人工智能原理与方法

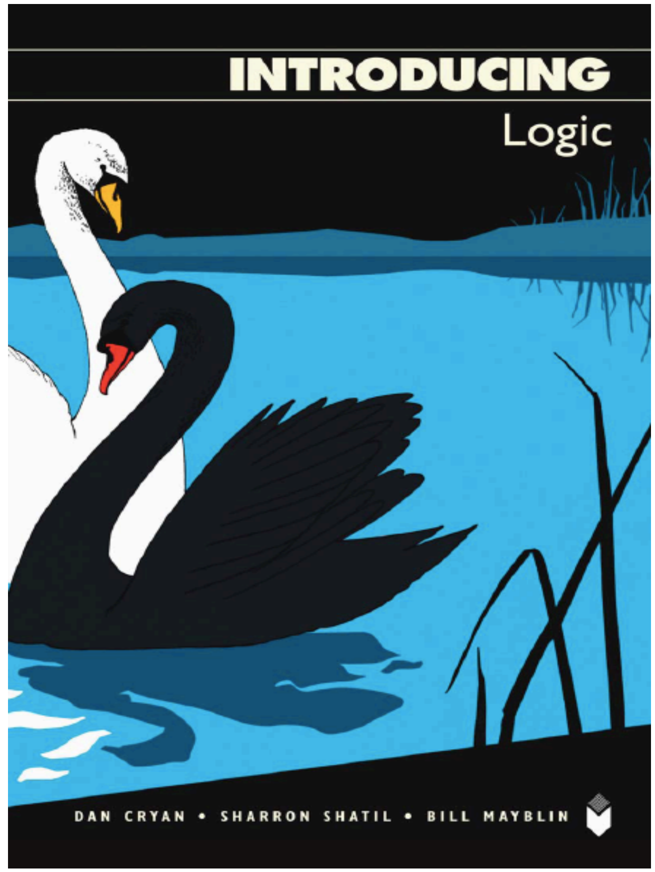
(二) 逻辑，语言与智能

Modern Artificial Intelligence & Applications

Zengchang Qin （秦曾昌）
zengchang.qin@gmail.com

Logic

- Highly recommended book:
- Introducing Logic – Dan Cryan, Sharron Shatil and Bill Mayblin, ICON Book UK/ Totem Books USA



What is logic?

Nothing is more natural to conversation than argument. We try to convenience the person we are arguing with that we are right, that our conclusion follows from something that they will accept.

- Syllogism

Leibniz's Law

1. "a = a"
e.g., "Socrates is Socrates."

2. If "**a** is **b**" and "**b** is **c**" then "**a** is **c**"

e.g., "All men are mortal, Socrates is a man, therefore Socrates is mortal."

Saying "**a** is **b**" is the same as saying that "all **a**'s are **b**".

3. "**a** = not (not **a**)"
e.g., "If Socrates is mortal then Socrates is not immortal."

4. " 'a is b' = 'not-b is not-a'"
e.g., "Socrates is a man means that if you are not a man then you are not Socrates."

Frege’s Propositional Calculus

- Frege shows the logical connectives such as If-Then can be transformed into other connectives.

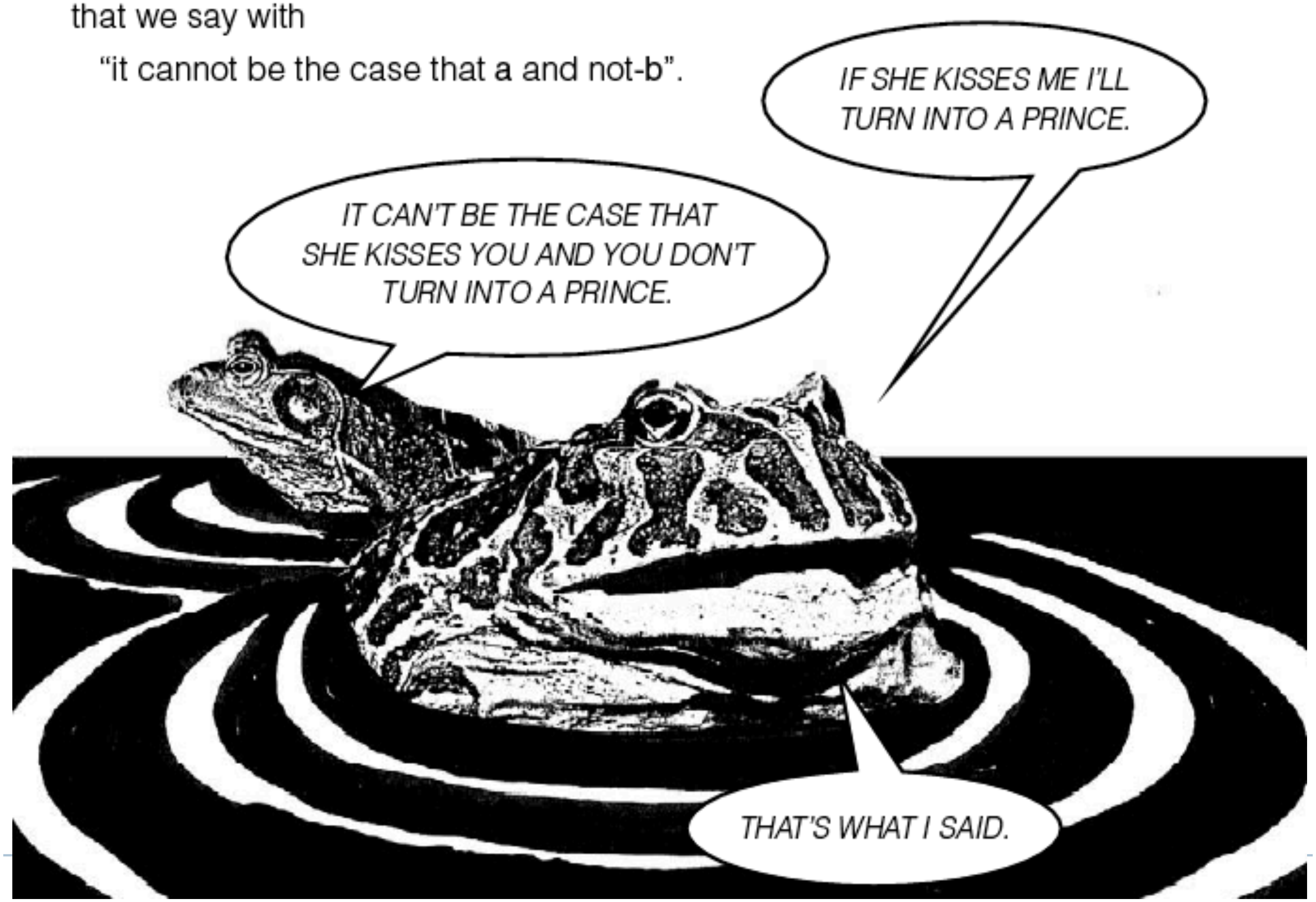
- P -> Q
- Not P $\lor$ Q
- Not (P $\land$ Not Q)

Another Example

"if **a** then **b**"

that we say with

"it cannot be the case that **a** and not-**b**".



Truth Table

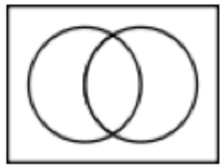
The truth value of compound sentences can then be determined by application of truth tables defined for each connective.

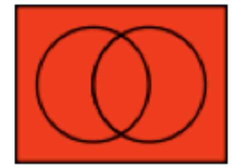
A	$\neg A$
t	f
f	t

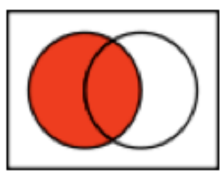
A	B	$A \wedge B$
t	t	t
f	t	f
t	f	f
f	f	f

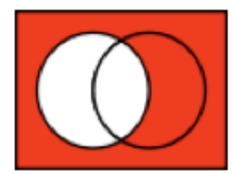
A	B	$A \rightarrow B$
t	t	t
f	t	t
t	f	f
f	f	t

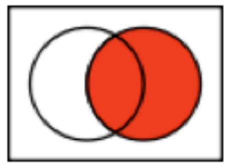
.......etc

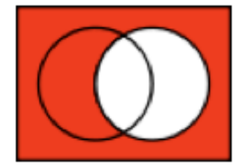
<table>
<tr><th colspan="4">Contradiction</th></tr>
<tr><th>Notation</th><th>Equivalent formulas</th><th>Truth table</th><th>Venn diagram</th></tr>
<tr><td>$\bot$</td><td>$P \wedge \neg P$
Opq</td><td><table><tr><td></td><td></td><td colspan="2">Q</td></tr><tr><td></td><td></td><td>0</td><td>1</td></tr><tr><td rowspan="2">P</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr></table></td><td></td></tr>
</table>

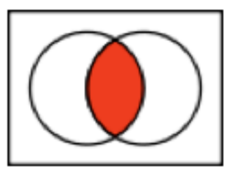
<table>
<tr><th colspan="4">Tautology</th></tr>
<tr><th>Notation</th><th>Equivalent formulas</th><th>Truth table</th><th>Venn diagram</th></tr>
<tr><td>$\top$</td><td>$P \vee \neg P$
Vpq</td><td><table><tr><td></td><td></td><td colspan="2">Q</td></tr><tr><td></td><td></td><td>0</td><td>1</td></tr><tr><td rowspan="2">P</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td><td></td></tr>
</table>

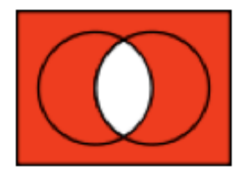
<table>
<tr><th colspan="4">Proposition P</th></tr>
<tr><th>Notation</th><th>Equivalent formulas</th><th>Truth table</th><th>Venn diagram</th></tr>
<tr><td>P</td><td>Ipq</td><td><table><tr><td></td><td></td><td colspan="2">Q</td></tr><tr><td></td><td></td><td>0</td><td>1</td></tr><tr><td rowspan="2">P</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td><td></td></tr>
</table>

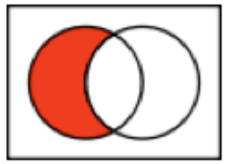
<table>
<tr><th colspan="4">Negation of P</th></tr>
<tr><th>Notation</th><th>Equivalent formulas</th><th>Truth table</th><th>Venn diagram</th></tr>
<tr><td>$\neg P$
$\sim P$</td><td>Np
Fpq</td><td><table><tr><td></td><td></td><td colspan="2">Q</td></tr><tr><td></td><td></td><td>0</td><td>1</td></tr><tr><td rowspan="2">P</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr></table></td><td></td></tr>
</table>

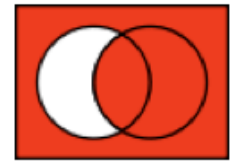
Proposition *Q*			
Notation	Equivalent formulas	Truth table	Venn diagram
Q	H*pq*	Q: 0 1 P 0: 0 1 P 1: 0 1	

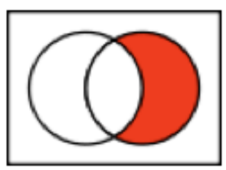
Negation of *Q*			
Notation	Equivalent formulas	Truth table	Venn diagram
$\neg Q$ $\sim Q$	N*q* G*pq*	Q: 0 1 P 0: 1 0 P 1: 1 0	

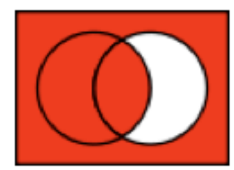
Conjunction			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \wedge Q$ $P \& Q$ $P \cdot Q$ P AND Q	$P \nrightarrow \neg Q$ $\neg P \nleftarrow Q$ $\neg P \downarrow \neg Q$ K*pq*	Q: 0 1 P 0: 0 0 P 1: 0 1	

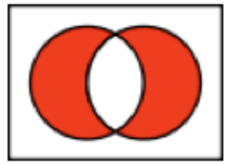
Alternative denial			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \uparrow Q$ $P \mid Q$ P NAND Q	$P \rightarrow \neg Q$ $\neg P \leftarrow Q$ $\neg P \vee \neg Q$ D*pq*	Q: 0 1 P 0: 1 1 P 1: 1 0	

Material nonimplication			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \nrightarrow Q$ $P \not\supset Q$	$P \land \neg Q$ $\neg P \downarrow Q$ $\neg P \nleftarrow \neg Q$ L*pq*	Q: 0, 1 P 0: 0, 0 P 1: 1, 0	

Material implication			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \rightarrow Q$ $P \supset Q$	$P \uparrow \neg Q$ $\neg P \lor Q$ $\neg P \leftarrow \neg Q$ C*pq*	Q: 0, 1 P 0: 1, 1 P 1: 0, 1	

Converse nonimplication			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \nleftarrow Q$ $P \not\subset Q$	$P \downarrow \neg Q$ $\neg P \land Q$ $\neg P \nrightarrow \neg Q$ M*pq*	Q: 0, 1 P 0: 0, 1 P 1: 0, 0	

Converse implication			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \leftarrow Q$ $P \subset Q$	$P \lor \neg Q$ $\neg P \uparrow Q$ $\neg P \rightarrow \neg Q$ B*pq*	Q: 0, 1 P 0: 1, 0 P 1: 1, 1	

Exclusive disjunction			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \not\leftrightarrow Q$ $P \not\equiv Q$ $P \oplus Q$ P XOR Q	$P \leftrightarrow \neg Q$ $\neg P \leftrightarrow Q$ $\neg P \not\leftrightarrow \neg Q$ Jpq	Q: 0, 1 P 0: 0, 1 P 1: 1, 0	

Biconditional			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \leftrightarrow Q$ $P \equiv Q$ P XNOR Q P IFF Q	$P \not\leftrightarrow \neg Q$ $\neg P \not\leftrightarrow Q$ $\neg P \leftrightarrow \neg Q$ Epq	Q: 0, 1 P 0: 1, 0 P 1: 0, 1	

History of notations

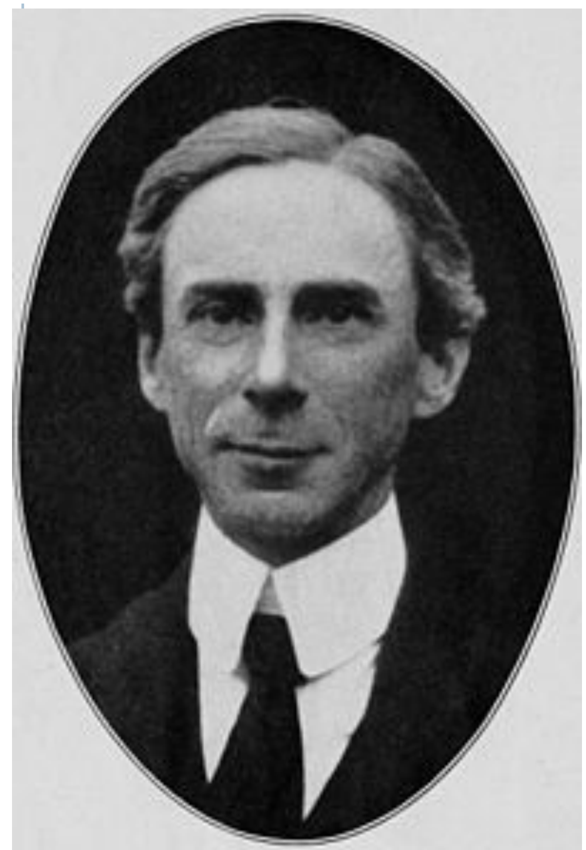
[edit]

- Negation: the symbol ¬ appeared in Heyting in 1929.[1][2] (compare to Frege's symbol —┬— A in his Begriffsschrift); the symbol ~ appeared in Russell in 1908;[3] an alternative notation is to add an horizontal line on top of the formula, as in $\overline{P}$; another alternative notation is to use a prime symbol as in P'.
- Conjunction: the symbol ∧ appeared in Heyting in 1929[1] (compare to Peano's use of the set-theoretic notation of intersection ∩ [4]); & appeared at least in Schönfinkel in 1924;[5] . comes from Boole's interpretation of logic as an elementary algebra.
- Disjunction: the symbol ∨ appeared in Russell in 1908 [3] (compare to Peano's use of the set-theoretic notation of union ∪); the symbol + is also used, in spite of the ambiguity coming from the fact that the + of ordinary elementary algebra is an exclusive or when interpreted logically in a two-element ring; punctually in the history a + together with a dot in the lower right corner has been used by Peirce,[6]
- Implication: the symbol → can be seen in Hilbert in 1917;[7] ⊃ was used by Russell in 1908[3] (compare to Peano's inverted C notation); ⇒ was used in Vax.[8]
- Biconditional: the symbol ≡ was used at least by Russell in 1908;[3] ↔ was used at least by Tarski in 1940;[9] ⇔ was used in Vax; other symbols appeared punctually in the history such as ⊃⊂ in Gentzen,[10] ~ in Schönfinkel[5] or ⊂⊃ in Chazal.[11]
- True: the symbol 1 comes from Boole's interpretation of logic as an elementary algebra over the two-element ring; other notations include $\bigwedge$ to be found in Peano.
- False: the symbol 0 comes also from Boole's interpretation of logic as a ring; other notations include $\bigvee$ to be found in Peano.

Russell's Paradox

- According to naive set theory, any definable collection is a set. Let R be the set of all sets that are not members of themselves.
- If R qualifies as a member of itself, it would contradict its own definition as a set containing all sets that are not members of themselves. On the other hand, if such a set is not a member of itself, it would qualify as a member of itself by the same definition.

- Other Forms:
- Barber's story and Reference Catalogue Story.



Russell Himself

- Russell's Paradox
- Russell's Teapot
- Type theory

In 1950, Russell was awarded the Nobel Prize in Literature, "in recognition of his varied and significant writings in which he champions humanitarian ideals and freedom of thought."

British philosopher, logician, mathematician, historian, and social critic.[2] At various points in his life he imagined himself a liberal, a socialist, and a pacifist, but he also admitted that he had never been any of these things in any profound sense.

Problem of China (1922)

China has an ancient civilization which is now undergoing a very rapid process of change. The traditional civilization of China had developed in almost complete independence of Europe, and had merits and demerits quite different from those of the West.

I believe that, if the Chinese are left free to assimilate what they want of our civilization, and to reject what strikes them as bad, they will be able to achieve an organic growth from their own tradition, and to produce a very splendid result, combining our merits with theirs.

There are, however, two opposite dangers to be avoided if this is to happen. The first danger is that they may become completely Westernized, retaining nothing of what has hitherto distinguished them, adding merely one more to the restless, intelligent, industrial, and militaristic nations which now afflict this unfortunate planet. The second danger is that they may be driven, in the course of resistance to foreign aggression, into an intense anti-foreign conservatism as regards everything except armaments.

An Example

"The present King of France is bald."

Russell can decompose it into:

1. There is a present King of France
2. There is exactly one present King of France
3. The present King of France is bald

To justify the collection of sentences.

Ludwig Wittgenstein

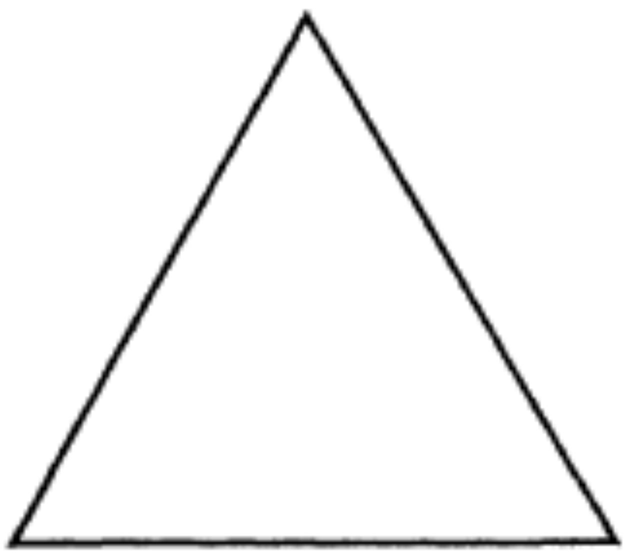
What any picture must have in common with reality in order to be able to depict it at all is logical form – the *form of reality*. For Wittgenstein, logic was something that both the world and language must have in common. It is only because language has something in common with the world that it can be used to picture the world, so it is only because of logic that our sentences have meaning at all.

Limitation of language: The world we see is biased with our language. What we can see are language-mirrored images.

Ludwig Wittgenstein

Wittgenstein's Example of Triangle

This triangle

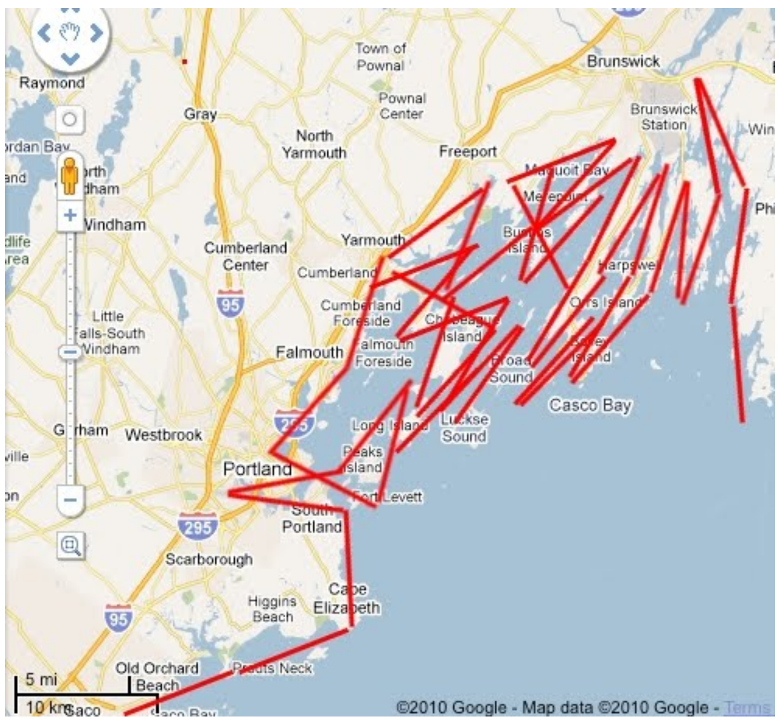


can be seen as
a triangular hole,
as a solid,
as a geometrical drawing,
as standing on its base,
as hanging from its apex,
as a mountain,
as a wedge,
as an arrow or pointer,
as an overturned object
which is meant to stand on
the shorter side of the
right angle,
as a half parallelogram,
and as various other things.
(PI p.200)

It seems we see it as an interpretation. But is it possible to SEE according to an INTERPRETATION?

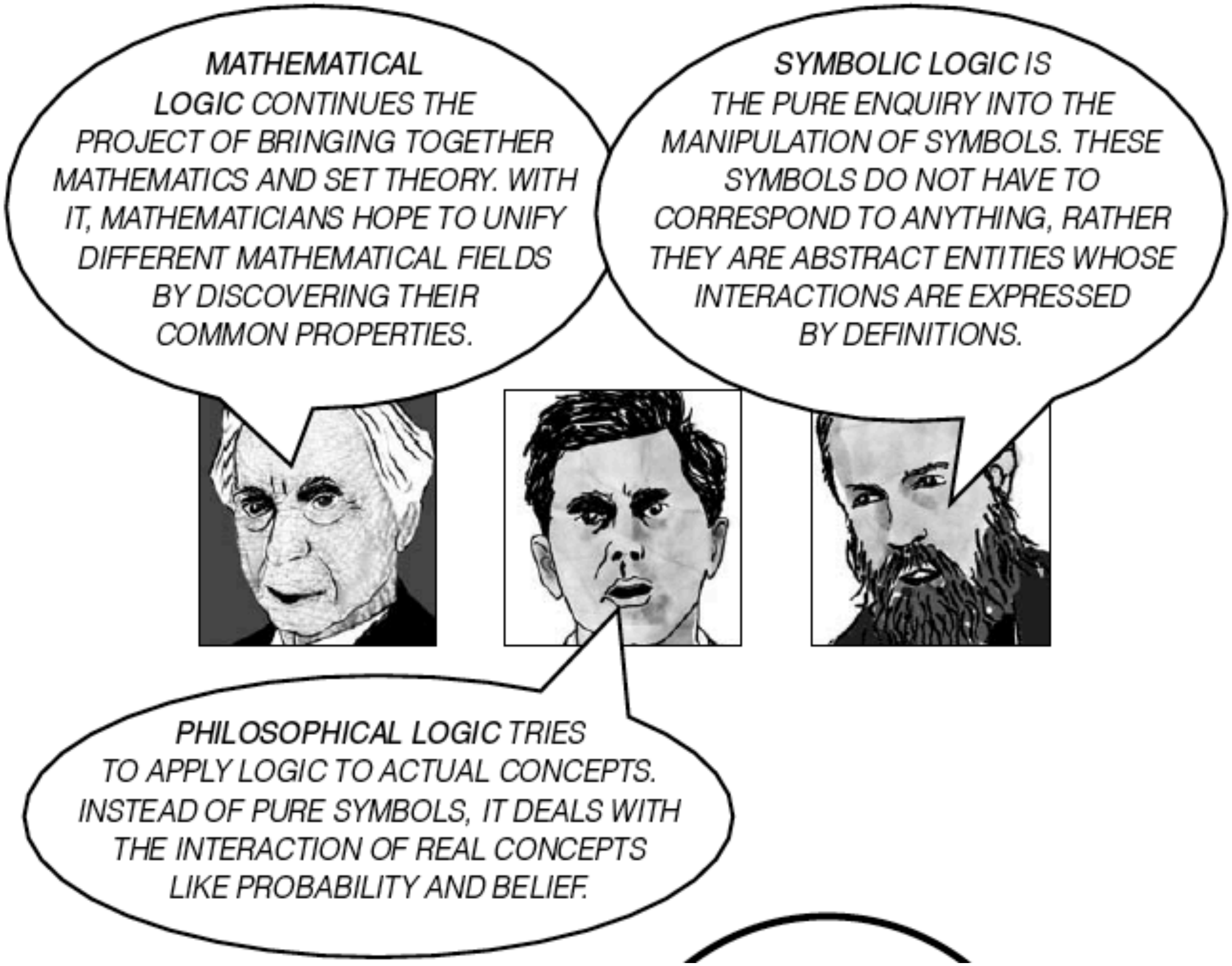
Simon's Cognitive Scissors

One blade is the limited cognitive ability of human



Another blade is structure of environment we hope to investigate.

Modern Logic

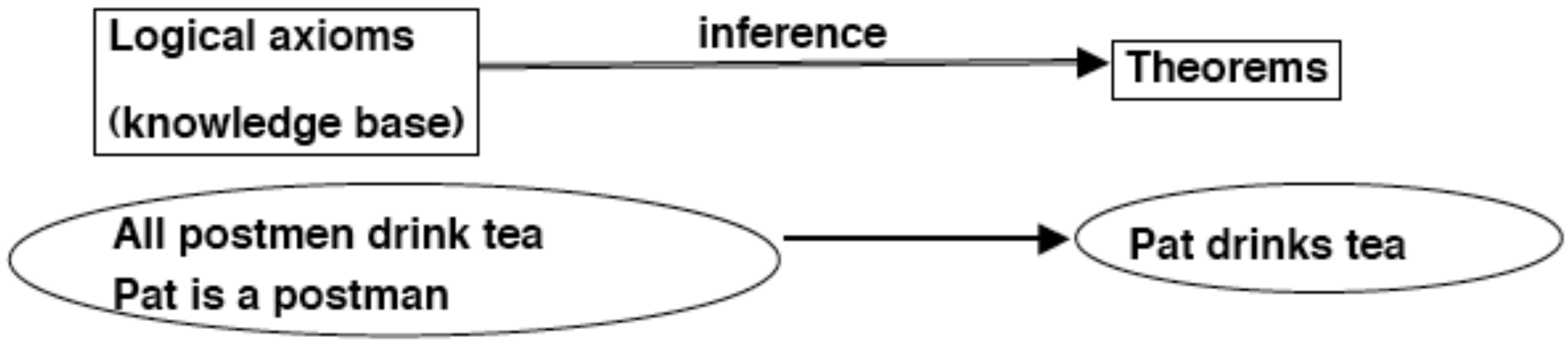


Logic and Programming

- Programming – Tell computer what to do.
- Logic – Science of reasoning, provide a precise language for expressing ideas, assumptions, relationships and arguments.

- Logic Programming – attempt to use techniques of logic in computer programming.
- Describe a problem to the computer
- Define assumptions and knowledge
- Ask computer questions

Logical deduction



express as X drinks tea if X is a postman

question : who drinks tea

X drinks tea if X is a postman

X drinks coffee if X is a fireman

Pat is a postman

Sam is a fireman

Prove it

Anyone who passes the exams is happy.

If you are lucky then you pass the exams.

If you are clever then you pass the exams.

Capricorns are clever

Gemini are lucky

Bill is a Capricorn

Anne is a Gemini

Therefore, Bill is happy and Anne is happy.

Prove it (2)

You can graduate if you pass the exams and pass the coursework.
If you are hardworking and punctual then you will pass the coursework.
If you are clever then you will pass the exams.
If you are lucky then you will pass the exams.
Pat is hardworking.
Sam is hardworking.
Sam is punctual.
Charlie is punctual.
Sam is clever.
Pat is lucky

Who can graduate?

Resolution Theorem Prover

$graduate(X) \leftarrow coursework(X) \wedge exams(X)$

$coursework(Y) \leftarrow punctual(Y) \wedge hardworking(Y)$

$exams(Z) \leftarrow clever(Z)$

$exams(W) \leftarrow lucky(W)$

$hardworking(pat)$

$hardworking(sam)$

$punctual(sam)$

$punctual(charlie)$

$clever(sam)$

$lucky(pat)$

- restrict to one positive literal per clause
- adopt the notation:

  ((graduate X) (coursework X) (exams X))

  …

  ((hardworking pat))

  …

  ? ((graduate A))
- solve query from left to right

  ?((graduate A))

  ?((coursework A) (exams A))

  ?((punctual A) (hardworking A) (exams A))
- search clauses from top to bottom

A Simple Knowledge Base

- this represents information on flights between various cities
  - "there is a flight from London to Dublin with flight number ba172"
- we wish to extract information from the computer
  - query *?((flight paris amsterdam af271))*
  - computer answers *yes*

```
((flight london dublin ba172))
((flight london paris ba345))
((flight paris amsterdam af271))
((flight edinburgh london ba215))
((flight london frankfurt lf419))
((flight amsterdam frankfurt ba119))
```

Queries

- Knowledge base + query $\Rightarrow$ execution

?((flight edinburgh london X))

((flight london dublin ba172))

((flight london paris ba345))

((flight paris amsterdam af271))

((flight edinburgh london ba215))

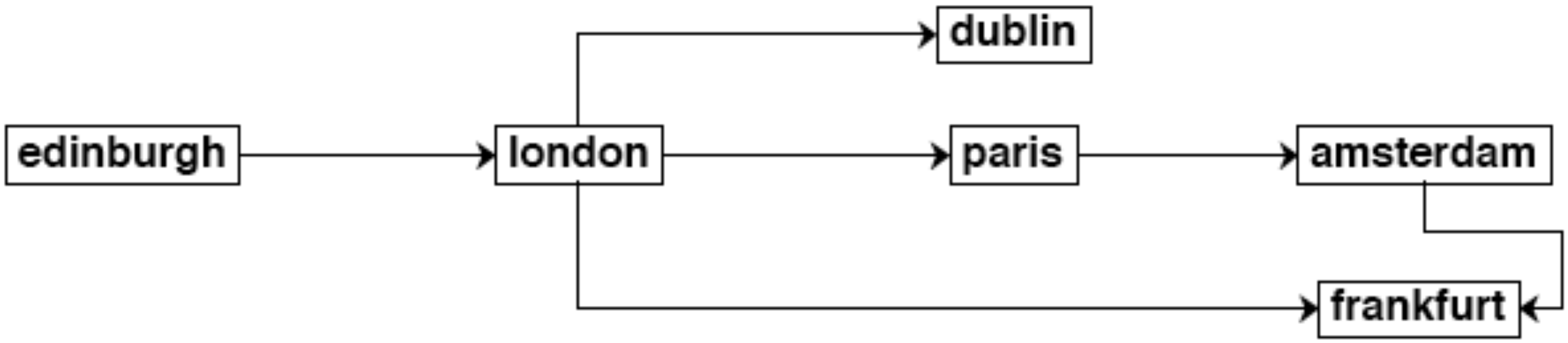
((flight london frankfurt lf419))

((flight amsterdam frankfurt ba119))

Multiple Queries

- Knowledge base + query $\Rightarrow$ execution

Rules



flight defines a relation between cities linked by a single flight. Suppose we want to define a relation *two_stage* between cities linked by a journey involving two flights

(i) As *facts*

((two_stage edinburgh dublin))
((two_stage edinburgh paris))
((two_stage edinburgh frankfurt))
((two_stage london amsterdam))
((two_stage paris frankfurt))

(ii) As a *rule*

((two_stage X Y) (flight X Z F1) (flight Z Y F2))

conclusion *if* condition1 *and* condition2

www.sics.se/isl/sicstuswww/site/highlights.html

SICStus

WELCOME TO

SICStus Prolog

Leading Prolog Technology

SICStus Home > SICStus Prolog Highlights | Download for Evaluation

About SICStus Prolog

- Start Page
- Feature Highlights
- Performance
- Platforms
- Documentation
- Coming Features
- Don't Panic!

Sales and Support

- Customers
- Evaluate Product
- Pricing/Order Info
- Download SICStus
- Support

Add-Ons & Third Party

- Extended Runtime
- Other

More Information

- Contact Information
- Community
- Links
- Bibliography

Feature Highlights

SICStus Prolog provides the means to integrate a very powerful Prolog engine in advanced, portable computing applications. It is the ideal choice for applications that require reasoning, search or pattern matching capabilities (e.g. intelligent web, rule engine, natural or formal language, and expert system style applications).

SICStus Prolog provides tools for delivering stand-alone applications as well as modules that can be linked with code written in other languages to create reliable solutions with Prolog embedded.

The latest version is **4.2.0**, featuring:

The hard facts about SICStus Prolog 4.2.0

Performance

- >61 MLips on a 2.67 GHz Intel Core i7. See Performance Summary.

Compliance

- International Standard ISO/IEC 13211-1 (PROLOG: Part 1—General Core).
- Internet Protocol versions 4 (IPv4) and 6 (IPv6).
- Unicode 5.0.

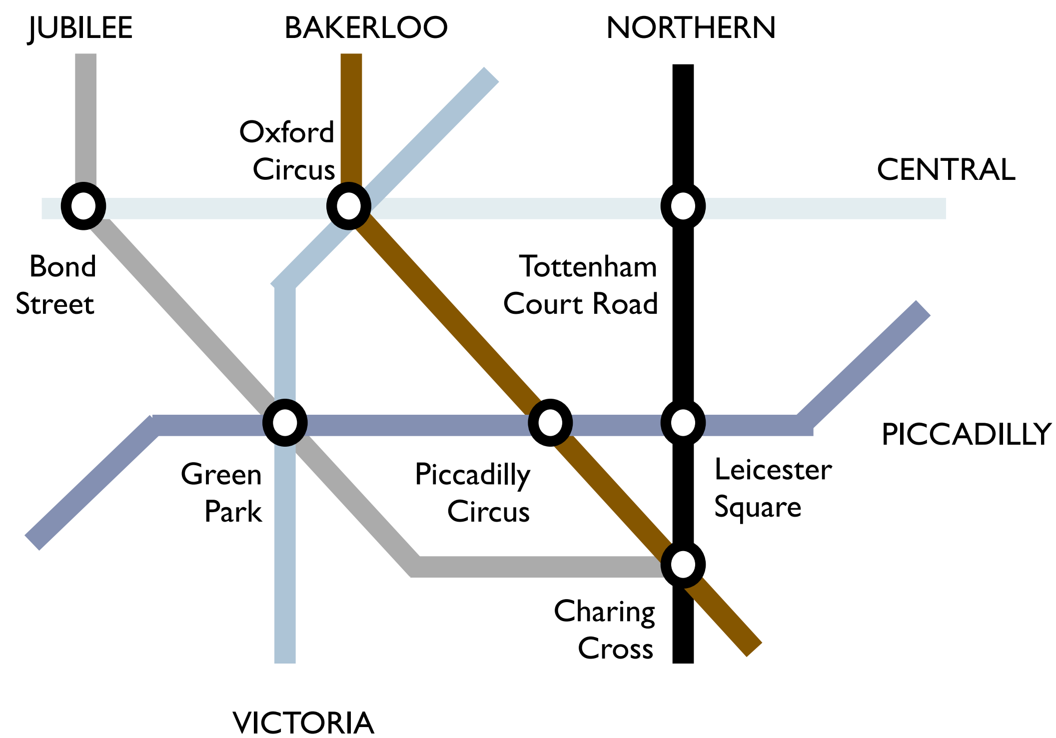
Prolog

```
mother_child(trude, sally).

father_child(tom, sally).
father_child(tom, erica).
father_child(mike, tom).

sibling(X,Y)      :- parent_child(Z, X), parent_child(Z,Y).

parent_child(X,Y) :- father_child(X,Y).
parent_child(X,Y) :- mother_child(X,Y).
```

This results in the following query being evaluated as true:

```
?- sibling(sally, erica).
Yes
```

London Underground example



UNDERGROUND

LRT Registered User No. 94/1954

London Underground in Prolog (1)

```
connected(bond_street,oxford_circus,central).
connected(oxford_circus,tottenham_court_road,central).
connected(bond_street,green_park,jubilee).
connected(green_park,charing_cross,jubilee).
connected(green_park,piccadilly_circus,piccadilly).
connected(piccadilly_circus,leicester_square,piccadilly).
connected(green_park,oxford_circus,victoria).
connected(oxford_circus,piccadilly_circus,bakerloo).
connected(piccadilly_circus,charing_cross,bakerloo).
connected(tottenham_court_road,leicester_square,northern).
connected(leicester_square,charing_cross,northern).
```

London Underground in Prolog (2)

Two stations are nearby if they are on the same line with at most one other station in between:

```
nearby(bond_street,oxford_circus).
nearby(oxford_circus,tottenham_court_road).
nearby(bond_street,tottenham_court_road).
nearby(bond_street,green_park).
nearby(green_park,charing_cross).
nearby(bond_street,charing_cross).
nearby(green_park,piccadilly_circus).
…
```

or better

```
nearby(X,Y):-connected(X,Y,L).
nearby(X,Y):-connected(X,Z,L),connected(Z,Y,L).
```

Exercise 1.1

Compare

```
nearby(X,Y):-connected(X,Y,L).
nearby(X,Y):-connected(X,Z,L),connected(Z,Y,L).
```

with

```
not_too_far(X,Y):-connected(X,Y,L).
not_too_far(X,Y):-connected(X,Z,L1),connected(Z,Y,L2).
```

This can be rewritten with don’t cares:

```
not_too_far(X,Y):-connected(X,Y,_).
not_too_far(X,Y):-connected(X,Z,_),connected(Z,Y,_).
```

Recursion (1)

A station is reachable from another if they are on the same line, or with one, two, … changes:

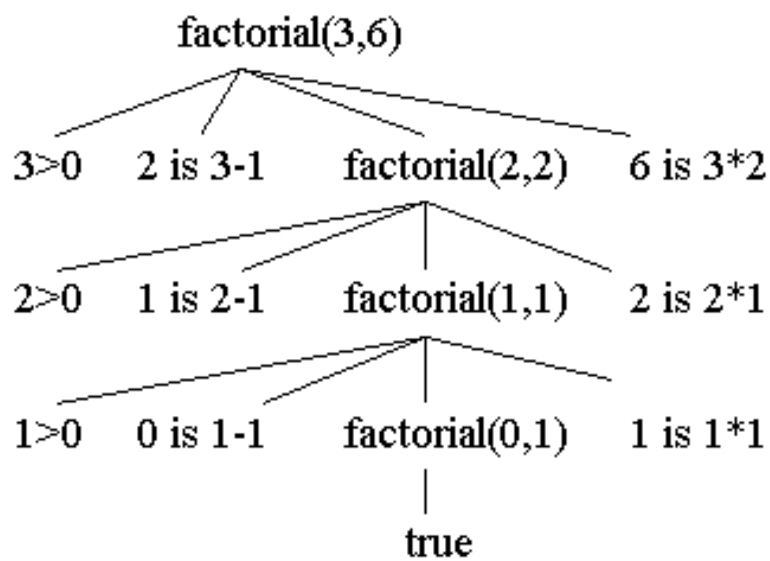
```
reachable(X,Y):-connected(X,Y,L).
reachable(X,Y):-connected(X,Z,L1),connected(Z,Y,L2).
reachable(X,Y):-connected(X,Z1,L1),connected(Z1,Z2,L2),
                connected(Z2,Y,L3).
…
```

or better

```
reachable(X,Y):-connected(X,Y,L).
reachable(X,Y):-connected(X,Z,L),reachable(Z,Y).
```

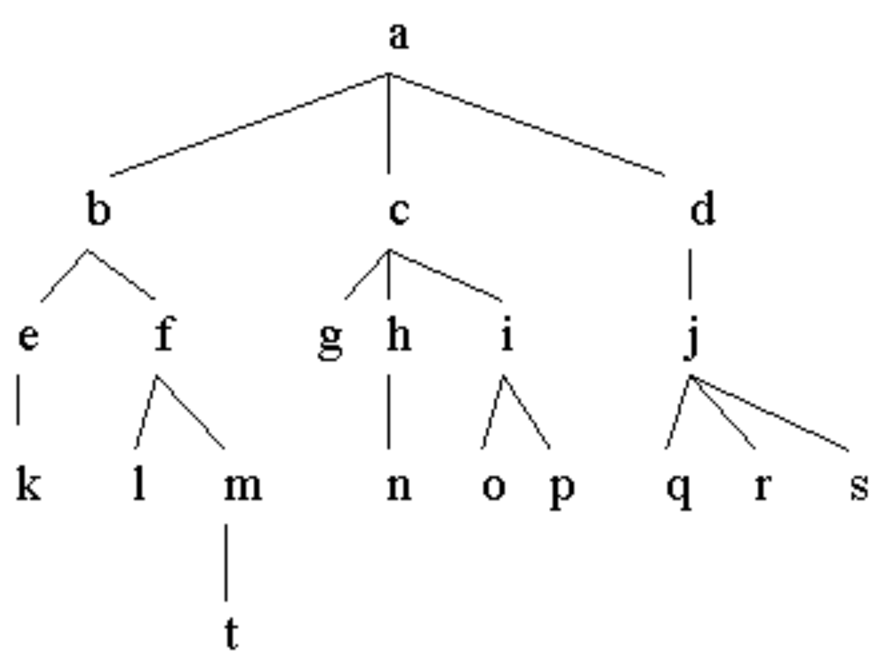
Recursions - factorial

```
factorial(0,1).

factorial(N,F) :-
   N>0,
   N1 is N-1,
   factorial(N1,F1),
   F is N * F1.


?-  factorial(3,W).
W=6
```



Tree Database

- Given tree.pl:



Summary

- Prolog has very simple syntax
  - constants, variables, and structured terms refer to objects
    - variables start with uppercase character
    - functors are never evaluated, but are used for naming
  - predicates express relations between objects
  - clauses express true statements
    - each clause independent of other clauses
- Queries are answered by matching with head of clause
  - there may be more than one matching clause
    - query answering is search process
  - query may have 0, 1, or several answers
  - no pre-determined input/output pattern (usually)