„Classic“ AI: Logic

The Idea

goal: draw conclusions from a set of data (observations, beliefs, etc)

logic is

- a powerful and well developed approach
- also a strong formal system suited for algorithms

challenges

- formalizing all real world facts (especially on a true/false basis)
- computational complexity

Logic in general

Logics

- formal languages
- for representing information
- such that conclusions can be drawn

Syntax defines the sentences in the language

Semantics define the "meaning" of sentences, i.e., define **truth** of a sentence in a world

Entailment

Entailment
one thing follows from another:

$$KB \models \alpha$$

- Knowledge base *KB*
- entails sentence α
- if and only if α is true
- in all worlds where *KB* is true

Models

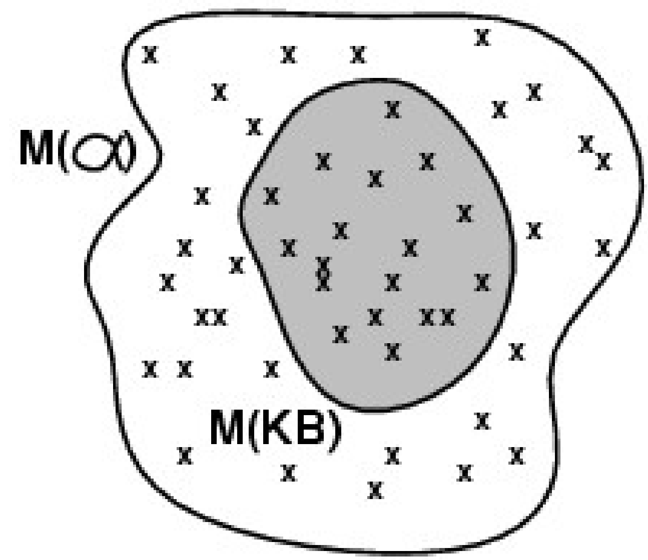
models

- formally structured worlds
- with respect to which truth can be evaluated

m is a **model** of a sentence α if α is true in *m*

M(α) is the set of all models of α

then KB $\models$ α iff *M(KB)* $\subseteq$ *M*(α)



Inference, Soundness, Completeness

Inference $KB \vdash_i \alpha$

sentence α can be derived from *KB* by procedure *i*

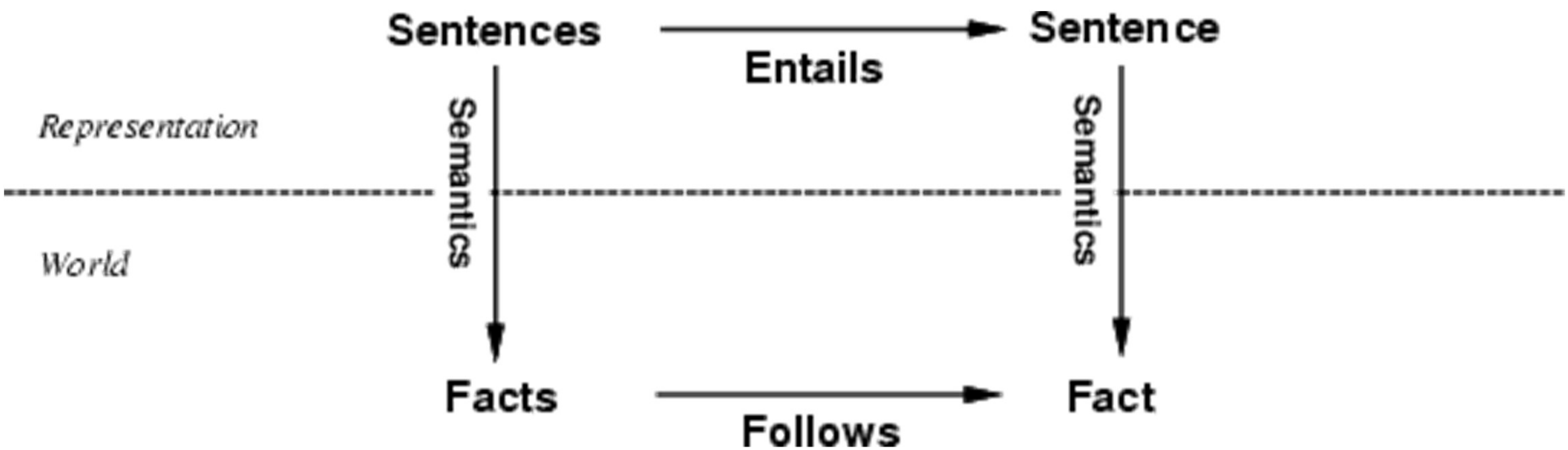
Soundness

- *i* is sound if $KB \vdash_i \alpha$ implies $KB \models \alpha$
- *i* derives only entailed sentences

Completeness

- *i* is complete if $KB \models \alpha$ implies $KB \vdash_i \alpha$
- *i* can derive any sentence that is entailed

Sentences and Facts



Semantics

- maps sentences in logic to facts in the world
- fact following from another ~ sentence is entailed by another

Propositional Logic
(aka Boolean Logic)

Propositional logic

- **Logical constants**: true, false
- **Propositional symbols**: P, Q, S, ...
- Wrapping **parentheses**: (…)
- **Connectives**:

$\neg$ ...not	[negation]
$\wedge$ ...and	[conjunction]
$\vee$ ...or	[disjunction]
$\Rightarrow$...implies	[implication / conditional]
$\Leftrightarrow$..is equivalent	[biconditional]

Propositional logic

- **Atomic sentences:** propositional symbols
- **Literal**: atomic sentence or negated atomic sentence (P, $\neg P$)
- Sentences are combined by **connectives**

e.g.:

$(P \wedge Q) \rightarrow R$

“If it is hot and humid, then it is raining”

$Q \rightarrow P$

“If it is humid, then it is hot”

Q

“It is humid.”

Propositional logic

- User defines a set of propositional symbols, like P and Q.
- User defines the **semantics** of each propositional symbol:
  - P means "It is hot"
  - Q means "It is humid"
  - R means "It is raining"
- A sentence (well formed formula) is defined as follows:
  - A symbol is a sentence
  - If S is a sentence, then $\neg$S is a sentence
  - If S is a sentence, then (S) is a sentence
  - If S and T are sentences, then ($S \vee T$), ($S \wedge T$), ($S \rightarrow T$), and ($S \leftrightarrow T$) are sentences
  - A sentence is formed by a finite number of the above rules

Backus Naur Form (BNF) Grammar of PL Sentences

```
S := <Sentence> ;
<Sentence> := <AtomicSentence> | <ComplexSentence> ;
<AtomicSentence> := "TRUE" | "FALSE" |
                    "P" | "Q" | "S" ;
<ComplexSentence> := "(" <Sentence> ")" |
          <Sentence> <Connective> <Sentence> |
          "NOT" <Sentence> ;
<Connective> := "AND" | "OR" | "IMPLIES" |
  "EQUIVALENT" ;
```

Precedences
(Saving Parentheses)

- proper use of parentheses can be tedious
- hence precedences to ease writing (much like in arithmetic)

Operator	Precedence
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\leftrightarrow$	5

e.g.,

- $(U \vee (\neg S \wedge T)) = U \vee \neg S \wedge T$
- $((S \wedge T) \rightarrow (U \vee V)) = S \wedge T \rightarrow U \vee V$

Terminology

- **valid sentence** (aka **tautology)**
  - sentence that is True under all interpretations
  - e.g., “It’s raining or it’s not raining.”
- **inconsistent sentence** (aka **contradiction**)
  - is a sentence that is False under all interpretations
  - e.g., “It’s raining and it’s not raining.”
- **P entails Q**, written P |= Q
  - whenever P is True, so is Q
  - i.e., all models of P are also models of Q

Truth tables

And

p	q	$p \cdot q$
T	T	T
T	F	F
F	T	F
F	F	F

Or

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

If . . . then

p	q	$p \supset q$
T	T	T
T	F	F
F	T	T
F	F	T

Not

p	$\sim p$
T	F
F	T

Truth tables

The five logical connectives:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

E.g., a complex sentence:

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
False	*False*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*True*	*True*	*False*	*True*

Notations

note that there are quite many notations

e.g.,

- AND as “*”, “&&”
- OR as “+”, “||”
- NOT as “-”, “′”, “¯”

De Morgan’s Laws

The complement of ANDs is the OR of complements
The complement of ORs is the AND of complements

$$(X + Y)' = X' Y'$$
$$(X Y)' = X' + Y'$$

- **proof** easy via **truth table**
- can also be generalized to **n** variables

Inference rules

Logical inference
- is used to create new sentences
- that logically follow from a given set of sentences (KB)

Deductive inference aka **Deduction**
- alternative to (enumerative) proofs with truth tables
- use of initial true sentence in KB and “re-write” rules
- i.e., logical equivalences (sound inference rules)

Inference rules

- an inference rule is **sound**
  - if every sentence X produced by an inference rule operating on a KB logically follows from the KB
  - i.e., the inference rule does not create any contradictions
- an inference rule is **complete**
  - if it is able to produce every expression that logically follows from (i.e., is entailed by) the KB
  - note the analogy to complete search algorithms

Sound Rules of Inference

Important Examples:

RULE	**PREMISE**	**CONCLUSION**
Modus Ponens	$A, A \rightarrow B$	B
And Introduction	A, B	$A \wedge B$
And Elimination	$A \wedge B$	A
Double Negation	$\neg\neg A$	A
Unit Resolution	$A \vee B, \neg B$	A
Resolution	$\mathbf{A \vee B, \neg B \vee C}$	$\mathbf{A \vee C}$

Sound Rules of Inference

- a rule is sound if its conclusion is true whenever the premise is true
- this can be shown using a truth table
- example modus ponens: premise A, A $\rightarrow$ B, conclusion B

A	B	A $\rightarrow$ B	OK?
True	True	True	√
True	False	False	√
False	True	True	√
False	False	True	√

Soundness Resolution

α	β	γ	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
False	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*True*	*True*
False	*True*	*False*	*True*	*False*	*False*
<u>*False*</u>	<u>*True*</u>	<u>*True*</u>	<u>*True*</u>	<u>*True*</u>	<u>*True*</u>
<u>*True*</u>	<u>*False*</u>	<u>*False*</u>	<u>*True*</u>	<u>*True*</u>	<u>*True*</u>
<u>*True*</u>	<u>*False*</u>	<u>*True*</u>	<u>*True*</u>	<u>*True*</u>	<u>*True*</u>
True	*True*	*False*	*True*	*False*	*True*
<u>*True*</u>	<u>*True*</u>	<u>*True*</u>	<u>*True*</u>	<u>*True*</u>	<u>*True*</u>

Proving with Inference

(deductive) proof

- a sequence of sentences
- where each sentence is either a premise
- or a sentence derived
  - from earlier sentences in the proof
  - by one of the rules of inference

last sentence = **theorem** (aka goal or query) that is to be proven

Proving with Inference

deduction relies on **monotonicity** of PL

- i.e.,
  - deriving a new sentence and adding it to the KB
  - does not affect what can be entailed from original KB
- can freely add true sentences
  - that can be derived in any order
- once something is proved true, it remains true

(there are non-monotonic logics)

Proving: Example

premises:

- $(P \wedge Q) \rightarrow R$
  - "If it is hot and humid, then it is raining"
- $Q \rightarrow P$
  - "If it is humid, then it is hot"
- Q
  - "It is humid"

goal: proof for R - "it is raining"

Proving: Example

1 Q	Premise - “It is humid”
2 $Q \rightarrow P$	Premise - “If it is humid, it is hot”
3 P	Modus Ponens (1,2) - “It is hot”
4 $(P \wedge Q) \rightarrow R$	Premise - “If it’s hot & humid, it’s raining”
5 $P \wedge Q$	And Introduction(1,3) - “It is hot & humid”
6 R	Modus Ponens(4,5) - “It is raining”

Note: inference is in general a search problem

Proof by Contradiction aka Refutation

show KB $\models \alpha$

- by proving that KB $\wedge \neg\alpha$ is *unsatisfiable*,
- i.e., by deducing False from KB $\wedge \neg\alpha$

- inference can use all the logical equivalences to derive new sentences
- or **resolution** as ***single*** inference rule
  - **sound**: only derives entailed sentences
  - **complete**: can derive any entailed sentence

Proof by Contradiction aka Refutation

So, Resolution is refutation complete:

if KB $\models \alpha$, then KB $\wedge \neg\alpha \vdash$ False

- but the sentences need to be pre-processed into a special form
- fortunately, all sentences *can* be converted into this form

Conjunctive Normal Form (CNF)

CNF aka clausal normal form

- a conjunction of clauses
- where a clause is a disjunction of literals

i.e., an AND of ORs of literals

e.g., $(A \vee \neg B \vee \neg C) \wedge (\neg A \vee D)$

Disjunctive Normal Form (DNF) is analog,

i.e., OR of ANDs of literals

Conjunctive Normal Form (CNF)

Replace all $\Leftrightarrow$ using iff/biconditional elimination

$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$

Replace all $\Rightarrow$ using implication elimination

$\alpha \Rightarrow \beta \equiv \neg\alpha \lor \beta$

Move all negations inward using

- double-negation elimination
  - $\neg(\neg\alpha) \equiv \alpha$
- de Morgan's rule
  - $\neg(\alpha \lor \beta) \equiv \neg\alpha \land \neg\beta$
  - $\neg(\alpha \land \beta) \equiv \neg\alpha \lor \neg\beta$

Apply distributivity of $\land$ over $\lor$

$\alpha \lor (\beta \land \gamma) \equiv (\alpha \lor \beta) \land (\alpha \lor \gamma)$

Resolution Refutation Algorithm

1. Convert all sentences in KB to CNF
2. Add negation of query (in CNF) to KB
3. Pick 2 sentences that have not been used before and can be used with the Resolution Rule of inference
4. If none, halt and answer that query is *NOT* entailed by KB
5. Compute resolvent and add it to KB
6. If False in KB
   - Then halt and answer that the query *IS* entailed by KB
   - Else Goto 3

use e.g. BFS for this (esp., step 3)

nice but can be computationally demanding

Horn Sentences

Horn sentence or **Horn clause** has the form:

$$P1 \wedge P2 \wedge P3 \ldots \wedge Pn \rightarrow Q$$

$(P \rightarrow Q) = (\neg P \vee Q)$

or alternatively

$$\neg P1 \vee \neg P2 \vee \neg P3 \ldots \vee \neg Pn \vee Q$$

where Ps and Q are non-negated atoms

proving for Horn sentences:

$P, Q, (P \wedge Q) \Rightarrow R \vdash R$

- Generalized Modus Ponens sufficient
- computationally efficient (more about this later)
- but Horn clauses are only a subset of PL

Limits Propositional Logic

Propositional Logic is a weak language

i.e., hard to

- identify "objects/individuals" (e.g., Mary, 3)
- talk about properties of individuals or relations between individuals (e.g., "Bill is tall")
- represent generalizations, patterns, regularities (e.g., "all triangles have 3 sides")

Example: Every person is mortal. Confucius is a person. Hence, Confucius is mortal. *How to express in PL?*