

# Artificial Intelligence

## III Knowledge and Reasoning

### III.3 Inference

F. C. Langbein

School of Computer Science  
and Informatics  
Cardiff University



1.4

## Overview

- Inference and theorem proving
  - Enumeration
  - Inference by searching
- Normal Forms
  - Conjunctive and disjunctive normal form
  - Horn form
- Resolution for propositional logic
  - Propositionalisation
  - Incompleteness

F. C. Langbein, Artificial Intelligence – III Knowledge and Reasoning: III.3 Inference

1

## Interacting with FOL KBs

- Suppose a wumpus-world agent is using a FOL KB:
  - It perceives a smell and a breeze (but no glitter) at  $t = 5$ :  
`TELL (KB, PERCEPT([Smell, Breeze, None], 5))`
  - `ASK (KB,  $\exists a$  Action(a,5))`  
i.e. does the KB entail any particular actions at  $t = 5$ ?
  - Answer: Yes,  $\{a/\text{Shoot}\}$  ← **substitution** (binding list)
- Given a sentence  $S$  and a substitution  $\sigma$ 
  - $S\sigma$  denotes the result of applying  $\sigma$  to  $S$
- Example:  $S = \text{Smarter}(x, y)$ ,  $\sigma = \{x/\text{Barrack}, y/\text{George}\}$ 
  - $S\sigma = \text{Smarter}(\text{Barrack}, \text{George})$
- `ASK(KB, S)` returns some/all  $\sigma$  such that  $KB \models S\sigma$

F. C. Langbein, Artificial Intelligence – III Knowledge and Reasoning: III.3 Inference

2

## Inference

- $KB \vdash_l \alpha$ : sentence  $\alpha$  can be derived from  $KB$  by method  $l$ 
  - E.g. consequences of  $KB$  are a haystack;  $\alpha$  is a needle
  - Entailment: needle is in haystack; inference: finding it
- **Soundness**:  $l$  is sound if  
whenever  $KB \vdash_l \alpha$ , it is also true that  $KB \models \alpha$
- **Completeness**:  $l$  is complete if  
whenever  $KB \models \alpha$ , it is also true that  $KB \vdash_l \alpha$
- First-order logic is expressive enough to say almost anything of interest
  - There exists a sound and complete inference procedure (answers any question whose answer follows from what is known; but otherwise we do not necessarily get an answer)

## Inference by Enumeration

- Simple inference method: enumeration
  - To verify  $KB \models \alpha$ , check all possible models ( $\alpha$  must be true whenever  $KB$  is true)
  - E.g.  $(KB = \{(A \vee C) \wedge (B \vee \neg C)\}) \models (\alpha = A \vee B)$

A	B	C	$A \vee C$	$B \vee \neg C$	KB	$\alpha$
false	false	false	false	true	false	false
false	false	true	true	false	false	false
false	true	false	false	true	false	true
false	true	true	true	true	true	true
true	false	false	true	true	true	true
true	false	true				
true	true	false				
true	true	true				

- Expensive
  - In propositional logic, need to check  $O(2^n)$  models where  $n$  is number of symbols
  - In FOL, also for all variable values (propositionalisation)

## Direct and Indirect Proofs

- Inference tries to prove a sentence based on a  $KB$
- A sentence is **valid** if it is true in all models  
(e.g.  $A \vee \neg A$ ,  $A \Rightarrow A$ ,  $(A \wedge (A \Rightarrow B)) \Rightarrow B$ )
  - Validity connected to inference (*deduction theorem*)
    - $KB \models \alpha$  iff  $KB \Rightarrow \alpha$  is valid (*direct proof*)
- A sentence is
  - **satisfiable** if it is true in some model (e.g.  $A \vee B$ ,  $C$ )
  - **unsatisfiable** if it is true in no model (e.g.  $A \wedge \neg A$ )
- Satisfiability linked to inference (*reductio ad absurdum*)
  - $KB \models \alpha$  iff  $KB \wedge \neg \alpha$  is unsatisfiable (*indirect proof*)
    - Note,  $KB \wedge \neg \alpha \equiv \neg(KB \Rightarrow \alpha)$

## Inference by Searching (direct proof)

► Proof process is **search** with **inference rules** as operators

- **Modus Ponens** (MP):

$$\frac{\alpha, \alpha \Rightarrow \beta \quad \text{At}(\text{Joe}, \text{Cardiff}), \quad \text{At}(\text{Joe}, \text{Cardiff}) \Rightarrow \text{OK}(\text{Joe})}{\beta \quad \text{OK}(\text{Joe})}$$

- **And-Introduction** (AI):

$$\frac{\alpha, \quad \beta \quad \text{OK}(\text{Joe}), \text{CS\_BSc}(\text{Joe})}{\alpha \wedge \beta \quad \text{OK}(\text{Joe}) \wedge \text{CS\_BSc}(\text{Joe})}$$

- **Universal Elimination** (UE):

$$\frac{\forall x \alpha \quad \forall x \text{At}(x, \text{Cardiff}) \Rightarrow \text{OK}(x)}{\alpha\{x/\tau\} \quad \text{At}(\text{Anne}, \text{Cardiff}) \Rightarrow \text{OK}(\text{Anne})}$$

$\tau$  must be a ground term, i.e. must not contain variables

## Example Search Proof

Bob is a buffalo	1. Buffalo(Bob)
Anne is a pig	2. Pig(Anne)
Buffaloes outrun pigs	3. $\forall x, y \text{ Buffalo}(x) \wedge \text{Pig}(y) \Rightarrow \text{Faster}(x, y)$

Bob outruns Anne

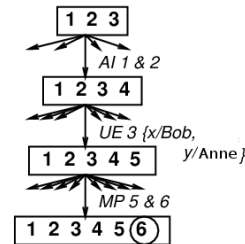
AI 1 & 2	4. Buffalo(Bob) $\wedge$ Pig(Anne)
UE 3, {x/Bob, y/Anne}	5. Buffalo(Bob) $\wedge$ Pig(Anne) $\Rightarrow$ Faster(Bob, Anne)
MP 4 & 5	6. Faster(Bob, Anne)

► Search with inference rules

- **Operators:** inference rules
- **States:** sets of sentences
- **Goal test:** is query sentence in state?

► AI, UE, MP is common pattern

► **Problem:** huge branching factor, esp. for UE



## Logical Equivalence

► Two sentences are logically equivalent iff they are true in the same models

$$\alpha \equiv \beta \text{ iff } \alpha \models \beta \text{ and } \beta \models \alpha$$

$(A \wedge B)$	$\equiv$	$(B \wedge A)$	$(A \Leftrightarrow B)$	$\equiv$	$(A \Rightarrow B) \wedge (B \Rightarrow A)$
$(A \vee B)$	$\equiv$	$(B \vee A)$	$\neg(A \wedge B)$	$\equiv$	$(\neg A \vee \neg B)$
$((A \wedge B) \wedge C)$	$\equiv$	$(A \wedge (B \wedge C))$	$\neg(A \vee B)$	$\equiv$	$(\neg A \wedge \neg B)$
$((A \vee B) \vee C)$	$\equiv$	$(A \vee (B \vee C))$	$(A \wedge (B \vee C))$	$\equiv$	$((A \wedge B) \vee (A \wedge C))$
$\neg(\neg A)$	$\equiv$	$A$	$(A \vee (B \wedge C))$	$\equiv$	$((A \vee B) \wedge (A \vee C))$
$(A \Rightarrow B)$	$\equiv$	$(\neg B \Rightarrow \neg A)$	$(A \Rightarrow B)$	$\equiv$	$(\neg A \vee B)$

► Rewrite rules for logical sentences

- Conversion into normal forms

## Normal Forms

- Some inference methods use syntactic operations on sentences
  - Simpler to use sentences in standardised forms
- **Conjunctive** normal form (CNF)
  - *Conjunction* of disjunctions of literals  
*clauses*
  - E.g.  $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
- **Disjunctive** normal form (DNF)
  - *Disjunction* of conjunctions of literals  
*terms*
  - E.g.  $(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$
- DNF and CNF exist for all *KBs*

## Normal Forms

- **Horn form**
  - *Conjunction* of *Horn clauses*
  - *Horn clause*: clause with at most one positive literal
    - E.g.  $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
  - Often written as set of implications:
 
$$B \Rightarrow A \quad (\equiv \neg B \vee A)$$

$$(C \wedge D) \Rightarrow B \quad (\equiv \neg C \vee \neg D \vee B)$$
  - *Definite clause*: a Horn clause with exactly one positive literal
    - If some facts are true, then *one* fact is implied
- Not all *KBs* can be written in Horn form

## Resolution in PL (indirect proof)

- To prove  $KB \models \alpha$ , show that  $KB \wedge \neg \alpha$  is unsatisfiable
  - Resolution uses *KB*,  $\neg \alpha$  in **CNF**
  - Resolution rule **combines** two clauses to make new one
  - Inference continues until **empty clause** is derived (contradiction) or no new sentences can be created
- PL **resolution rule**

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma} \equiv \frac{\neg \alpha \Rightarrow \beta, \beta \Rightarrow \gamma}{\neg \alpha \Rightarrow \gamma}$$
  - In general
 
$$\frac{\alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_n \vee \gamma, \beta_1 \vee \beta_2 \vee \dots \vee \beta_m \vee \neg \gamma}{\alpha_1 \vee \dots \vee \alpha_n \vee \beta_1 \vee \dots \vee \beta_m}$$
- Resolution is *sound* and *complete* for PL

## Example Resolution Proof

- ▶ KB in CNF
  1. BuffaloBob
  2. PigAnne
  3.  $\neg\text{BuffaloBob} \vee \neg\text{PigAnne} \vee \text{FasterBobAnne}$
- ▶ With negative query: 4.  $\neg\text{FasterBobAnne}$
- ▶ Repeated application of resolution rule
  - Resolution on 1 and 3 with BuffaloBob
    5.  $\neg\text{PigAnne} \vee \text{FasterBobAnne}$
  - Resolution on 2 and 3 with PigAnne
    6.  $\neg\text{BuffaloBob} \vee \text{FasterBobAnne}$
  - Resolution on 2 and 5 with PigAnne
    7. FasterBobAnne
  - Resolution on 4 and 7 with FasterBobAnne
    - ▶ Gives empty clause, i.e. contradiction, so  $\neg 4$  is true
- ▶ What about resolution in FOL?

## Universal Instantiation

- ▶ Every instantiation of a universal quantified sentence is entailed by it, i.e. for any ground term  $g$ ,
$$\frac{\forall v \alpha}{\alpha\{v/g\}}$$
- ▶ Example:  $\forall x \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$ 
  - ▶  $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$
  - $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$
  - $\text{King}(\text{Father}(\text{John})) \wedge \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))$
  - ...
- ▶ Universal instantiation can be applied *several times* to add new sentences
  - The new, expanded *KB* is *logically equivalent* to the old

## Existential Instantiation

- ▶ For any sentence  $\alpha$  and constant symbol  $K$  that does not appear elsewhere in the *KB*
$$\frac{\exists v \alpha}{\alpha\{v/K\}}$$
- ▶ Example:  $\exists x \text{Crown}(x) \wedge \text{OnHead}(x, \text{John})$ 
  - ▶  $\text{Crown}(C_1) \wedge \text{OnHead}(C_1, \text{John})$ 
    - provided  $C_1$  is a new constant symbol
    - $C_1$  is called a **Skolem constant**
- ▶ Existential instantiation can be applied *once* to replace the existential sentence
  - The new, expanded *KB* is *not equivalent* to the old
  - But is *satisfiable* iff the old *KB* was satisfiable

## Propositionalisation

- ▶ Every FOL  $KB$  can be *propositionalised so as to preserve entailment*
  - A ground sentence is entailed by new  $KB$  iff it is entailed by original  $KB$
- ▶ Idea: *propositionalise*  $KB$  and query and use resolution
  - But with function symbols there are infinitely many ground terms
    - E.g.  $\text{Father}(\text{Father}(\dots \text{Father}(\text{John}) \dots))$

## Propositionalisation and Resolution

- ▶ Herbrand, 1930: If a sentence is entailed by a FOL  $KB$ , it is entailed by a *finite* subset of the propositionalised  $KB$
- ▶ Apply iterative deepening approach to resolution

```
function  $\mathbb{B} \leftarrow \text{ID-RESOLUTION-FOL}(KB, \alpha)$ 
  for  $depth \leftarrow 0$  to  $\infty$ 
     $PKB \leftarrow \text{PROPOSITIONALISE}(KB, depth)$ 
    if  $\text{RESOLUTION-PL}(PKB, \alpha)$ , return true
```
- ▶ Fails to terminate if  $\alpha$  is not entailed
- ▶ Turing, 1936; Church 1936:
  - Entailment for FOL is semi-decidable
  - ▶ Algorithms exist that return true for every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence