Lecture 28

CS 6380 Artificial Intelligence

# Inference in First-Order Logic

Department of Computer Science and Engineering

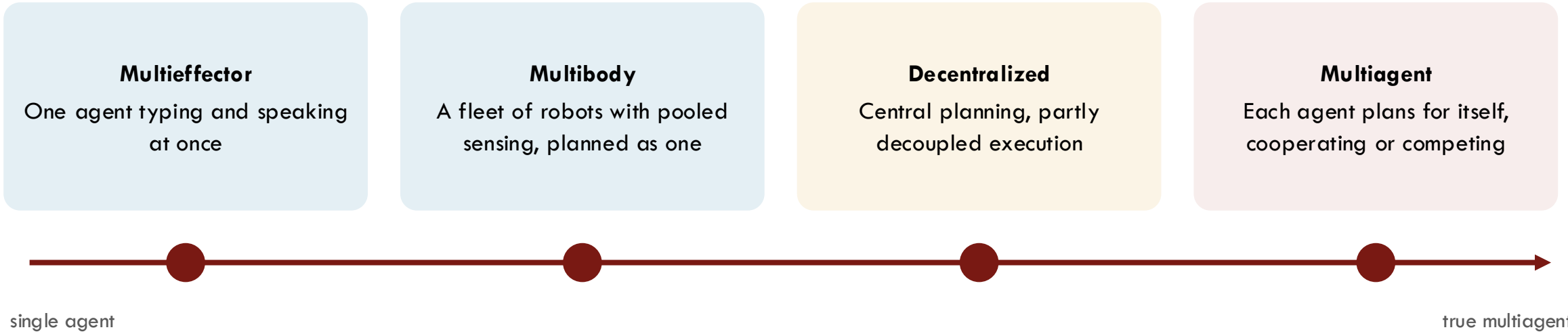
Indian Institute of Technology Madras (IITM), India



# Multiagent Planning

When more than one actor changes the world

# A Spectrum of Multiactor Settings




- With n actors a **joint action** is $\langle a_1, \ldots, a_n \rangle$, so the branching factor becomes $\mathbf{b^n}$
- Main research goal is to **decouple** actors so complexity grows linearly in n
- For **loosely coupled** problems, plan as if independent, then fix up the interactions

Source Section 11.4.1 of Russell and Norvig, Artificial Intelligence, A Modern Approach 3rd Ed.

# Doubles Tennis, Joint Plans

```
Actors(A, B)
Init(At(A, LeftBaseline) ∧ At(B, RightNet) ∧
     Approaching(Ball, RightBaseline))
Goal(Returned(Ball) ∧
     (At(a, RightNet) ∨ At(a, LeftNet)))
Action(Hit(actor, Ball),
  PRECOND Approaching(Ball, loc) ∧ At(actor, loc)
  EFFECT Returned(Ball))
Action(Go(actor, to),
  PRECOND At(actor, loc) ∧ to ≠ loc
  EFFECT At(actor, to) ∧ ¬At(actor, loc))
```

| | Plan 1 | Plan 2 |
|---|---|---|
| A | Go(A, RightBaseline), Hit(A, Ball) | Go(A, LeftNet), NoOp |
| B | NoOp, NoOp | Go(B, RightBaseline), Hit(B, Ball) |

Both joint plans work. If A follows Plan 2 and B follows Plan 1, **nobody** returns the ball

- A **concurrent action list** says which actions must or must not happen at the same time
  - Hit by a succeeds only if no other actor b hits at the same time
  - Carrying a heavy cooler needs two actors carrying together

Source Section 11.4.1 and 11.4.2 of Russell and Norvig, Artificial Intelligence, A Modern Approach 3rd Ed.

# Coordination Without a Central Planner

### Convention

Stick to your side of the court. Drive on one side of the road. Widespread conventions are social laws

### Communication

Shout Mine or Yours to agree on a joint plan

### Plan recognition

If A heads to the net, B infers Plan 2 and goes back to hit

### Emergence

Boids follow cohesion, separation and alignment, and a flock appears

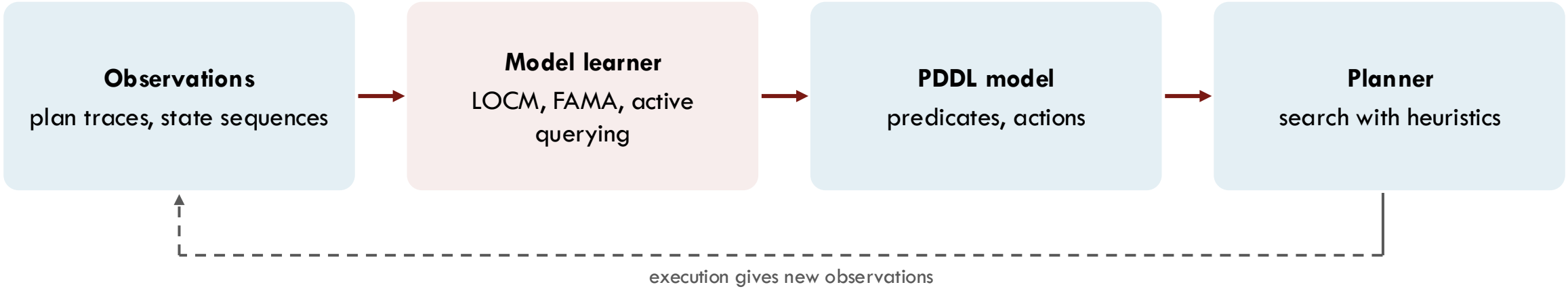
In CS6260 we look at **MA-PDDL** and cooperative multiagent planning

Source Section 11.4.2 of Russell and Norvig, Artificial Intelligence, A Modern Approach 3rd Ed.

# Learning for Planning

Where do the models come from

# Learning Action Models




- **Passive learning** from traces
  - LOCM and LOCM2 learn object centred models from action sequences alone
  - FAMA casts model learning as a planning problem
- **Active learning** by asking questions
  - Agent Interrogation Algorithm queries a black box agent to learn its model
  - Fewer, more informative examples

# What Else Can Be Learned

1

## Heuristics

Learn search guidance from solved problems instead of only deriving it from the model

2

## HTN methods

Save plans as methods and generalise them, so the agent improves with experience

3

## PPDDL models

Learn PPDDL operators and noisy deictic rules from noisy data

4

## Symbols

Invent predicates and abstractions from raw sensor data for planning

AIMA puts it simply. It is hard to imagine humans being so competent without some mechanism for learning and generalising methods

# Planning Meets Robotics, RL, and LLMs

## Task and motion planning

- Symbolic task planner chooses what to do
- Motion planner checks how to do it geometrically
- Integrated TAMP interleaves the two

## PDDL for RL

- Symbolic models give abstractions and options
- Plans can guide exploration and shape rewards
- RL handles low level control

## LLMs for planning

- Translate natural language into PDDL
- LLM proposes, sound planner verifies
- Combines flexibility with guarantees

Common thread, a **symbolic model** that something else grounds, learns or translates

# Key Takeaways

| | | |
|---|---|---|
| 1 | **Time** | Durations and resources turn planning into scheduling, which is NP-hard with shared resources |
| 2 | **Hierarchy** | HLAs encode how to do things and shrink search dramatically |
| 3 | **Uncertainty** | Belief states, contingent plans and replanning keep agents robust |
| 4 | **Many agents** | Joint plans need coordination through conventions, communication or recognition |
| 5 | **Learning** | Models, heuristics and methods can be learned, not just hand written |

# Inference in First Order Logic

# Recap: Where We Left Off

- FOL models = a universe of objects + an interpretation of the vocabulary
- Gödel (1929): FOL has sound and complete proof systems

  $T \models \varphi \text{ iff } T \vdash \varphi$

- But entailment in FOL is only semidecidable (Turing, Church 1936)
  - We can confirm every entailed sentence, but cannot always detect non-entailment
- Today: algorithms that answer questions posed in FOL
  - Given a knowledge base KB and a query $\alpha$, how do we establish $\mathrm{KB} \models \alpha$ ?

# Roadmap: Four Families of First-Order Inference

- Propositionalization: reduce FOL to propositional logic and reuse Chapter 7
- Unification and lifting: inference rules that work on FOL sentences directly
- Forward chaining: data-driven; deductive databases, production systems
- Backward chaining: goal-driven; logic programming and Prolog
- Resolution: refutation-complete for all of FOL; theorem provers

Forward and backward chaining handle *definite (Horn) clauses* only. Resolution handles any FOL sentence

# Universal Instantiation (UI)

- Substitute any ground term (a term with no variables) for a universally quantified variable
- SUBST(θ, α): apply substitution θ to sentence α
- Can be applied many times

$$\frac{\forall v \ \alpha}{\text{SUBST}(\{v/g\}, \alpha)}$$

for any variable v and ground term g

**Example: everyone who attends IITM is good at AI**

$\forall x \ \text{Attends}(x, \text{IITM}) \Rightarrow \text{GoodAt}(x, \text{AI})$

| Substitution | Result |
|---|---|
| {x/Alfred} | Attends(Alfred, IITM) ⇒ GoodAt(Alfred, AI) |
| {x/DaughterOf(Alfred)} | Attends(DaughterOf(Alfred), IITM) ⇒ GoodAt(DaughterOf(Alfred), AI) |

# Existential Instantiation (EI)

- Replace the variable by a single new constant that appears nowhere else in the KB
- The new name is a Skolem constant
  - A number a little above 2.71828 satisfies $d(x^y)/dy = x^y$. Call it $e$, not $\pi$
- Apply once, then discard the existential sentence
- New KB is inferentially equivalent: satisfiable exactly when the original is

$$\frac{\exists v \ \alpha}{\text{SUBST}(\{v/k\}, \alpha)}$$

k is a new constant symbol

**Example**

$\exists x \ \text{IsStick}(x) \wedge \text{Likes}(\text{DarkLord}, x)$

becomes

$\text{IsStick}(S_1) \wedge \text{Likes}(\text{DarkLord}, S_1)$

$S_1$ is a fresh name. Calling it Dobby would be wrong!

# Reduction to Propositional Inference

**First-order KB**

$\forall x$ Attends(x, IITM) $\wedge$ Studies(x, AI)
$\Rightarrow$ GoodAt(x, AI)

Attends(Alfred, IITM)

Studies(Alfred, AI)



**Instantiate x with every constant**

Attends(Alfred, IITM) $\wedge$ Studies(Alfred, AI) $\Rightarrow$ GoodAt(Alfred, AI)

Attends(IITM, IITM) $\wedge$ Studies(IITM, AI) $\Rightarrow$ GoodAt(IITM, AI)

Attends(AI, IITM) $\wedge$ Studies(AI, AI) $\Rightarrow$ GoodAt(AI, AI)

- Treat each ground atom, such as Attends(Alfred, IITM), as a proposition symbol
- Any complete algorithm from Chapter 7 now concludes GoodAt(Alfred, AI)
- This is propositionalization. Notice the silly instances it creates (in gray)

# The Catch: Infinitely Many Ground Terms

- With function symbols there are infinitely many ground terms

  $\text{Alfred},\ \text{DaughterOf}(\text{Alfred}),\ \text{DaughterOf}(\text{DaughterOf}(\text{Alfred})),\ \ldots$

- **Herbrand (1930)**: if a sentence is entailed, some finite subset of the propositionalized KB proves it
- So instantiate with terms of depth 0, then depth 1, then depth 2, … until a proof appears
- Complete: every entailed sentence *will* be proved
- Not entailed? The procedure may run forever, and we can't tell
  - Entailment in FOL is semidecidable: like the halting problem (Turing 1936, Church 1936)

# Propositionalization Is Wasteful

- To show $\mathrm{GoodAt}(\mathrm{Alfred}, \mathrm{AI})$, why generate sentences about $\mathrm{IITM}$ attending $\mathrm{IITM}$?
- A person reasons directly: find an $\mathrm{x}$ who attends IITM and studies AI. Then that $\mathrm{x}$ is good at AI
- Find a substitution $\theta$ that makes the rule's premises identical to facts in the KB
- Works even if the KB says *everyone* studies AI:

$$\mathrm{Attends}(\mathrm{Alfred}, \mathrm{IITM}), \quad \forall \mathrm{y}\ \mathrm{Studies}(\mathrm{y}, \mathrm{AI})$$

$$\theta = \{\mathrm{x}/\mathrm{Alfred}, \mathrm{y}/\mathrm{Alfred}\} \quad \Rightarrow \quad \mathrm{GoodAt}(\mathrm{Alfred}, \mathrm{AI})$$

# Generalized Modus Ponens (GMP)

$$\frac{p_1',\ p_2',\ \ldots,\ p_n',\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\mathrm{SUBST}(\theta, q)}$$

where $\mathrm{SUBST}(\theta, p_i') = \mathrm{SUBST}(\theta, p_i)$ for all i

| Facts | Rule |
|---|---|
| $p_1' = \mathrm{Attends}(\mathrm{Alfred}, \mathrm{IITM})$ | $p_1 = \mathrm{Attends}(x, \mathrm{IITM})$ |
| $p_2' = \mathrm{Studies}(y, \mathrm{AI})$ | $p_2 = \mathrm{Studies}(x, \mathrm{AI})$ |
| $\theta = \{x/\mathrm{Alfred}, y/\mathrm{Alfred}\}$ | $q = \mathrm{GoodAt}(x, \mathrm{AI})$ |

$\mathrm{SUBST}(\theta, q) =$ **GoodAt(Alfred, AI)**

- n + 1 premises: n atomic sentences and one implication
- Sound: follows from UI, since $p \models \mathrm{SUBST}(\theta, p)$
- A lifted Modus Ponens: from ground propositional logic up to FOL
- Makes *only* the substitutions a particular inference needs

# Unification

Finding substitutions that make two expressions identical is unification

$$\text{UNIFY}(p, q) = \theta \quad \text{where} \quad \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

Query: whom does John know? $\text{AskVars}(\text{Knows}(\text{John}, x))$

| p | q | UNIFY(p, q) |
|---|---|---|
| Knows(John, x) | Knows(John, Jane) | {x/Jane} |
| Knows(John, x) | Knows(y, Bill) | {x/Bill, y/John} |
| Knows(John, x) | Knows(y, Mother(y)) | {y/John, x/Mother(John)} |
| Knows(John, x) | Knows(x, Elizabeth) | fail |

The last fails: x cannot be John and Elizabeth at once

Try it: $\text{UNIFY}(\text{Likes}(\text{DarkLord}, x), \text{Likes}(y, \text{Location}(y))) = ?$

# Standardizing Apart and the Most General Unifier

## Standardizing apart

- Knows(x, Elizabeth) says *everyone* knows Elizabeth, so John should too
- It failed only because both sentences use the name x
- Fix: rename variables in one of them

UNIFY(Knows(John, x), Knows($x_{17}$, Elizabeth))
= {x/Elizabeth, $x_{17}$/John}

## Most general unifier (MGU)

- UNIFY(Knows(John, x), Knows(y, z)) has several answers:

{y/John, x/z} → Knows(John, z)
{y/John, x/John, z/John} → Knows(John, John)

- The first is more general: fewer restrictions on the variables
- Every unifiable pair has a single MGU, unique up to renaming

# The Unification Algorithm

```
function UNIFY(x, y, θ) returns a substitution
  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
    return UNIFY(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
  else if LIST?(x) and LIST?(y) then
    return UNIFY(x.REST, y.REST,
                 UNIFY(x.FIRST, y.FIRST, θ))
  else return failure

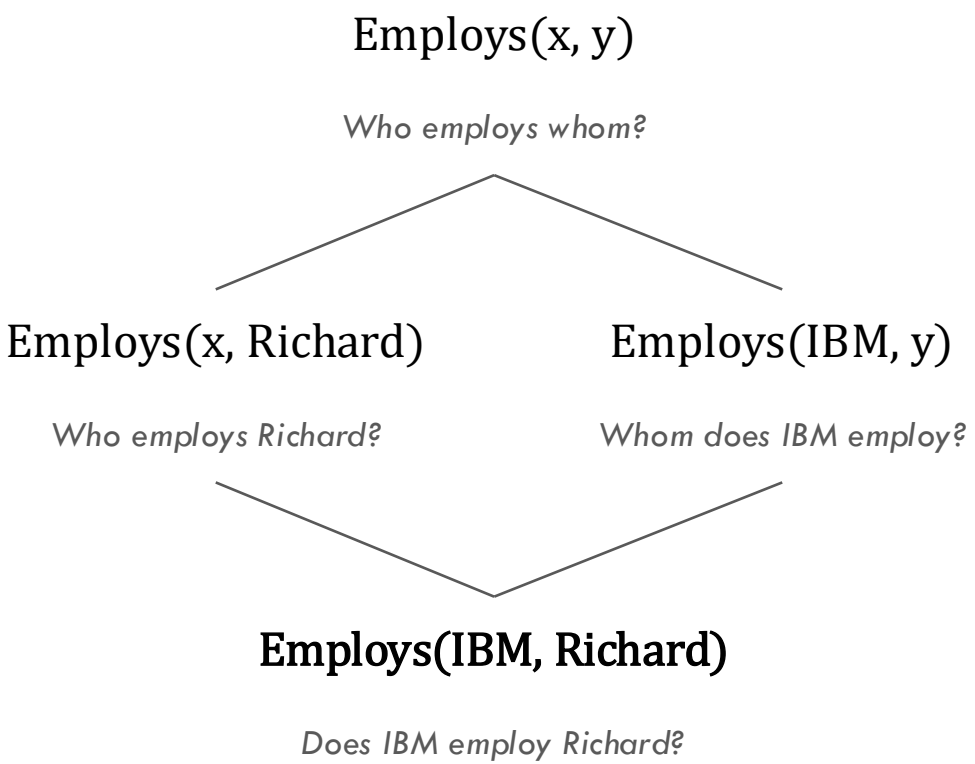
function UNIFY-VAR(var, x, θ) returns a substitution
  if {var/val} ∈ θ then return UNIFY(val, x, θ)
  else if {x/val} ∈ θ then return UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) then return failure
  else return add {var/x} to θ
```

- Walk both expressions side by side, building θ
- Occur check: a variable can't unify with a term containing it

  S(x) vs. S(S(x)): fail
- Makes UNIFY quadratic; Prolog skips it

# Storage and Retrieval: Indexing

- STORE(s) and FETCH(q) sit under TELL and ASK
- Simplest: one long list; unify the query with everything
- Predicate indexing: one bucket per predicate
- Huge buckets? Also index by arguments
- Store each fact under every query that unifies with it: a subsumption lattice
  - n arguments → $O(2^n)$ index entries




Subsumption lattice for one stored fact