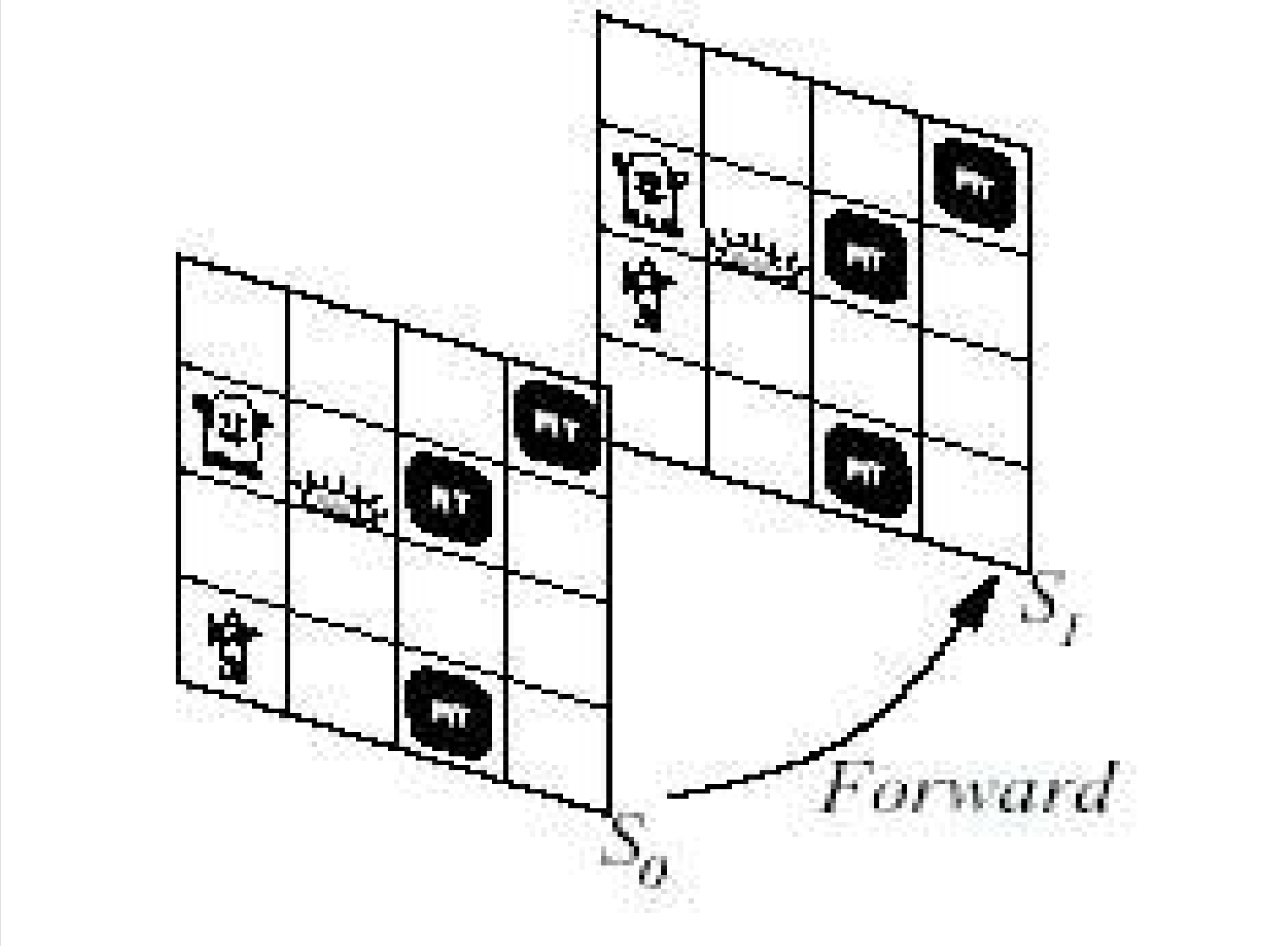
CSC421 Intro to Artificial Intelligence

UNIT 17:
Knowledge Engineering

Keeping track of change

- Facts hold in situations, rather than eternally
  - e.g. Holding(Gold, Now) rather than just Holding(Gold)
- Situation calculus is one way to represent change in FOL
  - Add a situation argument to every non-eternal predicate (for example Now is a situation)
- Situations are connected by the Result function
  - Result(a,s) is the situation that results from doing a in s

Situation Calculus



Describing axioms I

- "Effect" axiom – describe changes due to action
  - ∀s AtGold(s) => Holding(Gold, Result(Grab, s))
- "Frame" axioms – describe non-changes due to action
  - ∀s HaveArrow(s) => HaveArrow(Result(Grab, s))
- Frame problem : find elegant way to handle non-change (identified in 1969)
  - Representation : avoid frame axioms
  - Inference: avoid repeated "copy-overs" to keep track of state

Describing actions II

- Succesor-state axioms solve the representational frame problem
  - Each axiom is about a predicate
    - [P true afterwards] iff [an action made P true or P true already and no action made P false]
  - For holding the gold:
  - $\forall a,s$ Holding(Gold, Result(a,s)) <=> [a=Grab $\wedge$ AtGold(s)] $\vee$ [Holding (Gold) $\wedge$ a $\neq$ Release]
- R.Reiter 1991 UofT

Making Plans

- Initial condition in KB:
  - At(Agent, [1,1], S0)
  - At(Gold, [1,2], S0)
- Query: Ask(KB, ∃ s Holding(Gold, s))
  - i.e in what situation will I be holding Gold
  - Answer {s/Result(Grab, Result(Forward, So))}
- This assumes that the agent is interested in plans starting at S0 and that S0 is the only situation in KB

Making Plans: a better way

- Plans = actions sequences
- PlanResult(p,s) is the resulting of executing p in s
- Then the query Ask(KB, ∃ p Holding(Gold, PlanResult(p, S0))
- Answer: {p / [Forward, Grab]}
- Define PlanResult in terms of Result

Definition of PlanResult

- In terms of Result
  - ∀s PlanResult([], s) = s
  - ∀ a, p, s PlanResult([a|p], s) = PlanResult(p, Result(a,s))
- Planning systems are special purpose reasoners designed to do this type of inference more efficiently than a general purpose reasoner

Event Calculus

- Fluents hold at points in time
- Reasoning over intervals of time
  - Initiates(e,t,f) event e at time t cause fluent f to become true
  - Terminates(e,f,t) f ceases to be true
  - Happens(e,t)
  - Clipped(f,t,t2) terminated by some event between t and t2
- Many extensions
  - Duration, indirect effects, concurrent events

Generalized Events

- Aspects of some space-time chuck
- Generalizes: actions, locations, times, fluents and physical objects
  - Subevent(BattleofBritain, WorldWarII)
  - Subevent(WorldWarII, 20th century)
  - Duration(Period(WorldWarII)) > Year(5)
- $\exists w \quad w \in \text{CivilWars} \wedge \text{SubEvent}(w, 1640s) \wedge \text{In}(\text{Location}(w), \text{England})$

Processes (Liquid Events)

- Flying(Shankar)
- Any segment is still a member of Flying (Shankar)
- Fluent calculus
  - Both(e1, e2)
  - One of(e1, e2)
  - Either(e1, e2)
- Time intervals
  - Meet
  - Before/After
  - During
  - Overlap