# SATPlan

Course: CS40022
Instructor: Dr. Pallab Dasgupta

**Department of Computer Science & Engineering**
**Indian Institute of Technology Kharagpur**

# Planning Graphs

- Consists of a sequence of levels that correspond to time steps in the plan
- Each level contains a set of actions and a set of literals that *could* be true at that time step depending on the actions taken in previous time steps
- For every +ve and –ve literal C, we add a *persistence action* with precondition C and effect C

# Mutex Actions

- Mutex relation between two actions if:
  - Inconsistent effects – one action negates an effect of the other
  - Interference – one of the effects of one action is the negation of a precondition of the other
  - Competing needs – one of the preconditions of one action is mutually exclusive with a precondition of the other

# Mutex Literals

- Mutex relation between two literals if:
  - One is the negation of the other, or
  - Each possible pair of actions that could achieve the two literals is mutually exclusive (inconsistent support)

```
Function GraphPLAN( problem )
    // returns solution or failure
    graph ← Initial-Planning-Graph( problem )
    goals ← Goals[ problem ]
    do
        if goals all non-mutex in last level of graph
        then do
            solution ← Extract-Solution( graph )
            if solution ≠ failure then return solution
            else if No-Solution-Possible (graph )
                then return failure
        graph ← Expand-Graph( graph, problem )
```

# Planning with Propositional Logic

- The planning problem is translated into a CNF satisfiability problem
- The goal is asserted to hold at a time step T, and clauses are included for each time step up to T.
- If the clauses are satisfiable, then a plan is extracted by examining the actions that are true.
- Otherwise, we increment T and repeat

# SATPlan

Function SATPlan( problem, $T_{max}$ )
  // *returns solution or failure*

  for T = 0 to $T_{max}$ do
    *cnf, mapping* ← Trans-to-SAT(*problem*, T)
    *assignment* ← SAT-Solver( *cnf* )
    if *assignment* is not NULL then
        return Extract-Solution(*assignment, mapping*)

  return *failure*

# Modeling for SATPlan

- **Precondition Axioms**
  - Action occurrence requires the precondition to be satisfied
- **Action exclusion Axioms**
  - Prevent simultaneous actions
- **State constraints**
  - Generalization of exclusion axioms