**1. CLASSICAL PLANNING**

**Problem Solving**

* Sound: the solution is correct
* Complete: the solution is reachable
* Atomic representation: each state has 1 unique representation
  + Used by UCS and A\*
  + A\* heuristic must be admissible and consistent
* Factored representation: each state has multiple attribute/value pairs
  + Used by STRIPS and PDDL

**Logic Notation**

* Term: object (e.g. Box, Agent)
* Atom: description (e.g. Rich, Friend(Bob))
* Literal: atom or its negation (e.g. Rich, ~Rich)
* Ground literal: literal with no variable (e.g. At(NUS), At(x) is not as x is a variable)
* Sentence/formula: atoms joined by logical connectives
* Substitution: replace variables with term (e.g. {x/Bob})
* Unifier: returns substitution that makes 2 sentences identical (e.g. Unify(At(x, NUS), At(Bob, y)) = {x/Bob, y/NUS})
* Most general unifier (MGU): outputs of other unifiers can be derived from it

**STRIPS**

* Conjunction of positive, ground literals
* Closed world
* Goal partially specified: state satisfies goal if s contains all literals in g
* Action, precond, effect
* RESULT(s,a) = (s – DEL(a)) ∪ ADD(a)
  + DEL(a): negative effect literals of the action
  + ADD(a): positive effect literals of the action

**PDDL**

* Conjunction of positive and negative literals
* Allows variables and inequality predicates (e.g. On(b, x) ^ (b≠x))
* Forward search
  + Similar to tree search, expensive
  + Needs good heuristics
* Backward search
  + Uses descriptions: conjunction of fluents that applies to a set of states (e.g. (~Poor ^ Famous) describes (~Poor ^ Famous ^ Young)
  + Start from goal, s’ = (s – ADD(a)) ∪ Precond(a)
  + Relevant actions must have at least 1 effect that unifies with goal, no effect that negates an element of the goal
  + Can use MGUs to decrease branching

**SATPlan**

* + Conjunctive normal form (e.g. (A ˅ B) ^ (~A ˅ ~C)
  + Translating STRIPS to SAT:
    1. Express init state as ^ of positive and negative literals (literals not present due to closed world STRIPS)
    2. Define 1 possible goal using goal literals joined with ^
    3. Combine all possible goals using ˅
    4. Add successor state axiom: Ft+1 ↔ ActionCausesFt ˅ (Ft ^ ~ActionCausesNotFt)
    5. Add precond axioms: At → PRE(A)t
    6. Add action exclusion axiom: only 1 action at a time (e.g. ~(Ati ^ Atj) aka (~ Ati ˅ ~ Atj))
    7. Conjunction of all the axioms, transform into CNF
    8. Use SAT solver
  + Frame problem: for m actions and n fluents, need O(mn) axioms
  + Semidecidable: unsure how many steps required
  + Usually decidable due to finite states

**Planning Graph**

* For l literals, a actions, n levels
* Si: O(l) nodes, O(l2) mutex
* Ai: O(a+l) nodes, O(a+1)2 mutex, O(al+l) precond and effects
* Entire graph: O(n(a+l)2)
* Level-off: graph stops with 2 consecutive levels are the same
* GraphPlan algorithm: build graph starting from S0, search backwards
  + No-goods: store failed subproblems in lookup table
  + Will terminate since some properties increase/decrease monotonically
  + Increase: literals, actions
  + Decrease: mutex, no-goods

**Hierarchical Planning**

* High level actions, refined using other HLAs or primitives
* If a sequence of primitive actions does not appear in any of the refinement methods, will not appear in solution
  + Legal sequences would already be in refinement
* Check if reachable set of HLA overlaps with goal
  + Reachable set: set of states reachable by any implementation of the HLA
  + Represented by , meaning might add/remove
  + Reachable set of a sequence: union of reachable sets

**2. DECISION THEORY**

**Utility Axioms**

* Orderability: (A > B), (B > A) or (A ~ B)
* Transitivity: (A > B) ^ (B > C) → (A > C)
  + Only for single agent, group preferences may not be transitive
* Continuity: A > B > C → [p, A; (1-p), C] ~ B
* Substitutability: A ~ B → [p, A; (1-p), C] ~ [p, B; (1-p), C]
* Monotonicity: A > B → (if p > q, [p, A; (1-p), B] > [q, A; (1-q), B])
* Decomposability: lotteries are independent, can simplify using probability laws
  + [p,A; (1-p),[q, B; (1-q), C]] ~ [p,A; (1-p)q,B; (1-p)(1-q),C]

**Utility Functions**

* U([p1, S1; …; pn, Sn]) = ∑pi U(Si)
* Expected monetary value (EMV)
  + L1: [p, U($k); (1-p), U($k + high risk return)]
  + L2: U($k + low risk return)
  + Risk averse: L1 < L2
  + Risk seeking: L1 > L2

**Decision Network**

* Bayesian:
* Influence diagrams:
  + Chance node: random variable (circle)
  + Decision node: action choice (rectangle)
  + Utility node: util function (diamond)
* Evaluation:
  + 1. If observed, set parent variables to evidence
    2. Use bayesian to calculate posterior of parent nodes
    3. For each action at action node, use parent posterior to find expected utility
    4. Return action with highest utility

**Value of Information**

* No info:
* Perfect info:
* Value of info: VPIe(Ej) = E(EU(aej|e, ej) – EU(a|e) =
* VPI is not additive
* VPI is order independent
* Agent should choose to max VPI/Cost

**3. MARKOV DECISION PROCESS**

**MDP**

* Sequential, fully observable, stochastic, no history
* Can be represented by state transition graphs
* Value iteration
  + Bellman update:

* Number of iterations:

* Policy iteration:
  + Policy evaluation (a = π[s]):

* + Policy evaluation equations are linear, O(n3) time
  + Policy update:

**4. ONLINE SEARCH**

**Online Search**

* Alternating action and observation nodes, size = |A|D|S|D
* At observation node, compute expected values of children
* At action node, compute max of children
* Sparse sampling: sample k observations instead of |S|, tree size = |A|D|k|D
* MCTS
  + Return of a trial: rt(n) = R(s) + γrt(n’)
  + Actions should be chosen to balance exploration vs exploitation (e.g. using ɛ-greedy)
  + Average return of all trials:
  + Average return of trials from n using a: (s,a)
* UCT
  + Action selection:

* + Confidence interval decreases with time, converges
  + Worst time: Ω(exp(exp(…(exp(1)))), D-1 times, but good in practice

**5. PASSIVE REINFORCEMENT LEARNING**

* Fixed policy
* Learn transition, rewards, and state value

**ADP Agent**

* Monte Carlo Learning
  + is the return (estimate of U(s)) of just the ith trial
  + Note that s may appear more than once in 1 trial. Different ways to consider the returns
  + Both methods converge to the same: consider only first appearance, consider all appearances
  + Prediction error:

Error of the kth return:

* Need to wait for trail to be over to get return

**TD Agent**

* Learning rate, α, should be small

**6. ACTIVE REINFORCEMENT LEARNING**

**Exploration vs Exploitation**

* Agent policy update is greedy, need exploration
* ɛ-greedy: (1-ɛ) chance of greedy, ɛ chance of random
  + ɛ should decrease with time (e.g. ɛ = 1/t)
* Optimistic estimate
  + Similar to UCT
  + Exploration function: f(u, n), approaches u as n increases

**Model-based RL**

* ADP Agent
  + Replace policy evaluation with policy iteration
  + Choose action with exploration

**Model-free RL**

* Does not learn transition and rewards
* Q-learning
  + Similar to Bellman update
  + Off-policy: does not need to know next action a’
  + Takes the max of all next actions
  + s, a ← s’, action chosen with exploration
* SARSA
  + Similar to TD Agent but does not follow fixed policy
  + On-policy: must do policy update
  + Remembers bad actions

**Policy Search**

* Policy parameterised by θ
* Stochastic policy: probability of selecting action a in state s
* Continuous, can do gradient descent
* Softmax

  + Reinforce: uses MC update
* Use (Q(s, a) – B(s)) instead of Q(s, a) to reduce variance
  + If B(s) is estimated V(s),
  + Advantage function: A(s, a) = Q(s, a) – V(s)
  + Actor-critic: TD update using value function estimator V(s, w) where w is the parameter

**Function Approximation**

* Linear function of features:
* MC update: , uj(s) is the return on the jth trial
* TD update:
* Q-Learning:
* Non-Linear FA: DQN
  + Uses experience replay
* Note that using active learning with Non-linear FA tends to lead to divergence

**7. PARTIALLY OBSERVABLE MDP**

**POMDP**

* Sensor model: P(e|s)
* Transition: P(s’|s,a)
  + α is the normalising constant, makes sum = 1
* Probability of percept:
* Belief transition:
* Reward function:
* Conditional plan: p = π(b)
* Return of plan:
* Expected utility:
  + Up(b) is a linear function, hyperplane in belief space
  + Number of plans:
  + Dominated plans are never optimal
* Value iteration
  + Only works on small problems
* Dynamic Decision Network (DDN)
  + st = {Xt}, e = {Et}
  + Transition: P(Xt+1|Xt, At), Sensor model: P(Et|Xt)
  + POMCP
  + DESPOT

**8. GAME THEORY**

**Single-move Games**

* Pure strategy
  + Dominant strategy equilibrium
  + 1 unique nash equilibrium
  + Strongly dominates: s’ > s for all opponent’s strategies
  + Weakly dominates: s’ > s for at least 1 opponent strategy and not worse for the rest
  + Pareto optimal: no other outcomes that all players prefer
  + Pareto dominated: all players would prefer the other outcome
  + Iteratively eliminate strictly dominated strategies
* Mixed strategy
  + Assume probability distribution of actions
  + Find when other player is indifferent given the probabilities

**Zero-sum Game**

* Player 1 will try to maximise U1, player 2 will minimise U2
* Similar to mixed strategy

**Repeated Games**

* Use backwards induction
* Situation usually does not improve since outcome of games are independent from history