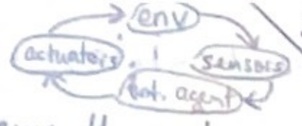
Intelligent agents



Def.: - perceives the env through sensors
- acts through actuators

Percept sequ.: complete history of perception

Agent func: Maps percept sequence to action
↳ Tabular func / agent programs

Rationality: select actions that max. expected performance
given: - performance measure - prior percept sequ.
- built-in knowledge - possible actions

Omniscient: knows actual outcomes of its actions

Task env description: - **P**erformance measures - **E**nvironment
- **A**ctuators - **S**ensors

Autonomy: Rely less on prior knowledge, use learned abilities instead

Task env properties:

→ **Fully observable:** Agent can detect complete env state, otherwise **partially**
(rational agent always wins/draws)

→ **Multi-agent / Single-agent**

→ **Deterministic:** Next state fully determined by current, otherwise **stochastic**

→ **Episodic:** Action in 1 episode don't affect later episodes, otherwise **sequential**

→ **discrete / continuous:** in space / time

→ **static:** Env only changes based on agent action, otherwise **dynamic**

→ **known:** Outcome (probabilities) of actions are known, otherwise **unknown** (→ env must be learned)

Simple Reflex Agent: Action based on current percept → only in fully obs. env

(extends)

Model-based Reflex agent: Handles partial obs. w/ world model
→ keeps internal state of prev. situation

Goal-based Agent: Keep track of goal
→ Reach goal by search + planning

Utility-based agent: Reach goal while max. utility → goal tradeoff to reveal the best course of action to maximize the perform. measure

Learning agent:
* Performance element
* Learning element
* Critic
* Problem generator

Uninformed Search

Tree Search: Init F

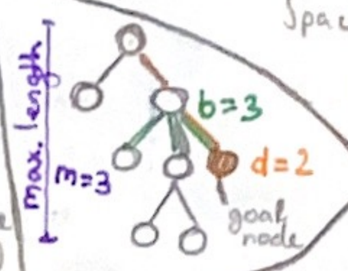
loop: if F = ∅ return Fail
pop node from F
if node = goal return solution
expand node, add children to F

Problem: loops for inf. search tree

Graph Search: Init F, Init E = ∅

loop: if F = ∅ return Fail
pop node from F
if node = goal return solution
add node to E
expand node, add children to F if ∉ {F, E}

BFS: → Sort nodes in F by shallowest depth
* F is FIFO queue
* Goal-test before adding a child to F
→ complete if d, b finite
→ optimal for equal step costs
→ Complexity: Time: $O(b^d)$ (d = depth)
Space: $O(b^d)$ (b = branching factor)



UCS/Dijkstra:
* F is priority queue by path cost
* Goal-test after popping from F
* When expanding, replace (only in BFS) frontier nodes w/ higher cost
→ complete if costs > 0
→ optimal if costs ≥ ε > 0 (ε = min. step cost)
→ Complex.: S/T: $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$ (C^* = cost of opt. solution)
goal node not in E

DFS: (iterative) order by depth
* like BFS w/ LIFO queue
* Complete if iterative implem., finite state space and E set
⇒ Does not terminate in inf. space w/ tree search (cyclic graph)
* Not optimal
* Complexity: Time: $O(b^m)$
Space: $O(bm)$

DLS: (recursive) "cutoff at limit"
* Goal-test when expanding a node
* Incomplete if $l < d$ (l = limit)
* not optimal if $l > d$
* Complexity: Time: $O(b^l)$
Space: $O(b \cdot l)$

IDS: uses DLS and iteratively increases l
* Complete if d finite
* Optimal for equal costs
* Complexity: Time: $O(b^d)$
Space: $O(bd)$

Bidirectional Search:
* Only if all actions are reversible and goal state is known

DFS: (recursive)
⇒ like iterative DFS + add goal node to F
* Properties of DLS

Informed Search

→ like UCS
→ eval func. $f(n)$ chooses next node in priority queue
(heuristic / estim. cost to goal)

GBFS: $f(n) = h(n)$
* $h(n) > 0$ and $h(\hat{n}) = 0$
→ expands node closest to goal
* Complete w/ graph search
* not optimal
* Complexity: T/S: $O(b^m)$

A*: $f(n) = g(n) + h(n)$ (g = path cost)
* $h(n) \leq g(n, goal)$. Must be admissible / lower than real cost
* Consistency for graph search
$h(n) \leq c(n, a, n') + h(n')$ (c = transition cost)
* complete if costs > 0
* optimal if costs > 0 and { h^* admissible, tree search; h consistent, graph search }
* Complexity: S/T: $O(b^{\varepsilon d})$ (ε = rel. error)
with $\varepsilon = \frac{h^* - h}{h^*}$ (h = estimated costs to goal, h^* = actual costs to goal)
⇒ Only paths with $f(n) \leq C^*$ are expanded (auto pruning) (C^* = cost of opt. path)
→ $f(n) > C^*$ never expanded

- Iterative-deepening A* → f(n) for cutoff
- Recursive BFS
- Memory-bounded A* → drop less promising paths

Effective branching factor b^*:
= avg # of nodes to expand at each step to reach a solution
N + 1 = 1 + $b^* + (b^*)^2 + \ldots + (b^*)^d$ (N = # of nodes generated by A* search; find iteratively)

- Derive h from relaxed problem
- comb. heuristics: $h(n) = \max\{h_1(n), h_2(n), \ldots\}$
- h_2 dominates h_1 if: $h_2(n) \geq h_1(n)$ ∀ n

CSP

is tuple (X, D, C)
(X = set of variables, D = set of domains, C = set of constraints)

D_i = set of allowed values
$C_i = \langle scope, rel \rangle = \langle$ tuple of particip. variables, connect. relation $\rangle$

Constraint	# of involved variables
Unary	1
Binary	2
Higher-Order	≥3

Hidden transformation: Turn n-ary constr. into binary constr.
1. Replace constr. with var. Z. dom(Z) is n-tuple and satisfies the constr. relation → dom(Z) = {(z1, z2) | ...}
2. New binary constr. match Z w/ neighbors (fst(Z) = A, ...)

Inference techniques:
→ **Forward checking:** Remove inconsistent vals of neigh. vars
→ **Arc consistency:** A is arc-cons. w/ B if for every value in dom(A) there is a value in dom(B) that satisfies the binary constraint of arc(A, B)

queue	removed?
(B, A)	true
(C, A)	⋮
⋮	

1. check if vals need to be removed from dom(B) to satisfy arc(B, A)
2. Then add neighbor arcs of B w/o A.

Backtracking Search: (→ DFS, all solutions at depth n w/ n vars)
Degree = # of constr. w/ unassigned neighbors
1. Init w/ max domain size and degree
2. Variable selection:
↳ **min.** # of remaining values (MRV)
↳ degree heuristic: highest degree
3. Value selection:
↳ least-constraining-value heuristic
4. Inference (Forward-Checking, Arc-constistency-alg)
↳ Prunes search tree
5. Backtrack to 3/2 if a domain size = 0
6. Repeat from 2.

=> Also use AC as preprocessing to reduce backtracking + one-step solution [initialize the queue w/ all arcs of the CSP]
→ Tree-structured CSP solved in $O(nd^2)$
→ General CSP: $O(d^n)$

Logical Agents (Propositional Logic)

Model: Instance which evaluates models to true or false.
Satisfaction: If sentence α is true in model m, then m satisfies α.
Entailment: $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$ ⇔ iff the set of models that evaluates α (to true) is contained in the set of models which eval β.
Validity: If a sentence is true in all models (→ tautology) (M(α) = AllModels)
Satisfiability: If a sentence is true in some model (→ unsatisf. M(α) = ∅)
→ α is valid if ¬α is not satisfiable
Logical equivalence: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
→ α is valid iff α ≡ True
→ α is unsatisfiable iff α ≡ False

Show KB ⊨ α:
1. Enumerate all models + check if α is true in every model where KB true
2. **Resolution Alg.:**
1. Convert KB ∧ ¬α to CNF: $\wedge_i \vee_j (\neg) \alpha_{ij}$
2. apply resolution
3. until: no new clauses (KB ⊭ α) / (KB ∧ ¬α is unsatisf. { empty clause (KB ⊨ α))

Modus Ponens: $\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$
And-Elimination: $\frac{\alpha \wedge \beta}{\alpha}$
Full Resolution ex: $\frac{A \vee B, \neg B \vee C}{A \vee C}$

Horn clause: • symbol **or**
• (conjunction of symb) ⇒ symbol
⇒ Only needs Modus Ponens for Inference
⇒ needed for Forw/Backw-Chain.

AND-OR graph L ∧ M ⇒ P
conj. **Forw. chaining:**
1) Fire rules whose premises satisfied by KB
2) add conclusion to KB
3) repeat until at goal

Standard Logical Equivalences:

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$
$\neg(\neg\alpha) \equiv \alpha$
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \Rightarrow \neg\beta)$
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ Implication elimination
$(\alpha \Leftrightarrow \beta) \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ Bicond. elim.
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ De Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$

$(A \Leftrightarrow \neg A) \equiv$ False
$A \wedge \neg A \equiv$ False
$A \vee \neg A \equiv$ True

P/Q	P/F	F/T	T/F	T/T
P ⇒ Q	T	T	F	T
P ⇔ Q	T	F	F	T

FOL

∀: extended interpr. w/ ∧
∃: extended interpr. w/ ∨

Syntax: Backus-Naur-Form

Sent := AtomSent | CpxSent
AtomSent := Pred | Pred(Term, ...) | Term = Term
CpxSent := (Sent) | [Sent] | ¬Sent | Sent ∧ Sent | Sent ∨ Sent | Sent ⇒ Sent | Sent ⇔ Sent | Quant Var ... Sent
Term := Func(Term, ...) | Const | Var
Const := A | X1 | John | ... (upper-case)
Pred := True | False | After | Loves | ... (returns true/false)

Other elements:

Var := a | x | s | ... (lower-case)
Quant := ∀ → usually main connective ⇒ ; ∃ → usually main connective ∧
Operator precedence: =, ¬, ∧, ∨, ⇒, ⇔
Func := Mother | LeftLeg | ... (returns obj)

Duality:

$\neg\exists x\ P \equiv \forall x\ \neg P$
$\neg\forall x\ P \equiv \exists x\ \neg P$
$\forall x\ P \equiv \neg\exists x\ \neg P$
$\exists x\ P \equiv \neg\forall x\ \neg P$
$(\forall x\ \phi) \Rightarrow \psi \equiv \exists x (\phi \Rightarrow \psi)$
$(\exists x\ \phi) \Rightarrow \psi \equiv \forall x (\phi \Rightarrow \psi)$
$\phi \Rightarrow (\exists x\ \psi) \equiv \exists x (\phi \Rightarrow \psi)$
$\phi \Rightarrow (\forall x\ \psi) \equiv \forall x (\phi \Rightarrow \psi)$

Inference in FOL

Universal Instantiation: $\frac{\forall v\ \alpha}{Subst(\{v/g\}, \alpha)}$ (g = const)
↳ all possible Subst.
⇒ new KB is log. equiv.
Existential Instantiation: $\frac{\exists v\ \alpha}{Subst(\{v/k\}, \alpha)}$
↳ new KB not log. equiv.

Backward-Chaining: Init goals = {goal}, Init θ = ∅ (to prove a goal sentence)
1. goals = {...}
2. q' ← Subst(θ, Fst(goals))
3. choose Standardize-Apart sentence from KB $p_1 \wedge \ldots \wedge p_n \Rightarrow q$
4. θ' ← Unify(q, q') s.t. Subst(θ, q) = Subst(θ, q')
5. goals' ← {$p_1 \wedge \ldots \wedge p_n$, Rest(goals)}
6. Break if goals' is empty

CNF:
1. Eliminate implications
2. Move ¬ before quantifiers inwards
3. Standardize variables
4. Skolemization (replace E.Q. vars with F(...) (is func of all vars from the U.Q.)
5. Drop U.Q.
⇒ $\wedge_i \vee_j (\neg) \alpha_{ij}$

Resolution: → just like PL + Unify variables (left from U.Q.)

Bayesian Networks

Marginalization: $P(X=x) = \sum_{y \in D_Y} P(X=x, Y=y)$

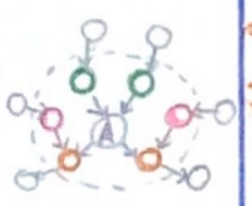
Bayes Rule: $P(X|Y) = \frac{P(X,Y)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$

Independence: $P(X|Y) = P(X)$ and $P(X,Y) = P(X)P(Y)$
↳ no common ancestry

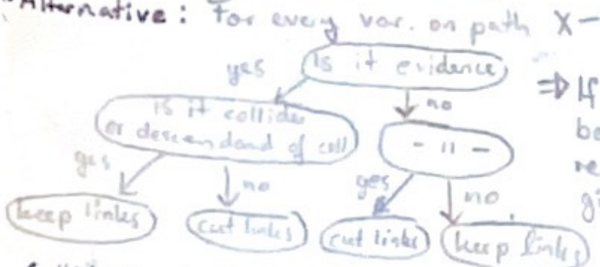
Cond. Independence: $P(X|Y,E) = P(X|E)$ or $P(Y|X,E) = P(Y|E)$

Netw.: Nodes ≙ RV w/ cond. prob. distr. $P(X_i|\text{Parents}(X_i))$
Arrows start at parents

Determine cond. ind.:
* Markov Blanket: of all other nodes given its parents, children & children's parents



* Alternative: For every var. on path X−Y check:



⇒ If no undir. links between X and Y remain → cond. ind. given E

Collider: Node C with neighbors A and B, s.t. A→C←B

Joint distr.: $P(X_1,\dots,X_n) = \prod_{i=1}^{n} P(X_i|\text{Parents}(X_i))$

Inference by enumeration:

e.g. $P(B|j,m) = \alpha \cdot P(B,j,m) = \alpha \sum_e \sum_a P(B,e,a,j,m)$
sum over hidden vars

Inference by variable elimination:
Start w/ eq. from enumeration; shorten eq. by reordering the sums

$P(B|j,m) = \alpha P(B) \sum_e P(e) \sum_a P(a|B,e) P(j|a) P(m|a)$

Choose axes, stay consistent! → multiply from right to left

$f_1(B)$ $f_2(E)$ $f_3(A,B,E)$ $f_4(A)$ $f_5(A)$

$= \alpha [P(b|j,m)\ P(\neg b|j,m)]$

Monte Carlo Simulation: Since exact inferece is too expensive
① Sample from given prob. distr.
② Deterministic Sim.
③ Aggr. of indiv. det. Sim. for exp. values of prob. distr.

→ Direct sampling:
Prob that a specific event is sampled $S_{PS}(x_1,\dots,x_n) = \prod_{i=1}^{n} P(x_i|\text{Parents}(x_i)) = P(x_1,\dots,x_n)$
Estimates are consistent $\hat{P}(x_1,\dots,x_n) = \frac{N_{PS}(x_1,\dots,x_n)}{N} \to P(x_1,\dots,x_n)$ for $n \to \infty$
($\hat{P}$: estimate; N_{PS}: # of samples for this event)

→ Rejection sampling: sample from hard-to-sample distr. given easy-to-sample one
Estimated distr. $\hat{P}(X|e) = \alpha N_{PS}(X,e) = \frac{N_{PS}(X,e)}{N_{PS}(e)} \approx \frac{P(X,e)}{P(e)} = P(X|e)$
⇒ Expensive if $P(e)$ small

→ Likelihood weighting: sample only events that include e
Sampling prob of non-evidence vars $S_{WS}(z,e) = \prod_i P(z_i|\text{Parents}(Z_i))$
Likelihood weight $w(z,e) = \prod_i P(e_i|\text{Parents}(E_i))$
⇒ $P(z,e) = S_{WS}(z,e) \cdot w(z,e)$

Rational decisions over time (Making complex decisions)

MDP: 5-tuple: (S, A, P, R, γ)
finite set of states S; finite set of actions A(s); $P_a(s,s') = P(s_{t+1}=s' | s_t=s, a_t=a)$; $R(s,a,s')$ immed. reward; γ discount factor

Value iteration alg.:
Bellman update: $U^{i+1}(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|a,s) U^i(s')$

Opt. policy: $\pi^*(s) = \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s'|a,s) U(s')$

Policy iteration alg.:
1. Policy evaluation: $U_i(s) = R(s) + \gamma \sum_{s'} P(s'|s,\pi_i(s)) U_i(s')$
↳ Solve sys. of lin. equations
2. Policy improvement: $\pi_{i+1}(s) = \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s'|s,a) U_i(s')$

→ stop when

HMMs

unobs. state X_t at each time step
obs. evidence E_t

Stochastic process: Sequence of RVs
Markov process: Stoch. process w/ Markov property
Markov property: If $P(X_n|X_{n-1},\dots,X_0) = P(X_n|X_{n-1})$
Stationary process: stoch. process whose joint prob. distr. does not change when shifted in time

$\forall t: P(X_n = x_i | X_{n-1} = x_j) = P(X_{n+t} = x_i | X_{n-1+t} = x_j)$

Stationary MP: = discrete stat. process w/ Markov property

$P(X_n = x_i) = \sum_{j=1}^{N} P(X_n = x_i | X_{n-1} = x_j) P(X_{n-1} = x_j)$

Trans. model $T = P(X_t | X_{t-1})$ cols sum to 1; T_{ij} ⇒ $P_n = T \cdot P_{n-1}$ w/ $(P_i)_n = P(X_n = x_i)$

Markov chain of m^{th} order depends on m previous states.
⇒ introduce states that correspond to a sequence of m states

e.g.
$\hat{X}_1 = (X_1, X_2)$, $\hat{X}_2 = (X_2, X_3)$, $\hat{X}_3 = (X_3, X_4)$, $\hat{X}_4 = (X_4, X_5)$
$P(X_n = x_1 | X_{n-1} = x_3, X_{n-2} = x_4) \triangleq P(X_n = \hat{x}_1 | X_{n-1} = \hat{x}_4)$
...
$P(X_n = x_2 | X_{n-1} = x_3, X_{n-2} = x_2) \triangleq P(X_n = \hat{x}_3 | X_{n-1} = \hat{x}_2)$
...

Convert to scalar RVs: to model task as HMM
e.g. $X = [\text{IsCat}, \text{LivesIn}] \to \hat{X} = [x_1, x_2, x_3, x_4]$
w/ $x_1 = [\text{true}, \text{Manila}]$, ...

Sensor/Obs. model: (only depend on current state)
$P(E_t | X_{0:t}, E_{0:t}) = P(E_t | X_t) = H$ (H_{ij}: x_j → e_i)
⇒ $\hat{P}_n = H \cdot P_n$ w/ $(P_i)_n = P(E_n = e_i)$

→ Joint state&evidence distr.:
$P(X_{0:t}, E_{0:t}) = \left[\prod_{i=1}^{t} \underbrace{P(E_i|X_i)}_{H} \underbrace{P(X_i|X_{i-1})}_{T}\right] P(X_0)$

Inference tasks in HMMs:

Filtering: What's the belief state given prev. evidence
$P(X_{t+1}|e_{1:t+1}) = \alpha \underbrace{P(e_{1:t+1}|X_{t+1})}_{H} \sum_{x_t} \underbrace{P(X_{t+1}|x_t)}_{T} P(x_t|e_{1:t})$

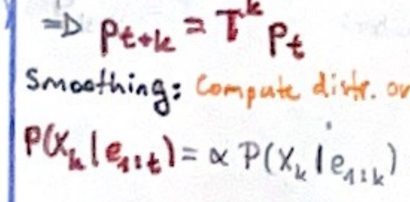
$f_{1:t+1} = \alpha O_{t+1} T f_{1:t}$ w/ $(f_i)_{1:t} = P(X_t = x_i | e_{1:t})$
initial $f_0 = \begin{pmatrix} P(x_0) \\ P(x_1) \\ \vdots \end{pmatrix}$
$(O_{ij})_{t+1} = \begin{cases} P(e_t|X_t = x_i), & j=i \\ 0, & \text{else} \end{cases}$

Prediction: Like filtering w/o new evidence
$P(X_{t+k+1}|e_{1:t}) = \sum_{x_{t+k}} P(X_{t+k+1}|x_{t+k}) P(x_{t+k}|e_{1:t})$
⇒ $p_{t+k} = T^k p_t$

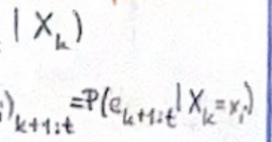
Smoothing: Compute distr. over past states given evidence up to the present
$P(X_k|e_{1:t}) = \alpha P(X_k|e_{1:k}) \times P(e_{1:k+t}|X_k)$
$= \alpha f_{1:k} \times b_{k+1:t}$ w/ $(b_i)_{k+1:t} = P(e_{k+1:t}|X_k = x_i)$
$b_{k+1:t} = T^T O_{k+1} b_{k+2:t}$ w/ $0 \le k < t$, $b_{t+1:t} = \vec{1}$

Most likely explanation: Find most likely sequence of states given the evidences
→ Viterbi alg.:
$\max_{x_1,\dots,x_t} P(x_1,\dots,x_t, X_{t+1}|e_{1:t+1}) = \alpha O_{t+1} P(e_{t+1}|X_{t+1}) \cdot \max_{x_t} \left[P(X_{t+1}|x_t) \max_{x_1,\dots,x_{t-1}} P(x_1,\dots,x_{t-1}|e_{1:t})\right]$
($\mu_{t+1}(X_{t+1})$)
⇒ $\mu_{t+1}(X_{t+1}) = \alpha O_{t+1} \max_{x_t} P(X_{t+1}|X_t)\, \mu_t(X_t)$

① calc. this indiv. for each X_t (col. in T, row in μ_t)
end node: $\operatorname{argmax}(\mu_3)$
② Mark most likely predecessors X_t



Robotics



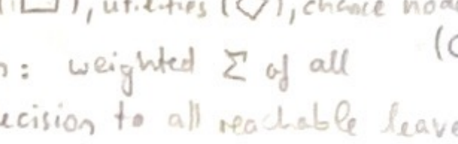
3 Laws of Rob.: 1) May not injure human or allow to come to harm
2) Must be given orders by human, except when conflict w/ 1)
3) Must protect itself unless this conflicts w/ 1) or 2)
Primary categories: Manipulators, Mobile Robots, Mob. manipulator
Passive sensors: Detect effect gen. by other source in env → Camera, GPS
Active sensors: Send signal into env and receive refl. → SONAR, LIDAR
Proprioceptive s.: Inform on robot's own state → angle sensor, acc. sensor
DOF: # independent directions for movement
↳ non-holonomic: more effective DOFs than controllable DOFs
→ holonomic: ... = ...
Localization: Find out where things are (incl. robot)
→ Landmarks: Stable, recognizable features in env for navig.
→ map to known map
→ SLAM: simultaneous localization and mapping

Rational Decisions (Making simple decisions)

Outcome prob.: $P(\text{Result}(a) = s') = \sum_e P(e) P(s'|s,a)$
Exp. utility: $EU(a|e) = \sum_{s'} P(\text{Result}(a) = s' | a, e)\, U(s')$
Rules: $EU(x,y) = P(x,y) U(x,y)$, $EU(x) = \sum_y EU(x,y)$
$EU(x|y) = \frac{EU(x,y)}{P(y)}$

Preference constraints: preference, indifference
Orderability $(A \succ B) \vee (B \succ A) \vee (A \sim B)$
Transivity $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$ (prize)
Continuity $A \succ B \succ C \Rightarrow \exists p\ [p, A; 1-p, C]$ (lottery)
Substitutability $A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$
Monotonicity $A \succ B \Rightarrow (p \succ q \Leftrightarrow [p, A; 1-p, B] \sim [q, A; 1-q, B])$
Decomposability $[p, A; 1-p, [q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$

Exp. utility of lottery: $U([p_1, s_1; \dots; p_n, s_n]) = \sum_i p_i U(s_i)$
Multiattribute utility: $U(X_1, \dots, X_n)$
→ strict dominance: $\forall i\ X_i(A) \ge X_i(B) \Rightarrow U(A) \ge U(B)$
→ Stoch. dominance: $\forall t \int_{-\infty}^{t} P_1(x)dx \le \int_{-\infty}^{t} P_2(x)dx \Rightarrow P_1$ dominates P_2



Decision trees: decisions (□), utilities (◇), chance nodes (○)
⇒ Exp. utility of decision: weighted Σ of all branches of the decision to all reachable leaves.

Value of information:
→ Val. of the curren best action: $MEU(\alpha|e) = \max_a \sum_{s'} P(\text{Result}(a) = s'|e) U(s')$ (best action, initial evidence)
→ Val of the new best action: $MEU(\alpha_{e_j}|e, e_j) = \max_a \sum_{s'} P(\text{Result}(a) = s'|e, e_j) U(s')$ (new best action, new evidence)

$VOI_e(E_j) = \left(\sum_k \underbrace{P(E_j = e_{jk}|e)}_{\text{weighted avg.}} \underbrace{MEU(\alpha_{e_{jk}}|e, E_j = e_{jk})}_{\text{max. exp. utility of getting the info for free}}\right) - \underbrace{MEU(\alpha|e)}_{\text{max. exp. utility w/o this info}}$

↓ non-negative, nonadditive: $VOI_e(E_j, E_k) \ne VOI_e(E_j) + VOI_e(E_k)$
~ Order-independent $VOI_e(E_j, E_k) = VOI_e(E_j) + VOI_{e+e_j}(E_k) = VOI_e(E_k) + VOI_{e+e_k}(E_j)$

Decision networks: smaller in size than decision trees

→ Partial ordering:
$X_0 < D_1 < X_1 < D_2 < \dots < X_{n-1} < D_n < X_n$
1) identify 1st dec. and all vars to make it
2) identify next dec and all vars. between D_1 and D_2
3) unrevealed vars at the end

→ optimal decision:
$\pi^*(D|e)$ $\pi^*(e) = \operatorname{argmax}_{d_i} EU(d_i|e)$
⇒ Optimal plan: * Eval decisions backwards in P.O.
* Consider all options of earlier dec, margenalize out later chance nodes
* continue with earlier decision, use optimal dec. π^* for later dec.

Learning (Decision trees and the information gain heuristic)

Decision trees: Map attribute vector to boolean
→ for n attributes we have 2^{2^n} possible trees
→ Entropy of boolean RV: $B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$ (q: truth prob)
$B(0.5) = 1$, $B(1) = 0$, $B(0) = 0$
$\log_2 x = \frac{\log x}{\log 2}$
→ Choose next best split: Max. info. gain (split attribute)

$Gain(A) = B\left(\frac{p}{n+p}\right) - \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$

parent entropy; all child nodes; prob to choose it; child entropy

⇒ Then find next best split for child data sets

Reinforcement learning:
Exploration: Random next action for new experience
Exploitation: Use exist. policy for next action
On-policy: Only data collected w/ recent policy used for policy update
Off-policy: Learning update uses any data
Model-based: Used state transition model P(s'|a,s) known
Model-free: Learning from sampling & simulation → Q-Learning, → Policy-search