Subiecte IS

# Multiple agent Decisions. Game Theory. AIMA17

## Morra Game.

**O1E1: E=2, O=-2;**

**O1E2: E=-3, O=3;**

**O2E1: E=-3, O=3;**

**O2E2: E=4, O=-4;**

**Show app of max technique;**

**Max equilibrium;**

**Explain Min-Max game tree**

-Poze de pus aici.

## Payoff matrix

Matrix with the values for all agents.

## Prisoners Dilemma

**2 prisoners: Alice and Bob. They can testify or refuse.**

**Ar, Br => -1, -1**

**At, Br => 0, -10**

**Ar, Bt => -10, 0**

**At, Bt => -5, -5**

Using min max we get for Bob testifies: Ur = -10, Ut = -5 => Testifying is a better choice.

Using min max we get for Bob refuses: Ur = -1, Ut = 0 => Testifying is a better choice.

Testifying is the dominant state. This state is also considered an equilibrium state.

## Nash equilibrium

Is achieved when the game reaches a point in which each agent knows that any attempt at drifting from the current state will result in the detriment of itself.

Can be more per game.

## Pareto Optimality

A state/strategy is so when we cannot change it without one of the other agents/players/individuals losing. Also, best state for all players.

Can be more per game.

# Knowledge in learning:

## Foil

**George x Mum -> [Margaret, Elizabeth]**

**Elizabeth x Phillip -> [Charles, Anne, Andrew, Edward]**

**Spencer x Kydd -> [Diana]**

**Diana x Charles -> [William, Harry]**

**Anne x Mark -> [Peter, Zara]**

**Andrew x Sarah -> [Beatrice, Eugenie]**

**Apply FOIL to learn the def of “ancestor/2” function.**

We presume we have the “parent/2” function.

-Solution

We divide the examples in positives and negatives. Used to determine if we need to shrink (AND) or widen (OR) the domain.

We first classify all the examples as true. => All examples are positives => we need to restrict.

We need to at least have 1 common attributes to each side.

Some optimizations are also present that help the algorithm use relevant functions for finding the solution.

Ockhams Razor: a heuristic that helps the alg. Determine what predicates to use next. Ex: by adding parent (a, b) => ancestor (a, b) -> restricts too much.

parent (a, b) = > ancestor (a, c) -> better but not restrictive enough => too many false-positives. (nu tre sa fie adevarate) => need to restrict the domain

parent (a, b) AND ancestor (b, c) => ancestor (a, c) -> we still have false-negatives (nu tre sa fie false). => we need to widen

(parent (a, b) AND ancestor (b, c)) OR parent (a, c) => ancestor (a, c)

## Explanation-Based Learning

**General rules from examples**

-having those examples, we get a deduction tree to which we add predicates till we solve it.

-extracted using variables instead of constants while building the trees

**Proof trees for simplification**

-the drawing to prove the predicate

**Improving efficiency**

- “Prune” -> we exclude some branches in order to obtain some more general rules. (Ex simp: we can stop at primitive/1 or at simplify(y+z,w)).

- 3 influencing factors: Branching factor (nr of used predicates, more is worse); Derived rules should be as general as possible; Choose rules that yield shorter proofs.

- Operationality -> short proogs

**Explicati pe Simplify(1\*(0+X),W)**

**a) Extragere regului generale din exemple**

**b) Imbunatatire eficienta**

-Poza

## Logical formulation of learning

-Find a hypothesis to classify our examples. => find a function

- 2 types of finding them: Current best vs Least statement

- For curr best: Operations to update the hypothesis: Generalization (Add a new case) or Specialization (remove a case).

- For least: At beginning, take all possible statements such that they are either too general or either too specific. Then take each example and test it for each statement => if the statement does not give the desired result, remove it.

Ex: grandparent = parent OR father OR mother OR (parent(x,z) AND parent (z,y)).

**Version space learning**

Is the space between the most general and the most specific hypothesis.

## Inductive logic programming

- 2 main approaches: Top down (FOIL. From general to specific), Inverse resolution.

**Inverse Resolution**

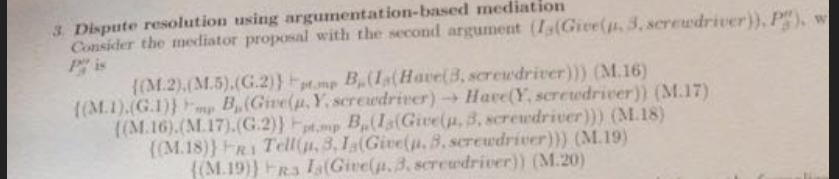
- Start from a specific example and get the general solution

- Poza grandparent

**Making decisions/discoveries (Inverse resolution)**

**-** Adds new predicates to the knowledge base if you need them to solve this function. (Ex. We may not have parent(G,\_), so we add one).

# Dispute resolution. Argumentation-based meditation. Trescak



-(1)Picture: Mediator tells an agent that he needs a screwdriver to hang the mirror and must request it from him.

-(2)Alternate Picture: How mediator figures out Y needs nail from other agent and that he needs to request it.

- A new agent, mediator, is used to solve disputes between multiple agents.

**Agent theories**

- Properties that must be respected by ALL agents (Agent theories)

G1-Ownership (X gives to Y sth => Y is the new owner of that sth)

G2-Reduction (Agent splits the big goal in smaller ones to solve first; Divid et impera)

G3-Generosity (Mediator wants to give all of his resources to other agents)

G4-Unicity (X, after giving someone sth, does not longer contain that sth)

G5-Benevolence (If X does not need sth, he will give it up if someone needs that sth)

G6-Parsimony (An agent will not do sth if he does not need to)

G7-Unique choice (If there are multiple ways to do sth, X will only chose one).

**Bridge rules**

-Properties for communication between agents.

R1/R2- Advice => Mediator will tell an agent how to operate, if the mediator thinks he knows what the agent is about to do OR how he wants to do sth.

R3- Trust in mediator => Agents trusts what the mediators tell them to do

R4- Request => If an agent needs sth, he will request it

R5- Accept request => If the agent can give a requested resource, he will give it.

**Arguments necessary to complete resolution.**

**-**(1) B ask A to give SCREW. B asks M to give SCREWDRIVER.

-(2) M infers that A to request NAIL from B. A requests nail from B.

# Making complex Decisions:

-When going from State S1 to state S2 we get a reward (can be + or -).

-A reward has a discount, that tells how far in the future that reward is.

## Value Iteration

**Utilities of states**

**-**The utility of a state is the expectancy of the sum of states that have a discount for a given policy.

-The TRUE utility of a state is the same but for the best possible policy.

**Bellman eq**

-Used to determine the utility of states.

-Utility(S1) = reward(S1) + discount(S2) \* max(utility). (utility of possible states from our current one)

-We can get multiple systems of bellman equations (that each have an unique solution) to determine the TRUE utility.

**Value iteration alg**

-We start from a random/approximate utility and we update it until the difference between 2 consecutive solutions is negligible.

-We use Bellman eq (Bellman update) to calculate the next utility: Uk+1 = max(Uk) \*discount(S2) + reward(S1).

**Policy iteration**

- Same as value but we update the policy.

- 2 Parts: Policy evaluation (Calc the utility of policies), Policy improvement (Choses policy with highest utility)

-U = policy evalution.

-foreach(state) Search what is the best move to do from all policies.

-\*\*\*\*

## Policy. Multiple sections.

**Policy loss**

**-** The most an agent can lose by using a policy other than the BEST policy.

**Modified policy iteration**

**-** We modify the Bellman eq: We need not calculate the max(U), since we will have a fixed policy. This applies to the policy evaluation step of the algorithm.

**Partialy observable MDP**

Similar to previous, but we no longer have states, we have belief states (B) and also sensor model (E).

We still have rewards, discounts and actions.

Can be split into 3 things: From a state B we do an action A and according to our policy we get a probability of the next state B1, that depends on B, A, and all the evidence our sensor S finds. One can apply the Rewords in a probabilistic way to get an approximate reward for this transition

Lage value space => hard to calculate

Alternative: Dynamic Decision Networks, Dynamic Beliefs, Dynamic Baycian and Dynamic Filters for belief states.

# Neural Networks. CBOW model. 1-word context

CBOW -> Continuous Bag Of Words

SkipGram -> mai greu

**Draw + explain the word2vec model under the one context simplification. CBOW**

The words from the Input Layer are copied to the Hidden Layer.

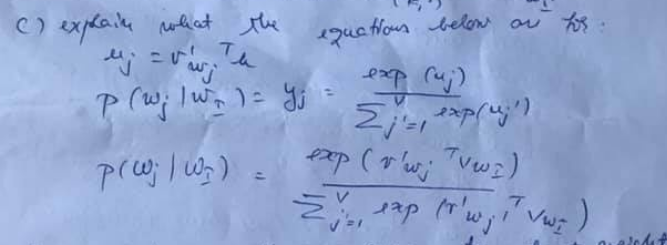
Given a context, the k-th row of W is copied H (Hidden layer). => the activation function for this is linear.

In W’ we use SoftMax as an activation function -> Obtains a multmodel distribution of words.

We then use backpropagation to update the weights in W’ and then W.

**Explain the working of the weights in the model: h=WTx = WT(k,\*) := vTW1**

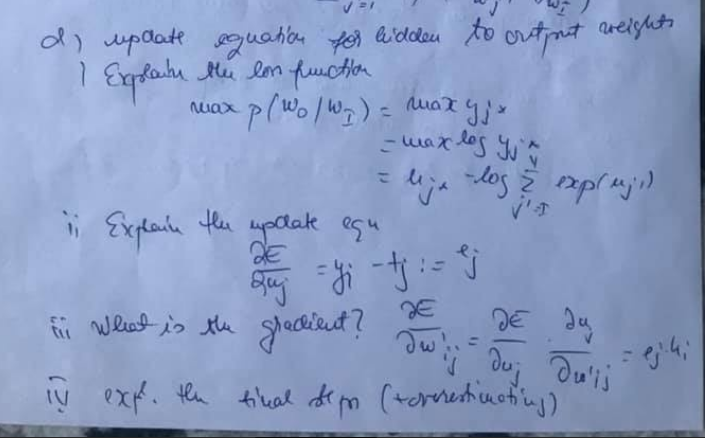
This model presents that the copy the words from the k-th row to the hidden layer



uj = the score of each word computed based on the vector from the hidden layer and the vectoral representation of a word from W’ (the column j of matrix W’).

softmax equation; yj the output

The last eq is the equation resulting from the replacement of uj with the value from the first equation in the second one.



We must maximize the probability that the output will be the correct one. -Loss.

We derivate E (Loss) according to uj and get the output j minus tj (1 if we found the correct word; 0 otherwise).

We derivate E according to w’ij by appling the normal partial derivation formula and get hi that is the second derivative (the el on the i-th position from the vector H). (This is the gradient)

We must now update the weights by subtracting the new weight with the learning rate times the computed derivative.

**Multi word grammar**

A bigger input layer with multiple W. Multi word context setting. An average will be computed for the vectors of the input context words.

**Skip Gramm**

Same but for output layer.

# Dynamic protocols for open agent Systems: CFCP

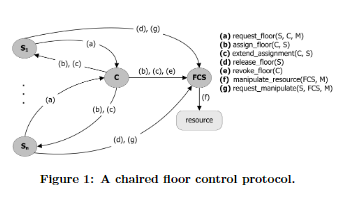
Used when a resource (floor) wants to be used by multiple agents (S). The assignment is handled by a chair (C).

There are multiple levels for this, in which the agents can vote to change the rules of this system.

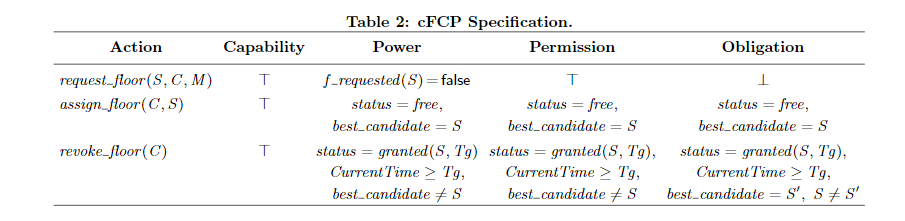
Level 0: The level that handles the resource.

Level N: Level that handles the protocols from the level N-1.

**Explain the figure for the cleaning floor control protocol**



**Table for the specification of the conditions of a CFCP participant with phys capabilities, some power, permissions and obligation to perform an action**



Capability => The action CAN be done

Power => If the agent can do the action

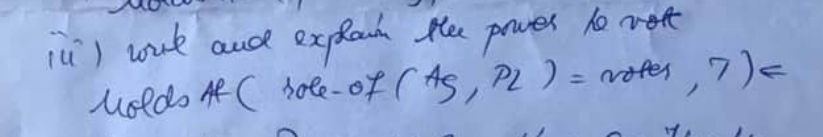
Permission => If the agent is allowed to do the action. Will be punished if the tires to and has no permission.

Obligation => What he must do, if the prerequisites are fulfilled. Will be punished if he does not do that.

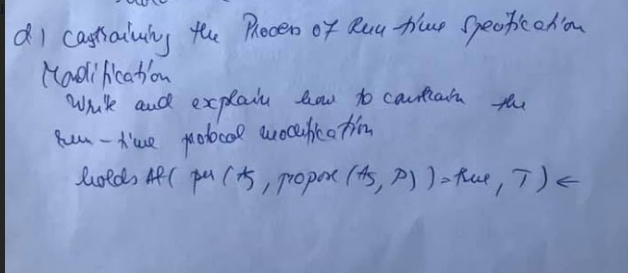
**Transition protocol: tule-set replacement; power to second a protocol;**

Transition protocol -> the protocol that must be followed to change a protocol at a lower level.

If a new motion is proposed, if at least one other agent seconds the motion, the motion can pass. If at least one other agent opposes to it, a vote will be held.



The agent Ag is a voter for the protocol level Pl at time T when he is the subj at level 0 when the protocol is executed (Pl > 0) and he is not punished/sanctioned.



The agent can propose a change to DOF at level Pl from OldVal to NewVal, if he is a subject at level 0, if there is no other protocol at level Pl + 1, if he knows what he wants to change is OldVal. (P is a function that changes the OldVal to the NewVal).

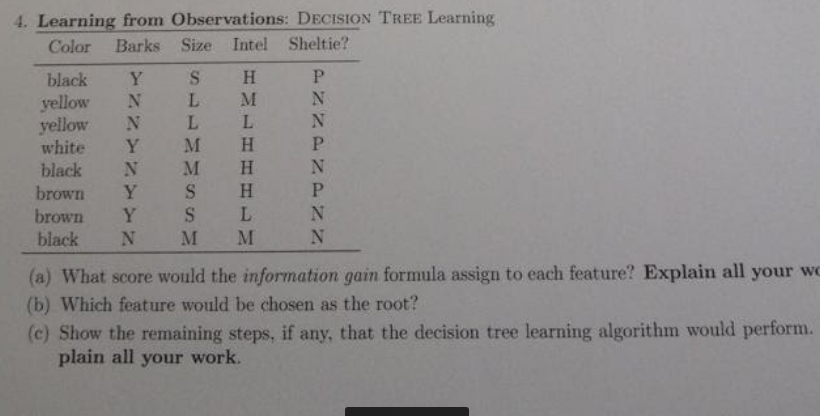
**Second a proposal**

One can second a proposal if one is subject at level 0 and there exists such a protocol that does not belong to him.

**Extra**

There is a constraint in place that does not allow the NewVal to differ greatly from OldVal. It is based on some kind of distance calculations that is compared to a given threshold.

# Learning from observation. Decision Tree learning



(a) Gain = B(p/p+n) – Remainder. (for each feature).

p- number of positives from the Target/What we want to prove (in our case Sheltie)

n- number of negatives from the Target (in our case Sheltie)

I(p(v1), p(v2), …) = sum(-p(vi) \* log2(p(vi)).

B(p) = -(p \* log2p + (1-p) \* log2(1-p)) = 0.95

Remainder = sum(((pi + ni)/p+n) \* B(pi/(pi+ni)))

Color: Black(1P,2N), Yellow(2N), White(1P), Brown(1P,1N). Gain = 0.95 – [3/8 \* B(1/3) + 2/8 \* B(0) + 1/8 \* B(1) + 2/8 \* B(1/2)] = 0.35

Barks: La fel. Gain = 0.54

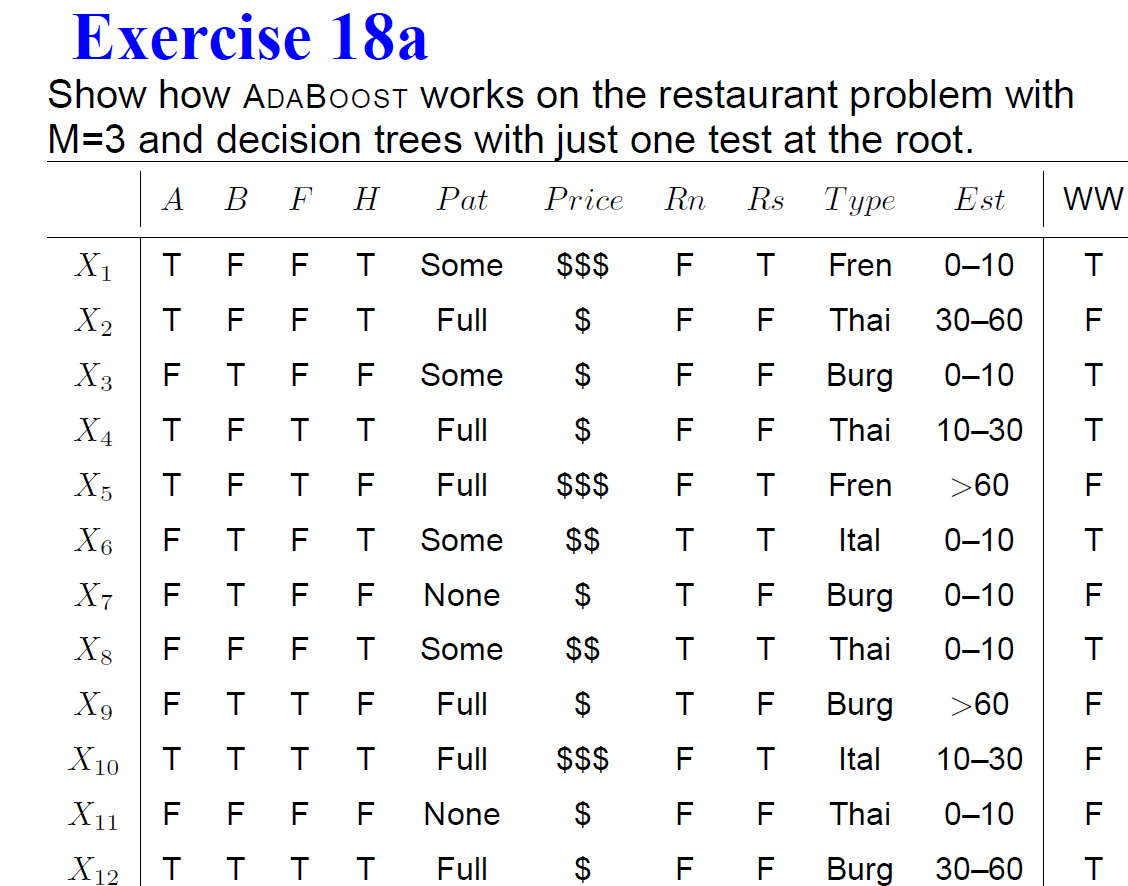
Size: La fel. Gain = 0.26

Intel: La fel. Gain = 0.54

(b) The highest Gain should be chosen as root. In our case BARKS can be chosen

(C) We separate depending on the values form barks and get only on one side both positives and negatives. We then apply the algorithm reclusively for those.

**AdaBoost problem.**



M - Nr of stumps that need to be computed.

The weights are at the beginning 1/12.

(M1)We first calculate all the gains and chose the biggest one as the root: PAT.

Will split in 3 parts: Some(all T), Full(T and F),None(all F).

Full is an issue: More F and we consider that it will take F (X4 and X12 are falsly classified), and we get the error 1/12 + 1/12 = 1/6.

The correctly classified ones are updated: new = old \* err/(1-err).

The weight of M1 is ln((1-err)/err) = ln(5);

We normalise the weights such that the sum is 1: 1/4 (X4,X12), 1/20 (Rest).

A method to use the weights is to update the table with new variables (that are duplicates of old ones) till we get a balance. In our case we add X4 and X12 (5-1) more times.

New max Gain is …

We get a new error

AND SO ON

Se dau urmatoarele features: F1 in {A,B}, F2 in

{C,D}, F3 in {E,G,H} si urmatoarele training examples:

A C H +

A D G +

A C G -

A D G +

B D H -

B D H -

B C H +

B C G -

(a) what score would the info gain formula assign to

each of 3 features?

(b) which would be the root of the tree?

(c) show remaining steps, if any, that decision tree

lerning algorithm would perform using the above

examples(d) what kind of search stratey?

# Communication.

## Semantic analysis

**Time + tense**

Refers to the case when we have the same verb in multiple tenses in our grammar. Ex John loves/loved Marry. To solve this case, we could add some lexical rules that encompass the time.



**Quantification**

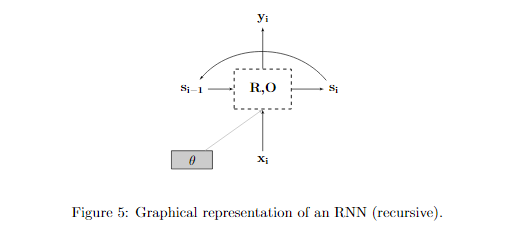
Refers how it can be interpreted based on existential and universal quantifiers. Ex. Every agent feels a breeze.

In this case, 2 interpretations are possible, 1 breeze per agent vs. 1 breeze for all agents. This ambiguity is solved using probabilities.

## RNN abstractions. Goldberg 10.1, pg 46

Instead of transporting the input form I to N (Feed forward Network), they also remember what happened in the past => states.

A new state is computed based on the curr state and the input.



## Ambiguation and Disambiguation. Cap 23

**Ambiguity syntactic/semantic**

Syntactic => Multiple ways to parse a sentence. Ex. I saw her duck. Leads to semantic.

Sematic => Different meanings.

**Metonymy**

A figure of speech in which one object is used in place of another. => Ambiguity

**Metaphor**

A sort of metonymy, but the words are similar, not synonims.

**Model for disambiguation**

4 Models

* The world model: One should know how the world actually works (Ex. I am dead)
* Mental model: Depending on WHO transmits, we must interpret it (What speaker thinks and what the speaker thinks the listener thinks)
* Language Model: Probability of choosing certain words to express yourself.
* Acoustic Model: Probability of certain sounds to be added when the words are already chosen (Ex. Can mai apare un IOI asa random la noi).

## Augmented grammars, AIMA 916

A grammar to which we add multiple rules to eliminate possible ambiguities.

Definite Clause Grammar – Showing how an augmented rule can be transformed into a logical sentence, that has a form of a definite clause.



For converting a grammar to a definite clause we do the following: Swap sides, make a /\ between all constituents and constrains and add variables si .



A language generation is the reverse application of the previous process.

Subject Verb agreement -> the verb will match the “nr” of the subject. Ex I am, you are, he is. We do that by splitting the substantives in 2 or 3 subcategories.

## Wumpus (De cautat pe rezolvari)

**It is a Wumpus.**

**The Wumpus is dead.**

**Wumpus is in 2.2**

**Can u define “It is a Wumpus” as “x belongs to Wumpus(x)”**

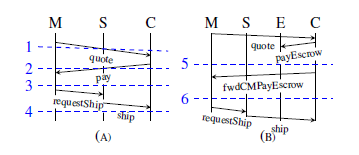
**Hint: “It was a a Wumpus”**

# Multiagent Communication for Interaction. Toska

Uses BSPL (Protocol specification) and CUPID (Commitment specification).

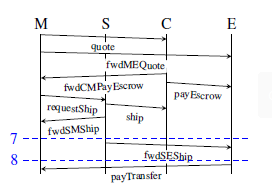
Protocols used for communication = Commitment alignment protocols. Explains how agents should communicate with each other (I/O needed to keep the communication working).

CUPID specifies what an agent knows he should do and after how much time he should do that. If a certain threshold is passed and a desynchronization appears, that agent should abandon that task.

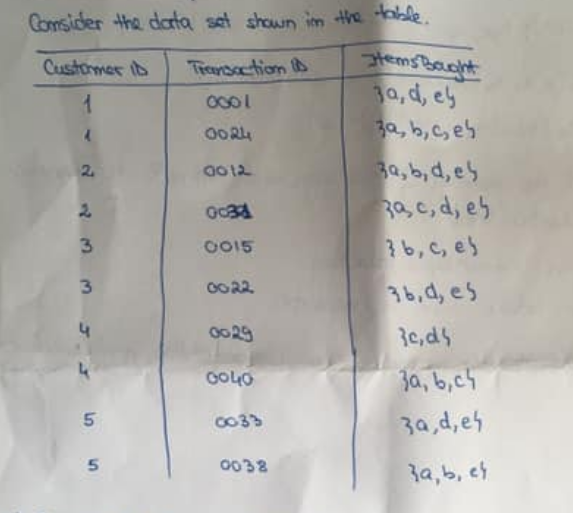


M-Merchant. S-Shipper. C-customer. E-Escrow

For this case, forwarding is needed for the merchant to receive confirmation that the shipment has been shipped.



# Association analysis



**Support for itemsets {e}, {b,d}, {b,d,e} by treating intersection as market basket.**

Nr of transitions that each appear together.

**Compute the confidence for some association rules. Are they a sym measure?**

Confidence from e to bd = support (bde) / support(e)

Non symmetrical.

**Treat each customer ID as a market bucket. Item = binary variable.**

Ex. C1: adeUabce = {abcde}

**Compute new confidence association.**

Use previous supports

## Apriori principle. CH6

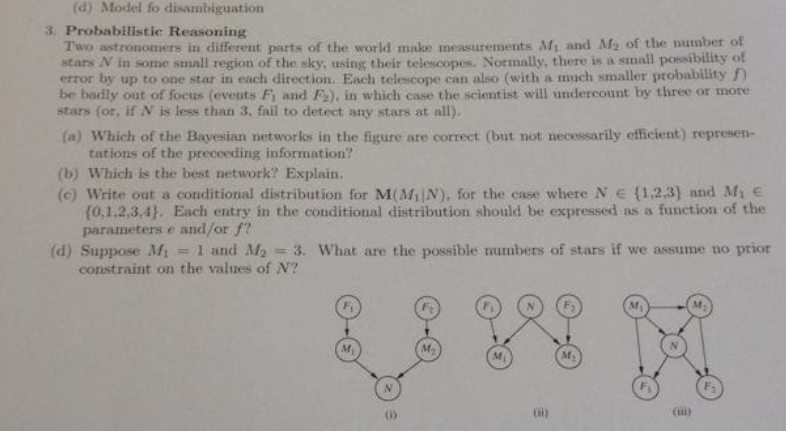
Used to generate frequent item sets. Ex. Most bought items.

If an itemset is frequent => all of its subsets MUST be frequent.

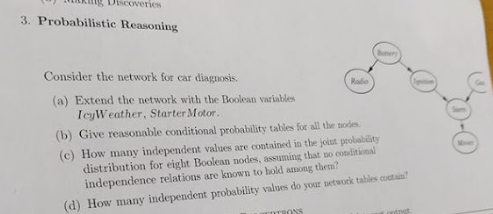
Used, as a start, from an empty item set. Search each individual item, and check if it passes a minimum support threshold. If one does not pass, remove it from future item sets. Compute new sets from the ones that pass the threshold. Repeat.

# Probabilistic Reasoning

## Astronomers AIMA SOL 132



## Car problem AIMA SOL 127



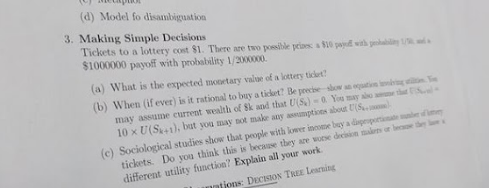
## Kalman Filters (Outdated)

**1-dimensinal example**

**Applicability**

# Making simple Decisions

## Lotto problem



# Value based plan selection in BDI agents.

## No pizza 4 u.

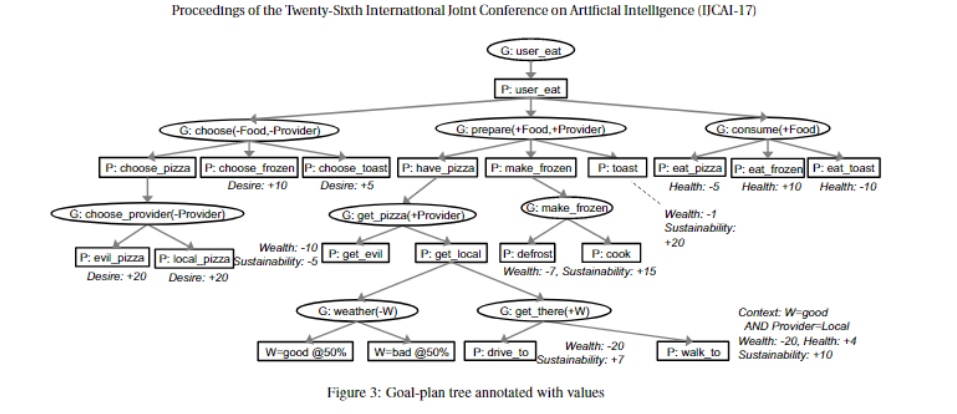
Explains how sets of values (Self-enhancement, Self-transcendence, Conservation, Openness to change) can be used with the help of concrete values.

We are given an example in which our Agent takes care of what a human eats. 3 values are chosen:

Enhancement -> To follow a users decision -> desired food

Conservation -> users’ health and wealth -> healthy + cheap food

Transcendence -> sustainable food



We see that the values are modified, either with + or with -. The goal of the agent is to balance these values.

To accomplish this, AgentSpeak is used.

We have an array with all the constraints that it can receive. The variables, that the agent must find to reach its goal, are specified: the provider and the choice.

We also know where each decision leads and the values that our variables get in certain nodes. When reaching a leaf, all variables should be instantiated. 

This formula presents what the agent should chose based on the importance of the values. The agent should minimize the sum.

# Agents that communicate

**write down the lexicon and the grammar rules for**

**"wumpus pidgin" (e0)**

**(a) which of the following are generated by the**

**grammar (why or why not?)?**

**se dau trei prop**

**(b) modify the rule for "the wumpus that the dogs see**

**stinks" to be generated**

**by the gramar**

**(c) give the parsing tree for this using your new rule**

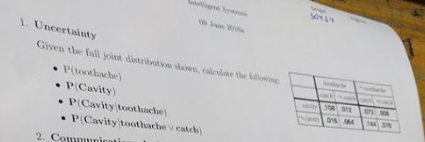
**(d) adjust grammar to allow 'the wumpus the dogs see**

**stinks", but to disallow "the wumpus the dogs I smell**

# Uncertainty (Posibil sa nu intre dar ii bulangiu si facem totusi)

## Wumpus world

## Dentist



## Extra

**Explicati esenta urmatoarelor sectiuni din acest capitol**

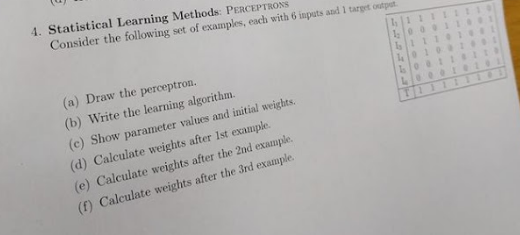
**a) The axioms of probability**

**b) Inference using ful join destribution**

**C) Independence**

# Statistical Learining Methods

## Perceptrons



# Planning (Outdated)

**Se da functie Ride(x,e,f1,f2) descrisa in STRIPS**

**care duce persoana x, cu liftul e, de la etajul f1 la**

**f2.**

**(a) write down a definition for Call(x,e,f) - care**

**cheam liftul la etajul f**

**(b) wite down an effect axiom for Ride**

**(c) write down a frame axiom needed for this world**

**(d) Jeb from floor 2 wants to go to 3, with the only**

**working elevator E, which is at floor 7. Using the**

**graphical notation for plans, give the initial**

**empty plan.**

**(e) Add the ride step to this plan**

**(f) Is there more than 1 way to do this?**

# (Outdated) Reinforcement Learning

**Explicati esenta urmatoarelor sectiuni din acest capitol**

**a) Passiv reinforcement learning**

**b) Activ reinforcement learning**

**c) Generalization in reinforcement learning**