Logikk og Resonnerende systemer

<u>Oversikt:</u>

- Øvinger

- Pensum

- Gjennomgang av pensum

Laget av: Marte Løge høsten 2012
Forelesere: Keith Downing og Pinar SomethingSomething

Obs! jeg tar ikke ansvar om det skulle stå noe feil i dokumentet (må også nevne at mine skrivekunnskaper IKKE er bra! Ikke mobb mine pinlige skrivefeil). Dette dokumentet er noe jeg lager for min egen del, så bruk det med måte! Alle har tilgang til å kommentere, så legg gjerne inn en kommentar om noe skulle være feil eller at noe mangler :)

Øvinger:

Øving1) (Robots)
- Environments
- Agents

Øving2) (Fraction puzzle and The linear checkers puzzle)
- A*
- Best-first search

Øving3) (Eggcartoon puzzle)
- Simulated annealing
- Local search (Dumb local search)
- Partial vs. complete solutions
- Objective function
- Representation, objective function, neighbor generation
- Heuristic vs. objective function

Øving4) (k-queen)
- Constraint satisfaction problems
- Backtracking
- Local Search with constraint reasoning (Intelligent local search)
- AC-3 algorithm

Øving5) (Logic)
- Models
- KB
- CNF
- Logikk

Øving6) (Planning)
- Graph planning
- Mutex
- Knowledgebase (KB)

Øving7) (Frame-based)

-

Pensum:

<u>Del1: “Artificial Intelligence”</u>

- **Kapittel1** - “Introduction”
- **Kapittel2** - “Intelligent Agents”

<u>Del2: “Problem-solving”</u>

- **Kapittel3** - “Solving problems by searching”
- **Kapittel4** - “Beyond Classical Search”
- **Kapittel5** - “Adversarial Search”
- **Kapittel6** - “Constraint Satisfaction Problems”

<u>Del3: “Knowledge, reasoning, and planning”</u>

- **Kapittel7** - “Logical Agents”
- **Kapittel8** - “First order logic”
- **Kapittel9** - “Inference in Firs-order Logic”
- **Kapittel10** - “Classical Planning”
- **Kapittel11** - “Planning and acting in the real world”
- **Kapittel12** - “Knowledge Representation”

<u>Del4: “Communicating, perceiving and acting”</u>

- **Kapittel22** - “Natural language Processing”

Kapittel2: “Intelligent Agents”

- Agent: Is as anything that can be viewed as percerving it environment through sensors and actiong upon that environment through actuators.

- PAGE

- Percepts
- Action
- Goal
- Environments

- Percept
- Percept sequence
- Agent functions
- Rationality
- Rational Agent

- PEAS:

- Performance measure
- Environment
- Actuators
- Sensors

Agent type	Performance measure	Environment	Actuators	Sensors
Autamated taxi driver	Safe, fast, legal, comfort, max profits	Roads, traffic, customers	Wheel, break, signal, horn, accelerator	Camera, speedometer, GPS, keyboard, engine sensors
Part picking robot	Percentage of parts in correct bins	Belt with parts, bin	Jointed arm and hand	Camera, joint, angle sensors
Medical diagnosis system	Frisk pasient, minimize costs	Patient, hospital, staff	Screen display	Keyboard

- Rationality != Omniscience (allvitenhet)

- Single agent vs. multiagent

- **Single agent:** Puzzle
- **Multi agent:** Chess
- **Competitive multiagent environment:** Feks i sjakk spiller man mot hverandre.
- **Cooperative multiagent environment:** Når man kjører bil samarbeider alle bilister.

Environments:

- Fully observable vs. partially observable environments

- **Fully observable:** The agents sensors give it access to the complete state of the environment at each point in time.
  A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choise og action
- **Partially observable:** Because of noisy and inaccurate sensors, ot because parts of state are simply missing from the sensor data.

- Deterministic vs. Stochastic environments

- **Deterministic:** Any action has a single quaranteed effect. There is no uncertainty about the state that will result from performing an action.
- **Strategic:** If the environment is deterministic expect from the actions of other agents, we say that the environmentis strategic.
- **Stochastic:** There is some uncertainty about the outccome of an action.

- Uncertain environments: Not fully observable nor deterministic

- Nondeterministic environments:

- Episodic vs. Sequential environments

- **Episodic:** In each episode the agent receives a percept and then performs a single action. The episodes is not effected by the privious or the next action. Feks en robot som skal sortere biter på bånd. Den vil behandle hver bit hver for seg. Selv om en bit er ødelagt så betyr ikke det at neste er det.
- **Sequential:** The current decision could affect all future decisions.

- Static vs. dynamic environments

- **Static:** Forblir uendret til agenten gjør en endring.
- **Dynamic:** Omgivelsene er i konstant ending. Når agenten får input kan det hende omgivelsene har endret seg før agenten får utført en handling.
- **Semidynamic:** If the environment itself does not change with the passage of time but the agents performance score does.

- Descrete vs. Continuous environment

- **Descrete:** Such as chess game has a finite number of distinct states . Chess has a descrete set of percepts and actions.
- **Continuous:** A taxidriver has continuous state and time.

- Known vs. unknown environment

- **OBS: Known/unknown environments != fully/partyally observable environments**
- **Known: If the agent knows the environment, then the agent does know the outcome of the actions**
- **Unknown: Opposite than know.**

Properties of environments:

Fully observable vs. partially observable	Deterministic vs. stochastic	Episodic vs. Sequential	Static vs. dynamic	Known vs. unknown

Agents:

- Simple reflex agents:

- Simplest kind of agent
- Selects actions on the basis of the current percept, ignoring the rest of the percept history
- Eks: Støvsugeren forholder seg til en ting og det er det som er skittent eller rent på plassen den står. Avgjørelsen da er da uavhengig om det var skittent/rent på forrige plass.
- The envoironment must be fully observable
- Evige løkker er ofte fallgruver i partyally observable environments
  → randomisering løser dette

- Model-based reflex agent

- How does the world(environment) evolves independently of the agent?
- How the agents own actions affect the world?

- Goal-based agents

- Knowing about the current state of the world is not always enough to decide what to do
  → We need a goal!
- Eks: at a road junction, the taxi can turn left, right or straight on. The correct decision depends on where the taxi is trying to get (what the goal is)
  → The destination is the goal that affects the action.
- Search and planning → reach goal

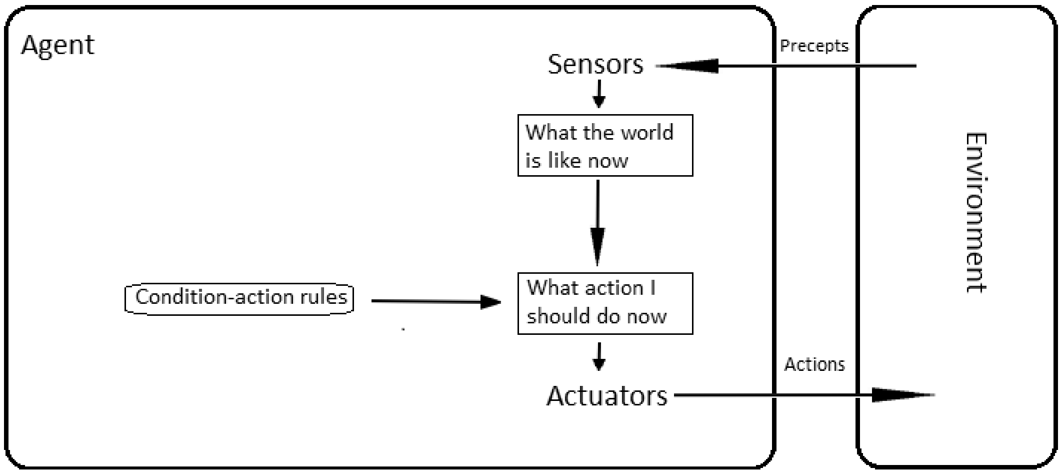
- **Utility-based agents:**

- Goal alone are not enough to generate high quality behaviour in most environments
- Eks: En taxi kan ta mange forskjellige veier til et mål, men noen tar lengre tid enn andre.
  → Ergo, noen beslutninger (actions) har høyere nytteverdi (utility)
- A utility function maps a state onto a real number, which describes the associated degree of satisfaction
- **Expected utility** → Forventer å maximere graden av tilfredsstillelse.

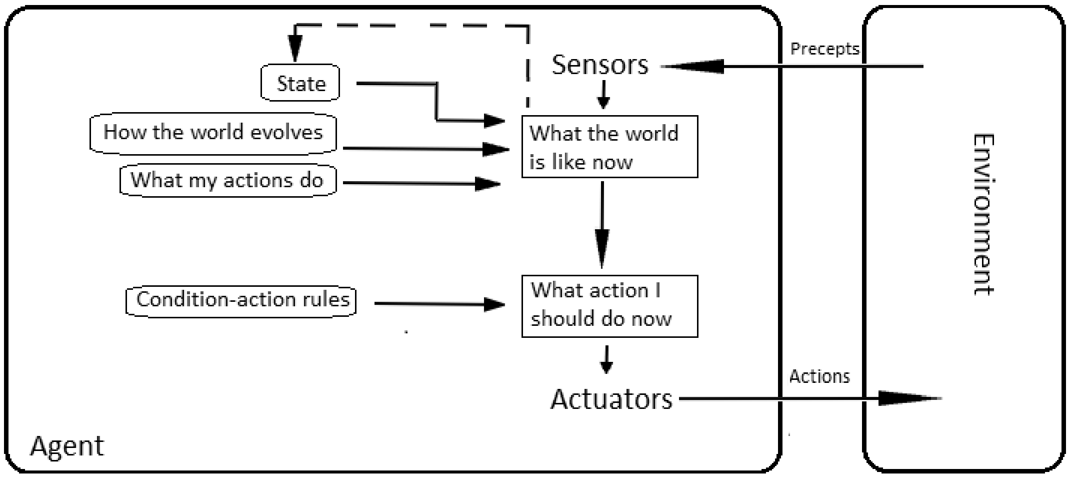
- Learning agents:

- Lærer av sine handlinger

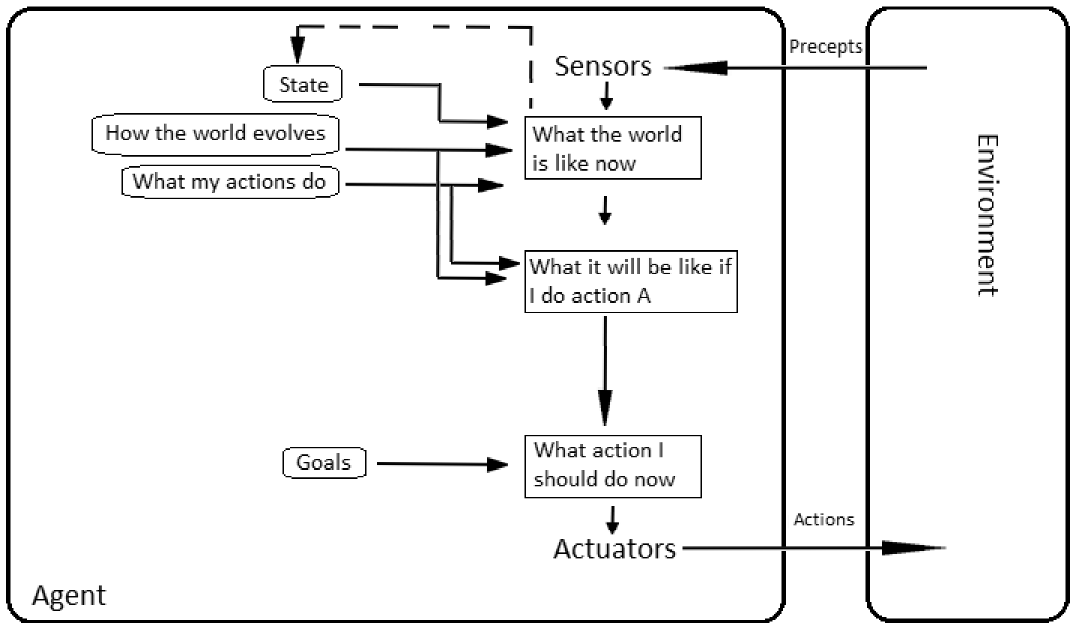
Simple reflex agent:



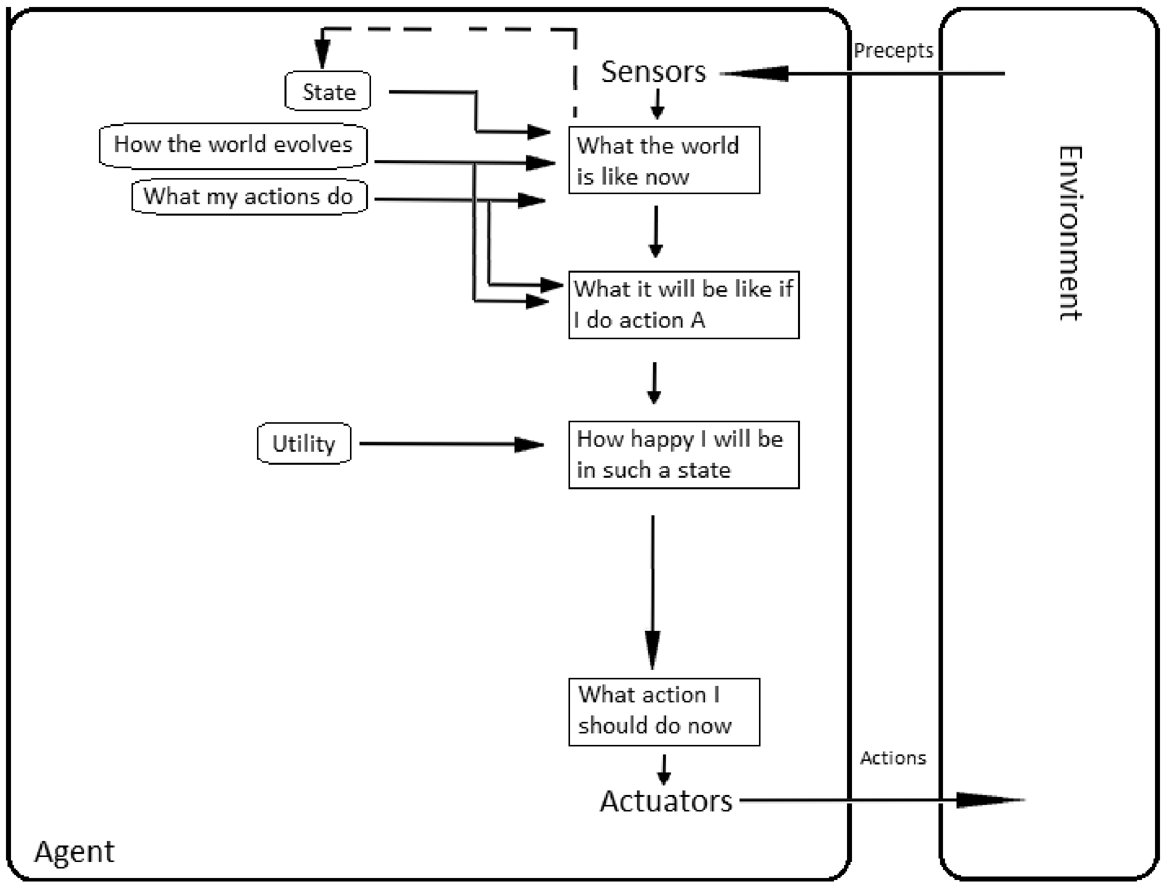
Model-based agent:



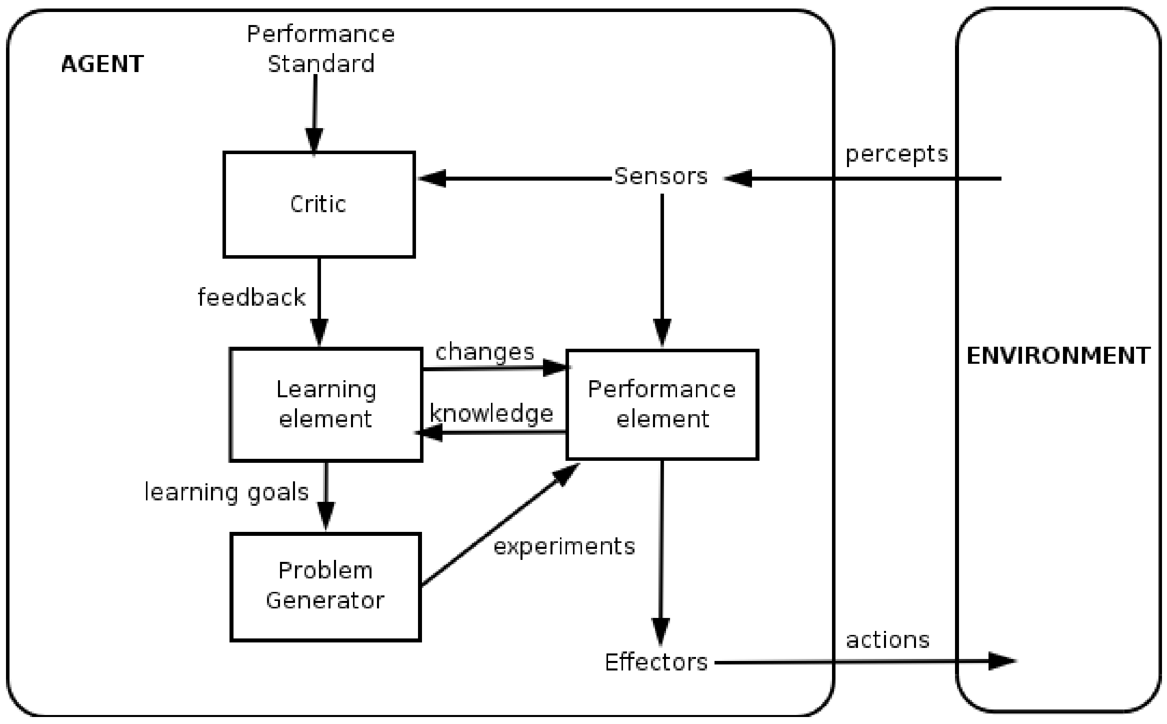
Goal-based agent:



Utility-based agent:


State
How the world evolves
What my actions do
Utility
Sensors
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Actuators
Agent
Precepts
Actions
Environment

Simple learning agent:



Kapittel3: “Solving problems by searching”

- **Uninformed vs. informed:** Do points in the search space give information that helps the searcher to determine the next step?
- **Partial vs. complete solutions:**
(Complete, local) Løsning, men er ikke nødvendigvis optimal
(Partial, backtracking) Følger en sti
- **Simple-Problem-Solving-Agent:** Planlegger, lukker øynene og utfører det man planla.
- **Offline vs online:** Offline (eyes closed). Online (Behandler på veien, men har ingen kunnskap)
- **Poblem types:**
  - Deterministic, fully observable → Single-state problem.
    Agent knows which state it will be in; Solution is a sequence
  - Non-Observable → Conformant problem
    Agent may have no idea where it is; solution (if any) is a sequence
  - Nondeterministic and/or partially observable → Contingency problem
    precepts provide new information about current state
    solution is a contingent plan or a policy
    ofter interleave serach, execution
  - Unknown state space → exploration problem (“online”)
- **A problem is defined by:** (Single-state problem formulation)
  - Initial state
  - Successor function (set of action)
  - Goal test
  - Path cost
  - Solution → sequence of actions

- The real world is very complex! Abstrahere for å gjøre det lettere, ta med det som er relevant for å finnne en løsning.
- **Tree search algorithms:** Offline, simulated exploration of state space by generating successors of already-explored states (a.k.a expanding states). Løsning? Hvis ikke, generer nytt nivå i tret.
- **Graph search:** Samme som Tree, bare at man ikke har redundante tilstander (ha<r en tilstand vert før, lager man bare en kompling til denne).
- **State vs. Node:** En node er et punkt i et tree/graf (struktur). En state er tilstanden inne i noden.
- **Search strategies:** A strategy is defined by picking the order of node expansion.
- **Strategies are evaluated** along the following dimensions:
  - Completeness - does it always fins a solution if one exists?
  - Time complexity - number of nodes generated/expanded
  - Space Complexity - maximum number of nodes in memory
  - Optimality - does it always find a least-cost solution? (Første løsning er den beste!!)
- **Measures:**
  - b - maximum branching factor of the search tree
  - d - depth of the least-cost solution
  - m - maximum depth of the state space (may be infinite)

Uniformed strategies

use only the information available in the problem definition

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Interative deepening search

- **Kunsten å lage gode søkalgoritmer** er å gi "hint" på veien. Uten hint har man vanlige metodiske søk som DFS og BFS.

- HINT: ".. han anbefalte å gjøre A* fra bunnen siden det nesten er garantert å få spørsmål om A* på eksamen".

- **A*:** F = G + H
  - F = Function
  - G = Path cost so far
  - H = Intelligent guess (Heuristic)
- **Breadth-first search:**
  - Complete? Yes (if b is finite)
  - Time? O(b^{d+1})
  - Space? O(b^{d+1}) (Keeps every node in memory!)
  - Optimal? Yes, but only if cost = 1 per step. Not optimal ingeneral!
- **Uniform-cost search:** Samme som BFS, bare den har forskjellig kostnader. Har en kø som er ordnet på kostnader (lavest først).

- Complete? Yes
- Time?
- Space?
- Optimal? Yes

- **Depth-first search:**
- Complete? No: fails in infinite-depth spaces
  Complete in finite spaces!
- Time? O(b^m): blir grusom om m er mye større enn d!
- Space? O(bm)
- Optimal?

- **Depth-limited search:**
- Går i dybden, men stopper ved et nivå l!
- Er ikke optimal siden en løsning kan ligge under grensen
- Finner slikt i spill → Til høyere level, til dypere søk gjør man

- **Iterative deepening search:**
- Er depth-limited search, bare at den repeterer den.
- Begynner med feks med en grense lik 1 → ingen løsning → utvid grense
  Starter på ny med grense lik 2 → ingen løsning → utvid grense
  etc...
- Complete? Yes
- Optimal? Yes, if step cost = 1

- **Repeated states:** Failure to detect repeated states can turn a linear problem into an exponential one! Tree Search → Graph Search

Informed search strategies:

- **Best-first search:**
- Idea: Use an evalution function for each node
- Expand most desirable unexpanded node
- Implementation: queue sorted in decreasing order of desirability
- Special cases:
  - Greedy search (Ignorerer tidligere steg. Ser på tilstanden den er i nå)
  - A* (Tar hensyn til historie)

- **Greedy search:**
- Evalution function h(x) (heuristic) = estimate of cost from x to y
- Greedy search expands the node that appears to be closest to the goal
- Complete? No, can get stuck in loops (optimal hvis man sjekker om man har vert der før)
- Optimal? No
- Time and space? (b^m)

- **A*** :
- Queue of expanded nodes. Always picks the node with the lowest f value (f = g + h)
- Optmail? YES! (unless there are infinitely many nodes with f<= f(G)
-

Kan være greit å kunne "bevise" hvorfor A* er optimal. Se forelesning om informed search strategies.

- **Heuristics:**
  - Admissible heuristics: underestimates the solution
    - 8 Puzzle: Number of misplaced tiles.
    - 8 Puzzle: Manhattan distance

  - **Does i matter to have a good heuristic?** YES, a good heuristic will expand less nodes than a bad one!
  - **Dominance:** If h2(n) >= h1(n) for all n (both admissible), then h2 dominates h1 and is better for search.
  - **How to find a admissible heuristic? → Relaxe the problem**
    Dette vil si at man forenkler søket og gjør det mer abstrakt. Feks at man fjerner regler i 8 puzzle som at det må være ledig for å flytte en brikke → Vi gjør det enklere å ser på selve avstanden i stedet for hvor mange moves vi trenger for å få en brikke på plass. I følge reglene er ikke dette lov, men det vil gi oss en pekepin på om vi beveger oss mot en løsning (et underesitmat).
  - **Key point:** the optimal solution cost of a relaxed problem is no grater than the optimal solution cost of the problem.

Local Search Methods:

- **Local search:**
  - The path is unimportant; only the final state matters
  - All search-space states = complete solutions (aka attempts)
  - Search ← → Modifying complete solutions
  - Partial credit still very important, nut now its given to whole(not partial) solutions.
  - Nå bruker vi ikke heuristics, men noe lignende som heter "Objective functions". Objective function svarer på: "hvor god er du, løsning"?
- **Properties of local search:**
  - Low space complexity - only need to save one (or a set) of current solutions, NOT paths back to the start state
  - Time complexity varies, through recent work indicates major improvements over incremental search for problems with densely-packed optimal solutions
  - Satisficing - can often find reasonably good solutions quickly
  - Requires representations that are easy to tweak to generate search-space neighbors
  - Uses an objective function to evaluate solutions. Similar ro a heuristic but for complete solutions → less guesswork
  - Often portrayed as movement in a landscape.
- **Incremental vs. local search**
  - Incremental tar vare på søketreet fordi stien er relevant for resultatet.

Local search jobber med komplette (ikke nødvendigvis optimal), så trenger ikke å ta vare på treet (grafen).

- Incremental bruker heuristics for å "spå fremtiden" (for å velge den beste vei).
  Local search bruker objective function som evaluerer hvor god løsningen er.

- **Example of local search problems**
  - Egg carton puzzle
  - K-Queen puzzle
- **Representation:**
  - Semantic representation: Ser på Egg carton puzzle som en egg carton with eggs.
  - Syntactic representation: Ser på Egg carton puzzle som en twodim-array med bits.
- **State space**
  - States in search space are the syntactic representation
  - Neighbor-generating operators are designed to manipulate syntax in simple wys; e.g. flip bits
  - **Direct representation:** Sometimes syntactic is almost equal to semantic representation and neighbor-generation takes semantic into account
    → Intelligent but more work
  - **Indirect representation:** Syntactic and semantic representations very different → bumb generator; all intelligence in objective function.
- **Search Landscape**
  - Objective function defines the landscape
  - Representation-modifying operators define legal moves in landscape.
- **Hill-Climbing**
  - Greedy: always moves to states with immediate benefits
  - Quick on smooth landscapes
  - Easily gets stuck on rough landscapes (Ser ikke en bedre vei, slutter/gets stuck)
- **Simulated Annealing (... type of hill-climbing)**
  - Hill-climbing with "jiggle"
  - Må ikke være like gråding som hill-climbing
- **Local Beam Search:** K parallel searches
  - K parallel searches are not independent since:
  - Best K neightbors chosen at each step, but som states may contribute 0 og 2+ neighbors to the next step
  - Good search = efficient distribution of resources (i.e,, the K states)
  - Jiggle can be introduced via Stochastic Beam Search: pick some of the K neighbors randomly, with probabilities weighted by their evaluations.
- **Stochastic local beam search:**
- **Generic algorithms:** Stochastic local beam search with crossovers.
  - Stochastic local beam search with (some of) the K neighbors generated by crossover
  - Supports long jumps in search space.. and combinations of the best of both parents.
- **Exploitation vs Exploration:**
  - Exploration ("Explore!"): "Velg random"
  - Expliotation ("Exploit!"): "Utforsk en retning som man har evaluert først"

- Just a little bit of random movement can soften help a lot (help you from getting stuck)!

Kapittel5: “Adversarial Search”

- “AI is what humans are currently better at..” - Jim Hendler

- **Classes of games:**
  - Perfect information: State of playing arena and other players known at all time
  - Imperfect information: Some information about arena or players not visible
  - Deterministic: Outcome of any action is certain
  - Stochastic: some actions have propabilistic outcomes, e.g. dealt cards, rolled dice, etc.

- **Adverserial Search Trees:**
  - My action and opponent actions are building a tree
  - One action → New game state
  - Level of diff. → Level in the tree
  - Evaluate bottom states and propagate values upwards.

- **Aplha-Beta Pruning:**
  - MAX/MIN nodes → I want to max my value and my opponent want to minimize my value
  - Er en metode man kan bruke for å få alpha-beta pruning mer effektivt.
  - Pruning means that you eliminate nodes that doesn't will affect the result from the search.
  - <alpha, beta>, lower and upper bounds
  -

- **Minimax**
  - Generer et tre, velg sti → Gå og beegyn på ny!
  - Bruker MAX/MIN nodes
  - Regner oss bakover fra bunnen opp til toppen.
  - For hver action krever det å generere et nytt tre!
  - DFS

- Viktig for eksamen er å forstå at det er mange måter å bruke søketrær på:

- **Generere et tre for hver action**
- **Generere et tre for å fine en sti av actions**
- **osv**

-”Typical exam questions: ” alpha-beta pruning: Arrange the integer evaluations 1, 2,,...., 8 in leaf nodes so that

- **The maximum number of nodes can be pruned**
- **All nodes need to be generated; no pruning possible**

Han hintet til at det å gjør denne oppgaven kunne hjelpe på eksamen.

- **AI Critics**
  - A good critic can reduce levels in a tree!

Kapittel6: “Constraint Satisfaction Problems (CSP)”

- **CSP: Map coloring**
  - Variables = {All the countries on the map}
  - Domains = {The colors}
  - Constraints = {Adjacent regions must have different colors}
- **Constraint graph:** nodes are variables, arcs shows constraints. General purpose CSP algorithms use the graph structure to speed up search. It will speed up the solution if you can divide the problem into subproblems.
- **Types of constraints:**
  - **Unary:** involves single a single variable
    South Australia != green
  - **Binary:** involves pairs of variables
  - **Higher-order constraints:** Involve 3 or more variables.
  - **Preferences (soft constraints):** “red is better than green”.
    Often representable by a cost of each variable assignment
    → Constrained optimization problems
    Should be satisfied but not necessarily.
  - **Absolute (hard):** Must be satisfied in any valid solution
- **Descrete variables:**
  - Finite domains:
  - Infinite domains
- **Continuous variables:**
- **Real world CSP:**
  - Assigment problems
    “who teaches what class?”
  - Time tabeling problems:
    “Which class is offered when and where?”
    Exam scheduling
  - Hardware configuration
  - Transportation scheduling
- **Domain reductions** (The domain is reduced by the constraints)
- **Constraint propagation:** Inference in CSP’s
  - **Node Consistency**
  - **Arc Consistency**
  - **Local Consistency**
  - **Path Consistency**
  - **Constant propagation**
  - **AC-3 algorithm**
  - **K-Consistency**
    - 1-consistency
      → Node consistency
    - 2-consistency

        → Arc consistency
    - 3-consistency:
        → Path consistency
- Strongly k-consistent
- Global constraints:
    - Alldiff
    - Resource constraint (AtMost)
    - Bounds propagation

- **Approaches to solving CSP’s**

- Inference
- Incremental Search + Inference
- Local Search + Assumption modification

- **CSP via Backtracking vs Local Search**

- **Backtracking search**

- The term backtracking search
- Improving effiency
- MRV - “Minimum remaining values”
- Degree heuristic

- **Structure of a problem**

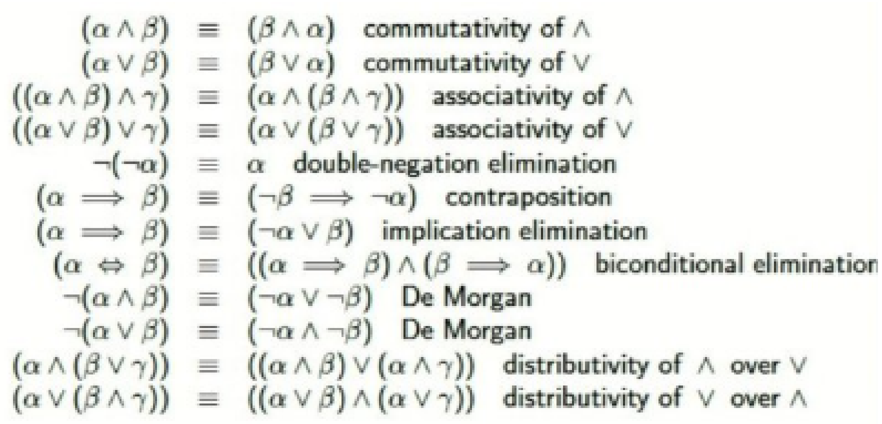
Kapittel7: “Logical Agents”

Logic in general:

- **Knowledge base (KB)**: Set of sentences in a formal language
- **Logic in general**:
  - Logics: are formal languages for representing information such that conclusions can be drawn.
  - Syntax: defines the sentences in the language
  - Semantics: defines the “meaning” of sentences
    i.e., define truth of a sentence in a world (model)
- **Entailment:** means that one thing follows from another:
  - KB|=a → KB “entails” a → KB is a subset of a
  - Entailment is a relationship between sentences (syntax) that is based on semantics
  - b |= a if and only if M(b) ⊆ M(a):
    every world in which b is true is also a world in which a is true
  - **Examples: (**⊆ means “subset”)
    A and B |= A? M(A and B) = {5,7} ⊆ {4,5,6,7} = M(a) → Ja
    A and B |= A or C? M(A and B) = {5, 7} ⊆ {1, 3, 4, 5, 6, 7} = M(A or C) → Ja
    A or C |= C? M(A or C) = {1,3,4,5,6,7} not ⊆ {1, 3, 5, 7} =M(C) → Nei
    X+Y = 4 “entails” 4 = X+Y
- **Worlds and logical models**
  - A possible world (a.k.a model) is a complete set of truth assigments for all logical variables.
  - M(a) of a logical sentence a is the collection of possible worlds in which a is true
  - We say m is a model of a sentence a if a is true in m
  - Example:
    M(A) = {4, 5, 6, 7}, M(C) = {1, 3, 5, 7}
    M(A and B) = {5,7}
    M(A or B) = {1, 3, 4, 5, 6, 7}

Propositional (boolean) Logic

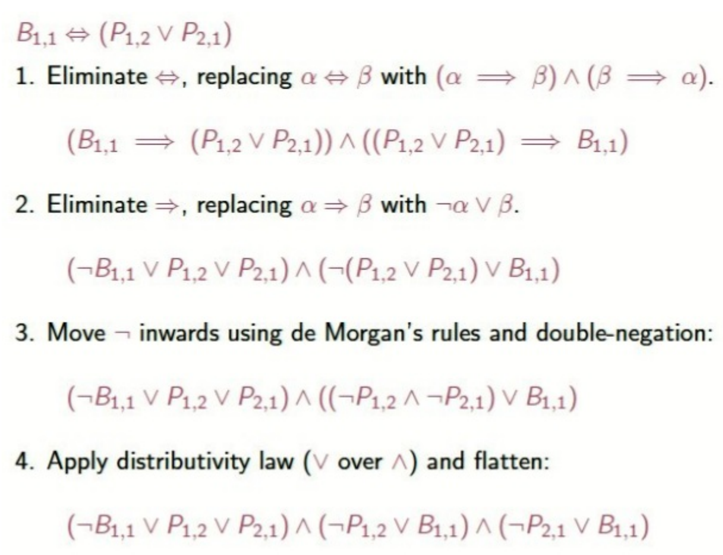
- **Inference**
  - KB |- (i) a = sentence a can be derived from KB by procedure i
  - Entailment = needle in haystack , inference = finding the needle
  - Soundness: procedure i is sound if whenever KB |- (i) a, it is also true that KB |= a
  - Completeness: procedure i is complete if whenever KB |= a, it is also true that KB |- (i) a
- **Logic rules:**



$(\alpha \land \beta)$	$\equiv$	$(\beta \land \alpha)$	commutativity of $\land$
$(\alpha \lor \beta)$	$\equiv$	$(\beta \lor \alpha)$	commutativity of $\lor$
$((\alpha \land \beta) \land \gamma)$	$\equiv$	$(\alpha \land (\beta \land \gamma))$	associativity of $\land$
$((\alpha \lor \beta) \lor \gamma)$	$\equiv$	$(\alpha \lor (\beta \lor \gamma))$	associativity of $\lor$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \implies \beta)$	$\equiv$	$(\neg\beta \implies \neg\alpha)$	contraposition
$(\alpha \implies \beta)$	$\equiv$	$(\neg\alpha \lor \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \implies \beta) \land (\beta \implies \alpha))$	biconditional elimination
$\neg(\alpha \land \beta)$	$\equiv$	$(\neg\alpha \lor \neg\beta)$	De Morgan
$\neg(\alpha \lor \beta)$	$\equiv$	$(\neg\alpha \land \neg\beta)$	De Morgan
$(\alpha \land (\beta \lor \gamma))$	$\equiv$	$((\alpha \land \beta) \lor (\alpha \land \gamma))$	distributivity of $\land$ over $\lor$
$(\alpha \lor (\beta \land \gamma))$	$\equiv$	$((\alpha \lor \beta) \land (\alpha \lor \gamma))$	distributivity of $\lor$ over $\land$

- **Logisk ekvivalens:** To setninger er ligisk ekvivalente om de er true i samme modeller. (a “entails” b and b “entails” a)
- **Tautologi(Valid):** alltid sann uansett verdier.
- **Satisfiable:** A sentence is satisfiable if it is true in some model
- **Kontradiksjon(unsatisfiable):** alltid uasann uansett verdi
- **Proof methods**
  - Application of inference rules
  - Model checking
- **Inference**
  - Modus ponens:
    "P implies Q; P is asserted to be true, so therefore Q must be true."
    $$\frac{P \to Q, P}{\therefore Q}$$
  - If it's raining, I'll meet you at the movie theater.
    It's raining.
    Therefore, I'll meet you at the movie theater.

- **Forward and backward chaining**
  - KB = **conjuntion** of **Horn clauses**
  - Conjuntion: Conjunctions join different parts of a sentence together. The very word "conjunction" comes from Latin words for 'join together'.
  - **Horn clause:** Disjunction of literals (i.e clauses) in which at most one of the literals is positive; rest is negative
    - (Conjunction of symbols) → Symbol
      (not (Conjunction of symbols) or symbol)
  - **Backward chaining:** Begynner med slutten og jobber seg bakover
  - **Forward chaining:** Starter med noe og jobber seg mot målet
- **Conjunction normal form (CNF):** produkt av sum ((a eller b eller c) og (d eller e eller f) og....)
- **Conversion to CNF:**



$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \implies \beta) \wedge (\beta \implies \alpha)$.

   $(B_{1,1} \implies (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \implies B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

- **Resolution:** proof by contradiction
- Propositional logic lacks expressive power

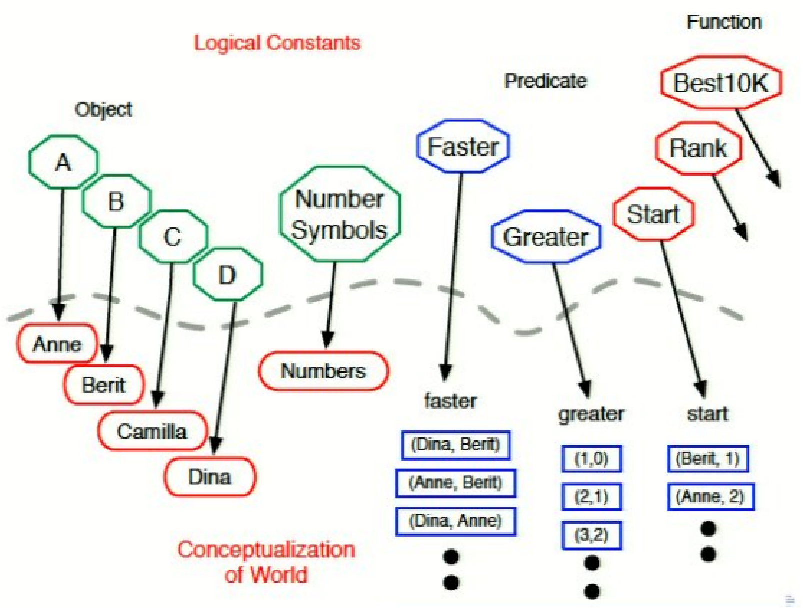
First order logic:

- Foil " Ski model": typisk eksamenspørsmål: Velge ut setninger som er en model for et problem.

- **Conseptualization of a possible world:**
  - **Objects:** physical objects or concepts in the possible world
  - **Functions:** these map one or more objects to single values, which are often other objects.
  - **Relations:** These represent relationships in the possible world. They must have a truth value.

- Predicates vs. functions:
- Predicates associate a set of objects with a truth value
  Color(X, red) → {True, False}
  Red(X) → {True, False}
- Functions map one or more objects to a unique object
  Color(X) → Red (where red is an object in this world)
  (Color(X) = green) → {True, False}

- **Intepretation:** Mapping from constant, function and predicate symbols of the representation to the conceptualization



Logical Constants
Function
Predicate
Object
Best10K
Rank
Faster
Start
Greater
A
B
C
D
Number Symbols
Anne
Berit
Camilla
Dina
Numbers
faster
(Dina, Berit)
(Anne, Berit)
(Dina, Anne)
greater
(1,0)
(2,1)
(3,2)
start
(Berit, 1)
(Anne, 2)
Conceptualization of World

Kapittel 9: “Inference in FOL”

- Universal instantiation (UI): “For alle x ….”
UI can be applied several times to add new sentences; the new KB is logically equivilent to the old
- Existential instantiation (EI): “Det eksisterer en x slik at ….”
EI can be applied once to replace the exisstential sentence; the new KB is NOT equivalent to the old, but is satisfiable iff the old KB was satisfiable.

$$\frac{\forall v \;\; \alpha}{\textsc{Subst}(\{v/g\}, \alpha)} \qquad \frac{\exists v \;\; \alpha}{\textsc{Subst}(\{v/k\}, \alpha)}$$

- **Skolem constant:**
∃ x Crown(x) ∧ OnHead(x, John) → Crown(C1) ∧ OnHead(C1, John)
C1 is a skolem constant.
- Unification:

$$\textsc{Unify}(\alpha, \beta) = \theta \text{ if } \alpha\theta = \beta\theta$$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	$fail$

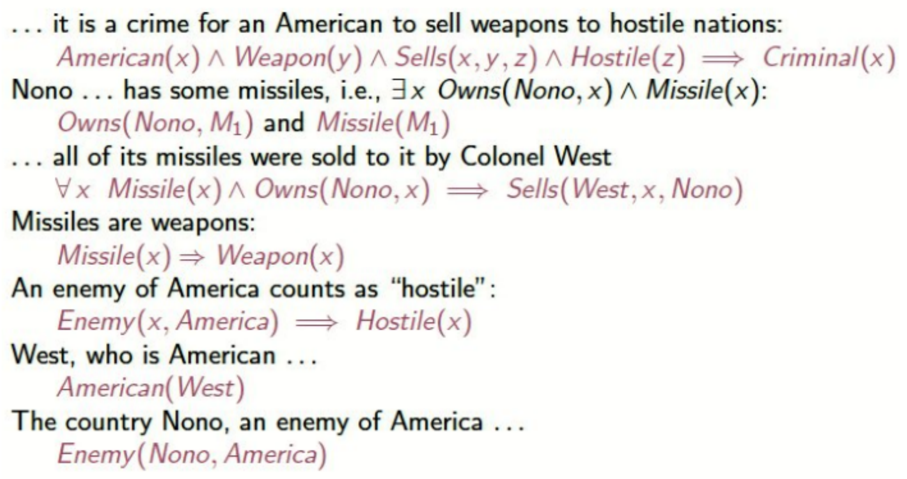
- Genaralized Modus Ponens:

$$\frac{p_1{}',\;\; p_2{}',\; \ldots,\; p_n{}',\;\; (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta} \qquad \text{where } p_i{}'\theta = p_i\theta \text{ for all } i$$

$p_1{}'$ is $King(John)$	p_1 is $King(x)$
$p_2{}'$ is $Greedy(y)$	p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$	q is $Evil(x)$
$q\theta$ is $Evil(John)$	

Kan si dette med et “lettere” eksempel:
Hungry(x) → Thirsty(x)
Hvis vi vet: Hungry(Marte), da vet vi også Thirsty(Marte).
Hvis en person er sulten så er også personen tørst.

- Reasoning eksempel med Forward- and Backward chaining:



... it is a crime for an American to sell weapons to hostile nations:

$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \implies Criminal(x)$

Nono ... has some missiles, i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West

$\forall x\ \ Missile(x) \wedge Owns(Nono,x) \implies Sells(West,x,Nono)$

Missiles are weapons:

$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":

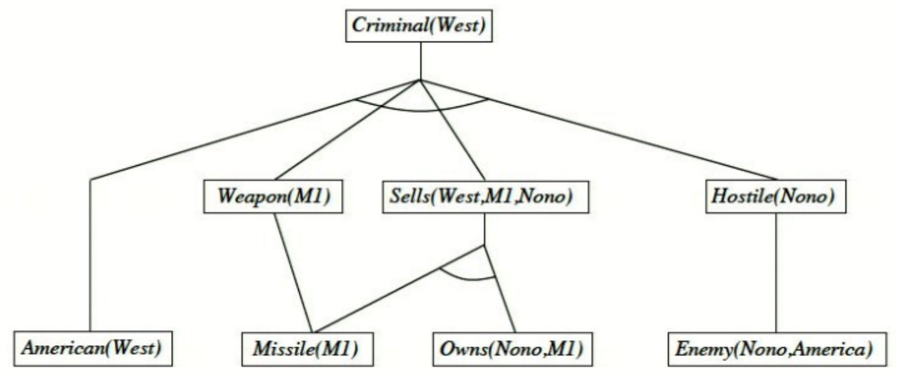
$Enemy(x, America) \implies Hostile(x)$

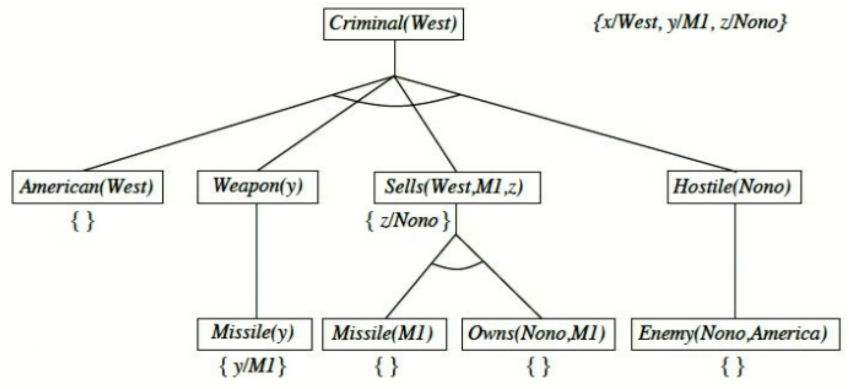
West, who is American ...

$American(West)$

The country Nono, an enemy of America ...

$Enemy(Nono, America)$





Backward chaining: Er ferdig når vi ikke har flere variabler som må bindes.

Conversion to CNF and Resolution Proof:

Everyone who loves all animals is loved by someone:

$$\forall x\ [\forall y\ \ Animal(y) \implies Loves(x,y)] \implies [\exists y\ \ Loves(y,x)]$$

1. Eliminate biconditionals and implications

$$\forall x\ [\neg\forall y\ \ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ \ Loves(y,x)]$$

2. Move $\neg$ inwards: $\neg\forall x, p \ \equiv \exists x\ \neg p,\quad \neg\exists x, p\ \equiv \forall x\ \neg p$:

$$\forall x\ [\exists y\ \ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ \ Loves(y,x)]$$
$$\forall x\ [\exists y\ \ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ \ Loves(y,x)]$$
$$\forall x\ [\exists y\ \ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ \ Loves(y,x)]$$

3. Standardize variables: each quantifier should use a different one

$$\forall x\ [\exists y\ \ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ \ Loves(z,x)]$$

4. Skolemize: a more general form of existential instantiation.
   Each existential variable is replaced by a Skolem function
   of the enclosing universally quantified variables:

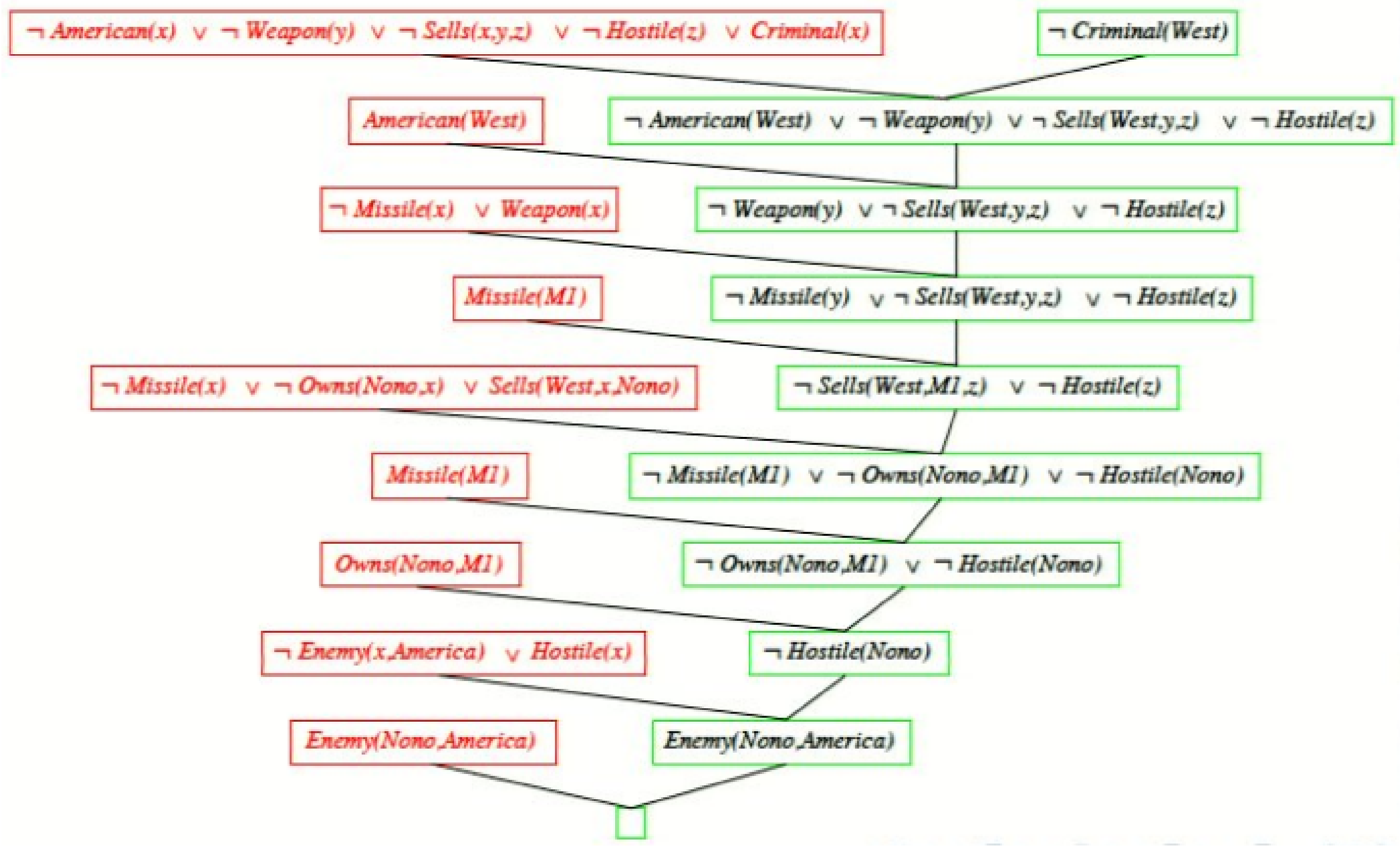
$$\forall x\ \ [Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$$

5. Drop universal quantifiers:

$$[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$$

6. Distribute $\wedge$ over $\vee$:

$$[Animal(F(x)) \vee Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$$


¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)
¬ Sells(West,M1,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
Owns(Nono,M1)
¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
¬ Enemy(x,America) ∨ Hostile(x)
¬ Hostile(Nono)
Enemy(Nono,America)
Enemy(Nono,America)

<u>Kapittel10: “Planning”</u>