Entity Behavior Modeling (Part 9)

Dr.Çağatay ÜNDEĞER

Öğretim Görevlisi
Bilkent Üniversitesi Bilgisayar Mühendisliği Bölümü

e-mail : cagatay@undeger.com
cagatay@cs.bilkent.edu.tr

Entity Behavior Modeling (Outline)

- Introduction to Artificial Intelligence (AI)
- AI Techniques & Architectures
- Some AI Algorithms/Techniques
  - Finite State Machines
  - Decision Trees
  - Artificial Neural Networks
  - Logic Programming
  - Production Systems
  - Genetic Algorithms
  - Path Planning
  - Script Programming
- Conclusion

Behavior Modeling

- Advance modeling and simulation systems
- Behavior Modeling = Artificial Intelligence

Artificial Intelligence (AI)

- Artificial intelligence deals with creating synthetic characters (agents, animats) with realistic behaviors.
- These are autonomous creatures with an artificial body situated in a virtual world.
- Job of AI developers is to give them unique skills and capabilities so that they can interact with their environment intelligently and realistically.

* Animats are artificial animals.
The term also includes physical robots and virtual simulations.

Artificial Intelligence

- AI is one of the most critical part of a simulation.
- High quality graphics, sounds, etc. attract viewers in the first sight.
- However, without a realistic AI, simulation will not serve its real purpose.

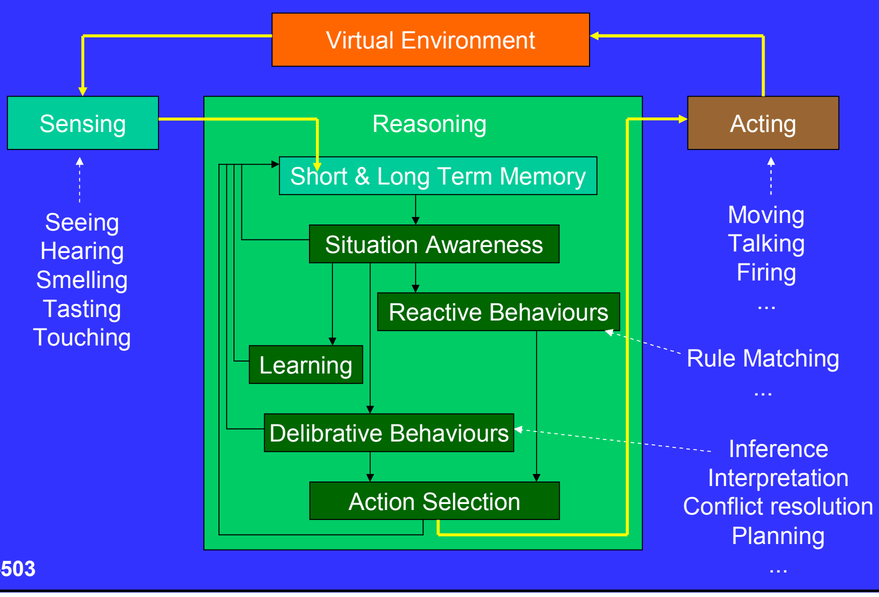
Kinds of Behaviors

- Reactive behaviors:
  - A simple decision is taken immediately depending on the developing situation (lying down when someone is shooting at you,...).
- Deliberative behaviors:
  - A complex decision is taken in a longer time by reasoning deeply (evaluation, planning, conflict resolution,...).
- Learning behaviors:
  - Learning how to act by observation, try and error.

Some AI Techniques

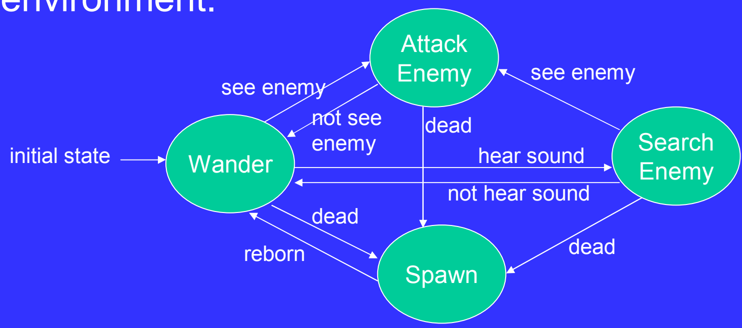
- Reactive behaviors:
  - Finite state machines
  - Fuzzy logic
- Learning (reactive) behaviors:
  - Decision trees
  - Neural networks
  - Bayesian belief networks
  - Support vector machines
  - Instance base learning
  - Reinforcement learning
- Deliberative behaviors:
  - Logic programming systems
  - Production systems
  - Theorem Provers
  - Semantic networks
  - Genetic algorithms
  - Path planning

An Example AI Architecture

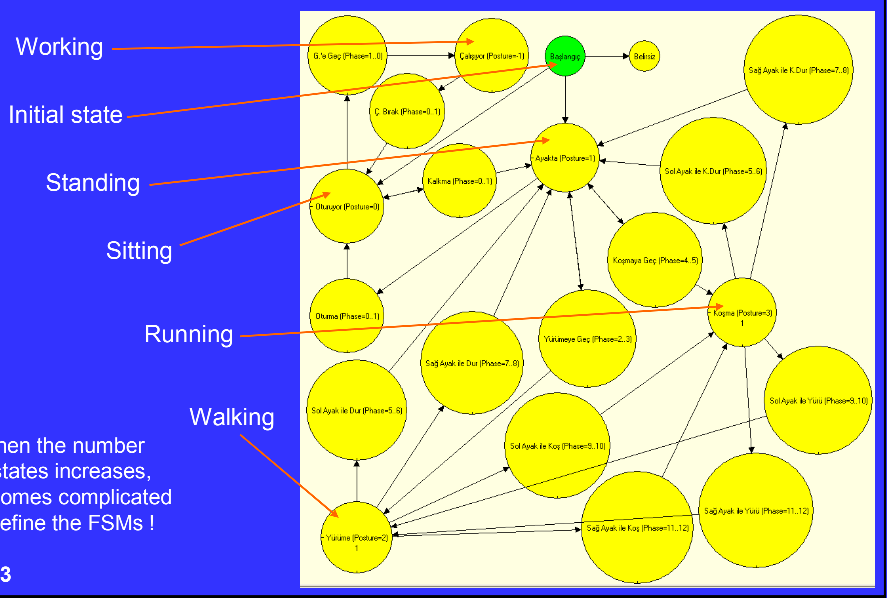


Finite State Machines (FSMs)

- Defined by a set of states and transitions between them.
- Transition from a state to another state is triggered by a change (event) in the environment.



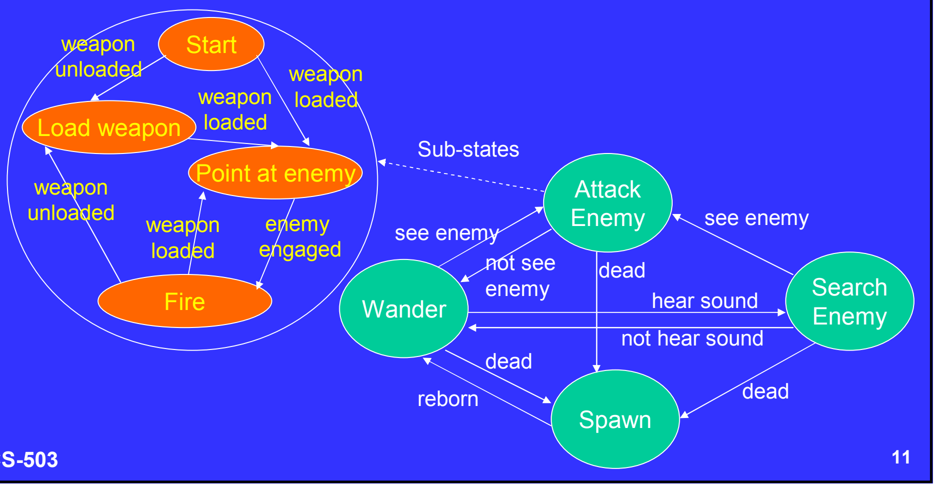
A Sample Navigation FSM



When the number of states increases, It becomes complicated to define the FSMs !

Hierarchical FSMs

- Define some higher level states (e.g. BTNs).
- Refine the details of states hierarchically.



Advantages

- Very fast to execute.
- Expressive enough for simple behaviors.
- Can create tools for non-programmer to build behaviors.
- Probabilistic transitions can be introduced to make unpredictable behaviors.

Disadvantages

- Number of states and arcs can grow very fast.
- Propositional representation:
  - Difficult to put in “pick up the best weapon”, “attack the closest enemy”
  - Expensive to count:
    - E.g. Wait until the third time I see the enemy, then attack
    - Need extra events: e.g. First time seen, second time seen, and/or extra states to take care of counting...

Decision Trees

- Rules are defined as a tree.
- Conditions are non-leaf nodes.
- Actions/decisions are leaf nodes.
- Decision trees can be learned (e.g. ID3).

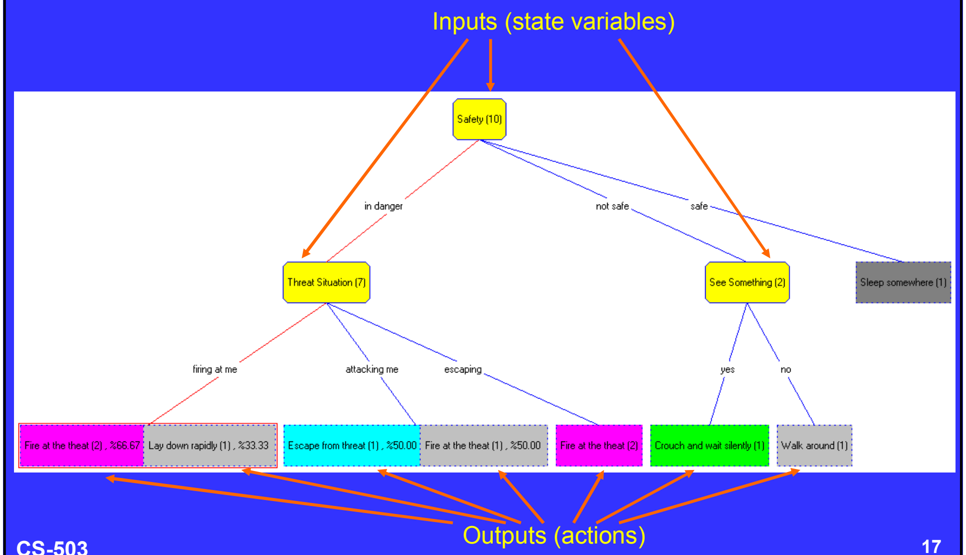
Sample Inputs & Outputs

- Inputs (State variables)
  - Safety (in danger, not safe, safe)
  - See something (yes, no)
  - Theat situation (firing at me, attacting me, escaping)
- Outputs (Actions)
  - Fire at the threat
  - Lay down rapidly
  - Escape from threat
  - Crouch and wait silently
  - Walk around
  - Sleep somewhere

Sample Instances For Learning

- If in danger, see something and threat firing at me
  - then Fire at the threat (count = 2)
- If in danger, see something and threat firing at me
  - then Lay down rapidly (count = 1)
- If in danger, see something and threat attacking me
  - then Escape from threat (count = 1)
- If in danger, see something and threat attacking me
  - then Fire at the threat (count = 1)
- If in danger, see something and threat escaping
  - then Fire at the threat (count = 2)
- If not safe and see something
  - then Crouch and wait silently (count = 1)
- If not safe and not see something
  - then Walk around (count = 1)
- If safe
  - then Sleep somewhere (count = 1)

A Sample Decision Tree



Advantages

- A simple and compact representation.
- Easy to create and understand:
  – Can also be represented as rules
- Decision trees can be learned.

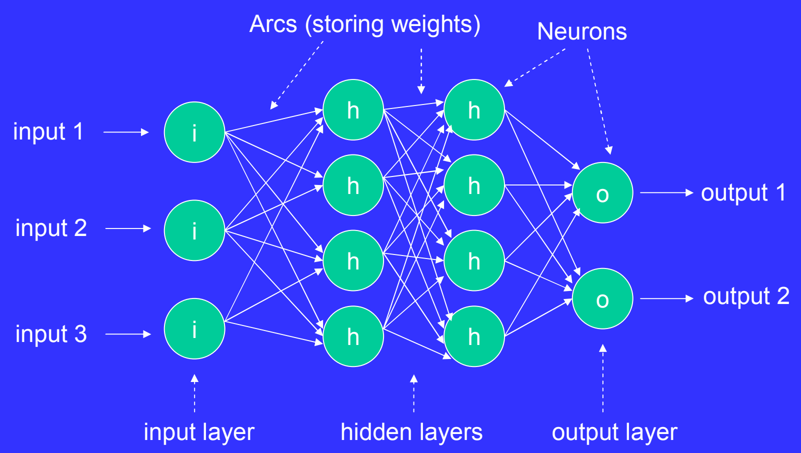
Disadvantages

- Decision tree learning algorithm is complex (hard to be coded).
- Need as many examples as possible.
- Sensitive to the errors in instances,
  - Learned decision trees may contain errors.

Artificial Neural Networks (NNets)

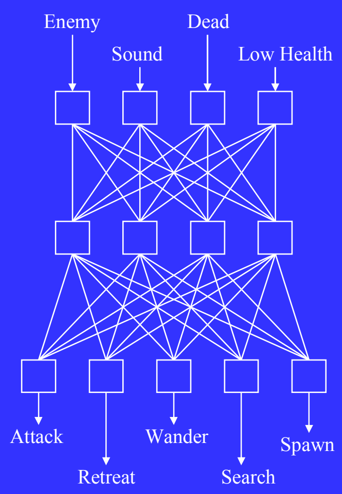
- Inspired by human brain.
- Fundamental functional units of brain are called neurons or nerve cells.
- Neurons are connected each other by axons.
- Neural networks use a similar approach and consist of neurons and arcs (axons) connecting neurons.
- Neural networks can be learned.

Neural Network Architecture



A Sample Neural Network

- The inputs are state variables that equal to either 0 (no) or 1 (yes).
- Outputs are action variables that are between 0 and 1.
- The action with the greatest output is selected.



Advantages

- Handle errors well.
- Graceful degradation.
- Can learn novel solutions.

Disadvantages

- Can’t understand how the learned network works, therefore they are the second best way to do something.
- Need as many examples as possible.
- Learning takes too much time / processing.
- Sometimes the network may not converge.

Logic Programming Systems

- Views the program and inputs as logical statements about the world.
- Use backward chaining technique with depth first search.
  - Given a query, they search for the answer.
- Prolog is the most widely used logic programming language.

CS-503 25

Prolog Language

- Consists of a set of inference rules.
- Rules are separate "If...then..." sentences.
- If conditions of a rule (premise) holds,
  - Its consequent can be inferred.
- In prolog,
  - Rules are written as reverse if-then rules.
- Consequent is in the front, and
- Premise is at the back.

If father(X,Z) and mother(Y,Z) then parents_of(X,Y,Z)

Prolog → parents_of(X,Y,Z) :- father(X,Z), mother(Y,Z)

CS-503 26

A Sample Prolog Query

Rule:

```
sister_of(X,Y) :- female(X), parents(X,M,F), parents(Y,M,F), X<>Y
```

Facts:

```
female(ayse)
parents(ayse,mustafa,zeynep)
parents(ali,mustafa,zeynep)
```

Query:

```
?-sister_of(X,ali)
```

Answer:

```
X = ayse
```


Another Sample Prolog Query

Rules:

```
action(take_out_weapon) :- safety(not_safe), weapon(in_rucksack)
action(reload_weapon) :- safety(not_safe), weapon(in_hand), weaponloaded(no)
action(put_in_weapon) :- safety(safe), weapon(in_hand)
action(walk_around) :- safety(safe), weapon(in_rucksack), sleepy(no)
action(sleep) :- safety(safe), weapon(in_rucksack), sleepy(yes)
```

Facts:

```
safety(safe)
weapon(in_hand)
```

Action query:

```
?- action(X)
```

Answer:

```
X = put_in_weapon
```

Advantages

- Not need to explicitly define all the facts.
- Can infer unknown facts from known facts.
- High expressiveness.

Disadvantages

- Inefficient and complex.
- Difficult to define the rules.
- Cannot use probabilities.

Production Systems (Rule-Based Systems)

- Logic programming systems use backward chaining techniques.
  - Given a query, they search for the answer.
- Production systems use the reverse (forward chaining) techniques.
  - Inference rules are applied to knowledgebase for finding new assertions.
  - The process is repeated forever, or until some stopping criteria is met.

Rules

- A set of inference rules.
- Rules are separate “If...then...” statements.
- If condition (premise) of a rule holds,
  - Its consequent can be asserted/executed.

A Cycle

- In each cycle,
  - The knowledge base (short term or working memory) is updated by the perception information.
  - Then the forward chainer is run to select an action to perform according to a set of condition-action rules (long term or rule memory).
    - Computes the subset of rules whose premise is satisfied (match phase)
    - Decides which of the satisfied rules are executed (conflict resolution)
  - Finally the selected actions are executed.
    - May add or remove elements from working memory.

Conflict Resolution

- Sometimes multiple rules may match at the same time.
- We need to select one/some of the rules.
- We perform conflict resolution:
  - Pick the first rule that matches.
  - Pick the most specific rule.
  - Pick the rule with the highest priority.
  - Pick the rule whose working memory elements are the most recent.
  - Pick all the rules.

A Sample Rule Set

if (not feeling safe & not carrying my weapon in hand)
    Take my weapon out from my rucksack

if (not feeling safe & carrying my weapon in hand & weapon is not loaded)
    Reload weapon

if (feeling safe & carrying my weapon in hand)
    Put my weapon back to my rucksack

if (feeling safe & not carrying my weapon in hand & not sleepy)
    Walk around

if (feeling safe & not carrying my weapon in hand & sleepy)
    Sleep

Soar

- Soar is one of the most well known production systems (1987).
- It has been used in many applications such as land force soldier modeling.

Advantages

- Not need to explicitly define all the facts.
- Can infer unknown facts from known facts.
- High expressiveness.
- Can solve problems.

Disadvantages

- Inefficient and complex.
- Difficult to define the rules.
- Cannot use probabilities.

Genetic Algorithms (GAs)

- An adaptive method which may be used to solve search, planning and optimisation problems.
- Based on genetic process of biological organisms:
  - Over many generations populations evolve.
  - Natural selection by survival of the fitness (Charles Darwin in the Origin of Species).
- Able to “evolve” solutions to real-world problems.

Genetics in Nature

- In Nature, individuals of a population compete with each other for:
  - Food, water, shelter, mate...
- Which are most successful in surviving and atracting mates will relatively have large number of offsprings.
- Poor ones will produce low, even may have no offsprings at all.

Genetics in Nature

- Genes from highly adapted “fit” individuals will spread more in each generation.
- The combination of good characterictics can produce “super-fit” offsprings whose fitness is greater than their parents.

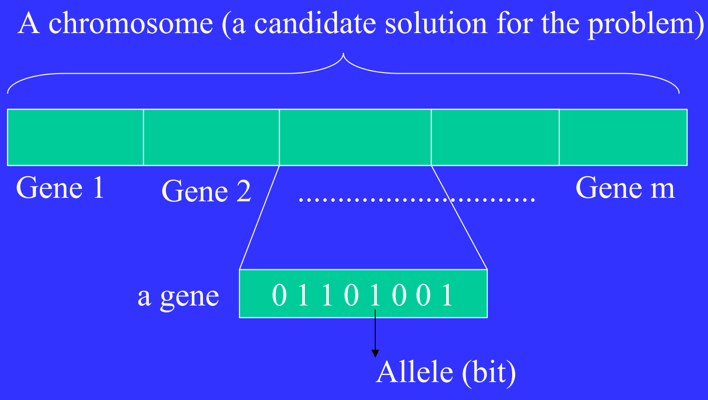
Properties of GA

- A class of stochastic search methods.
- Most stochastic search methods operator on a single solution, but GA operates on a polulation of solutions.
- Deals successfully with wide range of problem areas, especially those which are difficult for other methods to solve.
- Not guarantees to find the global optimum, but good at finding “acceptably good” solutions “acceptably quickly”.

Coding in GA

- In GA, a potential solution to a problem is represented as a set of parameters.
- Each parameter is called a "gene".
- These genes are joint together to form a string of values called a "chromosome" or a "genome" (a single potential solution to the problem).
- In general, each gene is a binary alphabet, a set of 0s and 1s.
- Each binary value (0 or 1) is called an "allele".

Coding in GA



A Population of Chromosomes

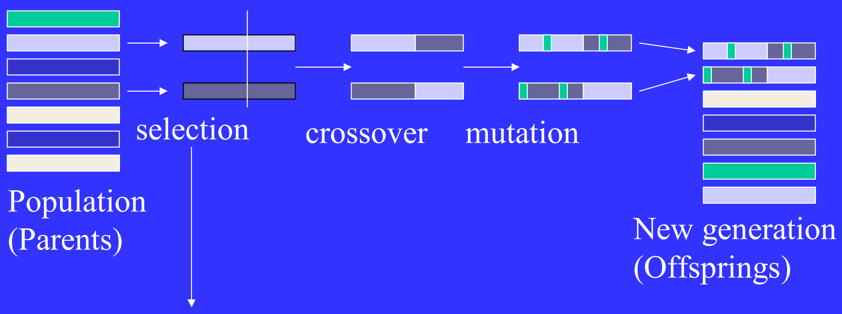
- In GA, a set of chromosomes (a polulation) is stored during the optimisation process.

n is the number of chromosomes stored in the population which is called "the population size"

Basic Algorithm

1) Initially a random population is generated.
2) The fitness of each chromosome is computed.
3) A set of genetic operator is applied to the current population. (selection, crossover, mutation)
4) A new population is created.
5) The new polulation is replaced with the current population (a new generation is formed).
6) The process is repeated until the population is converged (jump to 2).

Reproduction



Good individuals will probably be selected several times in a generation, poor ones may not be at all.

Advantages

- Can solve optimisation problems.
- Can offer sub-optimal solutions fast.
- Not domain specific.
- If a domain specific algorithm does not exist, you can adapt GA to the problem in order to have a solver.

Disadvantages

- A random search, so not guarantee good solutions.
- If it is possible to write a domain specific algorithm,
  - They will be the second best way to solve the problem.

Path Planning

- Path planning can be described as finding a sequence of moves from an initial state to a goal state.
- Path-planning algorithms:
  - Off-line,
  - On-line.

Off-Line and On-Line Algorithms

- Off-line algorithms;
  - Find the whole solution in advance,
  - Suffer from execution time.
- On the other hand, on-line algorithms;
  - Require planning and execution phases to be coupled.
  - Not designed to be optimal,
  - Usually find poor solutions.

Incremental Heuristic Search

- Hybrid solution:
  - Incremental heuristic search.
- Optimal and more efficient than off-line path planning algorithms.
- Still slow for some real-time applications.
- Considered as efficient off-line algorithms.

Changing Goals

- Common to assume that goal state is static.
- When relaxed for covering changing goals (moving target search),
  - The problem becomes very complicated.

Dealing with Changing Goals

- Off-line and incremental path planning:
  - Require re-planning towards the changing goal from scratch in each step.
- On-line search:
  - Usually designed for partially observable environments, but not for changing goals,
  - Store search information collected during the exploration.
  - There are few number of algorithms that can handle moving targets.

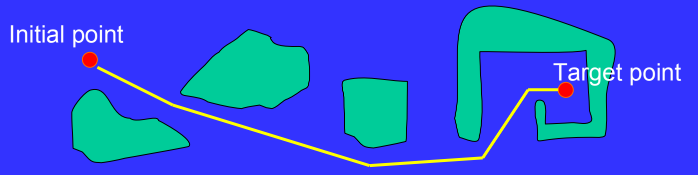
Multi-Agent Pursuit

- By increasing the number of predators involved in the environment,
  - Problem can be extended to a search against a static or moving prey with multiple coordinated agents.
- A recent research area called multi-agent pursuit,
- Not much study done so far.

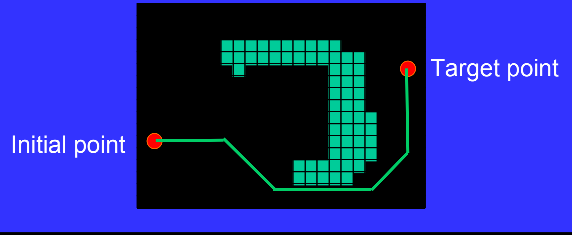
CS-503 55

Environment Representations

- Polygonal environments (e.g. Doom)



- Grid based environments (e.g. War Craft)



CS-503 56

A*

- It is common to use A* if environment is not very large.
- A* is optimal and efficient.
- The problem inputs are:
  – A graph of waypoints,
  – Initial point,
  – Target point.

CS-503

Efficiency Problems

- A* may be inefficient in very large graphs.
- Solutions:
  – Non-optimal A * versions may be used.
  – Random search algorithms (e.g. random trees, genetic algorithms, probabilistic roadmaps) may be used.
  – Increamental heuristic search algorithms (e.g. D*, D*-Lite) may be used.
  – Real-time search algorithms (e.g. RTA*, MTES, MAPS) may be used.

CS-503 58

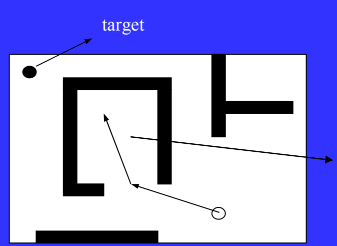
Real-Time A* (RTA*)

- Learning Real-Time A* (LRTA*) and
- Real-Time A* (RTA*)
  - Proposed by Korf,
  - Former generic heuristic search algorithms ***for fixed goals***.
  - Build and update a table containing heuristic estimates.
- LRTA* is able to learn optimal table values in single or multiple runs.
- RTA* performs better than LRTA* in the first run,
- Lack of learning optimal table values.

CS-503 59

Heuristic Depression

- The original RTA* only considers immediate successors to determine the next move.
- Stuck in semi-closed regions for a long time.
- ***A heuristic depression*** is a local maximum, whose heuristic values have to be filled up before the agent can escape from it.



The agent will be stuck in that semi-closed region for a long time, because the target is at north-west and the only out is at south-east.

CS-503 60

RTA* with *n*-Look-Ahead Depth

- RTA* can easily be extended to have any arbitrary ***look-ahead depth***.
- Instead of examining immediate neigbors of the current state, *n* level neighbors are used.
- Reduces the number of moves to reach the goal significantly,
- Exponential in the look-ahead depth.
- Large *n* is not preferred in practice.

Moving Target Search (MTS)

- RTA* and many variations of these algorithms are all limited to work on fixed goals,
- Ishida and Korf proposed Moving Target Search (MTS).
  - Built on LRTA*
  - Capable of pursuing a moving target.
- MTS is a poor algorithm.
- When the target moves, the learning process has to start all over again.
- A performance bottleneck in heuristic depressions.

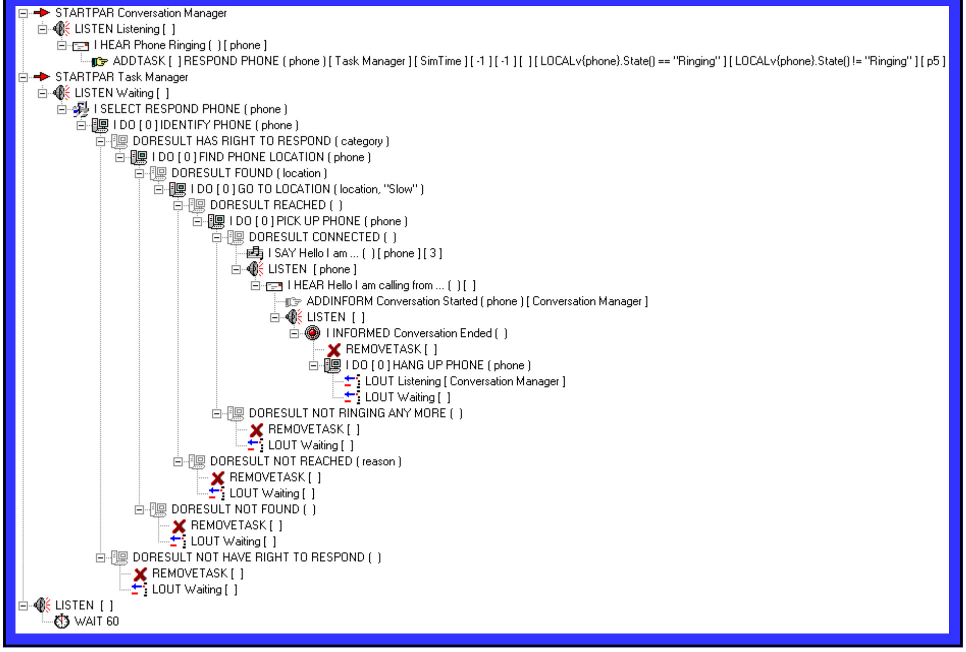
MTS-c & MTS-d

- Since original MTS is a poor algorithm, two MTS extensions:
  - *Commitment to Goal* (MTS-c) and
  - *Deliberation* (MTS-d) are proposed by Ishida.
- To use the learned table values more effectively,
  - MTS-c ignores some of the target's moves,
  - MTS-d performs an off-line search to update the heuristic values.

Script Programming

- In games, script programming is very commonly used.
- Some high level special script languages may be assumed as AI techniques.

A Sample High Level Script (HLTMS)



Conclusion

- There is no best algorithm.
- Solution depends on the style of game.
- Define the problem clearly.
- Develop an algorithm for the problem.