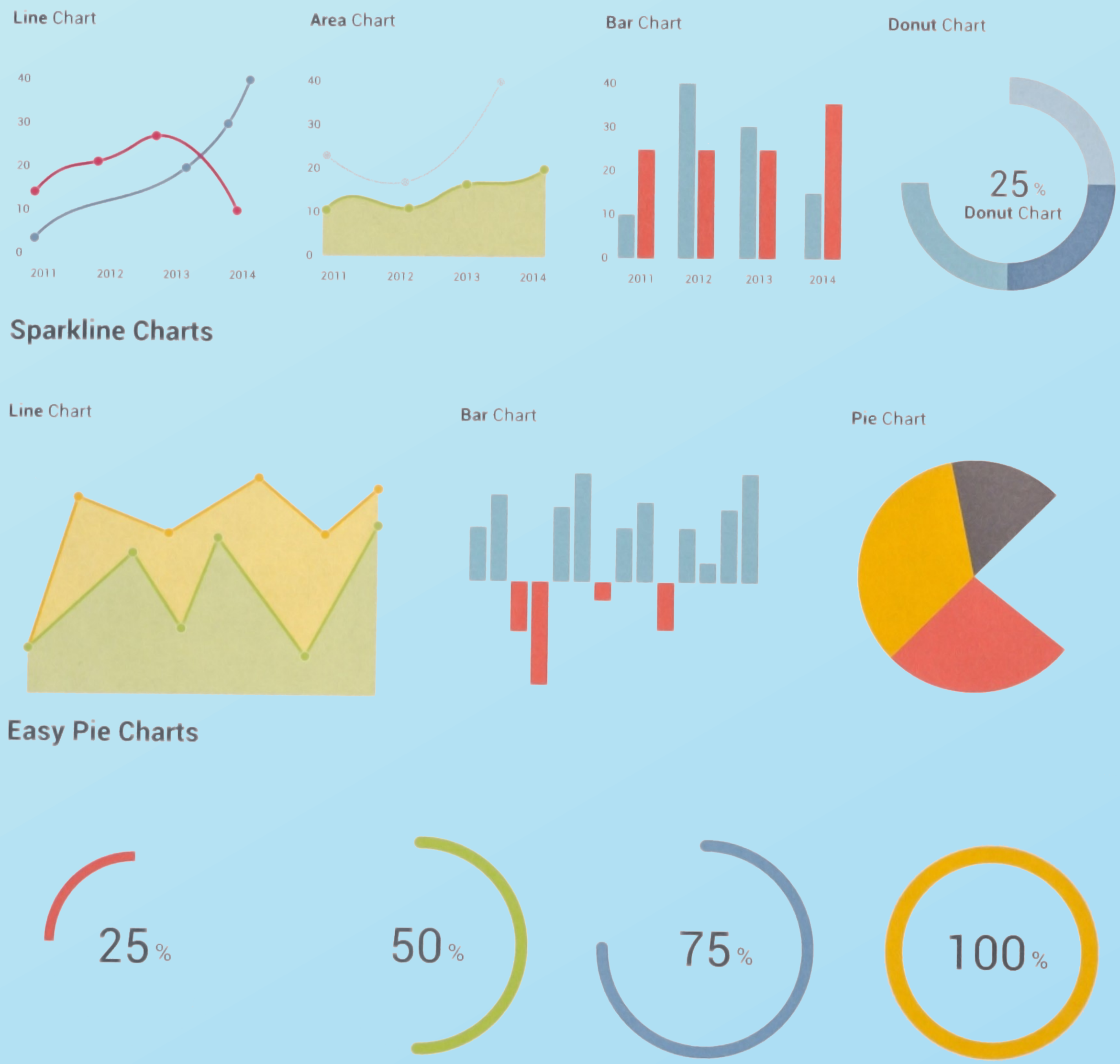
Intelligent Systems

Course 2021-22



Search and Logic

Search, Propositional logic, First order logic

Author: David Ramírez Arco

INTRODUCTION

In this Assignment, I am going to apply the *A* algorithm*, the *resolution algorithm*, and the *translation to first order logic algorithm* in detail. The document will be divided into three parts explaining how it works, and the application:

1. Search (A*):

Design of a search problem to apply step by step the studied A* algorithm, the solution length should have at least three steps

2. Propositional logic:

Create a knowledge base with at least 4 sentences and a conclusion(inference), then apply the resolution algorithm step by step

3. First order logic:

Translate at least 4 sentences from natural language to the first order logic. Every quantifier and operator must be used at least once in the set of sentences

Search A*

A* (or ‘A star’) is a computer algorithm that is widely used in pathfinding and graph traversal. The algorithm efficiently plots a walkable path between multiple nodes, or points, on the graph.

A* algorithm introduces a heuristic into a regular graph-searching algorithm, essentially planning ahead at each step so a more optimal decision is made. A* is an extension of Dijkstra's algorithm with some characteristics of breadth-first search.

Like Dijkstra, A* works by making a lowest-cost path tree from the start node to the target node. What makes A* different and better for many searches is that for each node, A* uses a function $f(n)$ that gives an estimate of the total cost of a path using that node. Therefore, A* is a heuristic function, which differs from an algorithm in that a heuristic is more of an estimate and is not necessarily provably correct.

A* expands paths that are already less expensive by using this function:

$$f(n)=g(n)+h(n)$$

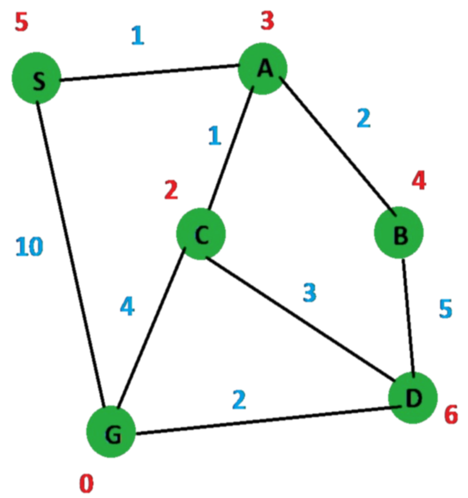
Where

- $f(n)$ = total estimated cost of path through node n
- $g(n)$ = cost so far to reach node n
- $h(n)$ = estimated cost from n to goal. This is the heuristic part of the cost function, so it is like a guess

The calculation of $h(n)$ can be done in different ways

If $h(n)$ = 0 A* becomes Dijkstra's algorithm, which guaranteed to find the shortest path.

Considering the following graph, the Search (A*) algorithm will be applied to find the solution



The number written with **blue** are the distance between the nodes and written with **red** are the heuristics value

The following table contains the heuristic values:

	S	A	B	C	D	G
Estimation	5	3	4	2	6	0

- Let's start at node S and set the goal to node G

First Iteration of the main loop:

- You can get from that to node A and node G

$$s \rightarrow A \Rightarrow f(n) = g(n) + h(n) = 1 + 3 = 4 \quad ✔$$

$$s \rightarrow G \Rightarrow f(n) = g(n) + h(n) = 10 + 0 = 10$$

We decide to follow the path $s \rightarrow A$

Second Iteration of the main loop:

- Node B and node C can be reached from node A

$$S \rightarrow A \rightarrow B \Rightarrow f(n) = g(n) + h(n) = 3 + 4 = 7$$

$$S \rightarrow A \rightarrow C \Rightarrow f(n) = g(n) + h(n) = 2 + 2 = 4 \quad ✔$$

We decide to follow the path $S \rightarrow A \rightarrow C$

Third Iteration of the main loop:

$$S \to A \to C \to D \Rightarrow f(n) = g(n) + h(n) = 3 + 4 = 7$$

$$S \to A \to C \to G \Rightarrow f(n) = g(n) + h(n) = 6 + 0 = 6 \quad ✔$$

We decide to follow the path $S \to A \to C \to G$ and we reach the goal node

We can see that the solution heavily depends on heuristics, but nevertheless is one of the best paths finding algorithms.

Propositional Logic

We are going to transform the Knowledge base into the Conjunctive normal form following this steps to show that KB $\models$ α:

1. Eliminate $\Leftrightarrow$ by replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.
2. Eliminate $\Rightarrow$ by replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$.
3. Move $\neg$ inwards by repeated application of:
    - $\neg(\neg\alpha) = \alpha$
    - $\neg(\alpha \land \beta) = \neg\alpha \lor \neg\beta$
    - $\neg(\alpha \lor \beta) = \neg\alpha \land \neg\beta$
4. Apply distributivity of $\lor$ over $\land$ whenever possible:
    - $\alpha \lor (\beta \land \gamma) = (\alpha \lor \beta) \land (\alpha \lor \gamma)$

If Manu had coronavirus, Paco (the father) will get sick. If Sergio had coronavirus, Maite (the mother) will get sick. If either Paco or Maite are sick, then Cristina (the grandmother) will die. Cristina is not dead. Consequently, Manu hasn't coronavirus and Sergio hasn't coronavirus.

Knowledge bases (KB):

ManuCoronavirus $\Rightarrow$ PacoSick
SergioCoronavirus $\Rightarrow$ MaiteSick
PacoSick $\lor$ MaiteSick $\Rightarrow$ CristinaDied
$\neg$CristinaDied

α:

$\neg$ManuCoronavirus $\land\neg$ SergioCoronavirus

CNF conversion algorithm:

1 $\neg$ManuCoronavirus $\lor$ PacoSick

2 $\neg$SergioCoronavirus $\lor$ MaiteSick

3 $\neg$PacoSick $\lor$ CristinaDied

4 $\neg$MaiteSick $\lor$ CristinaDied

5 $\neg$CristinaDied

6 ManuCoronavirus $\lor$ SergioCoronavirus

Resolution algorithm:

7	$\neg$MaiteSick	Resolve 5 and 4
8	$\neg$PacoSick	Resolve 5 and 3
9	$\neg$ManuCoronavirus	Resolve 8 and 1
10	$\neg$SergioCoronavirus	Resolve 7 and 2
11	SergioCoronavirus	Resolve 9 and 6
12	False	Resolve 11 and 10

Conclusion:
If the clause were invalid, we could have proved it by giving a counterexample.
So the argument was valid ✔

First Order Logic

Constants: Kira, David

Predicates:

- Owner(x) → x is an owner
- Dog(x) → x is a dog
- Is(x, y) → x is y
- Plays (x, y) → x plays y
- Own (x, y)→ x is owner of y

Sentences:

- Kira has an owner
  $\exists x\ Own(x, \text{Kir}a)$
- Some dog has no owner
  $\exists x\ [Dog(x) \wedge \neg \exists y\ Own(y, x)]$
- The dog is black or white
  Is(Cat, Black) ⇔ ¬ *Is*(Cat, White)
- An owner plays with a dog if and only if it is his own dog
  $\forall x \forall y[Owner(x) \wedge \text{Dog(y)} \wedge \text{plays(x, y)} \Leftrightarrow Own(x, y)]$
- David doesn't play with two different dogs
  $\neg \exists x \exists y[Dog(x) \wedge \text{Dog(y)} \wedge \text{plays(David, x)} \wedge \text{play(David, y)} \wedge \text{x = y}]$