Τμήμα: Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Μάθημα: Εργαστήριο Τεχνητής Νοημοσύνης.
Διδάσκοντες: Καθηγ. Μανόλης Μαρακάκης,
Δρ, Κονδυλακης Χαρίδημος
Ημερομηνία παράδοσης 6 Απριλίου 2022
Εαρινό εξάμηνο 2021-22

Εργασία 2

Να κατασκευάσετε κατηγόρημα breadthFirstSearch(InitialState, FinalState) το οποίο είναι αληθές εάν InitialState είναι μια αρχική κατάσταση σ' ένα δέντρο αναζήτησης και FinalState είναι μια τελική κατάσταση και βρίσκει με αναζήτηση πλάτος-πρώτα το μονοπάτι από την αρχική κατάσταση InitialState στη τελική κατάσταση FinalState και το εκτυπώνει. Εάν το FinalState είναι μεταβλητή να βρίσκει όλα τα μονοπάτια με αναζήτηση πλάτος-πρώτα από την αρχική κατάσταση InitialState στις τελικές καταστάσεις.
Το κατηγόρημα breadthFirstSearch/2 να υλοποιεί σε Prolog τον παρακάτω αλγόριθμο ο οποίος δίνεται σε ψευδογλώσσα. Αυτός ο αλγόριθμος εκτελεί αναζήτηση χώρου καταστάσεων «πλάτος-πρώτα (breadth-first)». **L1** είναι η λίστα των μη αναπτυγμένων/επεκταμένων κόμβων ή **ανοικτή FIFO** λίστα και **L2** είναι η λίστα των αναπτυγμένων κόμβων ή **κλειστό σύνολο** (τα στοιχεία δεν έχουν σειρά).

συνάρτηση αναζήτηση-πλάτος-πρώτα(ΑρχικήΚατάσταση,ΤελικήΚατάσταση)
αρχή
L1 := [ΑρχικήΚατάσταση]; **L2** := [ ]; **%** αρχικοποίηση
ενώ L1 ≠ [ ] **κάνε** **%** Υπάρχουν καταστάσεις για έλεγχο;
αρχή
αφαίρεσε από την αρχή της **L1** την κατάσταση **S**;
εαν η **S** είναι λύση **τότε** επέστρεψε **επιτυχία** **%** μια λύση βρέθηκε
διαφορετικά
αρχή
δημιούργησε τις επόμενες καταστάσεις (παιδιά) της S;
βάλε την κατάσταση **S** στην **L2**;
% έλεγχος για βρόγχο
απόρριψε κάθε παιδί της **S** εάν είναι ήδη στην **L1** ή στην **L2**;
βάλε τα υπόλοιπα παιδιά της S **στο τέλος** της **L1**; **%** ουρά
τέλος
τέλος
επέστρεψε αποτυχία **%** δεν έμειναν καταστάσεις για έλεγχο
τέλος

Οδηγίες υλοποίησης.

1. Η ανοικτή λίστα **L1** είναι **FIFO** λίστα, δηλαδή ουρά. Για την επεξεργασία των στοιχείων της θα χρησιμοποιήσετε τις πράξεις της ουράς όπως έχουν υλοποιηθεί

στην ενότητα 8.5 του βιβλίου «***Prolog: Προγραμματισμός σε Λογική για ΤΝ***» (έκδοση 2019)**.**

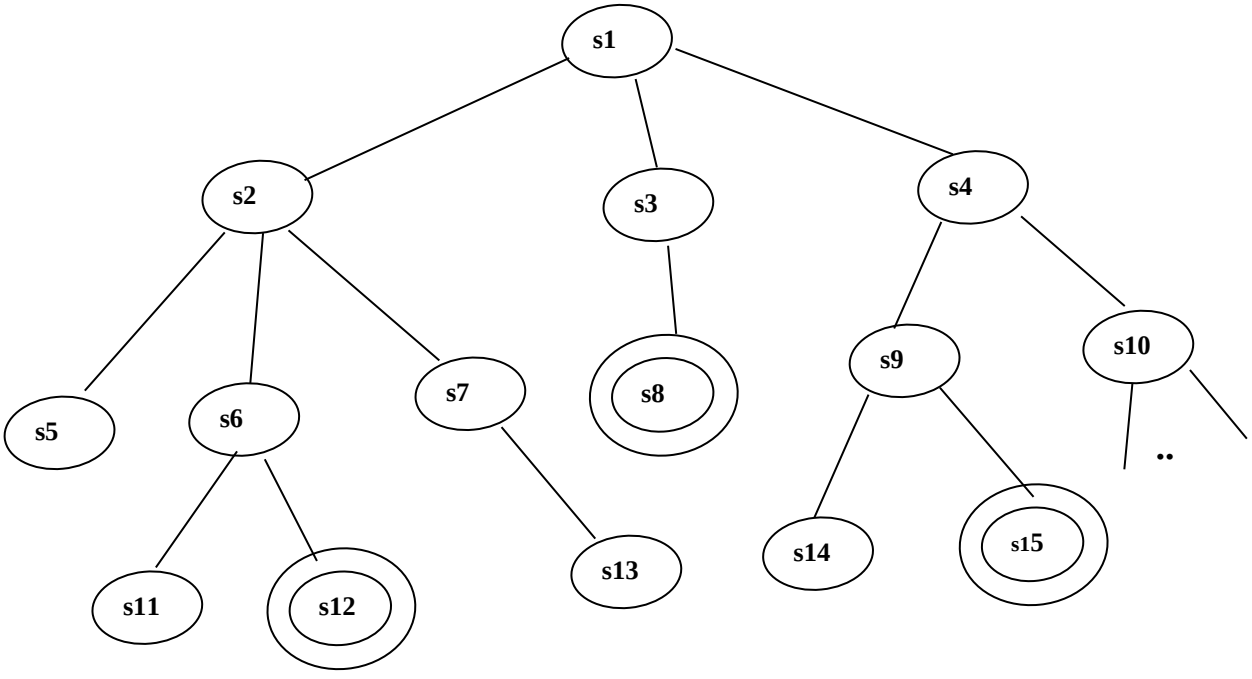
2. Η κλειστή λίστα **L2** να υλοποιηθεί ως σύνολο επειδή τα στοιχεία της δεν έχουν σειρά. Για την επεξεργασία των στοιχείων της θα χρησιμοποιήσετε τις πράξεις των συνόλων όπως έχουν υλοποιηθεί στην ενότητα 8.5 του βιβλίου «***Prolog: Προγραμματισμός σε Λογική για ΤΝ***»**..**
3. Να υλοποιήσετε το πρόβλημα χρησιμοποιώντας το πρόγραμμα 10.5 από το βιβλίο «***Prolog: Προγραμματισμός σε Λογική για ΤΝ***»**.** (έκδοση 2019) το οποίο βρίσκει τα μονοπάτια από μια αρχική κατάσταση προς στις τελικές καταστάσεις σε κυκλικό γράφο. Όσοι έχετε την παλιά έκδοση του βιβλίου (έκδοση 2014) αντί για το πρόγραμμα 10.5 του βιβλίου σας να χρησιμοποιήσετε την παρακάτω βελτιωμένη έκδοση αυτού του προγράμματος.

```
monopati(X, Y, Monopati) :-
        monopati1(X, Y, [X], Monopati).

monopati1(X, X, Monopati, Monopati) :-
        telike_katastase(X).
monopati1(X, Z, Monopati, Teliko_monopati) :-
        akme(X, Y),
        \+ member(Y, Monopati),
        append(Monopati, [Y], Neo_monopati),
        monopati1(Y, Z, Neo_monopati, Teliko_monopati).
```

4. Στην υλοποίηση σας να ακολουθήσετε το παράδειγμα της υλοποίησης του αλγορίθμου αναζήτησης βάθος-πρώτα το οποίο κάναμε στη τάξη.

Να εξετάσετε το πρόγραμμα σας στο παρακάτω δέντρο αναζήτησης, στο οποίο ο κόμβος **s1** είναι αρχική κατάσταση και οι κόμβοι **s8, s12** και **s15** είναι τελικές καταστάσεις. Η αναπαράσταση του δέντρου αναζήτησης να εκφράζεται ως ένα σύνολο από γεγονότα της μορφής «move(K1,K2)», που σημαίνει ότι η μετακίνηση από τον κόμβο K1 στον κόμβο K2 είναι έγκυρη και τα K1,K2 παίρνουν τιμές από το σύνολο {s1,s2, s3,s4, s5,s6, s7,s8, s9,s10, s11,s12, s13,s14, s15}. Για παράδειγμα τα γεγονότα « move(s4,s9), move(s9,s14), move(s14,s15)» παριστούν ένα μέρος του δέντρου αναζήτησης.



Να εξετάσετε το πρόγραμμα σας στους στόχους.

GOALS

- ?- bfs(s1,s15).
  Path of the solution: s1 --> s4 --> s9 --> s15 yes
- ?- bfs(s1,s8).
  Path of the solution: s1 --> s3 --> s8 yes
- ?- bfs(s1,s12).
  Path of the solution: s1 --> s2 --> s6 --> s12 yes
- ?- bfs(s1,G).
  Path of the solution: s1 --> s3 --> s8 G = s8 ? ;
  Path of the solution: s1 --> s2 --> s6 --> s12 G = s12 ? ;
  Path of the solution: s1 --> s4 --> s9 --> s15 G = s15 ? ;
  No solution found. true ? yes