Readme για το Pacman

Άγγελος Τσιτσόλη sdi2000200

Φεβρουάριος 26, 2022

Πρόβλημα 1

Στο πρώτο ερώτημα μας ζητείται να κάνουμε μια σειρά απο ενέργειες . Αρχικά για την δημιουργία του sentence1 και του sentence2 χρησιμοποιώ την συνάρτηση Expr , προκειμένου να μετατρέψω κάθε φορά τις λογικές εκφράσεις πυο δίνονται (π.χ A ,B , C, D) σε κατάλληλη μορφή Expr ώστε να δημιουργήσω μαυτά στην συνέχεια τις προτάσεις που ζητείται για την δημιουργία της τελικής λογικής έκφρασης που θα επιστρέψω . Συγκεκριμένα σκέφτηκα να φτιάχνω μικρές εκφράσεις με τις λογικές εκφράσεις μορφής Expr (που θα προκύψουν μέσω της κλήσης της συνάρτησης Expr() κάθε φορά) και στην συνέχεια να τα τοποθετώ σε λίστες ώστε να είναι μαζεμένα και πιο εύκολο να σχηματίσω έτσι την τελική λογική έκφραση που μου ζητείται.Για την δημιουργία ορισμένων προτάσεων χρησιμοποιώ την συνάρτηση disjoin η οποία προσθέτει το λογικό "or" μεταξύ εκφράσεων όπως επίσης και την συνάρτηση conjoin η οποία προσθέτει το λογικό "and" μεταξύ εκφράσεων .Στην συνέχεια αφού έχω δημιουργήσει τις προτάσεις (απο τις οποίες αποτελείται η τελική λογική έκφραση) και τις έχω φέρει κάπου μαζεμένες (όπως είπα σε μια λίστα πχ)τότε τις ενώνω μέσω της συνάρτησης conjoin ώστε να δημιουργηθεί έτσι μια έκφραση Expr που θα περιέχει όλες τις προτάσεις που έφτιαξα και επιστρέφω το αποτέλεσμα αυτής της ένωσης .Στην συνάρτηση sentence1() και στην συνάρτηση sentence2() κάνω πολύ παρόμοια πράγματα .Στην συνάρτηση Sentence3() κάνω τα ίδια πράγματα απλώς χρησιμοποιώ την συνάρτηση PropSymbolExpr() μέσω της οποίας φτιάχνω τις εκφράσεις που ζητείται όπως για παράδειγμα " Pacman is born at time 0." Για την δημιουργία των sentence1,sentence2,sentence3 συναρτήσεων απλώς ακολουθώ τις οδηγίες που δίνονται στην εκφώνηση.

Στην συνέχεια τεστάρω τις προτάσεις που δημιουργήθηκαν προηγουμένως στην συνάρτηση findmodel όπου και κρατάω σε σχόλια τα αποτελέσματα των κλήσεων της συνάρτησης μέσα στην συνάρτηση.

Για την υλοποίηση της findModelCheck ,θα πρέπει να φτιάξω ένα λεξιλόγιο που θα περιέχει την λογική έκφραση και την τιμή True ή False , ώστε η συνάρτηση να είναι παρόμοια με την συνάρτηση findModel.Συγκεκριμένα χρησιμοποιώ την κλάση dummyclass έτσι ώστε να μπορέσω να αναπαραστήσω μια λογική έκφραση στην περίπτωση μας το a χωρίς τα " " , διότι όσο προσπαθούσα να αναπαραστήσω την λογική έκφραση ως string τότε μου προέκυπτε συνεχώς λάθος , αλλά χρησιμοποιώντας την dummyclass η αναπαράσταση γίνεται όπως πρέπει δηλαδή όπως προκύπτουν απο την συνάρτηση Expr().Αυτό καθώς το έψαξα γενικότερα είδα ότι υλοποιείται χάρης την συνάρτηση repr όπου στην python έχει την ιδιότητα να κάνει αυτήν την συγκεκριμένη αναπαράσταση .Τέλος επιστρέφω το αποτέλεσμα που προκύπτει.

Στην συνάρτηση entails ζητείται ουσιαστικά εφαρμογή της γνωστής μεθόδου της ανάλυσης . Ουσιαστικά μέσω της ανάλυσης αποδεικνύουμε ότι μια λογική πρόταση A έπεται λογικά της knowledge base μας . Ο τρόπος με τον οποίο γίνεται η ανάλυση είναι ο εξής : παίρνουμε την αρνητική μορφή της πρότασης που θέλουμε να δείξουμε ότι έπεται λογικά και την κάνουμε ουσιαστικά conjoin με την βάση μας ώστε να προκύψει κάτι του τύπου $KB \wedge \neg A$. Στην συνέχεια προσπαθούμε μέσω απαλοιφών να δείξουμε ότι καταλήγουμε σε κενό , ώστε να μπορούμε να πούμε εν τέλει ότι αποδεικ΄νυετασι αυτό που θέλουμε . Όπως στην ανάλυση έτσι και εδώ θα κάνουμε τα ίδια ακριβώς βήματα απλώς θα χρησιμοποιήσουμε την συνάρτηση findModel για να ελέγξουμε αν υπάρχει ένα ικανοποιητικό μοντέλο . Σε περίπτωση που δεν υπάρχει ικανοποιητικό μοντέλο τότε αποδείξαμε αυτό που θέλουμε αλλιώς αποδεικνύεται ότι η πρόταση A δεν έπεται λογικά.Άρα αυτό εφαρμόζω και στην συνάρτηση entails.

Για την συνάρτηση pl_inverse απλά ακολούθησα ακριβώς τις οδηγίες .

Πρόβλημα 2

Προκειμένου η λογική μας πρόταση (που αποτελείται απο λογικές εκφράσεις) να είναι αληθής αν τουλάχιστον μία απο τις εκφράσεις είναι αληθής τότε πολύ απλά κάνω disjoin τις εκφράσεις . Καθώς αν έχω διάζευξη χρειάζεται μόνο μία απο τιθς εκφράσεις να είναι αληθής για να είναι συνολικά η πρόταση αληθής.

Προκειμένου η λογική μας πρόταση (που αποτελείται απο λογικές εκφράσεις) να είναι αληθής αν το πολύ μία απο τις εκφράσεις δωθεί ως αληθής ή και καμία τότε αρχικά χρησιμοποίησα την συνάρτηση itertools.combinations() η οποία αυτό που θα κάνει είναι ότι , με τις λογικές εκφράσεις που θα τις δώσουμε θα δημιουργήσει τους κατάλληλους συνδυασμούς μεταξύ των λογικών εκφράσεων . Αυτό δίνεται απο την εκφώνηση οπότε έτσι και αλλιώς θα έπρεπε να το κάνουμε οπότε έχοντας αυτά σκέφτηκα το εξής : σε περίπτωση που δίνονται οι λογικές εκφράσεις A,B,C,D και δημιουργούνται οι συνδυασμοί απο την κλήση της συνάρτησης που προαναφέρα τότε θα προκύψει κάτι του τύπου [(A, B), (A, C), (A, D), (B, C), (B, D), (C, D)]. Οπότε απο αυτήν την λίστα κάνω για κάθε συνδυασμό αρνητικές τις λογικές εκφράσεις απο τις οποίες αποτελείται , επίσης για κάθε συνδυασμό προσθέτω μεταξύ των λογικών εκφράσεων που αποτελούν τον συνδυασμό το λογικό or ώστε να ισχύει ή η μία λογική πρόταση ή η άλλη και τέλος εννώνω με το λογικό and όλους τους συνδυασμούς. Οπότε ουσιαστικά σε περίπτωση που δωθεί ότι το A είναι TRUE προκύπτει το εξής [(¬A (F) ∨ ¬B(T)) ∧ (¬A(F)∨ ¬C(T))∧ (¬A(F)∨ ¬D(T)∧ (¬B(T)∨ ¬C(T))∧ (¬B(T)∨ ¬D(T))∧ (¬C(T)∨ ¬D(T)] , άρα θα η τελική έκφραση θα είναι αληθής ενώ σε περίπτωση που δωθούν σε δύο ή παραπάνω εκφράσεις απο αυτές που δίνονται η τιμή True τότε θα προκύψει false ως τελικό αποτέλεσμα .Για παράδειγμα Αν δωθεί η τιμή True στις λογικές εκφράσεις A και B τότε προκύπτει [(¬A (F) ∨ ¬B(F)) ∧ (¬A(F)∨ ¬C(T))∧ (¬A(F)∨ ¬D(T)∧ (¬B(F)∨ ¬C(T))∧ (¬B(F)∨ ¬D(T))∧ (¬C(T)∨ ¬D(T)] άρα θα είναι False το τελικό αποτέλεσμα.Ομοίως αν καμία δεν δωθεί ότι είναι True θα προκύψει το επιθυμητό αποτέλεσμα [(¬A (T) ∨ ¬B(T)) ∧ (¬A(T)∨ ¬C(T))∧ (¬A(T)∨ ¬D(T)∧ (¬B(T)∨ ¬C(T))∧ (¬B(T)∨ ¬D(T))∧ (¬C(T)∨ ¬D(T)]

Για την συνάρτηση exactlyOne() κάνουμε σχεδόν το ίδιο με την προηγούμενη συνάρτηση , απλώς στην περίπτωση αυτή δίνεται ότι η τελική λογική έκφραση θα βγεί αληθής μόνο αν δωθεί ότι μία απο τις λογικές εκφράσεις(π.χ. A,B,C,D) είναι αληθής .Δηλαδή σε περίπτωση που δωθεί ότι καμία απο τις λογικές εκφράσεις δεν είναι αληθής ή πάνω απο μία δωθεί αρχικά αληθής τότε δεν θα έχουμε αληθή την τελική λογική έκφραση.Για να υλοποιηθεί αυτό θα χρησιμοποιήσω ότι και πριν απλώς θα προσθέσω και το εξής: σε περίπτωση που έχω τις εκφράσεις (A,B,C,D) τότε προσθέτω (A ∨ B ∨ C ∨ D) . Μ'αυτό τον τρόπο τον τρόπο 'σιγουρεύουμε' ουσιαστικά ότι μόνο αν δωθεί μόνο μία λογική έκφραση ότι είναι αληθής , μόνο τότε θα δειχθεί ότι η τελική έκφραση θα είναι αληθής.Για παράδειγμα αν δωθεί ότι το A είναι TRUE προκύπτει το εξής [(¬A (F) ∨ ¬B(T)) ∧ (¬A(F)∨ ¬C(T))∧ (¬A(F)∨ ¬D(T)∧ (¬B(T)∨ ¬C(T))∧ (¬B(T)∨ ¬D(T))∧ (¬C(T)∨ ¬D(T)) ∧ (A (T)∨ B(F) ∨ C(F) ∨ D(F))]. Ωστόσο σε περίπτωση που δωθεί ότι True είναι οι λογικές εκφράσεις A και B τότε προκύπτει [(¬A (F) ∨ ¬B(F)) ∧ (¬A(F)∨ ¬C(T))∧ (¬A(F)∨ ¬D(T)∧ (¬B(F)∨ ¬C(T))∧ (¬B(F)∨ ¬D(T))∧ (¬C(T)∨ ¬D(T) ∧ (A (T)∨ B(T) ∨ C(F) ∨ D(F))] άρα γίνεται κατανοητό ότι η τελική λογική πρόταση δεν θα είναι True.Ομοίως και σε περίπτωση που όλες είναι false [(¬A (T) ∨ ¬B(T)) ∧ (¬A(T)∨ ¬C(T))∧ (¬A(T)∨ ¬D(T)∧ (¬B(T)∨ ¬C(T))∧ (¬B(T)∨ ¬D(T))∧ (¬C(T)∨ ¬D(T) ∧ (A (F)∨ B(F) ∨ C(F) ∨ D(F))]

Πρόβλημα 3

Στην συνάρτηση pacmanSuccessorAxiomSingle αυτό που κάνω είναι απλά να επιστρέφω αυτό που αναφέρουν τα σχόλια δηλαδή το εξής : Current <==> (previous position at time t-1) (took action to move to x, y)

Για την συνάρτηση pacphysicsAxioms υλοποιώ τα εξής :

1. for all (x, y) in all_coords:
   If a wall is at (x, y) −− > Pacman is not at (x, y)

2. Pacman is at exactly one of the squares at timestep t.
3. Pacman takes exactly one action at timestep t.
4. Results of calling sensorModel(...), unless None.
5. Results of calling successorAxioms(...), describing how Pacman can end in various locations on this time step. Consider edge cases. Don't call if None.

Για το πρώτο απο τα παραπάνω ουσιαστικά αυτό που κάνω είναι να εκφράσω κάθε θέση ως θέση του που μπορεί να βρίσκεται το πακμαν και αλλά και ως θέση που μπορούμε να έχουμε τοίχο . Οπότε για κάθε θέση (all_coords) υλοποιώ την εξής έκφραση If a wall is at (x, y) −− > Pacman is not at (x, y) (βάζω σε μια λίστα τις θέσεις του πακμαν και σε μια άλλη τις θέσεις για τους τοίχους και χρησιμοποιώντας και την συνάρτηση zip φτιάχνω αυτές τις εκφράσεις που μου ζητείται για κάθε (x,y)).Όπου τέλος τα κάνω conjoin και τα προσθέτω στην βάση.

Για το δεύτερο μπούλετ ουσιαστικά κάνω την διαδικασία που υλοποιώ στην συνάρτηση exactly one **Γενικότερα , αυτό το κομμάτι κώδικα εμφανίζεται συνεχώς στον κώδικα , συγκεκριμένα είναι ο κώδικας που χρησιμοποιώ για να υλοποιήσω την συνάρτηση exactlyone . (Γνωρίζω ότι Θα μπορούσα να χρησιμοποιήσω απλώς την exactlyone απευθείας απλώς δεν είχα χρόνο να αλλάξω όλο τον κώδικα ξανά οπότε το άφησα έτσι εξηγώντας το παρακάτω.**:

```
lit=[]   ###### From here the following proc
for x in squares_list :
    lit.append(~x)
clauses=list(itertools.combinations(lit,2))
lit2=[]
lit3=[]
final=[]
lit3=disjoin(squares_list)
final.append(lit3)
for x in clauses:
    for y in x:
        lit2.append(y)
    final.append(atLeastOne(lit2))
    lit2.clear()
squares=conjoin(final) ######################
```

Εξηγώ κάθε εντολή παρακάτω. Στόχος αποτελεί να φέρουμε τις λογικές εκφράσεις που δίνονται σε συγκεκριμένη μορφή έτσι ώστε η τελική λογική έκφραση που θα προκύψει να είναι αληθής μόνο αν δωθεί ως αληθής μόνο μία λογική έκφραση απο αυτές που θα δωθούν εξ ' αρχής.Όπως στην συνάρτηση exactly one , δηλαδή εν τέλει να φτάσω σαυτήν την μορφή π.χ.[(¬**A** ∨ ¬**B**) ∧ (¬**A**∨ ¬**C**)∧ (¬**A**∨ ¬**D**∧ (¬**B**∨ ¬**C**)∧ (¬**B**∨ ¬**D**)∧ (¬**C**∨ ¬**D** ∧ (**A** ∨ **B** ∨ **C** ∨ **D**)]

lit=[] Ουσιαστικά στην λίστα αυτή θα βάλω τις αρνητικές μορφές κάθε λογικής έκφρασης.
for x in squares_list : ο βρόγχος αυτός χρησιμοποιείται για να κάνουμε αρνητική κάθε λογική έκφραση Το squares_list περιέχει τις λογικές μας εκφράσεις.
lit.append(¬x) βάζουμε κάθε αρνητική έκφραση στην λίστα.
clauses=list(itertools.combinations(lit,2))Ουσιαστικά εδώ γίνονται όσοι συνδυασμοί μεταξύ των λογικών εκφράσεων και δίνονται στην μεταβλητή clauses

lit2=[] Η λίστα αυτή χρησιμοποιείται για να μπαίνουν σαυτήν οι λογικές εκφράσεις ανα δύο κάθε φορά
lit3=[] Η λίστα αυτή χρησιμοποιείται για να κρατάμε μαζεμένμα σε ένα χώρο τις διαξεύξεις που δημιουργούνται
final=[] κρατείται εδώ το αποτέλεσμα απο την κλήση της συνάρτησης atLeastOne
lit3=disjoin(squares_list)Εδώ κρατάμε την μορφή (A ∨ B ∨ C ∨ D)
final.append(lit3) Το βάζουμε στην τελική λίστα
for x in clauses: Για κάθε στοιχείο χ μέσα στην clause δηλαδή για κάθε συνδυασμό καθώς κάνουμε iterate γίνονται τα παρακάτω
for y in x: για κάθε συνδυασμό (χ) παίρνουμε τα στοιχεία του το πρώτο του και το δεύτερο του .
lit2.append(y) τα βάζουμε στην λίστα lit2
final.append(atLeastOne(lit2)) μαυτήν την εντολή δίνοντας τα στοιχεία που έχει εκείνη την στιγμή η lit2 (δηλαδή του συγκεκριμένου συνδυασμού) και επιστρέφεται μία λογική έκφραση η οποία θα έχει την ιδιότητα ότι θα είναι αληθής μόνο αν δίνουμε σε μία απο τις λογικές εκφράσεις που αποτελούν την τελική λογική έκφραση την τιμή αληθής
lit2.clear() κάθε φορά που γεμίζει με δύο στοιχεία η λίστα αυτή δλδ με τα στοιχεία ενός συνδυασμού κάνουμε clear για να περάσουμε στον επόμενο συνδυασμό.
squares=conjoin(final) στο τέλος κάνουμε conjoin κάθε αποτέλεσμα για να φτάσουμε στην μορφή [(¬A ∨ ¬B) ∧ (¬A∨ ¬C)∧ (¬A∨ ¬D∧ (¬B∨ ¬C)∧ (¬B∨ ¬D)∧ (¬C∨ ¬D ∧ (A ∨ B ∨ C ∨ D)]

Αυτό τον κώδικα τον χρησιμοποιώ γενικα πολυ στο πρόγραμμα και σε αρκετές περιπτώσεις αλλάζει με βάση τις λογικές εκφράσεις που δέχεται αρχικά για παράδειγμα direction,squares_list και λοιπά

Ομοίως για το τρίτο μπουλετ κάνω την ίδια τακτική με αμέσως προηγούμενο μπούλετ απλά για τα actions που μπορεί να κάνε ιτο πακμαν.

Στο τέταρτο μπούλετ ουσιαστικά ελέγχο αν έχει δωθεί sensorModel διότι μπορέι να μην έχει δωθεί . Αν έχει δωθε΄ιτο συμπεριλαμβανω και αυτό στο τελικό αποτέλεσμα αλλιώς όχι.

Τελέυταίο μπούλετ ελέγχω αν υπάρχει successorModel και σε περίπτωση που υπάρχει ελέγχο τον χρόνο που δίνεται διότι αν ο χρόνο είναι μηδέν δεν υπάρχει το τρανζισιον που μας έφερε στο timestep t=0.Δηλαδή αν ειμαστε για t=0 δεν υπάρχει ενέργεια που έγινε προηγουμένως και φτάσαμε σαυτήν που είμαστε την t=0 διότι ουσιαστικά την t=0 ξεκίνησε το πακμαν δεν γίνεται να υπήρξε ενέργεια την t-1 για t=0.

Για την συνάρτηση checkLocationSatisfiability εκτελώ ότι μου ζητείται δηλαδή :

1. Προσθέτω στην KB το αποτέλεσμα της pacphysics axioms για τις απαραίτητες χρονικές στιγμές , δηλαδή για t= 0 και για t= 1, χωρίς την sensorModel και χρησιμοποιώντας την allLegalSuccessorAxioms.
2. Προσθέτω στην KB την θέση (χ0,y0)
3. Προσθέτω στο KB τις εκφράσεις το πακμαν κάνει την action0 και την action1

Έπειτα υλοποιώ τα model1 και model2 .Όπου το model1 εκφράζει ότι το πακμαν την χρονική στιγμή t= 1 βρίσκεται στην θέση (x1,y1) , ενώ το model2 εκφράζει ότι δεν βρίσκεται στην θέση αυτή.Οπότε στην μία περίπτωση κάνω conjoin το model1 στην βάση και καλώ την findmodel προκειμένου να ελέγξουμε αν υπάρχει ένα ικανοποιητικό μοντέλο και το αποτέλεσμα (True ή False)που προκύπτει απο την κλήση αυτή το κρατάω στην τελική λίστα που θα επιστρέψω . Στην άλλη περίπτωση κάνω το ίδιο με το model2 και ομοίως το αποτέλεσμα το κρατάω στην τελική λίστα που θα επιστραφεί.Την τελική λίστα την μετατρέπω σε tuple και το επιστρέφω.

Πρόβλημα 4

Για την συνάρτηση positionLogicPlan κάνω ομοίως ότι μου ζητείται απο την εκφώνηση .
Προσθέτω αρχικά στην KB την αρχική θέση του πακμαν την χρονική στιγμή t= 0
Μέσα στον βρόγχο για t=0 εώς t=49 το σκεπτικό είναι τα εξής: Εκτυπώνουμε κάθε

φορά την μεταβλητή t.
Αρχικά να σημειωθεί ότι πράττουμε ανάλογα με τον χρόνο t .Δηλαδή για τ= 0 πράττουμε αλλιώς και για τ> 0 πράττουμε αλλιώς.
t=0
Για t= 0 προσθέτω στην KB ότι το πακμαν μπορεί να βρίσκεται την t= 0 σε μόνο μία απο τις διαθέσιμες θέσεις . Οπότε με τις υπάρχουσες πληροφορίες κατευθείαν ελέγχω αν μπορεί να υπάρξει μια σειρά απο ενέργειες έτσι ώστε το πακμαν να βρίσκεται στην θέση στόχου που θέλουμε . Αν βρεθεί ότι είμαστε στον στόχο τότε θα επιστραφεί None διότι με το που ξεκινάμε βρισκόμαστε στον στόχο . Έπειτα προσθέτουμε στην KB την πληροφορία ότι το πακμαν μπορεί να διαλέξει μόνο μία ενέργεια προκειμένου να βρεθεί στην επόμενη διαθέσιμη θέση . Για t= 0 δεν ελέγχουμε το pacmanSuccessorAxiomSingle διότι την τ= 0 ουσιαστικά ξεκινάει το πακμαν και δεν υπήρχε προηγούμενη χρονική στιγμή μέσω της οποία φτάσαμε στην θέση που βρίσκεται την τ= 0 , δηλαδή σε περίπτωση που καλούσαμε την συνάρτηση θα υπολογίζαμε την ενέργεια που έκανε το πακμαν την τ= −1 που είναι άτοπο.
t>0
Για t>0 προσθέτουμε στην KB την πληροφορία ότι ο πακμαν μπορεί να βρίσκεται σε μία μοναδική θέση . Ωστόσο επειδή το t> 0 προσθέτουμε επίσης το αποτέλεσμα της συνάρτησης pacmanSuccessorAxiomSingle για κάθε θέση δηλαδή το τρανζισιον για κάθε θέση που εκφράζει πως βρέθηκε σε μια θέση την τωρινή χρονική στιγμή απο την αμέσως προηγούμενη στιγμή.Μαυτές τις πληροφορίες ελέγχουμε αν υπάρχει μια σειρά απο ενέργειες που εκφράζει ότι το πακμαν έφτασε μαυτόν τον τρόπο στον τελικό στόχο.Έπειτα προσθέτουμε την ενέργεια για την επόμενη θέση την επόμενη χρονική στιγμή που μπορεί να πάρει για τους μελλοντικούς ελέγχους.

Πρόβλημα 5

Η συνάρτηση foodLogicPlan είναι ουσιαστικά ίδια με την προηγούμενη συνάρτηση απλώς με μερικές διαφορές.
t=0
Για αυτήν την χρονική στιγμή κάνουμε το ίδιο με την προηγούμενη συνάρτηση, ωστόσο ο στόχος είναι διαφορετικός δηλαδή θα ελέγξουμε ως στόχο αν έχουν φαγωθεί όλα τα τα φαγητά απο κάθε θέση .Γιαυτό και ως στόχο αυτό που έκανα ουσιαστικά ήταν να εκφράσω την άρνηση για κάθε θέση που έχει φαγητό και την εισάγω στην KB. Σε περίπτωση που ισχύει αυτή η πληροφορία τότε σημαίνει ότι φτάσαμε στον τελικό στόχο.Παρολ 'αυτά σε περίπτωση που είμαστε στον στόχο την μηδενική χρονική στιγμή δεν υπάρχει κάποια σειρά απο ενέργειες καθώς τότε ξεκινάει το πακμαν , άρα πλά επιστρέφουμε το None.Επίσης για την χρονική στιγμή αυτή μετά τον έλεγχο αν είμαστε στον τελικό στόχο προσθέτουμε επίσης εκτός απο αυτά που έιχαμε και στο προηγούμενο ερώτημα την πληροφορία που αφορά το transition μεταξύ του φαγητού την χρονική στιγμή t , του φαγητού την χρονική στιγμή $t + 1$ και της ενέργειας που θα κάνει το πακμαν για να φτάσει εκεί .Για την συγκεκριμένη πληροφορία σκέφτηκα ότι , προκειμένου να υπάρχει φαγητό στην συγκεκριμένη θέση την t+1 θα πρέπει ο πακμαν να μην έχει βρεθεί στην συγκεκριμένη θέση που είχε φαγητό την προηγούμενη χρονική στιγμή . Σε περίπτωση που ο πακμαν βρέθηκε στην θέση αυτή που έχει φαγητό την χρονική στιγμή t τότε στην επόμενη χρονική στιγμή $\tau + 1$ δεν θα υπάρχει φαγητό στην συγκεκριμένη θέση. Άρα χρησιμοποίησα την συνεπαγωγή ¬(πακμαν στην θέση x,y την t) ∧ (φαγητό στην θέση x,y την t) ⇒ (φαγητό στην θέση x,y την t+1). **t>0**
Το ίδιο με τα προηγούμενα θα γίνει την t> 0 απλώς θα προσθέσουμε και τις πληροφορίες που προανέφερα.

Για τα επόμενα ερωτήματα τις βοηθητικές συναρτήσεις τις υλοποιώ κάθε φορά ξανά μέσα στις συναρτήσεις .

Πρόβλημα 6

Για το ερώτημα αυτό κάνω ακριβώς ότι μου ζητείται απο την εκφώνηση .Συγκεκριμένα σε μια λίστα (only_walls) βάζω τις θέσεις που έχουν τοίχο, ενώ σε μια λίστα (not_walls) βάζουμε τις θέσεις που δεν έχουν τοίχο και βάζω τις π ληροφορίες αυτές στην KB. Για την βοηθητική συνάρτηση Add pacphysics, action, and percept information to KB αυτό που κάνω ότι μου ζητείται . Για την συνάρτηση Find possible pacman locations with updated KB αυτό που κάνω είναι ότι εκεί που λέει "Can we prove

whether Pacman is at (x, y)? Can we prove whether Pacman is not at (x, y)? Use entails and the KB." ()χρησιμοποιώ την entails για τις θέσεις που το πακμαν μπορεί να βρεθεί και ξεχωριστά και τις θέσεις που δεν μπορεί να βρεθεί , και σε κάθε περίπτωση αν το αποτέλεσμα είναι αληθές τότε προστίθεται η έκφραση ότι βρίσκεται σαυτήν την θέση ή ότι δεν βρίσκεται σαυτην την θέση το πακμαν.Για την "If there exists a satisfying assignment where Pacman is at (x, y) at time t, add (x, y) to possible_locations" χρησιμοποιώ την findModel (γιατί λέει satisfying assignment άρα αναφέρεται σε ικανοποιητικό μοντέλο)και σε περίπτωση που δεν επιστρέψει false τότε προστίθεται στην λίστα των πιθανών θέσεων η συγκεκριμένη θέση .Έπειτα απλώς εκτελώ ότι λέει η εκφώνηση να κάνουμε.

Πρόβλημα 7

Στην συνάρτηση αυτή αρχικά αναφέρω ότι αν η known_map στην συγκεκριμένη θέση έχει 1 δηλαδή υπάρχει τοίχος εισάγουμε στην KB ότι υπάρχει τοιχός αλλιώς αν έχει μηδέν εισάγουμε στην KB ότι δεν υπάρχει τοίχος στην συγκεκριμένη θέση .Επίσης στην συνάρτηση Find provable wall locations with updated KBκάνω ότι μου ζητείται .

Πρόβλημα 8

Τέλος εκτελώ ότι μου λένε οι βοηθητικές συναρτήσεις απλώς στην αρχή κάνω και το εξής : για την χρονική θέση t= 0 το πακμαν βρίσκεται στην θέση αυτή άρα ενημερώνω την known_map με μηδέν στην συκγεκριμένη θέση διότι το πακμαν βρίσκεται στην θέση αυτή.Επίσης εισάγω στην KB ότι δεν υπάρχει τοίχος στην θέση αυτή .Στην συνέχεια και γενικότερα στο ερώτημα απλώς κάνω ότι μου ζητείται .