תרגיל מסכם אימות פורמלי וסינטזה

תאריך הגשה:

מגישים:

נועם דיאמנט 208520262

אורה רחל ווצלר 212058689

חלק ראשון:

1. נגדיר את ה FDS של לוח הסוקובן:

D = {V, , , J, C}

V – יש לנו n\*m משתנים שהם תאים בלוח, כאשר כל תא בלוח מהווה משתנה יחיד. נתאר את הלוח כמערך חד מימדי עם n\*m תאים. כל משתנה של תא בלוח יראה: board[i], יהיה התא בשורה (i/n) ובעמודה (i%n) כל משתנה בלוח לקבל אחד מבין הערכים הבאים:

{@, +, $, \*, #, ., -}[[1]](#footnote-1)

בנוסף, יש לנו משתנה המתאר את תנועת השומר:

- מצב התחלתי של המערכת – הלוח הנתון בתחילת כל משחק

*- המעברים האפשריים המערכת נתונים לפי המצבים הבאים:*

פונקציית המעברים של תנועת השומר היא לא דטרמיניסטית, ונתונה:

next(direction) := {l, u, r, d};

פונקציית המעברים של התאים בלוח:

הפונקציה נתונה עבור תא כלשהו. כאשר נריץ את הפייתון, הוא יציב את התא הספציפי בטבלה. כבר כאשר נריץ את הפייתון, נקבל שני פונקציות הצבה שונים.

אם הפייתון מזהה כי המצב ההתחלתי בלוח של תא [i][j] כלשהו הוא קיר ('#'), אזי הפייתון יציב לו בתור פונקציית מעבר את rho\_i, כיוון שהתא לעולם לא ישתנה:

next(SokobanBoard[i][j]) := hashtag;

אחרת, הפייתון יציב לו את פונקציית המעברים הבאה, כאשר תוך כדי הצבה, עבור כל תא שנבדק כאן ([i][j], [i+1][j], [i+2][j], [i][j+1]…) , הפייתון מראש יבדוק האם מדובר בתא שהבסיס שלו הוא רצפה, ולכן בכל תנאי הוא יבדוק ויציב רק אחד מהאפשרויות המתאימות מתוך הקבוצה

floor\_states = {-,@,$},

אחרת אם הבסיס של התא הוא מטרה, הפייתון יזהה זאת ויציב אפשרות מתאימה מתוך הקבוצה  
goal\_states = {.,+,\*}

בנוסף, אם הפייתון יזהה שתא סמוך, או אם רלוונטי- תא בשכנות מדרגה 2, הוא קיר- הוא לא יכתוב את התנאי ב nuXmv, מכיוון שזאת בדיקה מיותרת שתמיד תניב תוצאה שלילית.

    next(board[i][j]) :=

        case

            -- case keeper

            SokobanBoard[i][j] = (at | plus) & direction = l & SokobanBoard[i][j - 1] = (dash | dot) : (dash | dot);

            SokobanBoard[i][j] =  (at | plus) & direction = l & SokobanBoard[i][j-1] = (dollar | asterisk) & SokobanBoard[i][j-2] = (dash | dot) : (dash | dot);

            SokobanBoard[i][j] =  (at | plus) & direction = r & SokobanBoard[i][j+1] = (dash | dot) : (dash | dot);

            SokobanBoard[i][j] =  (at | plus) & direction = r & SokobanBoard[i][j+1] = (dollar | asterisk) & SokobanBoard[i][j+2] = (dash | dot) : (dash | dot);

            SokobanBoard[i][j] =  (at | plus) & direction = d & SokobanBoard[i+1][j] = (dash | dot) : (dash | dot);

            SokobanBoard[i][j] =  (at | plus) & direction = d & SokobanBoard[i+1][j] = (dollar | asterisk) & SokobanBoard[i+2][j] = (dash | dot) : (dash | dot);

SokobanBoard[i][j] =  (at | plus) & direction = u & SokobanBoard[i-1][j] = (dash | dot) : (dash | dot);

            SokobanBoard[i][j] =  (at | plus) & direction = u & SokobanBoard[i-1][j] = (dollar | asterisk) & SokobanBoard[i-2][j] = (dash | dot) : (dash | dot);

            -- case box

            SokobanBoard[i][j] = (dollar | asterisk) & direction = l & SokobanBoard[i][j-1] = (dash | dot) & SokobanBoard[i][j+1] =  (at | plus):  (at | plus);

            SokobanBoard[i][j] = (dollar | asterisk) & direction = r & SokobanBoard[i][j+1] = (dash | dot) & SokobanBoard[i][j-1] =  (at | plus):  (at | plus);

SokobanBoard[i][j] = (dollar | asterisk) & direction = u & SokobanBoard[i-1][j] = (dash | dot) & SokobanBoard[i+1][j] =  (at | plus):  (at | plus);

            SokobanBoard[i][j] = (dollar | asterisk) & direction = d & SokobanBoard[i+1][j] = (dash | dot) & SokobanBoard[i+1][j] =  (at | plus):  (at | plus);

            -- case floor

            SokobanBoard[i][j] = (dash | dot) & direction = l & SokobanBoard[i][j+1] =  (at | plus) :  (at | plus);

SokobanBoard[i][j] = (dash | dot) & direction = l & SokobanBoard[i][j+1] = (dollar | asterisk) & SokobanBoard[i][j+2] =  (at | plus) : (dollar | asterisk);

            SokobanBoard[i][j] = (dash | dot) & direction = u & SokobanBoard[i+1][j] =  (at | plus) :  (at | plus);

            SokobanBoard[i][j] = (dash | dot) & direction = u & SokobanBoard[i+1][j] = (dollar | asterisk) & SokobanBoard[i+2][j] =  (at | plus) : (dollar | asterisk);

            SokobanBoard[i][j] = (dash | dot) & direction = r & SokobanBoard[i][j-1] =  (at | plus) :  (at | plus);

            SokobanBoard[i][j] = (dash | dot) & direction = r & SokobanBoard[i][j-1] = (dollar | asterisk) & SokobanBoard[i][j-2] =  (at | plus) : (dollar | asterisk);

            SokobanBoard[i][j] = (dash | dot) & direction = d & SokobanBoard[i-1][j] =  (at | plus) :  (at | plus);

            SokobanBoard[i][j] = (dash | dot) & direction = d & SokobanBoard[i-1][j] = (dollar | asterisk) & SokobanBoard[i-2][j] =  (at | plus) : (dollar | asterisk);

            -- rho\_i

            TRUE:

                case

                    SokobanBoard[i][j] = (dash | dot) : (dash | dot);

                    SokobanBoard[i][j] =  (at | plus) :  (at | plus);

                    SokobanBoard[i][j] = (dollar | asterisk) : (dollar | asterisk);

                    -- to avoid nuXmv error. SHOULD NOT HAPPEN!!

                    TRUE : hashtag;

                esac;

        esac;

*J = – המשחק אמור להסתיים לאחר מספר סופי של צעדים, ולכן אין מצב שאמור לקרות אינסוף פעמים*

*C= – באופן דומה לדרישת הצדק*

*שאלה 2)*

*אנו רוצים שתהיה קיימת ריצה, כך שבסוף יש מצב שבו אין אף משתנה מסוג cell שהוא עדיין goal.*

*CTL EF !(*

1. לתשומת לב: מכיוון שחלק מהתווים הם תווים שמורים ב nuXmv, השתמשנו ב nuXmv בקבוצה הבאה: {dollar, asterisk, hashtag, at, plus, dot, dash}. לשם הנוחות, בקובץ הזה נשתמש בתווים עצמם. [↑](#footnote-ref-1)