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:

next(direction) :=

        case

            direction = u : d;

            direction = d : l;

            direction = l : r;

            direction = r : u;

        esac;

פונקציית המעברים של התאים בלוח:

הפונקציה נתונה עבור תא כלשהו. כאשר נריץ את הפייתון, הוא יציב את התא הספציפי בטבלה.[[2]](#footnote-2)

next(board[i]) :=

            case

                -- Any tile that is a wall always stays the same. Therefore it's : Rho\_i

                board[i] = hashtag : hashtag; -- This covers also tiles that are on the edge of the board

                TRUE:

                    case

                        -- If this tile is the guard, check if it can move towards the direction it needs.

                        -- Either the next tile is a floor (or goal), or that it has where to push the box (the tile after the box is floor or goal)

                        -- Otherwise - don't move

                        board[i] = (at | plus) :

                            case

                                (direction = u &

                                ((board[i-1\*row\_len] = (dash | dot)) |

                                ((board[i-1\*row\_len] = (dollar | asterisk)) & (board[i-2\*row\_len] = (dash | dot))))) |

                                (direction = d &

                                ((board[i+1\*row\_len] = (dash | dot)) |

                                ((board[i+1\*row\_len] = (dollar | asterisk)) & (board[i+2\*row\_len] = (dash | dot))))) |

                                (direction = l &

                                ((board[i-1] = (dash | dot)) |

                                ((board[i-1] = (dollar | asterisk)) & (board[i-2] = (dash | dot))))) |

                                (direction = r &

                                ((board[i+1] = (dash | dot)) |

                                ((board[i+1] = (dollar | asterisk)) & (board[i+2] = (dash | dot))))) :

                                    case

                                        board[i] = at : dash;

                                        board[i] = plus : dot;

                                    esac;

                                -- Rho\_i

                                TRUE :

                                    case

                                        board[i] = at : at;

                                        board[i] = plus : plus;

                                    esac;

                            esac;

                        -- In case this tile is a box, it can move only if the previous tile

                        -- (that coresponds to the direction) is the guard, and there is place to push the box

                        board[i] = (dollar | asterisk) :

                            case

                                (direction = u &

                                ((board[i+1\*row\_len] = (at | plus)) & (board[i-1\*row\_len] = (dash | dot)))) |

                                (direction = d &

                                ((board[i-1\*row\_len] = (at | plus)) & (board[i+1\*row\_len] = (dash | dot)))) |

                                (direction = l &

                                ((board[i+1] = (at | plus)) & (board[i-1] = (dash | dot)))) |

                                (direction = r &

                                ((board[i-1] = (at | plus)) & (board[i+1] = (dash | dot)))) :

                                    case:

                                        board[i] = dollar : at;

                                        board[i] = asterisk : plus;

                                    esac;

                                -- Rho\_i

                                TRUE :

                                    case

                                        board[i] = dollar : dollar;

                                        board[i] = asterisk : asterisk;

                                    esac;

                            esac;

                        -- In this case, the floor will change if the guard tries to go on it, or if the guard tries to push a box on to it

                        board[i] = (dash | dot) :

                            case

                                -- If the keeper wants to walk to this tile

                                ((direction = u &

                                ((board[i+1\*row\_len] = (at | plus)))) |

                                (direction = d &

                                ((board[i-1\*row\_len] = (at | plus)))) |

                                (direction = l &

                                ((board[i+1] = (at | plus)))) |

                                (direction = r &

                                ((board[i-1] = (at | plus))))) :

                                    case

                                        board[i] = dash : at;

                                        board[i] = dot : plus;

                                    esac;

                                -- If the keeper wants to push the box to this tile

                                ((direction = u &

                                ((board[i+1\*row\_len] = (dollar | asterisk)) & (board[i+2\*row\_len] = (at | plus)))) |

                                (direction = d &

                                ((board[i-1\*row\_len] = (dollar | asterisk)) & (board[i-2\*row\_len] = (at | plus)))) |

                                (direction = l &

                                ((board[i+1] = (dollar | asterisk)) & (board[i+2] = (at | plus)))) |

                                (direction = r &

                                ((board[i-1] = (dollar | asterisk)) & (board[i-2] = (at | plus))))) :

                                    case

                                        board[i] = dash : dollar;

                                        board[i] = dot : asterisk;

                                    esac;

                                -- Rho\_i

                                TRUE :

                                    case

                                        board[i] = dash : dash;

                                        board[i] = dot : dot;

                                    esac;

                            esac;

                    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)
2. נשים לב כי התאים בקצוות הם כבר מאותחלים להיות (#), ולכן לא נגיע למצב בעייתי שאנחנו בודקים את התאים שלידו (שלא קיימים).

   לעומת זאת, בתאים שאנו בודקים מה נמצא במרחק 2 תאים מהם- אנחנו קודם בודקים שהתא הצמוד אליהם הוא קופסא. לכן אם התא הזה הוא לא קופסא אלא קיר- בפרט לא נבדוק את התא במרחק 2 מהם, ולא ניכנס למצב בעייתי של בדיקת תא שלא קיים. [↑](#footnote-ref-2)