בס"ד

אקדמית תל אביב – יפו

בית ספר למדעי המחשב

תרגום של דיאגרמות מיצוב לשפת NuSmv, וביצוע בדיקת נכונות למודל

עבודה לפרויקט גמר

תואר שני במדעי המחשב

מגיש: בן ציון סויד, ת"ז: 305584542

מנחה: פרופסור שמואל טישברוביץ

תאריך הגשה: פברואר 2022

Abstract

The digital world is expanding. Every day, hundreds of new systems created around the world, and the need for validation is increasing due to the high value of creating those systems. When one wants to create a new software system, he first need to create a specification and model the system, only then, the development phase is starting. The later that you find a bug in life cycle of a project, the more it cost to fix it. As a result, the popularity of a validation tools are increasing.
One of the most common tool to model a system is *Statechart*.
This languages is very popular with designers of safety-critical systems, not only because of the intuitive graphical representation, but also because that there is a verity of tools that have built in simulation capabilities. Those methods give the designers the opportunity to model their system and to get intuitive observation about the way that their specifications runs. But still there is a problem, even if you model your system with statechart, there is not any guaranty that you model your system the right way, and safety critical systems need more formal approach to validate the correctness of a statechart-design.

One of the methods to prove that given system meets its specification is method that called *model checking [4]*. There is a verity of model checkers tools, one of them is NuSmv, a language that extend the old SMV symbolic model checker, that use CTL (computation tree logic), a branching time logic formula to validate the design. One can model his design in NuSmv language and test it by CTL formula that run above the NuSmv model.

In this paper, I suggest a solution for this complicate process, by creating a complete process from the statechart design throw the model checker, and by doing so, basically overcome the problems with the old and Sisyphean workflow.

The solution I give in paper divide into 4 steps:

1. Creating a statechart diagram
2. Convert the diagram into SCXML format
3. Convert the SCXML into NuSmv language
4. Run a CTL model checker to test the design

Every part of the steps gets it own solution. I searched and found the right tool for creating and drawing a statechart diagram (1), part of the condition for choosing the right tool is that it has converting to SCXML options (2). The SCXML format is an establish format by the W3C organization for representing a statechart by XML. So, I created a script written in Python that convert the SCXML format into the correlate NuSmv module (3), then I run CTL formula and validate the design with the NuSmv capabilities (4).

All these steps (beside the first one that is the creating of the diagram) is done automatically and save the designer a lot of work, and as a result, he can validate the safety and liveness of his system with ease.

1. מבוא

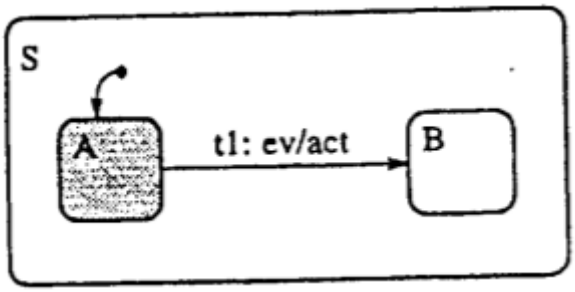
עולם מערכות המידע מתרחב, תוכנות ופרויקטים רבים נפתחים דבר יום ביומו. תכנון פרויקט תוכנה הינו משימה לא פשוטה, אך זהו שלב קריטי בבניה נכונה של התוצר שאנחנו רוצים לעשות. תכנון לקוי יתבטא בהוצאות כספיות רבות בהמשך שלבי הפרויקט, ובדחיית לו"ז ארוכה.
ישנם כלים רבים בשוק אשר תפקידם לסייע בתהליך אפיון הדרישות והגדרת ההתנהגות של התוכנה הרצויה, ובדיקת נכונותה עוד בטרם הושקע בפרויקט שעות עבודה רבות. אחד הכלים הנפוצים לצורך אפיון מערכות מידע הינו בניית דיאגרמות מיצוב (state chart). דיאגרמות מיצוב הן שפה עשירה של צורות וסמלים אשר מאפשרת לך לתאר במדויק את התנהגות המערכת שלך. ישנן כלים רבים ליצירה של דיאגרמות מסוג זה, וחלק מהכלים גם נבחנים כחלק מעבודת הגמר. יצירת דיאגרמת מיצוב למערכת זהו שלב קריטי בתהליך האפיון ותכנון של מערכות, אך שום דבר לא מבטיח כי הדיאגרמה שבנית באמת נכונה ומתארת את המערכת בצורה מדויק כפי שנדרש באפיון שלה. מתוך כך, גם אם בנו דיאגרמת מיצוב לתיאור המערכת, חסרה לנו רמת סמך גבוהה על מנת לוודא כי הדיאגרמה מתארת את המערכת שברצוני לאפיין, ולכן גם שלב האימות והחיות שברצוני לבדוק על המערכת לא בהכרח יהיה מדויק.

את הפער הזה אני מנסה לפתור בעבודת גמר. בעבודה שלי, אני מאפשר לעבור תהליך שלם, החל ביצירת ובציור דיאגרמת המיצוב (אשר משמשת לצורך פירוט הדרישות וההתנהגות של המוצר), עובר לפורמט SCXML ואז מומר לקוד NuSmv אשר נמנה עם קבוצת הכלים שמבצעים בדיקת נכונות למודלים, ובכך לאפשר למאפיין המערכת יכול לאמת ולוודא כי המערכת שהוא אפיין אכן נכונה, והדיאגרמה מתארת אותה בצורה שלמה ומדויקת.

2. הגדרות

STATE CHART

דיאגרמת מיצוב הינה שיטה אחת מתוך דיאגרמות UML [**Error! Reference source not found.**], אשר יעודה הינו למדל את התנהגותם של מערכות ולאפיין את הדרישות שלהם. הדיאגרמה משתמשת במצבים ואירועים על מנת לתאר את שטף התהליכים המתבצעים במערכת. כל מצב מתאר תנאי מסוים במערכת, מצב קיום כלשהו, והוא יכול להיות מושפע מאירועים, פנימיים או חיצוניים, שקורים במערכת. בכל אירוע המצב יכול להישאר אותו דבר או להשתנות, ובנוסף, לבצע פעולה מסוימת.
דוגמא לדיאגרמת מיצוב כלשהי:



באיור 1 אנו רואים דיאגרמה בסיסית המתארת 2 מצבים, ומעבר אחד (*transaction*). המצב A הינו המצב ההתחלתי (על פי הסימון המוסכם המפנה אליו בכניסה למצב S המכיל את תת המצבים), והמעבר בינו לבין B מוגדר על ידי המעבר t1 , המורכב מ ev – אירוע כלשהו שיעלה במערכת, ו act – הפעולה שתתבצע בזמן שאירוע ev עלה במערכת. לאחר ש t1 תקרה, המצב בו המערכת נמצאת יהיה B.
את הסמנטיקה של דיאגרמות המיצוב אנחנו נבסס על הסמנטיקה שהגדיר דוד הראל ואמנון נעמד במאמרם על הנושא [**Error! Reference source not found.**] .

SCXML

בשמו המלא: state chart xml, הינו סטנדרט לייצוג דיאגרמות מיצוב סופיות המבוססות על הייצוג החזותי שיצר והגדיר דוד הראל [**Error! Reference source not found.**], על ידי שימוש בשפת XML. הסטנדרט נקבע על ידי W3C , ארגון האחראי על קביעת סטנדרטים ופרוטוקולים באינטרנט, והינו הסטנדרט הרשמי שלהם לשימוש לייצוג דיאגרמות מיצוב [**Error! Reference source not found.**].

Model checkers

"בדיקת מודל" הינה שיטה או מתודה לבדיקה האם מודל של מערכת המתואר באמצעות מכונת מצבים סופית, עומד באפיון שהוגדרה לו. אפשר לקרוא לבדיקה הזאת "בדיקת נכונות" עבור המודל. השיטה משמשת בעיקר לבדיקת מערכות תוכנה או חומרה אשר קיים תנאי קריטי לחיות המערכת (מניעת מצב של deadlock) או תנאי הכרחי של בטיחות (דהיינו מניעת מצב של קריסת המערכת). המודל מיוצג בצורת פורמולה שאותה אפשר להמיר לקוד על פי התחביר של הכלי שבחרת להשתמש ב"בדיקת מודל" (שנכתב בשפת תכנות). תחום מרכזי בשיטות של בדיקת מודלים, הינו השיטה המבוססת על אפיון נכתב בצורה של LTL/CTL formula (פורמולות שמנוסחות בצורה התלויה בזמן

[4]). ישנם כלים רבים לבדיקת מודלים, אנחנו בפרויקט נתרכז בכלי: NuSmv.

NuSmv

כלי לבדיקת נכונות מודלים סמנטיים [8]. הכלי הינו הרחבה של הכלי SMV [6]. זהו כלי המבוסס על ה BDD-BASED (דיאגרמת החלטה בינארית, מבנה מידע לייצוג פונקציות בינאריות בצורה סימבולית [7]). אנחנו בפרויקט נשתמש בגרסה 2.6 של NuSmv. גרסה 2 של הספרייה מבוססת על הגרסה הראשונה (NuSmv כפי שתואר למעלה), אך מרחיבה אותה לכיוונים שונים, למשל השילוב של טכניקת בדיקה SAT-BASED (SAT SOLVER, בדיקה של האם סט של משפטים בינאריים יכולה להיות מסופקת [9]).

CTL

מודל לוגי דמוי עץ המייצר הסתעפויות בזמן בו העתיד איננו בטוח ויש כמה נתיבים שונים שכל אחד מהם יכול להתממש. המודל הזה משמש בד"כ בשיטות "מודלי בדיקה" (ר"ע למעלה) על מנת לאפיין מערכות לוגיות מסוימות ולבצע עליהם אימות (*verification*). התכונות שהמודל הזה בודק הן בעיקר בטיחות (שלא קורים דברים אסורים במודל, למשל חלוקה באפס במודל) וחיות (המודל ממשיך לעבוד כל הזמן ולא נתקע ב DEADLOCK). CTL הינו חלק ממחלקת המודלים לייצוג "משפטים" התלויים בזמן, ביניהם גם LTL (בו הנתיב\המסלול הוא קבוע ויש נתיב אחד בו אפשר לעבור במודל ועתיד ידוע מראש). ל CTL קיים מימוש ב NuSmv ואנחנו נשתמש בו בעבודת גמר על מנת לתקף (*validation*) את ההמרה שיצרנו לדיאגרמה.

3. תיאור מטרת העבודה וחשיבותה

מטרת הפרויקט הינה לייצר תהליך שלם משלב התכנון ואפיון המערכת, עד היכולת לבדוק את נכונות התכנון. כפי שתיארנו בסעיפים הקודמים, במערכות תגובתיות (תוכנה או חומרה), אשר ישנם אילוצים קריטיים לבדיקת נכונות המערכת (מניעת deadlock או מניעה מצב שבו המערכת קורסת), משתמשים לרוב בשיטת ה state diagram בשביל לבצע תכנון ואפיון. אך על מנת לבצע בדיקת נכונות לכל תיאור של דיאגרמות מיצוב, צריך להשתמש בכלי ה model checkers אשר מקבלים כקלט סט של פקודות והגדרות על בסיס שפת התכנות של הכלי שבחרת להשתמש בו.
בפרויקט שלנו אנחנו נאפשר לעבור מתכנון על בסיס הדיאגרמות מיצוב, לבדיקת נכונות המבוססת NuSmv בגרסה 2.6 שלה, בצורה אוטומטית, על מנת לייצר תהליך שלם מקצה לקצה, בצורה קלה ונוחה. בכך אנחנו נעזור בווידוא כי התכנון מכסה מקרי קצה, והמפרט המוצע למערכת עומד בתנאים ובאילוצים שאנחנו מצפים, ובכך נוכל להתקדם למימוש בלב שקט יותר.

4. סקירה ספרותית ועבודות קשורות

כחלק מעבודת הגמר, סקרתי מאמרים ועבודות שונות אשר יכולים להיות קשורים לפרויקט גמר שלי. מכל מאמר ניסיתי לזקק את התובנות שישמשו אותי בעבודה שלי, ומה ההבדלים והפערים בינם לבין העבודה שלי.
במאמר הזה [11] לדוגמא, משתמשים ב NuSmv ככלי למציאת מקרי בוחן למערכות המבוססות דיאגרמות מיצוב. כותבי המאמר יצרו שיטה ליצירה של מקרי בוחן אוטומטיים על בסיס אפיון של מערכת על ידי NuSmv, אך תחת הנחות מסוימות:

1. קיים אפיון מלא של דיאגרמת המיצוב על ידי מודל הנקרא במאמר בשם: "deterministic labeled transition system", מודל שמתאר את התנהגות הדיאגרמה בצורה של:
   $M_L = (S_L, s_{init}, R_L)$, כאשר S_L מבטא את כל המצבים במערכת, s_{init} מבטא את כל המצבים ההתחלתיים של המערכת, ו R_L מתאר את המעברים בין המצבים.
2. קיימת תוכנית SMV אשר מתארת את התנהגות המערכת.
3. המאמר מניח שלא יכול להיות אירוע בחלק של ה – action במעבר בין מצבים – הנחה שאיננו יכולים לקחת בעבודה הזאת.

המאמר מביא דוגמאות של דיאגרמות ומריץ עליהם את התהליך שהם יוצרים במהלך המאמר על מנת להוציא מקרי בוחן על המערכת מתוך דוגמא נגדית המבוססת הרצה של NuSmv.
המאמר מביא את הדוגמאות תוך הנחה שקיימות להן תוכניות NuSmv מוכנות

או שהוא יוצר ברקע (ללא התייחסות במאמר לאיך הוא יצר את התוכניות) תוכניות לדיאגרמות ועליהן מריץ את המנגנון שלו. בדוגמא היחידה הקיימת במאמר, ניתן לראות שהצורה שבה הוא מתייחס למעברים בין המצבים דומה מאוד לצורה שבה אני בעבודה שלי מתייחס אליהם:

```
1   MODULE main
2   VAR
3     pc : {S1, S2, S3};
4     ev : {ev1, ev2, ev3, ev4};
5
6   ASSIGN
7     init(pc) := S1;
8     next(pc) := case
9       pc = S1 & ev = ev1 : S2;
10      pc = S2 & ev = ev2 : S3;
11      pc = S2 & ev = ev4 : S1;
12      pc = S3 & ev = ev3 : S2;
13      1 : pc;
14    esac;
```

Figure 2. SMV program for Fig. 1

כך גם ההמרה שלי מאפיינת את תוכנית ה NuSmv , בצורה מאוד דומה, ועם התייחסות למצב הנוכחי, והאירוע שגורם למעבר.

במאמר אחר [12], בחנו המרה של דיאגרמת מיצוב ל Promela , שפת הקלט של מערכת האימות הנקראת SPIN . במאמר מתייחסים למאמר של דוד הראל [2] כסמנטיקה של דיאגרמות המיצוב, אך מגבילים אותה ויוצרים תת-שפה תחת ההנחות הבאות:

1. הביטויים במעברים בין המצבים יכולים להיות בוליאניים בלבד
2. ה action של מעברים יכול לייצר אירועים בלבד.

בנוסף, המאמר מדגיש כי הוא עובד תחת הנחה נוספת, שאין מעברים בין רמות (למשל כמו בדוגמא 1 למטה), ולכן מביא מאמר נוסף [13] בו הוא מתאר מעבר בין דיאגרמות מיצוב עם מעברים בין רמות ל EHA, פורמט של דיאגרמת מיצוב ללא מעברים כאלה. תחת ההנחות וההמרה ל EHA, המאמר מייצר תהליך המרה לשפת ה Promela. גם במאמר הזה, לא לוקחים בחשבון היסטוריה (ראו להלן גם התייחסות להיסטוריה בעבודת גמר שלי), שינויים במידע ובעיות תזמון מכיוון שהם התמקדו בפתרון לבעיה הבסיסית שאחרי זה ניתן להרחיב אותה. תהליך ההמרה שלהם כוללת יצירת "קונפיגורציה" (מצב המצבים הנוכחי במערכת) ובשילוב עם אירועים, מגדירים "status" וברכיב הזה משתמשים להגדרה של כלל ההמרה לשפה. בנוסף, המאמר עובד על מערכת סגורה ללא אירועים חיצוניים, ואם יש אירועים חיצוניים הם ממירים (ידנית) את הדיאגרמה לדיאגרמה סגורה על ידי יצירת דיאגרמות נוספות שמזריקות את האירועים החיצוניים ובכך שומרים על מערכת סגורה.

כמו בעבודה שלי, גם הם התמודדו עם הקושי בהמרה של דיאגרמה מקבילית, שכן גם ב Promela אין פתרון טבעי לכך.
המאמר מציע 2 פתרונות:

1. יצירה של Processes ב Promela ולחלק ביניהם את הדיאגרמות המקבילות – פתרון שבעייתי אצלנו מכיוון ש NuSmv איננו תומך כבר ב process.
2. ביצוע של flattening לדיאגרמה ויצירה של דיאגרמה סדרתית מקבילה – הפתרון הזה אפשרי, אך מסובך מאוד כשיש יותר מ 2 מצבים בכל חלק של המקביליות.

גם בפרויקט גמר הזה נתקלנו בבעיה עם המקביליות ללא פתרון מספק.

מאמר מאוד מעניין ביחסו לעבודה שלי, ***Modular Translation of Statecharts to SMV*** [14] , תיארו תהליך ארוך של המרה הדומה למטרת הפרויקט גמר שלי. המאמר עוסק בכלי הנקרא STP, כלי תרגום אוטומטי של דיאגרמות מיצוב של כלי ה STATEMATE [2] ל SMV , כלי שכותבי המאמר יצרו במיוחד לחברת ג'נרל מוטורס [חברה מסחרית אמריקאית לייצור מכוניות, ע"ע] בשנת 1998.
כותבי המאמר דנים בסוגי הדיאגרמות השונות, ביניהם דיאגרמות OR (הדיאגרמות הרגילות בהן המעבר בין המצבים הוא סידרתי) ודיאגרמות AND (דיאגרמות מקביליות), ותהליך העבודה שלהם להמרה ל SMV עובר על כל דיאגרמה ומפרק אותה לתת-דיאגרמות ומטפל בכל דיאגרמות בנפרד.
תיאור תהליך ההמרה שלהם (ב high level):
הקוד SMV שהם מייצרים מחולק ל 3 סוגי modules:

1. Main – עבור ה module הראשי
2. Module עבור כל תת-דיאגרמה של הדיאגרמה הראשית
3. Monitor module – המשמש לטיפול באירועים ותנאים

כל האירועים והתנאים מוגדרים ב main מכיוון שהם גלובליים במערכת. על מנת לפתור את הבעיה ש SMV אוכף סתירה בהשמה של משתנים (אירועים או תנאים) אז כל השמה של תנאי או אירוע מטופל ב monitor module משלו שרק הוא יכול לשנות אותו. כל מודול אחר שרוצה לשנות את פרמטר, מעלה דגל (שמוזרק לmonitor module הרלוונטי) וכך ה monitor module יודע לשנות את הערך של אותו פרמטר.

אחד האילוצים של SMV זה שכל מודול שמוגדר בה עובד בצורה מקבילית לכל שאר המודולים, ומאתחל את עצמו באותו זמן כמו כולם, כך שטכנית כל התת-דיאגרמות השונות פעילות בכל זמן נתון. על מנת להתגבר על בעיה זאת, יצרו פרמטר מאוד חשוב שיש לכל מודול: *active* . הפרמטר הזה נותן אינדיקציה האם התת-דיאגרמה פעילה או לא ומכתיב האם מתבצע מעברים בתוך הדיאגרמה הרלוונטית שלו, **אך עדיין בפועל כל דיאגרמה פעילה, והערך של ה- state שלה יהיה המצב האחרון בו הדיאגרמה הייתה.**

הפתרון שלהם לבעיית הדיאגרמות המקבילות (AND-state diagram) הוא פשוט להתייחס אליהם כדיאגרמות של OR רגילות שרצות במקביל (מעצם זה שהכל ממומש במודולים שונים ולכן גם ככה הכל רץ כל הזמן), עם ערך active פעיל בכל הדיאגרמות בו זמנית ($active_m$ = true לכל m = תת דיאגרמה מקבילית).
את המעברים בין המצבים הם יוצרים באמצעות TRANS Constraint , המגדיר את כל התנאים האפשריים לשינוי אירוע\תנאי. דהיינו, מעבר על כלל הדיאגרמה (ותתי הדיאגרמות שלה), מיפוי של כלל המקומות המשפיעים על כל אירוע או תנאי ויצירת TRANS Constraint מותאם לכל אירוע\תנאי.
המאמר הנ"ל מכתיב כמה אילוצים לכלי שלהם:

1. לא מאפשרים מעבר בין רמות (inter level transaction) – *אילוץ שלא קיים בכלי המרה שלי*
2. מערכת סגורה בלבד (ללא אירועים חיצוניים) – *אילוץ שלא קיים בכלי המרה שלי*
3. כל OR-state diagram יכול להכיל רק תת-דיאגרמה אחת (בגלל שהכל מומר בסוף ל OR-state diagram אז בעצם לכל דיאגרמה יש תת דיאגרמה אחת בלבד, אלא אם כן זה מקבילי ואז צריך מינימום 2) - *אילוץ שלא קיים בכלי המרה שלי.*
4. תנאים יכולים להיות בוליאניים בלבד (אין תמיכה בפרמטרים integer-ים – למשל מעבר שתלוי ב $x < 10$) - *אילוץ שלא קיים בכלי המרה שלי*
5. קיים רק action אחד לכל מעבר (*transition*) – *אילוץ שלא קיים בכלי המרה שלי*
6. אין התייחסות למימוש של היסטוריה

ההשמה של האירועים שלהם דומה להשמה של האירועים בפרויקט שלי מבחינת מבנה קוד.

השיטה שבה מתבצע התהליך המרה שלהם נראית יותר ספציפית לדיאגרמות הקיימות ב STATEMATE ומאוד הדוקות אליהן, כך שמצד אחד קיימת תמיכה בדיאגרמות מקביליות מצד אחד, אך סובלת מבעיות מהותיות מצד שני.

ההמרה שאני ביצעתי בפרויקט גמר שלי היא יותר נרחבת ומאפשרת מבחינת אופרטורים ותנאים, לעומת ההמרה שלהם שבהנחות עבודה יש אילוצים לא טבעיים לשפה.

ההמרה שלהם גם מאוד מושפעת מגודל הדיאגרמה וכמות התנאים והאירועים בה ולכן בדיאגרמות ב scale גבוה, הקוד SMV שיצא בסוף יהיה גדול מאוד ולא קריא. הם יוצרים module עבור כל תנאי וכל אירוע בנפרד, וגם משתנים ייחודיים לכל תנאי ואירוע לפי מספר התת-דיאגרמות במערכת, כך שכמות המודולים במערכת שלהם היא **לפחות** C*E*S (כאשר – S כמות התת-דיאגרמות הקיימות, כולל הראשית, E – כמות האירועים השונים במערכת, C –

כמות התנאים השונים במערכת) ומספר המשתנים הוא לפחות באותו גודל גם כן. מה שגורר קוד SMV מאוד מורכב לקריאה, ארוך ומסורבל (במאמר ישנה דוגמא בסוף של ההמרה שאפשר לבחון אותה), בעוד בהמרה שלי, קיים module אחד בלבד, מספר המשתנים שווה בדיוק למספר התנאים במערכת, ועוד 2 משתנים, אחד לביטוי של המצבים ואחד לאירועים.

בגלל הצורה שבה הם בונים את קוד ה SMV יוצא כי מורכב מאוד לכתוב specification בצורת CTL בשביל לבצע ולידציה לקוד שלהם, מכיוון שקיימים מודולים רבים ומשתנים רבים והכתיבה איננה טבעית, לעומת הקוד שלי שיוצר FLOW אחיד לכל אורך הקוד (למשל, state אחד פעיל לכל אורך התוכנית)

לסיכום המאמר הזה [14], אציין כי מצד אחד הוא נותן מענה מדויק יותר לדיאגרמות שיוצאות מכלי ה STATEMATE מה שנותן להם אפשרות להמיר בצורה יותר קלה את הפלט (בגלל החיבור ההדוק ל STATEMATE) וההמרה שלהם מאוד מחקרית וגם מתארת את הדיאגרמה בצורה קצת יותר מדויקת מהעבודה שלי, אך לעומת זאת העבודה שלי יותר נרחבת ומקלה (מבחינת אילוצים), תומכת ביותר אפשרויות והקלט שלה היא SCXML שזאת שפה גנרית לייצוג דיאגרמות מיצוב ולא דורש עבודה עם כלי ה STATEMATE בלבד. כמו כן, הקוד שלי יותר קצר, קריא ופשוט להבנה, וניתן לייצר מפרטי CTL בצורה קלה ופשוטה.
בעבודה שלי הצלחתי לשאוב מהם רעיונות לאיך לעצב את מימוש המקביליות על ידי שימוש ב modules שונים לכל תת דיאגרמה מקבילית, אך בשונה מהם, אני מנהל את כלל האירועים והמשתנים בmain ולא ב modules אחרים, ואני מזריק אותם אל תוך ה modules המקביליים.

פרויקט UMPLE

כחלק מהמחקר על מימוש הפרויקט, נחשפתי לפרויקט umple [15]. מטרת הפרויקט הינה לייצר שפה למידול דיאגרמות (תמיכה בכמה סוגים של דיאגרמות, ביניהן גם דיאגרמות מיצוב). הפרויקט הוא פרי עבודתם של חוקרים מאוניברסיטת אוטווה בארה"ב. הפרויקט מאוד דומה לעבודת גמר שלי, אך הוא מייצר שפה חדשה לגמרי. על ידי הגדרה של הדיאגרמה בשפה החדשה, הספרייה מסוגלת לשרטט דיאגרמה בהתאם. השפה יחסית פשוטה ללמידה אך איננה מכסה את כל התרחישים האפשריים לדיאגרמת מיצוב על פי ההגדרה של דוד הראל [2]. היתרון של הספרייה הזאת הינו שהוא מאפשר המרה לשפות אחרות, כך למשל, ביצירה של דיאגרמה מיצוב כלשהי על פי umple, אפשר להמיר לשפת java, c++ , וגם ל NuSmv ול scxml (ההמרה ל scxml הינה ניסיונית והפרויקט מצהיר כי אין לקחת אותה כתקינה בהכרח, כמו כן, הוא לא מצליח להמיר כל דיאגרמה ל scxml)
הפרויקט הזה שימש לי כהשראה בעבודת גמר שלי, מכיוון שמאפשר לוודא

מעבר מ scxml ל NuSmv על מנת להשוות אותו מול התוצר של העבודה שלי.
ההמרה שלהם ל NuSmv שימשה כהשראה למבנה שאני מייצר ב NuSmv בעבודת גמר שלי.
הפרויקט הינו פרויקט Online קל לשימוש, אך ההבניה של הדיאגרמות וההמרה שלהם לוקה מאוד בחסר ומפספסת חלק מהותי מהעושר הרב של דיאגרמות מיצוב.
בחלק של הדוגמאות, נראה השוואה בין תוצר של פרויקט umple לבין תוצר של הפרויקט שלי.

5. שלבי הפרויקט

מחקר כלים ליצירת state chart

בשלב הראשון של הפרויקט, נדרש מחקר על כלים שונים שיאפשרו שרטוט של דיאגרמות מיצוב בצורה נוחה וקלה, שתתן מענה לסמנטיקה של הדיאגרמות על פי דוד הראל. הקריטריונים שבחנתי בכלים:

1. יכולת שרטוט דיאגרמות בצורה נוחה וקלה
2. תמיכה בסמנטיקה רחבה ככל האפשר של דיאגרמות מיצוב על פי הסטנדרט של דוד הראל
3. כלי OS\חינמי שיהיה אפשר להפיץ לשימוש רחב
4. המרה נוחה ל XML

לאחר סקר כלים, נבחנו 3 הכלים האלה[10]:
UMLET – כלי בסיסי מאוד לציור רכיבים, שמתבסס בעיקר על קוד JAVA. אין אפשרות לייצא את הציור לקובץ XML או כל ייצוג טקסטואלי אחר. מבחינת יכולות ציור, הכלי מספק תרשימים בסיסיים מאוד ויהיה מאוד קשה לעבוד איתו ביצירת דיאגרמות מיצוב מורכבות. הכלי חינמי אך לא מתאים לשימוש בפרויקט שלנו.

STARUML – כלי מתקדם יותר לציור דיאגרמות. מאפשר יצירת של רכיבים מורכבים, והוספה של תנאים, אילוצים, אירועים ושמות לכל רכיב. יצירת תרשימים מורכבים אפשרי, פשוט יצטרך עבודה לא פשוטה. בעזרת תוסף (*plugin*), הם מאפשר הוצאה של הדיאגרמה לקובץ XML מה שמאפשר לנו לשלוח את התוצאה לתוכנה שאנחנו בונים בפרויקט. הכלי היה OS בגרסתו הראשונה, אך מאז (היום בגרסה 4) הוא כבר בתשלום. את גרסה 1 אפשר עדין למצוא ולעבוד איתה, אך היא לא בתמיכה ולא מתוחזקת. כמו כן, התוכנה מסובכת לעבודה וההמרה ל XML לא נוחה.

Yakindu – כלי מתקדם מאוד ליצירת דיאגרמות מסוגים שונים. הכלי מאפשר אופציות נרחבות מאוד, ונותן מענה לכל התחביר הנדרש

לדיאגרמות מיצוב. לכלי קיימת המרה מובנת ל SCXML ויחסית נוח לעבודה. הכלי מבוסס ECLIPSE IDE והוא מאפשר להריץ סימולציה של הדיאגרמות שבנית בצורה אינטראקטיבית ולבדוק אותם.
הכלי איננו OS, אך קיים רישיון חינמי לכל משתמש לא מסחרי, כמו כן, קיים גם רישיון אקדמאי בחינם גם כן, כך שבפועל, הכלי מונגש לכל מי שרוצה לעבוד איתו ויכול לשמש לצרכים אקדמאים גם בפרויקט שלנו. שני הרישיונות מאפשרים את כל היכולות שצריכים לפרויקט שלנו.
בחודש דצמבר 2021, החברה יצאה עם גרסה אינטרנטית (*WEB*) של הכלי עם כלל היכולות, מה שמקל מאוד להתחיל לעבוד עם הכלי מכיוון שגם הרישיונות מתקבלים הרבה יותר מהר, וגם זה מוריד את הצורך להתקין את הכלי על המחשב ולהסתבך עם העבודה איתו.
בעקבות כל היתרונות המצוינים למעלה, בחרתי בכלי הזה לעבודה בפרויקט הגמר.

המרה של השרטוט ל XML

הייצוג שנבחר לייצוג הדיאגרמה הינו XML, או ליתר דיוק, SCXML.
זהו ייצוג רשמי של דיאגרמות מיצוב ויש לו פירוט נרחב באינטרנט, והגדרה פורמלית על ידי ארגון W3C , לכן הבחירה בו הינה טבעית על מנת ליישר קו עם ייצוגים פופולריים ונפוצים בעבודות אחרות.
הכלי שבחרתי לעבוד איתו, yakindu, מאפשר המרה של הדיאגרמה שיצרנו לSCXML בצורה ישירה.

המרה מ XML ל NUSMV

את קובץ ה XML שקיבלנו מכלי yakindu ניקח כקלט לקוד שכתבתי בעבודת הגמר. הקוד עובר על הקובץ, ויוצר ממנו קובץ smv כפלט.
את הקוד כתבתי בשפת פייתון בגרסה 3.9 מכיוון שזוהי שפה מאוד עשירה וקלה, ומאפשרת הרצה של הקוד כסקריפט לנוחות שימוש בהעברה שלו ממחשב למחשב ויכול לרוץ על כל פלטפורמה.

קובץ קלט: input.scxml (צריך להיות קיים בתיקיית הסקריפט)

קובץ פלט: output.smv (ייווצר לאחר הרצה בתיקיית הסקריפט)

הרצה של הקובץ בעזרת NUSMV2.6

לאחר יצירת קובץ Output.smv , אשר מייצג את הדיאגרמה בשפת NUSMV, הסקריפט יפתח חלון CMD ויריץ את הקובץ בעזרת פקודה: "nusmv output.smv" . הפקודה תריץ את הקובץ בעזרת ספריית NuSmv בגרסה 2.6 , תבצע אימות (*validation*) על הדיאגרמה ותוציא

את הפלט שלה אם הדיאגרמה עברה ולידציה או לא (על פי הפלט של ספריית NuSmv).

5. ה FLOW של הקוד

הקוד שבנתי לצורך המרה בין scxml ל NuSmv מורכב מ 7 פונקציות מרכזיות: (על פי סדר הפעלתן)

הפונקציה	פירוט
initSmvFile	מייצרת את מסמך הפלט וכותבת אליו את המבנה של קובץ ה NuSmv
defVars	מייצרת את האירועים והמצבים של הדיאגרמה בשפת NuSmv, מגדירה אותם בחלק של ה VAR בקובץ פלט
defGuards	מייצרת את כל ה guards של הדיאגרמה ושומרת אותם לשימוש ברחבי הקוד
initVars	מאתחלת את ה guards על פי ההגדרה שלהם בדיאגרמה. כותבת אותם בחלק ה INIT של קובץ ה NuSmv
defineTrans	מגדירה את כל המעברים בדיאגרמה. הפונקציה עוברת על כל ה transitions בדיאגרמה, מגדירה את התנאים שלהם וכותבת אותם בחלק של ה DEFINE בקובץ NuSmv
assignSection	מגדירה את החלק של ASSIGN בקובץ NuSmv. זהו חלק מהותי שכן הוא מייצר את ההגדרות של המעברים של המצבים והאירועים של הדיאגרמה
ctlspec	מייצר את החלק האחרון בקובץ, האימות על פי CTL שכל האירועים נגישים.
parallelSectionHandler	פונקציה שאחראית לטיפול בחלקים המקביליים, היא פועלת רק אם היא מזהה שיש קטע מקבילי ב XML. הפונקציה יוצרת את ה Modules הנוספים של התת דיאגרמות המקביליות

בחרתי להוסיף ולידציה נוספת מעבר לקוד NuSmv רגיל שעובר בסימולציה על המצבים. הוולידציה נעשית על ידי CTL ומוודאת שכל המצבים נגישים ואין deadlocks. זוהי ולידציה שאיננה הכרחית למהות עבודת הגמר, אך היא נותנת חווי ראשוני טוב שהקוד שנוצר הוא טוב ומייצג את הדיאגרמה בצורה נאמנה ולכן הוספתי אותו (בהשראת umple).
שלבי הקוד:

1. יצירת הקובץ פלט ובנייה שלו כקובץ smv.
2. מעבר על קובץ xml והוצאה של האירועים והמצבים והגדרתם
3. מעבר על קובץ xml והוצאה של הguards , הגדרתם ואתחול שלהם
4. יצירת קטע של DEFINE בקובץ פלט, בו מוגדרים כל המעברים בדיאגרמה על פי תנאיהם
5. יצירת קטע ASSIGN בקובץ פלט, בו מוגדרים האירועים, ה guards והמצבים בשלבים הבאים שלהם (next) על פי התנאים שהוגדרו לכל רכיב
6. יצירת חלק ה CTL לצורך ולידציה.

הקוד מפיק קובץ smv כפלט ומריץ אותו על ידי פקודת NuSmv בחלון cmd.

6. <u>מורכבויות במימוש הפרויקט</u>

הפרויקט לא היה פשוט למימוש וכלל מורכבויות בכמה רבדים. במהלך הפרויקט נאלצתי להתמודד עם בעיות שונות שצצו במהלך העבודה, אפרט כאן כמה מין המורכבויות המרכזיות שנתקלתי בהן:

1. בעיות תחביר (*syntax*) – כמו בכל פרויקט שמטרתו להמיר בין פורמטים שונים, גם כאן נתקלתי בבעיות רבות בפער בין התחביר של שתי השפות. קיימים לא מעט הבדלים באכיפה של כל שפה והתחביר שלה כך שהייתי צריך לייצר מנגנון שיודע לתרגם ולהתמודד עם פערים אלה מלבד ההמרה הלוגית שבמהות הפרויקט. מספר דוגמאות:
   a. אכיפת שמות – state chart מאפשר שמות עם שילוב של נקודה באמצע אך NuSmv לא מאפשר דבר כזה – התמודדות: המרה של כל נקודה לקו תחתון בשמות המצבים והאירועים.
   b. אופרטורים - אופרטורים ב state chart שונים מאופרטורים ב NuSmv כך למשל: & צריך להפוך ל && ו == הופך ל = וכן הלאה
   c. משתנים בוליאניים – ב state chart משתנים בוליאניים הינם true ו false אך ב NuSmv הם TRUE ו FALSE וכמובן שזה case sensitive.
   d. אין ב NuSmv תמיכה באיברים מסוג integer וב state chart יש. המעקף שנדרש כאן הוא להגדיר ב NuSmv משתנים מספריים מוגבלים (בין 100- ל 100 בשביל לכסות את כל האופציות)
2. שפת NuSmv – השפה יותר מחוברת לעולם האקדמאי והיא איננה ידידותית למפתח. הדוקומנטציה שלה מאוד דלה (מסמך PDF אחד

ארוך ומסורבל) ואין ממשק נוח למפתח מתחיל בשפה. הלמידה של השפה מאוד סיזיפית ואין הרבה דוגמאות קוד טובות ללמידה מהן. מעבר מהיר על מה שמתיימר להיות ה programmer manual שלהם [16] ולראות כמה הוא ריק בשביל להבין את החוסר במידע שימושי בעת פיתוח לשפה הזאת.
בנוסף, השפה איננה בשימוש רחב בקהילה. מספר הפרויקט שעולים בחיפוש של NuSmv ב GitHub מעלה 169 פרויקטים בסך הכל, וב stack overflow בסך הכל 209 שאלות על הנושא הזה.
המרכיב הזה היה מאוד קריטי בהתמודדות עם השפה ועם היכולות שלה, וגורם מעכב משמעותי במיצוי הפוטנציאל של השפה.

3. מימוש של דיאגרמות מקבילות – כמו במאמרים שהבאתי בפרק של "סקירה ספרותית ועבודות קשורות", ההתמודדות עם דיאגרמות מקביליות הינה הקושי הגדול ביותר במימוש דיאגרמות מיצוב ב NuSmv, מאמרים שונים ניסו להתמודד עם הבעיה הזאת ללא פתרונות פרקטיים (ראו פירוט בפרק שם). המורכבות להתמודדות עם דיאגרמות מקבילות נובעת מכמה סיבות:
   a. לפי הדוקומנטציה הרשמית של NuSmv , אין יותר תמיכה באופרטור processes מה שמאפשר עבודה א-סינכרונית בין מודולים שונים.
   b. יצירה של דיאגרמה מקבילית שובר את ה FLOW הטבעי של ההרצה של הקוד, מכיוון שמפצל לך את המצבים ל – 2 מצבים שונים הקיימים בו"ז, לעומת המצב הרצוי בו משתנה state תמיד מכיל את המצב הנוכחי בו אתה נמצא – מקשה בעיקר בכתיבת specifications ב CTL
   c. יצירה של מקביליות סותרת את התפיסה הבסיסית של state chart שאם תת-דיאגרמה כלשהי לא פעילה, אז גם אף אחד מהמצבים הפנימיים שלה לא פעיל, ואין דרך לממש מקביליות בלי שאחד המצבים בפנים יהיה פעיל (בגלל העובדה ש NuSmv עובד במקביליות מלאה בין modules שונים, כולם רצים כל הזמן, ביחד)

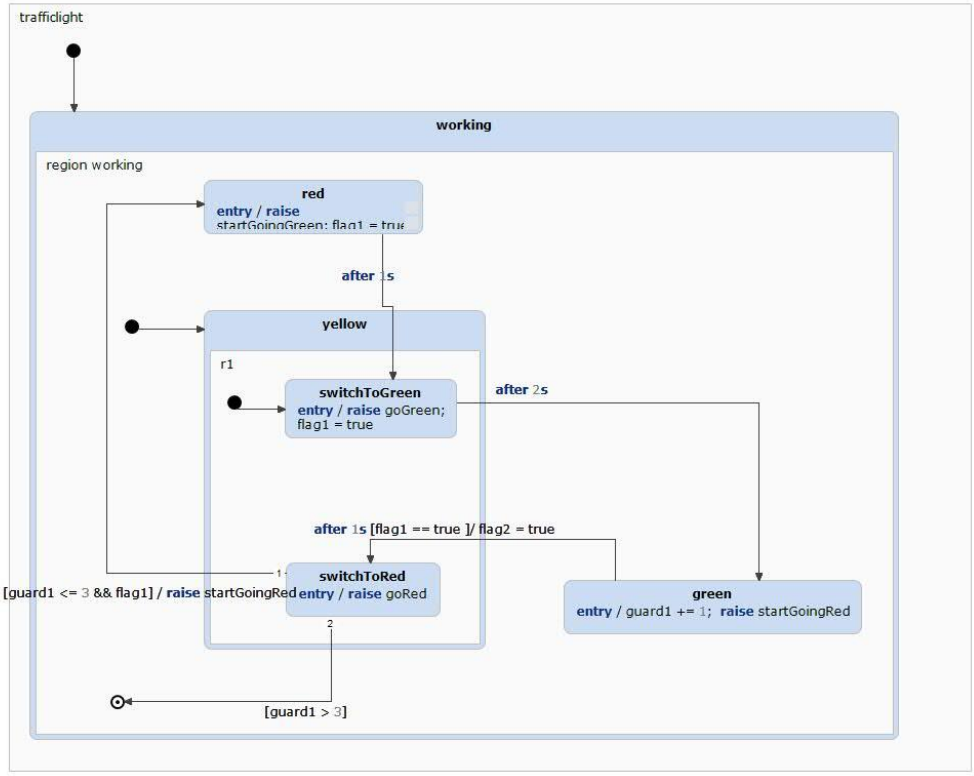
בשביל להתמודד עם הבעיה, פרסמתי שאלה ב stack overflow [17] הנוגעת להתמודדות הזאת, אך השאלה נותרה ללא אף מענה למרות שעברו ימים רבים.
למרות הקשיים מימשתי בעבודה שלי מקביליות על ידי שימוש במודולים שונים עבור התת דיאגרמות המקביליות. בגלל איך ש NuSmv עובד, כל התת מודולים עובדים כל הזמן במקביל ובסינכרוניות (בגלל אי שימוש באופרטור processes), את ההשראה קיבלתי ומימשתי בדומה למאמר [14] (ראו פירוט בפרק של הסקירה הספרותית), אך בשונה ממנו, אני מנהל את האירועים והguards במודול main ומזריק אותם לתת המודולים המקביליים.

בס"ד

7. דוגמאות

דוגמא 1 - trafficLight

דיאגרמת המיצוב



קובץ ה SCXML שנוצר מהדיאגרמה

trafficLight.scxml

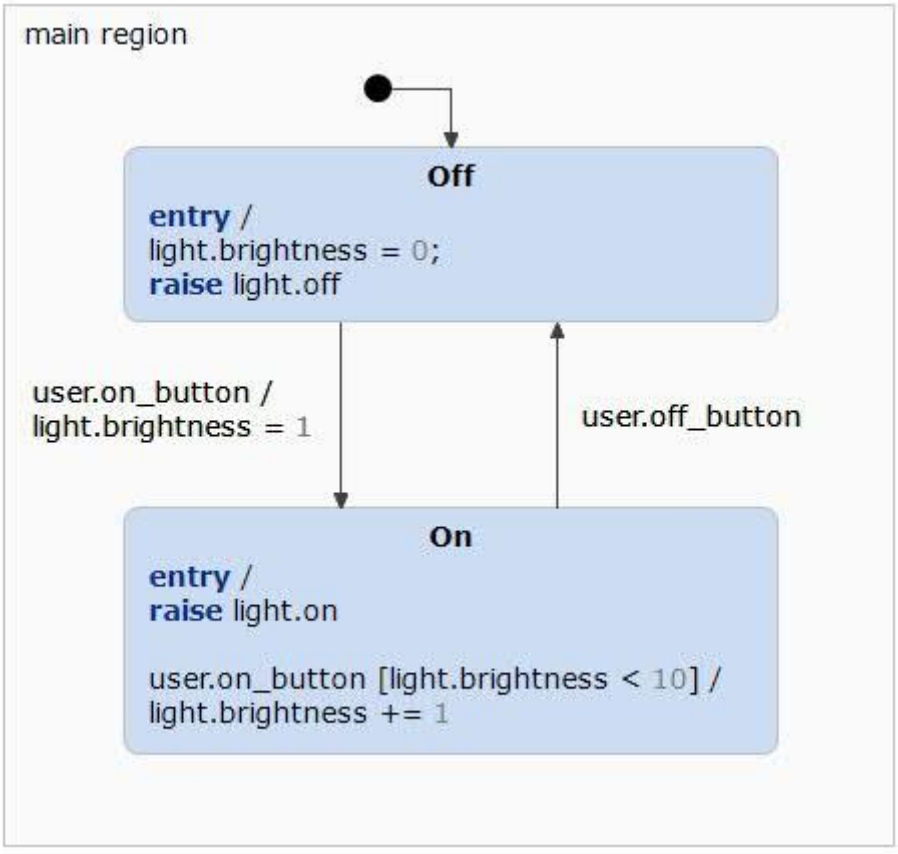
קובץ ה SMV שנוצר מהסקריפט

output.smv

הפלט של הרצת output.smv:

```
C:\Users\Ofek\Documents\[illegible]\state charts\examples\trafficlight>nusmv output.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification EF state = yellow  is true
-- specification EF state = switchToRed  is true
-- specification EF state = working  is true
-- specification EF state = switchToGreen  is true
-- specification EF state = trafficlight  is true
-- specification EF state = working_final_0  is true
-- specification EF state = red  is true
-- specification EF state = green  is true
```

דוגמא 2 – lightSwitch



בס"ד

קובץ SCXML שנוצר מהדיאגרמה

קובץ SMV שנוצר מהסקריפט

הפלט של ההרצה של הקובץ SMV :

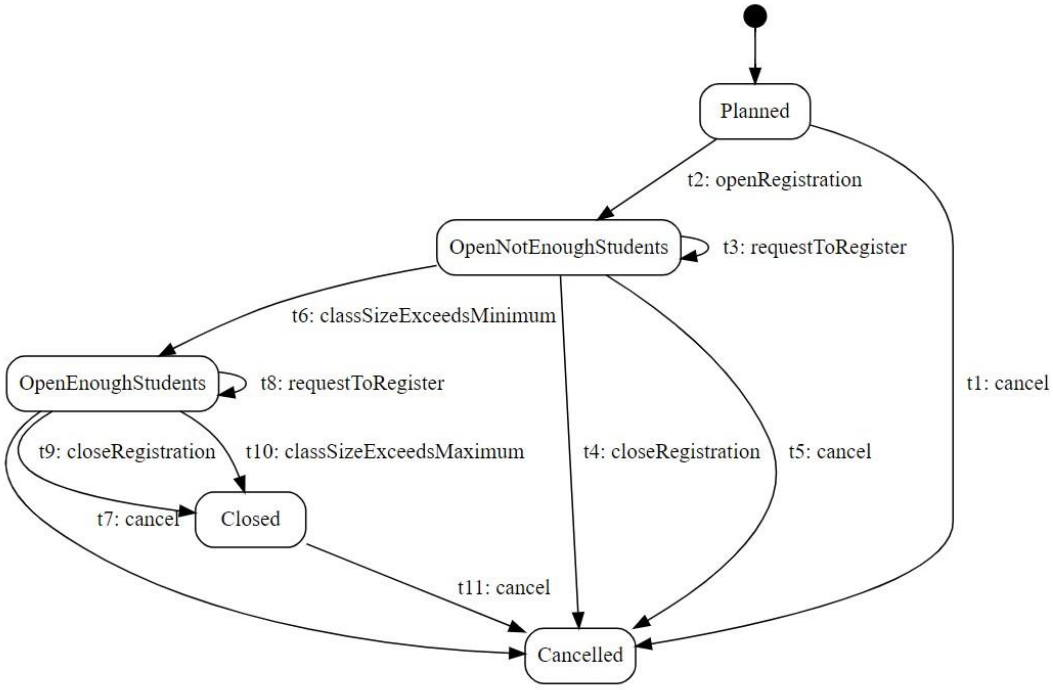
```
C:\Users\Ofek\Documents\[illegible]\state charts\examples\lightSwitch>nusmv output.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification EF state = main_region  is true
-- specification EF state = Off  is true
-- specification EF state = On  is true
```

בס"ד

דוגמא 3 – courseSection – השוואה מול Umple

בדוגמא הזאת, יצרנו שרטוט של דיאגרמה בפלטפורמה ה WEB-ית של Umple, והוצאנו גם קובץ SCXML וגם קובץ SMV של הדיאגרמה, ואז הרצתי את הסקריפט שלי על הקובץ SCXML והשוואתי את התוצאה לתוצאה של Umple.

הדיאגרמה:



הקובץ SCXML שיצא מהדיאגרמה:

השוואה בין שני ה NuSmv שיצאו:

הקובץ של Umple:

הקובץ שיצא מהסקריפט:

ניתן לראות ששני הקבצים מאוד מאוד דומים, מלבד העובדה שב umple מחלקים לתתי מודלים והסקריפט שלי לא. אך מעבר לזה, המעברים והתנאים שנוצרים מדויקים מאוד וניתן לראות שהתוצאה הינה אותה תוצאה.

בס"ד

הפלט של ההרצה של קובץ output.smv:

```
C:\Users\Ofek\Documents\[illegible]\state charts\examples\courseSection from umple>nusmv output.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification EF state = Cancelled  is true
-- specification EF state = Closed  is true
-- specification EF state = Planned  is true
-- specification EF state = OpenEnoughStudents  is true
-- specification EF state = OpenNotEnoughStudents  is true
```

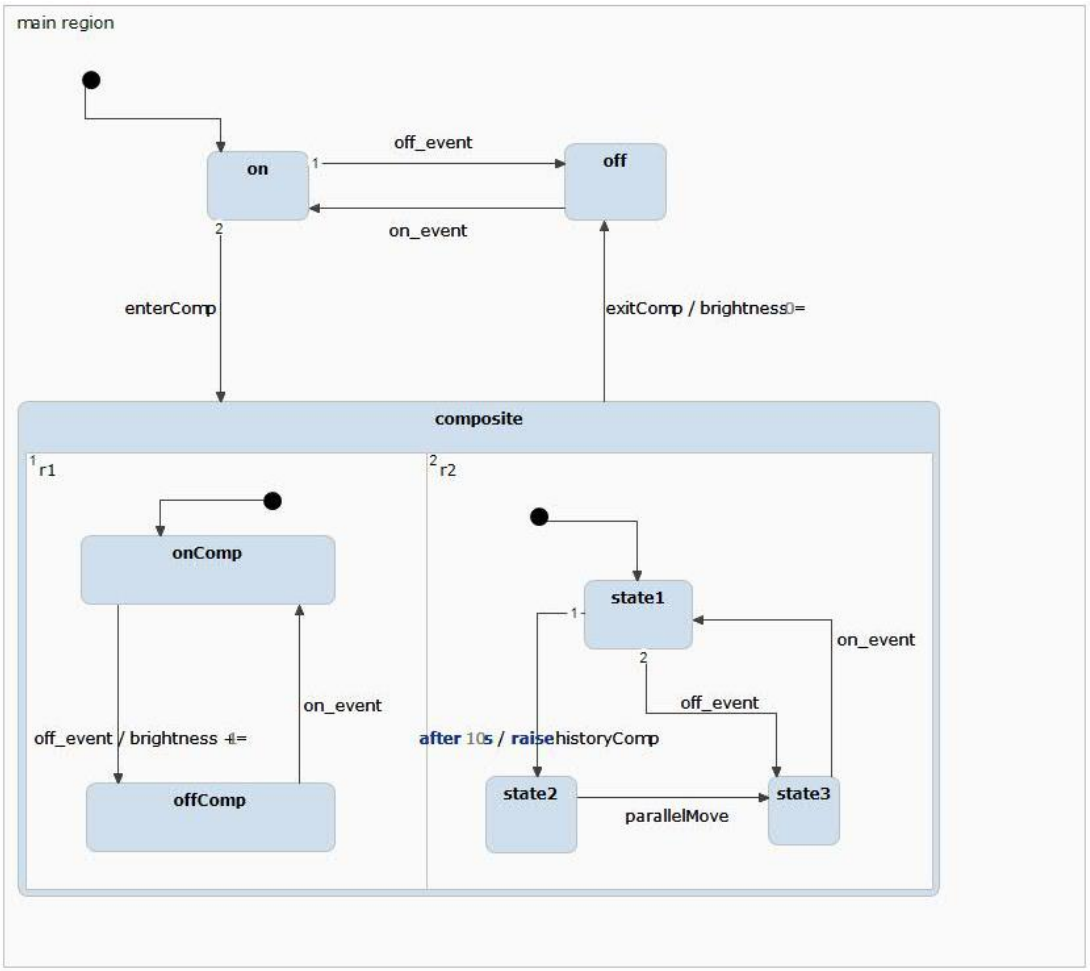
הרצה של הקובץ שנוצר ב umple:

```
C:\Users\Ofek\Documents\[illegible]\state charts\examples\courseSection from umple>nusmv courseSection.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification EF courseSectionStatus_Machine.courseSectionStatus.state = Status_Planned  is true
-- specification EF courseSectionStatus_Machine.courseSectionStatus.state = Status_OpenNotEnoughStudents  is true
-- specification EF courseSectionStatus_Machine.courseSectionStatus.state = Status_OpenEnoughStudents  is true
-- specification EF courseSectionStatus_Machine.courseSectionStatus.state = Status_Cancelled  is true
-- specification EF courseSectionStatus_Machine.courseSectionStatus.state = Status_Closed  is true
```

בס"ד

דוגמא 4 – parallel example

בדוגמא הזאת, יצרתי דיאגרמה מקבילית יחסית פשוטה להבנה על מנת להביא דוגמא למימוש המקבילי של הקוד:



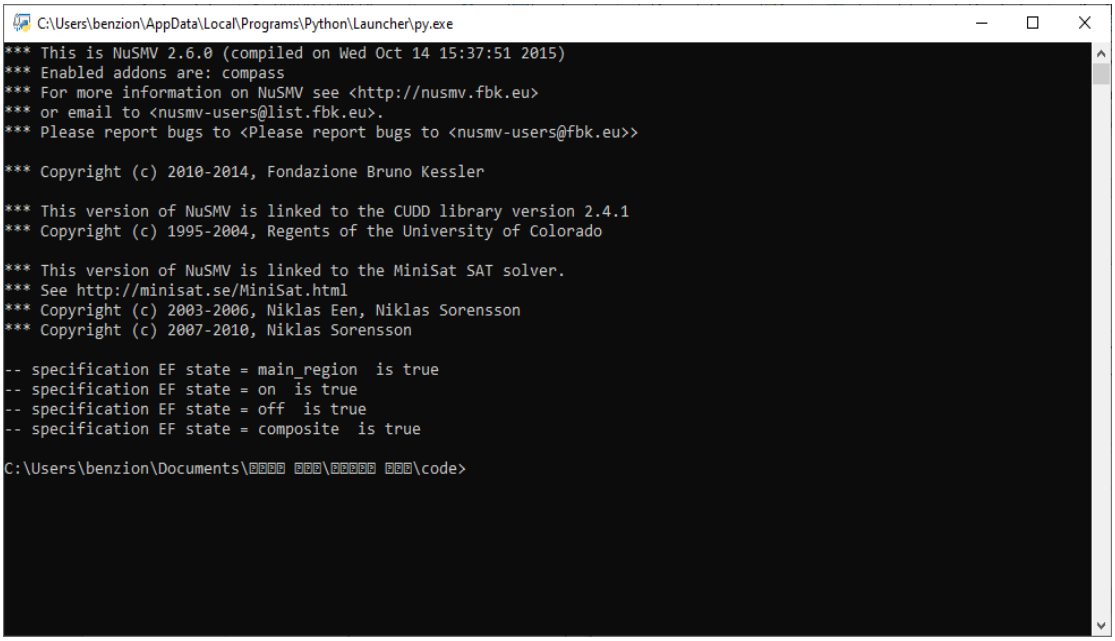
הקובץ SCXML שיצא מהדיאגרמה:

input.scxml

הקובץ SMV שיצא אחרי העיבוד:

output.smv

הפלט של ההרצה של קובץ output.smv:



בגלל הקטעים המקביליים בדיאגרמה, ה main "מכיר" רק את המצבים שלו ולכן יכולתי לייצר משפטי CTL לצורך אימות כברירת מחדל, רק על המצבים שלו. כמובן שצריך להוסיף בדיקות ידניות נוספות.

הפניות:

1. https://statecharts.github.io/what-is-a-statechart.html
2. ***The STATEMATE Semantics of Statecharts.*** David Harel, Amnon Naamad. ACM Trans. Softw. Eng. Methodol. 5(4): 293-333 (1996)
3. https://www.w3.org/TR/scxml
4. https://en.wikipedia.org/wiki/Linear_temporal_logic
5. ***NuSMV 2: An OpenSource Tool for Symbolic Model Checking.*** A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella. In Proceeding of International Conference on Computer-Aided Verification (CAV 2002). Copenhagen, Denmark, July 27-31, 2002.
6. ***Symbolic model checking***. Kenneth L. McMillan. Kluwer 1993, ISBN 978-0-7923-9380-1, pp. I-XV, 1-194.
7. ***Binary Decision Diagrams***. *Randal E. Bryant.* From "Handbook of model checking", 2018: 191-217.
8. ***NuSMV: a new symbolic model verifier***. A. Cimatti, E. Clarke, F. Giunchiglia and M. Roveri. In N. Halbwachs and D. Peled, editors. Proceeding of International Conference on Computer-Aided Verification (CAV'99). In Lecture Notes in Computer Science, number 1633, pages 495-499, Trento, Italy, July 1999. Springer.
9. ***SAT-Based Model Checking***. Armin Biere & Daniel Kröning. From "Handbook of model checking", 2018: 277-303.
10. **https://docs.staruml.io**
    https://www.itemis.com/en/yakindu/state-machine/documentation
    https://www.umlet.com/
11. ***Using the NuSMV Model Checker for Test Generation from Statecharts.*** Masaya Kadono, Tatsuhiro Tsuchiya, Tohru Kikuno, Osaka University, PRDC 2009: 37-42.
12. **Implementing Statecharts in PROMELA/SPIN.** Erich Mikk, Yassine Lakhnech, Michael Siegel, Gerard J. Holzmann: WIFT 1998: 90-101
13. **Hierarchical automata as model for statecharts**. E. Mikk, Y. Lakhnech, and M. Siegel. In *Asian Computing Science Conference (ASIAN'97)*, volume 1345 of *LNCS*. Springer Verlag, December 97
14. **Modular Translation of Statecharts to SMV**. E. Clarke, W. Heinle. August 2000.
15. https://cruise.umple.org/umpleonline

16. https://nusmv.fbk.eu/NuSMV/progman/progmanual-v2.html
17. https://stackoverflow.com/questions/70954645/what-is-the-best-way-to-implement-parallel-state-machine-in-nusmv-version-2-6