כלי הוכחות לוגיות

מגישים: רועי נוביץ' 203897186

אורן אור 203985262

מנחה: פרופסור דני קוטלר

Project Github link:

<https://github.com/roynovich1451/Final_project_logic_calculator>

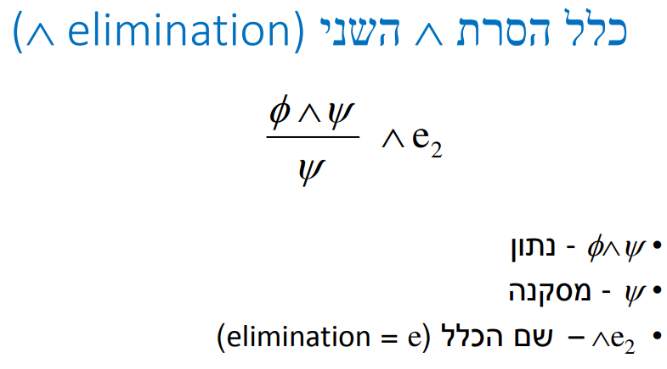
**תוכן עניינים**

1. **רקע**
2. **למה בחרנו בפרויקט**
3. **מטרת הפרויקט**
4. **Design**
5. **תהליך בניית הפרויקט**
   1. **בנייה דינאמית**
   2. **מימוש חוקי תחשיב הפסוקים**
   3. **שמירה וטעינה**
   4. **מימוש חוקי תחשיב היחסים**
   5. **שימוש בהוכחה ידועה**
6. **אלגוריתמיקה**
7. **תוצר סופי ועמידה ביעדים**
8. **סיכום**
9. **רקע**

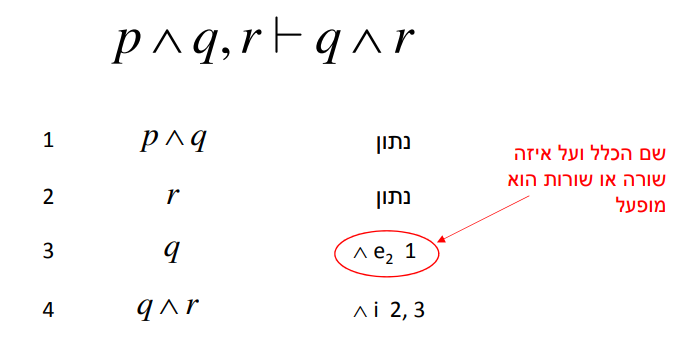
במתמטיקה הוכחה היא סדרה סופית של טענות הנובעות זו מזו בעזרת כללי היסק, תוך שימוש בהגדרות, אקסיומות, ובידע קודם שהוכח קודם לכן, המראה שטענה מסוימת היא נכונה. בלוגיקה מתמטית, הוכחה היא סדרה סופית של פסוקים במסגרת שפת תחשיב יחסים נתונה, המורכבת מאקסיומות ומגזירות באמצעות כלל היסק.

מבנה של הוכחה לוגית הוא טענה ונימוק, כאשר בקורס לומדים שניתן למספר את השורות ולרשום בעמודת הנימוק את השורות שבהן הוא נמצא.

כלל הגזירה " elimination∧":

****

שימוש הכלל בהוכחה:

****

1. **למה בחרנו בפרויקט הזה**

כאשר ביצענו פגישת סיעור מוחות ראשונית לגבי הפרויקט שאותו אנו רוצים לעשות היה לנו ברור שהמטרה הראשונה שלנו היא לתת פתרון למשהו שמבחינתנו היה חסר לאורך התואר, אם זה בקורס ספציפי או באופן כללי. באותו שלב בדיוק סיימנו את הקורס ב C# WPF .net עם דני ורצינו להמשיך ולהעמיק בשפת התכנות הזאת.

לאחר חשיבה ורעיונות מרובים נזכרנו בקורס לוגיקה (קורס חובה לכל הסטודנטים בתואר במדעי המחשב) ותהליך כתיבת והגשת שיעורי הבית המייגע.

מלאכת שיעורי הבית בקורס לוגיקה מצריכה כתיבה רבה. מעצם צורת כתיבת ההוכחה על כותב התשובה להיות ברור ומדויק ולשמור על סדר. בניגוד למקצועות רבים אחרים אשר כוללים בעיקר מלל ואולי סימונים מתמטיים סטנדרטים, הוכחה לוגית מצריכה סימונים שאינם טריוויאליים וסטודנט אשר רוצה לענות על שאלות אלו בוורד (או בכל תמלילן אחר) מוצא עצמו מבזבז זמן רב על חיפוש הסימנים וארגונם בטבלה, ולכן רבים נמנעים ובוחרים להגיש בכתב יד. הגשה בכתב בסוג התרגילים זה הקשה על הסטודנט מעצם צורת ההוכחה. לדוגמה, אם לאחר סיום הוכחה סטודנט מגלה שחסרה לו שורה באמצע הוכחה הוא נאלץ לשנות את כל המבנה של טבלת ההוכחה. בנוסף לכך ישנם סטודנטים בעלי כתב לא מובן או קושי בכתיבה מסודרת וברורה, מה שמקשה באופן ישיר גם על בודק התרגילים של הקורס.

רצינו לייצר תוכנה אשר תוכל לפתור את בעיות אלו ולתת ממשק עבודה נוח ופשוט לסטודנטים של הקורס.

הבנו שעם הכלים שרכשנו בשיעורי לוגיקה בנוסף על הקורס בC# נוכל ליצר כלי שיפתור בעיות אלו לסטודנטים בקורס.

1. **מטרות הפרויקט**

* **מטרה עיקרית:** יצירת כלי נוח לכתיבת ובדיקת הוכחות בלוגיקה, בין השאר בשיעורי הבית והתרגילים בקורס.
* **מטרות משנה:** 
  + יצירת ממשק עבודה על הוכחות לוגיות שיהיה כמה שיותר נוח.
  + מימוש בדיקת חוקי הלוגיקה בתוכנה.
  + יצירת תוכנה אשר תוחמת את הסטודנט אך בו בעת לא מאפשרת לו גישה ישירה לפתרון התרגיל.
  + בניית הפרויקט כ- Open source לצורך תיקונים עתידיים והרחבה.

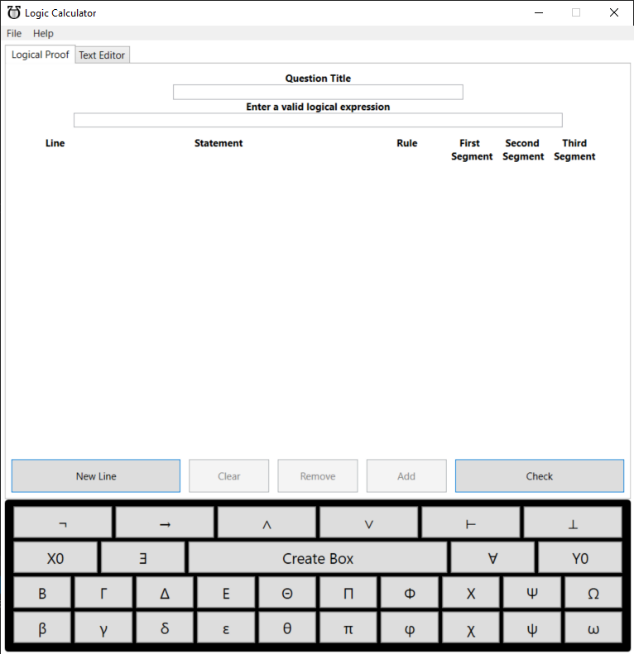
1. **Design**

התוכנה מחולקת לשלוש מחלקות עיקריות:

* 1. **Static Graphic user interface:**

חלק זה נבנה בתחילת הרצת התוכנה ואינו משתנה לכל אורך ריצת התוכנה.

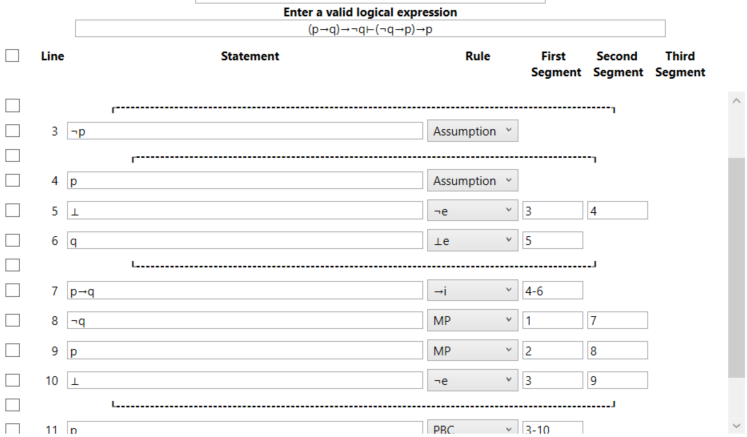
זהו השלד של הGUI המוצג למשתמש. למשל: מקלדת, תפריט, טאבים.



* 1. **Dynamic Graphic user interface:**

חלק זה תלוי בצרכי המשתמש, מאפשר שליטה למשתמש בכל מה שקשור בשורות ההוכחה, מאפשר הוספת/מחיקה/ניקוי שורות, כמו כן מתאים את עצמו לפי בחירת החוק הנבחר על ידי המשתמש בשורה הספציפית (משפיעה על כמות הסגמנטים המאותחלים בשורה).

חלק זה גם אחראי על יצירת תיבות ההוכחה.

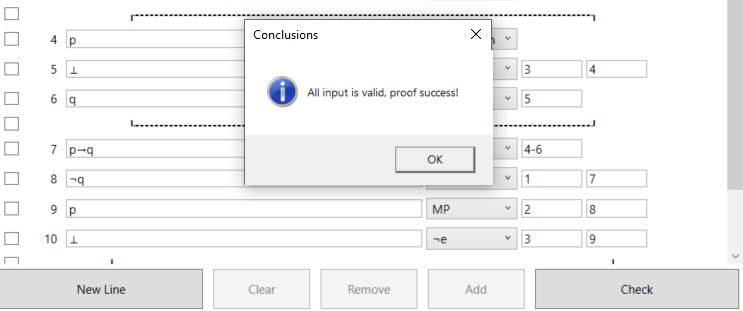


* 1. **Evaluation:**

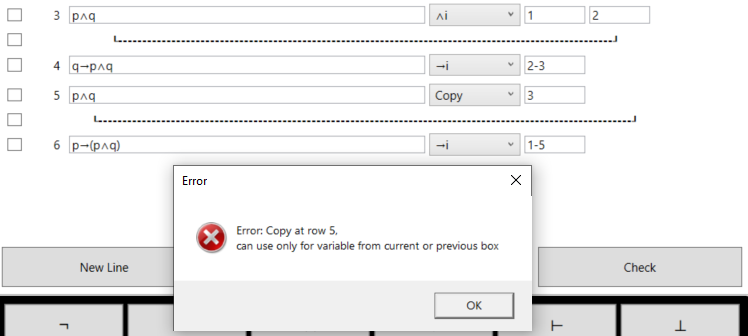
חלק זה של התוכנה הוא המהות שלה. בקוד עצמו הוא מופיע במחלקה נפרדת, שמכילה בתוכה את מימושי חוקי הלוגיקה על פי ספר הקורס:

Huth M., Ryan M., Logic in Computer Science, 2nd ed. Cambridge 200

ספר זה הוא מקור לימוד מקובל במקומות שונים. החלק הזה אחראי על בדיקת הנכונות של ההוכחה שנכתבה על ידי המשתמש עד השלב הנוכחי. הוא נכנס לשימוש רק מרגע לחיצה על כפתור "Check", על מנת שהשליטה תהיה בידי המשתמש. ראשית אנחנו בודקים את הסינטקס של הקלט ולאחר מכן את הנכונות הלוגית של ההוכחה, למשל, מספרי שורות, מיקום התיבות וכולי.

****

במקרה של בעיה בהוכחה, תקפוץ הודעה למשתמש שמכילה פרטים על מיקום הבעיה ומה הטעות שקרתה.



1. **בניית הפרויקט - ציר זמן**
   1. **איפון ראשוני**

בתחילה דיברנו עם פרופ' דני המנחה לגבי מה הבעיות והצרכים שעלינו לענות עליהם, ואיזה פונקציוליות הוא מעוניין שתהיה לתוכנה. לאחר מכן, ישבנו ויצרנו סקיצה ראשונית של מבנה התוכנה, ניסינו להבין כיצד להמיר את המבנה הברור של הוכחה לוגית לכדי תוכנה שמבוססת WPF.

רצינו לשים את נוחות השימוש במקום הראשון, שכן זאת אחת המטרות העיקריות של הפרויקט, ולבנות מודל אשר יהיה ברור למשתמש, זאת על ידי שמירה על עיצוב הדומה להוכחה לוגית סטנדרטית.

העיצוב נבנה בראשינו כאשר הסתכלנו על המחשב הנייד שלנו וממנו נלקח הרעיון של יצירת המקלדת מתחת למסך הראשי, כאשר מרכז התוכנה יהווה את החלק העיקרי בו המשתמש יבצע את פעולותיו ואילו בחלק העליון נוסיף אפשרויות עזר.

* 1. **בנייה סטאטית**

בשלב זה בנינו את דף הXAML, שהוא בעצם השלד של התוכנה, מטרת העל של השלב הזה הייתה להגיע לאב טיפוס ראשוני בכל מה שקשור בתצורת התוכנה, הרצון היה ליצר כמה שיותר מהר טופס סטאטי אשר יהווה מסגרת כלשהי עליה נוכל להתחיל לבנות ולהרחיב.

מכיוון שרוב העבודה בפרויקט הייתה בזמן הקורונה, וחלק נכבד היה תוך כדי עבודה ולימודים של שנינו, החלטנו לפצל את העבודה על מנת לפתח במקביל את התצורה הדינאמית של התוכנה ותוך כדי להתחיל במימוש חוקי תחשיב היחסים.

* 1. **מימוש חוקי תחשיב הפסוקים**

תחילת המימוש של החוקים הראשונים, הרעיון היה להפריד לגמרי בין מה שקשור לGUI והבניה שלו לבין מימושי החוקים ובדיקות הקלט של המשתמש, בשלב זה יצרנו את מחלקת Evaluation אשר הכילה בתוכה את המימושים הראשונים של החוקים.

במחלקה זו ישנו משתנה Is\_Valid אשר לפי ערכו בסוף הבדיקות המבוצעות על החוק הרלוונטי נקבע האם המשתמש השתמש בחוק בצורה נכונה או שישנה שגיאה ועלינו להתריע על כך.

בשלב זה נתקלנו בהתלבטות משמעותית, הוכחה לוגית נבנית בצורה אינדוקטיבית, המשמעות היא שבכל שורה אותה אנחנו כותבים ההנחה היא שהשורות לפניה **נכונות**, לכן התלבטנו האם עלינו לבדוק בכל פעם רק את השורה האחרונה שנוספה על ידי המשתמש או לעבור מהתחלה על כל שורות ההוכחה.

לבסוף החלטנו לבצע תמיד בדיקה על כל שורות ההוכחה מאחר ואנחנו נותנים למשתמש לבצע שינויים בכל שורה בה הוא רוצה ולכן אין הבטחה כי שורות ישנות שהוא הכניס לא עברו שינוי על ידו ועל כן יש לבדוק גם אותן.

* 1. **בנייה דינאמית**

המרת אב הטיפוס ובנייתו בצורה דינאמית, זהו היה שלב שדרש מאיתנו הרבה זמן והרבה קוד.

יש לציין שבקורס בC# לא נוגעים באופן בניית GUI דינאמי בכלל והיינו צריכים להבין כיצד ניתן לייצר אובייקטים חדשים של WPF וגם לבצע עליהם מניפולציות לאחר מכן, כל זה בעזרת הקוד.

רצינו לייצר כמה שיותר אפשרויות למשתמש לשינוי, עריכה והוספה לשורות ההוכחה וכמו כן לאפשר לו להשתמש רק בכמות סגמנטים רלוונטית לאחר בחירת החוק בו הוא רוצה להשתמש.

בשלב זה גם הכנסנו את כל השליטה בתיבות ההוכחה (Boxes) ויצירתן.

לקראת סוף שלב זה ביצענו Demo ראשון בפני דני, הצגנו לו את היכולות הנוכחיות, התייעצנו לגבי קשיים שצצו במהלך הדרך, והתדיינו לגבי המשך הדרך.

* 1. **שמירה וטעינה**

הוספנו ממשק בין התוכנה לקבצי Word בעזרת שימוש בחבילת "**Xceed docs**" אשר נתנה לנו API נוח לעבודה עם מסמכי Word.

היה לנו ברור ששלב זה קריטי וחשוב מאחר ומטרתה הסופית של התוכנה היא לעזור בהגשת שיעורי הבית ולכן היה לנו ברור שצריך לאפשרות לייצר פלט מסודר ומאורגן אותו הסטודנטים יוכלו להגיש.

בנוסף אפשרות הטעינה מאפשר למשתמש להפסיק את שיעורי הבית באמצע ולחזור אליהם בשלב מאוחר יותר וכמו כן מאפשר לבודק התרגילים לטעון את תרגילי הסטודנטים ולבצע בדיקה מהירה על נכונות התרגיל.

כמובן ששלב זה גם עזר לנו בהמשך בכל שלב הDebug ונתן לנו אפשרות לשמור ולטעון הוכחות אשר נמצאו בהן באגים.

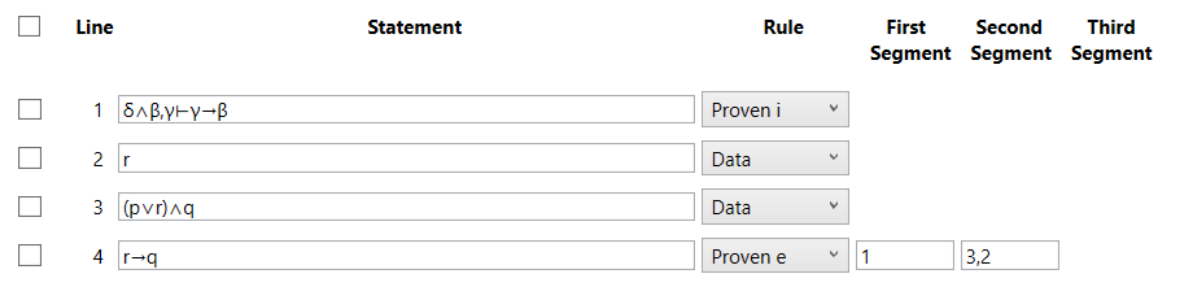
* 1. **מימוש חוקי תחשיב היחסים**

בשלב זה נתקלנו גם כן בבעיות, ידענו כי חוקים אלו קשים יותר למימוש אך בעיות נוספות צצו בבדיקות הקלט אשר התבצעו על כל הקלט של המשתמש.

היינו צריכים להתאים ולשנות חלק ניכר מבדיקות הקלט שלנו ולהוסיף מצבי קיצון שלא היה בהם צורך עד שלב זה.

* 1. **שימוש בהוכחה ידועה**

בעקבות הצעתו של דני הוספנו את היכולת להשתמש בהוכחה שכבר הוכחה כנכונה בהוכחה הנוכחית, הדבר מאפשר למשתמש לוותר על שלבים בהוכחה במידה ויש משהו שהוכח כבר ועל ידי כך לקצר את ההוכחה.

חוק זה הינו **גנרי** ומקל על שימוש המשתמש.

* 1. **Debug**

בסיום שלב הבניה של התוכנה, כאשר החלטנו שהגענו ליעד שהצבנו לעצמנו וכל הFeatures הוכנסו, התחלנו לבצע בדיקות מקיפות ולחפש אילו חוקים אינם עובדים כראוי ובאילו מקרים.

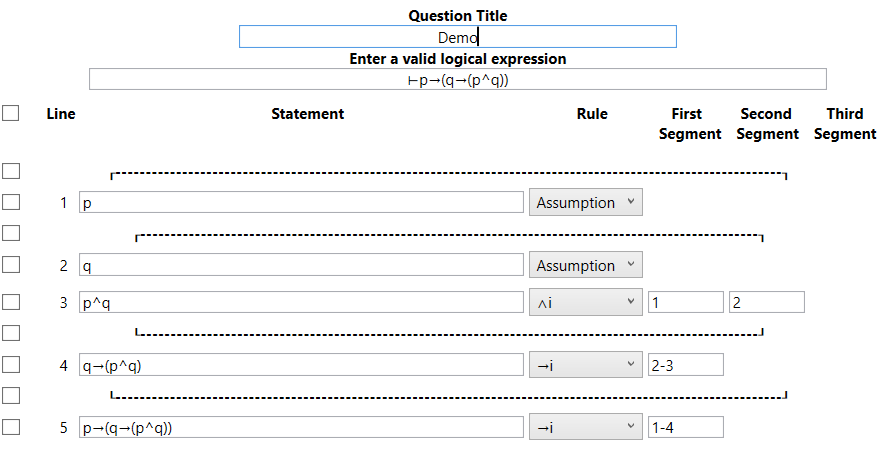
מאחר והמטרה הסופית שלנו היא שהתוכנה תהיה בשימוש של סטודנטים היה עלינו לנסות למצוא ולסגור כמה שיותר באגים על מנת להוציא תוכנה כמה שיותר אמינה.

השקענו שעות רבות בשימוש בתוכנה, השתמשנו בתרגילים ממבחנים, שיעורי בית וספר הקורס על מנת לנסות לראות כמה שיותר מצבי קיצון.

1. **אלגוריתמיקה**

אופן פעולת הבדיקה של התוכנה ממומש בצורה אינדוקטיבית, אנו בודקים את ההוכחה מהשורה הראשונה ועד האחרונה בצורה איטרטיבית , בצורה זו אנו יכולים להיות בטוחים שאם אנו מסתמכים בשורה הנוכחית על משהו שנגזר בשורות הקודמות הוא נבדק והוא נכון.

**נציג בקטע זה את האלגוריתם לבדיקת נכונות שורה,** נשתמש בדוגמא על שורה ספציפית אך אופן הפעולה זהה לכל שורה בהוכחה.

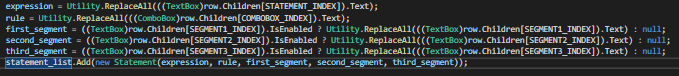
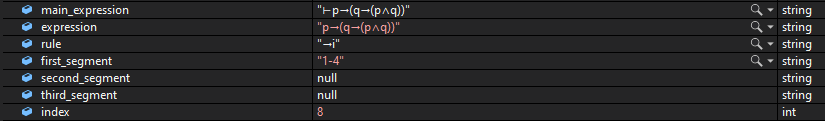


השורה הנבדקת

בדיקת נכונות שורת הוכחה מתחלקת לשלושה חלקים:

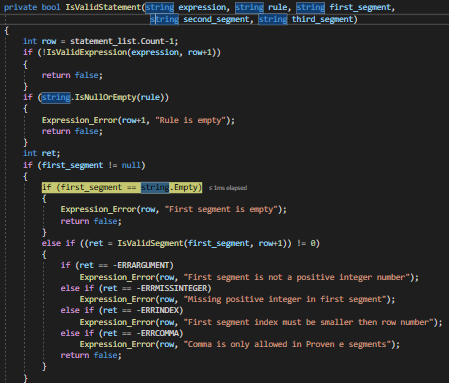
* 1. **קלט**

בשלב זה אנו קוראים את קלט המשתמש בשורה הנוכחית ויוצרים ממנו אובייקט Statement, אובייקט זה מכיל את כל שדות השורה הנוכחית:

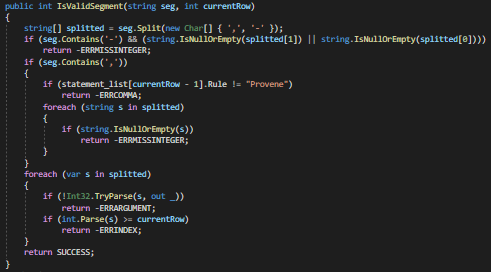
expression, rule, first\_segment, second\_segment, third\_segment  
אנו מוסיפים את השורה הנוכחית לתוך מערך של Statements.

לאחר מכן מגיע שלב בדיקת הקלט,

אנו בודקים את נכונות הסינטקטית של הExpression בשורה (שימוש בסורגים, שימוש בתווים מותרים, שימוש נכון במשתנים ובסימנים הלוגיים ועוד..)



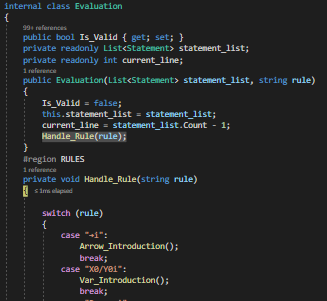
בסוף התהליך נבדקים הSegments (הרלוונטיים לחוק), מתבצעת בדיקה שהשורות שאליהן המשתמש הפנה אכן קיימות, ההפניה היא רק לשורה קודמות לשורה הנוכחית ואין שימוש בסימנים אסורים.



הודעת שגיאה מפורטת תוצג למשתמש במקרה שאחת הבדיקות יוצאת שלילית.

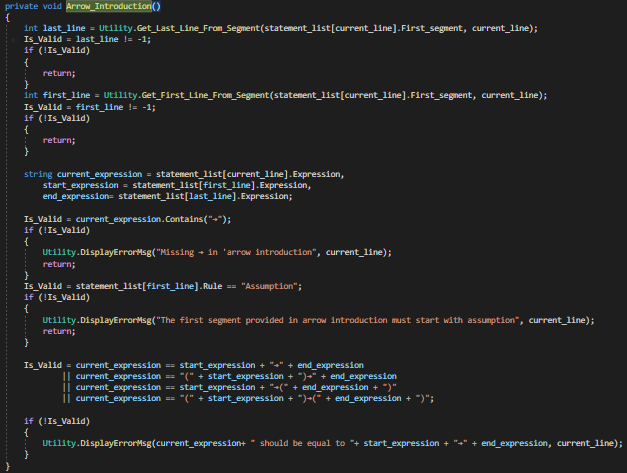
* 1. **שימוש בחוק**

כעת אנו מעבירים את רשימת הסגמנטים הנוכחית עם חוק השורה הנוכחית למחלקת Evaluation, תפקידה של המלקה היא בדיקת נכונות השימוש בחוק הנבחר על ידי המשתמש, במקרה של הדוגמא הנתונה "→i".





Handle\_Rule אחראית להעביר את טיפול השורה לפונקציה רלוונטית על פי סוג החוק.



31

21

1

בדיקת Arrow\_Introduction עושה Parsing על שורת הסגמנטים של החוק ולוקחת את הStatemets המופיעים בשורות הרלוונטיות (1), בודקת הופעת "→" בStatement הנוכחי (2) ומוודא כי אכן הStatement הוא הרכבה של Statements מגבולות התיבה (3).

במידה ואחת הבדיקות נכשלת הודעת שגיאה תצא למשתמש.

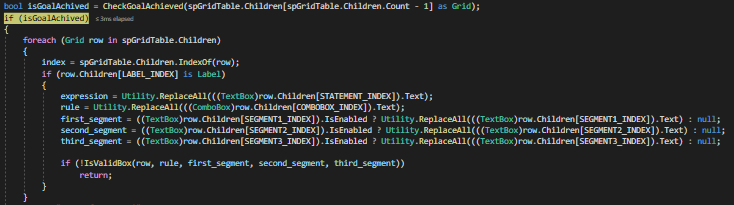
* 1. **תיבת הוכחה**

מנגנון בדיקת תיבת ההוכחה מופרדת מהבדיקות הקודמות אך אנו מתייחסים לחלק זה כחלק בלתי נפרד מאלגוריתם הבדיקה.

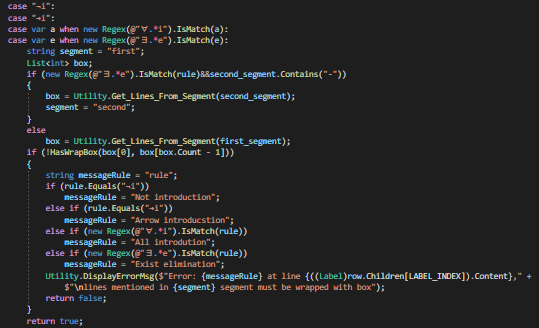
מאחר והתיבות (Boxes) מיוצרות על ידי הGUI הוא **אינו** נחשב כחלק מקלט המשתמש.

בעקבות החלטה נוספת בביצוע התוכנית ואופן יצירת התיבות בתוכנה החלטנו שבדיקות נכונות השימוש בתיבות ההוכחה תתבצע רק כאשר המשתמש הצליח להוכיח את מה שהוא רצה.

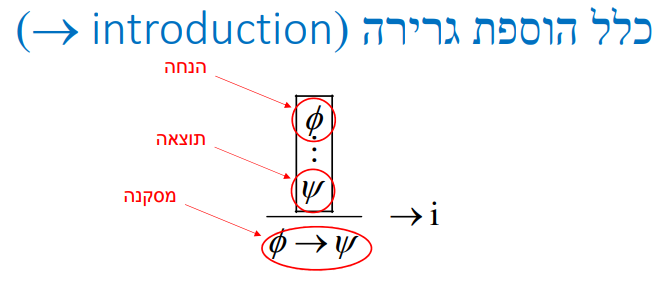
****בדיקות תיבות ההוכחה מתחילות אם ורק אם כל השורות בהוכחה נכונות מבחינה סינטקטית וגם ההוכחה הגיע לסיומה.

****במצב זה אנו נכנסים לאיטרציה נוספת על כל חלקי ההוכחה בה נבדקת אך ורק נכונות השימוש בתיבות ההוכחה, כמובן רק בחוקים אשר יש לתיבות משמעות.

בתוך IsValidBox אנו בודקים אם מתבצע השימוש בתיבות ההוכחה כמו שהחוק דורש

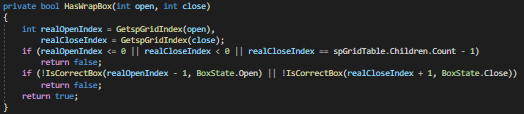
****

במקרה של הדוגמא שלנו:

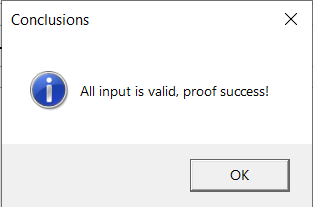


אנו צריכים לוודא כי ישנה תיבה אשר עוטפת את השורות אליהן הפנה המשתמש,

בדיקה זו מבוצעת ע"י הפונקציה HasWrapBox.



בסופו של התהליך, אם כל הבדיקות עברו המשתמש יקבל הודעה אשר תודיע לו שהכל תקין והוא סיים את ההוכחה.



1. **תוצר סופי ועמידה ביעדים**

התוצר הסופי הינו כלי אשר יודע להחזיר תשובה לגבי נכונות הוכחה שנכתבה על ידי המשתמש ובמידה ולא להתריע בדיוק מהי הסיבה לכישלון.

יצרנו תוכנה בעלת ממשק עבודה ברור ואינסטינקטיבי למשתמש, הממשק דומה מאוד לכל תוכנת Office ומקל מאוד על משתמשים אשר נחשפים לתוכנה בפעם הראשונה, מבנה הכנסת הקלט בכלי הינו זהה לתצורת כתיבת ההוכחה הנלמדת בקורס.

יצרנו דף משתמש (User manual) מפורט, בו הסברים על שימוש נכון בתוכנה, יכולותיה והסברים מפורטים על נכונות השימוש בכל אחד מהחוקים הממומשים.

יצרנו **קובץ התקנה** פשוט ומהיר אשר מאפשר לסטודנט התקנה קלה ונוחה ושימוש מידי על כל מחשב בעל מערכת הפעלה Windows.

אפשרות שמירה/טעינה מסמכי Word הופכת את התוכנה לרלוונטית וקלה לתפעול בכל הנוגע להגשת שיעורי הבית של הסטודנט, ובדיקת התרגילים.

מחלקת Evaluation אשר מכילה בתוכה את מימוש כל החוקים ניתנת להרחבה בקלות, ומאפשרת את הרחבת הפרויקט במידה ויש צורך בכך.

תצורת הודעת שגיאה למשתמש מאפשרת לו להבין בצורה מפורטת את הטעות אותה ביצע ביצירת ההוכחה כמו גם את מיקומה המדויק, אך בו בזמן אין התוכנה חושפת את הדרך הנכונה לפתרון.

הפרויקט נוהל מתחילתנו ועד סופו תחת Github, דבר אשר מאפשר גישה מכל מקום ומכל מחשב בעל חיבור לאינטרנט וכמו כן אפשרות עתידית להרחבת הפרויקט ושימוש חוזר בSource code.

1. **סיכום**

הפרויקט היה חשוב לנו מאוד,

התנסנו בזכותו בתהליך שלם של יצירת תוכנה, החל מרעיון ראשוני, מימוש, UI, UX, שימוש נרחב בשפת C#, Debugging אינטנסיבי וניהול קוד מלא בעזרת Git.

נעזרנו לכל אורך התהליך בפרופסור דני קוטלר, לקחנו את המלצותיו לתשומת לבינו וניסינו לייצר תוכנה נוחה ורלוונטית.

ניסינו לשים את עצמנו במקומו של המשתמש ולייצר חוויה טובה עד כמה שניתן.

אנחנו מרגישים שנתנו את המירב בפרויקט והתייחסנו אליו בכבוד הראוי.

בסופו של יום,

אנו מקווים שהתוכנה אכן תממש את מטרתה העיקרית ותהוה כלי בשימוש קבוע על ידי תלמידי קורס לוגיקה במכללה.

אנו מודים לדני על התמיכה והליווי לאורך התהליך.