אוניברסיטת בר אילן

המחלקה למתימטיקה ומדעי המחשב

יצירה אוטומטית של תוכניות והוכחת נכונותן באמצעות תהליך השלמה מורחב.

אלי פינצ'ובר

מוגש כמילוי חלקי של הדרישות למוסמך במדעי המחשב

רמת גן תשס"ב

1. תקציר

המשוואות הינן אבן יסוד בעולם המתימטיקה והמדע בכלל. באמצעות משוואות אנו מבטאים את הידע שלנו בתחום מסויים.

מכאן ואילך, קיימות אפשרויות שונות . לעיתים, ברצונינו לבדוק האם זהות מסויימת נובעת כמסקנה לוגית מהמשוואות הידועות. במקרים אחרים , ברצונינו למצוא פתרון למשוואה מסויימת. צורך נוסף הינו היכולת לפשט ביטוי מסויים לביטוי אחר, פשוט יותר.

היכולת להשתמש במשוואות ולהסיק מהן מסקנות הינו חיוני בתחומים רבים וביניהם חישובים מתימטים סימבולים, מערכות להיסק אוטומטי, הגדרה והוכחת נכונות של תוכניות,שפות תכנות מתקדמות ועוד.

שיכתוב *Rewriting* , הינו ענף העוסק בחישוביות באמצעות משוואות. השיכתוב מתבצע באמצעות משוואות מכוונות הנקראות כללי שכתוב. באמצעות כללי השיכתוב אנו מחליפים ביטויים בביטויים אחרים השווים להם, באמצעות המשוואות הנתונות אולם ההחלפה נעשית תמיד בכיוון אחד.

נושא מרכזי הקשור לשיכתוב הינו הביטוי הנורמלי *Normal Form* , ביטוי אשר לא ניתן לשכתב אותו יותר. החישוביות באמצעות שיכתוב מתבססת על ההגעה לביטוי נורמלי. כאשר הביטוי הנורמלי המתקבל מכל צורת שכתוב אפשרית הינו יחיד, הביטוי מהווה את ערכו של הביטוי הראשוני.

מערכת שיכתוב המשכתבת כל בטוי לביטוי נורמלי יחיד הינה מתכנסת *convergent* . במקרה כזה ניתן להשתמש במערכת השיכתוב ככלי לקבלת החלטה האם זהות מסויימת נכונה במערכת משוואות המתאימה למערכת השיכתוב.

לעיתים , יש בידינו מערכת שכתוב המתאימה למערכת משוואות מסויימת אולם מערכת השכתוב הינה חלקית בלבד ואינה מתכנסת, כלומר , מערכת השיכתוב אינה כוללת את כל חוקי השכתוב הנדרשים. כדי לקבל (אם אפשר) מערכת שכתוב מתכנסת הוצע על ידי

Knuth and Bendix תהליך השלמה *Completion* המאפשר להוסיף חוקי שכתוב חדשים.

תהליך ההשלמה (אם מסתיים בהצלחה) מוצא את כל כללי השכתוב החסרים והופך את מערכת השכתוב למערכת מתכנסת.

תהליך ההשלמה מתבסס על תהליך "דו צדדי" של הוספת חוקי שכתוב חדשים המתקבלים על ידי מציאת "זוגות קריטיים" *Critical Pairs* מצד אחד, וביצוע פישוט עד למחיקת חוקים קיימים כאשר מתקבלות זהויות באמצעות פישוט.

החסרון העיקרי בתהליך ההפשטה הינו העובדה שלעיתים התהליך מייצר מספר אינסופי של חוקי שכתוב חדשים ואינו מסתיים.

שימוש נוסף לתהליך ההשלמה הוצע ע"י Dershiwitz להוכחת משפטים.
ההוכחה מתבצעת בשיטה הקרויה "הוכחה באמצעות בדיקת עקביות" *Prove By Consistency* . השיטה מתבססת על בדיקת והשוואת מערכת השיכתוב הנתונה עבור מערכת המשוואות הבסיסית, אל מול מערכת שכתוב חדשה הנוצרת מביצוע תהליך השלמה על מערכת השכתוב הנתונה בצירוף המשוואה אותה רוצים להוכיח. ההוכחה מתבצעת על ידי בדיקת החוקים החדשים שהתוספו למערכת השכתוב הנתונה. במידה וכל מופע סגור של (צד שמאל של) החוקים החדשים ניתן לשכתוב על ידי המערכת המקורית , אזי המשוואה או המשפט אותו רוצים להוכיח אכן נכונים.
בעבודה זו נראה כיצד ניתן להשתמש בטכניקות של שיכתוב והשלמה לצורך ביצוע סינתיזה של תוכניות. התוכניות אותן נרצה לסנתז הינו תוכניות פונקציונאליות ובדרך כלל הן מבוססות על מבני נתונים בעלי מאפיינים ריקורסיביים כגון : מספרים טבעיים , רשימות, עצים וכד'.
הרעיון העיקרי מתבסס על שימוש במאגר או בסיס נתונים של תוכניות מוכרות, ידועות ומוכחות. מטרתינו הינה לסנתז באופן אוטומטי תוכנית חדשה. את התוכנית החדשה נגדיר באמצעות משוואות המתארות את המאפיינים אותם אמורה לבצע התוכנית. תהליך הסנתיזה אמור לייצר באופן אוטומטי תוכנית יעילה העונה למאפיינים שהוגדרו.
בעבודה זו השתמשנו כאמור בטכניקות של שכתוב והשלמה לצורך יצירת כללי שכתוב חדשים. כאשר אנו מוצאים מספיק כללי שכתוב יעילים המהווים תוכנית עבור האיפיון שהוגדר, אנו מקבלים תוכנית שנוצרה באופן אוטומטי.
כפי שצויין, הבעייה העיקרית בשימוש בתהליך השלמה נעוצה בעובדה שבמקרים רבים התהליך מייצר מספר אינסופי של חוקי שכתוב.
כדי להתגבר על הבעייה , מוצע כאן להשתמש ביוריסטיקות כדי לעצור את התהליך.
היוריסטיקות בהן השתמשנו :

זיהוי של חוקים בעלי תבנית דומה לצורך ניחוש של משוואה (משפט) כללית.
ביצוע השלמה בין חוקים סגורים *Ground Terms* שנוצרו בתהליך הסינתיזה ומציאת חוק חדש כללי המונע המשך ייצור אינסופי של חוקים מאותו סוג.
זיהוי ביטויים בעלי תבנית דומה לצורך החלפתם בצורה אוטומטית או כהמלצה להתערבות חיצונית.

היוריסטיקות מוכנסות לתוך תהליך ההשלמה ומונעות ביצוע אינסופי של התהליך.
כמובן , ששימוש ביוריסטיקות וניחוש משפטים מחייב הוכחה.
בעבודה זו השתמשנו בהוכחה על פי בדיקת עקביות המתבססת גם כן על תהליכים של שכתוב השלמה והכנסת יוריסטיקות כך שלמעשה אותם העקרונות משמשים גם לסינתיזה וגם להוכחה.

במסגרת העבודה נכתבה מערכת בשפת ML המבצעת תהליך של השלמה המתוגברת ביוריסטיקות שונות.

המערכת מסיקה חוקים חדשים באמצעות מנוע השלמה קלאסי . אולם , בתוך תהליך ההשלמה מתבצעים תהליכים של זיהוי תבניות וניחוש משפטים. המערכת מנסה להוכיח את נכונות המשפטים ומוסיפה כחוק חדש כל משפט שהוכח.

לצורך סיום התהליך המערכת בודקת האם החוקים שנוצרו עבור התוכנית המסונתזת מכסים את כל תחום הקלט הנדרש.

2. סקירה

בבסיס העבודה עומדת עבודתם ותרומתם החשובה של Knuth & Bendix,1970 אשר פיתחו את התאורייה והאלגוריתם עבור תהליך השלמה *Completion* .

תהליך ההשלמה נועד למצוא מערכת שיכתוב הדירה *Confluent* ומסיימת *Terminating*.

מערכת כזו, הנקראת מתכנסת *Convergent* , מוצאת לכל ביטוי , ביטוי נורמלי יחיד , לאחר מספר סופי של שיכתובים או הפעלה של חוקי שכתוב.

אלגוריתם ההשלמה מוצא את כל הזוגות הקריטיים עבור מערכת שכתוב מסיימת נתונה, ובודק האם ניתן לשכתב את שני חלקי הזוג לביטויים זהים. אם לא, מוסיפים את הזוג הקריטי תוך קביעת הכיוון כחוק נוסף למערכת השכתוב.

במידה והתהליך מסיים בהצלחה, התוצאה המתקבלת הינה מערכת שכתוב כנדרש. אולם, יתכן כי האלגוריתם יכשל או לא יסתיים.

מטרתם של Knuth & Bendix בפיתוח תהליך ההשלמה היתה למצוא שיטה בעזרתה ניתן יהיה לפתור את "בעיית המילה" *The Word Problem* כלומר , האם זהות מסויימת נכונה במרחב של מערכת משוואות נתונה.

תהליך ההשלמה, במידה והצליח, סיפק מערכת שיכתוב מתכנסת המתאימה למערכת המשוואות הנתונה. במקרה כזה, כל שנותר הוא להפעיל את מערכת השכתוב על שני חלקי המשוואה, ולבדוק האם הביטוי הנורמלי שהתקבל מחלקה הימני של הזהות, שווה בדיוק לביטוי הנורמאלי שהתקבל משכתוב חלקה השמאלי של הזהות.

תהליך ההשלמה נחקר בצורה רחבה Dershowitz[1989] , Bellegarde[1994] והוצע להשתמש בו במספר תחומים וביניהם עבור סינתיזה אוטומטית של תוכניות והוכחה אוטמטית של משפטים באמצעות בדיקת עקביות *Proof by Consistency* .

נושא מקביל ודומה לסינתיזה אוטומטית של תוכניות נחקר גם כן ומאמר מפתח בנושא הינו של Burstall & Darlinfton [1977] המתאר כיצד ניתן לייעל ולשפר את תוכניות פונקציונאליות באמצעות טרנספורמציות המתבססות על שימוש בכללי היסק מתוך מערכת המשוואות הנתונה.

כללי ההיסק נקראים Fold ו UnFold. הנושא מתואר גם ב Secher [2001] . במאמר זה מתואר כיצד ניתן לפשט ביטוי בתוכנית או בספציפיקציה ע"י התאמתו לצד שמאל של משוואה והחלפתו בצד ימין *UnFold* , או , ע"י התאמתו לצד ימין של משוואה והחלפתו בצד שמאל *Fold* .

התהליך בכללותו כולל ששה חוקים Bellegarde[1994] המכונים :

Definition, Instantiation, Unfolding, Folding, Abstraction, Law.

באמצעות טרנספורמציות על הביטויים בתוכנית ניתן לזהות ולמנוע חישוב מיותר של מבני נתונים זמניים , חישוב מרובה של אותו ביטוי וצורות נוספות של חוסר יעילות.

ב Dershowitz and Pinchover [1990] , Reddy[1989] , Dershowitz[1989] מתוארת גישה שונה לסינטיזה אוטומטית של תוכניות.

בגישה זו, קיימת מערכת שיכתוב או תוכניות פונקציונאליות עבור פונקציות בסיסיות. תוכניות אלו הינן שלמות ומסיימות *Ground Convergent* . בנוסף, אנו מגדירים את הפונקצייה אותה ברצונינו לסנתז באמצעות הפונקציות הקיימות. הגדרה זו אינה מהווה מערכת שכתוב שלימה או תוכנית פונקציונאלית המאפשרת בצורה קלה לחשב את הפונקצייה לכל קלט. במאמרים אלו מתוארת שיטה לביצוע סינתיזה אוטומטית לפונקצייה תוך שימוש בהשלמה.

תהליך ההשלמה בין הספציפיקצייה ומערכת השכתוב הנתונה מייצר מערכת שכתוב חדשה אשר (בין יתר החוקים שמתקבלים) כוללת את החוקים הנדרשים עבור הפונקצייה המוגדרת.

בעבודה זו הרחבנו את הסינתיזה בשני תחומים:

הוספנו למנגנון ההשלמה מערכת של יוריסטיקות וזיהוי תבניות.

שימוש ביוריסטיקות ובזיהוי תבניות חוזרות מאפשר לסיים תהליכי השלמה אינסופיים.

בנוסף , האינטואיציה בניחוש מבוססת גם על העובדה שאם קיימת תוכנית (או חוקי שיכתוב), ניתן לנחש אותם מתוך בדיקה של מספר מועט של חוקים המתקבלים בתהליך ההשלמה.

את החוקים המתקבלים על ידי ניחוש, נוכיח באותם כלים של השלמה ובדיקת החוקים המתקבלים בהשלמה נוספת זו האם הם עקביים למערכת הנתונה. על הוכחה באמצעות בדיקת עקביות ראה Bachmair L. and Dershowitz N. [1994] , Terese [1998] , Comon H. [2001] .

3. בסיס

בפרק זה נתאר מספר מונחים בסיסיים, משפטים וכלים בהם נשתמש בעבודה זו.

3.1. מבוא

התחום בו נעשתה עבודה זו הנו התחום של מערכות שכתוב,
(Term Rewriting Systems) TRS.
שכתוב, הינו תאורייה העוסקת בשינויים דיסקרטיים, הנעשים בשלבים, צעד צעד, על אובייקטים. מכיוון שהחישובים המתבצעים במדעי המחשב מאופיינים גם כן כסדרה של טרנספורמציות דיסקרטיות של אובייקטים, התאוריות והידע בתחום השכתוב מהווה יסוד חשוב במדעי המחשב בכלל. בין יתר הנושאים החשובים הקשורים לשכתוב ניתן למנות את : מימוש של תכנות פונקציונאלי, חישוב באמצעות משוואות, אוטומטיזציה של מערכות היסק, מערכות להוכחה אוטומטית של משפטים ועוד.
שכתוב הינו תחום ותיק ואף קודם למדעי המחשב. שורשיו הינם בתחום המתימטיקה והלוגיקה המתימטית. העיסוק בתחום השכתוב החל בשנות ה 30 של המאה הקודמת וקשור להופעתן של λ-calculus וקומבינטוריקה לוגית CL, תיאוריות המהוות את הבסיס לחישוביות.
בספרות ובאינטרנט קיים חומר רב על נושא השכתוב ומערכות TRS . נזכיר כאן מספר ספרים ומאמרים המהווים חלק מהחומר הרב הקיים בנושא:

הספר Nipkow Term Rewriting and All That [Baader,1998]
ספר נוסף על מערכות שכתוב *Term Rewriting Systems* [Terese,1998]
מאמר Term rewriting systems: a tutorial [Klop,1987]
מאמר Rewrite systems [Dershowitz,1990]
מאמר Inductionless Induction (Prove by Consistency) Comon, [2001]

במבוא זה נציג סיכום של מספר מושגים, מונחים ומשפטיבתחומים הרלוונטים.
את התיאור המלא והפורמאלי ניתן למצוא בספרים ובמאמרים הנ"ל.

3.2. מונחים בסיסיים - Abstract Reduction System ARS

כעיקרון,אנו נשתמש בעבודה זו במינוחים ובסימנים המקובלים בספרות הקיימת בתחום .
הפעולה הבסיסית בנושא שיכתוב הינה הרדוקצייה.
המונח רדוקצייה מבטא שם נרדף לפעולה של מעבר בסיסי, לצעד דיסקרטי הפועל על אובייקט (ביטוי) מסוים.

3.2.1. הגדרה.

ARS הינו זוג $(A,\rightarrow)$, כאשר הרדוקצייה $\rightarrow$ הינה יחס בינארי על הקבוצה A .
כלומר , $\rightarrow \subseteq A*A$.
במקום לכתוב $(a,b)\subseteq \rightarrow$, נכתוב $a \rightarrow b$.

המונח "רדוקצייה" (הפחתה, צמצום) נבחר, כיוון שבמקרים רבים (לא בכולם) , האובייקט עליו פועלת הרדוקצייה הולך וקטן בכל צעד עד לשלב בו לא ניתן להפעיל על האובייקט רדוקצייה נוספת.
הרדוקצייה הינה הפעולה הנמצאת במהותן של מערכות השכתוב.
סדרת רדוקציות ביחס ל $\rightarrow$ הינה סדרה סופית או אינסופית מהצורה:

$$a_0 \rightarrow a_1 \rightarrow a_2 \rightarrow \cdots$$

ותקרא נתיב רדוקציות או רדוקציות.

3.2.2. הגדרה.

יהיו R,S שתי רלציות $R\subseteq A*B$ ו $S\subseteq B*C$, החיבור שלהן יהיה :

$$R\circ S := \{(x,z\}\in A*C \mid \exists y\in B.\ \ (x,y)\in R \wedge (y,z)\in S\}$$

אנו מעונינים במיוחד בחיבור של רלצית הרדוקציה עם עצמה וביחסי הסגור,
נגדיר את המינוחים הבאים:

זהות $\overset{0}{\rightarrow} := \{(x,x) \mid x \in A\}$

הפעלת סדרה של רדוקציות $i \geq 0$ $\overset{i+1}{\rightarrow} := \overset{i}{\rightarrow} \circ \rightarrow$

הסגור הטרנזיטיבי $\overset{+}{\rightarrow} := \bigcup_{i>0} \overset{i}{\rightarrow}$

הסגור הרפלכסיבי טרנזיטיבי $\overset{*}{\rightarrow} := \overset{+}{\rightarrow} \bigcup \overset{0}{\rightarrow}$

הסגור הרפלכסיבי $\overset{=}{\rightarrow} := \rightarrow \bigcup \overset{0}{\rightarrow}$

הופכי inverse $\overset{-1}{\rightarrow} := \{(y,x) \mid x \rightarrow y\}$

הופכי inverse $\leftarrow := \overset{-1}{\rightarrow}$

הסגור הסימטרי $\leftrightarrow := \rightarrow \bigcup \leftarrow$

הסגור הטרנזיטיבי סימטרי $\overset{+}{\leftrightarrow} := (\leftrightarrow)^{+}$

הסגור הרפלכסיבי טרנזיטיבי סימטרי $\overset{+}{\leftrightarrow} := (\leftrightarrow)^{+}$

ניתן להתייחס אל ההגדרות הבאות גם במשמעות הבאה:

$x \overset{n}{\rightarrow} y$ אם קיים נתיב (רדוקציות) באורך n מ x ל y.

$x \overset{*}{\rightarrow} y$ אם קיים נתיב (רדוקציות) בעל אורך סופי מ x ל y.

נוסיף את המינוחים הבאים להגדרות הנ"ל:

x ניתן לצמצום (reducible) אם"ם קיים y כך שמתקיים $x \to y$

x בצורה נורמלית (normal form - irreducible) אם"ם x אינו reducible .

y הוא צורה נורמלית של x אם"ם $x \overset{*}{\to} y$ ו y בצורה נורמלית. אם ל x יש צורה נורמלית יחידה נסמן אותה $x \downarrow$.

x ו y מתאחדים (joinable) אם"ם קיים z כך שמתקיים $y \overset{*}{\leftarrow} z \overset{*}{\to} x$. את התוצאה אנו מסמנים $x \downarrow y$.

3.2.3. הגדרה.

יחס רדוקצייה $\to$ יקרא:

Rosser Church (CR) אם"ם $x \overset{*}{\leftrightarrow} y \Rightarrow x \downarrow y$

Confluent אם"ם $y_1 \overset{*}{\leftarrow} x \overset{*}{\to} y_2 \Rightarrow y_1 \downarrow y_2$

Locally Confluent(WCR-Weakly CR) אם"ם

$y_1 \leftarrow x \to y_2 \Rightarrow y_1 \downarrow y_2$

מסיים terminating אם"ם לא קיים $a_0 \to a_1 \to a_2 \to \cdots$ אינסופי.

normalizing אם לכל ביטוי יש צורה נורמלית.

convergent אם $\to$ הוא Confluent וגם terminating .

3.3. ביטויים והצבות

3.3.1. הגדרה.

חתימה Σ היא קבוצה של סמלי פנקציות, כאשר לכל $f \in \Sigma$ משוייך מספר טבעי n שיקרא ה ארי ה (arity) של f. לכל $n \geq 0$, אנו מציינים את קבוצת הסמלים בעלי ארי ה n ששייכים ל Σ בתור Σ^n. לאיברים של Σ^0 קוראים קבועים.

3.3.2. הגדרה

תהי Σ חתימה ותהי X קבוצה של משתנים (variables) כך שמתקיים $\Sigma \cap X = \varnothing$.
הקבוצה $T(\Sigma, X)$ של ביטויי Σ מעל X מוגדרת אינדוקטיבית כדלהלן:

$X \in T(\Sigma, X)$, כלומר, כל משתנה הוא ביטוי.

לכל $n \geq 0$, לכל $f \in \sum^{(n)}$ ולכל $t_1, \ldots, t_n \in T(\Sigma, X)$ קיים $f(t_1, \ldots, t_n) \in T(\Sigma, X)$ כלומר, השמה של פונקצייה על ביטויים יוצרת ביטוי.

3.3.3. הגדרה.

תהי Σ חתימה ותהי X קבוצה של משתנים (variables) כך שמתקיים $\Sigma \cap X = \varnothing$ וביטויים $s, t \in T(\Sigma, X)$.
קבוצת המיקומים (positions) של הביטוי s הוא קבוצה $Pos(s)$ של מחרוזות מעל האלפבית של הספרות ומוגדר אינדוקטיבית כדלהלן:

עבור $s = x \in X$ אזי $Pos(s) := \{\varepsilon\}$ כאשר ε מציין מחרוזת ריקה.

עבור $s = f(s_1,\ldots,s_n)$ אזי $Pos(s) := \{\varepsilon\} \cup \bigcup_{i=1}^{n} \{ip \mid p \in Pos(s_u)\}$

עבור $p \in Pos(s)$, תת הביטוי (subterm) של s במיקום p , הצויין כ $s|_p$ מוגדר אינדוקטיבית על האורך של p :

$s|_\varepsilon := s$

$f(s_1,\ldots,s_n)|_{iq} := s_i|_q$

עבור $p \in Pos(s)$, נציין ב $s[t]|_p$ את הביטוי המתקבל מ s כאשר מחליפים את תת הביטוי שנמצא במיקום p ע"י t.

ב $Var(s)$ נציין את קבוצת כל המשתנים הנמצאים בביטוי s.

3.3.4. הגדרה.

תהי Σ חתימה ותהי X קבוצה של משתנים (variables) כך שמתקיים $\Sigma \cap X = \varnothing$.
הביטוי $t \in T(\Sigma, X)$ יקרא סגור (ground) אם"ם $Var(t) = \varnothing$

ביטוי סגור הוא ביטוי שאינו מכיל משתנים.

3.3.5. הגדרה.

תהי Σ חתימה ותהי V קבוצה של משתנים . הצבת $T(\Sigma,V)$ או בקיצור , הצבה (substitution) (אם הקבוצה של הביטויים אינה רלוונטית או ברורה מההקשר) היא פונקצייה $\sigma : V \to T(\Sigma, X)$
כך ש $\sigma(x) \neq x$, לקבוצה סופית של משתנים. קבוצת המשתנים הסופית אותם σ לא ממפה לעצמם נקראת התחום של σ. עבור אותם משתנים נוכל לכתוב את σ בצורה הבאה:
$\sigma = \{x_1 \to \sigma(x_1),\ldots,x_n \to \sigma(x_n)\}$

קבוצת כל ההצבות תקרא Sub.

ביטוי t יקרא מופע (instance) של ביטוי s אם"ם קיימת הצבה σ כך ש $\sigma(s) = t$.

3.3.6. הגדרה.

תהי Σ חתימה ותהי V קבוצה של משתנים (variables) כך שמתקיים $\Sigma \cap V = \varnothing$.

הזהות $\Sigma - identity$ הינה קבוצה של זוגות $(s,t) \in \rightarrow T(\Sigma,V) \times T(\Sigma,V)$.

זהויות יכתבו בצורה $s \approx t$.

3.3.7. הגדרה.

תהי E קבוצה של $\Sigma - identities$. יחס הרדוקצייה $\rightarrow_E \subseteq T(\Sigma,V) \times T(\Sigma,V)$ מוגדר:

$s \rightarrow_E t$ אם"ם :

$$\exists (l,r) \in E, p \in Pos(s), \sigma \in Sub. \quad s|_p = \sigma(l) \quad and \quad t = s[\sigma(r)]_p.$$

3.3.8. הגדרה.

הזהות $l \approx r$ תקרא חוק שיכתוב אם"ם:

l אינו משתנה

מתקיים $Var(l) \supseteq \text{var}(r)$

במקרה כזה נכתוב $l \rightarrow r$.

מערכת שכתוב (TRS) Term rewriting system היא קבוצה של חוקי שכתוב.

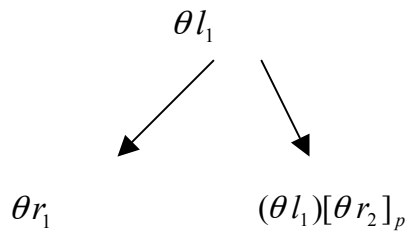
יחס הרדוקצייה $\rightarrow_R$ הינו היחס המושרה על ידי מערכת השכתוב R

3.4. זוגות קריטיים

3.4.1. זוג קריטי (critical pair)

יהיו $l_i \to r_i,\ i=1,2$ שני כללי שכתוב כלשהם אשר שמות המשתנים שלהם שוכתבו כך ש $Var(l_1,r_1)\cap Var(l_2,r_2)=\varnothing$. יהי $p\in Pos(l_1)$ כזה שמתקיים $l_1|_p$ אינו משתנה, ותהי θ mgu של $[\ l_1|_p\ ,\ l_2\ ]$.

מצב זה מגדיר זוג קריטי $\langle \theta r_1, (\theta l_1)[\theta r_2]_p \rangle$



במקרה כזה, שבו לשני חוקים יש זוג קריטי, אנו אומרים שהם חופפים.

3.4.2. לימה. (Critical Pair Lemma)

אם $s \to_R t_i,\ i=1,2$, אזי מתקיימת אחת משתי האפשרויות הבאות:

$t_1 \downarrow t_2$ או $t_i = s[u_i]_p, i=1,2$

כאשר $\langle u_1, u_2 \rangle$ או $\langle u_2, u_1 \rangle$ הינם זוג קריטי של R.

3.4.3. משפט הזוג הקריטי (Critical Pair Theorem)

מערכת שכתוב R , הינה בעלת התכונה WCR (local confluent) אם ורק אם כל הזוגות הקריטיים ב R מתאחדים (convergent)

3.4.4. לימת ניומן Newman's Lemma

מערכת שכתוב מסיימת הינה Confluent אם היא Local Confluent .

התוצאה ממשפט הזוג הקריטי ולימת ניומן הינו המשפט הבא המהווה בסיס לתהליך ההשלמה:

3.4.5. משפט

תהי R מערכת שכתוב בעלת תכונת SN (strongly normalizing) .
R הינה Confluent אם"ם כל הזוגות הקריטיים ב R מתאחדים (convergent) .

3.4.6. משפט Dershowitz N.[1989]

נניח שקיימת מערכת שכתוב עבור פונקצייה כלשהי f , הפונקציה f מוגדרת על ידי משוואות המתאימות למערכת משוואות E, והפונקציה f מוגדרת בתוך יחס סדר רלוונטי, אזי, התוצאה של תהליך השלמה הוגן על E יכיל מערכת שיכתוב עבור f .

משפט זה מבסס את השימוש בתהליך השלמה עבור ביצוע סינתיזה.

3.5. השלמה (Completion) .

תהליך , בו אנו מנסים להשלים מערכת שכתוב R (אם אמנם קיימת מערכת שלימה), עבור מערכת משוואות E , כך שהתוצאה המתקבלת הינה מערכת שיכתוב מתכנסת Convergent השקולה למערכת המשוואות הנתונה E.

תהליך ההשלמה הוגדר לראשונה באלגוריתם של Knuth-Bendix
. Knuth Bendix [1970]

תהליך ההשלמה מתבצע ע"י מציאת זוגות קריטיים, ביצועפישוט של החוקים החדשים ומחיקת זהויות.

3.5.1. משפט.

אם האלגוריתם של Knuth-Bendix מסתיים בהצלחה , המערכת המושלמת Rc , הינה מערכת שיכתוב שלמה עבור התאורייה של מערכת המשוואות E.

3.5.2. הגדרה.

מערכת שכתוב R נקראת irreducible אם לכל חוק של R $l \rightarrow r$ מתקיים:
r הוא ביטוי בצורה נורמלית ביחס ל R.
l הוא ביטוי נורמלי ביחס $R - \{ l \rightarrow r \}$.

3.6. הוכחה אינדוקטיבית באמצעות עקביות.

Inductive proofs by consistency .

במסגרת עבודה זו אנו זקוקים ליכולת הוכחה אוטומטית.
בעבודה זו בחרנו להשתמש בשיטה הנקראת proof by consistency .
הסיבה לבחירה זו הינה בהתאמתה לנושא העבודה סינתיזה אוטומטית של תוכניות, כיוון ששיטה זו מתבססת על אותם עקרונות בהן אנו משתמשים לצורך הסינתיזה.

3.6.1. הגדרה.

תהי (Σ, E) מערכת משוואות ו $s,t \in Ter(\Sigma)$ אזי

1. נאמר ש s=t הינו משפט אינדוקטיבי של E אם $I(\Sigma, E) \models s=t$
כלומר מתקיים $s\sigma \overset{*}{\leftrightarrow}_E t\sigma$ לכל הצבה סגורה $s\sigma \approx t\sigma$.
2. תהי (Σ, R) מערכת שכתוב עבור (Σ, E) .
המשוואה s=t הינה עקבית ביחס ל E (*consistent with* E) אם לכל ביטוי סגור ונורמלי תחת R (all closed (ground) R- normal forms) n , n' מתקיים $(\Sigma, E \cup \{s = t\}) \vdash n = n' \Rightarrow n \equiv n'$

3.6.2. משפט. [Dershowitz,1986]

תהי (Σ, R) מערכת שכתוב סגורה ושלמה (*ground complete*) עבור מערכת המשוואות (Σ, E) . המשוואה $s = t$ הינה משפט אינדוקטיבי של E אם ורק אם $s = t$ הינה קונסיסטנטית עם E.

משפט זה יוצר את הקשר בין עקביות של משוואה למערכת E , לבין היותה משפט אינדוקטיבי של E.

3.6.3. הגדרה.

תהי (Σ, R) מערכת שכתוב TRS. הביטוי $t \in Ter(\Sigma)$ יקרא *ground R-reducible* אם לכל ההצבות הסגורות σ , t^{σ} אינה צורה נורמלית תחת R.
כלומר, ניתן לשכתב ביטוי סגור של t תחת R.

משפט.
תהי (Σ, E) מערכת משוואות ותהי (Σ, R) מערכת שכתוב סגורה שלמה (*ground complete* TRS) עבור (Σ, E) . בנוסף, יהיו $s, t \in Ter(\Sigma)$, ותהי (Σ, R') מערכת שכתוב שלימה סגורה עבור $\{ s = t \}$ כך ש $R \subseteq R'$.
אזי, המשוואה $s = t$ הינה קונסיסטנטית עם מערכת המשוואות E אם ורק אם לכל כלל שכתוב חדש $l \to r \in R' - R$, l הינו *ground R-reducible*.

המשמעות של משפט זה הינה שניתן לבדוק קונסיסטנטיות של משוואה בדרך הבאה:
נניח שיש לנו מערכת שכתוב שלמה וסגורה (Σ, E) , עבור מערכת משוואות (Σ, E) (יתכן שמצאנו את המערכת בתהליך של השלמה). כדי להוכיח שהמשפט $s = t$ הינו קונסיסטנטי ב E , נפעיל את פרוצדורצת ההשלמה על המערכת R יחד עם המשוואה שברצונינו להוכיח s=t. אם התהליך מסתיים בהצלחה נקבל מערכת שכתוב חדשה (Σ, R').
לפי המשפט האחרון מספיק לבדוק אם כל צד שמאלי של חוק חדש שנמצא ב R' ולא נמצא ב R הינו *ground R-reducible*.

האינטואיציה העומדת מאחורי המשפט אומרת שאם ברצונינו להוכיח נכונות של משפט (משוואה) במסגרת של מערכת משוואות E, כלומר, המשוואה הינה משפט אינדוקטיבי של E , ולכן של R (שהיא מערכת סגורה ושלימה עבור E) , עלינו להראות שאם נמצא מערכת שכתוב שלימה

וסגורה שהיא איחוד של המערכת R (עבור E) ושל המשפט אותו רוצים להוכיח, אזי אם לא ניתן לשכתב במערכת החדשה ביטוי סגור נורמלי ב R , אזי המשפט נכון, כיוון שלא נוכל לקבל סתירה בין המערכת R למערכת השכתוב החדשה.

כל ביטוי סגור שניתן לשכתוב במערכת החדשה , ניתן לשכתוב זהה גם במערכת המקורי.

3.7. הפחתה לביטויים סגורים Ground Reducibility

כפי שראינו, כדי להשלים את ההוכחה באמצעות עקביות עלינו לבדוק את צד שמאל של החוקים החדשים שהתקבלו בתהליך ההשלמה ולקבוע האם הם Ground Reducibile במערכת השכתוב המקורית R.

נושא זה נחקר רבות ונסתפק בציון העובדה שקיים אלגוריתם למערכות שיכתוב שהן Left Linear.

לפירוט נוסף ראה [2001] ,Comon H. , Hofbauer D. To appear.

4. השיטה

4.1. משוואות, כללי שיכתוב ותכנות פונקציונאלי.

נקודת המוצא לביצוע סינתזה של תוכנית הינה קיומו של אוסף משוואות, המבטא ידע מסוים.
השימוש במשוואות נעשה בכמה היבטים:
המשוואות משמשות כבסיס ידע נתון המייצג עובדות ידועות ונתונות מראש.
המשוואות משמשות גם כמערכת שיכתוב. במצב זה אנו מתייחסים למשוואה $s = t$
כאל כלל שיכתוב $s \to t$.
המשוואות משמשות כתוכנית (פונקציונאלית).
למשל, נניח שנתונות לנו המשוואות הבאות :
כאשר : x, y הינם משתנים

s הינה פונקציה אונרית (עוקב)

$+$ הינה פונקציה בינארית (חיבור)

$$x + 0 = x$$
$$x + s(y) = s(x + y)$$

מערכת משוואות זו מבטאת למשל ידע לגבי פעולת חיבור על המספרים הטבעיים.
ניתן להתייחס למשוואות אלו כאל בסיס ידע נתון, ואשר המשוואות מבטאות בו את סך העובדות הנתונות והידועות לנו.
ניתן להתיחס למערכת המשוואות גם כאל כללי שכתוב :

$$x + 0 \to x$$
$$x + s(y) \to s(x + y)$$

בנוסף, ניתן לראות את אוסף המשוואות כתוכנית (פונקציונאלית) המממשת את פעולת החיבור על מספרים טבעיים.
לדוגמא , תוכנית פונקציונאלית (בשפת ML) למימוש פעולת חיבור עבור מספרים טבעיים. תוכנית זו מתבססת על המשוואות בתחילת העמוד.

לצורך המימוש בשפת ML הוגדר type nat המאפשר לייצג את המספרים הטבעיים בצורה הרקורסיבית הבאה:

```
datatype nat = z | s of nat
```

כאשר,

z הינו מספר טבעי מסוג nat (והוא מייצג את "0")

s(z) הינו המספר הטבעי העוקב של z .

```
fun plus x z = x
  | plus x (s y)  = s ( plus x y )  ;
```

4.2. תהליך הסינתיזה

4.2.1. השיטה, הבעיות והפתרונות המוצעים

נניח שברצוננו לסנתז תוכנית (מערכת שיכתוב) עבור פונקציה מסויימת $f(x_1,\ldots,x_n)$ בשלב הראשוני, אנו יודעים להגדיר (בקלות) את הפונקיה שעבורה ברצוננו ליצור תוכנית, באמצעות פונקציות אחרות מוגדרות מראש, ואשר עבורן יש בידינו תוכניות או מערכות שיכתוב יעילות.

הגדרה ראשונית זו מבטאת את תכונותיה ופעולתה של הפונקציה , אולם במקרים רבים אינה יכולה לשמש כתוכנית או מערכת שכתוב יעילה.

מטרתנו, לסנתז תוכנית עבור הפונקיה f , שתכיל ביטויים (terms) בסיסיים בלבד (primitive constructors).

אנו משתמשים בפרוצדורת השלמה (completion) אשר מבצעת השלמה בין ההגדרה הבסיסית הנתונה (specification) ובין מערכת המשוואות הנתונה עבור פונקציות ההגדרה.

למעשה, אנו משתמשים בהשלמה רחבה (paramodulation) המתבצעת כאשר הגדרות האפיון נלקחות בשני הכיוונים, כלומר מול $l \to r$, וגם מול $r \to l$.

בנוסף, נגדיר יחס סדר בין הביטויים השונים כך שניתן לכוון את כללי השכתוב המתקבלים.

על פי משפט Dershowitz N.[1989] תהליך השלמה הוגן זה ייצר בסופו של דבר אוסף חוקי שיכתוב אשר יכילו גם תוכנית עבור הפונקציה f .

הבעייה המעשית בשימוש בתהליך זה נובעת מהעובדה שאין בטחון שתהליך ההשלמה יסתיים, ובמקרים רבים התהליך מייצר מספר אינסופי של חוקי שכתוב.

הפתרון המוצע לבעיה בעבודה זו מתבסס על שימוש במודל מורחב של פרוצדורת ההשלמה.

ההרחבה תכלול שתי פעולות עיקריות תוך כדי ביצוע ההשלמה:

<u>ניחוש חוקים.</u>

ניחוש של חוקים (למות, משפטים). ניתן לנחש על ידי שילוב של יוריסטיקות שונות ועל בסיס הכללים שכבר נוצרו בתהליך ההשלמה עד לשלב הניחוש.

יוריסטיקות אפשריות:

מציאת חוקים יחודיים לפונקציה הנדרשת באמצעות השלמה פנימית.

מציאת תבניות כלליות על סמך דוגמאות של חוקים.

הפעלת היוריסטיקות וה"ניחוש" בתוך תהליך ההשלמה מיועד "לעקוף" את הבעייה של ייצור אינסופי של כללים ע"י תהליך השלמה שאינו מסתיים. מטרת הניחוש הינה לגלות חוקים על סמך ידע שנוצר בתהליך ההשלמה עד כה.
בהנחה שנוכל לגלות חוקים נכונים ולהוכיח אותם, נוכל למצוא את החוקים הדרושים עבור התוכנית הופנקציונאלית אותה אנו מנסים לסנתז.

הוכחת ההשערות.

ראינו בסעיף הקודם כי כדי למנוע פעולה איסופית של תהליך ההשלמה אנו מוסיפים לתהליך "מנוע ניחוש" שמנסה לגלות חוקים על סמך ידע שכבר נוצר.כמובן , שעלינו להוכיח כי ההשערות החדשות אכן נכונות.
השיטה בה השתמשנו בעבודה זו להוכחת ההשערות מתבססת על "הוכחה כתוצאה מעקביות" "prove by consistency".
בחרנו בשיטה זו כיוון שהיא מתבססת על "השלמה מורחבת".
כלומר, באמצעות אותם כלים בהם השתמשנו למציאת התוכנית וניחוש הכללים, אנו משתמשים להוכחת הכללים שהתגלו בתהליך של ניחוש והשערה, ואשר עדיין טעונים הוכחה.
בשיטה זו , הרעיון כפי שהוסבר בפרק הבסיס מתבסס על כך שהשערה מסויימת "נכונה" במערכת משוואות מסויימת אם ההשלמה של ההשערה עם מערכת המשוואות לא מייצרת כללי שכתוב אשר לא ניתנים לשכתוב במערכת המקורית.
כדי להוכיח בשיטה זו עלינו לבצע השלמה אולם שוב מתעוררת הבעייה של תהליך השלמה שמייצר אינסוף משוואות.
גם כאן, ננקוט באותה שיטה ממש כלומר, ננסה לנחש חוק חדש אשר "מכסה" מספר אינסופי של "מקרים פרטיים אשר נוצרים בתהליך ההשלמה. השערה תהיה נכונה כאשר אנו מגיעים למצב שבו לא קיימים זוגות קריטיים וכל הזוגות שנוצרו ניתנים לשכתוב (צד שמאל) במערכת המשוואות המקורית.

4.3. שלמות התוכנית המסונתזת.

בתהליך הסינתזה שתואר נוצרים הכללים אשר (חלקם) מהווים את מערכת השכתוב הנדרשת עבור הפונקציה f.

הסינתזה מסתיימת כאשר נוצרה קבוצה מספיקה של חוקים המכסה את כל תחום הקלט.

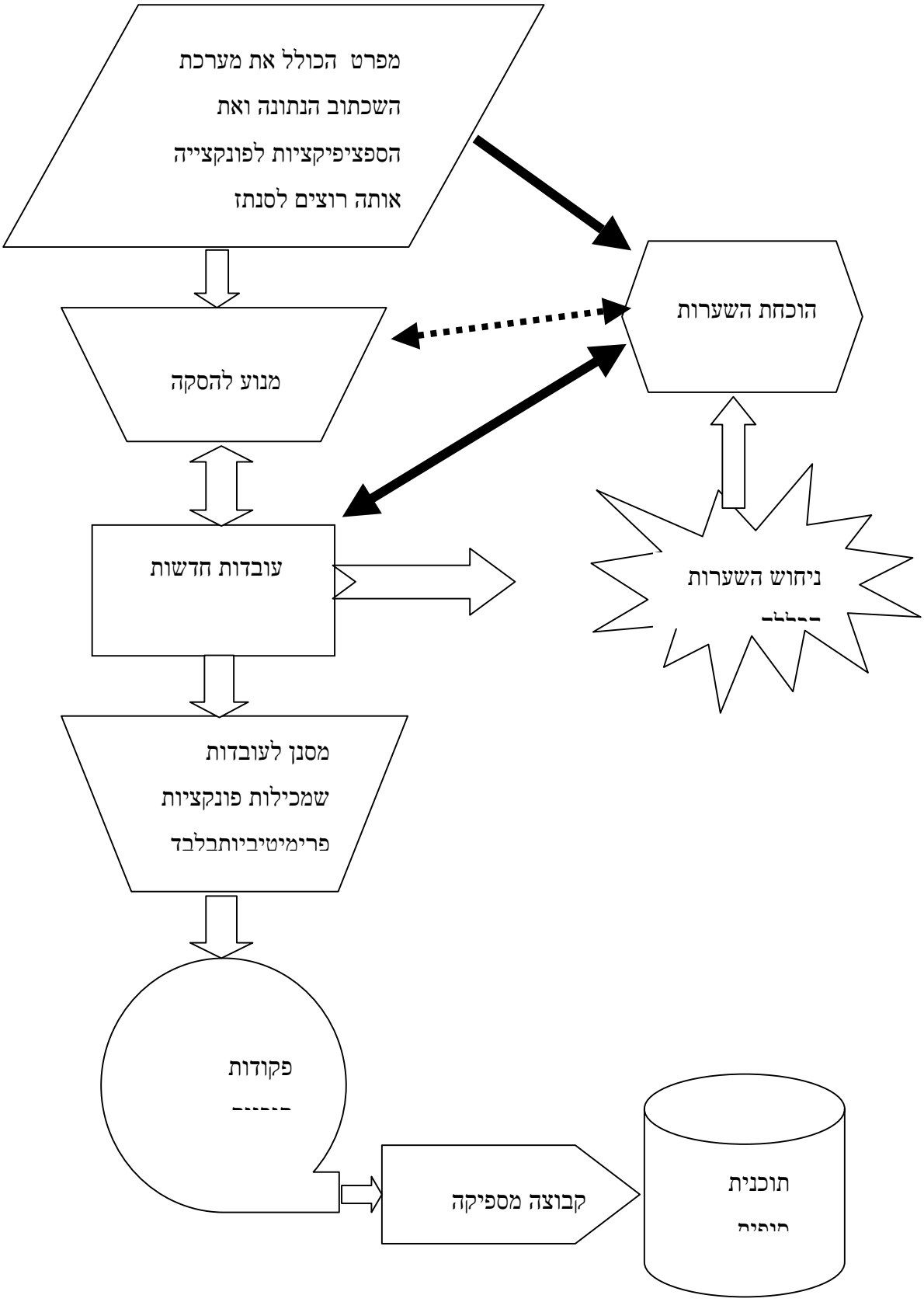
לצורך זה, יש להגדיר לכל תוכנית אותה רוצים לסנתז את כל תחום הקלט.

תחום הקלט מוגדר בצורה רקורסיבית ולכן ניתן להציגו כעץ המייצג את כל מרחב הקלט.

הכללים שנוצרו מהווים קבוצה מספקת לתוכנית אם הם מכסים את תת עץ הקלט מהשרש ועד לרמה כלשהיא בעץ בשלמותה.

5. מערכת

התהליך כולו מתואר בצורה סכימטית בתרשים הבא:



6. דוגמאות

6.1. דוגמא לסנתיזה של פונקצית הכפלה מעל המספרים הטבעיים

. $d(x) = x + x$

נניח שאנו מעונינים לסנתז תוכנית עבור פונקציה שמבצעת פעולת הכפלה על מספר טבעי.
נתונה לנו מערכת המשוואות עבור פעולת החיבור :

$x + 0 = x$

$x + s(y) = s(x + y)$

וכן המפרט (specification) עבור פונקצית ההכפלה (double) שתסומן d :

$d(x) = x + x$

ביצוע תהליך השלמה כנ"ל ייצר סדרה אינסופית של המשוואות הבאות :

$d(0) = 0$

$d(s(x)) = s(s(x) + x)$

$d(s(0)) = s(s(0)$

$d(s(s(x))) = s(s(s(s(x) + x))))$

$d(s(s(0))) = s(s(s(s((0))))$

.

.

.

כפי שניתן לראות, התהליך מייצר "תוכנית" בעלת מספר אינסופי של כללי שיכתוב.
אם נחלק את הכללים המתקבלים לאלה אשר לא ניתן לבצע עליהם השלמה נוספת ואלה שכן, נקבל את אוסף הכללים

$d(0) = 0$

$d(s(0)) = s(s(0)$

$d(s(s(0))) = s(s(s(s((0)))))$

.

.

.

אוסף כללים זה אכן מהווה "תוכנית" עבור הפונקציה d , אולם הוא אינו יותר מאשר לוח המכיל מספר אינסופי של מקרים אפשריים :

$\{ d(s^i(0) = s^{2i}(0) : i \geq 0 \}$

לצורך קבלת תוכנית שלמה, נדרש לגלות או לנחש את הכלל

$d(s(x)) = s(s(d(x))$

השיטה המוצעת בעבודה זו הנה ל"הרחיב" את פרוצדורת ההשלמה כך שתוך כדי תהליך ההשלמה יתבצעו תהליכים נוספים.

למשל, עבור סינתזה של התוכנית, אנו נבצע השלמה נוספת על הכללים שנוצרו ואשר מכילים כללים עם הפונקציה d בלבד.

השלמה כזו תייצר את הכללים הבאים:

$d(s(0)) = s(s(d(0))$

$d(s(s(0))) = s(s(d(s((0))))$

.

.

.

אם נתבונן בכללים החדשים שנוצרו בתהליך ההשלמה הנוסף, נגלה כי קיימת תבנית כללית עבור הכללים הנ"ל:

$d(s(0)) = s(s(d(0))$

אם נחליף את 0 , ■ <- 0 נקבל :

$d(s(■)) = s(s(d(■))$

$d(s(s(0))) = s(s(d(s((0))))$

אם נחליף את s(0) , ■ <- s(0) נקבל :

$d(s(■)) = s(s(d(■))$

אנו רואים כי עבור החוקים החדשים שהתקבלו קיימת תבנית כללית מהצורה :

$d(s(\blacksquare)) \quad = \quad s(s(d(\blacksquare))$

אשר היא בדיוק הכלל שהיה "חסר"

$d(s(x)) \quad = \quad s(s(d(x))$

התהליך בו אנו משתמשים לצורך גילוי התבנית הנו כזה שבו מוחלפים תת ביטויים זהים במשתנים זהים.

לתהליך זה קראנו most specific generalization - msg.

התהליך גם מכונה "איחוד שלילי" "anti unification".

האינטואיציה העומדת מאחורי תהליכים אלו נעוצה בעובדה שאנו מחפשים כללים אשר כוללים את הפונקציה המוגדרת גם בחלק הימני של הכלל.

הוכחת ההשערה.

לאחר שניחשנו את הכלל $d(s(x)) = s(s(d(x)))$ עלינו להוכיח את נכונותו.

הפעלת תהליך ההשלמה (יצירת זוגות קריטיים תוך כדי פישוט) יוצרת את החוקים הבאים:

$s(s(x) + x) \quad = \quad s(s(x+x))$

$s(s(s(s(x)) + x)) \quad = \quad s(s(s(s(x) + x)))$

ננחש שוב את החוק :

$s(x) + y \quad = \quad s(x + y)$

חוק זה מוכח ע"י עקביות.

חוק זה אינו מייצר זוגות קריטיים חדשים. לכן, כל שנותר לבדוק הוא האם כל הצבה סגורה של צד שמאל שלו, ניתנת לרדוקייה ע"י המערכת המקורית.

היות ותחום ההגדרה של התוכנית הינו המספרים הטבעיים כל ביטוי סגור יהיה עלינו לבדוק את הביטויים הבאים:

$s(0) + 0$

$s(0)+s(x)$

$s(s(x))+0$

$s(s(x))+s(y)$

מערכת השכתוב המקורית אכן משכתבת כל אחד מהביטויים הנ"ל.

הוספת החוק $s(x)+y=s(x+y)$ מסיימת את תהליך ההשלמה .

במצב זה המערכת כוללת את החוקים החדשים שנוצרו:

$d(0)=0$

$d(s(x)=s(s(d(x))$

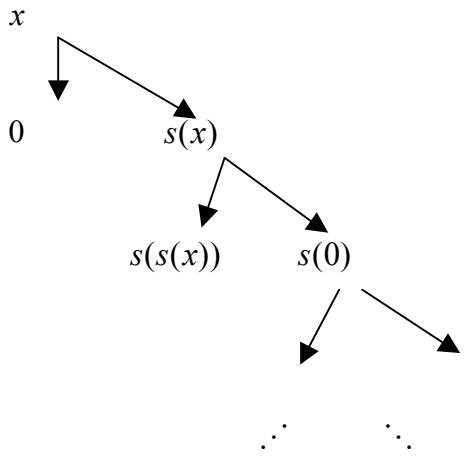
$s(x)+y=s(x+y)$

המערכת מסננת את החוקים כך שרק חוקים הכוללים ביטויים המורכבים מ $0,s,d$ עוברים כחוק שכתוב קביל למערכת המסונטזת.

כמובן שכאן ניתן לקבוע בצורה חיצונית (לתהליך האוטומטי) כללי קבלה של חוקים שנוצרו ע"י המערכת.

בשלב זה נותר לזהות קבוצה מספקת של חוקים המהווים תוכנית לפונקצייה d.

לצורך איתור קבוצת החוקים המספקת תוכנית עבור d, אנו בוחנים את העץ המכסה את התחום הרלוונטי, ובמקרה זה:



התוכנית :

$d(0)=0$

$d(s(x)=s(s(d(x))$

הינה שלימה כיוון שהיא מכסה את התחום המוגדר ע"י : $s(x)$, 0 .

6.2. דוגמא (I) לסנתיזה של פונקצית חילוק מעל המספרים הטבעיים

בדוגמא זו נבצע סינתיזה לתכנית שמבצעת פעולת חילוק על מספרים טבעיים.

גם בדוגמא זו נשתמש בתוכנית החיבור כמערכת השכתוב הבסיסית הנתונה:

$$x + 0 \rightarrow x$$

$$x + s(y) = s(x + y)$$

הספציפיקציה עבורת פעולת החילוק תוגדר באמצעות שתי המשוואות:

$$h(x + x) = x$$

$$h(s(x + x)) = x$$

תהליך ההשלמה (בתוך הסינתיזה) מתחיל לייצר את המשוואות הבאות:

$h(x + x) = x$	$h(s(x + x)) = x$
$h(0) = 0$	$h(s(0)) = 0$
$h(s\ (s(x) + x)\ ) = s(x)$	$h(s(s\ (s(x) + x)\ )) = s(x)$
$h(s(s(0))) = s(0)$	$h(s(s(s(0)))) = s(0)$
$h(s(s(\quad (s(s(x)) + x)\quad) = s(s(x))$	$h(s(s(s(\quad (s(s(x)) + x)\quad)) = s(s(x))$
$\vdots$	$\vdots$

על פי האיסטרטגייה הפועלת בתהליך הסינתיזה, המערכת "אוספת" את החוקים שלא ניתן לבצע עליהם השלמה נוספת (לא ניתן לייצר עבורם זוגות קריטיים) ובודקת אותם בנסיון למצוא כלל שכתוב חדש , ואכן מתוך בחינת החוקים:

$$h(0) = 0$$

$$h(s(0)) = 0$$

$$h(s(s(0))) = s(0)$$
$$h(s(s(s(0)))) = s(0)$$
$$h(s(s(s(s(0))))) = s(s(0))$$
$$\vdots$$

תהליך השלמה יוריסטיקת ההכללה על החוקים הנ"ל מייצר שתי השערות לחוקים חדשים:

$$h(s(x)) = h(x)$$
$$h(s(s(x))) = s(h(x))$$

ההשערה הראשונה נדחית כיוון שבתהליך ההוכחה אנו מפעילים את חוק הביטול ומקבלים

$$s(x) = x$$

את החוק השני אנו מוכיחים באמצעות בדיקת עקביות.

לצורך ההוכחה נחוץ לנו החוק לגבי קומוטטיביות החיבור $s(x) + y = s(x + y)$.

חוק זה ניתן להוסיף כחלק ממערכת חוקים (אקסיומות) ידועה ונתונה מראש. במקרה כזה, משתמש בחוקים אלה לצרכי פישוט ונירמול של ביטויים.

אפשרות נוספת הינה לזהות את התבנית החוזרת בתוך הביטויים המתקבלים בתהליך ההשלמה :

$$(s(x) + x)$$
$$(s(s(x)) + x)$$

אשר ניתן לנחש על פיהם את התבנית $s(x) + y$. במקרה כזה, המערכת תציע למשתמש לבדוק את האפשרות להוסיף למערכת אקסיומה שתשכתב את הבטויים הנ"ל.

בכל מקרה, לאחר הוספת החוק $s(x) + y = s(x + y)$, ההשלמה של ההשערה $h(s(s(x))) = s(h(x))$ עם המערכת הנתונה והספציפיקציות מסתיימת בהצלחה וללא הוספת חוקים נוספים.

לצורך השלמת ההוכחה יש לוודא שהחוק החדש הינו *Ground* Re*ducible* במערכת הנתונה.

6.3. דוגמא (II) לסנתיזה של פונקצית חילוק מעל המספרים הטבעיים

בדוגמא זו נבצע סינתיזה לתכנית שמבצעת פעולת חילוק על מספרים טבעיים.
הפעם, נשתמש בתוכנית ההכפלה כמערכת השכתוב הבסיסית הנתונה:

$$d(0) = 0$$
$$d(s(x) = s(s(d(x))$$

הספציפיקציה עבורת פעולת החילוק תוגדר באמצעות שתי המשוואות:

$$h(d(x)) = x$$
$$h(s(d(x))) = x$$

תהליך ההשלמה (בתוך הסינתיזה) מתחיל לייצר את המשוואות הבאות:

$h(d(x) = x$	$h(s(d(x)) = x$
$h(0) = 0$	$h(s(0)) = 0$
$h(s(s(d(x)))) = s(x)$	$h(s(s(s(d(x))))) = s(x)$
$h(s(s(0))) = s(0)$	$h(s(s(s(0)))) = s(0)$
$h(s(s(s(s(d(x)))))) = s(s(x))$	$h(s(s(s(s(s(d(x))))))) = s(s(x))$
⋮	⋮

היות והחוקים הסגורים שנוצרים כאן זהים לחוקים שנוצרו בדוגמא הקודמת, גם במקרה זה המערכת תנחש את ההשערה $h(s(s(x))) = s(h(x))$

מערכת השיכתוב הכוללת את החוקים הבאים :

$$d(0) = 0$$

$$d(s(x) = s(s(d(x))$$

$$h(d(x)) = x$$

$$h(s(d(x))) = x$$

$$h(s(s(x))) = s(h(x))$$

הינה שלמה .

החוק החדש נכון, כיוון שהוא מהווה חלק ממערכת שלימה וכל ביטוי סגור שלו ניתן לשכתוב גם במערכת המקורית.

7. תרומות ומסקנות.

בעבודה זו הוצגה יכולת לסנתז באופן אוטומטי, תוך שימוש במסגרת של משוואות ,לוגיקה משוואתית , מערכות שכתוב והשלמה מורחבת.

הסינתיזה רלוונטית עבור תוכניות פונקציונאליות הפועלות על מבני נתונים רקורסיביים כדוגמת המספרים הטבעיים , רשימות , עצים וכד'.

לצורך סינתיזה אוטומטית אנו משתמשים בשיטות דדוקטיביות ואינדוקטיביות.

ראינו כי כדי לאפשר סינתיזה אוטומטית בצורה מעשית, יש להשתמש ביוריסטיקות וניחושים אותם כמובן יש להוכיח.

בעבודה זו בצענו בצורה מוכללת סינתיזה מלאה הכוללת את כל השלבים מיצירת חוקים חדשים, הכללה וניחוש של חוקים חדשים , זיהוי תבניות חוזרות בתוך ביטויים שנוצרים בתהליך, הוכחה אוטומטית של משפטים אותם אנו משערים ,סינון וקבלה של התוכנית המוצעת.

בבואינו ליישם מערכות כלליות לסינתיזה אוטומטית של תוכניות, עלינו להתייחס למספר היבטים חשובים:

האם התוכנית שנוצרה נכונה במובן של שלימות וסיום?

איסטרטגיות ובקרה על יצירת המשוואות החדשות.

האם וכיצד ניתן להרחיב את הסינתיזה לתוכניות שאינן מבוטאות בצורת משוואות, למשל, משוואות מותנות, פסוקי הורן, פסוקים של לוגיקה מסדר ראשון ועוד.

בעבודה זו התמקדנו בסינתיזה של תוכניות פונקציונאליות המבוטאות כמשוואות.

השתמשנו בכלים של מערכות שכתוב והשלמה.

תרומתה של עבודה זו הינה בשילוב של מספר מתודות לכלל מערכת שלימה לסינתיזה אוטומטית.

היוריסטיקות שהוצעו בעבודה זו הכללה , ביצוע השלמה על תוצאות סגורות וזיהוי תבניות חוזרות, מתוות כיוון ואפשרויות להוספה של יוריסטיקות נוספות. למעשה המודל של הסינתיזה מאפשר להוסיף ולשנות את האסטרטגיות ליצירת החוקים וכן יוריסטיקות נוספות.

הנושא של סינתיזה אוטומטית של תוכניות נחקר בעיקר בהקשר של שיפור היעילות של תוכניות קיימות.

בעבודה זו הוצג לראשונה מודל שלם המאפשר לסנתז תוכנית עם ניחוש השערות והוכחתן, בדיוק כפי שאדם אינטלגנטי ניגש לכתוב תוכנית ולהוכיח השערות אותן הוא מניח על סמך ניסיון ואינטואיצייה.

כפי שראינו בעבודה זו, ניתן אכן לסנתז תוכנית שלימה ומוכחת בצורה אוטומטית.

כיוונים נוספים להמשך המחקר כוללים חקירת אסטרטגיות יעילות לביצוע הסינתיזה ושימוש ביוריסטיקות נוספות.

היכולת לבצע סינתיזה אוטומטית של תוכניות הינה מרתקת ובעלות השלכות שלא ניתן להגזים בערכן.

ביבליוגרפיה

Baader, F. and T. Nipkow [1998]. Term Rewriting and All That, Cambridge University Press.

Bachmair L. and Dershowitz N. [1994], `Equational inference, canonical proofs, and proof orderings', *J. of the Association for Computing Machinery* 41(2), 236-276.

Burstall R. M. and Darlington John[1977]. A transformation system for developing recursive programs. *Journal of the ACM*, 24(1):44-67

Comon H. [2001], Inductionless Induction, *in* A. Robinson and A. Voronkov, eds*,`Handbook of Automated Reasoning'*, Vol. I, Elsevier Science, chapter 14.

Dershowitz, N. [1986]. Applications of the Knuth-Bendix completion procedure, *in: Proceedings of the S_eminaire d'Informatique Th_eorique*, pp. 95-111.

Dershowitz N.[1989]. Completion and its applications. In *Resolution of Equations in Algebraic Structures*, volume 2: Rewriting Techniques, pages 31-86. Academic Press, San Diego.

Dershowitz Nachum and Pinchover Eli [1990]*. Inductive synthesis of equational programs*. *In* AAAI90, *Proceedings, Eighth National Conference on Artificial Intelligence*, pages 234-239. MIT Press.

Dershowitz N. and Jouannaud J.-P. [1990], Rewrite systems, *in* J. van Leeuwen, ed., *'Handbook of Theoretical Computer Science'*, Vol. B: Formal Methods and Semantics, NorthHolland, Amsterdam, chapter 6, pp. 243-320.

Dershowitz N., Kaplan S. and Plaisted D. A. [1991], 'Rewrite, rewrite, rewrite, rewrite,rewrite,. . . ', *Theoretical Computer Science* 83(1*),* 71-96.

Dershowitz N. and Reddy U. [1993], Deductive and inductive synthesis of equational programs, *J. Symbolic Computation* 15, pp. 467-494.

Dershowitz, N. and L. Vigneron [2001]. Rewriting Home Page, http://www.loria.fr/~vigneron/RewritingHP/.

Dershowitz N. and Plaisted D. [2001], Rewriting, *in* A. Robinson and A. Voronkov, eds*,`Handbook of Automated Reasoning'*, Vol. I, Elsevier Science, chapter 9.

Francoise Bellegarde[1994]. Automating synthesis by completion. *Technical Report CS/E 94-20*, Oregon Graduate Institute Of Science & Technology, Department of Computer Science and Engineering.

Hofbauer D. and Huber M. Test sets for the universal and existential closure of regular tree languages *in Information and Computation*, to appear.

Klop, J.W. [1987]. Term rewriting systems: a tutorial*, Bulletin of the European Association for Theoretical Computer Science 32*, pp. 143-183.

Klop, J.W. [1992]. Term rewriting systems, in: S. Abramsky, D.M. Gabbay and T.S.E. Maibaum (eds.), *Handbook of Logic in Computer Science*, Vol. 2, Oxford University Press, pp. 1-116.

Knuth, D.E. and P.B. Bendix [1970]. Simple word problems in universal algebra,*in*: J. Leech (ed.), *Computational Problems in Abstract Algebra*, Pergamon Press, pp. 263-297.

U. S. Reddy[1989]. Rewriting techniques for program synthesis. In *N. Dershowitz, editor,Rewriting Techniques and Applications,* pages 388-403. Springer-Verlag,. (LNCS Vol. 355).

Terese [1998]. Term Rewriting Systems.
URL: http://www.cs.vu.nl/~terese

J.P Secher [2001].Unfold/Fold Transformation, Unpublished Note,Department of Computer Science(DIKU), University of Copenhagen.

**Dershowitz, N. and M. Okada [1990]. A rationale for conditional equational pro-
gramming, Theoretical Computer Science 75, pp. 111{138.
Dershowitz, N., J.-P. Jouannaud and J.W. Klop [1993]. More problems in rewriting,
in: C. Kirchner (ed.), Proceedings of the Fifth Conference on Rewriting Tech-
niques and Applications (RTA '93), Lecture Notes in Computer Science 690,
Springer-Verlag, pp. 468{487. Included in the RTA list of open problems at
http://www.lri.fr/~rtaloop.
Dershowitz, N., L. Marcus and A. Tarlecki [1988]. Existence, uniqueness, and
construction of rewrite systems, SIAM Journal of Computing 17(4), pp. 629-639.
Dershowitz, N., M. Okada and G. Sivakumar [1988]. Canonical conditional rewrite
systems, in: E. Lusk and R. Overbeek (eds.), Proceedings of the Ninth Inter-
national Conference on Automated Deduction (CADE '88), Lecture Notes in
Computer Science 310, Springer-Verlag, pp. 538-549.**

**Dershowitz, N. [1987]. Termination of rewriting, Journal of Symbolic Computation
3(1), pp. 69-116. Corrigendum in 4(3), pp. 409-410.
Dershowitz, N. and S. Mitra [1993]. Path orderings for termination of associative-commutative rewriting, in: M. Rusinowitch and J.L. R_emy (eds.), Proceed-
Dershowitz N. and Plaisted D. [2001], Rewriting, *in* A. Robinson and A. Voronkov, eds*,`Handbook of Automated Reasoning'*, Vol. I, Elsevier Science, chapter 9.**

A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and

A. Smaill. Rippling: A heuristic for guiding inductive proofs.

Arti_cial Intelligence, 62(2):185{253, August 1993.

ings of the Third International Workshop on Conditional Term Rewriting Systems (CTRS '92), Lecture Notes in Computer Science 656, Springer-

Verlag, pp. 168{174.