<u>תרגיל בית 4 - תורת הקומפילציה</u>

<u>מגישות: סתיו אבירם ומיה אקהייזר</u>

<u>שאלה 1:</u>

סעיף א':

הדקדוק $G1$ ב-$LL(1)$ אם ורק אם לא קיים קונפליקט בדקדוק, כלומר לכל שני כללים בדקדוק עבור אותו המשתנה $A \to \alpha,\ A \to \beta$: $select\left(A \to \alpha\right) \cap select(A \to \beta) = \varnothing$. נרצה לחשב את תוצאת הפעלת פונקציית ה-$select()$ עבור כל כלל. מכיוון שעבור כל כלל צד ימין אינו אפיס, על מנת לחשב את פונקציית ה-$select()$ עלינו לחשב לשם כך רק את ה-$first()$ של הצורות הפסוקיות:

α	$lpar\ Binop\ rpar$	id	num	$add\ E\ E$	$sub\ E\ E$
$first(\alpha)$	$\{lpar\}$	$\{id\}$	$\{num\}$	$\{add\}$	$\{sub\}$

וכעת נחשב את פונקציית ה-$select()$:

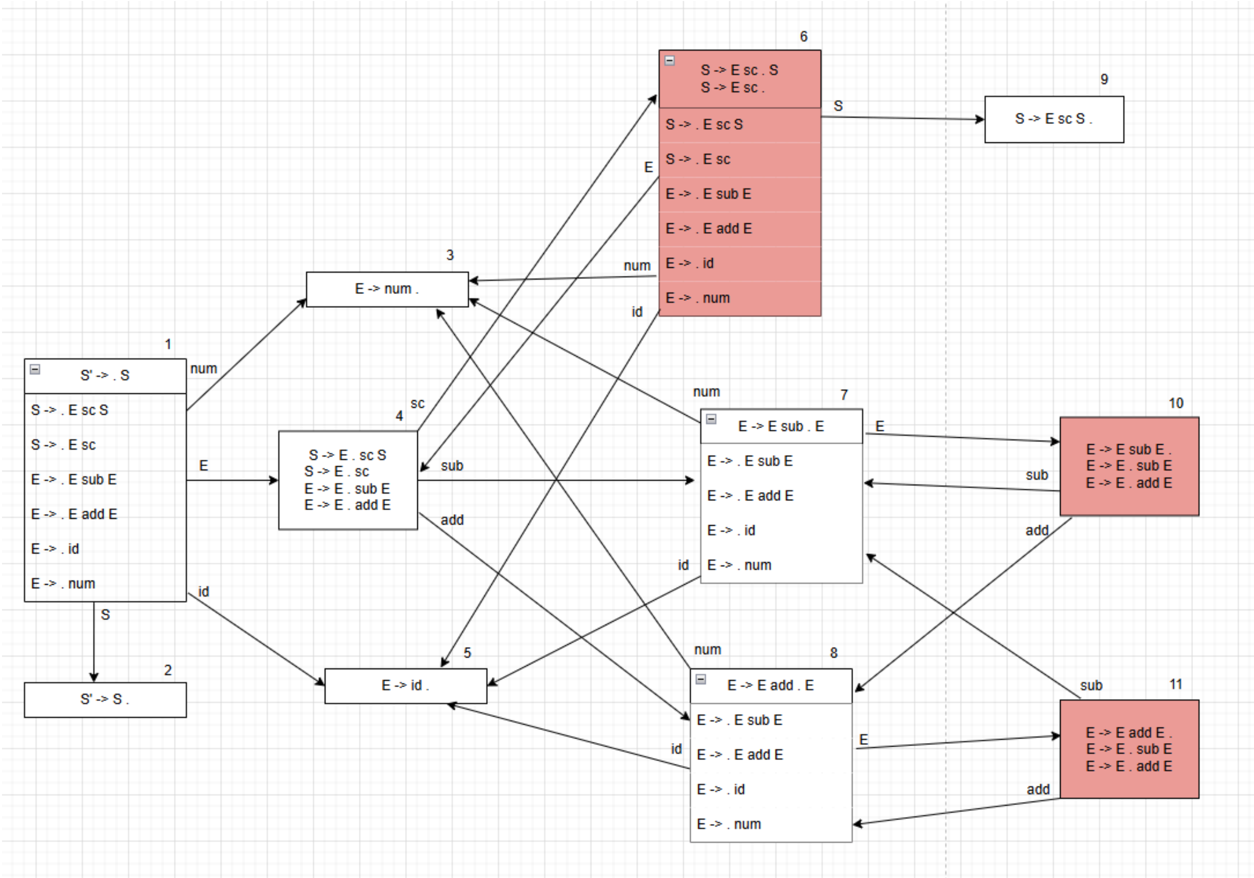
$rule$	$E \to lpar\ Binop\ rpar$	$E \to id$	$E \to num$	$Binop \to add\ E\ E$	$Binop \to sub\ E\ E$
$select$	$\{lpar\}$	$\{id\}$	$\{num\}$	$\{add\}$	$\{sub\}$

נבחין שגם עבור המשתנה E וגם עבור המשתנה $Binop$, הפעלת ה-$select()$ על כל כלל שלהם מייצרת קבוצות בהן איבר אחד ייחודי לכל כלל. לכן, לכל שני כללים בדקדוק עבור אותו המשתנה $A \to \alpha,\ A \to \beta$ מתקיים: $select\left(A \to \alpha\right) \cap select(A \to \beta) = \varnothing$.

מכאן ש**הדקדוק G_1 הוא ב-LL(1).**

סעיף ב':

נתחיל ביצירת כלל עזר: $S' \to S$, ולאחר מכן ניצור את מצבי האוטומט ואת המעברים ביניהם:



במצבים שצבועים באדום (מצבים 6, 10 ו-11) קיים קונפליקט $shift/reduce$ - בכל אחד ממצבים אלו קיים פריט $reduce$, וגם פריטי $shift$:

במצב 6 פריט ה-$reduce$ הוא $S \to E\ sc\ \cdot$, וישנם שני פריטי $shift$: $E \to \cdot\ id$ ו-$E \to \cdot\ num$.

במצב 10 פריט ה-$reduce$ הוא $E \to E\ sub\ E\ \cdot$, ושני הפריטים הנוספים הם פריטי $shift$: $E \to E \cdot\ sub\ E$ ו-$E \to E\ \cdot add\ E$.

במצב 11 פריט ה-$reduce$ הוא $E \to E\ add\ E\ \cdot$, ושני הפריטים הנוספים הם פריטי $shift$: $E \to E \cdot\ sub\ E$ ו-$E \to E\ \cdot add\ E$.

לכן, **הדקדוק G_2 אינו ב-LR(0)**.

סעיף ג':

בקונפליקטים מסוג $shift \,/\, reduce$ שנתקלנו בהם במצבים 10 ו-11, נבחין כי הקונפליקט נובע מכך שיש לנו גם פריט $reduce$ שמפרש את המשתנה E, כלומר מפרש את האופרנד השני בפעולת החיבור או החיסור בהתאמה, לבין פריטי $shift$ שממשיכים לקרוא את האסימונים המבטאים את פעולת החיבור או החיסור הבאה.

לכן, במקרה זה נעדיף קודם כל לסיים לפרש את האופרנד השני בפעולת החיבור או החיסור הנוכחית, ורק לאחר מכן להמשיך בקריאת האסימונים, כלומר להתקדם לפעולה הבאה.

לכן, נגדיר **אסוציאטיביות שמאלית גם לאסימון add וגם לאסימון sub**, על מנת לתעדף את פעולת ה-$reduce$ עבור מצבים 10 ו-11.

מכיוון שגם ל-add וגם ל-sub יש עדיפות זהה מבחינת סדר פעולות חשבון, נגדיר אותם בעדיפות זהה - כלומר, באותה השורה, כך שיהיו מסודרים זה לצד זה ולא זה מעל זה.

הקונפליקט מסוג $shift \,/\, reduce$ שנתקלנו בו במצב 6 מתרחש כאשר קראנו sc והאסימון הבא בקלט הוא id או num. במצב כזה, או שנפרש את ה-$statement$ שהגענו לסיומו בעת קריאת sc, או שנמשיך לקרוא מהקלט את האסימונים שמתחילים את ה-$statement$ הבא. מכיוון שבדקדוק קיים הכלל $S \rightarrow E\ sc\ S$, נרצה לאפשר מצב שבו נמשיך לקרוא את ה-$statements$ הבאים על מנת לפרשם בהמשך בתור S. לכן, **נגדיר בעדיפות גבוהה יותר את האסימונים id ו-num ביחס לעדיפות של האסימון sc** - כך הטרמינלים id ו-num יקבלו עדיפות גבוהה על הכלל $S \rightarrow E\ sc$.

סעיף ד':

נגדיר את התכונה הסמנטית הבאה:

$vars\_in\_statement$ - תכונה נוצרת שמכילה את קבוצת המשתנים שמופיעים במשתנה E בלבד - זאת מכיוון שכל $statement$ בתוכנית יפורש במהלך תהליך הפרסור לכדי משתנה E בשלב כלשהו.

הגדרה מונחית תחביר:

כלל סמנטי	כלל הגזירה
$S.new\_vars\ =\ \varnothing$	$S \rightarrow E\ sc\ S_1$
$S.new\_vars\ =\ \varnothing$	$S \rightarrow E\ sc$
$E.vars\_in\_statement\ = E_1.vars\_in\_statement \cup E_2.vars\_in\_statement$	$E \rightarrow E_1\ sub\ E_2$
$E.vars\_in\_statement\ = E_1.vars\_in\_statement \cup E_2.vars\_in\_statement$	$E \rightarrow E_1\ add\ E_2$
$E.vars\_in\_statement\ = \varnothing$	$E \rightarrow num$
$E.vars\_in\_statement\ = \{id.name\}$	$E \rightarrow id$

עבור חישוב new_vars, נבצע סריקה של העץ על ידי שימוש ב-$visitor$, ובמהלכה עבור כל צומת S נבצע את הפעולות הבאות:

1. עבור הבן E של S נבצע:

$$E.new\_vars = E.vars\_in\_statement \setminus S.new\_vars$$

2. אם קיים בן S_1 ל-S, נבצע:

$$S_1.new\_vars = E.vars\_in\_statement$$

הסבר:

תחילה, באמצעות האופן שבו הגדרנו את חישוב $vars\_in\_statement$, נוכל להבטיח שבעת כל ביקור בצומת E, הערך של $E.vars\_in\_statement$ יכיל בדויק את כל המשתנים המופיעים ב-$statement$ שמייצג E. לאחר מכן, נבטיח באמצעות האלגוריתם שמתואר לעיל חישוב נכון של התכונה הנורשת new_vars, כיוון שבכל ביקור בצומת S נדע שהערך $S.new\_vars$ מתאר במדויק את כל המשתנים שהופיעו באחיו השמאלי של S (אם קיים כזה), שהוא צומת מסוג E. כך, נוכל לחשב את הערך $E.new\_vars$ עבור בנו של כל משתנה S על ידי חיסור בין קבוצת כל המשתנים שמופיעים ב-E - כלומר $E.vars\_in\_statement$ - לבין קבוצת כל המשתנים שהופיעו בביטוי משמאל ל-E - כלומר $S.new\_vars$. כעת, אם לצומת הנוכחי S יש בן מסוג S - נסמנו S_1 - נרצה לפעפע כלפי מטה את ערך המשתנים שמופיעים בבנו של S מסוג E - כלומר, את ערך $E.vars\_in\_statement$ עבור הבן E של S. כך נמשיך בצורה רקורסיבית במורד העץ ובכך נבטיח חישוב נכון של $E.new\_vars$ לכל צומת E.

<u>שאלה 2:</u>

סעיף 1:

1. מטרת החישוב: בכל נקודה בתוכנית, נרצה למצוא את כל החישובים התקפים שבוצעו לפניה ושיכול להיעשות בהם שימוש שוב בנקודה זו.

נגדיר את הסריג:

האיברים:

על מנת להגדיר את האיברים בדומיין הסריג, נסמן את קבוצת הביטויים האריתמטיים בתור $Aexpr$, ואת קבוצת הביטויים הבוליאניים בתור $Bexpr$.

קבוצת האיברים בסריג, L, מוגדרת בתור קבוצת החזקה של הביטויים האריתמטיים: $L = \mathcal{P}(Aexpr)$.

פעולת יחס הסדר ($\sqsubseteq$):

אנחנו מעוניינים בכל נקודה בתוכנית לציין את הביטויים שבהכרח חושבו קודם ולא שונו, מכל המסלולים אשר מובילים לנקודה הזו. לכן, מדובר בבעיית $must$, ופעולת יחס הסדר החלקי בין האיברים בסריג מוגדרת כך: $\sqsubseteq = \supseteq$.

פעולת ה-$join$ ($\sqcup$):

מכיוון שמדובר בבעיית $must$, ונרצה לשמר ביטויים שהוגדרו בהכרח בכל המסלולים שמובילים לנקודה הנוכחית, פעולת ה-$join$ מוגדרת כך: $\cap \ = \ \sqcup$.

2.

תחילה, נגדיר:

$$FV : (Aexpr \cup Bexpr) \rightarrow \mathcal{P}(Vars)$$
$$in, out : \ Lab \rightarrow L$$

בנוסף, נגדיר לכל ביטוי e את $Aexpr(e)$ להיות קבוצת כל תתי הביטויים האריתמטיים שאינם אטומיים הנמצאים בביטוי e.

נגדיר לכל צומת B ב-CFG:

$in(B)$ - כל החישובים שבוצעו עד תחילת בלוק B ועדיין תקפים.

$out(B)$ - כל החישובים שבוצעו עד סוף בלוק B ועדיין תקפים.

כיוון זרימת המידע: קדמית.

כיוון שמתווספים איברים עם כל מעבר בבלוק שבו מתבצעים חישובים, לפני תחילת התוכנית אין כלל חישובים ולכן נאתחל לכל צומת B ב-CFG: $in(B) = out(B) = \varnothing$.

כעת, לכל צומת B ב-CFG נחשב את in ו-out באופן הבא:

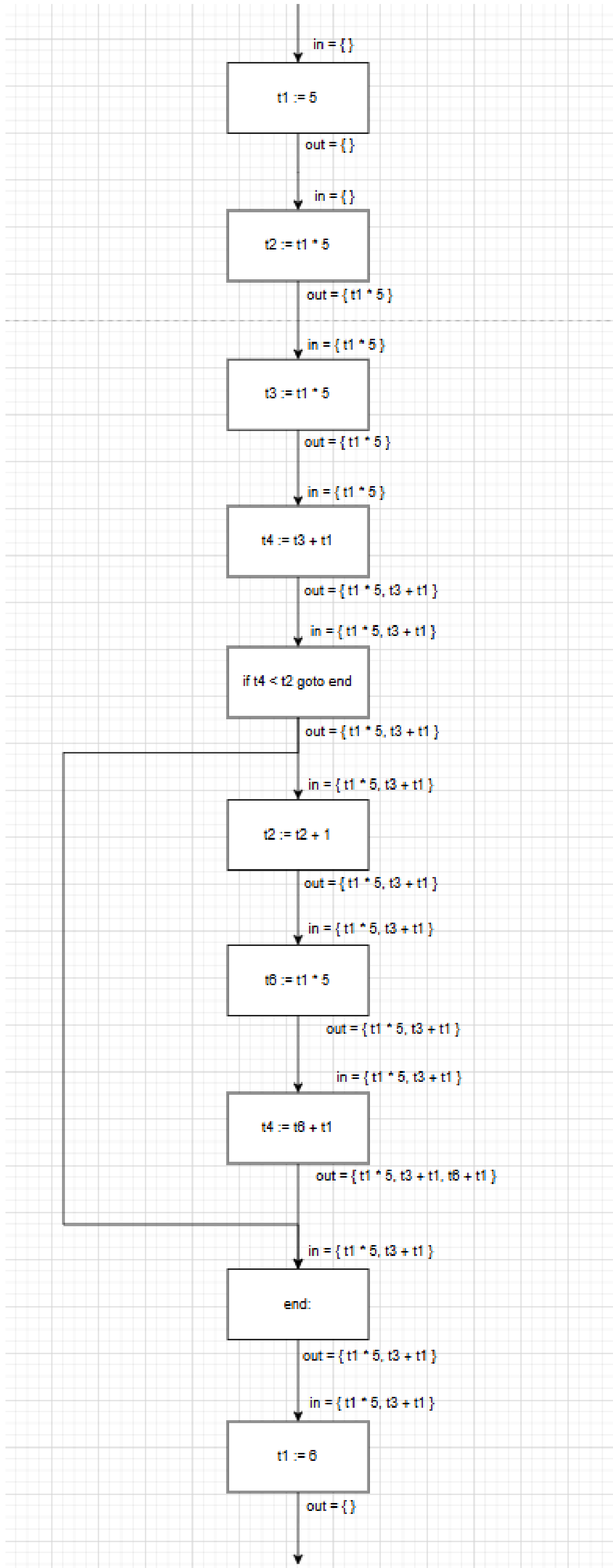
$$in(B) \ = \ \begin{cases} \varnothing & B \ is \ the \ initial \ block \ of \ the \ program \\ \bigcap \{out(B') \mid B' \in pred(B)\} & otherwise \end{cases}$$

$$out(B) =$$

Statement	$out(B)$
$x = expr$	$in(B) \setminus \underbrace{\{e \in Aexpr \mid x \in FV(e)\}}_{Kill(B)} \cup \underbrace{\{e \in Aexpr(expr) \mid x \notin FV(e)\}}_{Gen(B)}$
$goto\ label$	$in(B)$ $[Kill(B) = Gen(B) = \varnothing]$
$if\ expr_1\ relop\ expr_2\ goto\ label$	$in(B) \cup \underbrace{Aexpr(expr_1) \cup Aexpr(expr_2)}_{Gen(B)}$ $[Kill(B) = \varnothing]$
$label:$	$in(B)$ $[Kill(B) = Gen(B) = \varnothing]$

נקבל את המבוקש מה-DFA משום שלכל בלוק B, $out(B)$ יכיל את קבוצת החישובים התקפים שבוצעו עד אליו
(כולל) בתוכנית, כך שנוכל לדעת בכניסה לכל בלוק באילו חישובים נוכל להשתמש שוב.

סעיף 2:



in = { }
t1 := 5
out = { }
in = { }
t2 := t1 * 5
out = { t1 * 5 }
in = { t1 * 5 }
t3 := t1 * 5
out = { t1 * 5 }
in = { t1 * 5 }
t4 := t3 + t1
out = { t1 * 5, t3 + t1 }
in = { t1 * 5, t3 + t1 }
if t4 < t2 goto end
out = { t1 * 5, t3 + t1 }
in = { t1 * 5, t3 + t1 }
t2 := t2 + 1
out = { t1 * 5, t3 + t1 }
in = { t1 * 5, t3 + t1 }
t6 := t1 * 5
out = { t1 * 5, t3 + t1 }
in = { t1 * 5, t3 + t1 }
t4 := t6 + t1
out = { t1 * 5, t3 + t1, t6 + t1 }
in = { t1 * 5, t3 + t1 }
end:
out = { t1 * 5, t3 + t1 }
in = { t1 * 5, t3 + t1 }
t1 := 6
out = { }

סעיף 3:

מטרת החישוב: בכל נקודה בתוכנית, נרצה למצוא את כל החישובים התקפים שבוצעו לפניה ושיכול להיעשות בהם שימוש שוב בנקודה זו, יחד עם המשתנה אליו הושמו חישובים אלה.

לצורך תמיכה בדרישה החדשה, נרצה לשנות את ההגדרות משני הסעיפים הקודמים:

נגדיר תחילה קבוצה T:

$$T \;=\; \{(expr,\; var) \mid 1.\; var \in Vars \cup \{\$\}$$
$$2.\; expr \in Aexpr$$
$$3.\; (expr,\; var)\; satisfies\; one\; of\; the\; conditions\; below\; (*)\}$$

$(*)$ נדרוש שכל זוג $(expr,\; var) \in T$ יקיים אחד מהתנאים הבאים:

1. אם $var \in Vars$: ההשמה האחרונה שבוצעה לתוך var היא: $var := expr'$, וגם $expr$ הוא הביטוי שנקבל לאחר החלפה של כל משתנה ב-$expr'$ בהשמה תקפה שלו, עד אשר כל משתנה יוחלף בחישוב התקף הבסיסי ביותר שהוא מייצג. למשל, בדוגמה לעיל, בעת ביצוע ההשמה $t4 := t3 + t1$, נוכל להחליף את $t3$ בחישוב $t1 * 5$ מכיוון שלפני כן בוצעה השמה $t3 := t1 * 5$, והיא עוד תקפה. כך נקבל: $t4 := t1 * 5 + t1$, ולכן הזוג הסדור $(t1{*}t5 \;+\; t1 \;,\; t4)$ יוכנס כעת לתוך T, ונוציא אותו מ-T אם ורק אם הערך של $t1$ ישתנה או הערך של $t4$ ישתנה.
2. אם $var = \$$: החישוב $expr$ חושב במסגרת תנאי של קפיצה מותנית, ולא שונה מאז.

קבוצת האיברים בסריג, L, מוגדרת בתור קבוצת החזקה של T: $\mathcal{P}(T)$.

פעולת יחס הסדר ($\sqsubseteq$):

בדומה לסעיף 1 אנחנו מעוניינים בכל נקודה בתוכנית לציין את החישובים הבסיסיים ביותר שבהכרח חושבו קודם ולא שונו, מכל המסלולים אשר מובילים לנקודה הזו, והפעם נתייחס גם למשתנים שאליהם הושמו אותם החישובים התקפים (או לסימן $\$$ אם החישוב exp בוצע בתנאי של קפיצה מותנית). לכן, מדובר בבעיית $must$, ופעולת יחס הסדר החלקי בין האיברים בסריג מוגדרת כך: $\sqsubseteq = \supseteq$.

פעולת ה-$join$ ($\sqcup$):

מכיוון שמדובר בבעיית $must$, ונרצה לשמר ביטויים שהוגדרו בהכרח בכל המסלולים שמובילים לנקודה הנוכחית, פעולת ה-$join$ מוגדרת כך: $\sqcup \;=\; \cap$.

כעת לכל צומת B ב-CFG:

$in(B)$ - כל משתנה var שהייתה עבורו השמה תקפה של ביטוי אריתמטי $expr'$, ובצמוד אליו הביטוי האריתמטי התקף והמופשט ביותר שייצג $expr'$ ברגע ההשמה עד תחילת בלוק B.

$out(B)$ - כל משתנה var שהייתה עבורו השמה תקפה של ביטוי אריתמטי $expr'$, ובצמוד אליו הביטוי האריתמטי התקף והמופשט ביותר שייצג $expr'$ ברגע ההשמה עד סוף בלוק B.

כיוון זרימת המידע: קדמית.

מאותם נימוקים, נאתחל את $in(B)$ ואת $out(B)$ כמו קודם בתור קבוצות ריקות.

לכל צומת ב-CFG נחשב את in באותה הצורה כמו בסעיף הקודם, ונגדיר מחדש את out באופן הבא:

$out(B) =$

Statement	$out(B)$
$x = expr$	$in(B) \setminus Kill(B) \ \cup \ Gen(B)$ $Kill(B) = \{(e, v) \mid 1.\ e \in Aexpr$ $2.\ v \in Vars \cup \{\$\}$ $3.\ x \in FV(e)\ or\ x = v\}$ $Gen(B) = \{(expr', x) \mid expr'\ is\ the\ expression\ we\ get$ $when\ for\ each\ v \in Aexpr,$ $if\ (e', v) \in in(B)\ we\ replace\ v\ with\ e'\}$
goto label	$in(B)$ $[Kill(B) = Gen(B) = \varnothing]$
$if\ expr_1\ relop\ expr_2\ goto\ label$	$in(B) \cup \underbrace{\{(e, \$) \mid e \in Aexpr(expr_1) \cup Aexpr(expr_2)\}}_{Gen(B)}$ $[Kill(B) = \varnothing]$
label :	$in(B)$ $[Kill(B) = Gen(B) = \varnothing]$

בביצוע אנליזה מסוג $Forward\ Analysis$, השינויים מסייעים לנו לענות על הדרישה הנוספת:
עבור כל צומת ב-CFG, ערך ה-in עבור צומת זה מתאר במדויק את הפריטים הבאים:
1. כל משתנה var שהייתה עבורו השמה תקפה של ביטוי אריתמטי $expr'$, ובצמוד אליו הביטוי האריתמטי התקף והמופשט ביותר שייצג $expr'$ ברגע ההשמה - זאת מכיוון שהקבוצה in מתעדכנת בכל השמה, וכך אנו מוודאים שהביטויים שנכללים בה הם תמיד הביטויים התקפים המופשטים ביותר אליהם ניתן להגיע לפי השלבים הקודמים. כלומר, ניתן להגיד שאנחנו משמרים צורה "קנונית" של כל חישוב תקף עבור כל המשתנים שאליו הושם.
2. כל חישוב תקף שבוצע בתוך תנאי של קפיצה מותנית.

כעת, אם קיים בצומת הנוכחי חישוב כלשהו, יהיה ניתן לזהות האם החישוב כבר בוצע ואם הוא הושם למשתנה אחר לפני כן - זאת על ידי החלפת כל המשתנים בחישובים התקפים המתאימים להם עד שנגיע לצורה התקפה והמופשטת ביותר של החישוב שבצומת הנוכחי, ועל ידי סריקת ערך ה-in של הבלוק הנוכחי עבור זוג סדור שמכיל את החישוב שהתקבל. כך נוכל לזהות משתנים שמכילים את אותו החישוב (בצורתו ה"קנונית"), ו/או לזהות שהחישוב בוצע בתור תנאי לקפיצה מותנית, ובעזרת מידע זה נוכל לבצע את האופטימיזציה הנדרשת.