21.3.2022

מבחן סוף סמסטר – מועד ב׳

פתרון חלקי

מרצה אחראי: ד״ר שחר יצחקי

מתרגלים: יונתן יעקבי, מתן פלד, נדב רובינשטיין

הוראות:

א. בטופס המבחן 11 עמודים, וכן 5 דפי נוסחאות. בדקו שכל העמודים ברשותכם.
ב. משך המבחן שלוש שעות (180 דקות).
ג. כל חומר עזר חיצוני אסור לשימוש.
ד. בשאלות הפתוחות, ניתן לציין לגבי סעיף או שאלה ״לא יודע/ת״. תשובה זו תזכה ב-20% מהניקוד. תשובות שגויות לא יזכו בניקוד.
ה. קראו את כל המבחן לפני שאתם מתחילים לענות על השאלות.
ו. אין צורך להגיש את טופס מבחן זה בתום הבחינה.
ז. את התשובות לשאלות הסגורות יש לסמן בטופס התשובות הנפרד בלבד.
ח. התשובה היא תמיד ג׳

בהצלחה!

חלק א׳ - שאלות סגורות (50 נק׳)

שלבי קומפילציה

```
1   bool isPal(int n) {
2      int rev = 0;
3      int lr = -1;
4      int rem = 0;
5      while (n > rev) {
6         rem = n % 10;
7         n = n / 10;
8         if (lr < 0 && rem == 0)
9            return false;
10        lr = rev;
11        rev = 10 * rev * rem;
12        if (n == rev || n == lr)
13           return true;
14     }
15     return false;
16  }
```

בסעיפים הבאים (שאלות 1-4) מוצגים שינויים (בלתי תלויים) לקוד של התוכנית הרשומה מעלה.
עבור כל שינוי ציינו את השלב המוקדם ביותר שבו נגלה את השגיאה (2 נק׳ לשאלה).

הערה: לצורך שאלה זו נניח שב-FanC קיים האסימון ״%״ מסוג BINOP והוא משמש לפעולת השארית (מודולו) כמו ב-C.

שאלה 1 (2 נק׳)
נחליף את המילה while בשורה 5 במילה for.

א. שגיאה בניתוח לקסיקלי
ב. אין שגיאה
ג. שגיאה בניתוח תחבירי
ד. שגיאה בניתוח סמנטי
ה. שגיאה בייצור קוד
ו. שגיאה בזמן ריצה

שאלה 2 (2 נק׳)
נחליף את המילה while בשורה 5 במילה fork .

א. שגיאה בניתוח לקסיקלי
ב. אין שגיאה
ג. שגיאה בניתוח תחבירי
ד. שגיאה בניתוח סמנטי
ה. שגיאה בייצור קוד
ו. שגיאה בזמן ריצה

שאלה 3 (2 נק׳)
נחליף את רצף התווים && בשורה 8 ברצף התווים ^^ .

א. שגיאה בניתוח תחבירי
ב. אין שגיאה
ג. שגיאה בניתוח לקסיקלי
ד. שגיאה בניתוח סמנטי
ה. שגיאה בייצור קוד
ו. שגיאה בזמן ריצה

שאלה 4 (2 נק׳)

נחליף את המילה int בשורה 1 במילה byte .

א. שגיאה בניתוח לקסיקלי
ב. אין שגיאה
ג. שגיאה בניתוח סמנטי
ד. שגיאה בניתוח תחבירי
ה. שגיאה בייצור קוד
ו. שגיאה בזמן ריצה

שאלה 5 (2 נק׳)

באיזה שלב מתבצעת **ההחלטה** האם קודם לכפול את 10 ב-rev או קודם לכפול את rev ב-rem בשורה 11?

א. בשלב הניתוח הסמנטי
ב. בשלב הניתוח הלקסיקלי
ג. בשלב הניתוח התחבירי
ד. בשלב ייצור הקוד
ה. בזמן הריצה

שינויים בשפה

שאלה 6 (5 נק׳)

נרצה להוסיף לשפת FanC תמיכה במערכים בדומה לשפת C. באיזה שלב קומפילציה, מבין הרשומים כאן, *לא* נצטרך לערוך שינויים?

א. ניתוח לקסיקלי
ב. ייצור קוד
ג. נצטרך לערוך שינויים בכל אחד משלבי הקומפילציה הרשומים כאן.
ד. ניתוח תחבירי
ה. ניתוח סמנטי

שאלה 7 (5 נק׳)

בעת מימוש תמיכה במערכים עבור FanC, בשלב ייצור הקוד, האם יש להשתמש בטכניקת ה-backpatching כפי שהיא נלמדה בכיתה?

א. כן, אבל לא נצטרך לעשות emit לקוד נוסף, אפשר לפתור רק באמצעות קריאות ל-backpatch.
ב. לא, כי לא נעשה שינויים בשלב ייצור הקוד.
ג. לא, השינויים בשלב ייצור הקוד לא דורשים ביצוע backpatching.
ד. כן, ונצטרך לקרוא גם ל-emit וגם ל-backpatch במימוש.
ה. כן, אבל רק אם התמיכה במערכים כוללת bounds checking (בדיקה האם הגישה בגבולות המערך).

שאלה 8: תכנות מונחה עצמים (5 נק׳)

נתונה התוכנית הבאה בשפת תכנות מונחית עצמים הדומה לשפה שנלמדה בהרצאה (להזכירכם, בשפת התכנות הזו כל המתודות הן וירטואליות):

```
interface I {
  int f();
}

class A extends I {
  int f() { /* ... */ }
}

class B extends I {
  int f() { /* ... */ }
}

void g(A a) { /* ... */ }
void g(B a) { /* ... */ }

void func() {
  /* ...some code that declares a variable named “x”... */
  x.f();
  g(x);
}
```

מה ההבדל בין הקריאה x.f() לקריאה g(x)? בחרו בתשובה שתהיה נכונה עבור כל השלמה של הקוד בהערה, בהנחה ש-x הוא מטיפוס A או B.

א. בקריאה x.f() תיבחר הגרסה של f שתיקרא בזמן הקומפילציה, בעוד בקריאה g(x) תיבחר הגרסה של g שתיקרא בזמן הריצה.
ב. הקריאה x.f() תהיה מהירה יותר מהקריאה g(x).
ג. בקריאה x.f() תיבחר הגרסה של f שתיקרא בזמן הריצה, בעוד בקריאה g(x) תיבחר הגרסה של g שתיקרא בזמן הקומפילציה.
ד. הקריאה x.f() היא תקינה סמנטית, ואילו הקריאה g(x) איננה תקינה סמנטית.
ה. אין הבדל בין הקריאות.

שאלה 9: אופטימיזציה (5 נק׳)

נניח שבידינו קומפיילר אשר מייצר את קוד הביניים ישירות מה-CFG, כך שבסוף כל basic block יש goto מפורש לתחילת ה-basic block הבא, גם אם ה-basic block הבא נמצא מיד אחריו ברצף הפקודות. חברת ״פייזר״ שוקלת שימוש בקומפיילר הזה, והמהנדס הראשי תוהה מה תהיה ההשפעה של תכונה זו.

א. הקוד הסופי יהיה פחות יעיל כי הוא יהיה גדול יותר, ולכן עלול לחרוג מה-cache.
ב. הקוד הסופי יהיה יותר יעיל כי ה-goto המפורש משפר את ביצועי המעבד.
ג. הקוד הסופי יהיה יעיל בדיוק כמו קוד ללא ה-goto-ים הנ״ל כי הם יימחקו באחד משלבי הקומפילציה.
ד. הקוד הסופי יהיה פחות יעיל כי קפיצות הן פעולות יקרות בזמן ריצה.

דקדוקים

בשאלות הבאות, אותיות גדולות A,B,... הם משתנים, S הוא המשתנה ההתחלתי, ואותיות קטנות a,b,... הם טרמינלים.

שאלה 10 (5 נק׳)

נתון דקדוק G=(V,T,P,S). נגדיר את הדקדוק G_1 המתקבל מ-G להיות:

$$G_1 = (V, T \cup \{t\}, P \cup \{X \to t\}, S)$$

כאשר $X \in V, t \notin T$.

מה מהבאים הכי נכון:

א. אם G שייך ל SLR אז G_1 שייך ל.SLR
ב. אם G שייך ל LR(0) אז G_1 שייך ל LR(0).
ג. אף תשובה אינה נכונה.
ד. אם G שייך ל LR(1) אז G_1 שייך ל LR(1).
ה. תשובות ג׳ וד׳ נכונות.

שאלה 11 (5 נק׳)

נתונות פעולות אריתמטיות בינאריות #, @, &.

אליס הגדירה אותם בקובץ ה bison בצורה הבאה:

```
%%
...
/* remember: first is lowest */
% left '&'
% right '#'
% left '@'
...
%%
```

וכן הגדירה בהמשך הקובץ את כללי הגזירה הבאים:

```
E : E @ E { /* … */ }
  | E # E { /* … */ }
  | E & E { /* … */ }
...
```

נתון הביטוי:

1 # 2 # 3 & 4 @ 5 @ 6 & 7

אילו מהסוגריים הבאות מתאימות לסדר הפעולות בו יש לחשב?

א. ((1 # (2 # (3 & 4))) @ 5) @ (6 & 7)
ב. ((1 # 2) # 3) & ((4 @ (5 @ 6)) & 7)
ג. ((1 # (2 # 3)) & ((4 @ 5) @ 6)) & 7
ד. (1 # (2 # (3 & 4))) @ (5 @ (6 & 7))
ה. ((((1 # (2 # 3)) & 4) @ 5) @ 6) & 7

לפנייך שני מצבים מתוך מכונות מצבים של אוטומטי SLR של דקדוקים כלשהם (לא בהכרח אותו דקדוק):

1.	2.
$A \rightarrow a \bullet B$ $B \rightarrow \bullet BaBa$ $B \rightarrow \bullet c$	$A \rightarrow a \bullet B$ $B \rightarrow \bullet BbB$ $B \rightarrow \bullet c$

שאלה 12 (5 נק׳)

בחר במשפט הנכון ביותר לגבי מצב מספר 1:

א. ייתכן דקדוק SLR שבמכונת המצבים של אוטומט SLR שלו קיים מצב זה.
ב. ייתכן דקדוק LR(0) שבמכונת המצבים של אוטומט SLR שלו קיים מצב זה.
ג. תשובות א, ד נכונות.
ד. ייתכן דקדוק LR(1) שבמכונת המצבים של אוטומט SLR שלו קיים מצב זה.
ה. תשובות ב, ג, ד נכונות.
ו. אף תשובה אינה נכונה.

שאלה 13 (5 נק׳)

בחר במשפט הנכון ביותר לגבי מצב מספר 2:

א. ייתכן דקדוק SLR שבמכונת המצבים של אוטומט SLR שלו קיים מצב זה.
ב. ייתכן דקדוק LR(0) שבמכונת המצבים של אוטומט SLR שלו קיים מצב זה.
ג. אף תשובה אינה נכונה.
ד. ייתכן דקדוק LR(1) שבמכונת המצבים של אוטומט SLR שלו קיים מצב זה.
ה. תשובות ב, ג נכונות.
ו. תשובות ב, ג, ד נכונות.

חלק ב׳ - שאלות פתוחות (50 נק׳)

שאלה 1: אופטימיזציות (5 נק׳)

נתון הקוד הבא בשפת הביניים, שמייצג את הגרסה המקומפלת (ע״י קומפיילר כלשהו) של isPal משאלה קודמת.
הניחו כי n מתקבל כפרמטר, ונוסיף את return לפעולות בשפת הרביעיות.

```
1   rev = 0
2   lr = -1
3   rem = 0
4   deca = 10
5   if n <= rev goto 20
6   temp1 = n % deca
7   rem = temp1
8   temp2 = n / deca
9   n = temp2
10  if lr >= 0 goto 13
11  if rem != 0 goto 13
12  return false
13  lr = rev * 1
14  temp3 = deca * rev
15  rev = temp3 * rem
16  if rev == n goto 18
17  if n != lr goto 19
18  return true
19  goto 4
20  return false
```

הפעילו את האופטימיזציות מהרשימה להלן על הקוד הנתון, וכתבו את הקוד לאחר האופטימיזציות. ציינו את האופטימיזציות שהפעלתן ואת סדר ההפעלה. אין צורך לכתוב את תוצאות הביניים, רק את התוצאה הסופית לאחר ביצוע כל האופטימיזציות.

- Algebraic Simplification
- Copy Propagation
- Constant Propagation
- Branch Chaining
- החלפת קפיצה מותנית בקפיצה לא מותנית
- Useless Code Elimination
- Unreachable Code Elimination

שאלה 2: אנליזה סטטית (30 נק׳)

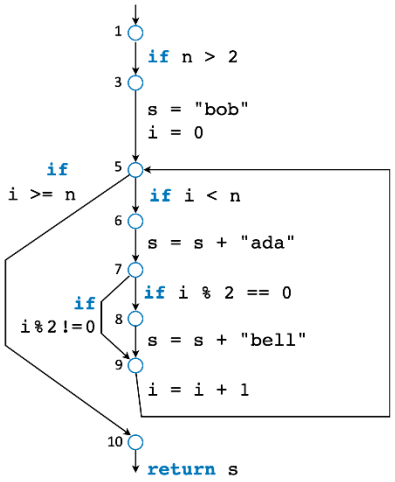
בוב מחבב ביותר מחרוזות תווים שבהם מספר התווים מסוג ׳a׳ גדול ממספר התווים מסוג ׳b׳. בוב גילה (לאחר שהתעייף מבניית אוטומטים פרפיקסיים וכיו״ב) שבהרצאה האחרונה בקורס 236360 מוצגת שיטה לניתוח תכונות אריתמטיות של תוכניות בעזרת הדומיין האבסטרקטי של אינטרוולים.

$$A = (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{\infty\})$$
$$[a,b] \sqsubseteq [c,d] \Leftrightarrow a \geq c \wedge b \leq d; [a,b] \sqcup [c,d] = [\min\{a,c\}, \max\{b,d\}]$$
$$\alpha(S) = [\min S, \max S]; \quad \gamma([a,b]) = \{v \mid a \leq v \leq b\}$$

ואז עלה במוחו הרעיון הבא: עבור כל משתנה מסוג מחרוזת, אפשר לעקוב אחרי מספר המופעים של ׳a׳ במחרוזת ועבור מספר המופעים של ׳b׳. באופן זה הדומיין יהיה <u>זוג</u> של אינטרוולים $A \times A$, שהראשון מביניהם מייצג את מופעי ׳a׳ והשני את מופעי ׳b׳. ואז, על ידי השוואת הטווחים, ניתן יהיה לקבוע מי מהם מופיע יותר פעמים.

א. (5 נק׳) הסבירו לבוב מדוע האנליזה שהוא מציע לא תהיה מדויקת, תוך שימוש בתוכנית הבאה כדוגמה (התוכנית כתובה בשפת Python; לא נדרש ידע ב-Python כדי לענות על הסעיף).

```
1   def charlie(n):
2     assert n > 2
3     s = "bob"
4     i = 0
5     while i < n:
6       s = s + "ada"
7       if i % 2 == 0:
8         s = s + "bell"
9       i = i + 1
10    return s
```



<u>בסעיף זה אין צורך להריץ</u> את האנליזה (שכן בוב עדיין לא הגדיר אותה), אלא להסביר מדוע הגישה שבוב הציע לא יכולה להביא לאנליזה שהיא גם נאותה, וגם מדויקת עבור התוכנית הזו.

בסעיף זה ובסעיפים הבאים ניתן להניח קיומו של אופרטור widening מתאים, אשר מבטיח את התכנסות האנליזה גם במקרה של לולאות לא חסומות כמו שיש בדוגמה. **<u>אין צורך</u>** לרשום את ההגדרה של widening או לציין מתי הוא מופעל.

ב. (15 נק׳) בוב לא מתייאש מהכישלון שנחל בניסיון הראשון, ומעלה רעיון חדש: במקום לעקוב אחרי מספר המופעים של תו יחיד, אפשר לעקוב אחר **ההפרש** בין מספר התווים מסוג ׳a׳ למספר התווים מסוג ׳b׳, ואם מספר זה חיובי, הרי שהמחרוזת זוהתה כחביבה.

בשיטה זו אין צורך בזוגות של אינטרוולים כמו ברעיון הקודם; כל משתנה מחרוזת ייוצג על-ידי אינטרוול בודד עבור ההפרש הנ״ל. משתנים מספריים ייוצגו גם הם על-ידי אינטרוולים באופן הרגיל. (נניח שבוב מוכן לספק לאנליזה כקלט סיווג של המשתנים לטיפוס מחרוזת ומספר.)

ממשו את האנליזה של בוב על-ידי שתשלימו את הסמנטיקה האבסטרקטית של ביטויים מעל מחרוזות לפי הפירוט הבא:

$[\![v = e]\!]^{\#}\sigma^{\#} = \sigma^{\#}[v \mapsto [\![e]\!]^{\#}\sigma^{\#}]$	(את ההגדרה הזו בוב העתיק מהמצגת של הרצאה 12)
$[\![\omega]\!]^{\#}\sigma^{\#} =$	כאשר ω הוא ליטרל (קבוע) מסוג מחרוזת
$[\![e_1 + e_2]\!]^{\#}\sigma^{\#} =$	כאשר e_1, e_2 הם <u>ביטויים</u> מטיפוס מחרוזת ו-׳+׳ הוא אופרטור שרשור מחרוזות
$[\![e_1 * e_2]\!]^{\#}\sigma^{\#} =$	כאשר e_1 הוא ביטוי מחרוזת, e_2 הוא ביטוי מטיפוס int, ו-׳*׳ הוא אופרטור שכפול מחרוזות (לדוג׳ "bab" * 3 = "babbabbab")

וכן את הסמנטיקה של משפטי התנאי הבאים:

כאשר e_1, e_2 הם <u>ביטויים</u> מטיפוס מחרוזת ו-'==' משווה תוכן מחרוזות	$[\![\text{if } e_1 == e_2]\!]^{\#}\sigma^{\#} =$
כאשר e_1, e_2 הם <u>ביטויים</u> מטיפוס מחרוזת ו-'in' בודק האם e_1 הוא תת-מחרוזת של e_2	$[\![\text{if } e_1 \texttt{ in } e_2]\!]^{\#}\sigma^{\#} =$

מותר להשתמש באופרטורים האבסטרקטיים $\hat{+}$ ו-$\hat{\cdot}$ שהוגדרו לביצוע פעולות חיבור וכפל בדומיין האינטרוולים, **<u>ואין צורך</u>** להגדירם מחדש.

ג. (10 נק') ציירו CFG של תוכנית אשר עבורה הרעיון הראשוני של בוב מסעיף א' יביא לתוצאה בלתי מדויקת ואילו האנליזה המשופרת מסעיף ב' תביא לתוצאה מדויקת. חשבו את תוצאת האנליזה (המשופרת) עבור התוכנית הזאת ונמקו מדוע התוכנית עומדת בתנאי שבמשפט הקודם.

כתבו את הפתרון שמצאה האנליזה (המשופרת) עבור כל אחד מהמיקומים ב CFG. **אין צורך** לפרט ערכי ביניים המתקבלים במהלך ריצת האנליזה.

טיפ. לא כדאי להשתמש בתוכנית מסעיף א' כדוגמה, מכיוון שגרף הבקרה שלה גדול, מה שידרוש מספר רב של חישובים כדי להגיע לתשובה. נסו למצוא תוכנית קטנה יותר.

שאלה 3: ייצור קוד (15 נק׳)

בתוכנה שאתם מפתחים הוחלט להתחקות אחרי סלילי DNA מיוחדים ולשם כך עליכם להבדיל בין וקטורים פלינדרומים לוקטורים שאינם פלינדרומים.

וקטור פלינדרומי הינו וקטור שקריאתו מימין לשמאל ומשמאל לימין היא זהה.

לשם כך נתון מבנה הבקרה הבא:

1) B -> foreach (ID_1 , ID_2) in palindrom Vector with B_1

2) Vector -> E , $Vector_1$ | E

כאשר Vector מייצג רשימה של ערכים הניתנים להשוואה .

הטרמינלים בדקדוק מסומנים בקו תחתון. נוספו אינדקסים לצורך התייחסות למופעים שונים של אותו סימן.

בכלל 1, תוצאת הביטוי הבוליאני B תהיה true אם ורק אם ה-Vector הינו פלינדרום, כאשר השוואה בין איברים בווקטור מוגדרת על-ידי ערכו של הביטוי הבוליאני B_1, כאשר המשתנים ששמותיהם ID_1, ID_2 מקבלים את ערכי האיברים להשוואה.

דוגמא:

if (foreach (a,b) in palindrom (1,2,3,4,5,5,4,3,2,1) with a==b) print “True!”;

הערות:

- הניחו שהווקטור (Vector העליון ביותר בעץ הגזירה) יכיל תמיד מספר זוגי של איברים.
- ניתן להשתמש אך ורק ב-queue וב-stack כמבני זיכרון בשאלה, עם הפעולות שלמדנו: pop, top, push ואתחול. כרגיל, במבנה מסוג stack, כל הפעולות מתבצעות על ראש המחסנית, ובמבנה מסוג queue, הפעולה push מכניסה איבר לסוף התור והפעולות pop, top מתייחסות לראש התור. שימו לב **שלא ניתן** לגשת לאיבר מסוים במבנה ע״י אינדקס.
- ניתן להשתמש במבני הנתונים הללו עבור תכונות של משתנים בזמן הקומפילציה **בלבד**, ולא עבור משתנים בזמן הריצה.
- ניתן להניח שיש מקום במחסנית זמן הריצה (רשומת ההפעלה) עבור כל ערכי E_i הנגזרים על-ידי Vector.

א. (5 נק׳) הציעו פריסת קוד לכללים 1,2 בשיטת backpatching עבור מבנה הבקרה הנ״ל. על הקוד הנוצר להיות יעיל ככל האפשר.

ב. (6 נק׳) כתבו סכימת תרגום לכללים 1,2 בשיטת backpatching המייצרת את פריסת הקוד שהצעתם בסעיף הקודם. על הסכימה להיות יעילה ככל האפשר, הן מבחינת זמן הריצה שלה והן מבחינת המקום בזיכרון הנדרש עבור התכונות הסמנטיות. כמו כן, הסבירו מהן התכונות שאתם משתמשים בהן עבור כל משתנה.

ג. (4 נק׳) מכיוון שסלילי DNA יכולים להיות ארוכים, ה overhead של מבני הזיכרון (הן בזמן קומפילציה והן בזמן ריצה) עלול להיות גדול. נניח שהחלטנו כי מספיק לבדוק רק את 20 האיברים הראשונים ו-20 האיברים האחרונים בבדיקת התכונה (כלומר, יש לבדוק שקריאת 20 האיברים הראשונים משמאל לימין וקריאת 20 האיברים האחרונים מימין לשמאל זהה), וכן שגודל כל סליל הוא בדיוק 100. כיצד תוכלו להקטין את צריכת הזיכרון ואת גודל הקוד ועדיין לשמור על נכונות, תחת הנחות אלה?

ניתן לתאר במילים כיצד פריסת הקוד הייתה משתנה, ואיך החיסכון היה מתבטא.

נוסחאות ואלגוריתמים

כל ההגדרות מתייחסות לדקדוק $G = (V, T, P, S)$.

Top Down

$first(\alpha) = \{ t \in T \mid \alpha \Rightarrow^* t\beta \wedge \beta \in (V \cup T)^* \}$

$follow(A) = \{ t \in T \cup \{\$\} \mid S\$ \Rightarrow^* \alpha A t \beta \wedge \alpha \in (V \cup T)^* \wedge \beta \in (V \cup T)^* (\varepsilon | \$) \}$

$$select(A \to \alpha) = \begin{cases} first(\alpha) \cup follow(A) & \alpha \Rightarrow^* \varepsilon \\ first(\alpha) & \text{otherwise} \end{cases}$$

הגדרה: דקדוק G הוא LL(1) אם ורק אם לכל שני כללים ב- G השייכים לאותו משתנה A מתקיים:
$select(A \to \alpha) \cap select(A \to \beta) = \varnothing$

הגדרת טבלת המעברים $M : V \times (T \cup \{\$\}) \to P \cup \{error\}$ עבור דקדוק LL(1):

$$M[A , t] = \begin{cases} A \to \alpha & t \in select(A \to \alpha) \\ error & t \notin select(A \to \alpha) \text{ for all } A \to \alpha \in P \end{cases}$$

אלגוריתם מנתח LL(1):

```
Q.push(S)
while !Q.empty() do
     X = Q.pop()
     t = next token
     if X ∈ T then
          if X = t then MATCH
          else ERROR
     else // X ∈ V
          if M[X , t] = error then ERROR
          else PREDICT(X , t)
     end if
end while
t = next token
if t = $ then ACCEPT
else ERROR
```

Bottom Up

פריט LR(0) הוא $(A \to \alpha \bullet \beta)$ כאשר $A \to \alpha\beta \in P$

סגור (closure) על קבוצת פריטים I מוגדר באופן אינדוקטיבי:

- בסיס: $closure(I) = I$.
- צעד: אם $(A \to \alpha \bullet B\beta) \in closure(I)$, אז לכל $B \to \gamma \in P$, גם $(B \to \bullet\gamma) \in closure(I)$

פונקציית המעברים של האוטומט:

$$\delta(I , X) = \bigcup \{ closure(A \to \alpha X \bullet \beta) \mid (A \to \alpha \bullet X\beta) \in I \}$$

פריט LR(1) הוא $(A \to \alpha \bullet \beta , t)$ כאשר $A \to \alpha\beta \in P$, $t \in T \cup \{\$\}$

סגור (closure) על קבוצת פריטים I מוגדר באופן אינדוקטיבי:

בסיס: $closure(I) = I$.

צעד: אם $(A \to \alpha \bullet B\beta , t) \in closure(I)$, אז לכל $B \to \gamma \in P$ ולכל $x \in first(\beta t)$, גם

$$(B \to \bullet\gamma , x) \in closure(I)$$

פונקציית המעברים של האוטומט:

$$\delta(I , X) = \bigcup \{ closure(A \to \alpha X \bullet \beta , t) \mid (A \to \alpha \bullet X\beta , t) \in I \}$$

הגדרת טבלת action למנתח SLR:

$$action[i , t] = \begin{cases} SHIFT_j & \delta(I_i ,t) = I_j \\ REDUCE_k & \text{rule k is } A \to \alpha,\ (A \to \alpha \bullet) \in I_i \text{ and } t \in follow(A) \\ ACCEPT & (S' \to S \bullet) \in I_i \text{ and } t = \$ \\ ERROR & \text{otherwise} \end{cases}$$

הגדרת טבלת action למנתח LR(1):

$$action[i , t] = \begin{cases} SHIFT_j & \delta(I_i ,t) = I_j \\ REDUCE_k & \text{rule k is } A \to \alpha \text{ and } (A \to \alpha \bullet , t) \in I_i \\ ACCEPT & (S' \to S \bullet , \$) \in I_i \text{ and } t = \$ \\ ERROR & \text{otherwise} \end{cases}$$

הגדרת טבלת goto למנתח SLR ו- LR(1):

$$goto[i , X] = \begin{cases} j & \delta(I_i , X) = I_j \\ error & \text{otherwise} \end{cases}$$

אלגוריתם מנתח shift/reduce:

```
Q.push(0)          // where 0 is the initial state of the prefix automaton
while true do
     k = Q.top().state
     t = next token
     do action[k , t]
end while
```

קוד ביניים

סוגי פקודות בשפת הביניים:

1. משפטי השמה עם פעולה בינארית
2. משפטי השמה עם פעולה אונרית
3. משפטי העתקה
4. קפיצה בלתי מותנה
5. קפיצה מותנה
6. הדפסה

```
x := y op z
x := op y
x := y
goto L
if x relop y goto L
print x
```

Data-Flow Analysis

ההגדרות מתייחסות ל – CFG: G=(V,E).

הצורה הכללית של המשוואות בחישוב סריקה קדמית:

$$in(B) = \bigcap_{(S,B)\in E} out(S) \quad \text{או} \quad in(B) = \bigcup_{(S,B)\in E} out(S)$$

$$out(B) = f_B(in(B))$$

הצורה הכללית של המשוואות בחישוב סריקה אחורית:

$$out(B) = \bigcap_{(B,S)\in E} in(S) \quad \text{או} \quad out(B) = \bigcup_{(B,S)\in E} in(S)$$

$$in(B) = f_B(out(B))$$

שפת FanC

אסימונים:

אסימון	תבנית
VOID	void
INT	int
BYTE	byte
B	b
BOOL	bool
CONST	const
AND	and
OR	or
NOT	not
TRUE	true
FALSE	false
RETURN	return
IF	if
ELSE	else
WHILE	while
BREAK	break
CONTINUE	continue
SC	;
COMMA	,
LPAREN	(
RPAREN	)
LBRACE	{
RBRACE	}
ASSIGN	=
RELOP	== | >= | <= | > | < | != |
BINOP	+ | - | * | /
ID	[a-zA-Z][a-zA-Z0-9]*
NUM	0 | [1-9][0-9]*
STRING	"([^\n\r\"\\]|\\[rnt"\\])+"

דקדוק:

44 $Program \rightarrow Funcs$
45 $Funcs \rightarrow \epsilon$
46 $Funcs \rightarrow FuncDecl\ Funcs$
47 $FuncDecl \rightarrow RetType\ ID\ LPAREN\ Formals\ RPAREN\ LBRACE\ Statements\ RBRACE$
48 $RetType \rightarrow Type$
49 $RetType \rightarrow VOID$
50 $Formals \rightarrow \epsilon$
51 $Formals \rightarrow FormalsList$
52 $FormalsList \rightarrow FormalDecl$
53 $FormalsList \rightarrow FormalDecl\ COMMA\ FormalsList$
54 $FormalDecl \rightarrow TypeAnnotation\ Type\ ID$
55 $Statements \rightarrow Statement$
56 $Statements \rightarrow Statements\ Statement$
57 $Statement \rightarrow LBRACE\ Statements\ RBRACE$
58 $Statement \rightarrow TypeAnnotation\ Type\ ID\ SC$
59 $Statement \rightarrow TypeAnnotation\ Type\ ID\ ASSIGN\ Exp\ SC$
60 $Statement \rightarrow ID\ ASSIGN\ Exp\ SC$
61 $Statement \rightarrow Call\ SC$
62 $Statement \rightarrow RETURN\ SC$
63 $Statement \rightarrow RETURN\ Exp\ SC$
64 $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement$
65 $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement\ ELSE\ Statement$
66 $Statement \rightarrow WHILE\ LPAREN\ Exp\ RPAREN\ Statement$
67 $Statement \rightarrow BREAK\ SC$
68 $Statement \rightarrow CONTINUE\ SC$
69 $Call \rightarrow ID\ LPAREN\ ExpList\ RPAREN$
70 $Call \rightarrow ID\ LPAREN\ RPAREN$
71 $ExpList \rightarrow Exp$
72 $ExpList \rightarrow Exp\ COMMA\ ExpList$
73 $Type \rightarrow INT$
74 $Type \rightarrow BYTE$
75 $Type \rightarrow BOOL$
76 $TypeAnnotation \rightarrow \epsilon$
77 $TypeAnnotation \rightarrow CONST$
78 $Exp \rightarrow LPAREN\ Exp\ RPAREN$
79 $Exp \rightarrow Exp\ BINOP\ Exp$
80 $Exp \rightarrow ID$
81 $Exp \rightarrow Call$
82 $Exp \rightarrow NUM$
83 $Exp \rightarrow NUM\ B$
84 $Exp \rightarrow STRING$
85 $Exp \rightarrow TRUE$
86 $Exp \rightarrow FALSE$
44 $Exp \rightarrow NOT\ Exp$
45 $Exp \rightarrow Exp\ AND\ Exp$
46 $Exp \rightarrow Exp\ OR\ Exp$
47 $Exp \rightarrow Exp\ RELOP\ Exp$
48 $Exp \rightarrow LPAREN\ Type\ RPAREN\ Exp$