מבחן סוף סמסטר – מועד א'

פתרון

מרצה אחראי: ד"ר הילה פלג

מתרגלים: הילה לוי, אלעד רון, תומר כהן, גיא ארבל

הוראות:

1. בטופס המבחן 16 עמודים, מתוכם 5 דפי נוסחאות. בדקו שכל העמודים ברשותכם.
2. משך המבחן שלוש שעות (180 דקות).
3. כל חומר עזר חיצוני אסור לשימוש.
4. בשאלות הפתוחות, ניתן לציין לגבי סעיף או שאלה "לא יודע/ת". תשובה זו תזכה ב-20% מהניקוד. תשובות שגויות לא יזכו בניקוד.
5. קראו את כל המבחן לפני שאתם מתחילים לענות על השאלות.
6. אין צורך להגיש את טופס מבחן זה בתום הבחינה.
7. את התשובות לשאלות הסגורות יש לסמן בטופס התשובות הנפרד בלבד. את התשובות לשאלות הפתוחות יש לכתוב במחברת הבחינה.
8. ודאו כי אתם מגישים טופס תשובות ומחברת בחינה בלבד.

התשובה הנכונה בשאלות האמריקאיות (בטופס זה) היא תמיד תשובה א'

בהצלחה!

חלק א' - שאלות סגורות (50 נק')

שלבי קומפילציה (20 נק')

נתונה התוכנית הבאה בשפת FanC:

```
1.  bool foo (int a, byte ca, byte fe, byte ba, byte be)
2.  {  1. byte
3.      int x0 = a;  -> cast (byte)a ——> semantic error
4.      int i = 0;
5.      while (i < 4) {    while(true) -> syntax error
6.          int x1 = x0 - x0 / 256 * 256;  x%(mod 256)   3. x1 = a % 256 <- lex error
7.          x0 = x0 / 256;  -> מילוי ערך אפס
8.          if (x1 != ca and x1 != fe and x1 != ba and x1 != be) {
9.              return false;
10.         }
11. 2. i++;  i = i + 1;  <- syntax error
12.     }
13.     return true;
14. }
```

בסעיפים הבאים (שאלות 1 עד 5) מוצגים שינויים **בלתי תלויים** לקוד של התוכנית. עבור כל שינוי ציינו את השלב המוקדם ביותר שבו נגלה את השגיאה:

שאלה מספר 1:

(2 נק') מחליפים את שורה 3 ב- "byte x0 = a;".

א. שגיאה בניתוח הסמנטי

ב. שגיאה בניתוח לקסיקלי

ג. שגיאה בניתוח תחבירי

ד. אין שגיאה

ה. שגיאה בייצור קוד

ו. שגיאה בזמן ריצה

שאלה מספר 2:

(2 נק') מחליפים את שורה 11 ל- "i++;".

א. שגיאה בניתוח תחבירי

ב. אין שגיאה

ג. שגיאה בניתוח לקסיקלי

ד. שגיאה בניתוח סמנטי

ה. שגיאה בייצור קוד

ו. שגיאה בזמן ריצה

שאלה מספר 3:

(2 נק') משנים את שורה 6 ל- "x1=a%256;".

א. שגיאה בניתוח לקסיקלי

ב. אין שגיאה

ג. שגיאה בניתוח תחבירי

ד. שגיאה בניתוח סמנטי

ה. שגיאה בייצור קוד

ו. שגיאה בזמן ריצה

שאלה מספר 4:

(2 נק') נוריד את הסוגריים המסולסלים שנמצאים בשורה 8 ובשורה 10.

א. אין שגיאה

ב. שגיאה בניתוח לקסיקלי

ג. שגיאה בניתוח תחבירי

ד. שגיאה בניתוח סמנטי

ה. שגיאה בייצור קוד

ו. שגיאה בזמן ריצה

שאלה מספר 5:

(2 נק') משנים את שורה 5 ל-"while {true}"

א. שגיאה בניתוח תחבירי

ב. אין שגיאה

ג. שגיאה בניתוח לקסיקלי

ד. שגיאה בניתוח סמנטי

ה. שגיאה בייצור קוד

ו. שגיאה בזמן ריצה

שאלה מספר 6:

(5 נק') נרצה להוסיף לשפת FanC את האופרטור הטרינארי שקיים בשפת c.
כידוע לכם, בשפת c ניתן לבצע את הפעולה הבאה:

```
int c = a==3 ? 4 : 3;
```

בשביל התמיכה בפיצ'ר החדש, באילו שלבי קומפילציה מבין הרשומים כאן, נצטרך לבצע שינויים?

א. נצטרך לערוך שינויים בכל אחד משלבי הקומפילציה שרשומים כאן.

ב. ניתוח לקסיקלי

ג. ייצור קוד

ד. ניתוח תחבירי

ה. ניתוח סמנטי

שאלה מספר 7:

(5 נק') נרצה להחליף את שפת הביניים של FanC בשפה שאין בה קפיצות מותנות. סמנו את התשובה הנכונה ביותר:

א. זה לא אפשרי כי while ו- if לא יעבדו

ב. זה אפשרי

ג. זה לא אפשרי כי int ו byte-לא יעבדו

ד. זה לא אפשרי כי כפל ו- if לא יעבדו

אופטימיזציות (10 נק')

נתון הקוד הבא בשפת הביניים, נוסיף את print לפעולות בשפת הרביעיות.

1. $y = y * y$
2. $y = y + 1$
3. $x = y$
4. $z = x$
5. $k = 2 + 1$
6. $k = k * 2$
7. $z = z * 2$
8. $x = z * k$
9. $y = y - 1$
10. $if\ y > 0\ goto\ 5$
11. $if\ y = 0\ goto\ 14$
12. $y = 2$
13. $goto\ 5$
14. $if\ z > 20\ goto\ 17$
15. $print\ z$
16. $goto\ 18$
17. $print\ 40$
18. $print\ end$

k = 3
k = 6 CP+CF
z = z << 1 AS
x = z * 6
y = y - 1
if

AS, ConstP, CF

שאלה מספר 8:

(5 נק') כמה leaders מזוהים על ידי אלגוריתם יצירת ה-CFG על הקוד הנתון?

א. 8

ב. 7

ג. 9

ד. 10

שאלה מספר 9:

(5 נק') על הבלוק שמתחיל בשורה 5 הריצו את האופטימיזציות הבאות: Algebraic Simplification, Constant Propagation, Constant Folding מספר רב של פעמים ככל הניתן.
מהי התוצאה אליה הגעתם?

(תיקון לשאלה במהלך המבחן: "בכל התשובות חסרה השורה $x = Z * 6$ לפני השורה $y = y - 1$")

א.

$$\begin{array}{c} k = 3 \\ k = 6 \\ z = z \ll 1 \\ x = z * 6 \\ y = y - 1 \\ if\ y > 0\ goto\ 5 \end{array}$$

ב.

$$\begin{array}{c} z = z \ll 1 \\ x = z * 6 \\ y = y - 1 \\ if\ y > 0\ goto\ 5 \end{array}$$

ג.

$$\begin{array}{c} k = 3 \\ k = k \ll 1 \\ z = z \ll 1 \\ x = z * 6 \\ y = y - 1 \\ if\ y > 0\ goto\ 5 \end{array}$$

ד.

$$\begin{array}{c} k = 6 \\ z = z \ll 1 \\ x = z * 6 \\ y = y - 1 \\ if\ y > 0\ goto\ 5 \end{array}$$

דקדוקים (20 נק')

יהי G_1 הדקדוק הבא:

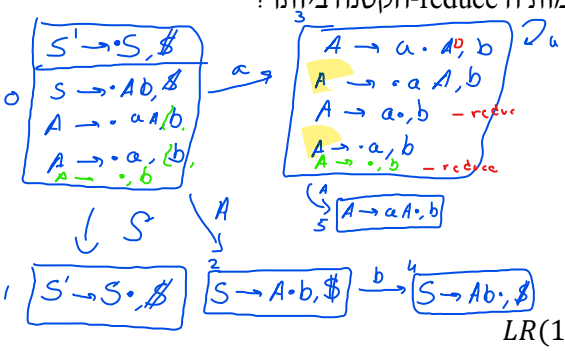
0 $S' \to S$
1 $S \to A\,b$
2 $A \to a\,A$
3 $A \to a$
(12) $A \to \varepsilon$

$S' \to \cdot S, \$$
$S \to \cdot A\,b, \$$
$A \to \cdot a A, b$
$A \to \cdot a, b$

שאלה מספר 10:

(4 נק') בטבלת ה-action של דקדוק G_1 במנתח $LR(1)$, באיזו עמודה כמות ה reduce-הקטנה ביותר?

	a	b	$	S	A
0	S3	r12		1	2
1			acc		
2		S4			
3	S3	r3/r12			5
4			r1		
5		r2			



א. a
ב. b
ג. $
ד. יש יותר מתו אחד

שאלה מספר 11:

(4 נק') כמה קשתות עם האות a מופיעות באוטומט עבור דקדוק G_1 במנתח $LR(1)$?

א. 2
ב. 1
ג. 3
ד. 4
ה. יותר מ- 4

שאלה מספר 12:

(4 נק') יהי G_2 דקדוק הזהה לדקדוק G_1 פרט להוספת הכלל $A \to \epsilon$.
מה ניתן להגיד על דקדוק G_2?

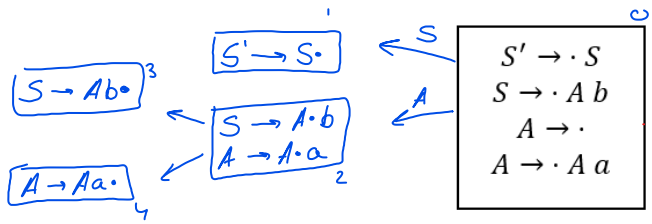
א. אף אחת מהתשובות לא נכונה
ב. נמצא ב-$LR(0)$
ג. נמצא ב-SLR
ד. נמצא ב-$LR(1)$

שאלה מספר 13:

(4 נק') יהי דקדוק: G_3

0 $S' \to \cdot S$
1 $S \to A\, b$
2 $A \to \varepsilon$
3 $A \to A\, a$

יוסי פיתח את אוטומט $LR(0)$ של דקדוק G_3 וקיבל את המצב ההתחלתי הבא:



יוסי טוען:

"במצב יש reduce לפי כלל $A \to \epsilon$ שיגרום לקונפליקט ולכן אין ספק שהדקדוק אינו ב- $LR(0)$".
סמנו את הטענה הנכונה ביותר.

	a	$	S	A
0	r2	r2	1	2
1	r0	r0		
2	s4	s3		
3	r1	r1		
4	r3	r3		

א. יוסי טועה והדקדוק כן ב- $LR(0)$

ב. יוסי צודק והדקדוק אינו ב- $LR(0)$

ג. יוסי טועה ועדיין הדקדוק אינו ב- $LR(0)$

ד. יוסי לא פיתח את המצב ההתחלתי נכון

שאלה מספר 14:

(4 נק') יהי דקדוק: G_4

$S \to A\, b \mid b\, C$
$A \to a$
$C \to c\, A$

$Sel(S \to Ab) = \{a\}$ $Sel(S \to bc) = \{b\}$
$Sel(A \to a) = \{a\}$
$Sel(C \to cA) = \{c\}$

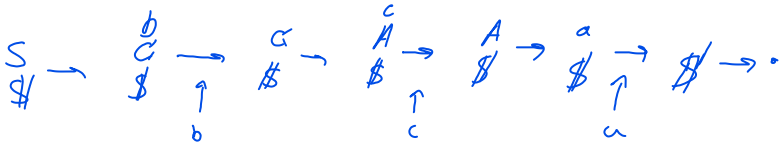
יוסי פיתח את המנתח $LL(1)$ של הדקדוק (הדקדוק נמצא במחלקה, אין צורך לוודא זאת) ובחר מילה כלשהי שנסמנה w.
יוסי התחיל להריץ את אלגוריתם המנתח ובאמצע המנתח הפסיק לעבוד.
מצב המחסנית כרגע הינו

M:

	a	b	c	$
S	$S \to Ab$	$S \to bC$		
A	$A \to a$			
C			$C \to cA$	

A
c
⋮

כלומר A נמצא בראש המחסנית.



סמנו את הטענה הנכונה ביותר.

1. יוסי טעה בהרצת אלגוריתם הניתוח
2. w בהכרח אינה בשפה
3. w בהכרח כן בשפה

1. bca$
2. ab$

חלק ב' - שאלות פתוחות (50 נק')

שאלה 1: ייצור קוד (20 נק')

ינון עשה טעות חמורה וניסה לתקן אותה על ידי הוספת מבנה בקרה חדש, if_according_to_b לשפת FanC:

הדקדוק שאיתו ממומש if_according_to_b הוא:

$$S \rightarrow if\_according\_to\_b\ (E\ ;\ B)\ \{L\}$$
$$L \rightarrow L_1\ ,S$$
$$L \rightarrow S$$

משמעות המבנה: נסמן את הערך של E ב-k. אם k שלילי או גדול ממספר ה-statements ש-L גוזר אין לבצע דבר.
אחרת, נבצע בדיקה של ערך הביטוי הבוליאני הנגזר מ- B ואז נפעל באופן הבא:

- אם ערך זה הוא **true** אז נבצע את k ה-statements **הראשונים**.
- אם ערך זה הוא **false** אז נבצע את k ה-statements **האחרונים**.

שימו לב, E מחושב בזמן ריצה, כלומר, ערכו יכול להיות שונה בריצות שונות.

דוגמה למבנה הבקרה החדש:

```
if_according_to_b (3 ; 2 < x) {
   printf("Best");
   printf("of");
   printf("luck");
   printf("to");
   printf("Inon");
   printf("and you");
}
```

עבור x=3 הערך שיודפס הוא: Best of luck
עבור x=1 הערך שיודפס הוא: to Inon and you

שימו לב:

- ניתן להניח שבקוד לא יהיו שגיאות קומפילציה
- אין לשנות את הדקדוק פרט להוספת מרקרים M, N
- ניתן להשתמש במרקרים M, N שנלמדו בכיתה בלבד
- למשתנים S, E, B ישנן התכונות שהוגדרו בכיתה בלבד
- למשתנים S, E, B ישנם כללי גזירה פרט לאלה המוצגים בשאלה
- אסור להשתמש במשתנים גלובליים
- המשתנה S תמיד יכיל nextlist

1. (7 נק') הציעו פריסת קוד המתאימה לשיטת backpatching עבור מבנה הבקרה הנ"ל. על הקוד הנוצר להיות יעיל ככל האפשר. הסבירו מהן התכונות הסמנטיות שאתם משתמשים בהן עבור כל משתנה.
2. (10 נק') כתבו סכימת תרגום בשיטת backpatching המייצרת את פריסת הקוד שהצעתם בסעיף הקודם. על הסכימה להיות יעילה ככל האפשר, הן מבחינת זמן הריצה שלה והן מבחינת המקום בזיכרון שנדרש עבור התכונות הסמנטיות.
3. (3 נק') הסבירו **בקצרה** כיצד הקוד היה משתנה לו הכלל הראשי כעת היה:

$$S \rightarrow if\_according\_to\_b\ (NUM\ ;\ B)\ \{L\}$$

ניתן להניח כי ל- NUM יש תכונה סמנטית value.

שאלה 2: אנליזה סטטית (30 נק')

רבקה מנהלת צוות פיתוח של קומפיילר ++C. היא רוצה לשפר את ביצועי הקוד שהקומפיילר מייצר, ולכן דורשת מהצוות שקריאה של פונקציה על אובייקט (למשל: x.foo()) תתורגם תמיד על ידי הקומפיילר ל-call ללייבל קבוע וידוע בזמן קומפילציה, וכך לחסוך את קריאת הזיכרון מה-dispatch vector. רוי, מפתח בצוות, טוען כי לא ניתן ליישם זאת.

א. (3 נק')

1. הסבירו בקצרה מה היה נחסך לו ניתן היה ליישם את הדרישה של רבקה ומדוע זה אכן היה מהיר יותר.
2. הסבירו מדוע רוי צודק, ולא ניתן ליישם את הדרישה של רבקה במקרה הכללי. בגלל שיש יירושים בעמ' רבים

טד, המוביל הטכני של הצוות, טוען שניתן ליישם את הדרישה בחלק מהמקרים. שותפו לשולחן בירד אומר שטד צודק, וקוראים לזה de-virtualization, ויש שני מקרים שבהם אפשר לעשות אופטימיזציה כזו. האחד פשוט, והשני ידרוש אנליזה שתעקוב אחרי הטיפוס הדינמי של אובייקטים שמשתנים מצביעים אליהם, ובעזרתה ניתן לבדוק מתי מותר לעשות את האופטימיזציה. עזרו לטד ובירד לפתח את האופטימיזציה.

ב. (4 נק') בירד וטד ניגשים למקרה בו יש להפעיל אנליזה כדי להחליט האם מותר להמיר קריאה לפונקציה על אובייקט ללייבל קבוע. הסבירו מתי מותר להמיר את הקריאה, ומה האנליזה תצטרך לוודא כדי להראות לנו מתי מותר לבצע את האופטימיזציה.

ג. (8 נק') הגדירו את האנליזה:

1. הגדירו את הדומיין: מהם האיברים? מהו יחס הסדר? מהי פעולת ⊔?
2. הגדירו את פונקציית המעברים עבור המשפטים השונים בדוגמה (השמה, משפט המכיל קריאה לפונקציה). הגדירו גם את הסמנטיקה עבור ביטויים רק אם אתם זקוקים לה.

```
class A {
public:
    virtual bool cond() {
       /*A's version*/ }
    virtual void foo() {
       /*A's version*/ }
};

class B : public A {
public:
    virtual bool cond() {
       /*B's version*/ }
    virtual void foo() {
       /*B's version*/ }
};

class C : public A {
public:
    virtual bool cond() {
       /*C's version*/ }
    virtual void foo() {
       /*C's version*/ }
};
```

```
class D {
public:
    virtual int xfunc() {
       /*D's version*/ }
    virtual void yfunc() {
       /*D's version*/ }
};

class E : public D {
public:
    virtual int xfunc() {
       /*E's version*/ }
};
```

```
1  void analyze_me(A *a, D *d, bool c1, bool c2) {
2      A *b = new B();
3      while (b->cond()) {
4          if (c1) {
5              b->foo();
6              b = a;
7          } else {
8              d = new E();
9              if (c2) {
10                 d->xfunc();
11             }
12         }
13     }
14     d->yfunc();
15 }
```

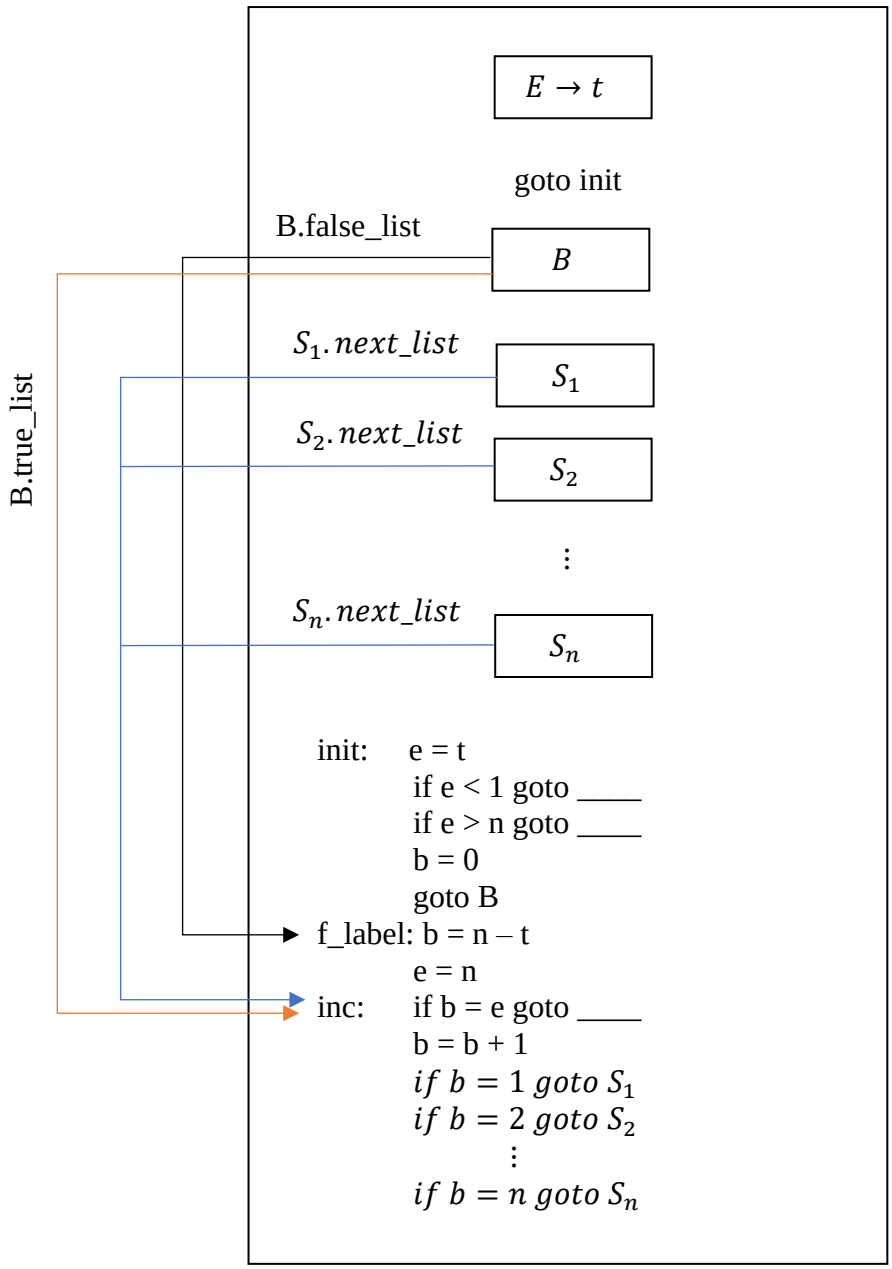
ד. (10 נק') בדקו את האנליזה שלכם על דוגמת הקוד של הצוות:
1. בנו את ה-CFG של הפונקציה analyze_me.
2. הריצו את האנליזה שלכם על הפונקציה. הנחה מקלה: ניתן להניח שאינכם יודעים דבר לגבי השיערוך של תנאים.

ה. (3 נק') לפי תוצאות האנליזה שלכם, פרטו האם ניתן לבצע את האופטימיזציה בשורות 3, 5, ו-10. הסבירו בקצרה כיצד האופטימיזציה תתבצע אם כן, או מדוע לא אם לא.

ו. (2 נק') רוי מצביע על שורה 14 בדוגמת הקוד ואומר שלמרות תוצאת האנליזה, ניתן להמיר את הקריאה לפונקציה ללייבל קבוע. הסבירו מדוע, והסבירו בקצרה את האופטימיזציה הכללית שניתן לגזור מכך.

שאלה 1: ייצור קוד- פתרון:

.1



התכונות הסמנטיות עבור S, B הן התכונות הסמנטיות שנלמדו בכיתה
התכונות הסמנטיות עבור L הן :

- next_list על מנת לשמור את כל כתובות ה goto מה-Sים אשר צריכים מילוי בהמשך.
- quad_list רשימה על מנת לשמור את כתובת ההתחלה של כל ה-Sים.

.2

```
L -> M S
{
        L.quad_list = make_list();
        L.quad_list.push(M.quad);
        L.next_list = S.next_list;
}

L -> L1, M S
{
        L.next_list = merge(L1.next_list, S.next_list);
        L.quad_list = L1.quad_list;
        L.quad_list.push(M.quad);
}

S -> if_according_to_b (E N ; M B) { L }
{
        bp(N.next_list, next_quad());
        e = new_temp();
        emit(e || " = " || E.place);
        S.next_list = make_list(next_quad());
        emit("if " || e || " < 1 goto ___")
        S.next_list = merge(S.next_list, make_list(next_quad()));
        emit("if " || e || " > " || L.quad_list.size() || "goto ___");
        b  = new_temp();
        emit(b || " = 0");
        emit("goto " || M.quad);
        bp(B.false_list, next_quad());
        emit(b || " = " || L.quad_list.size() || " - " || E.place);
        emit(e || " = " || L.quad_list.size());
        bp(B.true_list , next_quad());
        bp(L.next_list, next_quad());
        S.next_list = merge(S.next_list, make_list(next_quad()));
        emit("if " || b || " = " || e || "goto ____");
        emit( b || " = 1 +" || b);
        int i = 1;
        while(!(L.quad_list.empty()))
        {
                quad = L.quad_list.pop();
                emit("if " || b || " = " || i || "goto " || quad);
                i++;
        }
}
```

3. כעת, הערך של NUM קבוע בכל ריצה לעומת הערך של E שיכול להשתנות בין ריצות שונות.
על כן, נוכל לשערך את הקוד בהתאם לכך, שכן ידוע לנו כבר בשלב הקומפילציה מהו הערך של NUM, ובהתאם לכך מי הם הSים שנצטרך לבצע בהתאם לשערוך של B.
ישנם שני פתרונות אפשריים, האחד פשוט והשני מסובך יותר, שתי התשובות התקבלו כנכונות עם הסבר מתאים.
נשים לב שבשני הפתרונות לא ניתן לוותר על כתיבת הקוד של הSים לbuffer.
<u>פתרון פשוט-</u>
לכתוב את כל הקוד אותו הדבר בדיוק מלבד רצף הifים של ביצוע הSים שבו נכתוב רק את הSים הרלוונטיים לbuffer.
כלומר, את NUM הSים הראשונים וNUM הSים האחרונים.
<u>פתרון מסובך יותר-</u>
נשמור ברשימה את כל רשימות הnext_list של הSים.
כעת, נשים לב כי במידה ואין חפיפה בין הSים לביצוע בהתאם לערכו של NUM אז הפתרון הוא פשוט אך אם ישנה חפיפה נצטרך טיפול נוסף.
נפרוט את הפתרון למקרה של חפיפה:
נבצע bp של כל הSים אחד כלפי השני עם טיפול מיוחד עבור הS האחרון לביצוע במידה וB שוערך לtrue וטיפול מיוחד עבור S_n (הS- האחרון ברשימה).
נגדיר משתנה בוליאני X אשר לפי ערכו נדע אם B שוערך לtrue או לfalse.
במידה וB שוערך לtrue:
נקפוץ לקטע קוד שנותן למשתנה X ערך 1 ולאחריו נקפוץ לS_1.
במידה וB שוערך לfalse:
נקפוץ לקטע קוד שנותן למשתנה X ערך 0 ולאחריו נקפוץ ל$S_{NUM.value}$ (נשים לב כי כעת ניתן לדעת מי הוא הS הזה כיוון שNUM לא משתנה בריצות שונות).
במידה ויש חפיפה אז עבור $S_{NUM.value}$ (הS האחרון לביצוע כאשר B שוערך לtrue) נפנה את הnexl_list שלו לקוד אשר בודק את המשתנה הבוליאני X שהגדרנו ובמידה והוא שווה ל1 אז נצא אל מחוץ לקוד של S ואם הוא שווה ל0 אז נחזור ל$S_{NUM.value+1}$ ונמשיך לבצע את שאר הSים.
את S_n נפנה אל מחוץ לקוד של S.
אם אין חפיפה בין הSים עבור true וfalse לפי הערך של NUM אז את שניהם נפנה החוצה מהקוד של S.

שאלה 2: אנליזה סטטית- פתרון:

א. 1. הסבירו בקצרה מה היה נחסך לו ניתן היה ליישם את הדרישה של רבקה ומדוע זה אכן היה מהיר יותר.
כשנקראת פונקציה על אובייקט, כתובת ה-dispatch vector של האובייקט נטענת לרגיסטר מזיכרון האובייקט עצמו, מוסיפים לה את ה-offset של הפונקציה הנקראת, ואז מתבצעת קריאת זיכרון שניה לכתובת שחושבה על מנת לטעון את כתובת הגרסה של הפונקציה שרלוונטית לטיפוס הדינמי של האובייקט. לפיכך, קפיצה ישר לכתובת קבועה תחסוך שתי פקודות load.
2. הסבירו מדוע רוי צודק, ולא ניתן ליישם את הדרישה של רבקה במקרה הכללי
במקרה הכללי, מצביע לאובייקט יכול להצביע לאובייקט מכל טיפוס היורש מטיפוס המצביע (או הטיפוס עצמו), וכן האובייקט המוצבע יכול להתחלף בזמן ריצה. לכן, אם ברצוננו לתמוך בפולימורפיזם, אין מנוס אלא לבחור את הפונקציה הנכונה לפי הטיפוס הדינמי של האובייקט, על ידי גישה ל-DV.

ב. הסבירו מתי מותר להמיר את הקריאה, ומה האנליזה תצטרך לוודא כדי להראות לנו מתי מותר לבצע את האופטימיזציה.
במקרה ואנחנו יודעים בוודאות מה הטיפוס הדינמי של האובייקט, ניתן לקרוא ישירות לפונקציה שמוגדרת ב-DV עבור הטיפוס הזה במקום לפנות ל-DV על מנת להשיג את כתובתה. כדי לדעת את הטיפוס בוודאות, נצטרך לדעת שלא תיתכן ריצה כלשהי שבה יש אפשרות אחרת. לשם כך, ישנן שתי אנליזות אפשריות:

(1) האנליזה תעקוב אחרי כל המצביעים ותבדוק האם הטיפוס הדינמי של האובייקטים עליהם הם מצביעים הוא קבוע בכל נקודה, או
(2) האנליזה תעקוב אחרי כל הטיפוסים שיכולים להיות לאובייקטים שכל מצביע בתכנית מצביע אליהם בכל נקודה,

כדי שאם יכול להיות רק טיפוס אחד, נדע שניתן לבצע את האופטימיזציה המבוקשת.
מי שהיה ספציפי מספיק כאן כדי לציין שתנאי הנכונות הכללי יותר הוא כי לכל הטיפוסים שיכולים להיות יש את אותו המימוש לפונקציה קיבל נקודה ספייר.
ג. הגדירו את האנליזה
1. הגדירו את הדומיין
ישנם מספר דומיינים שאיתם ניתן לפתור את השאלה. השניים הנוחים ביותר (בהתאמה לתשובות לסעיף ב') הם:
נקרא לקבוצת כל המחלקות בתכנית $\mathcal{C}$

(1) דומיין הדומה לדומיין constant propagation: איברי הדומיין יהיו $L = \mathcal{C} \cup \{\bot, \top\}$. יחס הסדר הוא $\bot \sqsubseteq c \sqsubseteq \top$ לכל מחלקה $c \in \mathcal{C}$. פעולת $\sqcup$ הנובעת מיחס סדר זה היא:

$$\forall x \in L. \bot \sqcup x = x$$
$$\forall x \in L. x \sqcup \top = \top$$
$$\forall c_1, c_2 \in \mathcal{C}. c_1 \sqcup c_2 = \begin{cases} c_1 & c_1 = c_2 \\ \top & o.w. \end{cases}$$

(2) דומיין השומר את כל הטיפוסים העשויים להיות מוצבעים: איברי הדומיין יהיו תתי קבוצות של אובייקטים, $L = \mathcal{P}(\mathcal{C})$. מכיוון שאנחנו מעוניינים שכאשר משני מסלולים בתכנית מגיעים שני טיפוסים, התוצאה תהיה לשמור את שניהם (כלומר: להכשיל את האופטימיזציה), יחס הסדר יהיה $\sqsubseteq = \subseteq$, ונובע מכך כי $\sqcup = \cup$.

כעת נגדיר את הדומיין עבור תכנית מרובת משתנים: או דומיין מכפלה $L^{|Vars|}$ או פונקציה $Vars \to L$, כאשר Vars הם המשתנים הרלוונטיים בתכנית. השניים שקולים, והגדרת הסמנטיקה משתמשת בשני.
2. הגדירו את פונקציית המעברים עבור המשפטים השונים בדוגמה

אנחנו נדרשים לטפל במשפטים משלוש צורות: x=new Y() ,x=y, ו-x->func(). ניתן לטפל בשניים הראשונים ביחד ע"י איחודם ל-x=e והגדרת הסמנטיקה לביטויים שם משתנה וקריאה ל-constructor. הגרסה לכל דומיין כתובה בשיטה שונה, אבל ניתן להחליף בין השתיים.

(1) עבור דומיין CP:

$$[\![x = new\ Y(\quad)]\!]^{\#}\sigma^{\#} = \sigma^{\#}[x \mapsto Y]$$
$$[\![x = y]\!]^{\#}\sigma^{\#} = \sigma^{\#}[x \mapsto \sigma^{\#}(y)]$$
$$[\![x \to func()]\!]^{\#}\sigma^{\#} = \sigma^{\#}$$

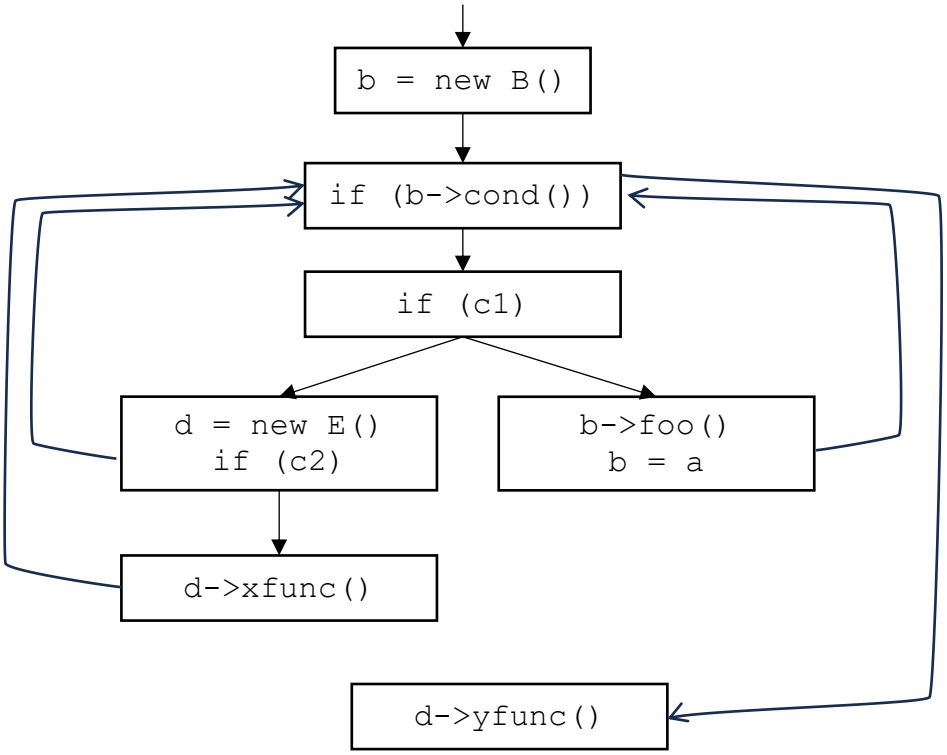
(2) עבור דומיין קבוצות הטיפוסים:

$$[\![x = e]\!]^{\#}\sigma^{\#} = \sigma^{\#}[x \mapsto [\![e]\!]^{\#}\sigma^{\#}]$$
$$[\![x \to func(\quad)]\!]^{\#}\sigma^{\#} = \sigma^{\#}$$
$$[\![new\ Y(\quad)]\!]^{\#}\sigma^{\#} = \{Y\}$$
$$[\![y]\!]^{\#}\sigma^{\#} = \sigma^{\#}(y)$$

על אף שלא ביקשנו זאת, מי שהגדיל והגדיר נכון (בשני הדומיינים) כי $[\![NULL]\!]^{\#}\sigma^{\#} = \bot$ קיבל פינוק.
מי שבחר בדומיין (2) יכול גם להגדיר ע"י kill/gen, אבל שימו לב כי אז x=y חייב לתת ל-x את הערך T או לכל היותר את קבוצת כל המחלקות שיורשות מטיפוס x, שכן מותר להיות תלויים רק בבלוק ולא בערך של in. למזל מי שבחר כך, זהו אובדן דיוק שלא משפיע על הדוגמה הנוכחית.

ד. בדקו את האנליזה שלכם על דוגמת הקוד של הצוות:

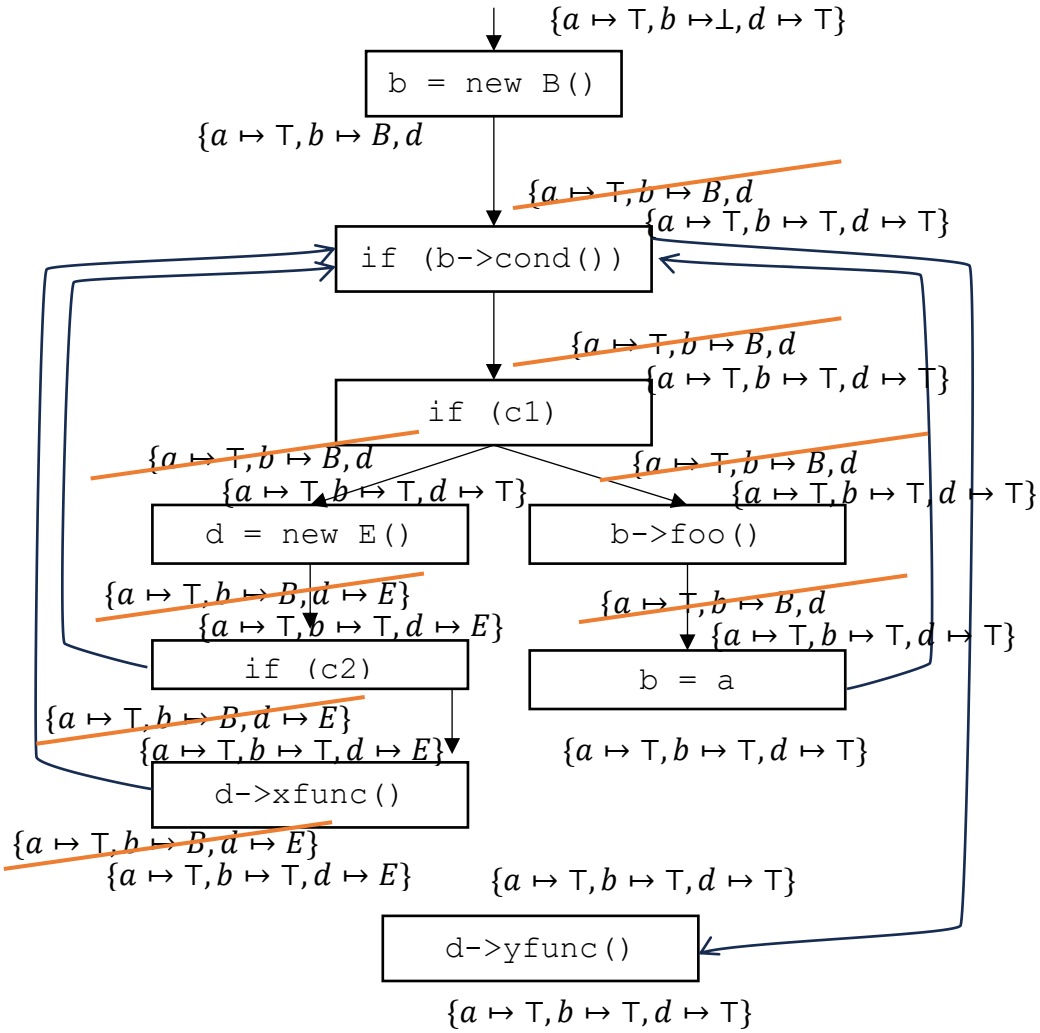
1. בנו את ה-CFG של הפונקציה me_analyze .



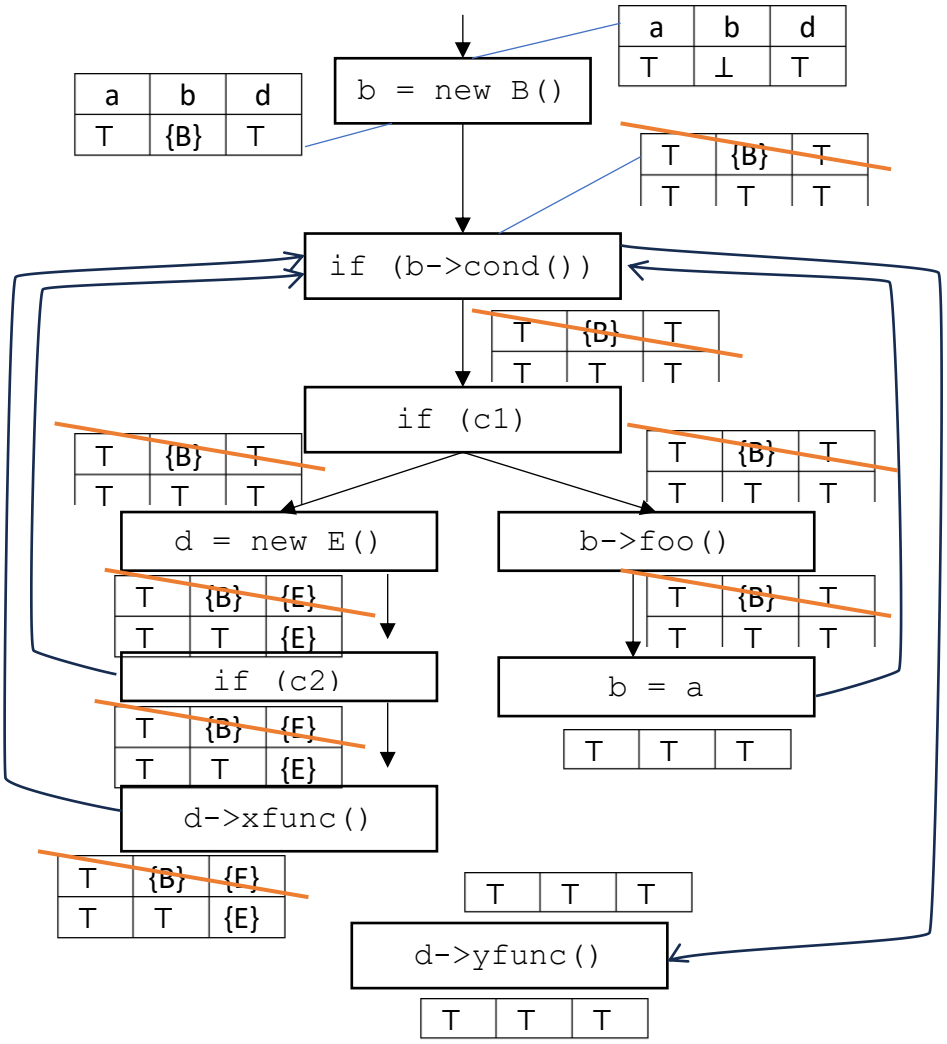
לשם הרצה נוחה יותר של האנליזה בתת-סעיף 2, נוכל להשתמש גם ב-CFG שבו כל בלוק בסיסי הוא פקודה אחת.

2. הריצו את האנליזה שלכם על הפונקציה.

כדי להריץ את האנליזה על הפונקציה, נשתמש בדומיין שהגדרנו כדי לעקוב אחרי כל המשתנים מסוג מצביע בתכנית: a,b,d. בכניסה לפונקציה b עדיין לא מאותחל, ולכן האתחול שלו הוא ⊥. שני הפרמטרים לפונקציה, a ו-d, הם פרמטרים ולא ידוע עליהם דבר. בשני הדומיינים האתחול הפשוט והנכון ביותר הוא ⊤. בדומיין (2) ניתן לצמצם את הקבוצה לכל הטיפוסים שמצביע מהטיפוס של הפרמטר יכול להצביע אליהם, ולאתחל את a להיות {A,B,C} ואת d להיות {D,E}.



או לפי דומיין 2:



ה. לפי תוצאות האנליזה שלכם, פרטו האם ניתן לבצע את האופטימיזציה בשורות 3, 5, ו-10. הסבירו בקצרה כיצד האופטימיזציה תתבצע אם כן, או מדוע לא אם לא.

לפני שורה 3 (כלומר, ב-in של הבלוק) בסוף האנליזה הערך של b הוא T בדומיין הראשון ו-T או {A,B,C} בדומיין השני. זאת אומרת שבנקודות שונות בריצות שונות יכול להיות כאן כל אובייקט, ואין לנו ברירה אלא לוותר על האופטימיזציה. בשורה 5 הערך של b הוא אותו הדבר ולכן המצב זהה. בשורה 10, לעומת זאת, הערך של d הוא E בדומיין הראשון או {E} בדומיין השני, ולכן בוודאות המצביע יצביע לאובייקט מסוג E – אז לפי האינפורמציה שהאנליזה סיפקה ניתן להחליף את קוד הקריאה מה-DV ב-call ישיר ל-E::xfunc.

ו. רוי מצביע על שורה 14 בדוגמת הקוד ואומר שלמרות תוצאת האנליזה, ניתן להמיר את הקריאה לפונקציה לייבל קבוע. הסבירו מדוע, והסבירו בקצרה את האופטימיזציה הכללית שניתן לגזור מכך

מכיוון שהמחלקה E יורשת את yfunc מ-D ולא דורסת אותו, ב-DV של שתיהן באינדקס של yfunc תהיה הכתובת של D::yfunc. לכן אם נקפוץ ישר ל-D::yfunc תמיד נקפוץ למקום הנכון. ניתן לגזור מכך את האופטימיזציה שאם לכל הטיפוסים שניתן להצביע אליהם עם המצביע הנוכחי יש את אותו מימוש לפונקציה, ניתן להמיר את הקריאה בקריאה קבועה. לא נדרשת אנליזה לשם כך, זו בדיקה סמנטית בלבד.

אלה מכם שהשתמשו בדומיין (2) והציעו בתור הפתרון הכללי שלהם שניתן לבדוק עבור כל הטיפוסים בערך האבסטרקטי של משתנה מהו המימוש של פונקציה, ואם לכל הטיפוסים בערך האבסטרקטי יש את אותו המימוש, ניתן להחליף את הקריאה בקריאה קבועה טופלו כמו אלה שהציעו את האופטימיזציה הזו כבר בסעיף ב'.

נוסחאות ואלגוריתמים

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

Top Down

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

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

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

<u>הגדרה:</u> דקדוק G הוא LL(1) <u>אם ורק אם</u> לכל שני כללים ב- G השייכים לאותו משתנה A מתקיים:

$$\text{select}(A \to \alpha) \cap \text{select}(A \to \beta) = \varnothing$$

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

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

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 מהצורה $G = (V, E)$:

הצורה הכללית של המשוואות בחישוב סריקה קדמית:

$$\mathrm{in}(B) = \bigsqcup_{(S,B)\in E} \mathrm{out}(S)$$
$$\mathrm{out}(B) = f_B\big(\mathrm{in}(B)\big)$$

הצורה הכללית של המשוואות בחישוב סריקה אחורית:

$$\mathrm{in}(B) = \bigsqcup_{(B,D)\in E} \mathrm{out}(D)$$
$$\mathrm{out}(B) = f_B\big(\mathrm{in}(B)\big)$$

Abstract Interpretation

בהינתן סריג L עם יחס סדר חלקי $\sqsubseteq$ ופעולת join $\sqcup$:

מצב אבסטרקטי הוא $\sigma^{\#} \in A = (Var \to L)$

ערך אבסטרקטי של משתנה $x \in \mathrm{Var}$ בהינתן מצב אבסטרקטי: $\sigma^{\#}(x)$

סמנטיקה אבסטרקטית של ביטוי e: $[\![e]\!]^{\#} : A \to L$

סמנטיקה אבסטרקטית של משפט s: $[\![s]\!]^{\#} : A \to A$

ערך אבס' של ביטוי e או מצב אבס' אחרי משפט s בהינתן מצב אבס' קודם: $[\![e]\!]^{\#}\sigma^{\#}$ / $[\![s]\!]^{\#}\sigma^{\#}$

מצב אבס' אחרי השמה של ערך אבס' $v \in L$ למשתנה $x \in \mathrm{Var}$: $\sigma^{\#}[x \mapsto v]$

שפת FanC

אסימונים:

אסימון	תבנית
VOID	void
INT	int
BYTE	byte
B	b
BOOL	bool
OVERRIDE	override
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:

1. $Program \rightarrow Funcs$
2. $Funcs \rightarrow \epsilon$
3. $Funcs \rightarrow FuncDecl\ Funcs$
4. $FuncDecl \rightarrow\ OverRide\ RetType\ ID\ LPAREN\ Formals\ RPAREN\ LBRACE\ Statements\ RBRACE$
5. $OverRide \rightarrow \epsilon$
6. $OverRide \rightarrow OVERRIDE$
7. $RetType \rightarrow Type$
8. $RetType \rightarrow\ VOID$
9. $Formals \rightarrow \epsilon$
10. $Formals \rightarrow FormalsList$
11. $FormalsList \rightarrow FormalDecl$
12. $FormalsList \rightarrow FormalDecl\ COMMA\ FormalsList$
13. $FormalDecl \rightarrow Type\ ID$
14. $Statements \rightarrow Statement$
15. $Statements \rightarrow Statements\ Statement$
16. $Statement \rightarrow LBRACE\ Statements\ RBRACE$
17. $Statement \rightarrow Type\ ID\ SC$
18. $Statement \rightarrow Type\ ID\ ASSIGN\ Exp\ SC$
19. $Statement \rightarrow ID\ ASSIGN\ Exp\ SC$
20. $Statement \rightarrow Call\ SC$
21. $Statement \rightarrow RETURN\ SC$
22. $Statement \rightarrow RETURN\ Exp\ SC$
23. $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement$
24. $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement\ ELSE\ Statement$
25. $Statement \rightarrow WHILE\ LPAREN\ Exp\ RPAREN\ Statement$
26. $Statement \rightarrow BREAK\ SC$
27. $Statement \rightarrow CONTINUE\ SC$
28. $Call \rightarrow ID\ LPAREN\ ExpList\ RPAREN$
29. $Call \rightarrow ID\ LPAREN\ RPAREN$
30. $ExpList \rightarrow Exp$
31. $ExpList \rightarrow Exp\ COMMA\ ExpList$
32. $Type \rightarrow INT$
33. $Type \rightarrow BYTE$
34. $Type \rightarrow BOOL$
35. $Exp \rightarrow LPAREN\ Exp\ RPAREN$
36. $Exp \rightarrow Exp\ BINOP\ Exp$
37. $Exp \rightarrow ID$
38. $Exp \rightarrow Call$
39. $Exp \rightarrow NUM$
40. $Exp \rightarrow NUM\ B$
41. $Exp \rightarrow STRING$
42. $Exp \rightarrow TRUE$
43. $Exp \rightarrow FALSE$
44. $Exp \rightarrow NOT\ Exp$
45. $Exp \rightarrow Exp\ AND\ Exp$
46. $Exp \rightarrow Exp\ OR\ Exp$
47. $Exp \rightarrow Exp\ RELOP\ Exp$
48. $Exp \rightarrow LPAREN\ Type\ RPAREN\ Exp$