מבחן סוף סמסטר – מועד ב׳

טור מקור

מרצה אחראי: ד״ר שחר יצחקי

מתרגלים: מתן פלד, אלעד רון, תומר כהן, הילה לוי

הוראות:

1. בטופס המבחן 14 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 בטופס זה בשאלות האמריקאיות היא תמיד תשובה א׳.

בהצלחה!

חלק א׳ - שאלות סגורות (50 נק׳)

שלבי קומפילציה (20 נק׳)

נתונה התוכנית הבאה בשפת FanC:

```
1   bool is_divisible_by_2(int a)
2   {
3      int tmp = a;
4      while (tmp > 0) {
5         tmp = tmp -2;
6      }
7      return tmp == 0;
8   }
9   bool three_n_plus_1(int a)
10  {
11     if (a == 1) {
12        return true;
13     }
14     if (is_divisible_by_2(a)) {
15        return three_n_plus_1(a/2);
16     }
17     else {
18        return three_n_plus_1(3*a + 1);
19     }
20  }
21  int main()
22  {
23     three_n_plus_1(30);
24     return 0;
25  }
```

ניתן להניח כי השפה הורחבה כך ש-ID יכולה להכיל קו תחתון. הפונקציה three_n_plus_1 בודקת שהמספר a מקיים את סדרת 3n+1 (בסדרה זאת, אם מספר כלשהו הוא זוגי אז המספר הבא בסדרה יהיה חלוקה שלו ב-2, אחרת נכפיל אותו ב-3 ונוסיף 1. בתיאוריה ישנה הנחה כי לכל מספר התחלתי שיתחיל את הסדרה נקבל סדרה שתסתיים ב-1.).

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 7 ב- return tmp is 0;.

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 2:
(2 נק׳)

מחליפים את שורה 23 ב- three_n_plus_1(0);.

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

שאלה מספר 3:
(2 נק׳)

מחליפים את טיפוס החזרה של three_n_plus_1 ל-byte.

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

שאלה מספר 4:
(2 נק׳)

מחליפים את שורה 11 ב- if(a=1).

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 5:
(2 נק׳)

משנים את שורה 3 ל- int tmp = (byte)a;.

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 6:
(5 נק׳)

נרצה להוסיף לשפת FanC תמיכה בלולאות For. באיזה שלב קומפילציה, מבין הרשומים כאן, לא נבצע שינויים?

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

שאלה מספר 7:
(5 נק׳)

בעת מימוש תמיכה בלולאות For עבור FanC, בשלב ייצור הקוד, האם יש להשתמש ב-backpatching כפי שהי נלמדה בכיתה?

א. כן, ונצטרך לקרוא גם ל-emit וגם ל-backpatch במימוש.
ב. כן, אבל לא נצטרך לעשות emit לקוד נוסף, אפשר לפתור רק באמצעות קריאות ל-backpatch.
ג. לא, כי לא נעשה שינויים בשלב ייצור הקוד.
ד. לא, השינויים בשלב ייצור הקוד לא דורשים ביצוע של backpatching
ה. כן, אבל רק אם מאתחלים משתנה חדש בכותרת של הלולאה.

אופטימיזציות (10 נק׳)

שאלה מספר 8:
(5 נק׳)
נתונה התוכנית הבאה בשפת הרביעיות שלמדנו בכיתה:

```
1   n = 0
2   x = 0
3   y = 1
4   z = 2
5   if x > y goto 15
6   t1 = x
7   t2 = y
8   x = y + z
9   y = t1 + z
10  z = t1 + t2
11  if z > x goto 8
12  if z > y goto 9
13  n = n + 1
14  goto 5
15  ret = x - n
```

בנו את ה-CFG של התוכנית (כטיוטא, אין צורך להגיש) וענו על השאלה הבאה לפי הגרף שיצרתם:

ו. יש 8 בלוקים בסיסיים והדרגת כניסה המקסימלית היא 2
ז. יש 7 בלוקים בסיסיים והדרגת כניסה המקסימלית היא 2
ח. יש 9 בלוקים בסיסיים והדרגת כניסה המקסימלית היא 2
ט. יש 8 בלוקים בסיסיים והדרגת כניסה המקסימלית היא 3
י. יש 7 בלוקים בסיסיים והדרגת כניסה המקסימלית היא 3
יא. יש 9 בלוקים בסיסיים והדרגת כניסה המקסימלית היא 3

שאלה מספר 9:
(5 נק׳)
נתבונן בקטע הקוד הבא. ידוע שהקוד ממשיך אחרי הקטע, כלומר שורה 11 היא יעד קפיצה חוקי:

```
16  x = 2
17  y = 3
18  ret = 0
19  if y == 0 goto 11
20  t1 = x
21  t2 = t1 + ret
22  ret = t2 + 0
23  t3 = y
24  y = t3 - 1
25  goto 4
```

מפעילים על הקוד את האופטימיזציות הבאות: copy propagation, constant propagation, arithmetic simplification, dead code elimination, useless code elimination, common sub-expression elimination, constant folding, branch chaining, המרת קפיצה מותנית בקפיצה לא-מותנית.
האופטימיזציות מופעלות בסדר אופטימלי כך שידוע שהפעלה נוספת של אופטימיזציה מהרשימה לא תגרום לשינוי בקוד.
בנוסף, ידוע כי בהמשך הקוד יש שימוש רק ב-ret ולא באף משתנה אחר המופיע בקטע הקוד.

מה תהיה שורה 1 בקוד לאחר האופטימיזציות?

יב. y = 3
יג. x = 2
יד. ret = 0
טו. ret = 6

דקדוקים (20 נק׳)

שאלות 11-10:
ברצוננו לבנות מנתח עבור דקדוקים כאשר מחיר הבנייה של מנתח מחושב לפי מחיר המחסנית בה המנתח משתמש.
קיימות מחסניות משני סוגים:
סופית - מחיר מחסנית בגודל X זוגות הוא 10 * X שקלים
אינסופית - מחיר מחסנית כזו הוא 1000 שקלים
כזכור, כל זוג שנכנס למחסנית מכיל מספר מצב ואלמנט בדקדוק.

שאלה מספר 10:
(4 נק׳)

$$S \to a\ A\ b$$
$$A \to A\ B\ c\ A\ d \mid \epsilon$$
$$B \to b$$

בשאלה זו נבנה מנתח SLR, מהו המחיר המינימלי למנתח עבור כל אחד מהדקדוקים הבאים?

א. 701-1200
ב. 0-300
ג. 301-700
ד. הדקדוק אינו ב-SLR

שאלה מספר 11:
(4 נק׳)

$$S \to a\ A\ b$$
$$A \to d\ B\ a\ d$$
$$B \to b$$

בשאלה זו נבנה מנתח LR(1), מהו המחיר המינימלי למנתח עבור כל אחד מהדקדוקים הבאים?

א. 0-600
ב. 601-1200
ג. 1201-2000
ד. הדקדוק אינו ב-LR(1)

שאלה מספר 12:
(4 נק׳)

נתון הדקדוק הבא:

$$S \to a\ B \mid B\ C \mid C\ B\ d \mid b$$
$$B \to b \mid \epsilon$$
$$C \to c \mid \epsilon$$

לפי הטבלה המתקבלת ממנתח LL(1) מה מספר הכללים הגדול ביותר שמכיל תא בודד בטבלה?
א. 3
ב. 2
ג. 4
ד. הדקדוק ב LL(1)

שאלה מספר 13:
(4 נק׳)

ברצוננו לבנות מנתח דמוי LL(1) עבור דקדוקים כאשר מחיר הבנייה הוא כדלקמן:
מחיר טבלה המכילה X ערכים בתוכה (**לכל גודל של טבלה**) הוא X שקלים
הערה – המחיר מתאים לכל טבלה הנוצרת מדקדוק כללי כפי שנלמד בתרגול, במקרה ותא מכיל מספר ערכים הם נחשבים כערכים **שונים**.

אנה ויוסי קיבלו דקדוק G שנמצא ב LL(1) המכיל 1000 כללים **שונים**. יוסי התחיל לחשב את ערכי ה- first וה-follow כפי שלמד בקורס תורת הקומפילציה, אך אנה עצרה אותו ואמרה "אנחנו יודעים שהדקדוק נמצא ב LL(1) ולכן כל תא מכיל לכל היותר כלל אחד ולכן המחיר של הטבלה למנתח חייבת להיות בדיוק 1000"
הניחו שכל דקדוק בשאלה הינו דקדוק שבו **כל הכללים ישיגים**, כלומר לכל משתנה X בדקדוק קיימת גזירה ממשתנה הגזירה ההתחלתי S בו X מופיע.

מה הטענה החזקה ביותר?

א. אנה טועה והמחיר יכול להיות גדול ממש מ 1000 עבור דקדוק ב LL(1)
ב. אנה צודקת גם עבור דקדוקים שאינם LL(1)
ג. אנה צודקת רק עבור דקדוקים שנמצאים LL(1)
ד. אנה טועה והמחיר יכול להיות קטן ממש מ 1000 עבור דקדוק ב LL(1)

שאלה מספר 14:
(4 נק׳)
נתון קובץ ה-Bison הבא.

```
%{
#include <cstdio>
#include <iostream>
extern int yylval;
int yylex(void) {
  char c = getchar();
  if (c <= '9' && c >= '0') { yylval = c - '0'; return NUM; }
  return c;
}
void yyerror(const char*) {}
int yyparse();
int main() { return yyparse(); }
%}

%token NUM
// precedence: highest to lowest
%right '$'
%left '#'
%right '@'

%%

prog: prog line
| ;
line : exp '\n' { std::cout << $1 << '\n'; } ;
exp :    NUM {$$ = $1;}
| exp '$' exp {$$ = $1 - $2;}
| exp '#' exp {$$ = $1 * $2;}
| exp '@' exp {$$ = $1 + $2;}
;
```

הפונקציה yylex המופיעה בתחילת הקובץ מהווה מנתח לקסיקלי פשוט שהופך כל תו בודד לאסימון: ספרות מפורשות כאסימון מסוג NUM שהערך הסמנטי שלו הוא הערך של הספרה, ושאר התווים מפורשים כאסימון שזהה לתו שנקרא מהקלט.
איזה מבין הקלטים הבאים מדפיס את הערך המספרי **הגדול** ביותר?

א. 1@2$3$4#5
ב. 1@2#3$4$5
ג. 1$2$3@4@5
ד. 1@2@3$4$5

חלק ב׳ - שאלות פתוחות (50 נק׳)

שאלה 1: ייצור קוד (20 נק')

הוסיפו לשפת FanC מבנה בקרה חדש, total_switch. הדקדוק שאיתו ממומש total_switch הוא:

$$S \rightarrow total\_switch\ (E\ ;B\ )\ \{\ CL\ \}$$
$$CL\ \rightarrow C\ CL1$$
$$\mathrm{CL}\ \rightarrow \mathrm{C}$$
$$C \rightarrow case\ (\ num\ )\ :\ S\ ;$$

המבנה החדש מבוצע באופן הבא, עבור כל ה- cases לפי הסדר: לכל case בצע בדיקה של ערך הביטוי הבוליאני הנגזר מ-B ואז תפעל באופן הבא:

- אם ערך זה הוא **true** אז נבצע את ה-case רק במידה ומתקיים כי המספר שב-case **שווה** לערך של E.
- אם ערך זה הוא **false** אז נבצע את ה-case רק במידה ומתקיים כי המספר שב- case **שונה** מהערך של E.

לאחר כל שלב נשערך מחדש את הערך של הביטוי הבוליאני, עד אשר עברנו על כל ה- cases ולאחר מכן נצא מהמבנה. דוגמה למבנה הבקרה החדש:

```
total_switch (3 ; func()) {
        case (0) : printf("I");
        case (3) : printf("I");
        case (2) : printf("will");
        case (3) : printf("can");
        case (3) : printf("do");
        case (3) : printf ("it!");
        case (4) : printf("do");
        case (6) : printf("it!");
}
```

עבור func שתמיד מחזירה true יודפס: I can do it!

עבור func שתמיד מחזירה false יודפס: I will do it!

עבור func שמחזירה true, true, true, true, false, false, true, true (משמאל לימין) יודפס: I can

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. 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. 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.

שאלה 2: אנליזה סטטית (30 נק')

במפקדת החלל האמריקנית נמשכים המאמצים לפתח קומפיילר עבור תוכניות חלל. סג״מ בוימלר קיבל מפרט של שפת ביניים מסוג רביעיות (3AC) שבה המשתנים הם מסוג int, ופקודה להריץ על התוכניות שלהלן אנליזה לבדיקת טווחי הערכים של המשתנים.

```
1  n = 0
2  if n >= 7 goto 8
3  n = n + 1
4  i = 0
5  if i >= n goto 2
6  i = i + 2
7  goto 5
8  return
```

א. (3 נק') הריצו אנליזת אינטרוולים (interval analysis) על התוכנית הזו כאשר כל שורה בתכנית היא בלוק בסיסי יחיד וכתבו את המצב האבסטרקטי שמתקבל **לפני** כל הוראה (in(B)). הניחו שבמימוש של בוימלר **אין widening**. טיפ מודיעיני. הסמנטיקה האבסטרקטית של תנאים מהצורה v >= e (כאשר v הוא משתנה ו-e הוא אופרנד שיכול להיות קבוע או משתנה) היא:

$$[\![\text{if } v >= e]\!]^{\#}\sigma^{\#} = \begin{cases} (\text{true}) & \sigma^{\#}[v \mapsto \sigma^{\#}v \sqcap [\inf [\![e]\!]^{\#}\sigma^{\#}, \infty]] \\ (\text{false}) & \sigma^{\#}[v \mapsto \sigma^{\#}v \sqcap [-\infty, \sup [\![e]\!]^{\#}\sigma^{\#} - 1]] \end{cases}$$

(inf מציין את הגבול התחתון של האינטרוול, sup את הגבול העליון שלו; ולמען הסר ספק, $\infty - 1 = \infty$).

שימו לב. **אין צורך** לכתוב את חישובי הביניים באנליזה.

ב. (2 נק') סג״מ מרינר רצתה להוכיח כי בשורה 6 (לפני ביצוע ההשמה) מתקיים התנאי $i < n$. הסבירו לה מדוע **לא ניתן** לקבוע זאת בהסתמך על תוצאות האנליזה הזו בלבד.

ג. (10 נק') על מנת לנסות לעזור לחבריה, סג״מ טנדי מגדירה את הדומיין הבא לאנליזה סטטית:

$$A = (\text{Vars} \times \text{Vars}) \to (\mathbb{Z} \cup \{-\infty\} \times \mathbb{Z} \cup \{\infty\})$$

כאשר Vars היא קבוצת המשתנים בתוכנית; מצב אבסטרקטי בדומיין הזה הוא מיפוי בין זוגות של משתנים לבין אינטרוולים של מספרים שלמים. יחס הסדר $\sqsubseteq$ הוא היחס הרגיל של דומיין האינטרוולים (pointwise). המשמעות של איבר $\sigma^{\#} \in A$ היא ש $\sigma^{\#}(\langle v_1, v_2\rangle)$ מייצג את טווח הערכים שיכול לקבל **ההפרש** $v_1 - v_2$.

עזרו לטנדי להשלים את הסמנטיקה האבסטרקטית של המשפטים הבאים בשפת הביניים:

$[\![v = v + c]\!]^{\#}\sigma^{\#} =$ (c קבוע, $v \in \text{Vars}$ משתנה)

$[\![\text{if } v_1 >= v_2]\!]^{\#}\sigma^{\#} =$ ($v_1, v_2 \in \text{Vars}$ משתנים)

ד. (5 נק') סג״מ מרינר החליטה לקצר תהליכים ולהגדיר את הסמנטיקה של שאר המשפטים בשפה כך:

$[\![v = \_\,]\!]^{\#}\sigma^{\#} = \sigma^{\#}[\langle v, v_1\rangle \mapsto \top, \langle v, v_2\rangle \mapsto \top, \ldots, \langle v_1, v\rangle \mapsto \top, \langle v_2, v\rangle \mapsto \top, \ldots]$

($\text{Vars} = \{v_1, v_2, \ldots\}$, ולכל השמה שאינה מהצורה שהופיעה בסעיף הקודם)

$[\![s]\!]^{\#}\sigma^{\#} = \sigma^{\#}$

(לכל יתר המשפטים)

(כזכור, $\top = [-\infty, \infty]$.)

הריצו את האנליזה הזו על התוכנית שבתחילת השאלה. הראו שכעת ניתן להוכיח את התכונה שמרינר ביקשה בסעיף ב. (טיפ. אם לא ניתן, כנראה שטעיתם בסעיף הקודם.)

ה. (10 נק') סג״מ ראת׳רפורד כתב את התוכניות הבאות:

1 x = 5	1 x = 5	1 x = 5
2 y = 10	2 y = 10	2 y = 10
3 c = read_int()	3 c = read_int()	3 c = read_int()
4 if c > 0 goto 7	4 if c > 0 goto 7	4 if c > 0 goto 7
5 x = x + (-5)	5 y = x	5 x = 0
6 y = y + (-5)	6 x = x + (-5)	6 y = 5
7 return	7 return	7 return

(דרך אגב, לראת׳רפורד יש שתל תוך-עיני והוא מדווח לנו שכל שלוש התוכניות שקולות סמנטית.)

הוא רוצה להוכיח שבשורה 7 מתקיים התנאי $x < y$. הוא הריץ את אנליזת האינטרוולים של בוימלר וקיבל (בכל המקרים)

$in(7) = \{x \mapsto [0, 5], y \mapsto [5, 10]\}$

הוא הריץ את אנליזת ההפרשים של טנדי-מרינר (סעיפים ג, ד) וקיבל (בכל המקרים)

$in(7) = \{(x, y) \mapsto \top, (y, x) \mapsto \top, \ldots\}$

ראת׳רפורד התאכזב מאד מכך שאפילו עם תוצאות שתי האנליזות לא עלה בידו להוכיח את התכונה. (ודאו שגם אתם מבינים זאת ומאוכזבים באותה מידה.)

תקנו את האנליזה של טנדי-מרינר כך שתטפל גם בהשמות מהצורה הבאה:

$[\![\ell\ v = c]\!]^{\#}\sigma^{\#} =$

$[\![\ell\ v_1 = v_2]\!]^{\#}\sigma^{\#} =$ (c קבוע, $v, v_1, v_2 \in Vars$ משתנים)

$[\![\ell\ v_1 = v_2 + c]\!]^{\#}\sigma^{\#} =$

הדרכה חשובה:

- התווית ℓ היא מספר השורה שבה נמצאת ההשמה.
- ניתן וצריך להשתמש בתוצאות האנליזה של בוימלר (האינטרוולים) לצורך הגדרת המעברים הללו. השתמשו בסימון $Bo[in(\ell)]$ עבור המצב האבסטרקטי בדומיין האינטרוולים בכניסה לשורה ℓ ובסימון $Bo[out(\ell)]$ עבור המצב ביציאה מהשורה ℓ. זכרו שמדובר בסריג מכפלה – כדי לקבל את הערך המתאים למשתנה מסוים כתבו, למשל, $Bo[in(\ell)](v)$.
- הגדירו אנליזה מדויקת ככל שתוכלו. ודאו (לעצמכם) שהאנליזה מצליחה להוכיח את התכונה $x < y$ בשורה 7 של כל אחת משלוש התוכניות שבתחילת הסעיף, כדי שראת׳רפורד לא יתאכזב שוב כבראשונה.

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