28.02.2017

מבחן סוף סמסטר – מועד ב'

מרצה אחראי: ד"ר אוהד שחם

מתרגלים: אבנר אליזרוב מיכל בדיאן הילה פלג עידן שורץ

הוראות:

א. בטופס המבחן 11 עמודים מהם 4 דפי נוסחאות. בדקו שכל העמודים ברשותכם.

ב. משך המבחן שלוש שעות (180 דקות).

ג. אסור כל חומר עזר פרט לדף הנוסחאות המצורף לבחינה.

ד. במבחן 5 שאלות. כל השאלות הינן חובה. משקל כל שאלה מופיע בראשיתה. (חלוקת המשקל בין הסעיפים בכל שאלה אינה בהכרח אחידה.)

ה. ניתן לציין לגבי סעיף או שאלה "לא יודע/ת". תשובה זו תזכה ב- 20% מהניקוד של הסעיף או השאלה. תשובות שגויות לא יזכו בניקוד.

ו. חובה לנמק כל תשובה. לא יינתן ניקוד על תשובות ללא נימוק.

ז. קראו את כל המבחן לפני שאתם מתחילים לענות על השאלות.

ח. אין צורך להגיש את הטופס בתום הבחינה.

ט. את התשובות לשאלות יש לרשום במחברת הבחינה בלבד.

בהצלחה!

שאלה 1 (12 נק'):סיווג מאורעות

נתון קטע הקוד הבא בשפת C:

```
int A[10];
int i = 0;
scanf("%d",&x);
```

בסעיפים הבאים ישנה **גישה שגויה** למערך. נמקו בקצרה מהו השלב המוקדם ביותר שבו תתגלה הגישה השגויה. בתשובתכם התייחסו לשלבים לקסיקלי, תחבירי, סמנטי ,אופטימיזציה וזמן ריצה.

1. A["i"]
2. A[x]
3. ש[9]
4. A[i+1.1]
5. A[i+1)
6. A(i)

שאלה 2 (28 נק'): רשומות הפעלה

אנו מעוניינים לבנות מנגנון רשומות הפעלה חדש כך שבכל שלב במהלך ריצת התוכנית לכל פונקציה תהיה לכל היותר רשומת הפעלה אחת.

1. (12 נקודות) תארו כיצד הייתם בונים את מנגנון רשומות ההפעלה. התיאור יכול לכלול כל מבני נתונים העולה לרוחכם וחייב לכלול התייחסות מלאה לשלבי הקריאה לפונקציה והחזרה מפונקציה. בנוסף התיאור חייב לכלול התייחסות מלאה לגבי הגישה לפרמטרים ומשתנים מקומיים.
2. (4 נקודות) הסבירו אילו תוכניות הנתמכות על ידי המנגנון המקורי לא יתמכו במנגנון החדש. הציעו אנליזה פשוטה ושמרנית לזיהוי סוג זה של תוכניות.
3. (12 נקודות) הרחיבו את המנגנון החדש לתמיכה ב dynamic scoping. הסבירו מהם היתרונות בשימוש במנגנון זה לdynamic scoping מול שימוש במנגנון הקיים. במידה ואינכם רואים יתרונות והתמיכה המוצעת איננה יעילה יותר, שנו את המנגנון מסעיף 1 על מנת לקבל תמיכה יעילה יותר מבמנגנון המקורי.
4. (2 נקודות) תנו דוגמא לשימוש ב dynamic scoping.

שאלה 3 (23 נקודות) אופטימיזציה ו DFA

א) (4 נקודות) תנו דוגמא למצב שבו לא נרצה לבצע את האופטימיזציות שנלמדו בכיתה (או חלקן) והסבירו מדוע.

ב) (6 נקודות) ענו על הסעיפים הבאים -

a. מה היתרון של ביצוע אופטימיזציות על קוד בשפת ביניים לעומת על קוד בשפת מכונה?
b. מה היתרון של ביצוע אופטימיזציות על קוד בשפת מכונה לעומת על קוד בשפת ביניים?
c. מה היתרון של ביצוע אופטימיזציות בזמן ריצה (JIT) לעומת בזמן קומפילציה?

ג) (13 נקודות) נתונה פקודה "סימון" חדשה מהצורה tag(p,k).
עבור פקודה זו, p הוא שם של משתנה ו k הוא גודל קבוע וידוע בזמן קומפילציה.
כמו כן נתונה פקודת untag(m) כך ש- m הוא גודל קבוע וידוע בזמן קומפילציה.
נאמר שמשתנה x עדיין מסומן לפני שורה מסוימת בתוכנית, אם הסימון האחרון של x על כל מסלול לשורה זו לא הוסר. סימון על מסלול כלשהו מוסר במקרים הבאים-

a. פקודת untag(m) מוחקת סימון לכל המשתנים p במסלול כך שפקודת הסימון האחרונה שלהם על המסלול היא tag(p,k) ו- k<m.
b. השמה מחדש למשתנה מוחקת את הסימון שלו.

למשל בקטע הקוד הבא:

```
1.  b:=0
2.  p:=3
3.  tag(p,8)
4.  tag(r,4)
5.  if(b>0) goto 9
6.  tag(p,30)
7.  untag(20)
8.  tag(r,30)
9.  r:=2
10. print(a)
```

לפני שורות 6 ו-7 המשתנים p ו r מסומנים.
לפני שורה 8 המשתנה p מסומן.
לפני שורה 9 המשתנים r ו p מסומנים.
לפני שורה 10 המשתנה p מסומן.

הראו אנליזת DFA המחשבת לכל בלוק בסיסי את קבוצת המשתנים המסומנים לפניו.
הערות:

- ניתן להגדיר שורה בודדת כבלוק בסיסי.
יורדו נקודות על אי <u>הגדרה מלאה</u> של DFA. כלומר, יש לציין מהם פריטי המידע,
האם האנליזה היא must/may, קדמית או אחורית, ושאר פרמטרי DFA שנלמדו בכיתה.

שאלה 4 (22 נק'): ניתוח תחבירי או סמנטי

בכל סעיפי השאלה, טרמינלים מסומנים באות קטנה ובקו תחתון, משתנים באות גדולה.

נתון דקדוק Polish Notation עבור אריתמטיקה. ב-Polish Notation, מופיע קודם האופרטור, ורק אחריו האופרנדים לפי הסדר שלהם בהפעלה:

E -> + E E
E -> - E E
E -> * E E
E -> num

כך ישוערך 3 2 + ל-5, וכן 3 1 1 + * ל-6.

הניחו כי ל-E אין אף תכונה סמנטית, ול-num יש תכונה val המכילה את הערך המספרי של הלקסמה.

א. (9 נקודות) האם הדקדוק הוא LL(1)? האם הוא SLR? נמקו את שתי התשובות.

ב. (8 נקודות) נרצה לבצע את הבדיקה הסמנטית הבאה על ביטוי בשפה: עבור פעולת כפל בה קיים אופרנד שהוא אפס נרצה לתת שגיאה אם האופרנד השני הוא ביטוי מורכב. אם הוא מספר, אין שגיאה.

למשל, הביטוי הבא תקין:

* - 3 + 1 2 1

אבל הביטוי הבא יגרום לשגיאה:

* - 3 + 1 2 + 5 6

האם ניתן לבצע בדיקה זו ללא מעבר נוסף על העץ? אם כן, הסבירו כיצד. אחרת, תארו את המניעה. יש להתייחס בתשובתכם לכל מחלקה אליה הראיתם שהדקדוק שייך בסעיף א', ואם התשובה שונה להסביר מדוע. אין צורך לענות עבור מחלקה אליה הדקדוק לא שייך.

ג. (5 נקודות) (לא תלוי בסעיפים הקודמים) האם ייתכן אוטומט פרפיקסי עבור LR(1) בו מתקיים בדיוק המצב הבא? אם כן, הראו דקדוק בו המצב מופיע. אם לא, הפריכו את הקיום.

A->b●cAA,a
A->b●cAA,c

שאלה 5 (15 נקודות): Backpatching

נתון מבנה הבקרה החדש :

$$S \rightarrow skipTo\ E1\ do\ E2{:}\ L\ end$$
$$L \rightarrow L_1\ \ S\ \mid\ S$$

מבנה הבקרה מקבל מספר פקודות ב – L (statements).
יש לבצע E2 פקודות אחת אחרי השנייה, החל מהפקודה מספר E1 (הספירה מתחילה מ1).
ניתן להניח ש E1 ו E2 הם מסוג rvalue.
יש לצאת מהמבנה במידה ויש שגיאה מהצורה :E1\E2 שליליים או חורגים ממספר הפקודות ב L.

דוגמת הרצה :
עבור הקוד הבא-

```
SkipTo 2 do 2:
print("Backpatching");
print("is");
print("FUN");
end
```

יודפס is FUN בלבד.

א. (5 נק') הציעו פריסת קוד המתאימה לשיטת backpatching עבור מבנה הבקרה הנ"ל. על הקוד הנוצר להיות יעיל ככל האפשר.הסבירו מהן התכונות שאתם משתמשים לכל משתנה.

ב. (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.

ג. אין לשנות את הדקדוק
ד. אין להשתמש בכללים סמנטיים באמצע כלל גזירה
ה. ניתן להשתמש במרקרים N,M שנלמדו בכיתה בלבד.
ו. אין להשתמש במשתנים גלובליים בזמן קומפילציה.
ז. למשתני S ישנן התכונות שהוגדרו בכיתה בלבד.
ח. למשתני S יש כללי גזירה פרט לאלו המוצגים בשאלה.

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 SHIFT
                else ERROR
        else    // X ∈ V
                if M[X , t] = error then ERROR
                else REPLACE(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
```

ניתוח סמנטי

אלגוריתם dfvisit לניתוח סמנטי עבור הגדרות L-attributed:

```
procedure dfvisit(node n) :
        foreach child m of n in left-to-right order do
                evaluate the inherited attributes of m
                dfvisit(m)
        end
        evaluate the synthesized attributes of n
```

ייצור קוד בשיטת Backpatching

פונקציות:

`makelist(quad)`	יוצרת רשימה ריקה עם איבר אחד (ה"חור" quad).
`merge(list1,list2)`	מחזירה רשימה ממוזגת של הרשימות list1, list2
`emit(code string)`	מדפיסה קוד בשפת הביניים ומאפשרת להדפיס פקודות קפיצה עם "חורים".
`nextquad()`	מחזירה את כתובת הרביעיה (הפקודה) הבאה שתצא לפלט.
`backpatch(list, quad)`	מקבלת רשימת "חורים" list וכתובת quad, ו"מטליאה" את הרשימה כך שבכל החורים תופיע הכתובת quad.
`newtemp()`	מחזירה שם של משתנה זמני חדש שאינו נמצא בשימוש בתכנית.

משתנים סטנדרטיים:

- S: גוזר פקודות (statements) בשפה. תכונות:
  - nextlist: רשימת כתובות של פקודות המכילות חור שיש להטליא בכתובת הפקודה הבאה לביצוע אחרי הפקודה הנגזרת מ- S.
- B: גוזר ביטויים בוליאניים. תכונות:
  - truelist: רשימת כתובות של פקודות המכילות חור שיש להטליא בכתובת אליה יש לקפוץ אם הביטוי הבוליאני מתקיים.
  - falselist: רשימת כתובות של פקודות המכילות חור שיש להטליא בכתובת אליה יש לקפוץ אם הביטוי הבוליאני אינו מתקיים.
- E: גוזר ביטויים אריתמטיים. תכונות:
  - E.place: שם המשתנה הזמני לתוכו מחושב הביטוי האריתמטי.

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. פרמטרים וקריאה לפרוצדורות
7. indexed assignments
8. השמה של כתובות ומצביעים

```
x := y op z
x := op y
x := y
goto L
if x relop y goto L
param x
call p, n
return y
x := y [ i ]
x [ i ] := y
x := addr y
x := * y
* x := y
```

Data-Flow Analysis

ההגדרות מתייחסות ל – CFG: G=(V,E).

הצורה הכללית של המשוואות בחישוב סריקה קדמית:

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

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

הצורה הכללית של המשוואות בחישוב סריקה אחורית:

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

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