28.1.2016

מבחן סוף סמסטר – מועד א'

מרצה אחראי: פרופ' ארז פטרנק

מתרגלים: יורי משמן עדי סוסנוביץ עידן שורץ

הוראות:

א. בטופס המבחן 9 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 (14 נק'):סיווג מאורעות

האם המאורעות הבאים קורים בזמן קומפילציה, בזמן ריצה או בזמן בניית הקומפיילר? אם מדובר על זמן קומפילציה כתבו באיזה שלב של הקומפילציה המאורע קורה. נמקו בקצרה.

א. (2 נק') בשפה מסוג dynamicly typed מתגלה שלפונקציה bar נשלח פרמטר מסוג int.

ב. (2 נק') מתגלה שגיאה כי המתכנת שכח לאתחל משתנה.

ג. (2 נק') מתגלה טעות בגלל שהמתכנת שם בטעות כרוכית ("@") בתוך שם משתנה בזמן הגדרת המשתנה.

ד. (2 נק') בשפה מסוג staticly typed מתגלה שגיאה בקריאה לפונקציה "bar(*a)" כי טיפוס הפרמטר שגוי.

ה. (2 נק') מתגלה טעות בגלל שהמתכנת החליף נקודה-פסיק (לאחר i=1) בנקודה בהגדרת לולאה הבאה: "for (i = 1 . i++; i<10) { ... }".

ו. (2 נק') מתגלה שלולאה אינה אינסופית, כלומר היא מסתיימת.

ז. (2 נק') מגלים שבהגדרת שפת התכנות קיימת דו משמעות.

שאלה 2 (23 נק'):

בשאלה זו נתכנן מערכת ניהול זיכרון דינאמי עבור מחשב מרכזי כלשהוא.

א) (3 נק) נניח שתוכנית משתמשת במרחב זיכרון גדול ולא עושה בו הרבה שינויים, על איזה אלגוריתם לניהול זיכרון דינאמי הייתם ממליצים (reference counting, mark and sweep, copying)? נמקו.

ב) (3 נק) עבור האלגוריתם שבחרתם, תארו בעיה מרכזית בשימוש בו ותנו דוגמא למהלך ריצה שיוצר בעיה. (תארו איך נראים האובייקטים המתאימים ב-heap וכיצד הם משתנים במהלך הריצה.)

ג) (2 נק) הציעו פתרון פשוט לבעיה שתיארתם.

ד) (15 נק) בכיתה תיארנו בעיה שיש לאלגוריתמי tracing עם concurent GC, בגלל שהגרף משתנה ע"י התוכנית בעת סריקתו על ידי אוסף הזבל. השאלה הבאה תנסה להבין את הבעייתיות באיסוף מקבילי כאשר משתמשים ב-reference counting .

a. (4 נק) ראשית, גם ב-reference counting יש סריקה. למשל, כאשר חוט מסויים מוחק מחוון אחרון אל מבנה גדול כמו עץ שלא נגיש משום מחוון אחר, האלגוריתם אוסף באופן רקורסיבי את כל הצמתים בעץ שהפך כולו ללא נגיש. האם נוצרת בעיה באיסוף של עץ כזה (ספציפית בדוגמה הזו של העץ) בגלל המקביליות? נמקו.

b. (3 נק) רישמו פסאודו קוד של ה-write barier של reference counting.

c. (4 נק) תארו מקרה שבו שני חוטים מבצעים במקביל שתי פעולות הגורמות לשני החוטים לרצות לעדכן את אותו מונה מחוונים בעת ביצוע ה-write barier . תארו את האובייקטים הרלוונטים ב-heap , מה המחוונים ביניהם, ומה הפעולה שרוצה כל חוט לבצע.

d. (4 נק) האם יש בעיה בכך ששני חוטים רוצים לעדכן את אותו מונה? כלומר, האם צריך להגן על העידכון בקטע קריטי?

שאלה 3 (23 נקודות): שאלת DFA (אנליזה סטטית)

בשאלה זו נרצה לתת סיווג לעוצמה של שגיאה.
לשם כך נגדיר עוצמה של שגיאה כמספר מסלולי החישוב ברוטינה שמגיעים לשגיאה . מסלול חישוב שמגיע לנקודה p בתוכנית הוא מסלול מהתחלת התוכנית עד לנקודה p.

למען הפשטות נתבונן בשפה שבה לתוכניות אין לולאות, וטיפוסי המשתנים יכולים להיות רק int או real. כמו כן יש בשפה רק פסוקי הגדרת טיפוסים של משתנים, הצבה של קבועים במשתנים או משתנים במשתנים אחרים, ופסוקי קפיצה מותנית. הגדרת טיפוס יכולה להופיע בכל מקום בתוכנית אבל לפני כל שימוש במשתנה חייבת להיות הגדרה של אותו משתנה.
אפשר להניח שיש בלוק כניסה יחיד ובלוק יציאה יחיד לרוטינה.

שגיאה תיווצר כאשר יש שימוש במשתנה אבל לפני כן לא היתה הגדרה שלו, או שהשימוש מציב במשתנה ערך שאינו תואם להגדרה שהיתה קודם, או שיש נקודה בתוכנית אליה ניתן להגיע ממסלול המגדיר משתנה כ-int וגם ממסלול אחר המגדיר משתנה כ-real.

א. (3 נק) נניח שלפקודה מסויימת i בבלוק בסיסי B מגיעים 120 מסלולי חישוב. האם יתכן שלפקודה אחרת j באותו בלוק בסיסי יגיע מספר שונה של מסלולי חישוב? אם כן, הסבירו איך זה קורה. אם לא נמקו מדוע זה לא ייתכן.

ב. (8 נק) תארו DFA המחשב כמה מסלולי חישוב מגיעים לתחילת כל בלוק בסיסי.

ג. (3 נק) הסבירו מה קורה ל-DFA שתיארתם בסעיף הקודם אם היו מרשים לולאות בשפת התוכנית. כלומר אם נפעיל את ה-DFA על תוכנית שיש בה לולאות.

ד. (9 נק) הציעו אלגוריתם DFA המקבל CFG עם בלוקים בסיסיים שימצא שגיאות טיפוסים בכל נקודה בתוכנית ויחשב את העוצמה של כל שגיאה כזו.

- עוצמה של הצבת ערך מטיפוס לא מתאים במשתנה היא מספר המסלולים המגיעים להצבה.
- באופן דומה, עוצמה של הצבת ערך למשתנה שהטיפוס שלו לא הוגדר (או שאפילו לא הוגדר בחלק מהמסלולים) היא מספר המסלולים המגיעים להצבה שבהם הטיפוס לא מוגדר.
- עוצמה של התנגשות בהגדרות (כלומר נקודה בתוכנית אליה ניתן להגיע עם שתי הגדרות שונות) היא מספר מסלולי החישוב שיכולים להגיע לנקודה כזו.

יש לתת הסבר קצר לנכונות העוצמות של השגיאות המדווחות. יש לבצע DFA יחיד.

הערות
בשאלה זו אנו מגמישים את ההגדרות של DFA כפי שנלמד בכיתה ובתרגולים, במובן שלא מספיק להגיד שהאנליזה may או must וצריך להגיד במפורש מה הנוסחת עדכון, אך תשתדלו היכן שאפשר להצמד כמה שיותר למבנה ה-DFA כפי שנלמד בתרגולים.
יש לתת פתרון יעיל עד כמה שניתן.

שאלה 4 (25 נק׳): ניתוח תחבירי

1) (10 נק׳) דנה כתבה קומפיילר עבור בדיקת שוויון בין מספרים, כשהיא מעוניינת לטפל בערכים מסוג int ו- float. הגרסה הראשונה שלה הייתה באופן הבא:

		$compare \rightarrow fltcmp \mid intcmp$
[0- 9]+	$\{return\ INT;\}$	$fltcmp \rightarrow fltval\ EQ\ fltval$
[0- 9]+ "."[0- 9]+	$\{return\ FLOAT;\}$	$intcmp \rightarrow intval\ EQ\ intval$
"= = "	$\{return\ EQ;\}$	$fltval \rightarrow FLOAT$
		$intval \rightarrow INT$

דנה השתמשה ב bison ליצירת המנתח, והתקבל מנתח תקין.
כעת דנה מעוניינת לאפשר גם השוואות בין ערכי int ל float (תוך המרת ה int ל float לפני ההשוואה), ומעדכנת את הדקדוק באופן הבא:

$$fltval \rightarrow FLOAT \mid INT$$

א. (5 נק׳) עבור הדקדוק החדש bison מתריע על קונפליקט. איזה קונפליקט ייווצר? האם וכיצד ניתן לפתור אותו באמצעות חוקי קדימויות ואסוציאטיביות ב bison? נמקו. יש להראות את הקונפליקט ע״י בניית המצב המתאים באוטומט המנתח.

ב. (5 נק׳) יוסי מציע לשנות את השפה ע״י הוספת אופרטור =~ עבור השוואות עם float. הסבירו האם הדבר פותר או לא את הקונפליקט מסעיף א׳. נמקו.

2) (5 נק׳) מצאו דוגמה לדקדוק שהוא לא left-factored , אך הוא כן ב- LL(1). הוכיחו תשובתכם.
על הדקדוק המוצע להכיל אך ורק משתנים ישיגים וטרמינליים.
<u>הגדרה</u>:
דקדוק **שאינו** <u>left-factored</u> מכיל שני כללים מהצורה: $X \rightarrow \alpha\beta_1, X \rightarrow \alpha\beta_2$, כאשר $\alpha \in (V \cup T)^+$.
משתנה גזירה X הוא <u>טרמינלי</u> אם קיימת גזירה בדקדוק $X \overset{*}{\Rightarrow} \alpha$ כאשר $\alpha \in T^*$.

3) (10 נק׳) נתונה שפה L באמצעות ביטוי רגולרי: 0*11

א. (5 נק׳) כתבו דקדוק SLR עבור השפה L אשר דורש θ(n) מקום במחסנית המנתח עבור קלט באורך n.
נמקו תשובתכם. (והוכיחו גם כי הדקדוק הוא SLR).

ב. (5 נק׳) כעת ענו על סעיף א׳ עם דרישת θ(1) מקום במחסנית. (כלומר יש להציע דקדוק SLR אחר לשפה L העומד בדרישות החדשות ולנמק תשובתכם, וכן להוכיח כי הדקדוק SLR).

שאלה 5 (15 נקודות): Backpatching

בשאלה זו נדון במבנה בקרה מסוג switch שהוחלט ליישם אותו על ביטויים בוליאניים.
התחביר של המבנה הוא כלהלן:

$$B \rightarrow switch(E)\{CL\}$$
$$\mathrm{CL} \rightarrow \mathrm{C}\ ,\mathrm{CL} \mid\ \mathrm{C}$$
$$\mathrm{C} \rightarrow case\ NUM : B$$

משמעות המבנה: ערך הביטוי הבוליאני שנגזר על ידי ה switch הוא כערך הביטוי הבוליאני המתאים ל case הראשון שעבורו יש התאמה של ערך הבטוי הנגזר על ידי E לערך הקבוע NUM של אותו ה case. אם אין אף case שעבורו יש התאמה כזו, אז ערך הבטוי הבולאיני הוא false.

דוגמת קוד:

```
int v1, v2,v3;
read(v1); read(v2); read(v3);
int var;
read(var);
if(  switch(var) {case 1: v1>10, case 2: v2>20, case 3: v3>30}  )
   print(″ok″);
```

בדוגמה לעיל, אם ערך var מהקלט הוא 1 וגם ערך v1 מהקלט הוא 11 אז הבטוי ישוערך ל true ויודפס ok.
לעומת זאת, אם ערך var מהקלט הוא 4, אז לא יהיה אף case מתאים והבטוי ישוערך ל false.

א. (5 נק׳) הציעו פריסת קוד, המתאימה לשיטת backpatching, עבור מבנה הבקרה הנ״ל. על הקוד הנוצר להיות יעיל ככל האפשר.
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 והן מבחינת המקום בזכרון שנדרש עבור התכונות הסמנטיות.

הערות:

1. אין להשתמש בכללים סמנטיים באמצע כלל גזירה.
2. אין להשתמש במשתנים גלובליים בזמן קומפילציה.
3. המשתנים E,B הם המשתנים הסטנדרטיים המופיעים בדף הנוסחאות, ויש להם כללי גזירה בנוסף לכלל המופיע בשאלה.
4. יש להשתמש רק במרקרים M,N שנלמדו.

בהצלחה!!

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:

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

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