מושגים בשפות תכנות, שנה"ל תשע"ח, סמסטר ב', מועד ב'

פרופ' מולי שגיב, עודד פדון

הנחיות כלליות:

- משך הבחינה 3 שעות.
- מותר להשתמש בכל חומר כתוב (אין להשתמש במחשבון, מחשב, או טלפון נייד).
- בכל השאלות עליכם לתת הסבר משכנע מדוע התשובה שכתבתם נכונה, אלא אם מצויין אחרת.
- במבחן 7 שאלות. שאלה 1 היא חובה, ועליכם לבחור לענות על 5 שאלות מבין שאלות 2-7.
- את כל התשובות יש לכתוב במחברת הבחינה בלבד, תשובות על טופס הבחינה לא תבדקנה.
- חלוקת הניקוד: שאלה 1, 10 נקודות, שאלות 2-7, 18 נקודות כל אחת. סה"כ 100 נקודות.

נא להקיף בעיגול את השאלות שבחרת לענות עליהן:

1 2 3 4 5 6 7

חובה לענות על השאלה הבאה:

שאלה 1 - שאלת חובה (10 נקודות - 2 נקודות לכל סעיף)

בשאלה זו עליכם לסמן נכון / לא נכון בכל אחד מהסעיפים (בשאלה זו אין צורך לנמק).

1. בשפת OCaml תמיד אפשר לשחרר את רשומת ההפעלה מהזיכרון בסוף ביצוע פונקציה.
2. בשפת JavaScript המתכנת משחרר זיכרון בצורה מפורשת.
3. קיים T כך שלביטוי (λx:T. λy:T. x y) יש טיפוס ב simply typed lambda calculus.
4. ב JavaScript יש תמיכה ב High-Order Functions.
5. התוכניות הבאות שקולות תחת Structural Operational Semantics:

- `while true do skip ; x := 1`
- `x := 2 ; while true do skip`

בחרו 5 שאלות מהשאלות הבאות. משקל כל שאלה 18 נקודות.

שאלה 2 - הגדרות אינדוקטיביות והוכחות באינדוקציה

נתון הדקדוק הבא, כאשר E הוא סמל ההתחלה:

```
R ::= R + T | T
T ::= id | ( R ) | ( R
```

א. כתבו את הדקדוק כהגדרה אינדוקטיבית של קבוצות, כפי שראינו בקורס.

ב. עבור כל אחת מהמילים הבאות, קבעו האם היא שייכת לשפה של הדקדוק הנ"ל. במקרה שהמילה בשפה - הראו עץ גזירה, במקרה שהמילה אינה בשפה - הוכיחו זאת.

```
id + (
( id + id
```

ג. הוכיחו שלכל מילה בשפה של הדקדוק הנ"ל מתקיים שמספר הסוגריים השמאליים גדול או שווה למספר הסוגריים הימניים.

שאלה 3 - סמנטיקה

נתונות שתי התוכניות הבאות:

```
P1 = (while b do S)
P2 = (while b do S) ; (while b do S)
```

הוכיחו שהתוכניות שקולות ב Natural Operational Semantics.

שאלה 4 - λ-calculus

א. ציירו את ה AST המתאים לביטוי הבא ב λ-calculus:

```
(λx. λy. x x) ((λx. x) (λx. x)) (λx. y)
```

ב. כתבו סדרת חישוב (reduction) לביטוי מסעיף א' תחת call by value semantics.

ג. כתבו סדרת חישוב (reduction) לביטוי מסעיף א' תחת lazy evaluation semantics.

ד. מצאו טיפוס T כך שקביעת הטיפוס הבא תתקיים, וכתבו עץ גזירה שמוכיח אותה לפי כללי הטיפוס (typing rules):

```
x : bool→bool→bool  ⊢ (λy:T. x (y true)) : T→T
```

שאלה 5 - OCaml

נרצה לבנות ב OCaml אינטרפרטר לשפת While לפי Structural Operational Semantics. לצורך כך, הגדרנו את הטיפוס `aexp` עבור ביטויים אריתמטיים, את הטיפוס `bexp` עבור ביטויים בוליאניים, ואת הטיפוס `statement` עבור פקודות בשפת While. להלן הגדרת הטיפוס `statement`:

```
type statement    = Skip
                  | Assign of string * aexp
                  | Seq of statement * statement
                  | If of bexp * statement * statement
                  | While of bexp * statement
```

הניחו שהגדרנו גם טיפוס `state` עבור מצב של התוכנית, וכן שהגדרנו את הפונקציות הבאות:

```
aeval : aexp -> state -> int
beval : bexp -> state -> bool
setvar: state -> string -> int -> state
```

כאשר הפונקציות beval ו aeval מממשות את הסמנטיקה של ביטויים בוליאניים וביטויים אריתמטיים בהתאמה, והפונקציה `setvar` מעדכנת ערך של משתנה במצב.

כדי לייצג קונפיגורציה של Structural Operational Semantics, הגדרנו את הטיפוס הבא:

```
type config       = Pair of statement * state
                  | Final of state
```

א. כתבו פונקציה:

```
sos: state -> statement -> config
```

שתקבל מצב s ופקודה S, ותחזיר קונפיגורציה γ כך ש

$\langle S,s \rangle \Rightarrow \gamma$

לפי Structural Operational Semantics.

ב. כתבו פונקציה:

```
run_sos: state -> statement -> state
```

שתקבל מצב s ופקודה S, ותחזיר מצב 's כך ש

$\langle S,s \rangle \Rightarrow^* s'$

לפי Structural Operational Semantics. אם אין מצב 's כזה, הפונקציה לא חייבת להסתיים.

הפונקציה `run_sos` צריכה לקרוא לפונקציה `sos` שהגדרתם בסעיף א'.

```
let rec sos env st = match st with
    | Skip -> Final  env
    | Assign (lhs, rhs) -> Final  (setvar env lhs (aeval rhs env))
    | Seq (s1, s2) -> (
            match sos env s1 with
            | Pair (sp, envp) -> Pair (Seq (sp, s2),  envp)
            | Final envp -> Pair (s2, envp)
        )
    | If (b, th, el) -> if (beval b env)
        then Pair (th, env)
        else Pair (el, env)
    | While (b, st) -> Pair (If (b, Seq (st, While (b, st)), Skip), env)

let rec run_sos env st = match (sos env st) with
    | Final envp -> envp
    | Pair (sp, envp) -> run_sos envp sp
```

שאלה 6 - JavaScript

נתון הקוד הבא בשפת JavaScript:

```
1   var arr = [16, 9, 2018, 0];
2   var i;
3   for (i = 0; i < 3; i++) {
4       setTimeout(function() { console.log(arr[i]); }, 1000 + 1000 * i);
5   }
```

תזכורת: פונקציית הספריה setTimeout(func, milliseconds) גורמת לתשתית לקרוא לפונקציה func לאחר זמן של milliseconds (שנתון באלפיות שניה).

א. בהרצת הקוד הנ"ל, אילו ערכים יודפסו אחרי שניה, שתי שניות, ושלוש שניות? הסבירו מדוע.

ב. תקנו את הקוד כך שהלולאה בשורות 3-5 תגרום להדפסה של שלושת הערכים הראשונים במערך, לאחר שניה, שתיים, ושלוש שניות בהתאמה. עליכם לשנות את שורה 4 בלבד. התיקון שלכם צריך לתמוך בכל אורך של מערך, במידה ואתחול המערך ותנאי הסיום של הלולאה ישונו בהתאם.

ג. הסבירו בקצרה מהו closure, וכיצד פועל הקוד המתוקן שכתבתם בסעיף ב'.

שאלה 7 - Types

נתון המימוש הבא של פונקציית OCaml שאמורה לבדוק האם שתי רשימות שוות (כלומר באותו אורך והאיברים שווים בהתאמה):

```
let rec eq xs ys = match xs with
    | [] -> match ys with
        | [] -> true
        | y::ys' -> false
    | x::xs' -> match ys with
        | [] -> false
        | y::ys' -> eq xs' ys'
```

א. נתחו את הטיפוס של הפונקציה eq ע"י אלגוריתם Hindley-Milner. עליכם לפרט את אופן פעולת האלגוריתם עד להגעה לטיפוס הכללי ביותר של הפונקציה.

ב. כיצד ניתן להבין מהטיפוס שיש באג במימוש?

ג. תקנו את הבאג.

ד. נתחו את הטיפוס של הפונקציה המתוקנת. עליכם להסביר בפירוט מה ישתנה בפעולת אלגוריתם Hindley-Milner על הקוד המתוקן לעומת סעיף א'.

ה. כתבו פונקציה <u>אחרת</u> כלשהי בעלת אותו הטיפוס בדיוק כמו הפונקציה המתוקנת.

בהצלחה,

מולי ועודד