מושגים בשפות תכנות, שנה"ל תשע"ו, 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 פלאפון).
- 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 ייבדקו.
- 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 עלייך לסמן נכון / לא נכון בכל אחד מהסעיפים (בשאלה זו אין צורך לנמק).

1. שפת Scala תומכת במבני נתונים אינסופיים.
2. בשפת Go לא מבוצעת בדיקת טיפוסים בזמן קומפילציה.
3. ביצוע של Y-combinator בסמנטיקת call-by-value גורם לחישוב אינסופי.
4. בשפת C יש צורך ב access link כדי לגשת למשתנים לא לוקאליים.
5. התוכניות הבאות שקולות תחת Natural Operational Semantics:

- `while false do skip`
- `while true do skip`

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

שאלה 2 - Parsing

נתבונן בדקדוק הבא עבור דקדוקים חסרי הקשר:

```
BNF → RULE | BNF ";" BNF
RULE → symbol "::=" SYMBOLS
SYMBOLS → symbol | SYMBOLS SYMBOLS
```

כאשר ה terminals הם:

```
symbol, "::=", ";"
```

וה non-terminals הם:

```
BNF, RULE, SYMBOLS
```

וסמל ההתחלה הוא BNF.

א. הראי שהדקדוק אינו חד משמעי ע"י מתן שני עצי גזירה שונים לאותה מילה.

ב. כתבי דקדוק חד משמעי שקול (עבור אותה שפה). ציירי את עץ הגזירה בדקדוק שכתבת עבור המילה שבה השתמשת בסעיף א'.

ג. האם הדקדוק שכתבת בסעיף ב' הוא LL(1)? אם לא, כתבי דקדוק LL(1) שקול.

ד. ציירי עץ גזירה לפי הדקדוק שכתבת בסעיף ג', עבור המילה הבאה:

```
bnf ::= r ;
bnf ::= bnf semi bnf ;
r ::= s arrow slst ;
slst ::= s ;
slst ::= slst slst
```

כאשר אין משמעות מיוחדת לירידת שורה, והמחרוזות הבאות מתורגמות ע"י ה lexer ל symbol:

```
bnf, r, s, slst, arrow, semi
```

שאלה 3 - סמנטיקה

נרצה להוסיף לשפת While את הפקודה הבאה:

while b_1 **repeat-if** b_2 **do** S

זהו מבנה של לולאה שמתנהג כמו לולאת while או כמו לולאת repeat לפי הערך של התנאי b_2 בכניסה ללולאה, כאשר b_1 הוא תנאי הלולאה. הסמנטיקה הרצויה של המבנה החדש היא שהפקודה הנ"ל שקולה לקוד הבא:

if b_2 **then** (**repeat** S **until** $\neg b_1$) **else (while** b_1 **do** S)

כלומר, אם התנאי b_2 מתקיים בכניסה ללולאה, אז נקבל לולאה שרצה לפחות פעם אחת, וממשיכה לרוץ כל עוד b_1 מתקיים, בדומה ללולאת repeat. אם b_2 לא מתקיים בכניסה ללולאה, אז נקבל לולאת while רגילה שרצה כל עוד b_1 מתקיים, בלי הבטחה שהיא תרוץ פעם אחת לפחות.

לדוגמה, עבור התוכנית הבאה (% הוא אופרטור השארית מחלוקה בשלמים):

while (n < 5**) repeat-if (**n % 2 = 0**) do** n := n + 2

אם נריץ את התוכנית ממצב בו n=10 היא תסתיים במצב בו n=12 (התנאי b_2 מתקיים בכניסה, לכן גוף הלולאה יבוצע פעם אחת, ולאחר מכן הלולאה תסתיים כי התנאי b_1 לא מתקיים).

אם נריץ את התוכנית ממצב בו n=11 היא תסתיים במצב בו n=11 (שני התנאים לא מתקיימים בכניסה, ולכן גוף הלולאה לא יבוצע כלל).

א. הרחיבי את ה Natural Operational Semantics כדי לטפל בפקודה החדשה. הכללים אינם יכולים להסתמך על מבנה לולאות while או repeat הרגילות (כלומר לא ניתן להתייחס בכללים למבנה לולאת while או repeat, אלא רק לפקודת הלולאה החדשה). הכללים חייבים להוות הגדרה אינדוקטיבית כפי שלמדנו בקורס, ואסור להם להוסיף משתנים נוספים לשפה.

ב. הדגימי את הסמנטיקה המורחבת שהגדרת בסעיף א' ע"י בניית עץ גזירה לתוכנית הדוגמה מפסקת הפתיחה ממצב התחלתי בו n=10.

ג. הרחיבי את ה Structural Operational Semantics כדי לטפל בפקודה החדשה. הכללים אינם יכולים להסתמך על מבנה לולאות while או repeat הרגילות (כלומר לא ניתן להתייחס בכללים למבנה לולאת while או repeat, אלא רק לפקודת הלולאה החדשה). הכללים חייבים להוות הגדרה אינדוקטיבית כפי שלמדנו בקורס, ואסור להם להוסיף משתנים נוספים לשפה.

ד. הדגימי את הסמנטיקה המורחבת שהגדרת בסעיף ג' ע"י בניית סדרת גזירה לתוכנית הדוגמה מפסקת הפתיחה ממצב התחלתי בו n=10.

שאלה 4 - λ -calculus

נתון הביטוי הבא ב λ -calculus:

```
(λt. λf. f) ((λs. λz. s z) (λx.x)) (λx.x) (λx.x)
```

א. ציירי את ה AST המתאים לביטוי.

ב. כתבי סדרת חישוב (reduction) לביטוי תחת call by value semantics.

ג. כתבי סדרת חישוב (reduction) לביטוי תחת lazy evaluation semantics.

ד. הסבירי בקצרה את ההבדל בין call by value ו lazy evaluation.

שאלה 5 - OCaml

הבהרה: בשאלה זו אסור להשתמש בפונקציות ספרייה מהמודול List

א. כתבי פונקציית OCaml בשם max_seq. הפונקציה max_seq מקבלת רשימה של איברים, ומחזירה את האיבר שחוזר על עצמו ברצף הכי ארוך ברשימה, כאשר רצף ברשימה הוא תת-רשימה שכולה מורכבת מחזרות על אותו איבר. שימי לב שלא מדובר באיבר שמופיע הכי הרבה פעמים סה"כ ברשימה, אלא באיבר שחוזר על עצמו הכי הרבה פעמים <u>ברצף</u>. אם יש כמה איברים כאלה, ניתן להחזיר כל אחד מהם. ניתן להניח שהרשימה לא ריקה. מותר להגדיר פונקציות עזר.

הנה מספר דוגמאות לשימוש ב max_seq:

```
# max_seq [1; 1; 2; 2; 2; 1; 1; 4; 4];;
- : int = 2   (* 2 appears in a sequence of length 3 *)
# max_seq ["a"; "a"; "c"; "c"; "c"; "c"; "a"; "a"; "a"; "b"];;
- : string = "c" (* "c" appears in a sequence of length 4 *)
# max_seq [3; 1; 2; 2; 3; 4; 4; 1; 1; 3];;
- : int = 1 (* can return any of: {1, 2, 4} but not 3 *)
```

ב. מהו הטיפוס הכללי ביותר של הפונקציה שכתבת בסעיף א'? אם כתבת פונקציות עזר, מהו הטיפוס של כל אחת מהן? בסעיף זה אין לפרט את תהליך הסקת הטיפוס, רק לכתוב מהו הטיפוס.

ג. מהי צריכת הזיכרון של הפונקציה max_seq שכתבת בסעיף א' כתלות באורך הרשימה שהיא מקבלת? הסבירי היכן זיכרון זה מוקצה, והאם ניתן להפעיל את הפונקציה על רשימה של מאה מיליון איברים? אם כתבת פונקציות עזר, אל תשכחי לכלול אותן בניתוח.

ד. אם צריכת הזיכרון של הפונקציה שכתבת בסעיף א' אינה קבועה, כתבי גרסה שלה שיכולה לפעול בזיכרון קבוע ללא תלות באורך הרשימה (רמז: רקורסיית זנב). מותר להגדיר פונקציות עזר.

שאלה 6 - JavaScript

נתון הקוד הבא בשפת JavaScript:

```
1   var x = 1;
2   var y = 2;
3   var z = 3;
4   {
5       // swapping x and y using z
6       var z = x;
7       x = y;
8       y = z;
9   }
10  console.log(x);
11  console.log(y);
12  console.log(z);
```

א. בהרצת הקוד הנ"ל, אילו ערכים יודפסו כתוצאה משורות 12-10? הסבירי מדוע.

ב. בשורות 4-9, הכוונה היא להחליף בין הערכים של המשתנים x ו y בלי שום השפעות נוספות. שני את שורות 4 ו 9 בלבד כדי להשיג את הכוונה המקורית. אסור לשנות את שורות 5-8. הסבירי אילו ערכים יודפסו לאחר השינוי, ומדוע השינוי שלך משיג את מטרתו.

שאלה 7 - Types

נתון המימוש הבא של פונקציית OCaml בשם absent שאמורה לבדוק האם איבר אינו מופיע ברשימה, כלומר להחזיר true אם האיבר אינו מופיע ברשימה ו false אם הוא כן מופיע:

```
let rec absent x xs = match xs with
 | [] -> true
 | h::t -> absent x t
```

א. נתחי את הטיפוס של הפונקציה absent ע"י אלגוריתם Hindley-Milner. עלייך לפרט את אופן פעולת האלגוריתם עד להגעה לטיפוס הכללי ביותר של הפונקציה.

ב. כיצד ניתן להבין מהטיפוס שיש באג במימוש?

ג. תקני את הבאג.

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

בהצלחה,

מולי ועודד