מושגים בשפות תכנות
תרגיל 2
להגשה עד 15/04/2019

הנחיות כלליות:

- "הספר" מתייחס ל:
Hanne Riis Nielson, Flemming Nielson: Semantics with Applications: A Formal Introduction
Available online at http://www.cs.kun.nl/~hubbers/courses/sc_1718/materiaal/wiley.pdf
- בתרגיל נעשה שימוש ב Python בגרסת 2.7. כל הפתרונות צריכים לרוץ עם גרסה זו.
- כל השינויים בקבצים צריכים להיות **במקומות המסומנים** בהם. אין לשנות בקבצים דבר מלבד במקומות אלה.
- בכל שאלות המימוש באחריותך לבדוק את הקוד שכתבת על דוגמאות נוספות ולוודא את נכונותו.
- בשאלות על שקילות ב-Structural Operational Semantics, ניתן (ומומלץ) להשתמש בתכונות של sequential composition שהוכחנו בתירגול (למה 2.19 ותרגיל 2.21 בספר).

1.

a. הוסיפו כללים לסמנטיקה של ביטויים אריתמטיים (טבלה 1.1 בספר) עבור טיפול בפעולות בינאריות (bitwise opearations) על מספרים טבעיים:
(x bit-and y)
(x bit-shift-left y)
(x bit-shift-right y)

b. הרחיבי את ה AST של שפת While בקובץ while_ast.py כדי שיכלול ביטויים עבור הפעולות מהסעיף הקודם: BitAnd, BitShiftLeft, BitShiftRight. הרחיבי את הפונקציה eval_arith_expr בקובץ expr.py כדי שתטפל בביטויים החדשים לפי הכללים שכתבת בסעיף a.

c. שני את המימוש של הפונקציה eval_bool_expr בקובץ expr.py ואת הקבצים nos.py, nos_tree.py, sos.py כך שערכי אמת ייוצגו ע"י הקבועים tt, ff (שמוגדרים ב expr.py), ולא ע"י הקבועים הבוליאניים True, False של שפת Python.

d. הריצי את התוכנית הבאה (אלגוריתם הכפל המצרי לחישוב 84*22), גם עם nos_tree.py וגם עם sos.py:

```
a := 84 ; b := 22 ; c := 0 ; while b ≠ 0 do (
    (if (b bit-and 1) ≠ 0 then c := c + a else skip) ;
    a := a bit-shift-left 1 ;
    b := b bit-shift-right 1
)
```

הוסיפי את פקודות הרצה ל main של הקבצים, ושמרי את הפלט ב 1nos.txt ו 1sos.txt (על הקבצים לכלול גם את הפלט של תוכנית prog הנתונה וגם את הפלט של אלגוריתם הכפל המצרי באותו פורמט).

2. בתרגיל מס' 1 הוספנו לשפת While את הפקודה הבאה:

do S **while** b

זוהי לולאה שתמיד מתבצעת פעם אחת לפחות, והביצוע שלה נפסק כאשר התנאי b אינו מתקיים. לדוגמה, הקוד הבא:

do x := x-10 **while** x>10

יסתיים במצב בו x=5 אם יתחיל במצב בו x=55, ויסתיים במצב בו x=-3 אם יתחיל במצב בו x=7.

a. הוסיפו ל while_ast.py את המבנה DoWhile לייצוג לולאת do-while, והוסיפי ל nos.py ול nos_tree.py מימוש לפי הכלל(ים) שהגדרתם בשאלה 5 של תרגיל מס' 1. בדקו את המימוש.
b. הוסיפו כלל(ים) לטבלה 2.2 בספר שיגדירו את ה Structural Operational Semantics של פקודת do while. הכלל(ים) אינם יכולים להסתמך על מבנה לולאת while בשפה (כלומר לא ניתן להתייחס בכללים למבנה לולאת while, רק ללולאת do while).
c. הוסיפו ל sos.py מימוש לפי הכלל(ים) שהגדרת בסעיף b. בדקו את המימוש.
d. הוכיחו את השקילות הבאה ב-Structural Operational Semantics:

do S **while** b ~ S **; if** b (**do** S **while** b) **else** skip

על פי הכללים שהגדרתם בסעיף ב.

3. הוכיחו שה-Structural Operational Semantics של שפת While הוא דטרמיניסטי. כלומר
(1) אם $\langle S, s\rangle \Rightarrow^* s_1$ וגם $\langle S, s\rangle \Rightarrow^* s_2$ אז $s_1=s_2$, וגם
(2) לא ייתכן שתוכנית S גם תעצור וגם תגיע ללולאה אינסופית החל ממצב מסוים s (כלומר לא ייתכן של-$\langle S, s\rangle$ קיימת גזירה סופית וגם קיימת גזירה אינסופית).

4. **א.** הוכיחו את השקילות הבאה ב-Structural Operational Semantics:

(S1;S2);S3 ~ S1;(S2;S3)

ב. הוכיחו שבמקרה הכללי **לא** מתקיימת השקילות ב-Structural Operational Semantics

S1;S2 ~ S2;S1

כלומר קיימים S1,S2 כך שהשקילות לא מתקיימת. (יש להוכיח שהשקילות לא מתקיימת.)

5. הראו (ע"י דוגמה נגדית) שלא בהכרח מתקיים שאם:

$\langle S_1;S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$

אז:

$\langle S_1, s\rangle \Rightarrow^k s'$

רמז: בחנו היטב את הביצוע של התוכנית משאלה 1 ע"י sos.py ובנו דוגמה נגדית בהשראתו.

6. **בונוס**

הוכיחו (עבור הכללים שהגדרתם בשאלה 2 סעיף d) את השקילות הסמנטית הבאה ב Structural Operational Semantics:

do S **while** b ~ S **; while** b **do** S

בהצלחה!