תורת הקומפילציה – ת"ב 4

מתרגל אחראי – מתן עין-אבי matan.mam@cs.technion.ac.il

שאלה 1 - *Backpatching*

בשאלה הזאת נדון במבנה בקרה חדש שתחבירו נתון על ידי:

S -> super-loop B @ L1 @ L2 S1

L -> L1 B | B

משמעות המבנה: אם ל-*B* ניתן ערך *true*, אז רק אם כל רשימת ה-*B* הראשונה מקבלת ערכי *true* יש לבצע את *S1* ולחזור לתחילת הלולאה. אם ל-*B* ניתן ערך *false* אז רק אם כל רשימת ה-*B* השניה מקבלת ערכי *true* יש לבצע את *S1* ולחזור לתחילת הלולאה (בתחילת הלולאה יש להעריך מחדש את כל התנאים הבוליאניים, כולל *B*), בכל מקרה יש לצאת החוצה מהמבנה.

א' (8 נק') הציעו פריסת קוד שמתאימה לשיטת *backpatching* עבור מבנה הבקרה המתואר, על הקוד הנוצר להיות יעיל ככל האפשר.

ב' (8 נק') הציעו פריסת קוד מתאימה בהנתן השינוי הבא: במקום לחשב מחדש את כל התנאים הבוליאניים כשחוזרים לתחילת הלולאה, מחשבים מחדש את כולם למעט *B*. כלומר, ערכו הראשונה של *B* נשאר קבוע מחשבים מחדש אך ורק תנאים בוליאניים שנגזרים מהרשימה שצריך לחשב בהתאם לערכו הראשוני של *B*.

ג' (9 נק') כתבו סכימת תרגום בשיטת *backpatching* המיצרת את פריסת הקוד שהצעתם בסעיף א'. על הסכימה להיות יעילה ככל האפשר, הן מבחינת זמן הריצה והן מבחינת המקום בזכרון שנדרש עבור התכונות הסמנטיות.

שאלה 2 – *Parsing*

נתון הדקדוק חסר ההקשר הבא:

$$S \rightarrow E$$
$$E \rightarrow L \mid L[\ num\ ]$$
$$L \rightarrow [\ EL\ ]$$
$$EL \rightarrow \varepsilon \mid num\ , EL$$

כאשר *S,E,L,EL* הם נונטרמינלים, ו- *num, [,],* ו-, (פסיק) הם טרמינלים.

א. (6 נק') האם הדקדוק הוא *LR(0)*? אם לא, הציגו את הקונפליקט, אם כן הציגו את טבלת הניתוח.
ב. (6 נק') האם הדקדוק הוא *LR(1)*? אם לא, הציגו את הקונפליקט, אם כן, הציגו את טבלת הניתוח.
ג. (6 נק') נתחו את המילה *[][num]*. הציגו את כל מצבי המחסנית במהלך הניתוח.
ד. (7 נק') נתחו את המילה *[num,num][num]*. הציגו את כל מצבי המחסנית במהלך הניתוח.

שאלה 3 – *DFA*

שפת התכנות *Chicken* מכילה משתנים מטיפוס *str* המייצג מחרוזת. אין חשיבות לאופן בו המחרוזת מיוצגת (למשל, אין צורך להתייחס לפרטי מימוש כמו *terminating null*, רק לתווים בתוך המחרוזת). על המחרוזות בשפת *Chicken* ניתן לבצע את הפעולות הבאות:

`'str'`	הגדרת קבוע מחרוזת
`x + y`	שרשור שתי המחרוזות *x* ו-*y*
`x * n`	שרשור של המחרוזת *x* עם עצמה *n* פעמים
`x.replace(y,z)`	יחזיר מחרוזת שהיא העתק של *x* בה כל מופע של רצף התווים *y* יוחלף ברצף התווים *z*
`x.upper()`	יחזיר מחרוזת שהיא העתק של *x* בה כל תו בין *a* ל-*z* יוחלף בתו הגדול המתאים (בין *A* ל-*Z*)
`x.lower()`	יחזיר מחרוזת שהיא העתק של *x* בה כל תו בין *A* ל-*Z* יוחלף בתו הקטן המתאים (בין *a* ל-*z*)
`x.find(y)`	יחזיר את המיקום הראשון של *y* בתוך *x*. אם *y* אינו תת-מחרוזת של *x*, יחזיר *-1*
`x = y`	השמה של ביטוי מחרוזת לתוך משתנה מחרוזת *x*

וכמובן אתחול עם מחרוזת קבועה והשמה של משתנה אחד מטיפוס *str* למשתנה אחר מטיפוס *str*.
(למידע נוסף על הפעולות של *Chicken* ניתן להסתכל על תיעוד הפעולות בשם זהה בפייתון.)

נרצה לבצע אנליזה לשתי הפונקציות הבאות:

```
1: def foo(str x, int n):
2:    str y = ''
3:    if n != 0:
4:          y = ('s' + x) * 2
5:    else:
6:          y = (x + 's') * 3
7:    assert(y.find('s') != -1)
8:
9: def bar(str x, int n):
10:   str y = ''
11:   while (n > 0):
12:         n = n-1
11:         if n != 0:
12:               y = x.replace('s','t')
13:         else:
14:               y = x.replace('s','w')
15:   assert(y.find('s') == -1)
```

סת' מציע להראות כי ה-*assert* יתקיים לכל ריצה על ידי אנליזה סטטית, ומציע שימוש בדומיין שיראה עבור כל מחרוזת את כל התווים ש**עשויים** להיכלל במחרוזת.

1. (5 נק') הגדירו עבור הדומיין שסת' מציע את הסריג: מהם האיברים ומהו יחס הסדר בינהם ($\sqsubseteq$). בנוסף, הגדירו את פעולת *join* עבור הדומיין ($\sqcup$).
2. (9 נק') הגדירו את הסמנטיקה האבסטרקטית של השפה: עבור כל אחת מהפעולות על מחרוזות הגדירו כיצד יראה המעבר שלה.
   רמז: כדי שתצליחו לבצע אנליזה של התכנית לעיל בסעיף הבא, כדאי לפצל את הסמנטיקות של * ושל `replace` לשלושה מקרים כל אחת.
   הנחה מקלה: השימוש ב-*find* יהיה תמיד בבדיקה *-1 ==* או *-1 !=*
3. (10 נק') הדגימו את ריצת האנליזה על הפונקציה *foo* לעיל: ציירו את ה-*CFG*, והשתמשו בסריג המכפלה כדי להריץ את האנליזה שלכם על הגרף. ניתן להתעלם מבדיקות תנאים בוליאניים בבקרת הריצה של התכנית.
4. (1 נק') האם בעזרת הרעיון של סת' ניתן להוכיח את ה-*assert* בשורה 7? מדוע?

מאט מציע דומיין חלופי לדומיין של סת' שישמור עבור כל מחרוזת את התווים ש**בוודאות** נכללים בה.

5. (5 נק') הגדירו מחדש את יחס הסדר ופעולת ה-*join* עבור הדומיין שהציע מאט.
6. (5 נק') כיצד משתנה הסמנטיקה האבסטרקטית של השפה?

רמז: האם שלושת המקרים שפיצלתם אליהם בסעיף 2 עדיין מתנהגים אותו הדבר?

7. (7 נק') הריצו את האנליזה של מאט על הפונקציה *foo*. האם ניתן להוכיח את ה-*assert* בשורה 7?
8. (8 נק') איזו מבין האנליזות כדאי להריץ כדי להוכיח את *bar*? נמקו את תשובתכם.