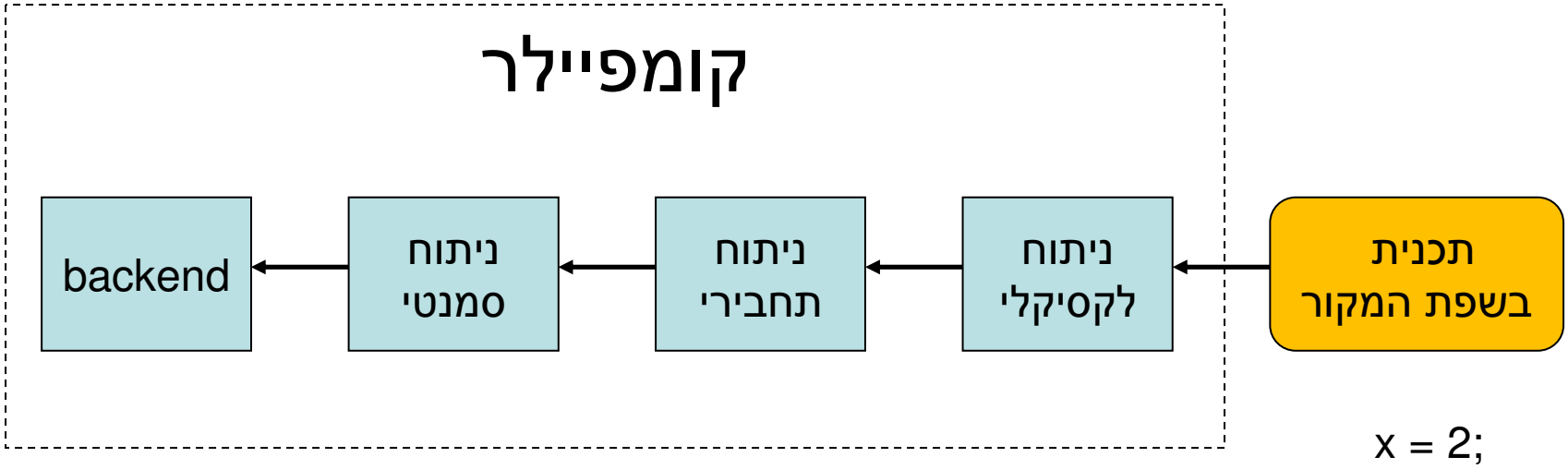
ניתוח לקסיקלי וכלי Lex

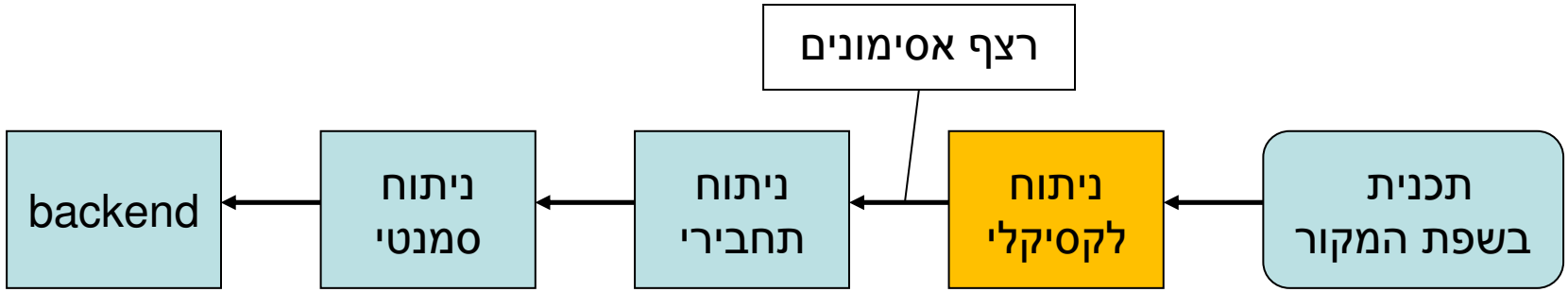
נכתב ע"י אלכס קוגן (sakogan@cs)

סמסטר חורף, תשס"ח

מבנה הקומפיילר



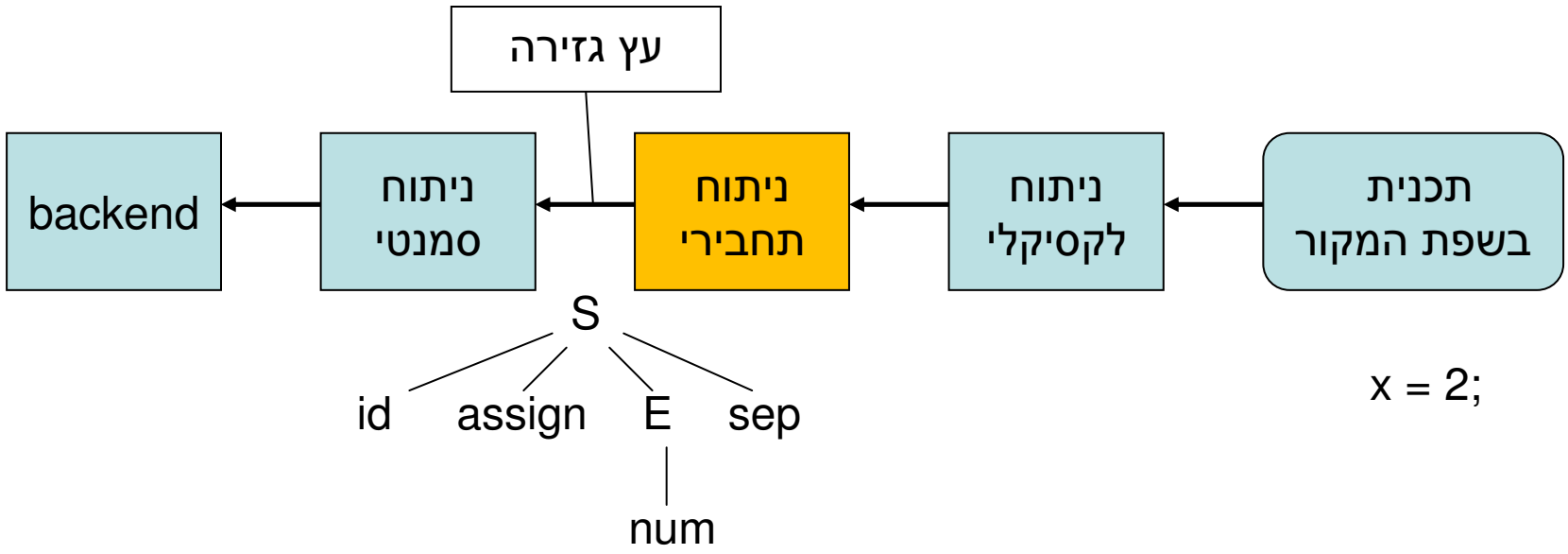
מבנה הקומפיילר



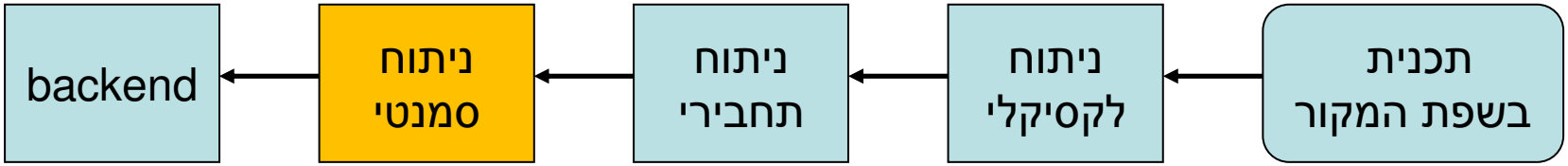
id assign num sep

x = 2;

מבנה הקומפיילר



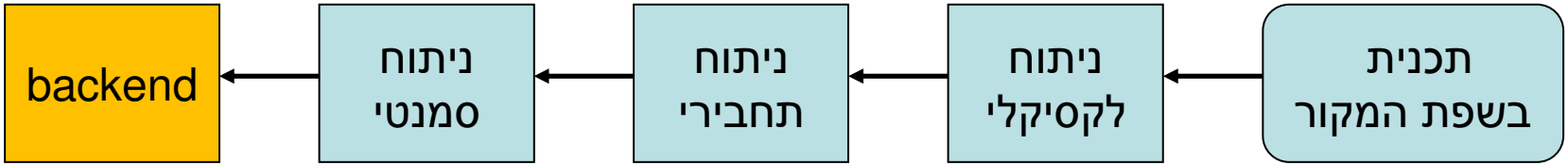
מבנה הקומפיילר



x = 2;

האם קיים משתנה ששמו x?
האם x מטיפוס int?

מבנה הקומפיילר



x = 2;

ייצור קוד בשפת היעד

הגדרות

- **אסימון (token)**: יחידה בסיסית המשמשת כטרמינל בדקדוק שגוזר את שפת התכנות

- **לקסמה (lexeme)**: מחרוזת בקלט שהמנתח הלקסיקלי התאים לאסימון כלשהו

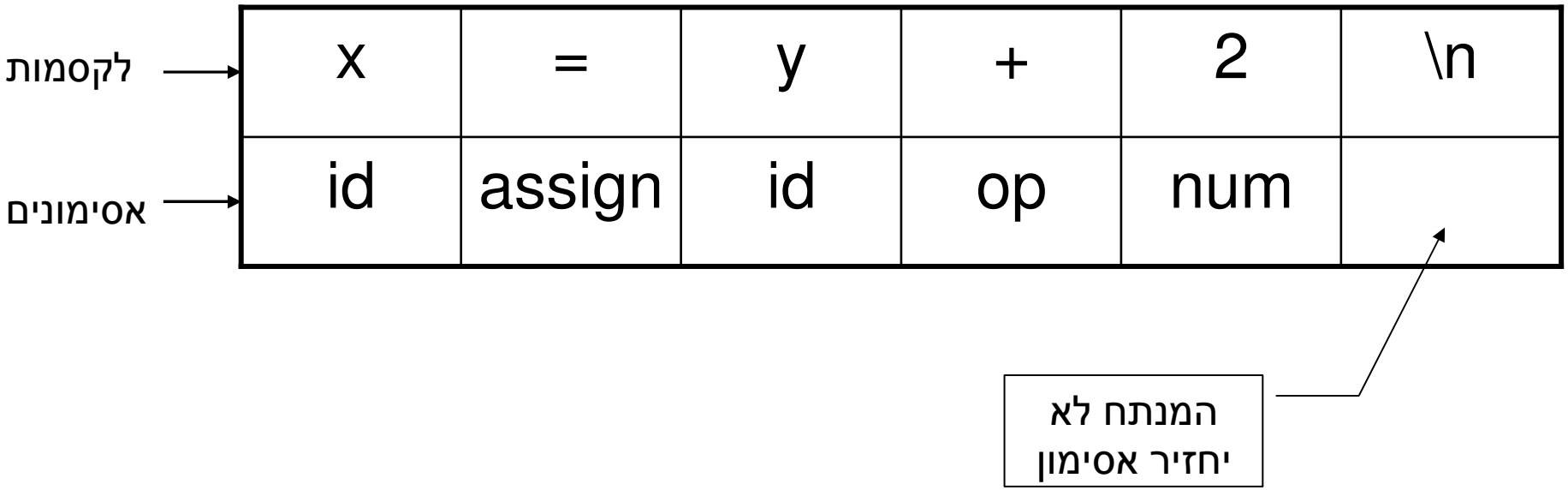
- **תבנית (pattern, regexp)**: ביטוי רגולרי שמגדיר את ההתאמה בין אוסף הלקסמות לאסימון מסוים
  - דוגמה: ניתן להגדיר את האסימון num ע"י התבנית
    (0+1+…+9)+

תפקיד המנתח הלקסיקלי

- מחלק את קוד המקור לאסימונים ("מילים")
- מסנן חלקים שאינם דרושים להמשך הניתוח
  - למשל, רווחים, ירידות שורה
  - מה עם comments?
- מזהה שגיאות לקסיקליות - מחרוזות שאינן להיות אף אסימון
  - למשל, "@" בשפת C

המנתח הלקסיקלי - דוגמה

- ניתוח לקסיקלי עבור "x = y + 2":



המנתח הלקסיקלי ותכונות סמנטיות

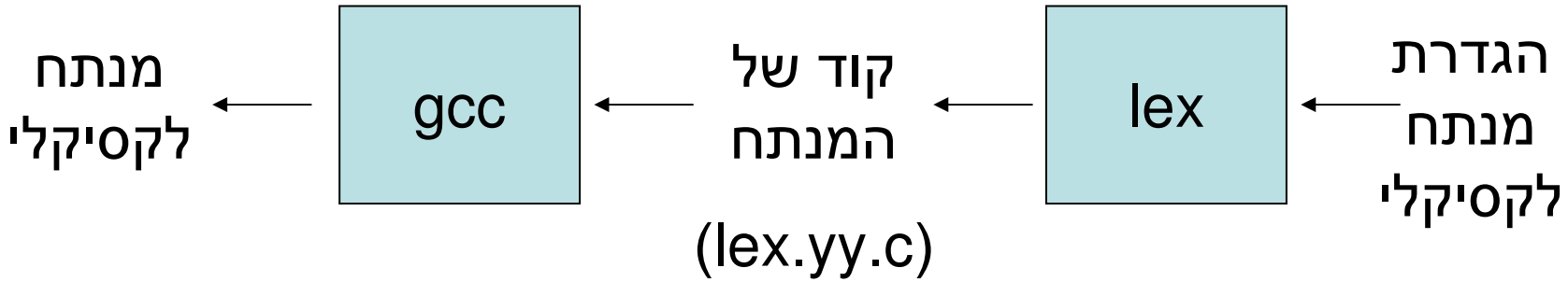
- הלקסמות אינן עוברות הלאה לשלבים הבאים בניתוח
  - הנמתח התחבירי אינו צריך אותן
  - המנתח הסמנטי כן צריך מידע עליהן:
    - האם קיים משתנה ששמו x?
    - איזו פעולה בדיוק הופעלה?

- המנתח הלקסיקלי מחשב לכל אסימון תכונות סמנטיות:
  - id {name = "x"}
  - op {type = "+"}

- התכונות הסמנטיות ידונו בהרחבה בהמשך

כלי Lex

- כלי לייצור מנתחים לקסיקליים
- צורת הפעלה:



- בקורס נעבוד עם כלי תואם Flex

בניית מנתח לקסיקלי בעזרת Lex

- עריכת קובץ הגדרת המנתח (source.lex)
- flex source.lex
- gcc –ll lex.yy.c

מבנה קובץ הגדרות המנתח

- קובץ הגדרות מורכב משלושה חלקים המופרדים בעזרת שורות שמכילות "%%" בלבד:

```
Definition section
%%
Rules section
%%
C code section
```

דוגמה למבנה קובץ הגדרות המנתח

```
1  %{
2  /* Declarations section */
3  #include <stdio.h>
4  void showToken(char *);
5  %}
6
7  %option yylineno
8  %option noyywrap
9  digit            ([0-9])
10 letter           ([a-zA-Z])
11 whitespace       ([\t\n ])
12
13 %%
14 {digit}+                      showToken("number");
15 {letter}+@{letter}+\.com       showToken("email address");
16 {whitespace}                  ;
17 .        printf("Lex doesn't know what that is!\n");
18 %%
19
20 void showToken(char * name) {
21         printf("Lex found token %s , name);
22         printf("the lexeme is %s , yytext);
23         printf("its length is %d\n", yyleng);
24 }
```

דוגמה למבנה קובץ הגדרות המנתח

Definitions section

```
%{
/* Declarations section */
#include <stdio.h>
void showToken(char *);
%}

%option yylineno
%option noyywrap
digit            ([0-9])
letter           ([a-zA-Z])
whitespace       ([\t\n ])

%%
{digit}+                        showToken("number");
{letter}+@{letter}+\.com        showToken("email address");
{whitespace}                    ;
.       printf("Lex doesn't know what that is!\n");
%%

void showToken(char * name) {
        printf("Lex found token %s , name);
        printf("the lexeme is %s , yytext);
        printf("its length is %d\n", yyleng);
}
```

דוגמה למבנה קובץ הגדרות המנתח

הגדרות של שפת C - קוד מועתק כפי שכתוב לקובץ ש-Lex מייצר

אופציות השולטות על צורת העבודה של Flex

הגדרת מקרואים בעזרת ביטויים רגולריים - לשימוש בחלק הבא

```
1  %{
2  /* Declarations section */
3  #include <stdio.h>
4  void showToken(char *);
5  %}
6
7  %option yylineno
8  %option noyywrap
9  digit            ([0-9])
10 letter           ([a-zA-Z])
11 whitespace       ([\t\n ])
12
13 %%
14 {digit}+                      showToken("number");
15 {letter}+@{letter}+\.com       showToken("email address");
16 {whitespace}                  ;
17 .        printf("Lex doesn't know what that is!\n");
18 %%
19
20 void showToken(char * name) {
21         printf("Lex found token %s , name);
22         printf("the lexeme is %s , yytext);
23         printf("its length is %d\n", yyleng);
24 }
```

ביטויים רגולריים של Lex

ביטוי רגולרי של Lex	ביטוי רגולרי "רגיל"	משמעות
a	a	התו a
.	Σ \ {\n}	כל תו פרט לירידת שורה
[xyz] [a-z]	x+y+z a+b+c+…+z	אחד מהתווים שבתוך הסוגריים
r* r+	r* r+	מספר כלשהו של r-ים כולל אפס / לא כולל אפס

דוגמאות נוספות ניתן למצוא באתר הקורס

דוגמה למבנה קובץ הגדרות המנתח

```
%{
/* Declarations section */
#include <stdio.h>
void showToken(char *);
%}

%option yylineno
%option noyywrap
digit               ([0-9])
letter              ([a-zA-Z])
whitespace          ([\t\n ])

%%
{digit}+                            showToken("number");
{letter}+@{letter}+\.com            showToken("email address");
{whitespace}                        ;
.       printf("Lex doesn't know what that is!\n");
%%

void showToken(char * name) {
        printf("Lex found token %s , name);
        printf("the lexeme is %s , yytext);
        printf("its length is %d\n", yyleng);
}
```

Rules section

הגדרות של
אסימונים ופעולות
ש-Lex צריך לבצע
בעת זיהוי שלהם

דוגמה למבנה קובץ הגדרות המנתח

```
1  %{
2  /* Declarations section */
3  #include <stdio.h>
4  void showToken(char *);
5  %}
6
7  %option yylineno
8  %option noyywrap
9  digit                ([0-9])
10 letter               ([a-zA-Z])
11 whitespace           ([\t\n ])
12
13 %%
14 {digit}+                          showToken("number");
15 {letter}+@{letter}+\.com          showToken("email address");
16 {whitespace}                      ;
17 .          printf("Lex doesn't know what that is!\n");
18 %%
19
20 void showToken(char * name) {
21         printf("Lex found token %s , name);
22         printf("the lexeme is %s , yytext);
23         printf("its length is %d\n", yyleng);
24 }
```

משתנים גלובליים של Lex

C code section

הגדרות פונקציות שהוכרזו בחלק ראשון

משתנים גלובליים של Lex

שם	טיפוס	משמעות
yytext	char *	הטקסט של הלקסמה האחרונה שזוהתה
yyleng	int	אורך הלקסמה האחרונה שזוהתה
yylineno	int	השורה הנוכחית בקלט
yylval	user defined	משמש לתקשורת עם המנתח התחבירי (פרטים בהמשך...)

דוגמה למבנה קובץ הגדרות המנתח

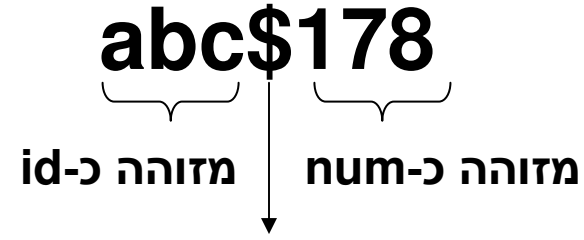
```
1  %{
2  /* Declarations section */
3  #include <stdio.h>
4  void showToken(char *);
5  %}
6
7  %option yylineno
8  %option noyywrap
9  digit            ([0-9])
10 letter           ([a-zA-Z])
11 whitespace       ([\t\n ])
12
13 %%
14 {digit}+                      showToken("number");
15 {letter}+@{letter}+\.com       showToken("email address");
16 {whitespace}                  ;
17 .        printf("Lex doesn't know what that is!\n");
18 %%
19
20 void showToken(char * name) {
21         printf("Lex found token %s , name);
22         printf("the lexeme is %s , yytext);
23         printf("its length is %d\n", yyleng);
24 }
```

תכונות המנתח הלקסיקלי ש-Lex בונה

- קורא קלט מ-stdin וכותב פלט ל-stdout
- רוב העבודה נעשית בפונקציה yylex שתפקידה:
  - לקרוא את הקלט
  - לזהות אסימונים
  - לבצע את פעולה המתאימה לאסימון
- yylex חוזרת רק כאשר:
  - המשתמש כתב return בפעולה של אסימון
  - מגיעה לסוף קלט

טיפול בשגיאות ניתוח

- שגיאת ניתוח נוצרת כאשר בנקודה מסוימת yylex לא מצליחה לזהות אף אסימון
- במקרה כזה, היא מעתיקה את התו הנוכחי לפלט ומנסה שוב מהתו הבא
- דוגמה:



טיפול בשגיאות ניתוח (המשך)

- כדי לתת הודעת שגיאה יותר מפורטת, ניתן להשתמש באסימון הנקודה (.) בתור אסימון אחרון – כפי שנעשה בדוגמה:

```
13  %%
14  {digit}+                        showToken("number");
15  {letter}+@{letter}+\.com        showToken("email address");
16  {whitespace}                    ;
17  .       printf("Lex doesn't know what that is!\n");
18  %%
```

פתרון קונפליקטים

- קונפליקט נוצר כאשר אותו קלט יכול להתאים למספר אסימונים
- דוגמה: נניח שמוגדרים האסימונים הבאים:

for “for”

id [a-z]+

מחרוזות שיגרמו לקונפליקט	ניתוחים אפשריים
abc	
ford	
for	

פתרון קונפליקטים

- קונפליקט נוצר כאשר אותו קלט יכול להתאים למספר אסימונים
- דוגמה: נניח שמוגדרים האסימונים הבאים:

for “for”

id [a-z]+

מחרוזות שיגרמו לקונפליקט	ניתוחים אפשריים
abc	a, ab, abc
ford	for, ford, f, fo
for	for (id) / for (for)

הכללים לפתרון הקונפליקט

1. המנתח חמדן – תמיד מעדיף את הלקסמה הארוכה ביותר שניתן לבחור
2. אם כלל (1) לא פתר את הקונפליקט, בוחרים באסימון בעל עדיפות גבוהה יותר
   - האסימון שמופיע ראשון

פתרון הדוגמה

- נניח שמוגדרים האסימונים הבאים:

for “for”

id [a-z]+

לפי כלל	הניתוח שייבחר	ניתוחים אפשריים	מחרוזות שיגרמו לקונפליקט
1	abc	a, ab, abc	abc
1	ford	for, ford, f, fo	ford
2	for (for)	for (id) / for (for)	for

מקורות נוספים

- דוגמאות לשימוש ב-Lex ניתן למצוא באתר הקורס
- שימוש ב-manual מותקן על Lex וביטויים רגולריים:

  man lex –

  man –s 5 regexp –
- אתר הבית של כלי Flex:
  http://flex.sourceforge.net
- מדריך לשימוש ב-Flex:
  http://www.gnu.org/software/flex/manual/html_node/flex_toc.html