שיטות הידור 046266

פרויקט – חלק ב'

מגישים: עידן גולן 204626444
דולב שמע 204559744

מבוא

בחלק זה של הפרויקט הוספנו parser מבוסס bison אל הlexer (מבוסס flex) שיצרנו בחלק הקודם.

קונפליקטים

את חוקי הדקדוק הנתון הזנו בקובץ parser.ypp. בדקדוק הנתון קיימים הקונפליקטים הבאים

(מסוג shift/reduce) :

1. Dangling else:

Cntrl → IF bexp THEN stmt · ELSE stmt
| IF bexp THEN stmt ·

עבור הקלט IF bexp THEN stmt · ELSE יש לבחור האם לבצע shift אל ELSE שנמצא בlookahead, או reduce. לפי חוקי הקדימות ב Cmm נדרש לבצע shift, זוהי גם ההתנהגות הדיפולטיבית של bison. על מנת להימנע מwarning בעת הקומפילציה של bison פתרנו את הקונפליקט בצורה מפורשת ע"י מתן קדימות לELSE על פני THEN באופן הבא: (עדיפות של חוק נקבעת לפי העדיפות שניתנת לטרמינל האחרון בכלל)

%precedence THEN
%precedence ELSE

2. עבור הכללים הבאים:

exp → exp ADDOP exp
| '(' type ')' exp

עבור הקלט '(' type ')' exp · ADDOP יש לבחור האם לבצע shift/reduce. לפי חוקי הקדימות של האופרטורים יש לבצע תחילה reduce, לכן נגדיר את '(' בעדיפות גבוהה יותר מ ADDOP:

%left ADDOP
%precedence ')'

3. קונפליקט ופתרון דומים ל(2) עבור:

exp → exp MULOP exp
| '(' type ')' exp

אסוציאטיביות

על מנת להתאים את האסוציאטיביות לדרוש בCmm (בדומה לCpp) הגדרנו את הtokenים בעלי אסוצ' ימנית ע"י %right, לדוג' %right NOT, ואת הtokenים בעלי אסוצ' שמאלית ע"י %left. כמו כן העדיפות בין הtokenים השונים נקבעה לפי סדר הופעתם כאשר התחתונים בעדיפות (לדוג' ביצוע MULOP לפני ADDOP).

מבני נתונים בשימוש

על מנת להדפיס את הparsing tree, השתמשנו במצביע לparserNode (הstruct הנתון בpart2_helpers). לכן, הגדרנו מצביע לparserNode כYYSTYPE, כך נקבל את טיפוס זה בlexer בגישה לyylval, ובparser בגישה לtoken/symbol.

אנו בונים את העץ ומעדכנים את הnodes שלו תוך שימוש בכללים הסמנטיים של כל חוק גזירה. בביצוע פעולת reduce לחוק גזירה אנו מייצרים parserNode חדש, מזינים את הערכים הרלוונטיים בstruct ומחברים אליו את הבנים שלו (אשר מחוברים ביניהם ברשימה מקושרת).

הוספנו פעולות עזר ליצירת node עבור symbol ועבור token, כאשר אלה משתמשות ב-makeNode (תיעודן זמין ב-part2_helper.h).