Winter 2007-2008
Compiler Construction
T11 – Recap

Mooly Sagiv and Roman Manevich
School of Computer Science
Tel-Aviv University

Exam – 30/4/2008

- Materials taught in class and recitations
- Example exams on web-site
  - Last year's exams in last year's course
- Popular types of questions
  - Introducing new features to language (IC)
    - Most reasonable Java features good candidates:
      - Access control,
      - Exceptions,
      - Static fields
      - Object oriented-related features
  - Parsing related questions
    - Building LR(0) parser,
    - Resolving conflicts,
    - Running parser on input
  - Activation records – concept level

Example program

```
// An example program
class Hello {
  boolean state;
  static void main(string[] args) {
    Hello h = new Hello();
    boolean s = h.rise();
    Library.printb(s);
    h.setState(false);
  }
  boolean rise() {
    boolean oldState = state;
    state = true;
    return oldState;
  }
  void setState(boolean newState) {
    state = newState;
  }
}
```


Scanning

```
// An example program
class Hello {
  boolean state;
  static void main(string[] args) {
    Hello h = new Hello();
    boolean s = h.rise();
    Library.printb(s);
    h.setState(false);
  }
  boolean rise() {
    boolean oldState = state;
    state = true;
    return oldState;
  }
  void setState(boolean newState) {
    state = newState;
  }
}
```

Issues in lexical analysis:

- Pattern matching conventions (longest match, priorities)
- Running scanner automaton
- Language changes:
  - New keywords,
  - New operators,
  - New meta-language features, e.g., annotations

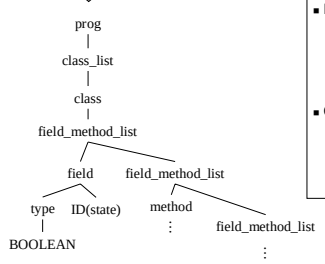
scanner read text and generate token stream

```
CLASS,CLASS_ID(Hello),LB,BOOLEAN,ID(state),SEMI …
```


Parsing and AST

```
CLASS,CLASS_ID(Hello),LB,BOOLEAN,ID(state),SEMI …
```

parser uses stream of token and generate derivation tree



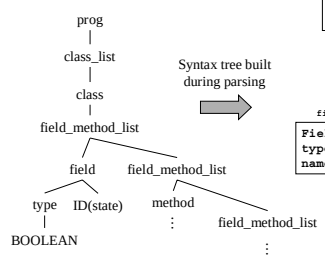
Issues in syntax analysis:

- Grammars: LL(k), LR(k)
- Building LR(0) parsers
  - Transition diagram
  - Parse table
  - Running automaton
- Conflict resolution
  - Factoring
  - In parse table
  - *Read TA1*

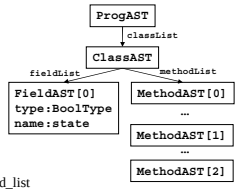
Parsing and AST

```
CLASS,CLASS_ID(Hello),LB,BOOLEAN,ID(state),SEMI …
```

parser uses stream of token and generate derivation tree



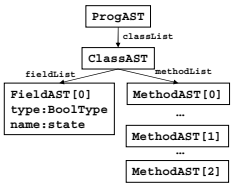
Syntax tree built during parsing



- Should know difference between derivation tree and AST
- Know how to build AST from input

Semantic analysis

- Representing scopes
- Type-checking
- Semantic checks
- Annotating AST



(Program)

Symbol	Kind	Type
Hello	class	Hello

(Hello)

Symbol	Kind	Type	Properties
state	field	boolean	instance
main	method	string[]->void	static
rise	method	void->boolean	instance
setState	method	boolean->void	instance

(setState)

Symbol	Kind	Type
newState	param	int

...

Semantic analysis

- The roles of scopes
  - Provide unique symbol for each identifier – use symbols in next phases instead of identifiers
  - Disambiguate identifiers
  - Determine scope rules: undeclared ids, double-declaration, declaration in wrong scope…
- Type-checking
  - Associate types with identifiers
  - Infer types for expressions
  - Check well-formed statements/classes etc.
  - *Get to know type rule notations*

Semantic conditions

- What is checked in compile-time and what is checked in runtime?

Event	C/R
Program execution halts	
All execution paths in function contain a return statement	
Array index within bound	
In Java the cast statement `(A) f` is legal	
In Java, method o.m(…) is illegal since m is private	

Semantic conditions

- What is checked in compile-time and what is checked in runtime?

Event	C/R
Program execution halts	**R** (undecidable in general)
All execution paths in function contain a return statement	**C** (number of simple paths from function entry to exit is finite)
Array index within bound	**R** (undecidable in general)
In Java the cast statement `(A) f` is legal	Depends: if `A` is sub-type of `f`'s type then checked during runtime (raising exception), otherwise flagged as an error during compilation
In Java, method o.m(…) is illegal since m is private	**C**

More semantic conditions

```
class A {…}
class B extends A {
  void foo() {
    B[] bArray = new B[10];
    A[] aArray = bArray;
    A x = new A();
    if (…) x = new B();
    aArray[5]=x;
  }
}
```

(a) Explain why the assignment aArray=bArray is considered well-typed in Java.
(b) Under what conditions should/could the assignment aArray[5]=x lead to a runtime error? Explain.
(c) How does Java handle the problem with the assignment aArray[5]=x?

Answer

(a) Since bArray[i] is a subtype of aArray[i] for every i

(b) At the mentioned statement aArray points to an array of objects of type B. Therefore, when the condition does not hold, x points to an object of type A, and therefore the assignment is not type-safe

(c) Java handles this by generating code to conduct type-checking during runtime. The generated code finds that the runtime type of x is X and the runtime type of the aArray is Y[] and checks that X is a subtype of X.
If this condition doesn't hold it throws a ClassCastException

Possible question

- Support Java override annotation *inside comments*
  - // @Override
  - Annotation is written above method to indicate it overrides a method in superclass
- Describe the phases in the compiler affected by the change and the changes themselves

Legal program

```
class A {
  void rise() {…}
}
class B extends A {
  // @Override
  void rise() {…}
}
```

Illegal program

```
class A {
  void rise() {…}
}
class B extends A {
  // @Override
  void raise() {…}
}
```

13

Answer

- The change affects the lexical analysis, syntax analysis and semantic analysis
- It does not effect later phases since the annotation is meant to add a semantic condition by the user

14

Changes to scanner

- Add pattern for @Override inside comment state patterns
- Add Java code to action to comments – instead of not returning any token, we now return a token for the annotation
- What if we want to support multiple annotations in comments?

```
boolean override=false;
%%
<INITIAL> // { override=false; yybegin(comment); }
<comment> @Override { override=true; }
<comment> \n { if (override)
                return new Token(…,override,…)
             }
```

15

Changes to parser and AST

- Suppose we have rule
  - method → static type name params '{' mbody '}'
    | type name params '{' mbody '}'
  - Since that annotation is legal only for instance methods we rewrite the rule into
    method → static type name params '{' mbody '}'
    | type name params '{' body '}'
    | OVERRIDE type name params '{' mbody '}'
- We need to add a Boolean flag to the method AST node to indicate that the method is annotated

16

Changes to semantic analysis

- Suppose we have an override annotation above a method *m* in class *A*
- We check the following semantic condition using the following inform
  1. We check that the class *A* extends a superclass (otherwise it does not make sense to override a method)
  2. We check the superclasses of A by going up the class hierarchy until we find the first method *m* and check that it has the same signature as *A.m*
     If we fail to find such a method we report an error
- We use the following information
  - Symbol tables
  - Class hierarchy
  - Type table (for the types of methods)

17

Translation to IR

- Accept annotated AST and translate functions into lists of instructions
  - Compute offsets for fields and virtual functions
- Issues: dispatch tables, weighted register allocation
- Support extensions, e.g., translate switch statements
- Question: give method tables for **`Rectangle`** and **`Square`**

```
class Shape {
  boolean isShape() {return true;}
  boolean isRectangle() {return false;}
  boolean isSquare() {return false;}
  double surfaceArea() {…}
}
class Rectangle extends Shape {
  double surfaceArea() {…}
  boolean isRectangle() {return true;}
}
class Square extends Rectangle {
  boolean isSquare() {return true;}
}
```

18

Answer

Method table for rectangle

Shape_isShape
Rectangle_isRectangle
Shape_isSqaure
Rectangle_surfaceArea

Method table for square

Shape_isShape
Rectangle_isRectangle
Sqaure_isSqaure
Rectangle_surfaceArea

19

LIR translation

```
// An example program
class Hello {
  boolean state;
  static void main(string[] args) {
    Hello h = new Hello();
    boolean s = h.rise();
    Library.printb(s);
    h.setState(false);
  }
  boolean rise() {
    boolean oldState = state;
    state = true;
    return oldState;
  }
  void setState(boolean newState) {
    state = newState;
  }
}
```

Compute method and field offsets

fieldToOffset

DVPtr = 0
state = 1

methodToOffset

_Hello_rise = 0
_Hello_setState=1

```
_DV_Hello: [_Hello_rise,_Hello_setState]

_Hello_rise:
  Move this,R0
  MoveField R0.0,R0
  Move R0,oldState
  Return oldState

_Hello_setState:
  Move this,R0
  Move newState,R1
  MoveField R1,newR0.0

_ic_main:
  __allocateObject(8),R0
  MoveField _DV_Hello,R0.0
  Move R0,h
  Move h,R0
  VirtualCall R0.0(),R0
  Move R0,s
  Library __printb(s),Rdummy
  Move h,R0
  VirtualCall R0.1(newState=0)
```

Suggestion: play around with microLIR

Sometimes real offsets computed during (assembly) code generation

20

Possible question

- Suppose we wish to provide type information during runtime, e.g., to support operators like `instanceof` in Java
- The operator returns true for `x instanceof A` iff x is **exactly** of type A (in Java it can also be subtype of A)
- Describe the changes in runtime organization needed to support this operator and the translation to IR

21

Answer

- As a very restricted solution, we can avoid any changes to the runtime organization by using the pointers to the dispatch table as the type indicators
- We translate `x instanceof A` as
  ```
  Move x,R0
  MoveField R0.0,R0
  Compare R0,_DV_A
  ```
- The comparison is true iff the dispatch table of the object pointed-to by x is the dispatch table of class A, meaning that the object is of type A
- If we want to support the Java operator we must represent the type hierarchy during runtime and generate code to search up the hierarchy (generating code with loops)

22

Code generation

- Translate IR code to assembly
- Not in exam:
  - Run simple example and draw frame stacks in different points of execution
  - Interaction between register allocation and caller/callee save registers
- Might be in exam:
  - Activation records and call sequences – conceptual level

23

Other issues:

- Liveness analysis

24

Sample questions: 2007 a,b

25

שאלה (25 נקודות) : סיווג מאורעות

סווגי את המאורעות הבאים לפי זמן ריצה, זמן קומפילציה, או זמן בניית הקומפיילר.
פרטי **מתי בדיוק מתרחש כל מאורע ואילו נתונים נדרשים** אליו על-ידי הקומפיילר או כותבת הקומפיילר, כלומר באילו מבני נתונים ואלגוריתמים משתמשים לצורך המאורע.
במידה ויש מספר תשובות נכונות, הסבירי את כולן:

א- (5 נק') נמצא כי אובייקט o אינו נגיש, ולכן ניתן להשתמש בזיכרון שהוקצה לו להקצאה אחרת.

ב- (5 נק') נמצא כי בביטוי x.f=f, ה-f מימין להשמה הינו משתנה מקומי וה-f משמאל להשמה הינו שדה של המחלקה A.

ג- (5 נק') נמצא כי גודל טבלת הפונקציות הוירטואליות (dispatch table) של אובייקטים מהמחלקה A הוא 16 בתים וגודל טבלת הפונקציות הוירטואליות של המחלקה B שיורשת מ-A הוא 20 בתים.

ד- (5 נק') התגלה כי הפרמטר הפורמאלי הראשון בעל טיפוס שונה מהפרמטר האקטואלי הראשון של פונקציה המוגדרת כ-extern (פונקציות המוכרזות כ-extern אינן מוגדרות ביחידת הקומפילציה הנוכחית).

ה- (5 נק') מתבצעת שמירה של ערך האוגר eax ברשומת ההפעלה.

26

תשובה:

א- האירוע מתרחש בזמן ריצה, בשלב הסריקה של אוסף הזבל (garbage collector).

ב- האירוע מתרחש בזמן קומפילציה בשלב הניתוח הסמנטי כאשר בונים את טבלאות הסמלים.

ג- האירוע מתרחש בשלב ייצור הקוד (קוד ביניים או קוד אסמבלי), כאשר מחשבים נתונים הנדרשים לייצוג אובייקטים ממחלקות שונות בזיכרון.

ד- האירוע מתרחש בזמן קומפילציה בשלב ה-linking, כאשר בודקים התאמת סמלים המוכרזים ביחידות הקומפילציה השונות לבין השימוש שלהם ביחידות קומפילציה אחרות.

ה- האירוע מתרחש בזמן ריצה, לקראת כניסה לפונקציה נקראת, כאשר נשמרים אוגרים המוגדרים כ- caller-saved

27

שאלה (20 נקודות) : ניתוח תחבירי

נתון הדקדוק הבא להשוואת מסלולי זיכרון בתכנית ע"י פקודת assert:

$S \rightarrow$ assert C
$S \rightarrow$ assert lp C rp
$C \rightarrow P$ eq P
$P \rightarrow$ id | id dot P

ה-tokens המופיעים בדקדוק מייצגים את המחרוזות המתאימות: lp:'(' rp:')' eq:'==' dot='.'.

דוגמאות לקלטים חוקיים הם הקלט "assert(x.a==y.b.c)" והקלט "assert x.n.n==y".
תזכורת: בשיטת LL(k) אלגוריתם הניתוח משתמש בטבלה הממפה כל k סימנים מהקלט (סדרה של טרמינלים ומשתני דקדוק) לכלל הגזירה שבו יש להשתמש כדי להחליף את משתנה הגזירה השמאלי ביותר.

א- (3 נק') מהו ערך של k הדרוש על-מנת לבצע ניתוח תחבירי לדקדוק הנתון בשיטת LL(k)? (כמה tokens קדימה צריך להסתכל כדי להחליט באיזה כלל גזירה לבחור?) הסבירי.

ב- (3 נק') האם ניתן להפחית את ערך k ע"י שינוי הדקדוק? אם כן, הראי כיצד. אם לא, הסבירי למה לא ניתן לעשות זאת.

ג- (4 נק') הראי שתת הדקדוק הניתן על-ידי כללי הגזירה של P (שני הכללים האחרונים) אינו שייך ל-LR(0).

ד- (10 נק') הראי כיצד ניתן לשכתב אותו כך שישתייך ל-LR(0). **רמז:** השתמשי בכלל החדש id→P'.
בני את דיאגרמת המצבים (האוטומט) של הדקדוק המשוכתב ואת טבלת הניתוח, והראי כיצד מתקבל הביטוי "x.n.n".

28

תשובה 2

א- הדקדוק נמצא ב-LL(2), זאת מכיוון שכדי להבדיל בין הכלל $S \rightarrow$ assert C לכלל $S \rightarrow$ assert lp C rp דרוש להסתכל על ה-token השני, כלומר זה שבא לאחר assert ולבדוק האם הוא lp או token אחר. כמו-כן, כדי להבדיל בין הכלל $P \rightarrow$ id לכלל $P \rightarrow$ id dot P דרוש לבדוק האם ה-token שלאחר id הוא dot או token אחר. בכל מקרה יש לבדוק את שני ה-tokens הבאים בקלט.

ב- כן, ניתן לבצע left-factoring ע"י הוצאת רישות משותפות של הכללים לכללים אחרים:

$S \rightarrow$ assert C'
$C' \rightarrow C$ | lp C rp
$C \rightarrow P$ eq P
$P \rightarrow$ id P'
$P' \rightarrow \varepsilon$ | dot P

כעת ניתן להבדיל בין הכללים הנגזרים מ-C' ומ-P' ע"י בדיקת ה-token הראשון בלבד. לכן הדקדוק המשוכתב שייך ל-LL(1).

ג- אוטומט המצבים יכלול מצב שבו קיימים שני items-LR(0) הבאים: $P \rightarrow$ id • dot -ו$P \rightarrow$ id • P, ולכן ישנו shift/reduce conflict, ולכן הדקדוק אינו שייך ל-LR(0).

ד- ניתן לשכתב אותו כך:

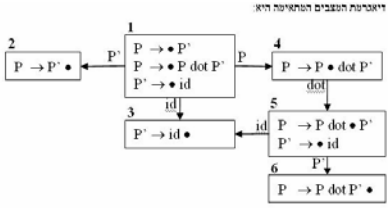
$P \rightarrow P'$
$P \rightarrow P$ dot P'
$P' \rightarrow$ id

(ניתן גם להוסיף לדקדוק סימן סוף קלט וכלל מתאים, אך התשובה הניתנת כאן היא ללא הכלל והסימן.)

29

המשך סעיף ד'

דיאגרמת המצבים המתאימה היא:



כפי שניתן לראות לא קיימים מצבים בעלי קונפליקטים ולכן הדקדוק שייך ל-LR(0). טבלת הניתוח המתאימה היא:

	id	dot	P	P'
1	3		4	2
2	reduce P → P'			
3	reduce P' → id			
4		5		
5	3			6
6	reduce P → P dot P'			

30

שאלה 4 : הקצאת אוגרים גלובאלית

להלן תוכנית בשפת ביניים ונתוני החיות של המשתנים המופיעים בה (תוצאת ההפעלה של אלגוריתם ה-liveness analysis שנלמד בכיתה). שימי-לב שנתוני החיות מתייחסים למצב אחרי ביצוע הפקודה המתאימה.

```
enter:
x := 8;  /* x, r1, r2 */
y := r1; /* x, y, r1 */
z := r2; /* x, y, z */
loop:
x := x - 1; /* x, y, z */
y := y + z; /* x, y */
z := 2;     /* x, y */
if x > 0 goto loop;
r1 := y;    /* r1 */
return;     /* r1 */
```

א- (10 נק') בני את גרף ההפרעות (ה-interference graph) המתאים לנתוני החיות של המשתנים ולתכנית.

ב- (15 נק') הפעילי את אלגוריתם ה-graph coloring על גרף ההפרעות מהסעיף הקודם והניחי שבמכונה יש בדיוק שלושה רגיסטרים r1, r2 ו-r3, כאשר r1 ו-r2 המופיעים בקוד מתאימים לרגיסטרים בעלי אותו שם.
כמו-כן השתמשי בקריטריון של Briggs לביצוע איחוד צמתים (coalescing): שני צמתים a ו-b ניתנים לאיחוד אם לצומת המאוחד יהיו פחות מ-k שכנים בעלי דרגה גדולה או שווה ל-k (בשאלה הנוכחית k=3).

להזכירך, אלגוריתם ה-graph coloring, כפי שנלמד בכיתה בוחר את הרגיסטר לצורך spill ע"י יוריסטיקה המשתמשת
ב-priority המוגדר כך:

priority = (uo + 10 * ui) / deg
uo = use + def outside of the loop
ui = use + def within the loop
deg = degree in the interference graph

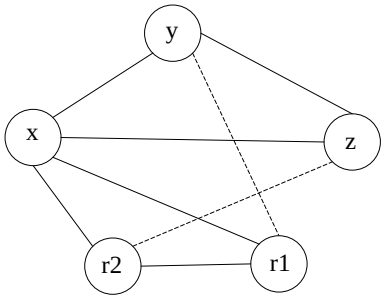
הרצת ה-parser על x.n.n

Stack	Input	Action
S1	id dot id dot id	shift
S1 id S3	dot id dot id	reduce P'→id
S1 P' S2	dot id dot id	reduce P→P'
S1 P S4	dot id dot id	shift
S1 P S4 dot S5	id dot id	shift
S1 P S4 dot S5 id S3	dot id	reduce P'→id
S1 P S4 dot S5 P' S6	dot id	reduce P→P dot P'
S1 P S4	dot id	shift
S1 P S4 dot S5	id	shift
S1 P S4 dot S5 id S3		reduce P'→id
S1 P S4 dot S5 P' S6		reduce P→P dot P'
S1		stop

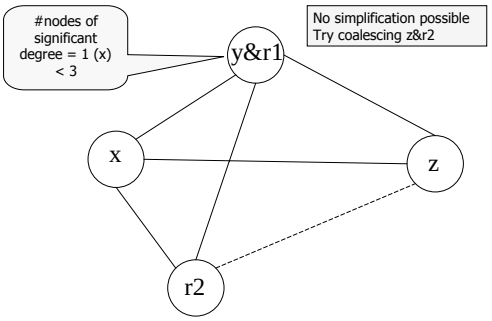
Spill priorities (uo + 10 * ui) / deg

temp	use+def out of loop	use+def within loop	degree	priority
x	1	3	4	7.75
y	2	2	2	11
z	1	2	2	10.5

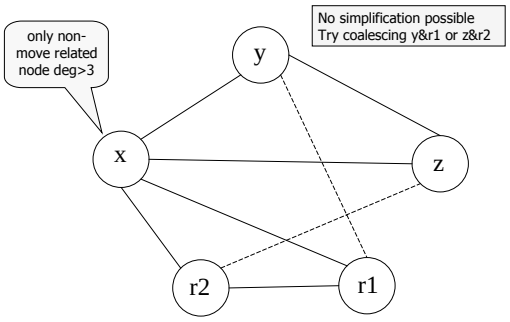
תשובה : גרף ההפרעות



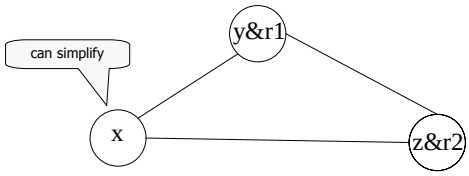
תשובה : הרצת אלגוריתם הצביעה



תשובה : הרצת אלגוריתם הצביעה

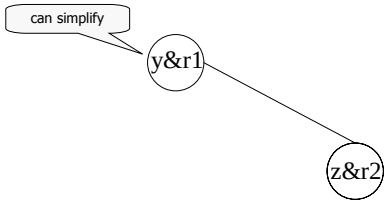


תשובה : הרצת אלגוריתם הצביעה



37

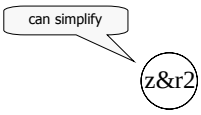
תשובה : הרצת אלגוריתם הצביעה



color stack
x

38

תשובה : הרצת אלגוריתם הצביעה



color stack
y&r1
x

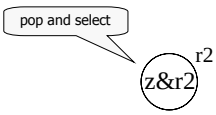
39

תשובה : הרצת אלגוריתם הצביעה

color stack
z&r2
y&r1
x

40

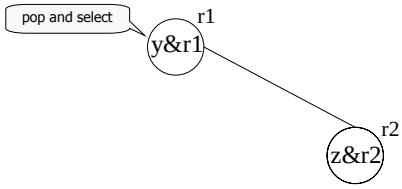
תשובה : הרצת אלגוריתם הצביעה



color stack
y&r1
x

41

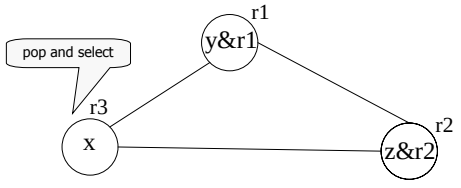
תשובה : הרצת אלגוריתם הצביעה



color stack
x

42

תשובה : הרצת אלגוריתם הצביעה



color stack

43

Good luck
in the exam!

44