CZ3007 Compiler Techniques 2018/2019 Semester 1

Solver: Tan Pei Ting Stella
Email Address: TA0004LA@e.ntu.edu.sg

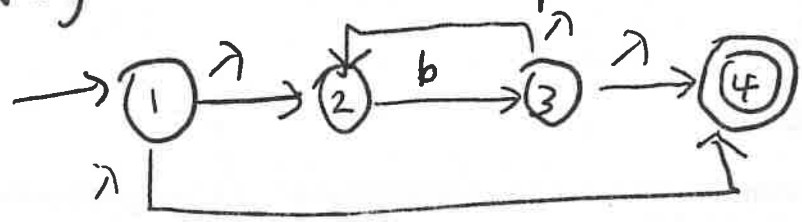
1ai.
Input: A stream of characters
Lexer to parser: A stream of tokens
Parser to semantic analyzer: An abstract syntax tree
Semantic analyser to code generator: An abstract syntax tree of the program

1aii.
Input: Source program + Data input
Output: Output of program + Error messages

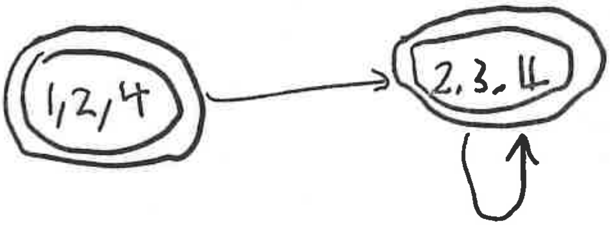
1b.
(0 | [1-9] ([0-9]?)^2 (','?[0-9]^3)*) '.' ([0-9][0-9]?)

1c.
McNaughton-Yamada-Thompson method is used to construct NFAs from regular expressions.
Subset construction algorithm is used to transform an NFA to a DFA.

McNaughton-Yamada-Thompson



Subset construction algorithm

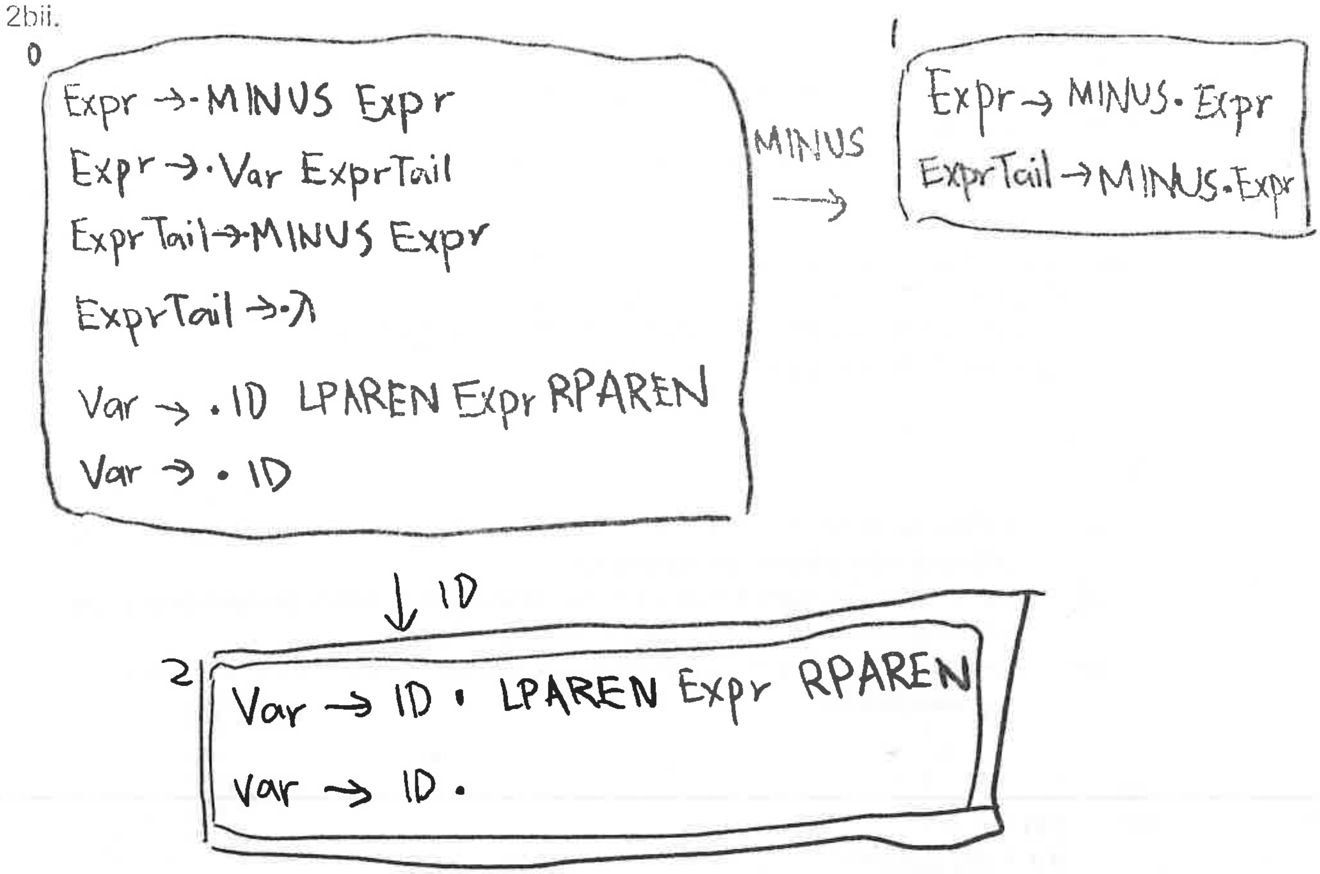


29
/4

2a.
LinearP -> ObjectiveFunction ConstraintList
ConstraintList -> Constraint ConstraintList
ObjectiveFunction -> Keyword Expression
Keyword -> MAXIMIZE
| MINIMIZE
Expression -> Expression AddOp Term
| Term
AddOp -> PLUS
| MINUS
Term -> ID
| INTLITERAL MUL ID
Constraint -> Expression CompOp INTLITERAL
CompOp -> EQEQ
| LEQ
| GEQ

2bi.
predict(p1) = {MINUS}
predict(p2) = {ID}
predict(p3) = {MINUS}
predict(p4) = {RPAREN}
predict(p5) = {ID}
predict(p6) = {ID}
Expr -> Var ExprTail
| ExprTail
ExprTail -> MINUS Expr
| λ
LL(1) requires a unique combination of a nonterminal and a lookahead symbol to decide which rule to use. If there is a common prefix, the LL(1) would not know which rule to use based on the first symbols, hence there will be an error.

2bii.



State 1: No conflict, shift-shift
State 2: Shift-reduce conflict
The contents of the itemFollow set determines the columns of the parse table and also which action to take for each stage.

3ai.

1. The semantic rules cannot be expressed with context-free grammars
2. Separation of concerns
3. To make the compiler easier to understand, extend and maintain

3aii.
Abstract grammar -> Java classes
Attribute grammar -> Java methods

3aiii.

1. Duplicate variable declaration
2. Undeclared variable
3. Inconsistent types in assignment
4. Inconsistent types in operand
5. Variable name not allowed
6. Array index out of range

30

2#

3bi.
Box = HORIZONTAL NUM.w NUM.h Box.l Box.r {: return new HBox(w, h, new Box(l), new Box(r)) :}
| ATOMIC NUM.w NUM.h {: return new ABox(w, h) :}

3bii.
```
public void HBox.dimensioncheck() {
        if (getLeft().xpos() != xpos()) return error;
        if (getRight().xpost() != xpox() + getLeft().width()) return error;
        getChild().dimentsioncheck();
}
```

4a.

(i) False, having n IR allows the compiler to break up the code generation process into simpler components

(ii) False, the JVM stack frames have limited size, which should only store variables needed in the function that is running

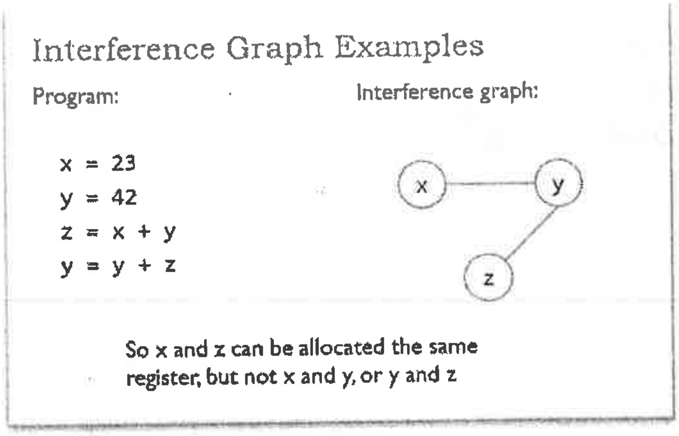
(iii) False, a + b > 2 has 2 operators and hence it is not in the 3-address representation

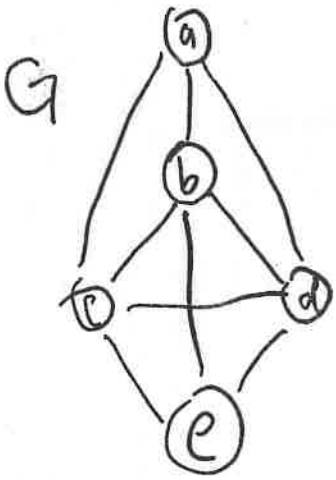
4b.
```
l0:     $i0 = n + 1;
        if x > $i0 goto l1;
        $i1 = y + 1;
        $i2 = y * $i1;
        y = y + $i2;
        if y > n goto l1;
        x = x + 1;
        goto l0;
l1:
```

4c.
```
l0:     load x
        load n
        load 1
        add
        ifgt l1
        load y
        load 1
        add
        load y
        mul
        load y
        add
        store y (note: do not remove store/load y here as the updated y value will be
        used in the next loop)
        load y
        load n
```

```
        ifgt l1
        load x
        load 1
        add
        store x
        load 1
        load 0
        ifgt l0
l1:
```

4d.
An interference graph is a graph containing one node for each local variable and undirected edges for variables that are live at the same time at some point int the program.

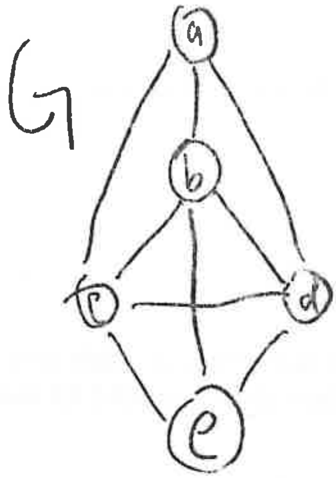




Stack: empty

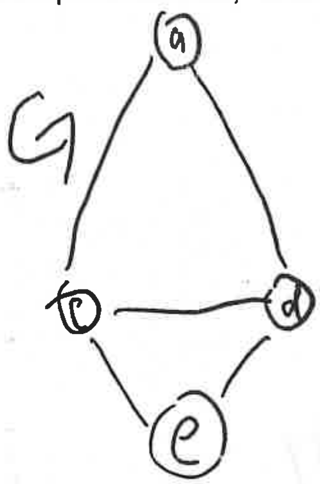
31
34

Remove nodes with less than 3 neighbours and push onto stack



Stack: empty

Choose spill candidate, remove and push onto stack: b is spill candidate

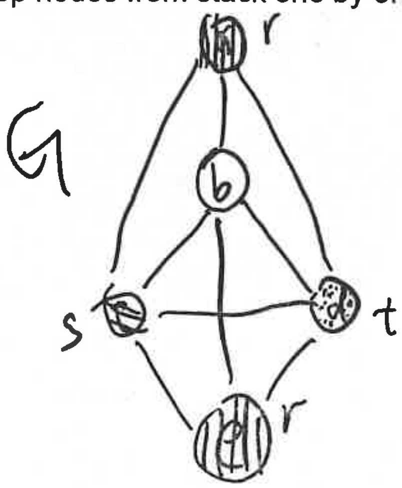


Stack: b*

Remove nodes with less than 3 neighbours and push onto stack

Stack: b*, a, c, d, e

Pop nodes from stack one by one



It is possible to allocate 3 registers r, s and t, but there is register spilling. Hence G is 3-colourable.

5a.

(i) Incorrect, as hardware usage can be limited, eg. Mobile phone cannot store many cores.

(ii) Incorrect, there can be many predecessors in a CFG

(iii) Correct, as if it is live after any predecessor nodes, it means the variable will be read in or after the current node, hence it must be live before the current node.

(iv) Correct

- Available Expressions is a *forward-must* analysis: whether an expression is available after a node depends on whether it is available before the node (but not vice versa), and an expression is available before a node if it is available after *every* predecessor node

5b.

Correctness Conditions

- There are two conditions that need to hold before we can eliminate a common subexpression:
  1. The expression must have been computed previously on every possible execution path, not just on one
  2. None of the variables involved in computing the expression may have been updated in the meantime
- Hence, we cannot eliminate subexpressions in this example:

```
x = y + z
y = y + 1
r = y + z
```

- Nor in this:

```
    if z > 0 goto l
    x = y + z
l:  r = y + z
```

5c.

$in_L(1) = out_L(1)$
$in_L(2) = out_L(2) \backslash z \cup \{y\}$
$in_L(3) = out_L(3) \cup \{x,z\}$
$in_L(4) = out_L(4) \cup \{x\}$
$in_L(5) = out_L(5) \cup \{z\}$
$in_L(6) = out_L(6) \cup \{y,z\}$
$in_L(7) = out_L(7) \cup \{x\}$
$in_L(8) = out_L(8)$

$out_L(1) = in_L(2)$
$out_L(2) = in_L(3)$
$out_L(3) = in_L(4) \cup in_L(7)$
$out_L(4) = in_L(5)$
$out_L(5) = in_L(6)$
$out_L(6) = in_L(3)$
$out_L(7) = in_L(8)$
$out_L(8) = \emptyset$

32

4 #

5d.
$in_L(1) = in_L(2)$
$in_L(2) = in_L(3) \setminus z \cup \{y\}$
$in_L(3) = in_L(4) \cup in_L(7) \cup \{x,z\}$
$in_L(4) = in_L(5) \cup \{x\}$
$in_L(5) = in_L(6) \cup \{z\}$
$in_L(6) = in_L(3) \cup \{y,z\}$
$in_L(7) = in_L(8) \cup \{x\}$
$in_L(8) = \emptyset$

	0	1	2	3	4	5
$in_L(1)$	∅	∅	y	x,y	x,y	x,y
$in_L(2)$	∅	y	x,y	x,y	x,y	x,y
$in_L(3)$	∅	x,z	x,z	x,z	x,y,z	x,y,z
$in_L(4)$	∅	x	x,z	x,y,z	x,y,z	x,y,z
$in_L(5)$	∅	z	y,z	x,y,z	x,y,z	x,y,z
$in_L(6)$	∅	y,z	x,y,z	x,y,z	x,y,z	x,y,z
$in_L(7)$	∅	x	x	x	x	x
$in_L(8)$	∅	∅	∅	∅	∅	∅