Thomas Mumtu
CS4250-Mishra
2/9/18

(i) Context Free Grammar for creating boolean expressions / identifiers in C language

~~<expression> → <unary op> <term> [<addop> <term>]*~~

<expression> → <term> [<addop> <term>]*

<term> → <signed factor> [<mulop> factor]*

<signed factor> → [<addop>] <factor>

<factor> → <integer> | <variable> | (<expression>)

For Boolean Expressions:

<b-expression> → <b-term> [<orop> <b-term>]*

<b-term> → <not-factor> [AND <not-factor>]*

<not-factor> → [NOT] <b-factor>

<b-factor> → <b-literal> | <b-variable> | (<b-expression>)

<relation> → <expression> [<relop> <expression>] (Numeric type)

<expression> → <term> [<addop> <term>]*

<term> → <signed factor> [<mulop> factor]*

<signed factor> → [<addop>] <factor>

<factor> → <integer> | <variable> | (<b-expression>)

AND → *

OR → +

NOT → -

=, <, >, !=, <=, =>

exp → term [OR term]

term → factor [AND factor]

factor → ID

factor → NOT factor

factor → LPAREN exp RPAREN

True & False are IDs

E → TE'

E' → OR T E'

E' → ϵ

T → FT'

T' → AND FT'

T' → ϵ

F → NF'

N → NOT

N → ϵ

F' → (E)

F' → id

Thomas Minton
CS425 - Mishra
2/9/18

(2) Remove indirect Left recursion from

$S \to Aa/Bb$ $A \to Aa/Abc/c/Sb$ $B \to bb$

Maximum common prefix: Aa

$S \to Aa/Bb$ $A \to S/ABc/c/Sb$ $B \to bb$

~~$A \to A'cf b$~~

~~$A' \to$~~

~~$A \to Aa/Abc/c/Aab/Bb$~~

~~$A \to A(a/bc/ab)/$~~

$A \to Aa/Abc/c/(Aa/Bb)b$

$\to Aa/Abc/c/Aab/Bbb$

$\to A(a/bc/ab)/(c/Bbb)$

α

~~$A \to BA'$~~

$A' \to \alpha A'/\epsilon$

~~$A' = (c/bbb)$~~

This goes 1st @ top of A

* Small letter are Terminal
* Capital letter = Non-Terminal

② $A \to c/Bbb/A$

③ $A' \to a/bc/ab/\epsilon$

① $S \to Aa/Bb$

④ $B \to bb$

Thomas Mantu
CS4250
HW-1

(3) Grammar:

<program> → begin <stmt-list> end

<stmt-list> → <stmt> / <stmt> ; <stmt-list>

<stmt> → <var> = <expression>

<var> → A / B / C

<expression> → <var> + <var> / <var> - <var> / <var>

is ~~A=B+C;~~

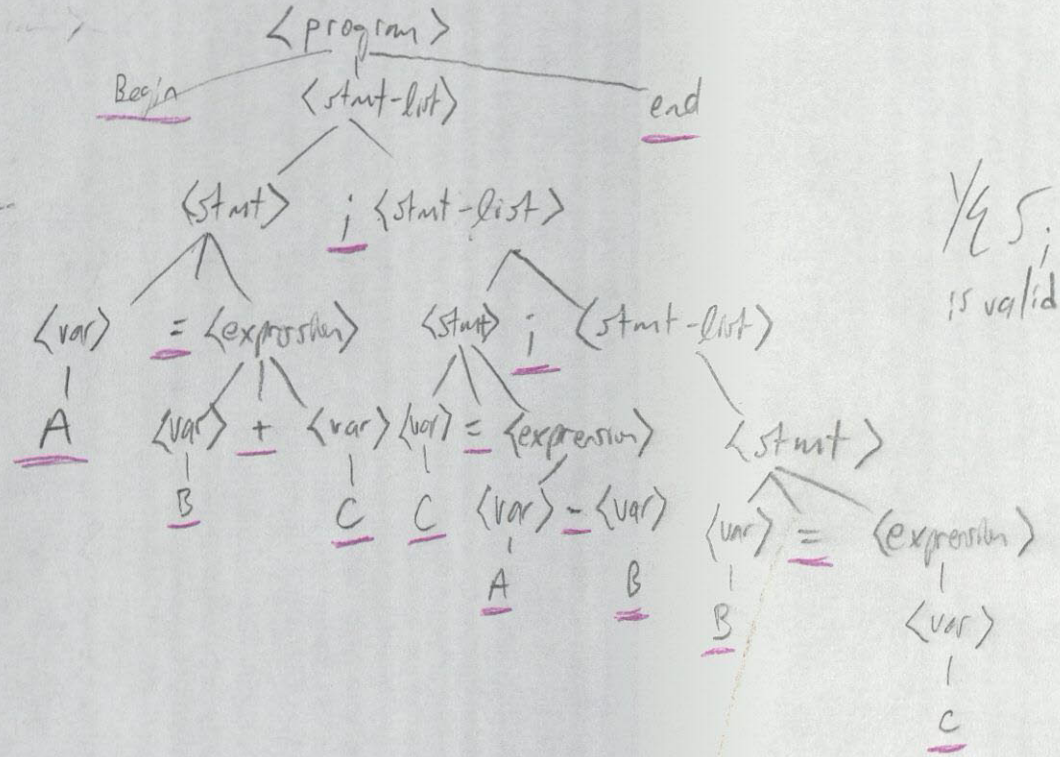
~~A=B+C;~~

is

begin A=B+C; C=A-B; B=C end

valid?

Assuming ment to be Upper case.



YES; A=B+C; C=A-B; B=C is valid for given grammar

Thomas Mundan
CS 4250 - Mishra
2/9/18

(4) <translation-unit> → (<external declaration>)

<external declaration> → <function definition> / <declaration>

<function-definition> → (<declaration specifier>) <declarator>

<declaration specifier> → <storage class-specifier> / <type-specifier> / <type qualifier>

<type-specifier> → <void> / <char> / <short> / <int> / <long> / <float> / <double>
<struct-or-union-specifier>

<struct-or-union-specifier> → <struct-or-union> <identifier> (<struct-declaration>)+ /
<struct-or-union> <identifier>

<struct-or-union> → <struct> / <union>

<struct-declarator> → <declarator> / <declarator> : <constant expression> / : <constant expression>

<declarator> → (<pointer>)? <direct declarator>

<pointer> → * <type-qualifier> * <pointer>?

<direct-declarator> → <identifier> / <direct-declarator> (<constant-expression>) /

<constant expression> → <conditional-expression>

<conditional-expression> → <logical or expression>

<logical-or-expression> → <logical and expression> / <logical and expression> / <logical or expression>

<logical and expression> → <inclusive or expression> / <logical and expression> && <inclusive or expression>

<inclusive or expression> → <exclusive or expression> / <inclusive or expression> / <exclusive or exp>

<exclusive or expression> → <and-expression> / <exclusive or exp> ^ <and expression>

<and expression> → <equality expression> / <and expression> & <equality expression>

Thomas Morton
CS4250 - Mishra
2/9/18

(5) No. Working HTML that is widely used is NOT context free grammar.
HTML 4.01 does NOT conform to SGML according to HTML 4.01 doctype definition.
However, HTML 5 is the first HTML standard to be defined before being implemented, ~~and~~ HTML 5 is the first HTML to strictly define parsing behavior of code. Pre-HTML 5 parsers vary, and each do their "best" to "guess" the intention of the code.