More Applications of The Pumping Lemma

The Pumping Lemma:

For infinite context-free language L

there exists an integer m such that

for any string $w \in L, \quad |w| \geq m$

we can write $w = uvxyz$

with lengths $|vxy| \leq m$ and $|vy| \geq 1$

and it must be:

$$uv^i xy^i z \in L, \quad \text{for all } i \geq 0$$

Non-context free languages

$\{a^n b^n c^n : n \geq 0\}$ $\{ww : w \in \{a,b\}\}$

Context-free languages

$\{a^n b^n : n \geq 0\}$ $\{ww^R : w \in \{a,b\}^*\}$

Theorem: The language

$$L = \{ww : w \in \{a,b\}^*\}$$

is **not** context free

Proof: Use the Pumping Lemma for context-free languages

$$L = \{ww : w \in \{a,b\}^*\}$$

Assume for contradiction that L

is context-free

Since L is context-free and infinite

we can apply the pumping lemma

$$L = \{ww : w \in \{a,b\}^*\}$$

Pumping Lemma gives a magic number m such that:

Pick any string of L with length at least m

we pick: $a^m b^m a^m b^m \in L$

$$L = \{ww : w \in \{a,b\}^*\}$$

We can write: $a^m b^m a^m b^m = uvxyz$

with lengths $|vxy| \leq m$ and $|vy| \geq 1$

Pumping Lemma says:

$$uv^i xy^i z \in L \quad \text{for all} \quad i \geq 0$$

$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

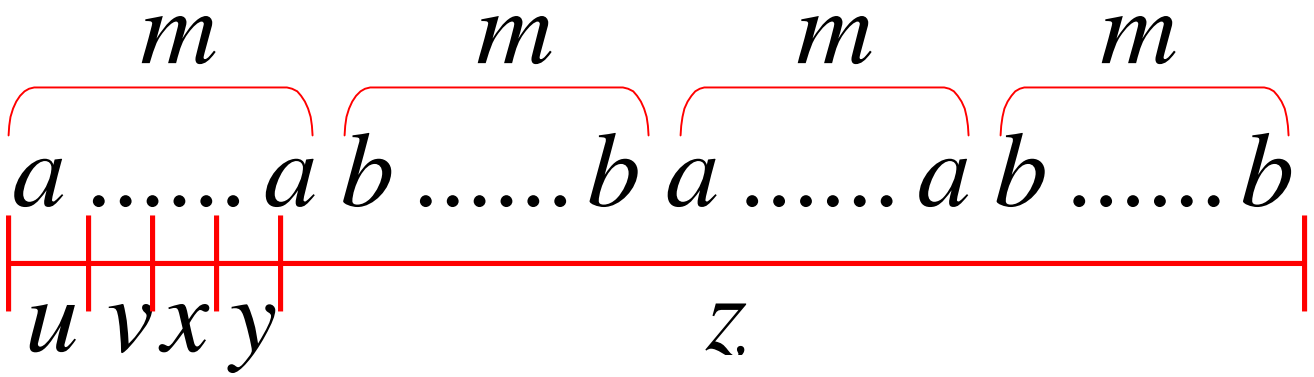
We examine **all** the possible locations of string vxy in $a^m b^m a^m b^m$

$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 1: vxy is within the first a^m

$$v = a^{k_1} \qquad y = a^{k_2} \qquad k_1 + k_2 \geq 1$$

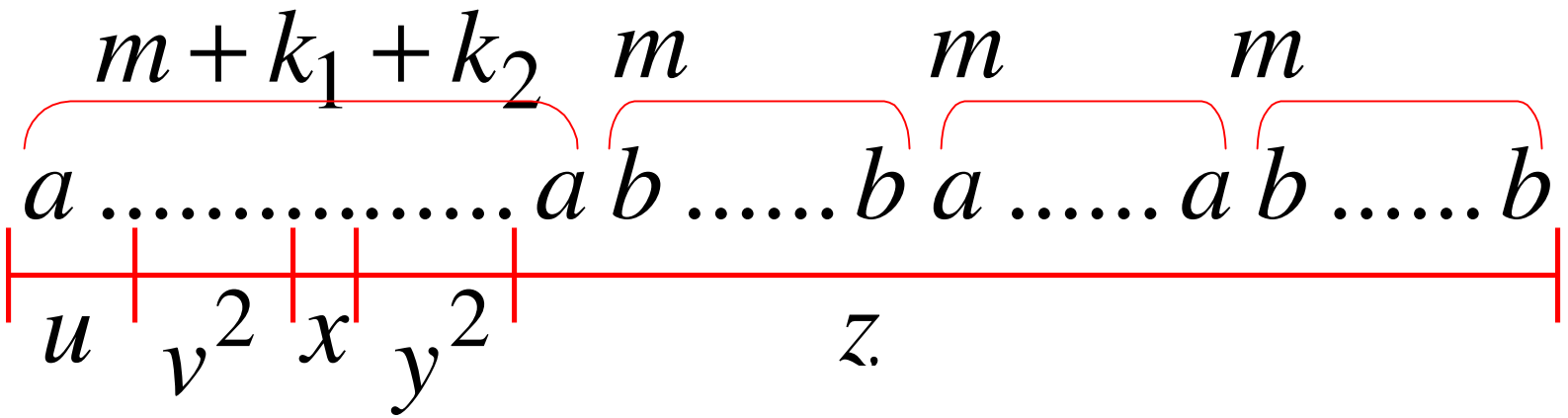


$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \le m \qquad |vy| \ge 1$$

Case 1: vxy is within the first a^m

$$v = a^{k_1} \qquad y = a^{k_2} \qquad k_1 + k_2 \ge 1$$



$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 1: vxy is within the first a^m

$$a^{m+k_1+k_2} b^m a^m b^m = uv^2 xy^2 z \notin L$$

$$k_1 + k_2 \geq 1$$

$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 1: vxy is within the first a^m

$$a^{m+k_1+k_2} b^m a^m b^m = uv^2 xy^2 z \notin L$$

However, from Pumping Lemma: $uv^2 xy^2 z \in L$

Contradiction!!!

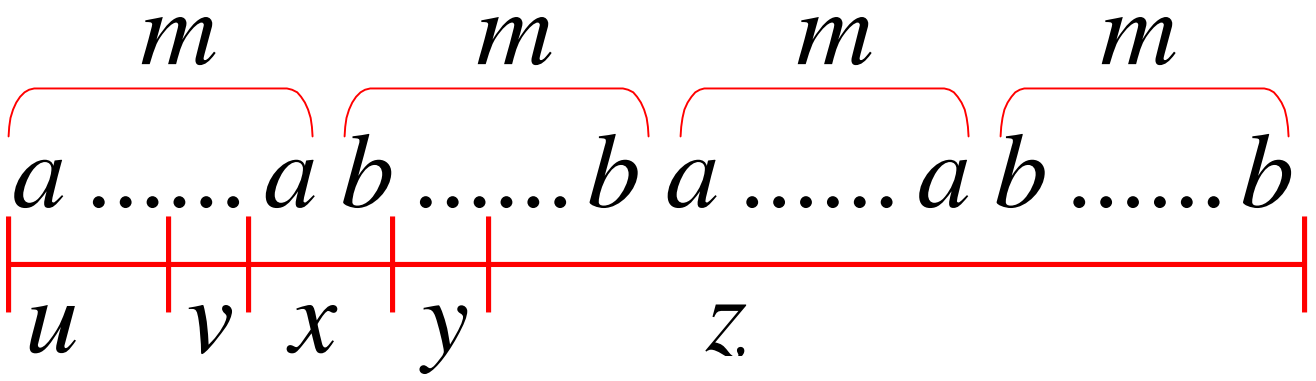
$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \le m \qquad |vy| \ge 1$$

Case 2: v is in the first a^m

y is in the first b^m

$$v = a^{k_1} \qquad y = b^{k_2} \qquad k_1 + k_2 \ge 1$$



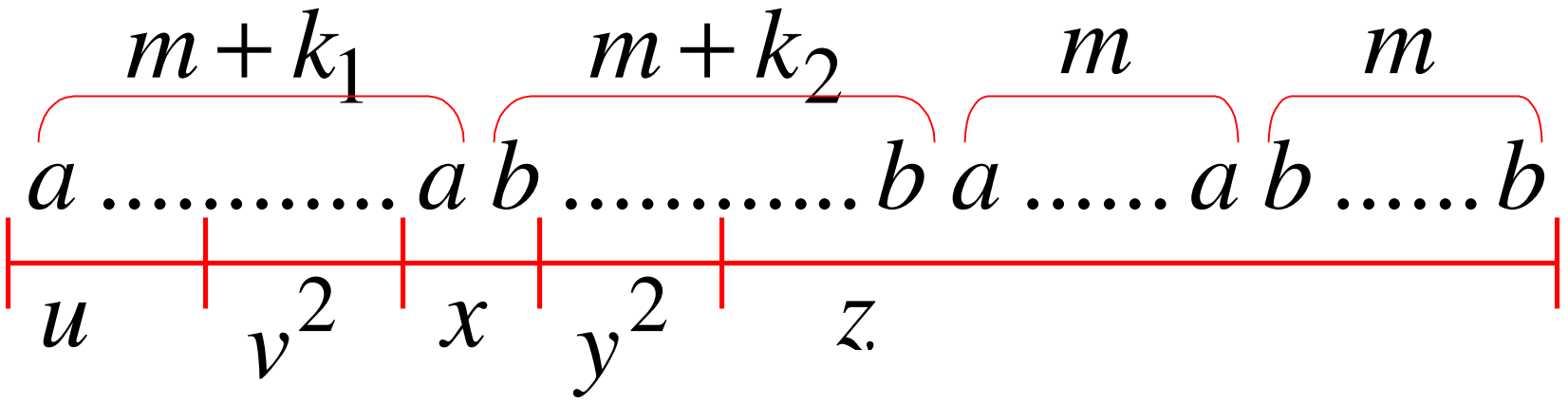
$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 2: v is in the first a^m

y is in the first b^m

$$v = a^{k_1} \qquad y = b^{k_2} \qquad k_1 + k_2 \geq 1$$



$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 2: v is in the first a^m

y is in the first b^m

$$a^{m+k_1} b^{m+k_2} a^m b^m = uv^2 xy^2 z \notin L$$

$$k_1 + k_2 \geq 1$$

$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 2: v is in the first a^m

y is in the first b^m

$$a^{m+k_1} b^{m+k_2} a^m b^m = uv^2xy^2z \notin L$$

However, from Pumping Lemma: $uv^2xy^2z \in L$

Contradiction!!!

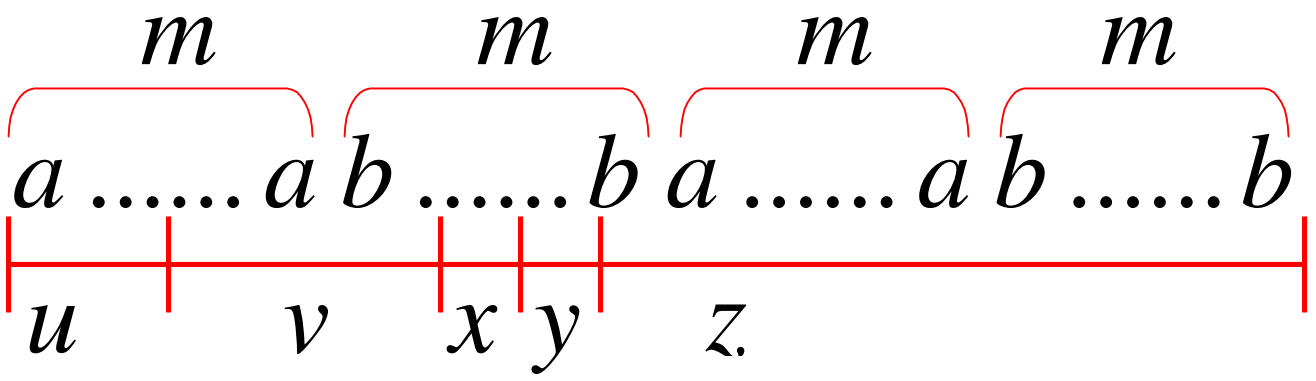
$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \le m \qquad |vy| \ge 1$$

Case 3: v overlaps the first $a^m b^m$

y is in the first b^m

$$v = a^{k_1} b^{k_2} \qquad y = b^{k_3} \qquad k_1, k_2 \ge 1$$



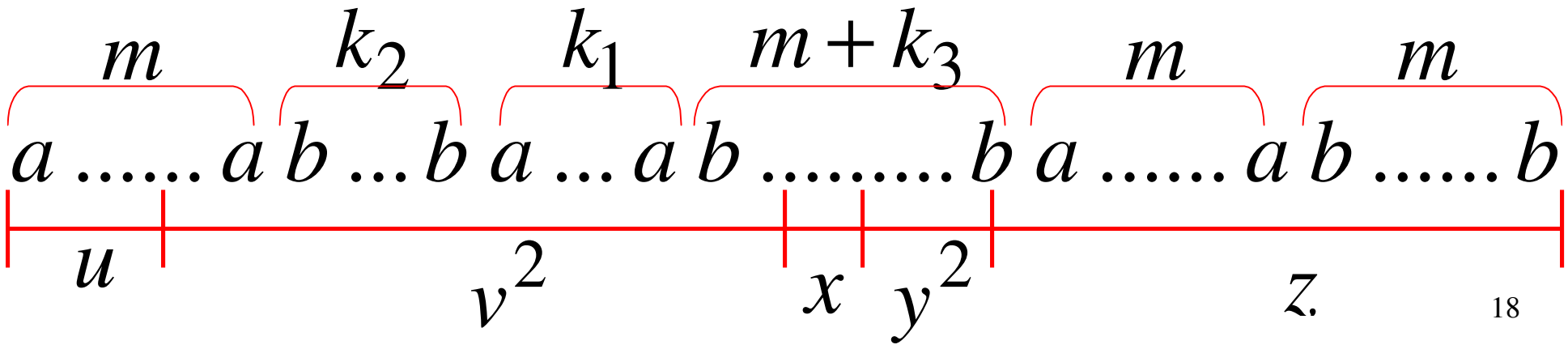
$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \le m \qquad |vy| \ge 1$$

Case 3: v overlaps the first $a^m b^m$

y is in the first b^m

$$v = a^{k_1} b^{k_2} \qquad y = b^{k_3} \qquad k_1, k_2 \ge 1$$



$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 3: v overlaps the first $a^m b^m$

y is in the first b^m

$$a^m b^{k_2} a^{k_1} b^{m+k_3} a^m b^m = uv^2xy^2z \notin L$$

$$k_1, k_2 \geq 1$$

$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 3: v overlaps the first $a^m b^m$

y is in the first b^m

$$a^m b^{k_2} a^{k_1} b^{k_3} a^m b^m = uv^2xy^2z \notin L$$

However, from Pumping Lemma: $uv^2xy^2z \in L$

Contradiction!!!

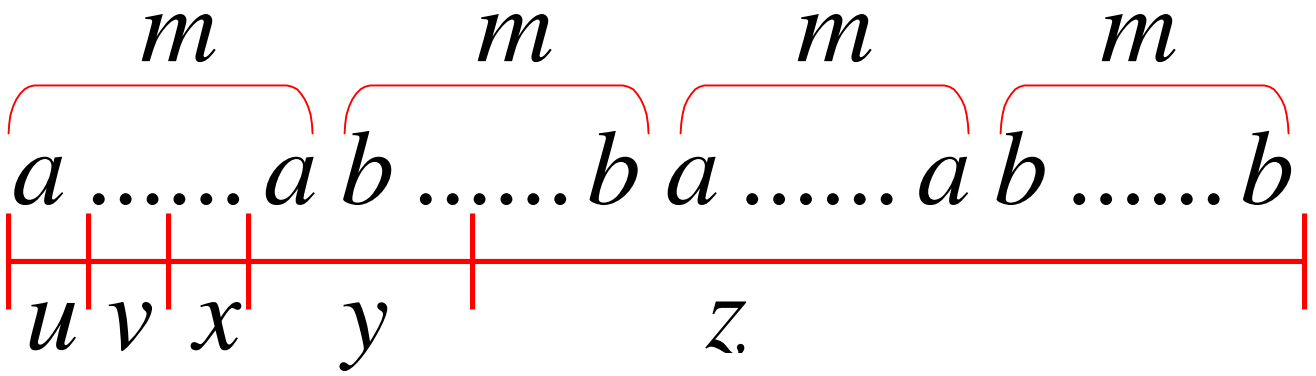
$$L = \{ww : w \in \{a,b\}^*\}$$

$$a^m b^m a^m b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 4: v in the first a^m

y Overlaps the first $a^m b^m$

Analysis is similar to case 3



Other cases: vxy is within $a^m b^m a^m b^m$

or

$a^m b^m a^m b^m$

or

$a^m b^m a^m b^m$

Analysis is similar to case 1:

$a^m b^m a^m b^m$

More cases: vxy overlaps $a^m b^m a^m b^m$

or

$a^m b^m a^m b^m$

Analysis is similar to cases 2,3,4:

$a^m b^m a^m b^m$

There are no other cases to consider

Since $|vxy| \leq m$, it is impossible vxy to overlap:

$$a^m b^m a^m b^m$$

nor

$$a^m b^m a^m b^m$$

nor

$$a^m b^m a^m b^m$$

In all cases we obtained a contradiction

Therefore: The original assumption that

$$L = \{ww : w \in \{a,b\}^*\}$$

is context-free must be wrong

Conclusion: L is not context-free

Non-context free languages

$\{a^n b^n c^n : n \geq 0\}$

$\{ww : w \in \{a,b\}\}$

$\{a^{n!} : n \geq 0\}$

Context-free languages

$\{a^n b^n : n \geq 0\}$

$\{ww^R : w \in \{a,b\}^*\}$

Theorem: The language

$$L = \{a^{n!} : n \geq 0\}$$

is **not** context free

Proof: Use the Pumping Lemma for context-free languages

$$L = \{a^{n!} : n \geq 0\}$$

Assume for contradiction that L

is context-free

Since L is context-free and infinite

we can apply the pumping lemma

$$L = \{a^{n!} : n \geq 0\}$$

Pumping Lemma gives a magic number m such that:

Pick any string of L with length at least m

we pick: $a^{m!} \in L$

$$L = \{a^{n!} : n \geq 0\}$$

We can write: $a^{m!} = uvxyz$

with lengths $|vxy| \leq m$ and $|vy| \geq 1$

Pumping Lemma says:

$$uv^i xy^i z \in L \quad \text{for all} \quad i \geq 0$$

$$L = \{a^{n!} : n \geq 0\}$$

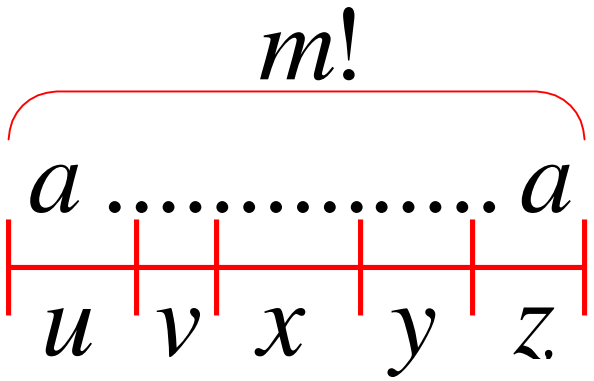
$$a^{m!} = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

We examine **all** the possible locations of string vxy in $a^{m!}$

There is only one case to consider

$$L = \{a^{n!} : n \geq 0\}$$

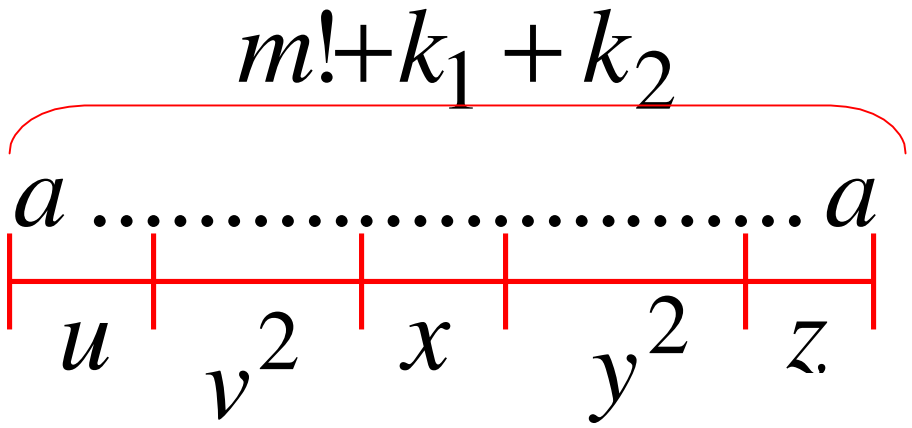
$$a^{m!} = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$



$$v = a^{k_1} \qquad y = a^{k_2} \qquad 1 \leq k_1 + k_2 \leq m$$

$$L = \{a^{n!} : n \geq 0\}$$

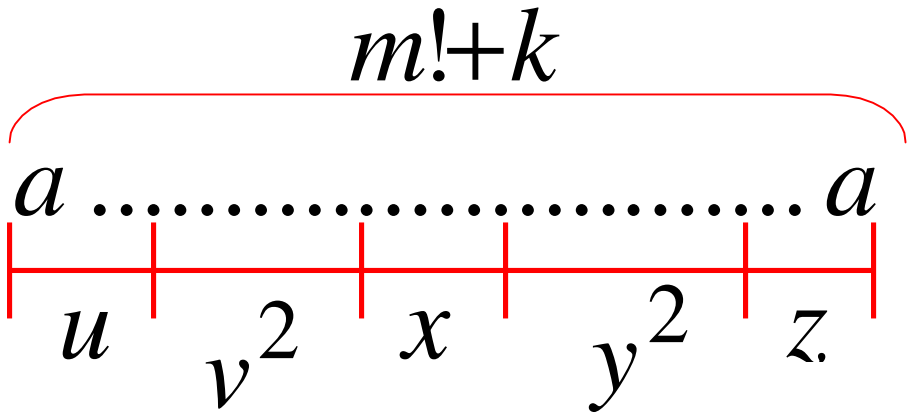
$$a^{m!} = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$



$$v = a^{k_1} \qquad y = a^{k_2} \qquad 1 \leq k_1 + k_2 \leq m$$

$$L = \{a^{n!} : n \geq 0\}$$

$$a^{m!} = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$



$$k = k_1 + k_2$$

$$v = a^{k_1} \qquad y = a^{k_2} \qquad 1 \leq k \leq m$$

$$L = \{a^{n!} : n \geq 0\}$$

$$a^{m!} = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

$$a^{m!+k} = uv^2xy^2z$$

$$1 \leq k \leq m$$

Since $1 \le k \le m$, for $m \ge 2$ we have:

$$m! + k \le m! + m$$

$$< m! + m!m$$

$$= m!(1+m)$$

$$= (m+1)!$$

$$m! < m! + k < (m+1)!$$

$$L = \{a^{n!} : n \geq 0\}$$

$$a^{m!} = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

$$m! < m! + k < (m+1)!$$

$$a^{m!+k} = uv^2xy^2z \notin L$$

$$L = \{a^{n!} : n \geq 0\}$$

$$a^{m!} = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

However, from Pumping Lemma: $uv^2xy^2z \in L$

$$a^{m!+k} = uv^2xy^2z \notin L$$

Contradiction!!!

We obtained a contradiction

Therefore: The original assumption that

$$L = \{a^{n!} : n \geq 0\}$$

is context-free must be wrong

Conclusion: L is not context-free

Non-context free languages

$\{a^n b^n c^n : n \geq 0\}$

$\{ww : w \in \{a,b\}\}$

$\{a^{n^2} b^n : n \geq 0\}$

$\{a^{n!} : n \geq 0\}$

Context-free languages

$\{a^n b^n : n \geq 0\}$

$\{ww^R : w \in \{a,b\}^*\}$

Theorem: The language

$$L = \{a^{n^2} b^n : n \geq 0\}$$

is **not** context free

Proof: Use the Pumping Lemma for context-free languages

$$L = \{a^{n^2} b^n : n \geq 0\}$$

Assume for contradiction that L

is context-free

Since L is context-free and infinite

we can apply the pumping lemma

$$L = \{a^{n^2} b^n : n \geq 0\}$$

Pumping Lemma gives a magic number m such that:

Pick any string of L with length at least m

we pick: $a^{m^2} b^m \in L$

$$L = \{a^{n^2} b^n : n \geq 0\}$$

We can write: $a^{m^2} b^m = uvxyz$

with lengths $|vxy| \leq m$ and $|vy| \geq 1$

Pumping Lemma says:

$uv^i xy^i z \in L$ for all $i \geq 0$

$$L = \{a^{n^2} b^n : n \geq 0\}$$

$$a^{m^2} b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

We examine **all** the possible locations

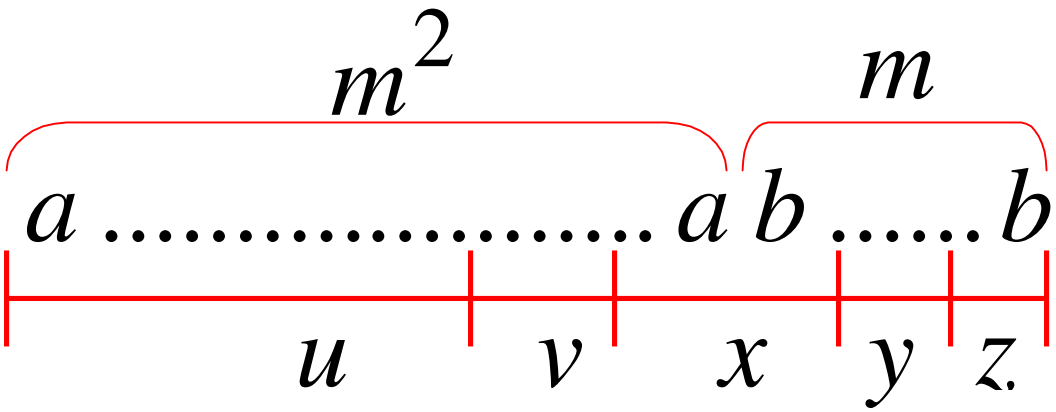
of string vxy in $a^{m^2} b^m$

$$L = \{a^{n^2} b^n : n \geq 0\}$$

$$a^{m^2} b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Most complicated case: v is in a^m

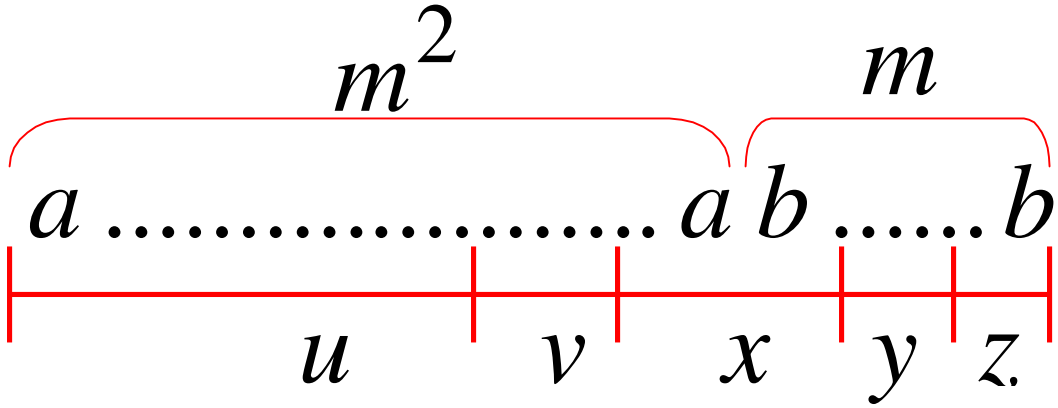
y is in b^m



$$L = \{a^{n^2} b^n : n \geq 0\}$$

$$a^{m^2} b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

$$v = a^{k_1} \qquad y = b^{k_2} \qquad 1 \leq k_1 + k_2 \leq m$$

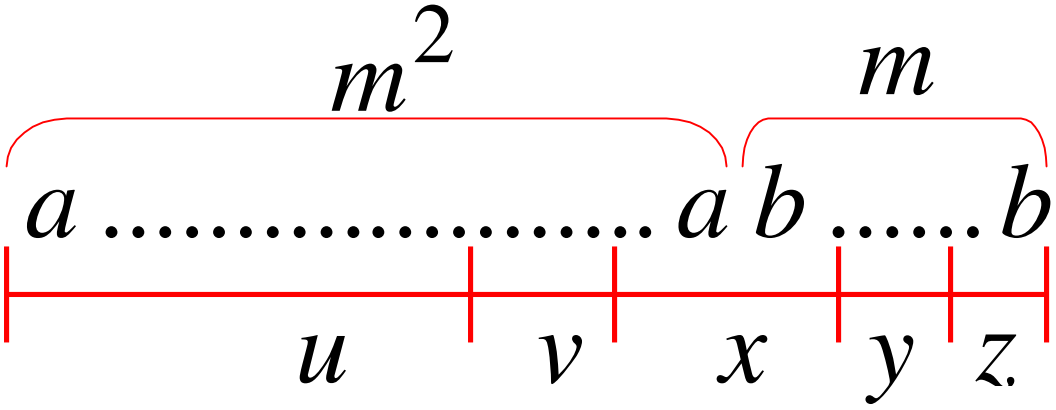


$$L = \{a^{n^2} b^n : n \geq 0\}$$

$$a^{m^2} b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Most complicated sub-case: $k_1 \neq 0$ and $k_2 \neq 0$

$$v = a^{k_1} \qquad y = b^{k_2} \qquad 1 \leq k_1 + k_2 \leq m$$

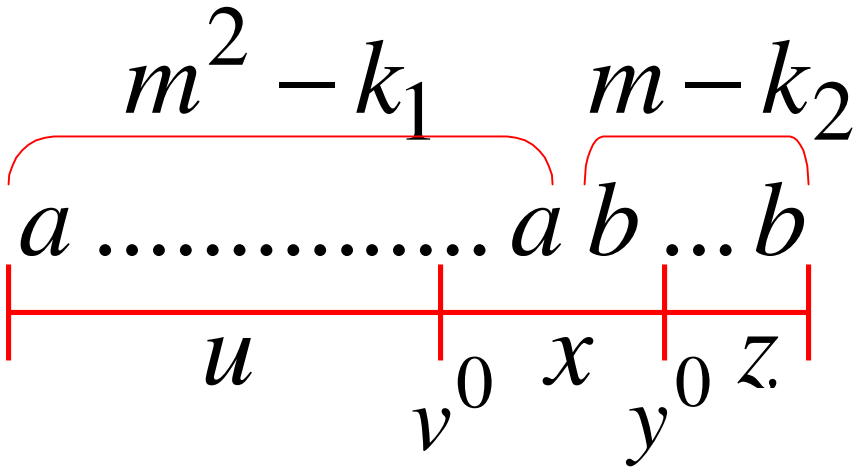


$$L = \{a^{n^2} b^n : n \geq 0\}$$

$$a^{m^2} b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Most complicated sub-case: $k_1 \neq 0$ and $k_2 \neq 0$

$$v = a^{k_1} \qquad y = b^{k_2} \qquad 1 \leq k_1 + k_2 \leq m$$



$$L = \{a^{n^2} b^n : n \geq 0\}$$

$$a^{m^2} b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Most complicated sub-case: $k_1 \neq 0$ and $k_2 \neq 0$

$$v = a^{k_1} \qquad y = b^{k_2} \qquad 1 \leq k_1 + k_2 \leq m$$

$$a^{m^2 - k_1} b^{m - k_2} = uv^0 xy^0 z$$

$k_1 \neq 0$ and $k_2 \neq 0$ $\qquad 1 \leq k_1 + k_2 \leq m$

$$(m-k_2)^2 \leq (m-1)^2$$

$$= m^2 - 2m + 1$$

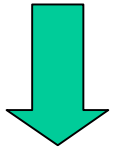
$$< m^2 - k_1$$

$$m^2 - k_1 \neq (m-k_2)^2$$

$$L = \{a^{n^2} b^n : n \geq 0\}$$

$$a^{m^2} b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

$$m^2 - k_1 \neq (m - k_2)^2$$



$$a^{m^2 - k_1} b^{m - k_2} = uv^0 xy^0 z \notin L$$

$$L = \{a^{n^2} b^n : n \geq 0\}$$

$$a^{m^2} b^m = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

However, from Pumping Lemma: $uv^0xy^0z \in L$

$$a^{m^2-k_1} b^{m-k_2} = uv^0xy^0z \notin L$$

Contradiction!!!

When we examine the rest of the cases we also obtain a contradiction

In all cases we obtained a contradiction

Therefore: The original assumption that

$$L = \{a^{n^2} b^n : n \geq 0\}$$

is context-free must be wrong

Conclusion: L is not context-free

YACC

Yet Another Compiler Compiler

Yacc is a parser generator

Input: A Grammar

Output: A parser for the grammar

Reminder: a parser finds derivations

Example grammar:

```
expr -> ( expr )
      | expr '+' expr
      | expr '-' expr
      | expr '*' expr
      | expr '/' expr
      | - expr
      | INT
      ;
```

The yacc code:

```
expr : '(' expr ')'
     | expr '+' expr
     | expr '-' expr
     | expr '*' expr
     | expr '/' expr
     | - expr
     | INT
     ;
```

Exampe Input:

10 * 3 + 4

Yacc Derivation:

expr => expr + expr => expr * expr + expr

=> 10*3 + 4

Resolving Ambiguities

```
%left '+', '-'
%left '*', '/'
%left UMINUS
%%

expr : '(' expr ')'
     | expr '+' expr
     | expr '-' expr
     | expr '*' expr
     | expr '/' expr
     | '-' expr          %prec UMINUS
     | INT
     ;
```

Actions

```
%left '+', '-'
%left '*', '/'
%left UMINUS
%%

expr : '(' expr ')'          {$$ = $2;}
     | expr '+' expr         {$$ = $1 + $3;}
     | expr '-' expr         {$$ = $1 - $3;}
     | expr '*' expr         {$$ = $1 * $3;}
     | expr '/' expr         {$$ = $1 / $3;}
     | '-' expr        %prec UMINUS    {$$ = -$2;}
     | INT                    {$$ = $1;}
     ;
```

A Complete Yacc program

```
%union{
  int int_val;
}

%left '+', '-'
%left '*', '/'
%left UMINUS

%token <int_val> INT
%type <int_val> expr

%start program

%%
```

```
program : expr        {printf("Expr value = %d \n", $1);}
        | error       {printf("YACC: syntax error near line %d \n", linenum);
                       abort();}
        ;

expr : '(' expr ')'     {$$ = $2;}
     | expr '+' expr    {$$ = $1 + $3;}
     | expr '-' expr    {$$ = $1 - $3;}
     | expr '*' expr    {$$ = $1 * $3;}
     | expr '/' expr    {$$ = $1 / $3;}
     | '-' expr         %prec UMINUS  {$$ = -$2;}
     | INT              {$$ = $1;}
     ;

%%

#include "lex.yy.c"
```

Execution Example

Input: 10 + 20*(3 - 4 + 25)

Output: Expr value = 490

The Lex Code

```
%{
int linenum=1;
int temp_int;
%}
%%

\n     {linenum++;}

[\t ]         /* skip spaces */;
\/\/[^\n]*    /* ignore comments */;

"+"     {return '+';}
"-"     {return '-';}
"*"     {return '*';}
"/"     {return '/';}
")"     {return ')';}
"("     {return '(';}
```

```
[0-9]+   {sscanf(yytext, "%d", &temp_int);
         yylval.int_val = temp_int;
         return INT;}

.  {printf("LEX: unknown input string found in line %d \n", linenum);
    abort();}
```

Compiling:

```
yacc YaccFile
lex LexFile
cc y.tab.c -ly -ll -o myparser
```

Executable: myparser

Another Yacc Program

```
%union{
  int int_val;
}

%left '+', '-'
%left '*', '/'
%left UMINUS

%token <int_val> INT
%type <int_val> expr

%start program

%%
```

```
program : stmt_list
        | error         {printf("YACC: syntax error near line %d \n", linenum);
                        abort();}
        ;

stmt_list : stmt_list stmt
          | stmt
          ;

stmt : expr ';'     {printf("Expr value = %d \n", $1);}
     ;
```

```
expr : '(' expr ')'      {$$ = $2;}
     | expr '+' expr     {$$ = $1 + $3;}
     | expr '-' expr     {$$ = $1 - $3;}
     | expr '*' expr     {$$ = $1 * $3;}
     | expr '/' expr     {$$ = $1 / $3;}
     | '-' expr          %prec UMINUS {$$ = -$2;}
     | INT               {$$ = $1;}
     ;

%%

#include "lex.yy.c"
```

Execution Example

Input:

```
10 + 20*(30 -67) / 4;

34 * 35 - 123 + -001;

17*8/6;
```

Output:

```
Expr value = -175
Expr value = 1066
Expr value = 22
```

Lex Code

```
%{
int linenum=1;
int temp_int;
%}
%%

\n     {linenum++;}

[\t ]        /* skip spaces */;
\/\/[^\n]*   /* ignore comments */;
```

```
"+"    {return '+';}
"-"    {return '-';}
"*"    {return '*';}
"/"    {return '/';}
")"    {return ')';}
"("    {return '(';}
";"    {return ';';}

[0-9]+   {sscanf(yytext, "%d", &temp_int);
          yylval.int_val = temp_int;
          return INT;}

.  {printf("LEX: unknown input string found in line %d \n", linenum);
    abort();}
```

Another Yacc Program

```
%union{
  int int_val;
  char *str_val;
}

%left '+', '-'
%left '*', '/'
%left UMINUS

%token PRINT
%token NEWLINE
%token <str_val> STRING
%token <int_val> INT
%type <int_val> expr

%start program
%%
```

```
program : stmt_list
        | error       {printf("YACC: syntax error near line %d \n", linenum);
                       abort();}
        ;

stmt_list : stmt_list stmt
          | stmt
          ;

stmt : expr ';'                  {printf("expression found\n");}
     | PRINT expr ';'            {printf("%d", $2);}
     | PRINT STRING ';'          {printf("%s", $2);}
     | PRINT NEWLINE ';'         {printf("\n");}
     ;
```

```
expr : '(' expr ')'     {$$ = $2;}
     | expr '+' expr    {$$ = $1 + $3;}
     | expr '-' expr    {$$ = $1 - $3;}
     | expr '*' expr    {$$ = $1 * $3;}
     | expr '/' expr    {$$ = $1 / $3;}
     | '-' expr         %prec UMINUS {$$ = -$2;}
     | INT              {$$ = $1;}
     ;

%%

#include "lex.yy.c"
```

Execution Example

Input:

```
print "The value of expression 123 * 25 is ";
print 123 * 25;
print newline;
10 + 5 * 8;
print "end of program";
print newline;
```

Output:

```
The value of expression 123 * 25 is 3075
expression found
end of program
```

Lex Code

```
%{
int linenum=1;
int temp_int;
char temp_str[200];
%}
%%

\n     {linenum++;}

[\t ]          /* skip spaces */;
\/\/[^\n]*     /* ignore comments */;
```

```
"+"         {return '+';}
"-"         {return '-';}
"*"         {return '*';}
"/"         {return '/';}
")"         {return ')';}
"("         {return '(';}
";"         {return ';';}
"print"     {return PRINT;}
"newline"   {return NEWLINE;}
```

```
[0-9]+   {sscanf(yytext, "%d", &temp_int);
          yylval.int_val = temp_int;
          return INT;}

\"[^"\n]*\" {strncpy(temp_str, &(yytext[1]), strlen(yytext)-2);
             temp_str[strlen(yytext)-2] = (char) 0;
              yylval.str_val = temp_str;
              return STRING;}

.  {printf("LEX: unknown input string found in line %d \n", linenum);
    abort();}
```