undefined

undefined

Example of Lex & Yacc

This is a simple example that demonstrate writting of a parser. In this example an input file describing a CRC card description in text format is taken as input and corresponding pseudo C++ code is generated. The input file is in a format:

```
  CRC
    CLASS <class name>
      RESPONSIBILITY
        <method name >  INT <attribute>
                        CHAR <attribute>
                        INT* <attribute>
                        CHAR* <attribute>
      END
        COLABRATION
           <RELATION>  CLASS    <class name>
        END
   END
```

e.g

```
     CRC
         CLASS  myclass
         RESPONSIBILITY
             assign  INT number
         END
         RESPONSIBILITY
             assign CHAR* name
         END
         COLABRATION
              USING  CLASS yourclass
         END
     END
```

The output of this translator is as follows:

```
   //  Class Name: myclass
   //  Function:
   //       assign int number
   //   Function:
   //       assign char *name
   //    Class Relations :
   //    This Class USES class yourclass
   //
  Class myclass  {
        public :
             assign(int number);
             assign(char *name);
         protected:
         private:
    };
```

Writting the Lex

```
Step 1. Identify all the Tokens.
        All the Keywords in the grammar forms the legal tokens.If one
        has the BNF then all the Terminals are the tokens generated by
```

the Lex.
Step 2. Write all the regular expressions describing the Tokens.
Step 3. Identify all lex substitutions (if possible) from the set of regular expressions.
Step 4. To avoid reduce/reduce errors identify possible start states and break conflicting tokens into multiple tokens.

According to the example,one can trace these steps
Step 1. Tokens are:
CLASS, CRC, END, RESPONSIBILITY, COLABRATION, INT, CHAR
*, USING, HAS_A, KIND_OF, variables,strings and spaces.
Step 2. R.E for keywords will be the same. R.E for strings and variable is [a-zA-Z][a-zA-Z0-9]*
Step 3. There are no possible lex substitutions.
Step 4. Since strings and variables represents the same R.E ,break them into multiple tokens and identify the start state.

The Start states are represented by

```
%s CL VAR METHOD
```

The White spaces and tabs are represented by

```
[ \t\n]*  { /* note there is no action taken */ }
```

Note:No action is taken here since whitespaces must be ignored,they are not the part of BNF. If any token is returned from this and not accounted in BNF then it will give yylook error during parsing.

When the keyword CLASS is identified ,the control is given to the State CL and then the string representing the <class name> is identified as token CLASSNAME. To start a state,lex has the Keyword BEGIN <state> and to return to the initial state, it is BEGIN INITIAL. The R.E to be matched on a perticular state is specified as <state> R.E { action }. Thus it will be <CL>[a-zA-Z][a-zA-Z0-9]* { /* action */}.
Note: The R.E's in a particular state are at the higher priority as compared to that in initial state. This means that if the token doesn't match any R.E in that state ,it will then try to match it in the initial state.

If one is planning to use yystack , then all the values from yytext must be copied to the stack. These can then be used by yacc with the help of pseudo variables. Thus to copy the class name we have

```
strcpy(yylval.stval,yytext);
```

where stval is of type char array defined in the yystack (%union) .yytext is of type char* which contains the value of the read token.

Given Below is the source code for the lex file.

---

crc.l

```
%{
  #include "dstruct.h"
  #include "y.tab.h"
  #include <string.h>
%}
%s CL VAR METHOD
%%
[ \t\n]*  {}
CLASS       {BEGIN CL;
             return CLASS; }
CRC         {

             return CRC; }
END         {return END; }
RESPONSIBILITY {BEGIN METHOD;return RESPONSIBILITY; }
COLABRATION    {return COLABRATION; }
INT            {BEGIN VAR;return INT; }
CHAR           {BEGIN VAR;return CHAR; }
"*"            {return PTR; }
USING          {return USING; }
HAS_A          {return HAS_A; }
KIND_OF        {return KIND_OF; }
<CL>[a-zA-Z][a-zA-Z0-9]*  {BEGIN INITIAL;
                strcpy(yylval.stval,yytext);
                return CLASSNAME; }
<VAR>[a-zA-Z][a-zA-Z0-9]* {BEGIN INITIAL;
                           strcpy(yylval.stval,yytext);
                           return VARIABLE; }
<METHOD>[a-zA-Z][a-zA-Z0-9]* {BEGIN INITIAL;
                   strcpy(yylval.stval,yytext);
                   return METHODNAME; }
[a-zA-Z][a-zA-Z0-9]*  { return STRING; }
%%
```

```
int yywrap(void)
{return 1;
}
```

Writting the Yacc

Step 1. Identify the Terminal and Non-Terminal Symbols from the BNF and Lex.

Step 2. Try coding all the grammar rules in yacc with empty actions Compile,link it to Lex and check for conflicts. This is an easy way of validating the BNF for reduce/reduce and shift/reduce conflicts.

Step 3. Search for any reduce/reduce conflict. Resolve it in Lex.

Step 4. Resolve any shift/reduce conflicts. Details on resolving it given later.

Step 5. Write rules for all possible syntax errors.Details on error handling are given later.

Step 6. Code the yyerror function in subroutine section.

Step 6. Design the Data Structure which can be easily integrated with the grammar rules for syntax directed translation.

Step 7. From the Data Structures and Lex needs ,formulate the correct Stack.The stack must have pointers for all the data structures.

Step 8. Do the appropiate type binding to all tokens and yacc variables (non-terminals).

Step 9. Write all the data structures in a seperate file and inlude it in yacc.

Step 10.Code all the actions.

Step 11.Restrict the actions in case of error, i.e no data structure should be built but parsing should continue to get more errors.

Eliminating shift/reduce errors

1. use -d switch of yacc to create debug file(y.output). This file will contain the full transition diagram description and the points at which any conflict arises.
2. Try assigning precedence and associativity to tokens and literals by using %left %right %noassoc %prec. Note that precedence level in the same line is same and down the line increases.
3. In majority of the cases shift/reduce conflict is always in the vicinity of left/right recursions. These might not be due to associativity or precedence relations.e.g consider the rules

```
s->XabY
a->E|aXAY   E=empty transition
b->E|bXBY
```

The syntax of these rules says that there is block XY which can have zero or more blocks of type A & B. These rules have shift/reduce conflict on the symbol X since in s->AabY for making a transition from literal a to b with input X it has no way to tell if it should reduce or shift another token . These rules can be rewritten as following

```
s->XaY
a->E|aXbY
b->A|B
```

Syntax checking and error recovery.It is one of the toughest part of parsing. There are many functions like yyerrok etc. However not all Yacc versions supports them. The simplest method is just to use the pseudo literal 'error'. In a rule whenever there is an error, yacc pushes a pseudo literal error and takes in next input. On identifying the rule it pops the stack and takes proper actions. Thus in this way the file pointer will always point to the right location and next rule can be looked for correctly. In our example code

```
error class {yyerror("Missing Relation"); }
```

says that if only class exists then error must be flaged. The literal error is pushed on the stack if relation is missing and then class is pushed. On reduction it calls yyerror with msg string.

Debugging The YACC

n rules never reduced

This means that the L.H.S or the yacc variable/non-termianl does not appear in any of the rules R.H.S. This rule might be redundant or is not used. Either remove it or check where it should be invoked from.

Illegal use of $n

The Yacc takes anything that is between {} as actions except pseudo-variable $n's. Thus if } is missing then yacc will take the next rule as an action till it encounters $n. At that point it will give the above error.Check for the missing }.

default action causes potential type clash.

In case of pseudo variables assignments $$=$1; if the 'C' type defined for $$ is different from $1, or in a rule types are defiend for the non-terminals but the actions does not have an assignment then this error is generated.
Check for all $n value's types and the types of L.H.S of the rule.

must specify type for X

This means that $X does not have a token type specified. i.e the token corresponding to $X does not have a type.
define the type of X %token < stack > X.

Given below is the source code for the Yacc file

---

crc.y

```
%{
  #include "dstruct.h"
  #ifndef debug
  #define debug 0
  #endif
  extern int  yylineno;
  extern char* yytext[];
  extern FILE* outFile_p;
  int noerror=1;
%}
%union{
 char stval[100];
 char* ptr;
 COLABR *colbr;
 ATTR *attr;
 RESPONS *resp;
 CARD *card;
 }
%token CLASS  CRC END
%token <stval> CLASSNAME
%token RESPONSIBILITY
%token COLABRATION STRING
%token INT CHAR PTR
%token <stval> VARIABLE
%token <stval> METHODNAME
%token USING HAS_A KIND_OF
%type <ptr> className
%type <ptr> class
%type <ptr> methodName
%type <colbr> colabrationType colabration colabrations

%type <attr> attributes attribute
%type <resp> responsibility responsibilities
%type <card> crc cards
%start data
%%
data:cards  {
              if(noerror)
                generatePseudoCode($1,outFile_p);
              if(debug)
                displayDataStruct($1);
              printf("Complete\n");
            }
cards:crc {
            if(noerror)
             { $$=$1;
               if(debug)
                printf("Assigned Ist Card\n");
             }
          }
     |cards crc {
                 if(noerror)
                  {
                   int no=2;
                   CARD *temp=$$;
                   while(temp->next)
                    { temp=temp->next;
                      no++;
                    }
                   temp->next=$2;
                   if(debug)
                    printf("Assigned %dth Card\n",no);
                  }
                }
     |error {
             yyerror("error in input file");
            }
crc:CRC class responsibilities colabrations END  {
          if(noerror)
           {
            CARD* crc=getNewCard();
            crc->className=$2;
            crc->responsibility=$3;
            crc->colabration=$4;
            $$=crc;
           }
          }
class:CLASS className {
                       if(noerror)
                        {
                         $$=$2;
                         if(debug)
                          printf("Assigned class name:%s\n",$2);
                        }
                      }
     |CLASS error  {
                     yyerror("Class Name not specified");
                   }
className:CLASSNAME  {
                      if(noerror)
                       {
                        char *name=(char*)malloc(strlen($1)*sizeof(char));
                        strcpy(name,$1);
                        $$=name;
                       }
                     }
responsibilities:responsibility  {
                if(noerror)
                 {
                  $$=$1;
                  if(debug)
                   printf("Assigned Ist Responsibility:%s\n",$1->methodName);
                 }
                }
                |responsibilities responsibility {
                   if(noerror)
                    {
                      int no=2;
                      RESPONS *temp=$$;
                      while(temp->next)
                        {temp=temp->next;
                          no++;
                        }
                      temp->next=$2;
                      if(debug)
                       printf("Assigned %dth Responsibility:%s\n",no,$2->methodName);
                    }
                  }
                |error {
                        yyerror("error in responsibility");
                       }
responsibility: RESPONSIBILITY methodName attributes END
      {

                  if(noerror)
                   {
                     RESPONS *res=getNewResp();
                     res->methodName=$2;
                     res->attribute=$3;
                     $$=res;
                   }
                 }
              | RESPONSIBILITY error attributes END  {
                                         yyerror("Method name not specified");
                                        }
methodName: METHODNAME {
                        if(noerror)
                        {
                         char *name=(char*)malloc(strlen($1)*sizeof(char));
                         strcpy(name,$1);
                         $$=name;
                        }
                       }
```

```
attributes: {
              $$=NULL;

            }
          |attributes attribute {
                                 if(noerror)
                                  {
                                   if($$)
                                    {
                                     int no=2;
                                     ATTR *temp=$$;
                                     while(temp->next)
                                      {temp=temp->next;
                                       no++;
                                      }
                                     temp->next=$2;
                                     if(debug)
                                      printf("Assigned %dth Attribute:%s\n",no,

                                                                   $2->attribute);

                                    }
                                   else
                                   {
                                    $$=$2;
                                    if(debug)
                                     printf("Assigned Ist Attribute:%s\n",

                                                             $2->attribute);

                                   }
                                 }
                               }
attribute :INT VARIABLE {
                         if(noerror)
                          {
                           ATTR* attr=getNewAttr();
                           strcpy(attr->attribute,"int ");

                           strcat(attr->attribute,$2);
                           $$=attr;

                          }
                        }
          |CHAR VARIABLE {
                          if(noerror)
                           {
                            ATTR* attr=getNewAttr();
                            strcpy(attr->attribute,"char ");
                            $$=attr;
                            strcat(attr->attribute,$2);

                           }
                          }
          |INT PTR VARIABLE {
                             if(noerror)
                              {
                               ATTR* attr=getNewAttr();
                               strcpy(attr->attribute,"int *");
                               strcat(attr->attribute,$3);

                               $$=attr;

                              }
                            }
          |CHAR PTR VARIABLE {

                              if(noerror)
                               {
                                ATTR* attr=getNewAttr();
                                strcpy(attr->attribute,"char *");
                                strcat(attr->attribute,$3);

                                $$=attr;

                               }
                             }
          | error STRING {yyerror("Missing Type for Variable"); }
          | error PTR STRING
{ yyerror("Missing Type for Variable"); }

          | INT error { yyerror("Missing Variable"); }
          | CHAR error { yyerror("Missing Variable"); }
          | INT PTR error { yyerror("Missing Variable"); }
          | CHAR PTR error { yyerror("Missing Variable"); }

colabrations: {
                if(noerror)
                  $$=NULL;

              }
            |colabrations colabration {
                 if(noerror)
                  {
                   if($$)
                    {
                      int no=2;
                      COLABR *temp=$$;
                      while(temp->next)
                        { temp=temp->next;
                          no++;
                        }
                      temp->next=$2;
                      if(debug)
                       printf("Assigned %dth Colabration\n",no);

                    }
                   else
                     {
                      $$=$2;
                      if(debug)
                        printf("Assigned Ist Colabration\n");

                     }
                  }
                 }
colabration:COLABRATION colabrationType END {
                               if(noerror)
                                  $$=$2 ;
                             }
           |COLABRATION error END {
                     yyerror("Colabration defined but is empty");

                    }
colabrationType:USING class {
                             if(noerror)
                              {
                               COLABR* col=getNewColbr();
                               col->relation=USING_R;
                               col->className=$2;
                               $$=col;
                              }
                            }
               |HAS_A class {
                             if(noerror)
                              {
                               COLABR* col=getNewColbr();
                               col->relation=HAS_A_R;
                               col->className=$2;
                               $$=col;
                              }
                            }
               |KIND_OF class {
                               if(noerror)
                                {
                                 COLABR* col=getNewColbr();
                                 col->relation=KIND_OF_R;

                                 col->className=$2;
                                 $$=col;

                                }
                              }
               |error class {yyerror("Missing Relation"); }
               |USING error {yyerror("Missing Class"); }
               |HAS_A error {yyerror("Missing Class"); }
               |KIND_OF error {yyerror("Missing Class"); }
%%
#include<stdio.h>
#include <iostream.h>
#include <string.h>
extern void yyerror(char* msg)
{
 noerror=0;
 if(strcmp(msg,"syntax error"))
  printf(" Syntax Error in Line : %d : %s\n",yylineno,msg);
}
```

Data Structures Header File

dstruct.h

```
#ifndef DSTRUCT_H
#define DSTRUCT_H

#include <stdio.h>
```

```
#include <string.h>
#include <malloc.h>
enum RELATION {NONE, USING_R,HAS_A_R, KIND_OF_R };
enum BOOL {FALSE,TRUE };

typedef struct COLABR {
                 char* className;
                 RELATION relation;
                 COLABR *next;
               }COLABR ;

typedef struct  ATTR {
               char *attribute;
               ATTR *next;
             } ATTR;

typedef struct RESPONS{
               char *methodName;
               ATTR *attribute;
               RESPONS *next;
               }RESPONS;

typedef struct CARD {
               char *className;
               RESPONS *responsibility;
               COLABR  *colabration;
               CARD *next;
               } CARD;

COLABR *getNewColbr();
RESPONS *getNewResp();
CARD *getNewCard();
ATTR *getNewAttr();
void generatePseudoCode(CARD *cardList,FILE* fp);
void displayDataStruct(CARD *cardList);
#endif
```

---

dstruct.c

```
#include "dstruct.h"

COLABR *getNewColbr()
{
  COLABR *col=(COLABR*)malloc(sizeof(COLABR));
  col->relation=NONE;
  col->next=NULL;
  return col;
}

ATTR *getNewAttr()
{
  ATTR* attr=(ATTR*)malloc(sizeof(ATTR));
  attr->attribute=(char*)malloc(50*sizeof(char));
  strcpy(attr->attribute,"");
  attr->next=NULL;
  return attr;
}

RESPONS *getNewResp()
{
  RESPONS *res=(RESPONS*)malloc(sizeof(RESPONS));
  res->attribute=NULL;
  res->next=NULL;
  return res;
}

CARD *getNewCard()
{
  CARD *crc=(CARD*)malloc(sizeof(CARD));
  crc->responsibility=NULL;
  crc->colabration=NULL;
  crc->next=NULL;
  return crc;
}

void displayDataStruct(CARD *cardList)
{
 /* Display DS for Debugging */
 while(cardList)
 {
   printf(" CLASS %s\n",cardList->className);
   RESPONS *tempResp=cardList->responsibility;
   while(tempResp)
   {
   printf(" RESPONSIBILITY\n");
    printf(" Method %s\n",tempResp->methodName);
    printf(" Attributes\n");
    ATTR *tempAttr=tempResp->attribute;
    while(tempAttr)
     {printf("   %s\n",tempAttr->attribute);
      tempAttr=tempAttr->next;
     }
    tempResp=tempResp->next;
  }
  COLABR *tempColbr=cardList->colabration;
  if(tempColbr)
    printf("  COLABRATORS\n");
  while(tempColbr)
    {
     switch(tempColbr->relation)
      {
         case NONE: printf("This Class has no relation specified with ");

                    break;
         case USING_R: printf("This Class USES class ");
                    break;
         case HAS_A_R:printf("This Class HAS A class ");
                    break;
         case KIND_OF_R: printf("This Class is a KIND OF class ");

                    break;
         default: break;

       }
     printf(" %s\n",tempColbr->className);
     tempColbr=tempColbr->next;
    }
   cardList=cardList->next;
 }
}

void generatePseudoCode(CARD *cardList,FILE *outFile_p)
{
 while(cardList)
 {
   fprintf(outFile_p,"// Class Name: %s\n",cardList->className);

   RESPONS *tempResp=cardList->responsibility;

   while(tempResp)

   {
    fprintf(outFile_p,"// Functions:\n");
    fprintf(outFile_p,"//   %s",tempResp->methodName);
    ATTR *tempAttr=tempResp->attribute;
    while(tempAttr)
     {
      fprintf(outFile_p," %s",tempAttr->attribute);
      tempAttr=tempAttr->next;
     }
     fprintf(outFile_p,"\n");
    tempResp=tempResp->next;
  }
  COLABR *tempColbr=cardList->colabration;
  fprintf(outFile_p,"// Class Relations :\n");
  while(tempColbr)
    {
     switch(tempColbr->relation)
      {
         case NONE: fprintf(outFile_p,"// This Class has no relation specified\n ");

                    break;
         case USING_R: fprintf(outFile_p,"// This Class USES class %s\n
                                                 ",tempColbr->className);

                    break;
         case HAS_A_R:fprintf(outFile_p,"// This Class HAS A class
                                        %s\n",tempColbr->className);
                      break;
         case KIND_OF_R: fprintf(outFile_p,"// This Class is a KIND OF class
                                                  %s\n",tempColbr->className);

                          break;
         default: break;

       }
     tempColbr=tempColbr->next;
    }

   fprintf(outFile_p,"//\n");
   fprintf(outFile_p,"Class %s { \n",cardList->className);

   fprintf(outFile_p,"    Public:\n");
   tempResp=cardList->responsibility;

   while(tempResp)
    {
      fprintf(outFile_p,"      %s(",tempResp->methodName);
      ATTR *tempAttr=tempResp->attribute;
      while(tempAttr)
       {
        if(tempAttr->next)
         fprintf(outFile_p,"%s,",tempAttr->attribute);
        else
         fprintf(outFile_p,"%s",tempAttr->attribute);
        tempAttr=tempAttr->next;
       }
      fprintf(outFile_p,");\n");
    tempResp=tempResp->next;
    }
    fprintf(outFile_p,"    Protected:\n");
    fprintf(outFile_p,"    Private:\n");
    fprintf(outFile_p,"};\n");

   cardList=cardList->next;
 }

}
```

main.c

```
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
extern int  yyparse();
extern FILE *yyin;
FILE *outFile_p;
main(int argc,char *argv[])
{
 if(argc<3)
  {
   printf("Please specify the input file & output file\n");
   exit(0);
  }
 FILE *fp=fopen(argv[1],"r");
 if(!fp)
 {
  printf("couldn't open file for reading\n");
  exit(0);
 }
  outFile_p=fopen(argv[2],"w");
  if(!outFile_p)
  {
   printf("couldn't open temp for writting\n");
   exit(0);
  }
  yyin=fp;
  yyparse();
 fclose(fp);
 fclose(outFile_p);
}
```

The Makefile

The Above code was compiled on Sun-sparc using C++ compiler. If one wants to use C compiler then many changes will have to be done in the code.

makefile

```
C++=/sun/pollux/home1/home/lang/CC-4.0.1 -Dsun4  -g
crc: main.o crcy.o crcl.o dstruct.o
        $(C++) -DEXTERNC -I/. crcl.o  crcy.o dstruct.o main.o  -o crc
crcl.o: ../lex/crc.l
        /usr/lang/lex++ ../lex/crc.l
        mv lex.yy.c crcl.cxx
        $(C++). -c crcl.cxx -o crcl.o
crcy.o: crc.y
        /usr/lang/yacc++ -dvt crc.y
        mv y.tab.c crcy.cxx
        $(C++) -c crcy.cxx -o crcy.o
main.o: main.cxx
        $(C++) -c main.cxx -o main.o
dstruct.o:dstruct.c
        $(C++) -c dstruct.c -o dstruct.o
```