CHƯƠNG TRÌNH DỊCH

Bài 5: Tự động sinh bộ PTTV

Nội dung

1. Bài tập và giải bài tập của các buổi trước
2. Giới thiệu về LEX
3. CsLex – phiên bản LEX cho C#

Phần 1

Bài tập và giải bài tập của các buổi trước

Hoàn thiện mã nguồn PTTV

Tham khảo slide buổi trước (từ slide 6 đến slide 12)

1. Bổ sung khả năng nhận ra cặp ngoặc:
   - Bổ sung hàm nextIsParethesis vào class PTTV: tương tự như hàm nextIsOperator nhưng thay thế phép toán bằng ngoặc
   - Thêm lời gọi hàm nextIsParethesis vào getNextWord
2. Bổ sung khả năng nhận ra các từ loại khác:
   - Nhận ra số nguyên
   - Nhận ra số thực
   - Nhận ra tên các hàm toán học (sqrt, log, exp, power)
   - Nhận ra tên biến
   - Lưu thông tin về vị trí của từ loại trong mã nguồn

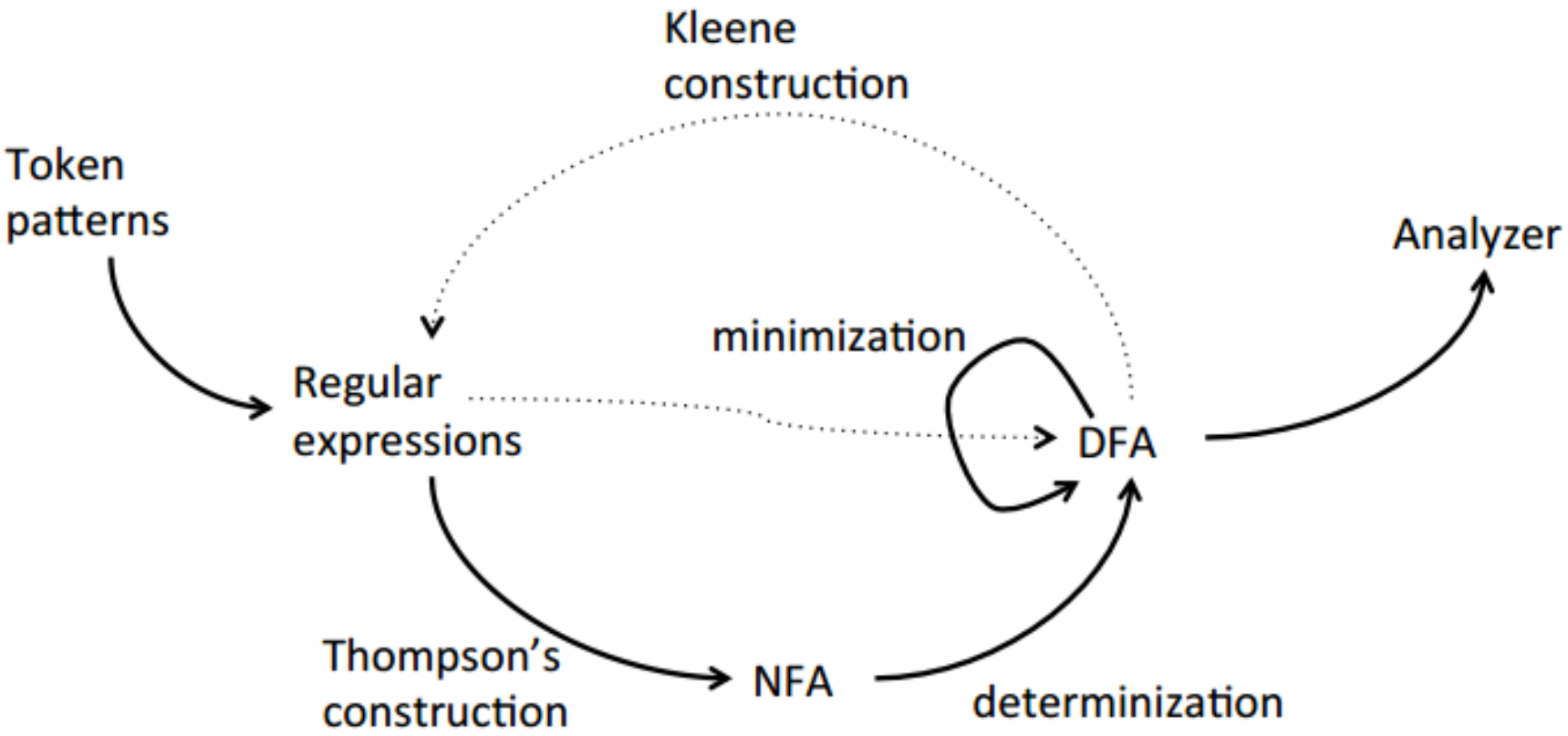
Hoàn thiện mã nguồn PTTV

- Có thể tham khảo mã nguồn đầy đủ hơn ở link: http://txnam.net/courseware/slide-to4-dung-dfa-de-phan-tich-tu-vung
- Bộ phân tích từ vựng đơn giản dễ hiểu nhưng không hiệu quả về mặt tốc độ, hãy sử dụng các kiến thức về DFA được học và chuyển đổi bộ phân tích trên thành bộ phân tích dựa trên DFA
  - Viết một DFA đơn giản bằng C# để nhận biết đồng thời các từ loại: số nguyên, số thực, tên biến
  - Chú ý: không nhất thiết phải sử dụng bảng chuyển trong việc triển khai DFA

Phần 2

Giới thiệu về LEX

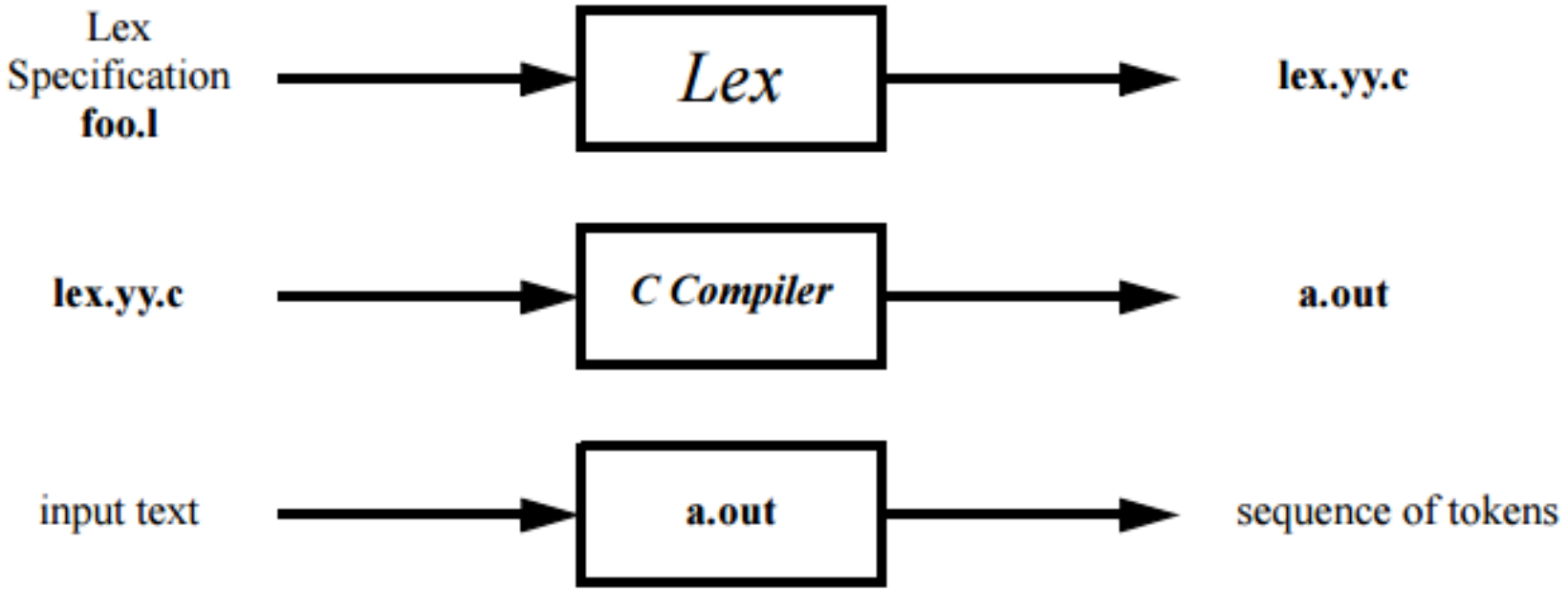
Từ patterns đến scanner



Các bước để tạo một bộ PTTV

- Các bước để tạo một bộ PTTV:
  1. Định nghĩa từ loại ở dạng các RE
  2. Chuyển các RE thành một NFA duy nhất
  3. Chuyển NFA thành DFA
  4. Tối ưu hóa DFA
  5. Viết mã xử lý DFA
  6. Xử lý các tình huống nhập nhằng hoặc đặc biệt
- Tự viết bộ PTTV (ad-hoc analyser): tự làm tất cả các bước trên
- Sử dụng LEX: làm bước 1 và bước 6, LEX làm các bước còn lại

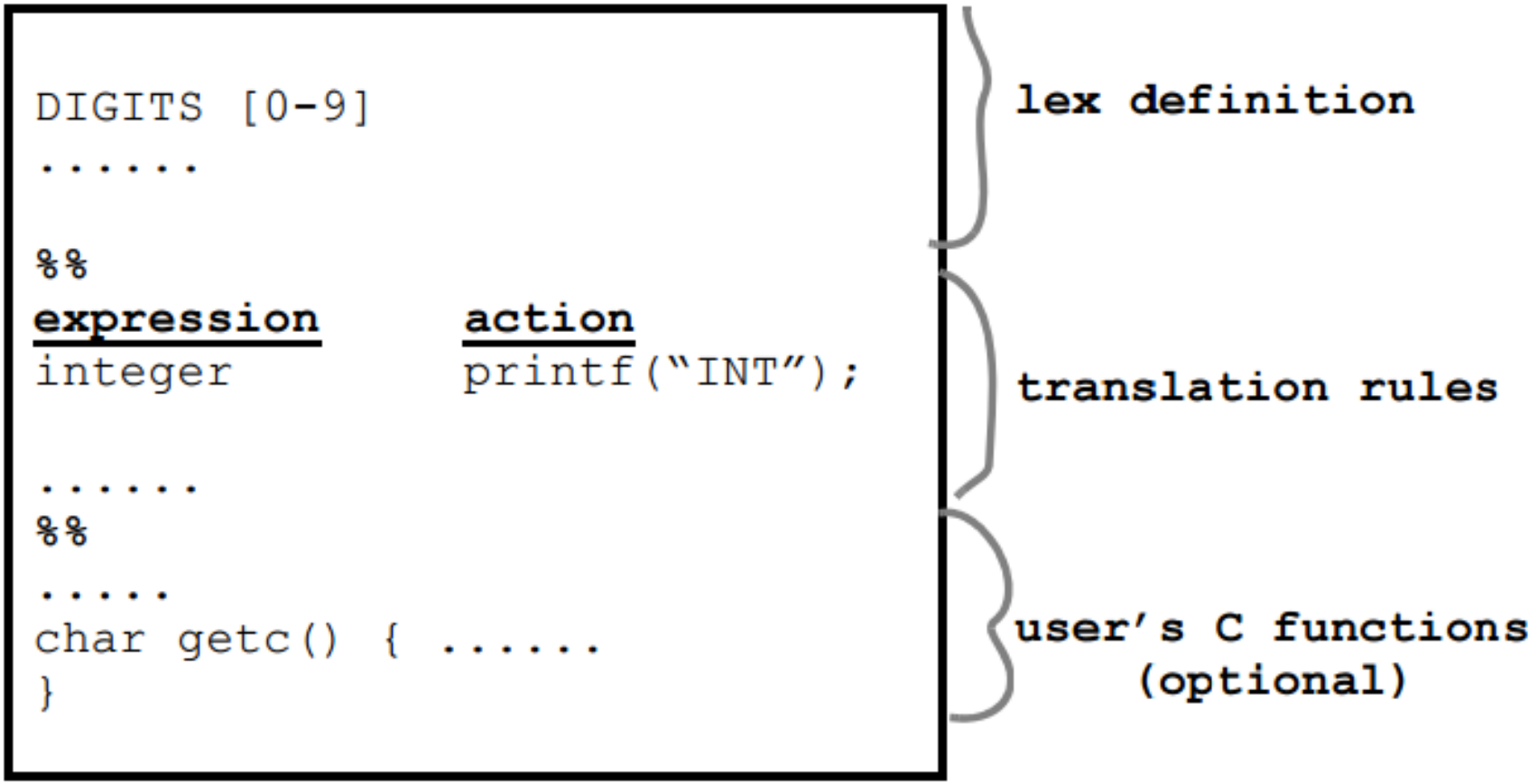
LEX: cách làm việc



Implementation of Lex:

Lex Spec -> NFA -> DFA -> Transition Tables + Actions -> yylex()

LEX: đầu vào



expression *is a regular expression ;* ***action*** *is a piece of C program;*

Ví dụ về LEX

- Cho ngôn ngữ X có văn phạm như sau:

```
stmt → if expr then stmt else stmt
     | if expr then stmt | ε
expr → term relop term | term
term → id | num
```

- Các từ loại:

```
if    → if
then  → then
else  → else
relop → < | <= | = | <> | > | >=
id    → letter (letter | digit)*
num   → digit+ (. digit+)? (E (+ | -)? digit+)?
delim → blank | tab | newline
ws    → delim+
```

Ví dụ về LEX

```
%{
/* đoạn mã định nghĩa các hằng: LT, LE, EQ, NE, GT, GE, IF, THEN,
ELSE, ID, NUMBER, RELOP */
#define LT 0
#define LE 1
…
}%
/* định nghĩa chính quy */
delim    [\t\n]
ws       {delim}+
letter   [A - Za -  z]
digit    [0 - 9]
id       {letter}({letter}| {digit})*
number   {digit}+(\.{digit}+)?(E[+\-]?{digit}+)?
```

Ví dụ về LEX

```
%%
/* xử lý khi gặp các ký hiệu */
{ws}        {/* Không có action, không có return */}
if          {return(IF); }
then        {return(THEN); }
else        {return(ELSE); }
{id}        {yylval = install_id( ); return(ID) }
{number}    {yylval = install_num( ); return(NUMBER) }
"<"         {yylval = LT; return(RELOP) }
"<="        {yylval = LE; return(RELOP) }
"="         {yylval = EQ; return(RELOP) }
"<>"        {yylval = NE; return(RELOP) }
">"         {yylval = GT; return(RELOP) }
">="        {yylval = GE; return(RELOP) }
```

Ví dụ về LEX

```
%%
/* các thủ tục bổ sung, viết bằng ngôn ngữ C */

/* Thủ tục phụ cài id vào trong bảng ký hiệu */
install_id ( ) {
    …
}

/* Thủ tục phụ cài một số vào trong bảng ký hiệu */
install_num ( ) {
   …
}
```

Phần 3

CsLex – phiên bản LEX cho C#

CsLex

- Cũng tương tự như Lex, nhưng sử dụng ngôn ngữ lập trình C#
- Trên GitHub: https://github.com/zbrad/CsLex
- Cấu trúc file input tương tự như Lex, nhưng có điều chỉnh cho phù hợp với C#

```
user code
%%
CsLex directives
%%
regular expression rules
```

Bài tập

- Viết file a.lex theo chuẩn của CsLex mô tả các luật từ vựng của ngôn ngữ A
- Sử dùng CsLex dịch a.lex sang file .cs tương ứng
- Viết các hàm (trong a.lex) để in ra các từ tố đoán nhận được trong quá trình phân tích
- Nộp bài cho thầy giáo vào email: namtx@wru.vn
- Deadline: 15/12/2017