Parsing

- Flex
  - 스트링을 입력 받아, 정규식(regular expression)으로 정의된 스트링 패턴에 따라 토큰을 생성함 (즉, 스트링을 토큰으로 변형함)
  - 입력 스트링이 정의된 패턴에 적합하지 않을 경우, 에러가 발생되며 중단됨

- Bison
  - 토큰을 입력 받아 문맥자유문법(context free grammar)으로 정의된 스트링 패턴에 따라 입력이 문법에 맞는 지를 점검한다.
  - 입력 토큰이 정의된 패턴에 적합하지 않을 경우, 에러가 발생되며 중단됨.
  - 문법에 맞는 경우, 목적에 따라 AST로 출력되거나, 직접 계산하여 결과 값을 출력함

예: 산술식

LEX

```
%%
[0-9]+  {yylval = atoi(yytext);
         return NUM;}
[ \t]   ;
.       return yytext[0];
%%
```

YACC

```
%%
exp: NUM                {$$=$1;}
     | exp '+' NUM      {$$=$1+$3;}
     | exp '-' NUM      {$$=$1-$3;}
;
%%
```

입력: $1 + 23 - 456$

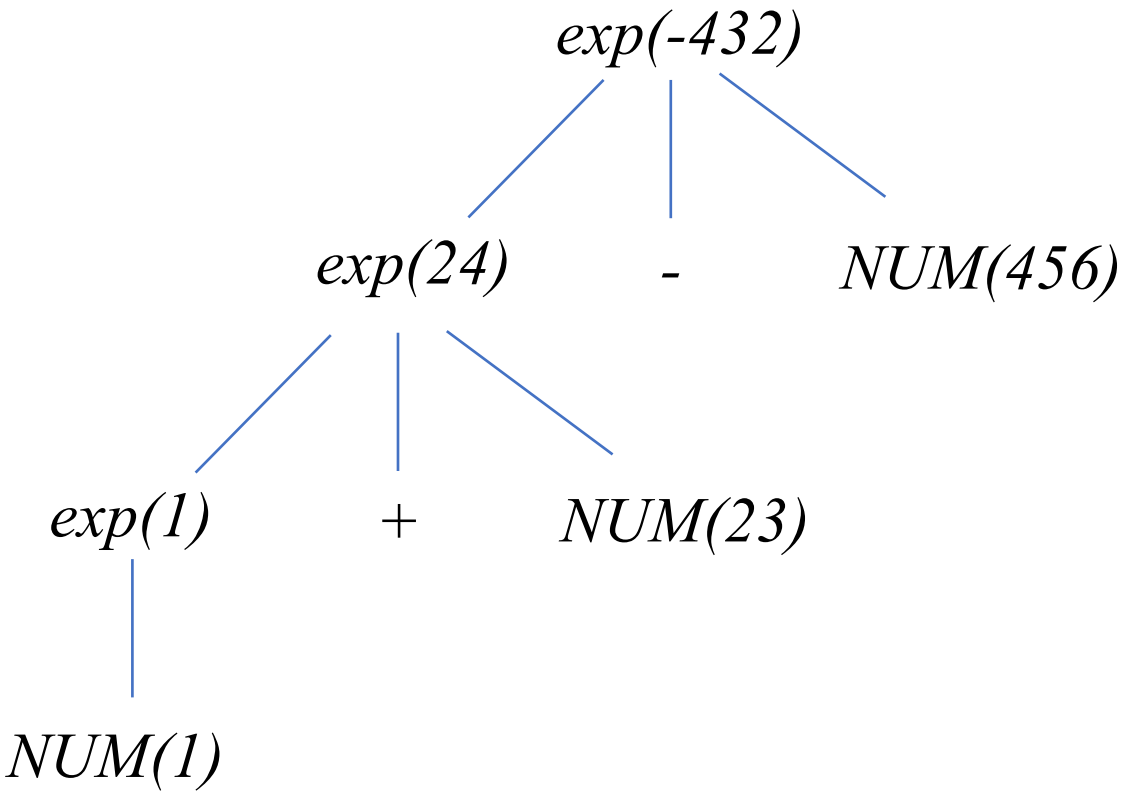
Flex 출력: $\mathrm{NUM}_1 + \mathrm{NUM}_{23} - \mathrm{NUM}_{456}$

- *exp*에 대한 Context-Free Grammar (이론적 표현) :
  - 이론은 오직 문법의 타당성만을 다루며, action에 대해서는 논의하지 않음.

$$exp \rightarrow NUM \mid exp + NUM \mid exp - NUM$$

$$exp \Rightarrow exp - NUM \Rightarrow exp + NUM - NUM \Rightarrow NUM_1 + NUM_{23} - NUM_{456}$$

Parse Tree로 표현되는 Action 수행 결과



Flex의 응용 (정규식의 응용)

- Flex의 스트링 패턴은 정규식(regular expression)으로 표현됨
- Flex는 정규식으로 정의된 패턴에 대한 Action을 C로 정의함.
- 정규식으로 프로그래밍 언어의 토큰을 정의할 수 있음.
- 그러나, 토큰 외에 다른 표현을 정의할 수 있으며, 목적에 따라 Flex는 언어의 Parsing 외에 다른 분야에 적용될 수 있음.
- Flex는 bison 없이, 독자적으로 사용될 수 있음

- 예: email 추출기, word counter, …. 등

Bison의 응용 (Context-Free Grammar의 응용)

- Bison의 스트링 패턴은 Context-Free Grammar로 표현됨
- Bison은 Context-Free Grammar로 정의된 패턴에 대한 Action을 C로 정의함.
- Context-Free Grammar로 프로그래밍 언어의 문법을 정의할 수도 있음
- 그러나, Context-Free Grammar는 프로그래밍 언어 문법 외에, 목적에 따라 다른 분야에도 적용될 수 있음.

- 예:
  - Jason parser, HTML Parser, Markup 언어 parser 등등

- 문제: Regular expression과 Context-Free Grammar의 표현력은 어떻게 다른가? 그에 따라 Flex와 Bison의 사용을 선택하게 됨.