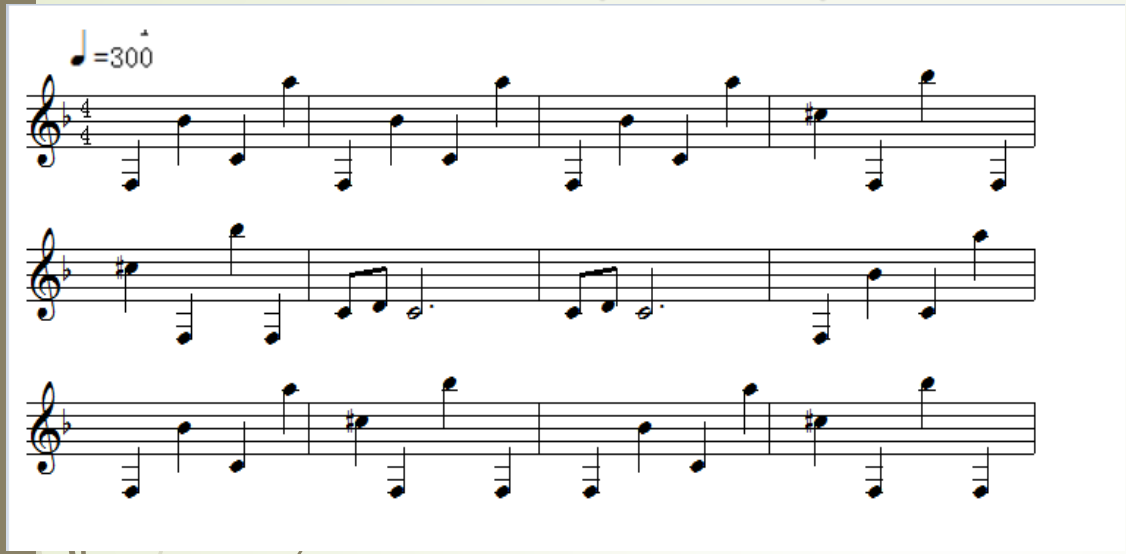
Rock Your Composition

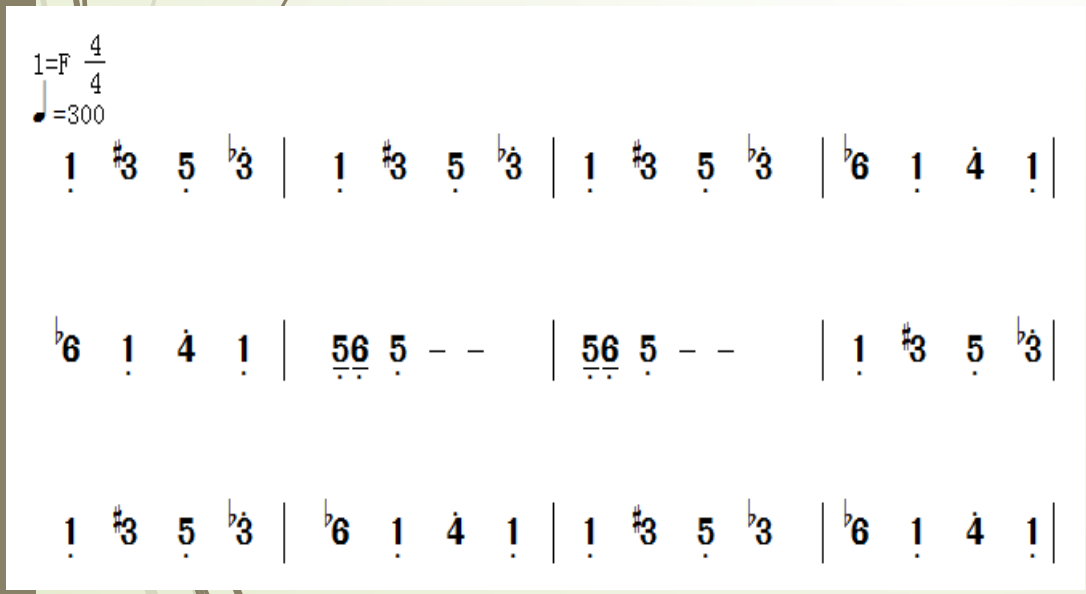
PRESENTERS

Project Manager	Xiaoji Li(xl2336)
Language Guru	Junyang Lu(jl3937)
System Architector	Ruizhi Liao(rl2643)
System Integrator	Dong Li(dl2768)
Tester	Zhuojun Huang(zh2193)

INTRODUCTION

RYC: rock your composition





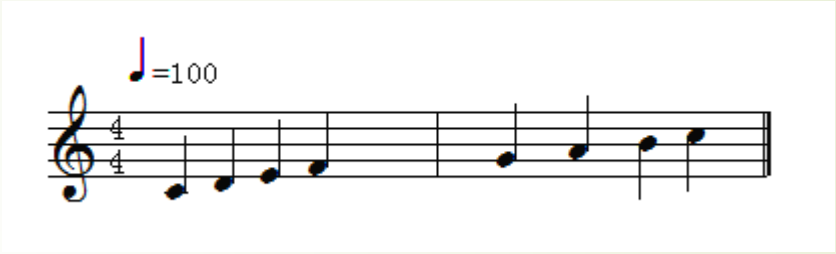

```
tempo = 300;
key = #3;
song = m2 a b c 1;
a = [_1 #3 _5 ^3 |];
b = [&6 _1 ^4 _1 |];
c = [(mordent _5 4)   |];
e = [(1, 2) (chord  2) |];
m2 a b c d = [
a a a b
b c c a
a b a b
b c d 1 (1, / 1 2) (1, / 1 2) 1 1 |
a e a a
b a a e 11
a e a a
b a e (_1, 2) (1, 2) |
];
chord p d = {
(p, d)
(+ p 2, d)
(+ p 4, d)
(+ p 7, d) } ;
mordent p d = [ (p, / d 8)
 (+ p 1, / d 8) (p, * 3 (/ d 4)) ] ;
```

INTRODUCTION

RYC: rock your composition

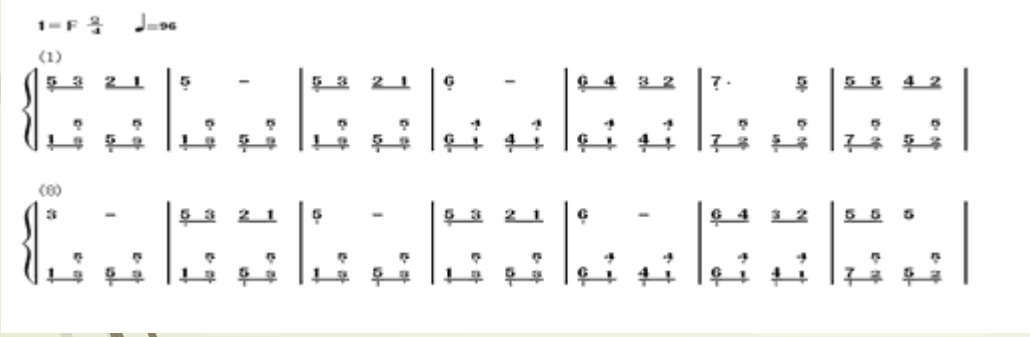
There are two ways of musical notation: numerical and staff

The staff notation is rather complicated, especially for one without professional musical background




```
key = #2;

song = [ 1 2 3 4 | 5 6 7 ^1 ];
```

We present RYC, the language for computer scientists and music lovers to play with music.



With RYC, one can write a piece of melody, do synchronization(parallelism). Since it's purely functional, you can even do lambda calculus.

INTRODUCTION

RYC: rock your composition

MUSICAL NOTATION CONVENTION:

0 : REST ^ : OTTAVA ALTA

1-7: NOTE _ : OTTAVA SOTTO

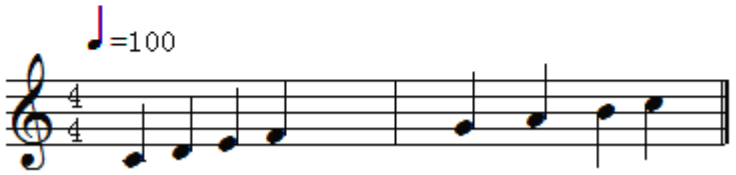
(6,8) : a note with a duration in the unit of basic length of the time signature

: SHARP & : FLAT

key = #2;

song = [1 2 3 4 | 5 6 7 ^1];

The corresponding staff notation is:



Main Features

RYC: rock your composition

```
tempo = 300;
key = 5;

song = repeat 100 m;
```

song: main function
tempo: optional function
key: optional function

```
repeat times melody =
if (<= times 0)
   []
   (seq melody (repeat (- times 1) melody));

m = { [ #1 0 3 1] ^[ (_5, 4)]} ;
```

Main Features

RYC: rock your composition

```
tempo = 300;
key = 5;

song = repeat 100 m;

repeat times melody =
if (<= times 0)
   []
   (seq melody (repeat (- times 1) melod

m = { [ #1 0 3 1] ^[ (_5, 4)]} ;
```

1 – 7: do - ti
0: rest
#: half tone higher
&: half tone lower
^: an octave higher
_: an octave lower

Main Features

RYC: rock your composition

```
tempo = 300;
key = 5;

song = repeat 100 m;

repeat times melody =
if (<= times 0)
   []
   (seq melody (repeat (- times 1) melody));

m = { [ #1 0 3 1] ^[ (_5, 4)]} ;
```

Note with default duration

Note with customized duration

Main Features

RYC: rock your composition

```
tempo = 300;
key = 5;

song = repeat 100 m;

repeat times melody =
if (<= times 0)
    []
    (seq melody (repeat (-
```

```
m = { [ #1 0 3 1] ^[ (_5, 4)]} ;
```

[]: Sequential melody

Main Features

RYC: rock your composition

```
tempo = 300;
key = 5;

song = repeat 100 m;

repeat times melody =
if (<= times 0)
    []
    (seq melody (repeat (- times 1)  melo

m = { [ #1 0 3 1] ^[ (_5, 4)]} ;
```

{}: Parallel melody

Main Features

RYC: rock your composition

```
tempo = 300;
key = 5;

song = repeat 100 m;

repeat times melody =
if (<= times 0)
   []
   (seq melody (repeat (- times 1) melody));

m = { [ #1 0 3 1] ^[ (_5, 4)]} ;
```

User-defined function

Main Features

RYC: rock your composition

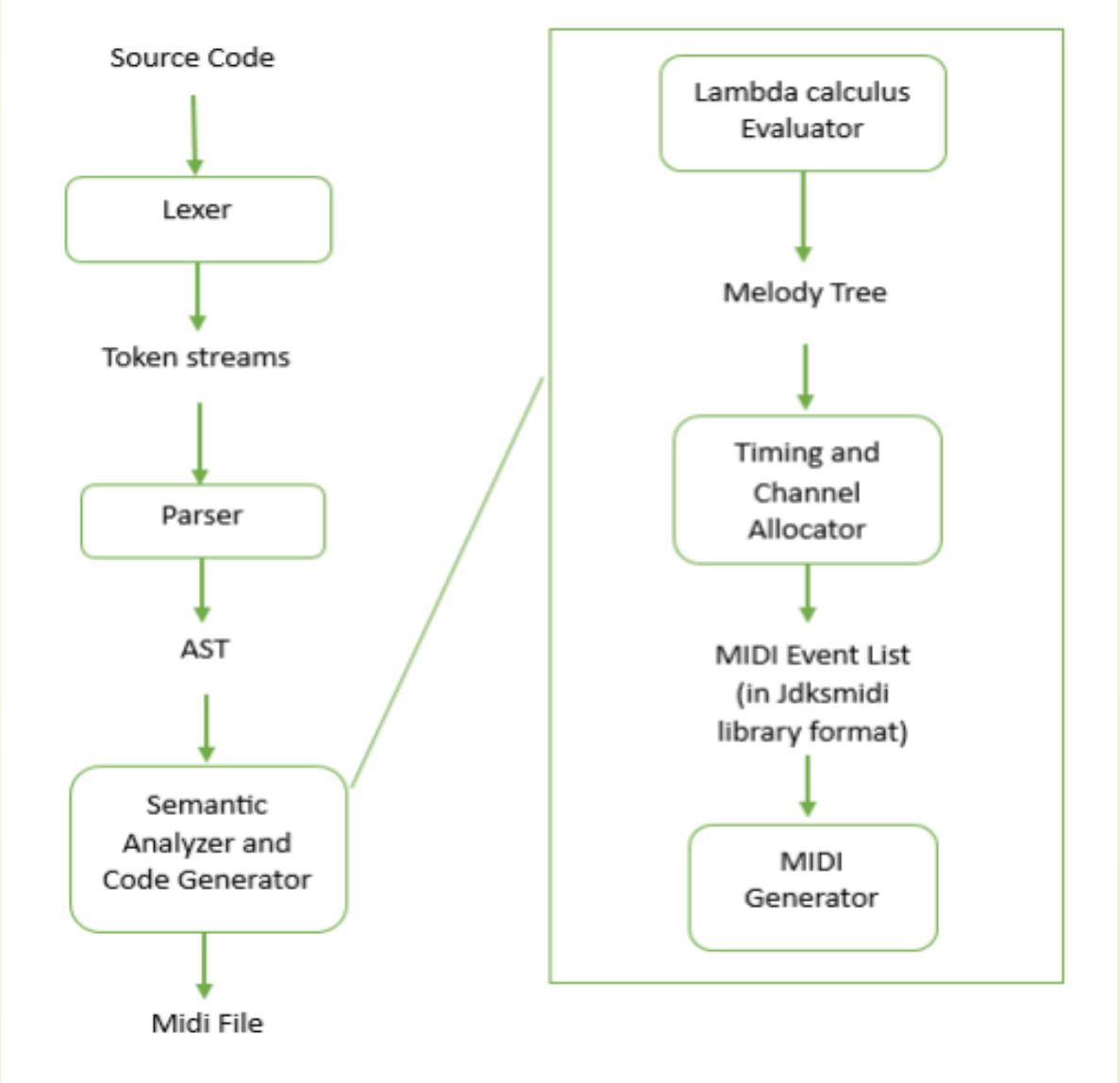
```
tempo = 300;
key = 5;

song = repeat 100 m;

repeat times melody =
if (<= times 0)
   []
   (seq melody (repeat (- times 1) melody));

m = { [ #1 0 3 1] ^[ (_5, 4)]} ;
```

Built-in function:

- Arithmetic operations
  + - * / %
- Logical operations
  == <> and or not *etc.*
- Melody operations
  seq par car cdr time *etc.*
- Control flow
  if

BLOCK DIAGRAM

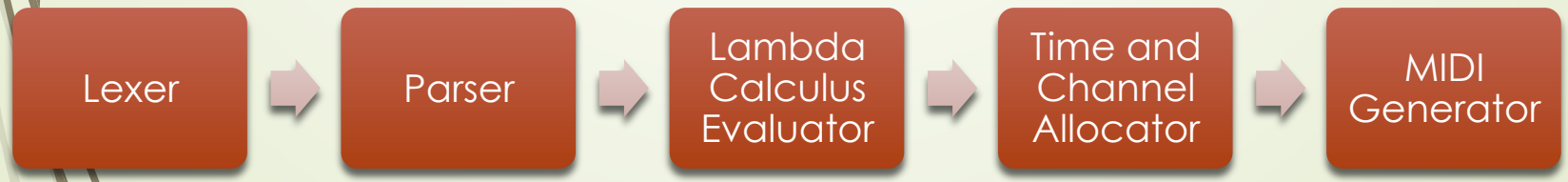
RYC: rock your composition



Input: Source Program

RYC: rock your composition

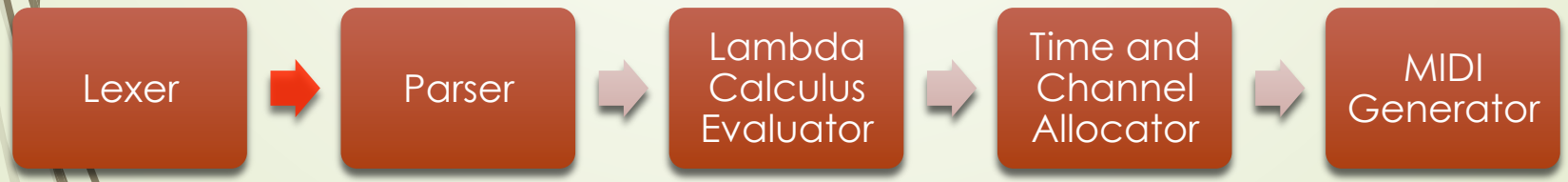
```
song = {[1 (f 4)] [(&3, 2)]} ;
f x = + x 1 ;
```



Token Stream

RYC: rock your composition

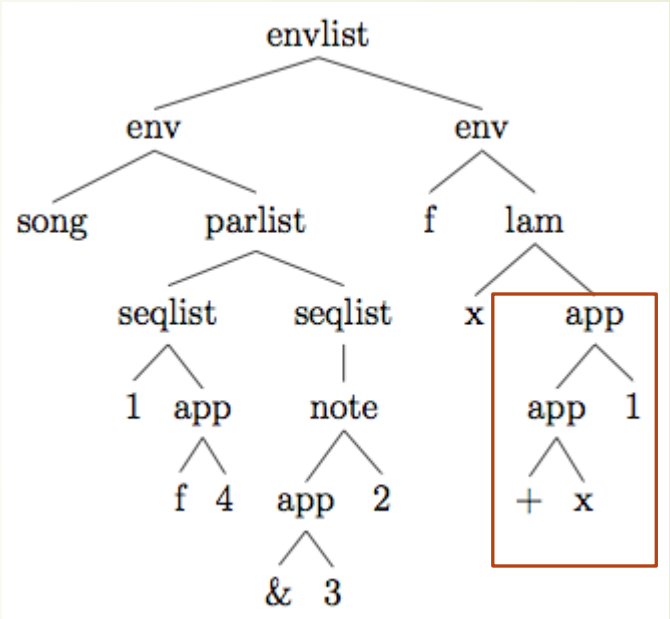
```
song = { [ 1 ( f 4 ) ] [ ( & 3 , 2 ) ] } ; f x =
+ x 1 ;
```



AST

RYC: rock your composition

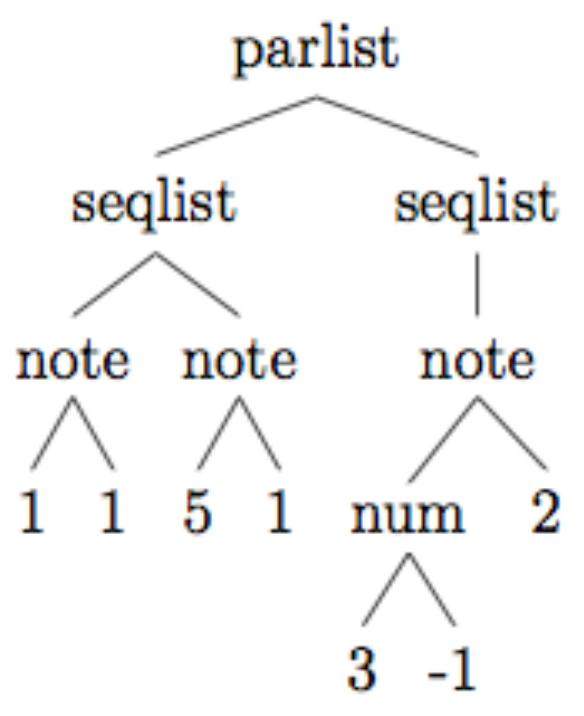
```
song = {[1 (f 4)] [(&3, 2)]} ;

f x = + x 1 ;
```



Lexer → Parser → Lambda Calculus Evaluator → Time and Channel Allocator → MIDI Generator

Melody Tree

RYC: rock your composition

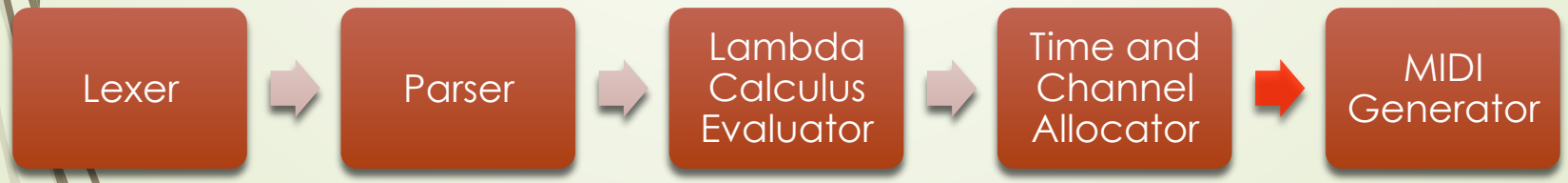


Lexer → Parser → Lambda Calculus Evaluator → Time and Channel Allocator → MIDI Generator

MIDI Event List

RYC: rock your composition

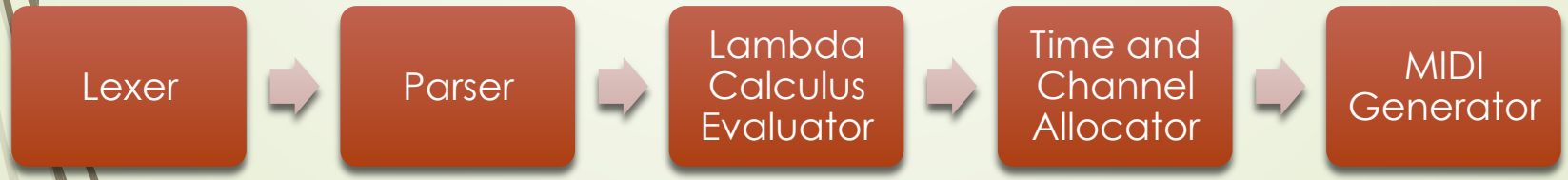
Pitch	Start time	End time	Channel
60	0	100	0
63	0	200	1
67	100	200	0



Output: MIDI file

RYC: rock your composition

a.mid



MIDI LIBRARY-JDKSMIDI

RYC: rock your composition

MIDI Event: msg

```
for (size_t i = 0; i < intervals.size(); ++i) {
  assert(intervals[i].channel != -1);
  msg.SetTime((unsigned long)(intervals[i].start + 0.5));
  msg.SetNoteOn(intervals[i].channel, intervals[i].note, 100);
  tracks.GetTrack(1)->PutEvent(msg);
  msg.SetTime((unsigned long)(intervals[i].end + 0.5));
  msg.SetNoteOff(intervals[i].channel, intervals[i].note, 100);
  tracks.GetTrack(1)->PutEvent(msg);
}
```

From gen_midi.c

DEVELOPMENT ENVIRONMENT

RYC: rock your composition

- Lexing and Parsing: lex, yacc
- Online Editor: Google Drive, Google Group
- Text Editors: Sublime, Xcode
- Version Control: Dropbox
- Test Suite: Make file for test suite of RYC files

TEST PLAN

RYC: rock your composition

UNIT TESTING

AST

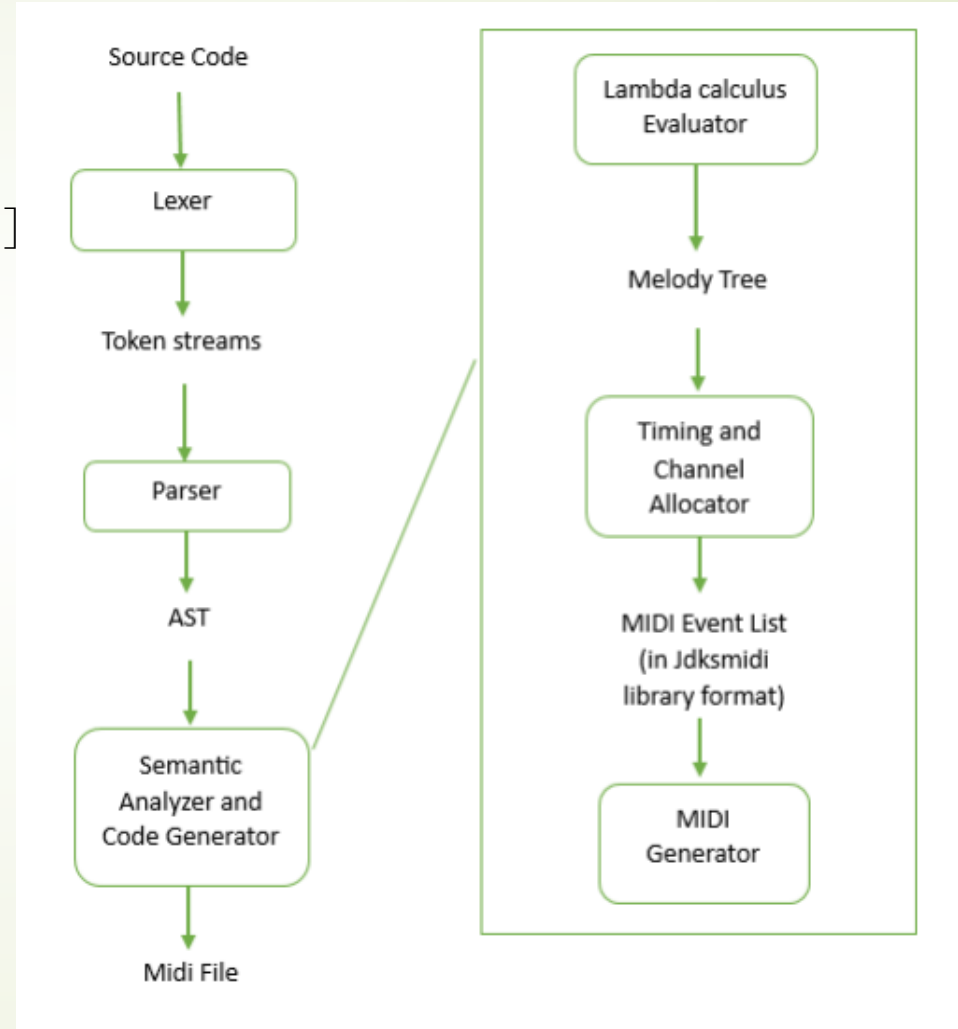
```
song := { [ (1.0, 1.0) (f 4.0) ]
[ ((& 3.0), 2.0) ] };
f := (\x -> ((+ x) 1.0));
```

Melody Tree

```
f{ [ (1.0, 1.0) (5.0, 1.0) ]
[ (3.0 - 1, 2.0) ] };
```

INTEGRATION TESTING

- sequenial melody
- parallel melody
- user-defined funcion
- built-in funcion



TEAM MANAGEMENT

RYC: rock your composition

- Timeline

Lessons learnt

RYC: rock your composition

Zhuojun: I have better understanding about diffrent phases of compiler

Junyang: the idea of lambda calculus and eazy evaluation fascinated me

Ruizhi: The semantic definition of each production is very tricky and educational

Dong: A lot of stuff needs to be taken into consideration when building a functional language

Xiaoji: It's been a lot of fun working with these people

Thank you! ♪ ♪ ♪

PRESENTERS

Project Manager	Xiaoji Li(xl2336)
Language Guru	Junyang Lu(jl3937)
System Architector	Ruizhi Liao(rl2643)
System Integrator	Dong Li(dl2768)
Tester	Zhuojun Huang(zh2193)