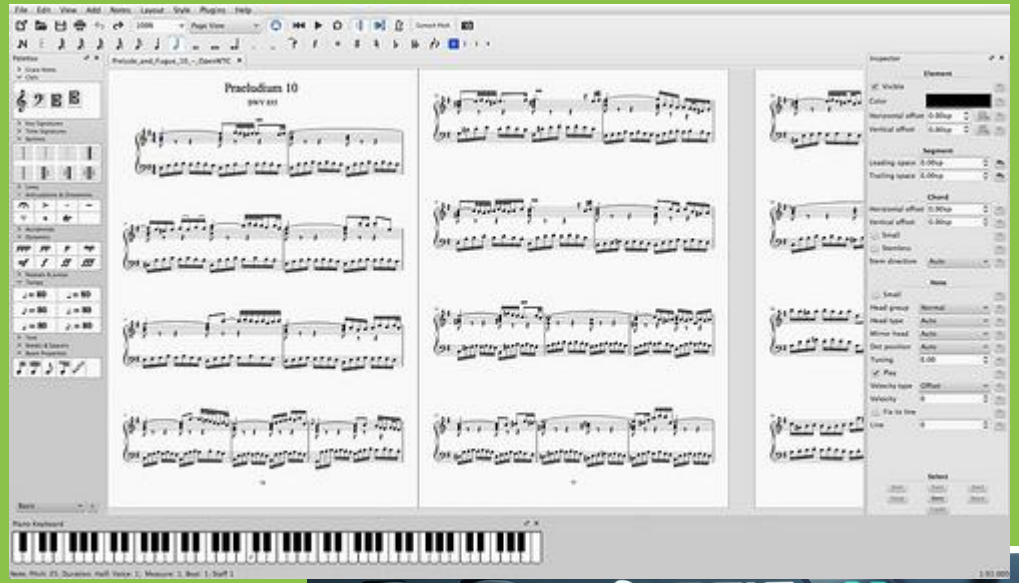
Beathoven

Eunice Kokor, Jake Kwon, Rodrigo Manubens, Ruonan Xu, & Sona Roy

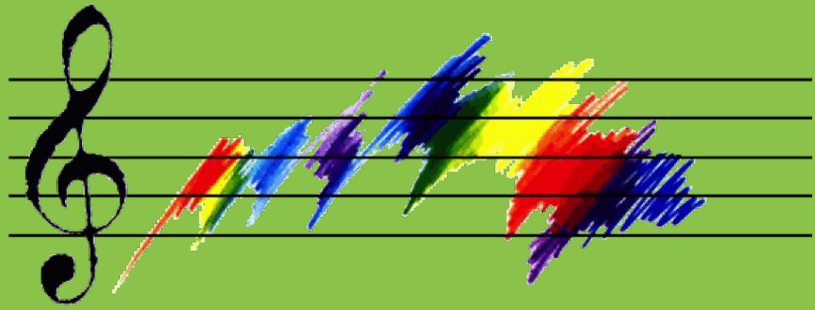
Introduction

MIDI

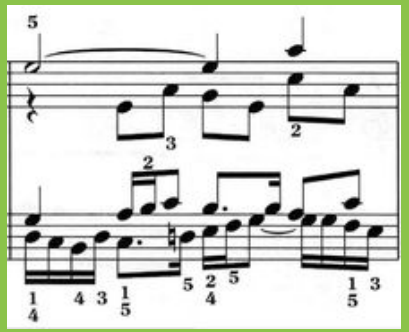
- Universal file format for digital music production
- Useful within compositions in practically all music software and devices



Goals



- MIDI File -- musical file, imported into composition software, flexible
- Stacked Music Scores -- simultaneous scores, same time + key, different (polyphonic) melodies
- Improvisation -- introduce randomness in note generation (chords + notes + improv = randomized music)
- Stretch goal (to build towards) -- melody + lyrics (tone, no beat = rap)



How do you use it?

Basic Datatypes

- Basic datatypes -- passing by value
  - bool, int, double, char, string
  - Music types: pitch, duration
- pitch:

```
pitch p1 = C3#;
pitch p2 = D5;
pitch p3 = E;
```

```
pitch re = 2;  /* equivalent to D4 */
pitch fa = 4^; /* equivalent to F5 */
pitch la = 6_; /* equivalent to A3 */
```

- duration:

```
duration quarter = 1/4;
duration quarter = 2/4;
duration quarter = 1/1;
```

More Datatypes -- passing by reference

- Struct
  - User-defined structs
  - Music type: Note

- Struct: Note
  - pitch: `note.p`
  - duration: `note.d`

```
Struct I_am_struct {
    int val;
}
Struct I_am_struct an_instance;
```

```
pitch p = F4#;
duration d = 1/16;
Note fSharpShort = p..d;
Note fa = 4..1/4;
Note defaultF = p; /* F4# pitch, 1/4 duration */
Note cWhole = ..1/1; /* C pitch, 1 duration */
```

More Datatypes -- passing by reference

- Array -- dynamically allocated
  - `datatype[] id;`
  - `int[] intarray = [1, 2, 3, 4, 5];`
  - Music type: Seq
- Seq
  - An array of Notes
  - <>
    - space-separated
    - int, pitch and duration wil be casted to Note.

```
Seq seq1 = <5..1/1 5 ..1/1>; // two G4 whole notes
Seq seq2 = <5..1/1 B5b..1/4 1 p note C
        p..1/2  p..h 1 ..h 7..h F..h >;
Seq twinkle = <1 1 5 5 6 6 5..3/8>;
```

Operations

- Arithmetic operator, Logical operators
- Array
  - Access: `array[idx]`
  - Subarray (create a copy):
    - Python-like:
      `array[0:4]; array[:7]; array[2:]; array[:]`
    - Deepcopy an array: `arr2 = arr1[:];`
  - Concatenate:
    `[arr1, arr2, ele1, ele2, arr3, ele3, ele4, ele5]`

Control Flow

- C-Like Syntax
  - If-Else Statement
  - While Statement
  - For Statement
  - Break
  - Return
- Syntax Sugar for `For`

```
int i;
for i in range(0, 100) {
    /* statements */
}
```

Function

```
/* overwrite the duration of `beats` to `melody` */
func Rhythm(Seq beats, Seq melody) -> Seq {
    if (len(beats) == 0) {
        print("empty beats");
        return melody;
    }
    else {
        int i = 0;
        int j;
        for j in range(0, len(melody)) {
            melody[j].d = beats[i].d;
            if (i + 1 < len(beats)) i = i + 1;
            else i = 0;
        }
    }
    return melody;
}
```

Function Listing

- print()

- len(Array array): Returns the number of array elements.
- str_of_pitch(pitch pitch'), str_of_duration(duration duration'), str_of_Note(Note): Returns the string of a pitch, a duration or a Note.

- render_as_midi(Seq seq): Output seq to a Midi file.
- render_seqs_as_midi(int num, Seq ...): Out sequences to a Midi file (multi-part)

Project Planning

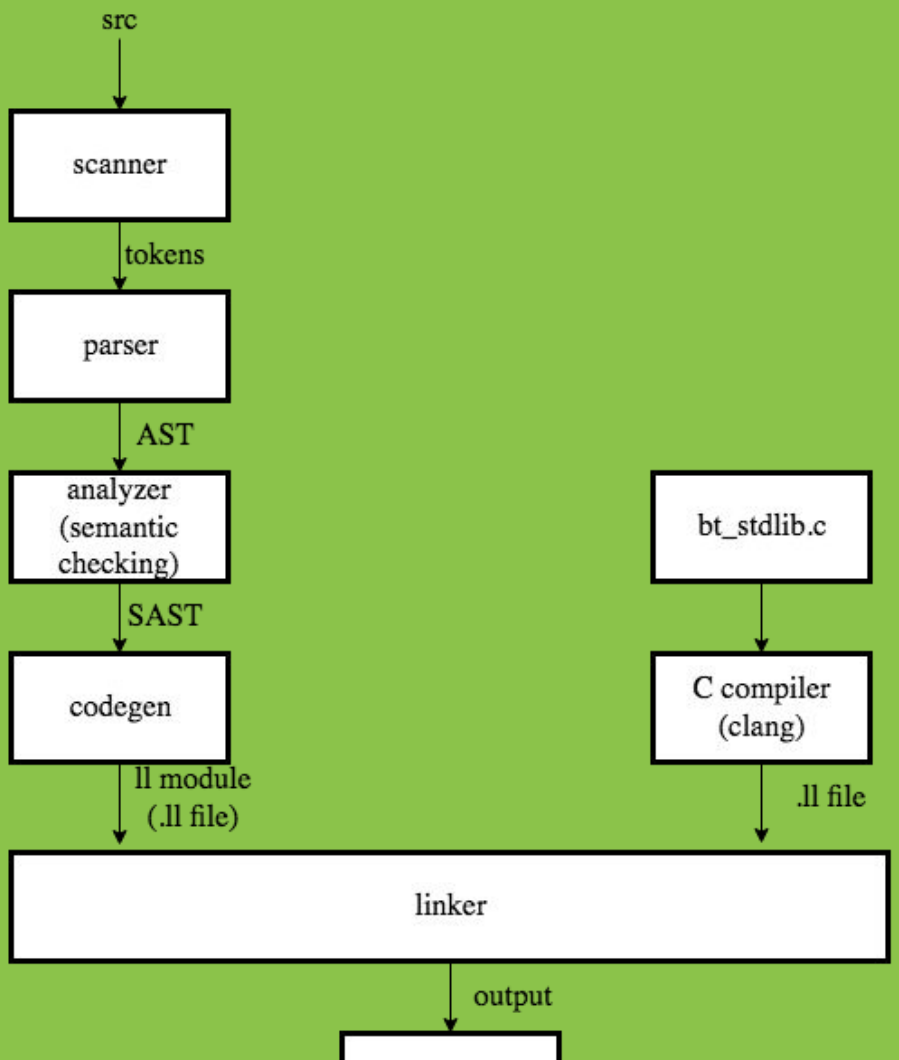
OCaml

- MicroC template -- scanner, parser, ast
- Codegen & restructure ast
- Codegen print function -- turning point
  - Variable assignment
  - Type of variable for printing
  - Create sast
- Codegen structs -- note data structure (pitch & duration)
- Iterative testing -- tests were added incrementally / feature

C Library / MIDI Roadmap

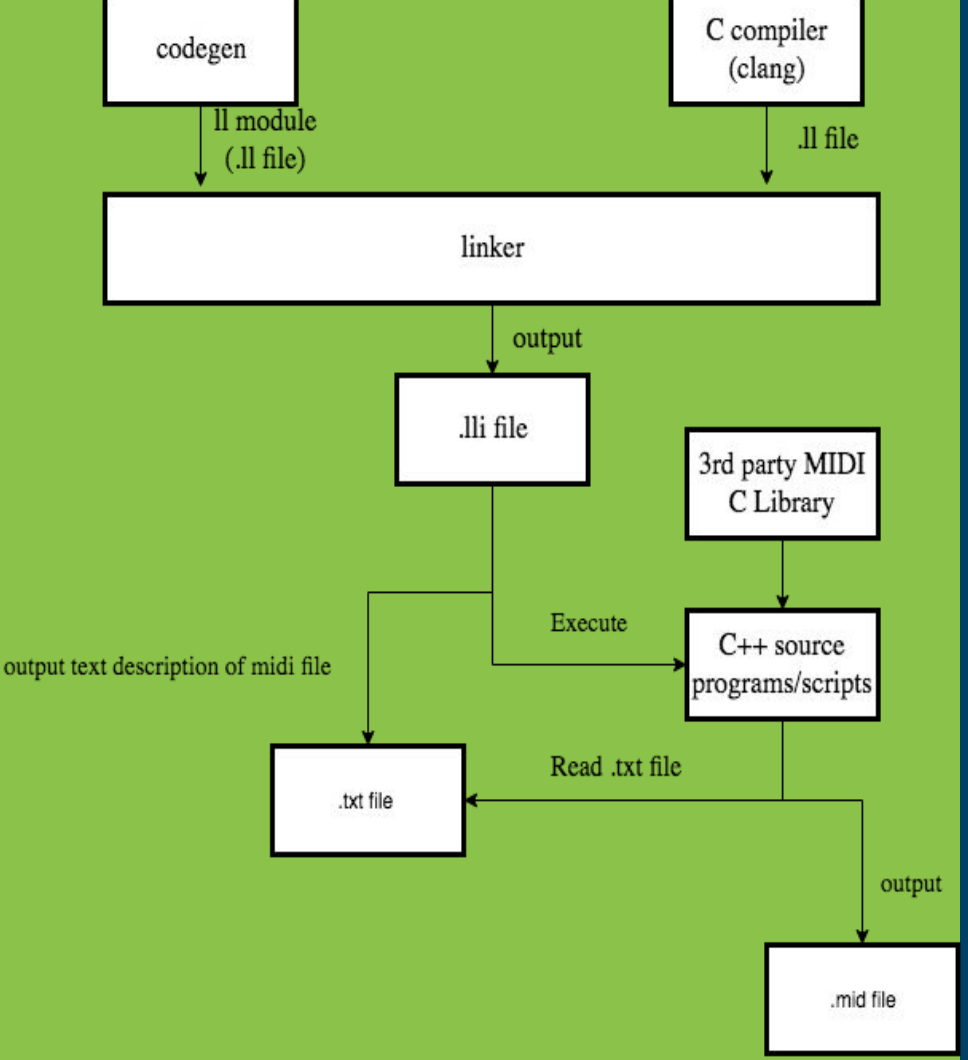
- Recreate MIDI Library -- too similar to interpreter
- LLVM Modules + C MIDI Library -- llvm program suite implementation was problematic given clang linking
- LLVM Modules + Wrapper Script + C MIDI Library -- Optimal given workaround and implementation of LLVM modules

Implementation



Compiler Architecture

3 separate parts, linked together

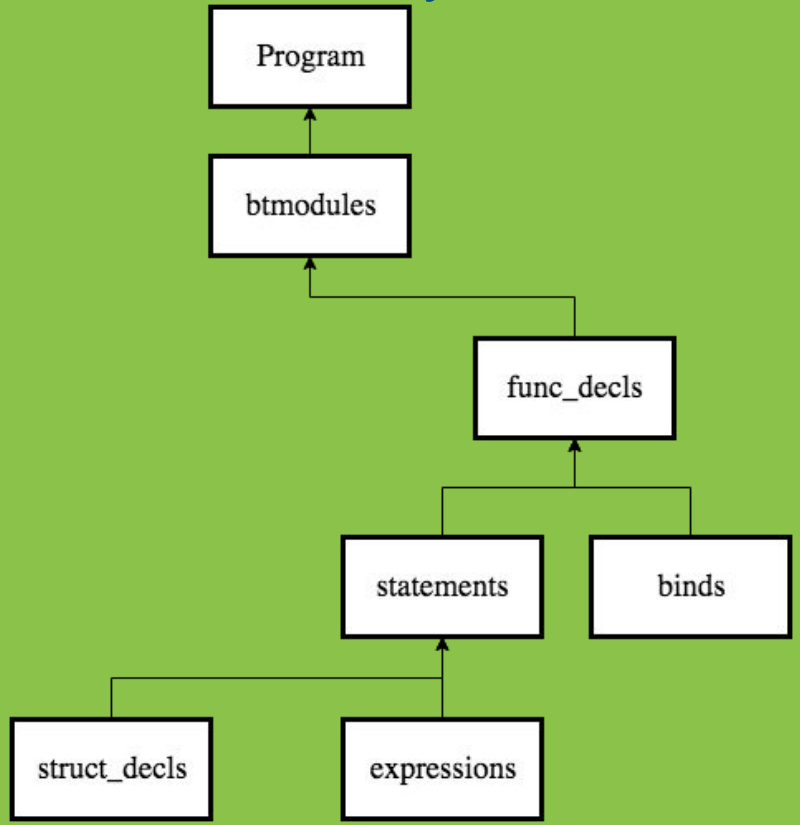


Compiler Architecture

3 separate parts, linked together

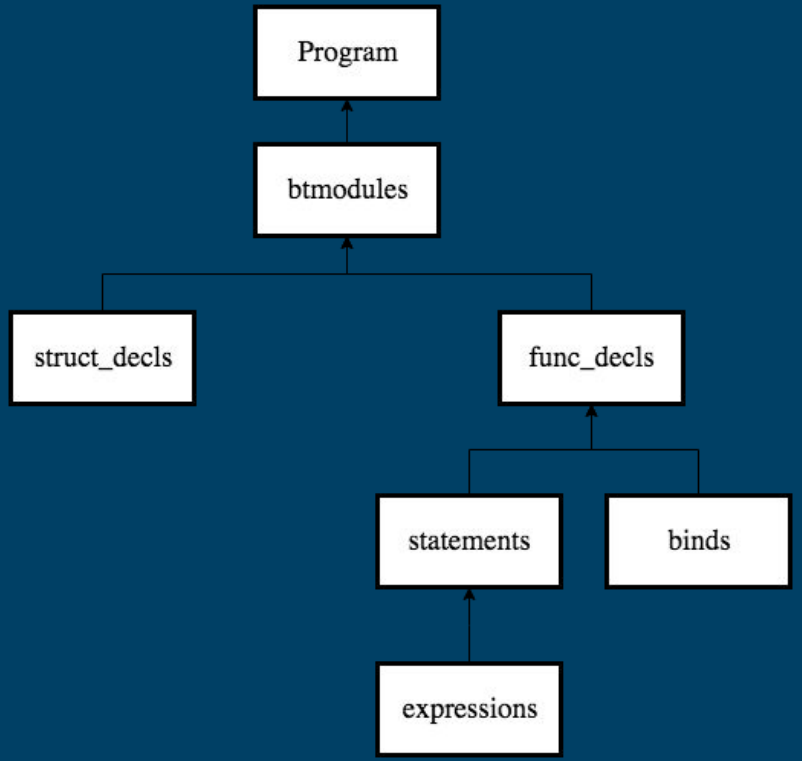
AST

Abstract syntax tree



SAST

Semantically-checked abstract syntax tree



Pitch Declaration and Assignment

1) pitch p = C4;
2) pitch q = 2;

0	1	2	3	4	5	6	7
H	C	D	E	F	G	A	B

LitInt(d) -> S.LitPitch(Char.chr (((d+1) mod 7 + 65)), 4, 0)

H	C	D	E	F	G	A	B
NA	67	68	69	70	71	65	66

Daniel Joseph Echikson <dje2125@columbia.edu> Oct 3

to Sonakshi, me, Rodrigo, Eunice

Hi Beathoven,

I'm emailing you all to (1) introduce myself and (2) set up a time for us to meet and discuss your proposal.

Introductions first. My name is Danny. My favorite color is green. I'm a history major, CS minor in CC. I'm a PLT vet; this is the second semester I'm TAing for Edward's PLT. I play the clarinet in Columbia's orchestra and Music Performance Program, so I know lots about music, which is why I chose to TA your group. Woohoo!

Ok now to business. Your proposal was overall good, but we need to discuss a few aspects of your project that could use a little tweaking. We can talk more in person. I hold my office hours in CSB 468 from 12:00 - 2:00 on Thursdays. Can at least 3/4ths of your group make it to my office hours?

If so, great! Please let me know that you are coming this Thursday. If not, please list all the times your group is available during the week, and I will let you know a time that works for me.

Only ONE of you should email be back. Please coordinate amongst yourselves first and have one person let me the results of your deliberations. Looking forward to working with all of you this semester!

Best,
Danny

Testing

```
Running Pass Test: note
Running... ../../beathoven.sh -c pass/note.bt logs/note.bt.ll
Running... lli logs/note.bt.ll > logs/note.bt.out
Comparing... logs/note.bt.out pass/note.out note.diff SUCCESS
Running Pass Test: pitch
Running... ../../beathoven.sh -c pass/pitch.bt logs/pitch.bt.ll
Running... lli logs/pitch.bt.ll > logs/pitch.bt.out
Comparing... logs/pitch.bt.out pass/pitch.out pitch.diff SUCCESS
Running Pass Test: printStr
Running... ../../beathoven.sh -c pass/printStr.bt logs/printStr.bt.ll
Running... lli logs/printStr.bt.ll > logs/printStr.bt.out
Comparing... logs/printStr.bt.out pass/printStr.out printStr.diff SUCCESS
Running Pass Test: sequence
Running... ../../beathoven.sh -c pass/sequence.bt logs/sequence.bt.ll
Running... lli logs/sequence.bt.ll > logs/sequence.bt.out
Comparing... logs/sequence.bt.out pass/sequence.out sequence.diff SUCCESS
Running Pass Test: string
Running... ../../beathoven.sh -c pass/string.bt logs/string.bt.ll
Running... lli logs/string.bt.ll > logs/string.bt.out
Comparing... logs/string.bt.out pass/string.out string.diff SUCCESS
Running Pass Test: stringequation
Running... ../../beathoven.sh -c pass/stringequation.bt logs/stringequation.bt.ll
Running... lli logs/stringequation.bt.ll > logs/stringequation.bt.out
Comparing... logs/stringequation.bt.out pass/stringequation.out stringequation.diff SUCCI
Running Pass Test: struct
Running... ../../beathoven.sh -c pass/struct.bt logs/struct.bt.ll
Running... lli logs/struct.bt.ll > logs/struct.bt.out
Comparing... logs/struct.bt.out pass/struct.out struct.diff SUCCESS
Running Pass Test: structarray
Running... ../../beathoven.sh -c pass/structarray.bt logs/structarray.bt.ll
Running... lli logs/structarray.bt.ll > logs/structarray.bt.out
Comparing... logs/structarray.bt.out pass/structarray.out structarray.diff SUCCESS
Running Pass Test: while
Running... ../../beathoven.sh -c pass/while.bt logs/while.bt.ll
Running... lli logs/while.bt.ll > logs/while.bt.out
Comparing... logs/while.bt.out pass/while.out while.diff SUCCESS
You have 0 out of 25 PASS errors

####  End of Pass Compiler Tests!  ####
➜  Beathoven (master)
```

```
###### Testing sequence
../../beathoven.sh -c pass/sequence.bt logs/sequence.bt.ll
Uncaught exception:

  Exceptions.VariableNotDefined("_bt.p")

Raised at file "analyzer.ml", line 28, characters 30-63
Called from file "analyzer.ml", line 92, characters 21-39
Called from file "analyzer.ml", line 310, characters 15-37
Called from file "analyzer.ml", line 411, characters 26-47
Called from file "analyzer.ml", line 423, characters 24-50
Called from file "list.ml", line 59, characters 20-23
Called from file "list.ml", line 59, characters 32-39
Called from file "analyzer.ml", line 426, characters 23-53
Called from file "analyzer.ml", line 508, characters 17-51
Called from file "analyzer.ml", line 551, characters 8-81
Called from file "list.ml", line 59, characters 20-23
Called from file "analyzer.ml", line 591, characters 27-64
Called from file "beathoven.ml", line 35, characters 17-41
lli logs/sequence.bt.ll > logs/sequence.bt.out
'main' function not found in module.
diff -b logs/sequence.bt.out pass/sequence.out > logs/ sequence.diff
FAILED logs/sequence.bt.out differs from pass/sequence.out
###### FAILED
```

General Testing Plan

- After Something Gets Implemented on Compiler
  - Add expected functional code to “testall.bt”
  - Grab individual test cases from functional code.
  - Start from basics, like assignment to using multiple things.
- Checking for exceptions
  - Exeptions.ml contains definitions of most exceptions raised throughout our compiler
  - Create fail tests for those exceptions
  - Also add parser failures to failure tests
    - Examples: reserved keywords not in correct order

System Music Testing

- Since the midi file is generated in a different directory, created a script to test all our example files midi files

Rhythm.bt

```
/* overwrite the duration of `beats` to `melody` */
func Rhythm(Seq beats, Seq melody) -> Seq {
    if (len(beats) == 0) {
        print("empty beats");
        return melody;
    }
    else {
        int i = 0;
        int j;
        for j in range(0, len(melody)) {
            melody[j].d = beats[i].d;
            if (i + 1 < len(beats)) i = i + 1;
            else i = 0;
            //print(i);
        }
    }
    return melody;
}

duration w = 1/1;
duration q = 2/4;
Seq melody = <1 1 5 5 6 6 5..2/4>;
Seq beats = <w w w w w w>;
Seq newBeat = melody[:];
Rhythm(beats, newBeat);
Seq all = [melody, newBeat];
render_as_midi(all);
```

Demo

Fur Elise, Sweet Child of Mine