




E-CATZ

Ethiopia Mengesha, Chianna Cohen, Annie Sui, Tim Vallancourt

:°◇(⊙⊕^⊕⊙)/♪☆♪◇:°



E-CATZ Overview

Goal: To enable users to explore their musical artistry through code.

Notable Features:

- Note Objects

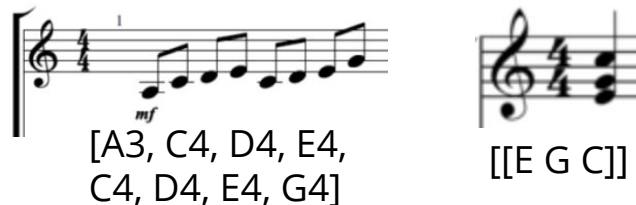


- Writing Notes to LilyPond

```
Note[] output = [[D3, E3, F#3, G3]];
```



- Sequences and Harmonies



- Sequences of Harmonies



E-CATZ in One Slide

Example of inversion



```
#include "stdlib.catz" _____ Include statement for preprocessing
/* 'invert' the array by flipping the distances between each note */
def Note[] inverse(Note[] input) { _____ Function definition
    Note[] output = new Note[input.length]; _____ Note array instantiation
    output[0] = input[0]; _____ Array access and assignment
    int initialPitch = noteToInt(input[0]); _____ Call to helper function in StdLib
    for (int i = 1; i < input.length; i = i + 1) { _____ For-loop
        int delta = noteToInt(input[i-1]) - noteToInt(input[i]); _____ Math operators on pitches
        Note newPitch = addToNote(output[i-1], delta);
        output[i] = new Note(newPitch.pitch, input[i].rhythm); _____ Note construction and
    } _____ field access
    return output;
}
Note[] bluesScale = [C4, D4, D#4, E4, G4, A5]; _____ Array literal
write(inverse(bluesScale), "filename" ); _____ Write function takes an array and
_____ outputs a LilyPond file
```

Compiler Architecture

source.catz



Parser



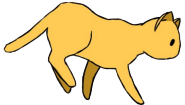
Semantic checking



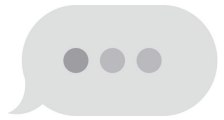
C library



Executable



Preprocessing



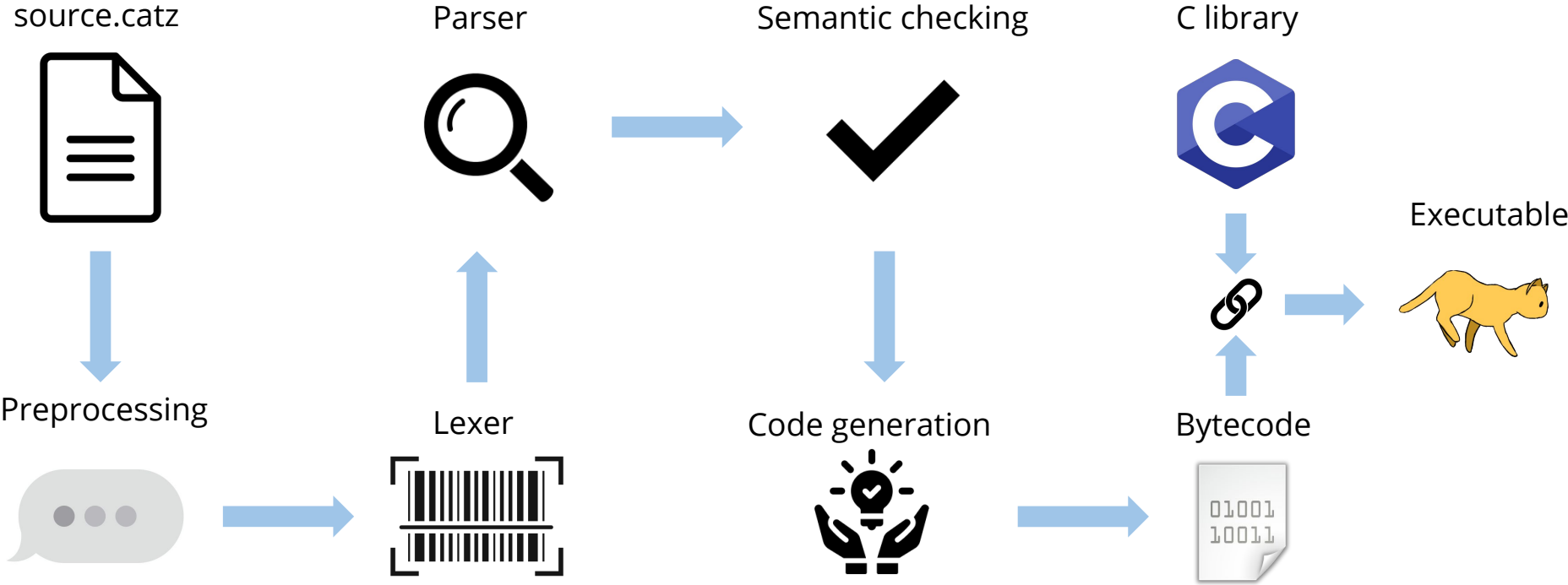
Lexer



Code generation



Bytecode



Notes

- Represented as structs with a pitch and rhythm literal

Note constructor

```
Note myNote = new Note(C#5, hf);
```

↑ ↑
pitch rhythm

Pitch literal, default quarter note

```
Note myNote = B3;
```

↑
pitch

- Access and modify the pitch and rhythm of a Note using dot (.) operator

```
Note x = new Note(F#3, wh);  
Note y = new Note(x.pitch, hf);  
x.rhythm = ts;
```

Arrays

- Construct arrays by specifying a length or using an array literal

```
int[] x = new int[10]; /* empty array of length 10 */  
int[] y = [3, 5, 6, 7]; /* array literal with 4 values */
```

- Use nested arrays to create harmonies

```
int[][] a = new int[][5]; /* array of 5 empty int arrays */  
int[][] b = [[3, 5, 7], [1, 2]]; /* array of 2 arrays of  
varied length */
```

Standard Library

- Written in E-CATZ
- Contains useful built-in functions
 - *noteToInt()*
 - *concatArray()*
 - *addToNote()*
 - *octaveChange()*

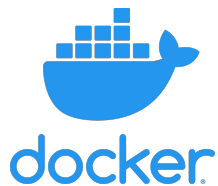
Write()

- Written in C
- Translates E-CATZ output into LilyPond
- Inserts LilyPond notation for Notes, Harmonies, and Sequences



How to Use E-CATZ

Set up Docker environment



Build E-CATZ compiler

make

Preprocess & execute program

```
./run-ecatz.sh [program.catz]
```

Open the .ly program
outputted by an E-CATZ
program in LilyPond



Acquire MIDI file



Listen to your work!



E-CATZ Live Demo

We will now show you a MIDI soundtrack we made from E-CATZ!



Future Work

Extended Features

- Capability to play notes of different rhythms in the same chord
- Dotted-note rhythms
- Time Signatures
- Rests
- Ability to read a LilyPond file into a program that one can manipulate
- More efficient memory management

Questions?