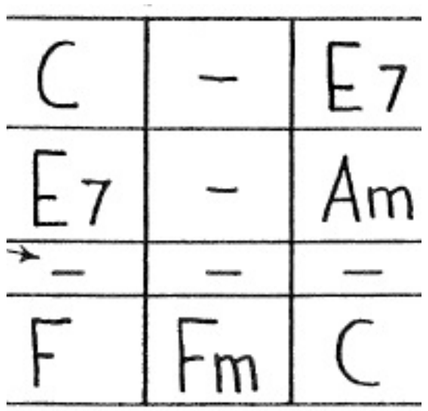
OPEN CHORDS CHARTS



I'm Christophe Benz

developer and jazz pianist

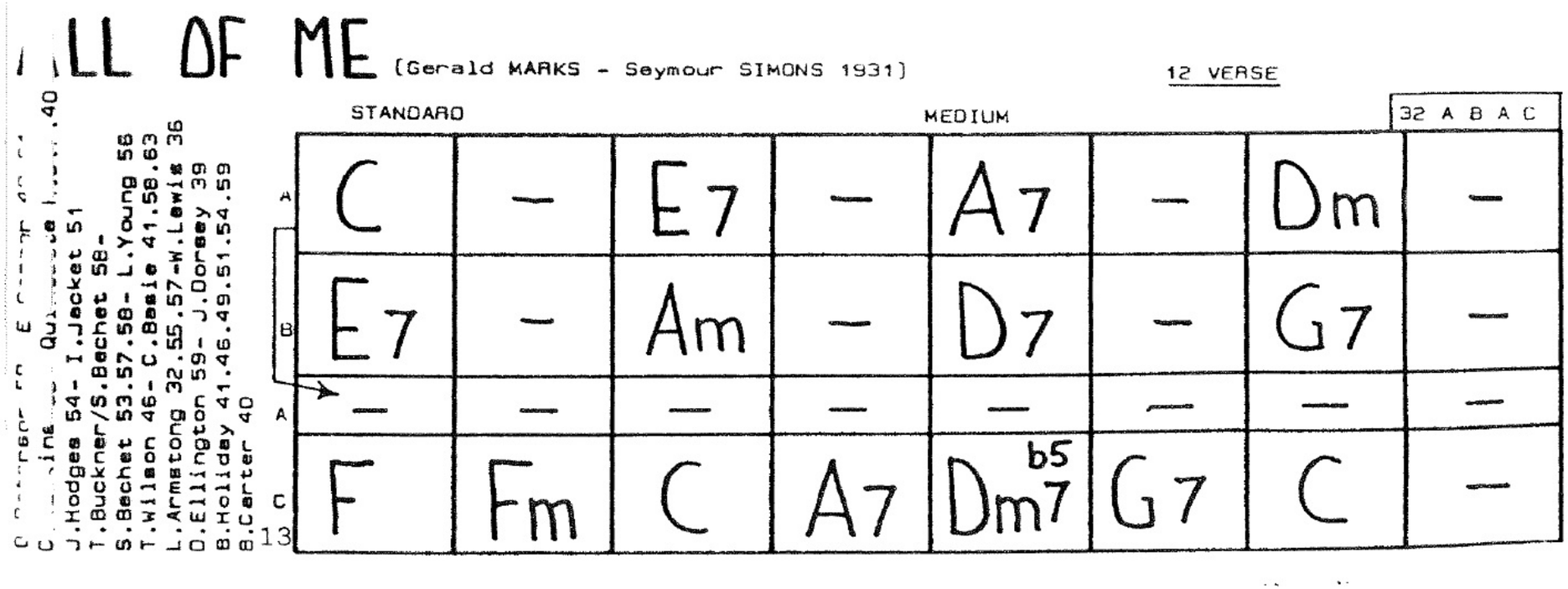
contact@cbenz.org

Jam session!

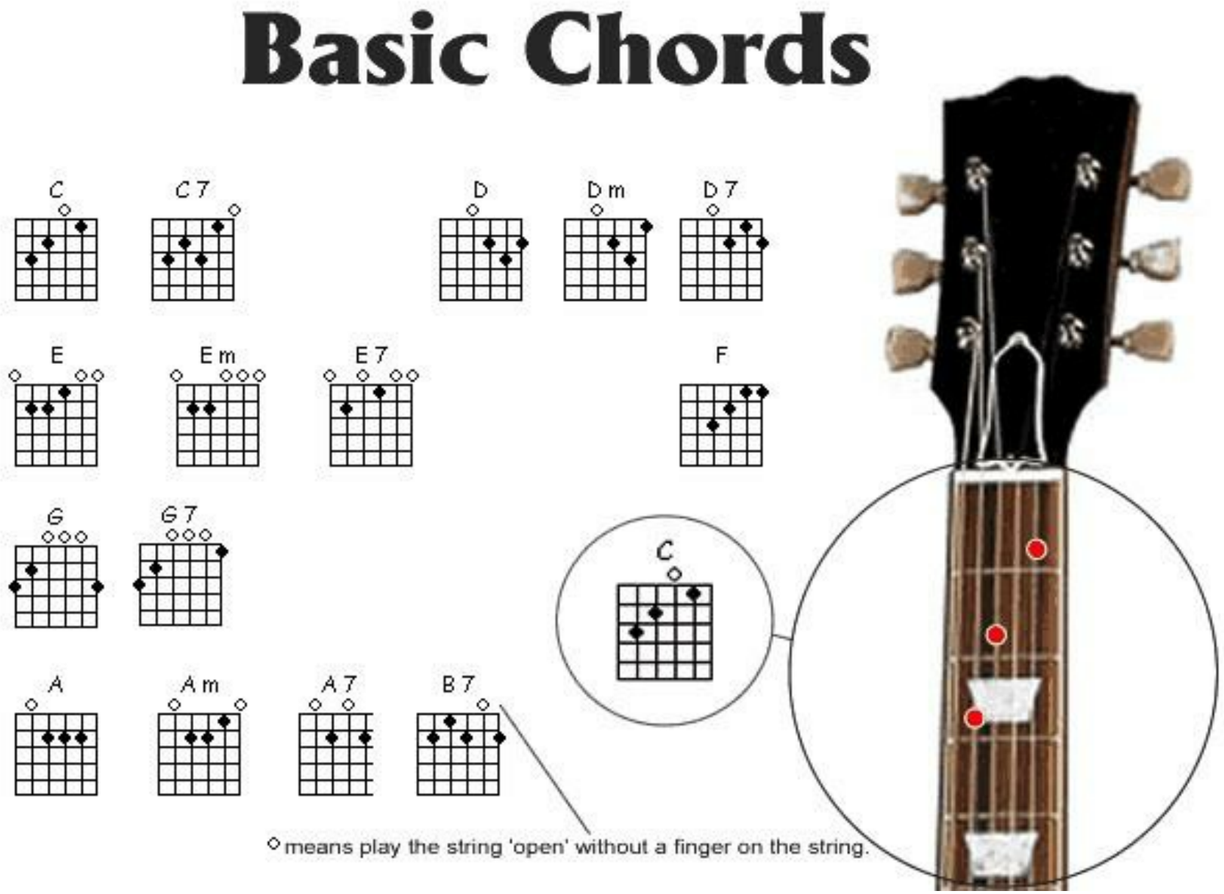
CHORDS CHARTS?

- song = melody + chords + tonality
- applies to rock, blues, jazz, folk
- transposing is changing tonality

EXAMPLE: JAZZ SONG



It's an image :-(



We just keep the "letters"

FOCUS ON THE FIRST CHORD

C

Means "C Major"

ANOTHER ONE

Fm

Means "F minor"

WHAT IS A CHORD?

- a "root" music note
  - A, B, C, D, E, F, G
- a quality
  - Major, minor, etc.

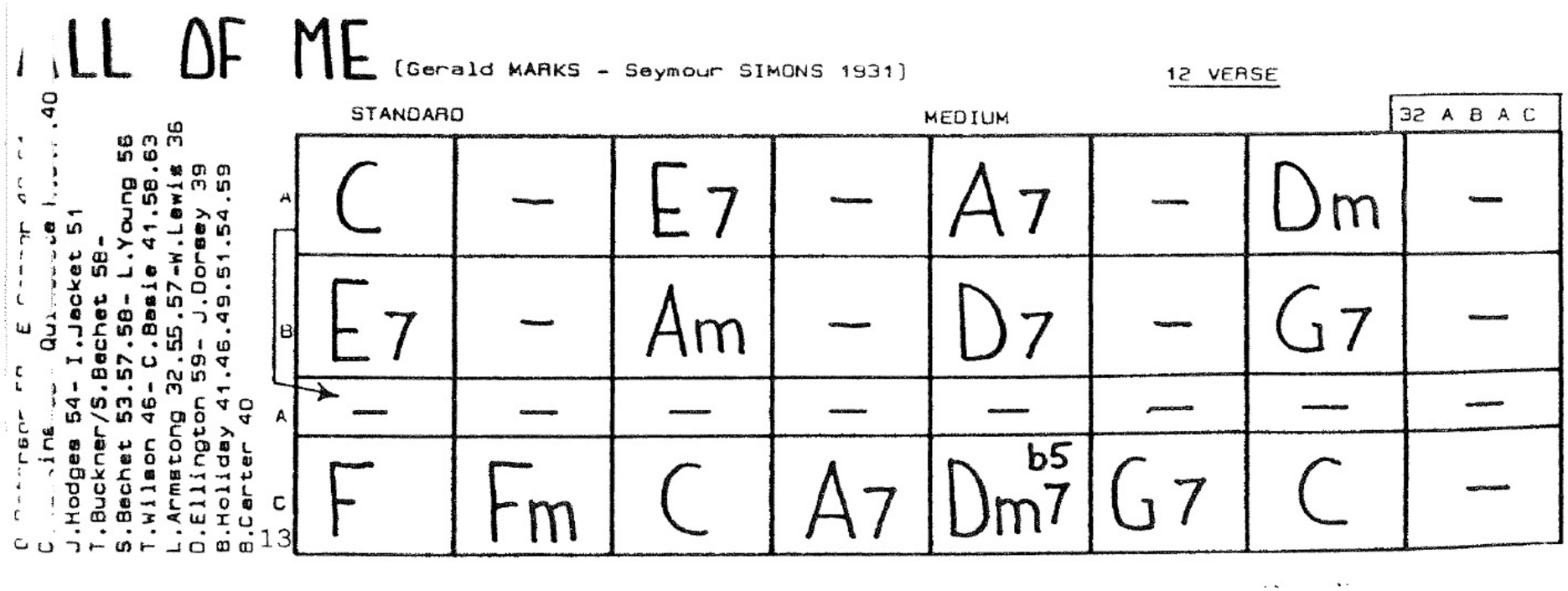
MUSIC NOTES IN ELM

```
type Note
    = A | Af | As
    | B | Bf | Bs
    | C | Cf | Cs
    ...
    | F | Ff | Fs
    | G | Gf | Gs

type alias OctaveIndex = Int

toOctaveIndex : Note -> OctaveIndex

toOctaveIndex Es == toOctaveIndex Ff -- enharmonics
```

CHORDS IN ELM

```
type Chord
    = Chord Note Quality

type Quality
    = Major
    | Minor
    | Augmented
    | MajorSixth
    | MinorSixth
    | Seventh
    ...
```

CHORDS CHARTS IN ELM



LL OF ME (Gerald MARKS - Seymour SIMONS 1931)

12 VERSE

STANDARD MEDIUM

32 A B A C

A	C	–	E7	–	A7	–	Dm	–
B	E7	–	Am	–	D7	–	G7	–
A	1	–	–	–	–	–	–	–
C	F	Fm	C	A7	Dm7 b5	G7	C	–

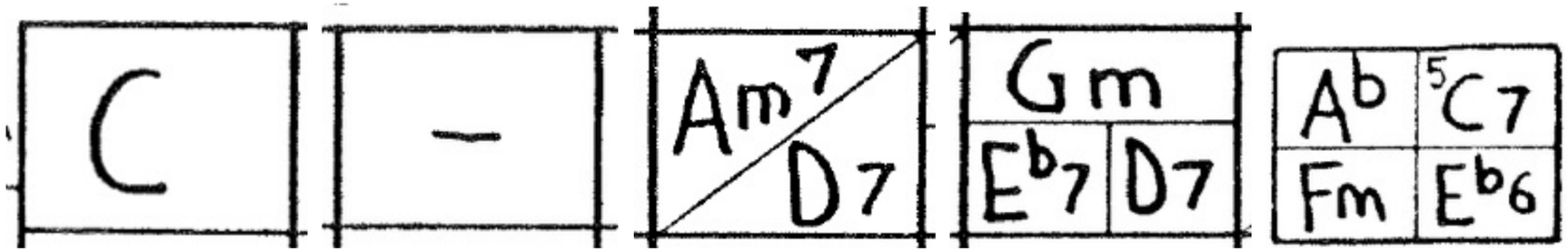
13

J.Hodges 54- I.Jacket 51
T.Buckner/S.Bechet 58-
S.Bechet 53.57.58- L.Young 56
T.Wilson 46- C.Basie 41.58.63
L.Armstong 32.55.57-W.Lewis 36
D.Ellington 59- J.Dorsey 39
B.Holiday 41.46.49.51.54.59
B.Carter 40

CHORDS CHARTS IN ELM

```
type alias Chart =
    { title : String
    , tonality : Note
    , parts : List Part
    }

type Part
    = Part String (List Bar)
    | PartRepeat String
```

BARS EXAMPLES



BARS IN ELM

```
type Bar
    = Bar (List Chord)
    | BarRepeat
```

TRANSPOSE A CHORDS CHART

```
transpose : Note -> Chart -> Chart
transpose key chart =
    let
        interval =
            Note.interval chart.key key

        newParts =
            chart.parts |> List.map (transposePart interval)
    in
        { chart
            | key = key
            , parts = newParts
        }
```

MAP A FUNCTION ON PART BARS

```
-- reminder
type Part
    = Part String (List Bar)
    | PartRepeat String

mapPartBars : (List Bar -> List Bar) -> Part -> Part
mapPartBars f part =
    case part of
        Part partName bars ->
            Part partName (f bars)

        PartRepeat _ ->
            part
```

TRANSPOSE A PART

```
transposePart : Interval -> Part -> Part
transposePart interval part =
    part |> mapPartBars (List.map (transposeBar interval))

transposeBar : Interval -> Bar -> Bar
transposeBar interval bar =
    bar |> mapBarChords (List.map (Chord.transpose interval))
```

TRANSPOSE A CHORD

```elm
-- Music.Chord
transpose : Interval -> Chord -> Chord
transpose interval (Chord note quality) =
    Chord (Note.transpose interval note) quality

-- Music.Note
transpose : Interval -> Note -> Note
transpose interval note =
    let
        octaveIndex =
            toOctaveIndex note
    in
        fromOctaveIndex (octaveIndex + interval)
```

CHART VIEWER / EDITOR

All of me **C**

A	C	–	E7	–	A7	–	Dm	–
B	E7	–	Am	–	D7	–	G7	–
A	–	–	–	–	–	–	–	–
C	F	Fm	C	A7	Dø	G7	C	–

Transpose to: C ▾

```
allOfMe : Chart
allOfMe =
    let
        partA = Part "A"
            [ Bar [ Chord C Major ]
            , BarRepeat
            ...
            ]
        partB = ...
        partC = ...
    in
        { title = "All of me", key = C
        , parts = [ partA, partB, PartRepeat "A", partC ]
        }
```

CHART VIEWER / EDITOR

```
type alias Model =
    { chart : Chart
    , viewedKey : Note
    }

view model =
    let
        viewedChart =
            model.chart
                |> Music.Chart.transpose model.viewedKey
    in
        ...
```

CHART VIEWER / EDITOR

DEMO

CHART VIEWER / EDITOR

```
type Msg
    = Edit
    | Save
    | SelectBar BarReference
    | SetChord BarReference ChordIndex Chord
    | SetBarRepeat BarReference Bool
    | SetViewKey Note

type alias BarReference =
    { partIndex : PartIndex
    , barIndex : BarIndex
    }
```

CHART VIEWER / EDITOR

All of me **C**

A	G	–	B7	–	E7	–	Am	–
B	B7	–	Em	–	A7	–	D7	–
A	–	–	–	–	–	–	–	–
C	C	Cm	G	E7	Aø	D7	G	–

Transpose to: G ▾

A DOMAIN SPECIFIC LANGUAGE

```
title: All of me
key: C

= A
C - E7 - A7 - Dm -

= B
E7 - Am - D7 - G7 -

= A

= C
F Fm C A7 Dø G7 C -
```

ELM-TOOLS/PARSER

```
chart : Parser Chart
chart =
    inContext "chart" <|
        succeed Chart
            |. spacesAndNewlines
            |. symbol "title:"
            |. spaces
            |= keepUntilEndOfLine
            |. newLine
            |. symbol "key:"
            |. spaces
            |= note
            |. spacesAndNewlines
            |= repeat oneOrMore (part |. spacesAndNewlines)
            |. end
```

CONCLUSION

- Elm and ADTs is awesome to model a domain like music
- Refactoring is a real pleasure
- Just the beginning!
- Questions?