# Languages and Patterns

Mark Hopkins

September 2021, **fp-syd**

@antiselfdual
antiselfdual.com

## Learning languages

triton

## A language drill tool for German

- Abstract grammatical model
- Generate sentences
- Pass to concrete English grammar model
  $\Rightarrow$ speak or print
- Pass to concrete German grammar model
  $\Rightarrow$ use to check user's translation

## Demo

## Turkish

|  |  |
|---|---|
| Turkish | • 80 million native speakers |
| Turkic | • over 35 languages<br>• 170 million native speakers |

## Turkish grammar

- no articles
- no grammatical gender
- incredibly regular
- agglutinative
- vowel harmony
- evidentiality
- interesting history

## It also sounds cool

Türkçe konuşabiliyorum.

Bambaşka

Çarşamba

Kırmızılı kız kızgın.

Şoförün tedbirlisi herkesin sevgilisi.

- Enayisin.

- Niçin enayiyim? / Niçin enayi imişim?

## Agglutinative

| | |
|---|---|
| Avrupa | Europe |
| Avrupalı | European |
| Avrupalılaş | Europeanise (intr.) |
| Avrupalılaştır | Europeanise (tr.) |
| Avrupalılaştırama | be unable to Europeanise |
| Avrupalılaştıramadık | one unable to be Europeanised |
| Avrupalılaştıramadıklar | those unable to be Europeanised |
| Avrupalılaştıramadıklarımız | those who we could not Europeanise |
| Avrupalılaştıramadıklarımızdan | of those who we could not Europeanise |
| Avrupalılaştıramadıklarımızdanmış | (reportedly) of we could not Europeanise |
| Avrupalılaştıramadıklarımızdanmışsınız | you are (reportedly) of those we could not Europeanise |
| Avrupalılaştıramadıklarımızdanmışsınızcasına | as if you were of those we could not Europeanise |

## Vowels

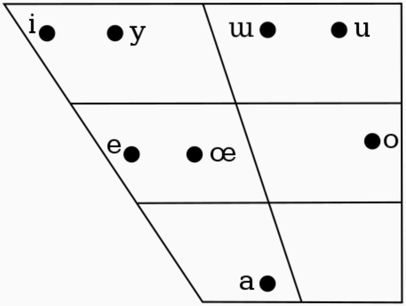


**Table 1:** Turkish vowels

| | Front | | Back | |
|---|---|---|---|---|
| | **Unrounded** | **Rounded** | **Unrounded** | **Rounded** |
| **High** | i /i/ | ü /y/ | ı /ɯ/ | u /u/ |
| **Low** | e /e/ | ö /œ/ | a /a/ | o /o/ |

**Table 2:** Examples of case endings

| | *house* | *idea* | *sky* | *book* | *ball* |
|---|---|---|---|---|---|
| Abs | ev | fikir | gök | kitap | top |
| Def. obj. | evi | fikri | gökü | kitabı | topu |
| Gen. | evin | fikrin | gökün | kitabın | topun |
| Dat. | eve | fikre | göke | kitaba | topa |
| Loc. | evde | fikirde | gökte | kitapta | topta |
| Abl. | evden | fikirden | gökten | kitaptan | toptan |

**Table 3:** Summary of case endings

| Last vowel of absolute | **e** or **i** | **ö** or **ü** | **a** or **ı** | **o** or **u** |
|---|---|---|---|---|
| Definite objective | **-(y)i** | **-(y)ü** | **-(y)ı** | **-(y)u** |
| Genitive *of* | **-(n)in** | **-(n)ün** | **-(n)ın** | **-(n)un** |
| Dative *to, for* | **-(y)e** | | **-(y)a** | |
| Locative *in, on, at* | **-de** | | **-da** | |
| Ablative *from, out of* | **-den** | | **-dan** | |

**Table 4:** Summary of case endings

| Last vowel of absolute | **F** and **U** | **F** and **R** | **B** and **U** | **B** and **R** |
|---|---|---|---|---|
| Definite objective | **-(y)i** | **-(y)ü** | **-(y)ı** | **-(y)u** |
| Genitive *of* | **-(n)in** | **-(n)ün** | **-(n)ın** | **-(n)un** |
| Dative *to, for* | **-(y)e** | | **-(y)a** | |
| Locative *in, on, at* | **-de** | | **-da** | |
| Ablative *from, out of* | **-den** | | **-dan** | |

```
declineObj :: String -> String
declineObj n = case lastVowel n of
  v | backness v == Front &&
      rounding v == Unrounded -> append n "y" "i"
  v | backness v == Front &&
      rounding v == Rounded   -> append n "y" "ü"
  v | backness v == Back &&
      rounding v == Unrounded -> append n "y" "ı"
  v | backness v == Back &&
      rounding v == Rounded   -> append n "y" "u"
```

```haskell
declineObj :: String -> String
declineObj n = case (backness &&& rounding) (lastVowel n) o
    (Front, Unrounded) -> append n "y" "i"
    (Front, Rounded)   -> append n "y" "ü"
    (Back, Unrounded)  -> append n "y" "ı"
    (Back, Rounded)    -> append n "y" "u"
```

```
pattern FrontVowel :: Vowel
pattern FrontVowel <- (backness -> Front)

pattern BackVowel :: Vowel
pattern BackVowel <- (backness -> Back)

pattern UnroundedVowel :: Vowel
pattern UnroundedVowel <- (rounding -> Unrounded)

pattern RoundedVowel :: Vowel
pattern RoundedVowel <- (rounding -> Rounded)
```

```
declineObj :: String -> String
declineObj n = case dup (lastVowel n) of
  (FrontVowel, UnroundedVowel) -> append n "y" "i"
  (FrontVowel, RoundedVowel)   -> append n "y" "ü"
  (BackVowel, UnroundedVowel)  -> append n "y" "ı"
  (BackVowel, RoundedVowel)    -> append n "y" "u"
```

```
declineObj :: String -> String
declineObj n = case lastVowel n of
  (dup -> (FrontV, UnroundedV)) -> append n "y" "i"
  (dup -> (FrontV, RoundedV))   -> append n "y" "ü"
  (dup -> (BackV, UnroundedV))  -> append n "y" "ı"
  (dup -> (BackV, RoundedV))    -> append n "y" "u"
```

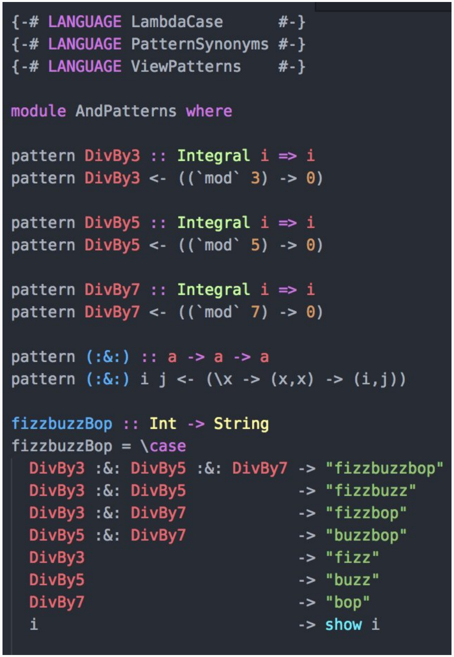
## And patterns!

```
pattern (:&:) :: a -> a -> a
pattern (:&:) a b <- (dup -> (a,b))
```

Thank you Arnold deVos (who did this for Scala)

## And patterns!

```
declineObj :: String -> String
declineObj n = case lastVowel n of
  FrontVowel :&: UnroundedVowel -> append n "y" "i"
  FrontVowel :&: RoundedVowel   -> append n "y" "ü"
  BackVowel  :&: UnroundedVowel -> append n "y" "ı"
  BackVowel  :&: RoundedVowel   -> append n "y" "u"
```



```haskell
{-# LANGUAGE LambdaCase      #-}
{-# LANGUAGE PatternSynonyms #-}
{-# LANGUAGE ViewPatterns    #-}

module AndPatterns where

pattern DivBy3 :: Integral i => i
pattern DivBy3 <- ((`mod` 3) -> 0)

pattern DivBy5 :: Integral i => i
pattern DivBy5 <- ((`mod` 5) -> 0)

pattern DivBy7 :: Integral i => i
pattern DivBy7 <- ((`mod` 7) -> 0)

pattern (:&:) :: a -> a -> a
pattern (:&:) i j <- (\x -> (x,x) -> (i,j))

fizzbuzzBop :: Int -> String
fizzbuzzBop = \case
  DivBy3 :&: DivBy5 :&: DivBy7 -> "fizzbuzzbop"
  DivBy3 :&: DivBy5            -> "fizzbuzz"
  DivBy3 :&: DivBy7            -> "fizzbop"
  DivBy5 :&: DivBy7            -> "buzzbop"
  DivBy3                       -> "fizz"
  DivBy5                       -> "buzz"
  DivBy7                       -> "bop"
  i                            -> show i
```

Please, sir
Can I have some ORs?

```
declineObj :: String -> String
declineObj n = case lastVowel n of
  Ve -> append n "y" "i"
  Vi -> append n "y" "i"
  Vö -> append n "y" "ü"
  Vü -> append n "y" "ü"
  Va -> append n "y" "ı"
  Vı -> append n "y" "ı"
  Vo -> append n "y" "u"
  Vu -> append n "y" "u"
```

```
declineObj :: String -> String
declineObj n = case lastVowel n of
  v | v == Ve || v == Vi -> append n "y" "i"
  v | v == Vö || v == Vü -> append n "y" "ü"
  v | v == Va || v == Vı -> append n "y" "ı"
  v | v == Vo || v == Vu -> append n "y" "u"
```

```
declineObj :: String -> String
declineObj n = case lastVowel n of
  Ve | Vi -> append n "y" "i"
  Vö | Vü -> append n "y" "ü"
  Va | Vı -> append n "y" "ı"
  Vo | Vu -> append n "y" "u"
```

Not yet legal Haskell, but see the OrPatterns language proposal.

```
declineObj :: String -> String
declineObj n = case lastVowel n of
  [o| Ve | Vi |] -> append n "y" "i"
  [o| Vö | Vü |] -> append n "y" "ü"
  [o| Va | Vı |] -> append n "y" "ı"
  [o| Vo | Vu |] -> append n "y" "u"
```

This uses the quasiquoter from the `or-patterns` package.

## Newsflash! (Nov 22)

```
declineObj :: String -> String
declineObj n = case lastVowel n of
  one of Ve, Vi -> append n "y" "i"
  one of Vö, Vü -> append n "y" "ü"
  one of Va, Vı -> append n "y" "ı"
  one of Vo, Vu -> append n "y" "u"
```

The OrPatterns proposal has at long last been accepted! But:

- with a different syntax (may change again)
- pattterns that bind a variable not currently supported

## The case for And and Or patterns

## Benefits to And and Or patterns

- no more exponential explosion of cases
- less duplication
- improved readability
- logic more closely matches specification, especially if in tabular form, so discrepencies easier to spot
- less tempting to use a catch-all case

$\Rightarrow$ lower cost and/or defect rate

## Demo