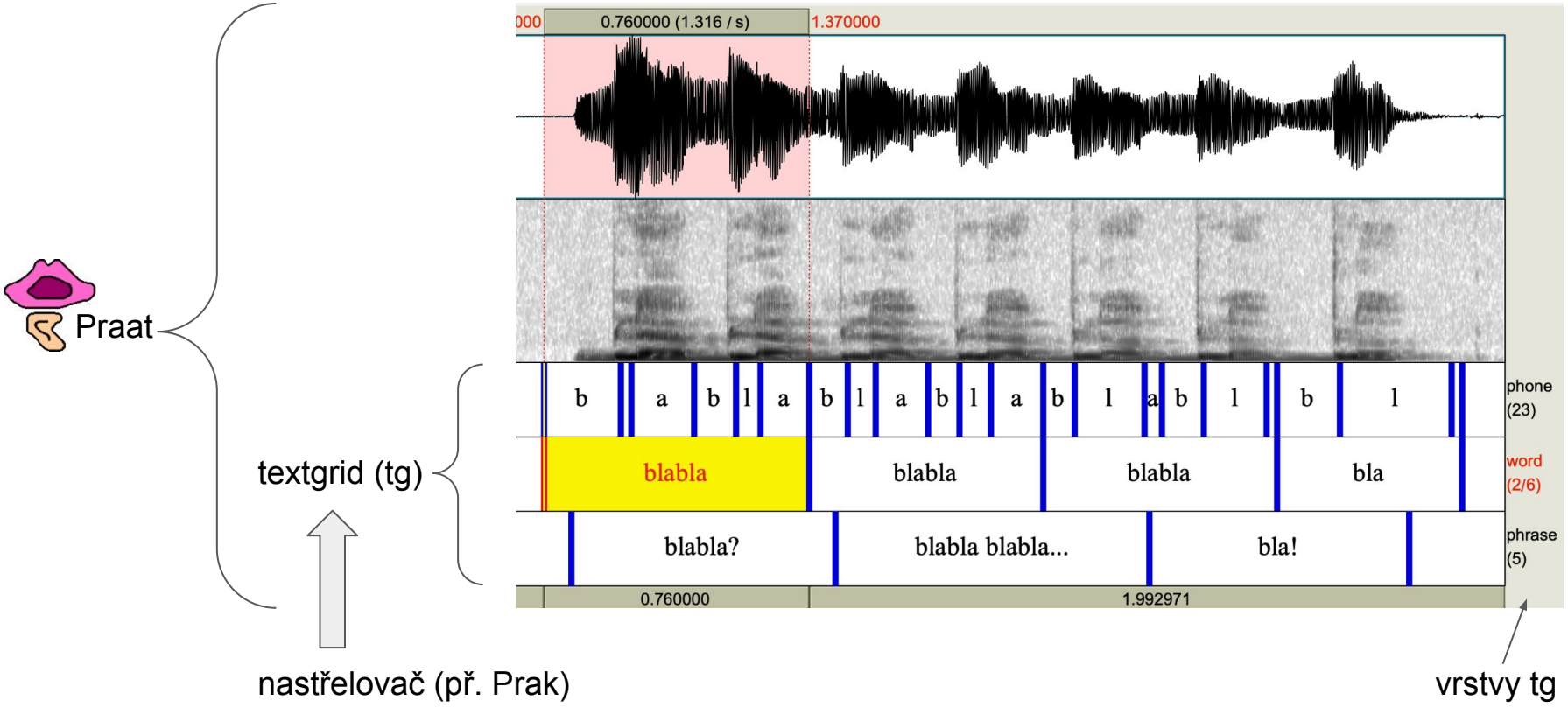
Prak

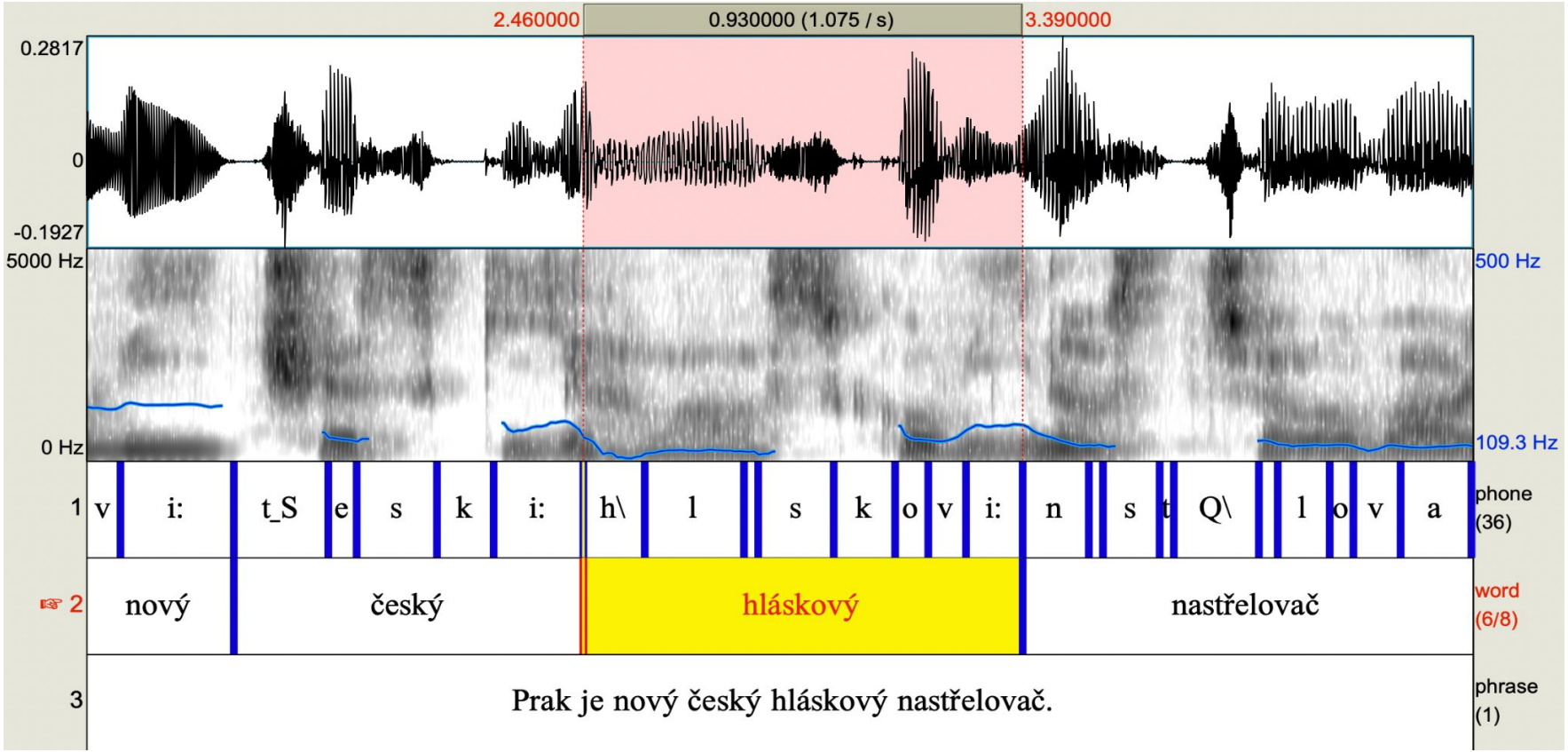
český hláskový nastřelovač

Adléta Hanžlová & Václav Hanžl

Pojmosloví


Praat
textgrid (tg)
nastřelovač (př. Prak)
000
0.760000 (1.316 / s)
1.370000
b a b l a b l a b l a b l a b l b l
blabla
blabla
blabla
bla
blabla?
blabla blabla...
bla!
phone (23)
word (2/6)
phrase (5)
0.760000
1.992971
vrstvy tg

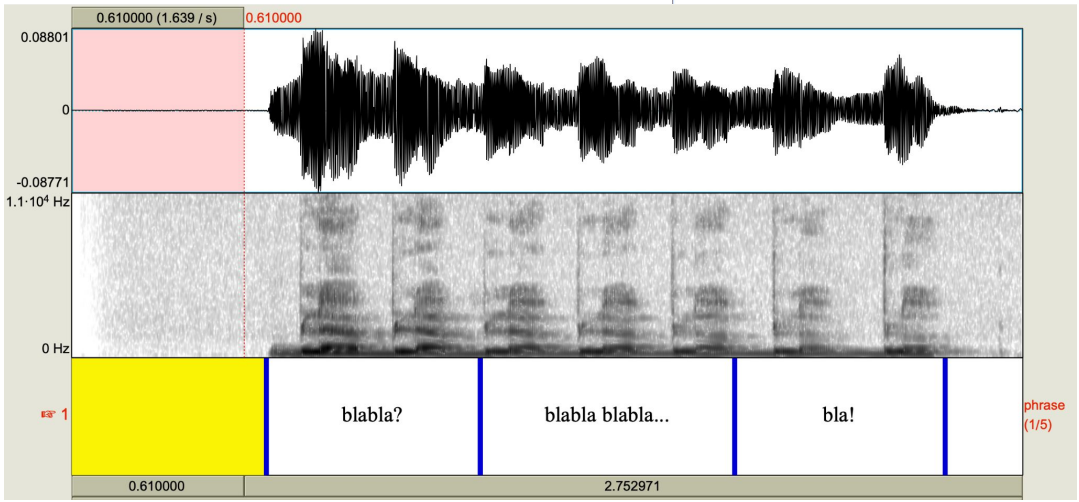
Nastřelovač: nahrávka, text ⇒ časově zarovnané hlásky

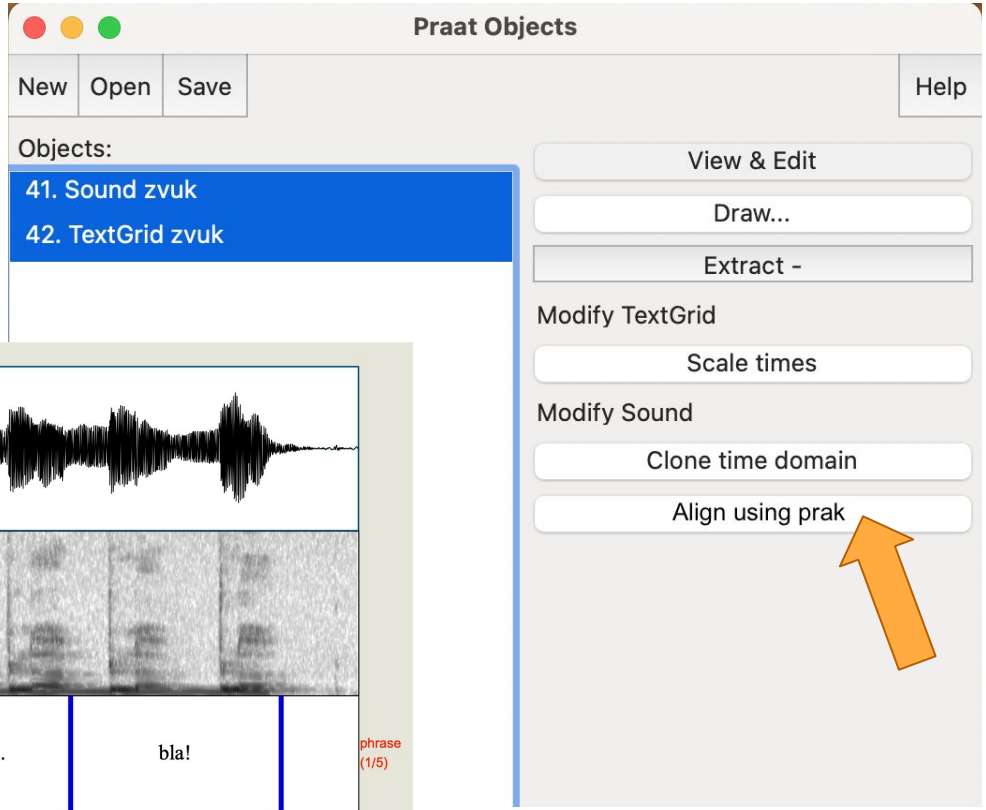


Jak použít prak?

1) nainstalovat
2) spustit

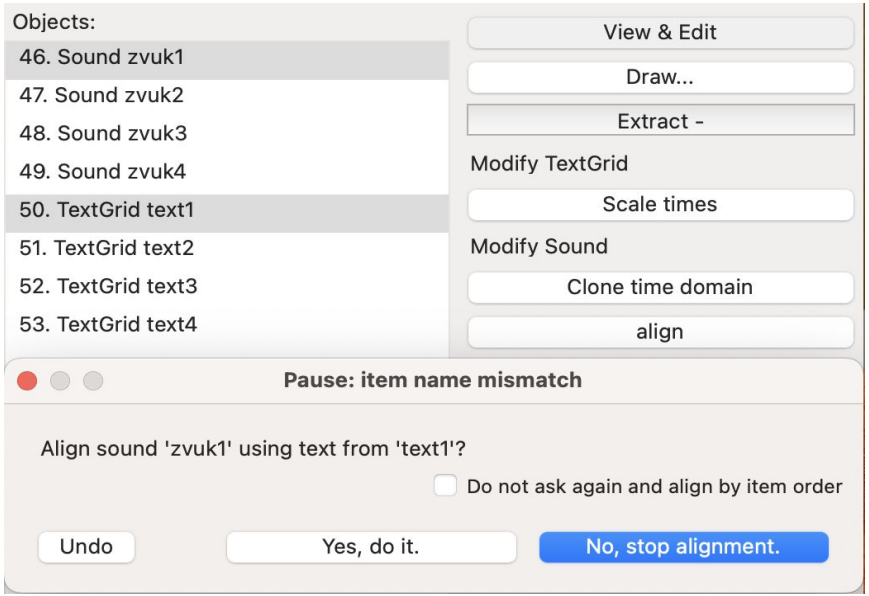
vstup: zvuk a tg s textem

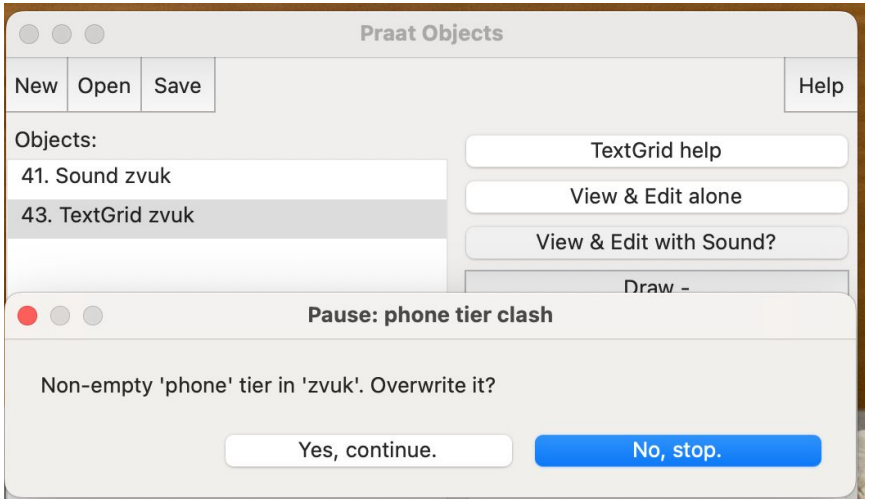




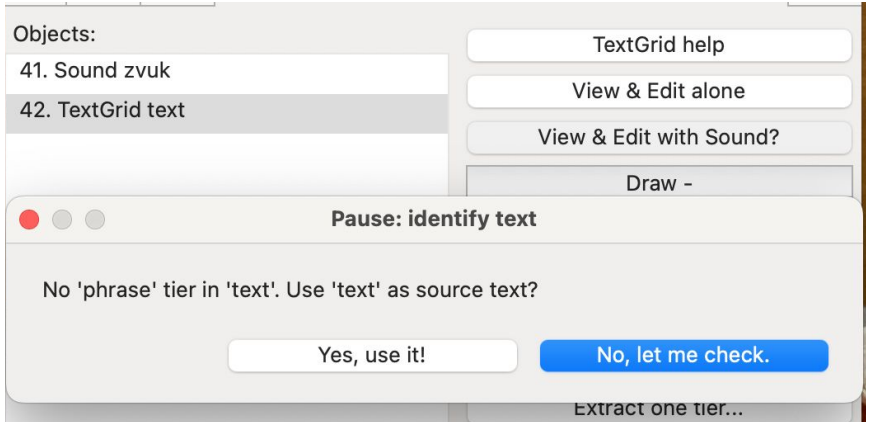
Kontroly před nastřelením

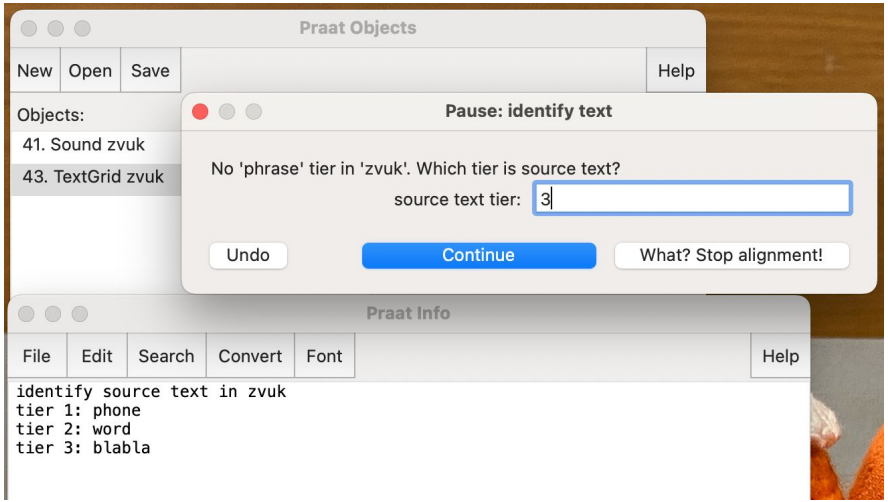
názvy položek, kontrola nastřelenosti





Identifikace správné vrstvy v textgridu

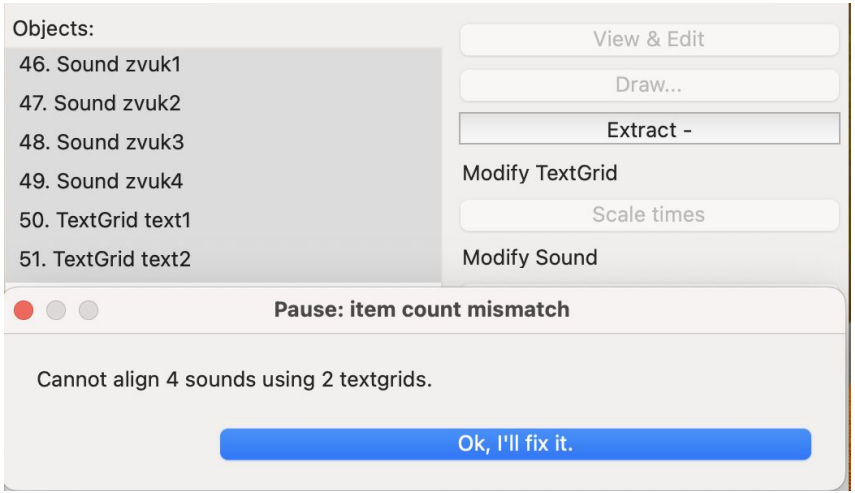


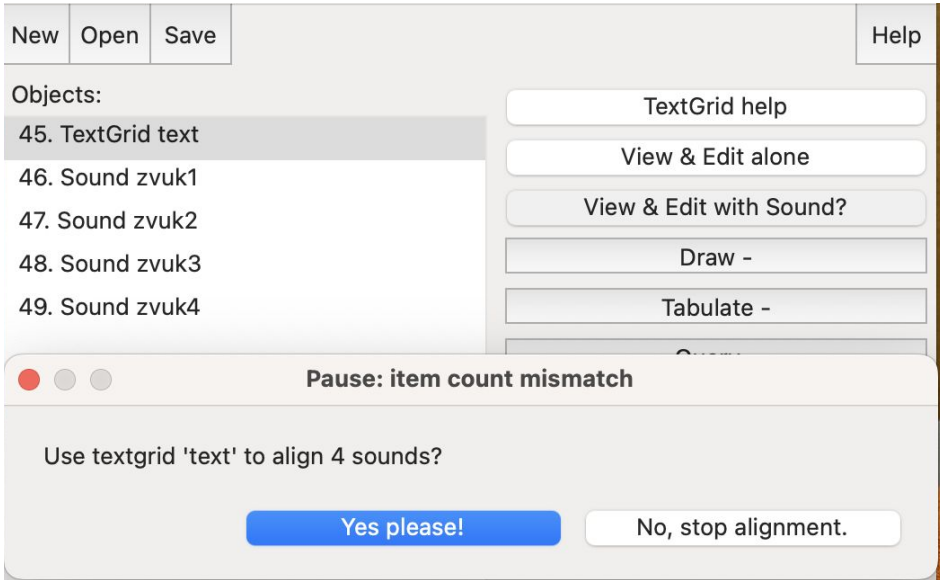


všechny vrstvy kromě vstupního textu budou smazány

Kontrola počtu označených zvuků a textgridů

Nesedí počet → :-(





ALE 1 tg a více zvuků → funguje! :-D

výstup: ke každému zvuku tg se stejným názvem

Výslovnost - prak_prongen.py

```
prak/prongen$ ./prak_prongen.py -p
--- reading stdin, printing possible pronunciations ---
asi jsem shodil sedm piv

          zh
ʔasɪ .sem sxoďɪl sed.m pɪf
     j              u

slyšel jsem asi galantní zpěv ptáků

           |ʔ
           _ʔ
slɪšel .sem_ asɪ galantňi: spjef pta:ku:
       j             nť
                     ňť

v USA pneumatiky za eura nekoupíš

      ʔ         eu
f=ʔu:=.es=.a: pn͡eumatɪkɪ za_ e͡ura nek͡oupi:š
v=        ʔ                _ʔ
                           |ʔ
```

```
Nic než oheň

        ž_
        š|ʔ
        š_ʔ
ňɪd͡z_neš_ oheň
  c_
  c|
```

```
to jsou úřední dokumenty

       |ʔ
       _ʔ
to .sóu̯_ u:řeďňi: dokumentɪ
   j         ď

nashromáždilo se to na tom se shodneme už dnes

                                    zh          ž_
na=sxroma:žďɪlo se to na tom se sxodneme_ uš|dnes
   zh                                      _ʔ
                                           |ʔ
```

Cizí slova v exceptions.txt

```
### exceptions.txt - additional pronunciation rules, see https://github.com/vaclavhanzl/prak
###
### Comment lines start with THREE hashes: "###".
### Longer match wins, order mostly does not matter. Write it the "normal" Czech way:
interview intervjů

### Help with ditini where needed, use "ň", "y" etc. (but it should be 98% correct by default):
nitrolak nytrolak

### Suggest seam. This rule helps for videoukázka, supervideoukázky etc.
videoukázk video=ukázk
sirouhlík siro=uhlík

### Use "+" for word begin/end. This rule wil NOT match e.g. "euroombudsman":
+room+ rům

### This rule allows pronunciations "jsem" and "sem", "jsi" and "si" etc. Can use variants! :)
+js s js

+nation+ nejšn nasijon

### REMEMBER: This file must be saved as utf-8. NOT utf-16, NOT latin-XXX.
```

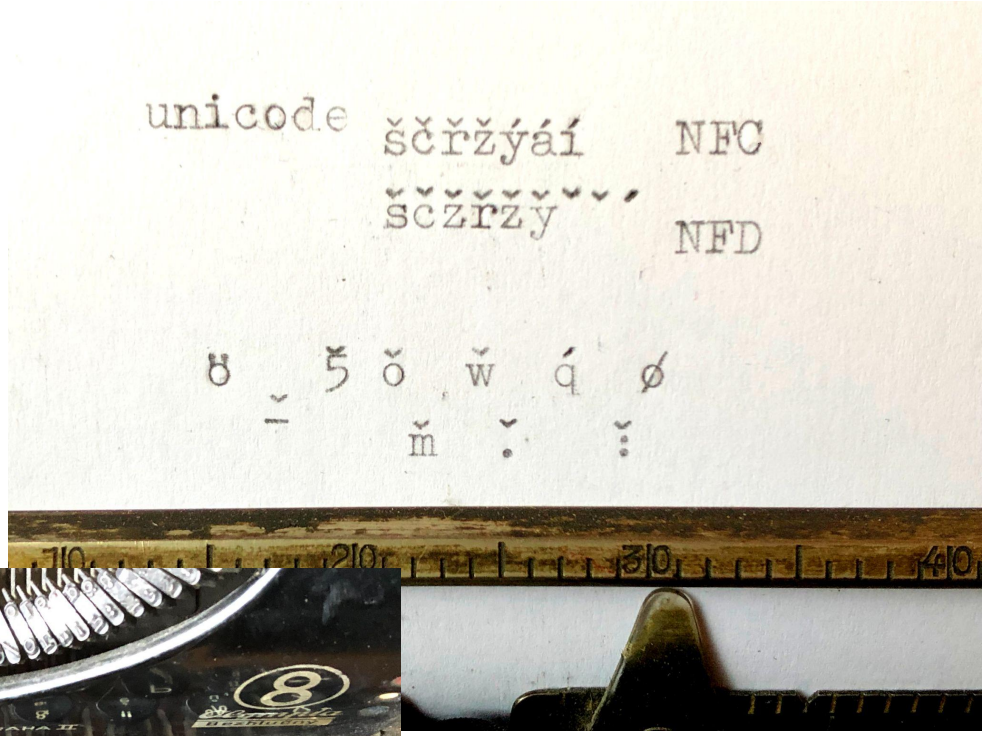
Co se skrývá v odhadu výslovnosti

- normalizace a čištění textu
- připojení některých předložek (od, bez, nad, pod, před, ob, z, v, s, u, k)
  - “bez nám voní bez nároku na honorář” - šev má trochu jiné vlastnosti
- “počeštění” cizích slov dle vnitřních pravidel (i “ditini”) a exceptions.txt
  - **ti**ká, ale automa**tyc**ky, ale **Valtic**ký... automaticky generovaná sada pravidel a výjimek z výjimek
- drobné následné asimilace znělosti (“ř”, počáteční “sh”)
- “logická” část zpětných vlivů (v podstatě konvoluce překladových automatů)
  - ráz, intervokalické “j”
  - asimilace znělosti
  - dtn/ďťň skupin jako “ntní”, bě/pě/vě/mě/fě
  - velarizace nk/ng
- drobné hláskové změny (např. zdvojené hlásky)
  - šš, ss, ts/c, tš/č... - většinou generovány obě možnosti
- optimalizace výsledného grafu (pro přehlednou prezentaci i pro výpočty)

Normalizace vstupního textu

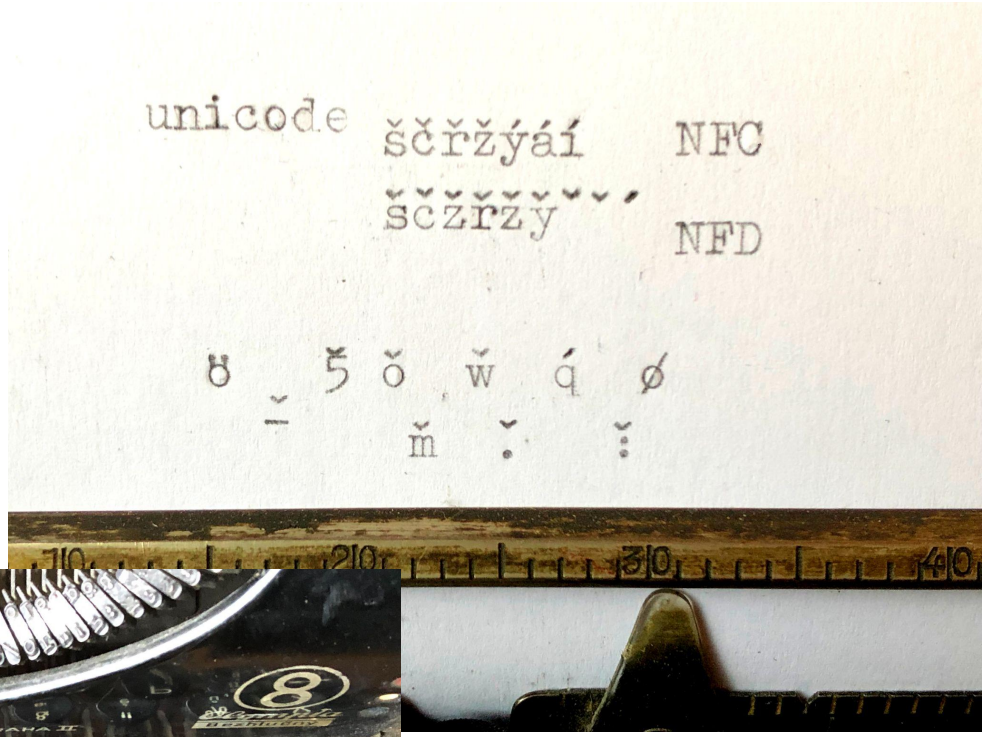
- převést do NFC (“composed”, “č” je jeden znak, ne “c” a háček)
- odstranit CR, LF (konce řádků, LF v Unixu, CR+LF na Windows)
- odstranit BOM (neviditelný znak)
- odstranit interpunkci: " ‘ . , ? ! „ “ - = – : ; — / _ | ~


Olympia


unicode ščřžýáí NFC
ščřžý NFD


OLYMPIA PSACÍ STROJE AKC. SPOL. PRAHA II
E R T Z U I O P
D F G H J K L
VYP
UZÁV

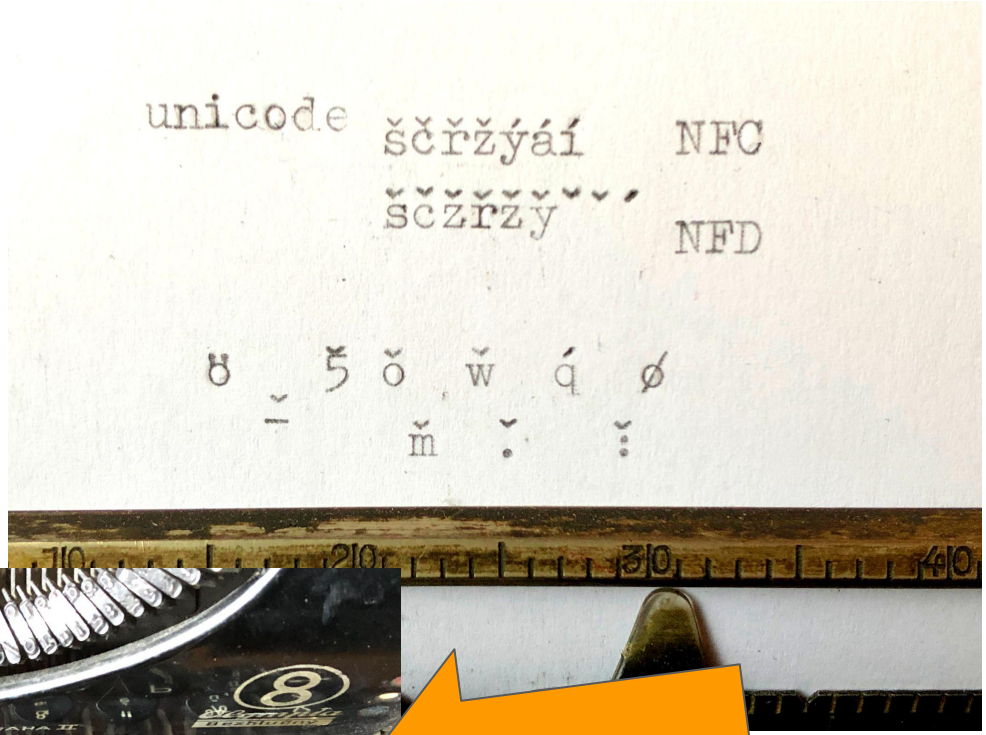

Olympia


unicode
ščřžýáí
NFC
ščřžý
NFD


OLYMPIA PSACÍ STROJE AKC. SPOL. PRAHA II
E R T Z U I O P
D F G H J K L
VYP
UZÁV

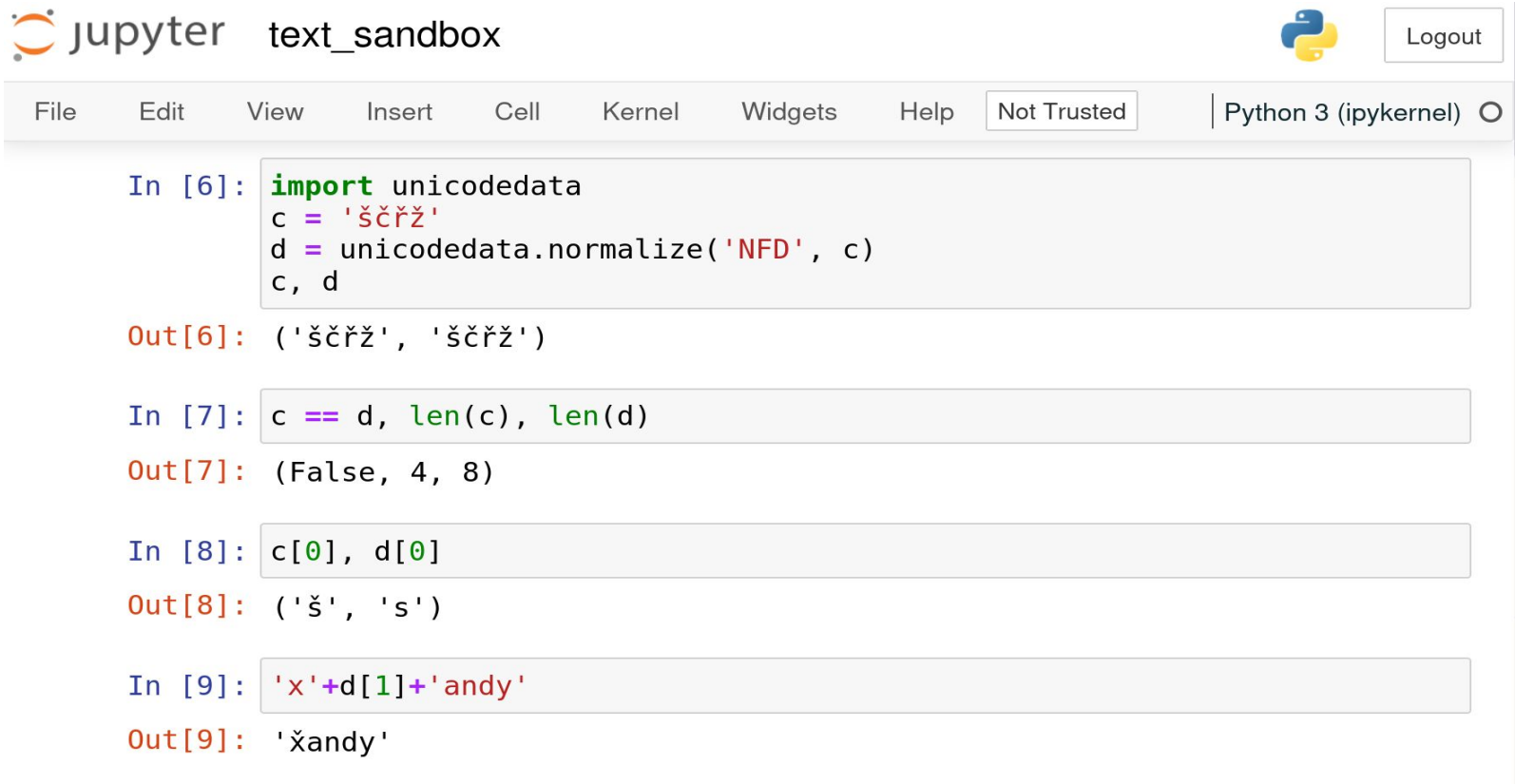
CR+LF


Olympia

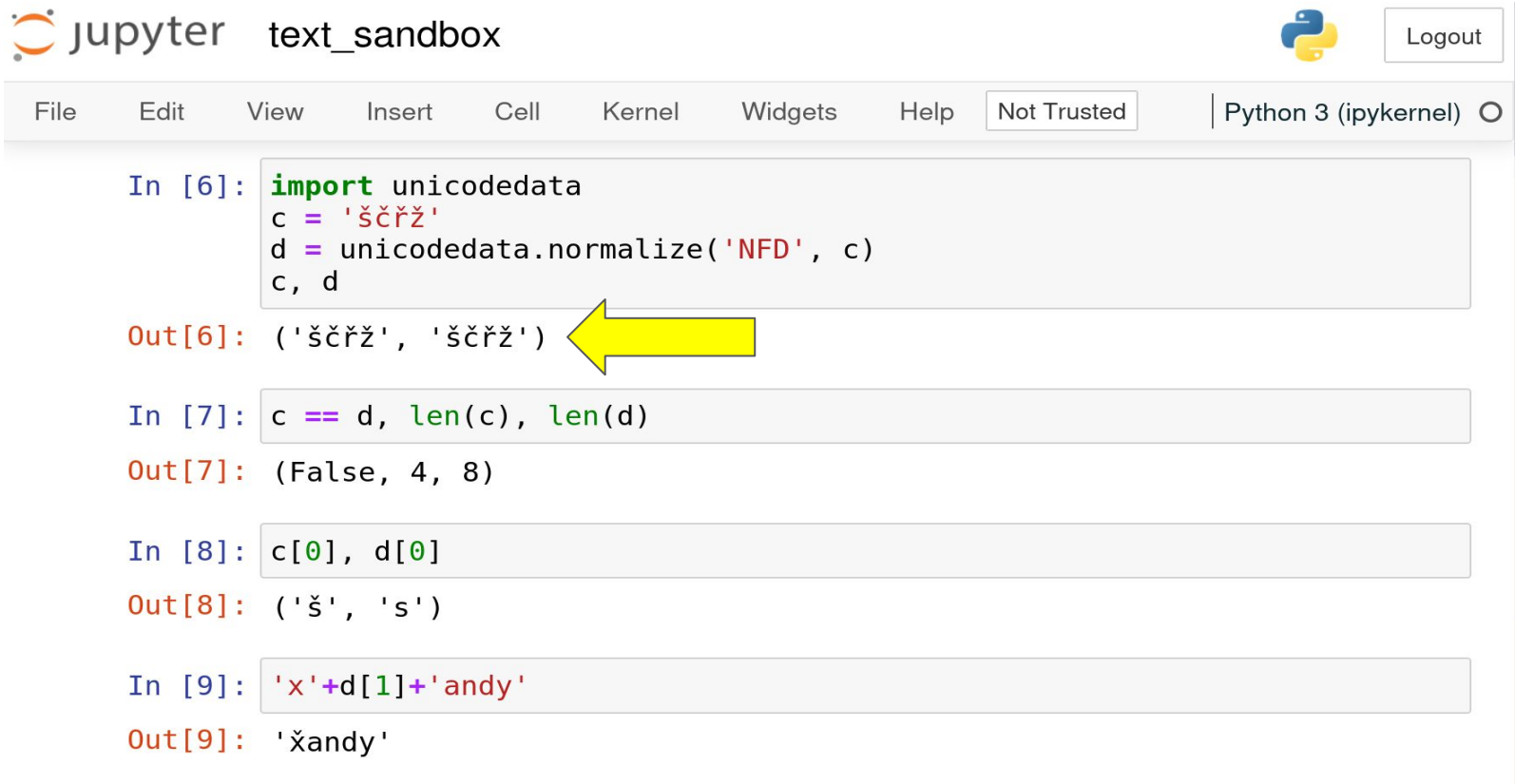

unicode
ščřžýáí
NFC
ščřžý
NFD


OLYMPIA PSACÍ STROJE AKC. SPOL. PRAHA II
E R T Z U I O P
D F G H J K L
VYP
UZÁV

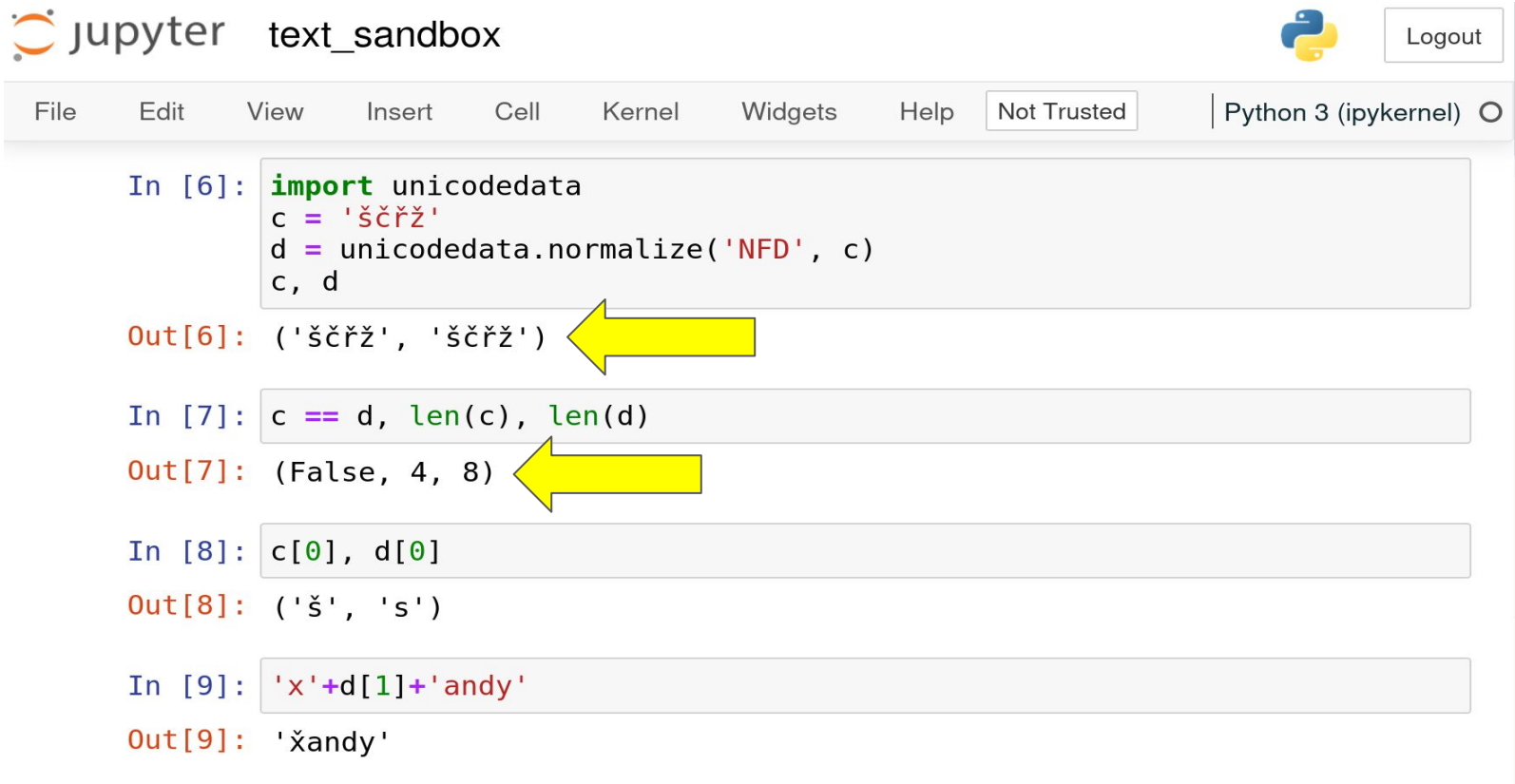
Unicode


jupyter
text_sandbox
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
In [6]:
import unicodedata
c = 'ščřž'
d = unicodedata.normalize('NFD', c)
c, d
Out[6]:
('ščřž', 'ščřž')
In [7]:
c == d, len(c), len(d)
Out[7]:
(False, 4, 8)
In [8]:
c[0], d[0]
Out[8]:
('š', 's')
In [9]:
'x'+d[1]+'andy'
Out[9]:
'x̌andy'

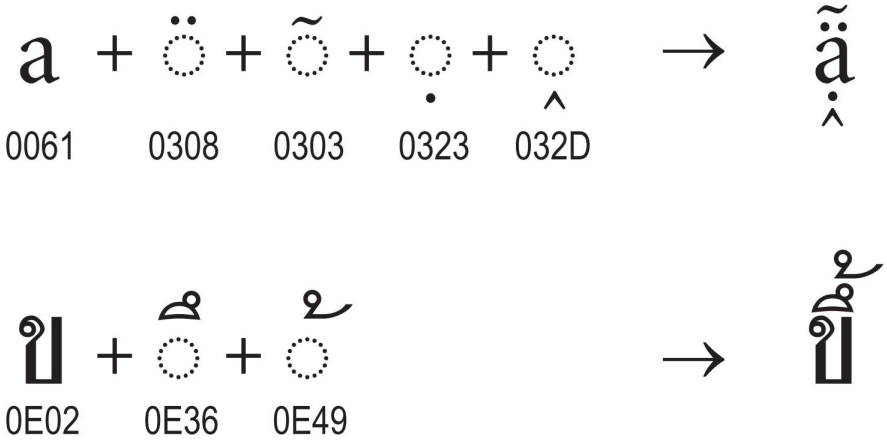
Unicode - NFC a NFD verze textu vypadají stejně

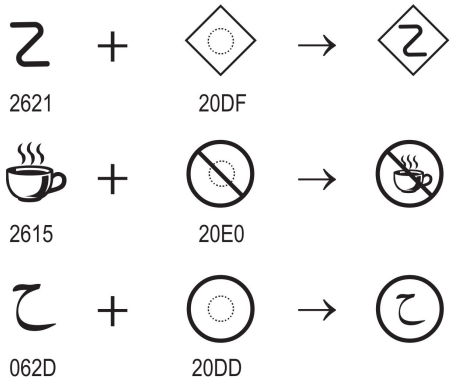


Unicode - NFC a NFD verze textu se nerovnají



Unicode - další znaménka (odměna za NFC/NFD bugy)





zdroj obrázků: Unicode Standard, “čepička” a “lasička” vlevo jsou thajský znak KHO KHAI + SARA UE + MAI THO, vpravo arabské HAH

Výslovnost - “logická” část zpětných vlivů (v podstatě konvoluce překladových automatů)

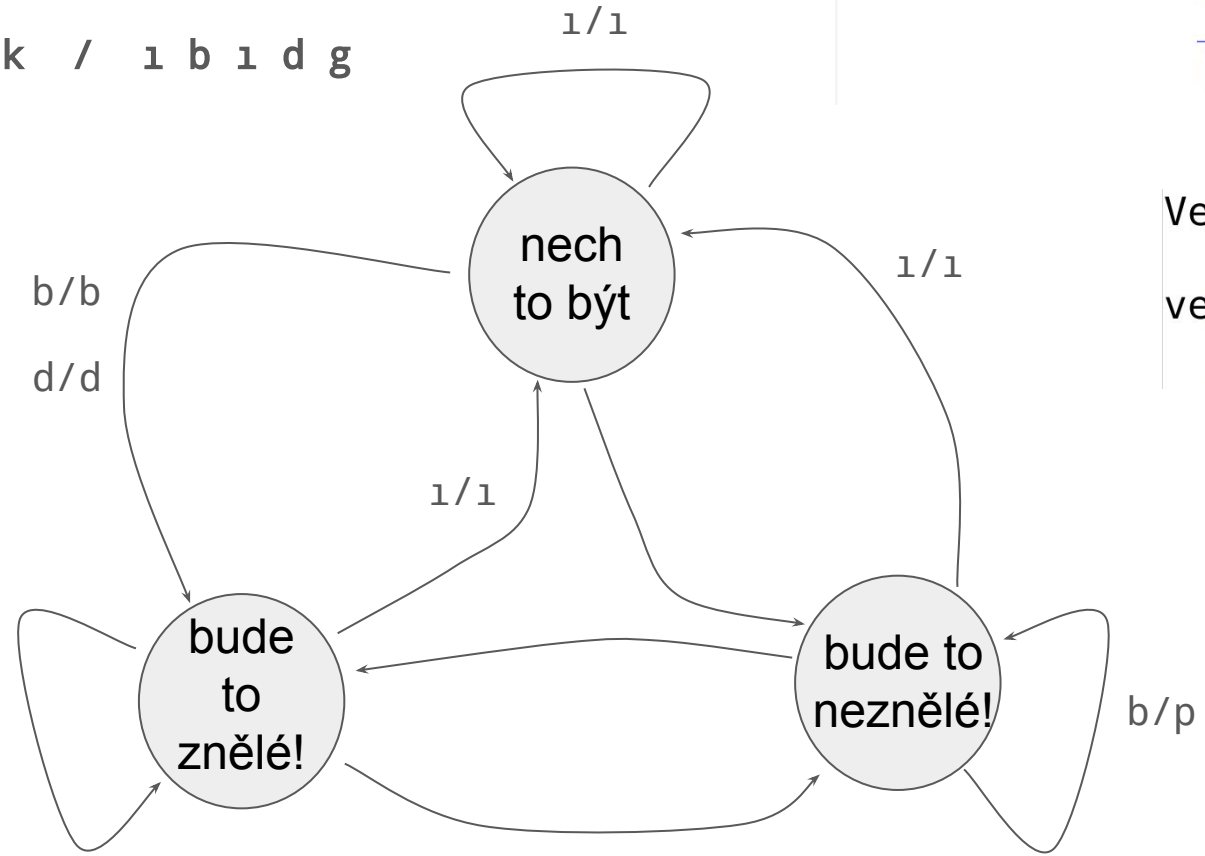
- ráz, intervokalické “j” (pavián ⇒ paviján)
- **asimilace znělosti**
- dtn/ďťň skupiny jako “ntní”, bě/pě/vě/mě/fě
- velarizace nk/ng

Zpětné asimilace

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

ɪ b ɪ d k / ɪ b ɪ d g



Vepř byl vypečený

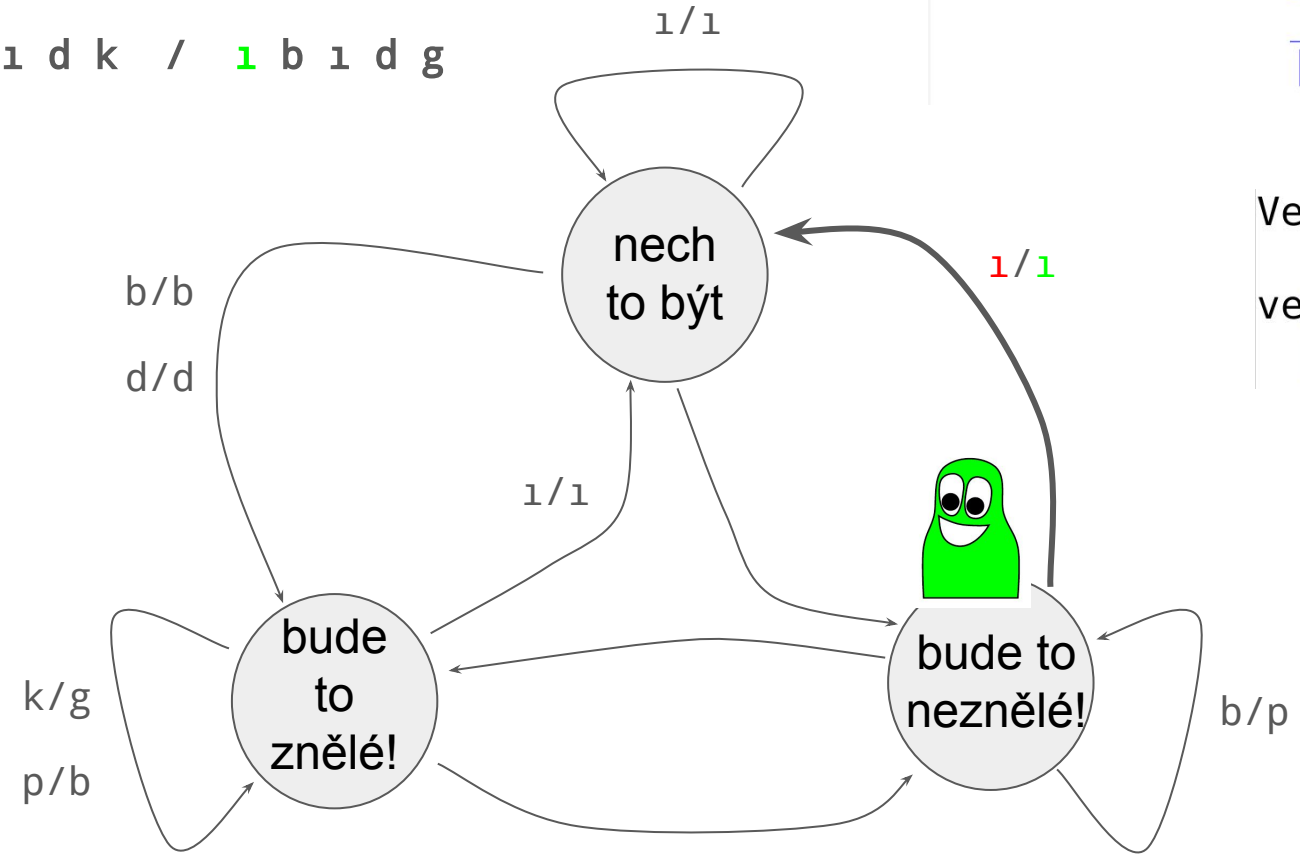
vebř_bɪl vɪpečeni:
př|

Zpětné asimilace

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

ɪ b ɪ d k / ɪ b ɪ d g



Vepř byl vypečený

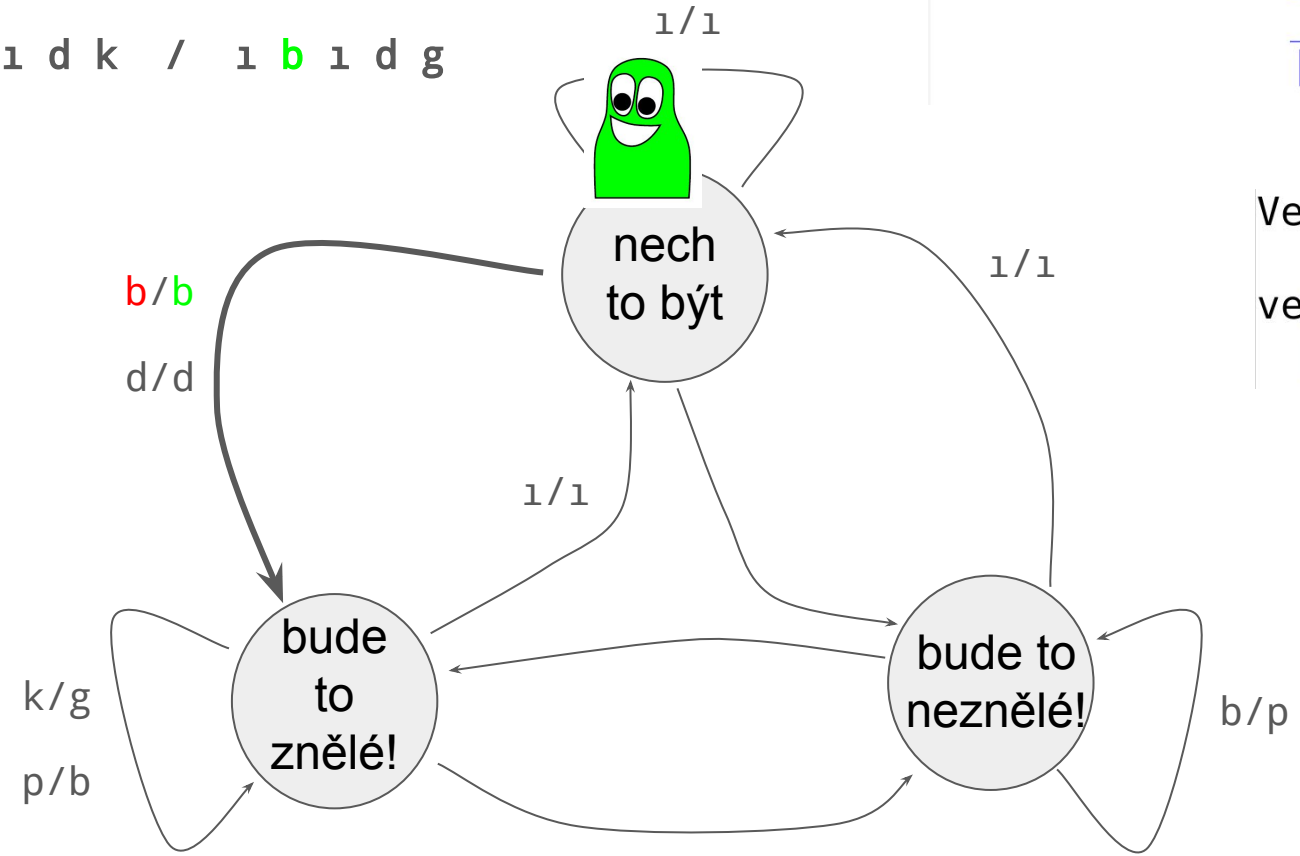
vebř_bɪl vɪpečeni:
př|

Zpětné asimilace

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

ɪ b ɪ d k / ɪ b ɪ d g



Vepř byl vypečený

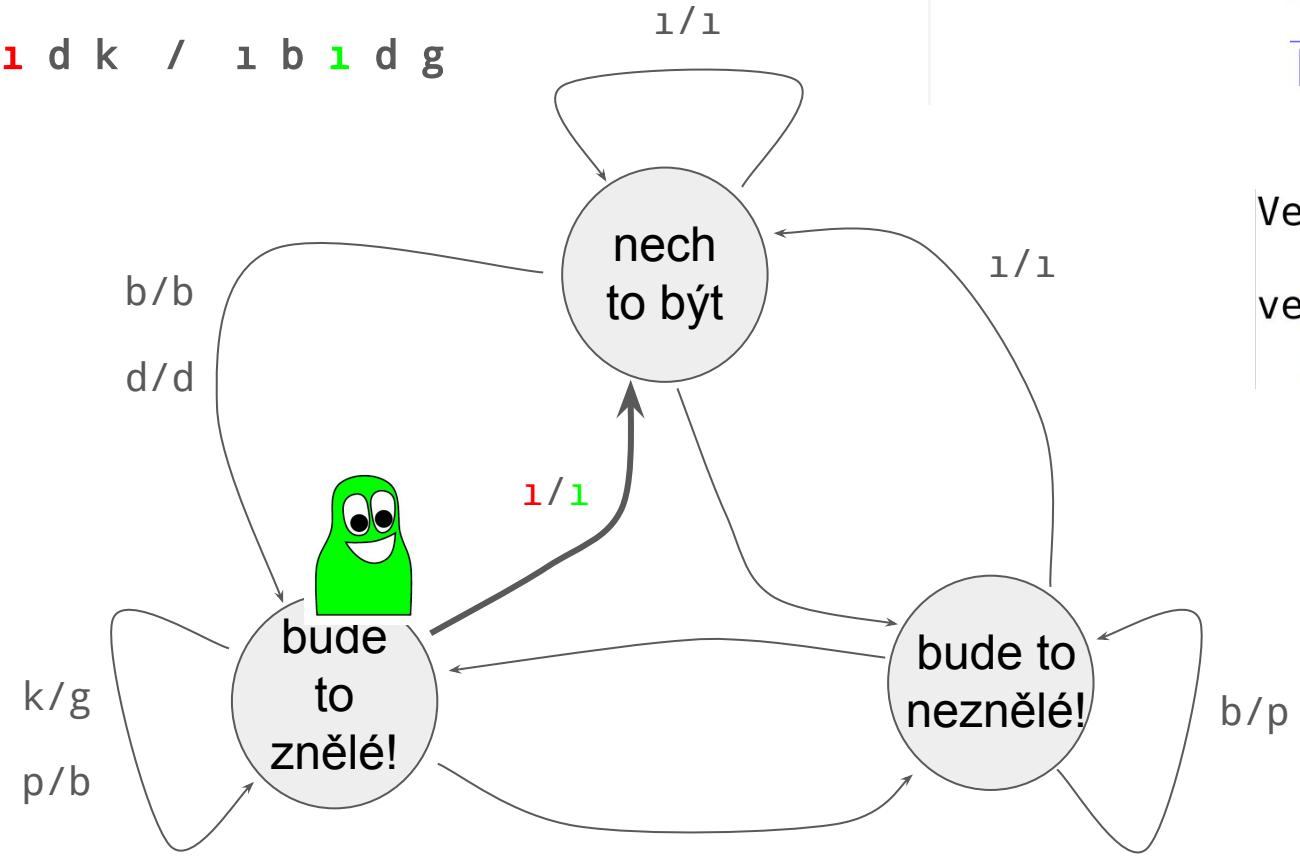
vebř_bɪl vɪpečeni:
př|

Zpětné asimilace

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

ɪ b ɪ d k / ɪ b ɪ d g



Vepř byl vypečený

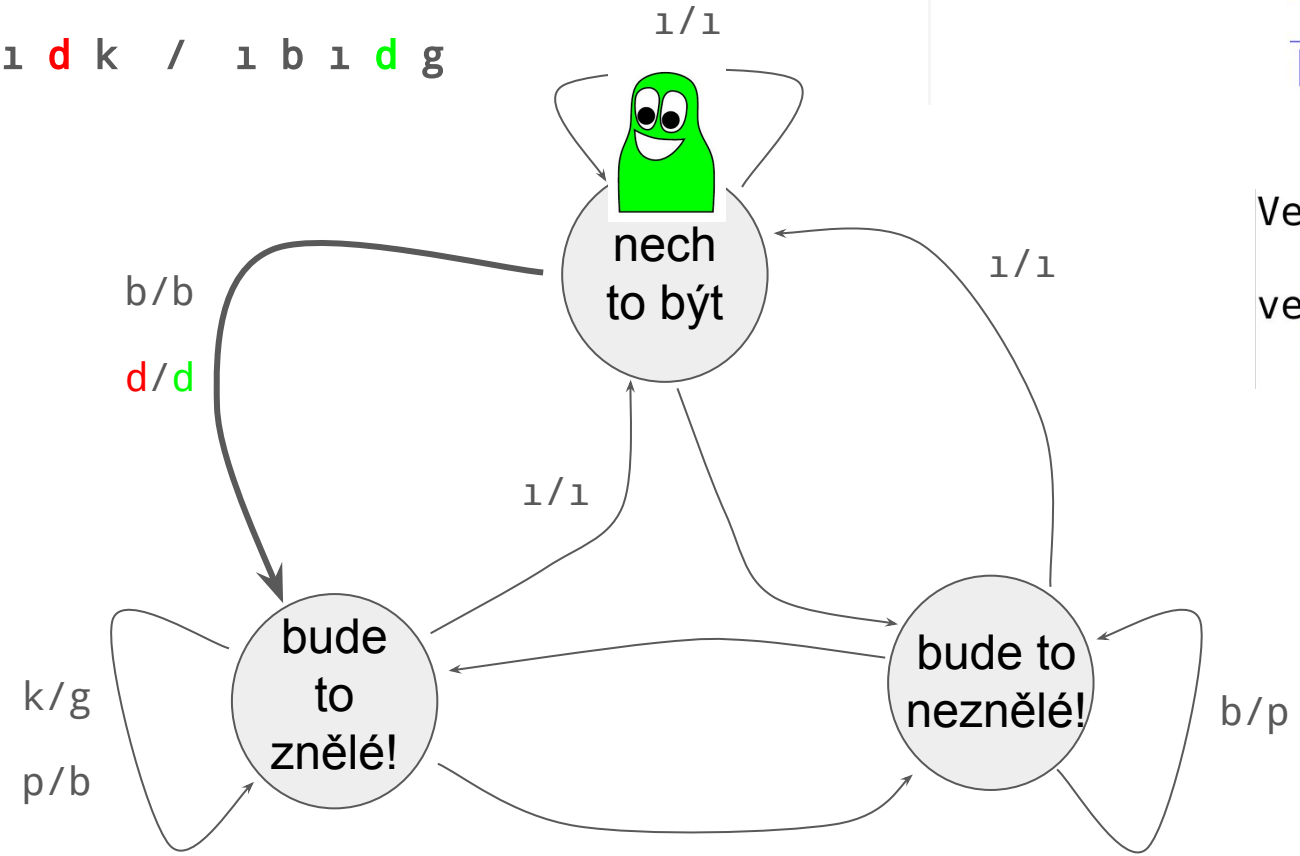
vebř_bɪl vɪpečeni:
př|

Zpětné asimilace

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

ɪ b ɪ d k / ɪ b ɪ d g



Vepř byl vypečený

vebř_bɪl vɪpečeni:
př|

Zpětné asimilace

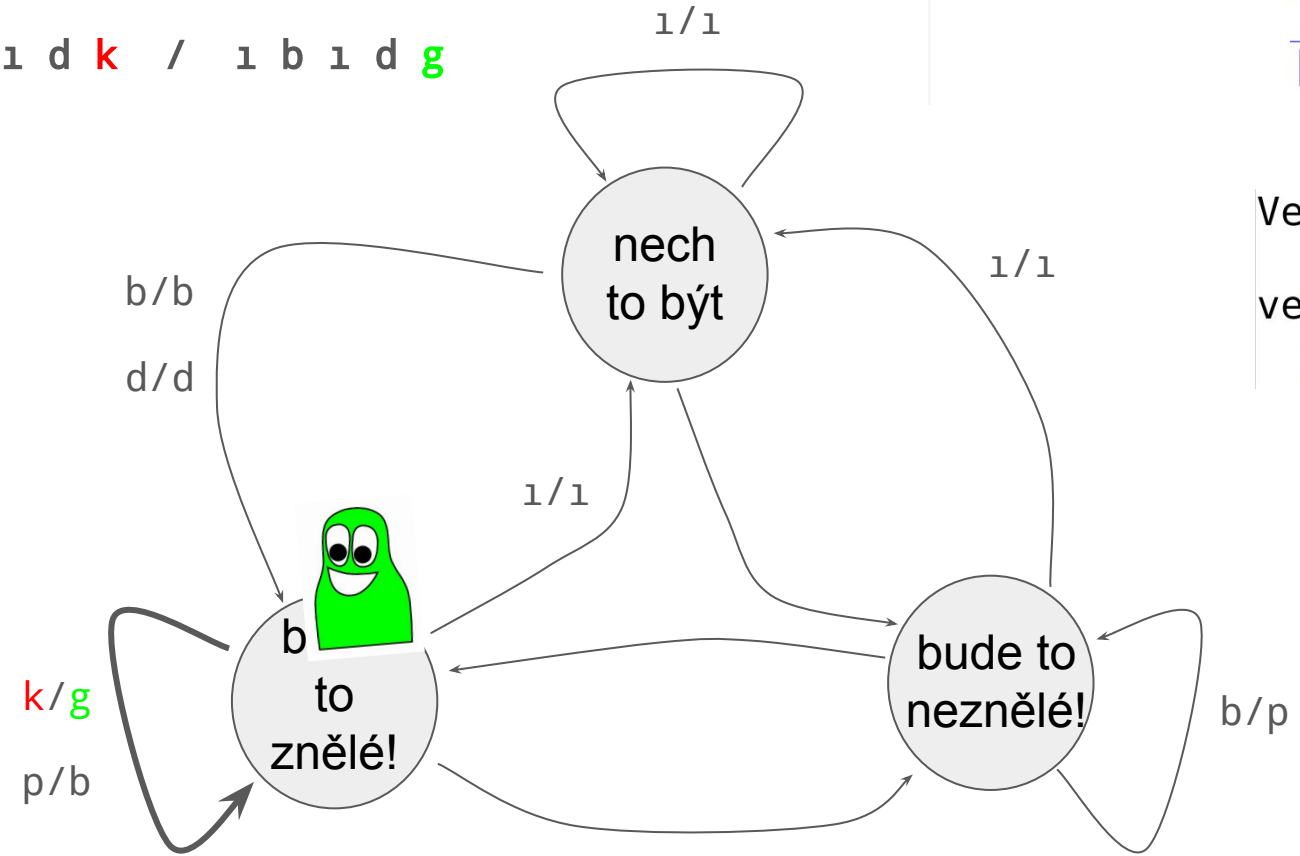
Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

Vepř byl vypečený

vebř_bɪl vɪpečeni:
př|

ɪ b ɪ d k / ɪ b ɪ d g

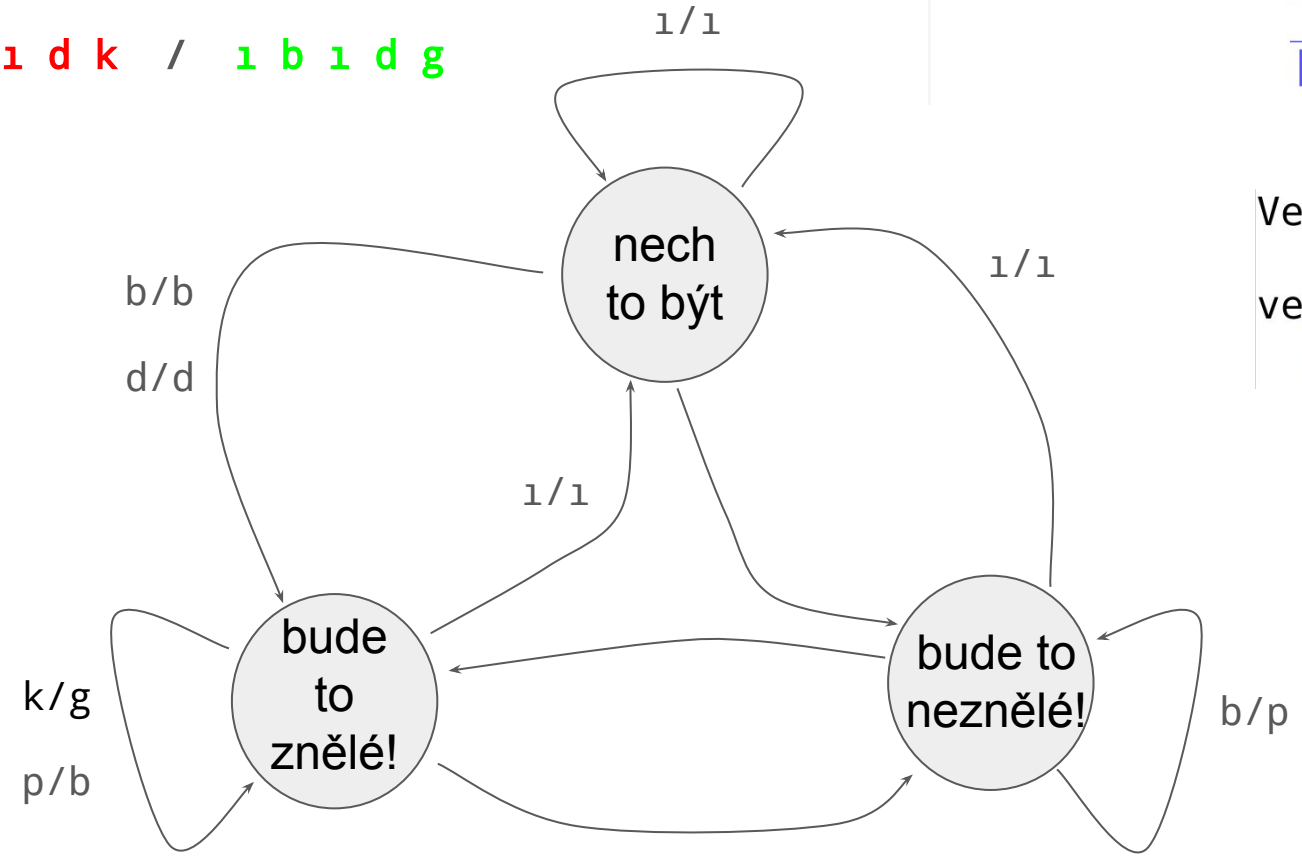


Zpětné asimilace

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

ɪ b ɪ d k / ɪ b ɪ d g



Vepř byl vypečený

vebř_bɪl vɪpečeni:
př|

Zpětné asimilace

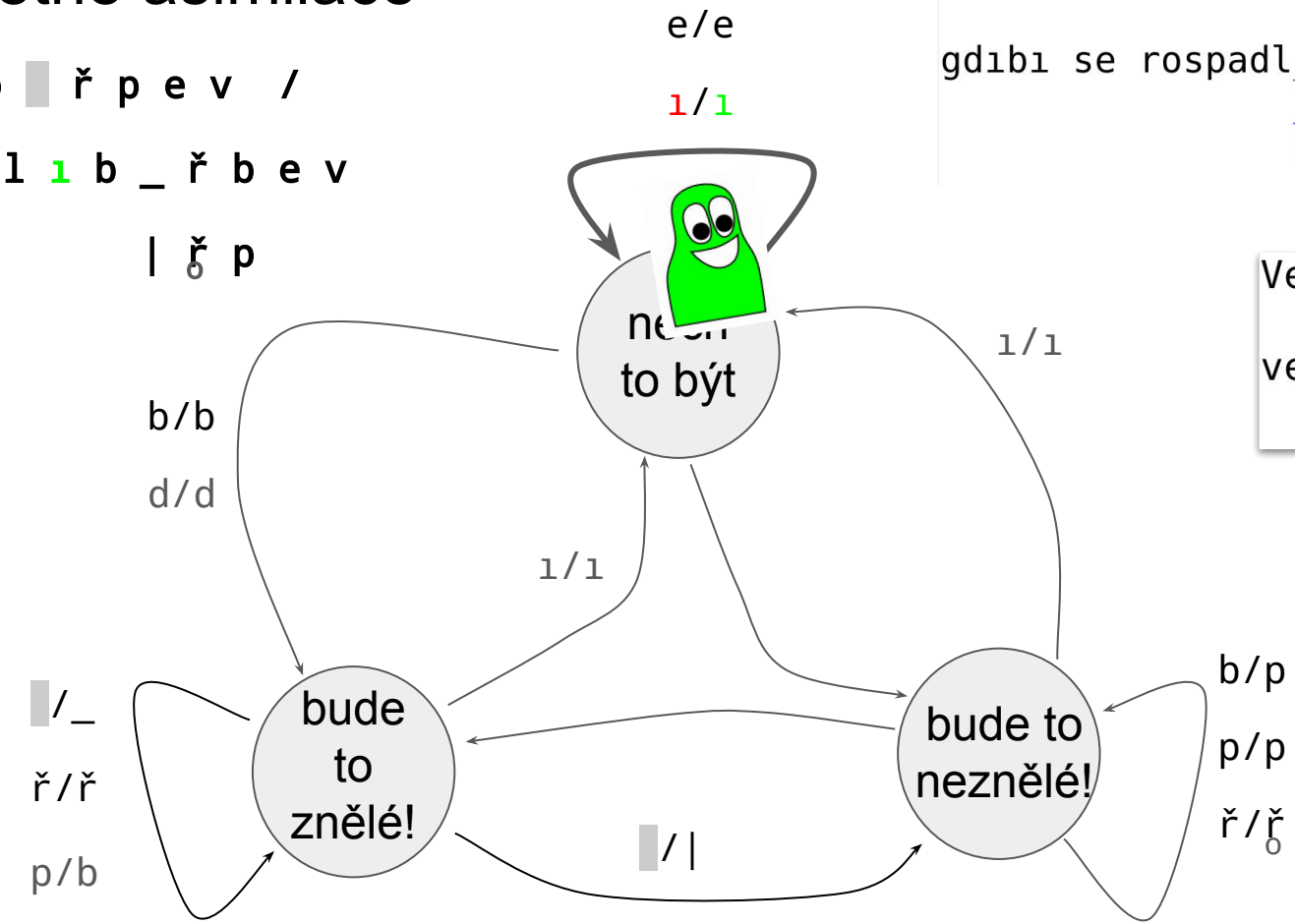
l ɪ b ▒ ř p e v / l ɪ b _ ř b e v | ř p

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

Vepř byl vypečený

vebř_bɪl vɪpečeni:
př̥|



_ ... spojovník

| ... rozdělovník

▒ ... obojí možné

Zpětné asimilace

l ɪ b ▒ ř p e v / /

l ɪ b _ ř b e v

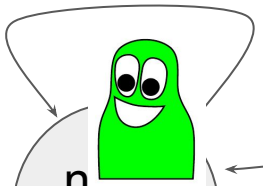
| ř p
o

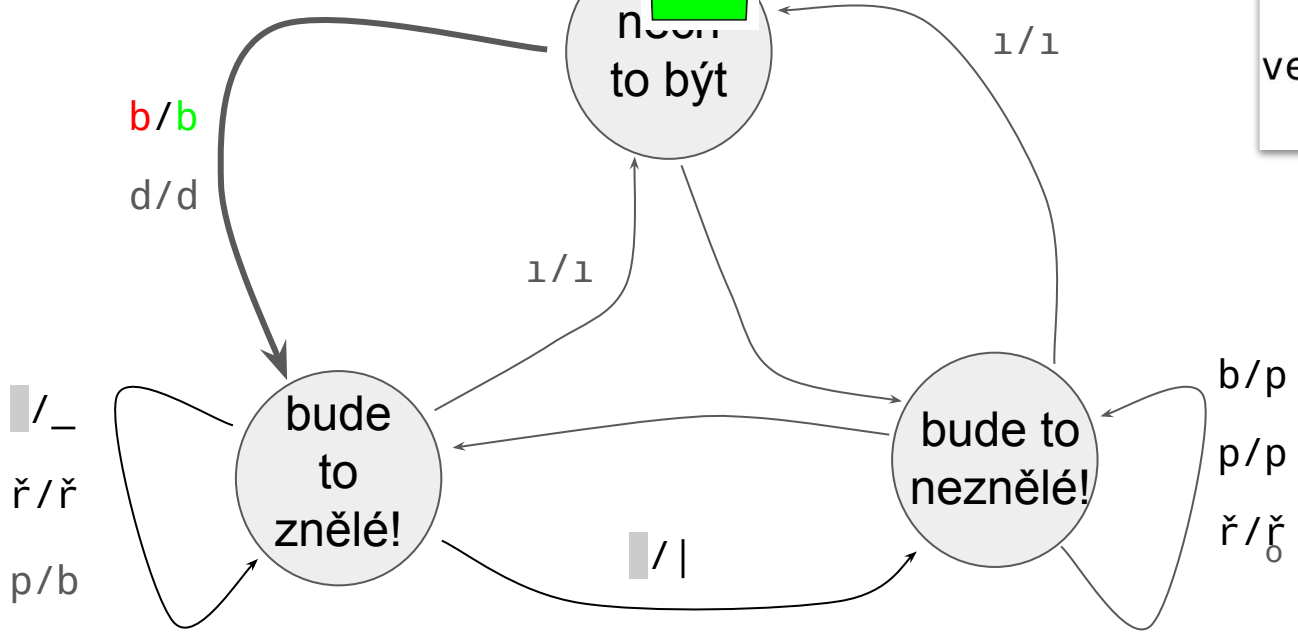
Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

Vepř byl vypečený

vebř_bɪl vɪpečeni:
př̥|





_ ... spojovník

| ... rozdělovník

▒ ... obojí možné

Zpětné asimilace

l ɪ b ▮ ř p e v /

l ɪ b _ ř b e v

| ř̥ p

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel

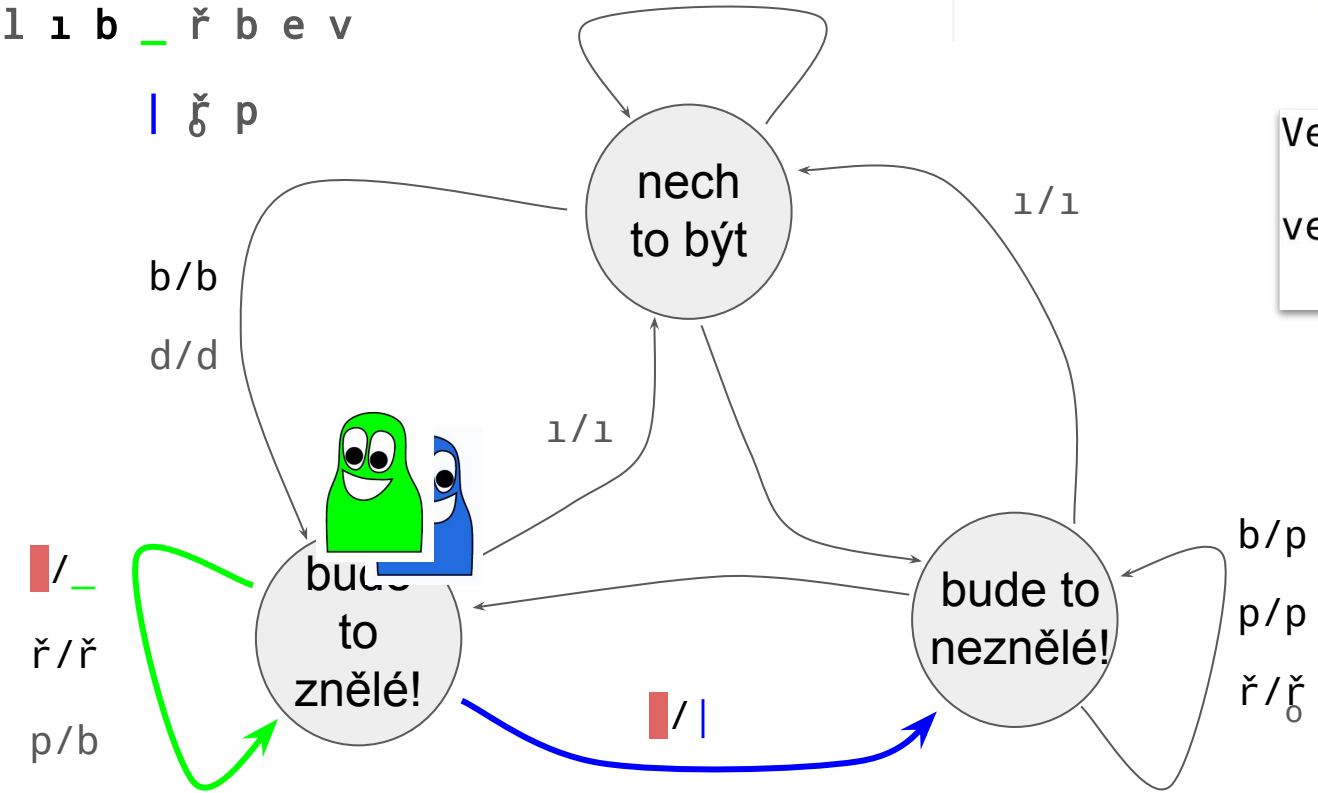
_?

|?

Vepř byl vypečený

vebř_bɪl vɪpečeni:

př̥|



_ ... spojovník

| ... rozdělovník

▮ ... obojí možné

Zpětné asimilace

l ɪ b ▒ ř p e v /

l ɪ b _ ř b e v

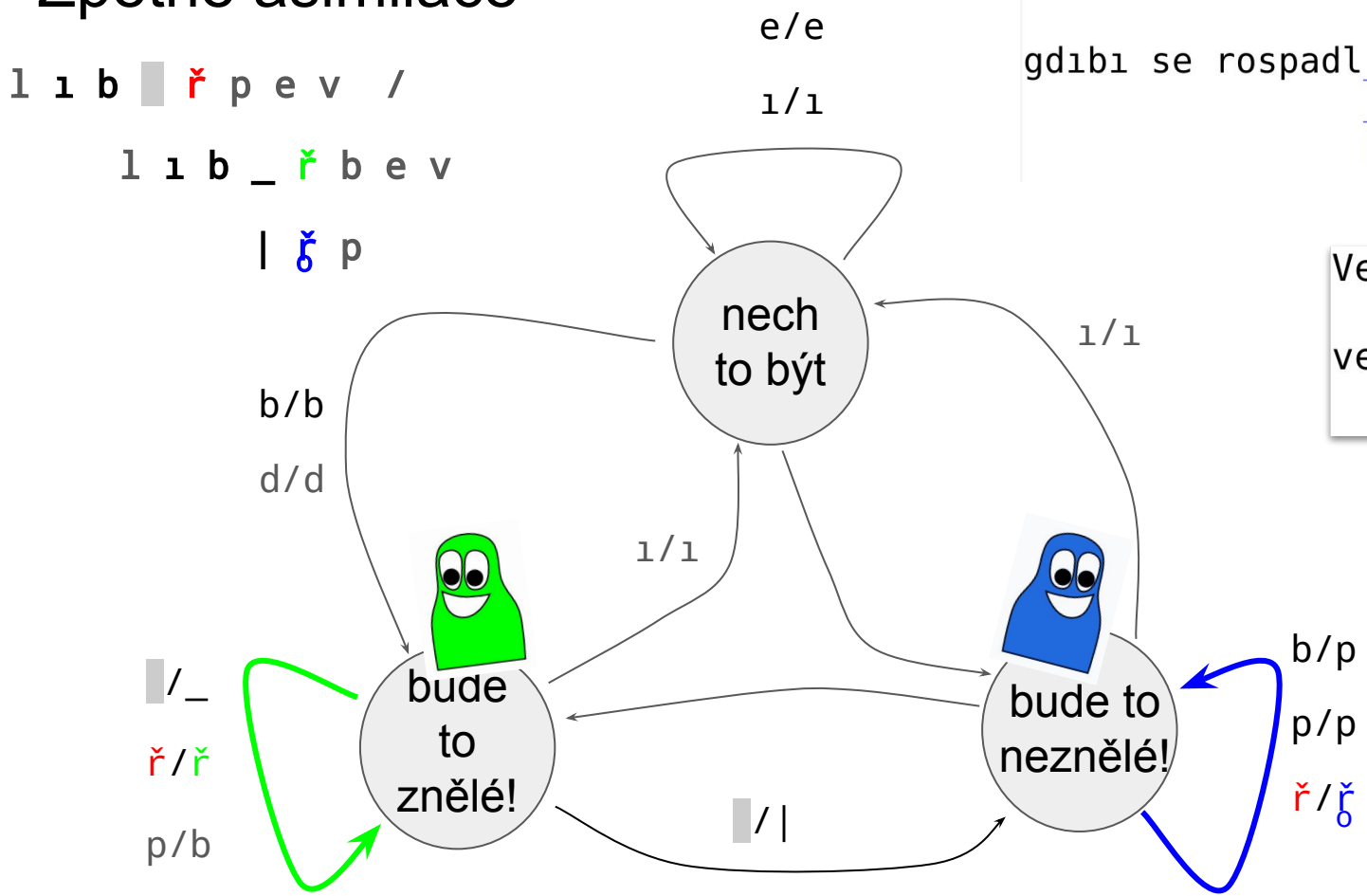
| ř̥ p

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel
_?
|?

Vepř byl vypečený

vebř_bɪl vɪpečeni:
př̥|



_ ... spojovník

| ... rozdělovník

▒ ... obojí možné

Zpětné asimilace

l ɪ b ▒ ř p e v / /

l ɪ b _ ř b e v

| ř̥ p

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel

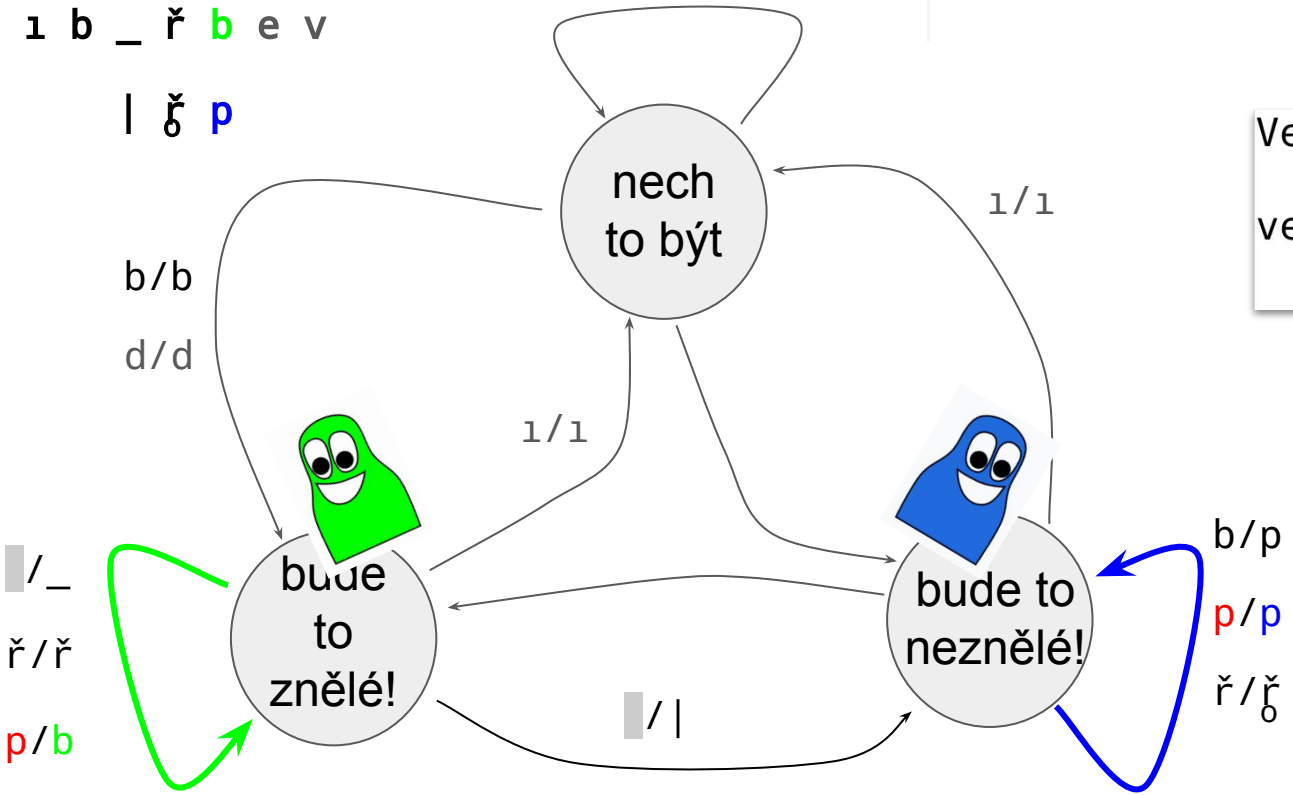
_?

|?

Vepř byl vypečený

vebř_bɪl vɪpečeni:

př̥|



_ ... spojovník

| ... rozdělovník

▒ ... obojí možné

Zpětné asimilace

l ɪ b ▒ ř p e v /

l ɪ b _ ř b e v

| ǩ p e

Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel

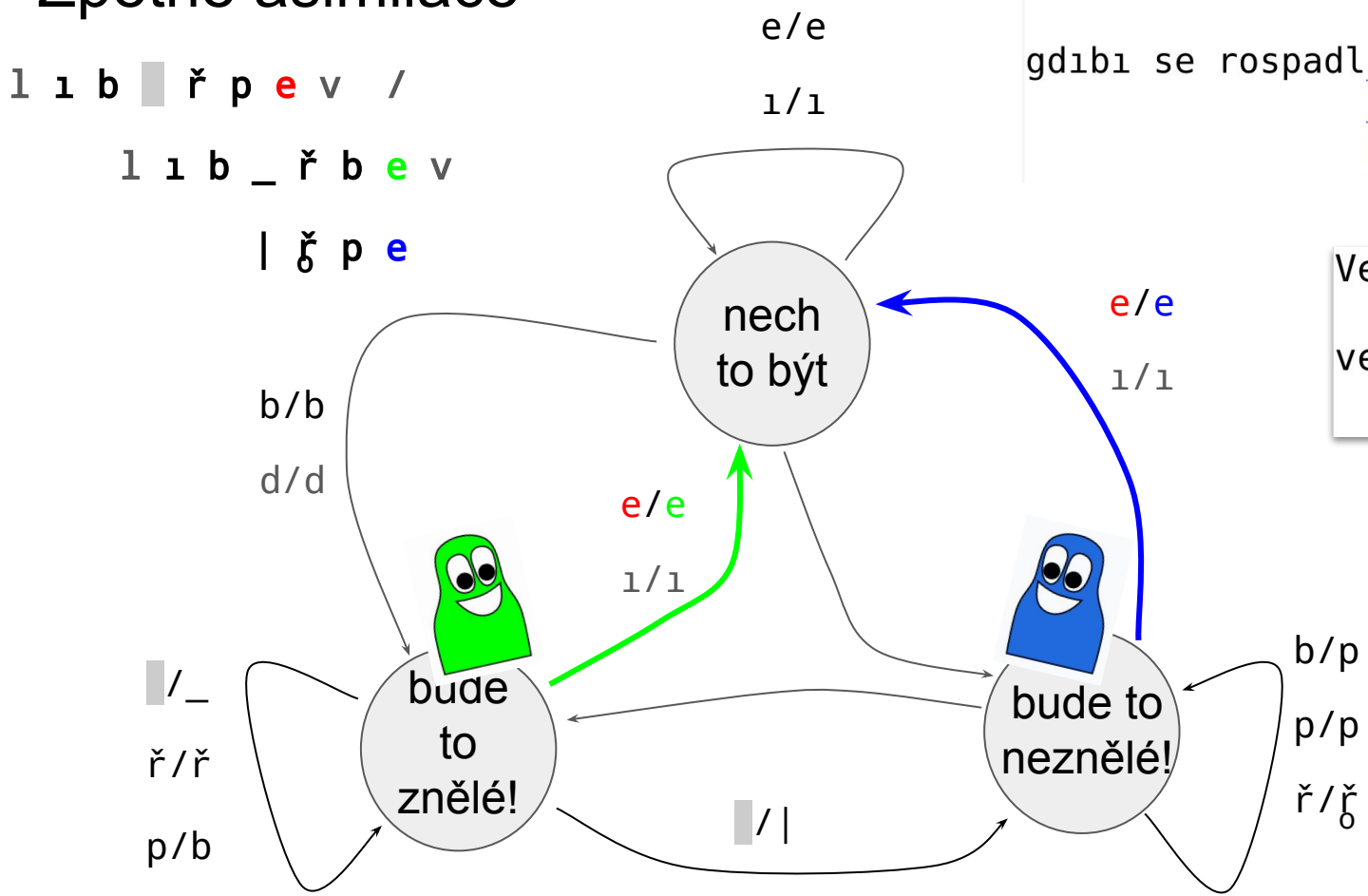
_?

|?

Vepř byl vypečený

vebř_bɪl vɪpečeni:

pǩ|



_ ... spojovník

| ... rozdělovník

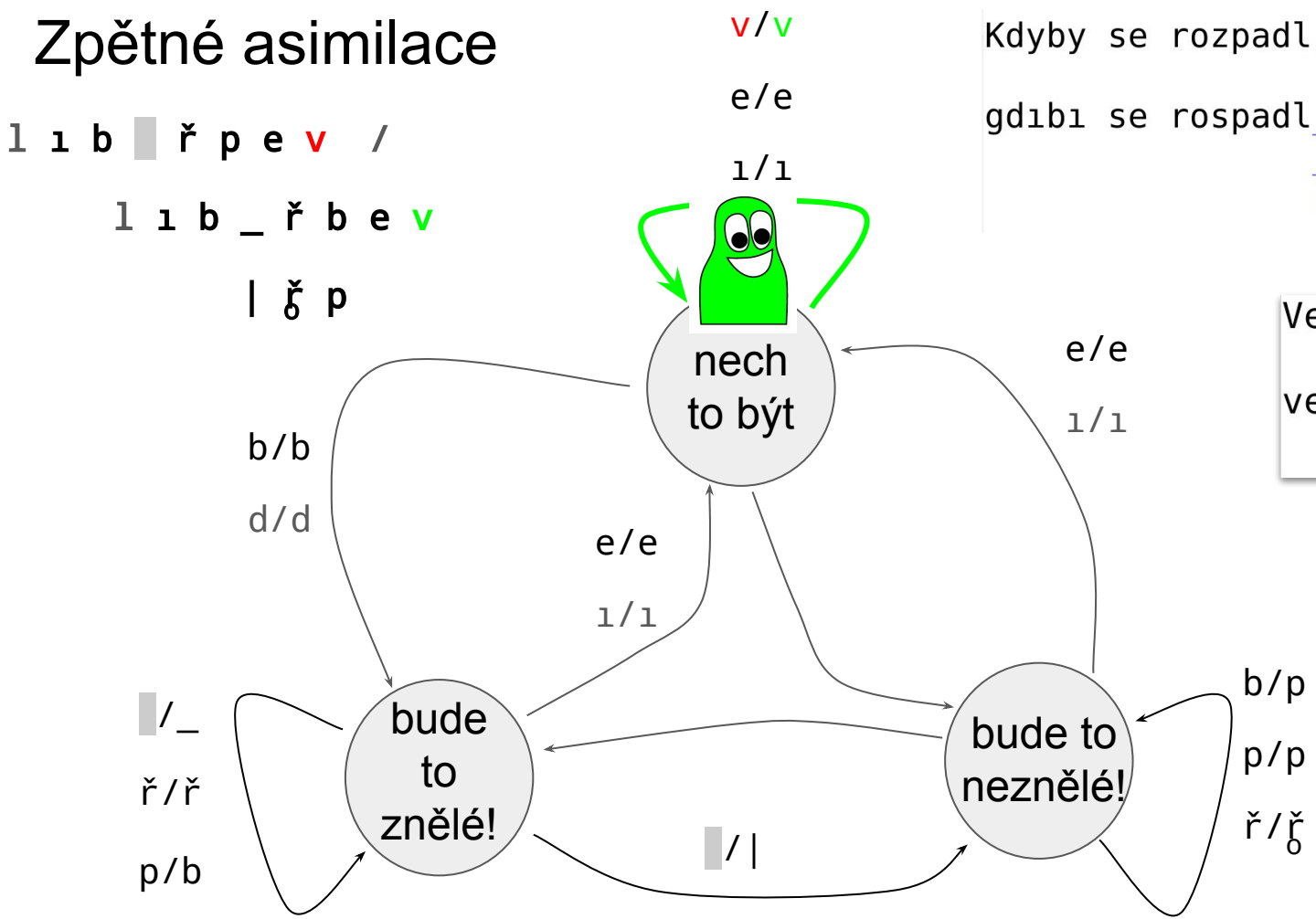
▒ ... obojí možné

Zpětné asimilace

l ɪ b ▒ ř p e v / /

l ɪ b _ ř b e v

| ǒ p



Kdyby se rozpadl, obci by prospěl.

gdɪbɪ se rospadl_ opcɪ bɪ prospjel

_?

|?

Vepř byl vypečený

vebř_bɪl vɪpečeni:

pǒ|

_ ... spojovník

| ... rozdělovník

▒ ... obojí možné

spoiler: Další překladový automat je **výrazně** jednodušší

Ráz

ı c b o l d a p z o r

/

ı c b o _ l d a p z o r

_ ?

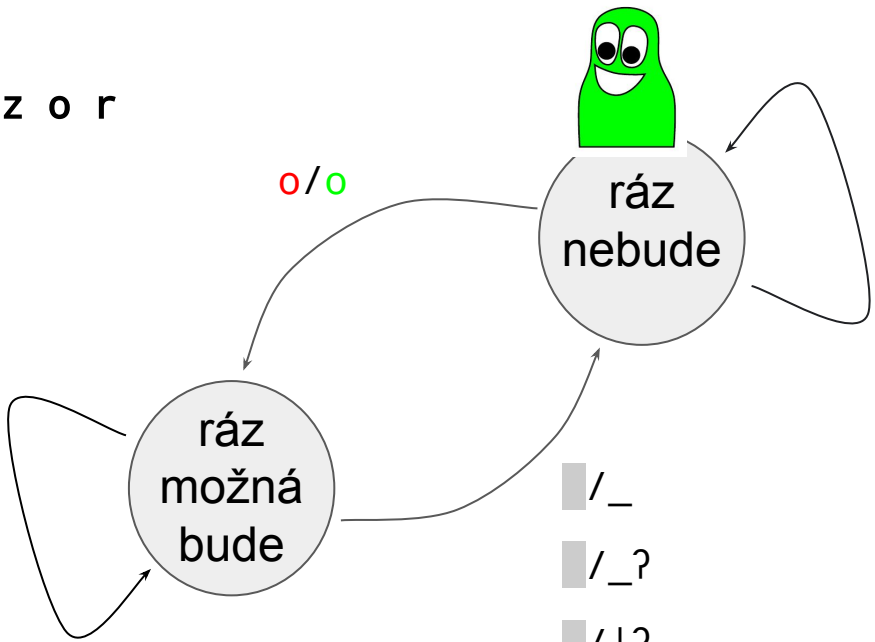
| ?

Kdyby se rozpadl, obci by prospěl.

gdıbı se rospadl_ opcı bı prospjel

_?

|?



/_

/_?

/|?

z/z

_ ... spojovník

| ... rozdělovník

... obojí možné

Ráz

ı c b o ▮ l d a p z o r

/

ı c b o _ l d a p z o r

_ ?

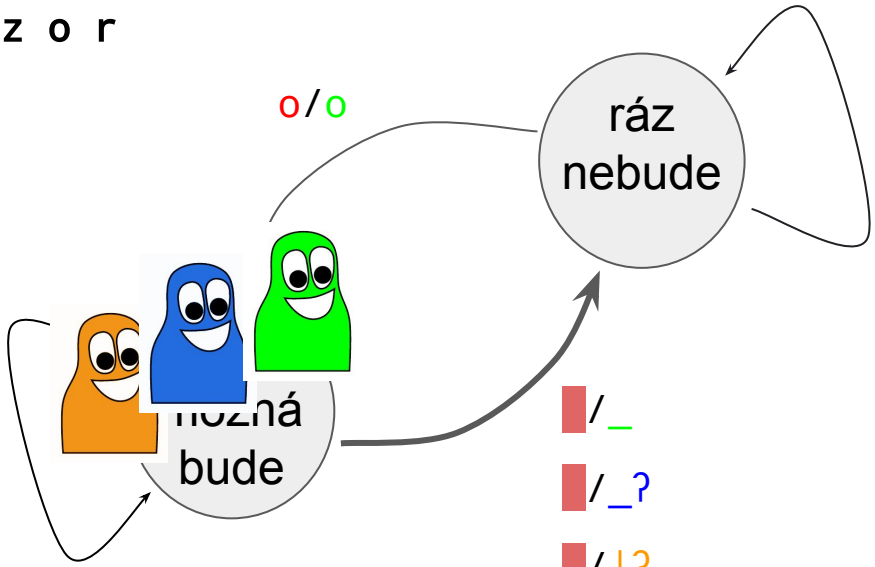
| ?

Kdyby se rozpadl, obci by prospěl.

gdıbı se rospadl_ opcı bı prospjel

_?

|?



_ ... spojovník

| ... rozdělovník

▮ ... obojí možné

Ráz

ı c b o l d a p z o r

/

ı c b o _ l d a p z o r

_ ?

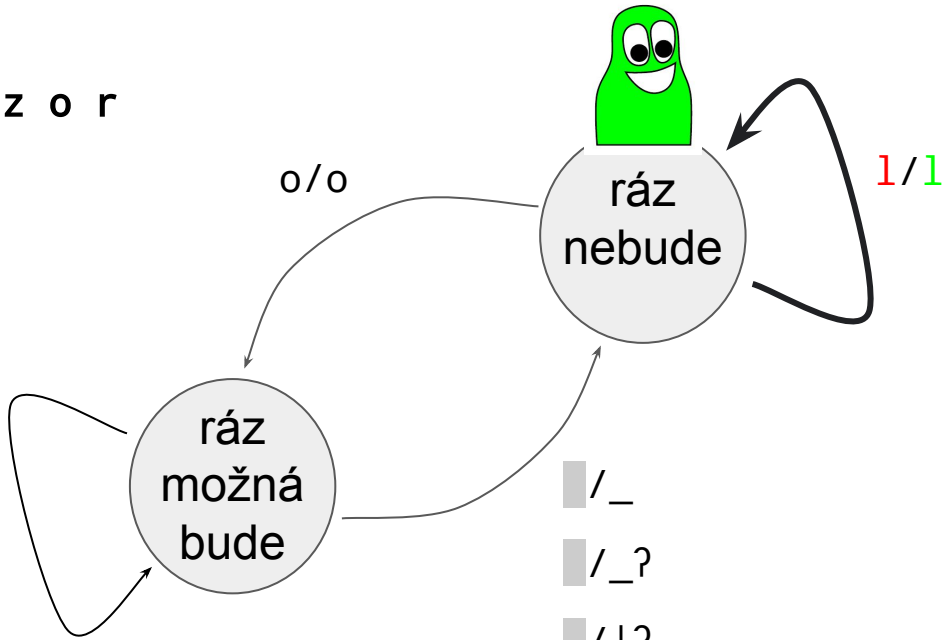
| ?

Kdyby se rozpadl, obci by prospěl.

gdıbı se rospadl_ opcı bı prospjel

_?

|?



/_

/_?

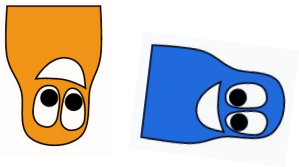
/|?

z/z

_ ... spojovník

| ... rozdělovník

... obojí možné



detaily tří automatů přeskočíme

Akustický model hlásek

v rozpoznávání řeči obvykle tvořen z přepsaných nahrávek na úrovni vět

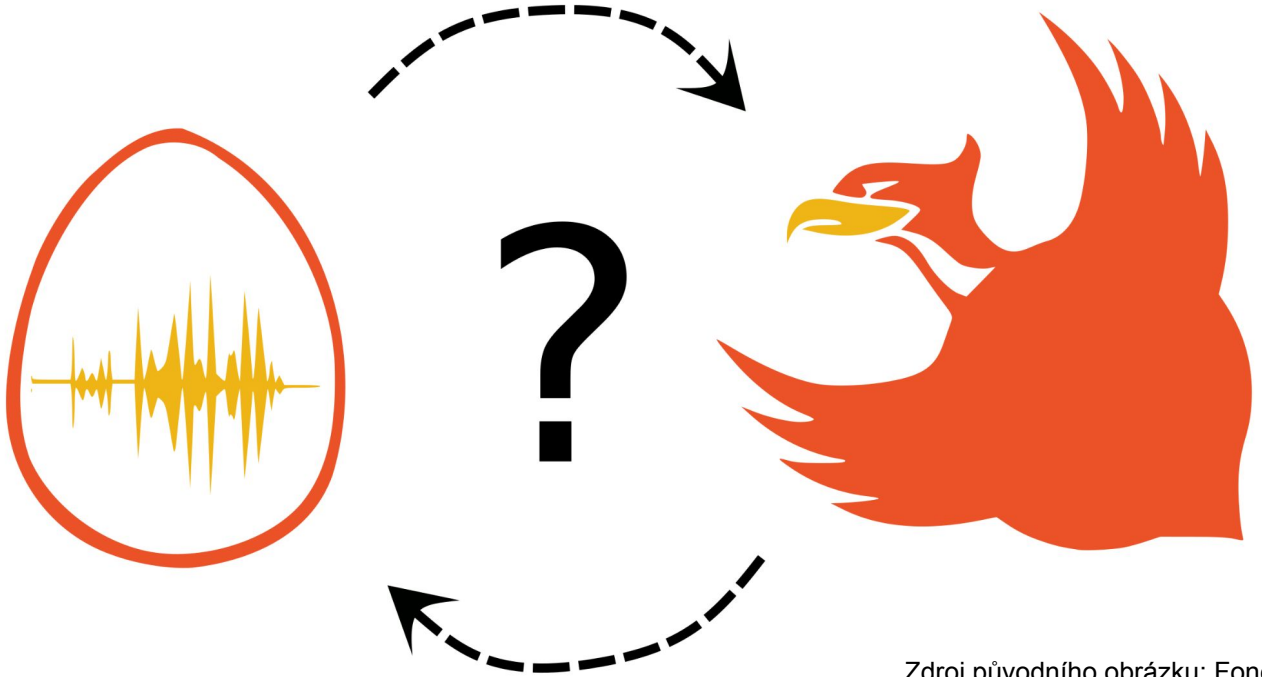
bez hranic slov či hlásek (to byl naposledy TIMIT a možná jedna verze WSJ)

ručně označit polohu hlásek je VELMI pracné

na tvorbu modelu se obvykle používá **řádově více dat**, než kdy přepsal FÚ

např. MLS: EN 45.289 h, DE 2.013 h, FR 1.106 h, ... (a česky 0 h)

Co bylo dřív?



Zdroj původního obrázku: Fonetický ústav

Časové zarovnání hlásek x akustický model hlásek

Časové zarovnání hlásek

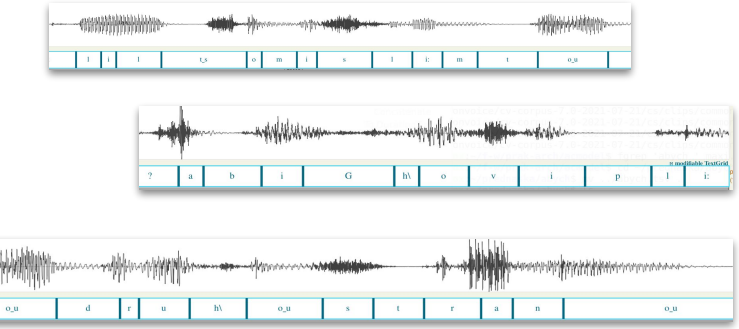
kdo ví, jak hlásky znějí, může je najít a ohraničit v nahrávce

Akustický model hlásek

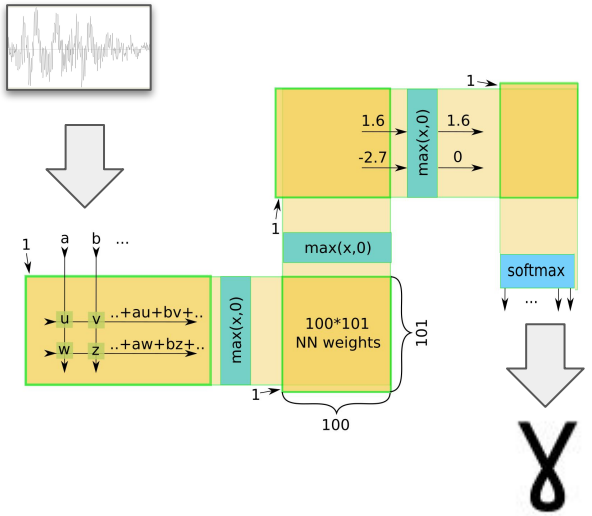
podle hlásek vystříhaných z časově zarovnaného audia se naučíte, jak znějí

Časové zarovnání hlásek x akustický model hlásek

Časové zarovnání hlásek



Akustický model



Časové zarovnání hlásek x akustický model hlásek

Časové zarovnání hlásek

kdo ví, jak hlásky znějí, může je najít a ohraničit v nahrávce (přibližně)

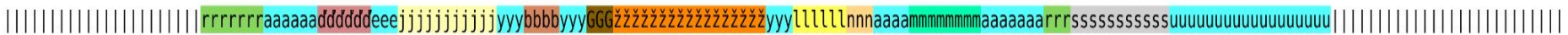
Akustický model hlásek

podle hlásek vystříhaných z časově zarovnaného audia se naučíte, jak znějí (přibližně)

Časové zarovnání hlásek na začátku a na konci učení NN

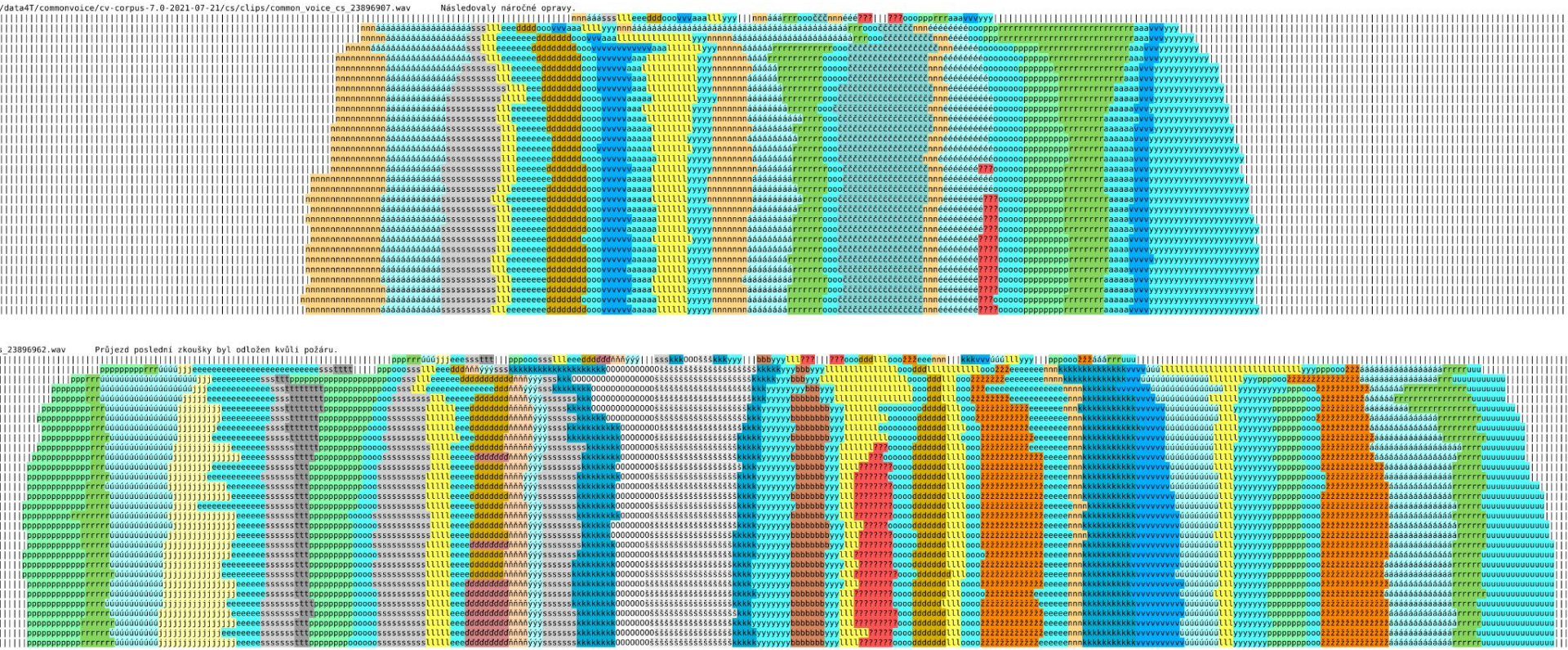
FAKE IT:

MAKE IT:

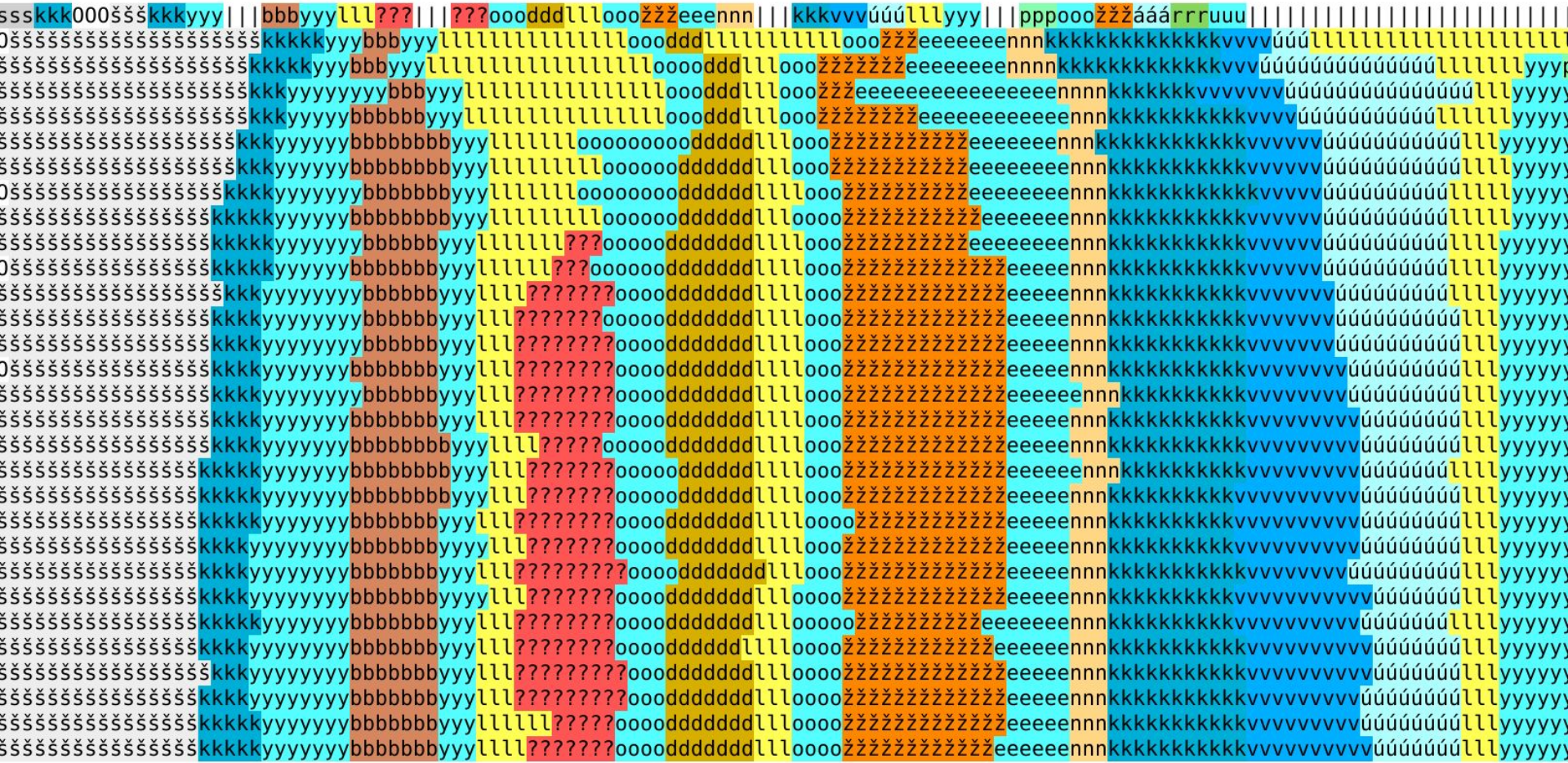


Postupná konvergence během trénování modelu

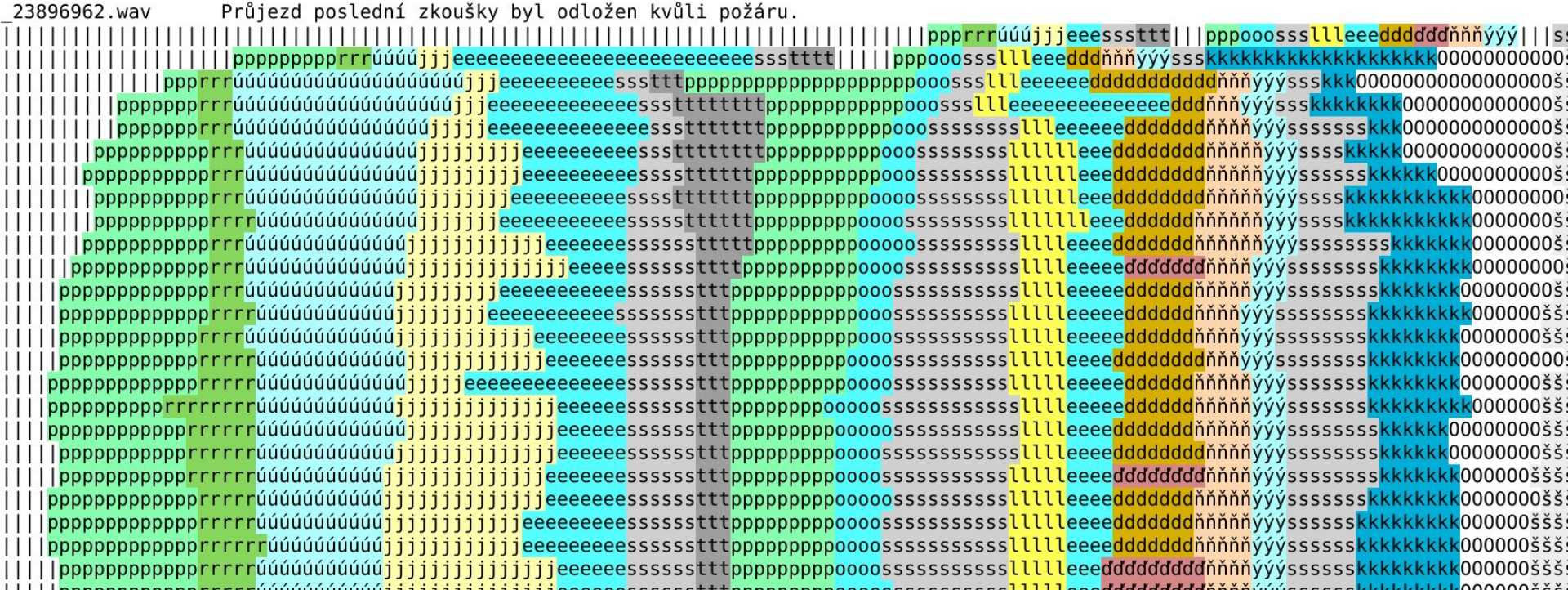
první řádek je zcela smyšlené a většinou nesprávné zarovnání
10ms/znak, 1 oběh šipek / řádek, barevné kódování dle dětské synestezie V. H.



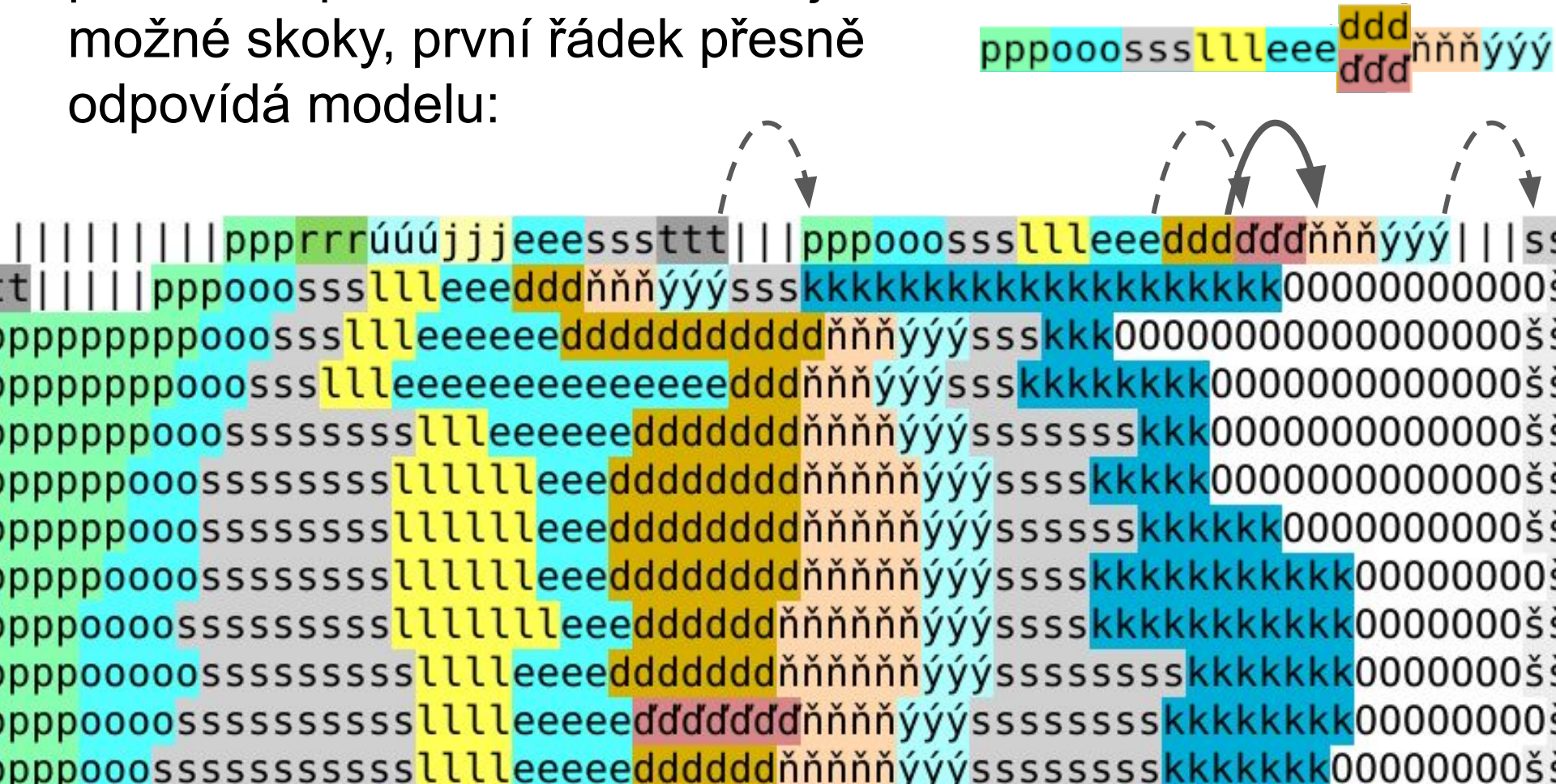
detail vpravo dole, nalezení rázu v “byl odložen”



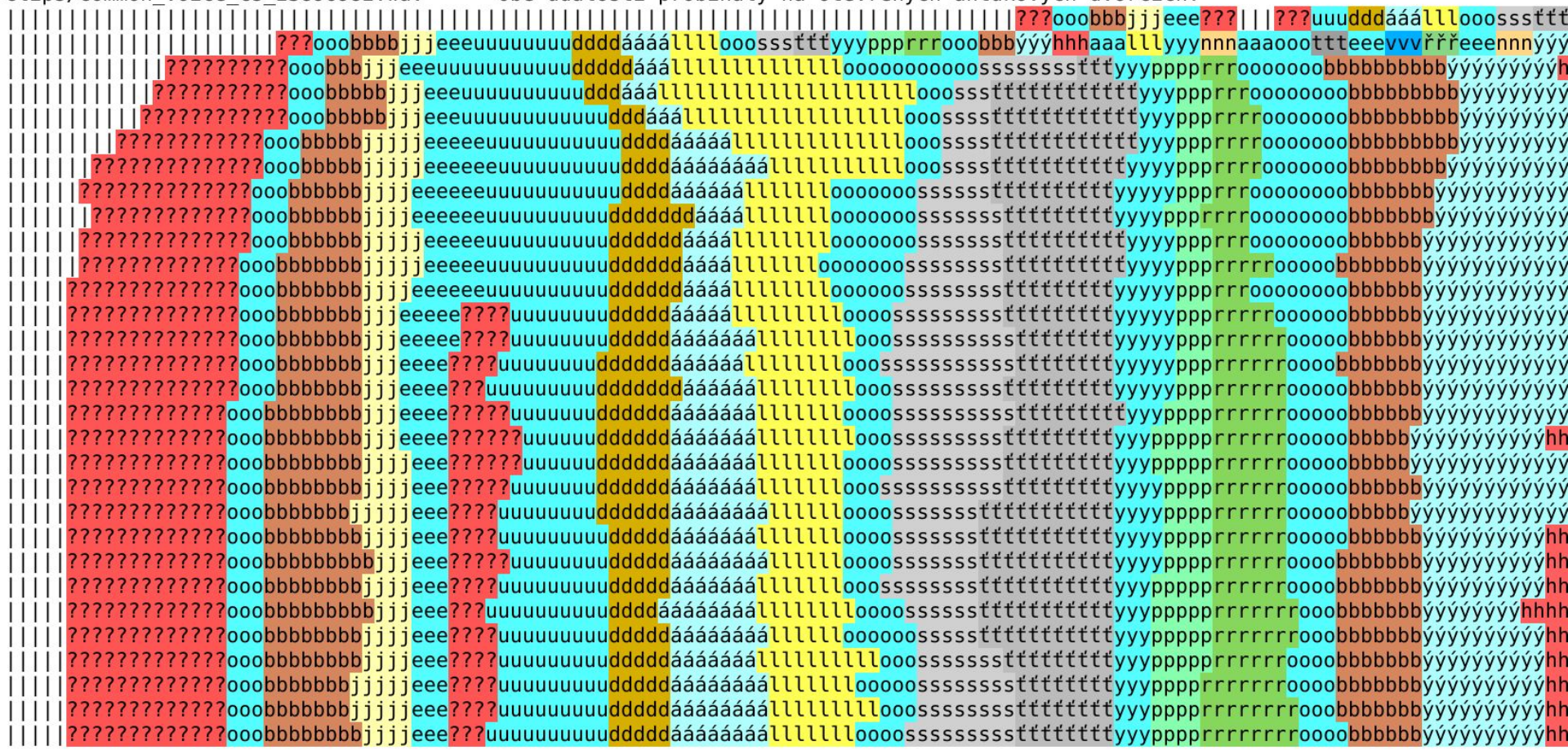
detail vlevo dole, rozhodování “poslední / posleďní”



poslední / posleďní - v modelu jsou někde možné skoky, první řádek přesně odpovídá modelu:

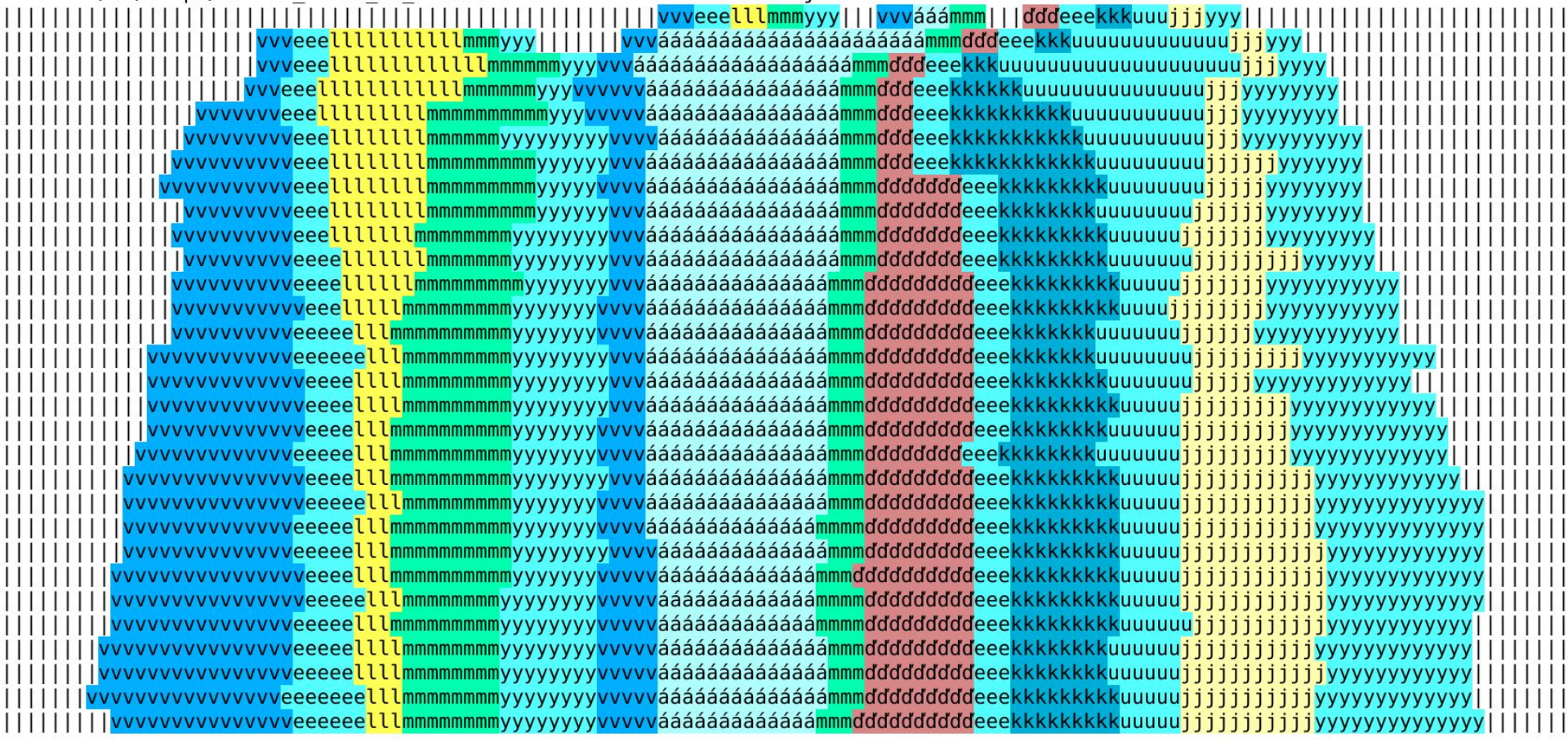


“obě události probíhaly ...”

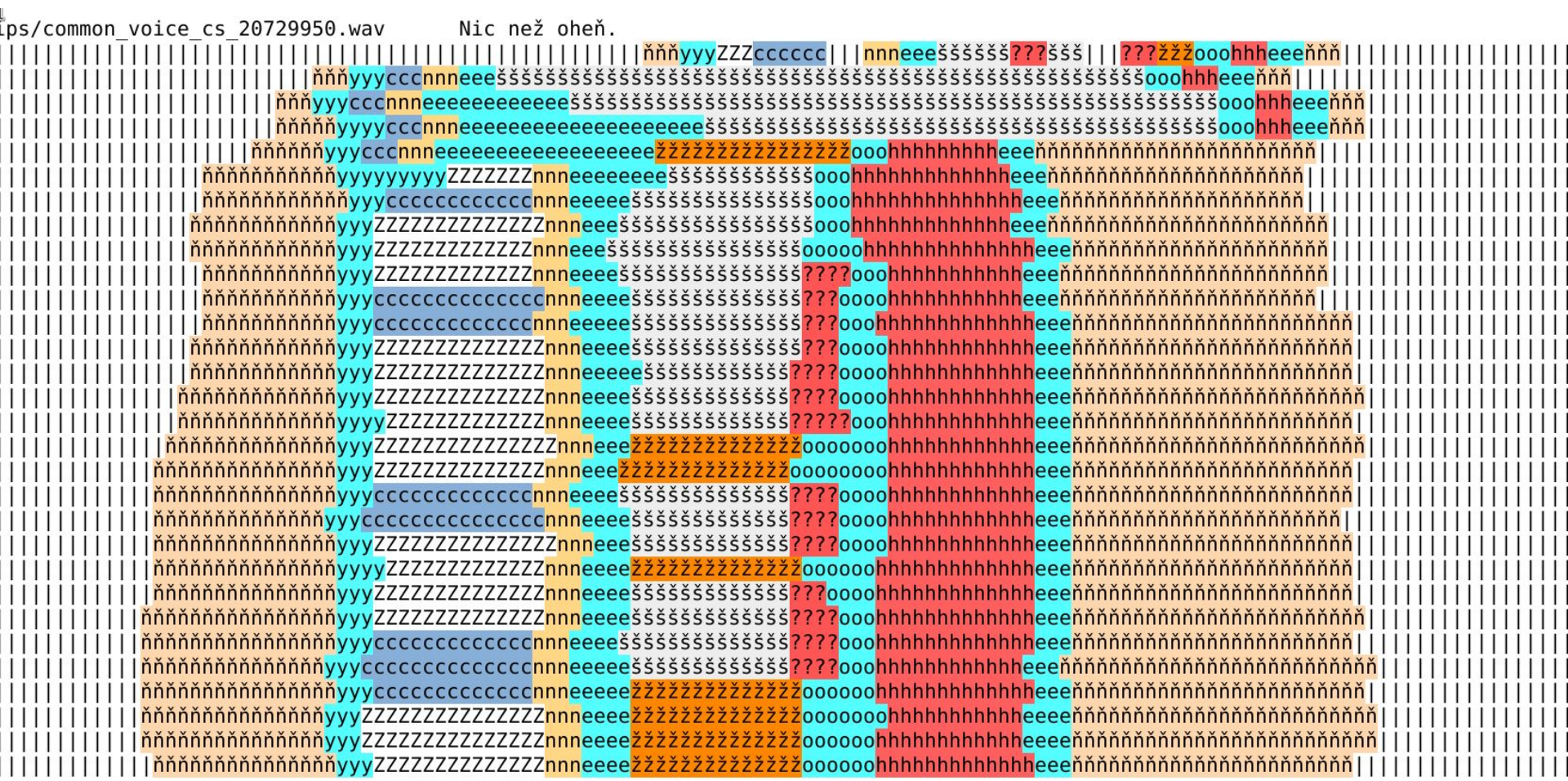


krátká nahrávka bez alternativ* “Velmi vám děkuji”

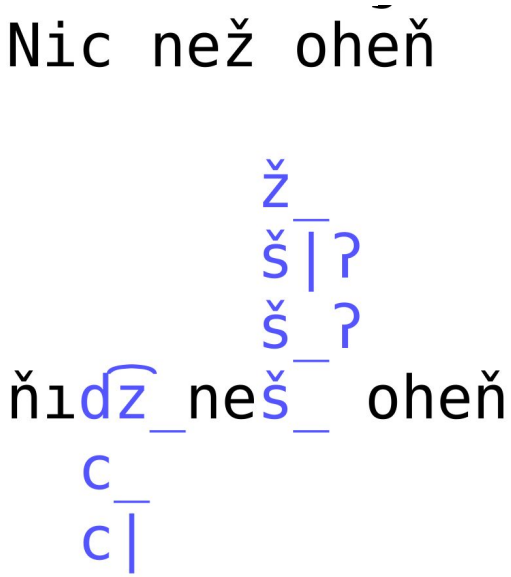
* ve skutečnosti má 4 varianty, mohou a nemusí tam být mezery



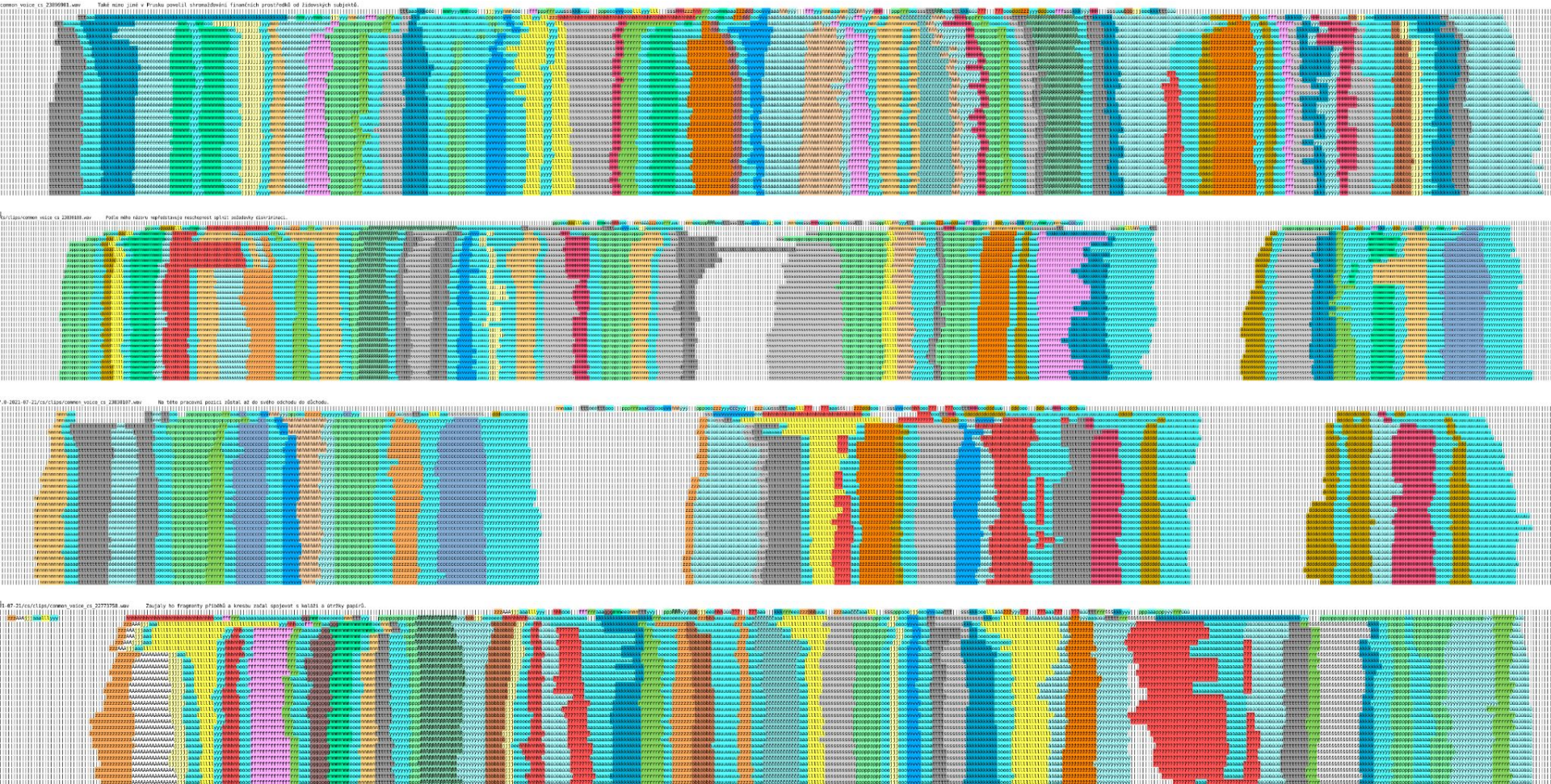
“Nic než oheň” - váhání mezi znělými a neznělými konci

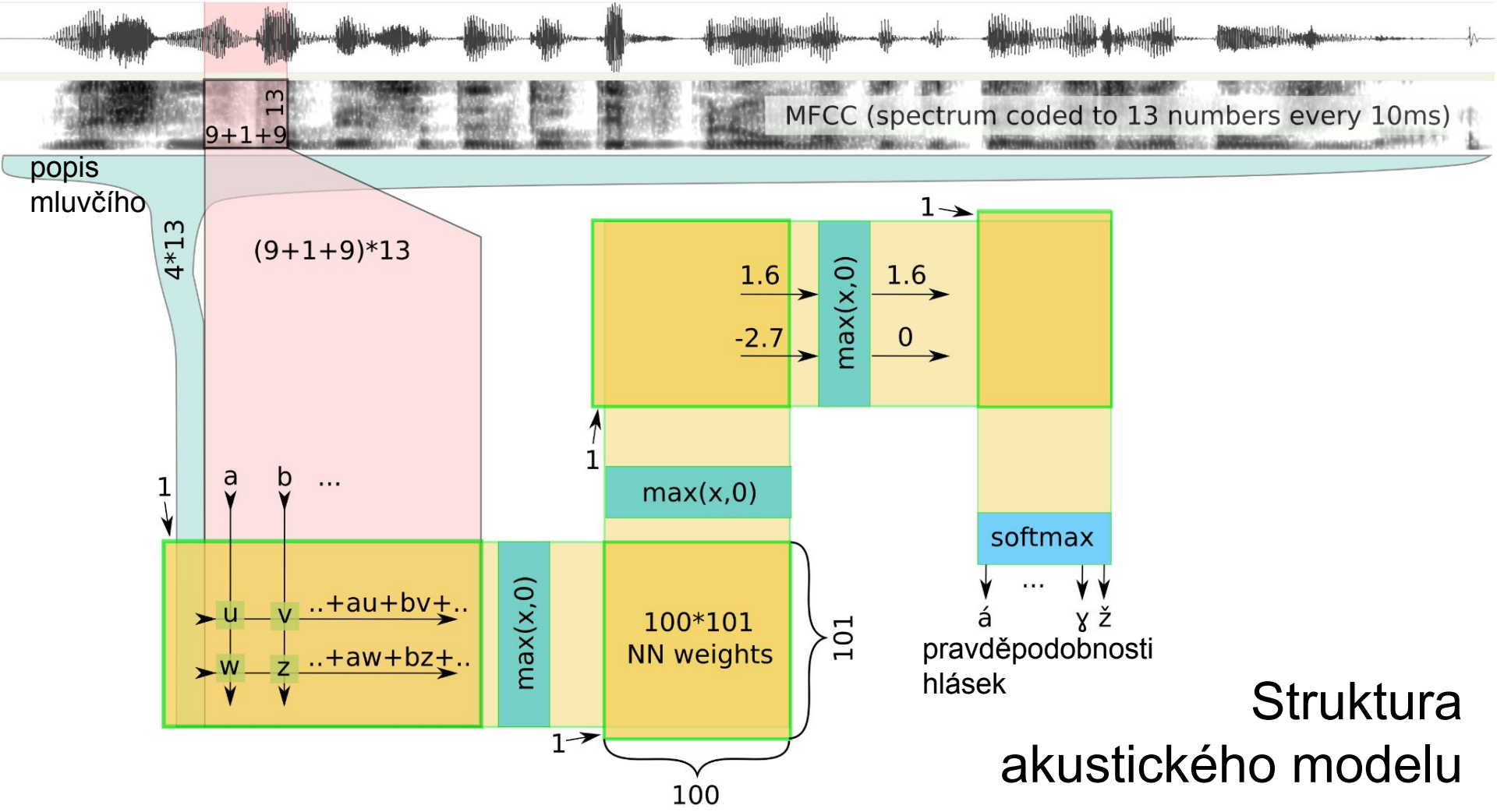


“Nic než oheň” - co připustil generátor výslovnosti



delší nahrávky





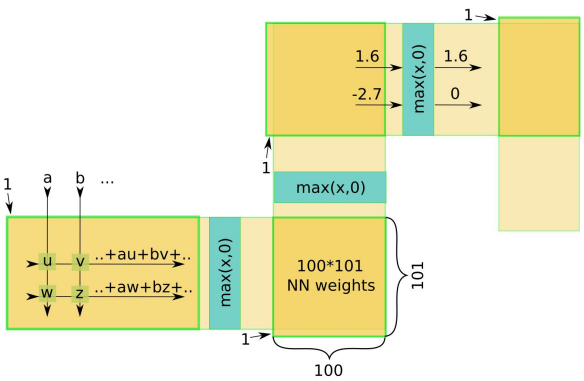
Struktura akustického modelu

PyTorch (populární NN knihovna pro python)

- optimalizované maticové operace, jako NumPy
- ale kromě CPU i GPU, MPS (Mac Metal Performance Shaders), TPU...
- autograd sleduje, čím byl výsledek ovlivněn, dle chyb lze posouvat váhy NN
- běžné NN struktury v knihovně
- torchaudio - MFCC, převzorkování

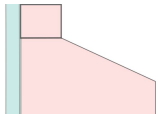
PyTorch - konstrukce NN

```
import torch
from torch import nn
from torch.utils.data import DataLoader
from torch.utils.data import Dataset
```




```
class NeuralNetwork(nn.Module):
    def __init__(self, in_size, out_size, mid_size=512):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(in_size, mid_size),
            nn.ReLU(),
            nn.Linear(mid_size, mid_size),
            nn.ReLU(),
            nn.Linear(mid_size, mid_size),
            nn.ReLU(),
            nn.Linear(mid_size, out_size)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```



PyTorch - trénovací data

```
class SpeechDataset(Dataset):
    def __init__(self, all_mfcc, all_targets, b_set, sideview = 9, speaker_vectors = None):
        self.all_mfcc = all_mfcc
        self.all_targets = all_targets
        self.sideview = sideview
        self.speaker_vectors = spe

        self.wanted_outputs = torc
        self.output_map = {}
        for i, b in enumerate(b_se
            self.output_map[b] = s
```

```
def __len__(self):
    return len(self.all_targets) - 2*self.sideview

def __getitem__(self, idx):
    idx += self.sideview
    mfcc_window = self.all_mfcc[idx-self.sideview:idx+self.sideview+1]

    nn_input = mfcc_window

    if self.speaker_vectors!=None:
        speaker_vector = self.speaker_vectors[idx]
        nn_input = torch.cat([mfcc_window, speaker_vector])

    return nn_input, self.output_map[self.all_targets[idx]]
```

PyTorch - trénování NN

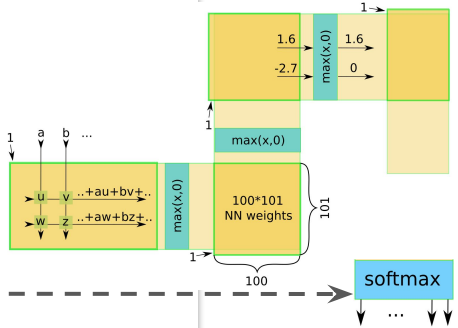
```
import torch.optim as optim

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

```
for i, data in enumerate(train_dataloader, 0):
    # get the inputs; data is a list of [inputs, labels]
    inputs, labels = data

    # zero the parameter gradients
    optimizer.zero_grad()

    # forward + backward + optimize
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()
```

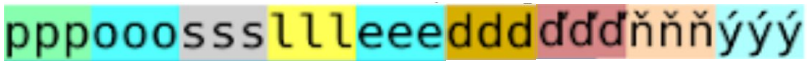


autograd kouzlo

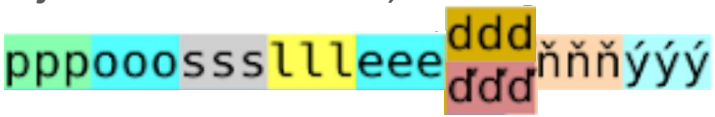
Trénování akustického modelu

CommonVoice - 10575 nahrávek (trénovací část), 14h (4h ticho, 10h řeč)

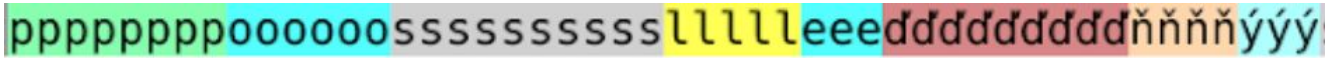
Začátek: Fikce, že každá hláska trvá přesně 30ms a ticho před a za je stejné



Počáteční iterace - častější zarovnání (model použit jako nastřelovač)



Pozdější iterace - 5 průchodů trénovacími daty na 1 zarovnání (nastřelení)



Lze natrénovat na CPU za půl dne, na GPU trochu rychlejší

Další (zatím nevyužité) možnosti trénování

Použít krásná, ručně označená data Fonetického ústavu !!!

... jsou ale kratší (nyní připraveno 5 hodin jednotně značených dat)

Další volně dostupná česká data:

- parlament ČR a EU, dlouhý dozvuk: ParCzech3.0 1290h, snemovna 454h, czparl2012 89h, VoxPopuli 60h
- OVM (televize): 30h, málo mluvčích
- vystadial2016   77h - těžká konverzační data, BUT-CZAS 5h
- komerční databáze - SpeechDAT, SpeeCon, ...
- není z čeho udělat MLS (LibriSpeech) či TEDx, není voxforge

Možnosti vylepšení s DSP triky

např. posunout hranici k výrazné změně spektra

vhodný trik závisí na hláskách, které hranice dělí

stačí přeložit knihu do programovacího jazyka :)

ohraničit dle krátkodobé energie

ohraničit dle distinktivního rysu? Lze NN...

Instalace (Windows, Mac, Linux)

https://github.com/vaclavhanzl/prak

Installation on Windows

Go to mambaforge, download mambaforge for windows (Mambaforge-Windows-x86_64). Run it, despite any protests from Windows (it will whine about unknown application, click small text 'more information' and th
Install to `C:\mambaforge`. This location is the only thing you have to change, otherwise go with
installation dialogs.

Run mambaforge console. Should be in Start menu as "Miniforge console" (name of the windo
Prompt"). Text prompt must start `(base)`. (If it does not, you ran something else. Find the right
hit Enter):

```
mamba install pytorch torchaudio -c pytorch
```

and then this (then Enter again):

```
pip install pysoundfile
```

If any Y/n questions pop up, just hit Enter (or type Y and Enter if you wish).

Download prak zip file (zip also available from green "Code" button up right on this GitHub pa
Uncompress it directly to `C:\` (or uncompress and move there), creating `C:\prak-main` folder
to `C:\prak`.

Run praat. Open script `C:\prak\prak_align_phrase.praat` and add it to menu (File > Add to
Class1: Sound, Class2: TextGrid, Command: Align using prak).

Set Praat **text files encoding to UTF-8** (Praat > Preferences > Text reading preferences > UTF
preferences > UTF-8).

Refer to Prak installation details if you need to know more.

Installation on Mac

Go to mambaforge. Choose and download the installation file here, either Mambaforge-MacOSX-x86_64 for older Intel-based Macs or Mambaforge-MacOSX-arm64 for new Macs with Apple M1 chips. Then make the downloaded file executable (using chmod) and run it in the terminal, for example (for Intel Mac):

```
chmod +x Downloads/Mambaforge-MacOSX-x86_64.sh
Downloads/Mambaforge-MacOSX-x86_64.sh
```

Answer a few questions (likely **yes** or just Enter but you must explicitly type **yes** where needed). If the licence agreement is too long to show, press spacebar untill you reach the end, then agree (yes). After installing mambaforge successfully, QUIT TERMINAL COMPLETELY and run it again. The prompt will now start with "(base)" and you can install python packages we need:

```
mamba install pytorch torchaudio -c pytorch
```

Get **prak**, preferably using **git** (makes future updates super easy):

```
git clone https://github.com/vaclavhanzl/prak.git
cd prak
```

(Or you can get zip as described above for Windows and uncompress it in your home folder.)

Run praat. Open script `C:\prak\prak_align_phrase.praat` and add it to menu (File > Add to dynamic menu > Class1: Sound, Class2: TextGrid, Command: Align using prak). Set read/write to UTF-8 (Praat > Preferences > Text reading preferences > UTF-8, then Text writing preferences > UTF-8).

Refer to Prak installation details if you need to know more.

Installation on Linux

Things happen too fast in the python world for apt package managers to keep up. So you most likely want a special package manager just for python, and there are really great tools to choose from.

For scientific work, **conda** package manager might be better than **pip**. In the conda world, there are still many options. You likely do not want the huge Anaconda but rather the more free and modular conda forge. To get it working, you still have multiple options from which **mambaforge** (faster conda) looks quite good. With this general guidance, it is now easy to google your way.

Big part of pytorch installation complexity stems from the CUDA GPU drivers installation. If you do not plan training big neural networks or do not have a decent GPU, you may very well use pytorch just on the CPU. **Prak** only uses CPU for phone alignment and even acoustic model can be reasonably trained on just the CPU.

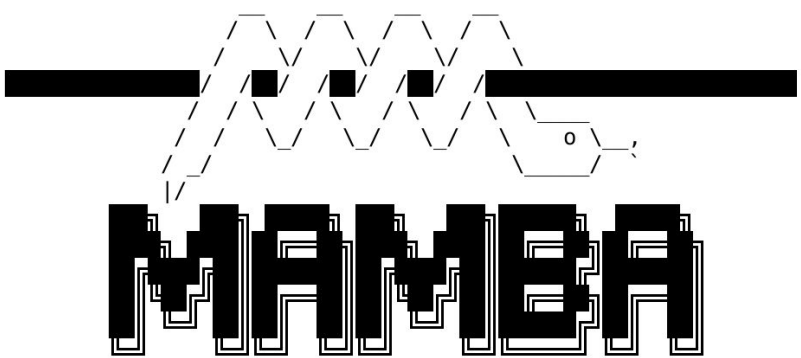
You need `pytorch` and `torchaudio` packages. As a linux hacker you likely have other things to do with python so **prak** tries to use its own virtual environment `prak` before resorting to `base` or even trying to run without a virtual

clone directly from your home like this:

same way as described above for Mac. If
g variables in this script.

PyTorch a TorchAudio instalujeme pomocí MambaForge

```
(base) hanzl@blackbox:~$ mamba activate prak
(prak) hanzl@blackbox:~$ mamba info
```

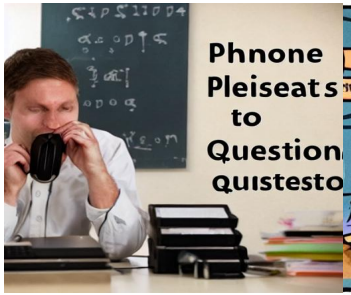


```
mamba (0.16.0) supported by @QuantStack

GitHub:  https://github.com/mamba-org/mamba
Twitter: https://twitter.com/QuantStack
```

```
     active environment : prak
    active env location : /home/hanzl/mambaforge/envs/prak
            shell level : 2
```

Dotazy?



nebo raději ...
Café Kamzík?

Obrázky vytvořeny na https://deepai.org/machine-learning-model/text2img , “phonetician asking many questions in the evening before going to bar” etc.