Tutorium 15 - 19.02.2024

Orga, Test-Exam, Regex (Exkurs)

Orga

Orga - Punkte, Vorstellen und einscannen

- Ich habe bei **allen** auf Blatt 12 (oder dem letzten korrigierten) `+6p` für das **verpasste Tutorium** vergeben
- Ich habe für **heute** bereits allen die **Anwesenheitspunkte + Vorstellen** eingetragen

Alle auf die das Zutrifft sind:

```
as2037, at359, au56, aw616, bo35, cl393, dk446, eh224, eh295, fk439, fv100,
ib180, jb1484, jx20, lf409, ln200, lp269, lp321, ls818, mk1518, mr824, mt367,
mw793, mz144, mz242, nm320, no43, pk375, rh295, rl173, rw208, sn205, tr211, ua28,
vb202, vb205, vr110, yp39, zj11
```

Bei Problemen oder Rückfragen einfach per mail nils@narl.io oder nach dem Tutorium

Orga - Klausur

- Klausur am 19.02.
- Es gibt vorraussichtlich zwei Termine
- 2 Stunden
- keine unterschiedlichen Klausuren
- Wo, Wann?
  - individuell
  - https://courses.laurel.informatik.uni-freiburg.de/courses/2023WS-EiP/exam
  - https://s.narl.io/s/termin
- Klausurumgebung ausprobieren unter
  - https://bwlehrpool.ruf.uni-freiburg.de/guacamole
  - https://s.narl.io/s/examvm

Orga - Vorbereitung auf Klausur

- Macht Altklausuren
- Übungsaufgaben im Git
  - https://git.narl.io/nvrl/eidp-klausuraufgaben-2023
  - https://s.narl.io/s/eidp-ub
- Wenn ihr die Probeklausur gut hinbekommen habt (**auch Zeitlich!!!**) seid ihr eig safe
- Zusatztutorium mit Dani und mir

Orga - Zusatztutorium von Dani und mir

- Wir machen Altklausuren/Übungsaufgaben
- Zu zweit kann man sich etwas persönlicher kümmern
- Gibt obv. keine Punkte, wir machen das auch nur freiwillig
- Wann, Wo?
  - Mittwoch
  - x.xx Uhr open end
  - Hier in 101
  - Es folgt auch noch eine E-Mail an alle über dessen Uni-Mail mit allen Infos

Test-Exam

Test-Exam - Datenklassen

- Ihr könnt **private** Attribute nicht in einer Unterklasser verwenden!
- Mit `super().post_init(...)` könnt ihr diese trotzdem setzen
- `self.__privet_attribute` in einer Unterklasse führt zu einem Fehler
- Es gibt `protected` welches von Außen nicht sichtbar ist, aber in Unterklassen
  - `_protected_attribute` welche mit einem `_` annotiert werden
  - Beißt sich leider etwas mit `InitVar[...]` von `dataclasses`
- Vergesst am besten `private`, `public` für die Klausur :) versprechen kann ich aber nichts

Test-Exam - Automata

- Bitte kein `T` oder Trash State in der Klausur, außer es ist explizit gefordert
  - Ein State bei dem invalide Eingaben hingeschoben werden
- Auch wenn das die Musterlösung von Exercise-13 gemacht hat
- Und auch wenn es eigentlich sinnvoller ist, weil wir wollen nicht bei einer falschen Eingabe dass unser Programm abstürzt

```
class State(Enum):
    q0 = auto()
    q1 = auto()
    q2 = auto()
```

Test-Exam - Automata

```python
def delta(state: State, input: str) -> State:
    match state, input:
        case State.q0, "a":
            return State.q1
        case State.q0, "b":
            return State.q2
        case State.q1, "a":
            return State.q0
        case State.q1, "b":
            return State.q1
        case State.q2, "a":
            return State.q2
        case State.q2, "b":
            return State.q1
        case _:
            raise ValueError("invalid state or input")
```

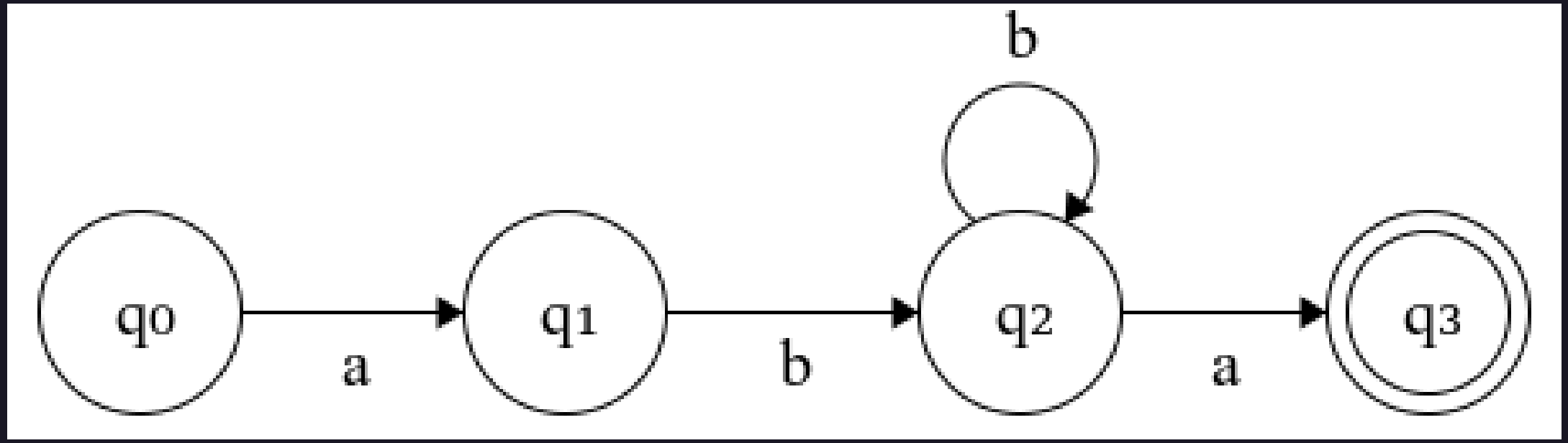
ReGex

Was ist ein ReGex?

- Ein regulärer Ausdruck ist ein **match pattern** in einem **text**
- Genau gesagt bildet es eine Menge von Zeichenketten (eine **Sprache**)
- Ihr habt bereits ReGex benutzt
  - Wenn ihr z.B. im Browser Ctrl+F drückt und nach einem Wort sucht
  - das Wort ist dann ein ReGex
- Es gibt aber auch deutlich komplexere ReGex

Automaten schon wieder

- Was ist wenn wir einen Eingabe-String überprüfen wollen ob er
  - mit `a` beginnt
  - dann mindest ein, aber beliebig viele `b` folgen
  - und mit einem `a` endet
- Wir können einen Akzeptor zeichnen! (nicht-deterministischen endlichen Automaten mit akzeptierenden Zustand)


b
q0
a
q1
b
q2
a
q3

ReGex - Python

- In Python haben wir `re` also Modul
- Ein ReGex ist eine Zeichenkette
  - `"ab"` akzeptiert `"ab"`
  - `re.fullmatch(r"ab", "ab")`
- Es gibt Sonderzeichen wie `*, +, (, ), ...` mit denen man komplexere Eingaben überprüfen kann
- Wir wollen `"ab...a"` von der vorherigen Slide matchen
  - `b*` möchte 0 bis unendlich `b`
  - `b+` möchte 1 bis unendlich `b`
- also `re.fullmatch(r"ab+a", "abbbbbbba")` ist ein Match

Weiter Sonderzeichen/Variabeln

- Mit `\d` kann man in Python eine beliebige Zahl meinen
- Mit `\s` kann man ein beliebigen Whitespace meinen
- So kann man z.B. eine beliebige Ip so darstellen
  - `r'(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})'`
- Nützlich zum Parsen oder auch Testen
  - Ich nutze z.b. ReGex um eure Aufgaben zu testen

Viel Erfolg bei der Klausur!