Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»
Мегафакультет компьютерных технологий и управления
Факультет программной инженерии и компьютерной техники

Лабораторная работа № 1
по дисциплине «Теория систем»

Выполнил: студент
Чайкин Вадим Константинович
группа P3324
Принял: преподаватель
Русак Алёна Викторовна

г. Санкт-Петербург
2024

Задание лабораторной работы

Требуется спроектировать управляющий конечный автомат.

Пример

Конечный автомат, управляющий стиральной машиной. Стиральная машина работает в трех режимах: залив, стирка, слив.

- Машина начинает работать по нажатию кнопки «Пуск».
- После этого происходит залив воды до тех пор, пока датчик $d1$ уровня воды не подаст сигнал о заполнения бака стиральной машины. Затем происходит стирка. Эта операция ограничивается с помощью таймера t.
- Если таймер исправен, то по истечении определенного времени он выдает сигнал о завершении стирки и стиральная машина переходит в режим слива воды.
- Если таймер неисправен, то стиральная машина переходит в состояние «дефект», т.е. в состояние ожидания, которое может быть прервано только после ремонта стиральной машины.
- Из состояния «дефект» стиральная машина не может возвратиться в исходное состояние, поэтому последующее нажатие кнопки «Пуск» не приведет к запуску стиральной машины.
- Из неисправного состояния после ремонта стиральная машина возвращается в исходное состояние по сигналу reset. Слив воды завершается при получении сигнала от датчика $d2$ уровня воды о том, что в баке воды нет. После этого стиральная машина возвращается в исходное состояние.

Выполнение лабораторной работы

Процесс в компьютере — это единица активности процессора, характеризуемая выполнением последовательности команд, текущим состоянием и связанным с ней множеством системных ресурсов.
Рассмотрим модель процесса с 5 состояниями:

- **New**: Процесс создан, но ещё не размещён в очереди процессов, готовых к исполнению.
- **Runnable (Ready)**: Процесс обладает всеми ресурсами для выполнения, но нет возможности исполняться.
- **On CPU**: Процесс в данный момент выполняется.
- **Wait (Blocked)**: Процесс заблокирован и ожидает события.
- **Exit**: Процесс завершил выполнение, но его структуры всё ещё существуют.

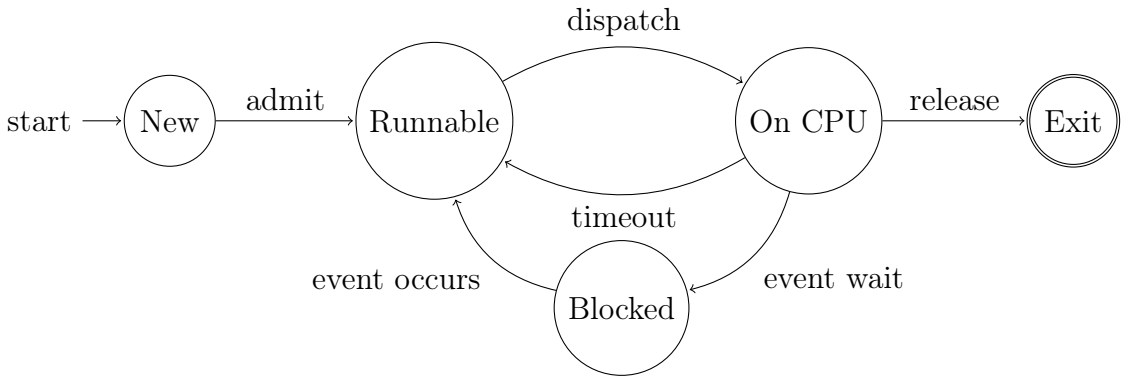

Рис. 1: Диаграмма состояний процесса с 5 состояниями

Описание конечного автомата на формальном языке

У этого автомата есть 1 начальное состояние (**New**), поэтому он инициальный.
Алфавит состояний: $X = \{\text{new, runnable, on\_cpu, blocked, exit}\}$.
Алфавит входа: $U = \{\text{admit, dispatch, timeout, event\_wait, event\_occurs, release}\}$.
Алфавит выхода: $Y = \{\text{new, runnable, on\_cpu, blocked, exit}\}$.
Функция перехода $\delta : X \times U \to X$ задаётся таблицей:

Вход / состояние	**new**	**runnable**	**on_cpu**	**blocked**	**exit**
admit	runnable	—	—	—	—
dispatch	—	on_cpu	—	—	—
timeout	—	—	runnable	—	—
event_wait	—	—	blocked	—	—
event_occurs	—	—	—	runnable	—
release	—	—	exit	—	—

Таблица 1: Таблица перехода между состояниями

Так как выход однозначно определяется состоянием (функция $\lambda : X \to Y$), то данный конечный автомат можно рассматривать как автомат Мура.

Реализация конечного автомата на Python

Реализация — см. GitHub.
Для реализации конечного автомата использована библиотека `transitions`.
Исходный код:

```
!pip install transitions
import transitions

from transitions import Machine, MachineError

# This object will have states
class Process(object):
    pass

lump = Process()

# Full list of states
states=['New', 'Runnable', 'On CPU', 'Blocked', 'Exit']

# Add table of state transitions
transitions = [
    { 'trigger': 'admit', 'source': 'New', 'dest': 'Runnable' },
    { 'trigger': 'dispatch', 'source': 'Runnable', 'dest': 'On CPU' },
    { 'trigger': 'timeout', 'source': 'On CPU', 'dest': 'Runnable' },
    { 'trigger': 'event_wait', 'source': 'On CPU', 'dest': 'Blocked' },
    { 'trigger': 'event_occurs', 'source': 'Blocked', 'dest': 'Runnable' },
    { 'trigger': 'release', 'source': 'On CPU', 'dest': 'Exit' },
]

    # trigger     an action that leads to state change,
    # source     state before transition,
    # dest     state after transition.

# Initialize machine
machine = Machine(lump, states=states, transitions=transitions, initial='New')

assert lump.state == 'New',\
  f'Incorrect state: initial state should be New, actually {lump.state}'

lump.admit()
assert lump.state == 'Runnable',\
  f'Incorrect state: state after admit should be Runnable, actually {lump.state}'

lump.trigger('dispatch')
assert lump.state == 'On CPU',\
  f'Incorrect state: state after admit should be On CPU, actually {lump.state}'
```

```
machine = Machine(lump, states=states, transitions=transitions,
    initial='New')
try:
    print(lump.state)

    lump.admit()
    print(lump.state)

    lump.dispatch()
    print(lump.state)

    lump.event_wait()
    print(lump.state)

    lump.event_occurs()
    print(lump.state)

    lump.dispatch()
    print(lump.state)

    lump.release()
    print(lump.state)
except MachineError as error:
    print(error)
```

Выводы

В ходе выполнения лабораторной работы №1 по дисциплине «Теория систем» были выполнены следующие действия:

- Построен конечный автомат на примере модели процесса с 5 состояниями: **New**, **Runnable**, **On CPU**, **Blocked** и **Exit**. Каждое из этих состояний отражает определённый этап жизненного цикла процесса.
- Построена диаграмма состояний, наглядно демонстрирующая переходы между состояниями, что позволило увидеть логику работы процесса и понять взаимосвязь между действиями (входными сигналами) и изменениями состояний.
- Сформулировано формальное описание конечного автомата с указанием:
  - множества состояний X,
  - множества входных сигналов U,
  - множества выходов: Y,
  - функции переходов $\delta : X \times U \to X$ с помощью таблицы переходов.
- Реализован конечный автомат на языке Python с использованием библиотеки `transitions`. Код продемонстрировал корректное выполнение переходов между состояниями:
  - Переход из состояния **New** в состояние **Runnable** по входному сигналу `admit`.
  - Переход из состояния **Runnable** в состояние **On CPU** по входному сигналу `dispatch`.
  - Другие переходы, соответствующие описанной логике модели.
- Проведено тестирование автомата с использованием утверждений (`assert`), что подтвердило соответствие фактического состояния автомата ожидаемым результатам на некоторых этапах выполнения.

Таким образом, лабораторная работа позволила:

- Углубиться в теоретические основы конечных автоматов и их применение в моделировании процессов в вычислительных системах.
- На практике реализовать конечный автомат, что демонстрирует важность формального описания систем для обеспечения их корректной работы.
- Получить практический опыт работы с языком Python и библиотекой `transitions`, что может быть полезно при разработке более сложных систем автоматизации и управления.

Список использованных источников

1. Google Classroom