

Homework #13

[Homework #1](#)

[Homework #2 \(graphs\)](#)

[Homework #3](#)

[Homework #4 \(alphabets\)](#)

[Homework #5](#)

[Homework #6](#)

[Homework #7](#)

[Homework #8](#)

[Homework #9](#)

[Homework #10](#)

[Homework #11](#)

[Homework #12](#)

[Homework #13](#)

▼ Homework #1

Task #5

Подогревает воду до 3 разных температур и поддерживает заданную температуру. Крышка на замке, пока требуемая температура не достигнута. Защита от отсутствия воды.

Термопот

- Термопот предназначен для нагрева воды до 3 разных температур и поддержания данных температур
- Имеет крышку на замке, которая блокируется на время нагрева воды
- Наличие воды фиксируется датчиком
- Температура воды фиксируется датчиком
- Имеются 3 кнопки для выбора температуры воды [которые также “запускают” термопот]
- Имеется кнопка для включения режима поддержания температуры

Требования и процессы

- **Параллельные процессы**

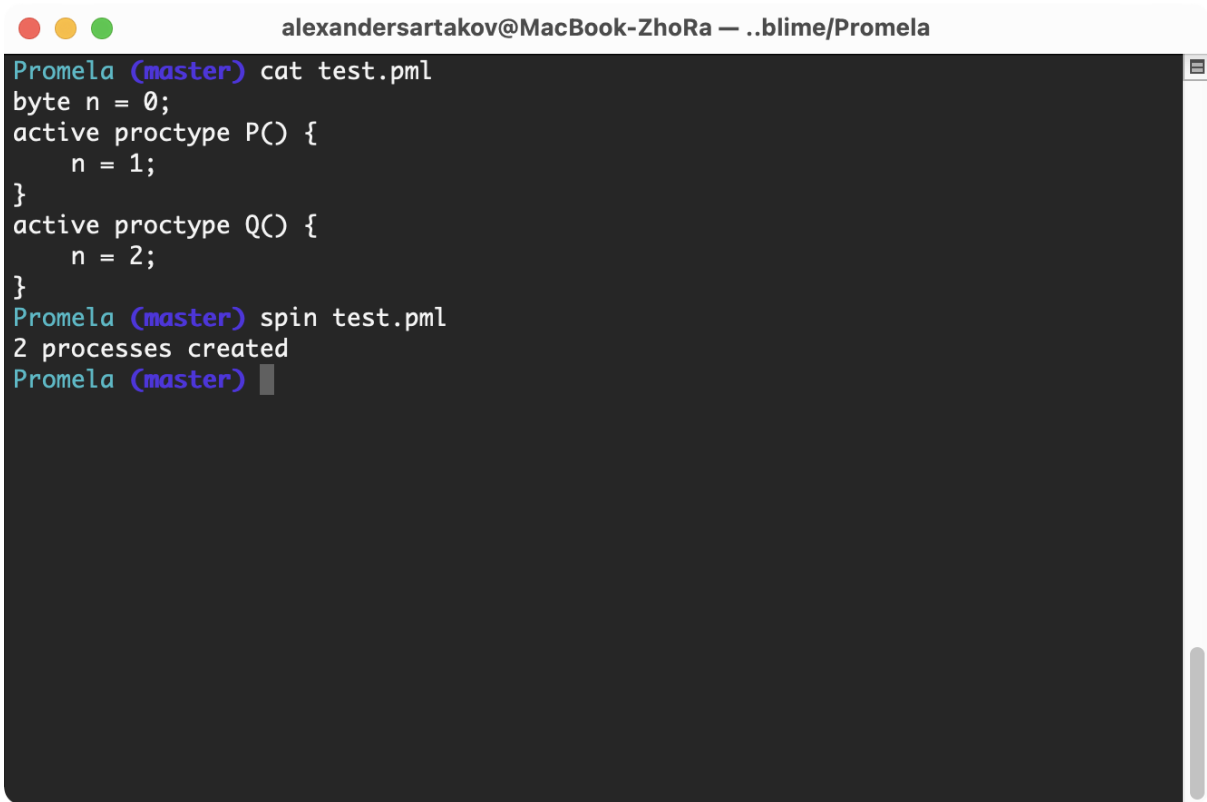
1. 3 кнопки выбора температуры
 2. Кнопка поддержания температуры
 3. Датчик температуры воды
 4. Датчик наличия воды
 5. Замок на крышке
 6. Обработчик сигналов датчиков (кнопок и регистрирующих температуру)
 7. Нагреватель воды
 8. Температура воды
 9. Вода
 10. Пользователь
- **Требования к системе**
 1. При отсутствии воды нагрев не производится
 2. Для старта нагрева воды крышка должна быть закрыта
 3. Нагрев начинается после нажатия на одну из кнопок при закрытой крышке и наличии воды
 4. При нагревании или поддержании температуры крышка блокируется
 5. При нажатии кнопки “поддержание температуры” термопот поддерживает последнюю заданную температуру

События-состояния процессы

- Пользователь
 - absent
 - pressing_1
 - pressing_2
 - pressing_3
 - pressing_temp
 - close_lid
 - open_lid
- Вода

- exist
 - not_exist
 - Нагреватель воды
 - turn_on
 - turn_off
 - Температура воды
 - increasing
 - decreasing
 - equals_1, equals_2, equals_3
 - Кнопки для выбора температуры
 - pressing_1, pressed_1, released_1
 - pressing_2, pressed_2, released_2
 - pressing_3, pressed_3, released_3
 - Кнопка “поддержание температуры”
 - pressing_temp, pressed_temp, released_temp
 - Датчик наличия воды
 - is_poured, not_poured
 - Датчик температуры
 - is_reached, not_reached
 - Крышка
 - is_closed, is_opened
 - is_locked, is_unlocked
 - Обработчик сигналов кнопок и датчиков, контроллер
 - Почти все перечисленные события всех процессов
-

Simple Promela code executed with SPIN

A terminal window titled 'alexandersartakov@MacBook-ZhoRa — ..blime/Promela' showing the execution of Promela code. The code defines two processes, P and Q, and then runs them with the SPIN tool. The output shows that two processes were created.

```
alexandersartakov@MacBook-ZhoRa — ..blime/Promela
Promela (master) cat test.pml
byte n = 0;
active proctype P() {
    n = 1;
}
active proctype Q() {
    n = 2;
}
Promela (master) spin test.pml
2 processes created
Promela (master) █
```

▼ Homework #2 (graphs)

User

Тут происходит следующее:

- Пользователя нет
- Пользователь открывает крышку и наливает или сливает воду, затем закрывает крышку
- Пользователь нажимает одну из кнопок, после чего может делать всё, что захочет (поэтому много связей возникает)

[pressing_temp] - кнопка режима “поддержания температуры”, для удобства, пусть пользователь нажимает на кнопку несколько раз для активации режима поддержания температуры

```
graph TD
a --> a[absent] & b(pressing_1) & c(pressing_2) & d(pressing_3) & g(open_lid)
g --> h(add_water) & i(decrease_water)
```

```

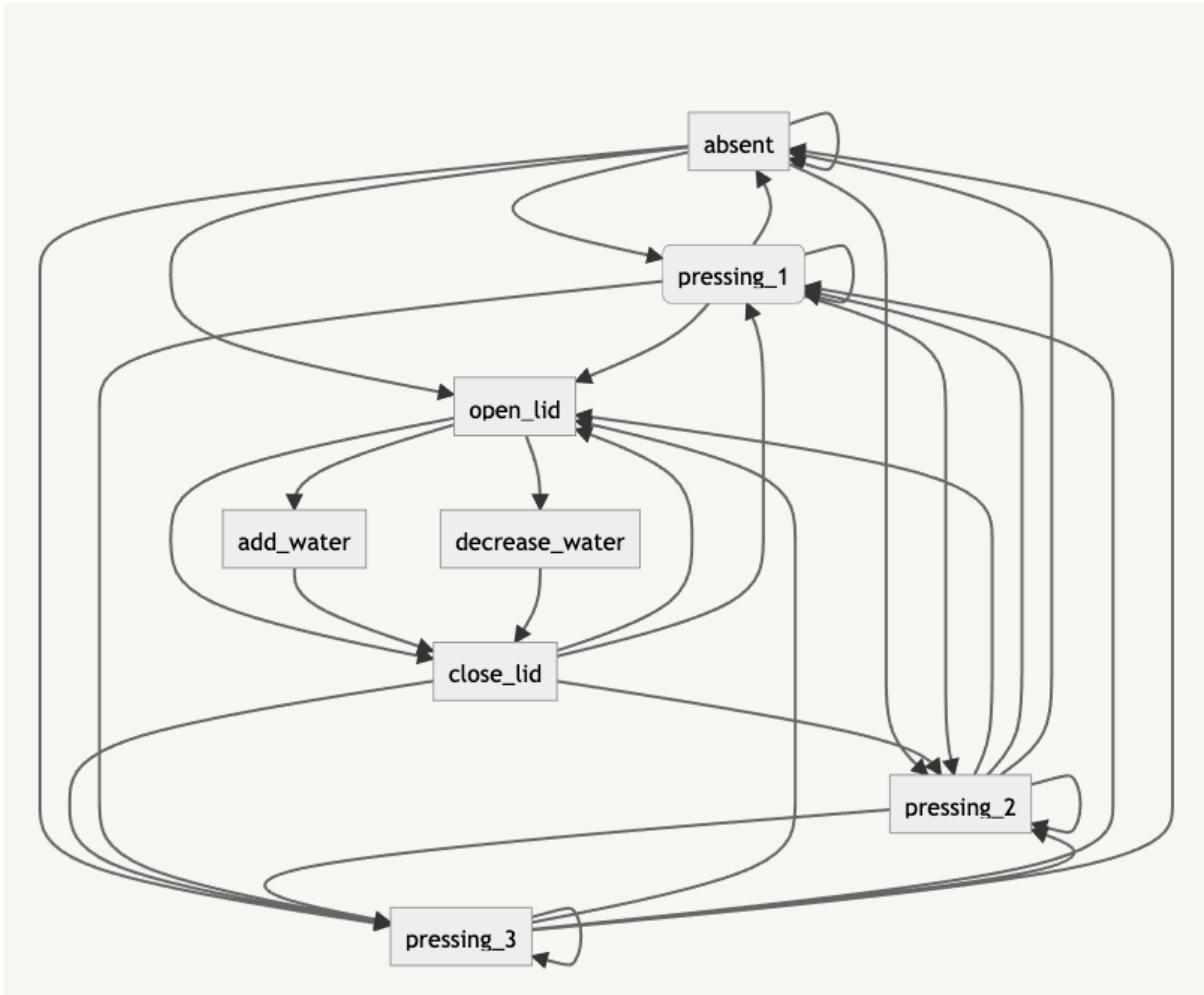
g & h & i --> f[close_lid]

b & c & d --> g

f --> g & b & c & d

b --> c & d & a & b
c --> b & d & a & c
d --> b & c & a & d

```



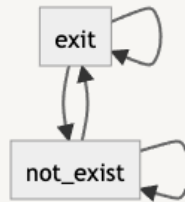
Вода

Ну, может есть, а может нет... Вечный вопрос

```

graph TD
    a[exit] --> b[not_exist]
    b --> a
    a --> a
    b --> b

```

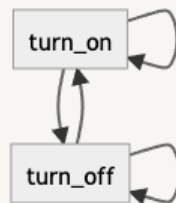


Нагреватель воды

Либо он работает, либо он не работает... Такой себе бинарник

```

graph TD
    a[turn_on] --> b[turn_off]
    b --> a
    a --> a
    b --> b
  
```



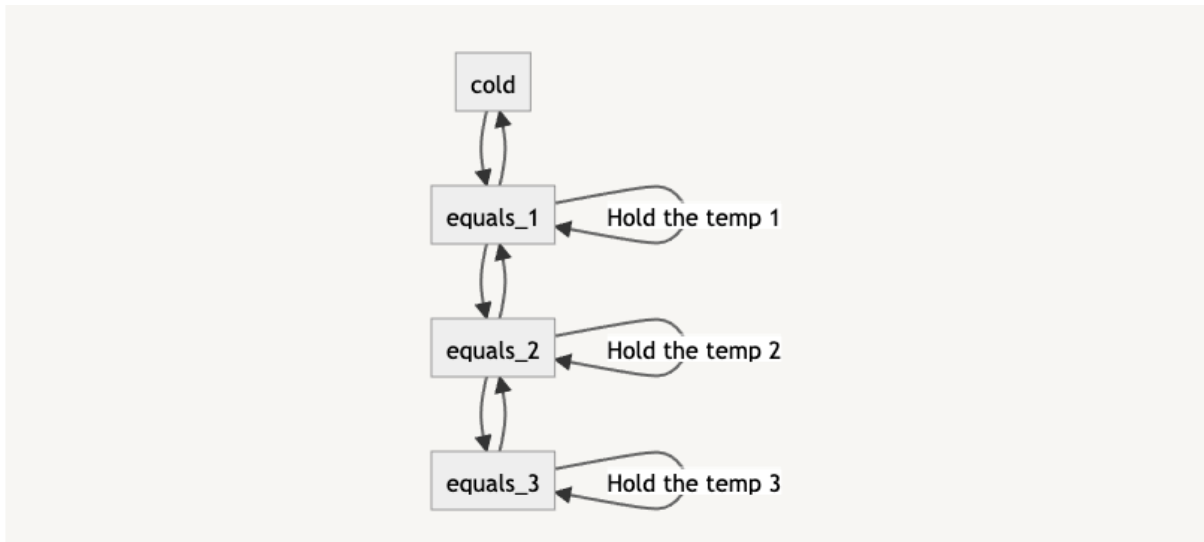
Температуры воды

Собственно, может увеличиваться, может уменьшаться и в это время становится одной из 3 нужных температур (+ если включена функция поддержания температуры, будет переходить из equals → equals)

Как видно из схемы, temp_1 > temp_2 > temp_3 (то есть установлен порядок)

```

graph TD
    cold --> equals_1 --> equals_2 --> equals_3
    equals_1 --> cold
    equals_3 --> equals_2 --> equals_1
    equals_1 --> |Hold the temp 1| equals_1
    equals_2 --> |Hold the temp 2| equals_2
    equals_3 --> |Hold the temp 3| equals_3
  
```



Кнопки для выбора температуры и поддержания температуры

Чтобы было понятно:

У кнопки есть цикл pressed → pressing → released

Кнопок 4 и из каждого конечного состояния можно перейти в начальное другой кнопки

Поэтому схема выглядит неприятно, по факту, тут 4 линейных структуры, у которых начало и концы соединены друг с другом

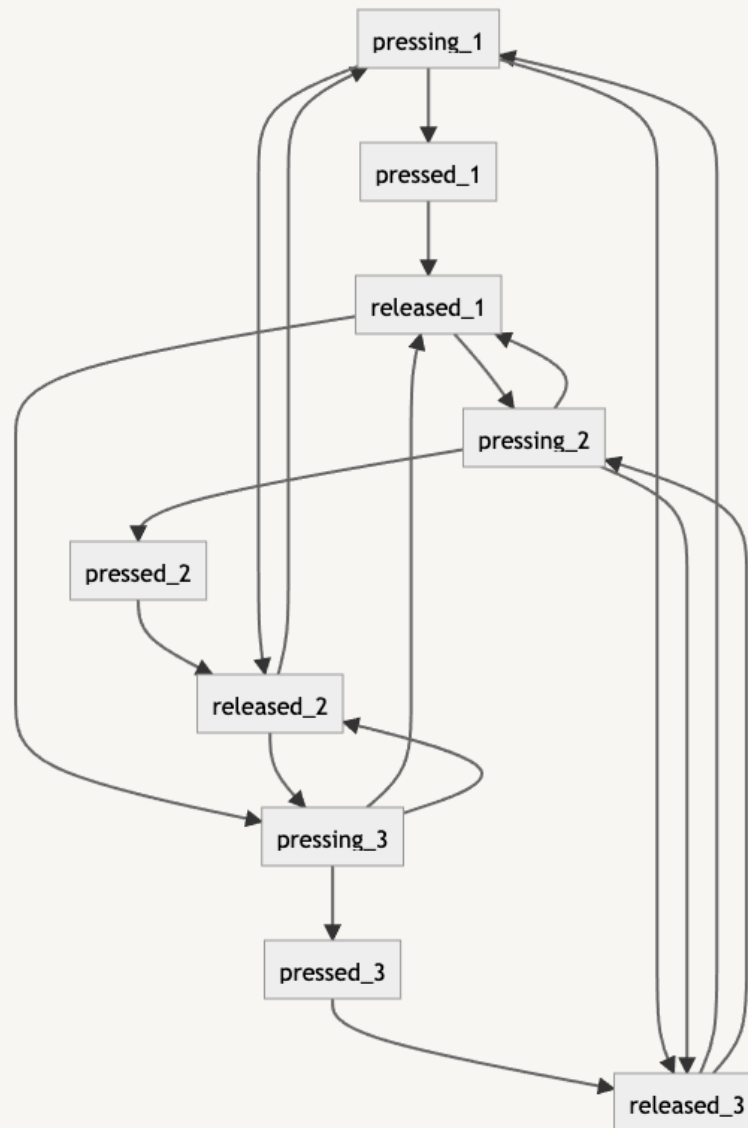
```

graph TD
    a1[pressing_1] --> a2[pressed_1] --> a3[released_1]
    b1[pressing_2] --> b2[pressed_2] --> b3[released_2]
    c1[pressing_3] --> c2[pressed_3] --> c3[released_3]

    a3 --> b1; b1 --> a3
    a3 --> c1; c1 --> a3

    b3 --> c1; c1 --> b3
    b3 --> a1; a1 --> b3

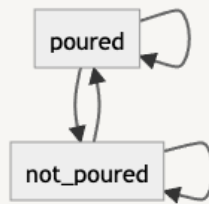
    c3 --> a1; a1 --> c3
    c3 --> b1; b1 --> c3
  
```



Датчик наличия воды

```

graph TD
    a[poured] --> b[not_poured]
    b --> a
    a --> a
    b --> b
  
```

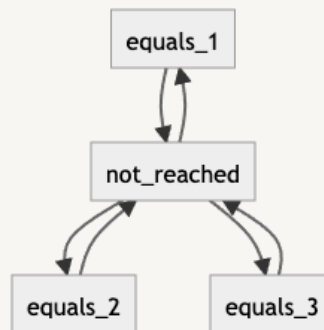



Датчик температуры

```

graph TD
    a[equals_1]
    b[equals_2]
    c[equals_3]
    d[not_reached]

    d --> a & b & c
    a & b & c --> d
  
```

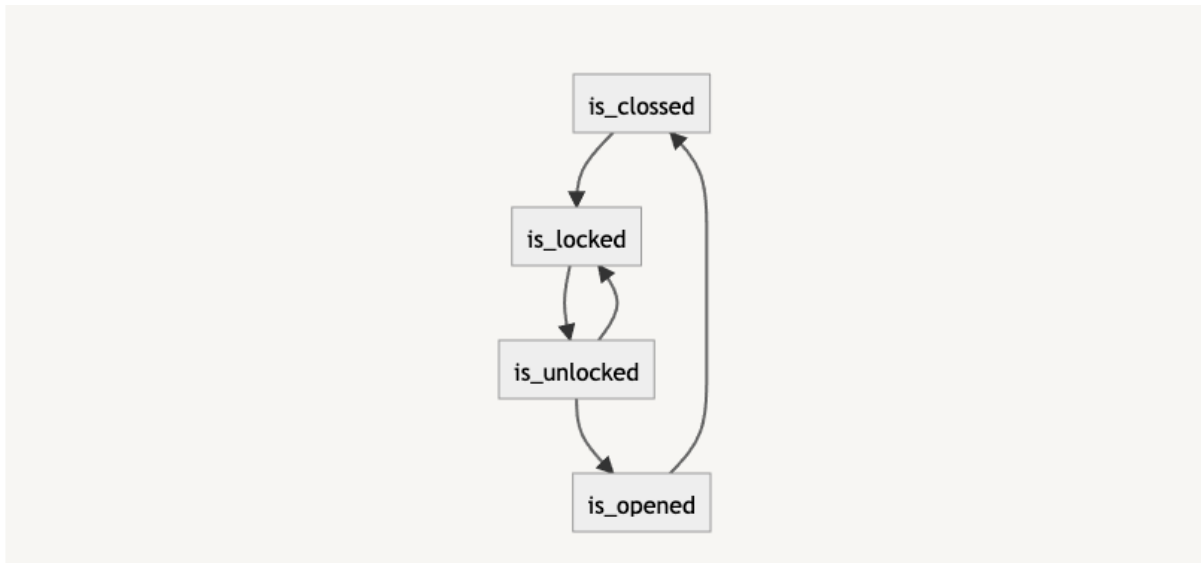


Крышка

```

graph TD
    a[is_closed]
    b[is_opened]
    c[is_locked]
    d[is_unlocked]

    a --> c --> d
    d --> b
    b --> a
    d --> c
  
```



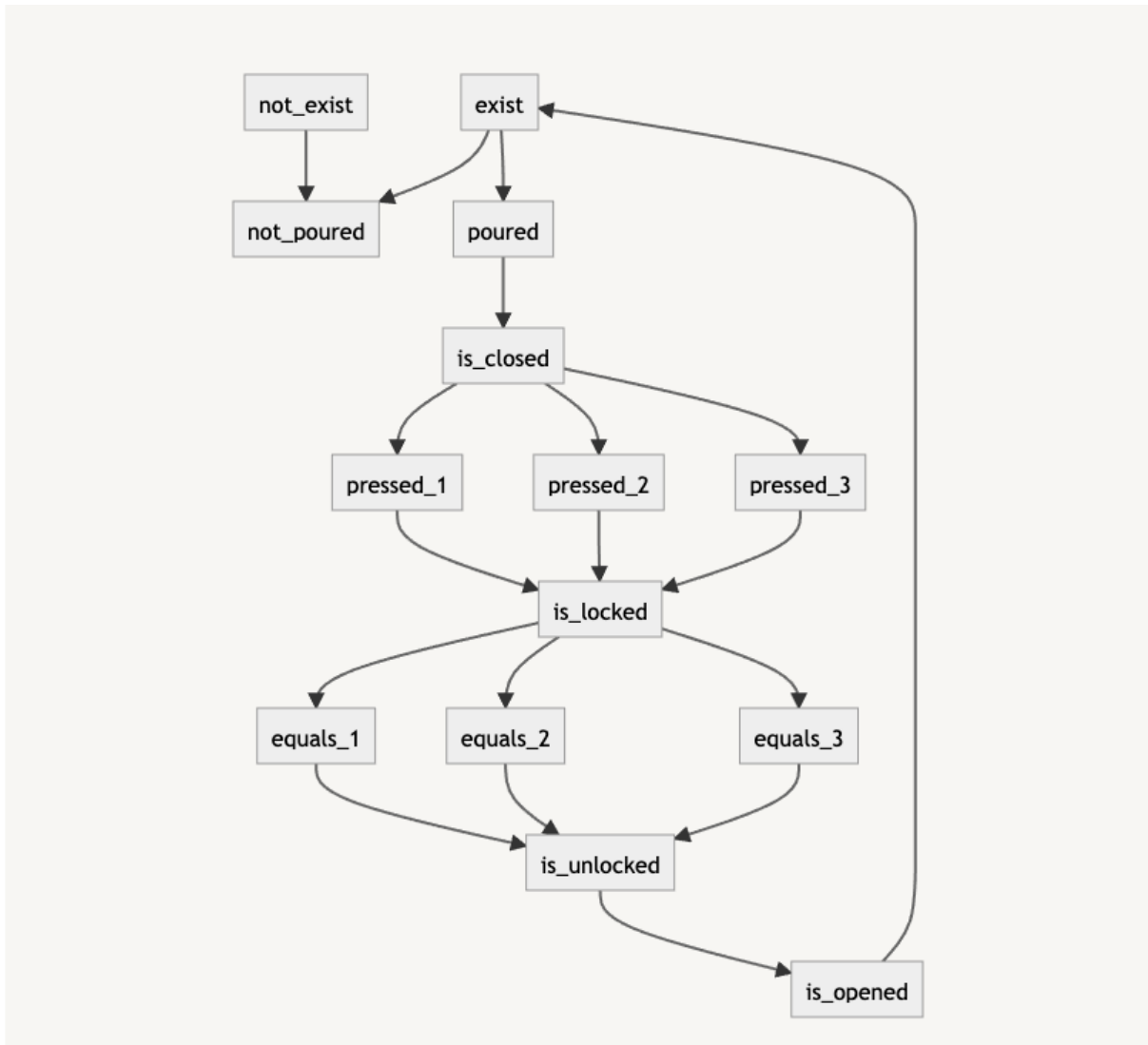
Обработчик сигналов кнопок и датчиков [контроллер]

It's gonna be a little complicated

Instead of 3 additional schemas of different temps added "any"

```

graph TD
    not_exist --> not_poured
    exist --> not_poured
    exist --> poured --> is_closed
    is_closed --> pressed_1 & pressed_2 & pressed_3
    pressed_1 & pressed_2 & pressed_3 --> is_locked --> equals_1 & equals_2 & equals_3 --> is_unlocked
    is_unlocked --> is_opened --> exist
  
```



▼ Homework #3

1. При отсутствии воды нагрев не производится

$G \neg (\text{turn_on} \ \&\& \ \text{not_poured})$

2. Для старта нагрева воды крышка должна быть закрыта

$G \neg (\text{turn_on} \ \&\& \ (\text{is_opened} \ || \ \text{is_unlocked}))$

3. Нагрев начинается после нажатия на одну из кнопок при закрытой крышке и наличии воды

$G (\text{pressed_any} \ \&\& \ (\text{poured} \ \&\& \ \text{is_closed} \ \&\& \ \text{is_locked}) \rightarrow \text{turn_on})$

$\text{pressed_any} = (\text{pressed_1} \ || \ \text{pressed_2} \ || \ \text{pressed_3})$

4. При нагревании или поддержании температуры крышка блокируется

$G (\text{turn_on} \rightarrow F \text{ is_locked})$

5. Крышка остается закрытой до окончания нагрева

$G (\text{is_locked} \rightarrow \text{is_locked } W \text{ turn_off})$

6. При нажатии кнопки “поддержание температуры” термопот поддерживает заданную температуру

$G (\text{pressed_temp} \rightarrow F \text{ equals_some})$

$\text{equals_some} = \text{equals_1} \parallel \text{equals_2} \parallel \text{equals_3}$

$\text{pressed_temp} = ($

$\text{pressed_1} \rightarrow F \text{ pressed_1} \parallel$

$\text{pressed_2} \rightarrow F \text{ pressed_2} \parallel$

$\text{pressed_3} \rightarrow F \text{ pressed_3}) \# \text{ need to push 2 times in a row}$

▼ Homework #4 (alphabets)

Water

- $\text{aWater} = \{\text{exist}, \text{not_exist}\}$
- $\text{WaterInit} = \text{exist} \rightarrow \text{Exist}$
- $\text{Exist} = \text{exist} \rightarrow \text{Exist} \mid \text{not_exist} \rightarrow \text{NotExist}$
- $\text{NotExist} = \text{not_exist} \rightarrow \text{NotExist} \mid \text{exist} \rightarrow \text{Exist}$

Water presence sensor

- $\text{aPresence} = \{\text{poured}, \text{not_poured}\}$
- $\text{PresenceInit} = \text{not_poured} \rightarrow \text{NotPour}$
- $\text{NotPour} = \text{not_poured} \rightarrow \text{NotPoured} \mid \text{poured} \rightarrow \text{Pour}$
- $\text{Pour} = \text{poured} \rightarrow \text{Pour}$

Water heater

- $\text{aHeater} = \{\text{turn_on}, \text{turn_off}\}$

- HeaterInit = turn_off → HeaterOff
- HeaterOff = turn_off → HeaterOff | turn_on → HeaterOn
- HeaterOn = turn_on → HeaterOn | turn_off → HeaterOff

Water temperature

- aTemp = {cold, equals_1, equals_2, equals_3}
- TempInit = cold → TempCold
- TempCold = cold → TempCold | equals_1 → cold → TempCold | cold → equals_1 → TempOne
- TempOne = equals_1 → TempOne | equals_2 → equals_1 → TempOne | equals_1 → equals_2 → TempTwo
- TempTwo = equals_2 → TempTwo | equals_3 → equals_2 → TempTwo | equals_2 → equals_3 → TempThree
- TempThree = equals_3 → TempThree | equals_3 → equals_2 → TempTwo

Water temperature sensor

- aSensor = {not_reached, equals_1, equals_2, equals_3}
- SensorInit = not_reached → SensorLow
- SensorLow = not_reached → SensorLow | equals_1 → SensorOne
- SensorOne = equals_1 → SensorOne | not_reached → SensorLow | equals_2 → SensorTwo
- SensorTwo = equals_2 → SensorTwo | equals_1 → SensorOne | equals_3 → SensorThree
- SensorThree = equals_3 → SensorThree | equals_2 → SensorTwo

Buttons

There's gonna 4 the same buttons, let me provide one to ease the checking process

- aButtons = {pressing, pressed, released}
- ButtonInit = released → Button

- Button = released → Button | pressing → pressed → released → Button

Lid

- aLid = {is_closed, is_locked, is_unlocked, is_opened}
- LidInit = is_opened → LidOpen
- LidOpen = is_opened → LidOpen | is_closed → LidClosed
- LidClosed = is_closed → LidClosed | is_opened → LidOpen | is_locked → LidLocked
- LidLocked = is_locked → LidLocked | is_unlocked → LidClosed

Controller

- aController = {not_exist, not_poured, exist, poured, is_closed, pressed_1, pressed_2, pressed_3, is_locked, equals_1, equals_2, equals_3, is_unlocked, is_opened, wait}
- ContlInit = wait → Pressed
- Pressed = wait → Pressed | pressed_n → (poured → is_closed → is_locked) → WorkN
- WorkN = turn_on → WorkN | equals_n → turn_off → (is_unlocked → is_opened) → Open
- Opened = open → Opened

where equals_n is shortened equivalent for 3 notes

User

- aUser = {absent, pressing_1, pressing_2, pressing_3, open_lid, add_water, decrease_water, close_lid}
- **non-deterministic behaviour**

▼ Homework #5

1. Формализуйте 3 вида трейсов длины 10 Контроллера.
 - a. C1 = <wait, pressed_1, poured, is_closed, is_locked, turn_on, equals_1, turn_off, is_unlocked, is_opened>

- b. $C2 = \langle \text{wait, pressed_2, poured, is_closed, is_locked, turn_on, equals_2, turn_off, is_unlocked, is_opened} \rangle$
 - c. $C3 = \langle \text{wait, pressed_3, poured, is_closed, is_locked, turn_on, equals_3, turn_off, is_unlocked, is_opened} \rangle$
- 2. Постройте сужение (\upharpoonright) трейсов относительно событий другого процесса.
 - a. $C1 \upharpoonright aLid = \langle \text{is_closed, is_locked, is_unlocked, is_opened} \rangle \quad \# \text{ fixed}$
 - b. $C1 \upharpoonright aWater = \langle \rangle$
 - c. $C1 \upharpoonright aPressence = \langle \text{poured} \rangle$
 - d. $C1 \upharpoonright aHeater = \langle \text{turn_on, turn_off} \rangle$
 - e. $C1 \upharpoonright aTemp = \langle \text{equals_1} \rangle$
 - f. $C1 \upharpoonright aSensor = \langle \text{equals_1} \rangle$
 - g. $C1 \upharpoonright aButton = \langle \text{pressed_1} \rangle$
- 3. Сравните исходные трейсы и их сужения
 - a. Они несравнимы, ибо стартовое состояние есть только у контроллера \rightarrow его нельзя сравнить с его сужением
- 4. Выберите короткий процесс и формализуйте его трейс.
 - a. $\bigcup m \geq 0 \bigcup n \geq 0 \bigcup k \geq 0 \{ s \mid s \leq (\langle \text{turn_on} \rangle^m \wedge \langle \text{turn_off} \rangle^n) k \} \quad \# \text{ fixed}$
- 5. Запишите процесс P after s , где P -- Контроллер, а s -- трейс длины 3.
 $s = \langle \text{wait, pressed_1, poured} \rangle$
 $P/s = \text{is_closed} \rightarrow \text{is_locked} \rightarrow \text{WorkN}$ (in this case, supposed to be WorkOne)
- 6. Определите, являются ли циклическими в CSP-смысле все процессы Вашей системы. Для одного из трейсов п.1 найдите кратчайший трейс, который приводит к исходному процессу: $\exists t : P / (c1^t) = P$.
 $C1 = \langle \text{wait, pressed_1, poured, is_closed, is_locked, turn_on, equals_1, turn_off, is_unlocked, is_opened} \rangle$
 $t = \langle \text{not_poured, wait} \rangle$

▼ Homework #6

1. Формализуйте спецификацию 3 свойств Контроллера в CSP-терминах.

- a. $\text{ThermopotSafety} = ((\text{tr} \downarrow \text{equals}) \leq (\text{tr} \downarrow \text{poured})) \# \text{нагрев не происходит без воды}$
- b. $\text{NormalActivity} = (\text{tr} \downarrow \text{poured}) = (\text{tr} \downarrow \text{is_closed}) \rightarrow (\text{tr} \downarrow \text{equals_n}) \leq (\text{tr} \downarrow \text{pressed_n})$
- c. $\text{PerfectUser} = ((\text{tr} \downarrow \text{is_opened}) \leq (\text{tr} \downarrow \text{is_unlocked})) \# \text{Пытается открыть крышку, когда она не под замком}$

2. Задайте параллельное исполнение двух процессов своей системы с пересекающимися алфавитами с помощью законов для параллельного исполнения опишите $P1 \parallel P2$ без использования \parallel . Должны быть приведены процессы $P1$, $P2$ и $P1 \parallel P2$.

$P = (a \rightarrow P1 \mid b \rightarrow P2)$ и $Q = (c \rightarrow Q1 \mid d \rightarrow Q2)$. $c \notin aP$.
Тогда $P \parallel Q = (a \rightarrow P1 \parallel Q) \mid (b \rightarrow P2 \parallel Q) \mid (d \rightarrow P \parallel Q2)$.

a. **Pressed** = wait \rightarrow Pressed | pressed_n \rightarrow (poured \rightarrow is_closed \rightarrow is_locked) \rightarrow WorkN

a. **Pressed** = pressed \rightarrow poured \rightarrow is_closed \rightarrow is_locked \rightarrow WorkN | wait \rightarrow Pressed

b. **LidClosed** = is_locked \rightarrow LidLocked | is_opened \rightarrow LidOpen

c. *By definition*

Pressed \parallel **LidClosed** = (pressed \rightarrow poured \rightarrow is_closed \rightarrow is_locked \rightarrow WorkN | wait \rightarrow Pressed) \parallel (is_locked \rightarrow LidLocked | is_opened \rightarrow LidOpen)

Applied L7: (c in aP) \Rightarrow

Pressed \parallel **LidClosed** = (pressed \rightarrow poured \rightarrow is_closed \rightarrow is_locked \rightarrow WorkN \parallel LidClosed) | (wait \rightarrow Pressed \parallel LidClosed) | (is_opened \rightarrow Pressed \parallel LidOpen) \Rightarrow

Removing “ \parallel ” 3 times \Rightarrow

Pressed \parallel **LidClosed** = (pressed \rightarrow poured \rightarrow is_closed \rightarrow ((is_locked \rightarrow WorkN) | (is_locked \rightarrow LidClosed))) | ((wait \rightarrow Pressed) | (wait \rightarrow LidClosed)) | ((is_opened \rightarrow Pressed) | (is_opened \rightarrow LidOpen)) \Rightarrow

Removing unnecessary brackets \Rightarrow

Pressed \parallel **LidClosed** = pressed \rightarrow poured \rightarrow is_closed \rightarrow ((is_locked \rightarrow WorkN) | (is_locked \rightarrow LidClosed)) | (wait \rightarrow Pressed | wait \rightarrow LidClosed) | (is_opened \rightarrow

$\text{Pressed} \mid \text{is_opened} \rightarrow \text{LidOpen}) \Rightarrow$

Separating the equations \Rightarrow

Pressed \parallel LidClosed = $\text{pressed} \rightarrow \text{poured} \rightarrow \text{is_closed} \rightarrow \text{is_locked} \rightarrow \text{WorkN} \mid$
 $\text{pressed} \rightarrow \text{poured} \rightarrow \text{is_closed} \rightarrow \text{is_locked} \rightarrow \text{LidClosed} \mid \text{wait} \rightarrow \text{Pressed} \mid \text{wait} \rightarrow$
 $\text{LidClosed} \mid \text{is_opened} \rightarrow \text{Pressed} \mid \text{is_opened} \rightarrow \text{LidOpen}$

▼ Homework #7

1. Записать процесс BOOL со стр. 22 лекции 7.

$\text{DD} = (\text{setorange} \rightarrow \text{O} \mid \text{setlemon} \rightarrow \text{L})$

$\text{O} = (\text{orange} \rightarrow \text{O} \mid \text{setlemon} \rightarrow \text{L} \mid \text{setorange} \rightarrow \text{O})$

$\text{L} = (\text{lemon} \rightarrow \text{L} \mid \text{setorange} \rightarrow \text{O} \mid \text{setlemon} \rightarrow \text{L})$

X5. The behaviour of a Boolean variable used by a computer program.

$\text{BOOL} = f(\text{DD})$

$f(\text{setorange}) = \text{assign0}$

$f(\text{setlemon}) = \text{assign1}$

$f(\text{orange}) = \text{fetch0}$

$f(\text{lemon}) = \text{fetch1}$

$\text{BOOL} = (\text{assign0} \rightarrow \text{O} \mid \text{assign1} \rightarrow \text{L})$

$\text{O} = (\text{fetch0} \rightarrow \text{O} \mid \text{assign1} \rightarrow \text{L} \mid \text{assign0} \rightarrow \text{O})$

$\text{L} = (\text{fetch1} \rightarrow \text{O} \mid \text{assign0} \rightarrow \text{O} \mid \text{assign1} \rightarrow \text{L})$

2. Определить процессы системы верхнего уровня, которые получаются изменением символов. Записать одну изменяющую функцию.

a. $\text{ButtonInit}, \text{LidInit}$

b. $\text{ButtonInit2} = f(\text{ButtonInit3})$

$f(\text{pressed_2}) = \text{pressed_3}$

$f(\text{pressing_2}) = \text{pressing_3}$

$f(\text{released_2}) = \text{released_3} \quad () \Rightarrow \text{can choose any of 3 buttons}$

3. Построить полное описание своей параллельной системы с использованием переименования и/или разметки процессов.

Отметить изменения в CSP-записях процессов (скорее всего, это контроллер) в результате переименований/разметки.

$\text{Termopot} = \text{ContlInit} \parallel \text{WaterInit} \parallel \text{HeaterInit} \parallel \text{LidInit} \parallel (\text{a} : \text{buttonInit}) \parallel (\text{b} : \text{waterTemperature})$
 $\parallel \text{User} \parallel \text{PresenceInit} \parallel \text{SensorInit}$

Controller

- $aController = \{not_exist, not_poured, exist, poured, is_closed, pressed_1, pressed_2, pressed_3, is_locked, equals_1, equals_2, equals_3, is_unlocked, is_opened, wait\}$
- $ContInit = wait \rightarrow Pressed$
- $Pressed = wait \rightarrow Pressed \mid a.pressed_n \rightarrow (poured \rightarrow is_closed \rightarrow is_locked) \rightarrow WorkN$
- $WorkN = turn_on \rightarrow WorkN \mid b.equals_n \rightarrow turn_off \rightarrow (is_unlocked \rightarrow is_opened) \rightarrow Open$
- $Opened = open \rightarrow Opened \mid a.pressed_n \rightarrow Pressed$

where $equals_n$ is shortened equivalent for 3 notes

4. Записать модифицированный процесс FOOTMAN. Доказать, что этот процесс гарантирует сытость философов (+5).

The effective solution is to buy more forks and plenty of spaghetti.

To guarantee that a seated philosopher will eventually eat modify the behaviour of footman:
Having helped a philosopher to his seat he waits until that philosopher has picked up both forks before he allows either of his neighbours to sit down

FOOT(j) - defines the behaviour of footman with j philosophers

$$\begin{aligned}
 FOOT_0 &= (x : D \rightarrow FOOT_1) \\
 FOOT_j &= (x : D \rightarrow FOOT_{j+1} \mid y : U \rightarrow FOOT_{j-1}) \quad \text{for } j \in \{1, 2, 3\} \\
 FOOT_4 &= (y : U \rightarrow FOOT_3) \\
 \blacksquare \quad U &= \bigcup_{i=0}^4 \{i.\text{gets up}\} \quad D = \bigcup_{i=0}^4 \{i.\text{sits down}\}
 \end{aligned}$$

Let's define **philosopher** and **fork** behaviour

Phil = sit \rightarrow get_one_fork \rightarrow get_second_fork \rightarrow put_one_fork \rightarrow put_second_fork \rightarrow stand \rightarrow **Phil**

Fork = left_up \rightarrow left_down \rightarrow Fork \mid right_up \rightarrow right_down \rightarrow **Fork**

Получаем, следующее поведение для лакея:

1. Лакей ждет философа
2. Лакей садит философа за стол и не дает сесть никому рядом с ним, пока философ не возьмет обе вилки в руки. (В это время лакей может посадить

философа через место от философа, тогда конкуренция за вилки не случится)

Запишем поведение лакея:

Footman_init = wait → Footman

Footman = wait → Footman | sit_phil → wait_for_philospher_taking_forks_1 →
take_first_fork → wait_for_philospher_taking_forks_2 → take_second_fork → Footman

Где wait_for_philospher_taking_forks - означает, что лакей не садит рядом с философом никого, пока тот не возьмет обе вилки

Требуется доказать **сытость философов**, следовательно, нужно определить, что это такое.

1. Все философы могут сесть за стол
2. Каждый философ может дождаться своей очереди для того, чтобы поесть == не возникает deadlock'a

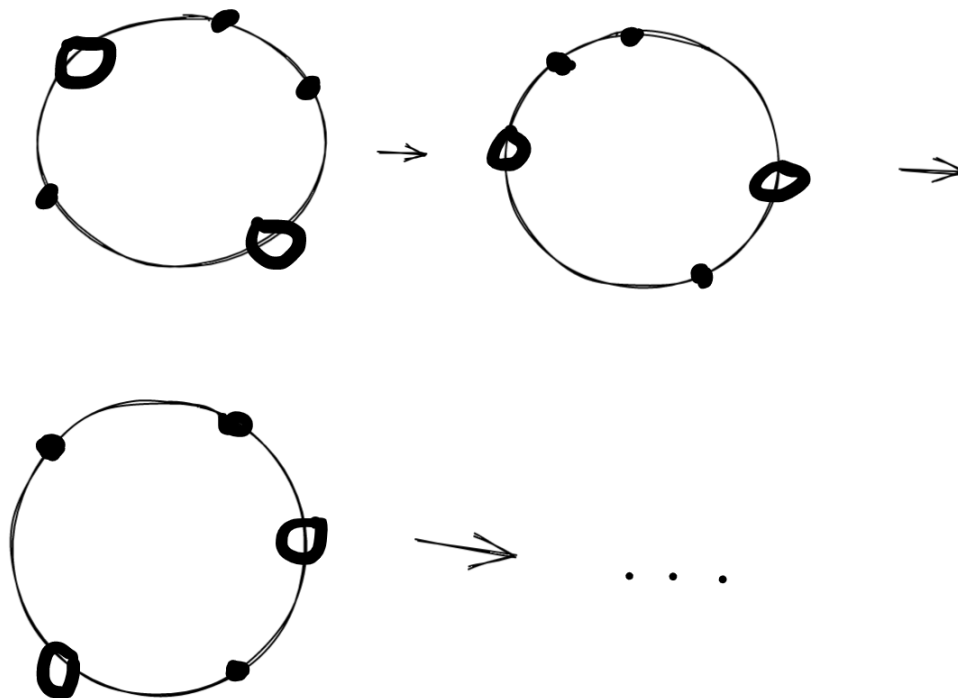
Доказательство:

1. Пока за столом никого нет, лакей может посадить философа на любое место, тогда философ может взять обе вилки в руки и не бороться за них.
2. Когда за столом сидит философ (и берет вилки или ест), лакей садит другого философа через место от философа, чтобы второй философ мог взять обе вилки и поесть (пока философ не возьмет вилки в руки рядом с ним нельзя садить другого)
3. Оставшиеся философы будут садиться рядом с уже обедающими философами (потому что пока они не взяли обе вилки в руки, садить рядом с ними нельзя) и брать вилку как только она освобождается от соседнего философа (он уходит и освобождает 2 вилки). В это время (пока философ сидит с одной или вовсе без вилок в руках) рядом с ним по условию нельзя садить других ⇒ deadlock не возникает (то есть нет варианта, когда все философы будут вечно сидеть и ждать вилки)

Так же это означает, что за стол можно посадить всех философов по очереди → первые два философа - очевидно из пункта 1 и 2, еще двух садим рядом с ними, оставшийся ждет. Когда один из философов поест и встанет, сидящий рядом с ним получает вилку и начинает есть, тогда последнего стоящего философа можно посадить за стол и таким образом запустить цикл опять

Таким образом, получаем, что каждый философ сможет насытиться

Если как-то пытаться нарисовать, то получаем такую картинку:



Где большие кружки - обедающие философы, которые меняются через итерацию (ну тут для наглядности представлено 2 итерации, то есть поел 2 философа (просто было трудно рисовать кружки))

▼ Homework #8

1. Найти в своей системе процессы, параллельная комбинация которых точно не имеет тупиков (deadlocks).

1. **Contorller** имеет пересечение со всеми процессами системы мощностью > 1 , кроме **User**, **Button** → deadlock не будет возникать с процессами User и Button, со всеми остальными deadlock может возникнуть
2. Параллельная комбинация всех остальных процессов (кроме **Water Temperature** и **Water temperature sensor**, так как они имеют пересечение алфавитов, поэтому их параллельная комбинация может привести к deadlock) не имеет deadlock'ов

2. Записать процесс пользователя (внешней среды), используя строго недетерминированный выбор.

- User:
 - $aUser = \{absent, pressing_1, pressing_2, pressing_3, open_lid, add_water, decrease_water, close_lid\}$
 - $UserInit = absent \rightarrow User$
 - $User_process = (absent \rightarrow User) \Pi$
 $(lid_process) \Pi$
 $(pressing_1 \rightarrow process) \Pi$
 $(pressing_2 \rightarrow process) \Pi$
 $pressing_3 \rightarrow process)$

 $process = (lid_process \Pi (pressing_1 \rightarrow process) \Pi (pressing_2 \rightarrow process) \Pi$
 $(pressing_3 \rightarrow process))$

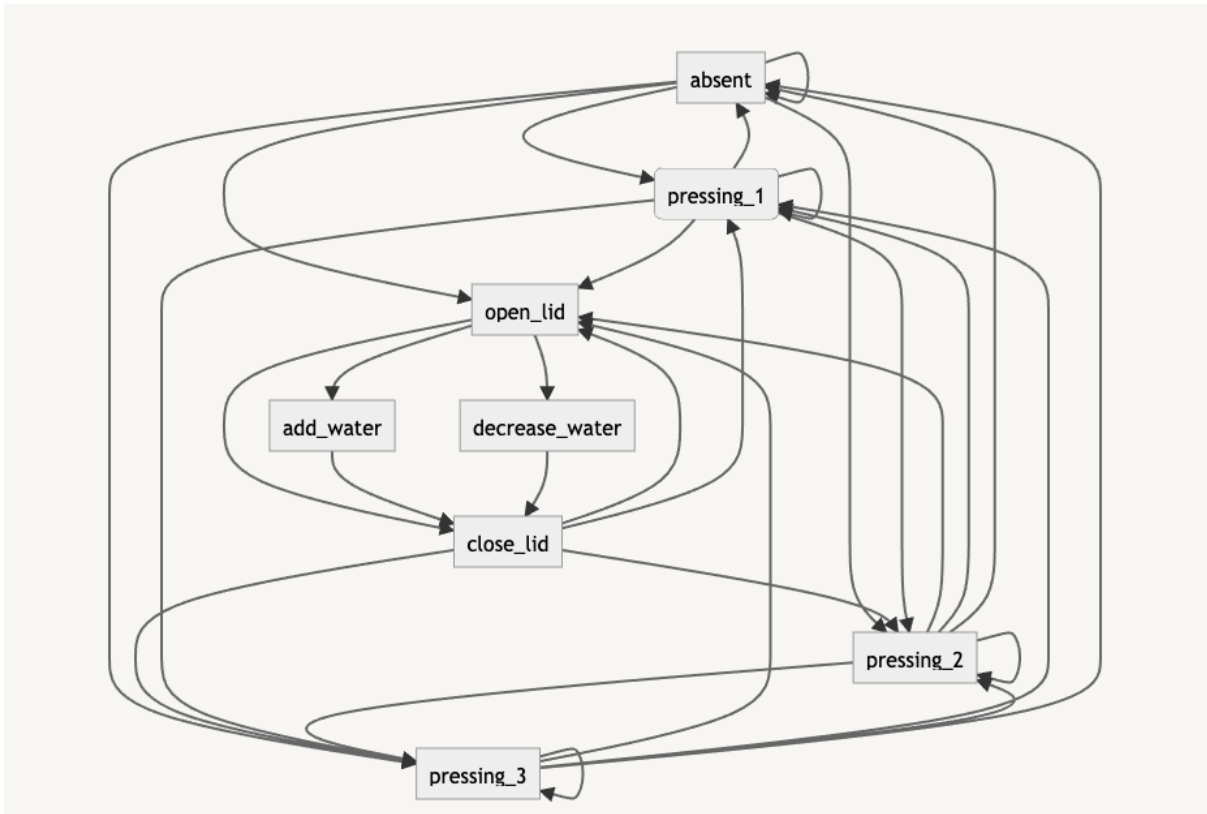
 $// pressing_n \text{ перед } process \text{ дает понять в каком состоянии находится}$
 система

 $lid_process = open_lid \rightarrow ((add_water \rightarrow CloseLid) \Pi (decrease_water \rightarrow CloseLid)$
 $\Pi CloseLid)$

 $CloseLid = close_lid \rightarrow process$

3. Записать спецификацию недетерминированного процесса пользователя (среды) с учётом отказов

$$CloseLid = (tr \downarrow open_lid > tr \downarrow close_lid \Rightarrow close_lid \notin ref)$$



▼ Homework #9

1. Выбрать процесс и множество его событий, в котором их сокрытие **не приводит к дивергенции**. Записать это множество событий, и процесс, получившийся в результате их сокрытия.

- $aButtons = \{pressing, pressed, released\}$
- $ButtonInit = released \rightarrow Button$
- $Button = released \rightarrow Button \mid pressing \rightarrow pressed \rightarrow released \rightarrow Button$

$C = \{pressed\}$

$ButtonInit \backslash C = released \rightarrow Button$

$Button \backslash C = (released \rightarrow Button \backslash C) \sqcap (pressing \rightarrow released \rightarrow Button \backslash C)$

2. Выбрать процесс и множество его событий (не равное алфавиту), в котором их сокрытие **приводит к дивергенции**. Записать это множество событий, и процесс,

получившийся в результате их сокрытия.

- $aButtons = \{pressing, pressed, released\}$
- $ButtonInit = released \rightarrow Button$
- $Button = released \rightarrow Button \mid pressing \rightarrow pressed \rightarrow released \rightarrow Button$

$C = \{released\}$

$ButtonInit \setminus C = Button \setminus C$

$Button \setminus C = (Button \setminus C)$ - Дивергентный $\Rightarrow ButtonInit \setminus C$ - дивергентный (\Rightarrow убирать нельзя)

▼ Homework #10

1. В основе примера лежит идея X8 из лекции

X8. *BUFFER sat right \leq left*

$$\begin{aligned}
 & BUFFER = P \\
 & P_{\langle \rangle} = left ? x \rightarrow P_{\langle x \rangle} \\
 & P_{\langle x \rangle \wedge s} = (left ? y \rightarrow P_{\langle x \rangle \wedge s \wedge y} \mid right ! x \rightarrow P_s)
 \end{aligned}$$

- This is the behaviour of a transparent communications protocol
 - the guarantee of delivering on the right
 - only those messages which have been submitted on the left, and
 - in the same order.
- The protocol achieves this in spite of the facts that
 - the place where the messages are submitted is separated from the place where they are received, and
 - the communications medium connecting the two places is somewhat unreliable.

1. Предположим, что пользователь может нажимать последовательно кнопки 12 и 21 быстрее, чем контроллер может обработать эти сигналы. Модифицировать процессы системы так, чтобы Buttons обрабатывал последовательность сигналов User и выдавал для Controller значение последнего вызова.

- Buttons
 - $aButtons = \{pressing, pressed, released\}$
 - $ButtonInit = released \rightarrow Button$
 - $Button = released \rightarrow Button \mid pressing \rightarrow pressed \rightarrow released \rightarrow Button$
- Buttons (modified)

- ButtonInit = released → Button
- Button = u ? pressing → c ! pressed → released
- Buttons (modified with brackets)
 - ButtonInit = released → Butsj
 - Buts = released → Buts | u ? pressing_1 → Buts<p1> | u ? pressing_2 → Buts<p2> | u ? pressing_3 → Buts<p3>
 - Buts<px> = u ? pressing_1 → Buts<p1> | u ? pressing_2 → Buts<p2> | u ? pressing_3 → Buts<p3> | c ! pressed_x → released → Buts
- **Controller**
 - aController = {not_exist, not_poured, exist, poured, is_closed, pressed_1, pressed_2, pressed_3, is_locked, equals_1, equals_2, equals_3, is_unlocked, is_opened, wait}
 - ContInit = wait → Pressed
 - Pressed = wait → Pressed | pressed_n → (poured → is_closed → is_locked) → WorkN
 - WorkN = turn_on → WorkN | equals_n → turn_off → (is_unlocked → is_opened) → Open
 - Opened = open → Opened

where equals_n is shortened equivalent for 3 notes
- **Controller** (modified)
 - Continit = wait → Cont
 - Cont = wait → Cont | c ? pressed_1 → is_locked → equals_1 → is_unlocked → Prep | c ? pressed_2 → is_locked → equals_2 → is_unlocked → Prep | c ? pressed_3 → is_locked → equals_3 → is_unlocked → Prep
 - Prep = is_opened → exist → poured → is_closed → Cont
- **User:**
 - aUser = {absent, pressing_1, pressing_2, pressing_3, open_lid, add_water, decrease_water, close_lid}
 - UserInit = absent → User
 - User_process = (absent → User) Π
(lid_process) Π
(pressing_1 → process) Π


```

(pressing_2 → process) Π
pressing_3 → process)

process = (lid_process Π (pressing_1 → process) Π (pressing_2 → process) Π
(pressing_3 → process))

    // pressing_n перед process дает понять в каком состоянии находится
    система

lid_process = open_lid → ((add_water → CloseLid) Π (decrease_water → CloseLid)
Π CloseLid)

CloseLid = close_lid → process

```

- User (modified)
 - UserInit = absent → User
 - User = (absent → User) Π (lid_process) Π (u ! pressing_1 → process) Π (u ! pressing_2 → process) Π (u ! pressing_3 → process)
 - process = (lid_process) Π ((u ! pressing_1 → process) Π (u ! pressing_2 → process) Π (u ! pressing_3 → process))
 - lid_process = open_lid → ((add_water → CloseLid) Π (decrease_water → CloseLid) Π CloseLid)
 - CloseLid = close_lid → process

▼ Homework #11

Задать CSP-спецификацию параллельного взаимодействия с подчинением для пары процессов системы.

(li: Lid//Pressed)

ContInit = wait → Pressed

- Pressed = wait → Pressed | pressed_n → (li.right ? opened → poured → li.left ! closing → li.right ? is_closed → li.left ! locking → li.right ? is_locked) → WorkN
- WorkN = turn_on → WorkN | equals_n → turn_off → (li.left ! unlocking → li.right ? is_unlocked → li.left ! opening → li.right ? is_opened) → Pressed

LidInit = is_opened → LidOpen

- LidOpen = is_opened → LidOpen | left ? closing → right ! is_closed → LidClosed
- LidClosed = is_closed → LidClosed | left ? opening → right ! is_opened → LidOpen | left ? locking → right ! is_locked → LidLocked
- LidLocked = is_locked → LidLocked | left ? unlocking → right ! is_unlocked → LidClosed

▼ Homework #12

Процессы получились достаточно разделенными друг от друга и пересечений не так много, поэтому так:

Buttons

- aButtons = {pressing, pressed, released}
- ButtonInit = released → Button
- Button = released → Button Δ pressing → pressed → released → Button (User_process)

Water temperature

- aTemp = {cold, equals_1, equals_2, equals_3}
- TempInit = cold → TempCold
- TempCold = cold → TempCold | equals_1 → cold → TempCold | cold → equals_1 → SensorOne (SensorOne)
- TempOne = equals_1 → SensorOne Δ equals_2 → equals_1 → SensorOne Δ equals_1 → SensorTwo (SensorOne)

Аналогично для TempTwo и TempThree

Controller

- $aController = \{not_exist, not_poured, exist, poured, is_closed, pressed_1, pressed_2, pressed_3, is_locked, equals_1, equals_2, equals_3, is_unlocked, is_opened, wait\}$
- $ContInit = wait \rightarrow Pressed$
- $Pressed = wait \rightarrow Pressed \Delta User_process \Delta poured \rightarrow Pour \Delta is_closed \rightarrow LidClosed \Delta is_locked \rightarrow LidLocked \rightarrow WorkN (User_process; Pour; LidClosed; LidLocked)$
- $WorkN = turn_on \rightarrow WorkN \Delta equals_n \rightarrow SensorN \Delta turn_off \rightarrow HeaterOff \rightarrow (is_unlocked \Delta is_opened \rightarrow LidOpen) \rightarrow Open (SensorN; HeaterOff; LidOpen)$

User

- $aUser = \{absent, pressing_1, pressing_2, pressing_3, open_lid, add_water, decrease_water, close_lid\}$
- $UserInit = absent \rightarrow User$
- $User_process = (absent \rightarrow User) \Pi (lid_process) \Pi (pressing_1 \rightarrow process) \Pi (pressing_2 \rightarrow process) \Pi pressing_3 \rightarrow process$
 - $process = (lid_process \Pi (pressing_1 \rightarrow process) \Pi (pressing_2 \rightarrow process) \Pi (pressing_3 \rightarrow process))$
 - $lid_process = open_lid \rightarrow ((add_water \rightarrow CloseLid) \Pi (decrease_water \rightarrow CloseLid) \Pi CloseLid)$
 - $CloseLid = close_lid \rightarrow process$

▼ Homework #13

Найти процесс системы, который могут разделить два пользователя. Записать соответствующую модификацию этих процессов с запросами и освобождениями.

В данной системе пользователи могут разделить процесс нажатия кнопки

b: Button // User_ini ||| User_ini

- $ButtonInit = released \rightarrow Button$
- $Button = released \rightarrow Button |$
 $acquire \rightarrow \mu X \bullet (left ? pressing \rightarrow toCont ! pressed \rightarrow released \rightarrow X | release \rightarrow Button$
 $pressing \rightarrow pressed \rightarrow released \rightarrow Button$

- $\text{User_ini} = \text{absent} \rightarrow (\text{User_ini} \sqcap \text{pressing1} \rightarrow \text{User1} \sqcap \text{pressing2} \rightarrow \text{User2})$
- $\text{User1} = \text{acquire} \rightarrow \mu X \bullet (\text{b: left ! pressing} \rightarrow \text{b: left ! pressed} \rightarrow \text{b: left ! releasing} \rightarrow X \sqcap \text{release} \rightarrow \text{User12}) \sqcap \text{absent} \rightarrow \text{User_ini}$
- $\text{User12} = (\text{acquire} \rightarrow \mu X \bullet (\text{b: left ! pressing} \rightarrow \text{b: left ! pressed} \rightarrow \text{b: left ! released} \rightarrow X \sqcap \text{release} \rightarrow \text{User_ini})) \sqcap \text{User2}$
- $\text{User1} = \text{acquire} \rightarrow \mu X \bullet (\text{b: left ! pressing} \rightarrow \text{b: left ! pressed} \rightarrow \text{b: left ! releasing} \rightarrow X \sqcap \text{release} \rightarrow \text{User21}) \sqcap \text{absent} \rightarrow \text{User_ini}$
- $\text{User21} = (\text{acquire} \rightarrow \mu X \bullet (\text{b: left ! pressing} \rightarrow \text{b: left ! pressed} \rightarrow \text{b: left ! released} \rightarrow X \sqcap \text{release} \rightarrow \text{User_ini})) \sqcap \text{User1}$