



# Verification in Isabelle/HOL of Hopcroft's algorithm for minimizing DFAs including runtime analysis

Vincent Trélat

supervised by Prof. Dr. Tobias Nipkow and Dr. Peter Lammich

Technical University of Munich  
Chair for Logic and Verification

September 2023

# Outline

## 1. Living in Munich

1.1 The city

1.2 Technical University of Munich

## 2. The Isabelle Refinement Framework

## 3. Hopcroft's algorithm

3.1 DFA minimization by example

3.2 Refinement steps

## 4. Conduction of the project

4.1 Gantt chart

4.2 Statistics



Figure: Location of Munich

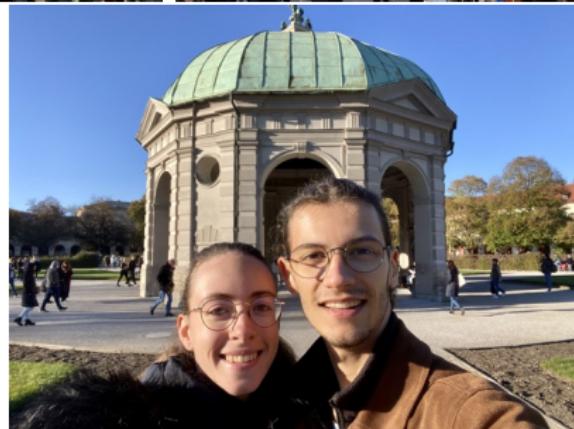


Figure: Some photos of Munich



Figure: Technical University of Munich (TUM), Garching campus

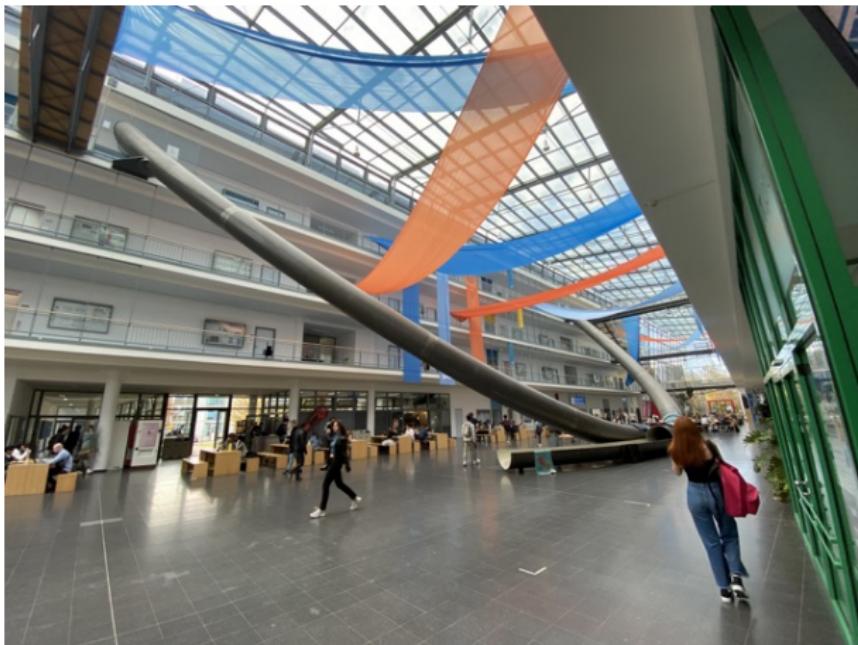


Figure: Technical University of Munich (TUM), Garching campus

















## 1. Living in Munich

1.1 The city

1.2 Technical University of Munich

## 2. The Isabelle Refinement Framework

## 3. Hopcroft's algorithm

3.1 DFA minimization by example

3.2 Refinement steps

## 4. Conduction of the project

4.1 Gantt chart

4.2 Statistics

# Understanding Refinement

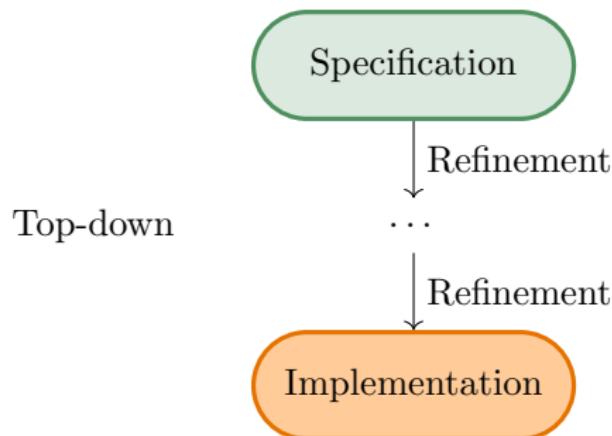
## Definition 1

Refinement is a systematic process of refining a high-level abstract specification into a concrete implementation.

# Understanding Refinement

## Definition 1

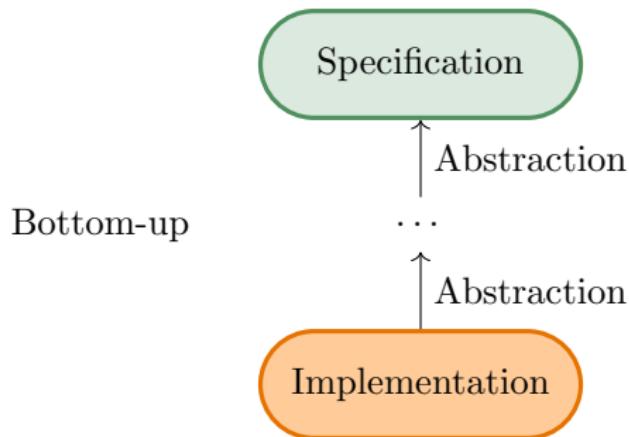
Refinement is a systematic process of refining a high-level abstract specification into a concrete implementation.



# Understanding Refinement

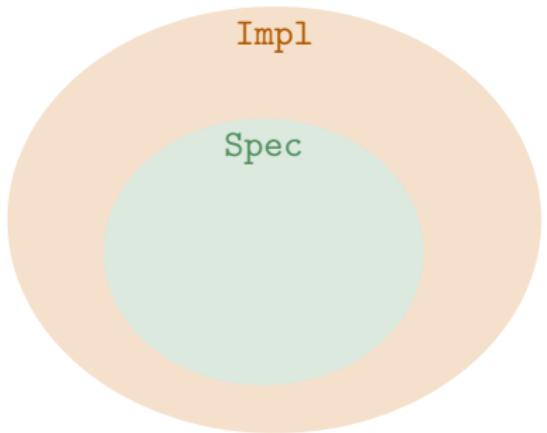
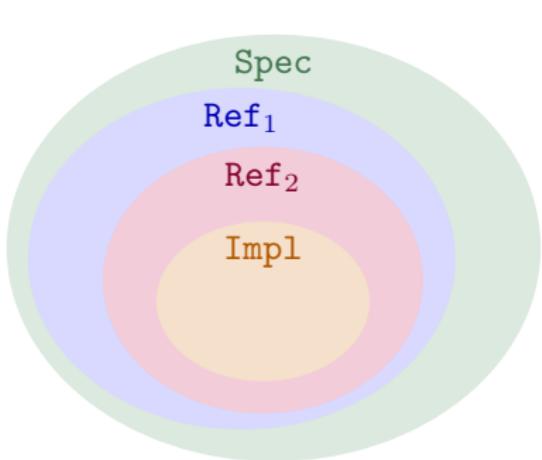
## Definition 1

Refinement is a systematic process of refining a high-level abstract specification into a concrete implementation.



Each refinement step preserves the intended behavior.

$$\text{Spec} \rightarrow \text{Ref}_1 \rightarrow \text{Ref}_2 \rightarrow \text{Impl}$$



# The Isabelle Refinement Framework

## Isabelle Refinement Framework

- Stepwise refinement approach to verified program development
- Formal and mathematical
- Ensures correctness at each step

Comes with the Isabelle Collection Framework, which provides an extensive library of reusable verified functional data structures (for data refinement).

## 1. Living in Munich

1.1 The city

1.2 Technical University of Munich

## 2. The Isabelle Refinement Framework

## 3. Hopcroft's algorithm

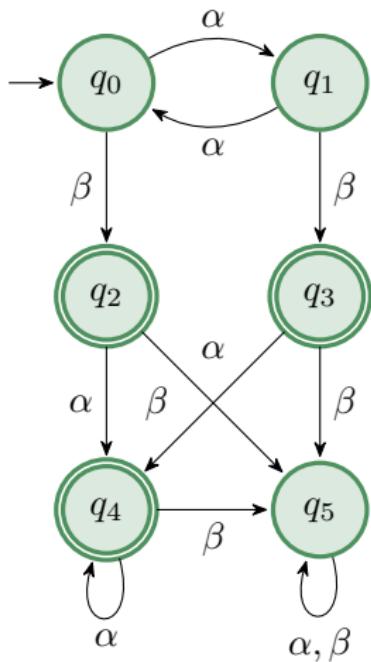
3.1 DFA minimization by example

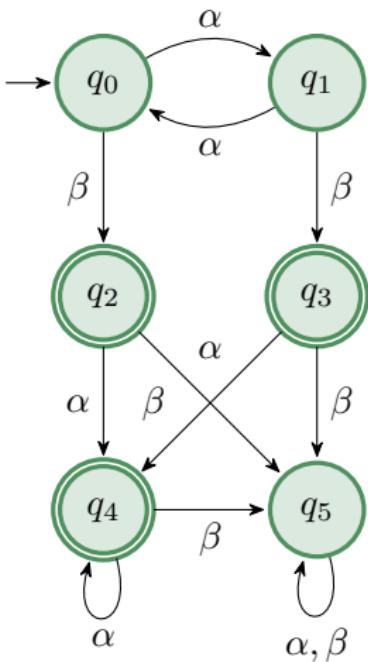
3.2 Refinement steps

## 4. Conduction of the project

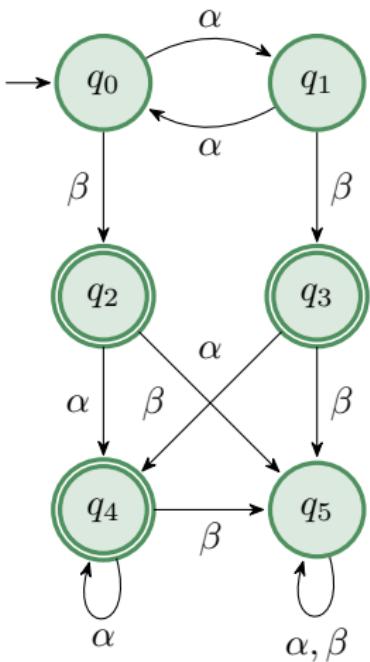
4.1 Gantt chart

4.2 Statistics

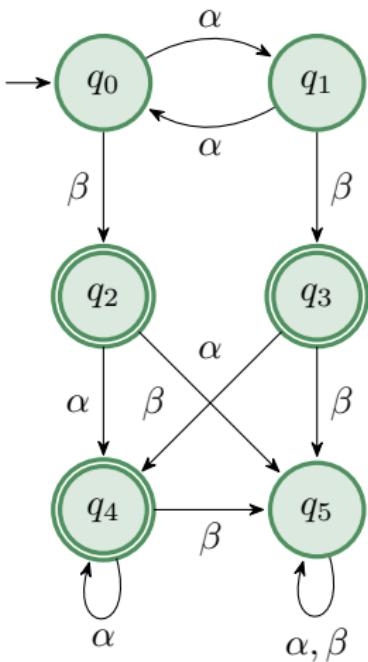




- Successively partitions the set of states into equivalence classes



- Successively partitions the set of states into equivalence classes
- Initial partition: accepting and non-accepting states



- Successively partitions the set of states into equivalence classes
- Initial partition: accepting and non-accepting states
- Each iteration: pick a splitter and split all blocks of the current partition

**Data:** a DFA  $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$

**if**  $\mathcal{F} = \emptyset \vee \mathcal{Q} \setminus \mathcal{F} = \emptyset$   
**return**  $\mathcal{Q}$

**else**

$\mathcal{P} := \{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}$

$\mathcal{W} := \{(a, \min\{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}), a \in \Sigma\}$

**while**  $\mathcal{W} \neq \emptyset$  **do**

Pick  $(a, C)$  from  $\mathcal{W}$  and remove it

**forall**  $B \in \mathcal{P}$  **do**

Split  $B$  with  $(a, C)$  into  $B_0$  and  $B_1$

$\mathcal{P} := (\mathcal{P} \setminus \{B\}) \cup \{B_0, B_1\}$

**forall**  $b \in \Sigma$  **do**

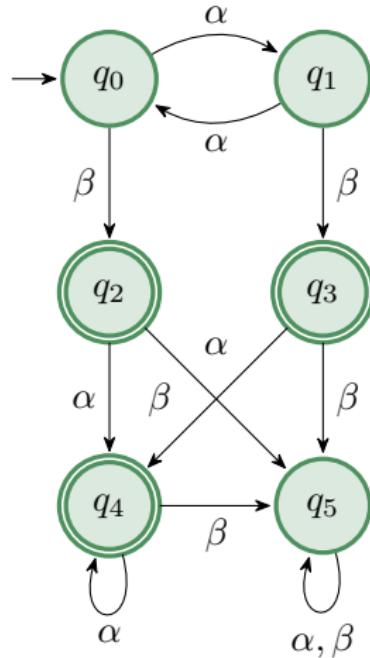
**if**  $(b, B) \in \mathcal{W}$

$\mathcal{W} :=$

$(\mathcal{W} \setminus \{(b, B)\}) \cup \{(b, B_0), (b, B_1)\}$

**else**

$\mathcal{W} := \mathcal{W} \cup \{(b, \min\{B_0, B_1\})\}$



**Data:** a DFA  $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$

if  $\mathcal{F} = \emptyset \vee \mathcal{Q} \setminus \mathcal{F} = \emptyset$   
return  $\mathcal{Q}$

else

$\mathcal{P} := \{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}$

$\mathcal{W} := \{(a, \min\{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}), a \in \Sigma\}$

while  $\mathcal{W} \neq \emptyset$  do

Pick  $(a, C)$  from  $\mathcal{W}$  and remove it

forall  $B \in \mathcal{P}$  do

Split  $B$  with  $(a, C)$  into  $B_0$  and  $B_1$

$\mathcal{P} := (\mathcal{P} \setminus \{B\}) \cup \{B_0, B_1\}$

forall  $b \in \Sigma$  do

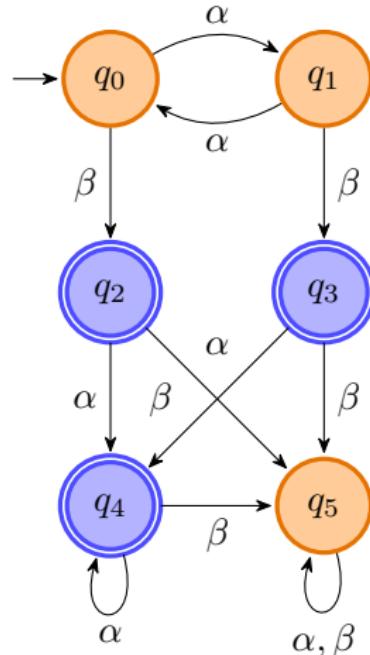
if  $(b, B) \in \mathcal{W}$

$\mathcal{W} :=$

$(\mathcal{W} \setminus \{(b, B)\}) \cup \{(b, B_0), (b, B_1)\}$

else

$\mathcal{W} := \mathcal{W} \cup \{(b, \min\{B_0, B_1\})\}$



Splitter	Partition	Workset
-	$\{q_0, q_1, q_5\} \{q_2, q_3, q_4\}$	$(\alpha, \{q_0, q_1, q_5\})$ $(\beta, \{q_0, q_1, q_5\})$

**Data:** a DFA  $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$

if  $\mathcal{F} = \emptyset \vee \mathcal{Q} \setminus \mathcal{F} = \emptyset$

    return  $\mathcal{Q}$

else

$\mathcal{P} := \{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}$

$\mathcal{W} := \{(a, \min\{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}), a \in \Sigma\}$

**while**  $\mathcal{W} \neq \emptyset$  **do**

        Pick  $(a, C)$  from  $\mathcal{W}$  and remove it

**forall**  $B \in \mathcal{P}$  **do**

            Split  $B$  with  $(a, C)$  into  $B_0$  and  $B_1$

$\mathcal{P} := (\mathcal{P} \setminus \{B\}) \cup \{B_0, B_1\}$

**forall**  $b \in \Sigma$  **do**

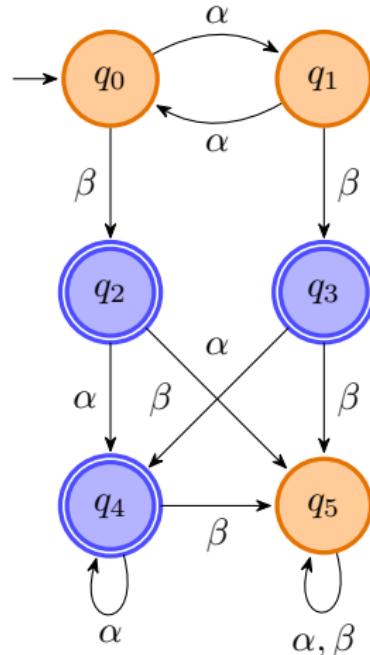
                if  $(b, B) \in \mathcal{W}$

$\mathcal{W} :=$

$(\mathcal{W} \setminus \{(b, B)\}) \cup \{(b, B_0), (b, B_1)\}$

**else**

$\mathcal{W} := \mathcal{W} \cup \{(b, \min\{B_0, B_1\})\}$



Splitter	Partition	Workset
-	$\{q_0, q_1, q_5\} \{q_2, q_3, q_4\}$	$(\alpha, \{q_0, q_1, q_5\})$ $(\beta, \{q_0, q_1, q_5\})$

**Data:** a DFA  $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$

if  $\mathcal{F} = \emptyset \vee \mathcal{Q} \setminus \mathcal{F} = \emptyset$

    return  $\mathcal{Q}$

else

$\mathcal{P} := \{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}$

$\mathcal{W} := \{(a, \min\{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}), a \in \Sigma\}$

**while**  $\mathcal{W} \neq \emptyset$  **do**

        Pick  $(a, C)$  from  $\mathcal{W}$  and remove it

**forall**  $B \in \mathcal{P}$  **do**

            Split  $B$  with  $(a, C)$  into  $B_0$  and  $B_1$

$\mathcal{P} := (\mathcal{P} \setminus \{B\}) \cup \{B_0, B_1\}$

**forall**  $b \in \Sigma$  **do**

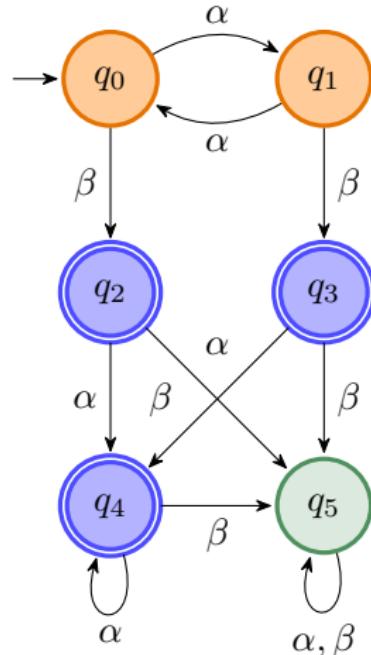
                if  $(b, B) \in \mathcal{W}$

$\mathcal{W} :=$

$(\mathcal{W} \setminus \{(b, B)\}) \cup \{(b, B_0), (b, B_1)\}$

**else**

$\mathcal{W} := \mathcal{W} \cup \{(b, \min\{B_0, B_1\})\}$



Splitter	Partition	Workset
$-$ $(\beta, \{q_0, q_1, q_5\})$	$\{q_0, q_1, q_5\} \{q_2, q_3, q_4\}$ $\{q_0, q_1\} \{q_5\} \{q_2, q_3, q_4\}$	$(\alpha, \{q_0, q_1, q_5\})$ $(\beta, \{q_0, q_1, q_5\})$ $(\alpha, \{q_0, q_1\})$ $(\alpha, \{q_5\})$

**Data:** a DFA  $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$

if  $\mathcal{F} = \emptyset \vee \mathcal{Q} \setminus \mathcal{F} = \emptyset$

    return  $\mathcal{Q}$

else

$\mathcal{P} := \{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}$

$\mathcal{W} := \{(a, \min\{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}), a \in \Sigma\}$

**while**  $\mathcal{W} \neq \emptyset$  **do**

        Pick  $(a, C)$  from  $\mathcal{W}$  and remove it

**forall**  $B \in \mathcal{P}$  **do**

            Split  $B$  with  $(a, C)$  into  $B_0$  and  $B_1$

$\mathcal{P} := (\mathcal{P} \setminus \{B\}) \cup \{B_0, B_1\}$

**forall**  $b \in \Sigma$  **do**

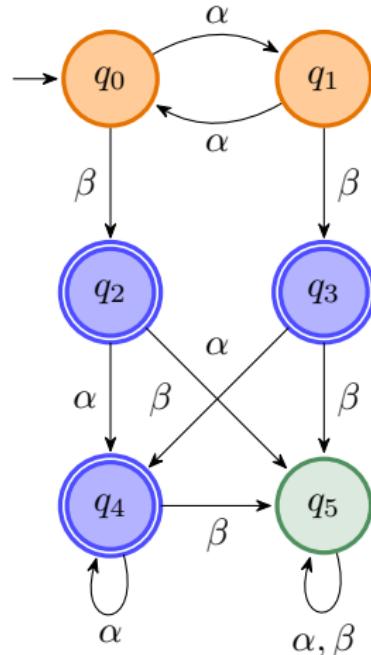
                if  $(b, B) \in \mathcal{W}$

$\mathcal{W} :=$

$(\mathcal{W} \setminus \{(b, B)\}) \cup \{(b, B_0), (b, B_1)\}$

                else

$\mathcal{W} := \mathcal{W} \cup \{(b, \min\{B_0, B_1\})\}$



Splitter	Partition	Workset
—	$\{q_0, q_1, q_5\} \{q_2, q_3, q_4\}$	$(\alpha, \{q_0, q_1, q_5\})$ $(\beta, \{q_0, q_1, q_5\})$
$(\beta, \{q_0, q_1, q_5\})$	$\{q_0, q_1\} \{q_5\} \{q_2, q_3, q_4\}$	$(\alpha, \{q_0, q_1\})$ $(\alpha, \{q_5\})$
$(\alpha, \{q_0, q_1\})$	$\{q_0, q_1\} \ {q_5} \{q_2, q_3, q_4\}$	$(\alpha, \{q_5\})$

**Data:** a DFA  $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$

if  $\mathcal{F} = \emptyset \vee \mathcal{Q} \setminus \mathcal{F} = \emptyset$

    return  $\mathcal{Q}$

else

$\mathcal{P} := \{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}$

$\mathcal{W} := \{(a, \min\{\mathcal{F}, \mathcal{Q} \setminus \mathcal{F}\}), a \in \Sigma\}$

**while**  $\mathcal{W} \neq \emptyset$  **do**

        Pick  $(a, C)$  from  $\mathcal{W}$  and remove it

**forall**  $B \in \mathcal{P}$  **do**

            Split  $B$  with  $(a, C)$  into  $B_0$  and  $B_1$

$\mathcal{P} := (\mathcal{P} \setminus \{B\}) \cup \{B_0, B_1\}$

**forall**  $b \in \Sigma$  **do**

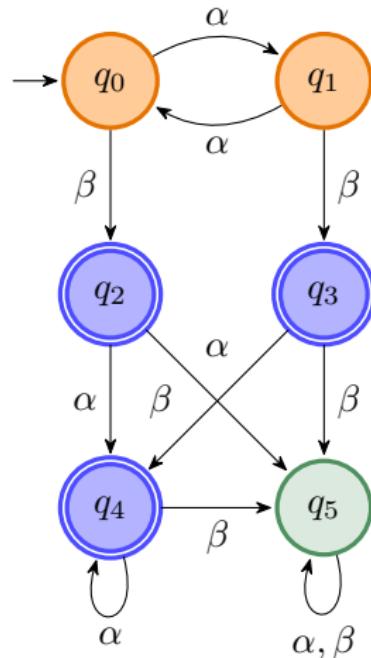
                if  $(b, B) \in \mathcal{W}$

$\mathcal{W} :=$

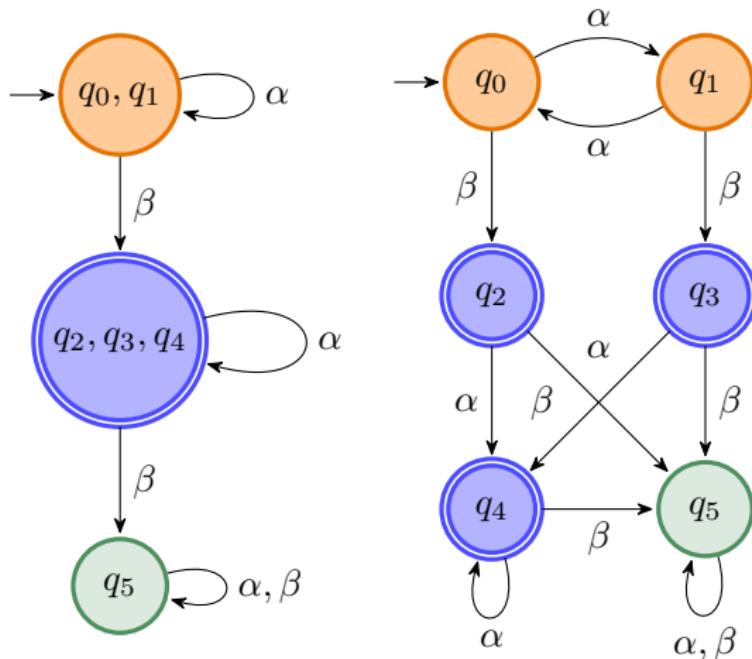
$(\mathcal{W} \setminus \{(b, B)\}) \cup \{(b, B_0), (b, B_1)\}$

                else

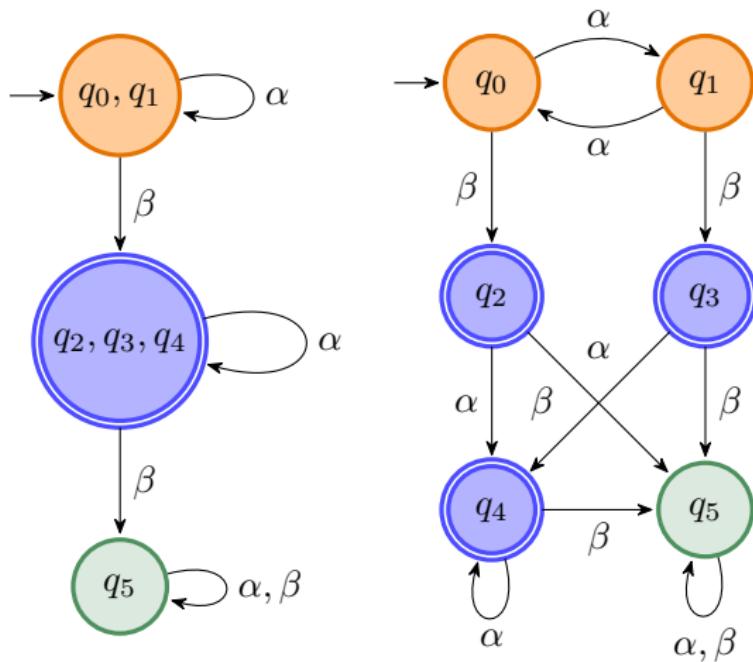
$\mathcal{W} := \mathcal{W} \cup \{(b, \min\{B_0, B_1\})\}$



Splitter	Partition	Workset
-	$\{q_0, q_1, q_5\} \{q_2, q_3, q_4\}$	$(\alpha, \{q_0, q_1, q_5\})$ $(\beta, \{q_0, q_1, q_5\})$
$(\beta, \{q_0, q_1, q_5\})$	$\{q_0, q_1\} \{q_5\} \{q_2, q_3, q_4\}$	$(\alpha, \{q_0, q_1\})$ $(\alpha, \{q_5\})$
$(\alpha, \{q_0, q_1\})$	$\{q_0, q_1\} \{q_5\} \{q_2, q_3, q_4\}$	$(\alpha, \{q_5\})$
$(\alpha, \{q_5\})$	$\{q_0, q_1\} \{q_5\} \{q_2, q_3, q_4\}$	$\emptyset$

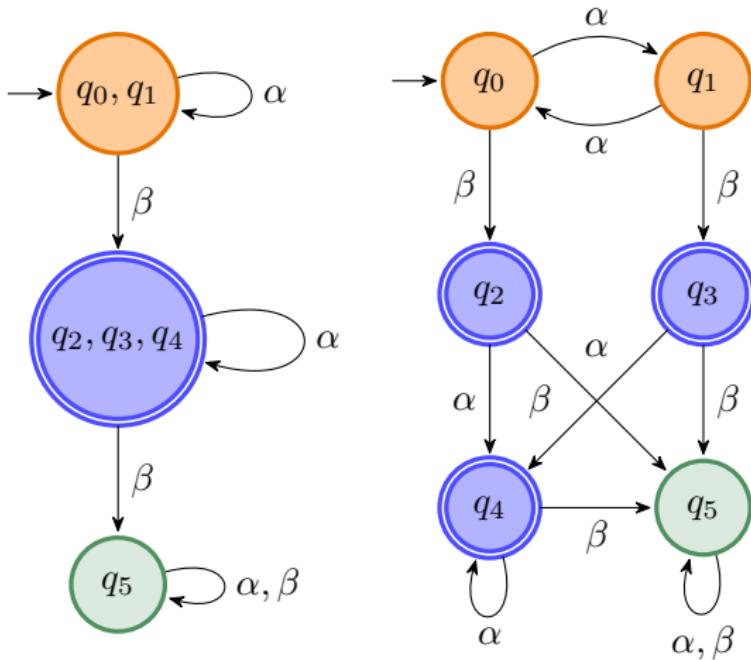


$$\mathcal{L} = \alpha^* \beta \alpha^*$$



$$\mathcal{L} = \alpha^* \beta \alpha^*$$

$$O(|\mathcal{Q}| |\Sigma| \log(|\mathcal{Q}|))$$



$$\mathcal{L} = \alpha^* \beta \alpha^*$$

$O(n \log(n))$  where  $n = |\mathcal{Q}|$

## 1. Living in Munich

1.1 The city

1.2 Technical University of Munich

## 2. The Isabelle Refinement Framework

## 3. Hopcroft's algorithm

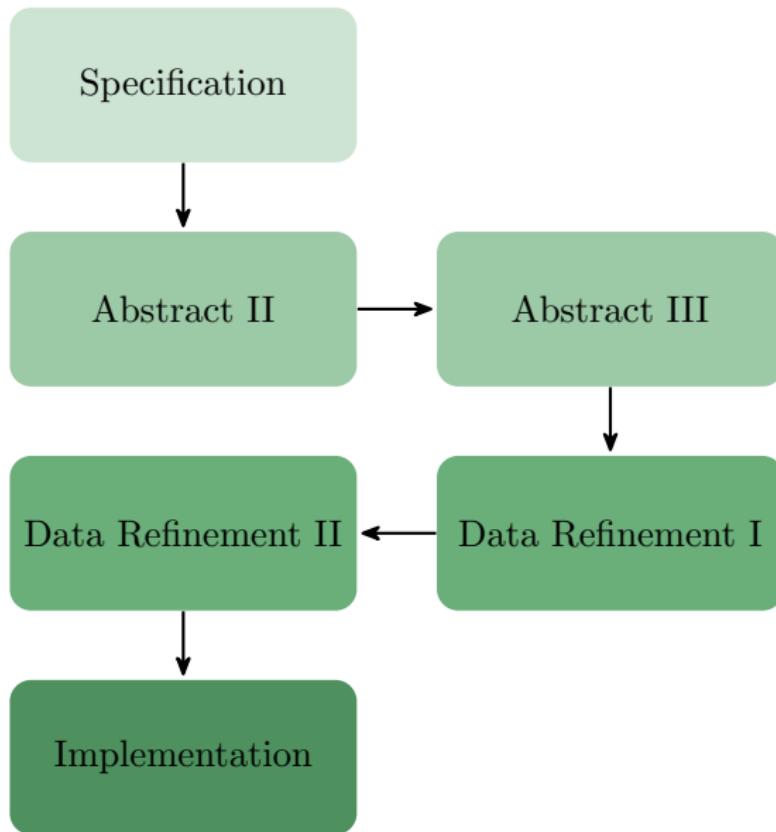
3.1 DFA minimization by example

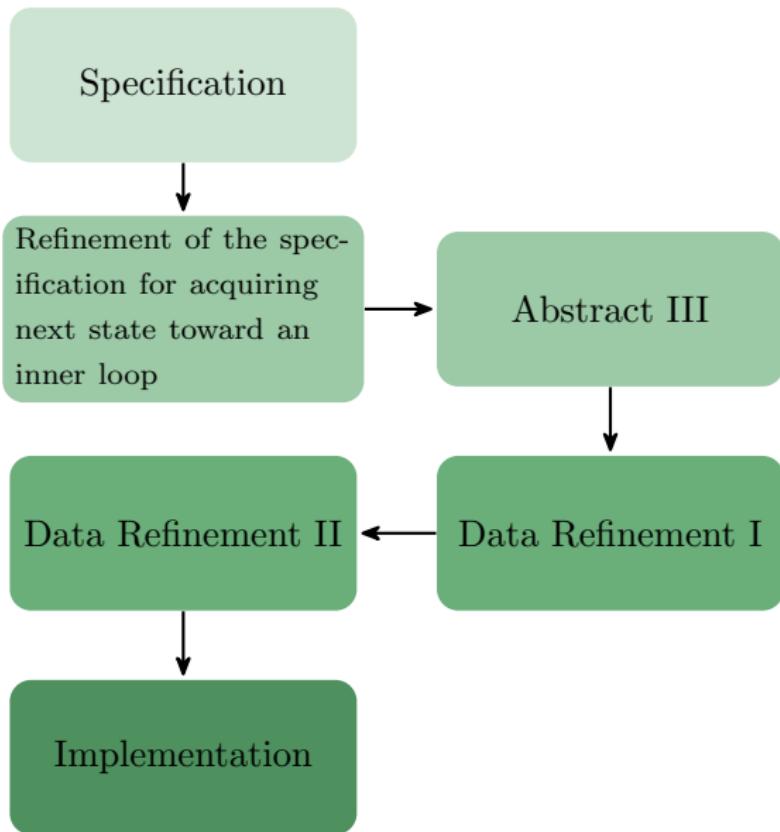
3.2 Refinement steps

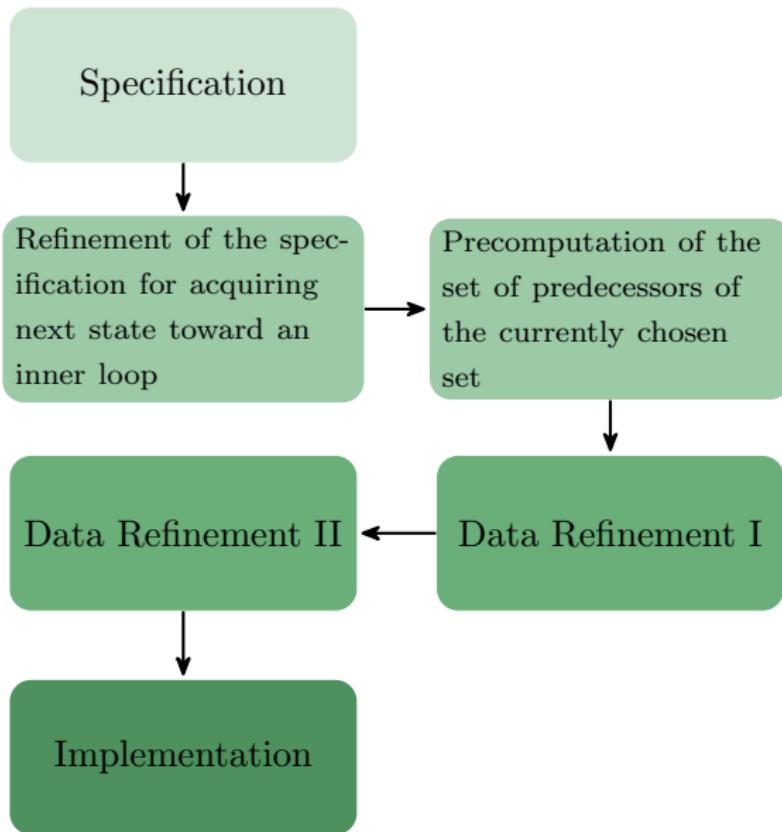
## 4. Conduction of the project

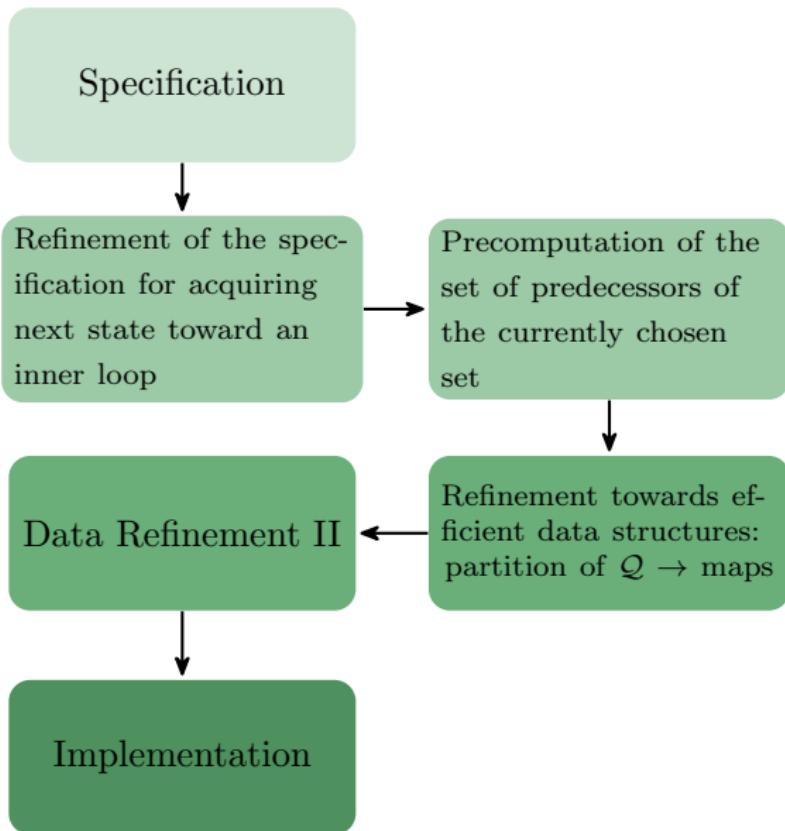
4.1 Gantt chart

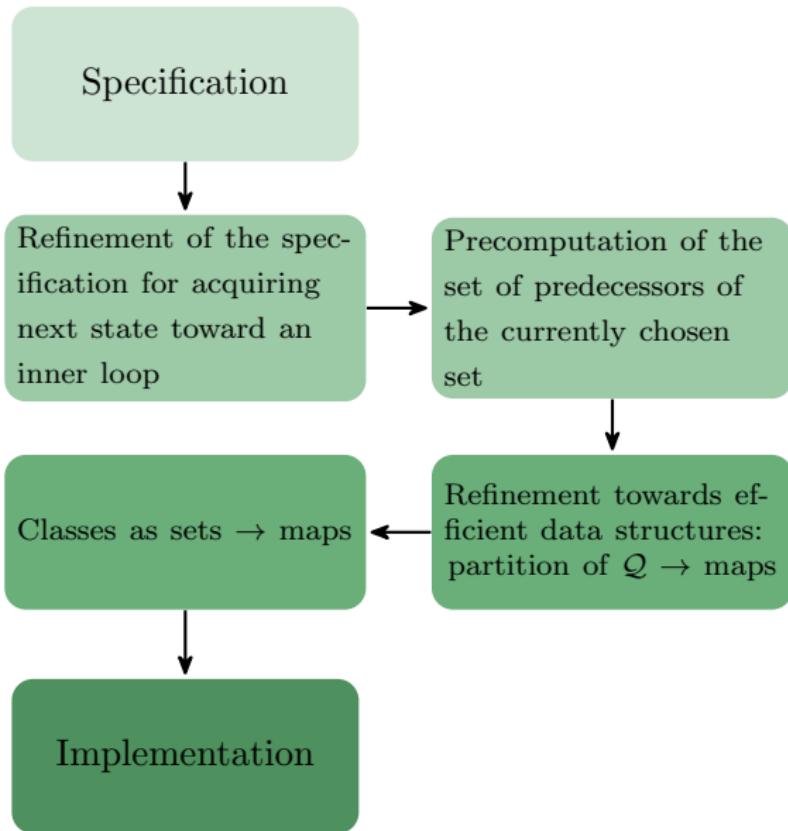
4.2 Statistics

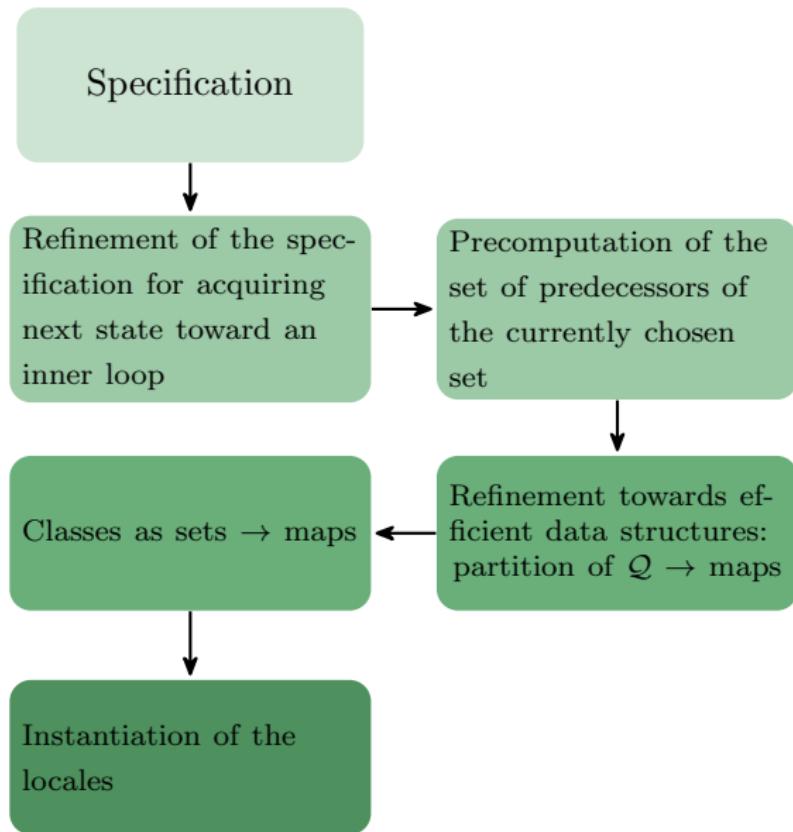


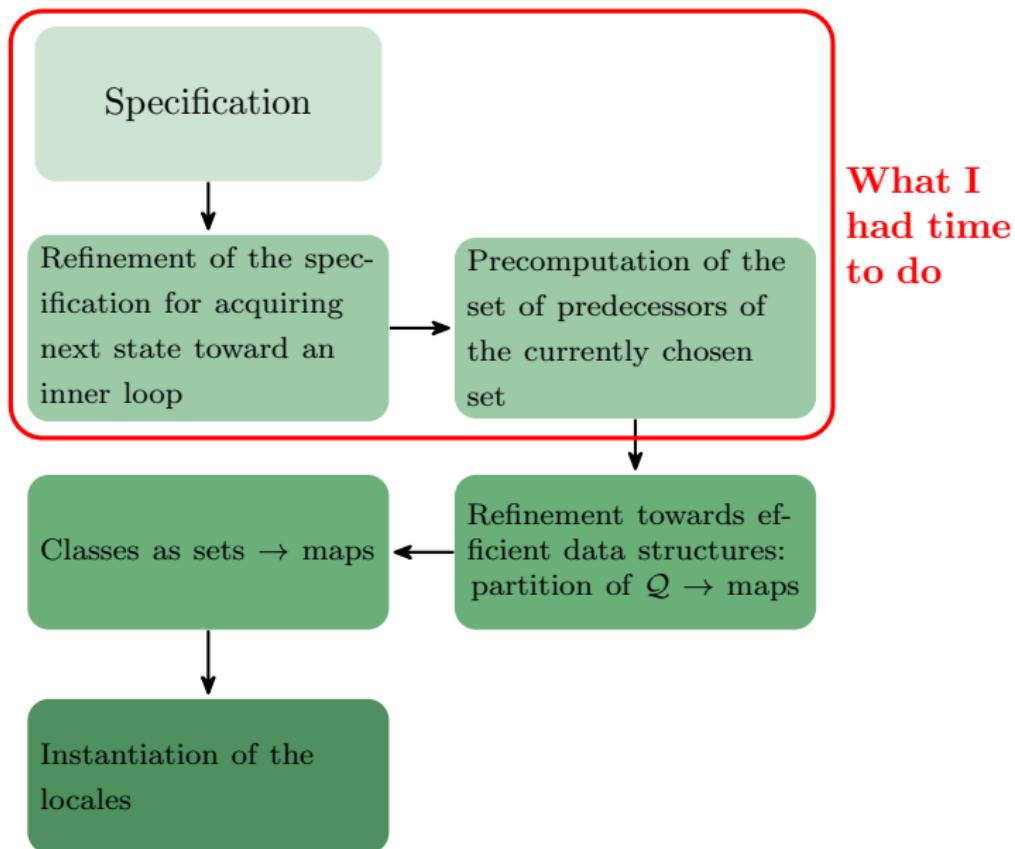








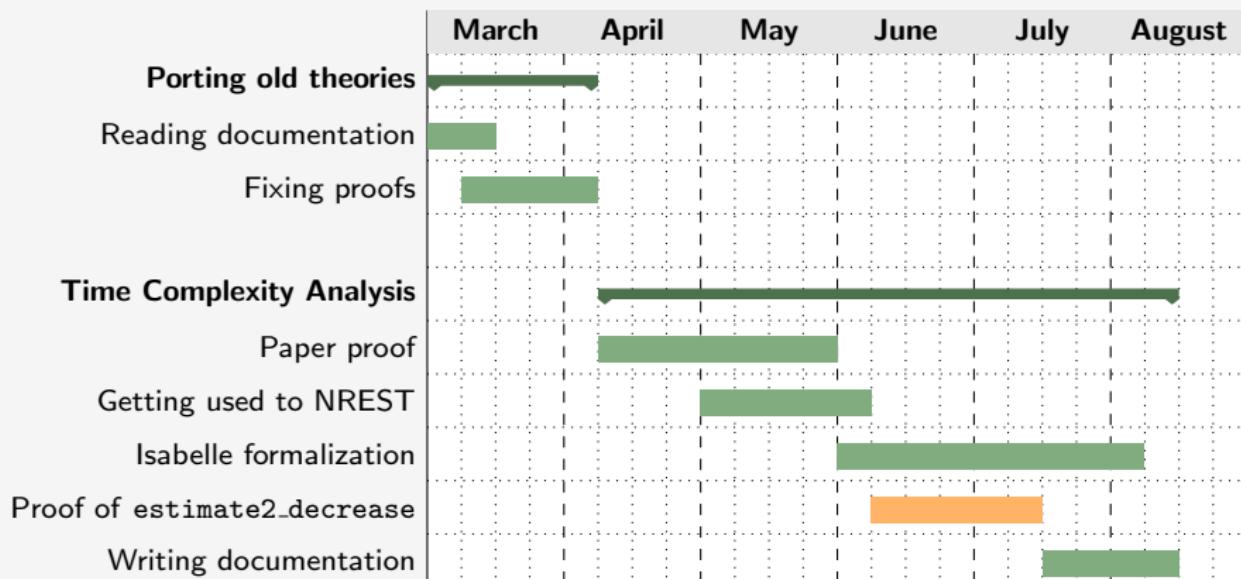




1. Living in Munich
2. The Isabelle Refinement Framework
3. Hopcroft's algorithm
4. Conduction of the project
  - 4.1 Gantt chart
  - 4.2 Statistics



```
vtrelat@home:~$ echo gantt chart
```





```
vtrelat@home:~$ show stats
```

Beginning of the project (in SLoC):

Current state (in SLoC) after approx. 900h:



```
vtrelat@home:~$ show stats
```

Beginning of the project (in SLoC):

```
$ find . -type f -name '*' | xargs wc -l | grep 'total'  
477020 total
```

Current state (in SLoC) after approx. 900h:



```
vtrelat@home:~$ show stats
```

Beginning of the project (in SLoC):

```
$ find . -type f -name '*' | xargs wc -l | grep 'total'  
477020 total
```

```
$ find . -type f -name '*.thy' | xargs wc -l | grep 'total'  
139052 total
```

Current state (in SLoC) after approx. 900h:



```
vtrelat@home:~$ show stats
```

Beginning of the project (in SLoC):

```
$ find . -type f -name '*' | xargs wc -l | grep 'total'  
477020 total  
$ find . -type f -name '*.thy' | xargs wc -l | grep 'total'  
139052 total
```

Current state (in SLoC) after approx. 900h:

```
$ find . -type f -name '*' | xargs wc -l | grep 'total'  
691703 total
```



```
vtrelat@home:~$ show stats
```

Beginning of the project (in SLoC):

```
$ find . -type f -name '*' | xargs wc -l | grep 'total'  
477020 total  
$ find . -type f -name '*.thy' | xargs wc -l | grep 'total'  
139052 total
```

Current state (in SLoC) after approx. 900h:

```
$ find . -type f -name '*' | xargs wc -l | grep 'total'  
691703 total  
$ find . -type f -name '*.thy' | xargs wc -l | grep 'total'  
273752 total
```



```
vtrelat@home:~$ show stats
```

Beginning of the project (in SLoC):

```
$ find . -type f -name '*' | xargs wc -l | grep 'total'  
477020 total  
$ find . -type f -name '*.thy' | xargs wc -l | grep 'total'  
139052 total
```

Current state (in SLoC) after approx. 900h:

```
$ find . -type f -name '*' | xargs wc -l | grep 'total'  
691703 total  
$ find . -type f -name '*.thy' | xargs wc -l | grep 'total'  
273752 total
```

