Mechanical Verification of a Constructive Proof for FLP

according to Hagen Völzer

Bisping Brodmann Jungnickel
Rickmann **Seidler** Stüber Wilhelm-Weidner
Peters Nestmann

24 August 2016

Models and Theory of Distributed Systems

Consensus – Motivation

Example

- distributed database
- each at different state
- decide whether to apply transaction

Consensus – Motivation

Example

- distributed database
- each at different state
- decide whether to apply transaction

- exchange messages
- all have to arrive at same decision

Consensus – Motivation

Example

- distributed database
- each at different state
- decide whether to apply transaction

- exchange messages
- all have to arrive at same decision

Problem

processes may crash

The FLP Theorem

Theorem (Fischer, Lynch, Paterson, 1985)

impossible to ensure consensus, if processes may crash

The FLP Theorem

Theorem (Fischer, Lynch, Paterson, 1985)

impossible to ensure consensus, if processes may crash

Theorem (Völzer, 2004)

more constructive proof of FLP

The FLP Theorem

Theorem (Fischer, Lynch, Paterson, 1985)

impossible to ensure consensus, if processes may crash

Theorem (Völzer, 2004)

more constructive proof of FLP

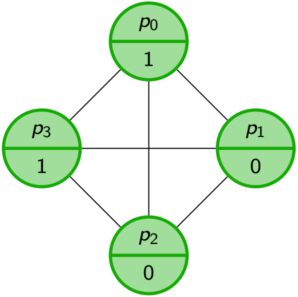
Our Work

- based on the more constructive paper of Völzer
- formalizing this proof in Isabelle/HOL
- ... including "fairness", which was just stated

Consensus

Model

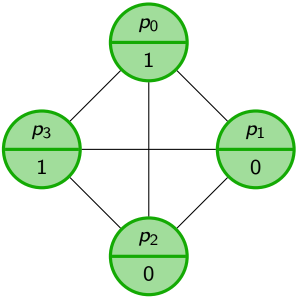
- finite set of sequential processes
- *asynchronous* communication channels between all pairs



Consensus

Model

- finite set of sequential processes
- *asynchronous* communication channels between all pairs



Definition: Binary Consensus

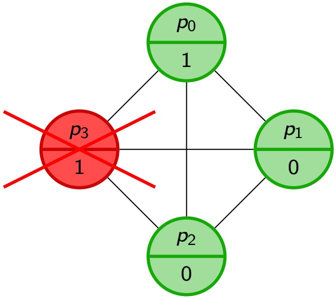
Each process gets an input value from $\{0, 1\}$ and may irrevocably decide on a final output value such that:

- *Agreement*: No two processes decide differently.
- *Validity*: The output value is the input value of some process.
- *Termination*: Each process eventually decides or crashes.

Consensus

Model

- finite set of sequential processes
- *asynchronous* communication channels between all pairs



Definition: Binary Consensus

Each process gets an input value from $\{0, 1\}$ and may irrevocably decide on a final output value such that:

- *Agreement*: No two processes decide differently.
- *Validity*: The output value is the input value of some process.
- *Termination*: Each process eventually decides or crashes.

Fairness

- easy to obtain undesired behaviour
- "block" process by not processing its messages

Fairness

- easy to obtain undesired behaviour
- "block" process by not processing its messages

Definition: Fair Execution

Each message is processed (as long as receiver not crashed).

Fairness

- easy to obtain undesired behaviour
- "block" process by not processing its messages

Definition: Fair Execution

Each message is processed (as long as receiver not crashed).

- unfair execution practically irrelevant

The FLP Theorem

Theorem (Völzer, 2004)

There is no consensus algorithm such that

- *a process may crash*
- *validity*
- *agreement*
- *every fair execution terminates*

The FLP Theorem

Theorem (Völzer, 2004)

There is no consensus algorithm such that

- *a process may crash*
- *validity*
- *agreement*
- *every fair execution terminates*

fundamental result in distributed computing

The FLP Theorem

Theorem (Völzer, 2004)

Every consensus algorithm such that

- *a process may crash*
- *validity*
- *agreement*

has an infinite fair execution that does not decide.

The FLP Theorem

Theorem (Völzer, 2004)

Every consensus algorithm such that

- *a process may crash*
- *validity*
- *agreement*

has an infinite fair execution that does not decide.

$\rightsquigarrow$ constructive

The FLP Theorem

Theorem (Völzer, 2004)

Every consensus algorithm such that

- *a process may crash*
- *validity*
- *agreement*

has an infinite fair execution that does not decide.

$\rightsquigarrow$ constructive

Idea of proof

- find invariant that ensures non-decided
- find proper way to extend finite execution, keeping the invariant
- infinite fair run

Initial Lemma

Non-uniform

There are processes p, q such that

- crash of p allows decision 0
- crash of q allows decision 1

Initial Lemma

Non-uniform

There are processes p, q such that

- crash of p allows decision 0
- crash of q allows decision 1

Initial Lemma

There is a non-uniform initial configuration.

Initial Lemma

Non-uniform

There are processes p, q such that

- crash of p allows decision 0
- crash of q allows decision 1

Initial Lemma

There is a non-uniform initial configuration.

Small error in Völzer's proof

- used same symbol for different configurations
- required adaption in proof

Extension Lemma

Extension Lemma - Völzer's version

For each non-uniform configuration c and each process p there is a configuration c' such that $c \Rightarrow^* c'$ and crash of p in c' allows for both decisions.

Extension Lemma

Extension Lemma - Völzer's version

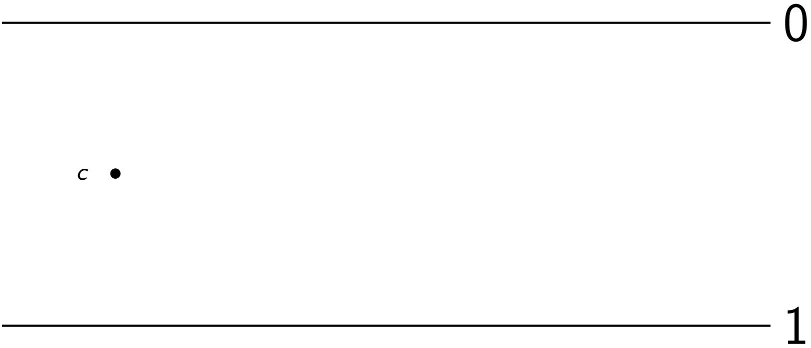
For each non-uniform configuration c and each process p there is a configuration c' such that $c \Rightarrow^* c'$ and crash of p in c' allows for both decisions.

Extension Lemma – our version

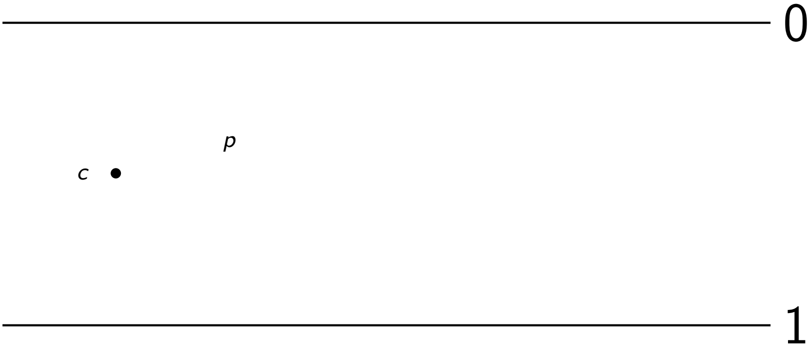
- choose message (p, m) – receiver p, content m
- apply Extension Lemma for this p
- can safely consume message (keeping invariant)

all put into single extension

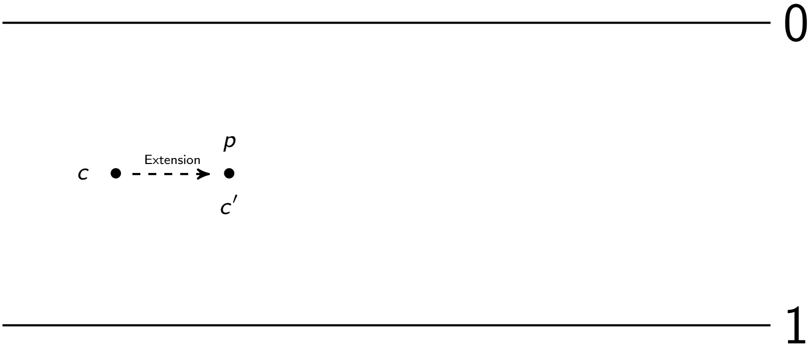
Extension – Picture



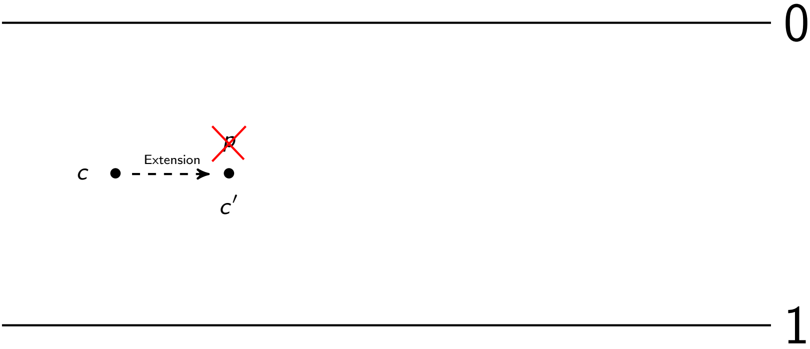
Extension – Picture



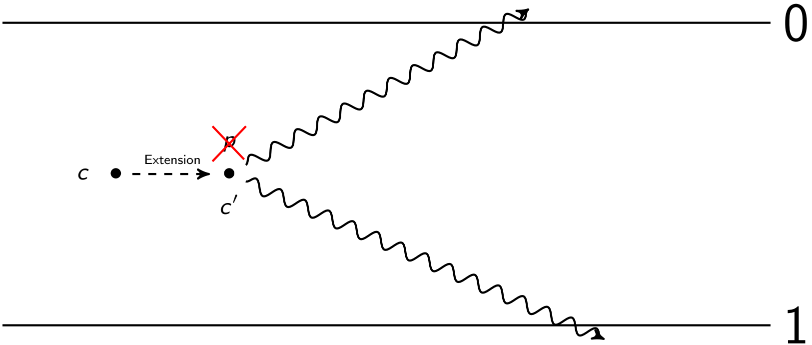
Extension – Picture



Extension – Picture



Extension – Picture



FLP-Theorem

FLP-Theorem

Each possible consensus algorithm has a fair infinite execution that does not decide.

FLP-Theorem

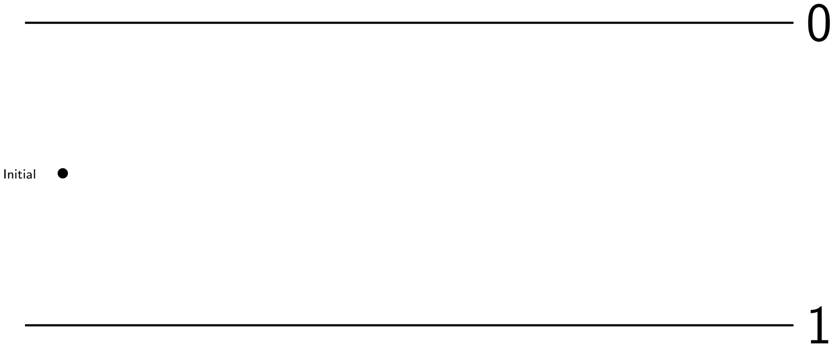
FLP-Theorem

Each possible consensus algorithm has a fair infinite execution that does not decide.

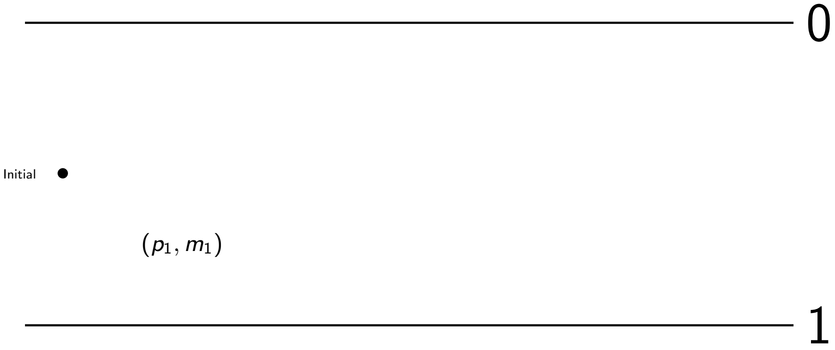
Proof by Völzer

- start with non-uniform initial configuration
- take message with minimal enabling time
- extend execution using Extension Lemma, ending with non-uniform configuration
- repeat this process

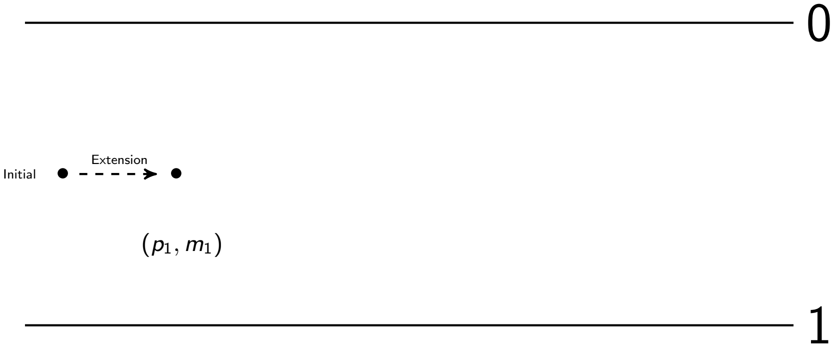
Proof Idea



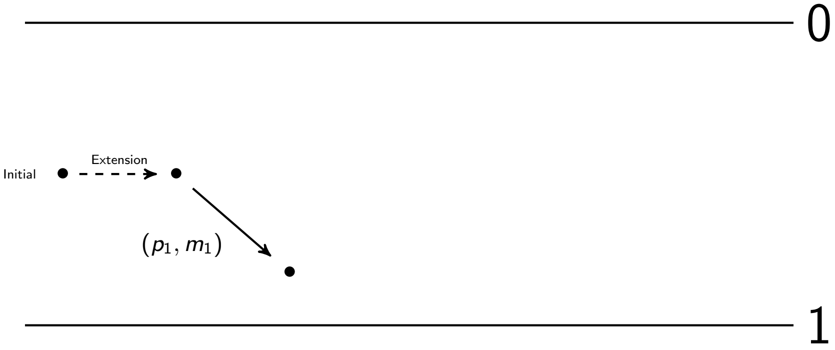
Proof Idea



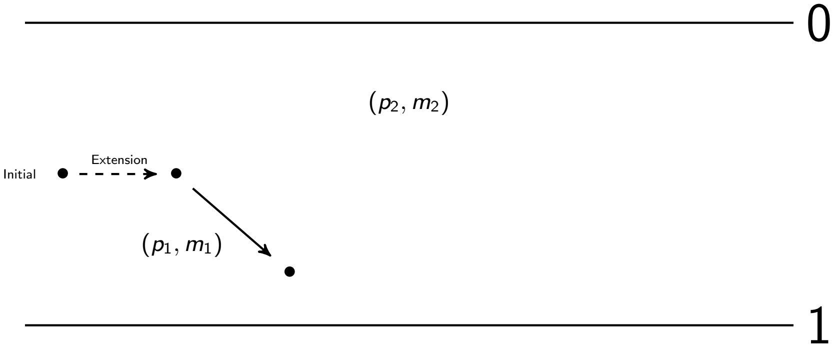
Proof Idea



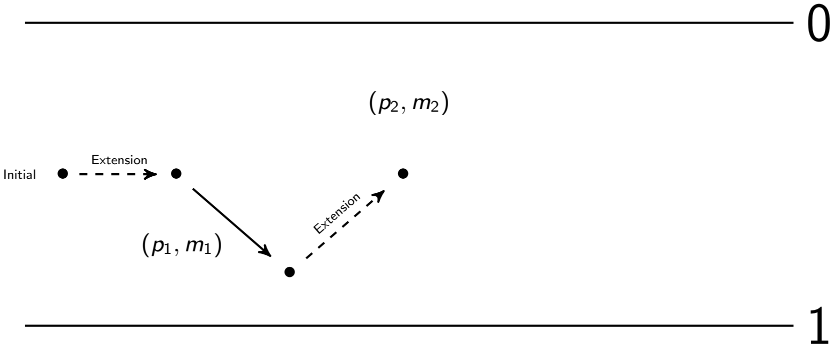
Proof Idea



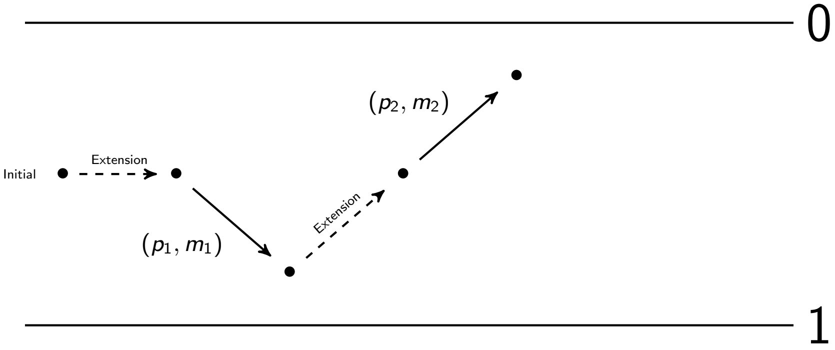
Proof Idea



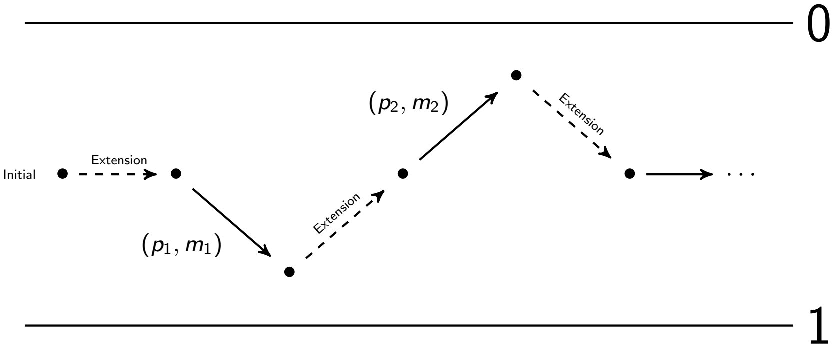
Proof Idea



Proof Idea



Proof Idea



Infinite Executions

Problem

- fairness/correctness defined for single (infinite) execution
- construction yields sequence of finite executions

Infinite Executions

Problem

- fairness/correctness defined for single (infinite) execution
- construction yields sequence of finite executions

Infinite executions – our model

- as function from natural numbers to finite executions

Infinite Executions

Problem

- fairness/correctness defined for single (infinite) execution
- construction yields sequence of finite executions

Infinite executions – our model

- as function from natural numbers to finite executions

```
definition infiniteExecution ::
  "(nat ⇒ (('p, 'v, 's) configuration list))
  ⇒ (nat ⇒ (('p, 'v) message list)) ⇒ bool"
where
  "infiniteExecution fe ft ≡
    ∀ n . execution trans sends start (fe n) (ft n) ∧
      prefixList (fe n) (fe (n+1)) ∧
      prefixList (ft n) (ft (n+1))"
```

Proof of Fairness

Völzer: "We obtain a fair execution
where all processes are correct
and that is always eventually
non-uniform and hence does not decide. □"

Proof of Fairness

```
assumes
  Cfg: "initial cfg" "nonUniform cfg"
shows "∃ fe ft.
  (fe 0) = [cfg]
  ∧ fairInfiniteExecution fe ft
  ∧ (∀ n . nonUniform (last (fe n))
      ∧ prefixList (fe n) (fe (n+1))
      ∧ prefixList (ft n) (ft (n+1))
      ∧ (execution trans sends start (fe n) (ft n)))"
proof -
  have BC:  [10 lines]
  obtain fStepCfg fStepMsg where FStep: "∀ cfgList msgList . ∃cfgList' msgList' . [27 lines]
  def Fe: fe == "infiniteExecutionCfg cfg fStepCfg fStepMsg" and [1 lines]

  have BasicProperties: "(∀n. nonUniform (last (fe n))  [26 lines]
  have Fair: "fairInfiniteExecution fe ft" [737 lines]
  show ?thesis proof (rule exI[of _ fe], rule exI[of _ ft]) [6 lines]
qed
```

Völzer: "We obtain a fair execution where all processes are correct and that is always eventually non-uniform and hence does not decide. □"

Proof of Fairness

```
assumes
  Cfg: "initial cfg" "nonUniform cfg"
shows "∃ fe ft.
  (fe 0) = [cfg]
  ∧ fairInfiniteExecution fe ft
  ∧ (∀ n . nonUniform (last (fe n))
       ∧ prefixList (fe n) (fe (n+1))
       ∧ prefixList (ft n) (ft (n+1))
       ∧ (execution trans sends start (fe n) (ft n)))"
proof -
  have BC:  [10 lines]
  obtain fStepCfg fStepMsg where FStep: "∀ cfgList msgList . ∃cfgList' msgList' . [27 lines]
  def Fe: fe == "infiniteExecutionCfg cfg fStepCfg fStepMsg" and [1 lines]

  have BasicProperties: "(∀n. nonUniform (last (fe n))  [26 lines]
  have Fair: "fairInfiniteExecution fe ft" [737 lines]
  show ?thesis proof (rule exI[of _ fe], rule exI[of _ ft]) [6 lines]
qed
```

Völzer: "We obtain a fair execution where all processes are correct and that is always eventually non-uniform and hence does not decide. □"

Conclusions

```
theorem ConsensusFails:
assumes
  Termination:
    "⋀ fe ft . (fairInfiniteExecution fe ft ⟹ terminationFLP fe ft)" and
  Validity: "∀ i c . validity i c" and
  Agreement: "∀ i c . agreementInit i c"
shows
  "False"
```

Conclusions

```
theorem ConsensusFails:
assumes
  Termination:
    "⋀ fe ft . (fairInfiniteExecution fe ft ⟹ terminationFLP fe ft)" and
  Validity: "∀ i c . validity i c" and
  Agreement: "∀ i c . agreementInit i c"
shows
  "False"
```

Conclusions

- formalization of Völzer's proof in Isabelle/HOL
- $2\frac{1}{2}$ pages $\rightarrow$ 4000 LOC
- precise list of preconditions for individual proofs
- proof of fairness
- correctness up to correctness of Isabelle/HOL

Conclusions

```
theorem ConsensusFails:
assumes
  Termination:
    "⋀ fe ft . (fairInfiniteExecution fe ft ⟹ terminationFLP fe ft)" and
  Validity: "∀ i c . validity i c" and
  Agreement: "∀ i c . agreementInit i c"
shows
  "False"
```

Conclusions

- formalization of Völzer's proof in Isabelle/HOL
- $2\frac{1}{2}$ pages $\rightarrow$ 4000 LOC
- precise list of preconditions for individual proofs
- proof of fairness
- correctness up to correctness of Isabelle/HOL

Thank you very much for your attention.