FLP

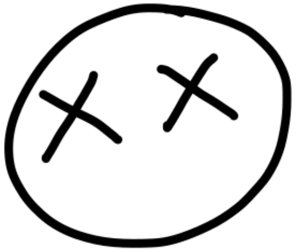
Impossibility of Distributed Consensus with One Faulty Process

ByzaRe, 2017-12-07, mail@maria-a-schett.net

FLP

[1]

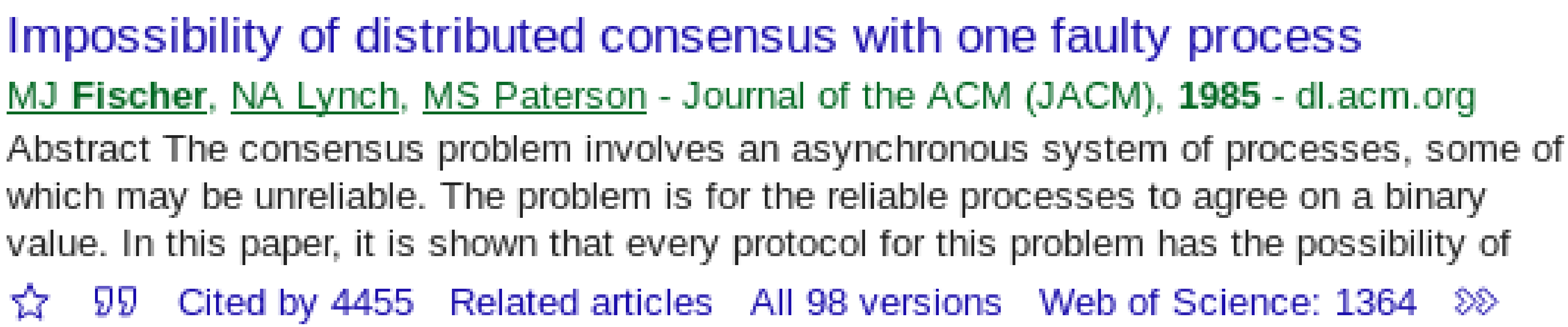
“no **completely asynchronous consensus protocol** can tolerate even a single **unannounced process death**”



[1] M. Fischer, N. Lynch, and M. Paterson. Impossibility of Distributed Consensus With One Faulty Process. Journal of the ACM, 32(2), 1985 https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf

Why?

Paxos, PBFT…?

- to scrutinize how consensus protocols deal with this impossibility
- 

Impossibility of distributed consensus with one faulty process

MJ **Fischer**, NA Lynch, MS Paterson - Journal of the ACM (JACM), **1985** - dl.acm.org

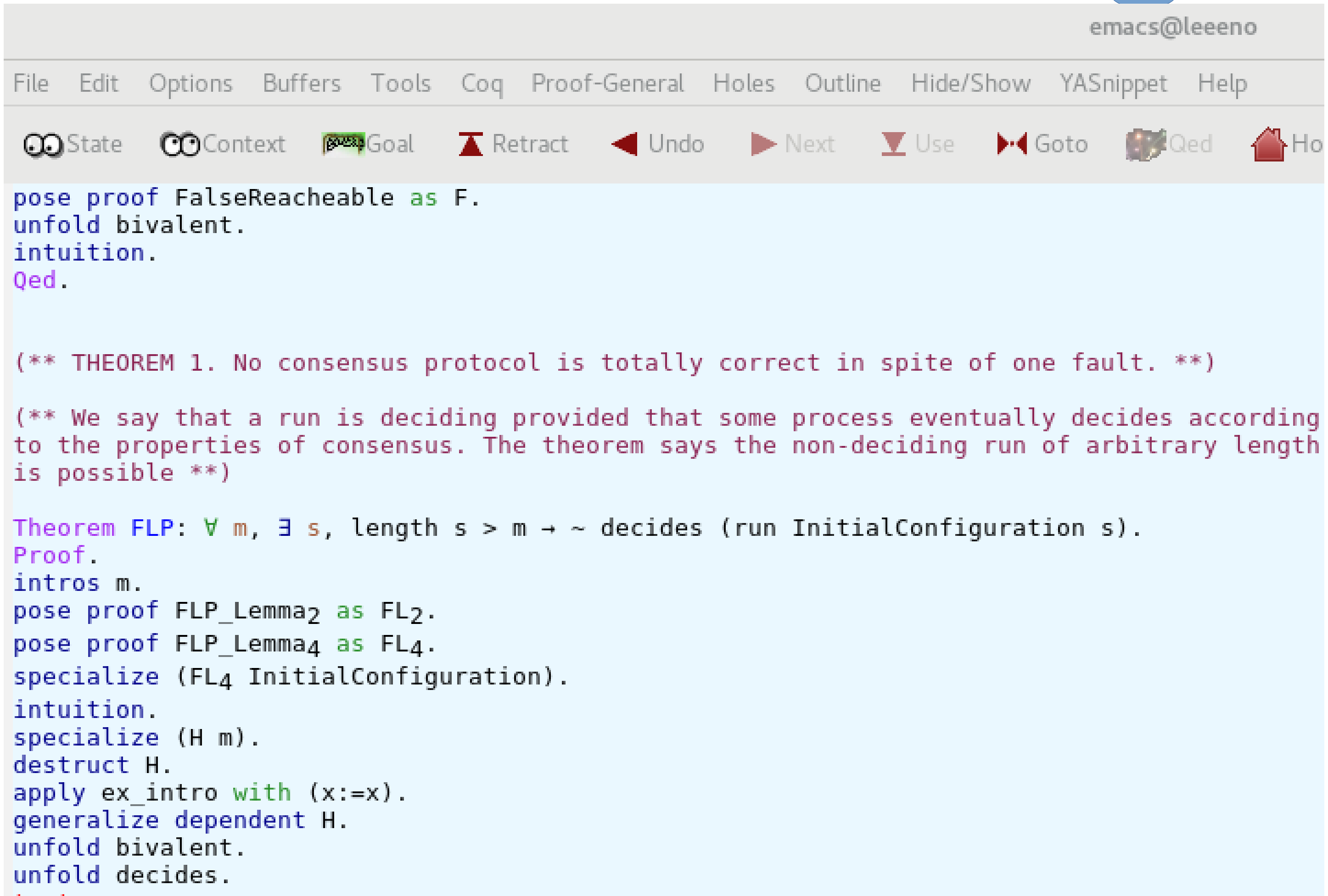
Abstract The consensus problem involves an asynchronous system of processes, some of which may be unreliable. The problem is for the reliable processes to agree on a binary value. In this paper, it is shown that every protocol for this problem has the possibility of

Cited by 4455 Related articles All 98 versions Web of Science: 1364

- curious about how proofs are

Mechanization: COQ

- github.com/kushti/flp/




```
pose proof FalseReacheable as F.
unfold bivalent.
intuition.
Qed.


(** THEOREM 1. No consensus protocol is totally correct in spite of one fault. **)

(** We say that a run is deciding provided that some process eventually decides according
to the properties of consensus. The theorem says the non-deciding run of arbitrary length
is possible **)

Theorem FLP: ∀ m, ∃ s, length s > m → ~ decides (run InitialConfiguration s).
Proof.
intros m.
pose proof FLP_Lemma2 as FL2.
pose proof FLP_Lemma4 as FL4.
specialize (FL4 InitialConfiguration).
intuition.
specialize (H m).
destruct H.
apply ex_intro with (x:=x).
generalize dependent H.
unfold bivalent.
unfold decides.
```

Mechanization: Isabelle

- www.isa-afp.org/entries/FLP.html

A Constructive Proof for FLP

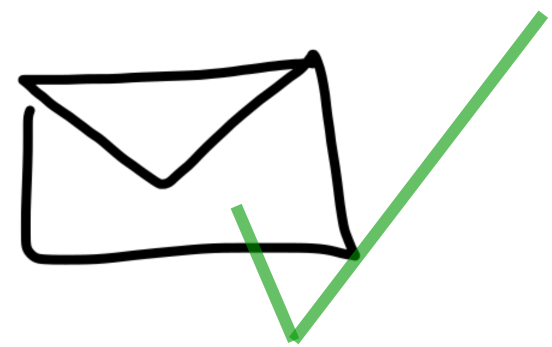
- Home
- About
- Submission
- Updating Entries
- Using Entries
- Search
- Statistics
- Index
- Download

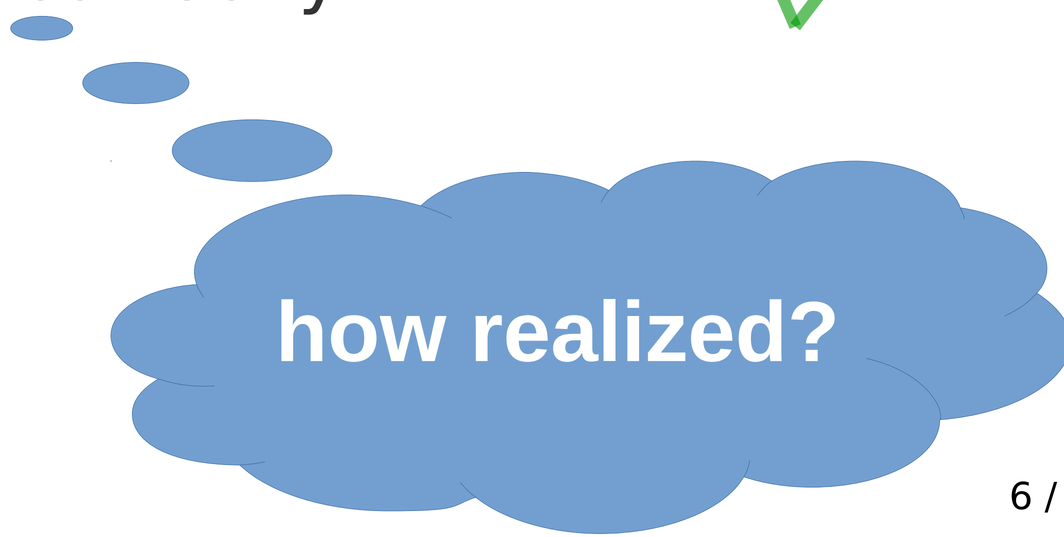
Title:	A Constructive Proof for FLP
Authors:	Benjamin Bisping (benjamin /dot/ bisping /at/ campus /dot/ tu-berlin /dot/ de), Paul-David Brodmann (p /dot/ brodmann /at/ tu-berlin /dot/ de), Tim Jungnickel (tim /dot/ jungnickel /at/ tu-berlin /dot/ de), Christina Rickmann (c /dot/ rickmann /at/ tu-berlin /dot/ de), Henning Seidler (henning /dot/ seidler /at/ mailbox /dot/ tu-berlin /dot/ de), Anke Stüber (anke /dot/ stueber /at/ campus /dot/ tu-berlin /dot/ de), Arno Wilhelm-Weidner (arno /dot/ wilhelm-weidner /at/ tu-berlin /dot/ de), Kirstin Peters (kirstin /dot/ peters /at/ tu-berlin /dot/ de) and Uwe Nestmann
Submission date:	2016-05-18
Abstract:	The impossibility of distributed consensus with one faulty process is a result with important consequences for real world distributed systems e.g., commits in replicated databases. Since proofs are not immune to faults and even plausible proofs with a profound formalism can conclude wrong results, we validate the fundamental result named FLP after Fischer, Lynch and Paterson. We present a formalization of distributed systems and the aforementioned consensus problem. Our proof is based on Hagen Völzer's paper "A constructive proof for FLP". In addition to the enhanced confidence in the validity of Völzer's proof, we contribute the missing gaps to show the correctness in Isabelle/HOL. We clarify the proof details and even prove fairness of the infinite execution that contradicts consensus. Our Isabelle formalization can also be reused for further proofs of properties of distributed systems.

Assumptions (1)

- no Byzantine

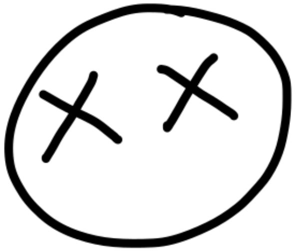
- reliable messages
  - exactly once & correctly





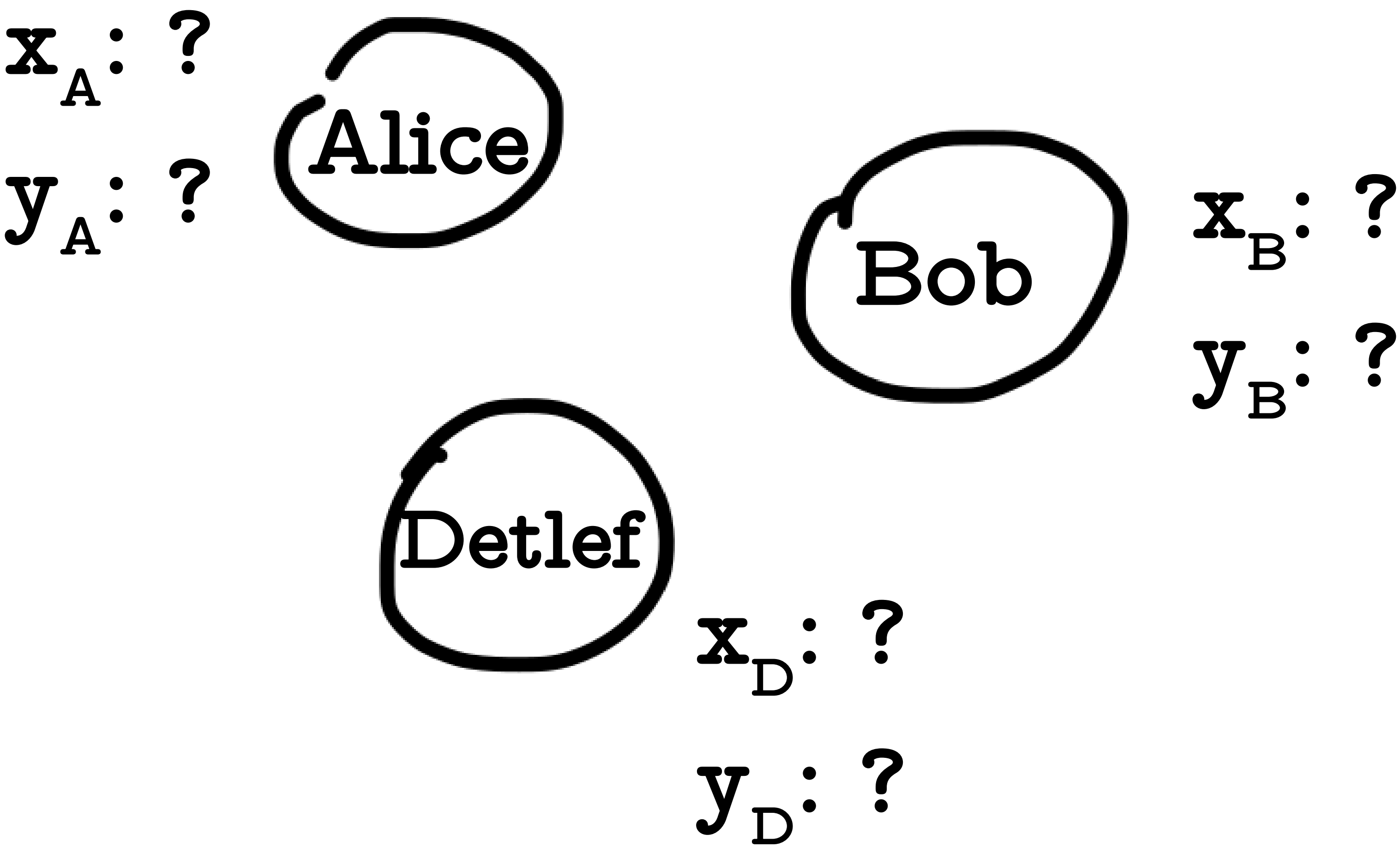
Assumptions (2)

- completely asynchronous
  - arbitrary speed of processes/message delivery
  - no central clock
- impossible to detect dead processes

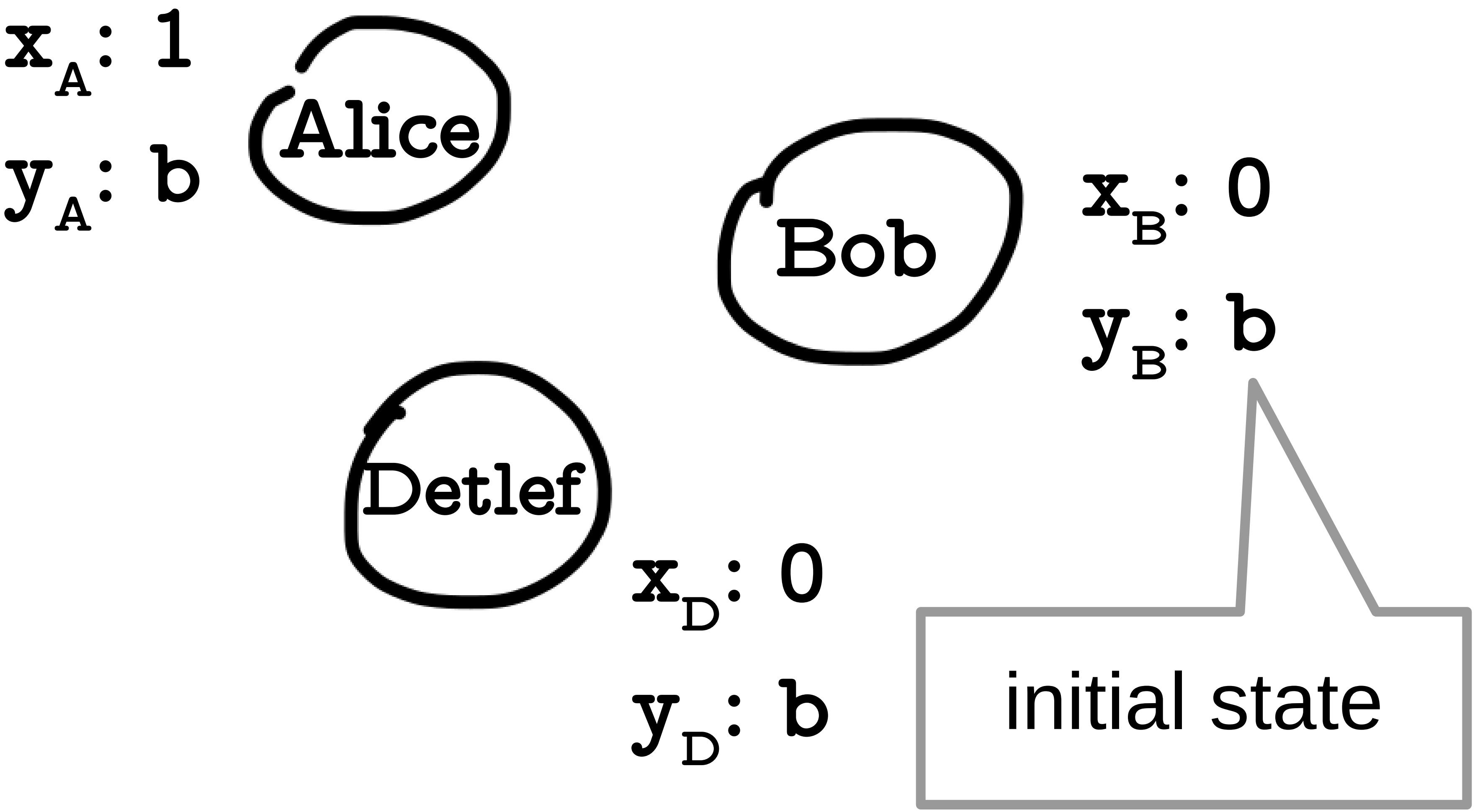


follows from asynch?

Processes >= 2



Register Values = {0, 1, b}



Goal – weak!

is it?

x_A: ?

y_A: 1

Alice

Bob

x_B: ?

y_B: ?

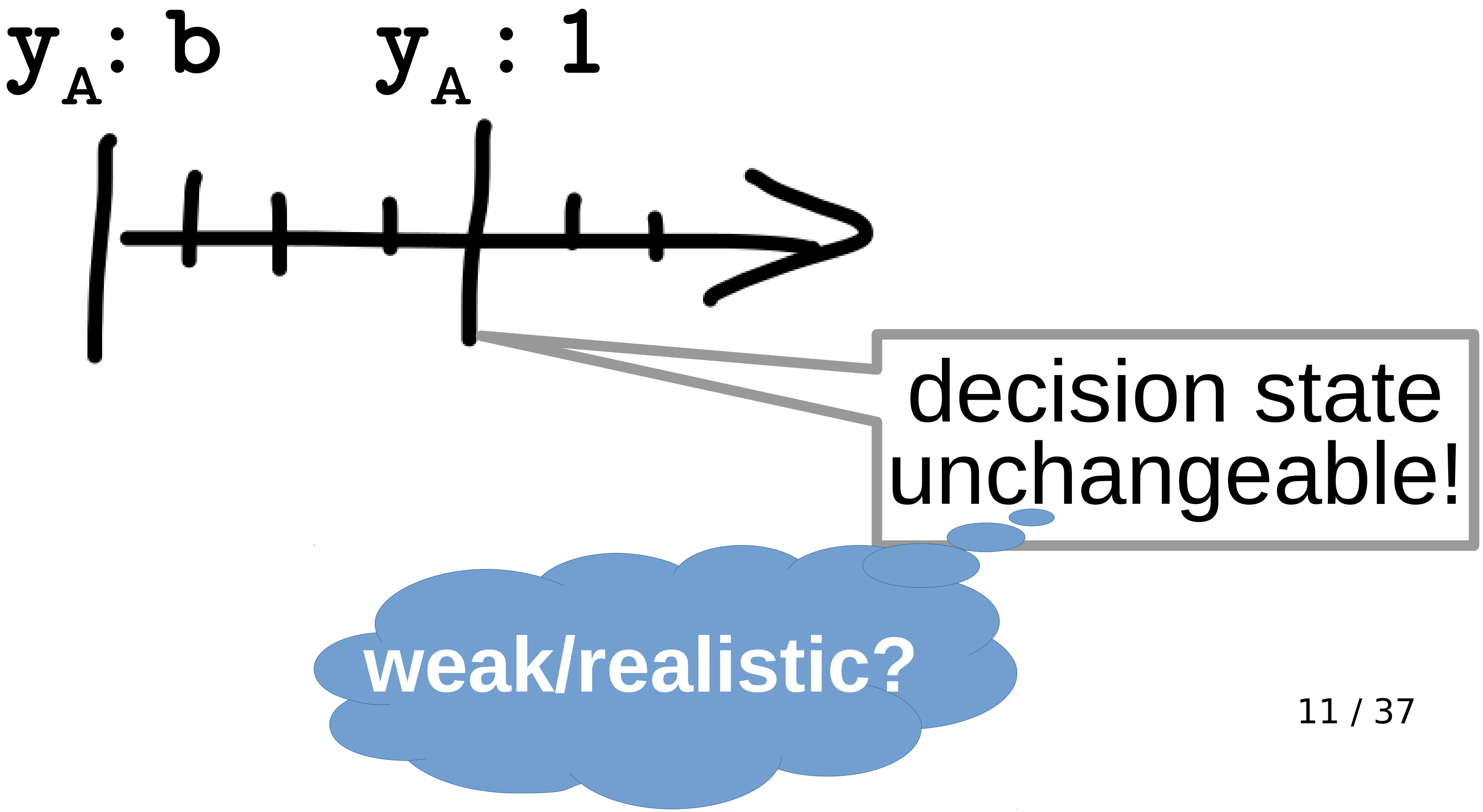
Detlef

x_D: ?

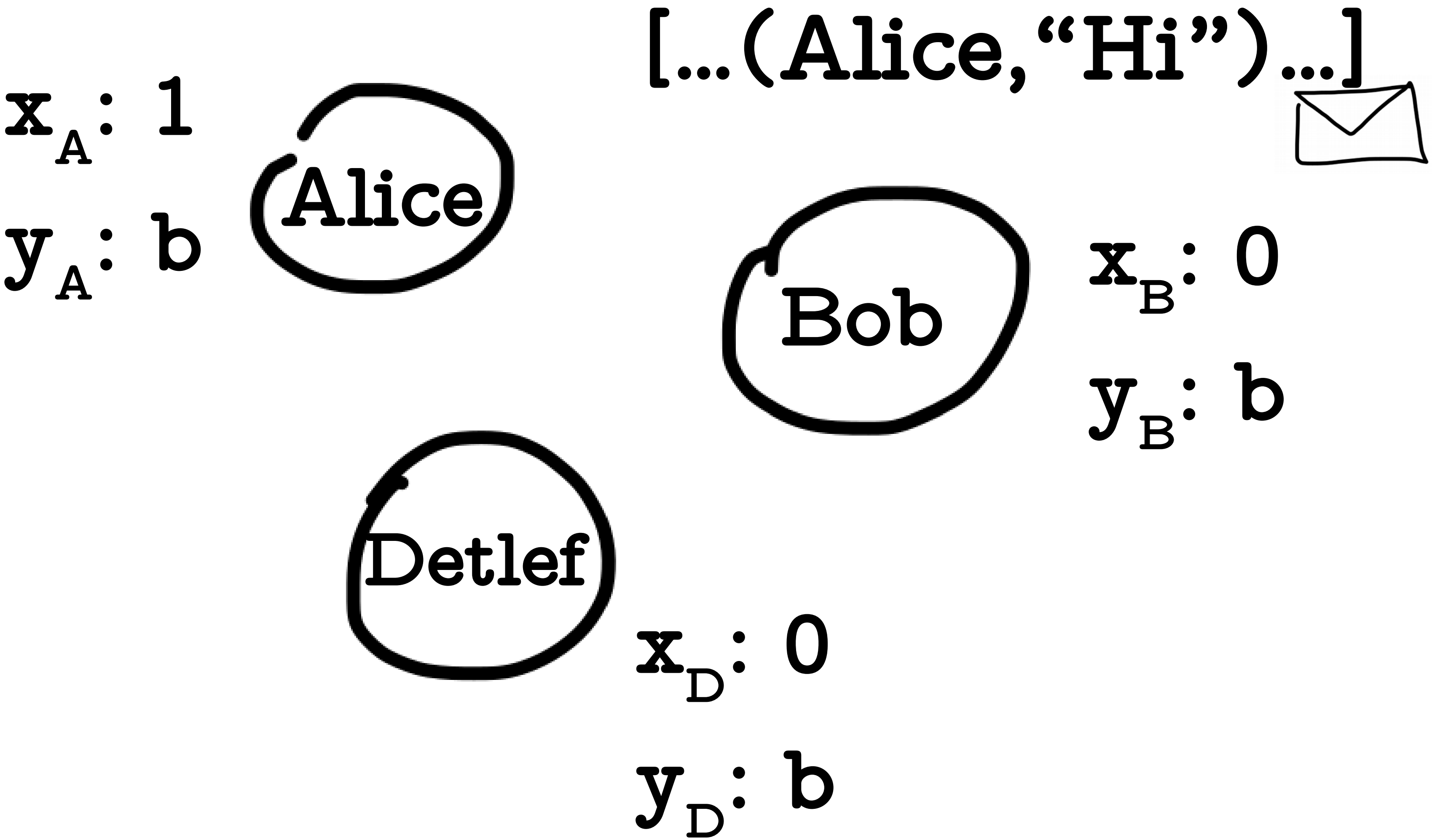
y_D: ?

Decision Values = {0, 1}

- Assumption: “write-once”



Configuration



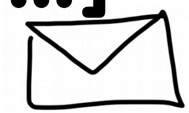
Configuration→Configuration

x_A: 1

y_A: b

Alice

[...(Alice,“Hi”)...]



- A reads message m from buffer
- *changes internal state*
- puts finite m_i messages in buffer

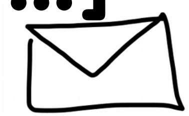
Configuration→Configuration

x_A: 1

y_A: b

Alice

[...(Alice,“Hi”)...]



- event $\mathbf{e} = (\mathbf{p}, \mathbf{m})$ determines configuration after $\mathtt{C}$, apply $\mathbf{e}$ to $\mathtt{C}$
- schedule $\boldsymbol{\sigma} = \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \ldots \mathbf{e}_n$
- run

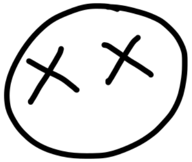
Theorem 1.

No consensus protocol is totally correct in spite of one fault.

- *Proof Idea*
  - Assume consensus protocol `P` totally correct in spite of one fault; contradiction.

Theorem 1.

No consensus protocol is totally correct in spite of one fault.

- *totally correct in spite of*  *:=*
  - *partially correct*
  - *every admissible run is deciding*

contradiction
for assumed **P**

$\mathtt{P}$ indecisive forever

1) exists initial configuration

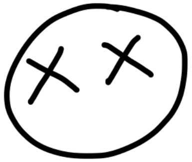
- in which decision is not determined

2) some step(s)

- which don’t go towards decision

Theorem 1.

No consensus protocol is totally correct in spite of one fault.

- *totally correct in spite of*  *:=*
  - *partially correct*
  - *every admissible run is deciding*

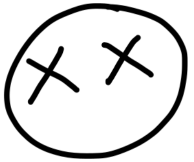
Theorem 1.

- no <u>accessible</u> configuration has more than one decision value

- for each decision value **v** in {**0, 1**} some accessible configuration has **v**

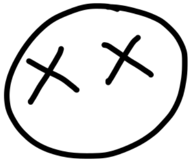
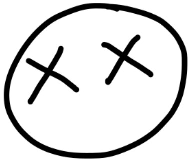
No

<u>**co**</u>

- *totally correct in spite of*  *:=*
  - <u>*partially correct*</u>
  - *every admissible run is deciding*

Theorem 1.

No consensus protocol is totally correct in spite of one fault.

- not  if process takes infinitely many steps
- else 

`P` indecisive forever

1) exists initial configuration

- in which decision is not determined

Lemma 2

2) some step(s)

- which don't go towards decision

Lemma 3

Bi/Uni/0/1 - Valent

- decision values {0,1} /
- one decision value /
- 0 /
- 1

reachable from configuration C

P indecisive forever

1) exists initial configuration

- in which decision is not determined

bivalent C_{init}

2) some step(s)

- which don't go towards decision

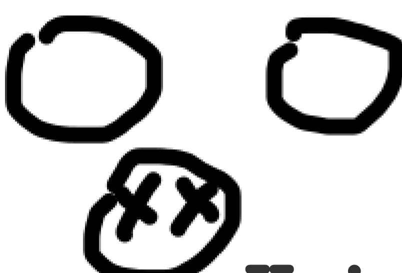
bivalent C → bivalent C'

Lemma 2.

`P` has bivalent initial configur’n.

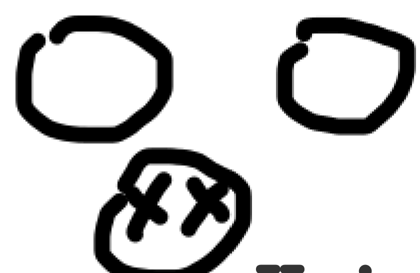
Assume not, by `P` we can construct:

C_0 0 valent



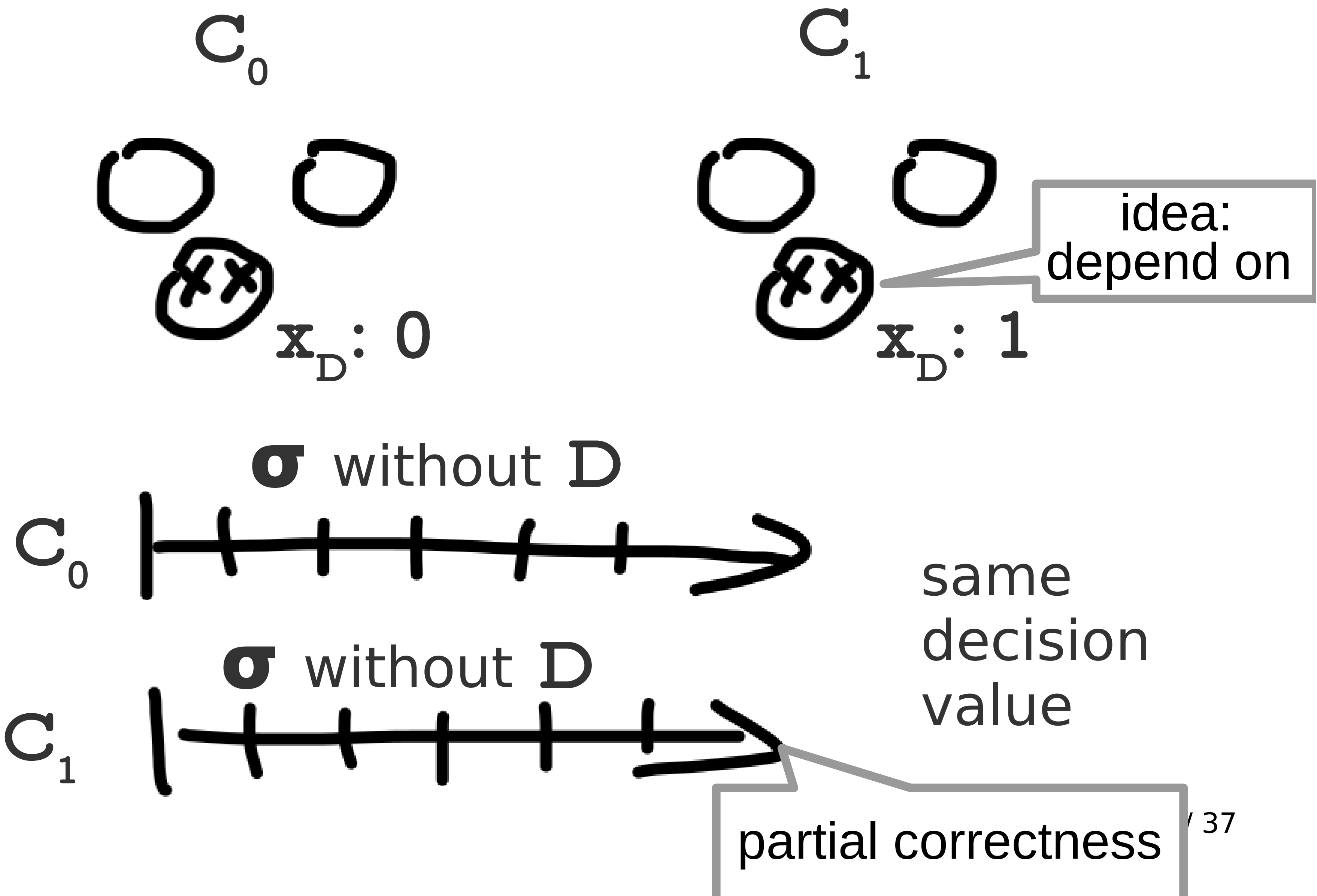
x_D: 0

C_1 1 valent



x_D: 1

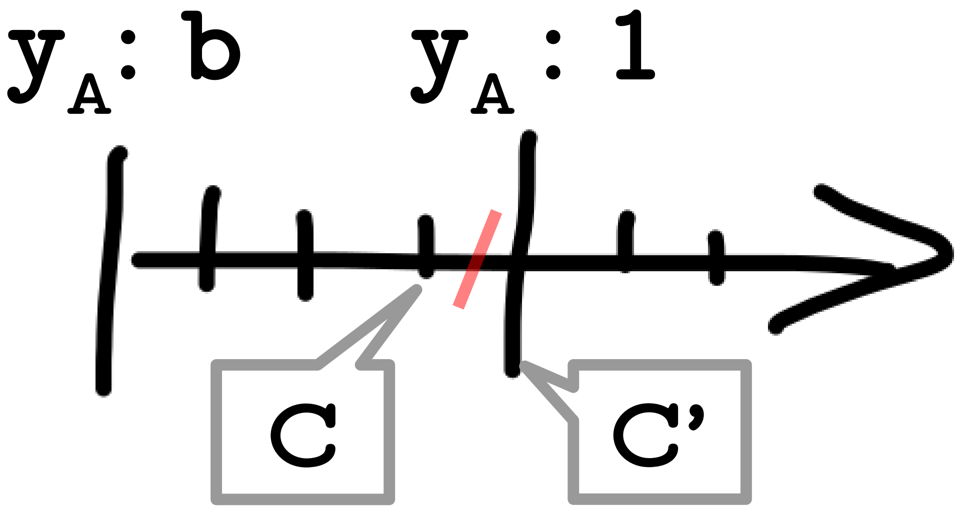
Lemma 2.



C_0
C_1
x_D: 0
x_D: 1
idea:
depend on
σ without D
C_0
σ without D
C_1
same
decision
value
partial correctness

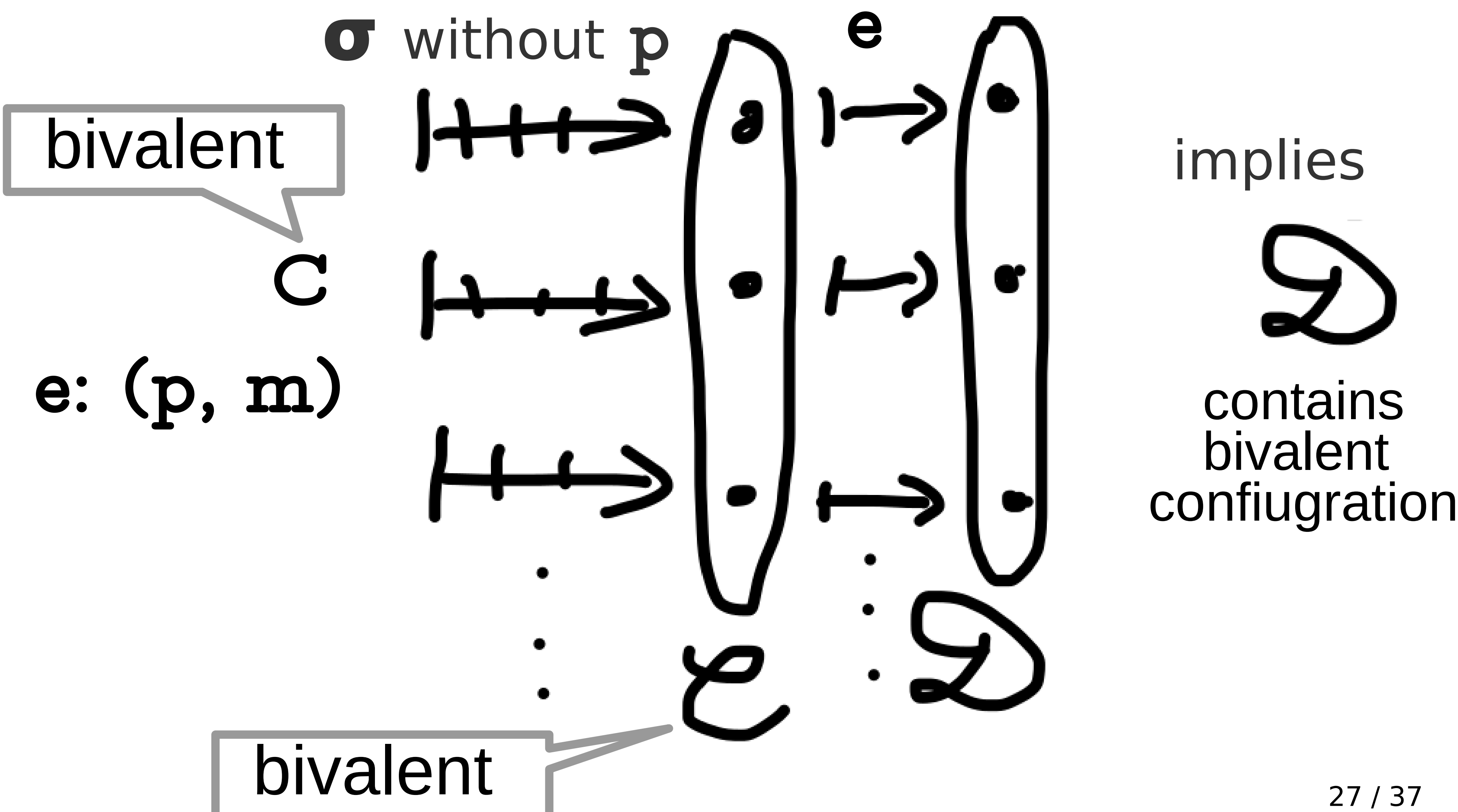
Idea: Lemma 3.

- step from bivalent C to univalent C'



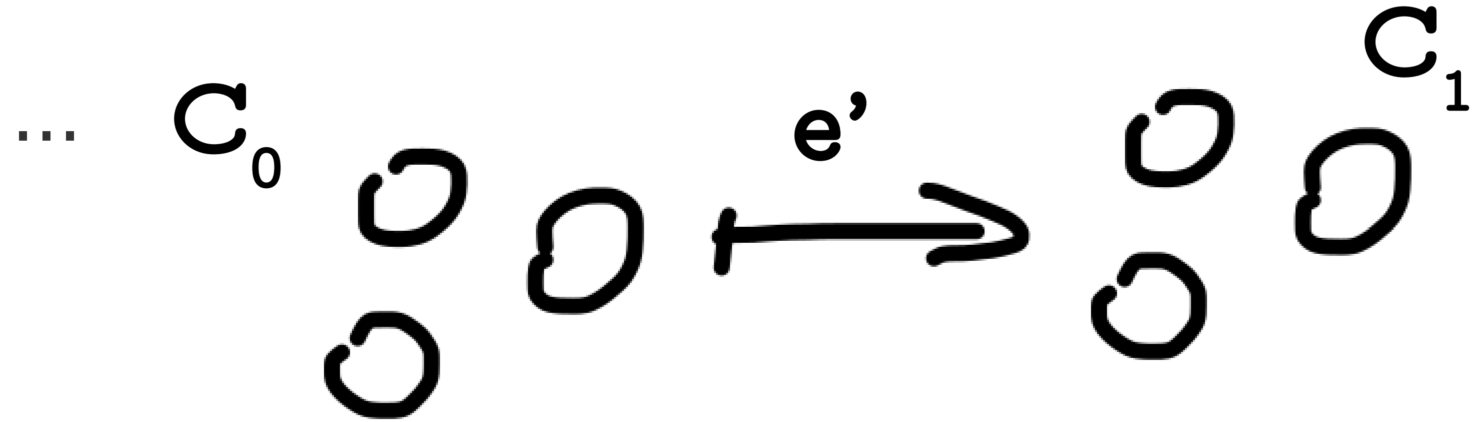
- Plan:
  Don't take that step—take another!

Lemma 3.



σ without p
e
bivalent
C
e: (p, m)
bivalent
implies
contains
bivalent
confiugration

Proof Sketch: Lemma 3.



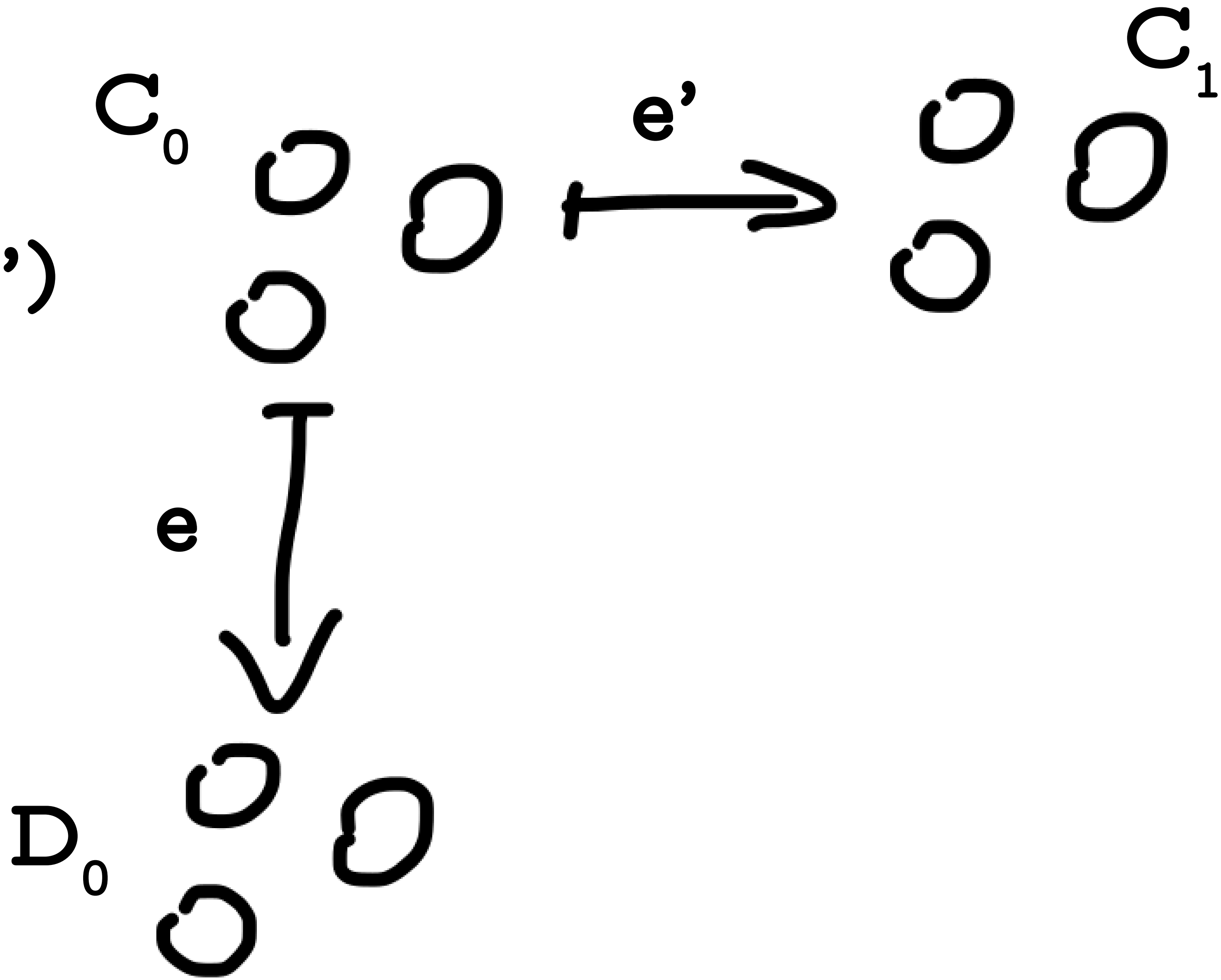
reason by contradiction

- C_0 C_1 in $\mathcal{C}$ — bivalent
- D_0 D_1 in $\mathcal{D}$ — univalent

Proof Sketch: Lemma 3.

e’: (p’, m’)

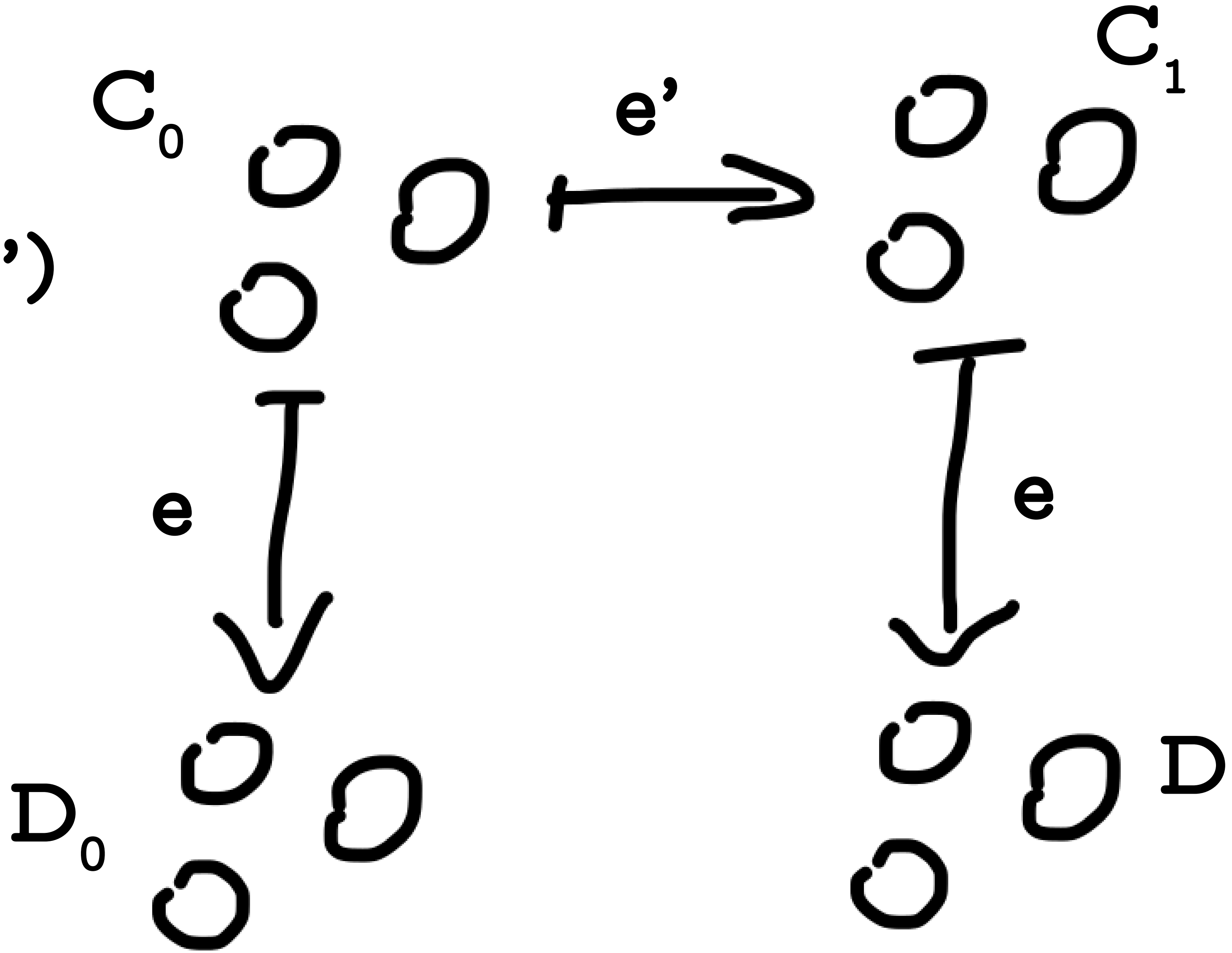
e: (p, m)



Proof Sketch: Lemma 3.

e’: (p’, m’)

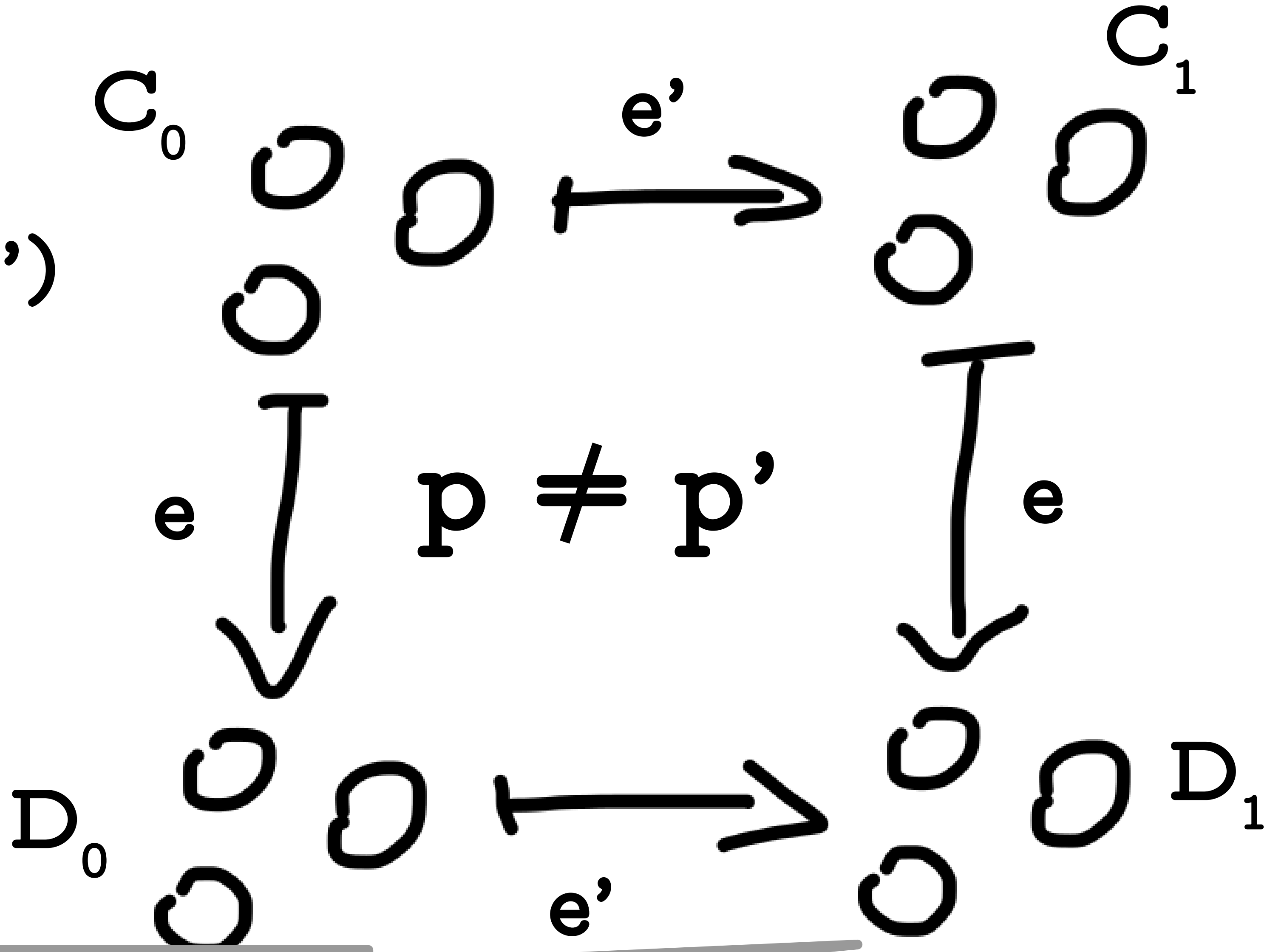
e: (p, m)



Proof Sketch: Lemma 3.

e': (p', m')

e: (p, m)

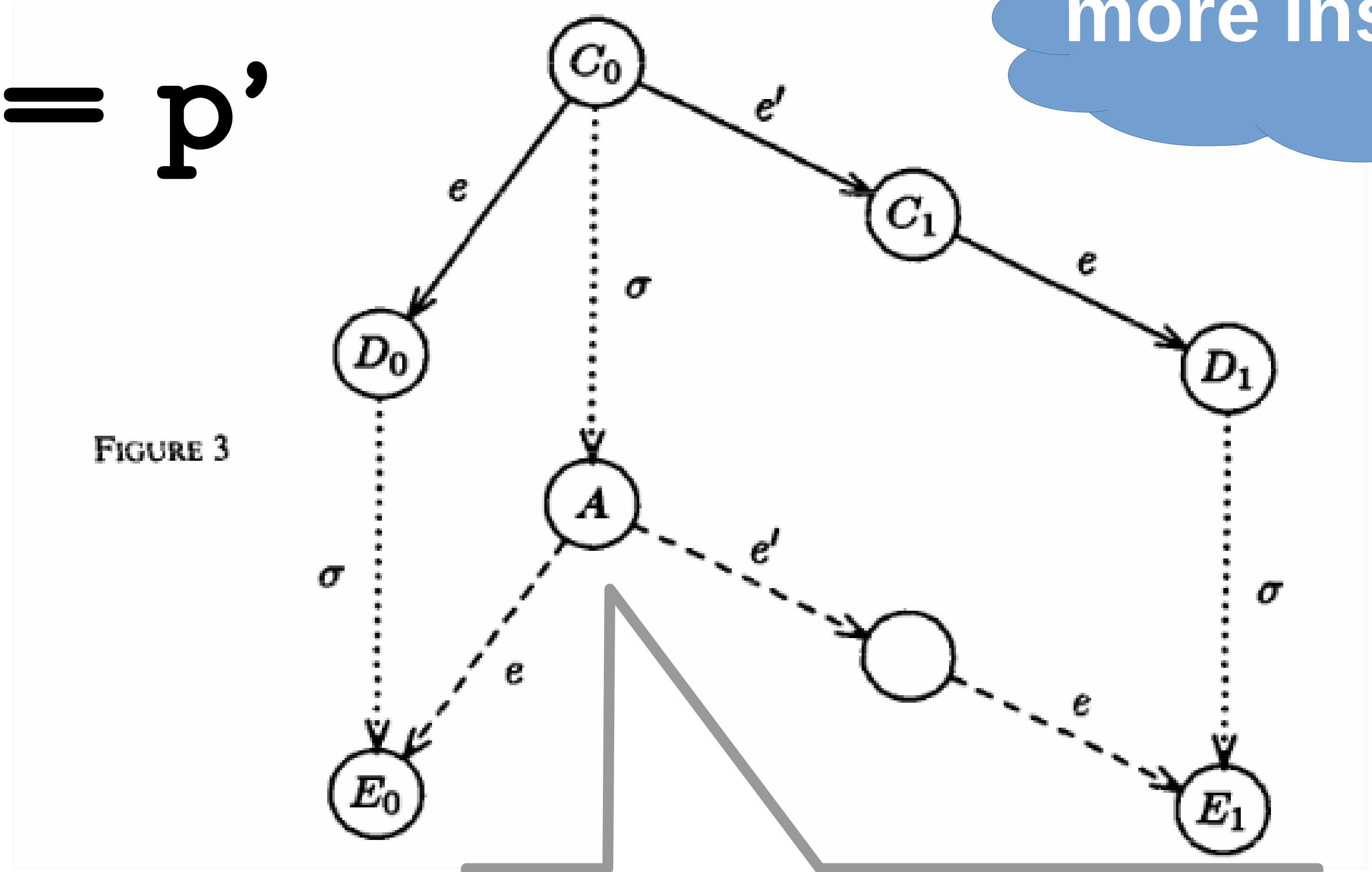


1-valent

Proof Sketch: Lemma 3.

more insight?

p = p'



Assume univalent

`P` indecisive forever

1) exists initial configuration

– in which decision is not determined

Lemma 2

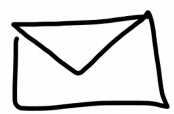
2) some step(s)

– which don’t go towards decision

Lemma 3

Theorem 1.

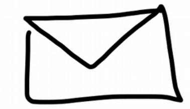
No consensus protocol is totally correct in spite of one fault.

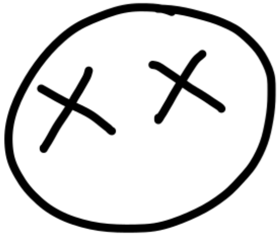
- •  to non-faulty nodes eventually received

- *totally corre*
  - *partially corr*
  - *every <u>admissible</u> run is deciding*

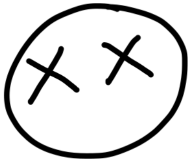
contradiction
for assumed **P**

Admissable Run

- queue of processes `A, B, D`
- message buffer FIFO 
- by Lemma 2 & 3. indecisive
  → contradiction; `P` not totally correct

Initially 

protocol to
detect proc.

- possible to detect initially 

→ partially correct consensus protocol if more than half of the initial processes were alive & no process dies

2f +1

Summary

- appreciate abstract model
- proofs by contradiction are difficult to keep track