

# Failure Detectors

Seif Haridi - Royal Institute of Technology

Peter Van Roy - Université catholique de Louvain

haridi(at)kth.se

peter.vanroy(at)uclouvain.be

## Modeling Timing Assumptions

- Tedious to model eventual synchrony (partial synchrony)
- Timing assumptions mostly needed to detect failures
  - Heartbeats, timeouts, etc...
- Use **failure detectors** to **encapsulate timing assumptions**
  - Black box giving **suspicion**s regarding node failures
  - Accuracy of suspicions depends on model strength
  - Invention of failure detectors around 1990 was a major step toward the maturity of the field of distributed algorithms as opposed to a disorganized bunch of complicated algorithms

## Implementation of Failure Detectors

### ■ Typical Implementation

- Periodically exchange **heartbeat** messages
- **Timeout** based on **worst case** msg round trip
- If timeout, then **suspect** node
- If recv msg from suspected node, **revise suspicion** and increase time-out

19/3/2012

Based on Ghods's slides

3

## Completeness and Accuracy

### ■ Two important types of requirements

- **1. Completeness requirements**
  - **Requirements regarding actually crashed nodes**
    - When do they have to be detected?
- **2. Accuracy requirements**
  - **Requirements regarding actually alive nodes**
    - When are they allowed to be suspected?
- How to trivially achieve either? **[d]**
  - Both impossible in an asynchronous system!

19/3/2012

Based on Ghods's slides

4

## Formal Model of FD

- Augment formal model with failure detectors (FD)
- A configuration consists of
  - State of each node
  - **FD-state** of each node
- Transition function on node  $i$  gets **extra parameter**:
  - FD-state of node  $i$
- FD-state updated in  $\text{comp}(i)$  by **another function**
  - **FD-function**
  - Not modeled explicitly, but must satisfy some properties

19/3/2012

Based on Ghodsi's slides

5

## Requirements: Completeness

- **Strong Completeness**
  - Every crashed node is **eventually** detected by **all correct** nodes
- There exists a time after which all crashed nodes are detected by all correct nodes
  - The book only studies detectors with this property
- Is it realistic? **[d]**

19/3/2012

Based on Ghodsi's slides

6

## Requirements: Completeness

- **Weak Completeness**

- Every crashed node is *eventually* detected by *some* correct node

- There exists a time after which all crashed nodes are detected by some correct node

- A privileged node with good view of the others!
- But: possibly detected by *different* correct nodes
  - Several privileged nodes, with good views to some

19/3/2012

Based on Ghodsi's slides

7

## Requirements: Accuracy

- **Strong Accuracy**

- No correct node is ever suspected

- For all nodes p and q,

- p does not suspect q, unless q has crashed

- Is it realistic? **[d]**

- Strong assumption, requires synchrony
- I.e. no premature timeouts

19/3/2012

Based on Ghodsi's slides

8

## Requirements: Accuracy

- **Weak Accuracy**
  - There exists a correct node which is never suspected by any node
- There exists a correct node P
  - Such that all nodes will never suspect P
- This is still quite a strong assumption
  - There is a privileged node that is always seen

19/3/2012

Based on Ghodsi's slides

9

## Requirements: Accuracy

- **Eventual Strong Accuracy**
  - After some finite time the FD provides **strong accuracy**
- **Eventual Weak Accuracy**
  - After some finite time the detector provides **weak accuracy**
- After some time, the requirements are fulfilled
  - Prior to that, any behavior is possible!
- Quite weak assumptions **[d]**
  - When can eventual weak accuracy be achieved?

19/3/2012

Based on Ghodsi's slides

10

## Four Main Established Detectors

- Four detectors with **strong completeness**

- **Perfect Detector (P)**
    - Strong Accuracy
  - **Strong Detector (S)**
    - Weak Accuracy
- } Synchronous Systems

- **Eventually Perfect Detector ( $\Diamond P$ )**
    - Eventual Strong Accuracy
  - **Eventually Strong Detector ( $\Diamond S$ )**
    - Eventual Weak Accuracy
- } Asynchronous Systems

19/3/2012

Based on Ghodsi's slides

11

## Four Less Interesting Detectors

- Four detectors with **weak completeness**

- **Detector Q**
    - Strong Accuracy
  - **Weak Detector (W)**
    - Weak Accuracy
- } Synchronous Systems

- **Eventually Detector Q ( $\Diamond Q$ )**
    - Eventual Strong Accuracy
  - **Eventually Weak Detector ( $\Diamond W$ )**
    - Eventual Weak Accuracy
- } Asynchronous Systems

19/3/2012

Based on Ghodsi's slides

12

## Interface of Perfect Failure Detector

### ■ Module:

- Name: PerfectFailureDetector (P)

### ■ Events:

- **Indication:**  $\langle \text{crash} \mid p_i \rangle$ 
  - Notifies that node  $p_i$  has crashed

### ■ Properties:

- **PFD1 (strong completeness)**
- **PFD2 (strong accuracy)**

19/3/2012

Based on Ghodsi's slides

13

## Properties of P

### ■ Properties:

- **PFD1 (strong completeness)**
  - Eventually every node that **crashes** is permanently detected by every correct node (**liveness**)
- **PFD2 (strong accuracy)**
  - If a node  $p$  is detected by any node, then  $p$  has crashed (**safety**)

### ■ Safety or Liveness?

19/3/2012

Based on Ghodsi's slides

14

## Implementing P in Synchrony

- Assume **synchronous system**
  - Max transmission delay between 0 and  $\delta$  time units
- Each node every  $\gamma$  time units
  - Send <heartbeat> to all nodes
- Each node waits  $\gamma + \delta$  time units
  - If did not get <heartbeat> from  $p_i$ 
    - Detect <crash |  $p_i$ >

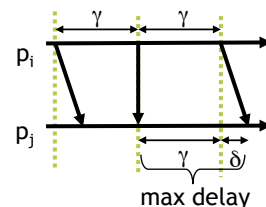
19/3/2012

Based on Ghodsi's slides

15

## Correctness of P

- **PFD1 (strong completeness)**
  - A crashed node doesn't send <heartbeat>
    - Eventually every node will notice the absence of <heartbeat>
- **PFD2 (strong accuracy)**
  - Assuming local computation is negligible
  - Maximum time between 2 heartbeats
    - $\gamma + \delta$  time units
  - If alive, all nodes will recv hb in time
    - No inaccuracy



19/3/2012

Based on Ghodsi's slides

16



## Interface of EPFD

- **Module:**
  - Name: EventuallyPerfectFailureDetector ( $\Diamond P$ )
- **Events:**
  - **Indication:**  $\langle \text{suspect} \mid p_i \rangle$ 
    - Notifies that node  $p_i$  is suspected to have crashed
  - **Indication:**  $\langle \text{restore} \mid p_i \rangle$ 
    - Notifies that node  $p_i$  is not suspected anymore
- **Properties:**
  - *PFD1 (strong completeness)*
  - *PFD2 (eventual strong accuracy).* **Eventually**, no correct node is suspected by any correct node.

19/3/2012

Based on Ghodsi's slides

17

## Implementing $\Diamond P$

- Assume **partially synchronous system**
  - Eventually some bounds exists
- Each node every  $\gamma$  time units
  - Send  $\langle \text{heartbeat} \rangle$  to all nodes
- Each node waits  $T$  time units
  - If did not get  $\langle \text{heartbeat} \rangle$  from  $p_i$ 
    - Indicate  $\langle \text{suspect} \mid p_i \rangle$  if  $p_i$  is not in **suspected**
    - Put  $p_i$  in **suspected** set
  - If get HB from  $p_i$ , and  $p_i$  is in **suspected**
    - Indicate  $\langle \text{restore} \mid p_i \rangle$  and remove  $p_i$  from **suspected**
    - Increase timeout  $T$

19/3/2012

Based on Ghodsi's slides

18

## Correctness of $\Diamond P$

- **EPFD1 (strong completeness)**
  - Same as before
- **EPFD2 (eventual strong accuracy)**
  - Each time  $p$  is inaccurately suspected by a correct  $q$ 
    - Timeout  $T$  is increased at  $q$
    - Eventually system becomes synchronous, and  $T$  becomes larger than the **unknown bound**  $\delta$  ( $T > \gamma + \delta$ )
    - $q$  will receive HB on time, and never suspect  $p$  again

## Leader Election

## Leader Election vs Failure Detection

- Failure detection captures failure behavior
  - Detect **failed** nodes
- Leader election (LE) also captures failure behavior
  - Detect **correct** nodes (a single & same for all)
- Formally, **leader election is a FD**
  - Always suspects all nodes except one (leader)
  - Ensures some properties regarding that node

19/3/2012

Based on Ghodsi's slides

21

## Leader Election vs Failure Detection

- We'll define two leader election algorithms
  - **Leader election (LE)** which "matches"  $P$
  - **Eventual leader election ( $\Omega$ )** which "matches"  $\Diamond P$

19/3/2012

Based on Ghodsi's slides

22

## Matching LE and P

- P's properties
  - P always eventually detects failures (strong completeness)
  - P never suspects correct nodes (strong accuracy)
- Completeness of LE
  - Informally: eventually ditch crashed leaders
  - Formally: **eventually** every correct node trusts **some** correct node
- Accuracy of LE
  - Informally: never ditch a correct leader
  - Formally: No two **correct** nodes trust different **correct** nodes
    - Is this really accuracy? [d]
    - Yes! Assume two nodes trust different correct nodes
      - One of them must eventually switch, i.e. leaving a correct node

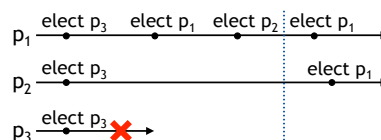
19/3/2012

Based on Ghods's slides

23

## LE desirable properties

- LE always eventually detects failures
  - **Eventually every correct node trusts some correct node**
- LE is always accurate
  - **No two correct nodes trust different correct nodes**
- But the above two permit the following



- But P<sub>1</sub> is “inaccurately” leaving a correct leader

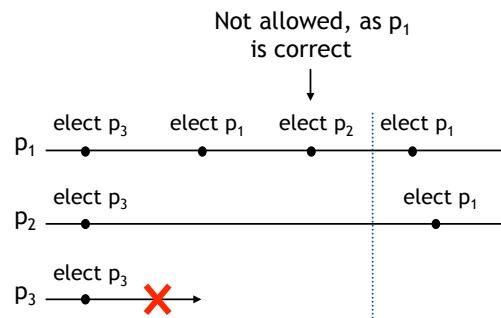
19/3/2012

Based on Ghods's slides

24

## LE desirable properties

- To avoid “inaccuracy” we add
  - **Local Accuracy:**
    - If a node is elected leader by  $p_i$ , all previously elected leaders by  $p_i$  have crashed



19/3/2012

Based on Ghodsi's slides

25

## Interface of Leader Election

- **Module:**
  - Name: LeaderElection (le)
- **Events:**
  - **Indication:**  $\langle \text{leLeader} \mid p_i \rangle$ 
    - Indicate that leader is node  $p_i$
- **Properties:**
  - **LE1 (eventual completeness).** Eventually every correct node trusts some correct node
  - **LE2 (agreement).** No two correct nodes trust different correct nodes
  - **LE3 (local accuracy).** If a node is elected leader by  $p_i$ , all previously elected leaders by  $p_i$  have crashed

19/3/2012

Based on Ghodsi's slides

26

## Implementing LE

- Globally rank all nodes
  - E.g. rank ordering  $p_1 > p_2 > p_3 > p_4$
  - Represented by **function  $r$**  which returns all nodes with higher ranking
  - $r(p_1) = \emptyset$ ,
  - $r(p_2) = \{p_1\}$ ,
  - $r(p_3) = \{p_1, p_2\}$
  - $r(p_4) = \{p_1, p_2, p_3\}$

## Implementing LE (2)

- **Implements:** LeaderElection (le).
- **Uses:** PerfectFailureDetector (P).
- **upon event  $\langle \text{init} \rangle$  do**
  - $\text{suspected} = \emptyset$ ;  $\text{leader} := \text{highest}(r)$
  - **trigger**  $\langle \text{leLeader} \mid \text{leader} \rangle$
- **upon event  $\langle \text{crash} \mid p_i \rangle$  do**
  - $\text{suspected} := \text{suspected} \cup \{p_i\}$
- **upon exists  $p_i$  s.t.  $r(p_i) \subseteq \text{suspected} \wedge p_i \notin \text{suspected}$  do**
  - $\text{leader} := p_i$
  - **trigger**  $\langle \text{leLeader} \mid p_i \rangle$

## Matching $\Omega$ and $\Diamond P$

- $\Diamond P$  weakens  $P$  by only providing eventual accuracy
  - Weaken LE to  $\Omega$  by only guaranteeing **eventual agreement**

■ **LE Properties:**

- **LE1 (eventual completeness).** Eventually every correct node trusts some correct node
- **LE2 (agreement).** No two correct nodes trust different correct nodes
- ~~**LE3 (local accuracy).** If a node is elected leader by  $p_i$ , all previously elected leaders by  $p_i$  have crashed~~

eventual

19/3/2012

Abdullah Al-Hajjaj

29

## Interface of Eventual Leader Election

- **Module:**
  - Name: EventualLeaderElection ( $\Omega$ )
- **Events:**
  - **Indication:**  $\langle \text{leLeader} \mid p_i \rangle$ 
    - Notify that  $p_i$  is trusted to be leader
- **Properties:**
  - **ELD1 (eventual completeness).** **Eventually** every correct node trusts some correct node
  - **ELD2 (eventual agreement).** **Eventually** no two correct nodes trust different correct nodes

19/3/2012

Abdullah Al-Hajjaj

30

## Eventual Leader Detection $\Omega$

- In crash-stop process abstraction
  - $\Omega$  is obtained directly from  $\Diamond P$
  - Each node trusts the node with highest id among all nodes not suspected by  $\Diamond P$
  - Eventually, exactly one correct process will be trusted by all correct processes

## Implementing $\Omega$

- **Implements:** EventualLeaderElection ( $\Omega$ ).
- **Uses:** EventuallyPerfectFailureDetector ( $\Diamond P$ ).
- **upon event**  $\langle \text{init} \rangle$  **do**
  - $\text{suspected} := \emptyset$ ;  $\text{leader} := p_n$ ;
  - **trigger**  $\langle \text{leLeader} \mid \text{leader} \rangle$
- **upon event**  $\langle \text{suspect} \mid p_i \rangle$  **do**
  - $\text{suspected} := \text{suspected} \cup \{p_i\}$
- **upon event**  $\langle \text{restore} \mid p_i \rangle$  **do**
  - $\text{suspected} := \text{suspected} \setminus \{p_i\}$
- **upon exists**  $p_i$  **s.t.**  $r(p_i) \subseteq \text{suspected} \wedge p_i \notin \text{suspected}$  **do**
  - $\text{leader} := p_i$
  - **trigger**  $\langle \text{leLeader} \mid p_i \rangle$



## $\Omega$ for Crash Recovery

- Can we elect a recovered node? [d]
  - Not if it keeps crash-recovering infinitely often!
- Basic idea
  - Count number of times you've crashed (**epoch**)
  - Distribute your **epoch** periodically to all nodes
  - Elect leader with lowest (**epoch, node\_id**)
- Implementation
  - Similar to  $\Diamond P$  and  $\Omega$  for crash-stop
  - Piggyback **epoch** with heartbeats
  - Store and load leader upon crash

## Reductions

## Reductions

- We say  $X \leq Y$  if
  - X can be solved given a solution of Y
  - Read X is reducible to Y
  - Informally, problem X is easier than or as hard as Y

19/3/2012

Based on Ghodsi's slides

35

## Preorders, partial orders...

- A relation  $\sim$  is a **preorder** on a set A if for any  $x, y, z$  in A
  - $x \sim x$  (**reflexivity**)
  - $x \sim y$  and  $y \sim z$  implies  $x \sim z$  (**transitivity**)
- Difference between preorder and partial order
  - Partial order is a preorder with **antisymmetry**
    - $x \sim y$  and  $y \sim x$  implies  $x = y$
  - I.e. two **different** objects  $x$  and  $y$  cannot be symmetric
    - i.e. it isn't possible that  $x \sim y$  and  $y \sim x$  for two **different**  $x$  and  $y$

19/3/2012

Based on Ghodsi's slides

36

## $\leq$ is a preorder

- $\leq$  is a preorder
  - **Reflexivity.**  $X \leq X$ 
    - $X$  can be solved given a solution to  $X$
  - **Transitivity.**  $X \leq Y$  and  $Y \leq Z$  implies  $X \leq Z$ 
    - Since  $Y \leq Z$ , use impl. of  $Z$  to impl.  $Y$ .  
use impl. of  $Y$  to impl.  $X$ . Hence we impl.  $X$  from  $Z$ 's impl.
- $\leq$  is **not** antisymmetric, thus not a partial order
  - Two different  $X$  and  $Y$  can be equivalent
    - Distinct problems  $X$  and  $Y$  can be solved from the other's solution

19/3/2012

Based on Ghodsi's slides

37

## Shortcut definitions

- We write  $X \approx Y$  if
  - $X \leq Y$  and  $Y \leq X$
  - Problem  $X$  is **equivalent** to  $Y$
- We write  $X < Y$  if
  - $X \leq Y$  and not  $X \approx Y$
  - or equivalently,  $X \leq Y$  and not  $Y \leq X$
  - Problem  $X$  is **strictly weaker** than  $Y$ , or
  - Problem  $Y$  is **strictly stronger** than  $X$

19/3/2012

Based on Ghodsi's slides

38

## Example

- It is true that  $\Diamond P \leq P$ 
  - Given P, we can implement  $\Diamond P$ 
    - We just return P's suspicions.
    - P always satisfies  $\Diamond P$ 's properties
- In fact,  $\Diamond P < P$  in the asynchronous model
  - Because it is not true that  $P \leq \Diamond P$
- Reductions common in computability theory
  - If  $X \leq Y$ , and if we know X is **impossible** to solve
    - Then Y is impossible to solve too
  - If  $\Diamond P \leq P$ , and some problem Z can be solved with  $\Diamond P$ 
    - Then Z can also be solved with P

## Weakest FD for a problem?

- Often P is used to solve problem X
  - But P is not very practical (needs synchrony)
  - Is X a “practically” solvable problem?
    - Can we implement X with  $\Diamond P$ ?
    - Sometimes a weaker FD than P will not solve X
      - Proven using **reductions**

## Weakest FD for a problem

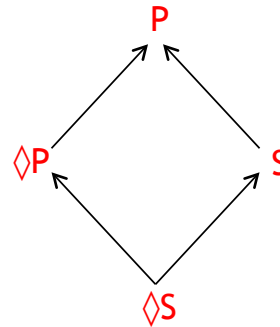
- We might know that  $X \leq P$  (X solvable with P)
  - Can we solve X with a weaker FD than P, say  $\Diamond P$ ?
  - Or is it impossible, i.e.  $\Diamond P < X$
- Common proof to show P is weakest FD for X
  - Prove that  $P \leq X$
  - I.e. P can be solved given X
- If  $P < X$  then  $\Diamond P < X$ 
  - Because we know  $\Diamond P < P$  and  $P \approx X$ , i.e.  $\Diamond P < P \approx X$ 
    - If we can solve X with  $\Diamond P$ , then
    - we can solve P with  $\Diamond P$ , which is a contradiction

## How are the detectors related

## Trivial Reductions

### Strongly complete

- $\Diamond P \leq P$ 
  - P is always strongly accurate, thus also eventually strongly accurate
- $\Diamond S \leq S$ 
  - S is always weakly accurate, thus also eventually weakly accurate
- $S \leq P$ 
  - P is always strongly accurate, thus also always weakly accurate
- $\Diamond S \leq \Diamond P$ 
  - $\Diamond P$  is always eventually strongly accurate, thus also always eventually weakly accurate



19/3/2012

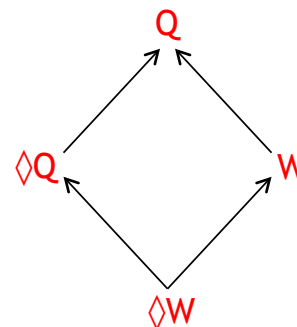
Based on Ghods's slides

43

## Trivial Reductions (2)

### Weakly complete

- $\Diamond Q \leq Q$ 
  - Q is always strongly accurate, thus also eventually strongly accurate
- $\Diamond W \leq W$ 
  - W is always weakly accurate, thus also eventually weakly accurate
- $W \leq Q$ 
  - Q is always strongly accurate, thus also always weakly accurate
- $\Diamond W \leq \Diamond Q$ 
  - $\Diamond Q$  is always eventually strongly accurate, thus also always eventually weakly accurate



19/3/2012

Based on Ghods's slides

44

## Completeness “Irrelevant”

- Weak completeness **trivially reducible** to strong
- Strong completeness **reducible** to weak
  - i.e. can get strong completeness from weak
    - $P \leq Q, S \leq W, \Diamond P \leq \Diamond Q, \Diamond S \leq \Diamond W$ ,
  - They're **equivalent**!
    - $P = Q, S = W, \Diamond P = \Diamond Q, \Diamond S = \Diamond W$

$\neg Q, S=W, \Diamond P=\Diamond Q, \Diamond S=\Diamond W$

		Accuracy			
Completeness		Strong	Weak	Eventual Strong	Eventual Weak
	Strong	P	S	$\Diamond P$	$\Diamond S$
	Weak	Q	W	$\Diamond Q$	$\Diamond W$

19/3/2012

Based on Ghodsi's slides

45

## Proving Irrelevance of Completeness

- Weak completeness ensures
  - Every crash is eventually detected by some correct node
- Simple idea
  - Every node  $q$  broadcasts suspicions **Susp** **periodically**
  - **upon event receive**  $\langle S, q \rangle$ 
    - $\text{Susp} := (\text{Susp} \cup S) - \{q\}$  ← also works like a heartbeat
- Every crash is eventually detected by all correct  $p$ 
  - Can this violate some accuracy properties?

19/3/2012

Based on Ghodsi's slides

46

## Maintaining Accuracy

- Strong and Weak Accuracy aren't violated
- Strong accuracy
  - No one is ever inaccurate
  - Our reduction never spreads inaccurate suspicions
- Weak accuracy
  - Everyone is accurate about at least one node  $p$ 
    - No one will spread inaccurate information about  $p$

19/3/2012

Based on Ghodsi's slides

47

## Maintaining Eventual Accuracy

- Eventual Strong and Eventual Weak Accuracy aren't violated
- Proof is almost same as previous page
  - Eventually all faulty nodes crash
  - Inaccurate suspicions **undone**
    - Will get heartbeat from correct nodes and revise ( $\neg\{q\}$ )

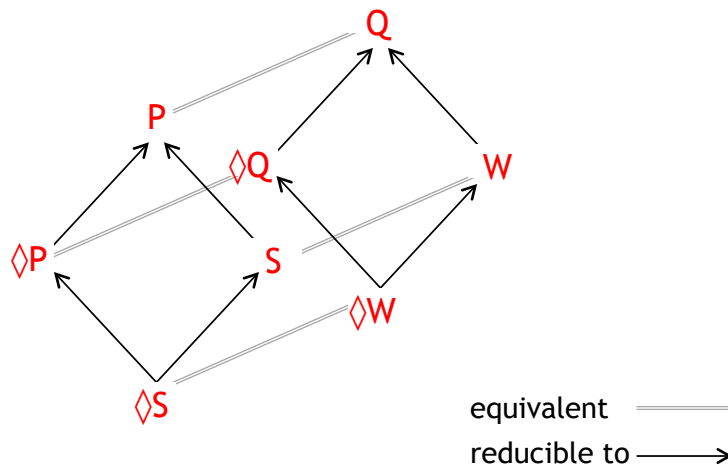
19/3/2012

Based on Ghodsi's slides

48



## Relation between FDs



19/3/2012

Based on Ghodsi's slides

49

## Omega also a FD

- Can we implement  $\diamond S$  with  $\Omega$ ? [d]
  - I.e. is it true that  $\diamond S \leq \Omega$
  - Suspect all nodes except the leader given by  $\Omega$ 
    - **Eventual Completeness**
      - All nodes are suspected except the leader (which is correct)
    - **Eventual Weak Accuracy**
      - Eventually, one correct node (leader) is not suspected by anyone
  - Thus,  $\diamond S \leq \Omega$

19/3/2012

Based on Ghodsi's slides

50

## $\Omega$ equivalent to $\Diamond S$ (and $\Diamond W$ )

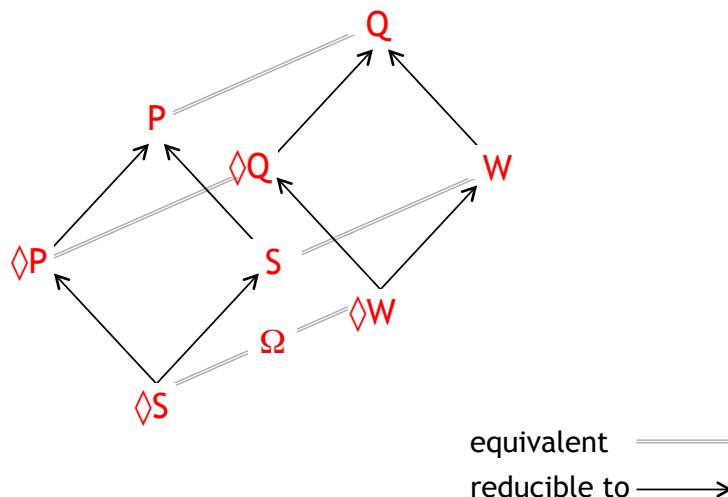
- We showed  $\Diamond S \leq \Omega$ , it turns out we also have  $\Omega \leq \Diamond S$ 
  - I.e.  $\Omega \approx \Diamond S$
- Due to the famous CHT Theorem
  - CHT Theorem (1996):
    - If consensus implementable with detector D, then  $\Omega$  can be implemented using D
  - I.e. if  $\text{Consensus} \leq D$ , then  $\Omega \leq D$ 
    - Since  $\Diamond S$  can be used to solve consensus, we have  $\Omega \leq \Diamond S$
  - Implies  $\Diamond W$  is weakest detector to solve consensus
  - Only important proof that is omitted in this course!

19/3/2012

Based on Ghodsi's slides

51

## Relation between FDs (2)



19/3/2012

Based on Ghodsi's slides

52

## Combining Abstractions

- **Fail-stop (synchronous)**
  - Crash-stop process model
  - Perfect links + Perfect failure detector (P)
- **Fail-silent (asynchronous)**
  - Crash-stop process model
  - Perfect links
- **Fail-noisy (partially synchronous)**
  - Crash-stop process model
  - Perfect links + Eventually Perfect failure detector ( $\Diamond P$ )
- **Fail-recovery**
  - Crash-recovery process model
  - Stubborn links + ...

19/3/2012

Based on Ghodsi's slides

53

## Rest of course: “from P to $\Diamond P$ ”

- Assume crash-stop system with a **perfect failure detector** (fail-stop)
  - Give algorithms
- Then try to make a weaker assumption
  - For example, **eventually perfect failure detector**
  - Revisit the algorithms:
    - Do they still work? Can they be made to work?

19/3/2012

Based on Ghodsi's slides

54