TECHNISCHE UNIVERSITÄT CHEMNITZ

Dependable Systems
Winter term 2015/2016

Dependable Systems

5. Chapter

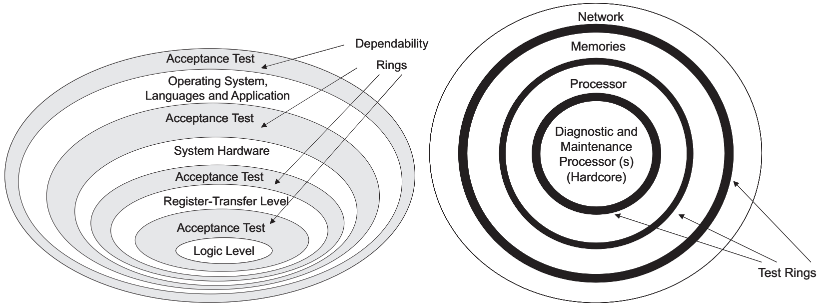
Error Diagnosis

Prof. Matthias Werner

Operating Systems Group

5.1 Introduction

- Diagnosis is a standard approach in design of dependable systems
- Use:
  - Error diagnosis to fault masking
  - Error diagnosis to fault containment (avoidance of fault propagation)
- Containment may be structured in a layered or component-oriented way



Tests

- Diagnosis includes several aspects
  - Detect that a fault exists ➡ **fault detection**
  - Localization of a fault
- Fault detection is done by **testing/checking**
- Evaluation criterion of a test is **coverage** ➡ see 3.12

Attention

In case a test detects a fault it is not implied that the testee is faulty.
It is also possible that the tester is faulty!

- More in Section “System Level Diagnosis”

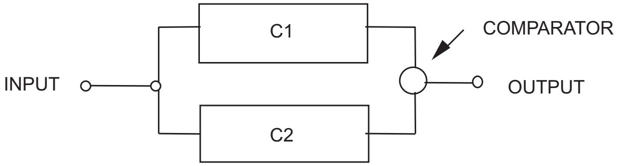
5.2 Standard Fault Diagnosis Techniques

Test/Checks

- **Standard checks**
  - Replication checks
  - Timing checks
  - Reversal checks
  - Coding checks
  - Reasonableness checks
  - Structural checks
  - Diagnostic checks
  - Algorithmic checks
- Categories are not orthogonal ➡ for some methods classification is not always clear

Replication Checks

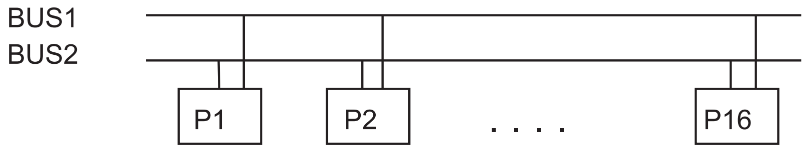
- Everything is done multiple times
- Efficient but expensive
- Variants:
  - Identical replication
  - Different designs
  - Repeated execution
  - Comparison with standard execution ➡ **diagnosis checks**



Timing Checks

- Tests execution against timing constraints
- Variants
  - Additional unit for monitoring of timing (watchdog)
  - Passive mutual check
  - Active mutual check

Example Tandem: “I’am alive” each second, “Are you okay?” every two seconds



Reversal Checks

- Usually, outputs depend deterministically on inputs
- Calculating inputs based on outputs
- Comparison with input
- Examples:
  - Read after write
  - Mathematical functions such as
    - $(\sqrt{x})^2 \stackrel{!}{=} x$
    - $\mathbf{A} \cdot \mathbf{A}^{-1} \stackrel{!}{=} \mathbf{I}$

Coding Checks

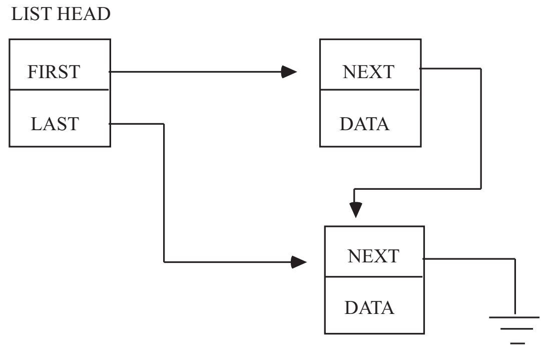
- Redundant representation of data
- Examples:
  - Parity bit ➡ Odd or even
  - Berger code ➡ Counts 1 or 0
  - Checksum ➡ Adding data elements of a block
  - Hamming code ➡ increases Hamming distance
  - Cyclic Redundancy Check ➡ Based on remainder theorems for residue arithmetic
- More in section on memory....

Reasonableness Checks

- Using common sense
- Using knowledge regarding internal design and structures
- Examples:
  - Range checks (e.g., $0° \leq \alpha < 360°$, index of array within defined range)
  - Consistency checks (e.g., aircraft on ground vs. state of wheels)
  - Type checks (e.g., is result integer?)

Structural Checks

- Checks consistent structure of data or system structure
- Examples:
  - Number of elements
  - Redundant pointers



  - List of Plug-and-Play-devices

Diagnostic Checks

- Test component using known input/output-pairs
- Typically used for hardware diagnosis (e.g., BIOS)
- Examples:
  - Memory tests (further discussed later)
  - Exception tests
  - Load tests
  - Environmental tests

Algorithmic Checks

- Checking invariants
- Examples:
  - Sorting: Number of entries, checksum, other properties
  - Checksum for matrix multiplication

$$\left[\begin{array}{cc} 1 & 2 \\ 3 & 5 \\ \hdashline 4 & 7 \end{array}\right] \times \left[\begin{array}{cc:c} 2 & 4 & 6 \\ 1 & 3 & 4 \end{array}\right] = \left[\begin{array}{cc:c} 4 & 10 & 14 \\ 11 & 27 & 38 \\ \hdashline 15 & 37 & 52 \end{array}\right]$$

5.3 Coding Checks: Main Memory

Motivation

- Memory appears in computers in large amounts
- A single bit fault may lead to a system failure
- Becomes worse with increased capacity and decreased structure size
- ➡ Efficient tests needed

- **Typical approaches:** codes and check sums (that are codes, too)
- Two application principle: offline (production, startup) or online

Design Criterions

- **Coverage**
  - Total coverage
  - Coverage with respect to a given type of error
- **Overhead**
  - Hardware (additional circuitry, additional memory)
  - Software
  - Runtime (time overhead for encoding and decoding)
  - # Checkbits
- **Application case**
  - Detection
  - Localization
  - Correction

Example: Two-dimensional Parity

	n bits/word					Row Parity Register	Parity Error?
k words	0	0	1	1	1	0	No
	1	0	0	0	1	1	No
	1	1	0	1	0	1	Yes
	0	1	1	0	1	0	No
Column Parity Register	1	1	0	1	0	0	Overall parity check bit
Parity Error?	No	No	Yes	No	No		

- Detects all 1-bit faults, all 2-bit and 3-bit faults within k words, and many more
- Localizes all 1-bit faults within k words

BERGER Code

- Number of 1 is added to data
- k information bits need $\lfloor \log_2 k + 1 \rfloor$ check bits
- 100% coverage for single errors
- Coverage calculation tricky for double and other errors
- Low overhead

HAMMING Codes

- Provide error detection and error correction
- Example: $(n,k)=(7,4)$
  7 bits including 3 check bits
- **Hamming distance** between two code words:
  Number of different corresponding bit positions
  $H_d(1001001, 1100101)=3$
- **Hamming distance of a code:** minimal Hamming distance of two (different) code words
- A distance H_d allows to detect H_d-1 bit faults
  or
  to correct $\lfloor \frac{H_d-1}{2} \rfloor$ bit faults
- Typically: $H_d=3$

Construction of HAMMING Codes

Construction by R.W. HAMMING, 1950

- Each j. position, $j=2^{i-1}$ with $i=1,\ldots,k$ is a **check bit** (parity bit) c_i.
- The remaining bits are data bits d_l with $l=1,\ldots,m$
  - **Example** for $(7,4)$: $d_4,d_3,d_2,c_3,d_1,c_2,c_1=h_7,h_6,h_5,h_4,h_3,h_2,h_1$
- Each check bit forms parity over a number of bits
- **Calculation rule:** $c_j=h_{2^{j-1}}$ is used for all data bits with $(i \mod 2^j)\geq 2^{j-1}$ (i relates to h_i-number)
  - **Example** for $(7,4)$:
    - Parity of h_1,h_3,h_5,h_7 ➡ $c_1=d_1\oplus d_2\oplus d_4$
    - Parity of h_2,h_3,h_6,h_7 ➡ $c_2=d_1\oplus d_3\oplus d_4$
    - Parity of h_4,h_5,h_6,h_7 ➡ $c_3=d_2\oplus d_3\oplus d_4$

Example: (7,4)-HAMMING Code

- Example for $(7,4)$-Hamming Code
- There are no two code words (third column) with a hamming distance less than 3

Value	Binary	Hamming
0	0000	0000000
1	0001	0000111
2	0010	0011001
3	0011	0011110
4	0100	0101010
5	0101	0101101
6	0110	0110011
7	0111	0110100
8	1000	1001011
9	1001	1001100
10	1010	1010010
11	1011	1010101
12	1100	1100001
13	1101	1100110
14	1110	1111000
15	1111	1111111

Generator Matrix

- HAMMING Code is a **linear** code
  ➡ can be calculated using a generator matrix with $h=d\cdot G$ (multiplication modulo 2)
- Example for (7,4) code: $G=\begin{pmatrix}1&0&0&1&0&1&1\\0&1&0&1&0&1&0\\0&0&1&1&0&0&1\\0&0&0&0&1&1&1\end{pmatrix}$
- **Example:** $d=(0011), h=d\cdot G=(0011110)$

Parity Matrix

- The **parity matrix** P describes parity calculation in form of a matrix
- Example for (7,4) code: $P = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- Each column of P is equal to one of the formulas of parity calculation
- Please note: $G \cdot P = 0$ (again: multiplication modulo 2)

Checking Hamming Codes

- Code word h is checked by multiplication modulo 2 with parity matrix
- The resulting vector S is called a *syndrom*
- $h \cdot P = S$
- If syndrom is zero vector no fault is present (with respect to fault model)
- In case of a single fault syndrom decodes bit position

Fault-free Case

- Example 1: No fault
  - $d = 1101, h = H(d) = 1100110$
  - $$S = h \cdot P$$
    $$= \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$$

Faulty Case

- Example 2: Fault
  - $d = 1101, h = H(d) = 11\underline{1}0110$
  - $$S = h \cdot P$$
    $$= \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$$
  - The syndrom equals $(1\ 0\ 1)$ meaning that the fault is at position 5 of h (counted from right)

Parity and Complement

- Parity is usually used for fault detection, but not localization
- Localization can be done with an algorithmic "trick"
- **Idea**: Writing the complement to same address to correct error
- **Fault model:** at most 1 bit stuck-at-X
- Example:

1^{st} write	1 1 0 1 0 0 1 1 0	original data
1^{st} read	1 1 0 1 0 1 1 1 0	parity error
$D \to \bar{D}$	0 0 1 0 1 0 0 0 1	data complement
2^{nd} write	0 0 1 0 1 0 0 0 1	complemented data
2^{nd} read	0 0 1 0 1 1 0 0 1	parity error
$D \to \bar{D}$	1 1 0 1 0 0 1 1 0	data complement (corrected data)

System-Level Diagnosis

The subject of *system-level diagnosis* or *system diagnosis* is to determine by mutual checks of execution units which unit is correct and which unit is faulty.

- Given a test (node A tests node B) that delivers "correct" or "faulty"
- **Goal**: Nodes recognized as faulty should no longer be used
- **Problem**: Wrong testing results are possible if testing node is faulty
- **Questions**
  - Is a solution possible (characterization)?
  - What is the solution?

t-Diagnosability and Syndrom

- A system is called **t-diagnosable** if for any distribution of up to t faults each of those faults can be recognized and located
- **Assumption**: There is an external observer who "collects" and evaluates the results of the tests
- The set of results is called **syndrom**
- A system is t-diagnosable if and only if there are **distinguishable** syndroms for any distribution of up to t faults.
- **Remark**: In systems using t for time other symbols are used, e.g., f-diagnosability

PMC Model

- PMC-Model by Preparata, Metze and Chien, 1967
- Assumptions regarding test results:

Testing unit	Tested unit	Test result
correct	correct	correct
correct	faulty	faulty
faulty	correct	undefined
faulty	faulty	undefined

- For simplification: test result "correct" is noted by 0 and "faulty" by 1

Diagnosability in the PMC Model

- Under the assumption that two nodes do not test each other mutually holds:

Theorem 5.1

A system is t-diagnosable if

- $n \geq 2t + 1$
- *Each node is tested by at least t other nodes*

Diagnosability in the PMC Model (cont.)

- Generic case:

Theorem 5.2

A system $G(V, E)$ is t-diagnosable if and only if

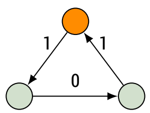
- $n \geq 2t + 1$
- $\forall v \in V : |\Gamma^{-1}(v)| \geq t$
- $\forall p \in \mathbb{N}, 0 \leq p < t, \forall X \subseteq V, |X| = n - 2t + p \rightarrow |\Gamma(X)| > p$

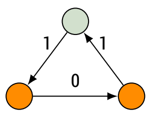
- $\Gamma(Z)$: Set of nodes tested by nodes from set Z
  $\Gamma^{-1}(Z)$: Set of nodes testing nodes of Z (tester of Z)

Proof of Theorem 5.1

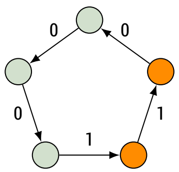
Necessity. To prove necessity it is enough to give a counter-example.

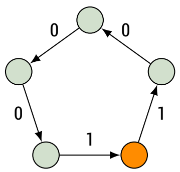
- Necessity of $n \geq 2t + 1$:





- Necessity of $|\Gamma^{-1}(v)| \geq t$:





Proof of Theorem 5.1 (cont.)

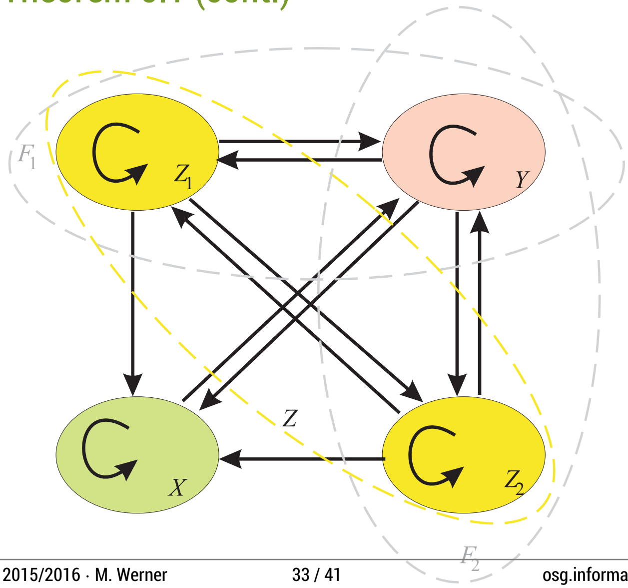
Sufficiency. Proof by contradiction – assume: $\mathcal{S}$ is a system with

$$n \geq 2t + 1 \tag{A.1}$$
$$|\Gamma^{-1}(v)| \geq t \tag{A.2}$$

- $\mathcal{S}$ is not t-diagnosable, $\Rightarrow$ There are two different fault sets F_1 und F_2 leading to the same syndrom S.

In order to make F_1 und F_2 indistinguishable, **no** element that is **only** in F_1 **or** in F_2 can be tested by an element of $V \backslash (F_1 \cup F_2)$ (always fault-free).

Proof of Theorem 5.1 (cont.)



Proof of Theorem 5.1 (cont.)

Notations:

- $Y = F_1 \cap F_2$ (Set of elements that are always faulty)
- $Z_1 = F_1 - Y$ (Set of elements faulty only in F_1)
- $Z_2 = F_2 - Y$ (Set of elements faulty only in F_2)
- X is the set of elements that are correct in both cases, $X = V - (F_1 \cup F_2)$

- Let $E(A, B)$ be the set of tupels (a, b) with $a \in A$ and $b \in B$ with "a tests b"
- $|E(A, B)|$ is the number of tests if members of A test members of B

Proof of Theorem 5.1 (cont.)

- How many tests are possible within a set A of elements if there is no mutual test?
- One element can test at maximum $|A| - 1$ others, the next $|A| - 2$, etc.

Lemma 5.3

If there is no mutual test of two elements of a set A:

$$|E(A, A)| \leq \frac{|A| \cdot (|A| - 1)}{2}$$

$$|E(A \cup B, A \cup B)| \leq |A| \cdot |B| \qquad \text{(lemma 5.3a)}$$

if $A \cap B = \emptyset$

Proof of Theorem 5.1 (cont.)

Consider tests of elements from Z_1 (according to (A.2)):

$$\begin{aligned} |Z_1| \cdot t \leq \left| \sum_{v \in Z_1} \Gamma^{-1}(v) \right| &\leq |E(Z_1, Z_1)| + |E(Z_2, Z_1)| + |E(Y, Z_1)| \\ &\leq |E(Z_1, Z_1)| + |E(Z_2, Z_1)| + |Y| \cdot |Z_1| \qquad (1) \end{aligned}$$

Analogical for tests of elements from Z_2:

$$\begin{aligned} |Z_2| \cdot t \leq \left| \sum_{v \in Z_2} \Gamma^{-1}(v) \right| &\leq |E(Z_2, Z_2)| + |E(Z_1, Z_2)| + |E(Y, Z_2)| \\ &\leq |E(Z_2, Z_2)| + |E(Z_1, Z_2)| + |Y| \cdot |Z_2| \qquad (2) \end{aligned}$$

Proof of Theorem 5.1 (cont.)

Adding (1)+(2):

$$(|Z_1| + |Z_2|) \cdot t \leq |E(Z_1, Z_1)| + |E(Z_2, Z_2)| + \\ + |Y| (|Z_1| + |Z_2|) + |E(Z_1, Z_2)| + |E(Z_2, Z_1)|$$

Considering lemma 5.3 and lemma 5.3a:

$$(|Z_1| + |Z_2|) \cdot t \leq \frac{1}{2} (|Z_1| (|Z_1| - 1) + |Z_2| (|Z_2| - 1)) + \\ + |Y| (|Z_1| + |Z_2|) + |Z_1| \, |Z_2|$$

$$2 \cdot t \leq |Z_1| + 2 \cdot |Y| + |Z_2| - 1$$

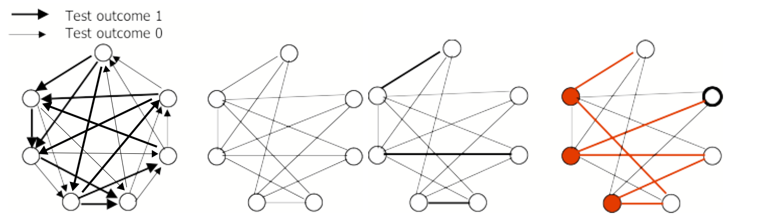
$$2 \cdot t \leq \underbrace{|Z_1| + |Y|}_{=|F_1| \leq t} + \underbrace{|Z_2| + |Y|}_{=|F_2| \leq t} - 1$$

➡ **Contradiction to (A.1)** □

Example for a Diagnosis Algorithm

Algorithm by Sullivan (modified)

1. Creating the L-graph (disagreement-graph)
   An edge between v_1 and v_2 exists if the assumption "v_1 is correct" leads directly or indirectly to "v_2 is faulty"
2. Find a **maximal matching** in the L-graph
3. Assign the state "correct" to a node that is **not** in the matching and make a diagnosis of the system from that node



BGM Model

- BGM model by Barsi, Grandoni and Maestrini, 1976
- Assumptions regarding test results:

Testing unit	Tested unit	Test result
correct	correct	correct
correct	faulty	faulty
faulty	correct	undefined
faulty	faulty	faulty

Diagnosability in the BGM Model

Necessary condition for t-diagnosability in the BGM model

Theorem 5.4

A system $\mathcal{S}$ is t-diagnosable in the BGM model. Then it holds:

$$n \geq t + 2$$

There is also a general (necessary and sufficient) condition for t-diagnosability in the BGM model.

Problem Variations

- There are a number of further variation of the system-level diagnosis, e.g.:
  - **Sequential diagnosis:** SIngle elements will be detected as faulty and replaced; then the diagnosis continues
  - **Alternative diagnosis models**
  - **Set diagnosis:** Determine a set X of elements, $|X| > f$, where X contains all faulty elements
  - **Result propagation:** How to collect test outcomes, if the result have to be transmitted by participating nodes?