Introduction to Software Testing

Chapter 3
Syntactic Logic Coverage Criteria - DNF

Paul Ammann & Jeff Offutt

Updated by Sunae Shin

DNF Criteria

- Revisit the testing of Boolean expressions
- Syntactic Logic Coverage Criteria
- Structure of the predicate as expressed in a disjunctive normal form (DNF) representation

Disjunctive Normal Form

- **Common Representation for Boolean Functions**
  - **Slightly Different Notation for Operators**
  - **Slightly Different Terminology**
- **Basics:**
  - **A *literal* is a clause or the negation (overstrike) of a clause**
    - **Examples: a, $\overline{a}$**
  - **A *term* is a set of literals connected by logical "and"**
    - **"and" is denoted by adjacency instead of $\wedge$**
    - **Examples: ab, $a\overline{b}$, $\overline{a}\overline{b}$ for $a \wedge b$, $a \wedge \neg b$, $\neg a \wedge \neg b$**
  - **A *(disjunctive normal form) predicate* is a set of terms connected by "or"**
    - **"or" is denoted by $+$ instead of $\vee$**
    - **Examples: $abc + \overline{a}b + a\overline{c}$**
    - **Terms are also called "implicants"**
      - **If a term is true, that implies the predicate is true**

Implicant Coverage

- **Obvious coverage idea: Make each implicant evaluate to "true".**
  - **Problem: Only tests "true" cases for the predicate.**
  - **Solution: Include DNF representations for negation.**

<u>Implicant Coverage (IC)</u> : Given DNF representations of a predicate f and its negation $\bar{f}$, for each implicant in f and $\bar{f}$, TR contains the requirement that the implicant evaluate to true.

- **Example:** $f = ab + b\bar{c}$ $\quad \bar{f} = \bar{b} + \bar{a}c$
  - **Implicants:** $\{ ab, b\bar{c}, \bar{b}, \bar{a}c \}$
  - **Possible test set: {TTF, FFT}**
- **Observation: IC is relatively weak**

Improving on Implicant Coverage

- **Additional Definitions:**
  - A *proper subterm* is a term with one or more clauses removed
    - Example: *abc* has 6 proper subterms: *a, b, c, ab, ac, bc*
  - A *prime implicant* is an implicant such that no proper subterm is also an implicant.
    - Example: $f = ab + a\bar{b}c$
    - Implicant *ab* is a prime implicant
    - Implicant $a\bar{b}c$ is not a prime implicant (due to proper subterm *ac*)
  - A *redundant implicant* is an implicant that can be removed without changing the value of the predicate
    - Example: $f = ab + ac + b\bar{c}$
    - *ab* is redundant
    - Predicate can be written: $ac + b\bar{c}$

Unique True Points

- A ***minimal DNF representation*** **is one with only prime, nonredundant implicants**.
- A ***unique true point*** **with respect to a given implicant is an assignment of truth values** so that
  - the given implicant is true, and
  - all other implicants are false
- Hence a unique true point test focuses on just one implicant
- A minimal representation guarantees the existence of at least one unique true point for each implicant

<u>Unique True Point Coverage (UTPC)</u> : Given minimal DNF representations of a predicate *f* and its negation $\bar{f}$, TR contains a unique true point for each implicant in *f* and $\bar{f}$.

Unique True Point Example

- **Consider again:** $f = ab + b\bar{c}$ $\quad \bar{f} = \bar{b} + \bar{a}c$
  - **Implicants:** $\{ ab, b\bar{c}, \bar{b}, \bar{a}c \}$
  - **Each of these implicants is prime**
  - **None of these implicants is redundant**
- **Unique true points:**
  - ab: **{TTT}**
  - $b\bar{c}$: **{FTF}**
  - $\bar{b}$: **{FFF, TFF, TFT}**
  - $\bar{a}c$: **{FTT}**
- **UTPC is fairly powerful**
  - **Exponential in general, but reasonable cost for many common functions**
  - **No subsumption relation wrt any of the ACC or ICC Criteria**

Near False Points

- A *near false point* with respect to a clause *c* in implicant *i* is an assignment of truth values such that *f* is false, but if *c* is negated (and all other clauses left as is), *i* (and hence *f*) evaluates to true.
- Relation to *determination*: at a near false point, *c* determines *f*
  - Hence we should expect relationship to ACC criteria

Unique True Point and Near False Point Pair Coverage (CUTPNFP) : Given a minimal DNF representation of a predicate *f*, for each clause *c* in each implicant *i*, TR contains a unique true point for *i* and a near false point for *c* such that the points differ only in the truth value of *c*.

- Note that definition only mentions *f*, and not $\bar{f}$.

CUTPNFP Example

- Consider *f = ab + cd*
  - For implicant *ab*, we have 3 unique true points: {TTFF, TTFT, TTTF}
    - For clause *a*, we can pair unique true point <u>T</u>TFF with near false point <u>F</u>TFF
    - For clause b, we can pair unique true point T<u>T</u>FF with near false point T<u>F</u>FF
  - For implicant *cd*, we have 3 unique true points: {FFTT, FTTT, TFTT}
    - For clause *c*, we can pair unique true point FF<u>T</u>T with near false point FF<u>F</u>T
    - For clause *d*, we can pair unique true point FFT<u>T</u> with near false point FFT<u>F</u>
- CUTPNFP set: {TTFF, FFTT, TFFF, FTFF, FFTF, FFFT}
  - First two tests are unique true points; others are near false points
- Rough number of tests required: # implicants * # literals

Karnaugh Map

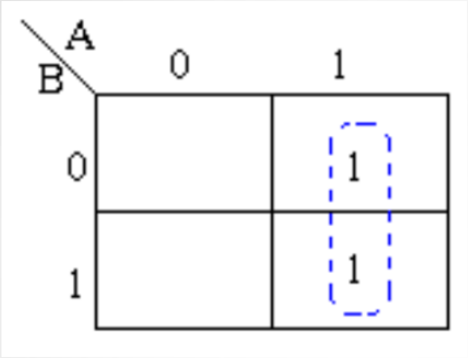
- **Karnaugh map**
  - Provides a simple and straight-forward method of minimizing Boolean expressions
  - A tabular representation of a predicate
    - Groupings of adjacent table entries correspond to simple DNF representation
  - Up to four and even six variables can be simplified

cd \ ab	00	01	11	10
00			1	
01			1	
11	1	1	1	1
10			1	

Karnaugh map table for the predicate "ab + cd"

Karnaugh Maps – Example 1

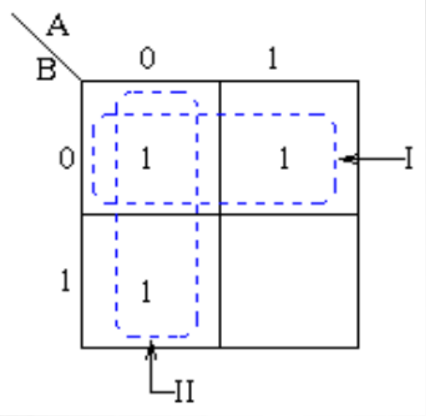
- Consider the following map
- The function plotted is: $Z = f(A, B) = A\overline{B} + AB$



- Referring to the map above, the two adjacent 1's are grouped together
- Through inspection it can be seen that variable B has its true and false form within the group
- This eliminates variable B leaving only variable A which only has its true form
- The minimized answer therefore is Z = A

Karnaugh Maps – Example 2

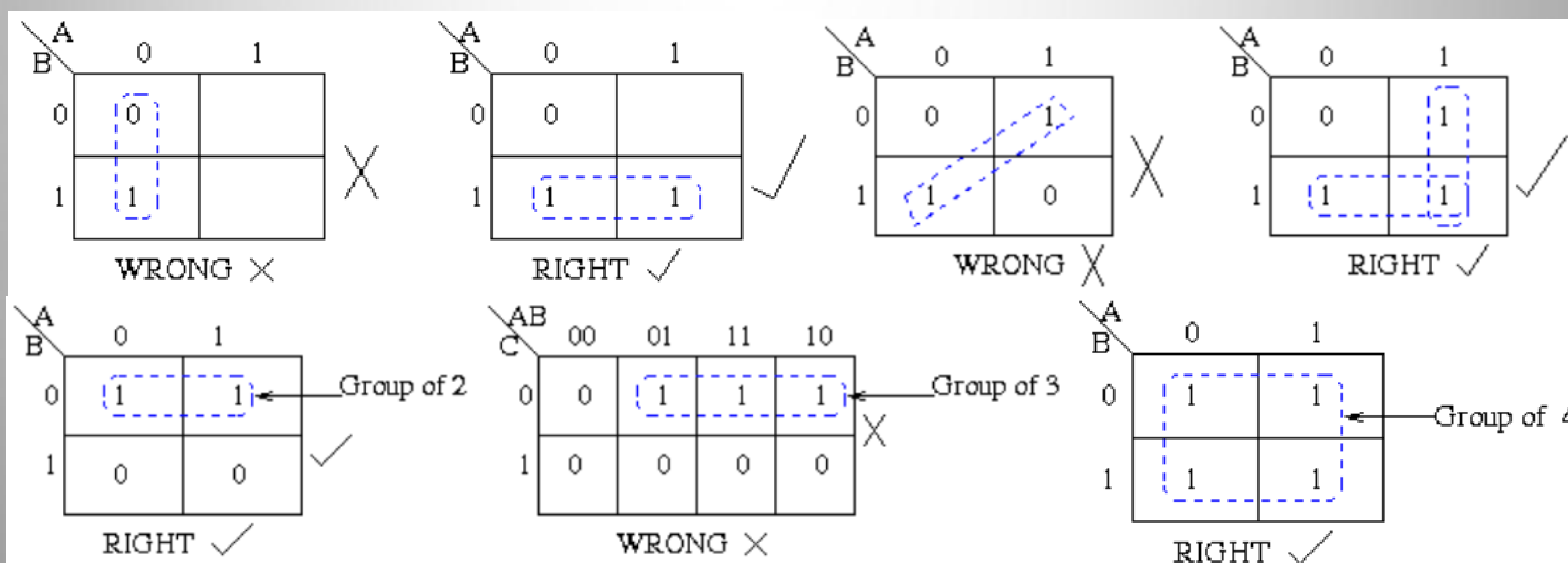
- **Consider the expression $Z = f(A, B) = \overline{A}\,\overline{B} + A\overline{B} + \overline{A}B$ plotted on the Karnaugh map:**



- **The first group**
  - **Correspond to the area of the map where B = 0 contains 1s, independent of the value of A**
  - **The expression of the output will contain the term $\overline{B}$**
- **The second group**
  - **Corresponds to the area of the map where A = 0**
  - **The group can therefore be defined as $\overline{A}$**
- **Hence the simplified answer is $Z = \overline{A} + \overline{B}$**

Karnaugh Maps – Rules of Simplication

- Groups may not include any cell containing a zero
- Groupd may be horizontal or vertical, but not diagonal
- Groups must contain 1, 2, 4, 8, or in general 2^n cells
- Each group should be as large as possible
- Groups may overlap



K-Map: Negation of a predicate

- **Consider the predicate: $f = ab + bc$**
- **Draw the Karnaugh Map for the negation**
  - **Identify groups**
  - **Write down negation: $\overline{f} = \overline{b} + \overline{a}\,\overline{c}$**

c \ ab	00	01	11	10
0			t	
1		t	t	

c \ ab	00	01	11	10
0	t	t		t
1	t			t