Tests Using Predicate Syntax

Dr. Mark C. Paulk

SE 4367 – Software Testing, Verification, Validation, and Quality Assurance

Jonsson School of Engineering and Computer Science

Topics: Software Testing

<u>Part II: Test Generation</u>

3. Domain Partitioning

4. Predicate Analysis
- **Domain Testing**
- **Cause-Effect Graphing**
- **Tests Using Predicate Syntax**
- **Tests Using Basis Paths**
- **Scenarios and Tests**

5. Test Generation from Finite State Models

6. Test Generation from Combinatorial Designs

Singular

A Boolean expression is singular if each variable in the expression occurs only once.

$E = e_1$ bop e_2 bop … bop e_k

e_i and e_j are mutually singular if they do not share any variable

- **e_i is a singular component of E iff e_i is singular and is mutually singular with every other component of E**
- **e_i is nonsingular iff it is nonsingular by itself and mutually singular with the remaining components of E**

DNF and CNF

A Boolean expression is in <u>disjunctive normal form</u> if it is represented as a sum of product terms

pq + ~rs

A Boolean expression is in <u>conjunctive normal form</u> if it is represented as a product of sums

(p + ~r)(p + s)(q + ~r)(q + s)

Any Boolean expression in CNF can be converted to an equivalent DNF and vice versa

Predicate Testing

Targets three classes of faults

- **Boolean operator fault**
- **relational operator fault**
- **arithmetic expression fault**

Fault Causes

A Boolean operator fault is caused when
- an incorrect Boolean operator is used
- a negation is missing or placed incorrectly
- parentheses are incorrect
- an incorrect Boolean variable is used

A relational operator fault is caused when
- an incorrect relational operator is used

An arithmetic expression fault is caused when
- the value of an arithmetic expression is off by an amount equal to ε

Boolean Operator Faults

Correct predicate: $(a<b) \lor (c>d) \land e$

$(a<b) \land (c>d) \land e$	**Incorrect Boolean operator**
$(a<b) \lor !(c>d) \land e$	**Incorrect negation operator**
$(a<b) \land (c>d) \lor e$	**Incorrect Boolean operators**
$(a<b) \lor (e>d) \land c$	**Incorrect Boolean variable**

Relational Operator Faults

Correct predicate: $(a<b) \vee (c>d) \wedge e$

$(a=b) \vee (c>d) \wedge e$	**Incorrect relational operator**
$(a=b) \vee (c<d) \wedge e$	**Two relational operator faults**
$(a=b) \vee (c>d) \vee e$	**Incorrect relational and Boolean operators**

Arithmetic Expression Faults

Correct predicate: **Ec: e_1 relop1 e_2**
Incorrect predicate: **Ei: : e_3 relop2 e_4**

E_i has an off-by-ε fault if $|e_3 - e_4| = \varepsilon$ for any test case for which $e_1=e_2$

E_i has an off-by- ε * fault if $|e_3 - e_4| \leq \varepsilon$ for any test case for which $e_1=e_2$

E_i has an off-by- ε^+ fault if $|e_3 - e_4| > \varepsilon$ for any test case for which $e_1=e_2$

Arithmetic Expression Fault Example

Correct predicate: $E_c = a < b + c$, a and b integer
$\varepsilon = 1$

Three incorrect versions of E_i

$a < b$	**given c = 1, off-by-1 fault in E_i, $\|a - b\| = 1$ for a test case for which $a = b + c$, e.g., (2,1,1)**
$a < b + 1$	**given c = 2, off-by-1* fault in E_i, $\|a - (b+1)\| \geq 1$ for any test case for which $a = b + c$, e.g., (4,2,2)**
$a < b - 1$	**given c > 0, off-by-1+ fault in E_i, $\|a - (b-1)\| > 1$ for any test case for which $a = b + c$, e.g., (3,2,1)**

Goal of Predicate Testing

Given a correct predicate p_c, generate a test set T such that

- **there is at least one test case $t \in$ T for which**
- **p_c and**
- **its faulty version p_i**
- **evaluate to different truth values**

Predicate Testing Example

Suppose that p_c: $a < b + c$ and p_i: $a > b + c$

Consider test set $T = \{t_1, t_2\}$ where

- **t_1: <a=0, b=0, c=0>**
- **t_2: <a=0, b=1, c=1>**

The fault in p_i is not revealed by t_1 as both p_c and p_i evaluate to false when evaluated against t_1

The fault is revealed by t_2 since

- **p_c evaluates to true**
- **p_i evaluates to false**

when evaluated against t_2

Predicate Constraints

Let BR denote {t, f, <, =, >, +ε, -ε}

BR stands for Boolean and relational

Any element of the BR set is a BR-symbol

- **specifies a constraint on a Boolean variable or relational expression**

+ε is a constraint on E': $e_1 < e_2$

- **$0 < e_2 - e_1 \leq \varepsilon$**

-ε is a constraint on E': $e_1 < e_2$

- **$-\varepsilon \leq e_1 - e_2 < 0$**

Constraints

BR symbols t and f specify constraints on Boolean variables and expressions

Constraints on relational expressions use <, =, and >

t and f can also be use to specify constraints on a simple relational expression

- **p_r: a < b**

Mathur, Example 4.8

E: a < c + d

Constraint C: (=) on E

Satisfying C requires at least one test case such that a = c + d

<a=1, c=0, d=1>

1 = 0 + 1

E: $a < c + d$

Constraint C: $(+\varepsilon)$ on E

Let $\varepsilon = 1$

Satisfying C requires at least one test case such that $0 < a - (c + d) \leq 1$

$<a=4, c=2, d=1>$

$0 < 4 - (2 + 1) \leq 1$

E: b

E is a Boolean expression

Constraint C: (t) on E

Given a Boolean expression E:b, the constraint "t" is satisfied by a test case that sets variable b to true.

Predicate Constraints

Let p_r denote a predicate with $n > 0$ $\vee$ and $\wedge$ operators.

A <u>predicate constraint</u> C for predicate p_r is a sequence of (n+1) BR symbols

- **one for each Boolean variable or relational expression in p_r**

Note that n ($\vee$ and $\wedge$) operators implies n+1 components in the predicate.

Mathur, Example 4.9

Given predicate p_r: $b \wedge r < s \vee u \geq v$

p_r: $b \wedge (r < s) \vee (u \geq v)$

One possible BR-constraint for p_r is C: (t, =, >)

A test case that satisfies C for p_r is
<b=true, r=1, s=1, u=1, v=0)

A test case that does not satisfy C for p_r is
<b=true, r=1, s=1, u=1, v=2)

Test Cases for Constraints

Test case t satisfies constraint C for predicate p_r if each component of p_r satisfies the corresponding constraint in C when evaluated against t.

Constraint C for predicate p_r guides the development of a test for p_r

- **offers hints on what the values of the variables should be for p_r to satisfy C**

Infeasible Constraints

A constraint C is considered <u>infeasible</u> for predicate p_r if there exists no input values for the variables in p_r that satisfy c.

For example, the constraint (>, >) is infeasible for the predicate $a > b \wedge b > d$ if it is known that $d > a$

True and False Constraints

$p_r(C)$ denotes the value of predicate p_r evaluated using a test case that satisfies C

C is referred to as

- **a true constraint when $p_r(C)$ is true**
- **a false constraint otherwise**

A set of constraints S is partitioned into subsets S^t and S^f, respectively, such that

- **for each C in S^t, $p_r(C)$ = true**
- **for any C in S^f, $p_r(C)$ = false**
- **$S = S^t \cup S^f$**

Mathur, Example 4.10

p_r: $(a < b) \land (c > d)$

C_1: $(=, >)$

- if $a = b$, then $a < b$ is false
- any test case that satisfies C_1 on p_r makes p_r false → C_1 is a false constraint

C_2: $(<, +\varepsilon)$ for $\varepsilon = 1$

- $0 < c - d \leq 1$ implies $c > d$
- any test case that satisfies C_2 on p_r makes p_r true → C_2 is a true constraint

$S = \{C_1, C_2\}$ $S^t = \{C_2\}$ $S^f = \{C_1\}$

Predicate Testing Criteria

Given a predicate p_r, we want to generate a test set T such that

- **T is minimal**
- **T guarantees the detection of any fault in the implementation of p_r**
  - **faults correspond to the fault model discussed earlier**

BOR = Boolean operator

BRO = Boolean and relational operator

BRE = Boolean and relational expression

BOR Testing Criterion

A test set T that satisfies the BOR-testing criterion for a compound predicate p_r guarantees the detection of single or multiple Boolean operator faults in the implementation of p_r.

T is referred to as a <u>BOR-adequate test set</u>

- **T_{BOR}**

BRO Testing Criterion

A test set T that satisfies the BRO-testing criterion for a compound predicate p_r guarantees the detection of single or multiple

- **Boolean operator**
- **relational operator faults**

in the implementation of p_r.

T is referred to as a <u>BRO-adequate test set</u>

- **T_{BRO}**

BRE Testing Criterion

A test set T that satisfies the BRE-testing criterion for a compound predicate p_r guarantees the detection of single or multiple

- **Boolean operator**
- **relational expression**
- **arithmetic expression faults**

in the implementation of p_r.

T is referred to as a <u>BRE-adequate test set</u>

- **T_{BRE}**

Guaranteeing Fault Detection

Let T_x, $x \in$ {BOR, BRO, BRE}, be a test set derived from predicate p_r.

Let p_f be another predicate obtained from p_r by injecting single or multiple faults of one of three kinds:

- **Boolean operator fault**
- **relational operator fault**
- **arithmetic expression fault**

T_x is said to guarantee the detection of faults in p_f if for some $t \in T_x$, $p(t) \neq p_f(t)$.

Mathur, Example 4.11

$p_r = (a < b) \land (c > d)$

Constraint set S = {(t,t), (t,f), (f,t)}

$T_{BOR} = \{t_1, t_2, t_3\}$ is a BOR-adequate test set that satisfies S

t_1: <a=1, b=2, c=1, d=0 >

- **satisfies (t,t), i.e. a<b is true and c>d is also true**

t_2: <a=1, b=2, c=1, d=2 >

- **satisfies (t,f) since 1>2 is false**

t_3: <a=1, b=0, c=1, d=0 >

- **satisfies (f,t) since 1<0 is false**

T is BOR-adequate

Predicate	t_1	t_2	t_3
$a < b \wedge c > d$	true	false	false
Single Boolean operator fault			
$a < b \vee c > d$	true	true	true
$a < b \wedge c < d$	false	true	false
$a > b \wedge c > d$	false	false	true
Multiple Boolean operator faults			
$a < b \vee c < d$	true	true	false
$a > b \vee c > d$	true	false	true
$a > b \wedge c < d$	false	false	false
$a > b \vee c < d$	false	true	true

Cross Product

The <u>(cross) product</u> of two sets A and B is defined as

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$$

Mathur, Example 4.12

Let $A = \{t, =, >\}$ and $B = \{f, <\}$

$$A \times B = \{(t, f), (t, <), (=, f), (=, <), (>, f), (>, <)\}$$

Onto Product

The <u>onto product</u> of two sets A and B is defined as

$$A \otimes B = \{(u, v) \mid u \in A, v \in B\}$$

- **such that each element of A appears at least once as u and each element of B appears at least once as v**
- **$A \otimes B$ is a minimal set**

Mathur, Example 4.12

Let $A = \{t, =, >\}$ and $B = \{f, <\}$

$$A \otimes B = \{(t, f), (=,<), (>,<)\}$$

Other possibilities for $A \otimes B$

- $A \otimes B = \{(t, f), (=,<), (>,f)\}$
- $A \otimes B = \{(t, f), (=,f), (>,<)\}$
- $A \otimes B = \{(t, <), (=,f), (>,<)\}$
- $A \otimes B = \{(t, <), (=,f), (>,f)\}$

"An" Element

$\{t_1\}$ where $t_1 \in A$ means that t_1 is <u>an</u> element of A

Let A = {(<), (=), (>)}

$\{t_1\} = \{(<)\}$

Other possibilities for $\{t_1\}$:

- **$\{t_1\} = \{(=)\}$**
- **$\{t_1\} = \{(>)\}$**

Conventions That Ease Grading

There are legitimate alternatives for ONTO product and for {t_x} or {f_x} in this problem.

- **using one of the alternatives is legitimate**
- **if you went a different (legal) way, that's acceptable**

Conventions

- **order {(t), (f)}, {(<), (=), (>)}, {(-ε), (=), (+ ε)} in initial sets**
- **match corresponding ONTO terms until reaching the end of the shorter set; then continue matching with the last item in the shorter set**
- **pick the first item for a {t_x} or {f_x}**

A Minimal BOR-Constraint Set BOR-CSET 1

Input

- **an abstract syntax tree for predicate p_r: AST(p_r)**
- **p_r contains only singular expressions**

Output

- **BOR-constraint set for p_r attached to the root node of AST(p_r)**

Procedure BOR-CSET

Step 1. Label each leaf node N of AST(p_r) with its constraint set S_N. For each S_N = {t, f}.

BOR-CSET 2

Step 2. Visit each non-leaf node in AST(p_r) in a bottom-up manner.

- **Let N_1 and N_2 denote the direct descendants of node N, if N is an AND-node or OR-node.**
- **If N is a NOT-node, then N_1 is its direct descendant.**
- **S_{N1} and S_{N2} are the BOR-constraint sets for nodes N_1 and N_2 respectively.**
- **For each non-leaf node N, compute S_N as follows.**

BOR-CSET 2.1

N is an OR-node.

$$S_N^t = (S_{N1}^t \times \{f_2\}) \cup (\{f_1\} \times S_{N2}^t)$$
$$\text{where } f_1 \in S_{N1}^f \text{ and } f_2 \in S_{N2}^f$$

$$S_N^f = S_{N1}^f \otimes S_{N2}^f$$

BOR-CSET 2.2

N is an AND-node.

$S_N^t = S_{N1}^t \otimes S_{N2}^t$

$S_N^f = (S_{N1}^f \times \{t_2\}) \cup (\{t_1\} \times S_{N2}^f)$
where $t_1 \in S_{N1}^t$ and $t_2 \in S_{N2}^t$

BOR-CSET 2.3

N is a NOT-node.

$S_N^t = S_{N1}^f$

$S_N^f = S_{N1}^t$

BOR-CSET 3

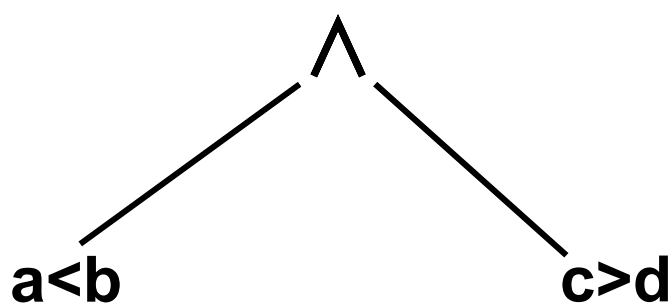
Step 3. The constraint set for the root of AST(p_r) is the desired BOR-constraint set for p_r.

End of procedure BOR-CSET

Mathur, Example 4.13

We want to generate T_{BOR} for: p_r: $a < b \wedge c > d$

Generate AST(p_r)



S_N is the constraint set for node N in AST(p_r).

S_N^t is the true constraint set for node N in AST(p_r).

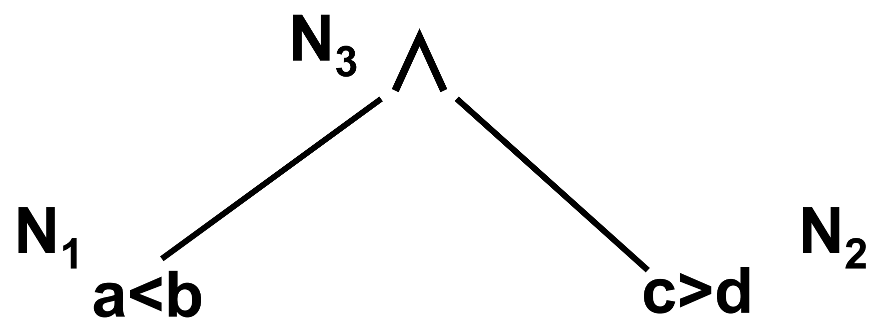
S_N^f is the false constraint set for node N in AST(p_r).

$$S_N = S_N^t \cup S_N^f$$

Label each leaf node with the constraint set {(t), (f)}.

Label the nodes as N_1, N_2, and so on for convenience.

$S_{N1}{}^{t} = \{t\}$
$S_{N1}{}^{f} = \{f\}$



$S_{N2}{}^{t} = \{t\}$
$S_{N2}{}^{f} = \{f\}$

$S_{N1} = \{(t), (f)\}$ $S_{N2} = \{(t), (f)\}$

Note that N_1 and N_2 are direct descendants of N_3 which is an AND-node.

Compute the constraint set for the next higher node in the syntax tree, in this case N_3.
For an AND-node:

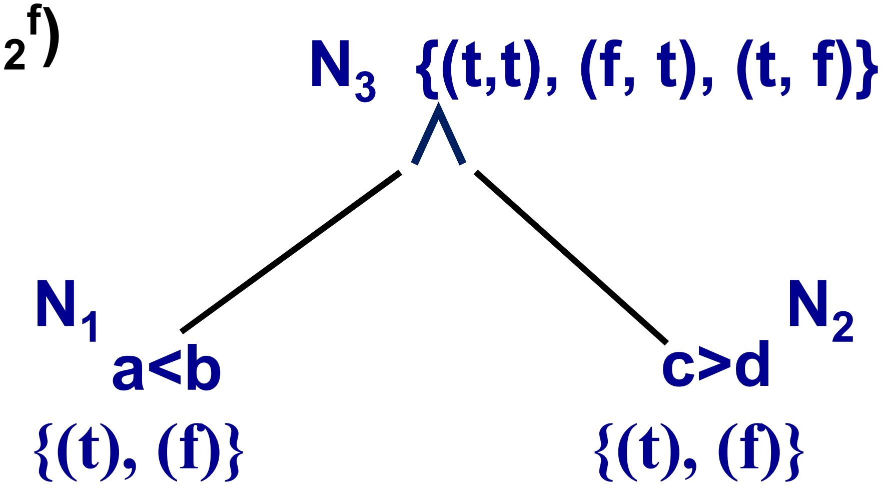
$$S_{N3}^{t} = S_{N1}^{t} \otimes S_{N2}^{t} = \{(t)\} \otimes \{(t)\} = \{(t, t)\}$$

$$S_{N3}^{f} = (S_{N1}^{f} \times \{t_2\}) \cup (\{t_1\} \times S_{N2}^{f})$$

$$= (\{(f)\} \times \{(t)\}) \cup (\{(t)\} \times \{(f)\})$$

$$= \{(f, t)\} \cup \{(t, f)\}$$

$$= \{(f, t), (t, f)\}$$



$S_{N3} = S_{N3}{}^t \cup S_{N3}{}^f = \{(t,t), (f,t), (t,f)\}$

S_{N3} contains a sequence of three constraints

- a minimal test set consists of three test cases

One possible test set:

$T_{BOR} = \{t_1, t_2, t_3\}$

$t_1 = \langle a=1, b=2, c=6, d=5 \rangle$ (t, t)

$t_2 = \langle a=1, b=0, c=6, d=5 \rangle$ (f, t)

$t_3 = \langle a=1, b=2, c=1, d=2 \rangle$ (t, f)

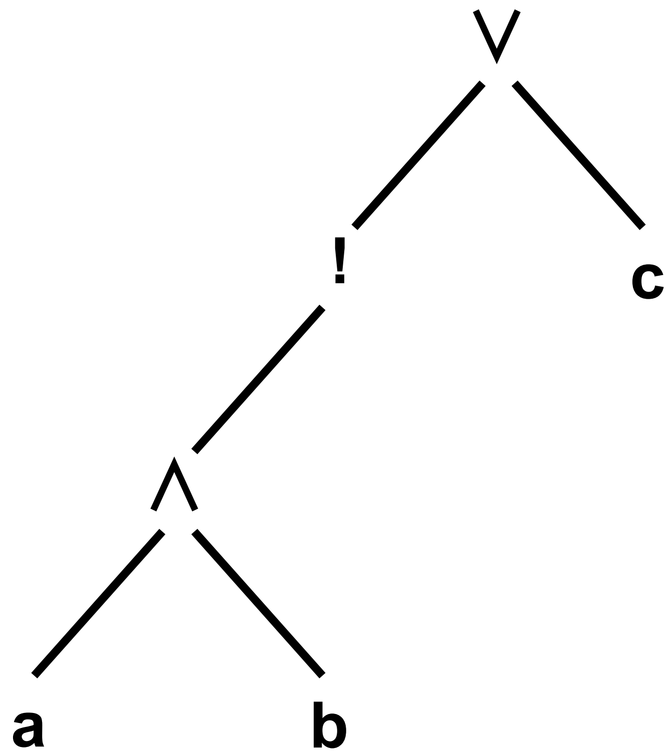
BOR Example #1

Generate a BOR-adequate test set T_{BOR} for

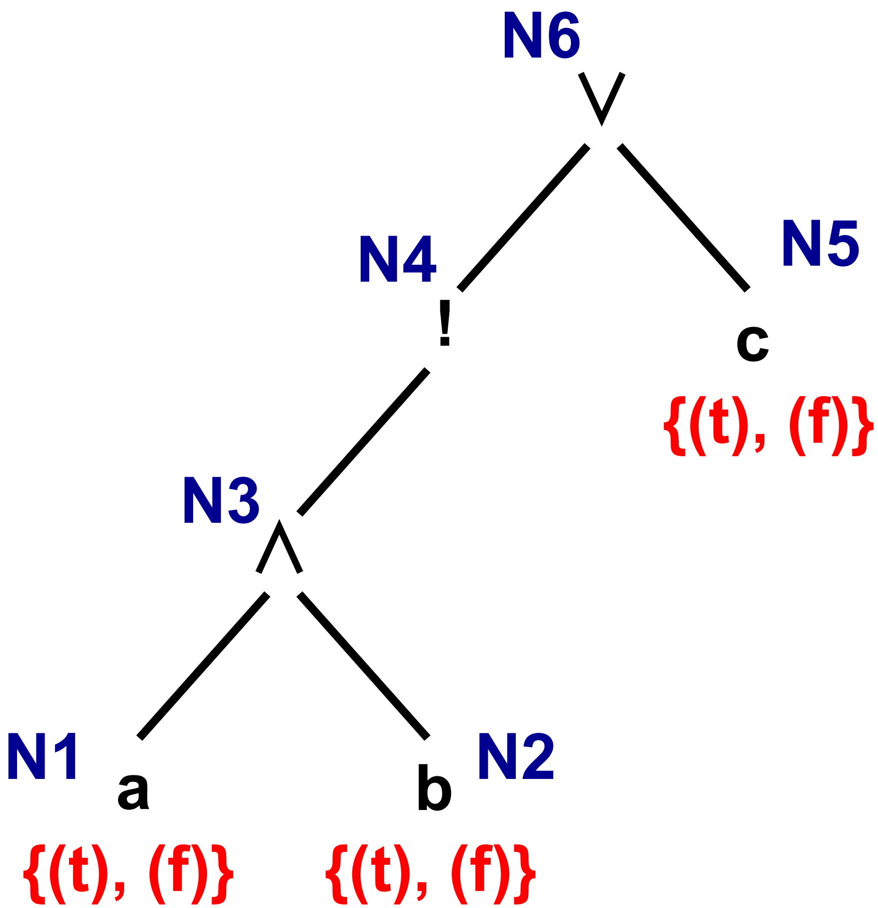
p_r: $!(a \wedge b) \vee c$

Show all the steps in generating T_{BOR}

AST for $!(a \wedge b) \vee c$



BOR-CSET Step 1



$$S_{N1}^{t} = S_{N2}^{t} = S_{N5}^{t} = \{t\}$$
$$S_{N1}^{f} = S_{N2}^{f} = S_{N5}^{f} = \{f\}$$

BOR-CSET Step 2.2 for N3

N3 is an AND-node

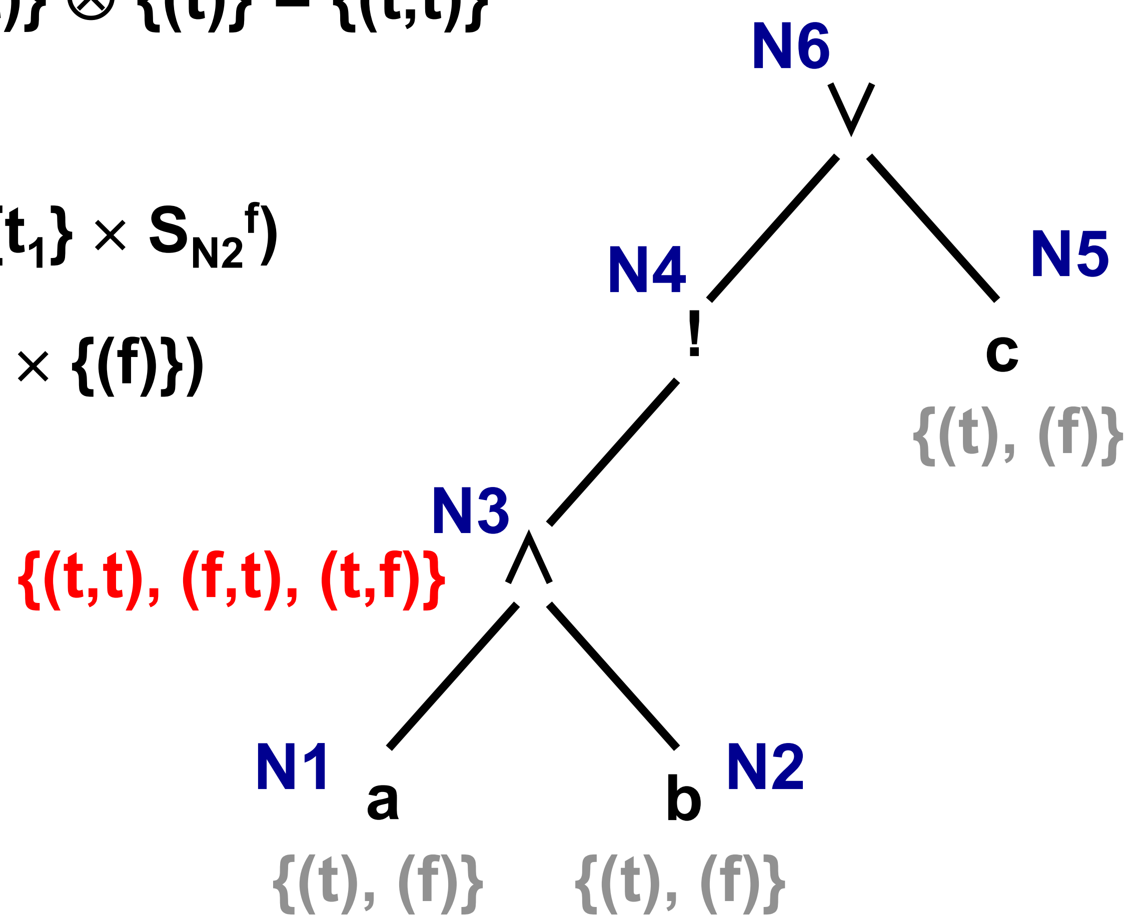
$S_{N3}^{t} = S_{N1}^{t} \otimes S_{N2}^{t} = \{(t)\} \otimes \{(t)\} = \{(t,t)\}$

$S_{N3}^{f} = (S_{N1}^{f} \times \{t_2\}) \cup (\{t_1\} \times S_{N2}^{f})$

$= (\{(f)\} \times \{(t)\}) \cup (\{(t)\} \times \{(f)\})$

$= \{(f,t)\} \cup \{(t,f)\}$

$= \{(f,t), (t,f)\}$

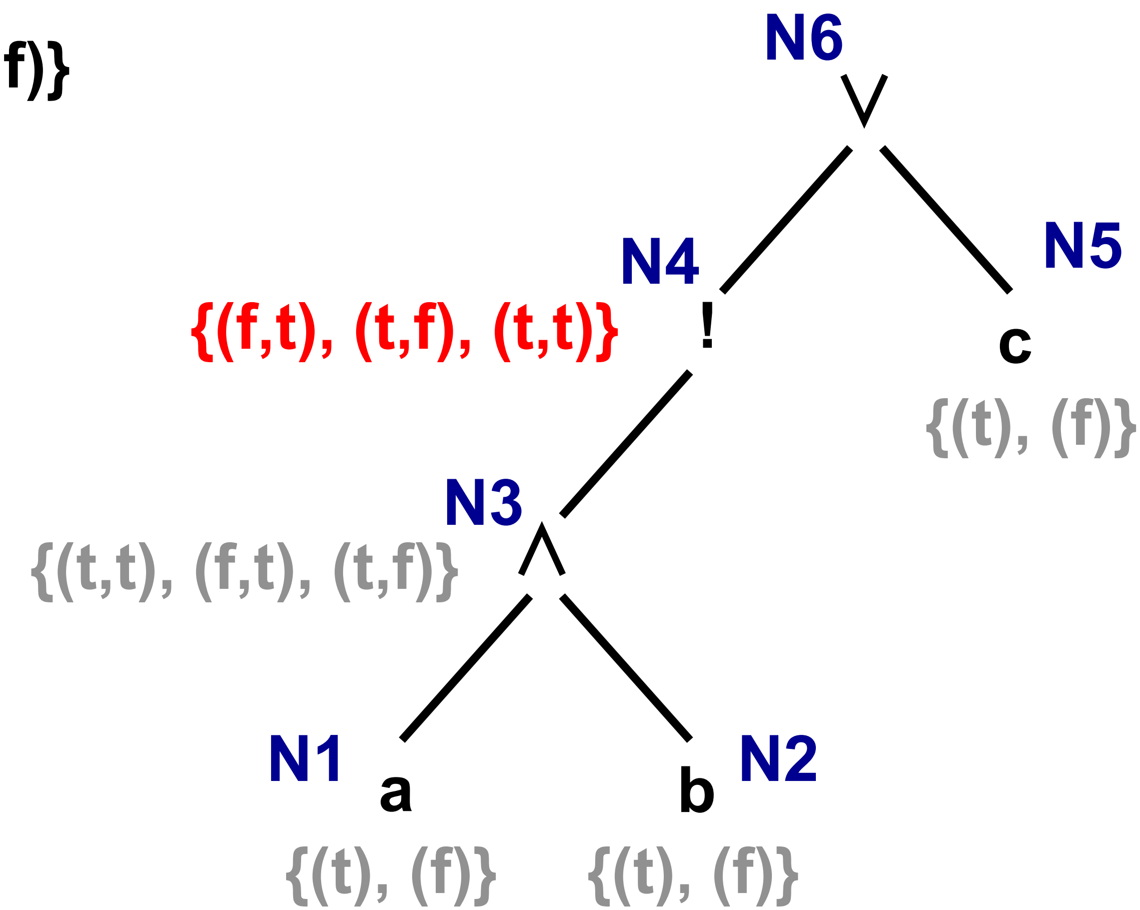


BOR-CSET Step 2.3 for N4

N4 is a NOT-node.

$S_{N4}^{t} = S_{N3}^{f} = \{(f,t), (t,f)\}$

$S_{N4}^{f} = S_{N3}^{t} = \{(t,t)\}$

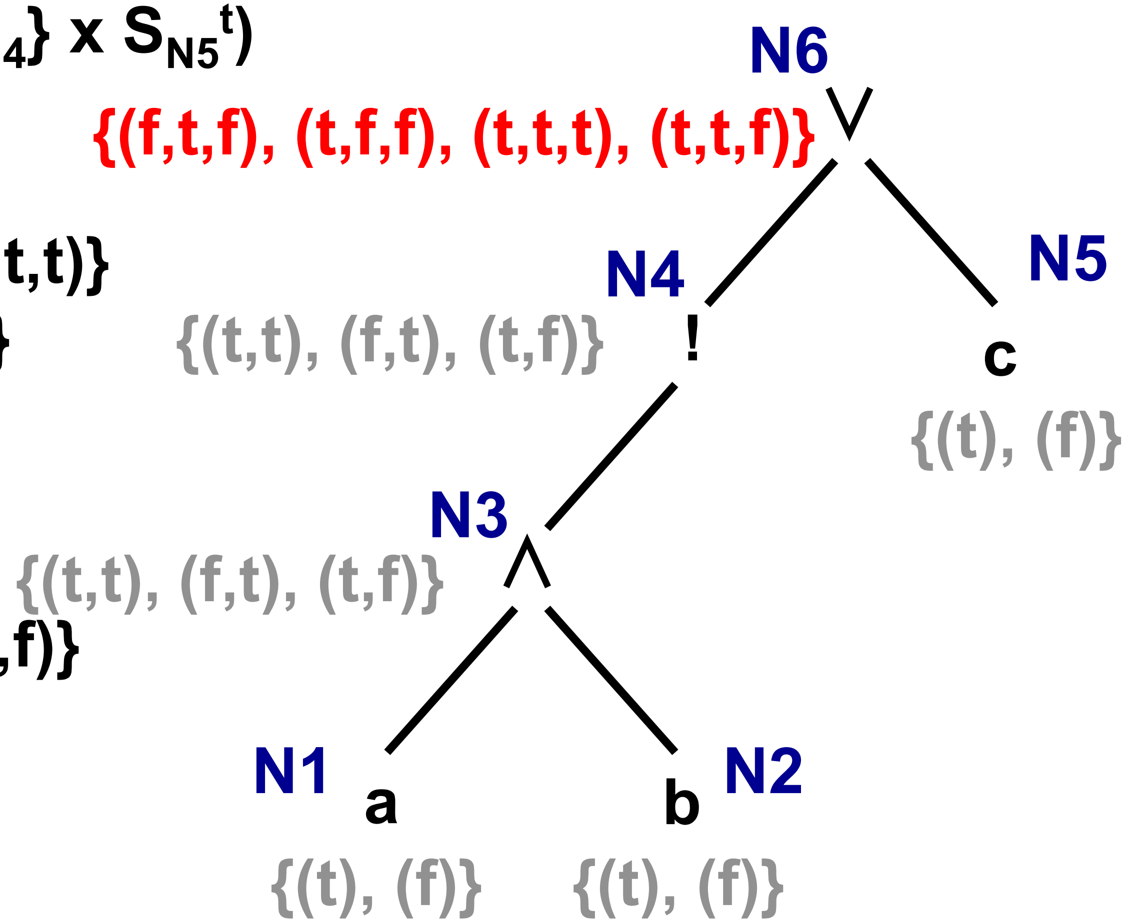


BOR-CSET Step 2.1 for N6

N6 is an OR-node.

$$\begin{aligned} S_{N6}^{t} &= (S_{N4}^{t} \times \{f_{N5}\}) \cup \{(f_{N4}\} \times S_{N5}^{t}) \\ &= (\{(f,t), (t,f)\} \times \{(f)\}) \\ &\quad \cup \ (\{(t,t)\} \times \{(t)\}) \\ &= \{(f,t,f), (t,f,f)\} \cup \{(t,t,t)\} \\ &= \{(f,t,f), (t,f,f), (t,t,t)\} \end{aligned}$$

$$\begin{aligned} S_{N6}^{f} &= S_{N4}^{f} \otimes S_{N5}^{f} \\ &= \{(t,t)\} \otimes \{(f)\} = \{(t,t,f)\} \end{aligned}$$



T_{BOR}

The test cases for the Boolean variables a, b, c are:

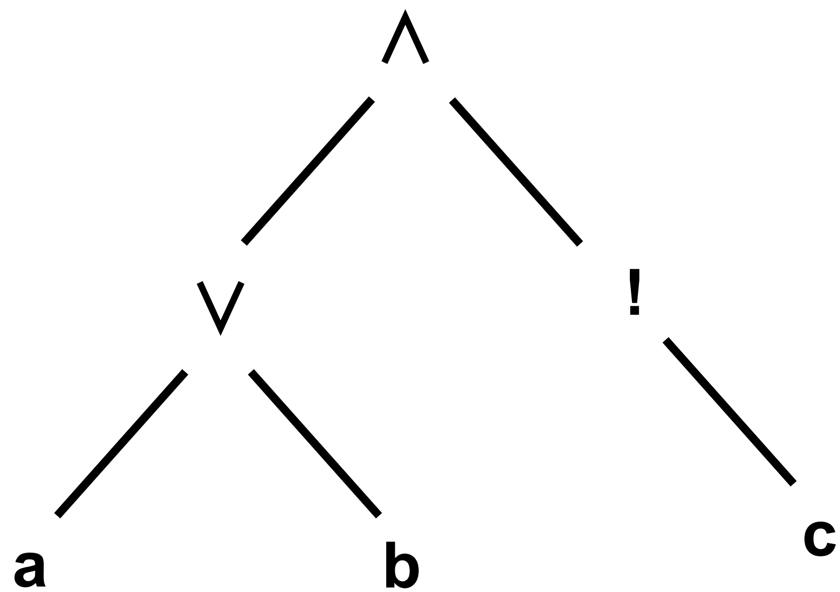
t_1: (true,true,false)
t_2: (false,true,false)
t_3: (true,false,false)
t_4: (true,true,true)

BOR Example #2

Generate a BOR-adequate test set T_{BOR} for
p_r: $(a \vee b) \wedge\ !c$

<u>Show all the steps</u> in generating T_{BOR}

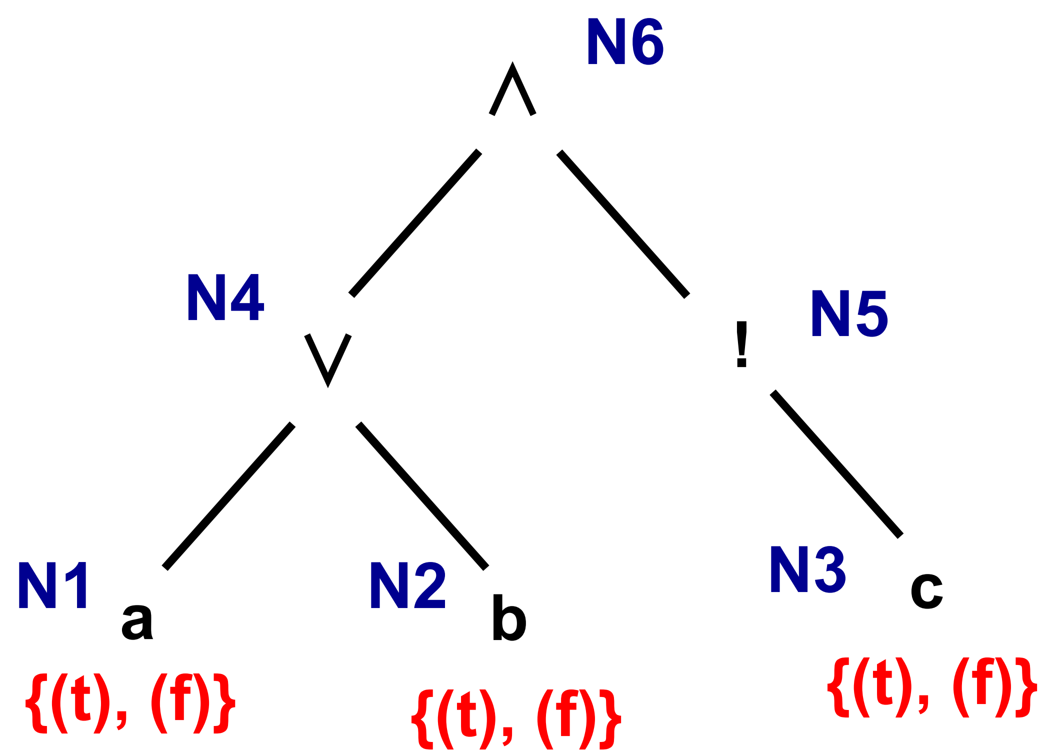
AST for $(a \vee b) \wedge !c$



BOR-CSET Step 1

$S_{N1}^{t} = S_{N2}^{t} = S_{N3}^{t} = \{(t)\}$

$S_{N1}^{f} = S_{N2}^{f} = S_{N3}^{f} = \{(f)\}$

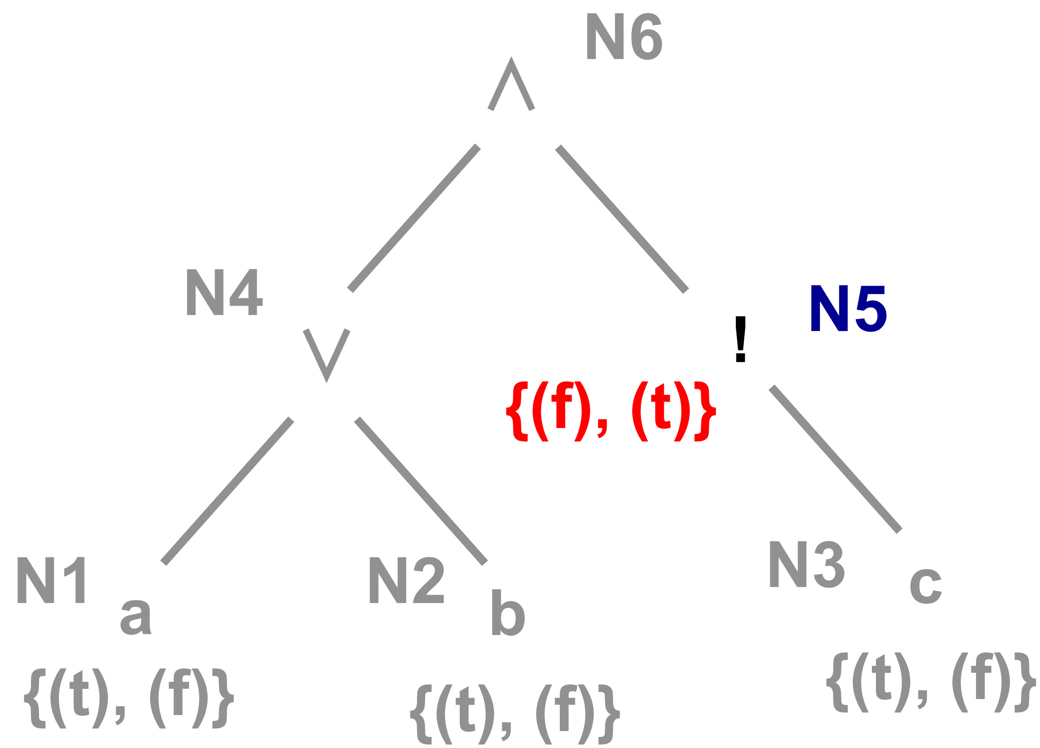


BOR-CSET Step 2.3 for N5

N5 is a NOT-node.

$S_{N5}^{t} = S_{N3}^{f} = \{(f)\}$

$S_{N5}^{f} = S_{N3}^{t} = \{(t)\}$

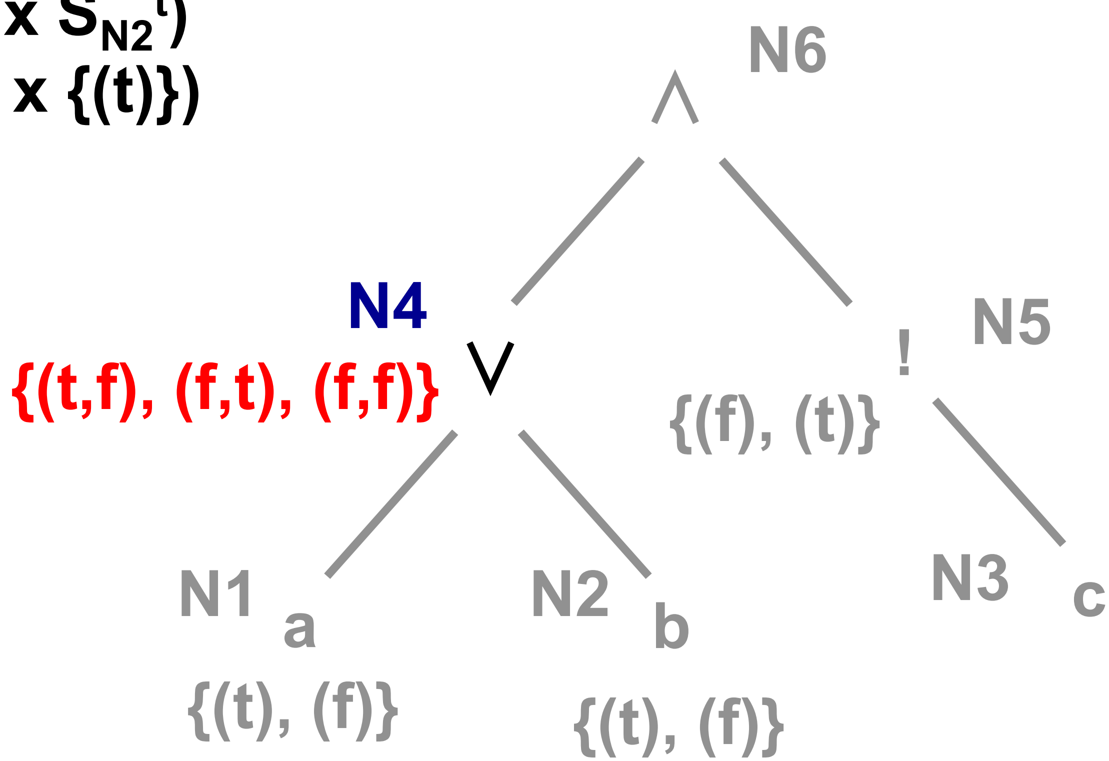


BOR-CSET Step 2.1 for N4

N4 is an OR-node.

$$
\begin{aligned}
S_{N4}^{t} &= (S_{N1}^{t} \times \{f_{N2}\}) \cup \{(f_{N1}\} \times S_{N2}^{t}) \\
&= (\{(t)\} \times \{(f)\}) \cup (\{(f)\} \times \{(t)\}) \\
&= \{(t,f)\} \cup \{(f,t)\} \\
&= \{(t,f), (f,t)\}
\end{aligned}
$$

$$
\begin{aligned}
S_{N4}^{f} &= S_{N1}^{f} \otimes S_{N2}^{f} \\
&= \{(f)\} \otimes \{(f)\} = \{(f,f)\}
\end{aligned}
$$



BOR-CSET Step 2.2 for N6

N6 is an AND-node

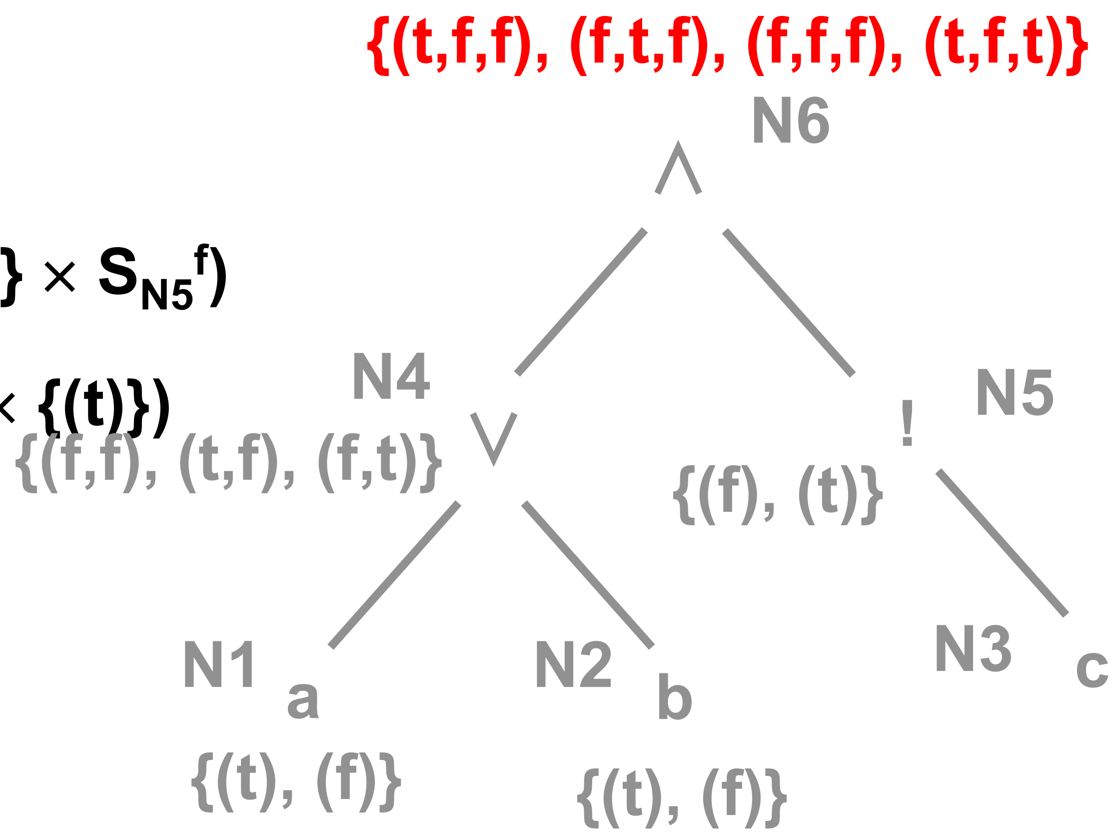
$$S_{N6}^{t} = S_{N4}^{t} \otimes S_{N5}^{t} = \{(t,f), (f,t)\} \otimes \{(f)\}$$
$$= \{(t,f,f), (f,t,f)\}$$

$$S_{N6}^{f} = (S_{N4}^{f} \times \{t_{N5}\}) \cup (\{t_{N4}\} \times S_{N5}^{f})$$

$$= (\{(f,f)\} \times \{(f)\}) \cup (\{(t,f)\} \times \{(t)\})$$

$$= \{(f,f,f)\} \cup \{(t,f,t)\}$$

$$= \{(f,f,f), (t,f,t)\}$$



T_{BOR} *for the Example*

The test cases for the Boolean variables a, b, c are:

t_1: (true,false,false)
t_2: (false,true,false)
t_3: (false,false,false)
t_4: (true,false,true)

Alternate Choice BOR-CSET Step 2.2 for N6

N6 is an AND-node

$S_{N6}^{t} = S_{N4}^{t} \otimes S_{N5}^{t} = \{(t,f), (f,t)\} \otimes \{(f)\}$
$= \{(t,f,f), (f,t,f)\}$

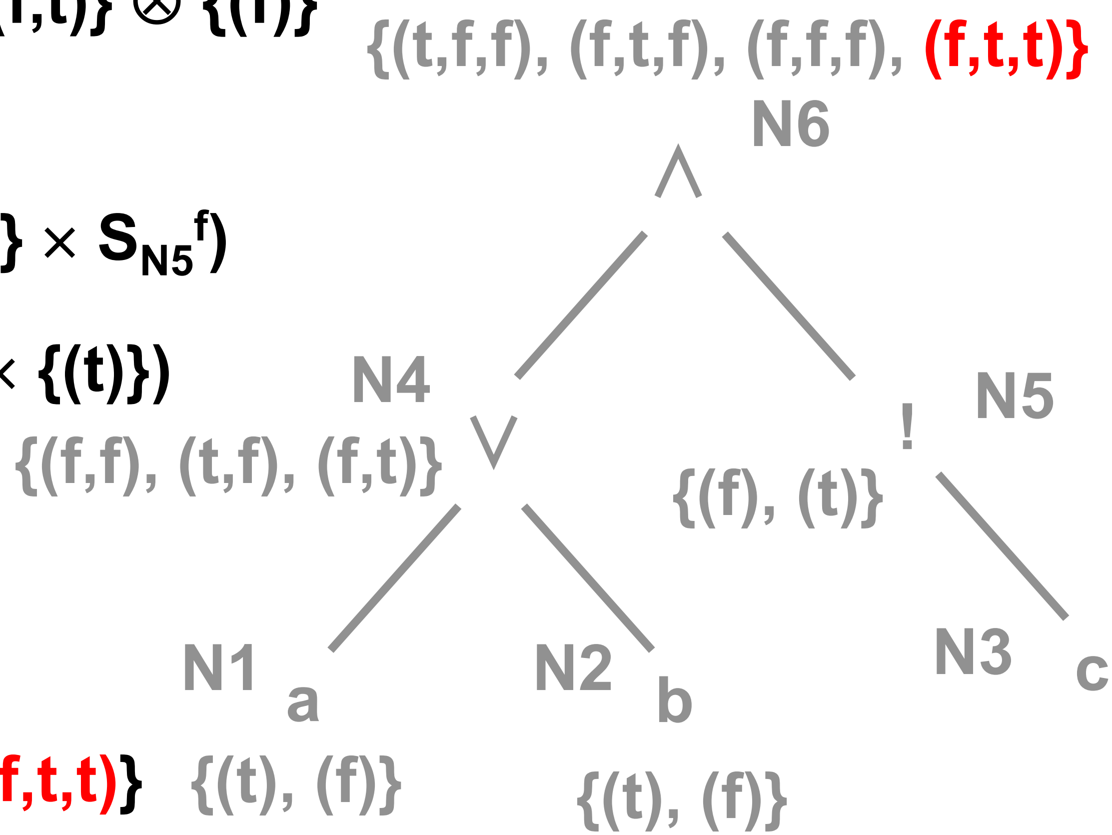
$S_{N6}^{f} = (S_{N4}^{f} \times \{t_{N5}\}) \cup (\{t_{N4}\} \times S_{N5}^{f})$

$= (\{(f,f)\} \times \{(f)\}) \cup (\{(f,t)\} \times \{(t)\})$

$= \{(f,f,f)\} \cup \{(f,t,t)\}$

$= \{(f,f,f), (f,t,t)\}$

$S_{N6} = \{(t,f,f), (f,t,f), (f,f,f), (f,t,t)\}$

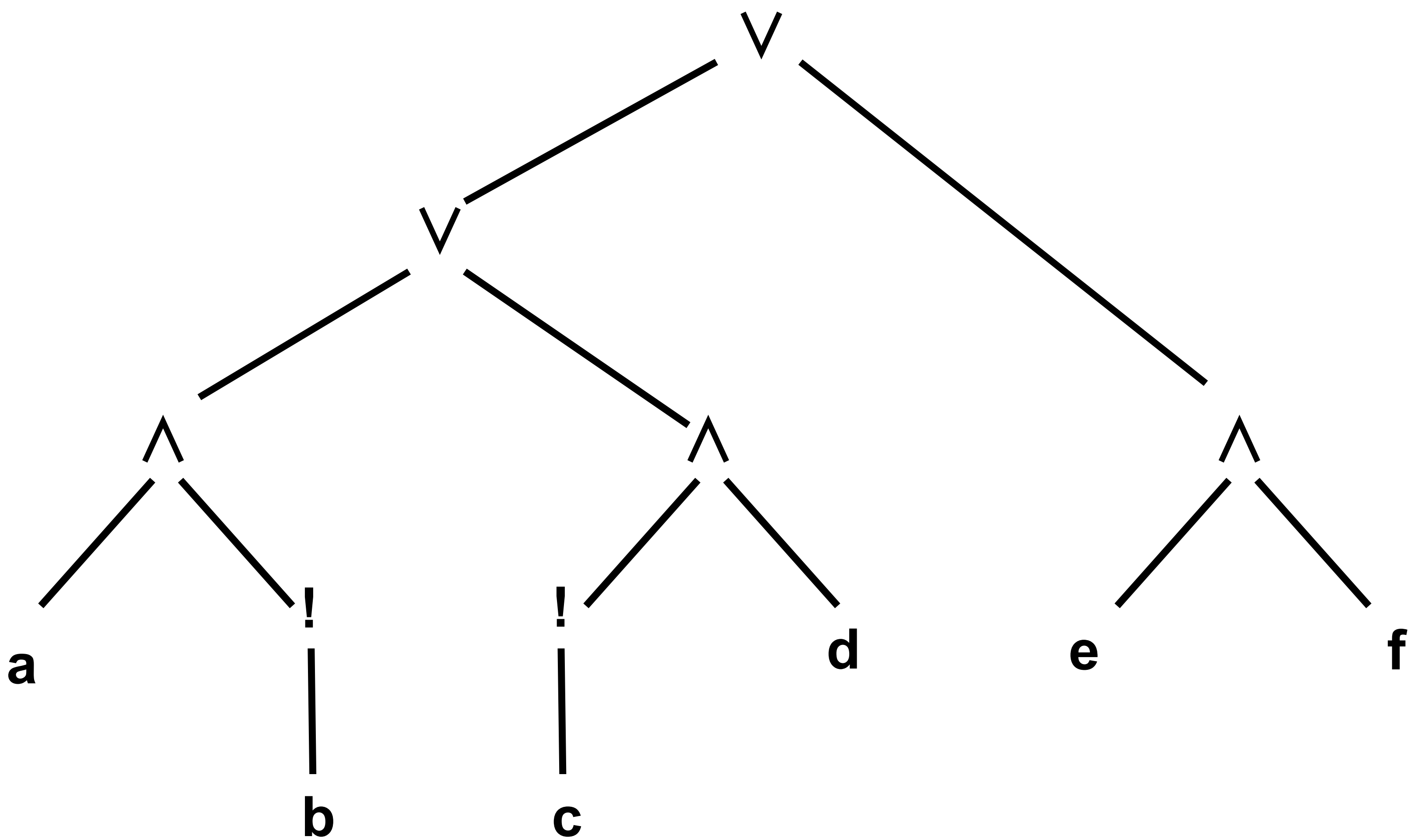


BOR Example #3

Generate a BOR-adequate test set T_{BOR} for
p_r: a!b + !cd + ef

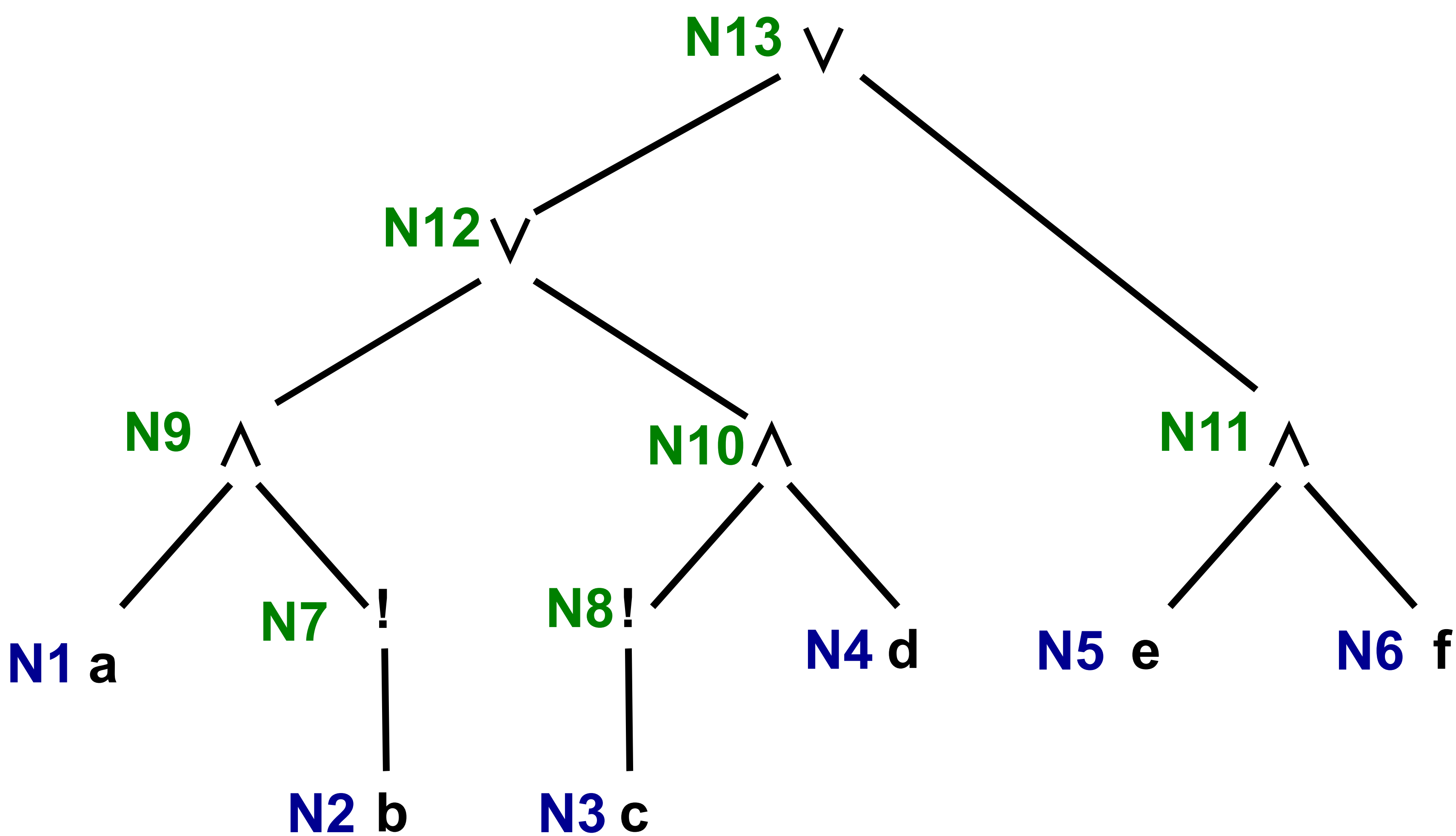
Show all the steps in generating T_{BOR}

AST for a!b + !cd + ef



BOR-CSET Step 1

Label AST (a!b + !cd + ef)



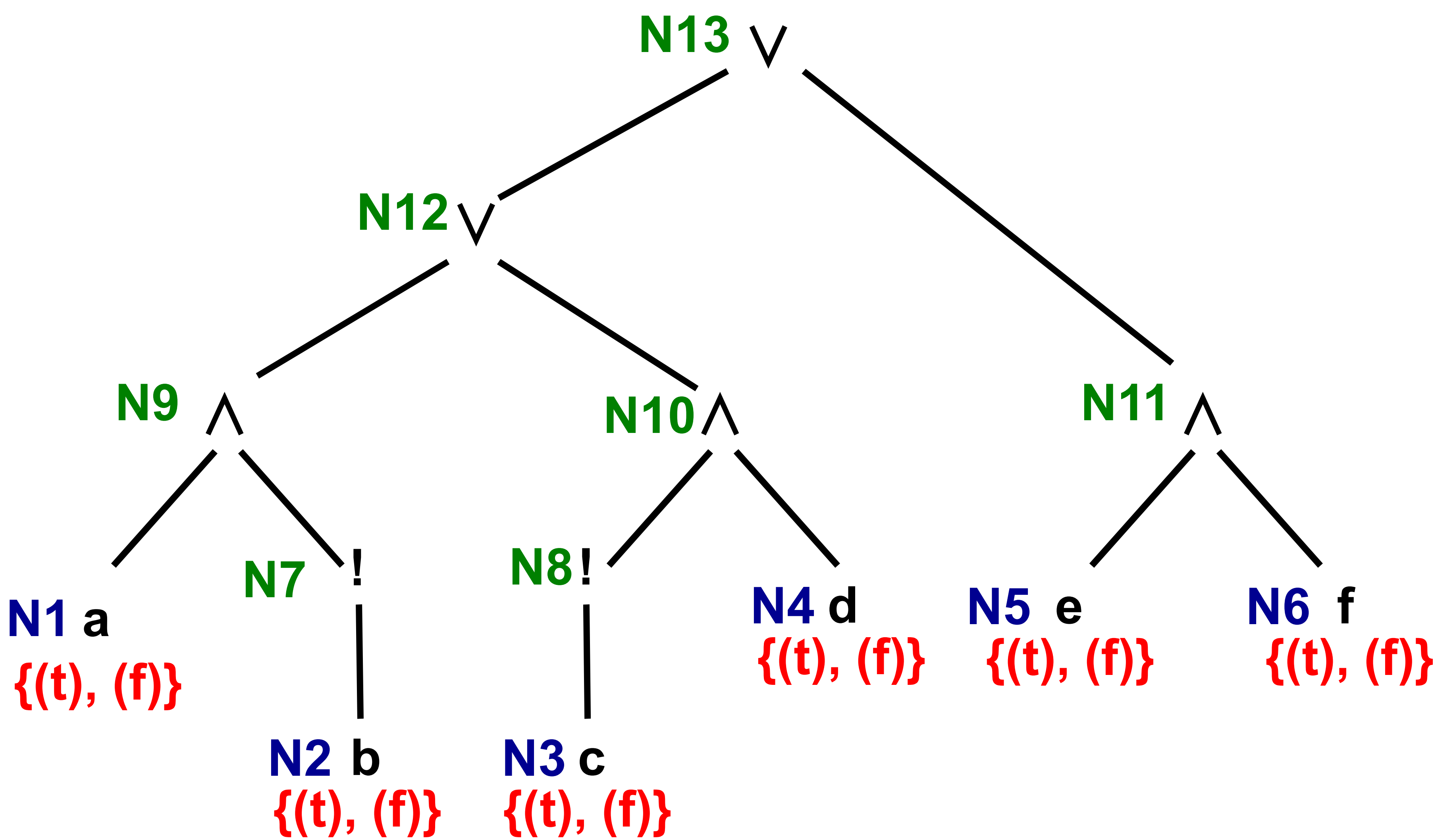
BOR-CSET Step 1
Label Leaf Nodes

$$S_{N1}^{t} = S_{N2}^{t} = S_{N3}^{t} = S_{N4}^{t} = S_{N5}^{t} = S_{N6}^{t} = \{(t)\}$$

$$S_{N1}^{f} = S_{N2}^{f} = S_{N3}^{f} = S_{N4}^{f} = S_{N5}^{f} = S_{N6}^{f} = \{(f)\}$$

BOR-CSET Step 1

Label AST (a!b + !cd + ef)



BOR-CSET Step 2.3 for NOT-Nodes

N7, N8 are NOT-nodes.

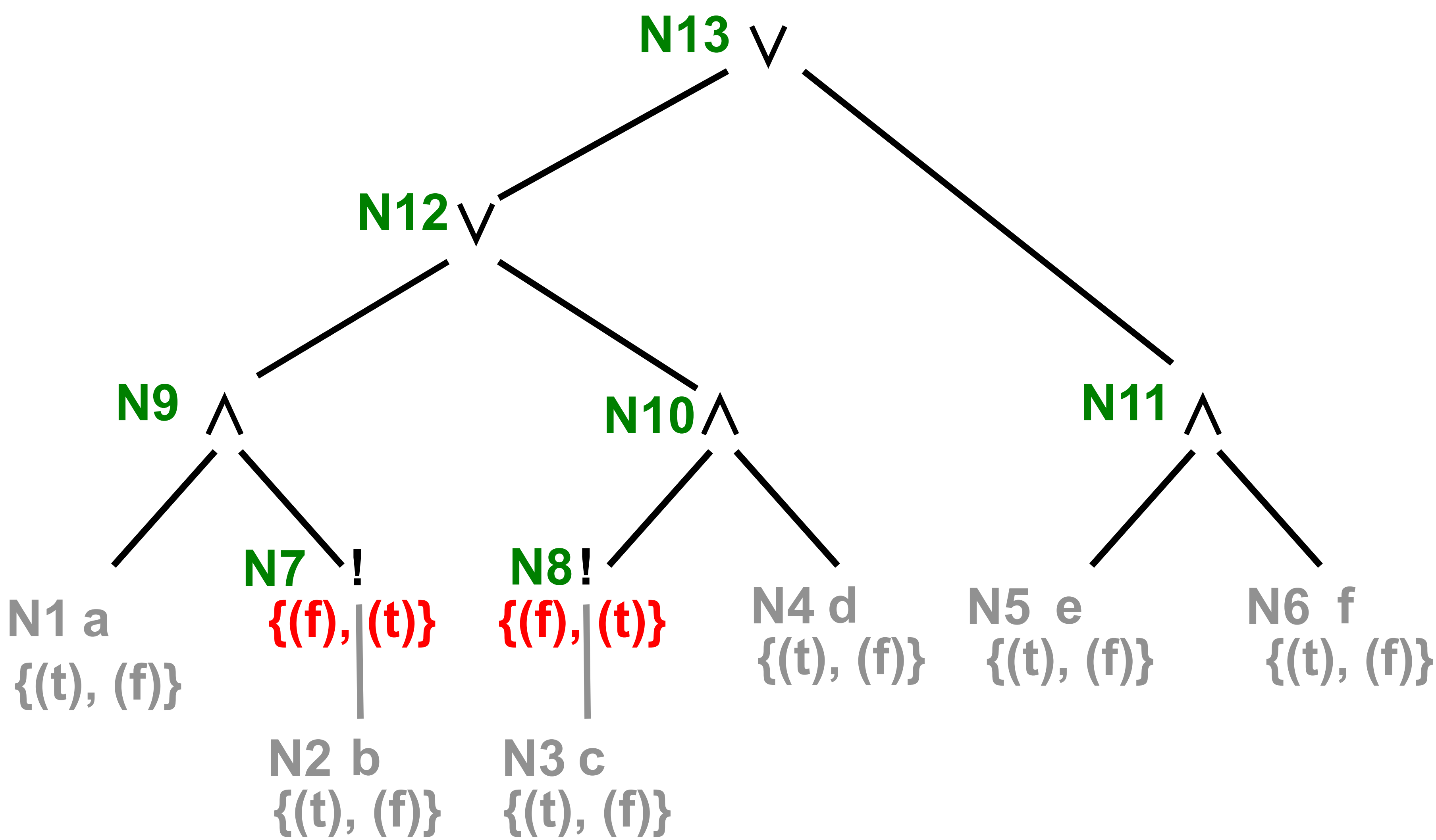
$S_{N7}^{t} = S_{N2}^{f} = \{(f)\}$

$S_{N7}^{f} = S_{N2}^{t} = \{(t)\}$

$S_{N8}^{t} = S_{N3}^{f} = \{(f)\}$

$S_{N8}^{f} = S_{N3}^{t} = \{(t)\}$

BOR-CSET Step 2.3
Label NOT-Nodes



BOR-CSET Step 2.2 for N_9

N_9 is an AND-node for f(a,b)

$$S_{N9}^{t} = S_{N1}^{t} \otimes S_{N7}^{t}$$
$$= \{(t)\} \otimes \{(f)\} = \{(t,f)\}$$

$$S_{N9}^{f} = (S_{N1}^{f} \times \{t_{N7}\}) \cup (\{t_{N1}\} \times S_{N7}^{f})$$

$$= (\{(f)\} \times \{(f)\}) \cup (\{(t)\} \times \{(t)\})$$

$$= \{(f,f)\} \cup \{(t,t)\}$$

$$= \{(f,f), (t,t)\}$$

BOR-CSET Step 2.2 for N_{10}

N_{10} is an AND-node for f(c,d)

$$S_{N10}^{t} = S_{N8}^{t} \otimes S_{N4}^{t}$$
$$= \{(f)\} \otimes \{(t)\} = \{(f,t)\}$$

$$S_{N10}^{f} = (S_{N8}^{f} \times \{t_{N4}\}) \cup (\{t_{N8}\} \times S_{N4}^{f})$$

$$= (\{(t)\} \times \{(t)\}) \cup (\{(f)\} \times \{(f)\})$$

$$= \{(t,t)\} \cup \{(f,f)\}$$

$$= \{(t,t), (f,f)\}$$

BOR-CSET Step 2.2 for N_{11}

N_{11} is an AND-node for f(e,f)

$$S_{N11}^{t} = S_{N5}^{t} \otimes S_{N6}^{t}$$
$$= \{(t)\} \otimes \{(t)\} = \{(t,t)\}$$

$$S_{N11}^{f} = (S_{N5}^{f} \times \{t_{N6}\}) \cup (\{t_{N5}\} \times S_{N6}^{f})$$

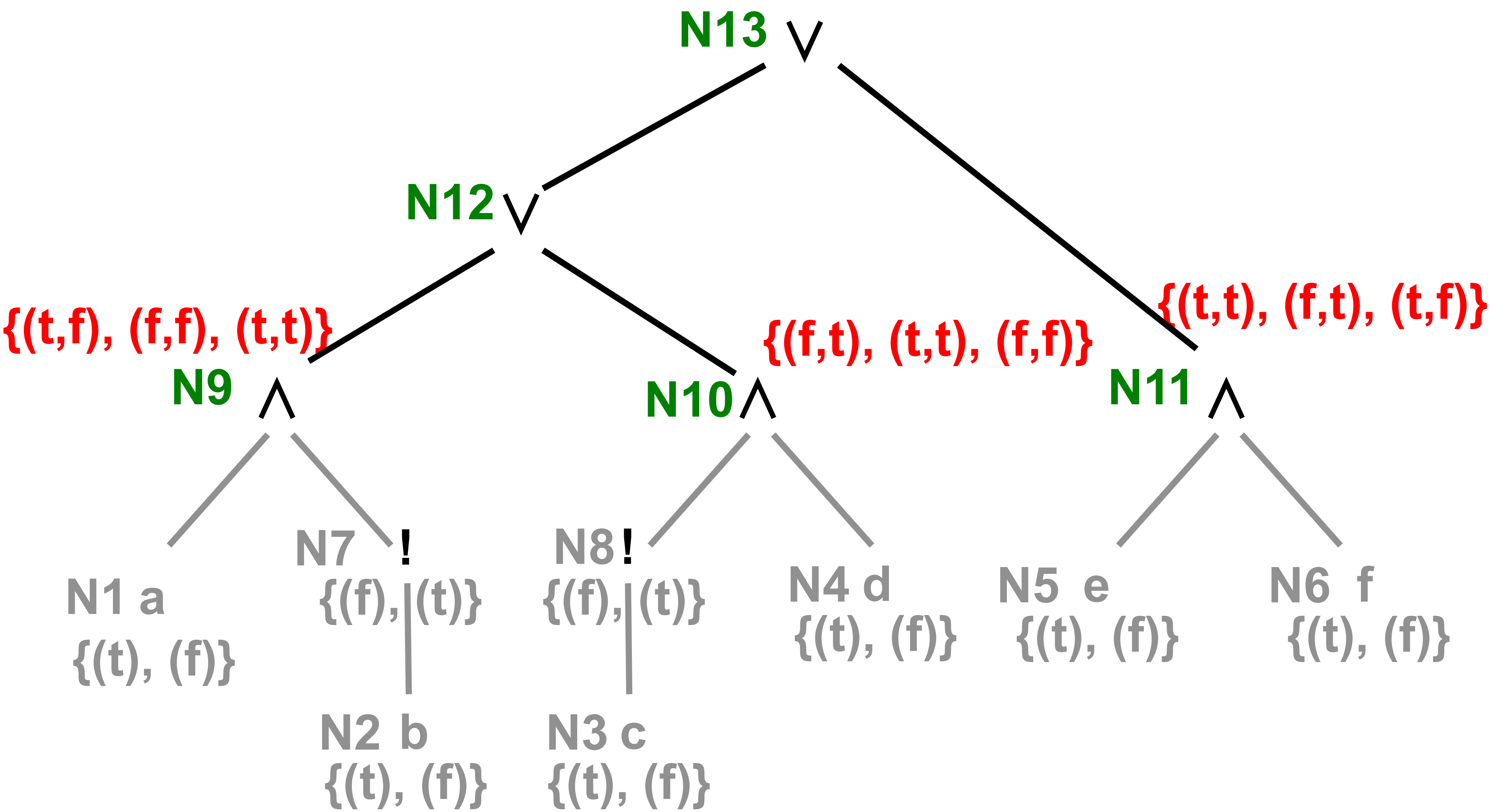
$$= (\{(f)\} \times \{(t)\}) \cup (\{(t)\} \times \{(f)\})$$

$$= \{(f,t)\} \cup \{(t,f)\}$$

$$= \{(f,t), (t,f)\}$$

BOR-CSET Step 2.2

Label AND-Nodes



BOR-CSET Step 2.1 for N_{12}

N_{12} is an OR-node for f(a,b,c,d).

$$
\begin{aligned}
S_{N12}^{t} &= (S_{N9}^{t} \times \{f_{N10}\}) \cup (\{f_{N9}\} \times S_{N10}^{t}) \\
&= (\{(t,f)\} \times \{(t,t)\}) \cup (\{(f,f)\} \times \{(f,t)\}) \\
&= \{(t,f,t,t)\} \cup \{(f,f,f,t)\} \\
&= \{(t,f,t,t), (f,f,f,t)\}
\end{aligned}
$$

$$
\begin{aligned}
S_{N12}^{f} &= S_{N9}^{f} \otimes S_{N10}^{f} \\
&= \{(f,f), (t,t)\} \otimes \{(t,t), (f,f)\} \\
&= \{(f,f,t,t), (t,t,f,f)\}
\end{aligned}
$$

$\{f_{N10}\}$ could also be $\{(f,f)\}$
$\{f_{N9}\}$ could also be $\{(t,t)\}$

ONTO product could be reversed

BOR-CSET Step 2.1 for N_{13}

N_{13} is an OR-node for f(a,b,c,d,e,f).

$$S_{N13}^{t} = (S_{N12}^{t} \times \{f_{N11}\}) \cup (\{f_{N12}\} \times S_{N11}^{t})$$
$$= (\{(t,f,t,t), (f,f,f,t)\} \times \{(f,t)\}) \cup (\{(f,f,t,t)\} \times \{(t,t)\})$$
$$= \{(t,f,t,t,f,t), (f,f,f,t,f,t)\} \cup \{(f,f,t,t,t,t)\}$$
$$= \{(t,f,t,t,f,t), (f,f,f,t,f,t), (f,f,t,t,t,t)\}$$

$$S_{N13}^{f} = S_{N12}^{f} \otimes S_{N11}^{f}$$
$$= \{(f,f,t,t), (t,t,f,f)\} \otimes \{(f,t), (t,f))\}$$
$$= \{(f,f,t,t,f,t), (t,t,f,f,t,f)\}$$

$$S_{N13} = \{(t,f,t,t,f,t), (f,f,f,t,f,t), (f,f,t,t,t,t), (f,f,t,t,f,t), (t,t,f,f,t,f)\}$$

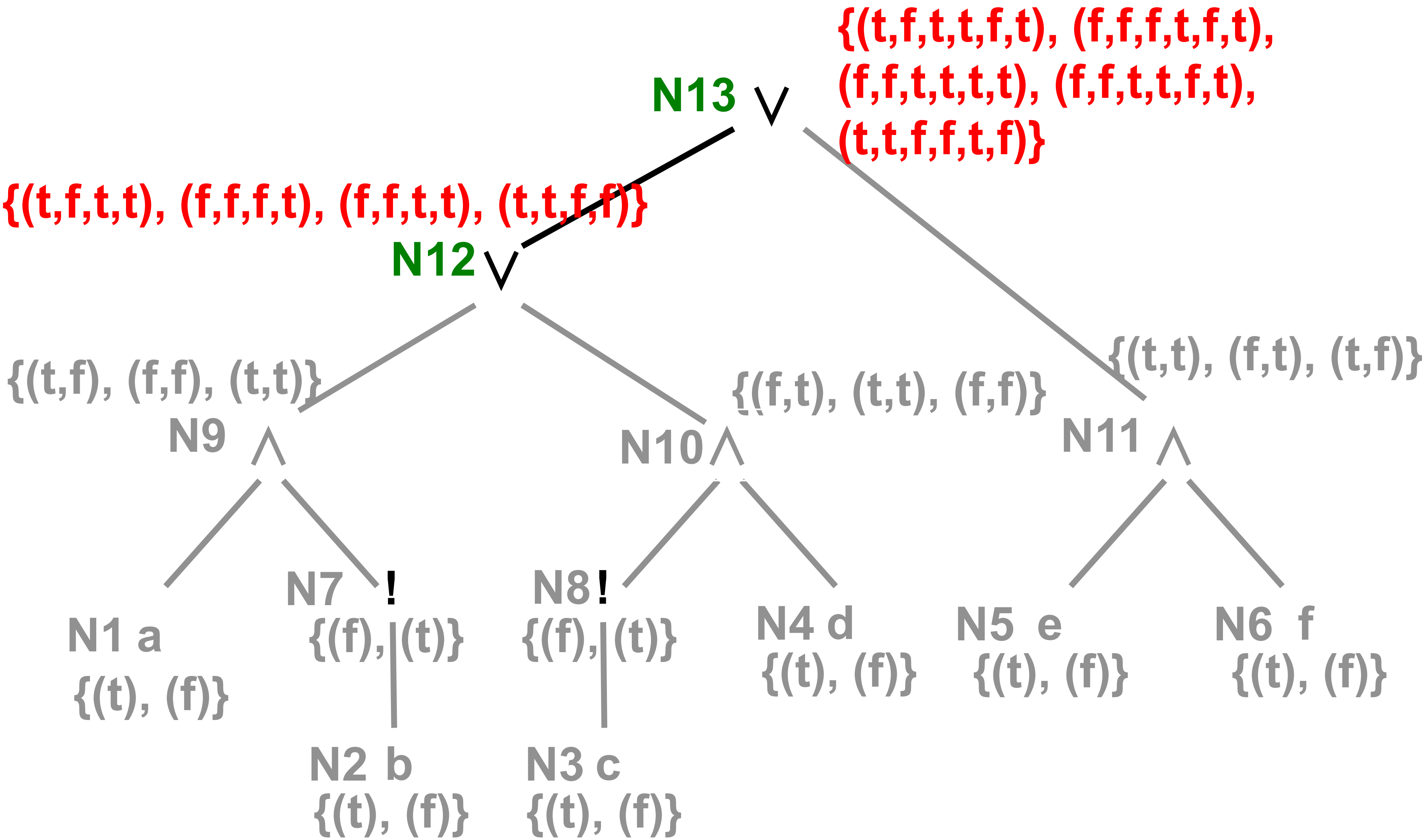
$\{f_{N11}\}$ could also be {(t,f)}

$\{f_{N12}\}$ could also be {(t,t,f,f)}

ONTO product could be reversed

BOR-CSET Step 2.1

Label OR-Nodes



T_{BOR}

The test cases for the Boolean variables a, b, c, d, e, f are:

t_1: <a=true,b=false,c=true,d=true,e=false,f=true>

t_2: <a=false,b=false,c=false,d=true,e=false,f=true>

t_3: <a=false,b=false,c=true,d=true,e=true,f=true>

t_4: <a=false,b=false,c=true,d=true,e=false,f=true>

t_5: <a=true,b=true,c=false,d=false,e=true,f=false>

What If?

What if you were told to use the BOR procedure on the predicate

a!bc + cd!e

… which is not singular (c is repeated).

A correct way to solve this assignment would be to rewrite the predicate as

c(a!b + d!e)

which is singular.

A Minimal BRO-Constraint Set

A test set adequate with respect to a BRO constraint set for predicate p_r, guarantees the detection of all combinations of single or multiple Boolean operator and relational operator faults.

- **p_r contains only singular expressions**

Label each leaf node that is a relational expression S_N = {(<), (=), (>)}.

Compute S_N for each non-leaf node using the BOR-CSET steps 2.1, 2.2, and 2.3.

S^t and S^f for the BRO Constraint Set

Separating the BRO-constraint S into its true (S^t) and false (S^f) components

relop: >	$S^t = \{(>)\}$	$S^f = \{(<), (=)\}$
relop: ≥	$S^t = \{(=), (>)\}$	$S^f = \{(<)\}$
relop: =	$S^t = \{(=)\}$	$S^f = \{(<), (>)\}$
relop: <	$S^t = \{(<)\}$	$S^f = \{(=), (>)\}$
relop: ≤	$S^t = \{(<), (=)\}$	$S^f = \{(>)\}$

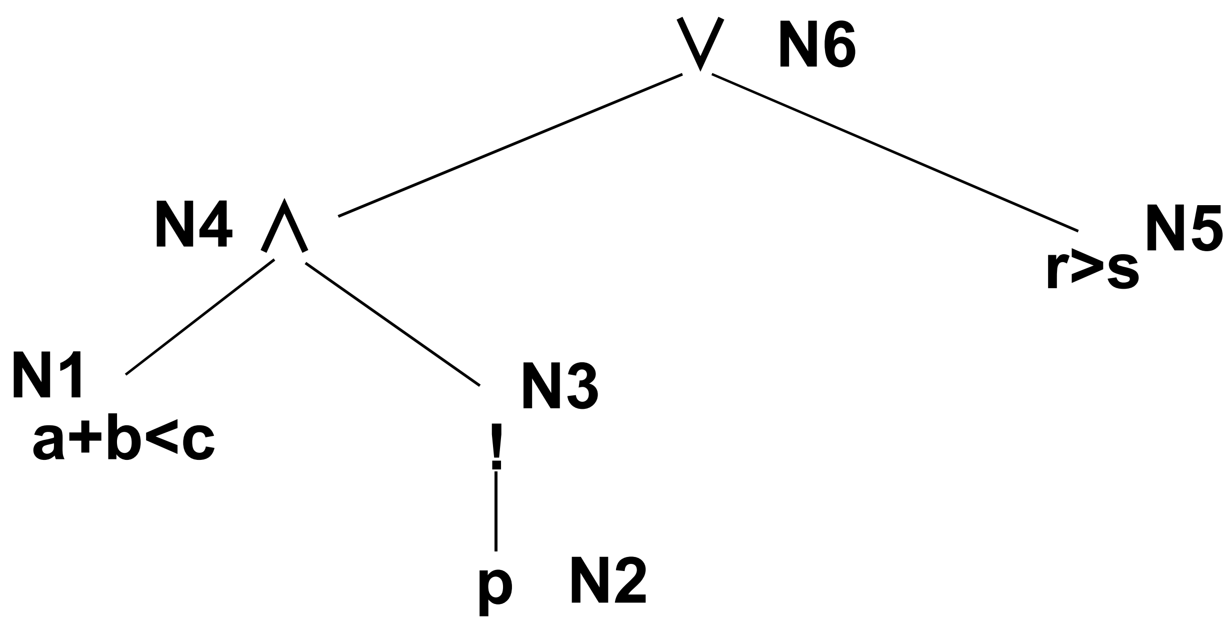
t_N denotes an element of S^t_N

f_N denotes an element of S^f_N

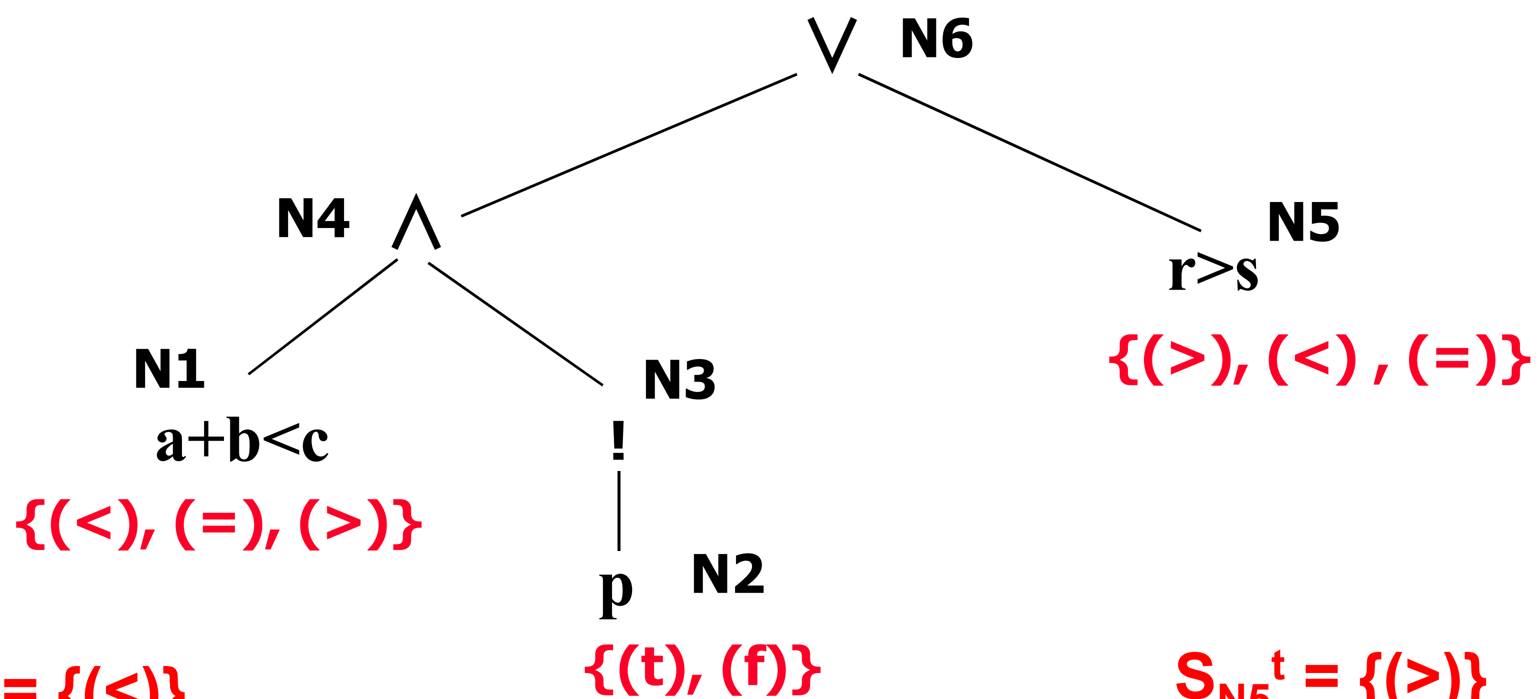
Mathur, Example 4.15

p_r: (a+b < c) ∧ !p ∨ (r > s)

Construct AST(p_r)



Label each leaf node with its constraint set S.



$S_{N1}{}^{t} = \{(<)\}$
$S_{N1}{}^{f} = \{(=), (>)\}$

$S_{N2}{}^{t} = \{(t)\}$
$S_{N2}{}^{f} = \{(f)\}$

$S_{N5}{}^{t} = \{(>)\}$
$S_{N5}{}^{f} = \{(<), (=)\}$

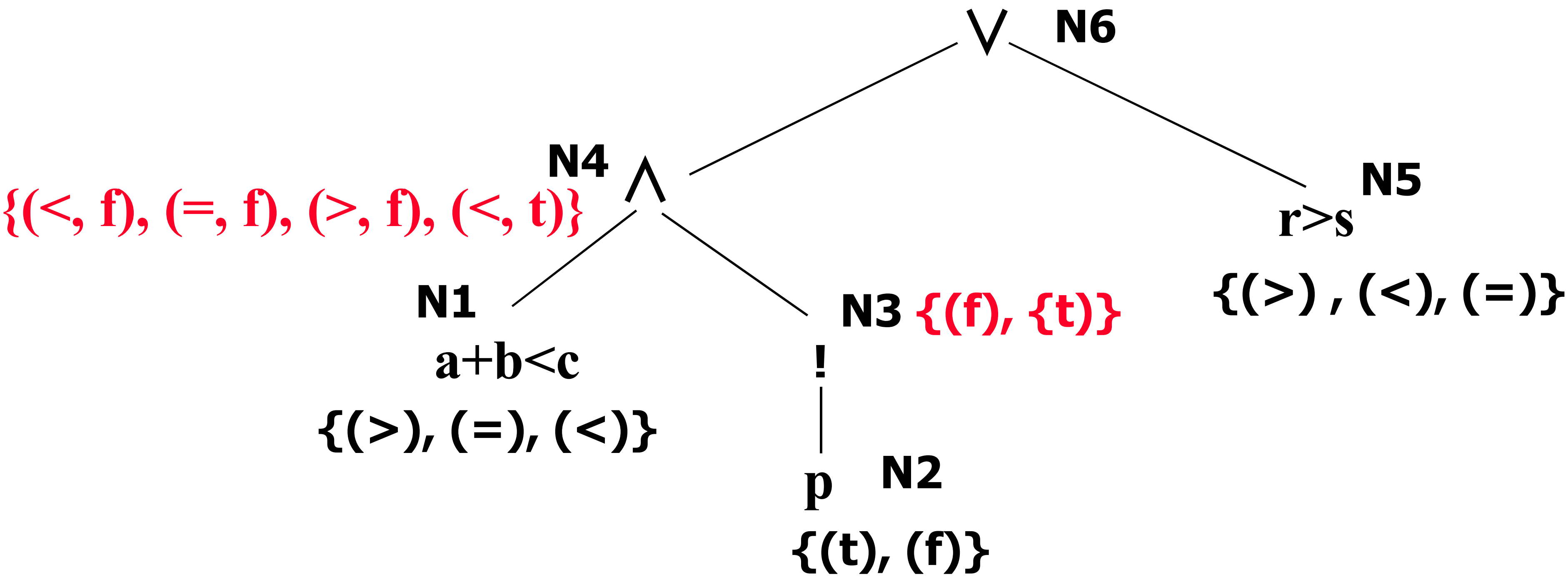
Traverse the tree and compute the constraint set for each internal node.

$$S^t_{N3} = S_{N2}{}^f = \{(f)\}$$

$$S^f_{N3} = S_{N2}{}^t = \{(t)\}$$

$$S^t_{N4} = S_{N1}{}^t \otimes S_{N3}{}^t = \{(<)\} \otimes \{(f)\} = \{(<, f)\}$$

$$\begin{aligned} S^f_{N4} &= (S^f_{N1} \times \{(t_{N3})\}) \cup (\{(t_{N1})\} \times S^f_{N3}) \\ &= (\{(=), (>)\} \times \{(f)\}) \cup \{(<)\} \times \{(t)\}) \\ &= \{(=, f), (>, f)\} \cup \{(<, t)\} \\ &= \{(=, f), (>, f), (<, t)\} \end{aligned}$$


∨ N6
N4 ∧
{(<, f), (=, f), (>, f), (<, t)}
N5
r>s
{(>) , (<), (=)}
N1
a+b<c
{(>), (=), (<)}
N3 {(f), {t)}
!
p N2
{(t), (f)}

Compute the constraint set for the root node (this is an OR-node).

$$S^t_{N6} = (S^t_{N4} \times \{(f_{N5})\}) \cup (\{(f_{N4})\} \times S^t_{N5})$$

$$= (\{(<,f)\} \times \{(<)\}) \cup \{(=,f)\} \times \{(>)\})$$

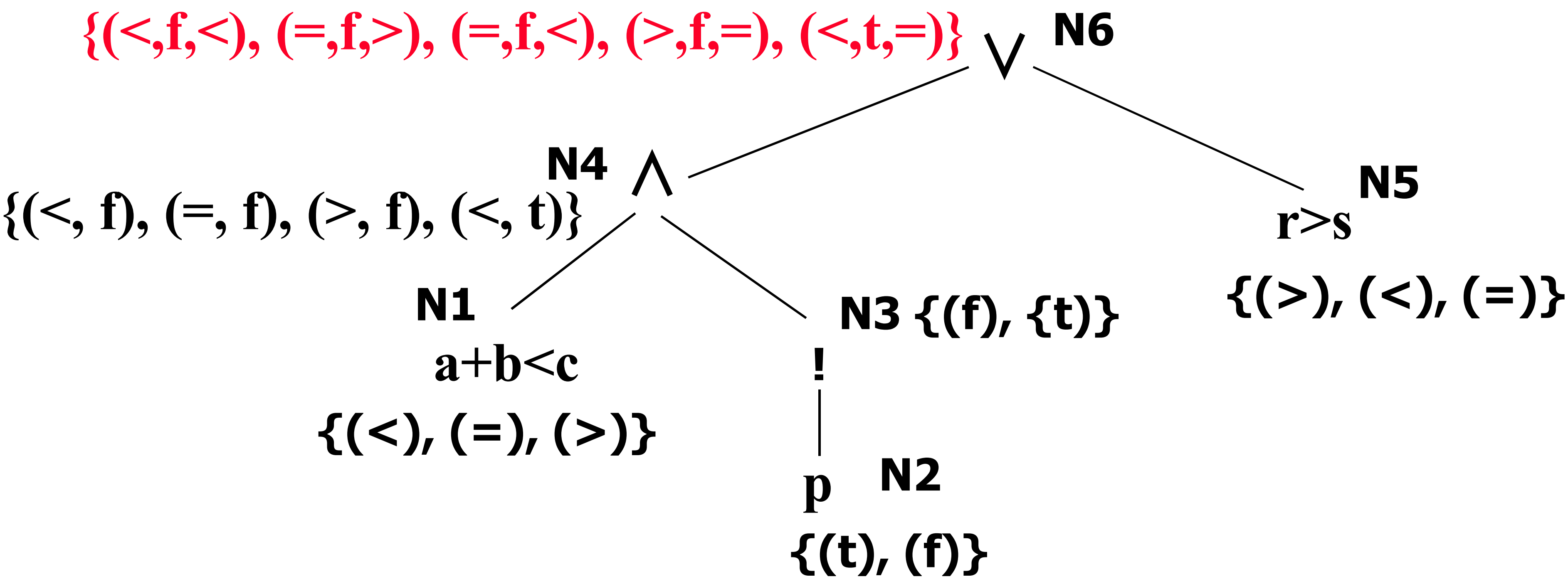
$$= \{(<,f,<)\} \cup \{(=,f,>)\}$$

$$= \{(<,f,<), (=,f,>)\}$$

$$S^f_{N6} = S^f_{N4} \otimes S^f_{N5}$$

$$= \{(=, f), (>, f), (<, t)\} \otimes \{(<), (=)\}$$

$$= \{(=,f,<), (>,f,=), (<,t,=)\}$$



{(<,f,<), (=,f,>), (=,f,<), (>,f,=), (<,t,=)}
∨
N6
N4
∧
{(<, f), (=, f), (>, f), (<, t)}
N5
r>s
{(>), (<), (=)}
N1
a+b<c
{(<), (=), (>)}
N3 {(f), {t)}
!
p
N2
{(t), (f)}

Given the constraint set for p_r

$$(a + b < c) \wedge !p \vee (r > s)$$

construct T_{BRO}

{(<,f,<), (=,f,>), (=,f,<), (>,f,=), (<,t,=)}

	a+b>c	p	r>s	Test case <a,b,c,p,r,s>
t_1	<	f	<	<1,1,3,false,1,2>
t_2	=	f	>	<1,0,1,false,2,1>
t_3	=	f	<	<1,1,2,false,1,2>
t_4	>	f	=	<2,2,3,false,0,0>
t_5	<	t	=	<1,1,3,true,2,2>

The BRE-Constraint Set

A test set adequate with respect to a BRO constraint set for predicate p_r, guarantees the detection of any Boolean operator, relation operator, arithmetic expression, or combination thereof faults.

- **p_r contains only singular expressions**

The BRE-constraint set for a relational expression is {(-ε), (=), (+ε)}, ε > 0.

***e_1 relop e_2* is separated into S^t and S^f based on**

+ε	**$0 < e_1 - e_2 \leq +\varepsilon$**
-ε	**$-\varepsilon \leq e_1 - e_2 < 0$**

S^t and S^f for the BRE Constraint Set

Separating the BRE-constraint S into its true (S^t) and false (S^f) components

relop: >	$S^t = \{(+\varepsilon)\}$	$S^f = \{(-\varepsilon), (=)\}$
relop: ≥	$S^t = \{(=), (+\varepsilon)\}$	$S^f = \{(-\varepsilon)\}$
relop: =	$S^t = \{(=)\}$	$S^f = \{(-\varepsilon), (+\varepsilon)\}$
relop: <	$S^t = \{(-\varepsilon)\}$	$S^f = \{(=), (+\varepsilon)\}$
relop: ≤	$S^t = \{(-\varepsilon), (=)\}$	$S^f = \{(+\varepsilon)\}$

BRE-CSET

Label each leaf node that is a relational expression S_N = {(-ε), (=), (+ε)}.

Compute S_N for each non-leaf node using the BOR-CSET steps 2.1, 2.2, and 2.3.

Maximum Size of the Test Sets

If a predicate contains n AND/OR operations, then the maximum size of the BOR-adequate test set is n + 2.

The maximum size of a BRO- or BRE-adequate test set is 2n + 3.

Note that the BOR-CSET procedure generates significantly smaller test sets than the cause-effect graph decision table.

- **fault detection effectiveness of BOR-CSET is slightly less than that of the CEGDT procedure**

BOR Constraints for Nonsingular Expressions

Test generation procedures described so far are for singular predicates.

- **a singular predicate contains only one occurrence of each variable**

How to generate BOR constraints for nonsingular predicates?

Look at some nonsingular expressions

- **disjunctive normal forms (DNF)**
- **mutually singular components**

Examples of Nonsingular Expressions and DNF

Predicate (p_r)	DNF	Mutually singular components in p_r
ab(b+c)	abb+abc	a b(b+c)
a(bc+ bd)	abc+abd	a (bc+bd)
a(bc+!b+de)	abc+a!b+ade	a bc+!b de

Generating BOR Constraints for Nonsingular Expressions

The modified BOR strategy to generate tests from predicate p_r uses

- **the BOR-CSET procedure**
- **another procedure called the Meaningful Impact (MI) procedure**

MI procedure generates tests from any Boolean expression p_r, singular or nonsingular

- **p_r must be in DNF**

A literal occurrence in a Boolean formula is said to have a *meaningful impact* on the value of the formula for a given test case if, everything else being the same, a different truth value assignment to that literal would have resulted in the formula evaluating to a different value.

MI-CSET Procedure
MI-CSET 1

Input

- **Boolean expression $E = e_1 + e_2 + \ldots e_n$ in minimal DNF**

Output

- **a set of constraints S_E that guarantees the detection of missing or extra NOT operator faults in a faulty version of E**

Start of procedure MI-CSET

Step 1. For each term e_i, $1 \leq i \leq n$, construct T_{ei} as the set of constraints that make e_i true

MI-CSET 2-3

Step 2. Let $TS_{ei} = T_{ei} - \cup_{j=1,i\neq j}^{n} T_{ej}$

- **note that for $i \neq j$, $TS_{ei} \cap TS_{ej} = \emptyset$**
- **note that TS contains only the unique trues**

The *unique true points* are of interest because they demonstrate the meaningful impact of each literal of a term on the evaluation of the formula to true.

Step 3. Construct S_E^t by including one constraint from each $TS_{ei,}$ $1 \leq i \leq n$

- **note that for each constraint c in S_E^t, E(c) = true**

MI-CSET 4

Step 4. Let e_i^j denote the term obtained by complementing the j^{th} literal in term e_i, for 1≤l≤n and 1≤j≤ l_i

- **count the literals in a term from left to right, leftmost first**

Construct F_{eij} as the set of constraints that make e_i^j false

MI-CSET 5-7

Step 5. Let $FS_{eij} = F_{eij} - \cup_{k=1}^{n} T_{ek}$

- **for any constraint c in FS_{eij}, E(c) = false**

Step 6. Construct S_E^f that is minimal and covers each FS_{eij} at least once

Step 7. Construct the desired constraint set for E as $S_E = S_E^t \cup S_E^f$

End of procedure MI-CSET

Effectiveness of the MI Procedure

As discussed by Chen and Lau (2001), the basic meaningful impact procedure guarantees finding all occurrences of

- **Expression Negation Fault (ENF)**
- **Literal Negation Fault (LNF)**
- **Term Omission Fault (TOF)**
- **Operator Reference Fault (ORF)**
- **Literal Omission Fault (LOF)**

MI is not guaranteed to find

- **Literal Insertion Fault (LIF)**
  - **a literal not appearing in a term is inserted in that term, e.g., ab!c + de → ab!cd + de**
- **Literal Reference Fault (LRF)**
  - **a literal is replaced by another literal not appearing in the term, e.g., ab!c + de → abd + de**

Operator Fault Categories *(Badhera)*

Operator Reference Fault (ORF): In this class of fault, a binary logical operator '.' is replaced by '+' or vice versa.

Expression Negation Fault (ENF): A sub-expression in the statement is replaced by its negation (!).

Variable Negation Fault (VNF): An atomic Boolean literal is replaced by its negation (!).

Associative Shift Fault (ASF): This fault occurs when an association among conditions is incorrectly implemented due to misunderstanding about operator evaluation properties.

- **Parenthesis omission fault (POF): A pair of parentheses has been incorrectly omitted from the Boolean expression.**
- **Parenthesis insertion fault (PIF): A pair of parentheses has been incorrectly inserted from the Boolean expression.**

Missing Variable Fault (MVF): A condition in the expression is missing with respect to original expression.

Variable Reference Fault (VRF): A condition is replaced by another input which exists in the statement.

Clause Conjunction Fault (CCF): A condition a in expression is replaced by a.b, where b is a variable in the expression.

Clause Disjunction Fault (CDF): A condition a in expression is replaced with a+b, where b is a variable in the expression.

Stuck at 0: A condition a is replaced with 0 in the function.

Stuck at 1: A condition a is replaced with 1 in the function.

Mathur, Example 4.17

Consider the nonsingular predicate: a(bc + !bd)

DNF equivalent is

E = abc + a!bd

a, b, c, and d are Boolean variables (literals)

Each literal represents a condition

- **a could represent r < s**

Step 1. For each term e_i, $1 \le i \le n$, construct T_{ei} as the set of constraints that make e_i true

Express E in DNF notation

- $E = e_1 + e_2$
  - where $e_1 = abc$ and $e_2 = a!bd$

Step 1: Construct a constraint set T_{e1} for e_1 that makes e_1 true

- $T_{e1} = \{(t,t,t,t), (t,t,t,f)\}$

Construct T_{e2} for e_2 that makes e_2 true

- $T_{e2} = \{(t,f,t,t), (t,f,f,t)\}$

Step 2. Let $TS_{ei} = T_{ei} - \cup_{j=1,i\neq j}^{n} T_{ei}$

- note that for $i \neq j$, $TS_{ei} \cap TS_{ei} = \varnothing$

Step 2: From each T_{ei} , remove the constraints that are in any other T_{ej}

There are no common constraints between T_{e1} and T_{e2} in our example

TS_{e1} = {(t,t,t,t), (t,t,t,f)}

TS_{e2} = {(t,f,t,t), (t,f,f,t)}

Step 3. Construct S_E^t by including one constraint from each $TS_{ei,}$ $1 \le i \le n$

- note that for each constraint c in S_E^t, E(c) = true

Step 3: Construct S_E^t by selecting one element from each TS_{ei}

S_E^t = {(t,t,t,f), (t,f,f,t)}

There are four possible S_E^t

- Note that Mathur picked the last elements

For each constraint x in S_E^t we get E(x) = true

S_E^t is minimal

Step 4. Let e_i^j denote the term obtained by complementing the j^{th} literal in term e_i, for $1 \le i \le n$ and $1 \le j \le l_i$

- **count the literals in a term from left to right, leftmost first**

Step 4: For each term in E, obtain terms by complementing each literal, one at a time.

- **e_1 = abc and e_2 = a!bd**

e_1^1 = !abc e_1^2 = a!bc e_1^3 = ab!c

e_2^1 = !a!bd e_2^2 = abd e_2^3 = a!b!d

MI-CSET Step 4 (cont). Construct F_{eij} as the set of constraints that make e_i^j true

From each term e above, derive constraints F_e that make e true

$Fe_1^1 = \{(f,t,t,t), (f,t,t,f)\}$!abc
$Fe_1^2 = \{(t,f,t,t), (t,f,t,f)\}$ a!bc
$Fe_1^3 = \{(t,t,f,t), (t,t,f,f)\}$ ab!c

$Fe_2^1 = \{(f,f,t,t), (f,f,f,t)\}$!a!bd
$Fe_2^2 = \{(t,t,t,t), (t,t,f,t)\}$ abd
$Fe_2^3 = \{(t,f,t,f), (t,f,f,f)\}$ a!b!d

Step 5. Let $FS_{eij} = F_{eij} - \cup_{k=1}^{n} T_{ek}$

- for any constraint c in FS_{eij}, $E(c)$ = false

Step 5: Construct FS_e by removing from F_e any constraint that appeared in any of the two sets T_e constructed earlier

- constraints common with T_{e1} and T_{e2} are removed
- T_{e1} = {(t,t,t,t), (t,t,t,f)}
- T_{e2} = {(t,f,t,t), (t,f,f,t)}

$FSe_1^1 = Fe_1^1$ = {(f,t,t,t), (f,t,t,f)}
FSe_1^2 = {(t,f,t,f)} {(t,f,t,t), (t,f,t,f)}
$FSe_1^3 = Fe_1^3$ = {(t,t,f,t), (t,t,f,f)}

$FSe_2^1 = Fe_2^1$ = {(f,f,t,t), (f,f,f,t)}
FSe_2^2 = {(t,t,f,t)} {(t,t,t,t), (t,t,f,t)}
$FSe_2^3 = Fe_2^3$ = {(t,f,t,f), (t,f,f,f)}

Step 6. Construct S_E^f that is minimal and covers each FS_{eij} at least once

Step 6: Construct S_E^f by selecting one constraint from each FS_e

S_E^f = {(f,t,t,f), <u>(t,f,t,f)</u>, <u>(t,t,f,t)</u>, (f,f,t,t)}

FSe_1^1 = {(f,t,t,t), (f,t,t,f)}
FSe_1^2 = {<u>(t,f,t,f)</u>}
FSe_1^3 = {<u>(t,t,f,t)</u>, (t,t,f,f)}

FSe_2^1 = {(f,f,t,t), (f,f,f,t)}
FSe_2^2 = {(t,t,f,t)}
FSe_2^3 = {<u>(t,f,t,f)</u>, (t,f,f,f)}

Step 7. Construct the desired constraint set for E as
$S_E = S_E^t \cup S_E^f$

Step 7: Now construct $S_E = S_E^t \cup S_E^f$

S_E = {(t,t,t,f), (t,f,f,t), (f,t,t,f), (t,f,t,f), (t,t,f,t), (f,f,t,t)}

Each constraint in S_E^t makes E true

Each constraint in S_E^f makes E false

The BOR-MI-CSET Procedure

Takes a nonsingular expression E as input

Generates a constraint set that guarantees the detection of Boolean operator faults in the implementation of E

BOR-MI-CSET

Step 1. Partition E into a set of n mutually singular components $E = \{E_1, E_2, \ldots E_n\}$

Step 2. Generate the BOR-constraint set for each singular component in E using the BOR-CSET procedure

Step 3. Generate the MI-constraint set for each nonsingular component in E using the MI-CSET procedure

Step 4. Combine the constraints generated in steps 2 and 3 using Step 2 from the BOR-CSET procedure to obtain the constraint set for E

Mathur, Example 4.18

Consider a nonsingular Boolean expression

E = a(bc + !bd)

Step 1. Partition E into a set of n mutually singular components $E = \{E_1, E_2, \ldots E_n\}$

Mutually singular components of E

$e_1 = a$

$e_2 = bc + !bd$

Step 2. Generate the BOR-constraint set for each singular component in E using the BOR-CSET procedure

Use the BOR-CSET procedure to generate the constraint set for the singular component e1 = a

For component e_1 = a we get

$S_{e1}^{t} = \{t\}$ $S_{e1}^{f} = \{f\}$

S_{e1}^{t} is the true constraint set for e_1
S_{e1}^{f} is the false constraint set for e_1

Step 3. Generate the MI-constraint set for each nonsingular component in E using the MI-CSET procedure

Use the MI-CSET procedure for the DNF nonsingular component e_2 = bc + !bd

e_2 can be written as e_2 = u + v
where u = bc and v = !bd

Apply the MI-CSET procedure to obtain the BOR constraint set for e_2

T_u = {(t,t,t), (t,t,f)} T_v = {(f,t,t), (f,f,t)}

- **note that the tuples are (b,c,d) by position**

MI-CSET Step 2. Let $TS_{ei} = T_{ei} - \cup_{j=1,i\neq j}^{n} T_{ei}$

$TS_u = T_u = \{(t,t,t), (t,t,f)\}$ $\quad$ $TS_v = T_v = \{(f,t,t), (f,f,t)\}$

MI-CSET Step 3. Construct S_E^t by including one constraint from each TS_e

$S_{e2}^t = \{(t,t,f), (f,t,t)\}$

MI-CSET Step 4. Let e_i^j denote the term obtained by complementing the j^{th} literal in term e_i, for $1 \leq i \leq n$ and $1 \leq j \leq l_i$

- **u = bc and v = !bd**

u_1 = !bc $\quad$ **u_2 = b!c**

v_1 = bd $\quad$ **v_2 = !b!d**

MI-CSET Step 4 (cont). Construct F_{eij} as the set of constraints that make e_i^j true

- $u_1 = !bc$ $\quad u_2 = b!c$
- $v_1 = bd$ $\quad v_2 = !b!d$

F_{u1}= {(f,t,t), (f,t,f)} $\quad F_{u2}$ = {(t,f,t), (t,f,f)}
F_{v1}= {(t,t,t), (t,f,t)} $\quad F_{v2}$ = {(f,t,f), (f,f,f)}

MI-CSET Step 5. Let $FS_{eij} = F_{eij} - \cup_{k=1}^{n} T_e$

- T_u = {<u>(t,t,t)</u>, (t,t,f)} T_v = {<u>(f,t,t)</u>, (f,f,t)}

FS_{u1} = {(f,t,f)} $\quad FS_{u2}$ = {(t,f,t), (t,f,f)}
FS_{v1} = {(t,f,t)} $\quad FS_{v2}$ = {(f,t,f), (f,f,f)}

MI-CSET Step 6. Construct S_E^f that is minimal and covers each FS_{eij} at least once

- FS_{u1} = {<u>(f,t,f)</u>} FS_{u2} = {<u>(t,f,t)</u>, (t,f,f)}
- FS_{v1} = {<u>(t,f,t)</u>} FS_{v2} = {<u>(f,t,f)</u>, (f,f,f)}

S_{e2}^f = {(f,t,f), (t,f,t)}

MI-CSET Step 7. Construct the desired constraint set for E as $S_E = S_E^t \cup S_E^f$

- S_{e2}^t = {(t,t,f), (f,t,t)}

S_{e2} = {(t,t,f), (f,t,t), (f,t,f), (t,f,t)}

Recap

From Step 2 of BOR-MI-CSET, for component $e_1 = a$

$S_{e1}^t = \{t\}$ $S_{e1}^f = \{f\}$

From Step 3 of BOR-MI-CSET, for component $e_2 = bc + !bd$

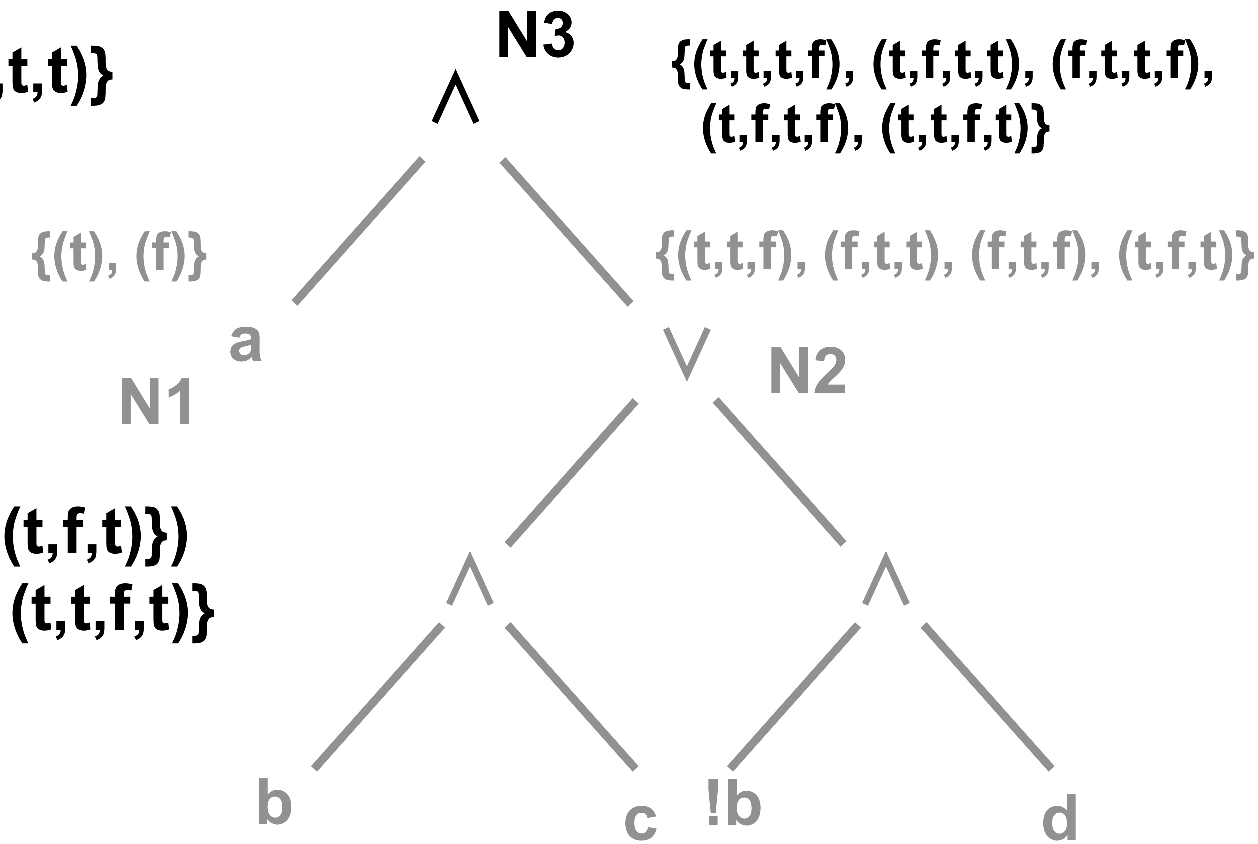
$S_{e2}^t = \{(t,t,f), (f,t,t)\}$ $S_{e2}^f = \{(f,t,f), (t,f,t)\}$

Step 4. Combine the constraints generated in steps 2 and 3 using Step 2 from the BOR-CSET procedure to obtain the constraint set for E

AND Node

$$S_{N3}{}^{t} = S_{N1}{}^{t} \otimes S_{N2}{}^{t}$$
$$= \{(t)\} \otimes \{(t,t,f), (f,t,t)\}$$
$$= \{(t,t,t,f), (t,f,t,t)\}$$

$$S_{N3}{}^{f} = (S_{N1}{}^{f} \times \{t_2\}) \cup (\{t_1\} \times S_{N2}{}^{f})$$
$$= (\{(f)\} \times \{(t,t,f)\}) \cup (\{(t)\} \times \{(f,t,f), (t,f,t)\})$$
$$= \{(f,t,t,f), (t,f,t,f), (t,t,f,t)\}$$



Another BOR-MI Example

Use the BOR-MI-CSET procedure to derive the constraint set for the following predicate

p_r: a + b + cd + !ce

where a,b,c,d,e are Boolean variables.

BOR-MI-CSET 1

Consider the nonsingular Boolean expression

E = a + b + cd + !ce

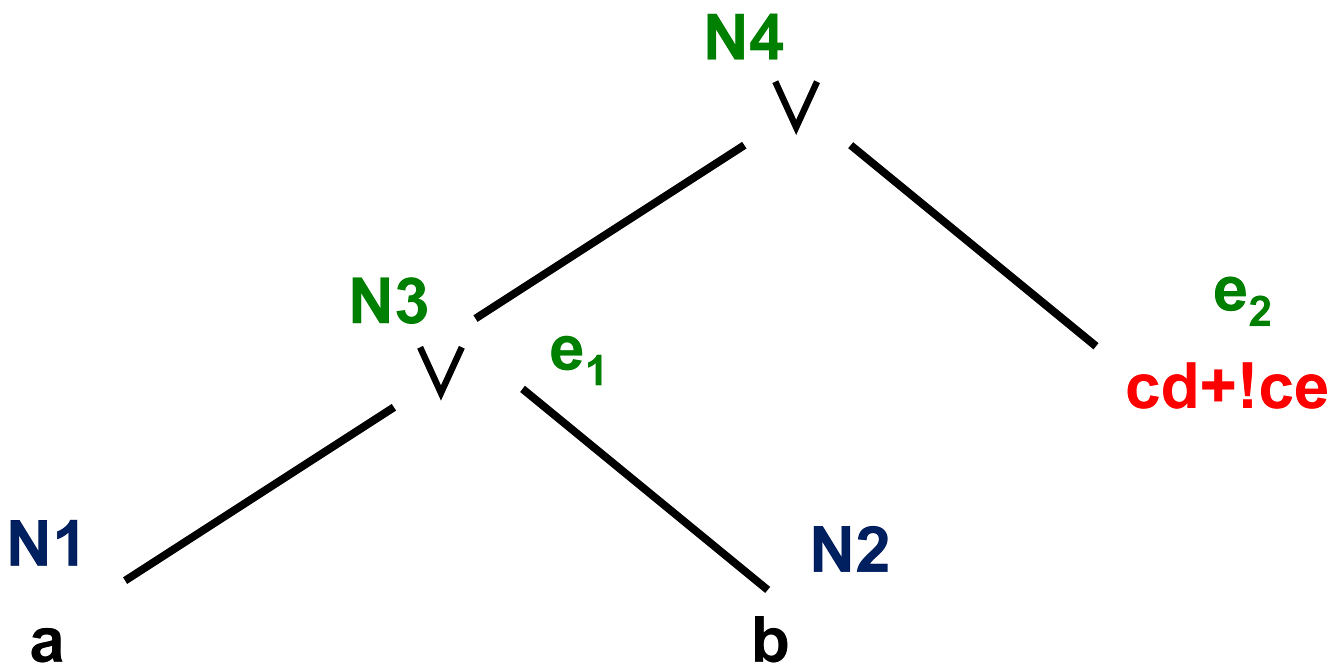
Step 1. Partition E into a set of n mutually singular components E = {E_1, E_2, … E_n}

Mutually singular components of E

e_1 = a + b

e_2 = cd + !ce

$AST\ for\ a + b + cd + !ce$



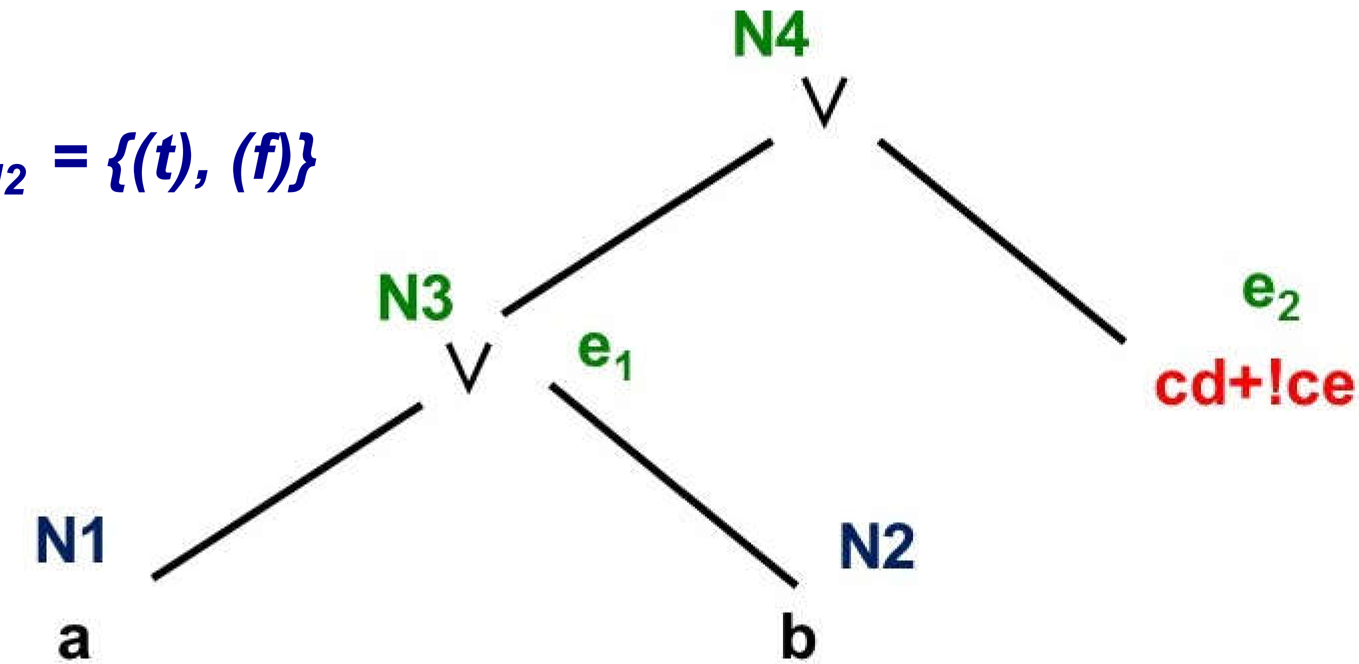
BOR-CSET Step 1

Label Leaf Nodes

$S_{N1}^{t} = S_{N2}^{t} = \{(t)\}$

$S_{N1}^{f} = S_{N2}^{f} = \{(f)\}$

$S_{N1} = S_{N2} = \{(t), (f)\}$



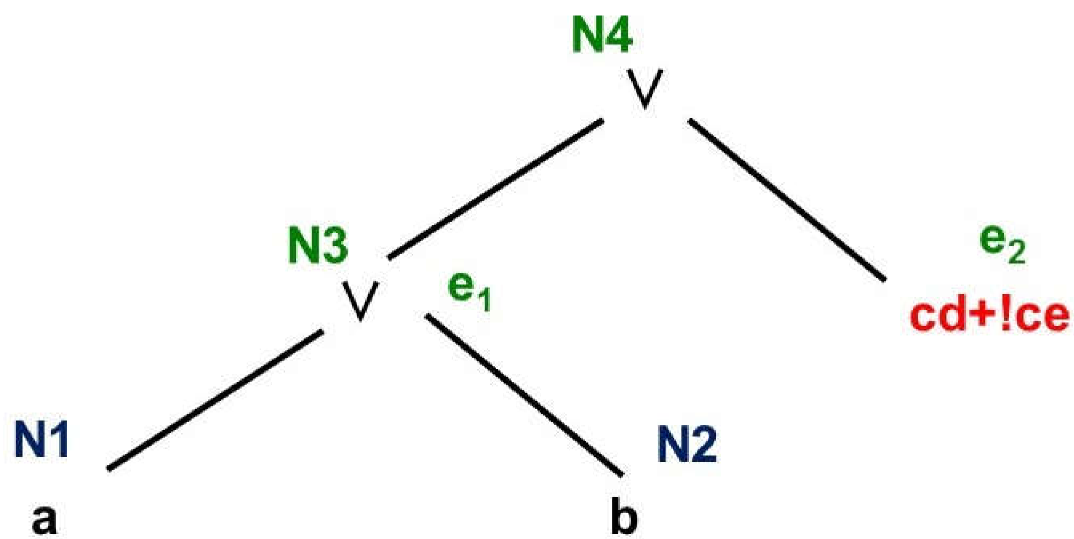
BOR-CSET Step 2.1 for N3

N3 is an OR-node for f(a,b).

$$S_{N3}^{t} = (S_{N1}^{t} \times \{f_{N2}\}) \cup (\{f_{N1}\} \times S_{N2}^{t})$$
$$= (\{(t)\} \times \{(f)\}) \cup (\{(f)\} \times \{(t)\})$$
$$= \{(t,f)\} \cup \{(f,t)\}$$
$$= \{(t,f), (f,t)\}$$

$$S_{N3}^{f} = S_{N1}^{f} \otimes S_{N2}^{f}$$
$$= \{(f)\} \otimes \{(f)\}$$
$$= \{(f,f)\}$$

$$S_{N3} = \{(t,f), (f,t), (f,f)\}$$



BOR-MI-CSET 2

Step 2. Generate the BOR-constraint set for each singular component in E using the BOR-CSET procedure

Use the BOR-CSET procedure to generate the constraint set for the singular component $e_1 = a + b$

For component $e_1 = a + b$ we get

$S_{e1}^{t} = \{(t,f), (f,t)\}$ **$S_{e1}^{f} = \{(f,f)\}$**

S_{e1}^{t} is the true constraint set for e_1
S_{e1}^{f} is the false constraint set for e_1

BOR-MI-CSET 3

Step 3. Generate the MI-constraint set for each nonsingular component in E using the MI-CSET procedure

Use the MI-CSET procedure for the DNF nonsingular component e_2 = cd + !ce

e_2 can be written as e_2 = u + v
where u = cd, v = !ce

Apply the MI-CSET procedure to obtain the BOR constraint set for e_2

T_u = {(t,t,t), (t,t,f)} T_v = {(f,t,t), (f,f,t)}

- **note that the tuples are (c,d,e) positionally**

MI-CSET 2

MI-CSET Step 2. Let $TS_{ei} = T_{ei} - \cup_{j=1,i\neq j}^{n} T_{ej}$

- **$T_u = \{(t,t,t), (t,t,f)\}$**
- **$T_v = \{(f,t,t), (f,f,t)\}$**

There are no duplicate Ts.

$TS_u = T_u = \{(t,t,t), (t,t,f)\}$

$TS_v = T_v = \{(f,t,t), (f,f,t)\}$

MI-CSET 3

MI-CSET Step 3. Construct S_E^t by including one constraint from each TS_e

- $TS_u = \{(t,t,t), (t,t,f)\}$
- $TS_v = \{(f,t,t), (f,f,t)\}$

$S_{e2}^t = \{(t,t,t), (f,t,t)\}$

MI-CSET 4

MI-CSET Step 4. Let e_i^j denote the term obtained by complementing the j^{th} literal in term e_i, for $1 \le i \le n$ and $1 \le j \le l_i$

- u = cd, v = !ce

u_1 = !cd	u_2 = c!d
v_1 = ce	v_2 = !c!e

Construct F_{eij} as the set of constraints that make e_i^j false

F_{u1}= {(f,t,t), (f,t,f)}	F_{u2} = {(t,f,t), (t,f,f)}
F_{v1}= {(t,t,t), (t,f,t)}	F_{v2} = {(f,t,f), (f,f,f)}

MI-CSET 5

MI-CSET Step 5. Let $FS_{eij} = F_{eij} - \cup_{k=1}^{n} T_e$

- **T_u = {(t,t,t), (t,t,f)}**
- **T_v = {(f,t,t), (f,f,t)}**

Eliminate candidate Fs that are really true.

FS_{u1}= {~~(f,t,t)~~, (f,t,f)}
FS_{v1}= {~~(t,t,t)~~, (t,f,t)}

FS_{u2} = {(t,f,t), (t,f,f)}
FS_{v2} = {(f,t,f), (f,f,f)}

MI-CSET 6

MI-CSET Step 6. Construct S_E^f that is minimal and covers each FS_{eij} at least once

- FS_{u1}= {(f,t,f)} FS_{u2} = {(t,f,t), (t,f,f)}
- FS_{v1}= {(t,f,t)} FS_{v2} = {(f,t,f), (f,f,f)}

S_{e2}^f = {(f,t,f), (t,f,t)}

MI-CSET 7

MI-CSET Step 7. Construct the desired constraint set for E as $S_E = S_E^t \cup S_E^f$

- $S_{e2}^t = \{(t,t,t), (f,t,t)\}$
- $S_{e2}^f = \{(f,t,f), (t,f,t)\}$

$S_{e2} = \{(t,t,t), (f,t,t), (f,t,f), (t,f,t)\}$

BOR-CSET and MI-CSET

Recap

From Step 2 of BOR-MI-CSET, for component $e_1 = a + b$

$S_{e1}^{t} = S_{N3}^{t} = \{(t,f), (f,t)\}$ $S_{e1}^{f} = S_{N3}^{f} = \{(f,f)\}$

From Step 3 of BOR-MI-CSET, for component $e_2 = cd + !ce$

$S_{e2}^{t} = \{(t,t,t), (f,t,t)\}$

$S_{e2}^{f} = \{(f,t,f), (t,f,t)\}$

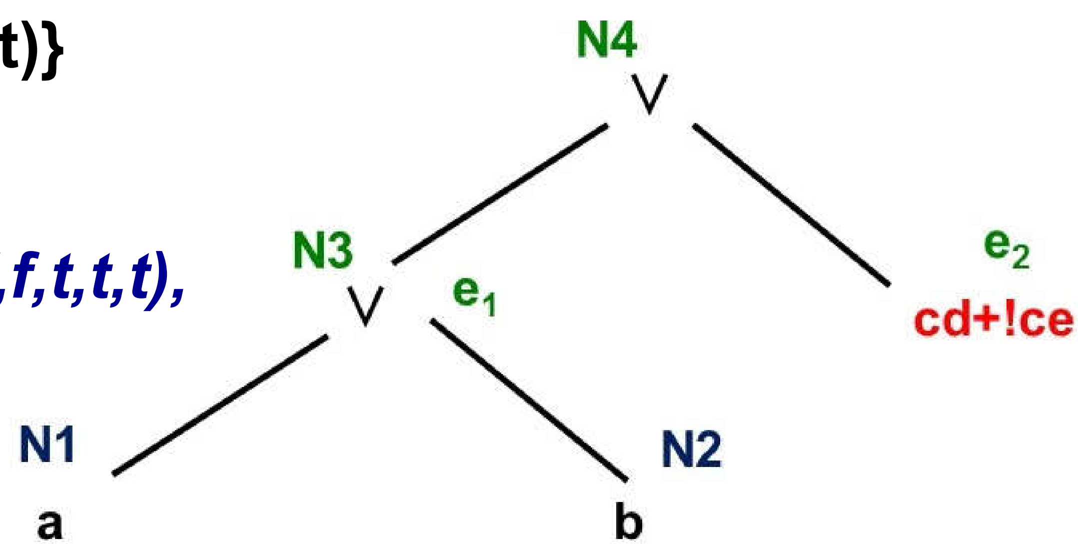
BOR-CSET Step 2.1 for N4

N4 is an OR-node for f(a,b,c,d).

$$S_{N4}^{t} = (S_{N3}^{t} \times \{f_{e2}\}) \cup (\{f_{N3}\} \times S_{e2}^{t})$$
$$= (\{(t,f), (f,t)\} \times \{(f,t,f)\}) \cup (\{(f,f)\} \times \{(t,t,t), (f,t,t)\})$$
$$= \{(t,f,f,t,f), (f,t,f,t,f)\} \cup \{(f,f,t,t,t), (f,f,f,t,t)\}$$
$$= \{(t,f,f,t,f), (f,t,f,t,f), (f,f,t,t,t), (f,f,f,t,t)\}$$

$$S_{N4}^{f} = S_{N3}^{f} \otimes S_{e2}^{f}$$
$$= \{(f,f)\} \otimes \{(f,t,f), (t,f,t)\}$$
$$= \{(f,f,f,t,f), (f,f,t,f,t)\}$$

$S_{N4} = \{(t,f,f,t,f), (f,t,f,t,f), (f,f,t,t,t), (f,f,f,t,t), (f,f,f,t,f), (f,f,t,f,t)\}$



Test Set for BOR-MI

S_{N3} = {(t,f,f,t,f), (f,t,f,t,f), (f,f,t,t,t), (f,f,f,t,t), (f,f,f,t,f), (f,f,t,f,t)}

$T_{BOR\text{-}MI}$ = {t_1: <a=true, b=false, c=false, d=true, e=false>,
t_2: <a=false, b=true, c=false, d=true , e=false>,
t_3: <a=false, b=false, c=true, d=true , e=true>,
t_4: <a=false, b=false, c=false, d=true , e=true>,
t_5: <a=false, b=false, c=false, d=true, e=false>,
t_6: <a=false, b=false, c=true, d=false, e=true>}

Faulty Test Sets

Predicate E	t_1	t_2	t_3	t_4	t_5	t_6
a + b + cd + !ce	t	t	t	t	f	f

Faulty predicates						
ab!cde	f	f	f	f	f	f
a + b + cd + !cd	t	t	t	t	t	f
a + b + d + e	t	t	t	t	t	t

MI vs BOR-MI

Note that Example 4.17 and Example 4.18 both have the nonsingular predicate: a(bc + !bd)

Example 4.17 illustrates using the MI-CSET procedure.

- **MI procedure requires that p_r be in DNF**
- **a(bc+!bd) = abc + a!bd**
- **result: S_E = {(t,t,t,t), (t,f,f,f), (f,t,t,f), (t,f,t,f), (t,t,f,t), (f,f,t,t)}**

Example 4.18 illustrates using the BOR-MI-CSET procedure.

- **mutually singular components of E are e_1 = a and e_2 = bc + !bd**
- **result: S_{N3} = {(t,t,t,f), (t,f,t,t), (f,t,t,f), (t,f,t,f), (t,t,f,t)}**

MI-CSET generates six test cases
BOR-MI-CSET generates five test cases

Combining Test Techniques

Equivalence partitioning and boundary value analysis are the most commonly used methods for test generation while doing functional testing.

Given a function f to be tested in an application, apply these techniques to generate tests for f.

Most requirements contain conditions under which functions are to be executed.

- **Predicate testing generates tests to ensure that each condition is tested adequately.**

To combine equivalence partitioning, boundary value analysis, and predicate testing procedures to generate tests for a requirement of the following type:

if condition then action 1, action 2, … action n;

For the condition – apply predicate testing.

For actions – apply equivalence partitioning, boundary value analysis (and predicate testing if there are nested conditions).

Summary – Things to Remember

Singular, mutually singular, DNF

BOR (n+2) and BRO (2n+3) test generation

- **singular predicates**

MI test generation

- **nonsingular DNF predicates**

BOR-MI test generation

- **smaller test sets, more powerful than MI**

Questions and Answers