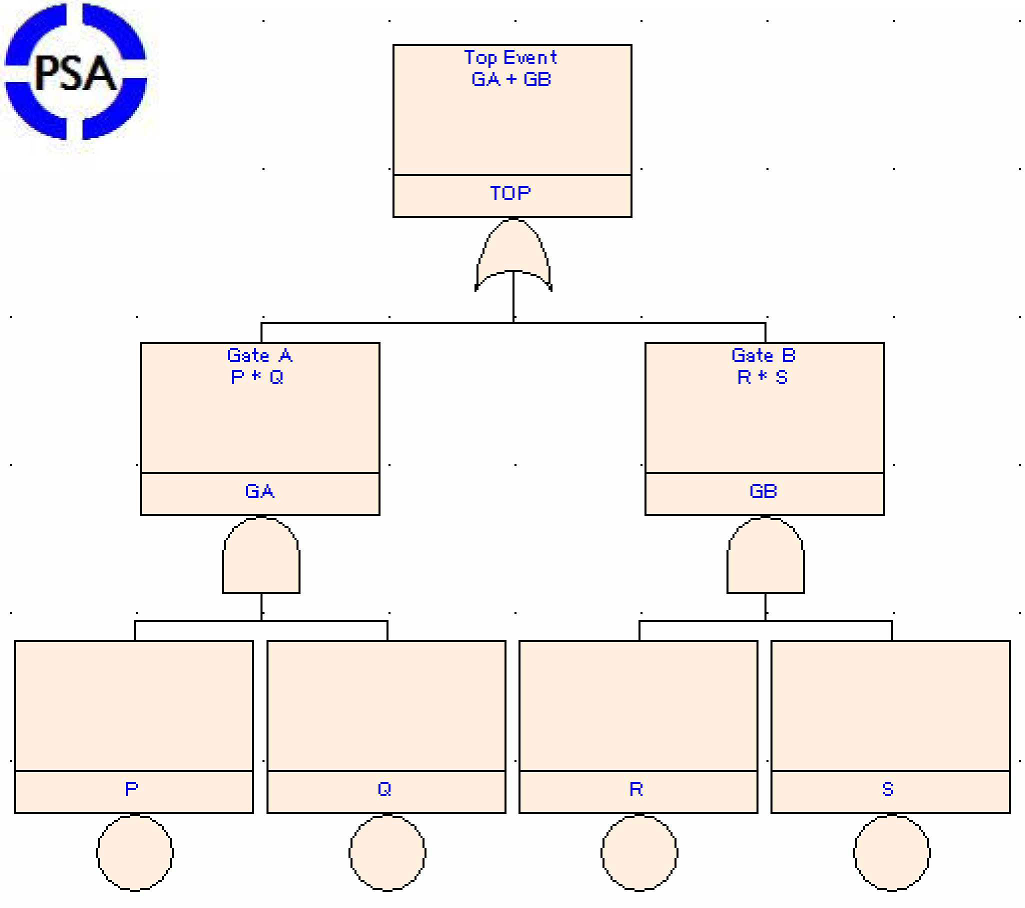
PSA

The Destructive Truth Table Method with Truncation



			Pmin
p	q	TRUE	P(p) * P(q)
p	not q	FALSE	P(p) *(1- P(q))
not p	q	FALSE	(1-P(p)) * P(q)
not p	not q	FALSE	(1-P(p)) *(1- P(q))
GA		TRUE	P(p) * P(q)
		FALSE	P(p) *(1- P(q)) + (1-P(p)) * P(q) + (1-P(p)) *(1- P(q))
r	s	TRUE	P(r) * P(s)
r	not s	FALSE	P(r) *(1- P(s))
not r	s	FALSE	(1-P(r)) * P(s)
not r	not s	FALSE	(1-P(r)) *(1- P(s))
GB		TRUE	P(r) * P(s)
		FALSE	P(r) *(1- P(s)) + (1-P(r)) * P(s) + (1-P(r)) *(1- P(s))
GA	GB	TRUE	P(p) * P(q) + P(r) * P(s)
GA	not GB	TRUE	P(p) * P(q) + (1-(P(r) * P(s)))
not GA	GB	TRUE	(1-(P(p) * P(q))) + P(r) * P(s)
not GA	not GB	FALSE	(1-(P(p) * P(q))) + (1-(P(r) * P(s)))
TOP		TRUE	(P(p) * P(q) + P(r) * P(s)) + (P(p) * P(q) + (1-(P(r) * P(s)))) + (1-(P(p) * P(q))) + P(r) * P(s)
		FALSE	(1-(P(p) * P(q))) + (1-(P(r) * P(s)))

The Method

1. Start at the bottom of the tree with an empty truth table;
2. At each gate, add its inputs to the truth table;
3. If a line in the truth table is lower than the truncation limit, add it to Delta and eliminate from the table;
4. Solve the probability for the gate;
5. Collapse the truth table by substituting the gate name for its inputs;
6. Continue up the tree.

The Claim

Ptop <= Pexact <= Ptop+Delta

Example #1 sent to us …

Truth Table Method for Solving T = x + y

T = x + y	Pr	a	0.01
x = a /(b + d)	Pr	b	0.01
y = d	Pr	d	0.01
Pr(d)	d	0.01	TRUE
	/d	0.99	FALSE
Collapse for y	y	0.01	TRUE
	/y	0.99	FALSE
expand for (b + d)			
	y	0.01	TRUE
	/y	0.99	FALSE
	bd	0.0001	TRUE
	b/d	0.0099	TRUE
	/bd	0.0099	TRUE
	/b/d	0.9801	FALSE
Collapse for (b+d)	y	0.01	TRUE
	/y	0.99	FALSE
	(b + d)	0.0199	TRUE
	/(b + d)	0.9801	FALSE
Rename (b+d) to Z	Z	0.0199	TRUE
	/Z	0.9801	FALSE
Expand for a /Z	y	0.01	TRUE
	/y	0.99	FALSE
	Z	0.0199	TRUE
	/Z	0.9801	FALSE
	a	0.01	TRUE
	/a	0.99	FALSE
	aZ	0.000199	FALSE
	a/Z	0.009801	TRUE
	/aZ	0.019701	FALSE
	/a/Z	0.970299	FALSE
Collapse for x	y	0.01	TRUE
	/y	0.99	FALSE
	x	0.009801	TRUE
	/x	0.990199	FALSE
Expand for x+y	y	0.01	TRUE
	/y	0.99	FALSE
	x	0.009801	TRUE
	/x	0.990199	FALSE
	xy	0.00009801	TRUE
	x/y	0.00970299	TRUE
	/xy	0.00990199	TRUE
	/x/y	0.98029701	FALSE
Collapse for T	T	0.01970299	TRUE
	/T	0.98029701	FALSE

The true value for T as calculated by Aralia is: 1.9801e-02.

In this example, if 0 <= Delta <= 1e-4, then Pexact lies outside the upper bound: Ptop = (Ptop+Delta) <= Pexact

… example #2 sent to us …

Truth Table Method for Solving T = x + y

T = x + y	Pr	a	0.01
x = a /(b + d)	Pr	b	0.01
y = /d	Pr	d	0.01
Pr(d)	d	0.01	TRUE
	/d	0.99	FALSE
Collapse for y	y	0.99	TRUE
	/y	0.99	FALSE
expand for (b + d)			
	y	0.99	TRUE
	/y	0.99	FALSE
	bd	0.0001	TRUE
	b/d	0.0099	TRUE
	/bd	0.0099	TRUE
	/b/d	0.9801	FALSE
Collapse for (b+d)	y	0.01	TRUE
	/y	0.99	FALSE
	(b + d)	0.0199	TRUE
	/(b + d)	0.9801	FALSE
Rename (b+d) to Z	Z	0.0199	TRUE
	/Z	0.9801	FALSE
Expand for a /Z	y	0.99	TRUE
	/y	0.99	FALSE
	Z	0.0199	TRUE
	/Z	0.9801	FALSE
	a	0.01	TRUE
	/a	0.99	FALSE
	aZ	0.000199	FALSE
	a/Z	0.009801	TRUE
	/aZ	0.019701	FALSE
	/a/Z	0.970299	FALSE
Collapse for x	y	0.99	TRUE
	/y	0.99	FALSE
	x	0.009801	TRUE
	/x	0.990199	FALSE
Expand for x+y	y	0.99	TRUE
	/y	0.99	FALSE
	x	0.009801	TRUE
	/x	0.990199	FALSE
	xy	0.00970299	TRUE
	x/y	0.00970299	TRUE
	/xy	0.98029701	TRUE
	/x/y	0.98029701	FALSE
Collapse for T	T	0.99970299	TRUE
	/T	0.00029701	FALSE

The true value for T as calculated by Aralia is: 9.9000e-01

In this example, with any Delta, Pexact lies below Ptop by 9e-2: Pexact < Ptop <= (Ptop+Delta)

So why does this happen?

Consider the two simple cases:

1) F+F where $p(F)=0.1$: With the method you get (even without a Delta) $p(F+F) = p(F)+p(F)-p(F).p(F) = 0.19$ which is an overestimation

2) F.F where $p(F)=0.1$: with the method you get $p(F.F) = p(F).p(F) = 0.01$ which is an underestimation.

You could argue that one can detect this repeated F case. But it is easy to hide it by taking two formulae F and F' that are structurally close, but not the same, and very far apart in the tree.

This developer has not followed the “Gate Collapse Rule” to be sure that all such cases are eliminated.

So why does this happen?

Consider the two simple cases:

1) F+F where p(F)=0.1: With the method you get (even without a Delta)
p(F+F) = p(F)+p(F)-p(F).p(F) = 0.19 which is an overestimation

o

the

t all

Gate Collapse Rule

A gate *g* can be collapsed iff:

1. All other gates which share at least one of its inputs are in the truth table;
2. Gate *g* has all inputs resolved in expanded form;
3. All gates which share inputs fulfill the above two conditions.

We don’t present this example to criticize the algorithm, but, rather, to show that an attempt must be made to PROVE the algorithm’s correctness and limits.

We have had techniques for over 40 years to demonstrate algorithm and program correctness.

- Floyd (1967, *Assigning Meanings to Programs*)
- Hoare (1969, *An Axiomatic Basis for Computer Programs*)
- Scott & Strachey (1972, *Denotational Semantics*)

Algorithm development is a branch of mathematics.

We must concern ourselves with accuracy of calculations and the proof of such before concerning ourselves with speed. Remember, good cooking takes the time it takes.