

Formal Methods in Software Engineering

Engr. Madeha Mushtaq
Department of Computer Science
Iqra National University

Dijkstra Game

- Suppose we have a jar full of white and black balls.
- The game is that, we get two balls out from the jar, depending upon the color of those two balls, we put one ball back in the jar.
- If we get a white and a black ball out, we put a white ball back in the jar.
- But if we get 2 white balls or 2 black balls i.e. 2 balls of the same color, then we will put a black ball back in the jar.
- Now the problem is could we tell the color of the last ball, given the no. of white and black balls in the jar.

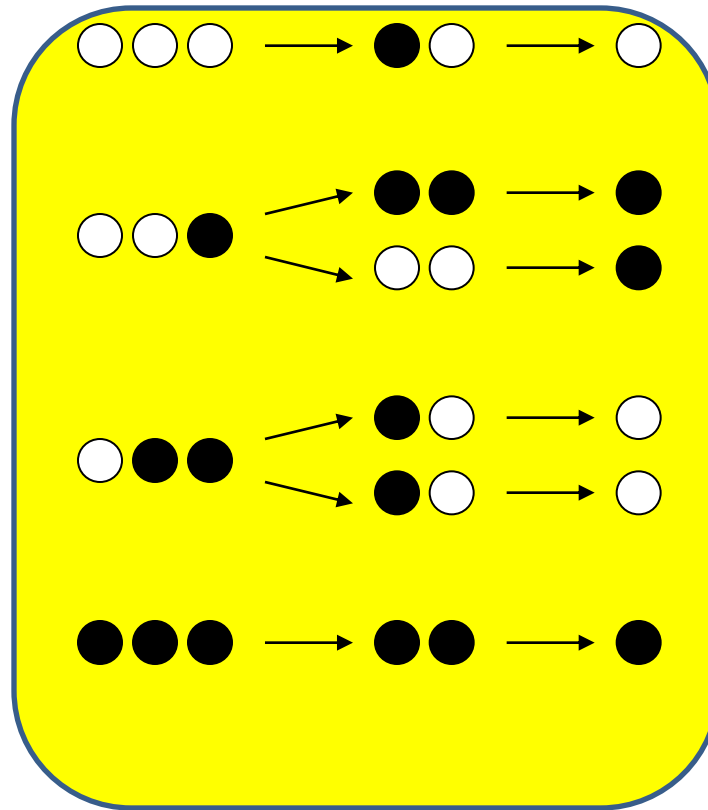
Dijkstra Game

- Lets try to play this game. We will start with a single ball.
- Suppose we have only one ball in the jar, when we take that ball out, obviously that is the color of the last ball.
- If it's a white ball, white will be the color of the last ball out, if it's a black ball, we will have the black color last ball.
- Now going towards a two-ball game. We have two balls in the jar, we take them out.
- Now there are 3 possible combinations.
- We can have a white and a black out, in this case we will put a white ball back in the jar.
- So the color of the last ball will be white.

Dijkstra Game

- The other 2 possibilities are, if we get 2 white balls or 2 black balls out, in this case we will put a black ball back in the jar.
- So the color of the last ball will be black.
- Now lets move towards the three-ball game.
- We have three balls in the jar, in this case we have 4 possible combinations.
 - We have 3 white balls
 - We have 2 white and 1 black ball
 - We have 2 black and 1 white ball
 - We have 3 black balls

Dijkstra Game



A THREE-BALL
GAME

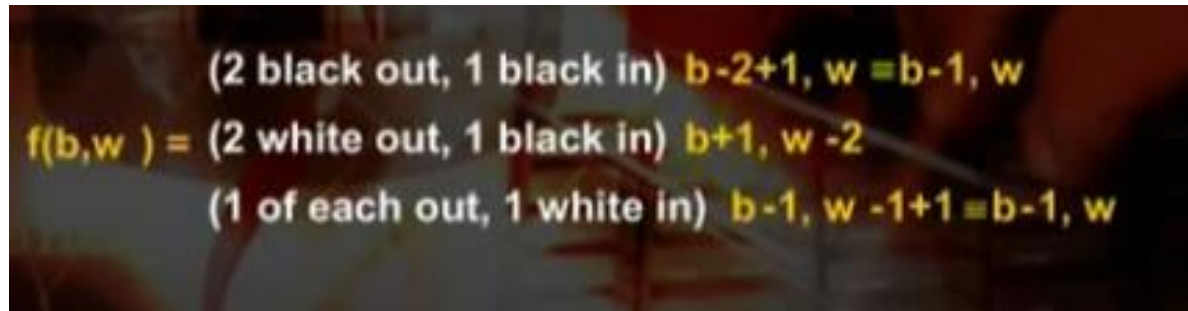
Dijkstra Game

- So from the three-ball game, can we come up with the color of the last ball??
- WE observe that the color of the last ball is dependent upon the number of white balls in the jar.
- If we have even number of white balls, we will end up with a black ball, and if we have odd number of white balls then we will end up with a white ball.
- Is this hypothesis correct?
- If we have 100 balls in the jar, could we be sure about the outcome of the last ball?

Mathematical Model

- How many games do we need to play to be sure that our hypothesis is correct?
- We need a formal proof to show that the hypothesis is correct.
- For formal proof, first we need to develop a mathematical model for our game.
- We need to understand the steps of the game in order to develop a mathematical model.

Mathematical Model



(2 black out, 1 black in) $b-2+1, w = b-1, w$
 $f(b,w) =$ (2 white out, 1 black in) $b+1, w-2$
(1 of each out, 1 white in) $b-1, w-1+1 = b-1, w$

- Let $f(b,w)$ be the function that puts in and gets ball out of the jar.
- Let w be the number of white balls and b be the number of black balls.

Mathematical Model

- If we look at the mathematical model, the total number of balls is reduced by exactly one in each move.
- Parity of the white balls does not change. i.e. if we have even number of white balls, it will remain even. If we have odd number of white balls, it will remain odd.
- Hence we reach at the conclusion that the parity of the white balls can determine the outcome of the game.
- Hence our hypothesis is correct/true.

Tools for FM in SW Engineering

- There are two basic tools for Formal Methods in software engineering.
- Logic and
- Set Theory
- We will use these two tools to develop mathematical models and provide proofs.

Logic and Propositional Calculus

- Proposition is a statement that can either be true or false.
- For example, I say, “I live in Peshawar”.
- This statement can either be true or false, so this is a proposition.
- For example, the statement $a=b$, now depending on the values of a and b , this statement can either be true or false.
- And hence this is a proposition.
- Other examples of propositions are $a<b$, $a>b$ etc.

Logic and Propositional Calculus

- “Read this book carefully.” Now this statement is not a proposition.
- As this statement can’t be true or false.
- Once we have a proposition, we can associate symbols with those propositions.
- We can use and join these propositions with the help of logical connectors.
- Some examples of logical connectors are And, Or, Not, Implication and If and only if.

Implication

- We use this sign \Rightarrow for implication.
- Let P and Q be two propositions.

Implication Truth Table

P	Q	$P \Rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

Bi-Conditional If and Only If

- This is a bi-directional implication.
- Let P and Q be two propositions.
- If P is true, then Q is true. If P is false, then Q is false.
- So we can use Bi-Conditional If and Only If for equivalence.
- If we look closely at the truth table, we will see it is opposite of XOR.

P	Q	$P \Leftrightarrow Q$
T	T	T
T	F	F
F	F	T
F	T	F

$$P \Leftrightarrow Q \text{ means } P \Rightarrow Q \wedge Q \Rightarrow P$$

Tautology

- A compound proposition that is always true, irrespective of the truth values of the comprising propositions, is called a tautology.
- For example, this statement will always be true.

$$p \vee \neg p$$

Contradiction

- Contradiction is the opposite of tautology.
- It is a proposition that is always false, no matter what is the values of the comprising propositions.
- For example, $p \wedge \neg p$.
- It will always be false.

Logical Equivalence

- The propositions **p** and **q** are called logically equivalent if $\mathbf{p} \Leftrightarrow \mathbf{q}$ is tautology.
- It is written as ,
$$\mathbf{p} \equiv \mathbf{q}$$
- For example: $\neg (\mathbf{p} \vee \mathbf{q}) \equiv \neg \mathbf{p} \wedge \neg \mathbf{q}$
- This bi-directional If and Only If is always true and hence is a tautology and logically equivalent.

Some Useful equivalences

$p \text{ or true} \equiv \text{true}$

$p \text{ or false} \equiv p$

$p \text{ and true} \equiv p$

$p \text{ and false} \equiv \text{false}$

$\text{true} \Rightarrow p \equiv p$

$\text{false} \Rightarrow p \equiv \text{true}$

$p \Rightarrow \text{true} \equiv \text{true}$

$p \Rightarrow \text{false} \equiv \text{not } p$

$p \text{ or } p \equiv p$

$p \text{ and } p \equiv p$

$\text{not not } p \equiv p$

$p \text{ or not } p \equiv \text{true}$

$p \text{ and not } p \equiv \text{false}$

Some Useful equivalences

- And, Or and “Bi-Conditional If and Only If” are associative and commutative.
- De-Morgan’s Law holds for Implication and “Bi-Conditional If and Only If”.
- And is distributive over Or, Or is distributive over And.
- Or is distributive over \Rightarrow , \Rightarrow is distributive over And, Or and \Rightarrow .
- \Rightarrow is distributive over \Leftrightarrow as shown in the next slide.

Some Useful equivalences

distributivity of

- and over or
- or over and
- or over \Rightarrow
- \Rightarrow over and
- \Rightarrow over or
- \Rightarrow over \Rightarrow
- \Rightarrow over \Leftrightarrow

associativity of

\vee , \wedge , and \Leftrightarrow

Commutativity of

\vee , \wedge , and \Leftrightarrow

Demorgan's law

- Implication
- if and only if

End of Slides