CSC 404 - Foundations of Computation

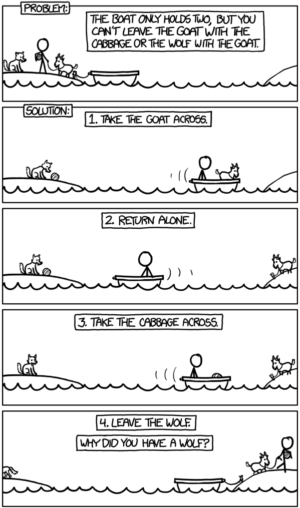
Section 1.0 – Riddles and Computation

Farmer River Crossing Problem

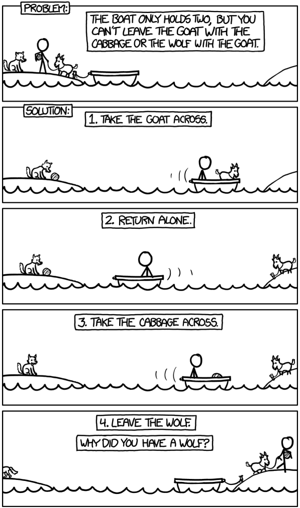
Problem 1.0.1.

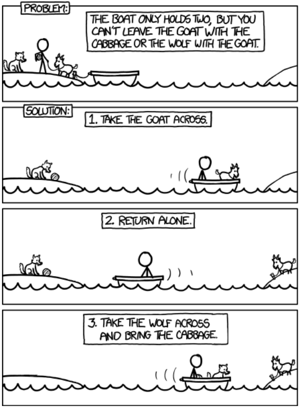
A farmer wants to cross a river and take with him a wolf, a goat, and a cabbage. There is a boat that can fit the farmer plus either the wolf, the goat, or the cabbage. If the wolf and the goat are alone on one shore, the wolf will eat the goat. If the goat and the cabbage are alone on the shore, the goat will eat the cabbage. How can the farmer bring the wolf, the goat, and the cabbage across the river?

Farmer River Crossing Problem



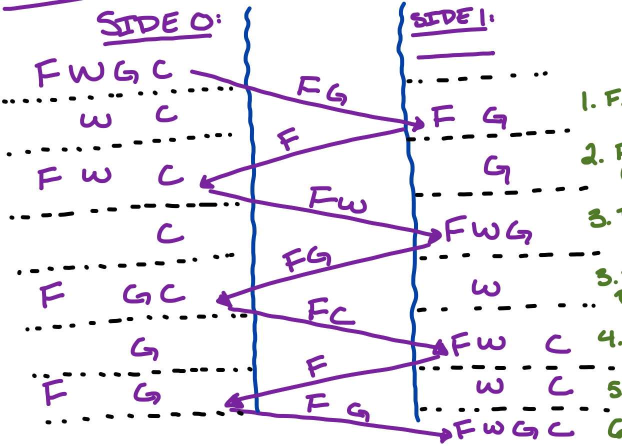
Farmer River Crossing Problem





Farmer River Crossing Problem

IDEAS



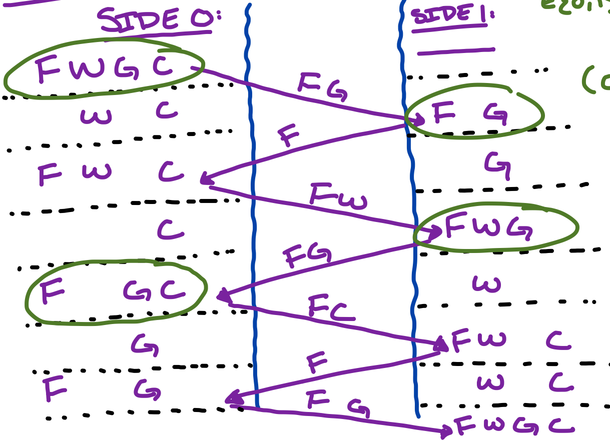
1. FARMER BRINGS GOAT (0→1)
2. FARMER RETURNS TO SIDE 0 (LEAVES GOAT)
3. FARMER BRINGS WOLF (0→1)
3. FARMER LEAVES WOLF AND BRINGS GOAT BACK (1→0)
4. FARMER LEAVES GOAT AND BRINGS CABBAGE (0→1)
5. FARMER RETURNS TO SIDE 0
6. FARMER BRINGS GOAT (0→1)

Farmer River Crossing Problem

IDEAS

(FARMER, WOLF, GOAT, CABBAGE)

$\in \{0,1\}$ $\in \{0,1\}$ $\in \{0,1\}$ $\in \{0,1\}$



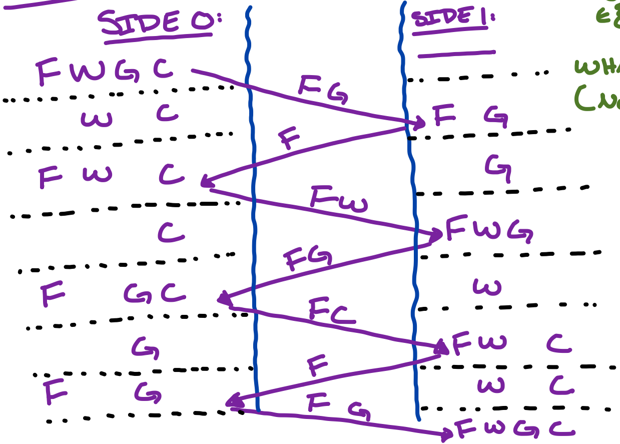
$(0,0,0,0) \to (1,0,1,0)$

$(1,1,1,0) \to (0,1,0,0)$

IDEAS

$(\underbrace{\text{FARMER}}_{\in\{0,1\}}, \underbrace{\text{WOLF}}_{\in\{0,1\}}, \underbrace{\text{GOAT}}_{\in\{0,1\}}, \underbrace{\text{CABBAGE}}_{\in\{0,1\}})$

WHAT ARE POSSIBLE STATES?
(NO GOAT OR CABBAGE EATEN)



Farmer River Crossing Problem

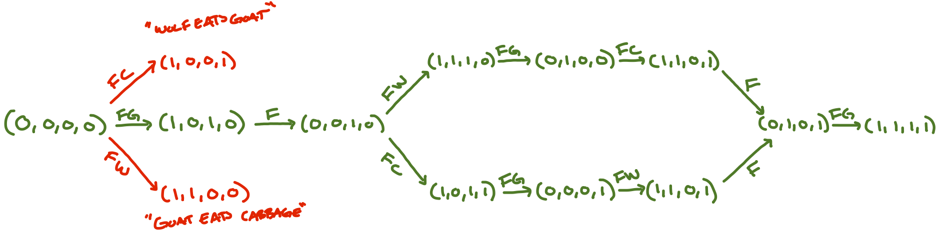
(0, 0, 0, 0) (0, 0, 0, 1) (0, 0, 1, 0) (0, 0, 1, 1)
(GOAT EATS CABBAGE)

(0, 1, 0, 0) (0, 1, 0, 1) (0, 1, 1, 0)
(WOLF EATS GOAT)
(0, 1, 1, 1)
NO SUPERVISION!
WOLF EATS GOAT
OR GOAT EATS CABBAGE

(1, 0, 0, 0)
NO SUPERVISION!
WOLF EATS GOAT
OR GOAT EATS CABBAGE
(1, 0, 0, 1)
WOLF EATS GOAT
(1, 0, 1, 0) (1, 0, 1, 1)

(1, 1, 0, 0)
GOAT EATS CABBAGE!
(1, 1, 0, 1) (1, 1, 1, 0) (1, 1, 1, 1)

Farmer River Crossing Problem



Farmer River Crossing Problem

MORE DETAILS!

HOW TO ENCODE MOVEMENT? (RECALL (F, W, G, C))

$$(0,0,1,0) \xrightarrow{FW} \underbrace{(1,1,1,0)}_{\underbrace{(0+1,0+1,1,0)}_{\underbrace{(0,0,1,0)}_{\text{START}} + \underbrace{(1,1,0,0)}_{\text{FW}}}}$$

$$(1,1,1,0) \xrightarrow{FW} \underbrace{(0,0,1,0)}_{\underbrace{(1,1,1,0)+(1,1,0,0)}_{\underbrace{(1+1,1+1,1+0,0+0)}_{\underbrace{(2,2,1,0)}_{(0,0,1,0) \bmod 2}}}}$$

$1+1 = 2 \equiv 0 \bmod 2$

Reduce each component modulo 2!

IN GENERAL

$$(a_0,a_1,a_2,a_3) \xrightarrow{FW} (a_0,a_1,a_2,a_3)+(1,1,0,0) = (a_0+1,a_1+1,a_2,a_3) \bmod 2$$

$$(a_0,a_1,a_2,a_3) \xrightarrow{FG} (a_0,a_1,a_2,a_3)+(1,0,1,0) = (a_0+1,a_1,a_2+1,a_3) \bmod 2$$

$$(a_0,a_1,a_2,a_3) \xrightarrow{FC} (a_0,a_1,a_2,a_3)+(1,0,0,1) = (a_0+1,a_1,a_2,a_3+1) \bmod 2$$

$$(a_0,a_1,a_2,a_3) \xrightarrow{F} (a_0,a_1,a_2,a_3)+(1,0,0,0) = (a_0+1,a_1,a_2,a_3) \bmod 2$$

Farmer River Crossing Problem

WHAT IS A VALID MOVE?

$(1,1,1,0) \xrightarrow{FW} (0,0,1,0)$ VALID!

$(1,1,1,0) \xrightarrow{FC} (0,1,1,1)$ INVALID! (FARMER AND CABBAGE ON OPPOSITE SIDES!)

$(a_0, a_1, a_2, a_3) \xrightarrow{FW}$ Valid if $a_0 = a_1$

$(a_0, a_1, a_2, a_3) \xrightarrow{FG}$ Valid if $a_0 = a_2$

$(a_0, a_1, a_2, a_3) \rightarrow$ Valid if $a_0 = a_3$

$\xrightarrow{F}$ is always valid (Farmer always has the boat → Really the goat should be allowed to steer the boat :)

What is a valid state (No Goat or cabbage eaten)?

$(1,1,1,0)$ Valid! $(1,0,0,1)$ Invalid!

(a_0, a_1, a_2, a_3) Is Invalid when
$\begin{cases} a_0 \neq a_1 \text{ (Farmer and wolf apart)} \\ a_1 = a_2 \text{ (wolf and Goat together)} \\ \text{OR} \\ a_0 \neq a_2 \text{ (Farmer and Goat apart)} \\ a_2 = a_3 \text{ (Goat and cabbage together)} \end{cases}$

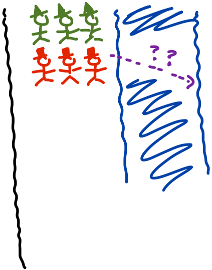
Missionaries and Cannibals Problem

Problem 1.0.2.

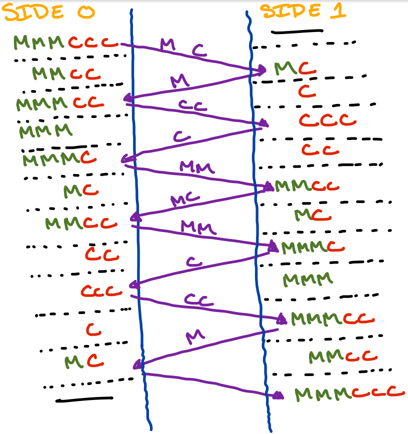
On one bank of a river are three missionaries and three cannibals. There is one boat available that can hold up to two people and that they would like to use to cross the river. If the cannibals ever outnumber the missionaries on either of the river's banks, the missionaries will get eaten. How can the boat be used to safely carry all the missionaries and cannibals across the river?

 = MISSIONARY (OR A WIZARD)

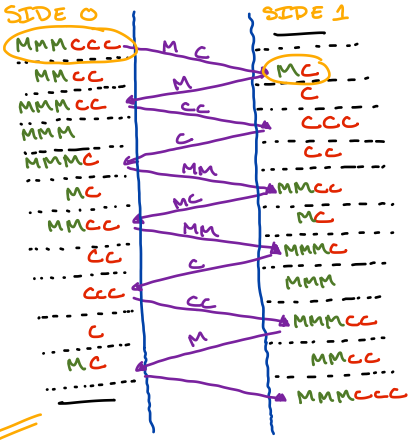
 = CANNIBAL (BECAUSE CANNIBALS WEAR TOP HATS?)



Missionaries and Cannibals Problem



Missionaries and Cannibals Problem



How To Encode?

OPTION 1: $(\underbrace{M_1, M_2, M_3, C_1, C_2, C_3}_{\text{on side 0 or side 1 (each in } \{0,1\})})$

1. (0,0,0,0,0,0) → (1,0,0,1,0,0)

[Note—each M and C is treated uniquely!]

OPTION 2: $(\underbrace{M \text{ SIDE } 0}_{\in\{0,1,2,3\}}, \underbrace{C \text{ SIDE } 0}_{\in\{0,1,2,3\}})$

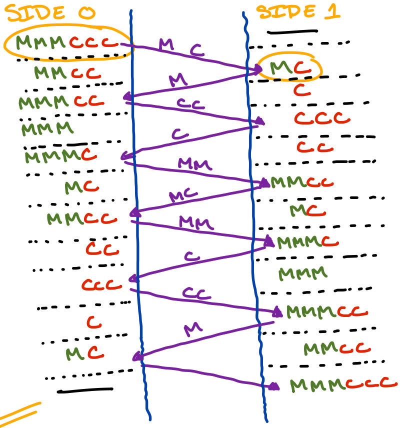
[NOTE THIS TELLS US HOW MANY ARE ON SIDE 1! (TOTAL OF 3 EACH SO, 3 − M SIDE 0 GIVES M SIDE 1)]

1. (3,3) → (2,2)
2. (2,2) → (3,2)

⋮

0: (3,3) → (2,2) → (3,2) → (3,0) → (3,1) → (1,1) → (2,2) → (0,2) → (0,3) → (0,1) → (1,1) → (0,0)

Missionaries and Cannibals Problem



How To Encode?

OPTION 1: $(\underbrace{M_1, M_2, M_3, C_1, C_2, C_3}_{\text{ON SIDE 0 OR SIDE 1 (each in } \{0,1\})})$

1. $(0,0,0,0,0,0) \rightarrow (1,0,0,1,0,0)$

[Note - each M and C is treated uniquely!]

OPTION 2: $(\underbrace{M\ \text{SIDE } 0}_{\in \{0,1,2,3\}}, \underbrace{C\ \text{SIDE } 0}_{\in \{0,1,2,3\}})$

[NOTE THIS TELLS US HOW MANY ARE ON SIDE 1! (TOTAL OF 3 EACH SO, 3 − M SIDE 0 GIVES M SIDE 1)]

1. $(3,3) \rightarrow (2,2)$

[SIDE 1: $(3-2, 3-2) = (1, 1)$]

0: $(3,3) \rightarrow (2,2) \rightarrow (3,2) \rightarrow (3,0) \rightarrow (3,1) \rightarrow (1,1) \rightarrow (2,2) \rightarrow (0,2) \rightarrow (0,3) \rightarrow (0,1) \rightarrow (1,1) \rightarrow (0,0)$

1: $(0,0) \rightarrow (1,1) \rightarrow (0,1) \rightarrow (0,3) \rightarrow (0,2) \rightarrow (2,2) \rightarrow (1,1) \rightarrow (3,1) \rightarrow (3,0) \rightarrow (3,2) \rightarrow (2,2) \rightarrow (3,3)$

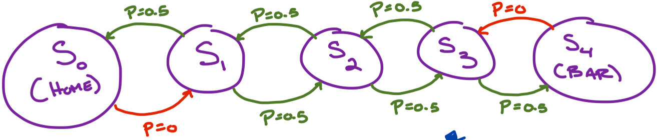
Missionaries and Cannibals Problem

Missionaries and Cannibals Problem

Drunkard's Walk

Example 1.0.3.

A drunkard walks along a four-block stretch of Park Avenue. If they are at corner 1, 2, or 3, then they walks forward with probability 0.5 (p) or backwards with probability 0.5 ($1 - p$). The process continues until they reaches corner 4 (Bar) or corner 0 (Home). If the drunkard reaches either home or the bar, they stays there.



$$\begin{cases} 1 = \text{GO FOWARD}(S_i \to S_{i+1}) \\ 0 = \text{GO BACKWARD}(S_i \to S_{i-1}) \end{cases}$$

where will the drunkard end up?

Example 1.0.3.

A drunkard walks along a four-block stretch of Park Avenue. If they are at corner 1, 2, or 3, then they walks forward with probability 0.5 (p) or backwards with probability 0.5 ($1 - p$). The process continues until they reaches corner 4 (Bar) or corner 0 (Home). If the drunkard reaches either home or the bar, they stays there.

Question 1.0.4.

a. What is the probability that the process will eventually reach an absorbing state (Home or Bar)?

b. What is the probability that the process will end up in a given absorbing state?

c. On the average, how long will it take for the process to be absorbed?

d. On the average, how many times will the process be in each transient state?

Example 1.0.3.

A drunkard walks along a four-block stretch of Park Avenue. If they are at corner 1, 2, or 3, then they walks forward with probability 0.5 (p) or backwards with probability 0.5 ($1 - p$). The process continues until they reaches corner 4 (Bar) or corner 0 (Home). If the drunkard reaches either home or the bar, they stays there.

Question 1.0.4.

a. What is the probability that the process will eventually reach an absorbing state (Home or Bar)?

The drunkard will 'eventually' end up at home or the bar. (Theoretically could bounce from S_1 to S_2 forever)

b. What is the probability that the process will end up in a given absorbing state?

START AT 1: 0.75 Home(0) and 0.25 Bar(4)

START AT 2: 0.50 Home (0) and 0.50 Bar(4)

START AT 3: 0.25 Home (0) and 0.75 Bar(4)

c. On the average, how long will it take for the process to be absorbed?

START AT 2: Expect 4 moves to absorption

START AT 1 OR 3: Expect 3 moves to absorption

d. On the average, how many times will the process be in each transient state?

Remark 1.0.5.

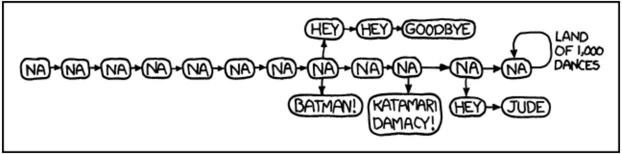
A practical finite state automaton is basically four things:

1. A bunch of functions, or things that need to get done.
2. A bunch of events, or reasons to call these functions.
3. Some piece of data that tracks the ‘state’ this bunch of functions is in.
4. Code inside the functions that says how to ‘transition’ or ‘change’ into the next state for further processing.

Remark 1.0.5.

A practical finite state automaton is basically four things:

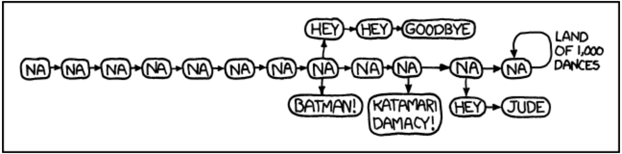
1. A bunch of functions, or things that need to get done.
2. A bunch of events, or reasons to call these functions.
3. Some piece of data that tracks the 'state' this bunch of functions is in.
4. Code inside the functions that says how to 'transition' or 'change' into the next state for further processing.

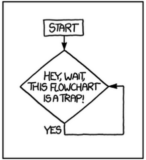


Remark 1.0.5.

A practical finite state automaton is basically four things:

1. A bunch of functions, or things that need to get done.
2. A bunch of events, or reasons to call these functions.
3. Some piece of data that tracks the 'state' this bunch of functions is in.
4. Code inside the functions that says how to 'transition' or 'change' into the next state for further processing.





Why Study Automata Theory?

Remark 1.0.6.

Automata theory deals with the definitions and properties of models of computation. These models play a role in several applied areas of computer science. For example, finite automation and context-free grammar are used in the following:

1. Software for designing and checking the behavior of digital circuits.
2. The 'lexical analyzer' of a typical compiler, that is, the compiler component that breaks the input text into logical units, such as identifiers, keywords, and punctuation.
3. Software for scanning large bodies of text, such as collections of Web pages, to find occurrences of words, phrases, or other patterns.
4. Software for verifying systems of all types that have a finite number of distinct states, such as communications protocols or protocols for secure exchange of information.
5. Programming languages and artificial intelligence.

Remark 1.0.7.

- (Computability and Decidability) What can a computer do at all? This study is called 'decidability,' and the problems that can be solved by computer are called 'decidable.'
- (Complexity) What can a computer do efficiently? This study is called 'intractability,' and the problems that can be solved by a computer using no more time than some slowly growing function of the size of the input are called 'tractable.' Often, we take all polynomial functions to be 'slowly growing,' while functions that grow faster than any polynomial are deemed to grow too fast.
- One applied area that has been affected directly by complexity theory is the field of cryptography. Cryptography is unusual because it specifically requires computational problems that are hard, rather than easy, because secret codes should be hard to break without the private key. Complexity theory points cryptographers in the direction of computationally hard problems around which revolutionary cryptosystems and signature algorithms have emerged.