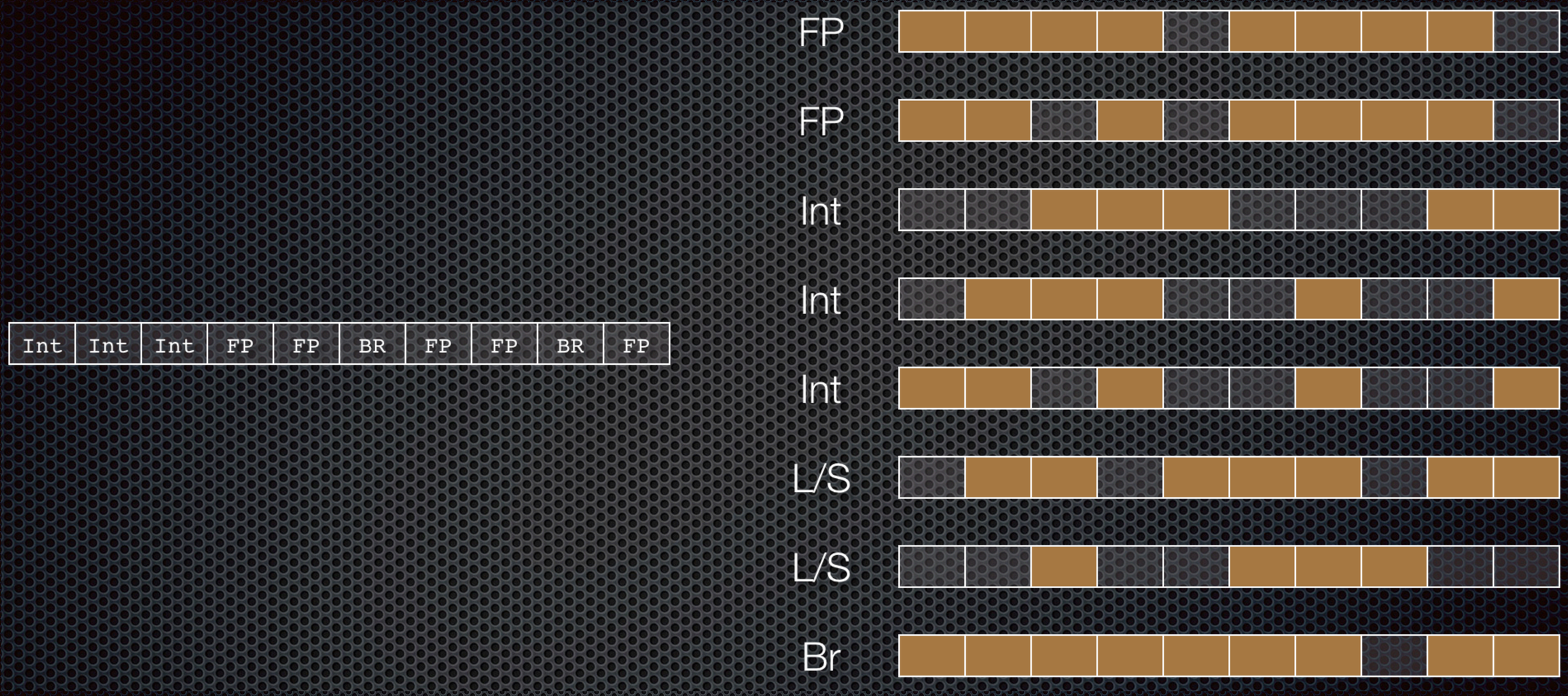
Branch Prediction

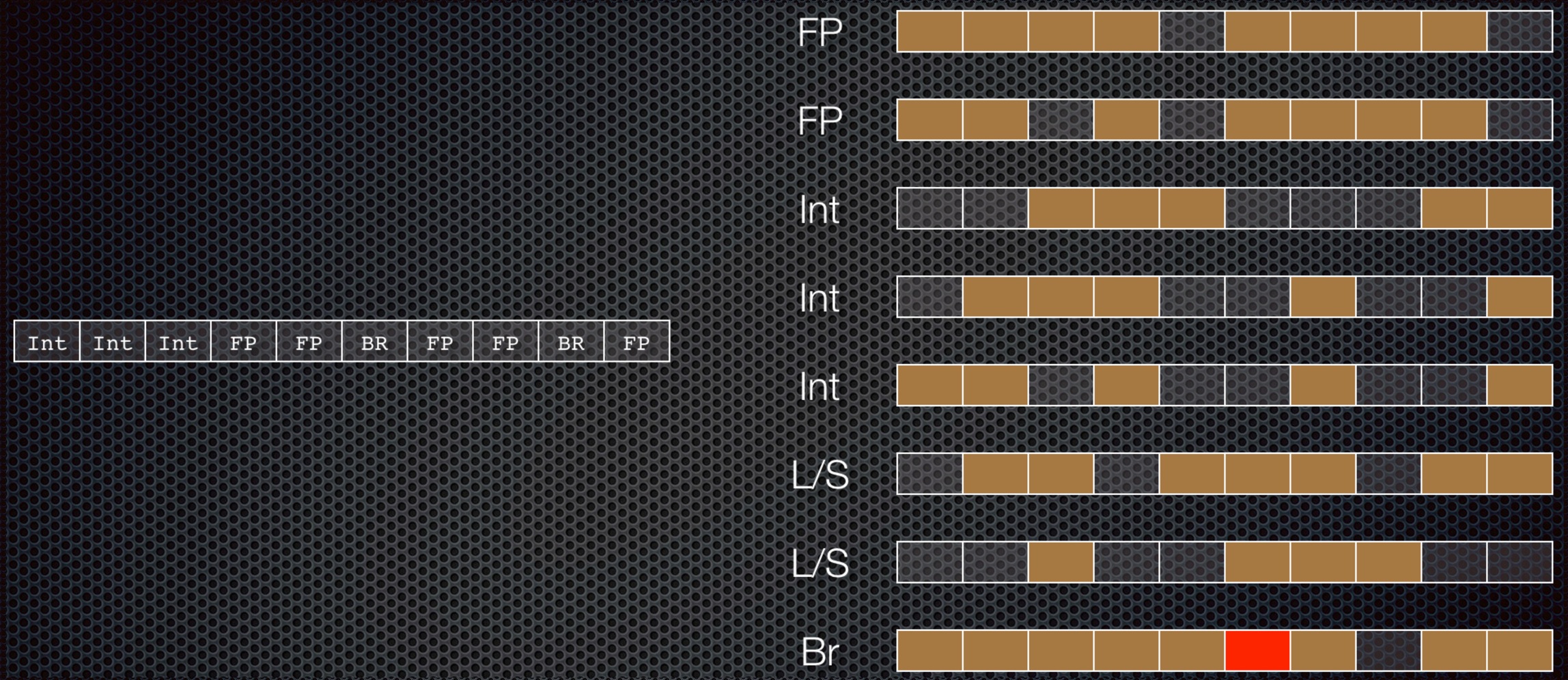
Reducing Pipeline Bubbles

The Problem

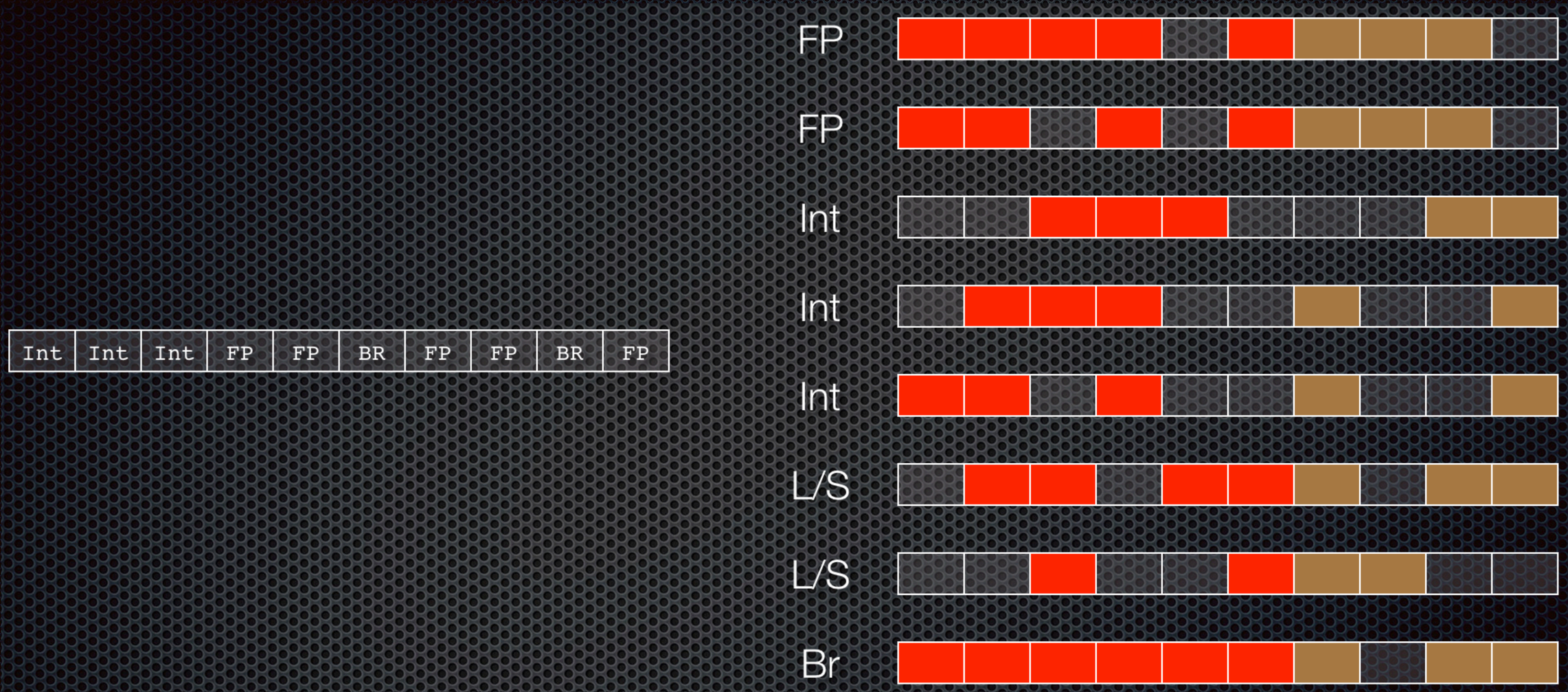
- Branches occur approximately every 5 instructions
- Superscalar pipelines have on the order of 100 instructions in flight
- To get instructions in to the pipelines we have to predict the ways that branches will go
- Branch outcome may not be computed until several stages into the pipeline
- A mispredicted branch means wrong instructions are in the pipes, so they must be flushed



A branch occurs with most issue slots



Misprediction of a branch may be detected late



Many dependent instructions must be flushed

Pipeline Efficiency = $1/(1+P_j\ P_t\ P)$

$P_j = 0.2$ (20% jumps)
$P_t = 0.1$ (10% mispredicts)
$P = 30$ (penalty)
Efficiency = 62.5%

Predictor Context

- Local: Specific to a particular branch, or its history
- Global: Using information from other branches (correlations between branches can be more predictive)
- Hybrid: Selecting different prediction mechanisms for different branches

Static Predictions

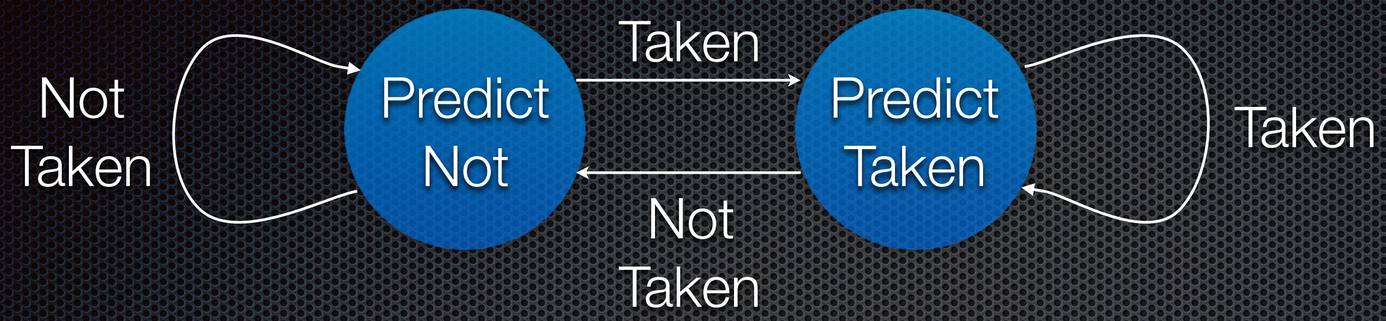
- Predictions that do not depend on run time
- Always-taken (or always not taken)
- Backward taken (loop return) and forward not (loop exit) -- uses sign of relative address in branch instruction

Dynamic Prediction

- Uses information from branch history
- For example, predict same as last outcome
  - Need to record state of the branch

Local Dynamic Branch Predictors

Reducing the Probability that a Branch is “Taken”

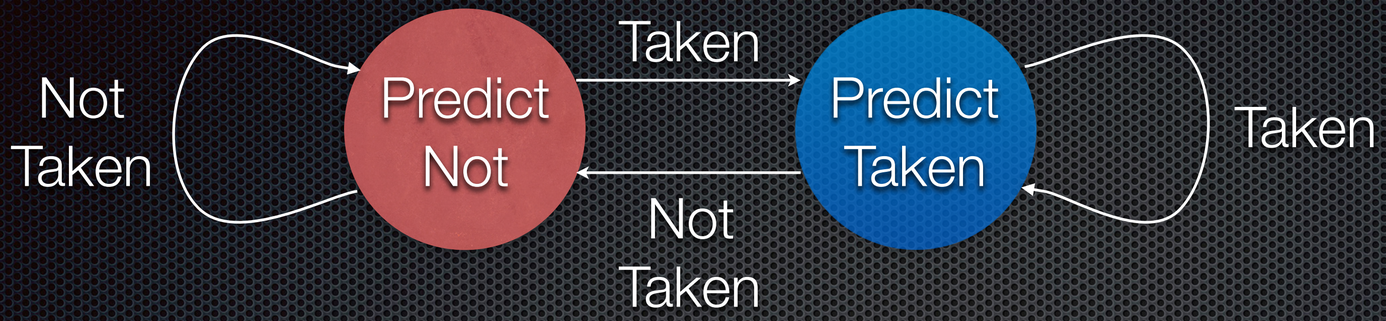


Only one bit is required to remember the last outcome of a branch.

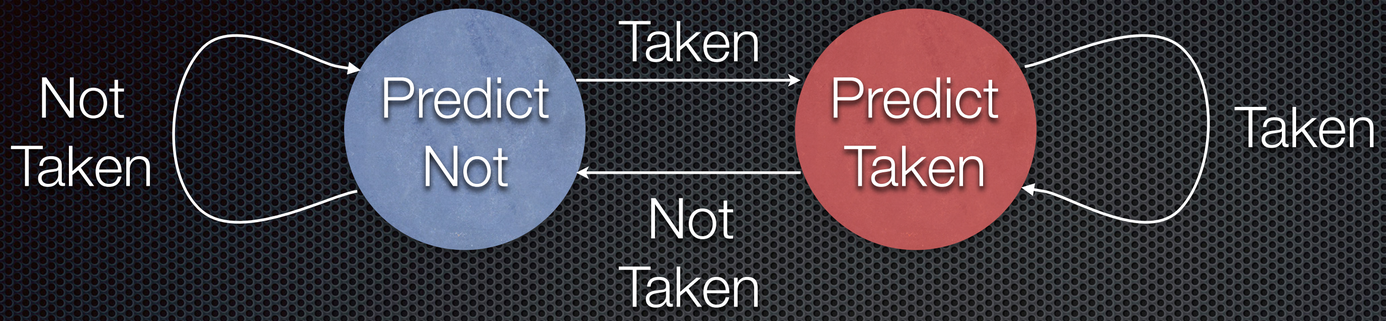
When the branch is taken, set the bit to 1, when it is not taken, set the bit to 0.

When the bit is 1, predict that the branch is taken. When it is 0, predict not taken.

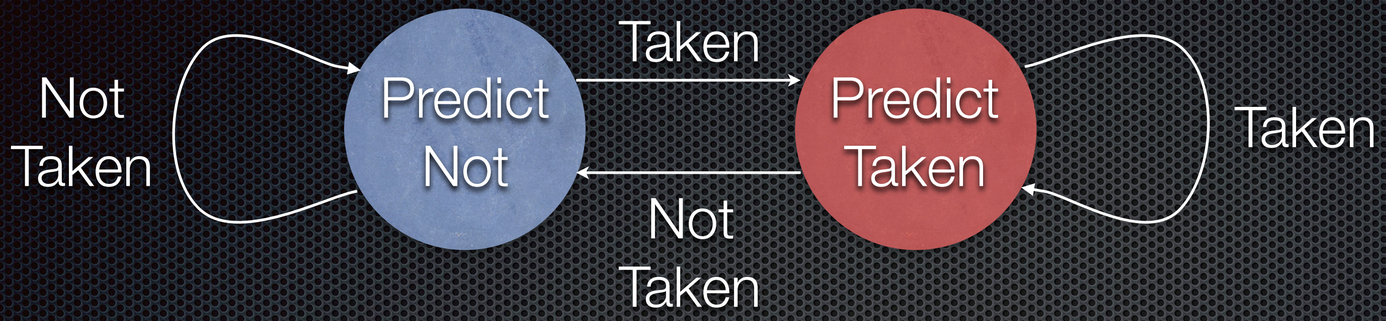
This can be represented as a FSM with two states.



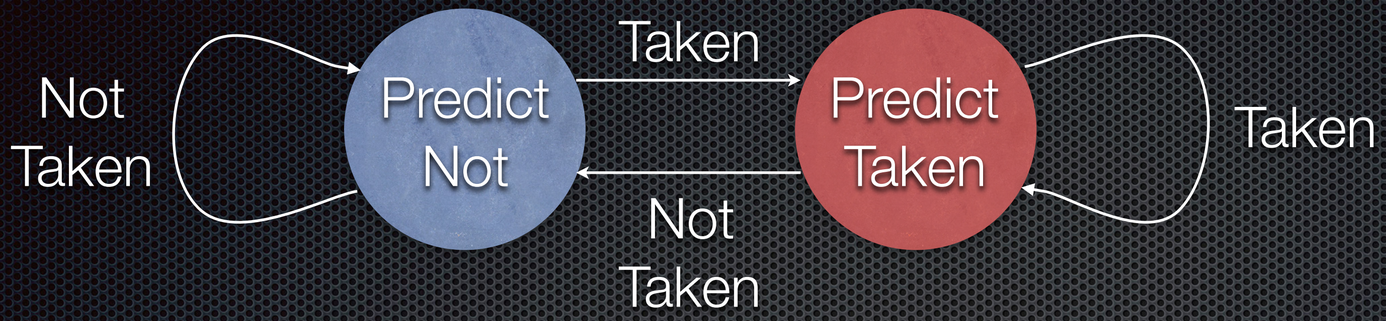

```
Prediction: N
Outcome:    T T T N N T T T N N T T T T
Correct?:   N
```



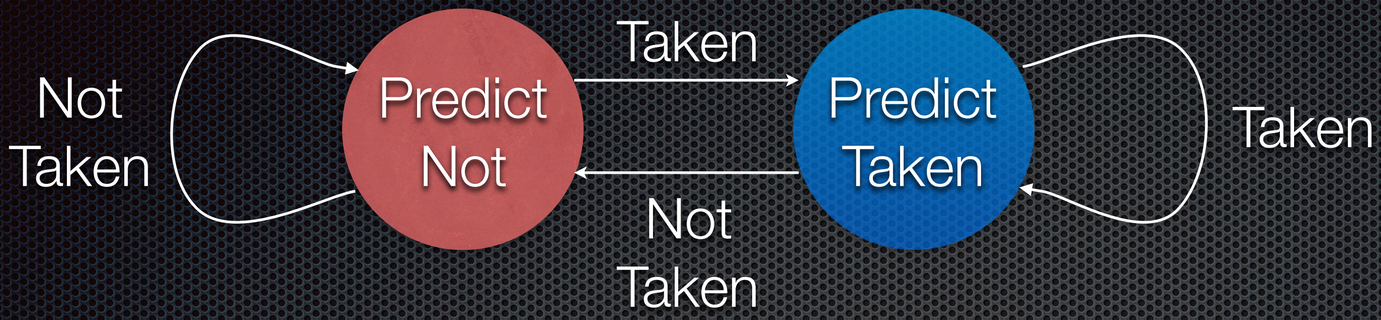

```
Prediction: N T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y
```



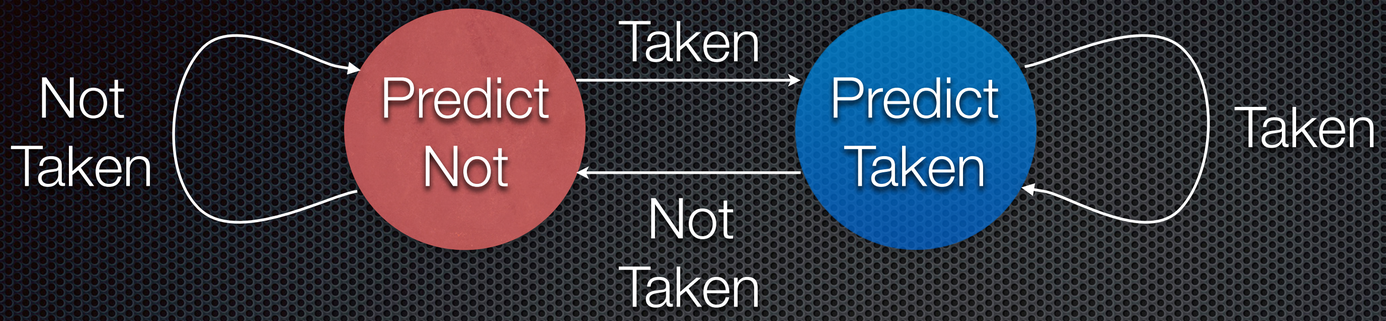

```
Prediction: N T T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y
```



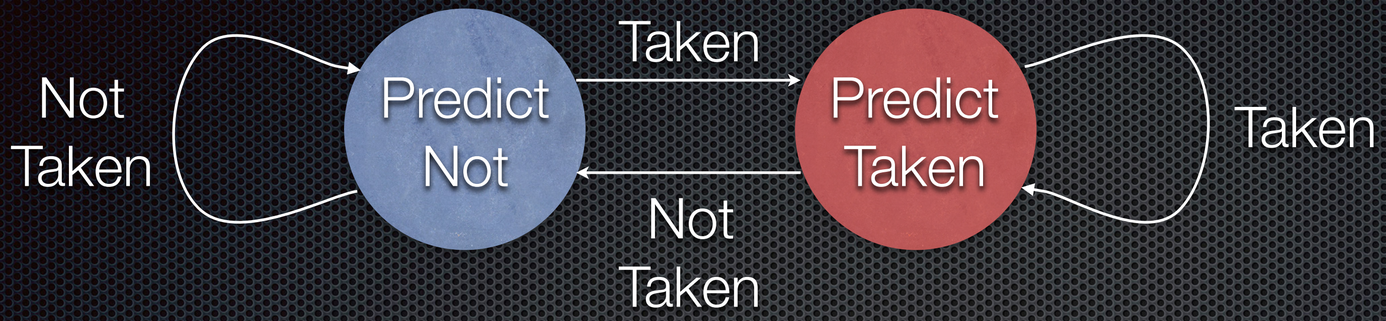

```
Prediction: N T T T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N
```



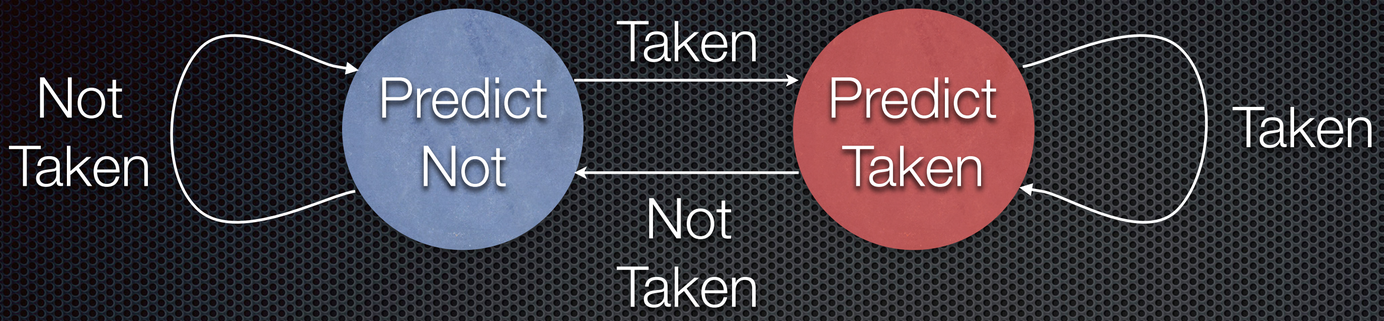

```
Prediction: N T T T N
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y
```



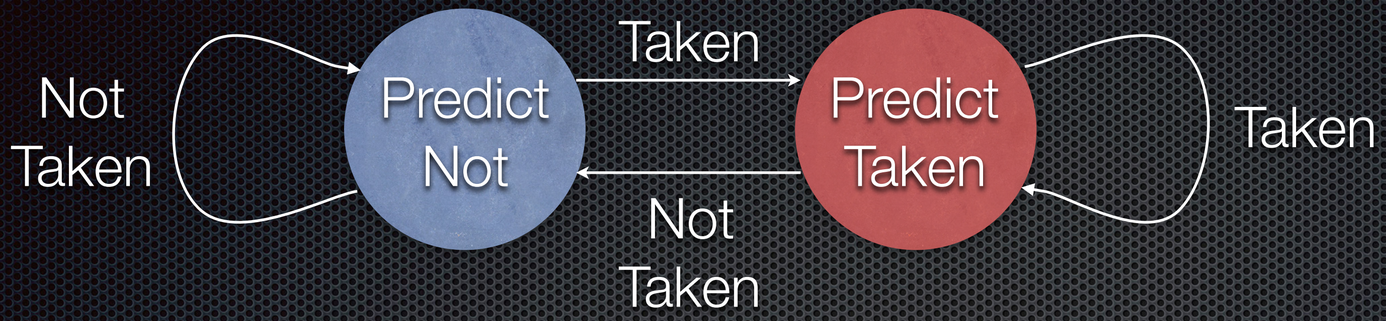

```
Prediction: N T T T N N
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N
```



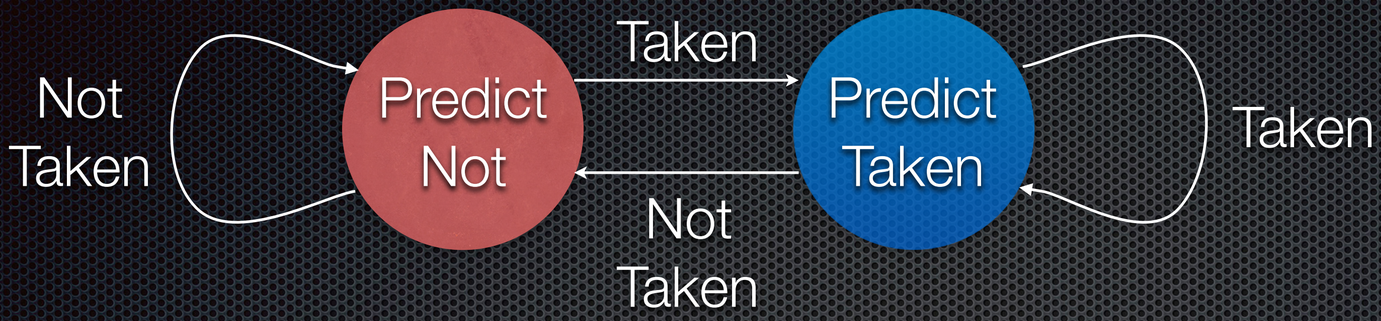

```
Prediction: N T T T N N T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N Y
```



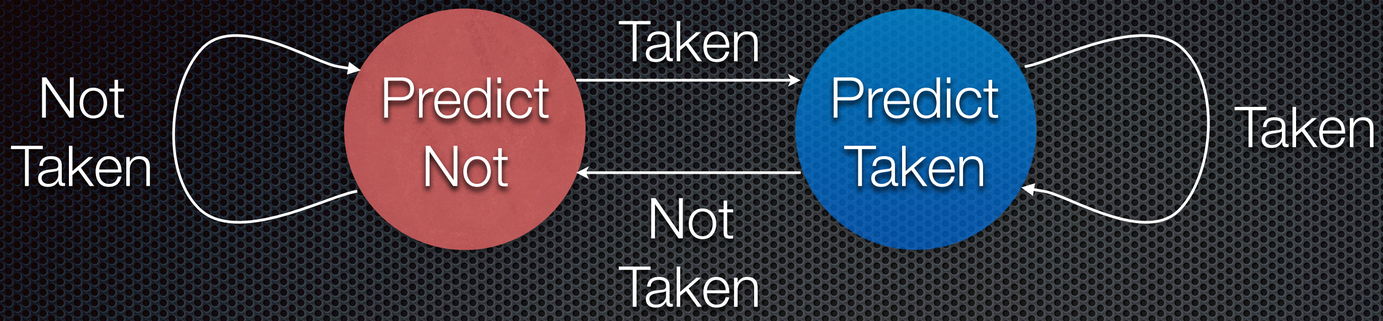

```
Prediction: N T T T N N T T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N Y Y
```



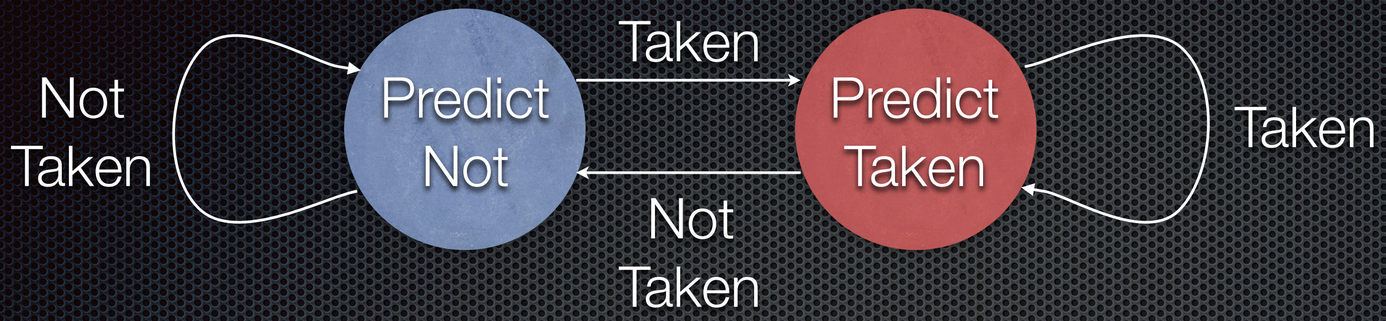

```
Prediction: N T T T N N T T T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N Y Y N
```



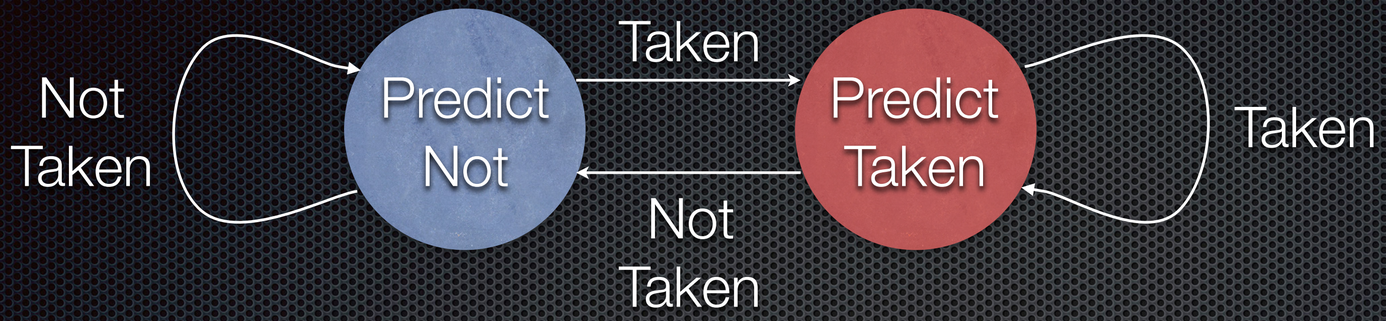

```
Prediction: N T T T N N T T T N
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N Y Y N Y
```



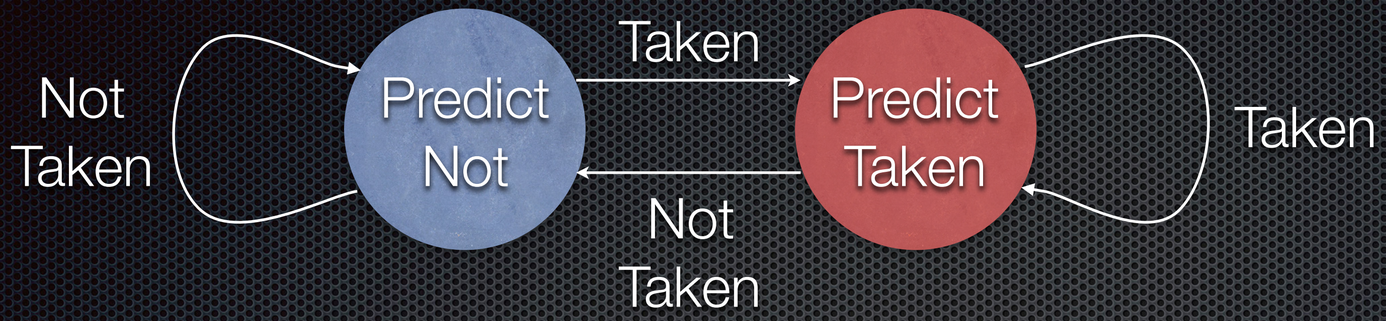

```
Prediction: N T T T N N T T T N N
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N Y Y N Y N
```



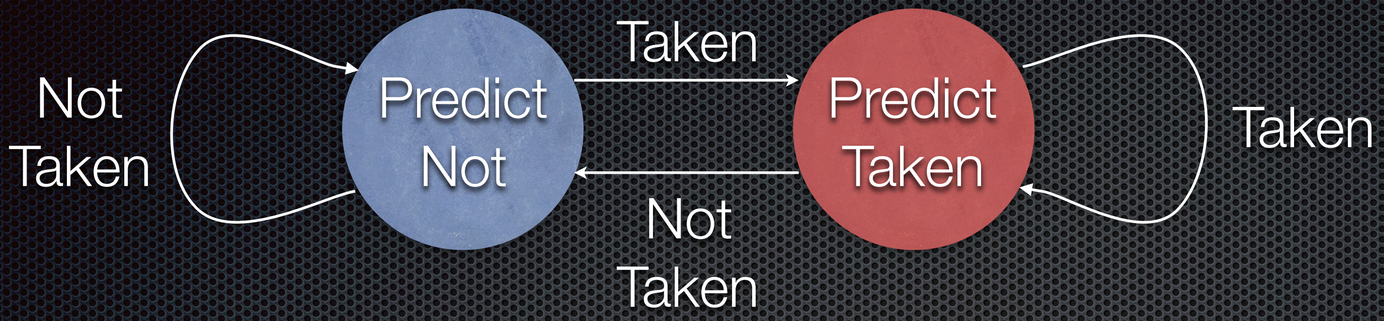

```
Prediction: N T T T N N T T T N N T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N Y Y N Y N Y
```




```
Prediction: N T T T N N T T T N N T Y
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N Y Y N Y N Y Y
```




```
Prediction: N T T T N N T T T N N T Y Y
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N Y Y N Y N Y Y Y
```

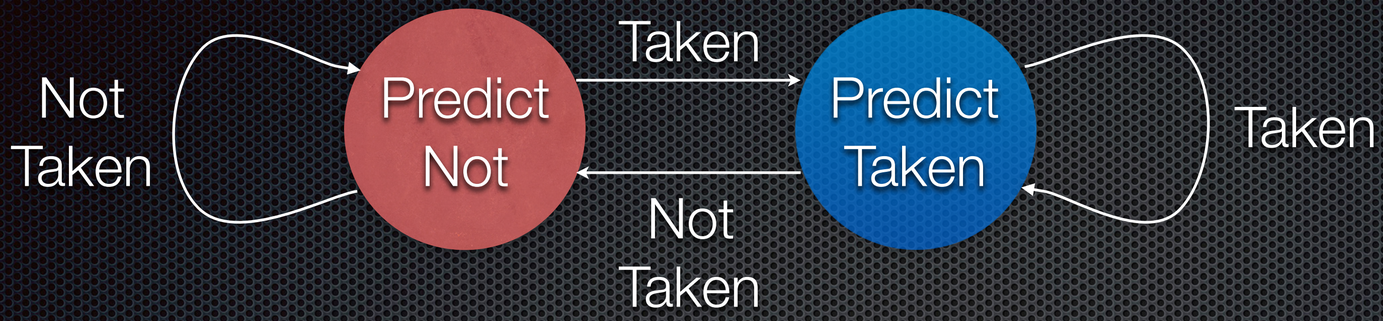



```
Prediction: N T T T N N T T T N N T Y Y
Outcome:    T T T N N T T T N N T T T T
Correct?:   N Y Y N Y N Y Y N Y N Y Y Y
```

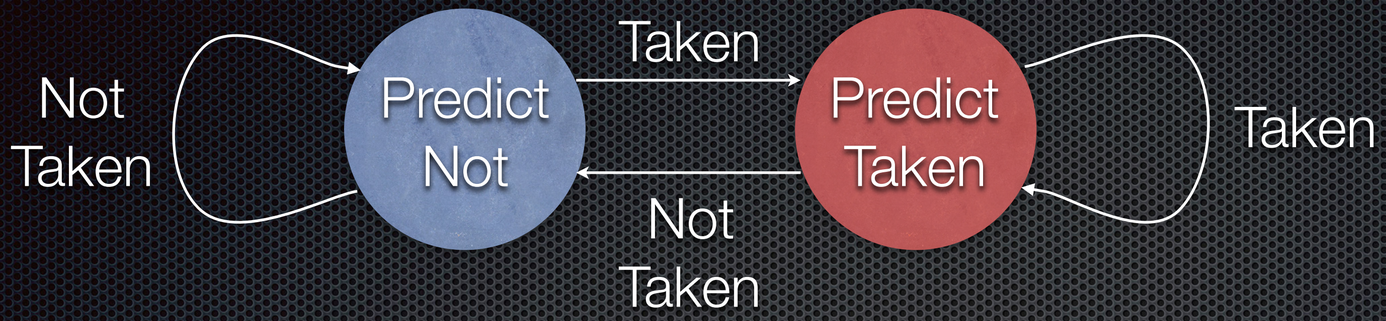
Accuracy = # Correct / # Predictions

Accuracy = 9 / 14 = 64%

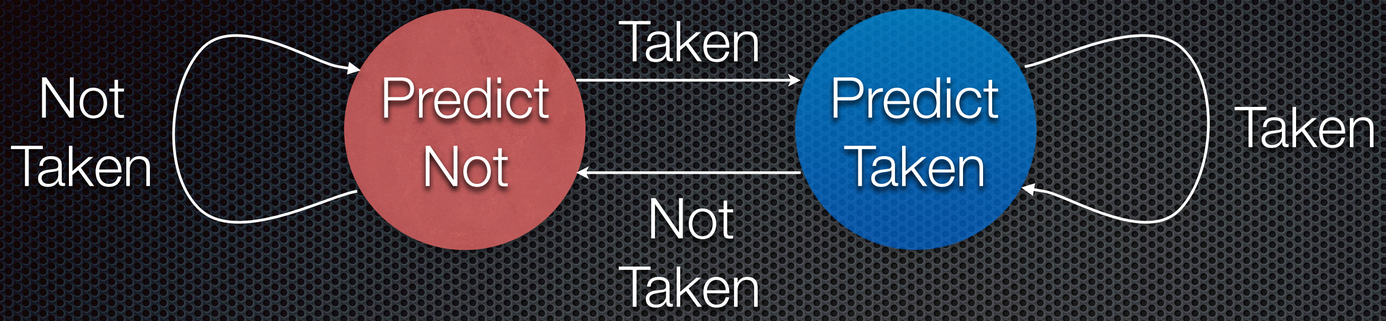
Another Example



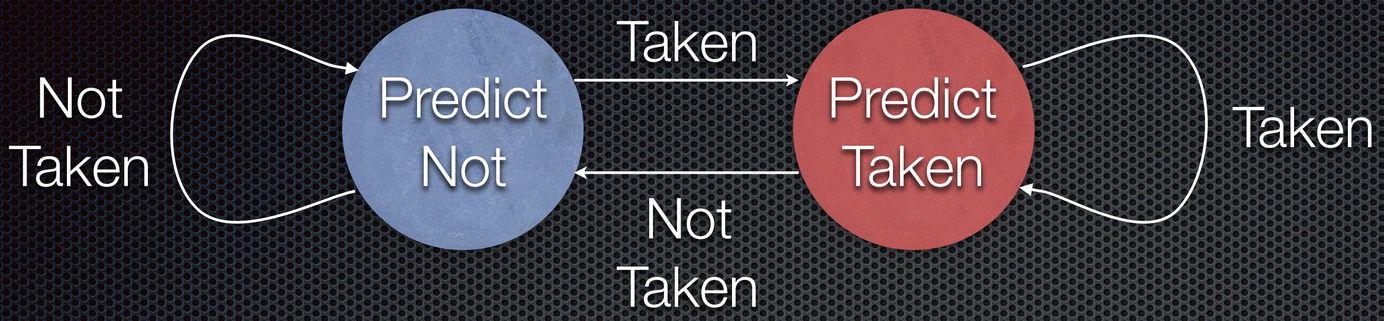

```
Prediction: N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N
```



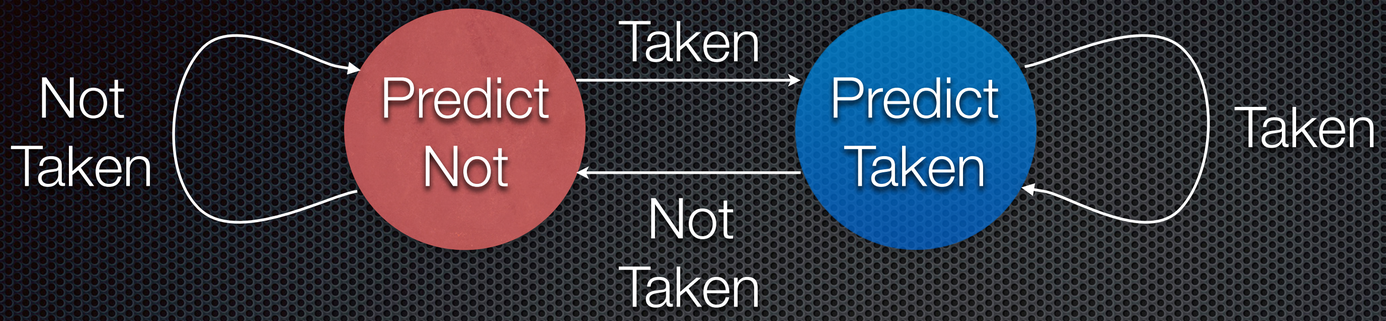

```
Prediction: N T
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N
```



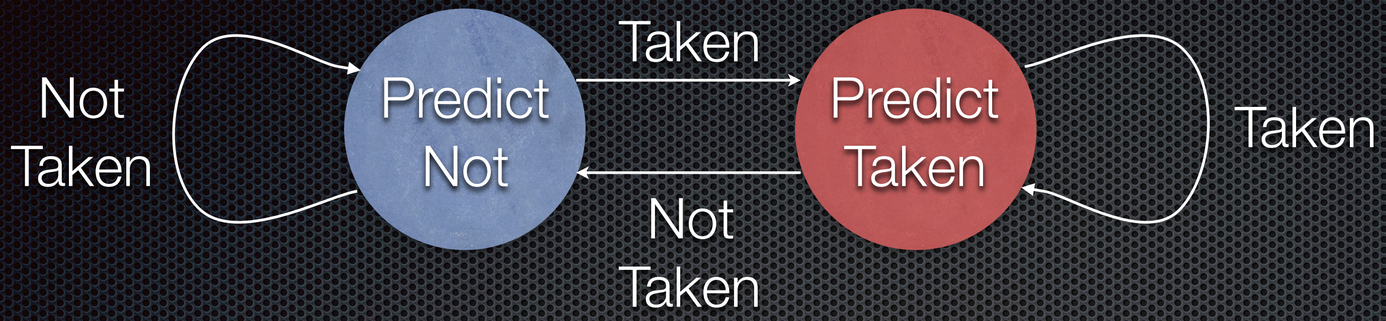

```
Prediction: N T N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N
```



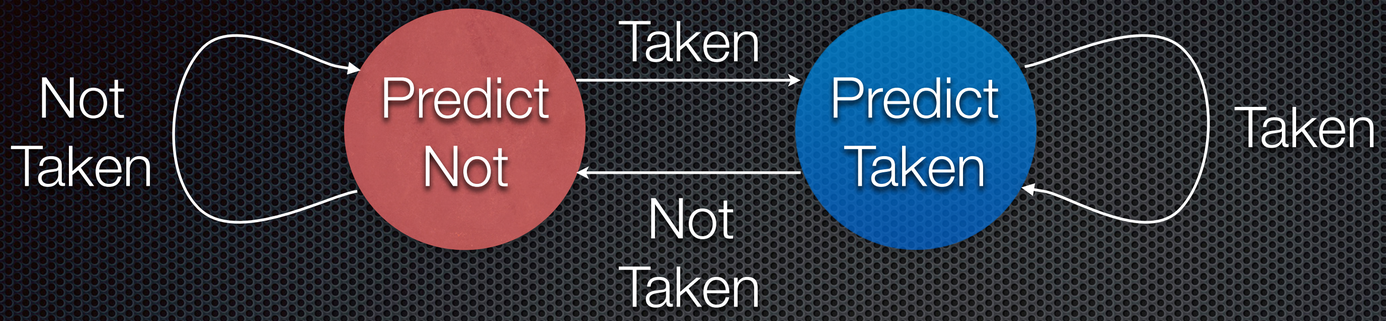

```
Prediction: N T N T
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N
```



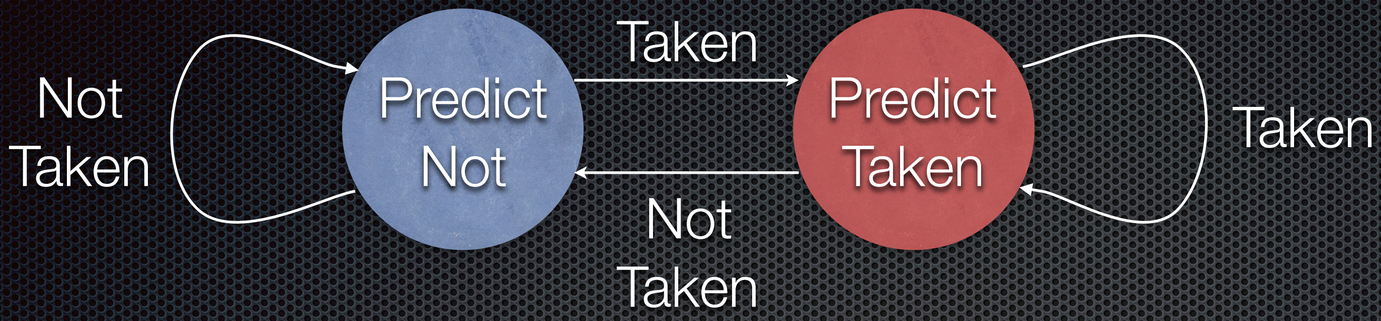

```
Prediction: N T N T N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N
```



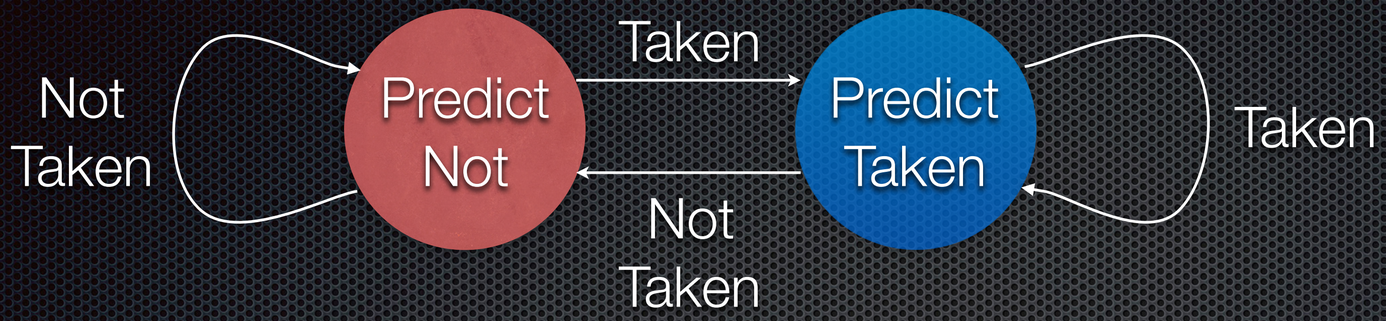

```
Prediction: N T N T N T
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N N
```



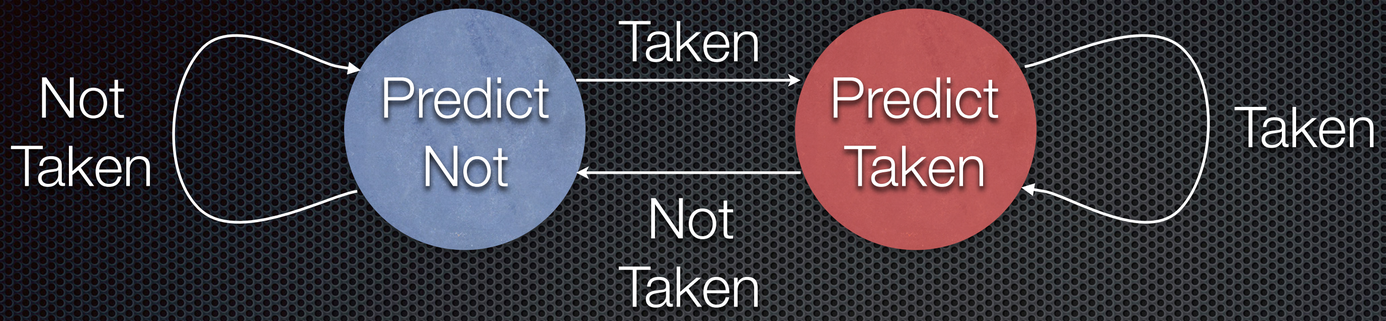

```
Prediction: N T N T N T N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N N N
```



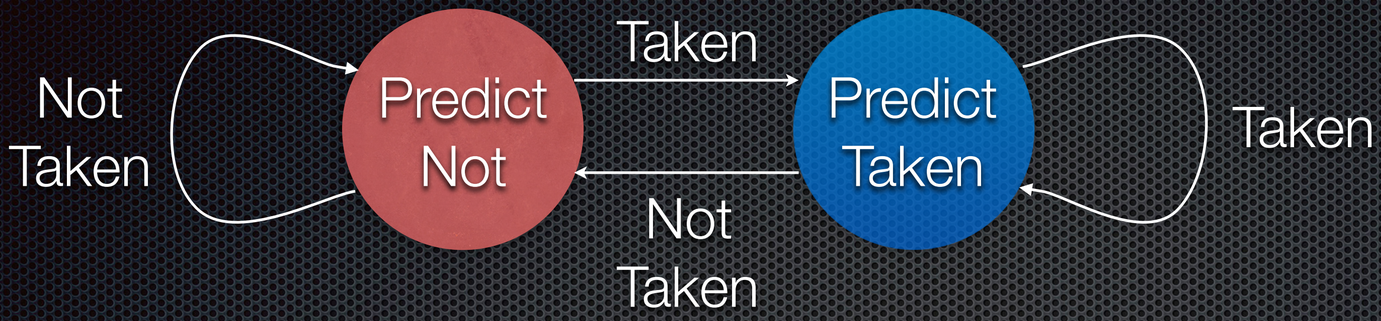

```
Prediction: N T N T N T N T
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N N N N
```



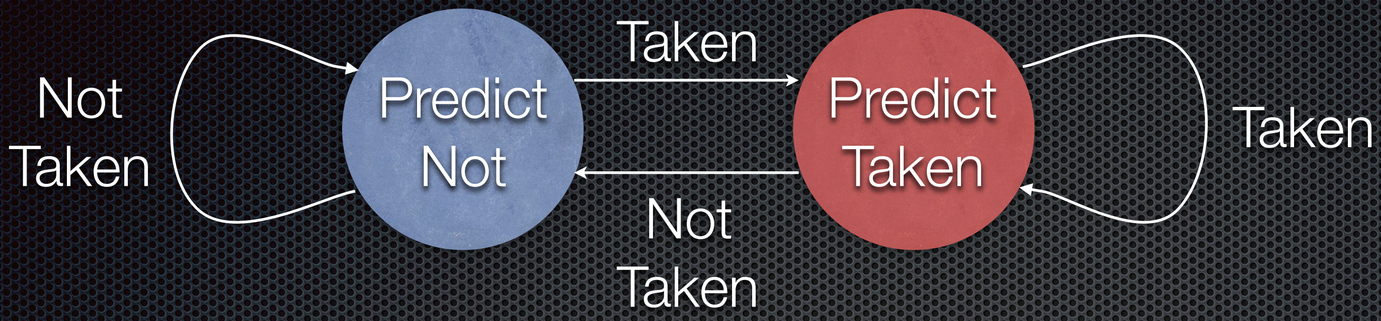

```
Prediction: N T N T N T N T N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N N N N N
```



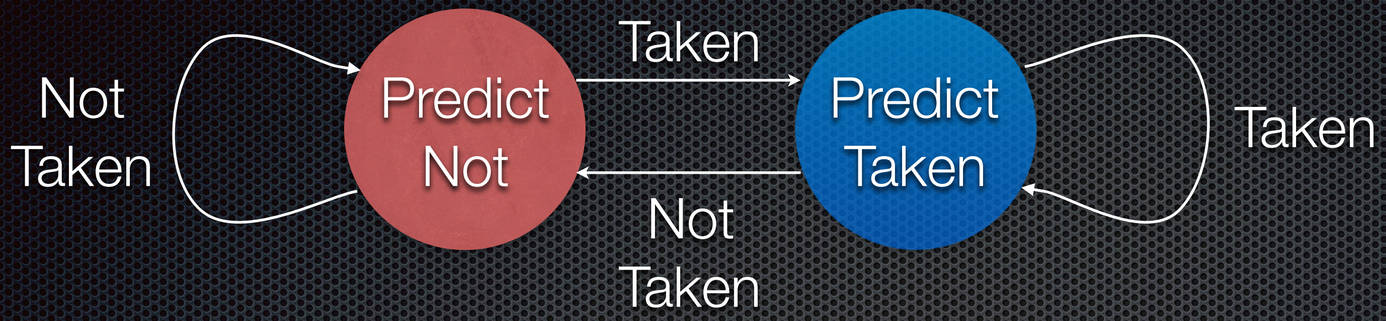

```
Prediction: N T N T N T N T N T
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N N N N N N
```



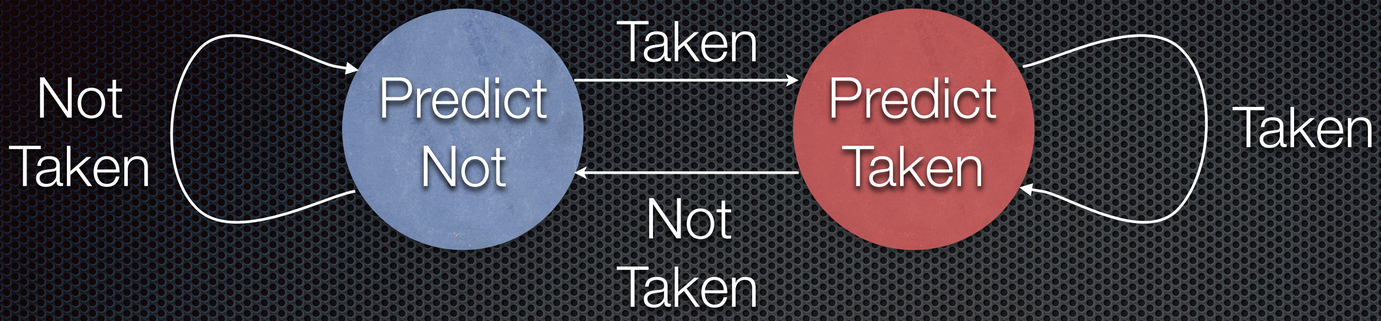

```
Prediction: N T N T N T N T N T N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N N N N N N N
```




```
Prediction: N T N T N T N T N T N T
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N N N N N N N N
```




```
Prediction: N T N T N T N T N T N T N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N N N N N N N N N
```

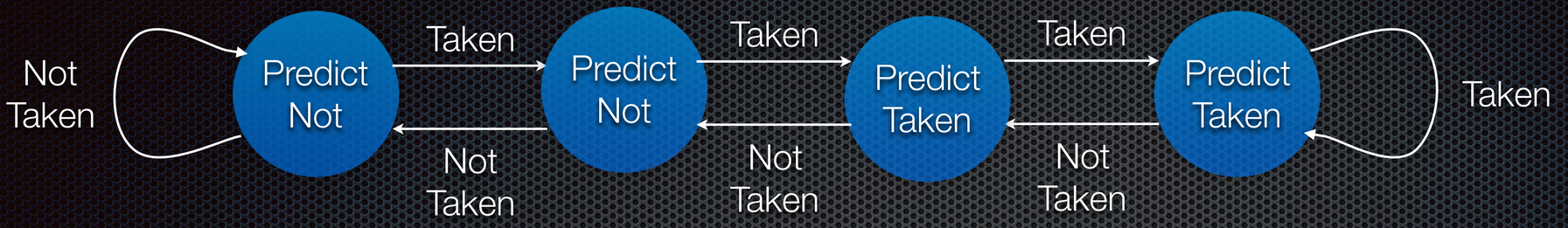



```
Prediction: N T N T N T N T N T N T N T
Outcome:    T N T N T N T N T N T N T N
Correct?:   N N N N N N N N N N N N N N
```

Accuracy = 0 / 14 = 0%

In other words: 100% WRONG!

- How can we fix this?
- Remember more
- Require two mispredicts before changing to the alternate prediction
- Need four states in the FSM (2 bits)

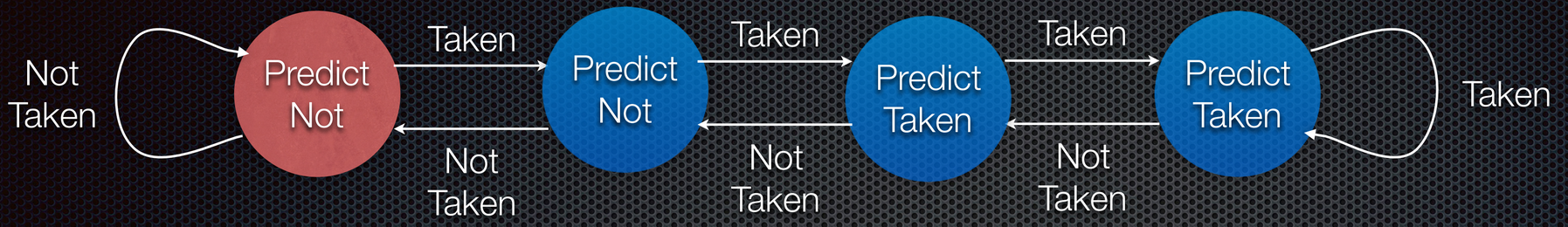


00 01 10 11

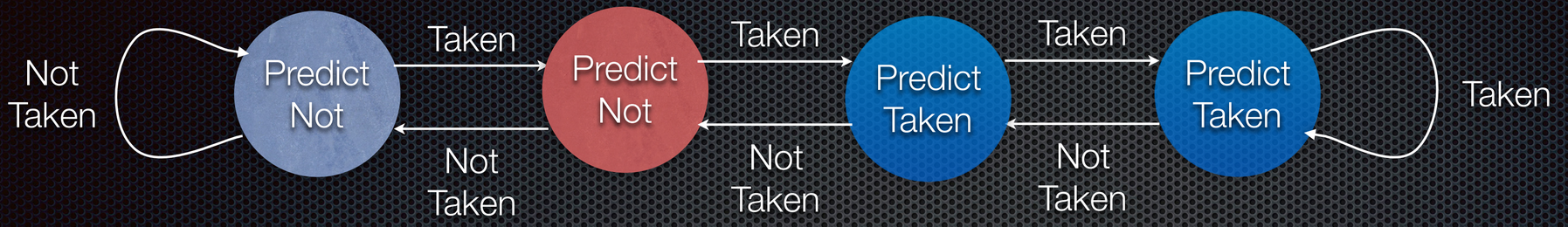
Two-bit Predictor

If the state bits are assigned as shown, then this can be implemented with a two-bit, “saturating,” up-down counter

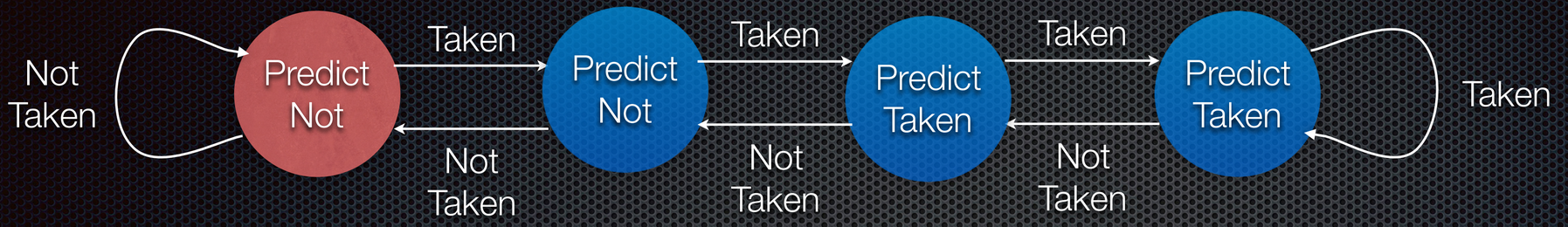
Replay of Last Example



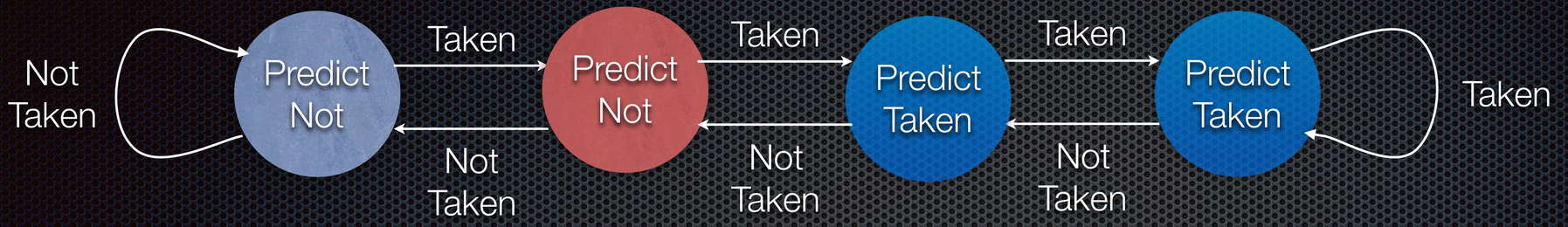

```
Prediction: N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N
```




```
Prediction: N N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N Y
```

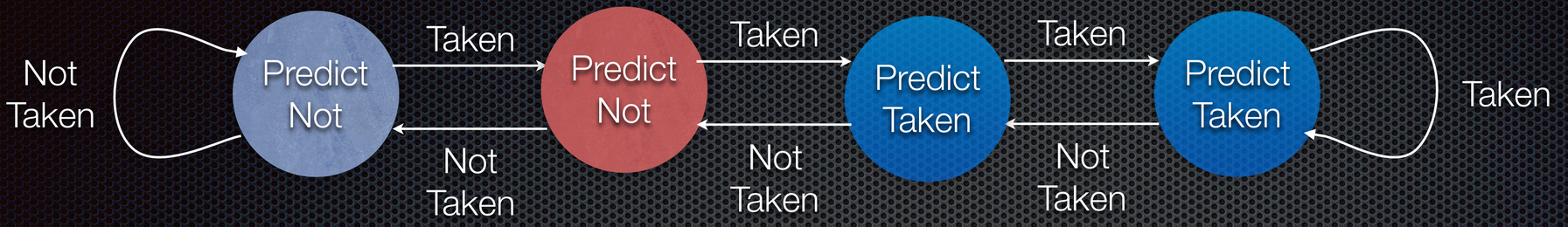



```
Prediction: N N N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N Y N
```




```
Prediction: N N N N
Outcome:    T N T N T N T N T N T N T N
Correct?:   N Y N Y
```

Fast Forward

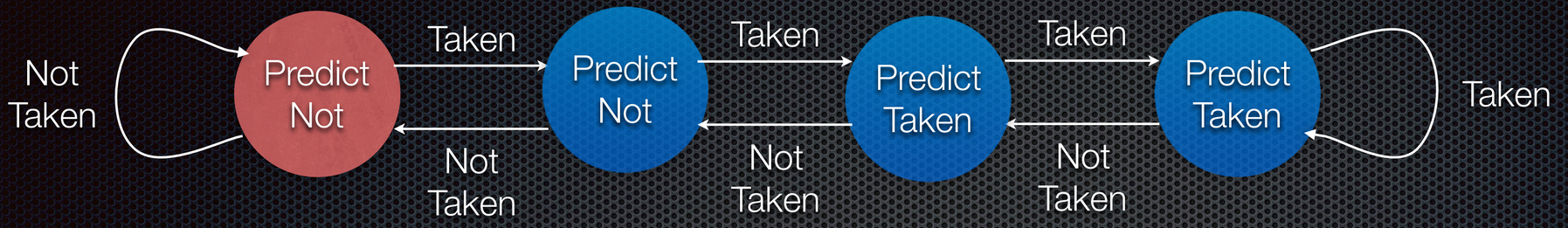



```
Prediction: N N N N N N N N N N N N N N
Outcome:      T N T N T N T N T N T N T N
Correct?:     N Y N Y N Y N Y N Y N Y N Y
```

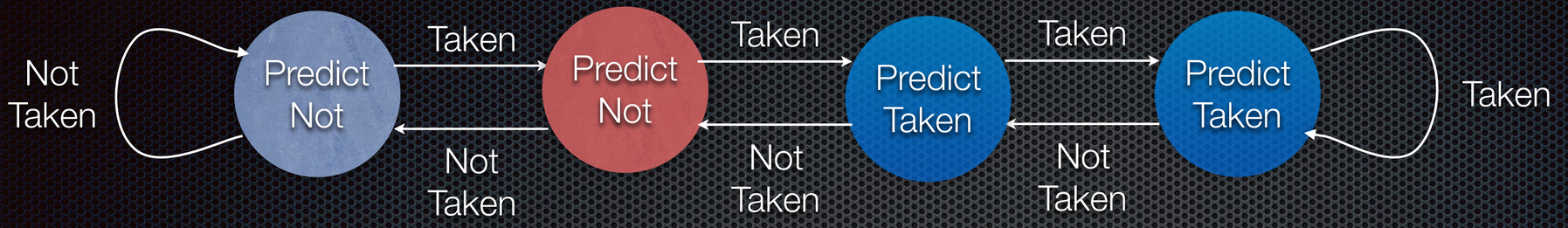
Accuracy = 7 / 14 = 50%

Still not great, but no worse than static prediction

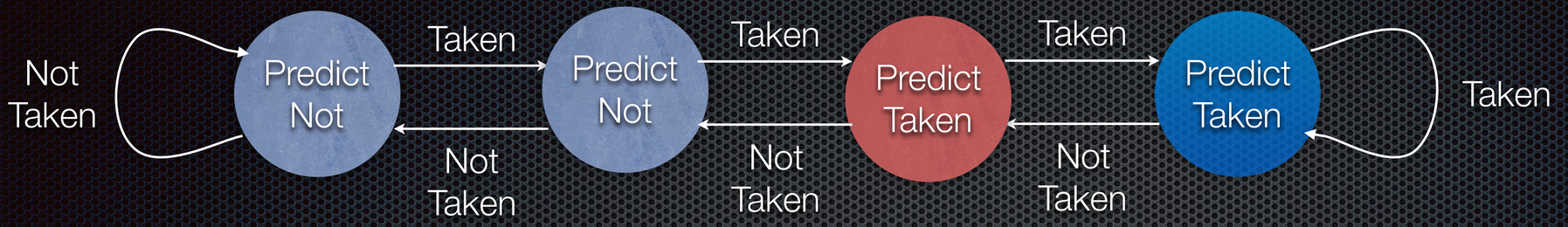
How about the first case?



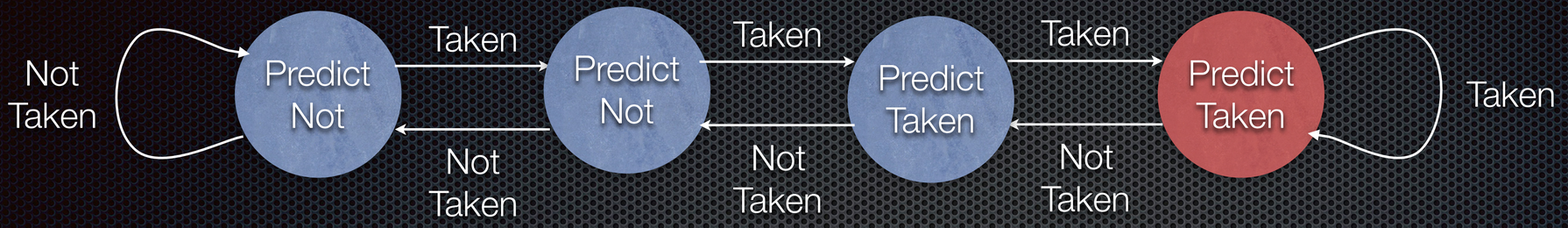

```
Prediction: N
Outcome:    T T T N N T T T N N T T T T
Correct?:   N
```



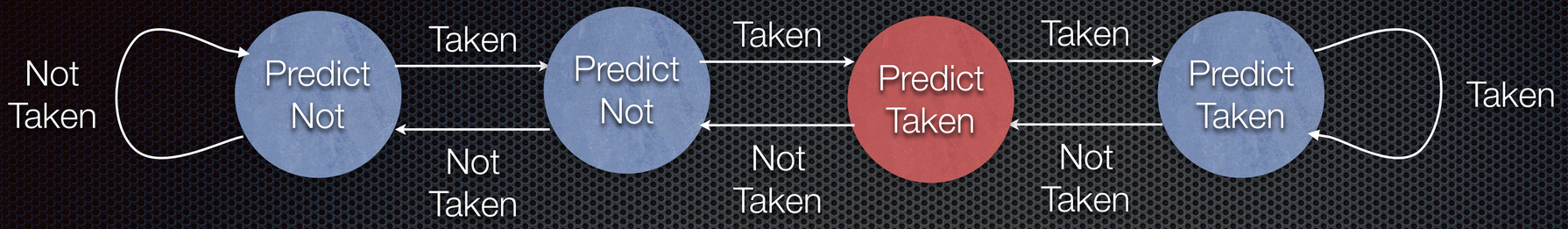

```
Prediction: N N
Outcome:    T T T N N T T T N N T T T T
Correct?:   N N
```



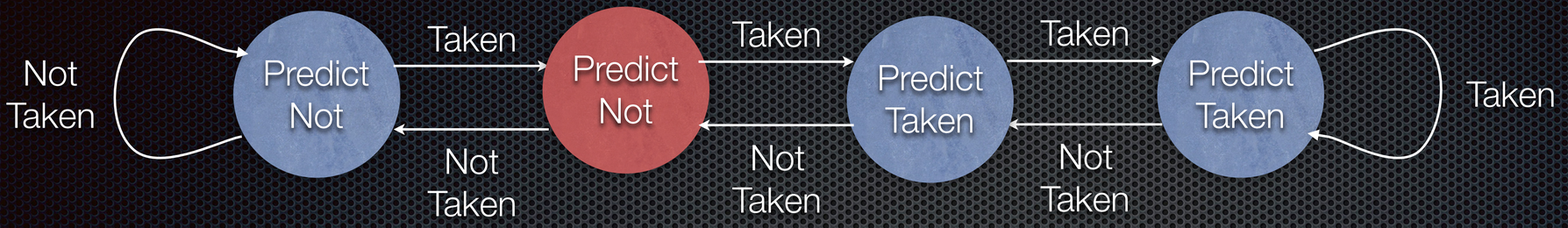

```
Prediction: N N T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N N Y
```



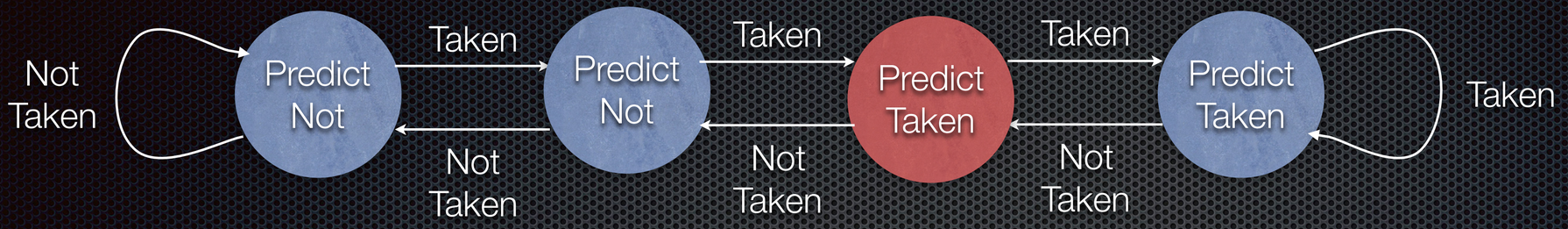

```
Prediction: N N T T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N N Y N
```



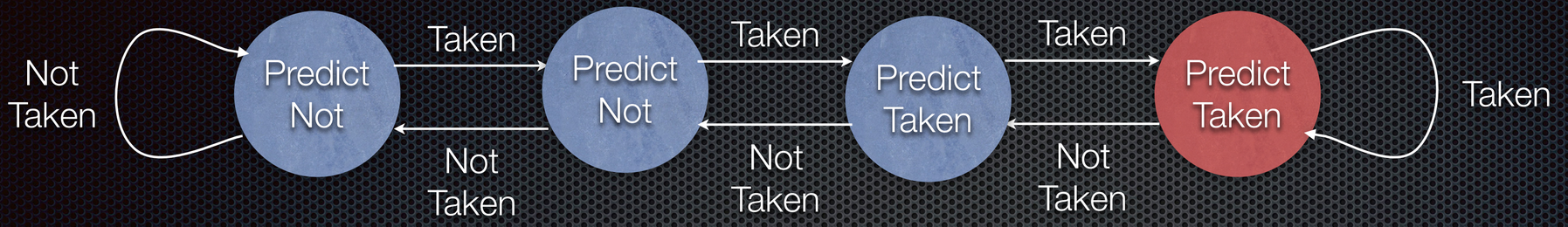

```
Prediction: N N T T T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N N Y N N
```



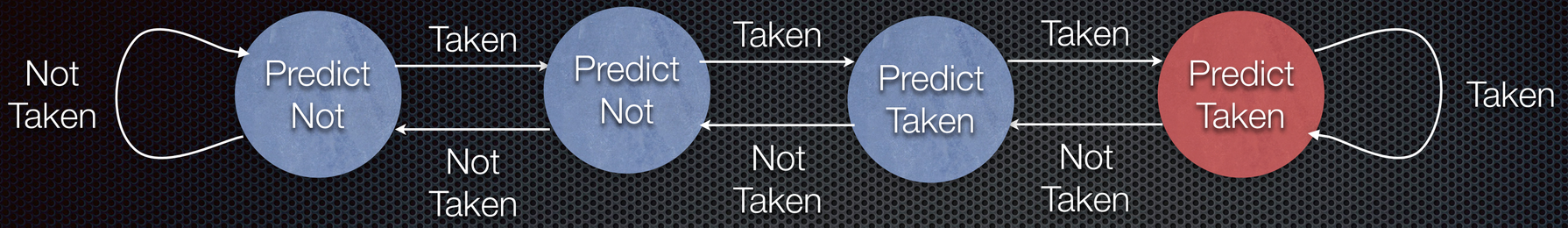

```
Prediction: N N T T T N
Outcome:      T T T N N T T T N N T T T T
Correct?:     N N Y N N N
```



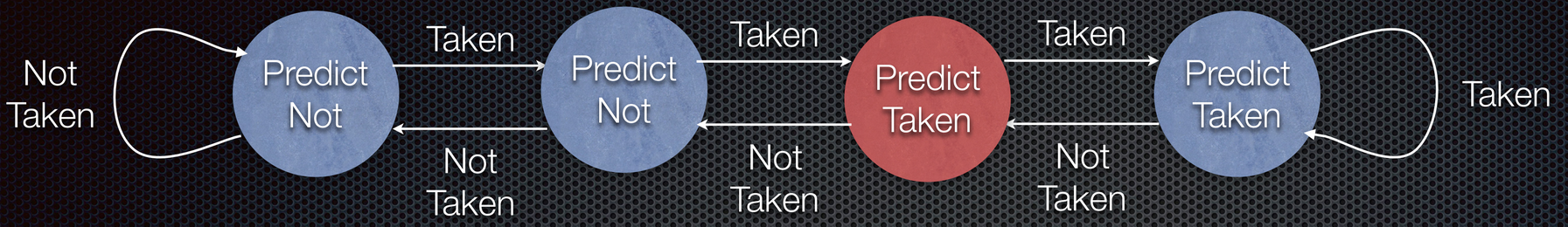

```
Prediction: N N T T T N T
Outcome:      T T T N N T T T N N T T T T
Correct?:     N N Y N N N Y
```



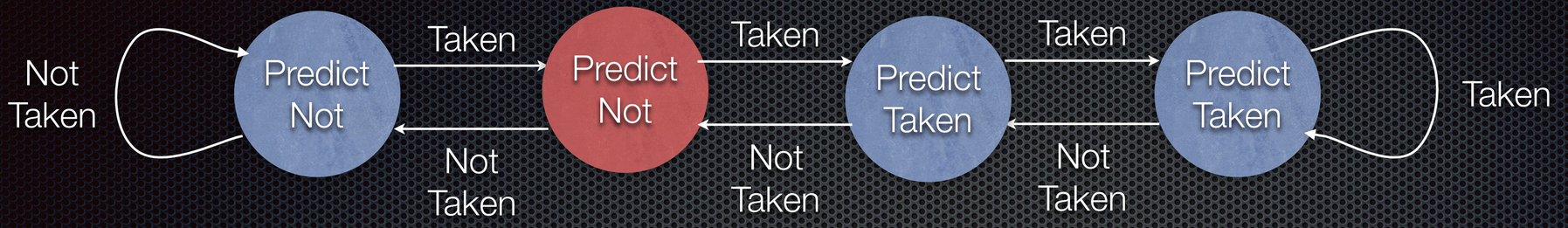

```
Prediction: N N T T T N T T
Outcome:      T T T N N T T T N N T T T T
Correct?:     N N Y N N N Y Y
```



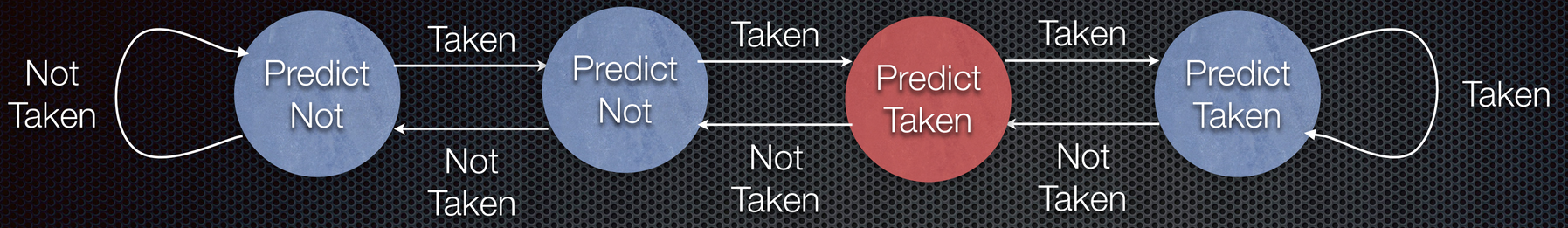

```
Prediction: N N T T T N T T T
Outcome:      T T T N N T T T N N T T T T
Correct?:     N N Y N N N Y Y N
```



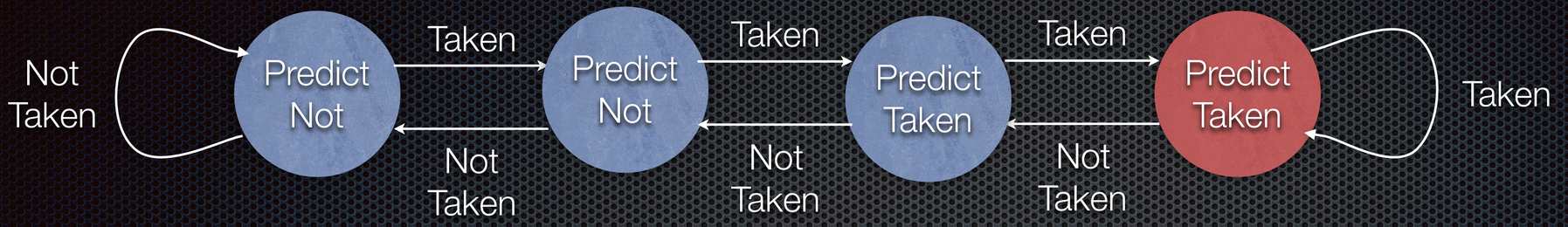

```
Prediction: N N T T T N T T T T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N N Y N N N Y Y N N
```



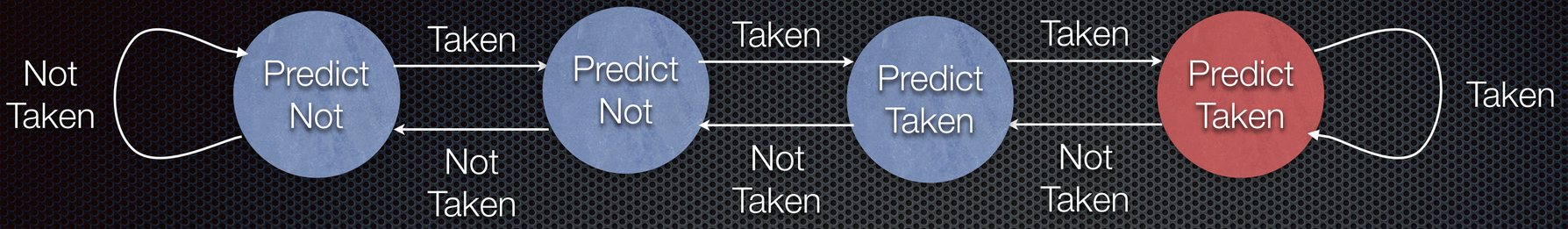

```
Prediction: N N T T T N T T T T N
Outcome:      T T T N N T T T N N T T T T
Correct?:     N N Y N N N Y Y N N N
```




```
Prediction: N N T T T N T T T T N T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N N Y N N N Y Y N N N Y
```




```
Prediction: N N T T T N T T T T N T T
Outcome:    T T T N N T T T N N T T T T
Correct?:   N N Y N N N Y Y N N N Y Y
```



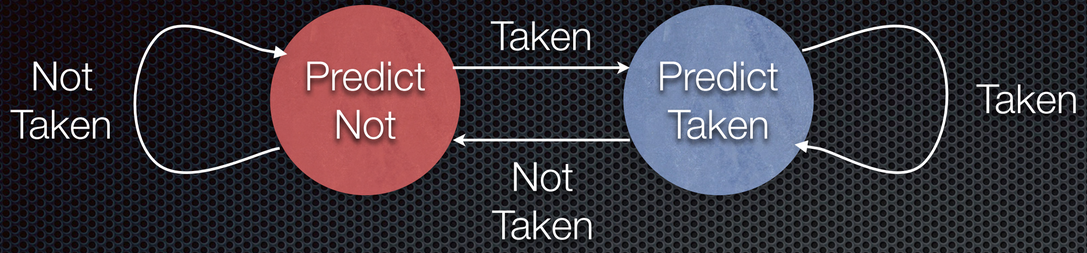

```
Prediction: N N T T T N T T T T N T T T
Outcome:      T T T N N T T T N N T T T T
Correct?:     N N Y N N N Y Y N N N Y Y Y
```

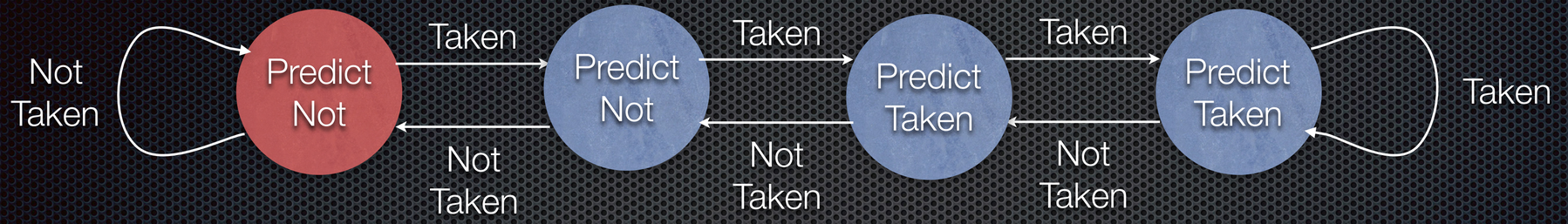
Accuracy = 6 / 14 = 43%

vs. 1-bit predictor:

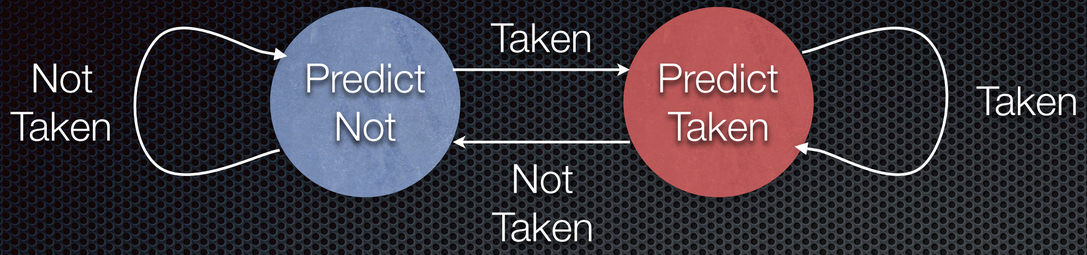
Accuracy = 9 / 14 = 64%

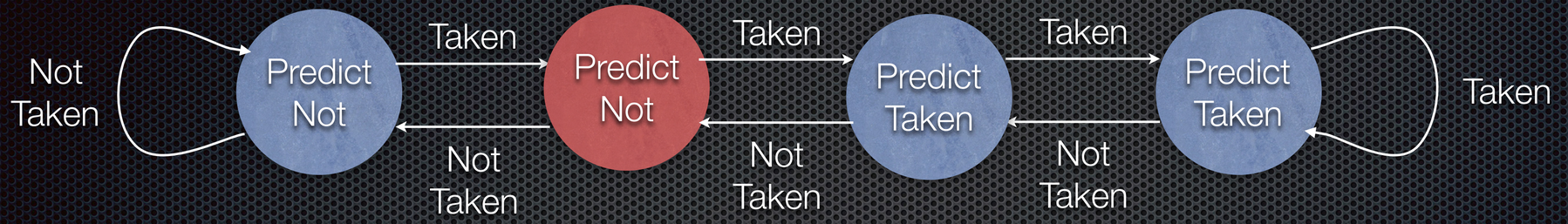
- Two-bit is better than one-bit for worst case
- One-bit is better for earlier example
- When is two-bit better?
- Another example



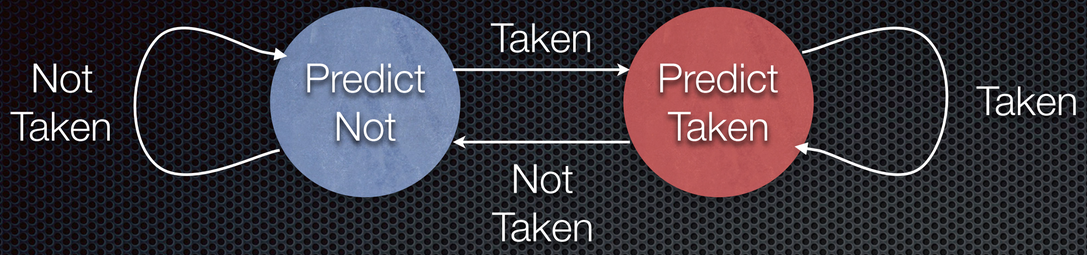


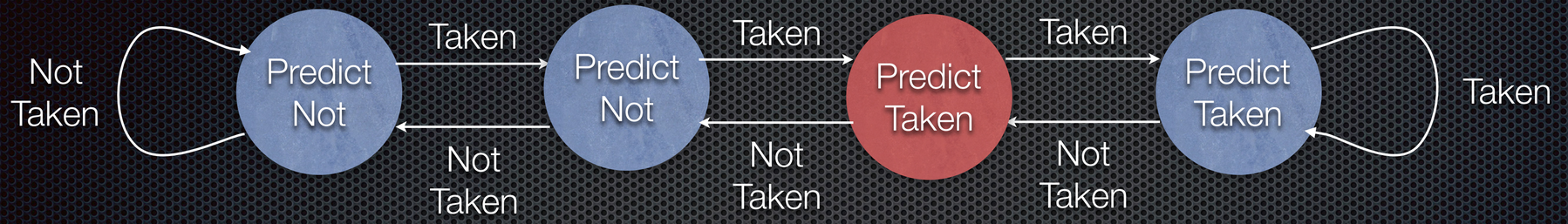

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N
1-bit corr:  N
2-bit pred:  N
2-bit corr:  N
```



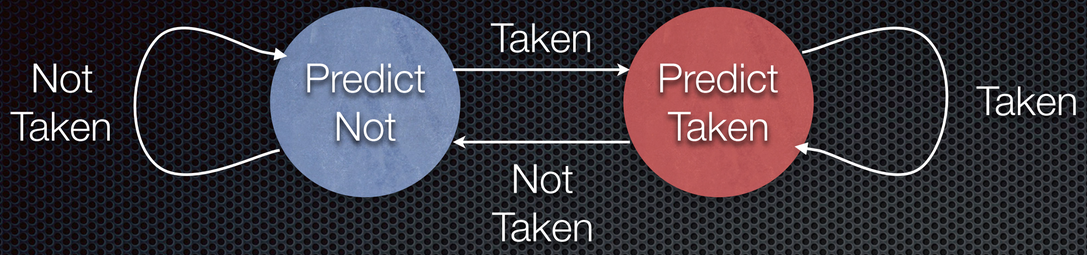


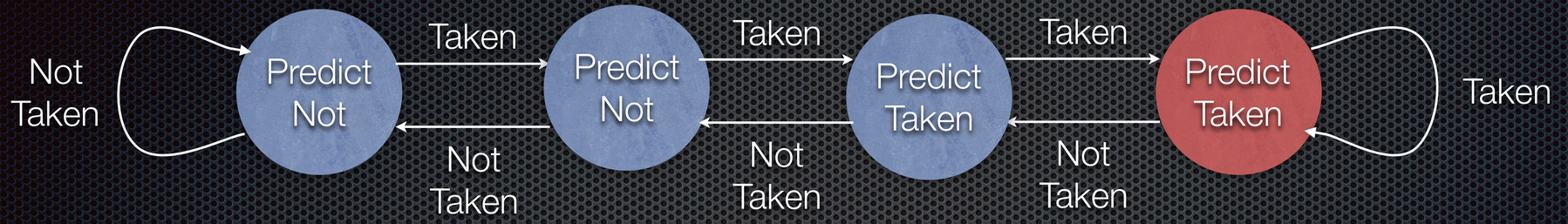

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T
1-bit corr:  N Y
2-bit pred:  N N
2-bit corr:  N N
```



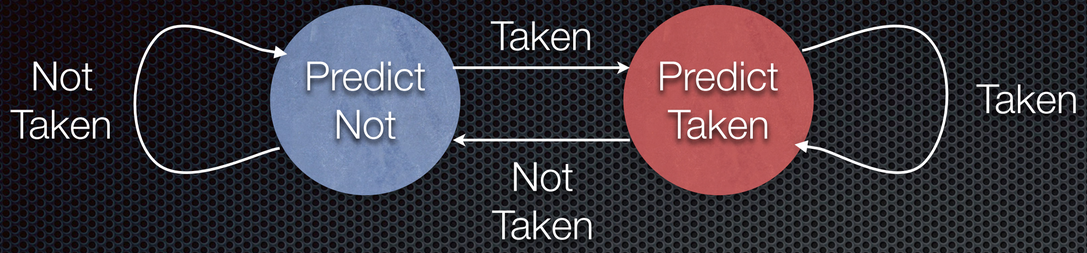


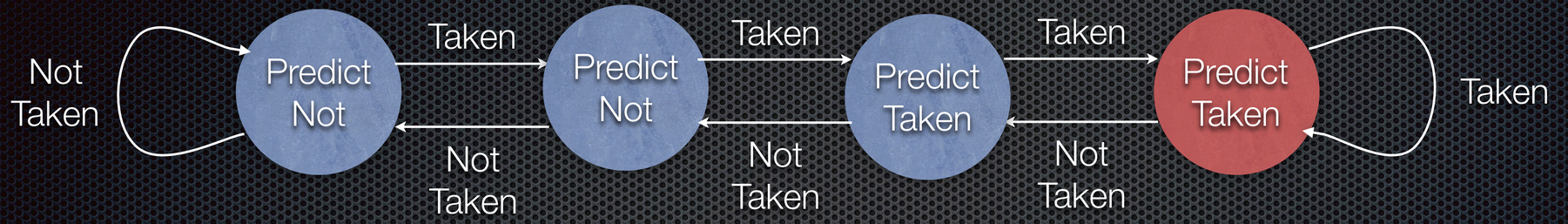

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T
1-bit corr:  N Y Y
2-bit pred:  N N T
2-bit corr:  N N Y
```



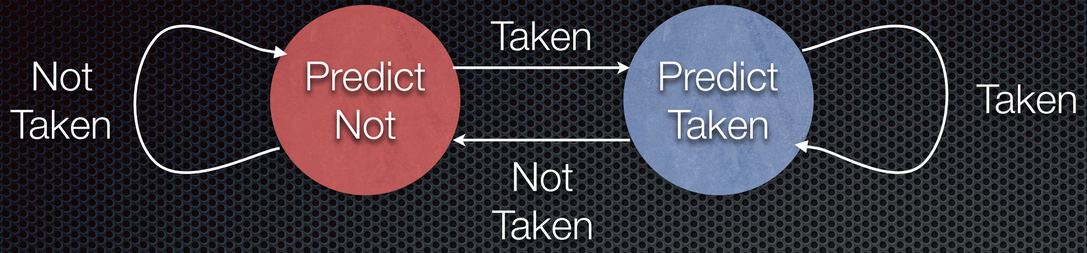


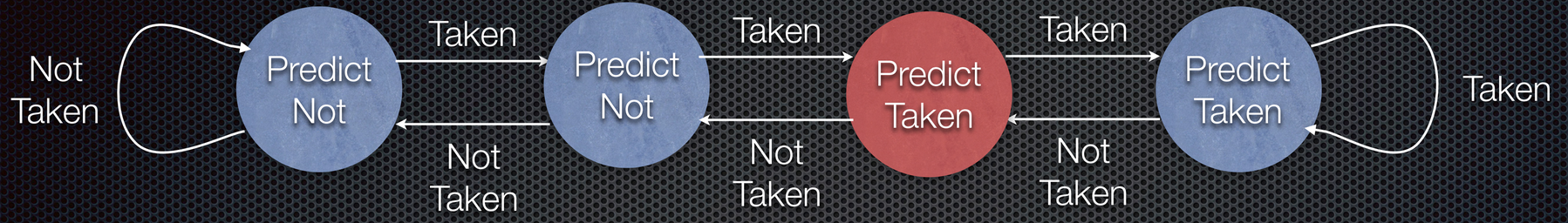

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T T
1-bit corr:  N Y Y Y
2-bit pred:  N N T T
2-bit corr:  N N Y Y
```



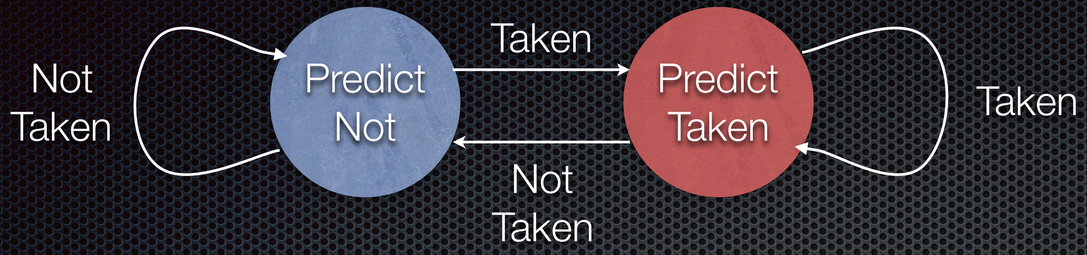


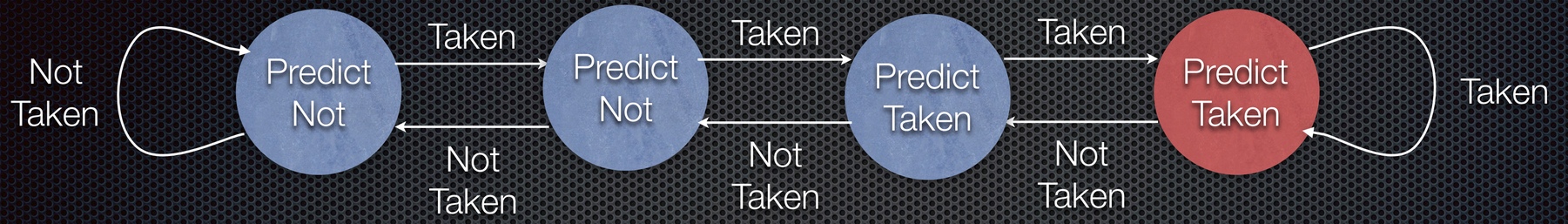

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T T T
1-bit corr:  N Y Y Y N
2-bit pred:  N N T T T
2-bit corr:  N N Y Y N
```



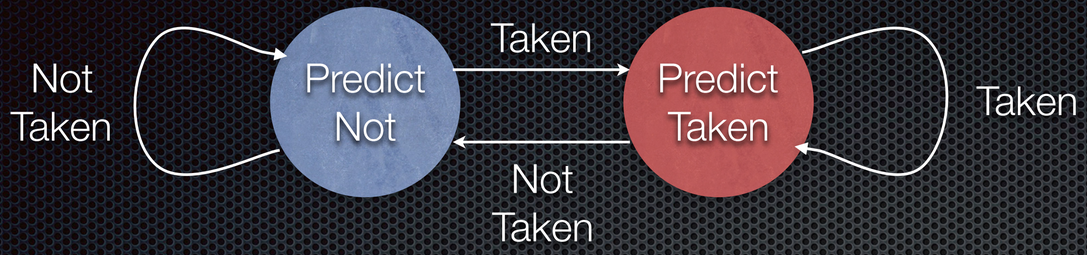


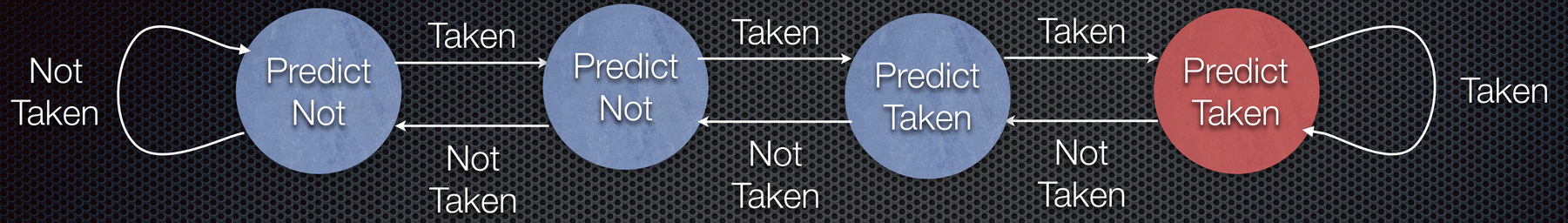

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T T T N
1-bit corr:  N Y Y Y N N
2-bit pred:  N N T T T T
2-bit corr:  N N Y Y N Y
```



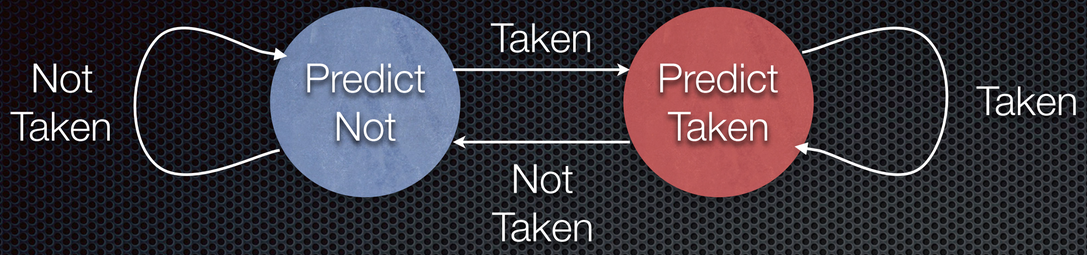


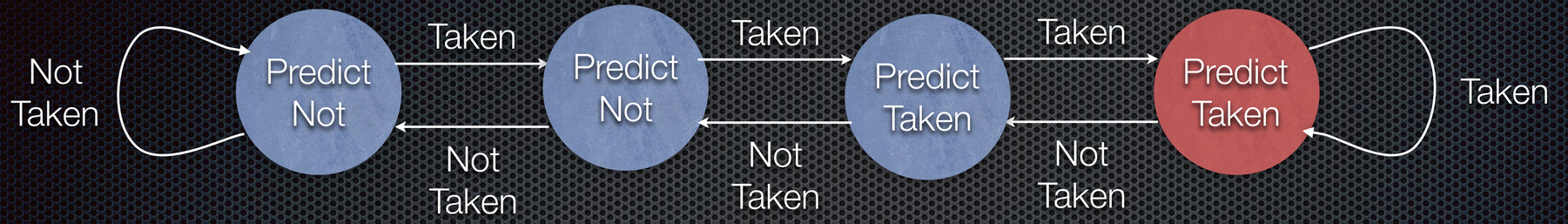

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T T T N T
1-bit corr:  N Y Y Y N N Y
2-bit pred:  N N T T T T T
2-bit corr:  N N Y Y N Y Y
```



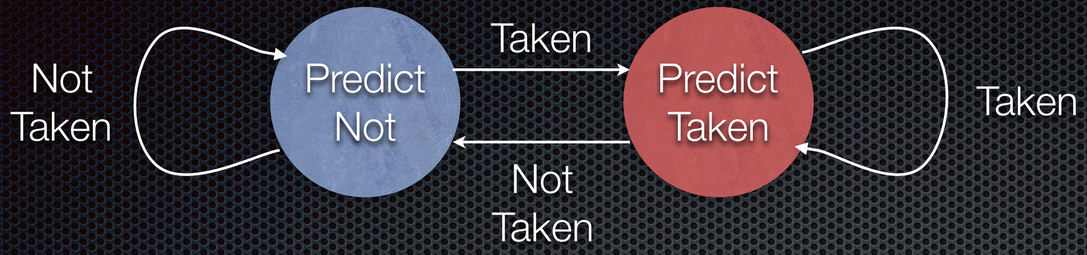


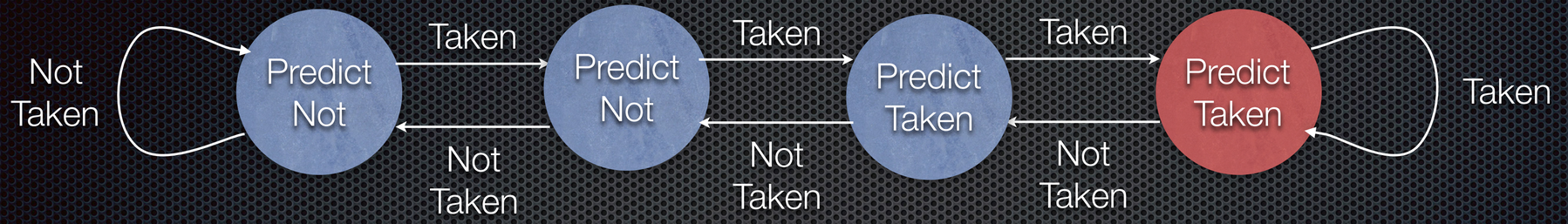

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T T T N T T
1-bit corr:  N Y Y Y N N Y Y
2-bit pred:  N N T T T T T T
2-bit corr:  N N Y Y N Y Y Y
```



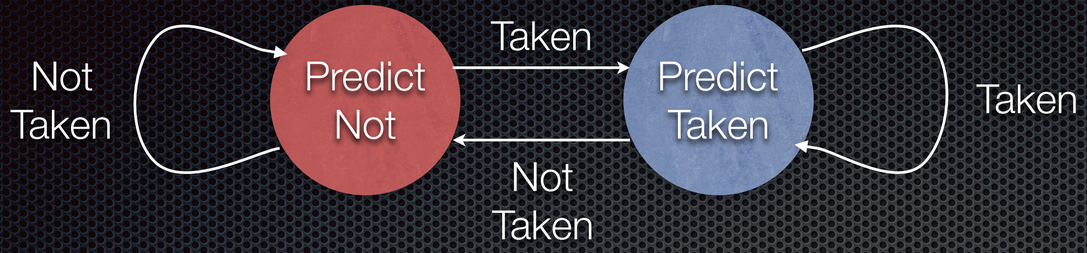


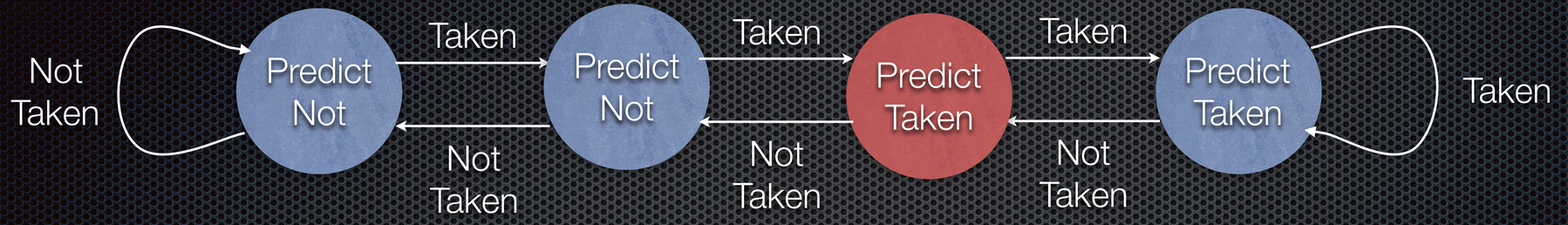

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T T T N T T T
1-bit corr:  N Y Y Y N N Y Y Y
2-bit pred:  N N T T T T T T T
2-bit corr:  N N Y Y N Y Y Y Y
```



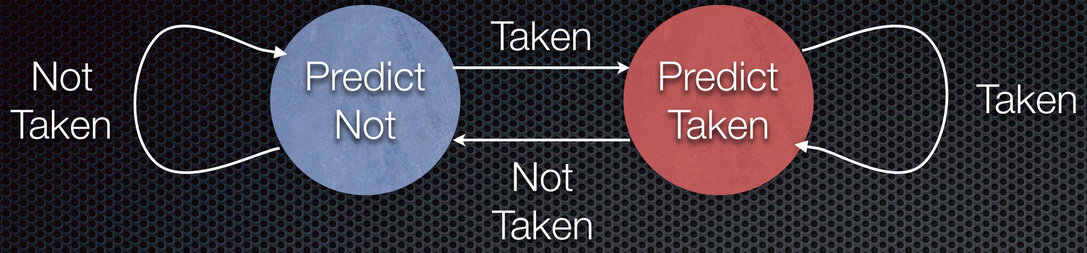


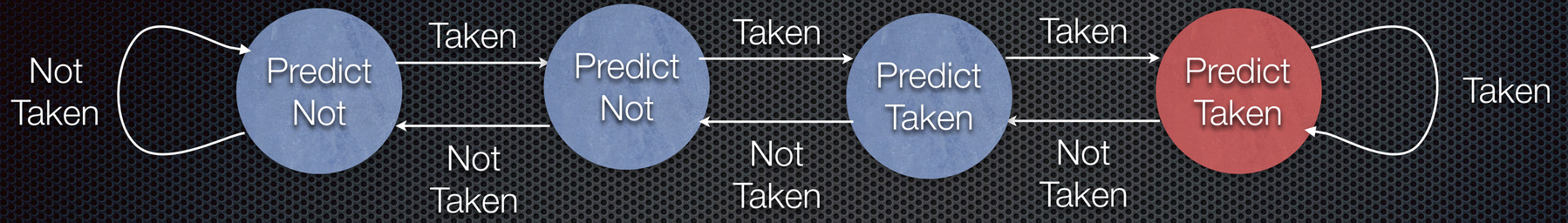

```
Outcome:      T T T T N T T T T N T T T T
1-bit pred:   N T T T T N T T T T
1-bit corr:   N Y Y Y N N Y Y Y N
2-bit pred:   N N T T T T T T T T
2-bit corr:   N N Y Y N Y Y Y Y N
```



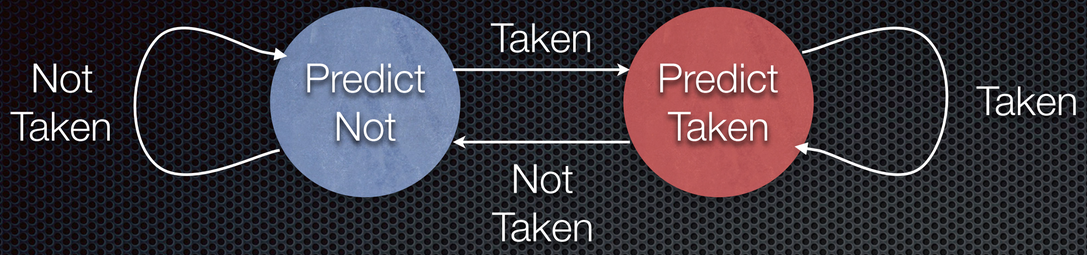


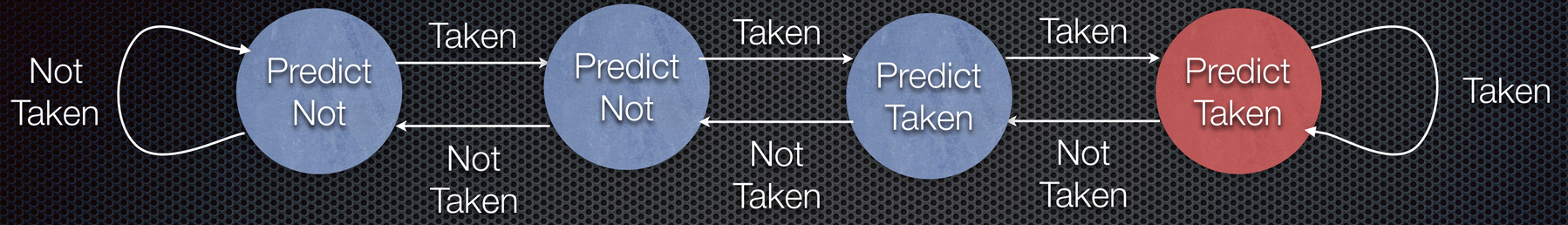

```
Outcome:      T T T T N T T T T N T T T T
1-bit pred:   N T T T T N T T T T N
1-bit corr:   N Y Y Y N N Y Y Y N N
2-bit pred:   N N T T T T T T T T T
2-bit corr:   N N Y Y N Y Y Y Y N Y
```



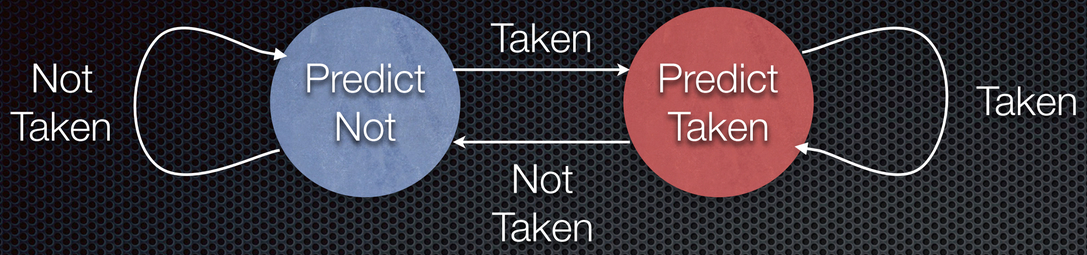


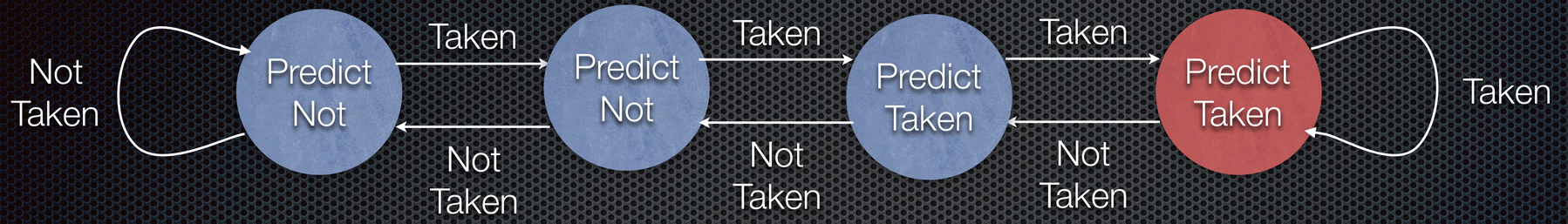

```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T T T N T T T T N T
1-bit corr:  N Y Y Y N N Y Y Y N N Y
2-bit pred:  N N T T T T T T T T T T
2-bit corr:  N N Y Y N Y Y Y Y N Y Y
```






```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T T T N T T T T N T T
1-bit corr:  N Y Y Y N N Y Y Y N N Y Y
2-bit pred:  N N T T T T T T T T T T T
2-bit corr:  N N Y Y N Y Y Y Y N Y Y Y
```






```
Outcome:     T T T T N T T T T N T T T T
1-bit pred:  N T T T T N T T T T N T T T
1-bit corr:  N Y Y Y N N Y Y Y N N Y Y Y
2-bit pred:  N N T T T T T T T T T T T T
2-bit corr:  N N Y Y N Y Y Y Y N Y Y Y Y
```

1-bit accuracy = 64% 2-bit accuracy = 71%

What Have We Learned?

- 2-bit predictor is better for branches that mostly take the same path, and only go down the other path once in a while (like loops and exceptions)
- 2-bit predictor takes longer to initially “train”
- 1-bit predictor is more effective on branches with random short sequences of path choices (like… ?)

Predictor Implementation

- Up to now, we've assumed that the predictor is associated with the branch
- How does this association happen?
- To answer this, we first need to know:
  - Where do the predictor bits live?
- In a specialized memory called the branch history table

Branch History Table

Addr	Bit 0	Bit 1
0		
1		
10		
11		
100		
101		
110		
111		

Association

- A branch can be stored at any address
- How do we connect the predictor address with a given branch address?
- Need a mapping function
- Because branch table is smaller than memory, this must be a many-to-one map

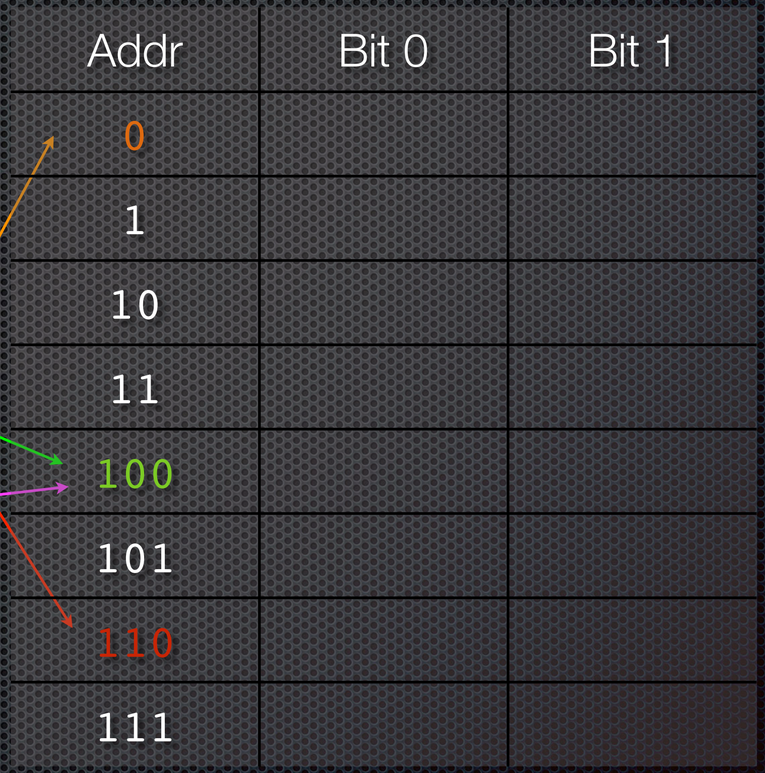
Use Low-Order Bits

- Because prediction must happen fast, we need a simple mapping function
- If the table has 2^N entries, we use the low order N bits of the word-address of the branch (we ignore byte address bits)

```
Branch 1: 00101101101101110
Branch 2: 00101110100101100
Branch 3: 00101101101110000
```

So far, so good. But...

```
Branch 1: 00101101101101110
Branch 2: 00101110100101100
Branch 3: 00101101101110000
Branch 4: 00101000001011100
```



Branches 2 & 4 conflict in the table

Aliasing

- When branches conflict, they are aliases of each other in the mapping
- The actions of both affect the predictor FSM at that location in the table
- If they have similar behavior (e.g., both are exceptions) they reinforce each other (constructive aliasing)
- Most aliasing is destructive -- reduces accuracy of the predictions

Reducing Aliasing

- Make table bigger -- fewer conflicts
  - If too big, table will be slower
- Have compiler rearrange or pad instructions to reduce branch conflicts
  - Only works locally -- can’t control how loader relocates code

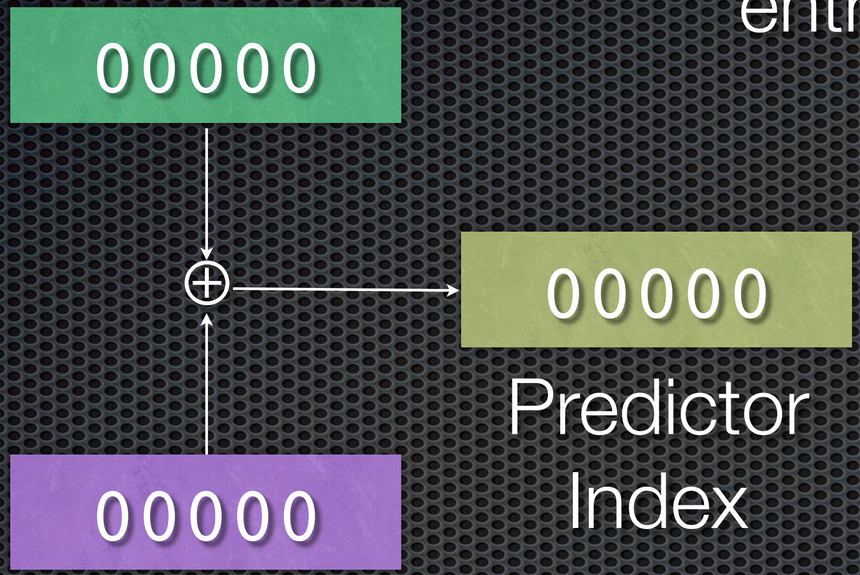
What Effect Does Prediction Have?

- Changes PC to predicted branch target as soon as branch is decoded (or even earlier)
- Branch table may be augmented with branch target buffer that remembers last target address, avoiding need to recompute it
- No branch penalty if prediction is correct
- Incorrect prediction squashes instructions behind branch, and restarts pipe (same as if no predictor)

G-Share Branch Predictor

Using Global History to Distinguish Branch Behavior Patterns

Branch Address



Predictor Index

Global History

Predictor table with two-bit entries

Index	State
00000	
00001	
00010	
00011	
00100	
00101	
00110	
00111	
01000	
01001	
01010	
01011	
01100	
01101	
01110	
01111	
10000	
10001	
10010	
10011	
10100	
10101	
10110	
10111	
11000	
11001	
11010	
11011	
11100	
11101	
11110	
11111	

Simple Case

- One branch, with low order address bits 00111
- Alternating Pattern: TNTNTNTNTNTNTNTN

Predict: Not Taken
Outcome: Taken

Branch Address

00111

00111

Predictor Index

00000

Global History

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	0
00110	0
00111	0
01000	0
01001	0
01010	0
01011	0
01100	0
01101	0
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

Predict: Not Taken
Outcome: Not Taken

Branch Address

00111

⊕

00001

Global History

00110

Predictor Index

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	0
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	0
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

Predict: Not Taken
Outcome: Taken

Branch Address

00111

00101

Predictor Index

00010

Global History

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	0
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	0
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

Predict: Not Taken
Outcome: Not Taken

Branch Address

00111

⊕

00010

Predictor Index

00101

Global History

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	0
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

Predict: Not Taken
Outcome: Taken

Branch Address

00111

01101

Predictor Index

01010

Global History

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	0
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

Predict: Not Taken
Outcome: Not Taken

Branch Address

00111

10101

Global History

10010

Predictor Index

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	1
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

Predict: Not Taken
Outcome: Taken

Branch Address

00111

01101

Predictor Index

01010

Global History

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	1
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

Accessed 2x

Predict: Not Taken
Outcome: Not Taken

Branch Address

00111

10010

Predictor Index

10101

Global History

Accuracy = 50%

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	2
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

Accessed 2x

Predict: Taken
Outcome: Taken

Branch Address

00111

01101

Predictor Index

01010

Global History

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	3
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

Accessed 2x

Accessed 3x

Predict: Not Taken
Outcome: Not Taken

Branch Address

00111

10010

Predictor Index

10101

Global History

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	3
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Accessed 1x

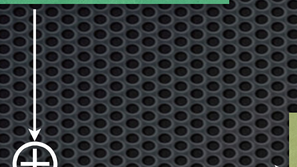
Accessed 2x

Accessed 3x

Predict: Taken
Outcome: Taken

Branch Address

00111



01101

Predictor Index

01010

Global History

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	3
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

After 4 mispredicts, now trained --
Accuracy will be 100% from now on

Predict: Not Taken
Outcome: Not Taken

Branch Address

00111

10010

Predictor Index

10101

Global History

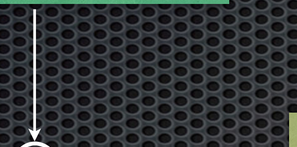
Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	0
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	3
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

Conflicts with four other entries during training, reducing accuracy

Predict: Taken
Outcome: Taken

Branch Address

00111



01101

Predictor Index

01010

Global History

A single branch now uses two entries, increasing aliasing

Index	State
00000	0
00001	0
00010	0
00011	0
00100	0
00101	1
00110	1
00111	1
01000	0
01001	0
01010	0
01011	0
01100	0
01101	3
01110	0
01111	0
10000	0
10001	0
10010	0
10011	0
10100	0
10101	0
10110	0
10111	0
11000	0
11001	0
11010	0
11011	0
11100	0
11101	0
11110	0
11111	0

G-Share Summary

- Global history can improve prediction accuracy for complex branches by separating different streams of behaviors, based on correlation with other branches
- During training, prediction accuracy may be low
- The training phase can disrupt other entries in the predictor table, reducing their accuracy
- The more complex the behavior, the longer the training phase takes

Hybrid Branch Predictor

Best of both?

Two In One

- Combine a local and global predictor
- Third table, predictor selector, chooses which predictor is associated with a given branch
- Enough mispredicts from one predictor can trigger a switch to the other predictor

3-bit Predictor Selector

Addr	Sel
000	G
001	L
010	L
011	G
100	L
101	G
110	G
111	L

1-bit Local Predictor

Addr	Pred
000	T
001	N
010	T
011	T
100	T
101	N
110	T
111	N

2-bit Global Predictor

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

For simplicity of illustration, states are not shown, and are only partly represented by colors

Branch Address

010 →

Addr	Sel
000	G
001	L
010	L
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	T
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

010

Addr	Sel
000	G
001	L
010	L
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	T
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

010

Addr	Sel
000	G
001	L
010	L
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	T
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

010

Addr	Sel
000	G
001	L
010	L
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	T
011	T
100	T
101	N
110	T
111	N

Mispredict

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Assume this is the second mispredict for this location

Addr	Sel
000	G
001	L
010	L
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Updated

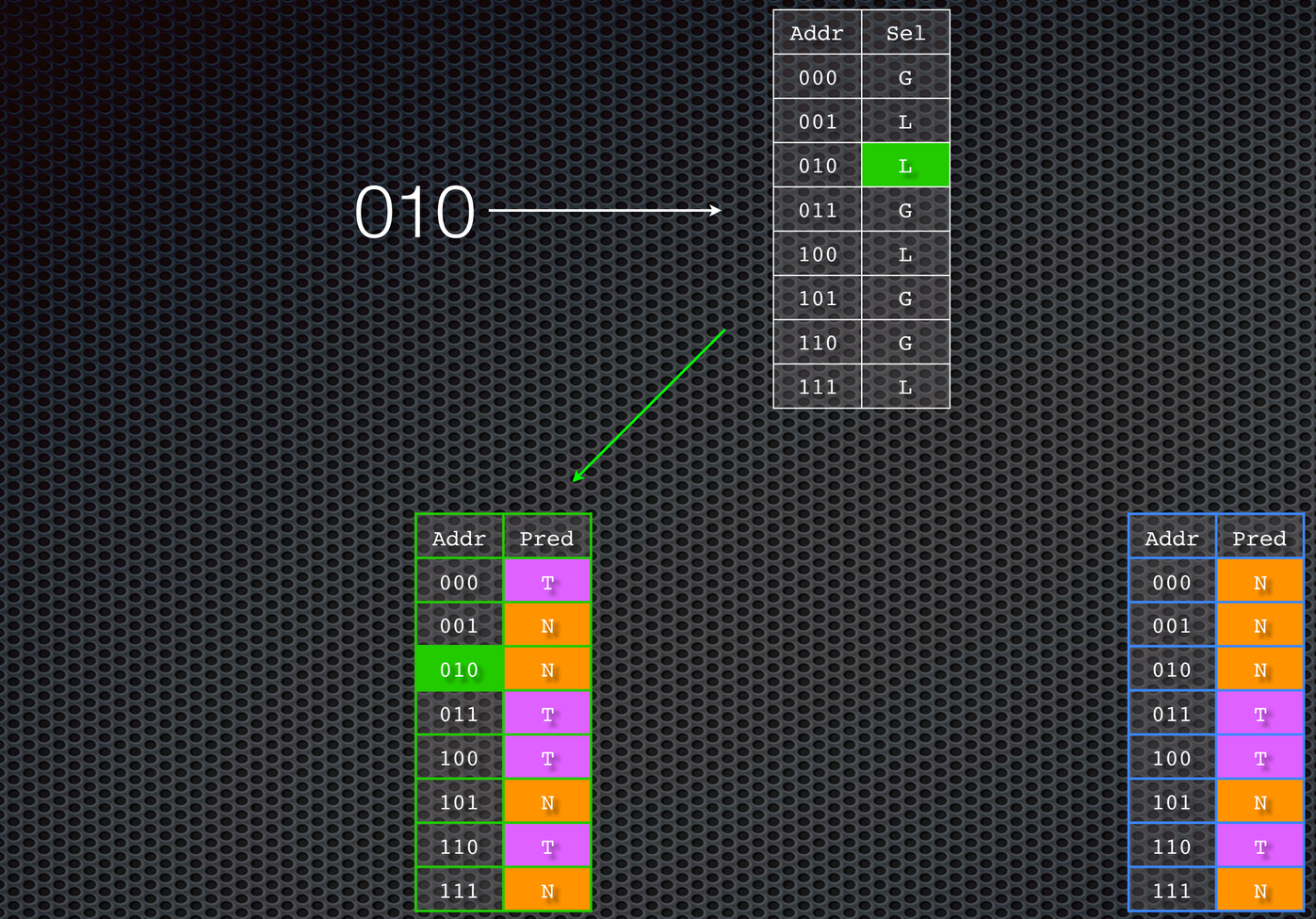
Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

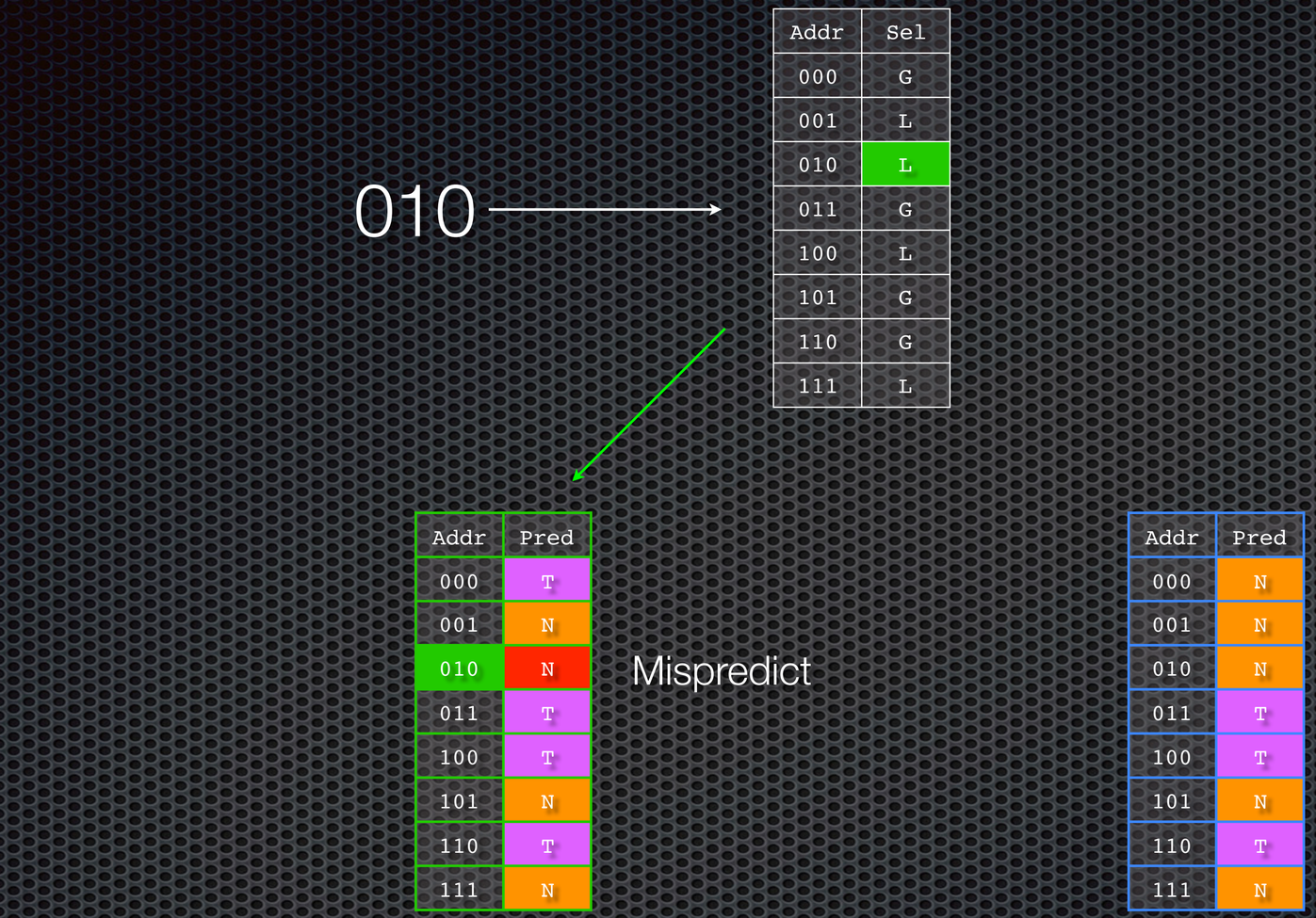
010

Addr	Sel
000	G
001	L
010	L
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N


010
Addr
Sel
000
G
001
L
010
L
011
G
100
L
101
G
110
G
111
L
Addr
Pred
000
T
001
N
010
N
011
T
100
T
101
N
110
T
111
N
Addr
Pred
000
N
001
N
010
N
011
T
100
T
101
N
110
T
111
N


010
Addr
Sel
000
G
001
L
010
L
011
G
100
L
101
G
110
G
111
L
Addr
Pred
000
T
001
N
010
N
011
T
100
T
101
N
110
T
111
N
Mispredict
Addr
Pred
000
N
001
N
010
N
011
T
100
T
101
N
110
T
111
N

Addr	Sel
000	G
001	L
010	L?
011	G
100	L
101	G
110	G
111	L

Updated

Addr	Pred
000	T
001	N
010	T
011	T
100	T
101	N
110	T
111	N

Updated

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

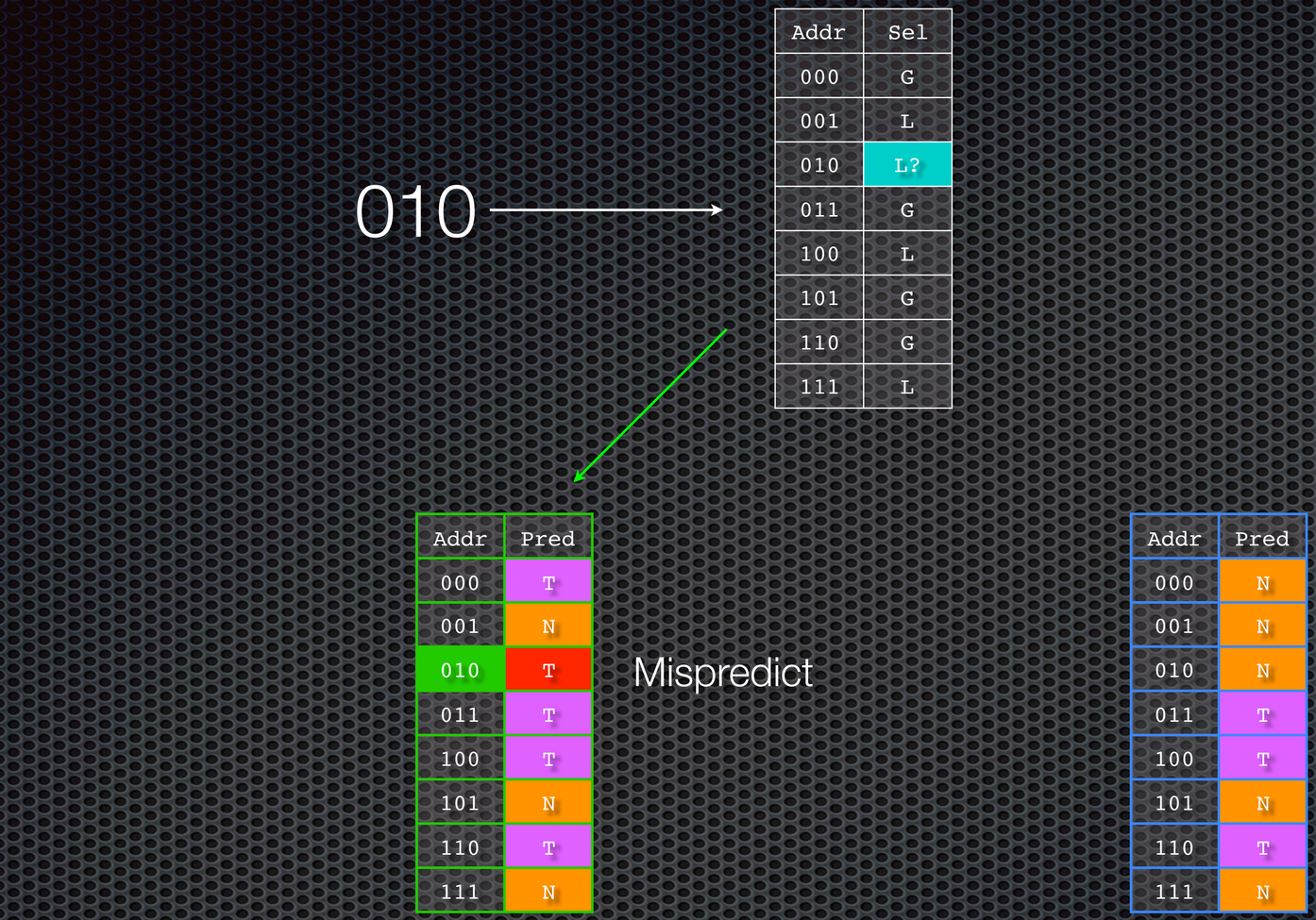
After 3 mispredicts by the local predictor, the predictor selector has reached the threshold of changing to global

010

Addr	Sel
000	G
001	L
010	L?
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	T
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N


010
Addr
Sel
000
G
001
L
010
L?
011
G
100
L
101
G
110
G
111
L
Addr
Pred
000
T
001
N
010
T
011
T
100
T
101
N
110
T
111
N
Mispredict
Addr
Pred
000
N
001
N
010
N
011
T
100
T
101
N
110
T
111
N

Addr	Sel
000	G
001	L
010	G?
011	G
100	L
101	G
110	G
111	L

Updated

Addr	Pred
000	T
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Updated

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

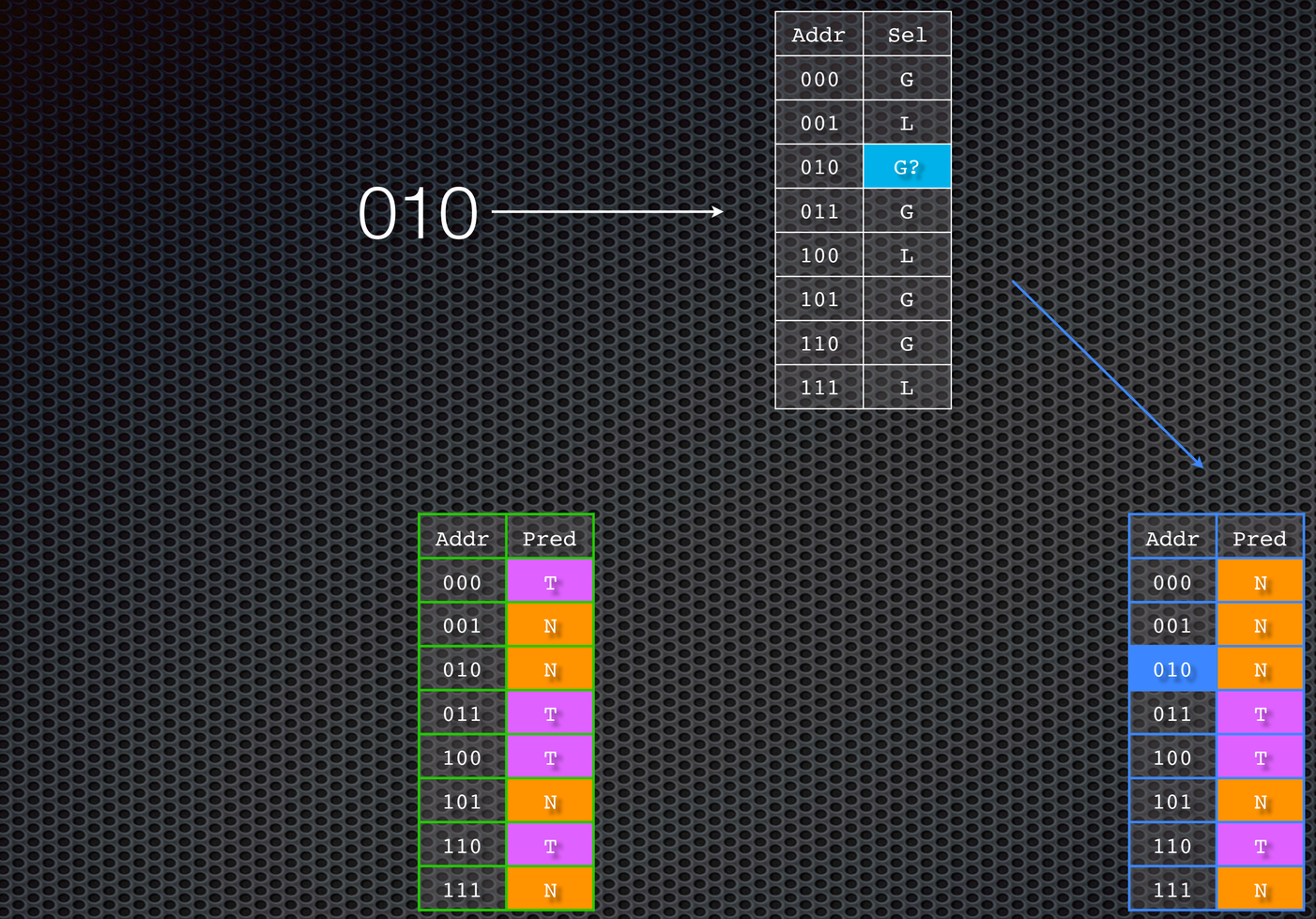
The local predictor failed four times -- give global a try

010

Addr	Sel
000	G
001	L
010	G?
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N


010
Addr
Sel
000
G
001
L
010
G?
011
G
100
L
101
G
110
G
111
L
Addr
Pred
000
T
001
N
010
N
011
T
100
T
101
N
110
T
111
N
Addr
Pred
000
N
001
N
010
N
011
T
100
T
101
N
110
T
111
N

010

Addr	Sel
000	G
001	L
010	G?
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Correct

Addr	Sel
000	G
001	L
010	G
011	G
100	L
101	G
110	G
111	L

Updated

Addr	Pred
000	T
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

With one success, now solidly in global mode. Had it failed, then local would get another try.

010

Addr	Sel
000	G
001	L
010	G
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

010

Addr	Sel
000	G
001	L
010	G
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

010

Addr	Sel
000	G
001	L
010	G
011	G
100	L
101	G
110	G
111	L

Addr	Pred
000	T
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Addr	Pred
000	N
001	N
010	N
011	T
100	T
101	N
110	T
111	N

Correct

Some Additional Notes

- Hybrid predictors can take longer to train
- Possible to ping-pong between predictors, especially when global predictor isn’t trained
- Selector threshold fixed, but could vary
- Predictors don’t have to be same size (cheaper local predictor could have more entries)
- Much of benefit comes from reduced aliasing in global predictor
- Branch hint flags can help initialize predictors

Cliff Young ISCA 1995

A Comparative Analysis of Schemes for Correlated Branch Prediction

Dividers and Predictors

- Many branches are correlated with control flow streams that pass through different paths in the code
- A divider is a mechanism that separates out the streams to different predictors
- Predictors can be many flavors. Focus on profile-driven path-based static predictors and dynamic pattern-based predictors

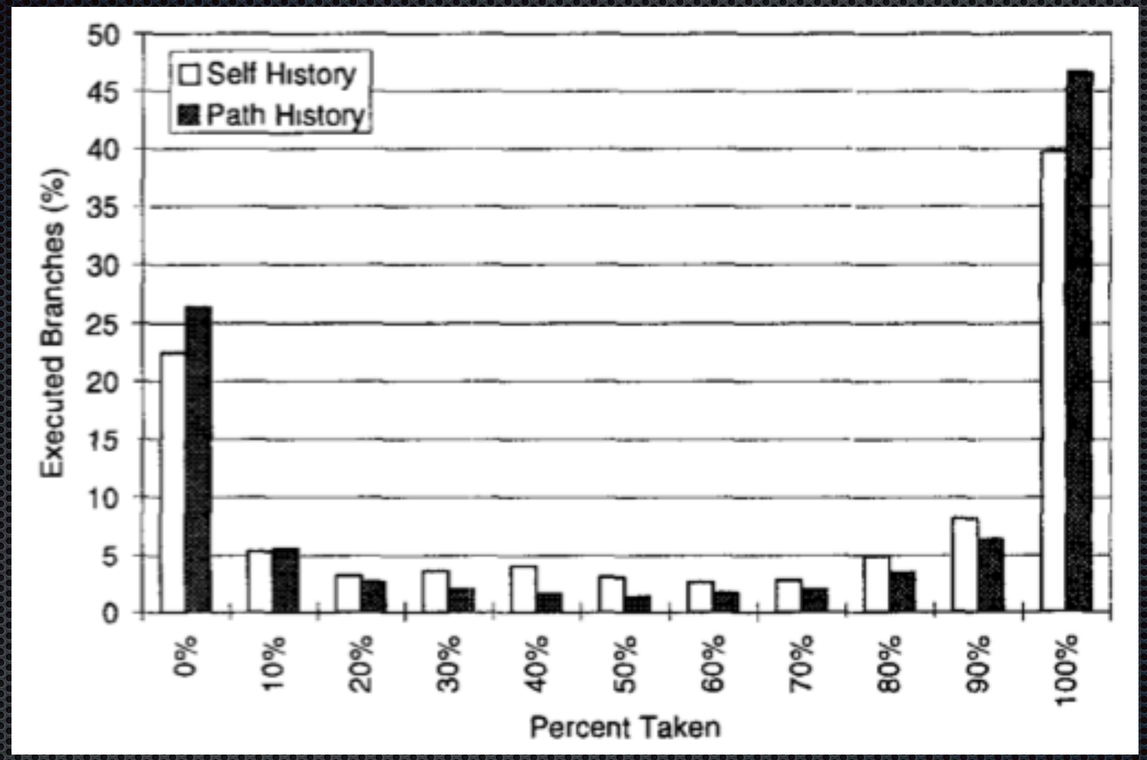
Paths vs. Patterns

- A path is a sequence of branch instructions leading to the branch of interest
- A pattern is a record of recent branch outcomes
- Patterns often represent paths, but there are exceptions
- Compilers can determine control flow paths
- Patterns are based on global history (as in G-Share)

Profiling Static Predictor

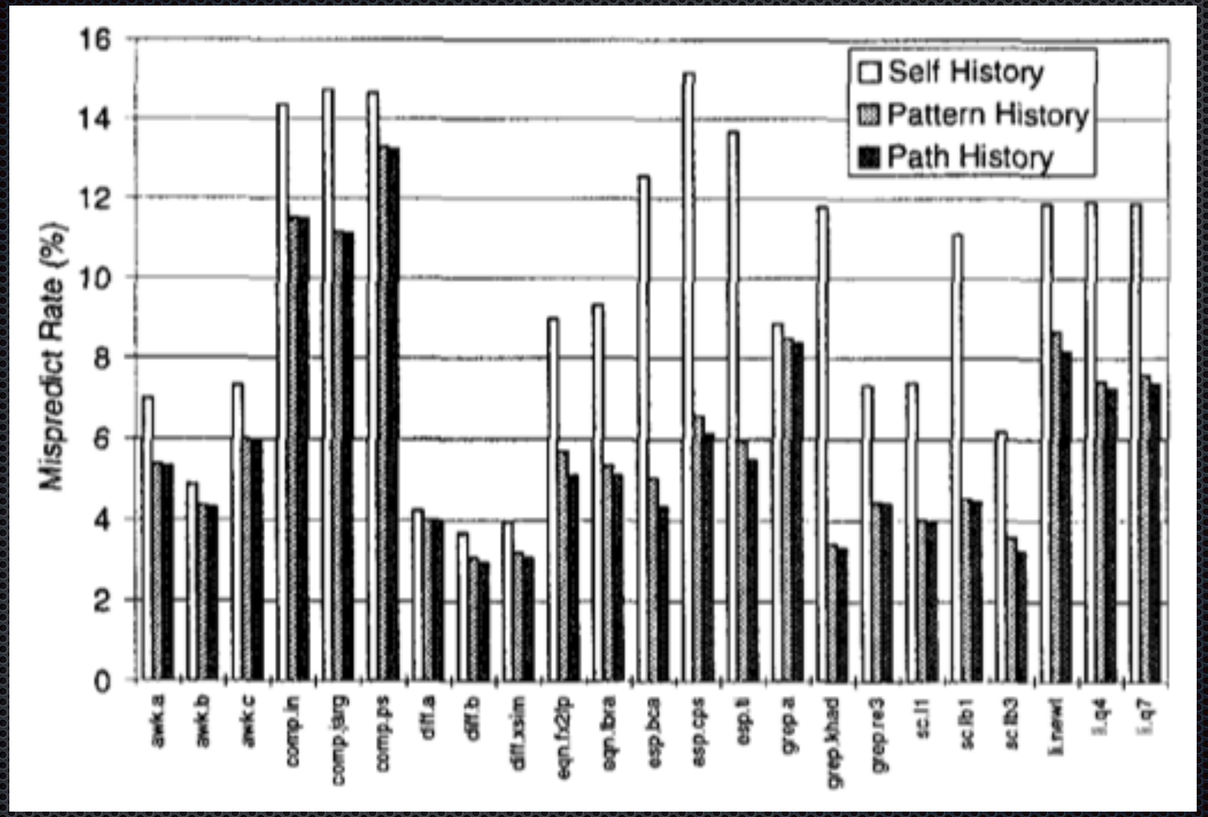
- Profiling determines most common outcome of branch
- Compiler reorganizes code (replicates branches) so that paths with different behavior streams go to different branches
- Compiler tags branches with likely outcome, which is the basis for prediction

Branch Bias



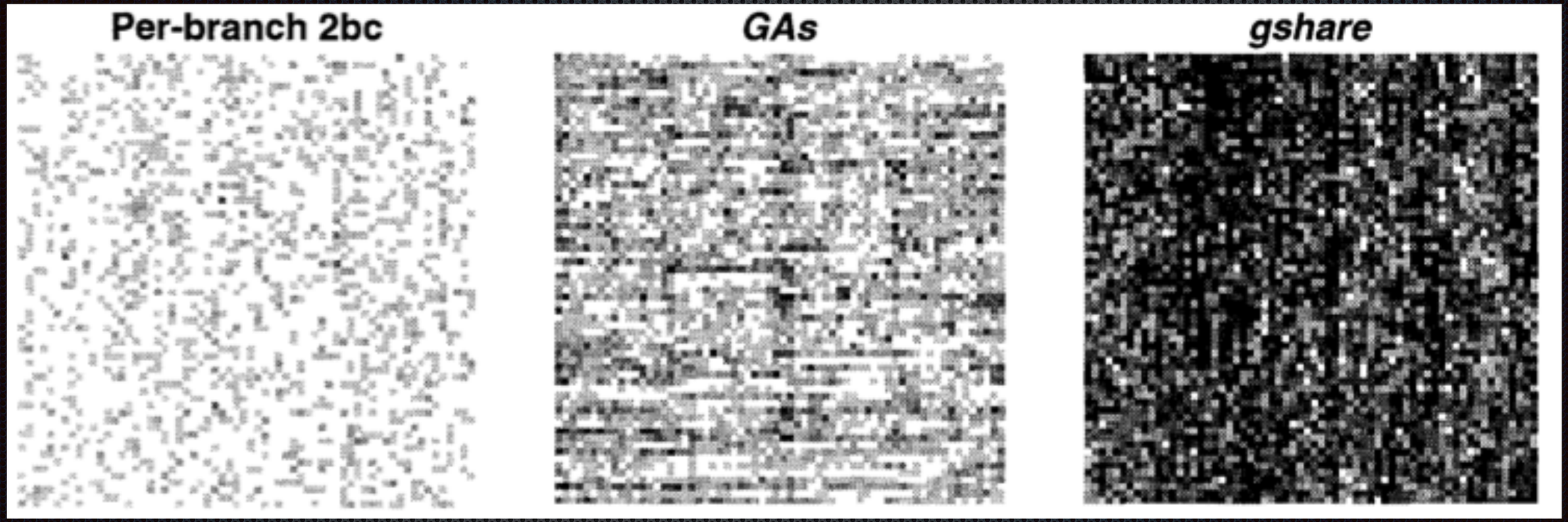
Using path history helps distinguish branches more than self history for the most common cases

Path vs. Pattern



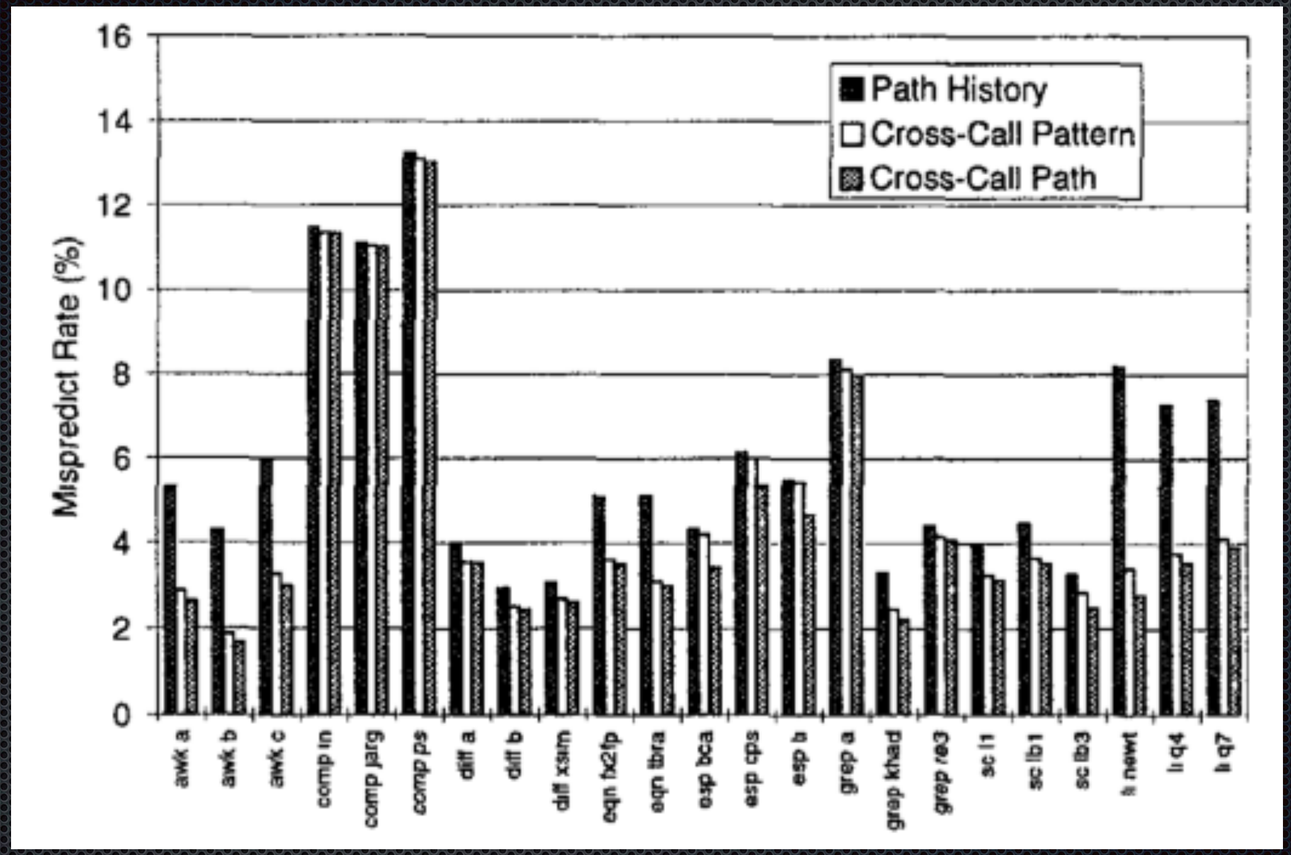
Paths do slightly better than patterns
(smaller is better)

Aliasing



More global history in divider increases aliasing
(also showed most aliasing is destructive)

Cross-Call Effect



G-Share crosses calls automatically, boosting its prediction rate -- Hypothetical cross-call path

Discussion

Marius Evers ISCA 1998

An Analysis of Correlation and Predictability: What Makes Two-Level Branch Predictors Work

Some Branches are Related

```
branch Y: if (cond1)
...
branch X: if (cond1 AND cond2)
```

(a)

```
branch Y: if (cond1) a = 2;
...
branch X: if (a == 0)
```

(b)

```
branch Y: if (cond1)
...
branch Z: if (cond2)
...
branch X: if (cond1 AND cond2)
```

(c)

A Few Good Branches

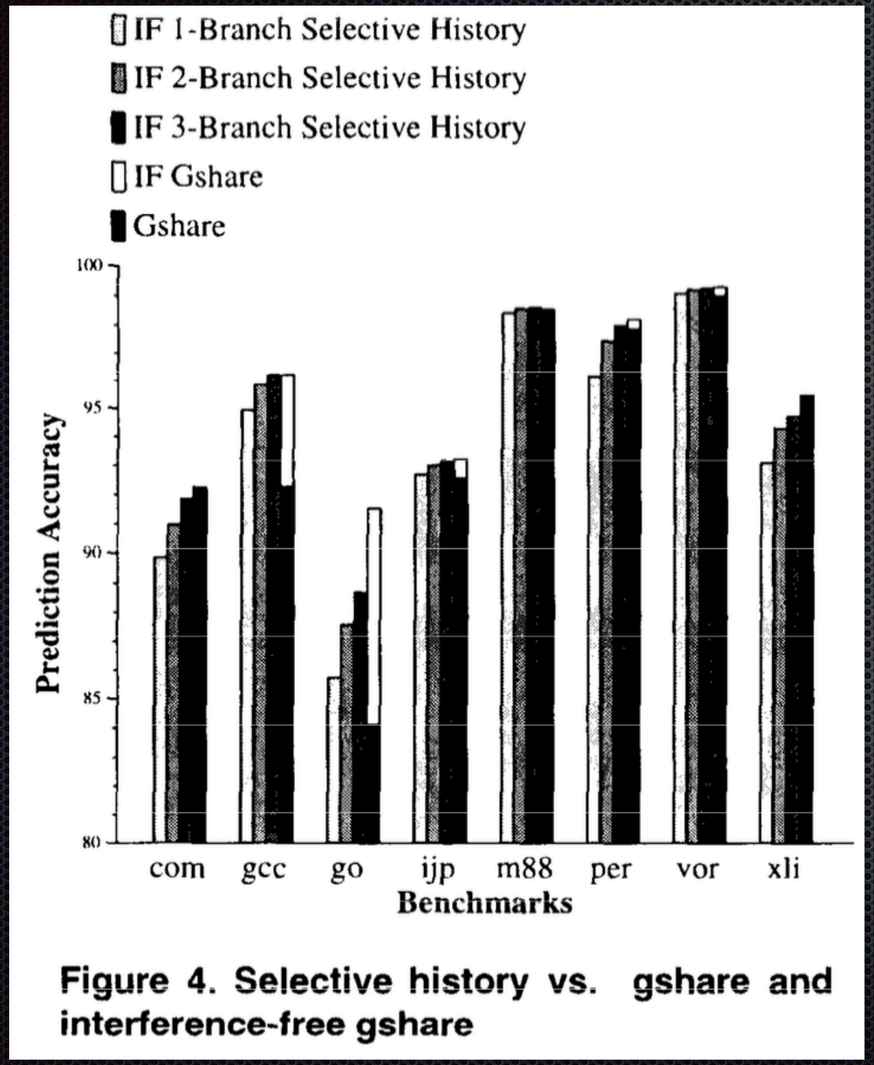


Figure 4. Selective history vs. gshare and interference-free gshare

Many branches are irrelevant. If we can identify the ones that are correlated and use only those (so they are interference-free) Then those few will be as predictive as a larger path

Interference Causes Aliasing

- gshare is good for correlated branches, but its reliance on global history results in destructive aliasing that reduces its effectiveness
- Some branches don’t need global history -- their own behavior is indicative of future actions (per address)
- Some basically always behave the same (static)

gshare Hybrid vs. Correlated

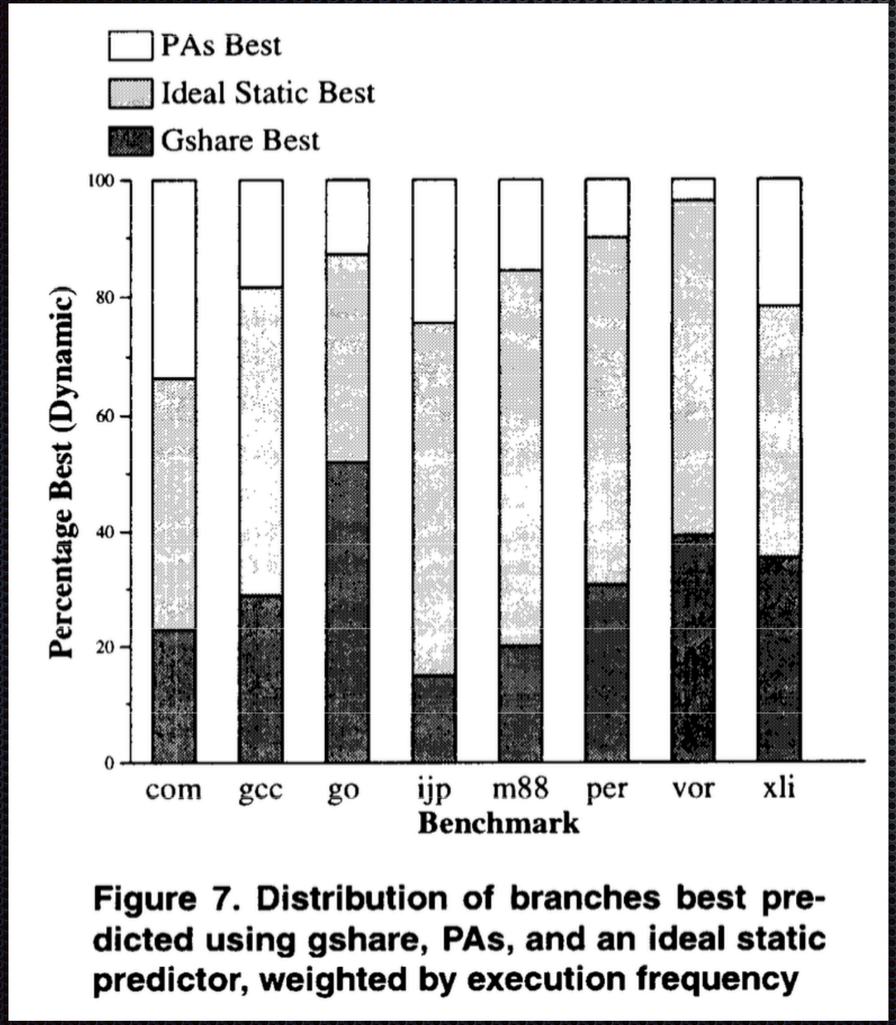


Figure 7. Distribution of branches best predicted using gshare, PAs, and an ideal static predictor, weighted by execution frequency

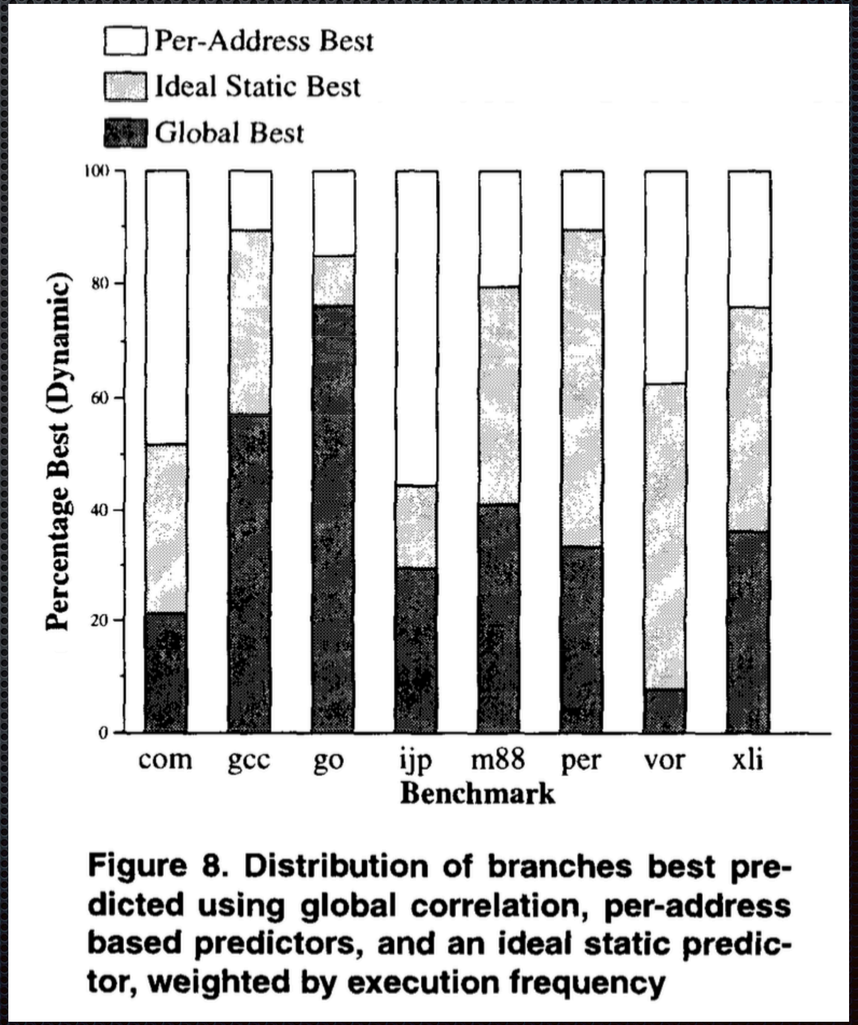


Figure 8. Distribution of branches best predicted using global correlation, per-address based predictors, and an ideal static predictor, weighted by execution frequency

Conclusion

- There is room for improvement beyond a hybrid of gshare and per-address (PA)
- Global correlation can capture some of that gap
- That leaves 40% of branch behaviors that aren’t being caught
- Of those, 92% show strong bias, leaving only 8% (3.2% overall) that are not predictable

Discussion