



# TOAD

**Tips for Object Architecture Development**

*SEAN WOODS*

# The Paper

- ✱ Lessons learned from [ab]using TclOO

# Good Coding Practice

- ✱ Eliminate copy and paste
- ✱ Eliminate copy and paste
- ✱ Eliminate copy and paste

```
method foo theunmungedarg {  
  # Load our variables  
  variable bar  
  variable bat  
  set arg [my munge $theunmungedarg]  
  # Begin with a blank result  
  set theResult {}  
  
  ...  
  (the actual method)  
  ...  
  
  # Remunge our result  
  return [my remunge $theResult]  
}
```

```
method foo theunmungedarg {
  # Load our variables
  variable bar
  variable bat
  set arg [my munge $theunmungedarg]
  # Begin with a blank result
  set theResult {}

  ...
  (the actual method)
  ...

  # Remunge our result
  return [my remunge $theResult]
}
```

```
method baz theunmungedarg {
  # Load our variables
  variable bar
  variable bat
  set arg [my munge $theunmungedarg]
  # Begin with a blank result
  set theResult {}

  ...
  (the actual method)
  ...

  # Remunge our result
  return [my remunge $theResult]
}
```

```
method foo theunmungedarg {
  # Load our variables
  variable bar
  variable bat
  set arg [my munge $theunmungedarg]
  # Begin with a blank result
  set theResult {}

  ...
  (the actual method)
  ...

  # Remunge our result
  return [my remunge $theResult]
}
method baz theunmungedarg {
  # Load our variables
  variable bar
  variable bat
  set arg [my munge $theunmungedarg]
  # Begin with a blank result
  set theResult {}

  ...
  (the actual method)
  ...
```

```
method bing theunmungedarg {
  # Load our variables
  variable bar
  variable bat
  set arg [my munge $theunmungedarg]
  # Begin with a blank result
  set theResult {}

  ...
  (the actual method)
  ...

  # Remunge our result
  return [my remunge $theResult]
}
```

```
method foo theunmungedarg {  
  # Load our variables  
  variable bar  
  variable bat  
  set arg [my munge $theunmungedarg]  
  # Begin with a blank result  
  set theResult {}  
  
  ...  
  (the actual method)  
  ...  
  
  # Remunge our result  
  return [my remunge $theResult]  
}
```

```
set ::preamble {  
  # Load our variables  
  variable bar  
  variable bat  
  set arg [my munge $theunmungedarg]  
  # Begin with a blank result  
  set result {}  
}
```

```
set ::postamble {  
  # Remunge our result  
  return [my remunge $theresult]  
}
```

```
proc pageMethod {class methodname body} {  
  set methodbody $::preamble  
  append methodbody \n $body \n  
  append methodbody \n $::postamble \n  
  oo::class define $class $methodname \  
    theunmungedarg $methodbody  
}
```

# IMPLEMENTATIONS ARE MUCH SHORTER, AND MUCH EASIER TO READ

```
set ::preamble {
  # Load our variables
  variable bar
  variable bat
  set arg [my munge $theunmungedarg]
  # Begin with a blank result
  set result {}
}
set ::postamble {
  # Remunge our result
  return [my remunge $theresult]
}

proc pageMethod {
  class methodname body
} {
  # Build a buffer that starts
  # with our preamble
  set methodbody $::preamble
  # Tacks on our body
  append methodbody \n $body \n
  # Tack on code that
  # transforms the result
  append methodbody \n \
    $::postamble \n

  # With the actual body built
  # define the method
  oo::class define $class \
    $methodname \
    theunmungedarg \
    $methodbody
}
```

```
oo::class create pageMake

pageMethod pageMake foo {
  ... foo body ...
}
pageMethod pageMake bar {
  ... bar body ...
}
pageMethod pageMake baz {
  ... baz body ...
}
```

# Klingon Programming

- \* Code does not have parameters
- \* It has arguments
- \* AND IT ALWAYS WINS THEM

# Read the Paper

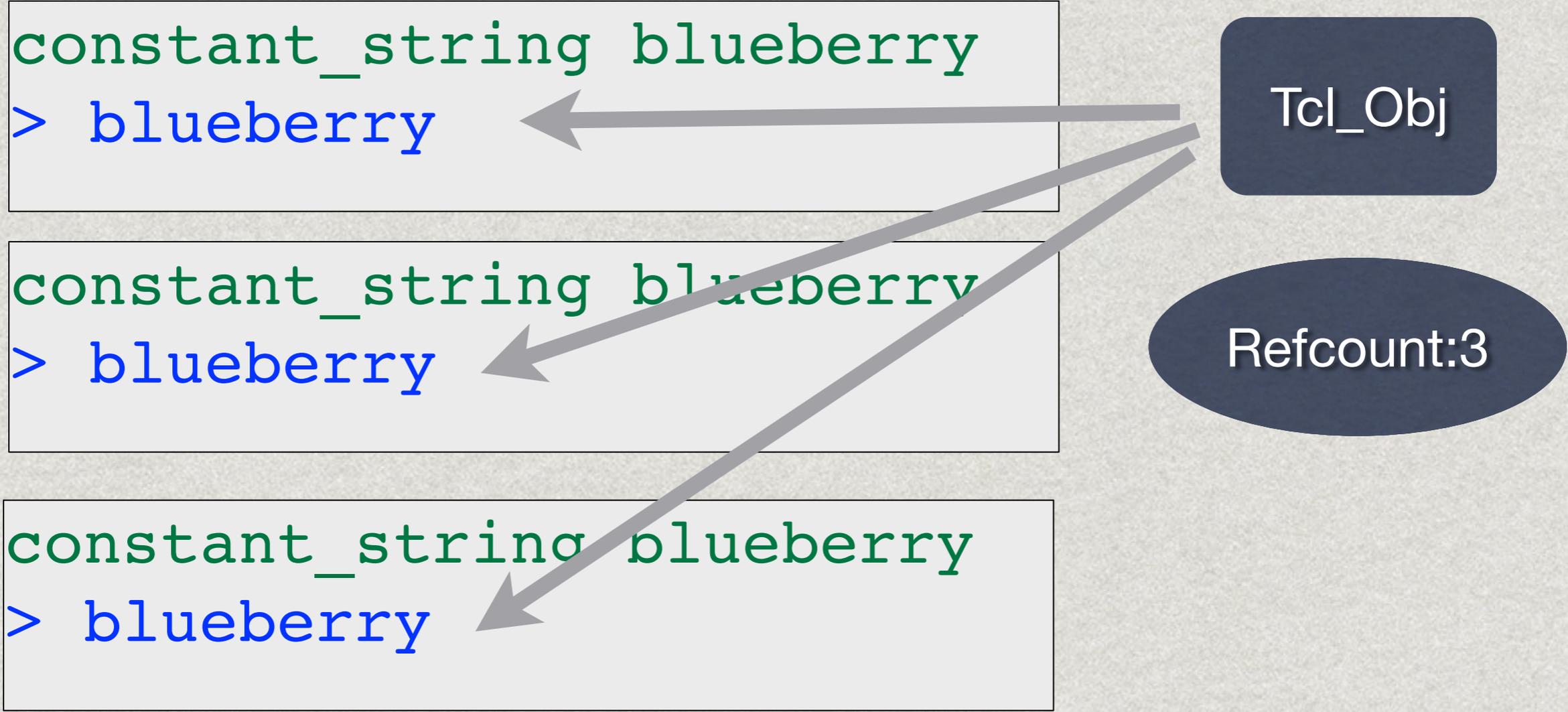
- \* Using code wrappers
- \* Winning Arguments with Dicts
- \* Sanity Saving Standardized Methods
- \* Better living through forwarding
- \* A C implementation of a memory saving technique

# Constant Strings

```
dict set inmemdb $field $value
```

```
dict set inmemdb \  
  [constant_string $field] $value
```

# Copies of the same Tcl\_Obj\*

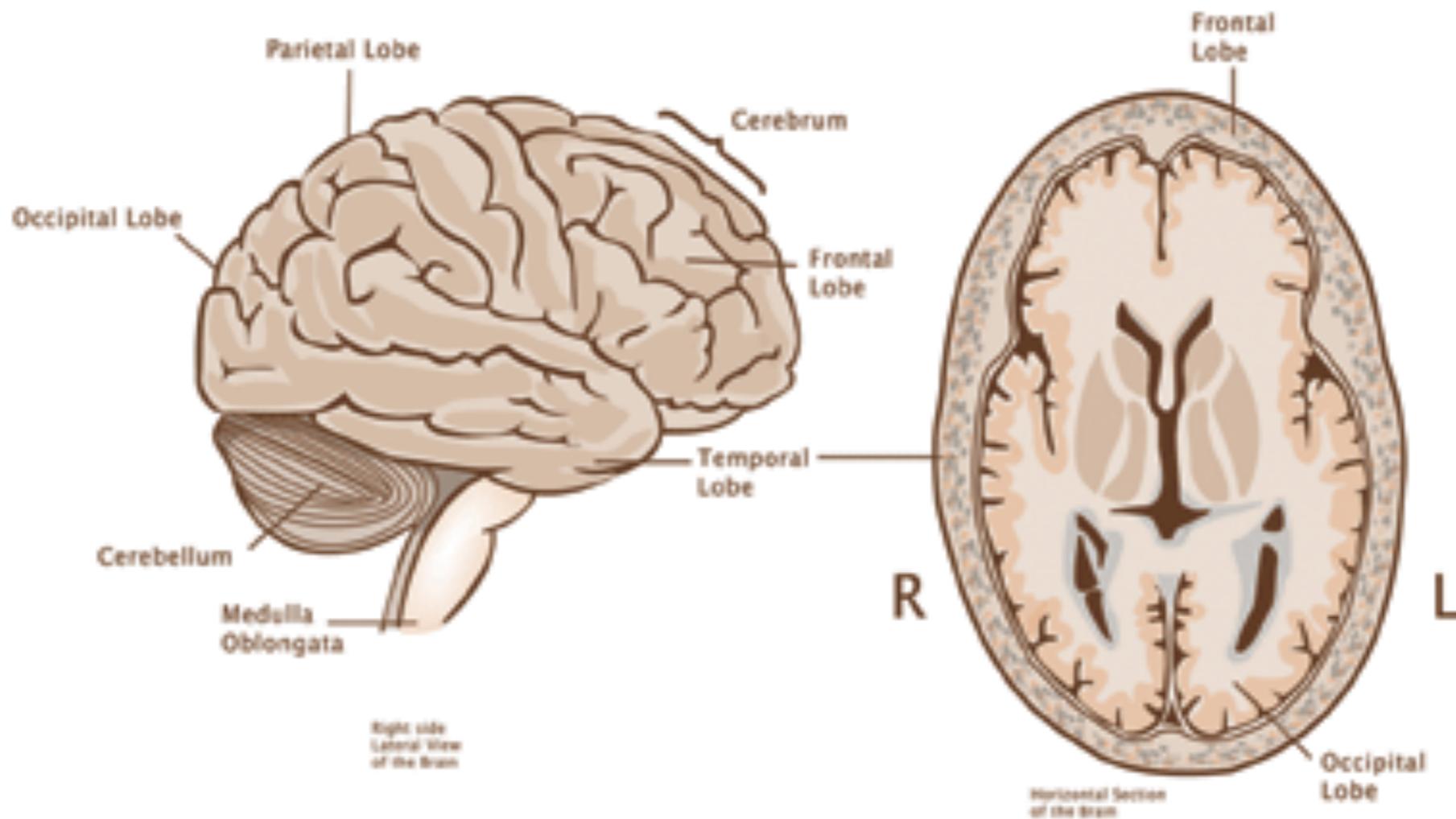


# In practice

- ✱ Best used for field names that are not likely to change
- ✱ In one app, memory usage reduced from 95mb to 19mb

# Model Human Behavior

## Anatomy of the Brain

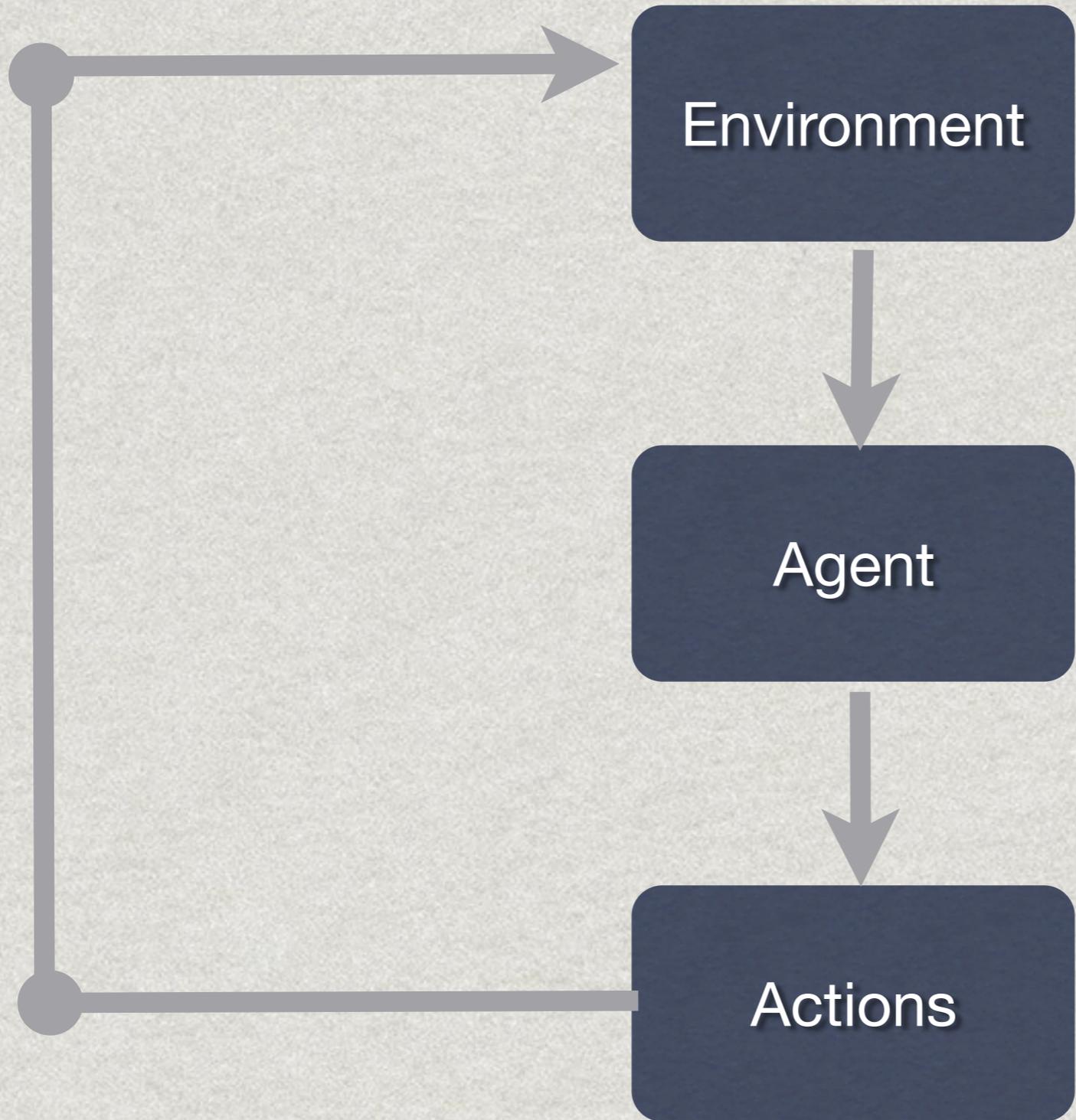


# Why TclOO?

- ✱ It's built in
- ✱ It allows me to do devious and interesting things
- ✱ "It's very, very fast"

# The Agent Based Model

- \* Simulates coordinated human effort
- \* Models the “fog of war”
- \* Agents act on imperfect knowledge

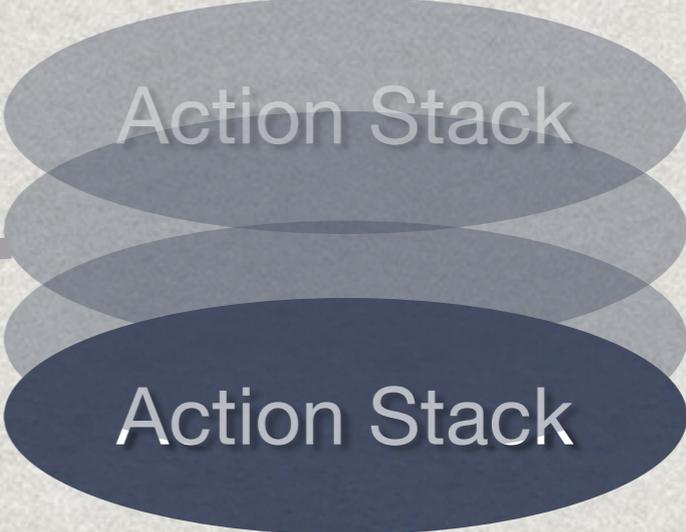




**FACT TABLE**

c141	on fire
v13	is hot

Behaviors  
Behaviors  
Behaviors  
Behaviors



# Complex Case

- \* Objective 1: Flee to Safety
- \* Objective 2: Report location of fire

# FLY TO SAFETY

Walk to:  
Refuge point

# REPORT LOCATION OF FIRE

Walk to:  
Phone

Use. Phone

Talk to: Operator

Refuge point

# Two places at once...

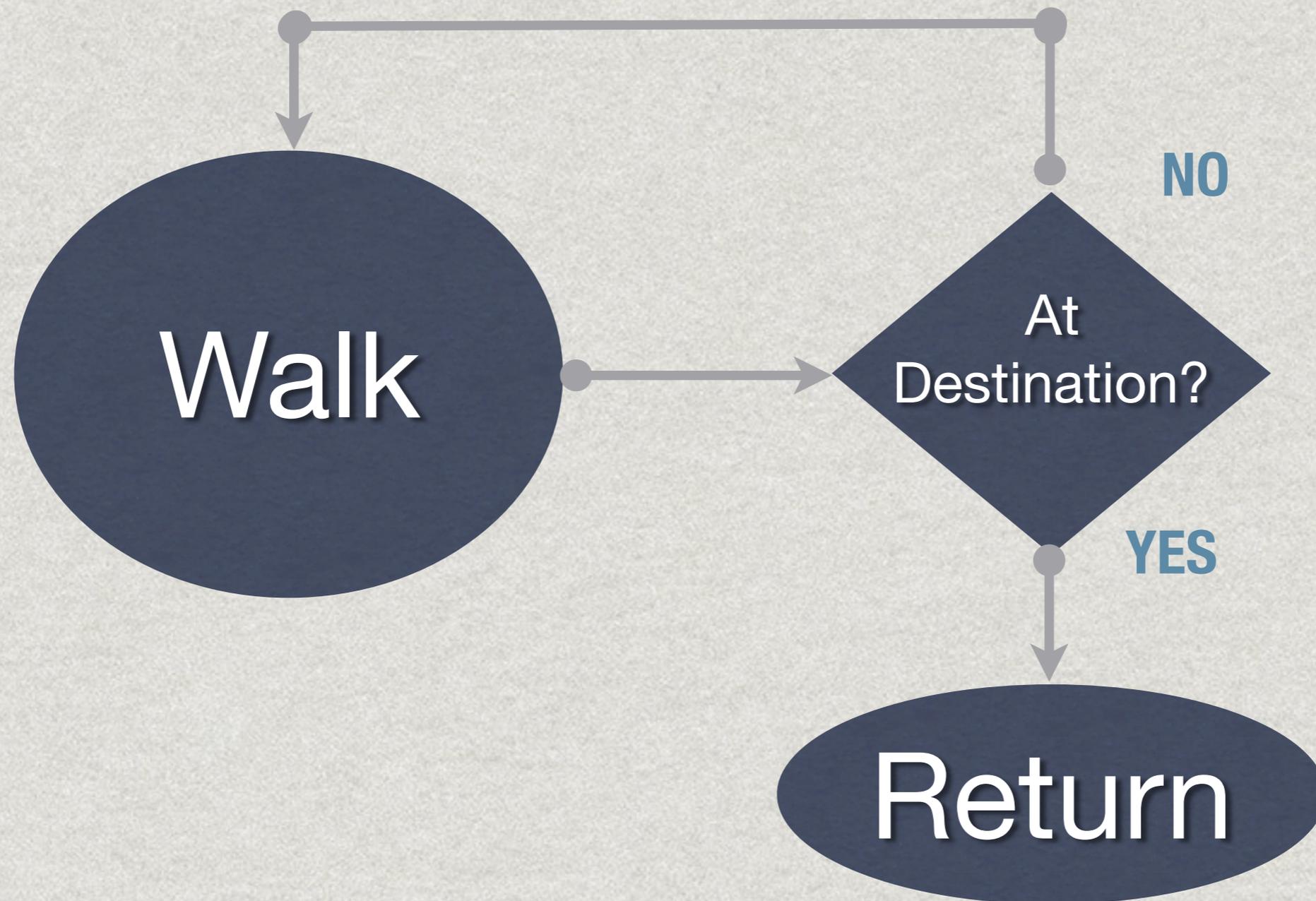
Walk to:  
Phone

Use: Phone

Talk to: Operator

Refuge point

# FSMs...



But in some cases...

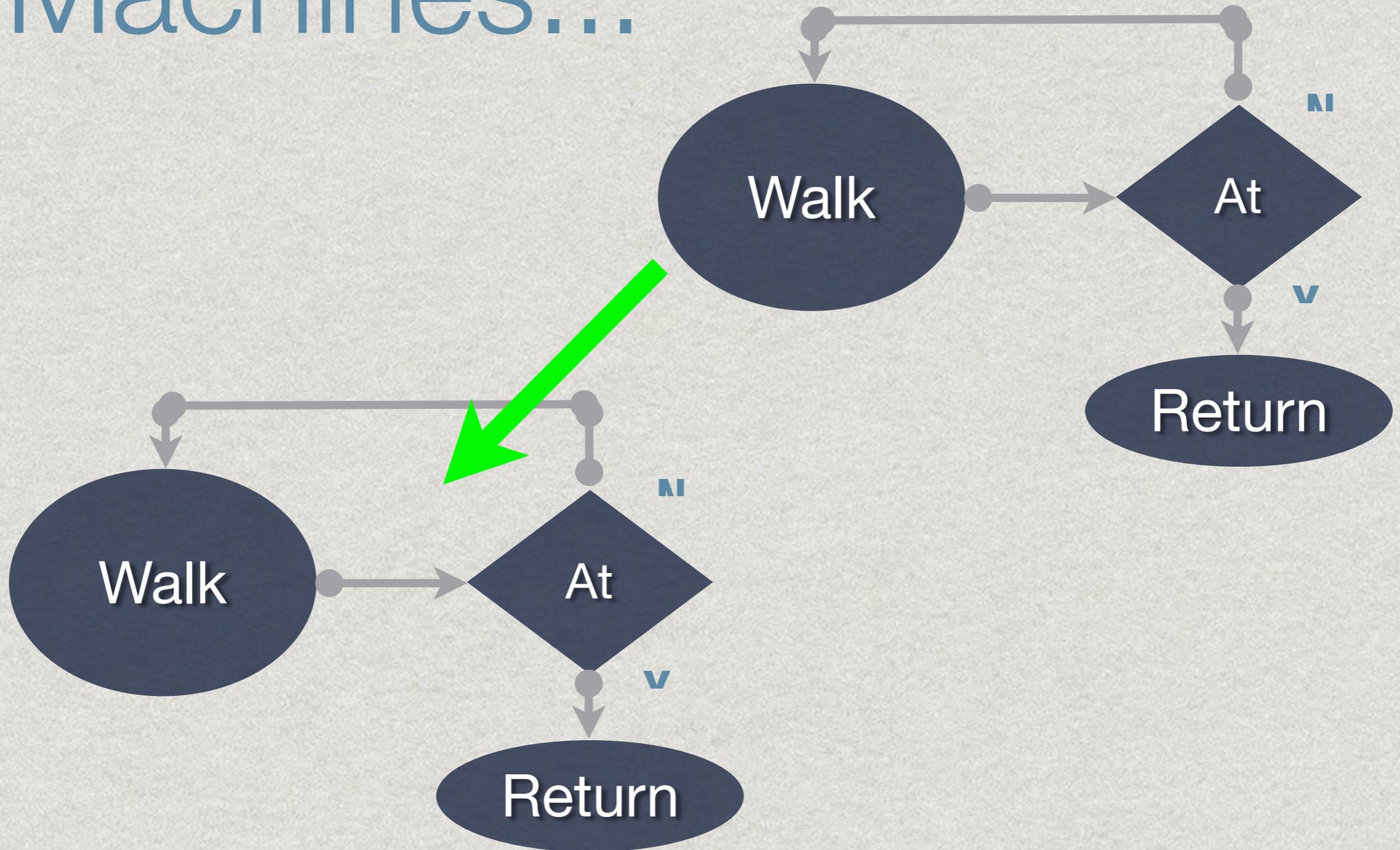
Walk to:  
Phone

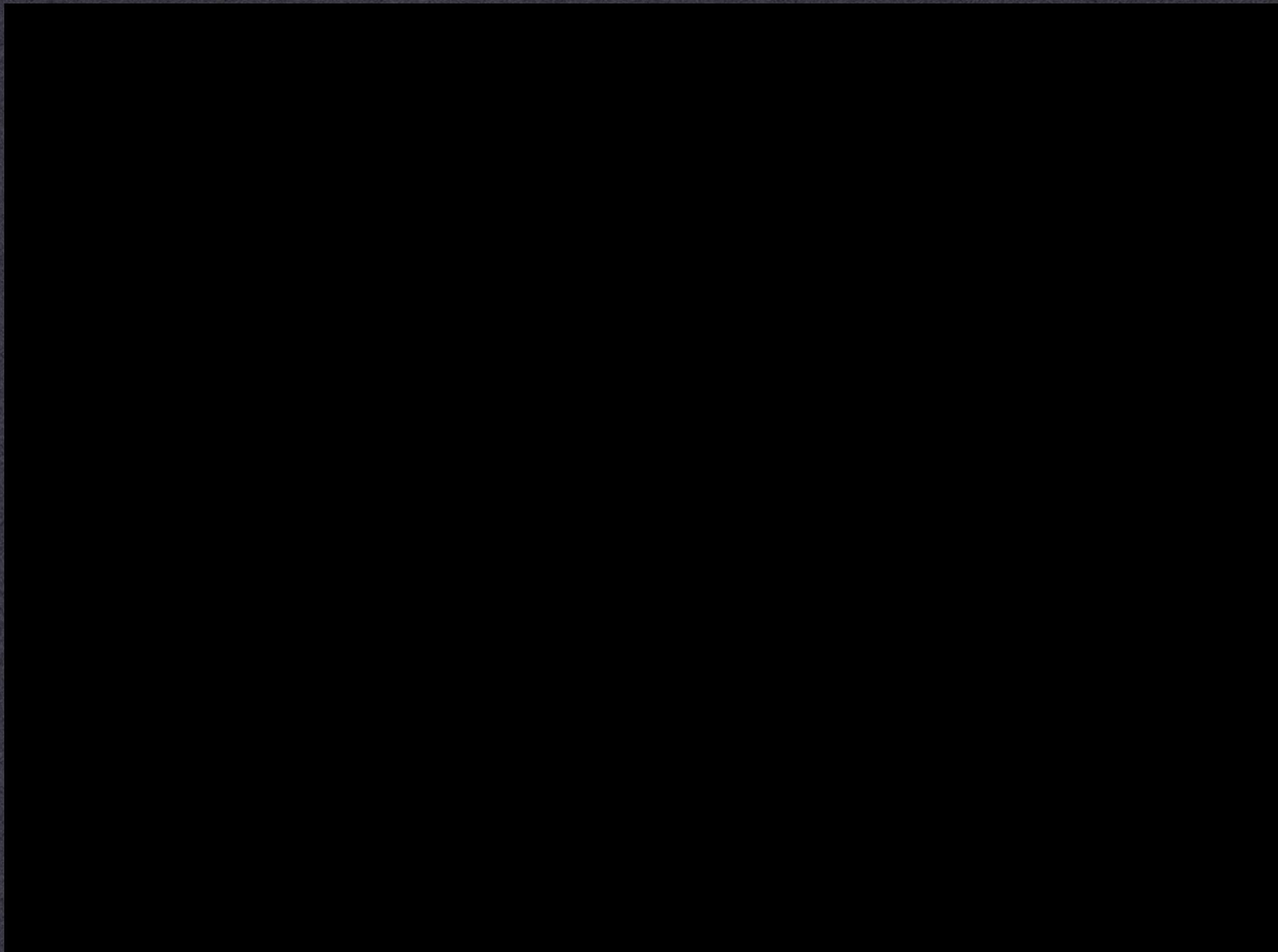
Use. Phone

Talk to doctor

Walk to:  
Refuge point

# F#@%&\*g State Machines...





And Now...

*A live demo*