

Models of Attack and Defense

Christian Collberg

University of Arizona

Models of ~~Attack~~ and Defense

Christian Collberg

University of Arizona

Models of ~~Attack~~ and Defense 2

Christian Collberg

University of Arizona

Surreptitious Software — Problems & Techniques

Problem

Technique

Surreptitious Software — Problems & Techniques



Problem

Technique

Software Piracy



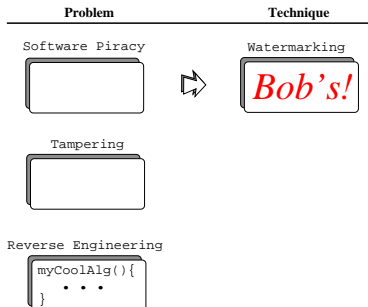
Surreptitious Software — Problems & Techniques

Problem	Technique
Software Piracy	
Tampering	

Surreptitious Software — Problems & Techniques

Problem	Technique
Software Piracy	
Tampering	
Reverse Engineering	<pre>myCoolAlg(){ . . . }</pre>




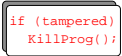
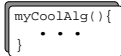

Surreptitious Software — Problems & Techniques







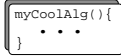

Surreptitious Software — Problems & Techniques

Problem		Technique
Software Piracy	⇒	Watermarking <i>Bob's!</i>
Tampering	⇒	Tamperproofing <code>if (tampered) KillProg();</code>
Reverse Engineering		<code>myCoolAlg(){ ... }</code>

Surreptitious Software — Problems & Techniques





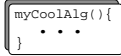

Problem		Technique
Software Piracy	⇒	Watermarking
		
Tampering	⇒	Tamperproofing
		
Reverse Engineering	⇒	Obfuscation
		

Surreptitious Software — Problems & Techniques

Problem		Technique
Software Piracy	⇒	Watermarking
		
Tampering	⇒	Tamperproofing
		
Reverse Engineering	⇒	Obfuscation
		

- Techniques are **code transformations**.

Surreptitious Software — Problems & Techniques

Problem		Technique
Software Piracy	⇒	Watermarking
		
Tampering	⇒	Tamperproofing
		
Reverse Engineering	⇒	Obfuscation
		

- Techniques are **code transformations**.
- Tools **compile** unprotected programs to protected programs.

- **Observations:**
 - 1 many software protection algorithms use similar transformations.

- **Observations:**
 - ① many software protection algorithms use similar transformations.
 - ② protection schemes in the natural world seem to use similar transformations.

- **Observations:**
 - ① many software protection algorithms use similar transformations.
 - ② protection schemes in the natural world seem to use similar transformations.
- **Conjectures:**
 - ① protection in the natural world can teach us something about protection in our artificial world.

- **Observations:**
 - ① many software protection algorithms use similar transformations.
 - ② protection schemes in the natural world seem to use similar transformations.
- **Conjectures:**
 - ① protection in the natural world can teach us something about protection in our artificial world.
 - ② there is a finite number of transformations.

Program

- **Observations:**
 - ① many software protection algorithms use similar transformations.
 - ② protection schemes in the natural world seem to use similar transformations.
- **Conjectures:**
 - ① protection in the natural world can teach us something about protection in our artificial world.
 - ② there is a finite number of transformations.
- **Program:**
 - identify a set of primitives that describe and classify software protection algorithms.

Goals

- We want a model that
 - ① helps us analyze published algorithms,

Goals

- We want a model that
 - 1 helps us analyze published algorithms,
 - 2 serves as a design pattern for future algorithms.

Goals

- We want a model that
 - ① helps us analyze published algorithms,
 - ② serves as a design pattern for future algorithms.
- This talk:
 - ① Model notation;
 - ② The primitives of the model;
 - ③ Examples from biology, history, and computing;
 - ④ Discussion (useful? complete?).

Model notation

The model notation consists of

- ① Frames
- ② Properties
- ③ Transformations
- ④ Demons

Model notation

The model notation consists of

- ① Frames
 - ② Properties
 - ③ Transformations
 - ④ Demons
- **Frames** are a knowledge representation device used in AI.

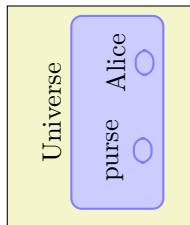
Model notation

The model notation consists of

- ① Frames
 - ② Properties
 - ③ Transformations
 - ④ Demons
- **Frames** are a knowledge representation device used in AI.
 - Basic idea: **Frame-to-frame transformations** represent transformations from unprotected to protected universes.

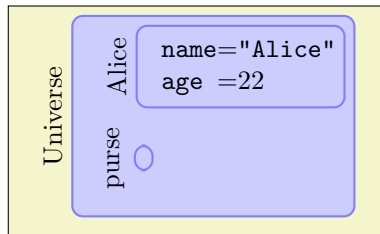
Frames

Frames represent a universe of object.



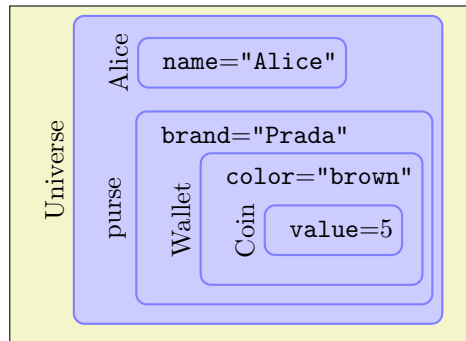
Frames

Frame slots describe **object properties**.



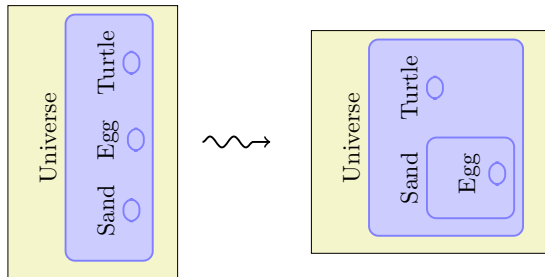
Frames

Frames can **contain** other frames.



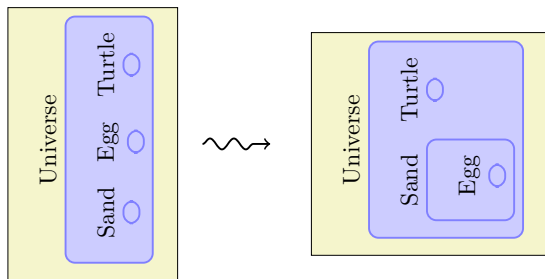
Protection Strategies

- Protection strategies: functions mapping **frames to frames**.



Protection Strategies

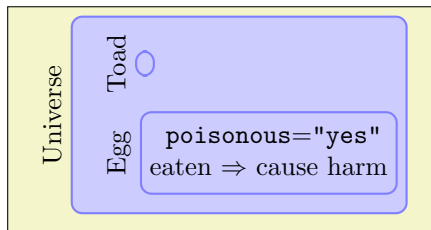
- Protection strategies: functions mapping frames to frames.



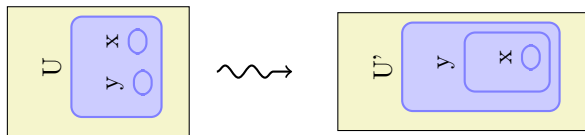
- Defense-in-depth**: Layer protection schemes using function composition.

Demons

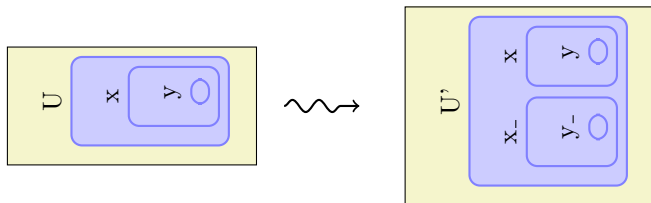
- Slots can have **demons** which fire under the right circumstances.



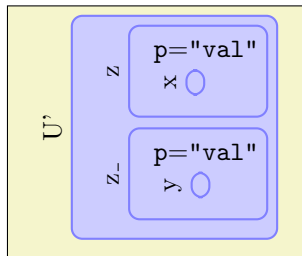
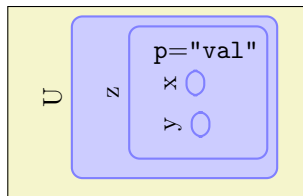
cover



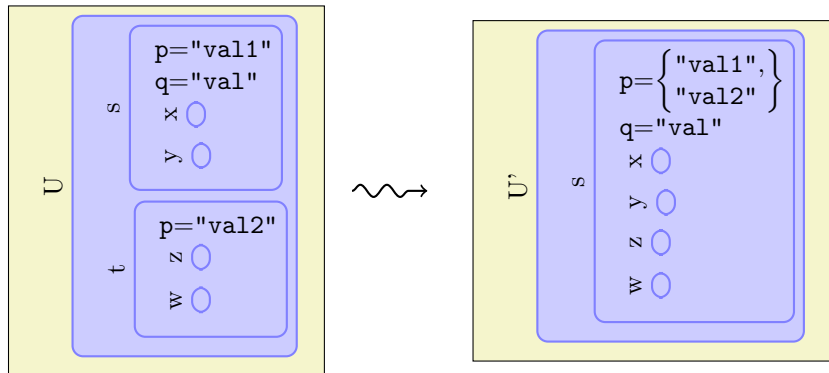
duplicate



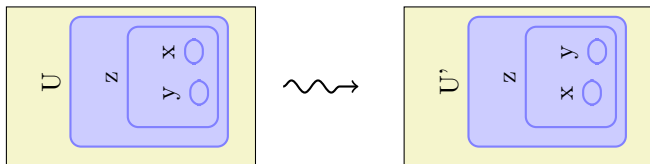
split



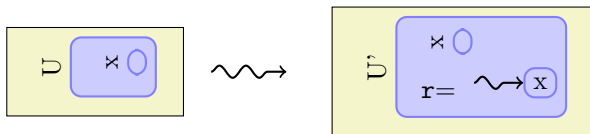
merge



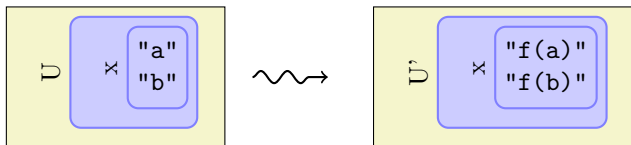
reorder



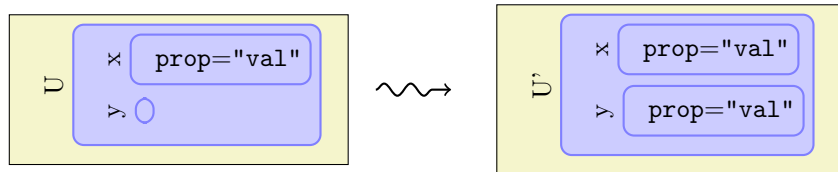
indirect



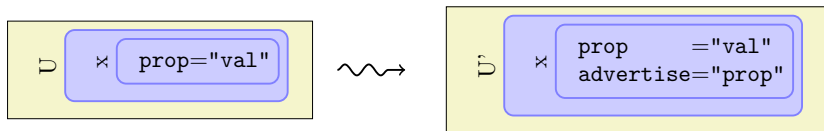
map



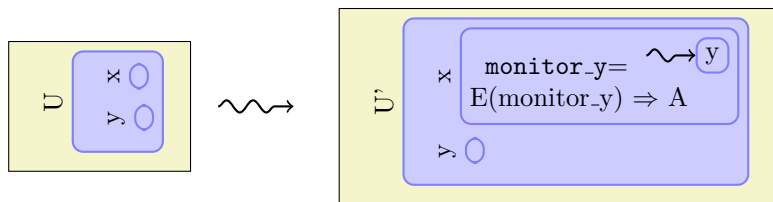
mimic



advertise



detect-respond



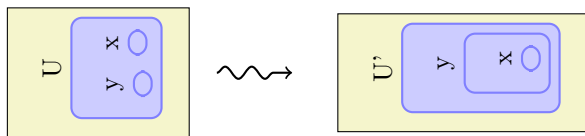
dynamic

$$x \rightarrow fx \rightarrow f(fx) \rightarrow f(f(fx)) \dots$$

compose

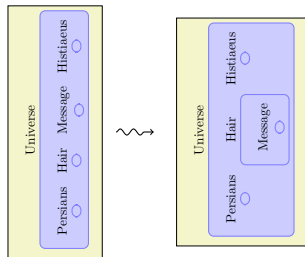
$$(f \circ g)(x) = f(g(x))$$

Primitive #1: Cover

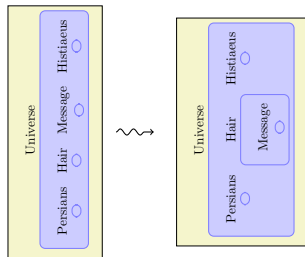


- Fundamental way of protecting something: **cover** it with another object!

The Cover Primitive — History



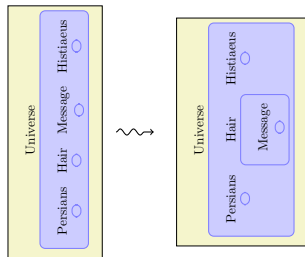
The Cover Primitive — History



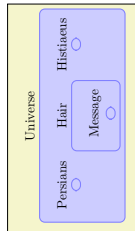
The Cover Primitive — History



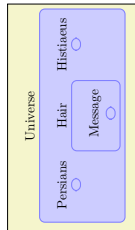
Simulation!



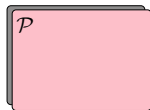
The Cover Primitive — History



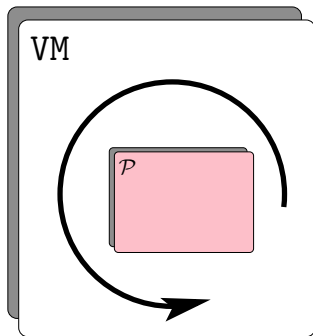
The Cover Primitive — History



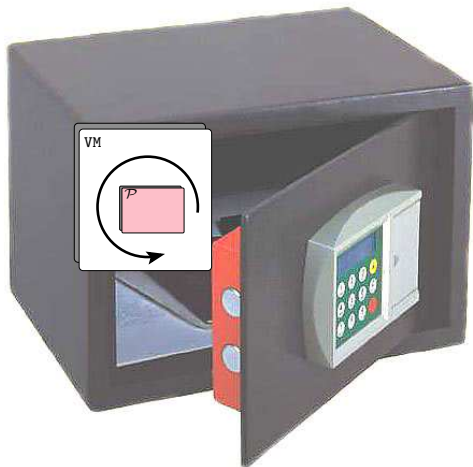
The Cover Primitive — Software/Hardware



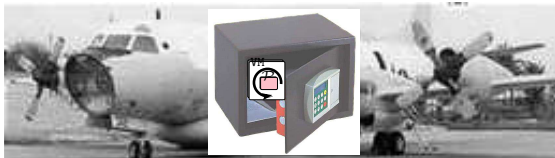
The Cover Primitive — Software/Hardware



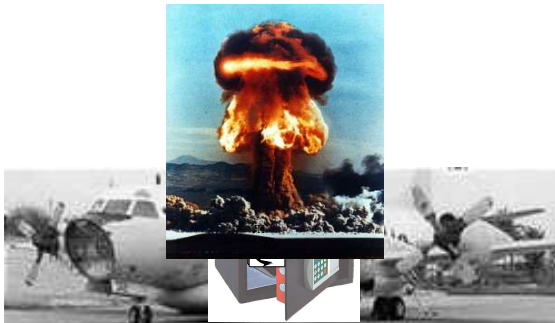
The Cover Primitive — Software/Hardware



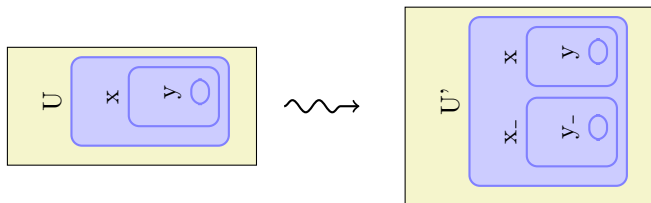
The Cover Primitive — Software/Hardware



The Cover Primitive — Software/Hardware

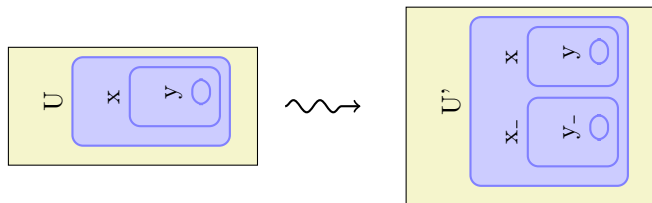


Primitive #2: Duplicate



- Add **decoy** objects to force an attacker to consider more items.

Primitive #2: Duplicate



- Add **decoy** objects to force an attacker to consider more items.
- Add a **clone** of an object to force an attacker to destroy both copies.

The Duplicate Primitive — Biology



California newt

The Duplicate Primitive — Biology



California newt



- **clone:** 7-30 eggs;

The Duplicate Primitive — Biology



California newt



- **clone:** 7-30 eggs;
- **cover:** Eggs covered by a gel-like membrane;

The Duplicate Primitive — Biology



California newt

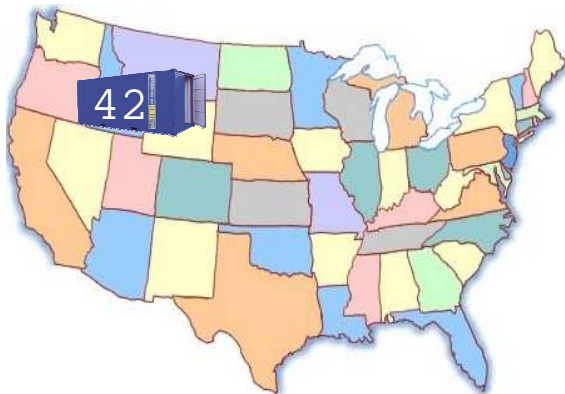


- **clone:** 7-30 eggs;
- **cover:** Eggs covered by a gel-like membrane;
- **detect-respond:** Membrane contains tarichatoxin.

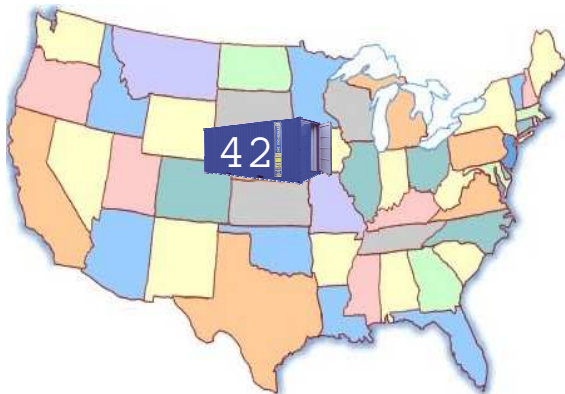
The Duplicate Primitive — History



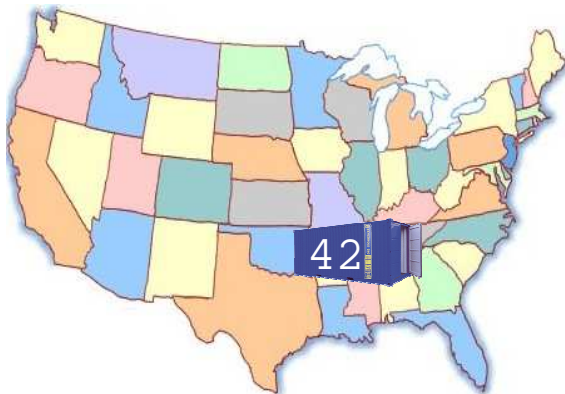
The Duplicate Primitive — History



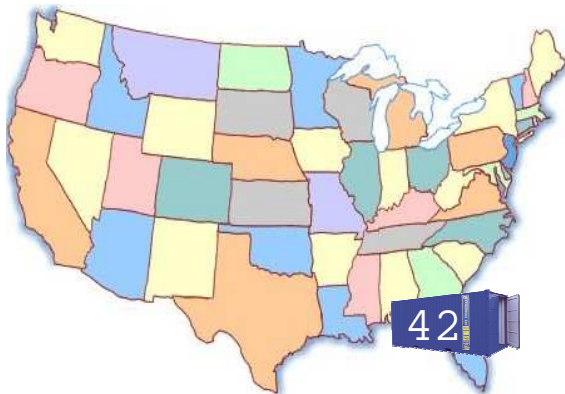
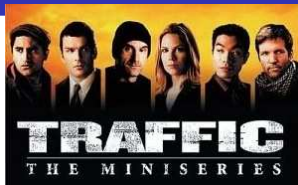
The Duplicate Primitive — History



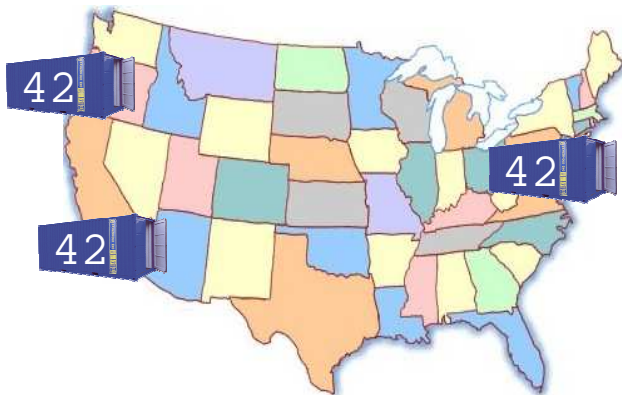
The Duplicate Primitive — History



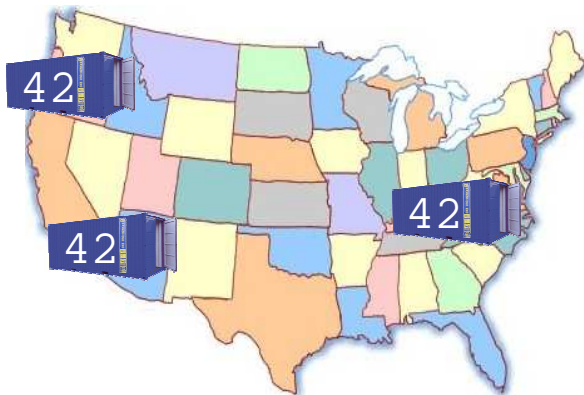
The Duplicate Primitive — History



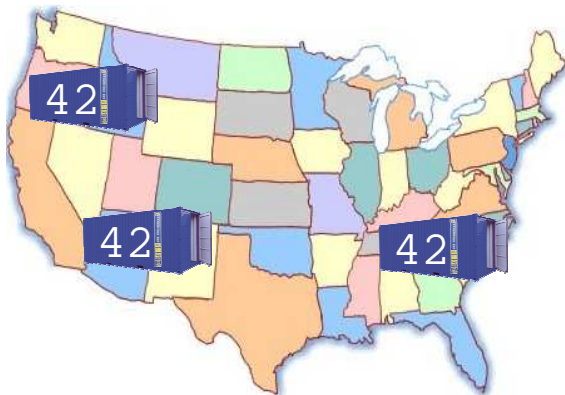
The Duplicate Primitive — History



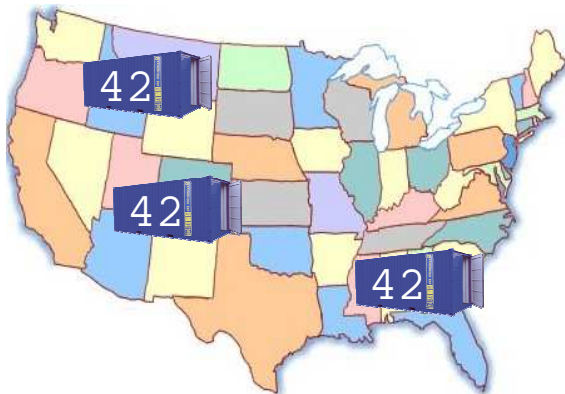
The Duplicate Primitive — History



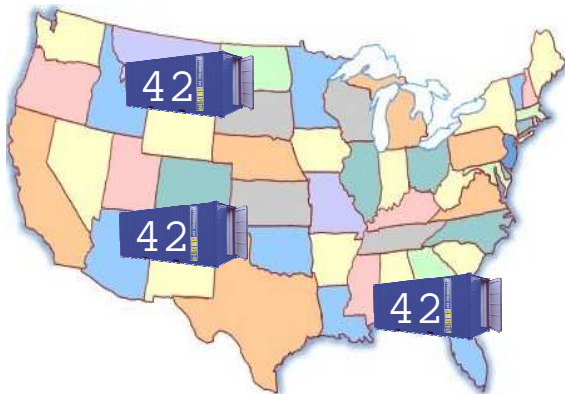
The Duplicate Primitive — History



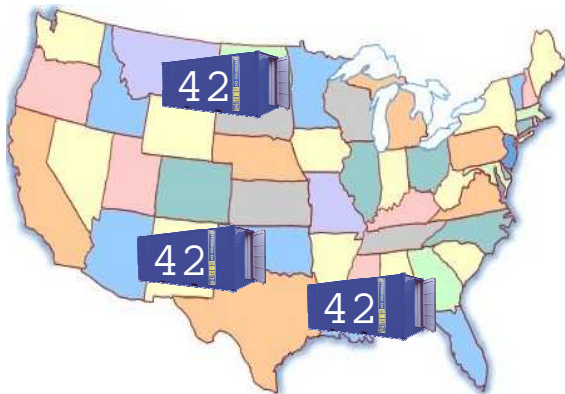
The Duplicate Primitive — History



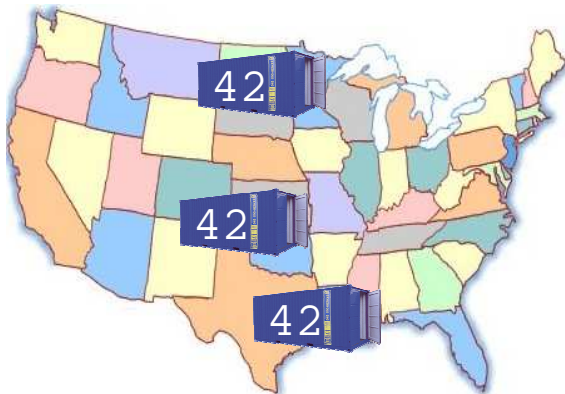
The Duplicate Primitive — History



The Duplicate Primitive — History



The Duplicate Primitive — History

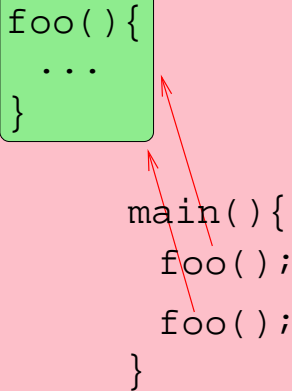


The Duplicate Primitive — Software

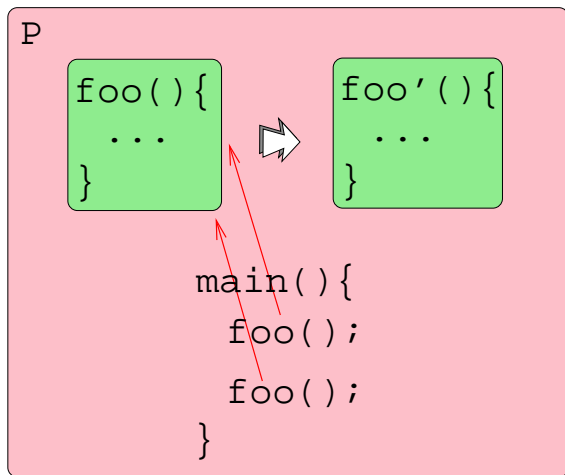
P

```
foo() {  
  ...  
}
```

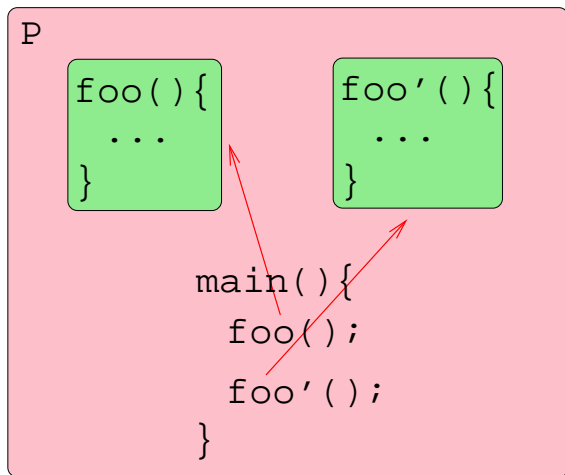
```
main() {  
  foo();  
  foo();  
}
```



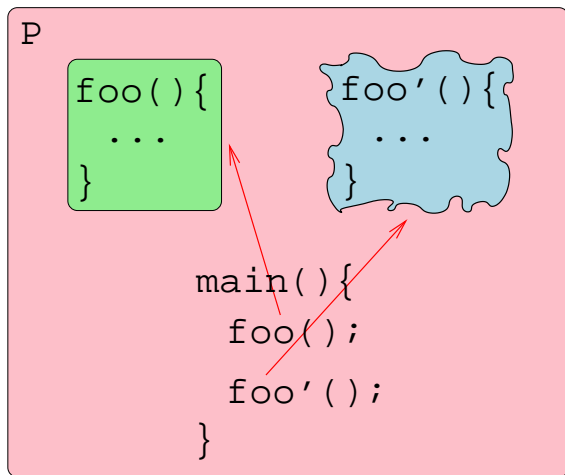
The Duplicate Primitive — Software



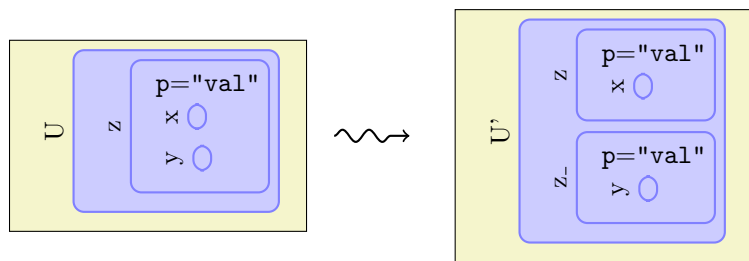
The Duplicate Primitive — Software



The Duplicate Primitive — Software

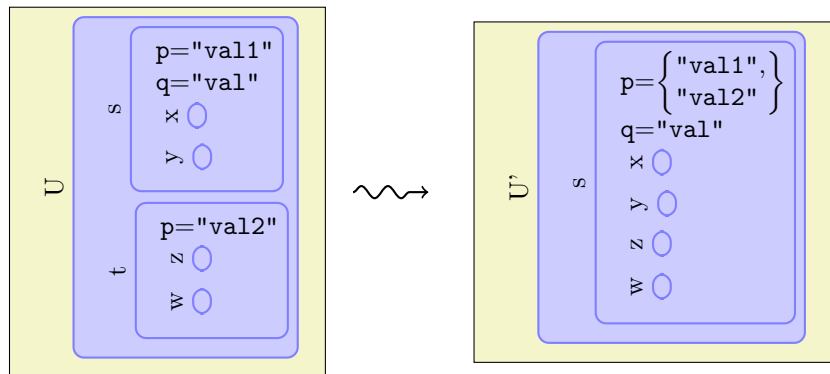


Primitives #3-4: Split/Merge



- Split an object, hide/protect the pieces!

The Split/Merge Primitives



- Merge unrelated object to sow confusion!

The Split/Merge Primitives — Biology



- **Autotomy** — when attacked, **split**, and give up on one part.

The Split/Merge Primitives — Biology



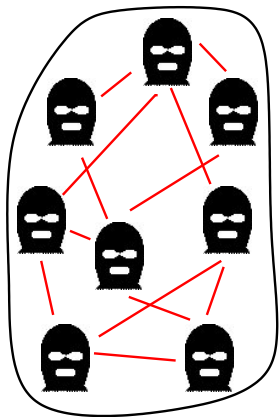
- **Autotomy** — when attacked, **split**, and give up on one part.

The Split/Merge Primitives — Biology

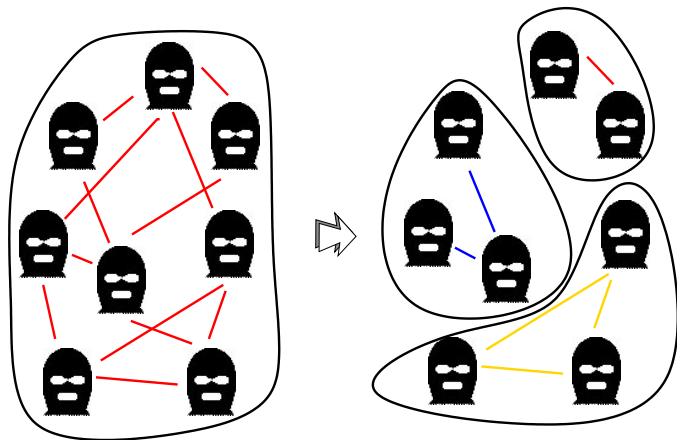


- **Autotomy** — when attacked, **split**, and give up on one part.
- Often combined with **detect-respond** or regeneration.

The Split/Merge Primitives — History

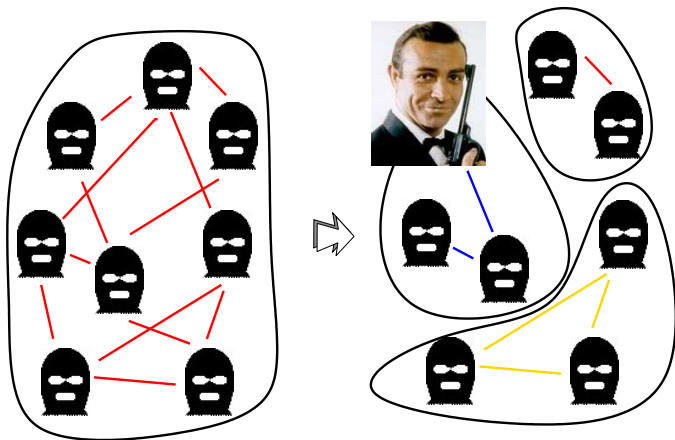


The Split/Merge Primitives — History



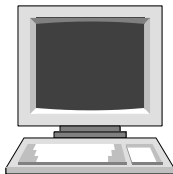
- Terrorist networks split into autonomous cells.

The Split/Merge Primitives — History



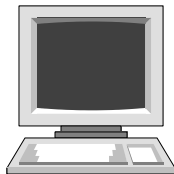
- Terrorist networks split into autonomous cells.

The Split/Merge Primitives — Software



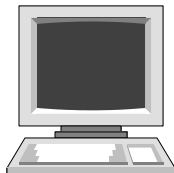
```
10000000010011110001000000000000  
0000000011100000000000000010000  
0000000000110111111000000000000  
00000000111000000010000000011000  
00000000001111110000000000000000  
10100101010101101010110010101010  
10100000000011100000000000000110
```

The Split/Merge Primitives — Software



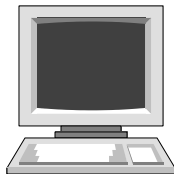
```
10000000010011110001000000000000  
000000001110000000000000000010000  
0000000001101111110000000000000  
00000000111000000010000000011000  
0000000001111110000000000000000  
10100101010101101010110010101010  
10100000000011100000000000000110
```

The Split/Merge Primitives — Software



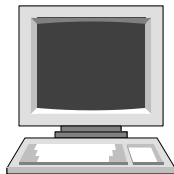
```
10000000010011110001000000000000  
00000000111000000000000000010000  
00000000011011111100000000000000  
00000000111000000010000000011000  
00000000011111100000000000000000  
10100101010101101010110010101010  
10100000000011100000000000000110
```

The Split/Merge Primitives — Software



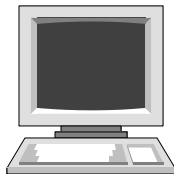
```
10000000010011110001000000000000  
0000000011100000000000000010000  
000000000110111111000000000000  
00000000111000000010000000011000  
00000000001111110000000000000000  
10100101010101101010110010101010  
10100000000011100000000000000110
```

The Split/Merge Primitives — Software



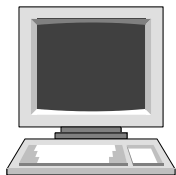
```
10000000010011110001000000000000  
0000000011100000000000000010000  
000000000110111111000000000000  
00000000111000000010000000011000  
00000000011111100000000000000000  
10100101010101101010110010101010  
10100000000011100000000000000110
```


The Split/Merge Primitives — Software



```
10000000010011110001000000000000  
0000000011100000000000000010000  
000000000110111111000000000000  
0000000011100000001000000011000  
0000000001111110000000000000000  
10100101010101101010110010101010  
1010000000001110000000000000110
```

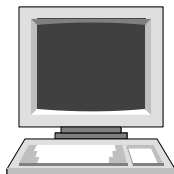
The Split/Merge Primitives — Software



```
10000000010011110001000000000000  
0000000011100000000000000010000  
0000000000110111111000000000000  
00000000111000000010000000011000  
00000000001111110000000000000000  
10100101010101101010110010101010  
10100000000011100000000000000110
```

Key?

The Split/Merge Primitives — Software



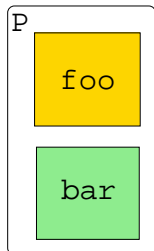
```
10000000010011110001000000000000
0000000011100000000000000010000
000000000110111111000000000000
00000000111000000010000000011000
00000000001111110000000000000000
10100101010101101010110010101010
10100000000011100000000000000110
```



```
10101001010011110001000000000000
00000000111000001010101000010000
00000000011011111100000000000000
00000000111000000010000000011000
0000000000111110101010100000000000
0000111100000011110000000000000000
10100000000011100000000001010110
```

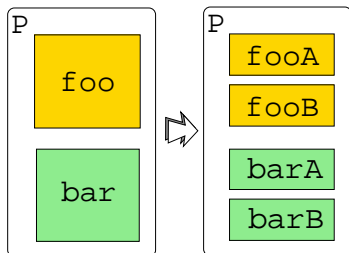
The Split/Merge Primitives — Software

- **Split/Merge** for code obfuscation.



The Split/Merge Primitives — Software

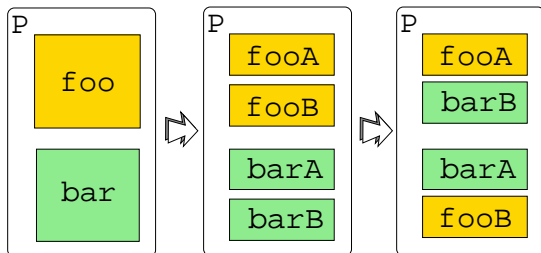
- **Split/Merge** for code obfuscation.



- 1 **Split** functions,

The Split/Merge Primitives — Software

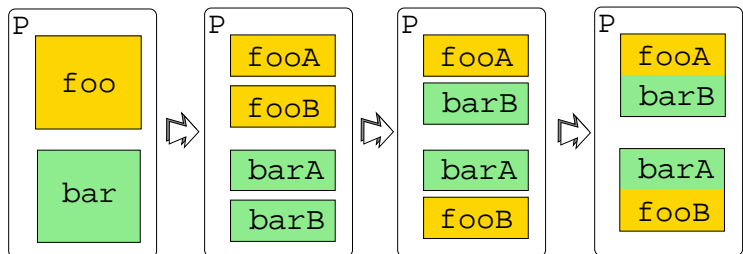
- **Split/Merge** for code obfuscation.



- 1 **Split** functions,
- 2 **Reorder** the pieces,

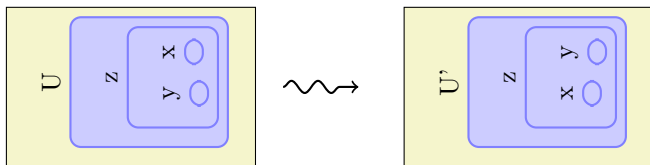
The Split/Merge Primitives — Software

- **Split/Merge** for code obfuscation.



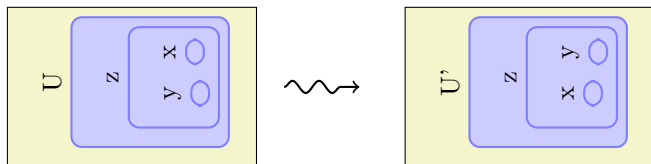
- 1 **Split** functions,
- 2 **Reorder** the pieces,
- 3 **Merge** back together.

Primitive #5: Reorder



- Randomly reorder to **sow** confusion.

Primitive #5: Reorder



- Randomly reorder to **sow** confusion.
- Reorder to **convey** information.

The Reorder Primitive — History



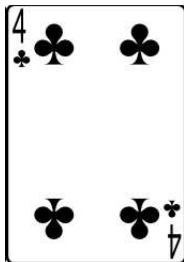
The Reorder Primitive — History



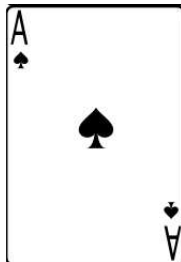
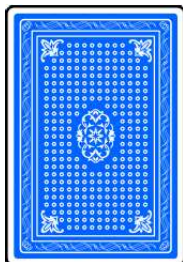
The Reorder Primitive — History



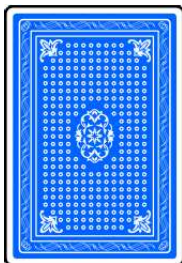
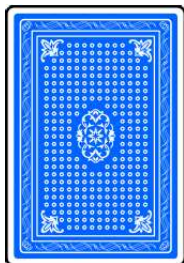
The Reorder Primitive — History



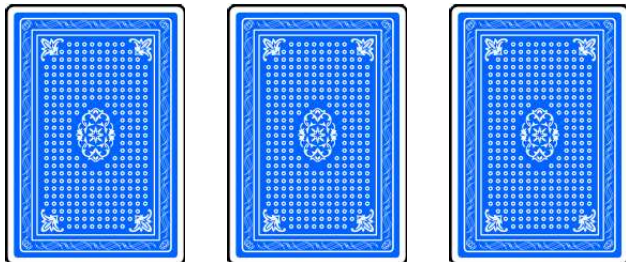
The Reorder Primitive — History



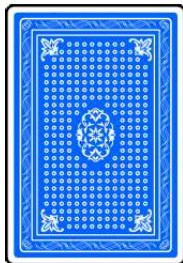
The Reorder Primitive — History



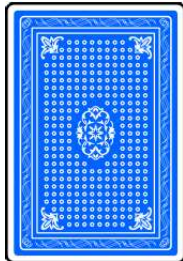
The Reorder Primitive — History



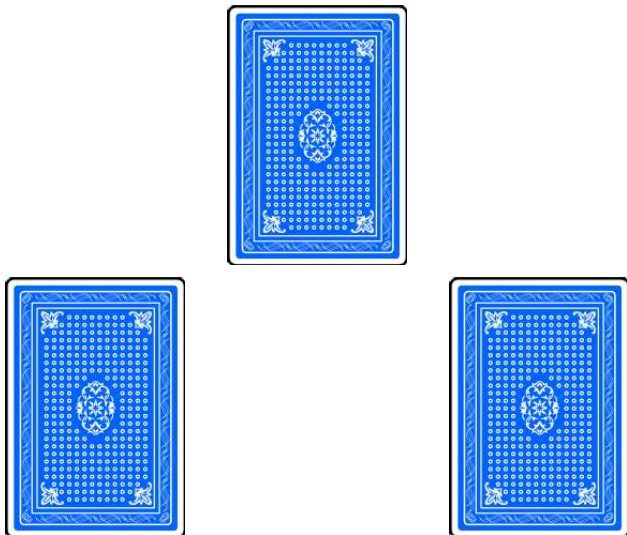
The Reorder Primitive — History



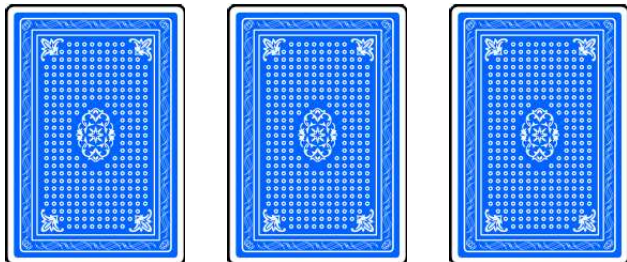
The Reorder Primitive — History



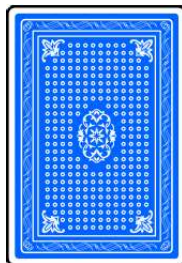
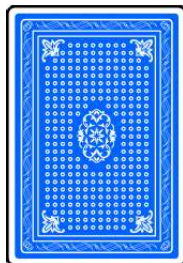
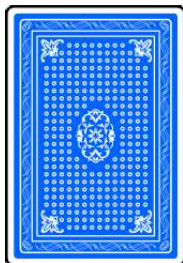
The Reorder Primitive — History



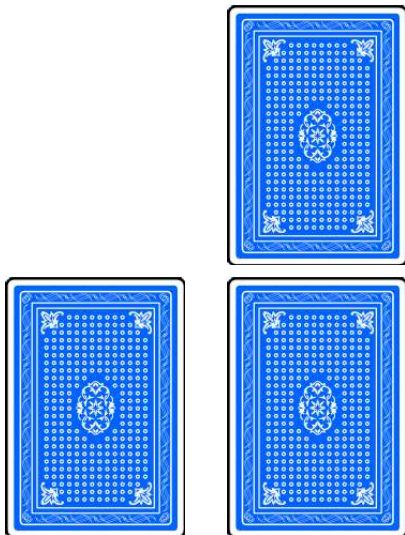
The Reorder Primitive — History



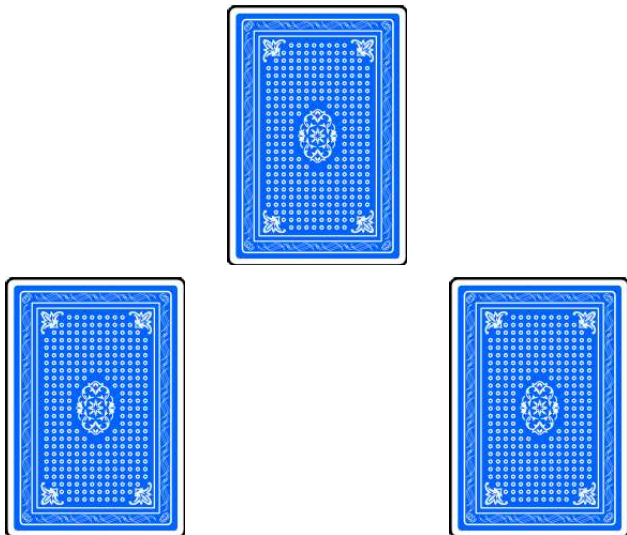
The Reorder Primitive — History



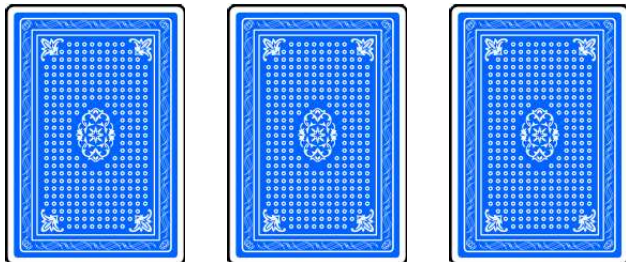
The Reorder Primitive — History



The Reorder Primitive — History



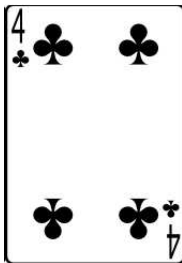
The Reorder Primitive — History



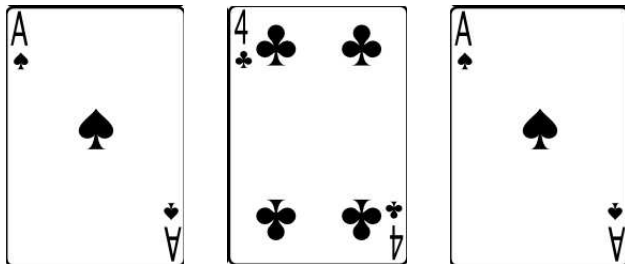
The Reorder Primitive — History

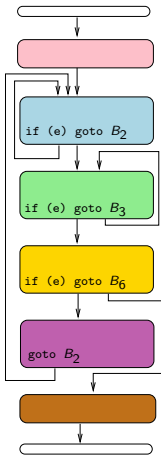


The Reorder Primitive — History

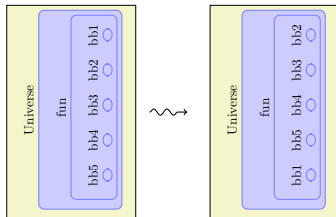


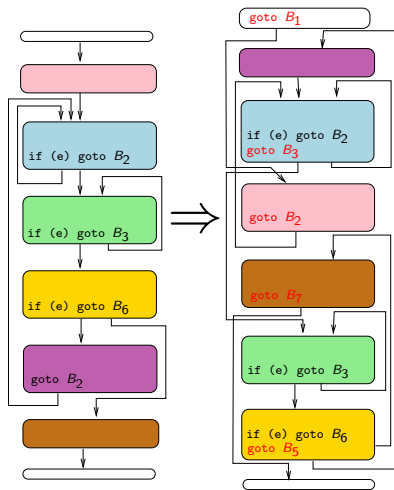
The Reorder Primitive — History



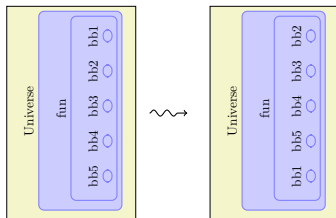


Reorder \Rightarrow watermark

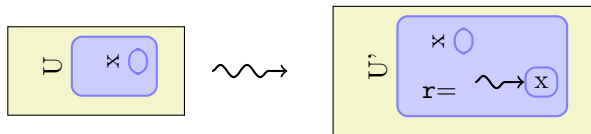




Reorder \Rightarrow watermark



Primitive #6: Indirect



- Add confusing levels of indirection!

The Indirect Primitive — History



Decoy

- Stop opposing team from stealing signs!

The Indirect Primitive — History



Decoy



Decoy

- Stop opposing team from stealing signs!

The Indirect Primitive — History



Decoy



Decoy



Indicator!

- Stop opposing team from stealing signs!
- Real sign follows **indicator** sign.

The Indirect Primitive — History



Decoy



Decoy



Indicator!



Real!

- Stop opposing team from stealing signs!
- Real sign follows **indicator** sign.

The Indirect Primitive — History

- **Indirection** is a common adventure movie plot device.
- A sequence of clues, each one pointing to the next one, leads to the treasure:



The Indirect Primitive — History

- **Indirection** is a common adventure movie plot device.
- A sequence of clues, each one pointing to the next one, leads to the treasure:



The Indirect Primitive — History

- **Indirection** is a common adventure movie plot device.
- A sequence of clues, each one pointing to the next one, leads to the treasure:



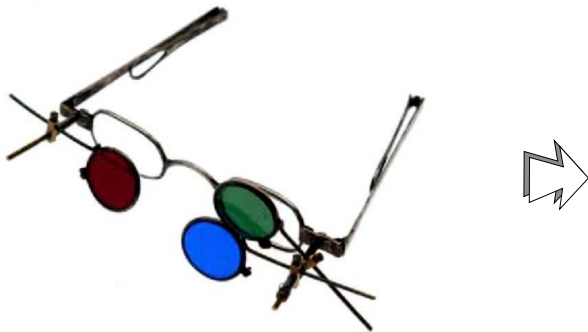
The Indirect Primitive — History

- **Indirection** is a common adventure movie plot device.
- A sequence of clues, each one pointing to the next one, leads to the treasure:



The Indirect Primitive — History

- **Indirection** is a common adventure movie plot device.
- A sequence of clues, each one pointing to the next one, leads to the treasure:



The Indirect Primitive — History

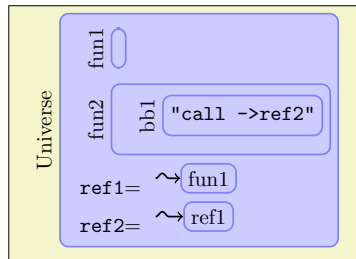
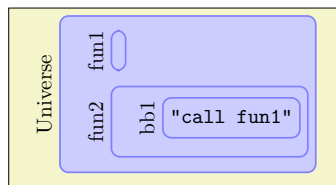
- **Indirection** is a common adventure movie plot device.
- A sequence of clues, each one pointing to the next one, leads to the treasure:



© 2004 Buena Vista Pictures and Jerry Bruckheimer Inc.

The Indirect Primitive — Software

```
void bar(){}  
void foo(){  
    bar();  
}
```

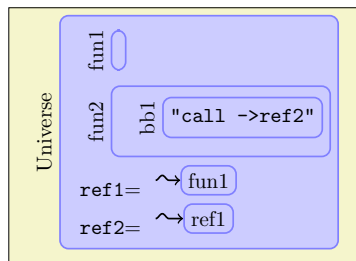
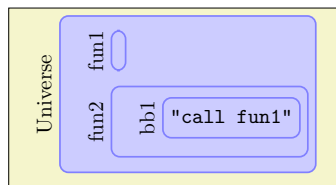


The Indirect Primitive — Software

```
void bar(){}  
void foo(){  
    bar();  
}
```



```
void bar(){}  
void (*x)() = &bar;  
void (**y)() = &x;  
void foo(){  
    (**y)();  
}
```

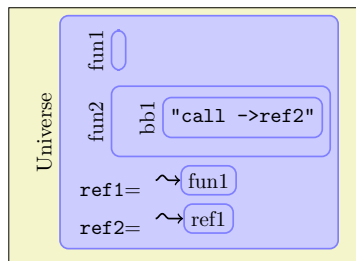
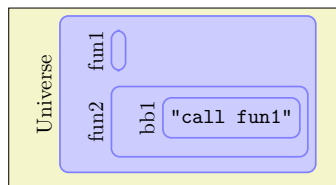


The Indirect Primitive — Software

```
void bar(){}  
void foo(){  
    bar();  
}
```



```
void bar(){}  
void (*x)() = &bar;  
void (**y)() = &x;  
void foo(){  
    (**y)();  
}
```

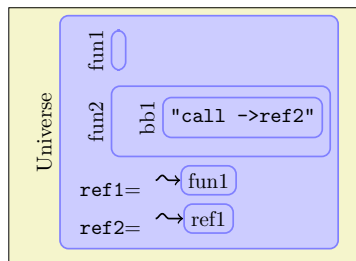
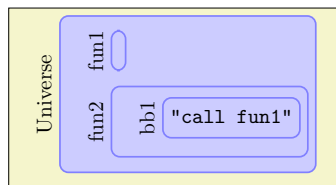


The Indirect Primitive — Software

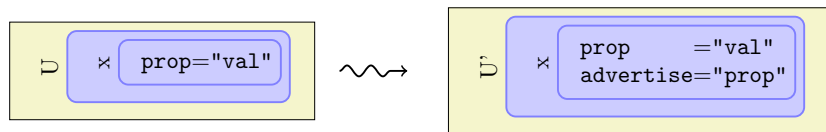
```
void bar(){}  
void foo(){  
    bar();  
}
```



```
void bar(){}  
void (*x)() = &bar;  
void (**y)() = &x;  
void foo(){  
    (**y)();  
}
```

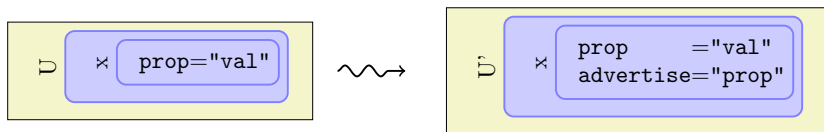


Primitive #9: Advertise



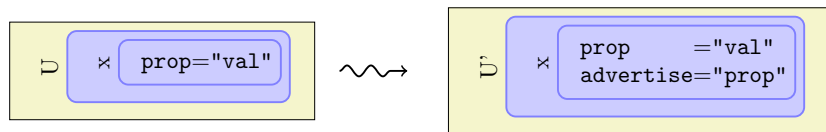
- The model assumes that objects keep all information about themselves secret. **Advertise** breaks this secrecy.

Primitive #9: Advertise



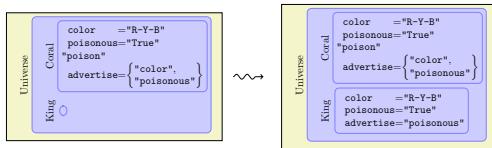
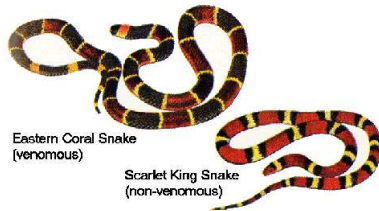
- The model assumes that objects keep all information about themselves secret. **Advertise** breaks this secrecy.
- **Advertise** your strengths!

Primitive #9: Advertise



- The model assumes that objects keep all information about themselves secret. **Advertise** breaks this secrecy.
- **Advertise** your strengths!
- Falsely **advertise** to hide your weaknesses!

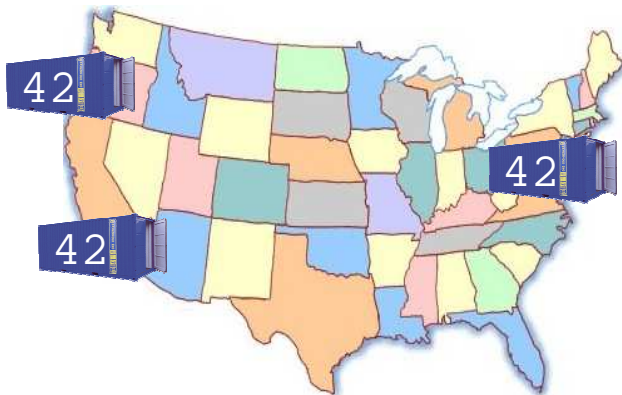
- **Aposematic coloration**: Toxic species use bright colors to **advertise** their harmfulness.
- Red-yellow-black stripes of the poisonous Coral snake.
- The non-venomous King snake uses **mimicking** and false **advertising**.



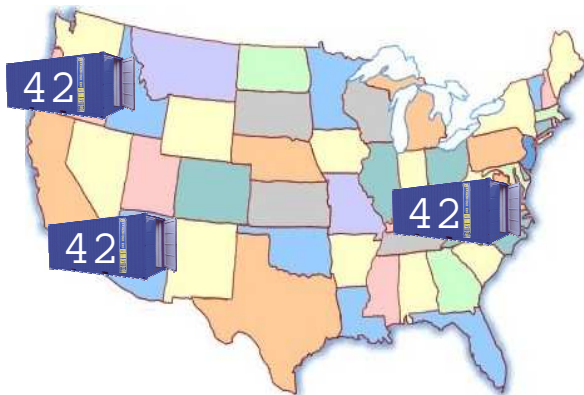
The Advertise Primitive — History



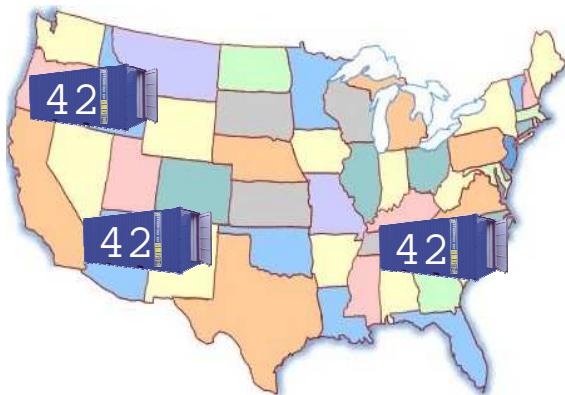
The Advertise Primitive — History



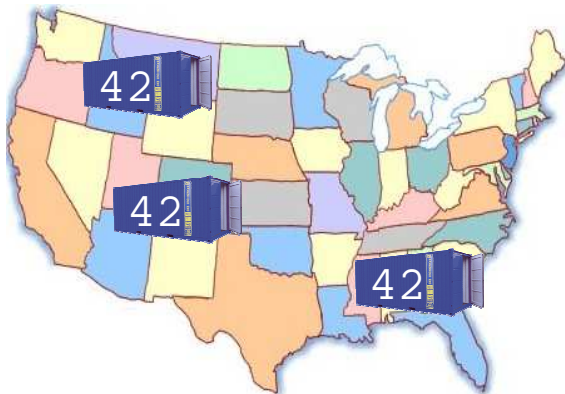
The Advertise Primitive — History



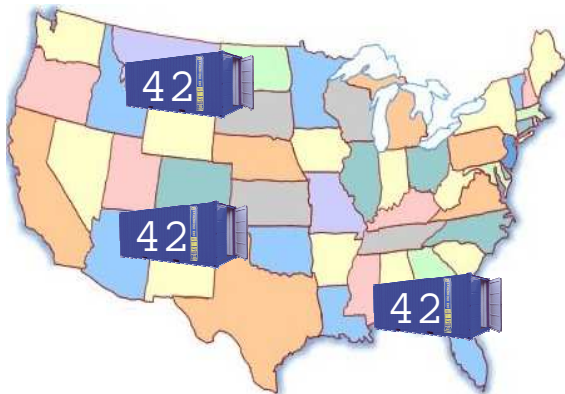
The Advertise Primitive — History



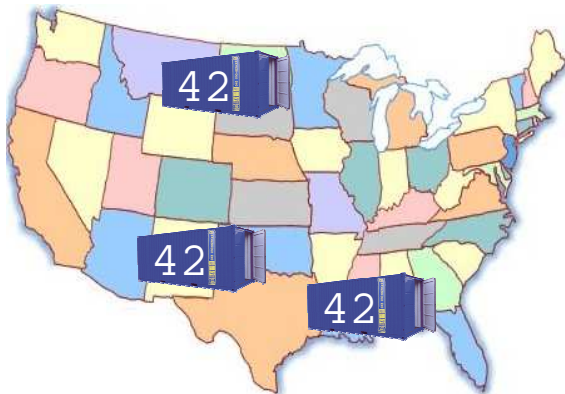
The Advertise Primitive — History



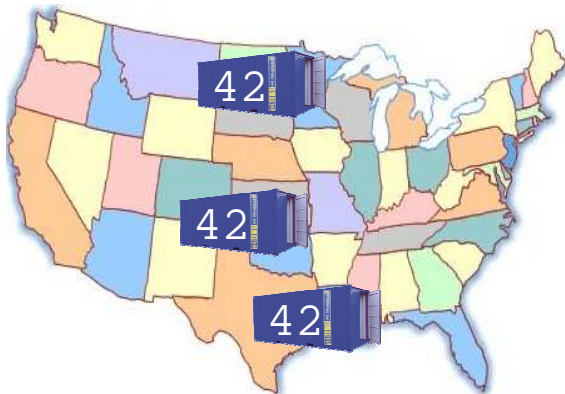
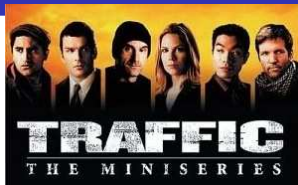
The Advertise Primitive — History



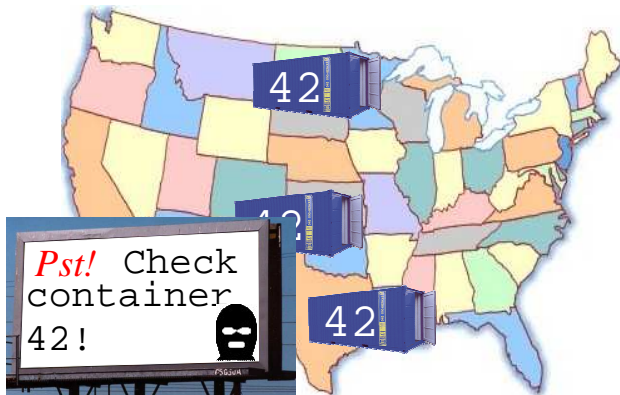
The Advertise Primitive — History



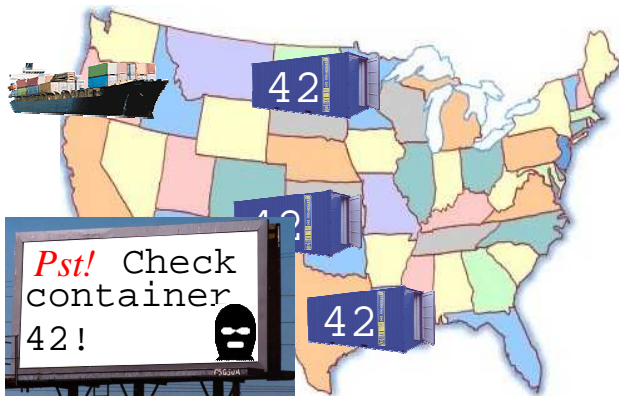
The Advertise Primitive — History



The Advertise Primitive — History



The Advertise Primitive — History



Primitive #11: Dynamic

x

- Repeatedly apply a primitive f to an object x .
- Fast movement, unpredictable movement, continuous evolution of defenses. . .

Primitive #11: Dynamic

$$x \rightarrow fx$$

- Repeatedly apply a primitive f to an object x .
- Fast movement, unpredictable movement, continuous evolution of defenses. . .

Primitive #11: Dynamic

$$x \rightarrow fx \rightarrow f(fx)$$

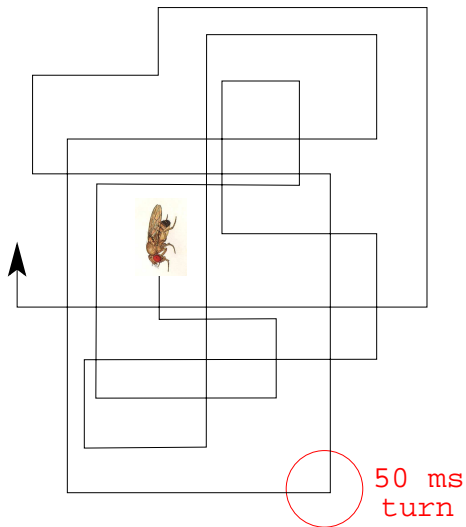
- Repeatedly apply a primitive f to an object x .
- Fast movement, unpredictable movement, continuous evolution of defenses. . .

Primitive #11: Dynamic

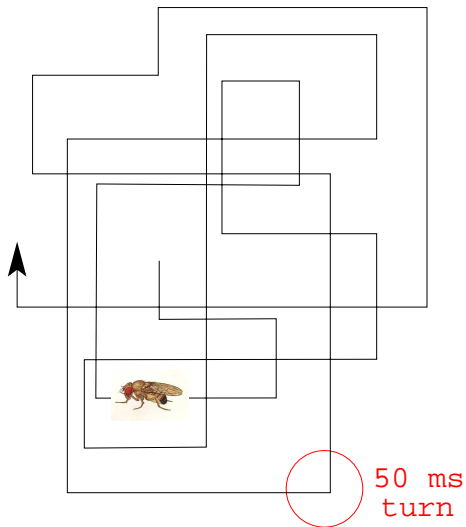
$$x \rightarrow fx \rightarrow f(fx) \rightarrow f(f(fx)) \dots$$

- Repeatedly apply a primitive f to an object x .
- Fast movement, unpredictable movement, continuous evolution of defenses. . .

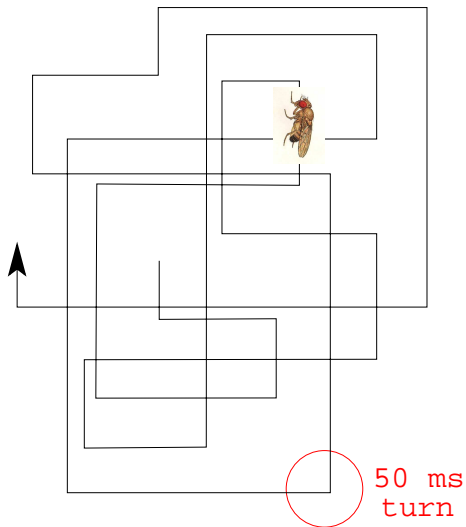
The Dynamic Primitive — Animals



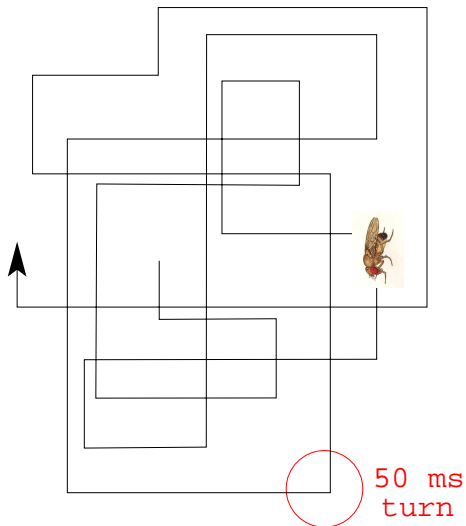
The Dynamic Primitive — Animals



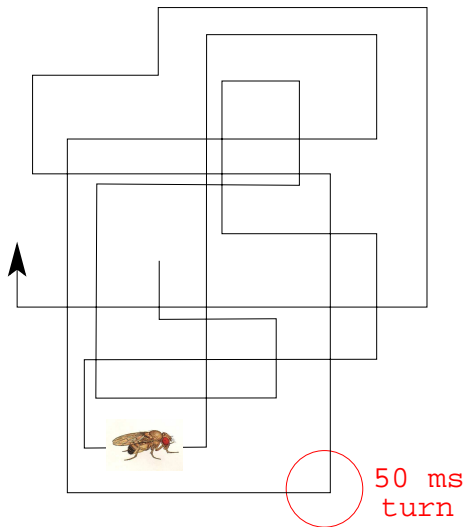
The Dynamic Primitive — Animals



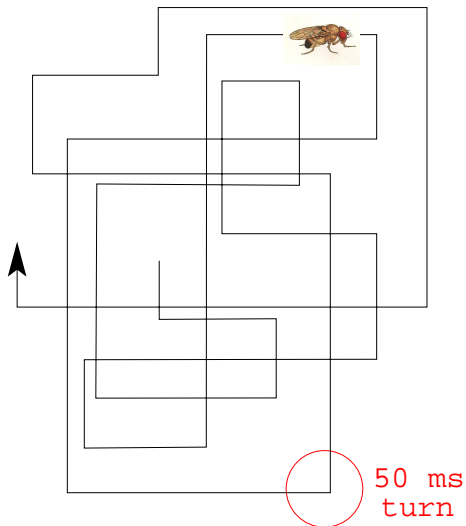
The Dynamic Primitive — Animals



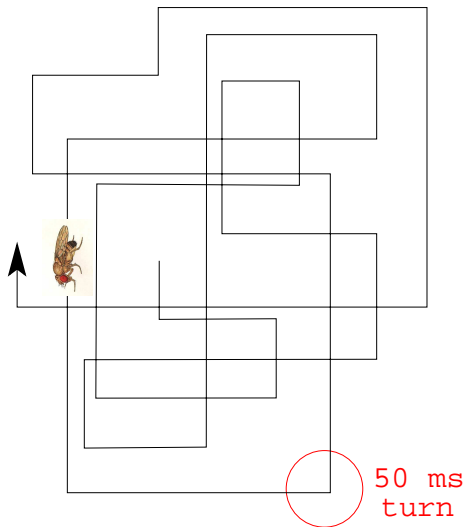
The Dynamic Primitive — Animals



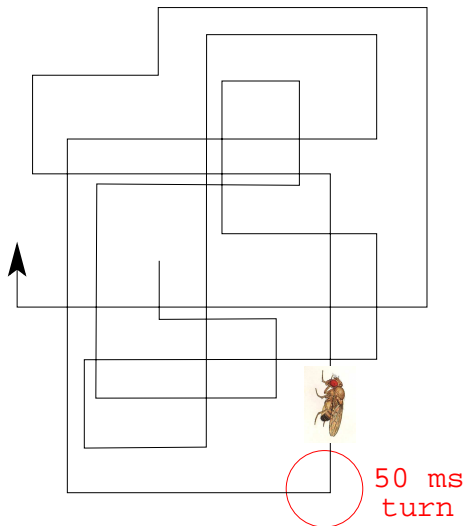
The Dynamic Primitive — Animals



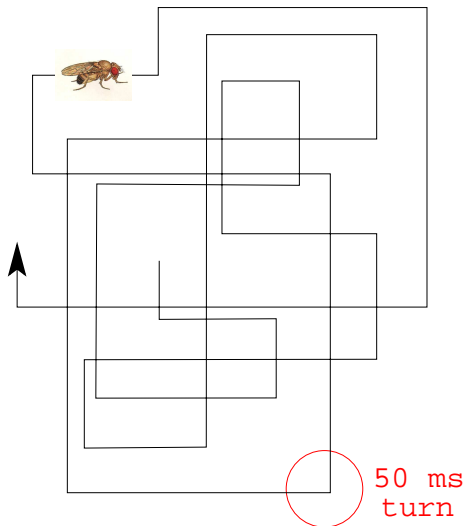
The Dynamic Primitive — Animals



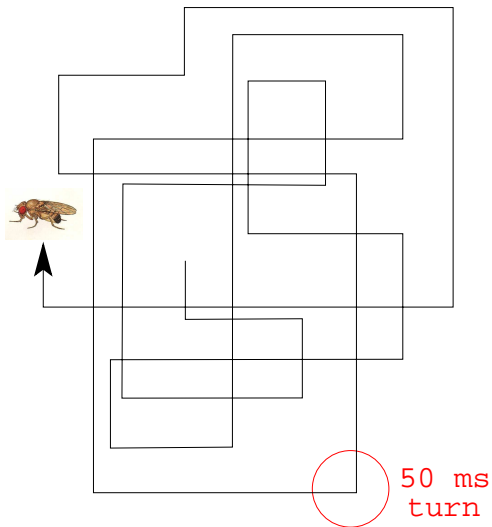
The Dynamic Primitive — Animals



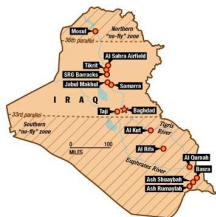
The Dynamic Primitive — Animals



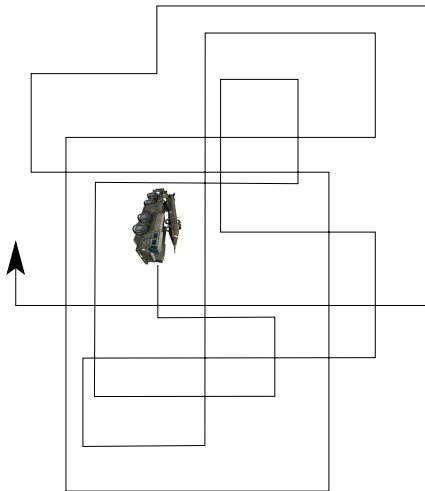
The Dynamic Primitive — Animals



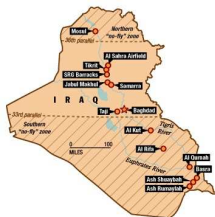
The Dynamic Primitive — History



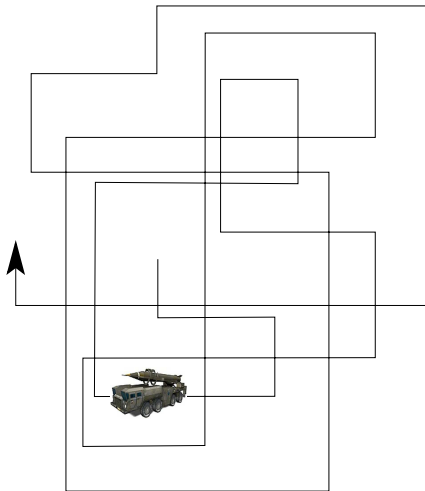
“even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable”



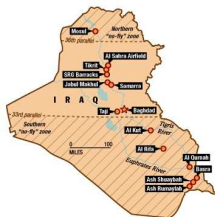
The Dynamic Primitive — History



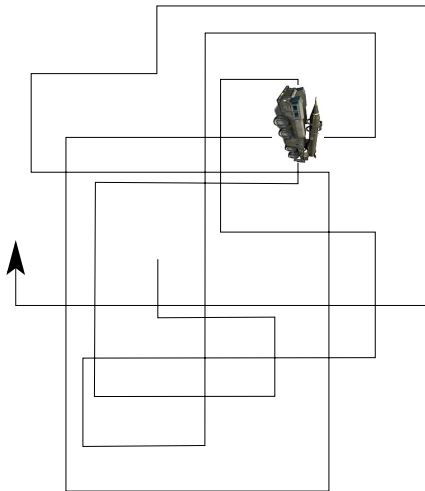
“even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable”



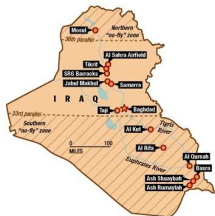
The Dynamic Primitive — History



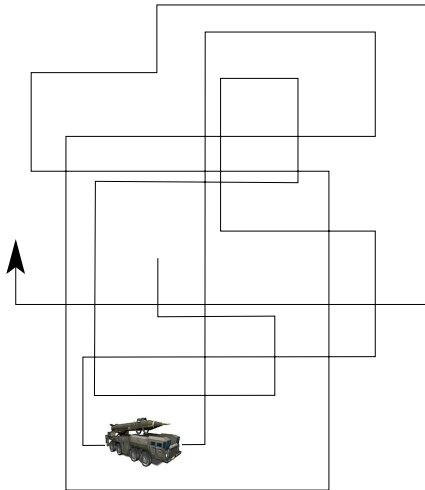
“even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable”



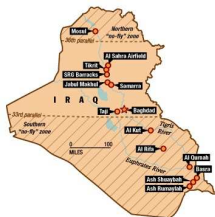
The Dynamic Primitive — History



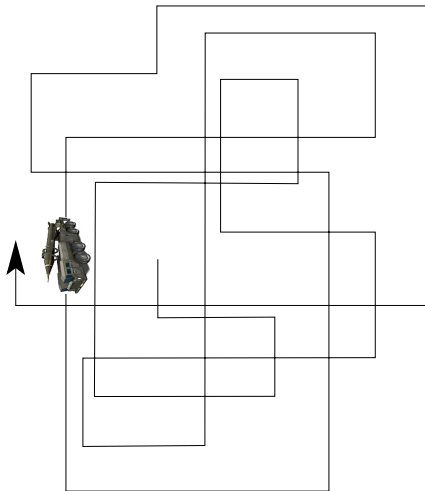
“even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable”



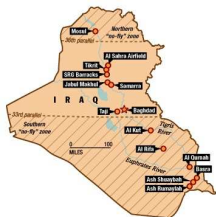
The Dynamic Primitive — History



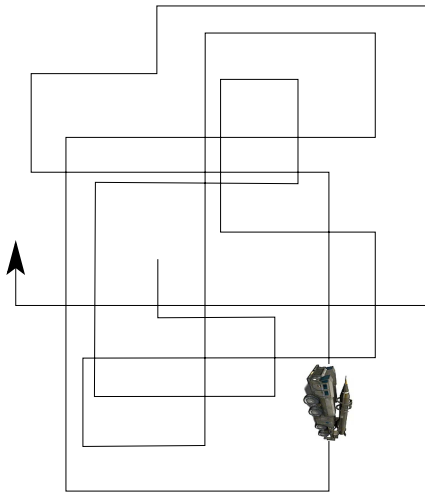
“even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable”



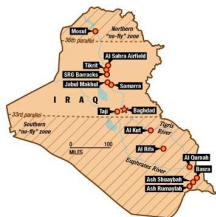
The Dynamic Primitive — History



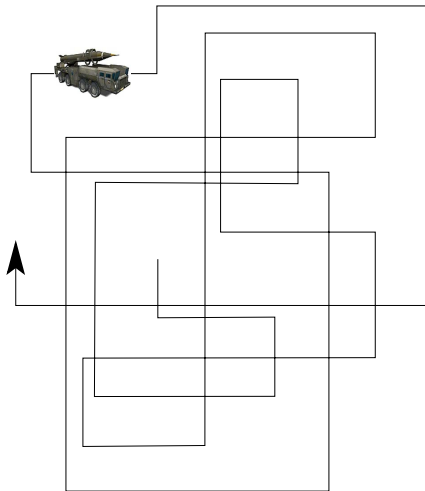
“even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable”



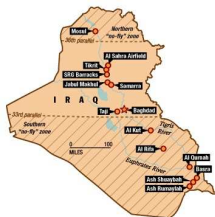
The Dynamic Primitive — History



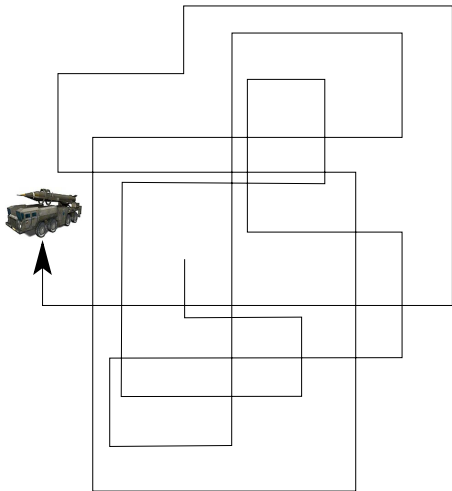
“even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable”



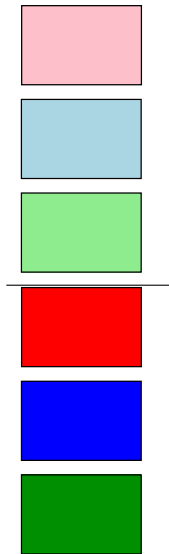
The Dynamic Primitive — History



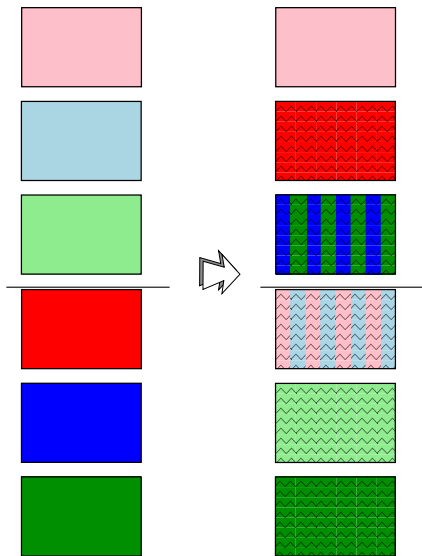
“even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable”



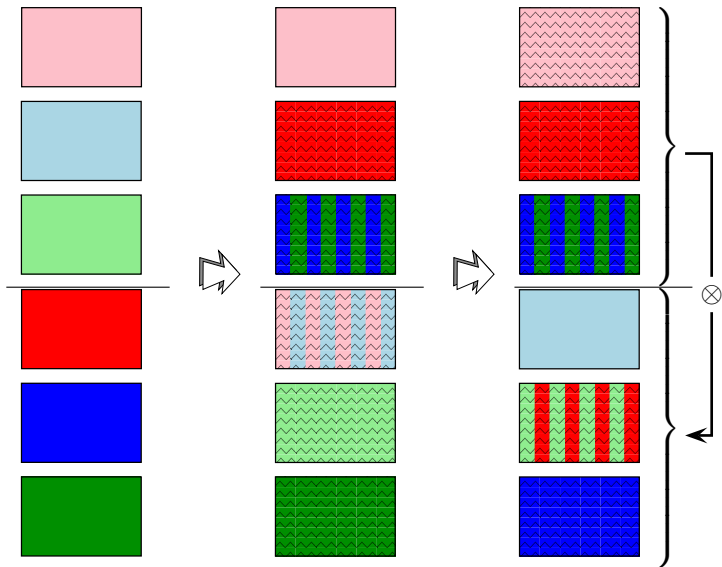
The Dynamic Primitive — Aucsmith



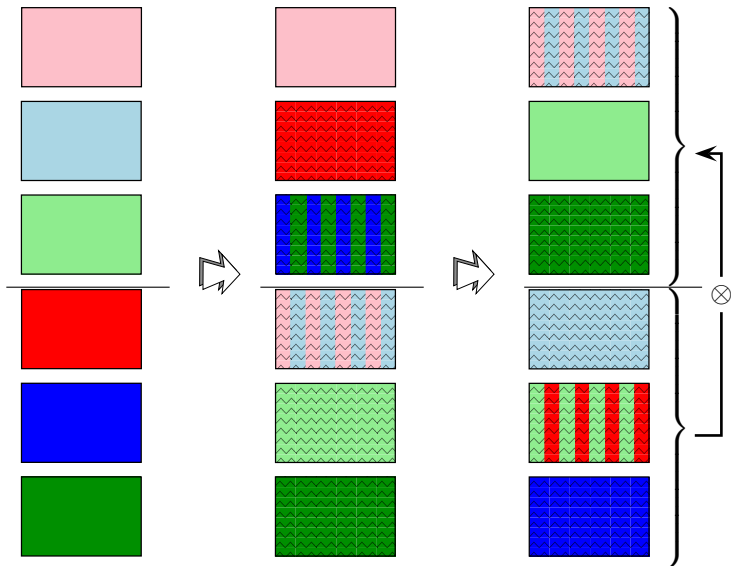
The Dynamic Primitive — Aucsmith



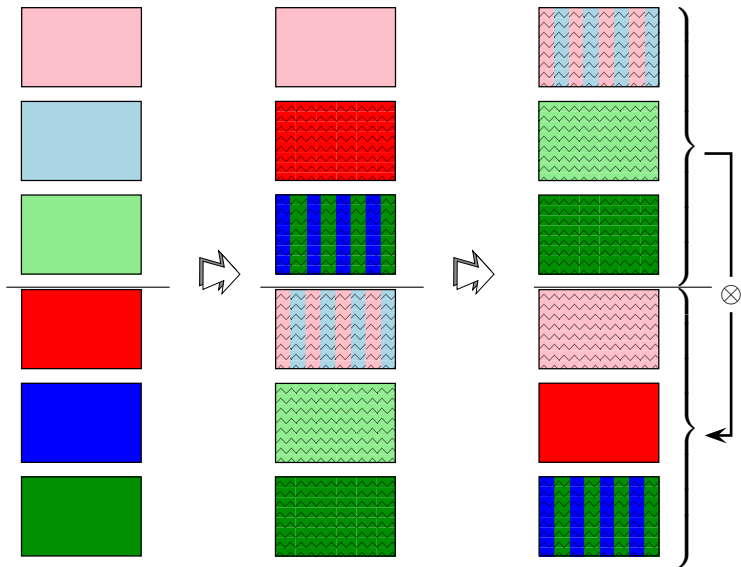
The Dynamic Primitive — Aucsmith



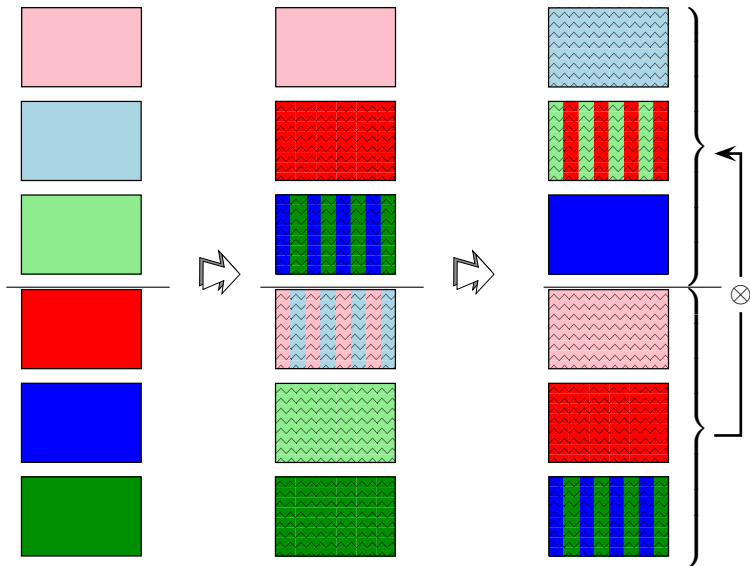
The Dynamic Primitive — Aucsmith



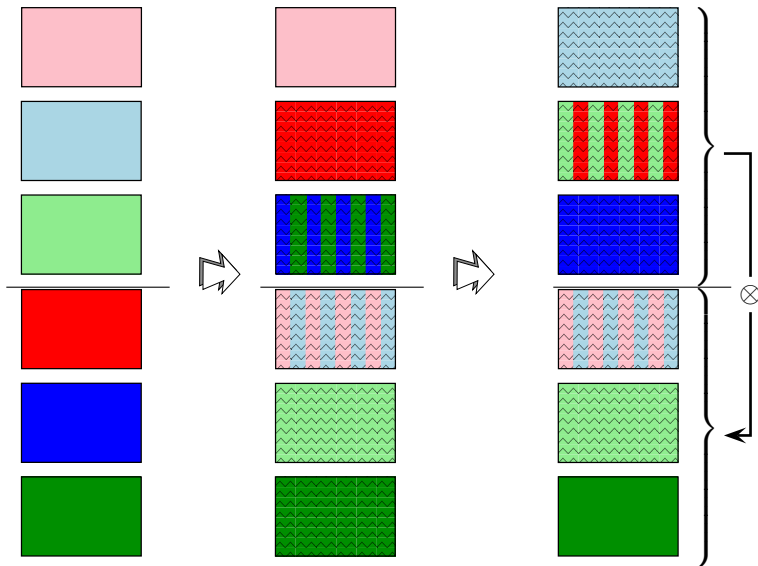
The Dynamic Primitive — Aucsmith



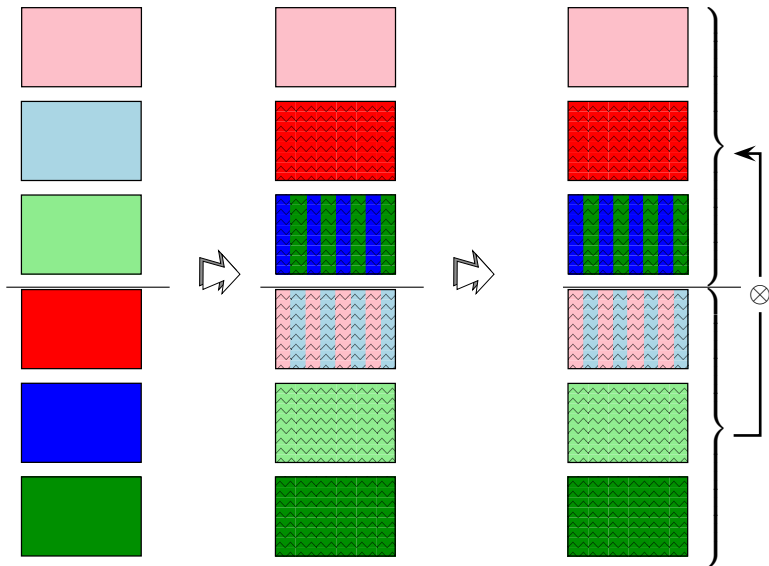
The Dynamic Primitive — Aucsmith



The Dynamic Primitive — Aucsmith



The Dynamic Primitive — Aucsmith



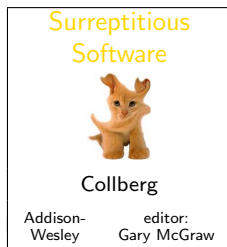
Discussion

- Are these observations **useful** at all?

Discussion

- Are these observations **useful** at all?

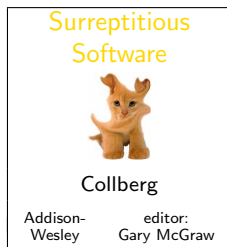
① Useful to **me!**



Discussion

- Are these observations **useful** at all?

① Useful to **me!**

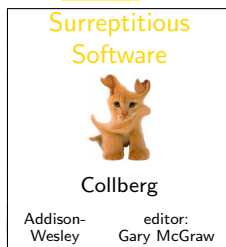


② Can we model **existing** /predict **future** algorithms?

Discussion

- Are these observations **useful** at all?

- 1 Useful to **me!**

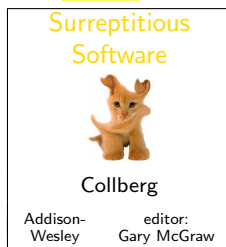


- 2 Can we model **existing** /predict **future** algorithms?
- Extend the model to incorporate **attacks**:

Discussion

- Are these observations **useful** at all?

- 1 Useful to **me!**

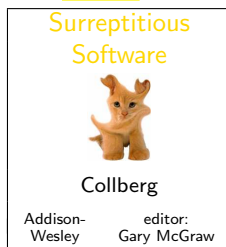


- 2 Can we model **existing** /predict **future** algorithms?
- Extend the model to incorporate **attacks**:
 - 1 Attacks peel off layers of compositions.

Discussion

- Are these observations **useful** at all?

- 1 Useful to **me!**

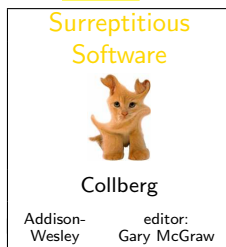


- 2 Can we model **existing** /predict **future** algorithms?
- Extend the model to incorporate **attacks**:
 - 1 Attacks peel off layers of compositions.
 - 2 Model the difference in cost between defenses and attacks.

Discussion

- Are these observations **useful** at all?

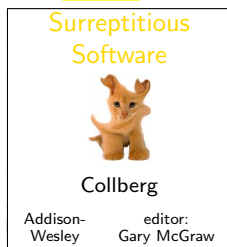
- 1 Useful to **me!**



- 2 Can we model **existing** /predict **future** algorithms?
- Extend the model to incorporate **attacks**:
 - 1 Attacks peel off layers of compositions.
 - 2 Model the difference in cost between defenses and attacks.
 - Develop a formal semantics!

- Are these observations **useful** at all?

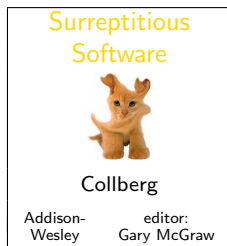
- ① Useful to **me!**



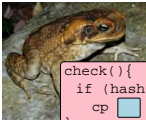
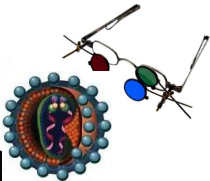
- ② Can we model **existing** /predict **future** algorithms?
- Extend the model to incorporate **attacks**:
 - ① Attacks peel off layers of compositions.
 - ② Model the difference in cost between defenses and attacks.
- Develop a formal semantics!
- Find unexpressible natural/computational scenarios!

- Are these observations **useful** at all?

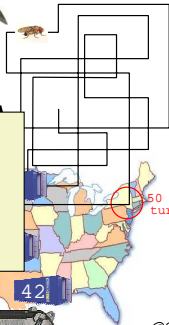
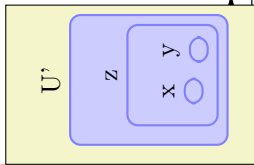
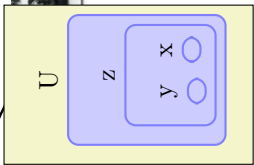
- 1 Useful to **me!**



- 2 Can we model **existing** /predict **future** algorithms?
- Extend the model to incorporate **attacks**:
 - 1 Attacks peel off layers of compositions.
 - 2 Model the difference in cost between defenses and attacks.
 - Develop a formal semantics!
 - Find unexpressible natural/computational scenarios!
 - Eliminate redundant primitives!



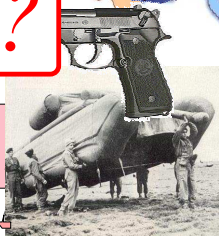
```
check(){
  if (hash(...)!=42)
    cp ■ ■
}
```



Questions?



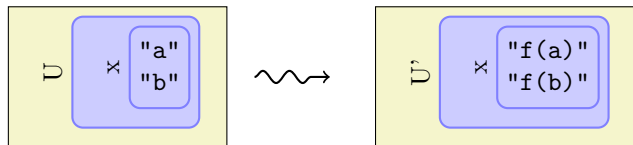
WATERMARK_IS_HERE =
"Copyright Bob"



The Washington Post



Primitive #7: Map



- Translate every component into something different.
- Creates confusion or encodes data.
- Keep the inverse of the mapping function secret.

The Map Primitive — History



I will now catch up with my correspondence and attend to my household chores and hobbies. My Rock Garden is beautiful just now. As to my doll collection I am trying to purchase a few foreign ones no longer in the shops.

I just secured a lovely Siamese Temple Dancer, it had been damaged, that is tore in the middle, but it is now repaired and I like it very much. I could not get a mate for this Siam dancer, so I am redressing just a small plain ordinary doll into a second Siam doll.

I cannot say that I like this ~~would~~ be future Siam doll now, yet I hope after a while I will perform a miracle and have a mate to my Siamese dancer. My future production will be rather smaller than the original repaired one yet I am copying the costume

The Map Primitive — History



I will now catch up with my correspondence and attend to my household chores and hobbies. My Rock Garden is beautiful just now. As to my doll collection I am trying to purchase a few foreign ones no longer in the shops.

I just secured a lovely Siamese Temple Dancer, it had been damaged, that is tore in the middle, but it is now repaired and I like it very much. I could not get a mate for this Siam dancer, so I am redressing just a small plain ordinary doll into a second Siam doll.

I cannot say that I like this ~~would~~ be future Siam doll now, yet I hope after a while I will perform a miracle and have a mate to my Siamese dancer. My future production will be rather smaller than the original repaired one yet I am copying the costume

Siamese Temple Dancer = aircraft carrier warship

The Map Primitive — History



I will now catch up with my correspondence and attend to my household chores and hobbies. My Rock Garden is beautiful just now. As to my doll collection I am trying to purchase a few foreign ones no longer in the shops.

I just secured a lovely Siamese Temple Dancer, it had been damaged, that is forein the middle, but it is now repaired and I like it very much. I could not get a mate for this Siam dancer, so I am redressing just a small plain ordinary doll into a second Siam doll. I cannot say that I like this ~~would~~ be future Siam doll now, yet I hope after a while I will perform a miracle and have a mate to my Siamese dancer. My future production will be rather smaller than the original repaired one yet I am copying the costume

Siamese Temple Dancer = aircraft carrier warship
tore in the middle = torpedoed in the middle


```
class DRM {  
    int secretKey = 0xff004587;  
    int decrypt (int data) {  
        ....  
    }  
}
```

```
class DRM {
  int secretKey = 0xff004587;
  int decrypt (int data) {
    ....
  }
}
```

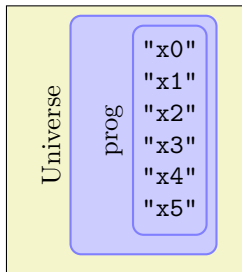
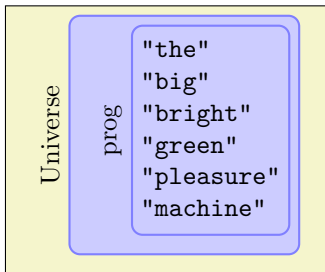


```
class C1 {
  int i1 = 0xff004587;
  int m1 (int x1) {
    ....
  }
}
```

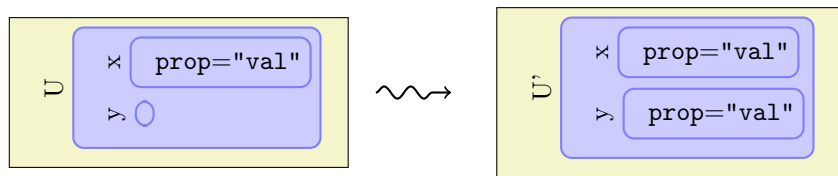
```
class DRM {
  int secretKey = 0xff004587;
  int decrypt (int data) {
    ....
  }
}
```



```
class C1 {
  int i1 = 0xff004587;
  int m1 (int x1) {
    ....
  }
}
```

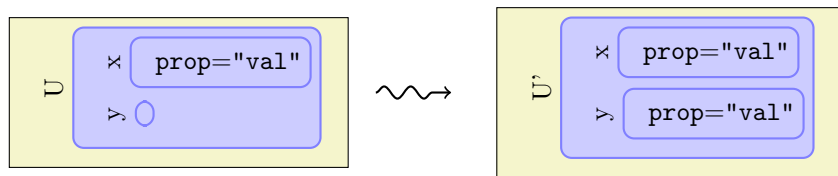


Primitive #8: Mimic



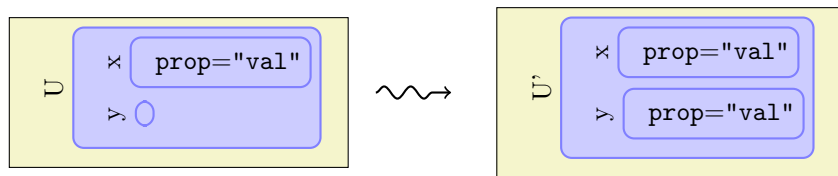
- Use as **camouflage** — blend in with the background!

Primitive #8: Mimic



- Use as **camouflage** — blend in with the background!
- Use as **deterrent** — look scary!

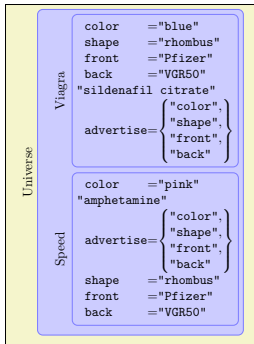
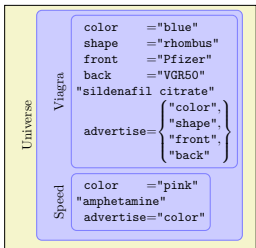
Primitive #8: Mimic

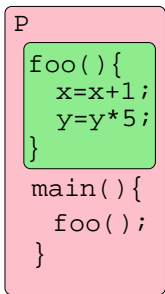


- Use as **camouflage** — blend in with the background!
- Use as **deterrent** — look scary!
- Copy a property from object A to B .

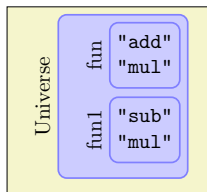
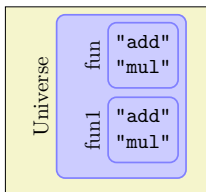
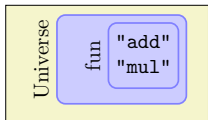
The Mimic Primitive — Biology

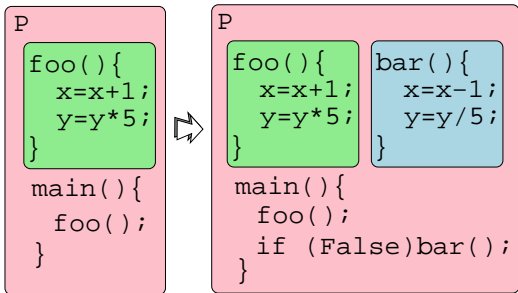




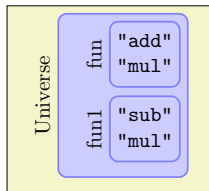
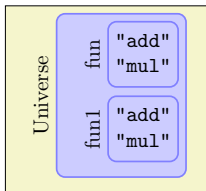
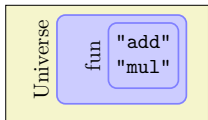


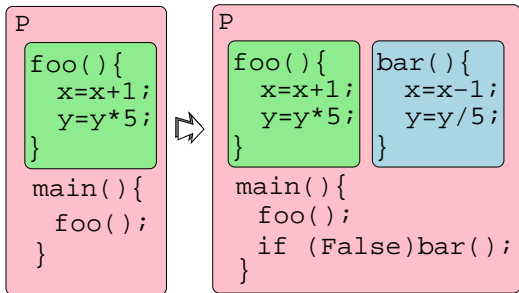
42





42

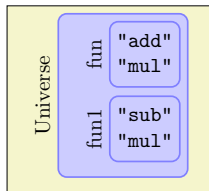
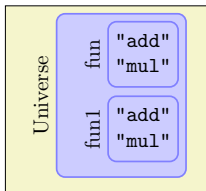
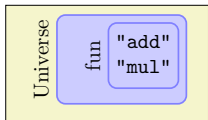




42



42



Primitive #10: Detect-respond

Tamperproofing has two parts:

- 1 detecting that an attack has occurred

The reaction can be a combination of

Primitive #10: Detect-respond

Tamperproofing has two parts:

- 1 detecting that an attack has occurred
- 2 reacting to this.

The reaction can be a combination of

Primitive #10: Detect-respond

Tamperproofing has two parts:

- ① detecting that an attack has occurred
- ② reacting to this.

The reaction can be a combination of

- ① self-destructing (in whole or in part),

Primitive #10: Detect-respond

Tamperproofing has two parts:

- 1 detecting that an attack has occurred
- 2 reacting to this.

The reaction can be a combination of

- 1 self-destructing (in whole or in part),
 - 2 destroying objects in the environment (including the attacker),
- or

Primitive #10: Detect-respond

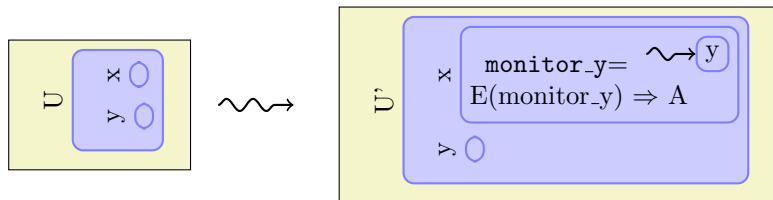
Tamperproofing has two parts:

- 1 detecting that an attack has occurred
- 2 reacting to this.

The reaction can be a combination of

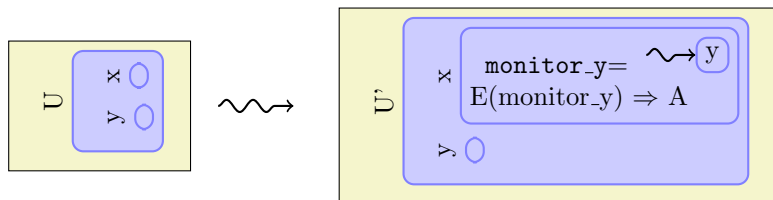
- 1 self-destructing (in whole or in part),
- 2 destroying objects in the environment (including the attacker),
or
- 3 regenerating the tampered parts.

The Detect-respond Primitive



- A monitors the health of x .

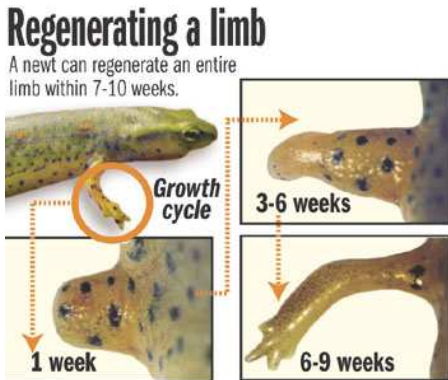
The Detect-respond Primitive



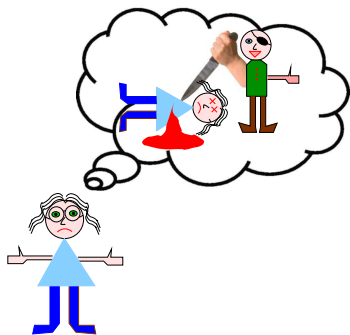
- A monitors the health of x .
- If T should happen, execute tamper-response E .

The Detect-respond Primitive — Biology

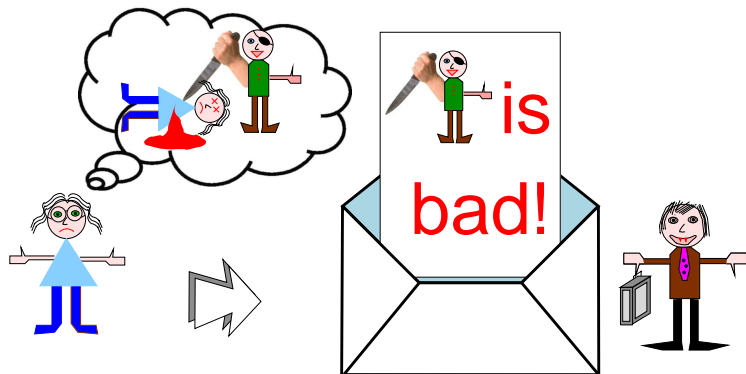
- Some animals can **regenerate** destroyed parts of their bodies after an attack:



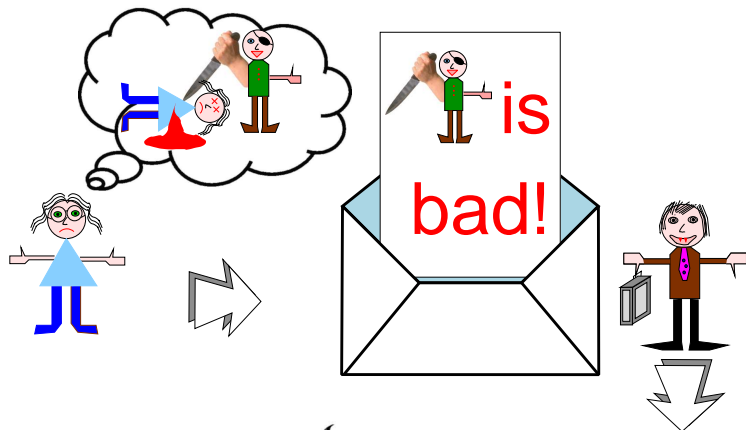
The Detect-respond Primitive — History



The Detect-respond Primitive — History



The Detect-respond Primitive — History



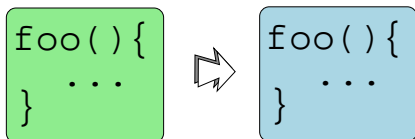
The Washington Post

SATURDAY, JUNE 28, 2003

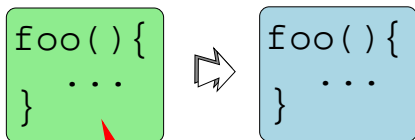
The Detect-respond Primitive — Software



```
foo() {  
    ...  
}
```


The Detect-respond Primitive — Software



The Detect-respond Primitive — Software



```
check() {  
  if (hash(foo) != 42)  
    cp    
}
```

The diagram shows a pink rounded rectangle containing the code for a `check()` function. The function has an `if` statement that checks `hash(foo) != 42`. If true, it executes `cp` followed by two small boxes: a blue one containing `foo() { ... }` and a green one containing `foo() { ... }`. A red arrow points from the `foo` parameter in the `if` statement to the green box, indicating that the function object is being copied.

The Detect-respond Primitive — Software

