Automating Formal Proofs for Reactive Systems

Daniel Ricketts, Valentin Robert,
Dongseok Jang, Sorin Lerner

University of California, San Diego

Zachary "Danger" Tatlock

University of Washington

Proof Assistant Based Verification

Proof Assistant Based Verification

Testing tool for C compilers [Yang et al. PLDI 11]

Proof Assistant Based Verification

Testing tool for C compilers [Yang et al. PLDI 11]

Compiler	*Bugs Found*

Proof Assistant Based Verification

Testing tool for C compilers [Yang et al. PLDI 11]

Compiler	*Bugs Found*
GCC	122

Proof Assistant Based Verification

Testing tool for C compilers [Yang et al. PLDI 11]

Compiler	*Bugs Found*
GCC	122
Clang/LLVM	181

Proof Assistant Based Verification

Testing tool for C compilers [Yang et al. PLDI 11]

Compiler	*Bugs Found*
GCC	122
Clang/LLVM	181
CompCert	?

Proof Assistant Based Verification

Testing tool for C compilers [Yang et al. PLDI 11]

Compiler	*Bugs Found*
GCC	122
Clang/LLVM	181
CompCert	0

Proof Assistant Based Verification

Testing tool for C compilers

[Yang et al. PLDI 11]
[Le et al. PLDI 14]

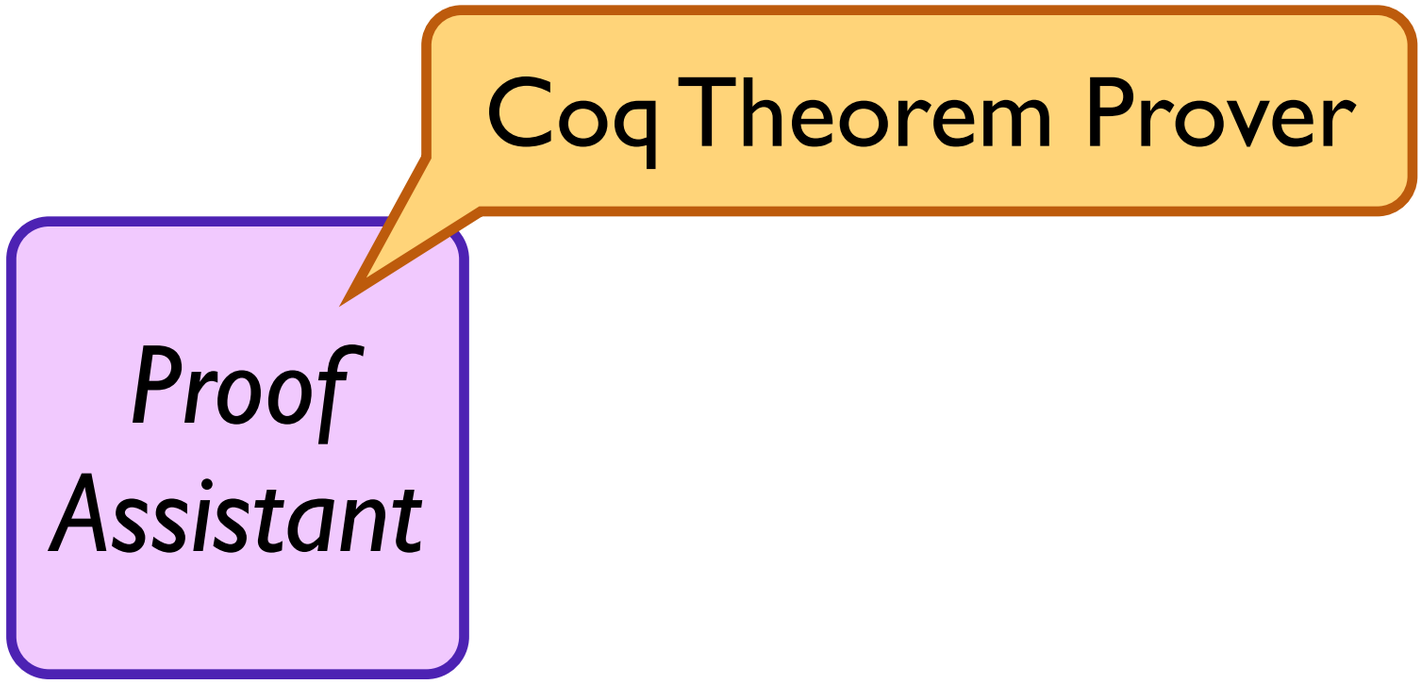
Compiler	*Bugs Found*
GCC	122
Clang/LLVM	181
CompCert	0

Proof Assistant Based Verification

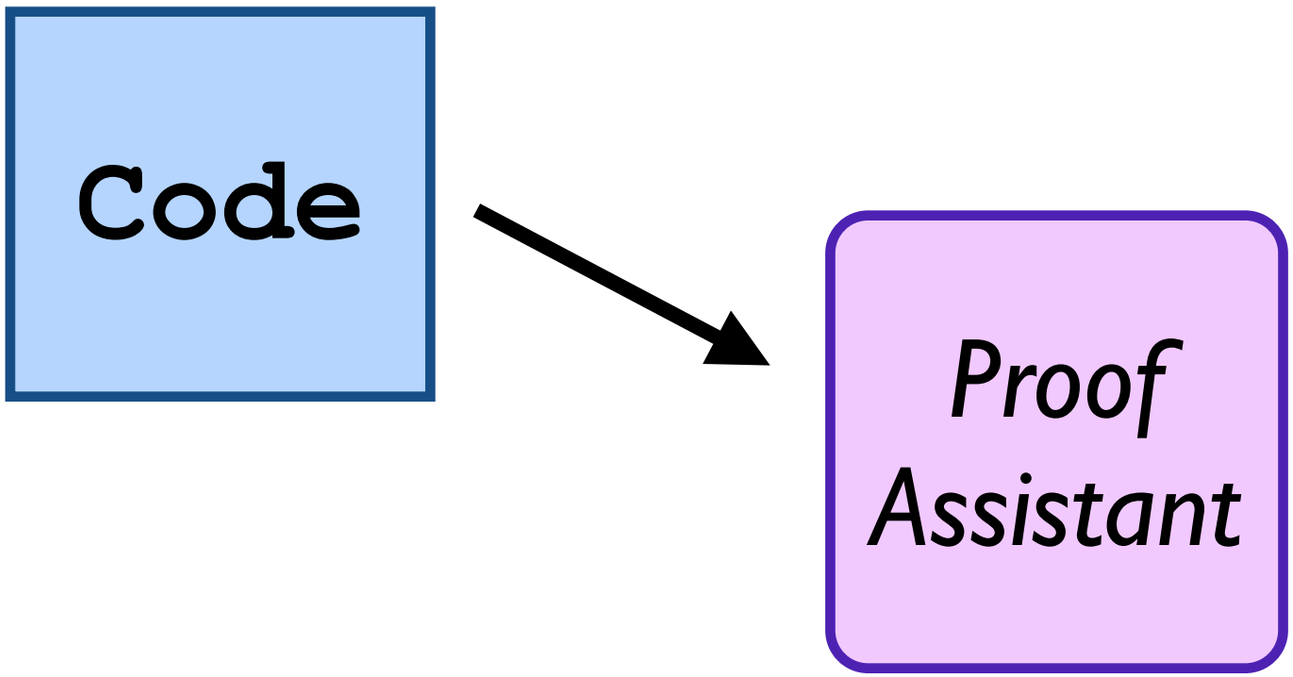
Proof Assistant Based Verification

Proof Assistant

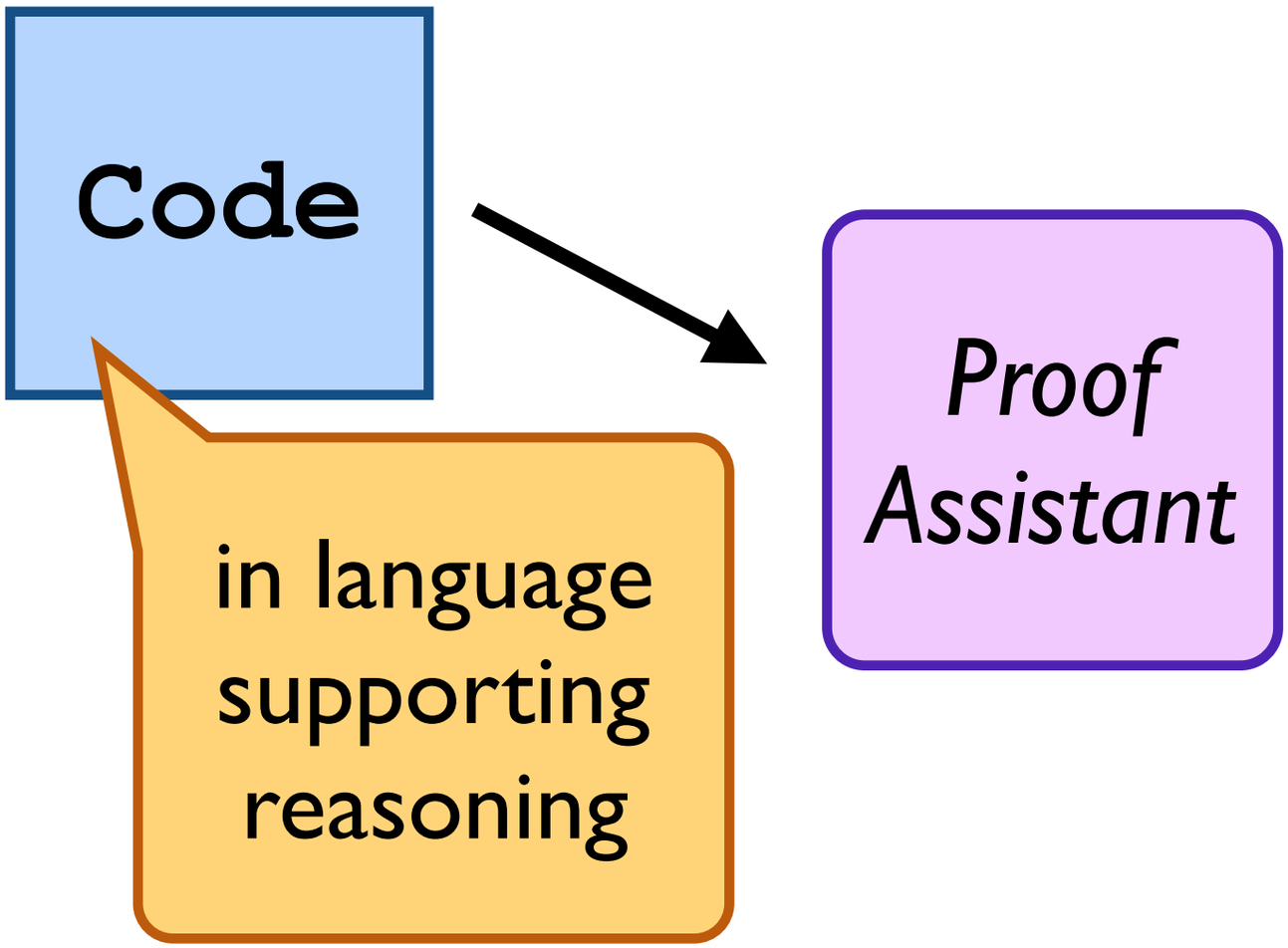
Proof Assistant Based Verification



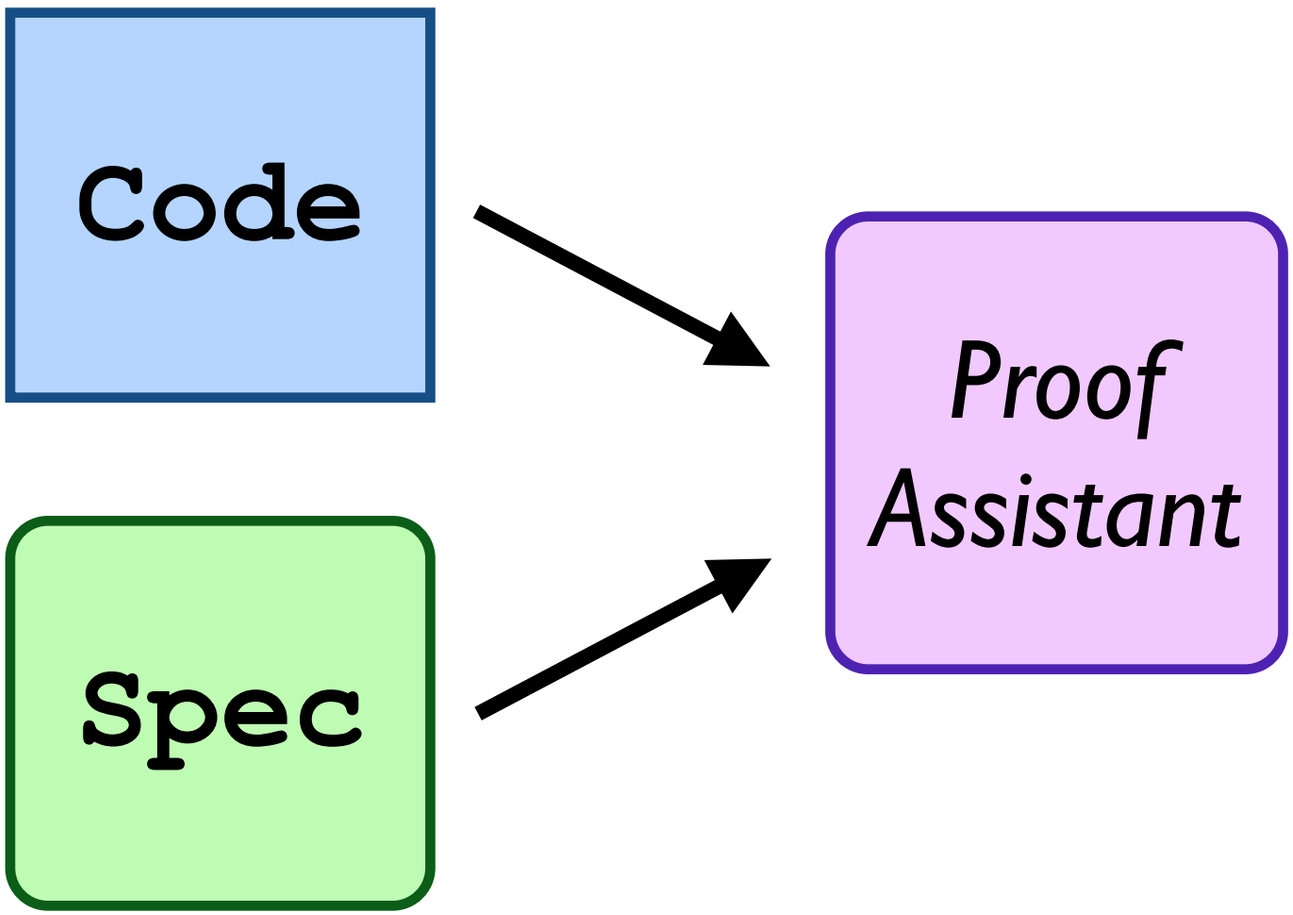
Proof Assistant Based Verification



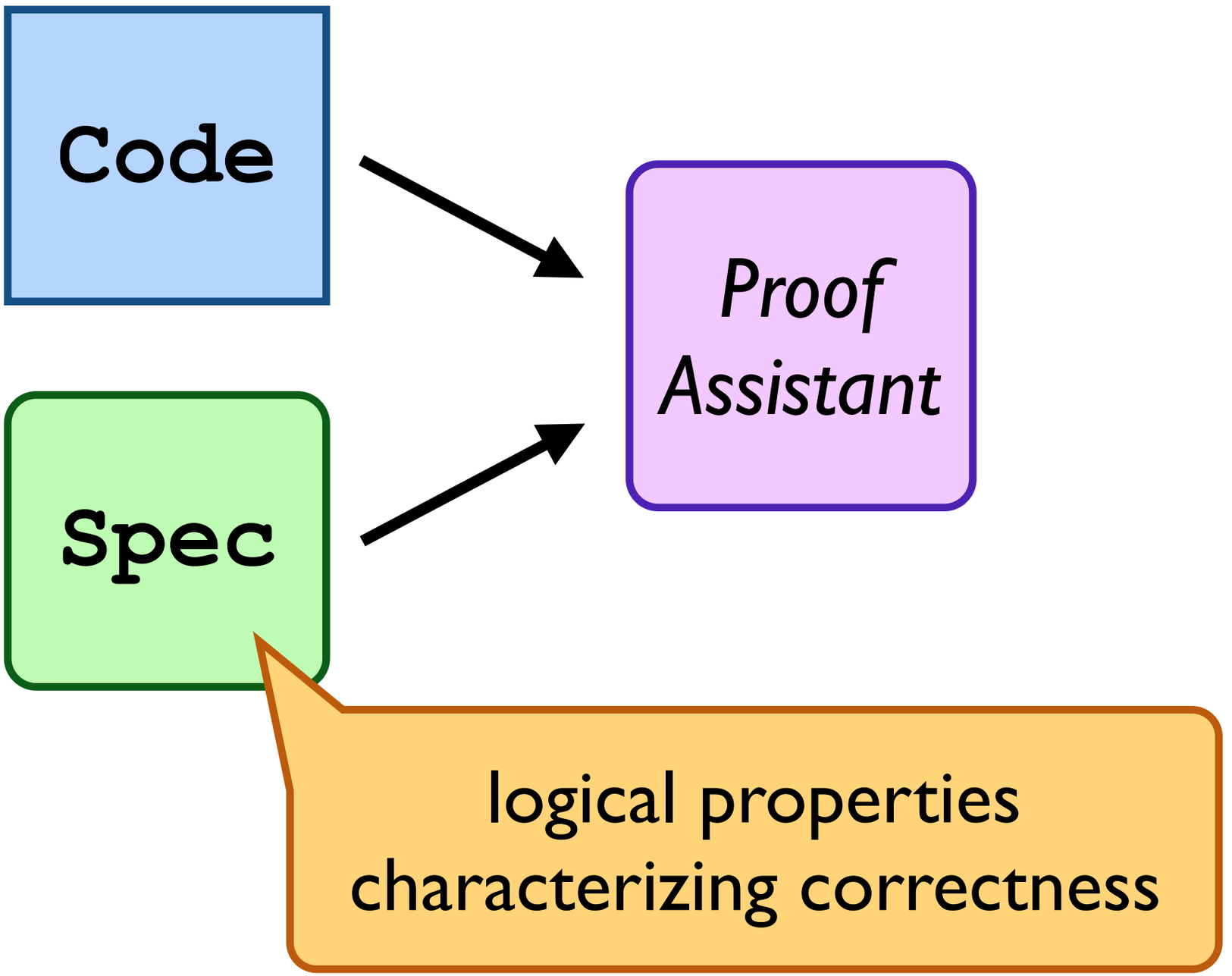
Proof Assistant Based Verification



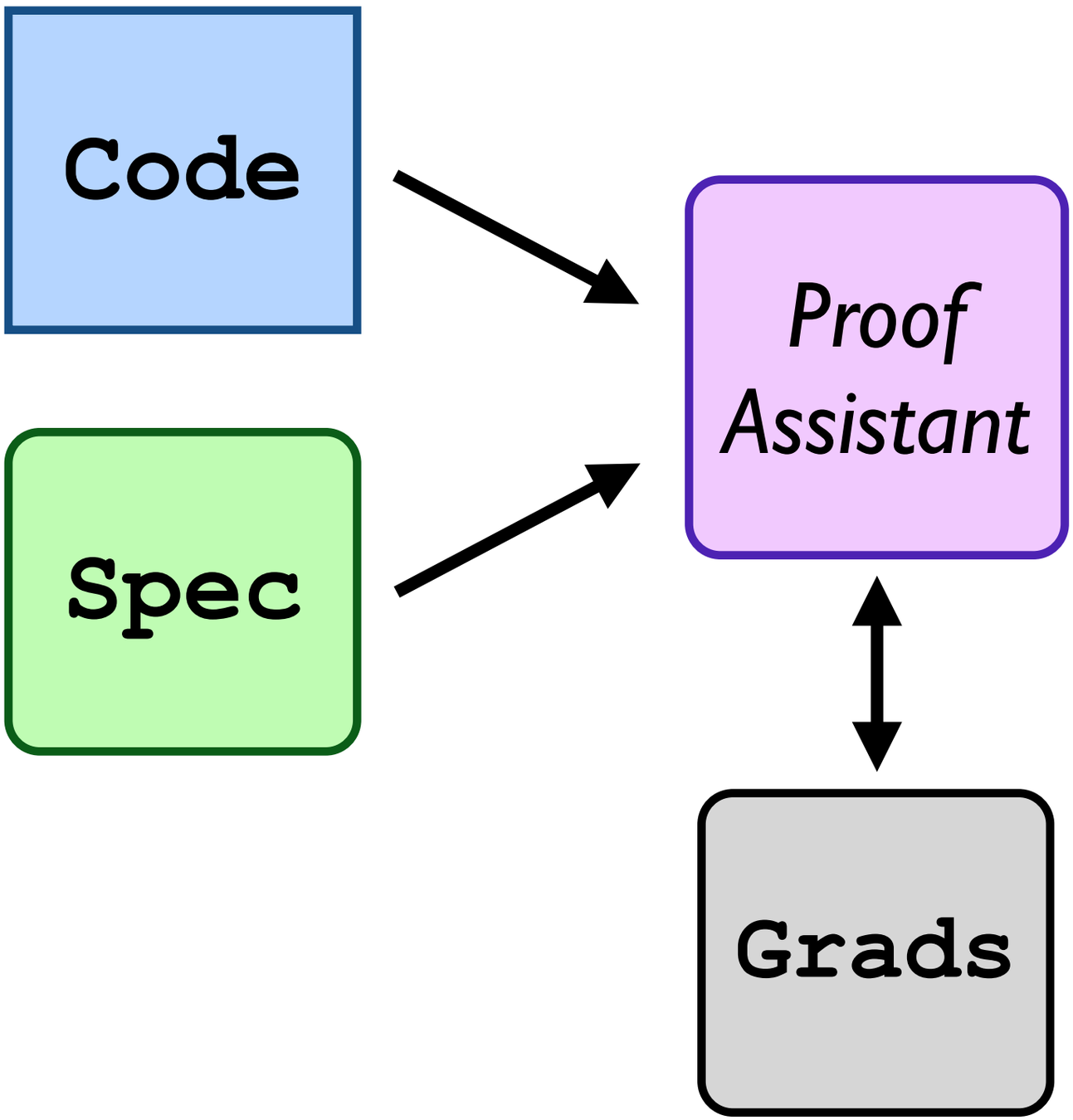
Proof Assistant Based Verification



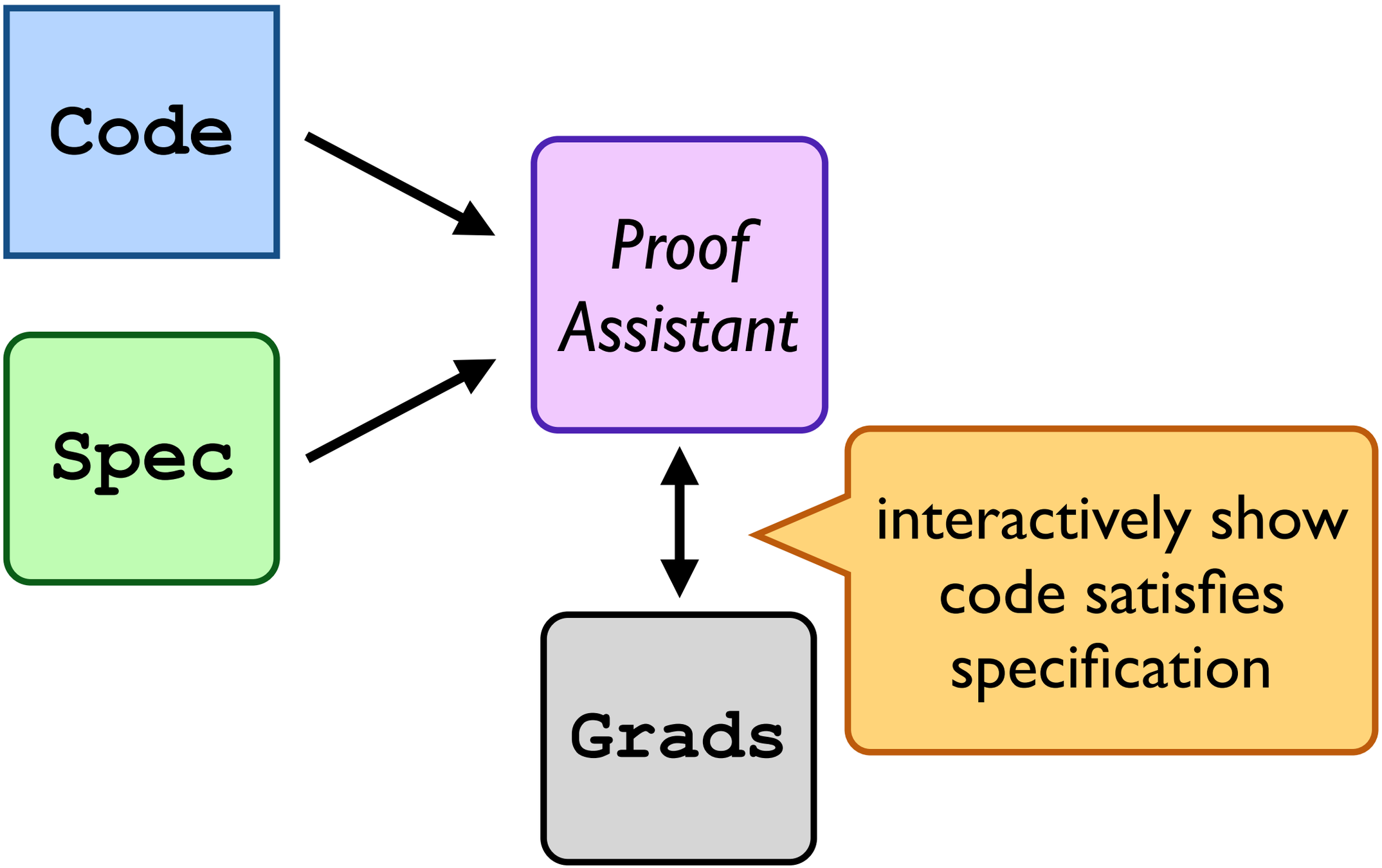
Proof Assistant Based Verification



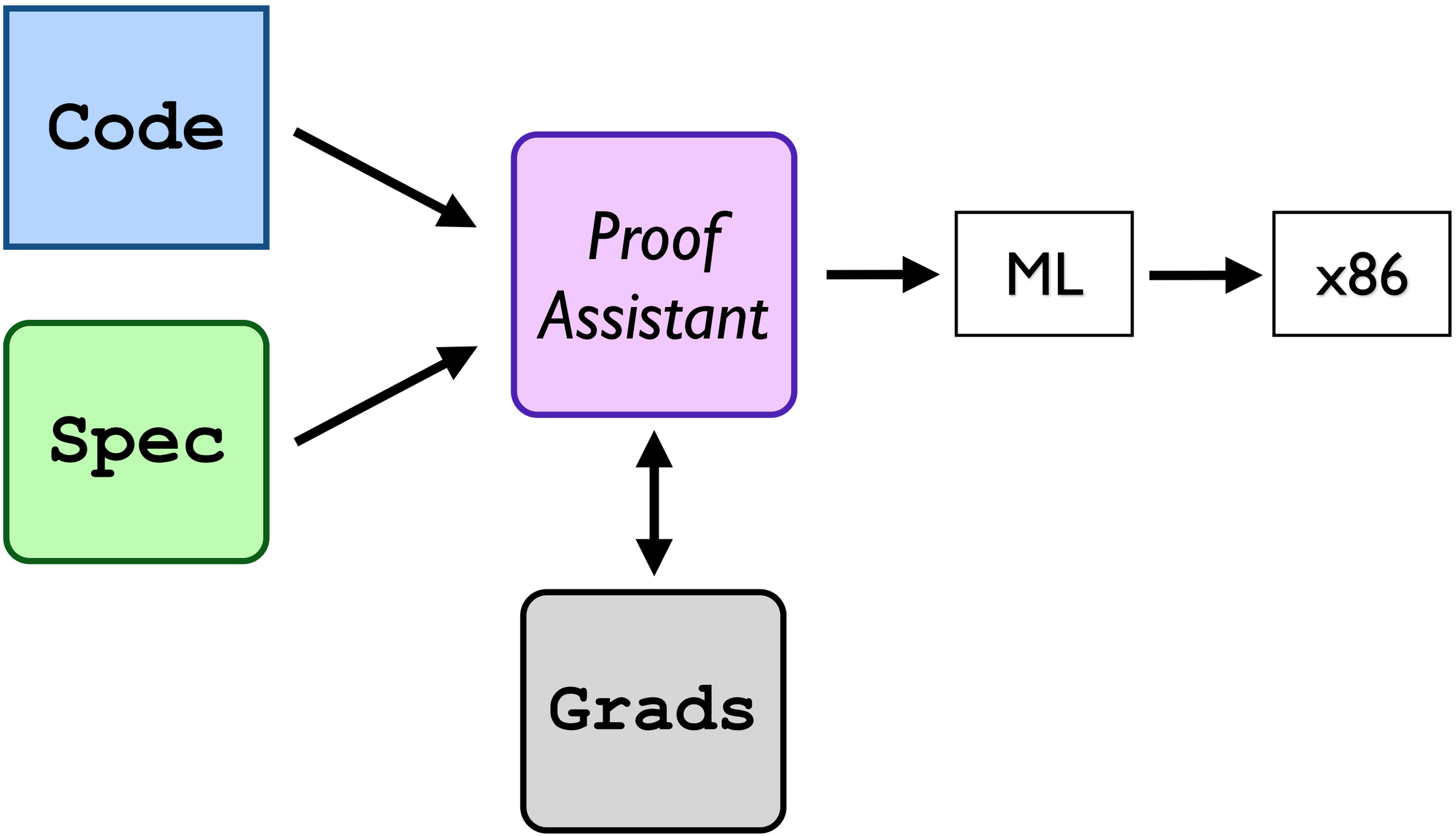
Proof Assistant Based Verification



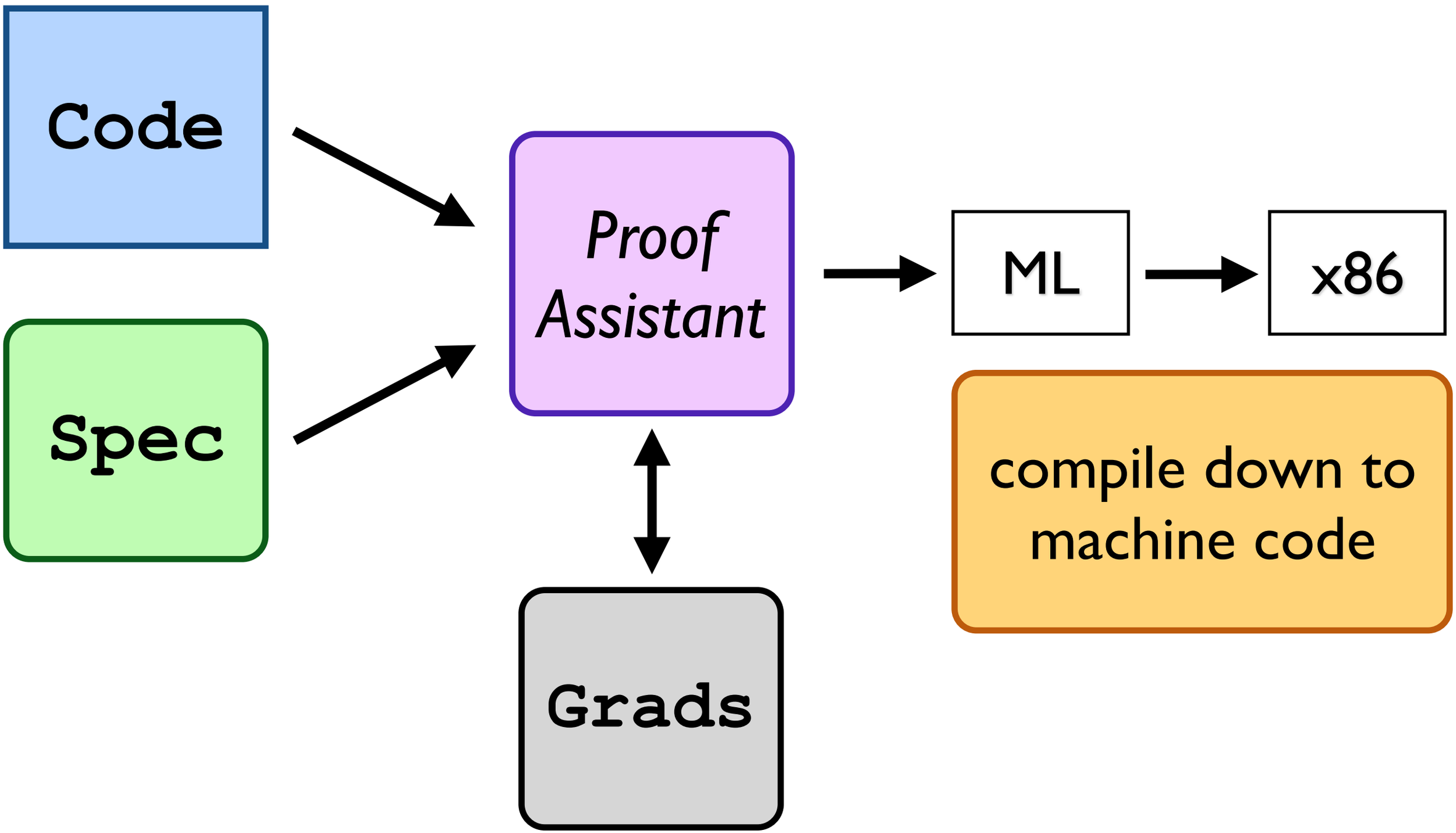
Proof Assistant Based Verification



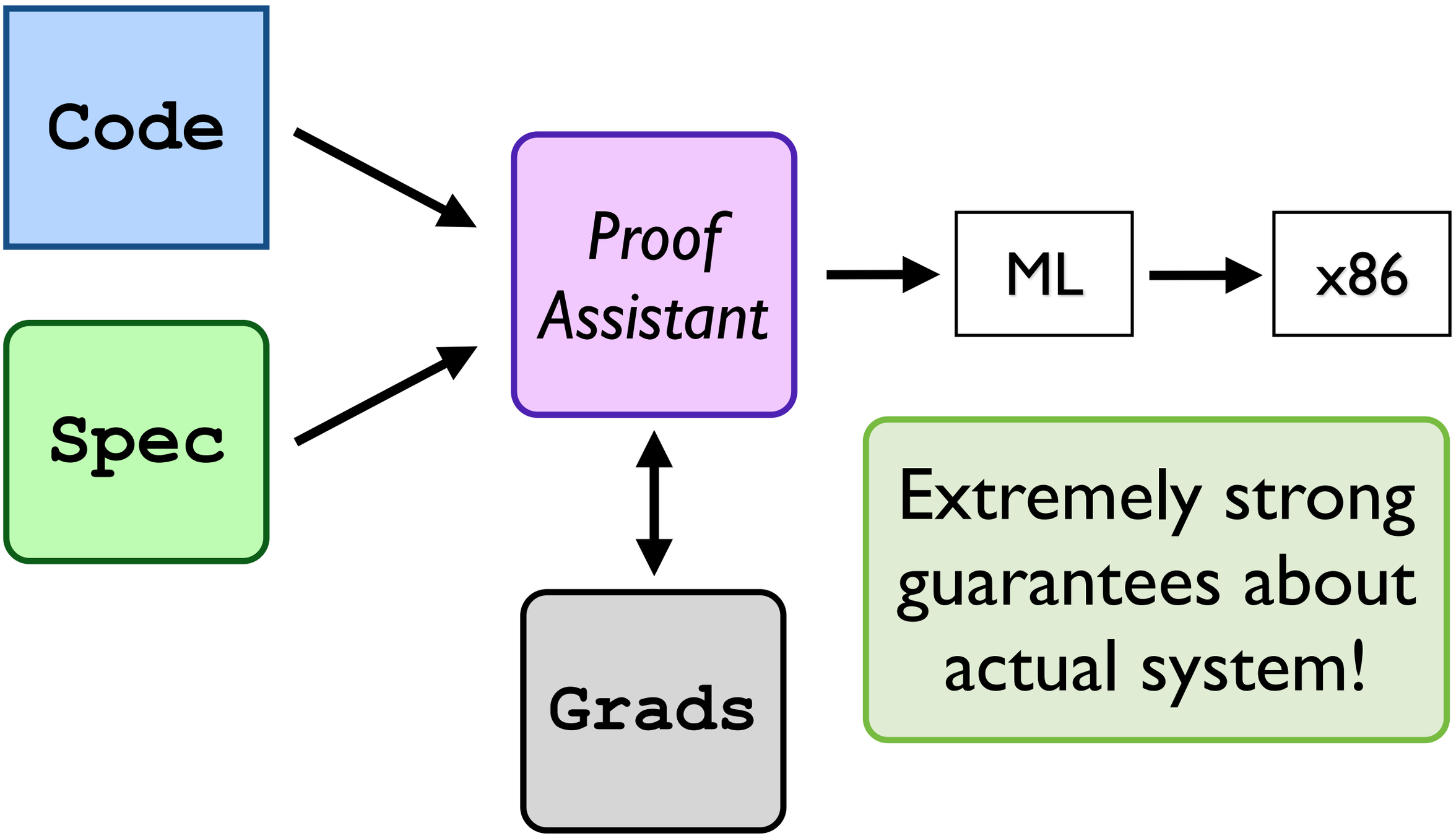
Proof Assistant Based Verification



Proof Assistant Based Verification



Proof Assistant Based Verification



Proof Assistant Based Verification

Verified Compiler: CompCert

[Leroy POPL 06]

Proof Assistant Based Verification

Verified Compiler: CompCert

[Leroy POPL 06]

Verified OS micro-kernel: seL4

[Klein et al. SOSP 09]

Proof Assistant Based Verification

Verified Compiler: CompCert

[Leroy POPL 06]

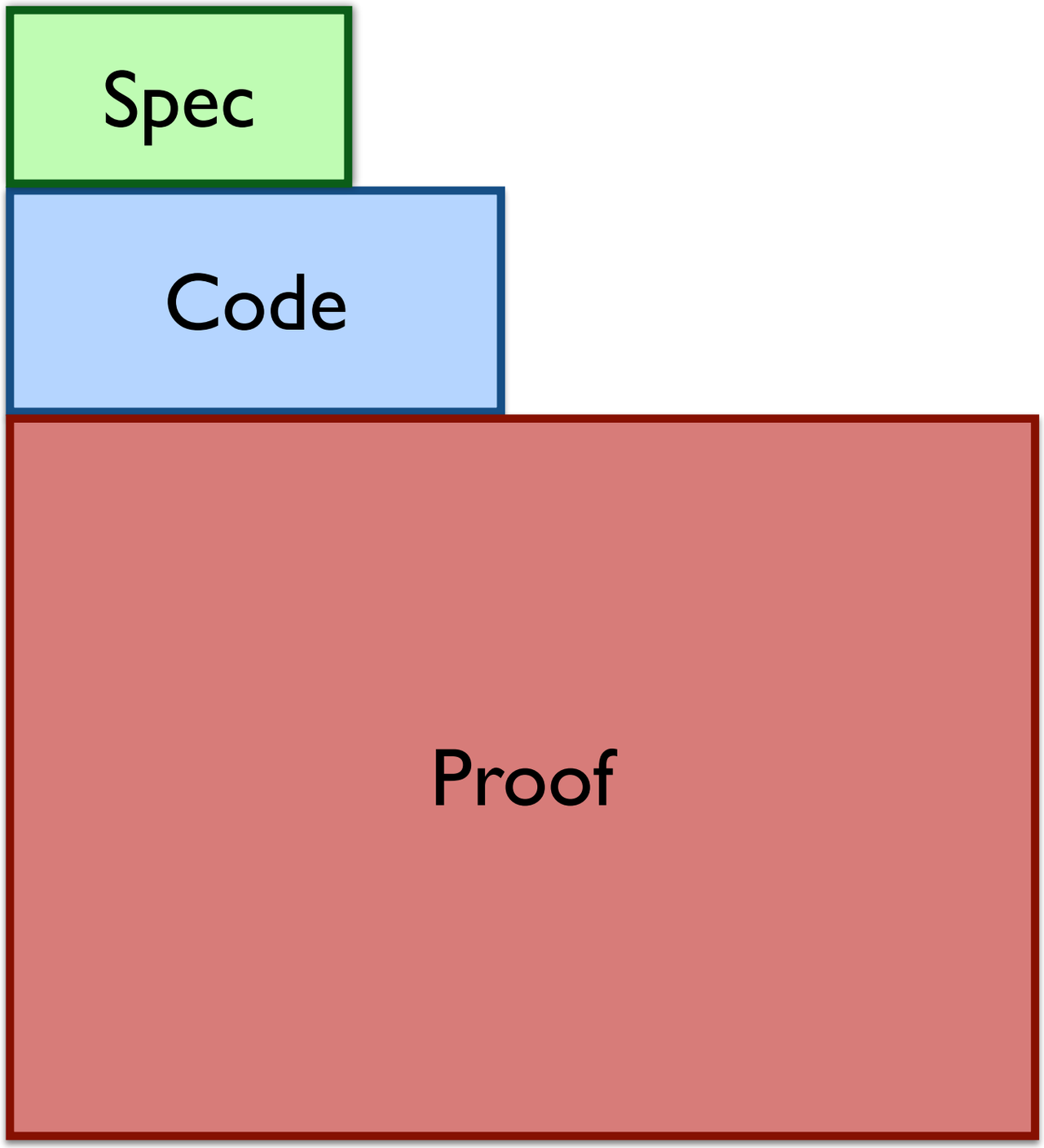
Verified OS micro-kernel: seL4

[Klein et al. SOSP 09]

Verified Web browser: Quark

[Jang et al. Security 12]

Manual Proof Burden



Manual Proof Burden

Verified OS micro-kernel: seL4

Manual Proof Burden

Verified OS micro-kernel: seL4

9,000 lines of C code

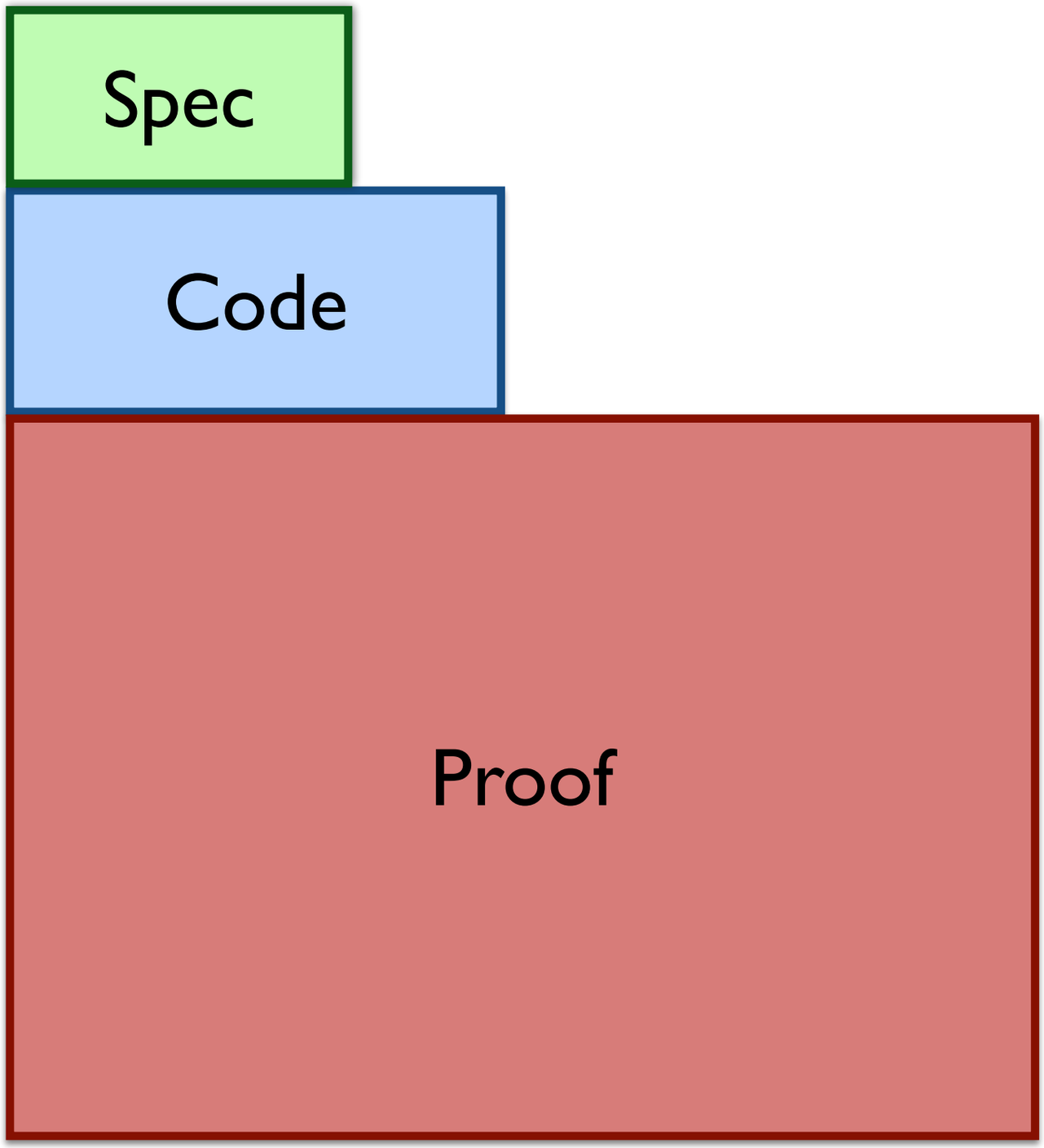
Manual Proof Burden

Verified OS micro-kernel: seL4

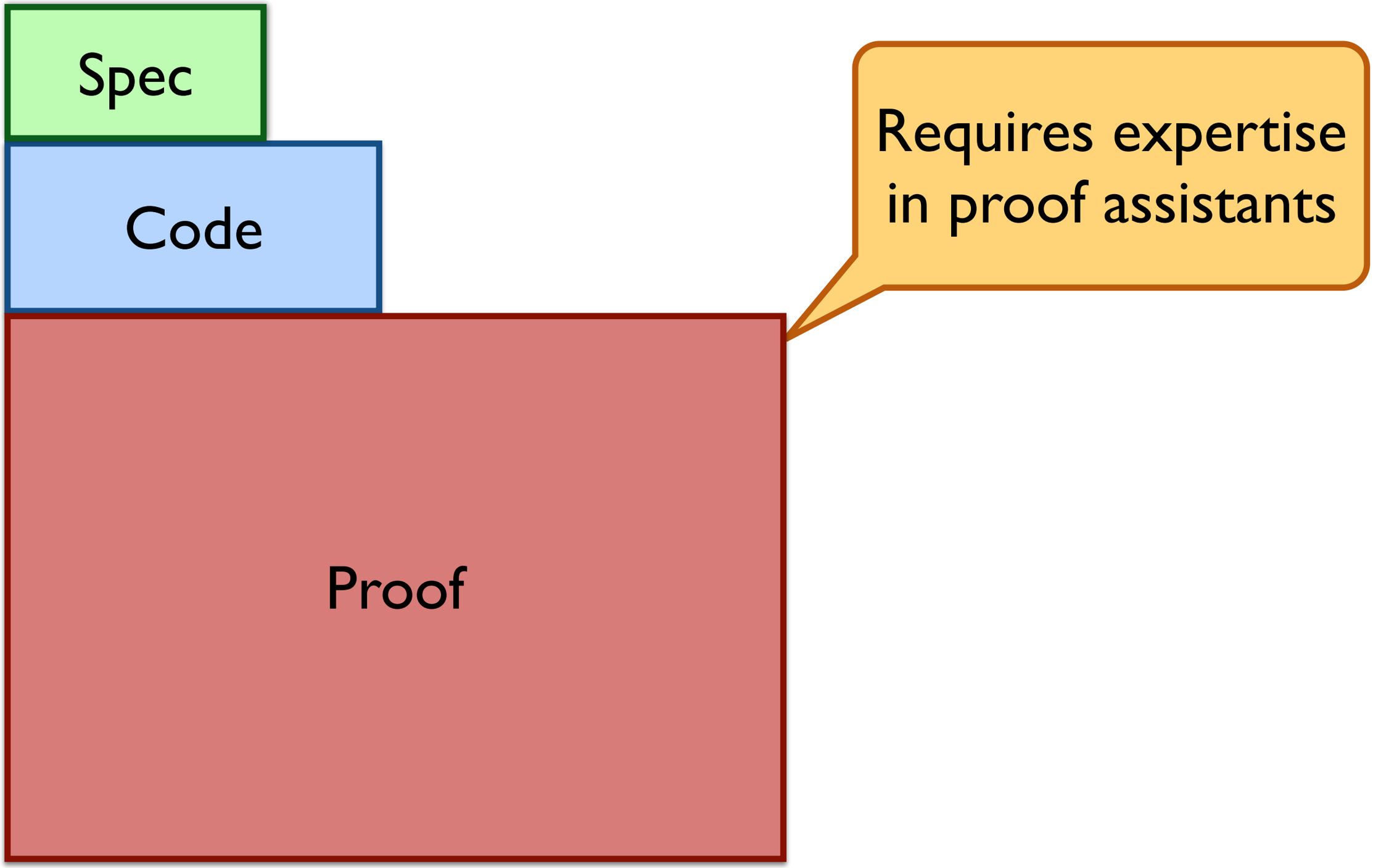
9,000 lines of C code

20 person-years for verification

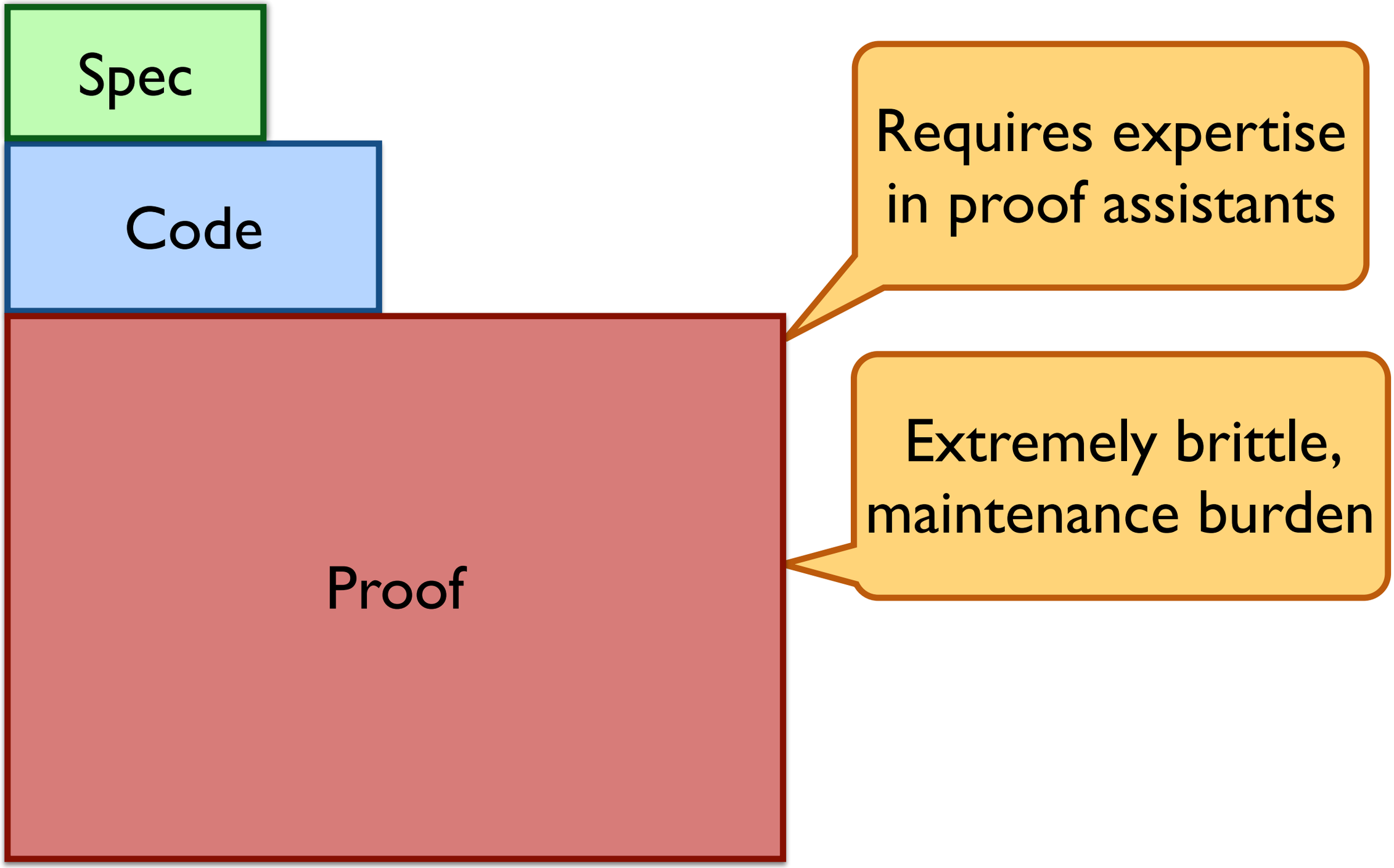
Manual Proof Burden



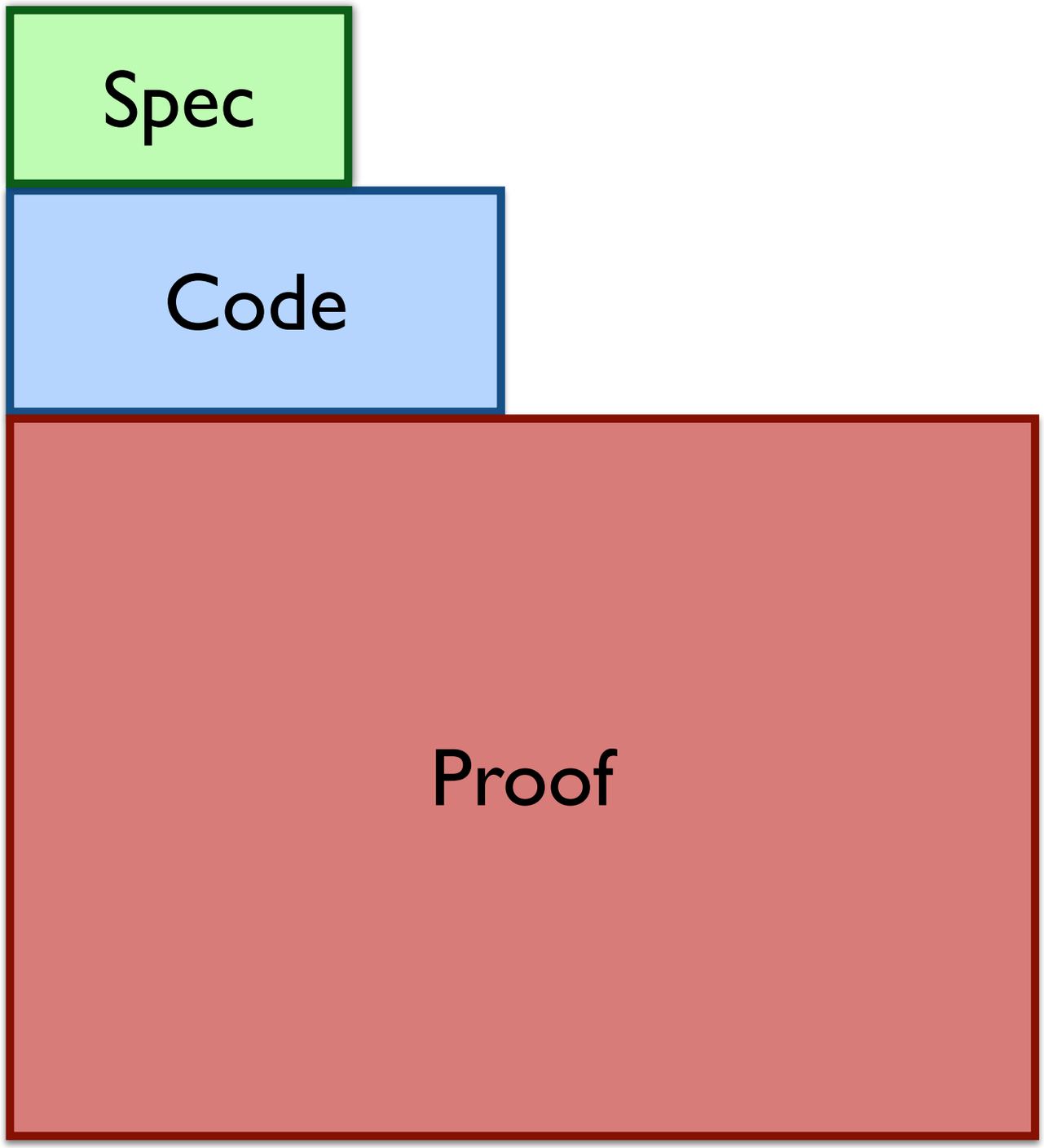
Manual Proof Burden



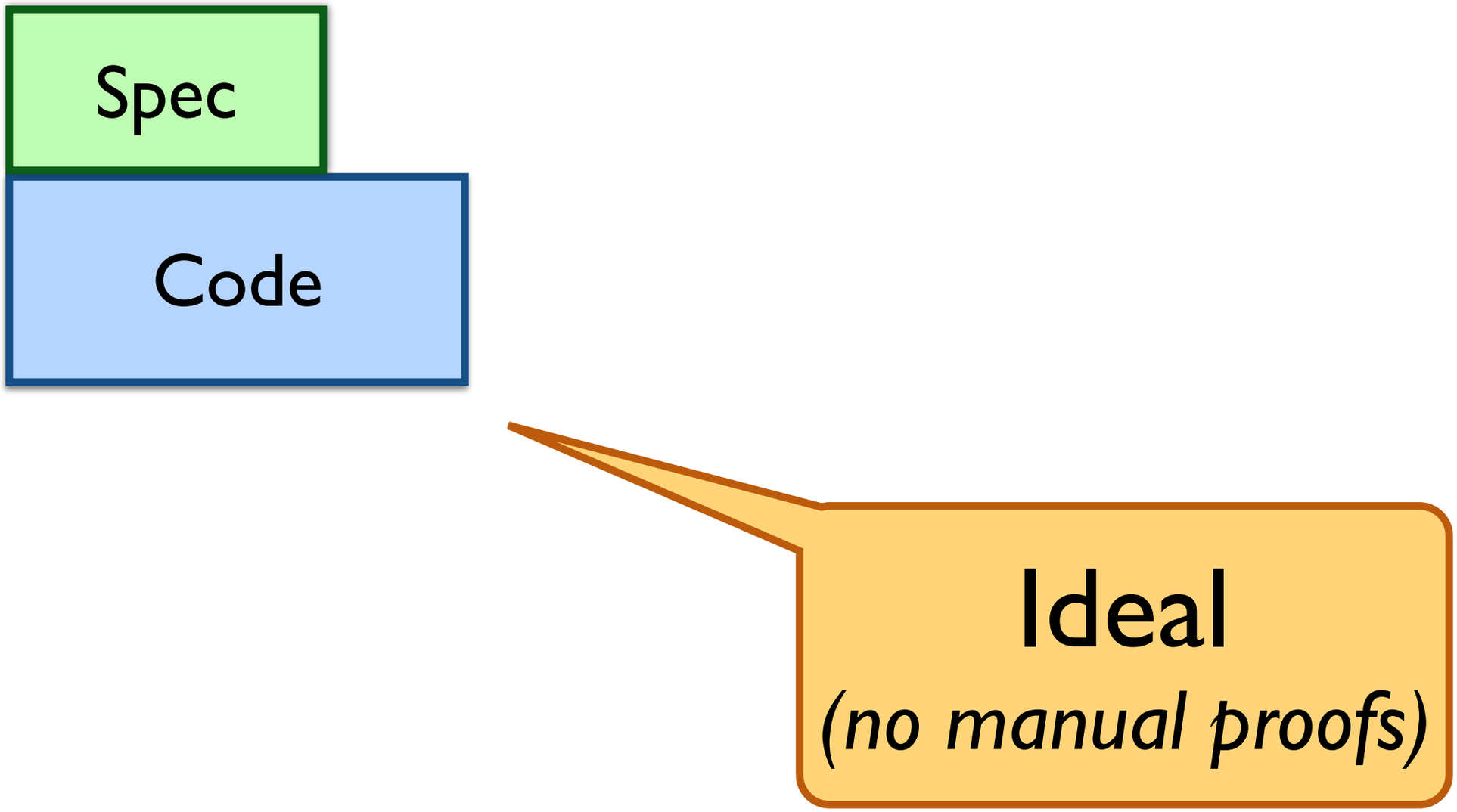
Manual Proof Burden



Manual Proof Burden

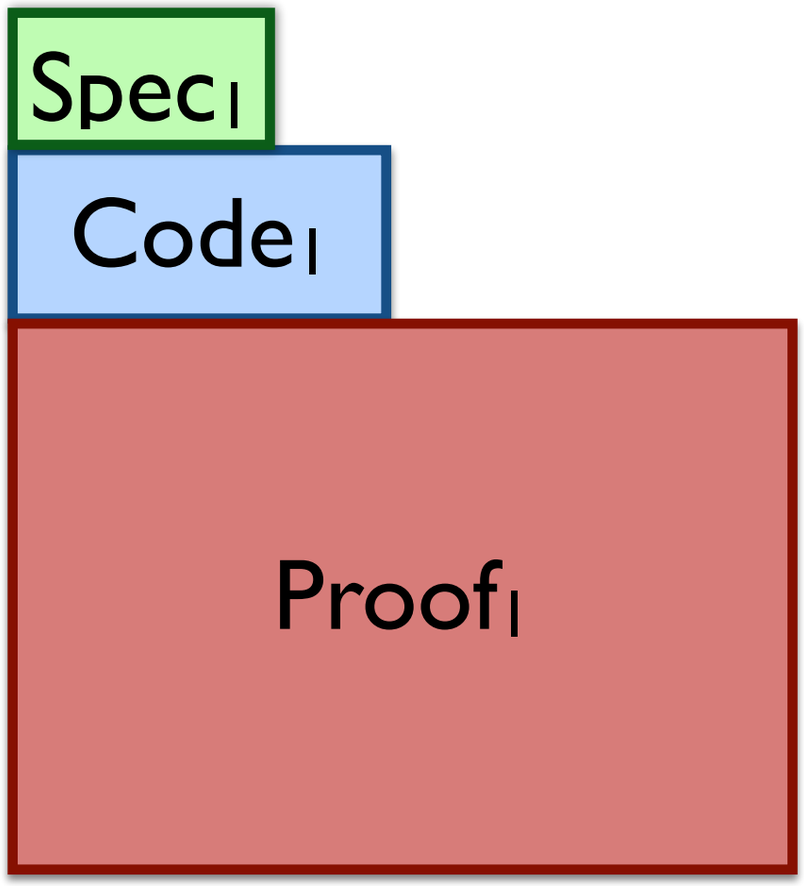


Manual Proof Burden

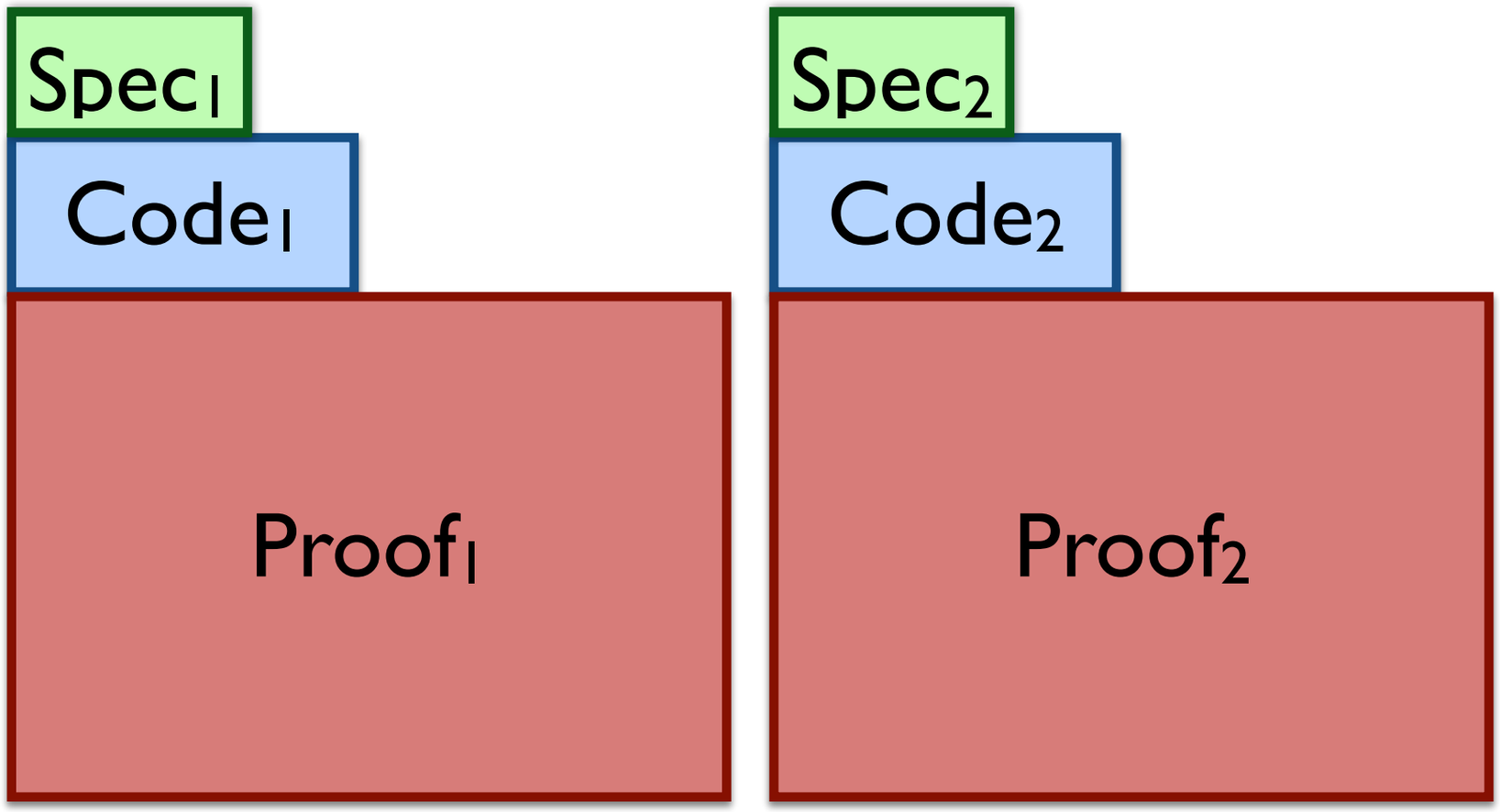


Single application domain

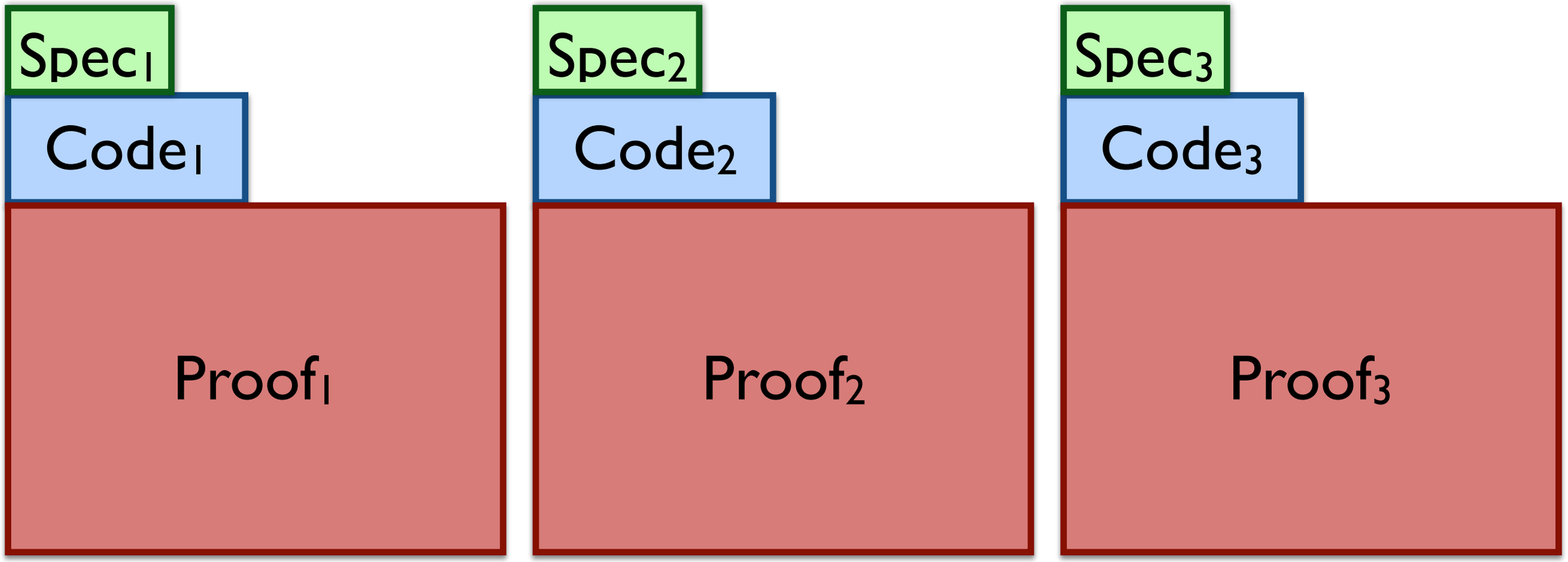
Single application domain



Single application domain

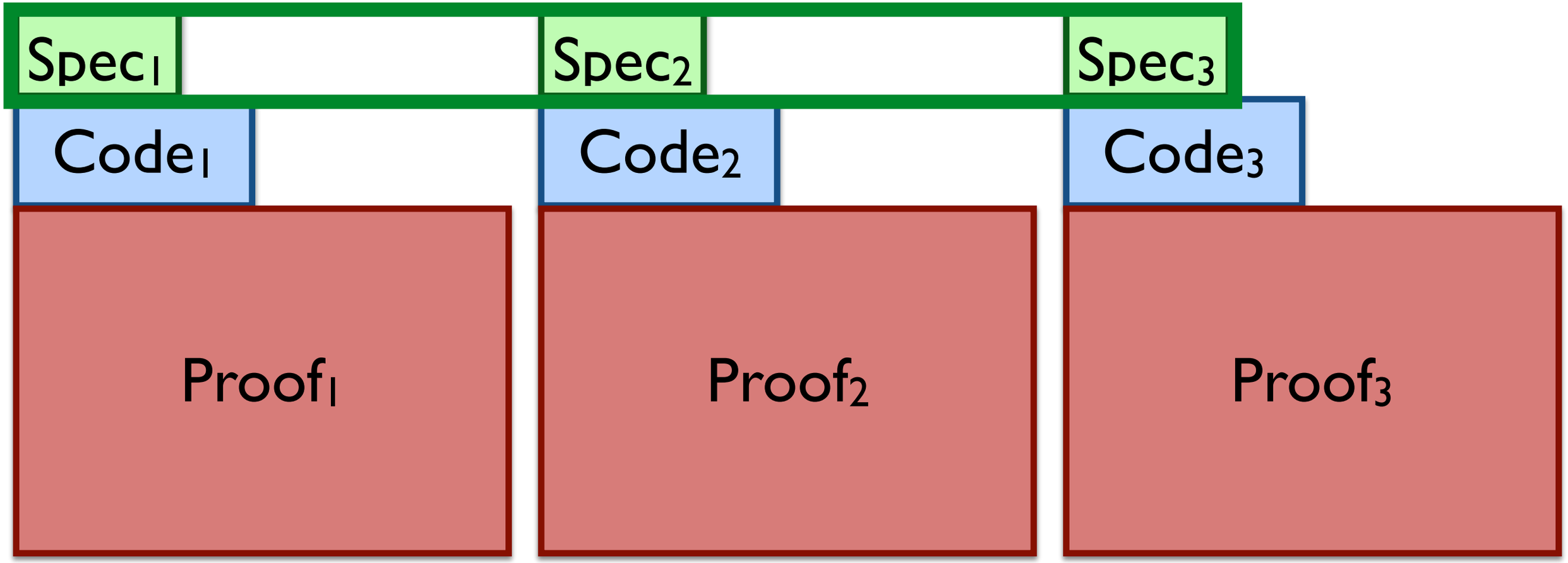


Single application domain



Single application domain

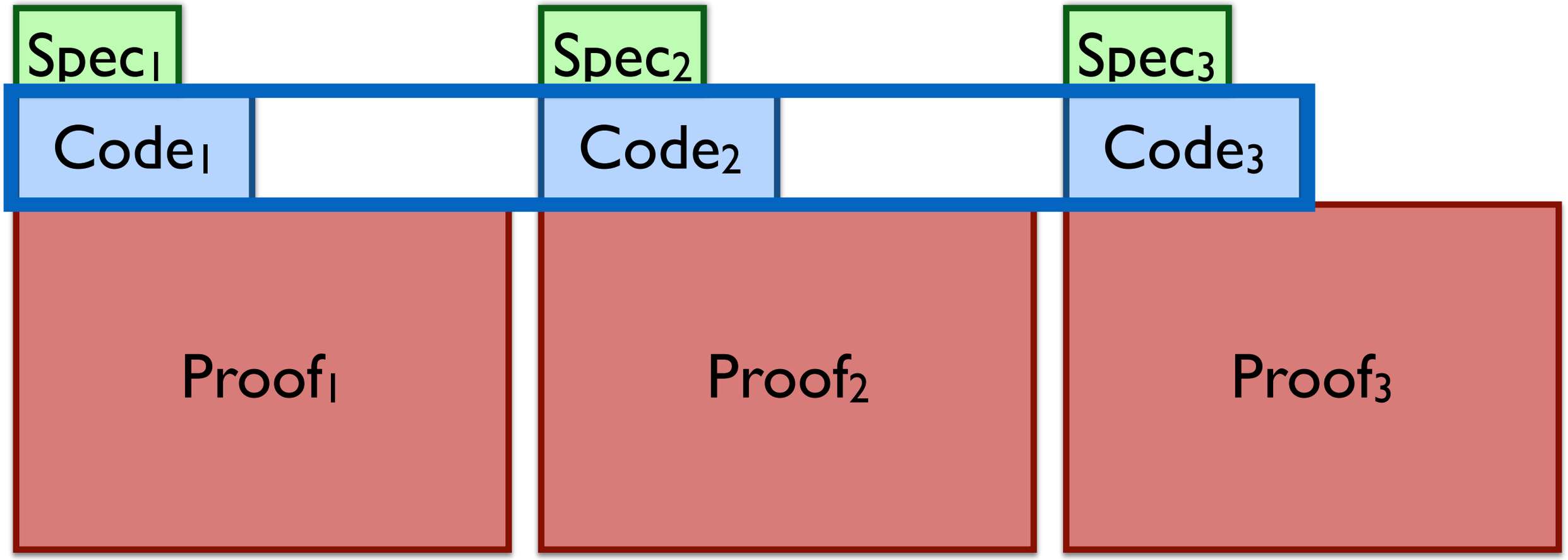
Similar properties



Single application domain

Similar properties

Similar architecture

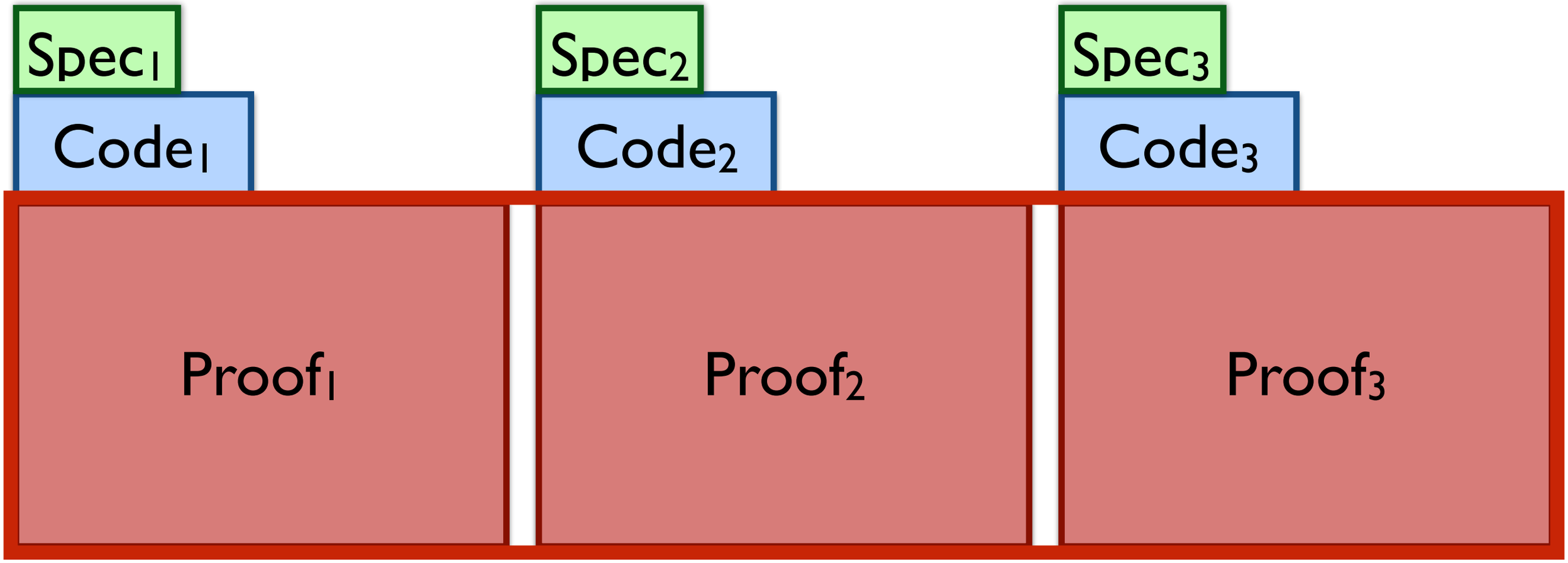


Single application domain

Similar properties

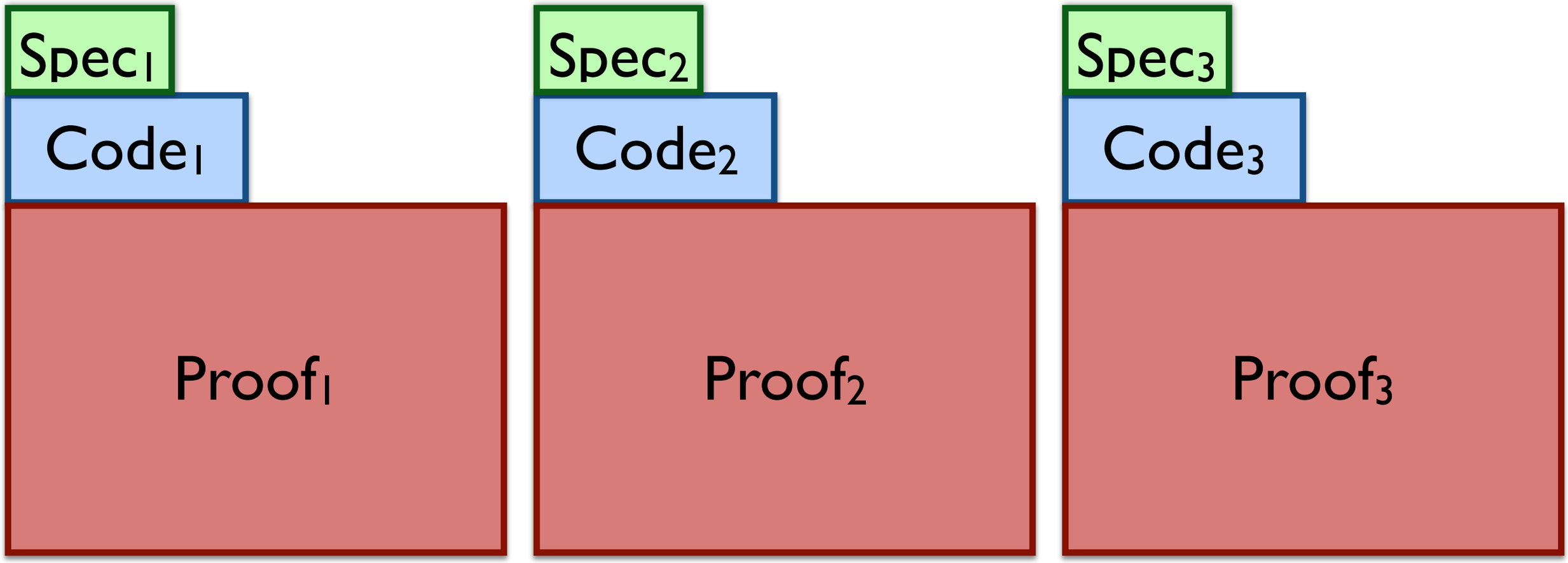
Similar architecture

Similar reasoning



Single application domain

DSL for Specs

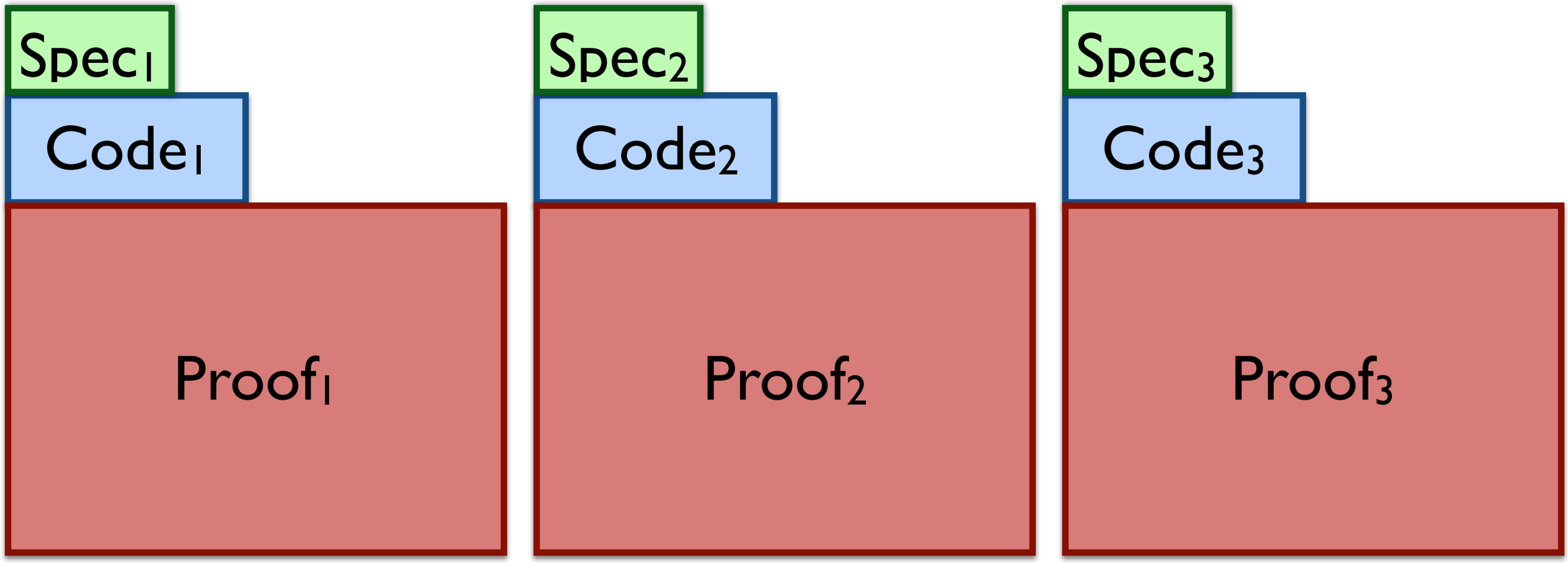


Single application domain

DSL for Specs

Similar architecture

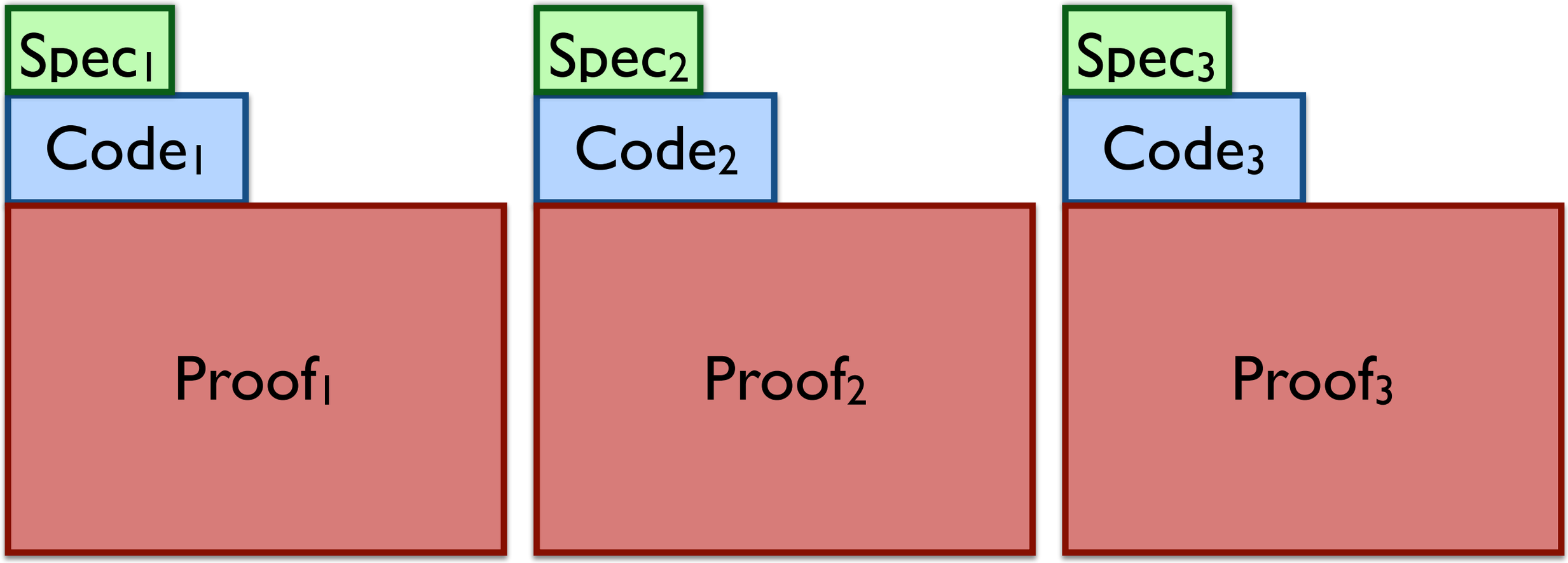
Similar reasoning



Single application domain

DSL for Specs

DSL for Code

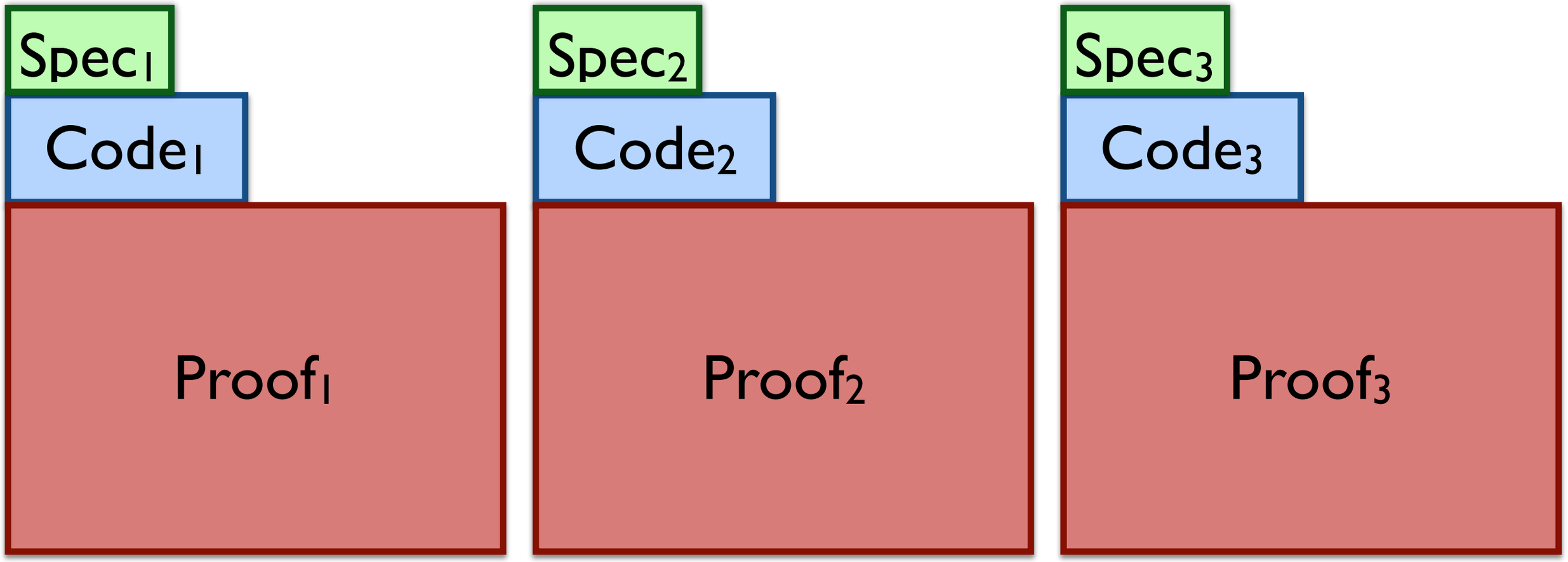


Single application domain

DSL for Specs

DSL for Code

Similar reasoning

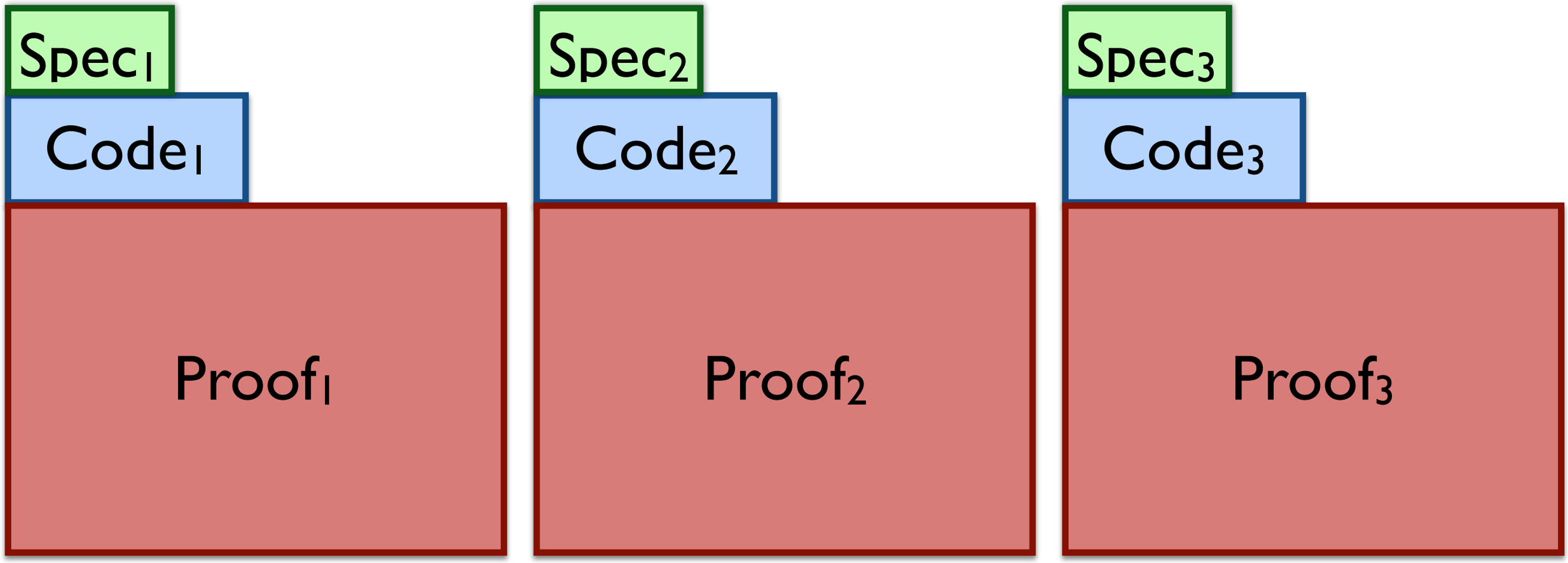


Single application domain

DSL for Specs

DSL for Code

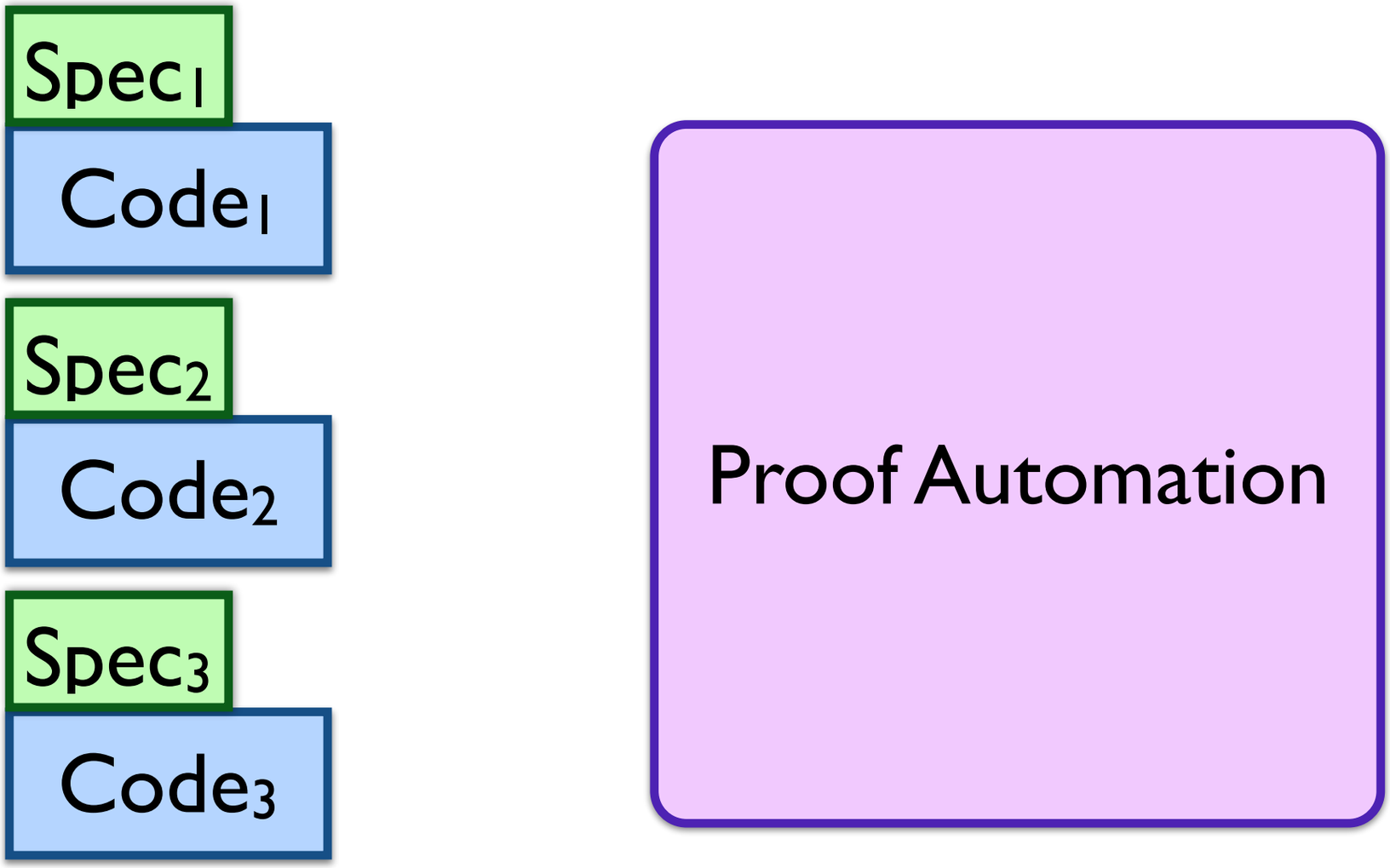
Proof Automation



Single application domain

DSL for Specs

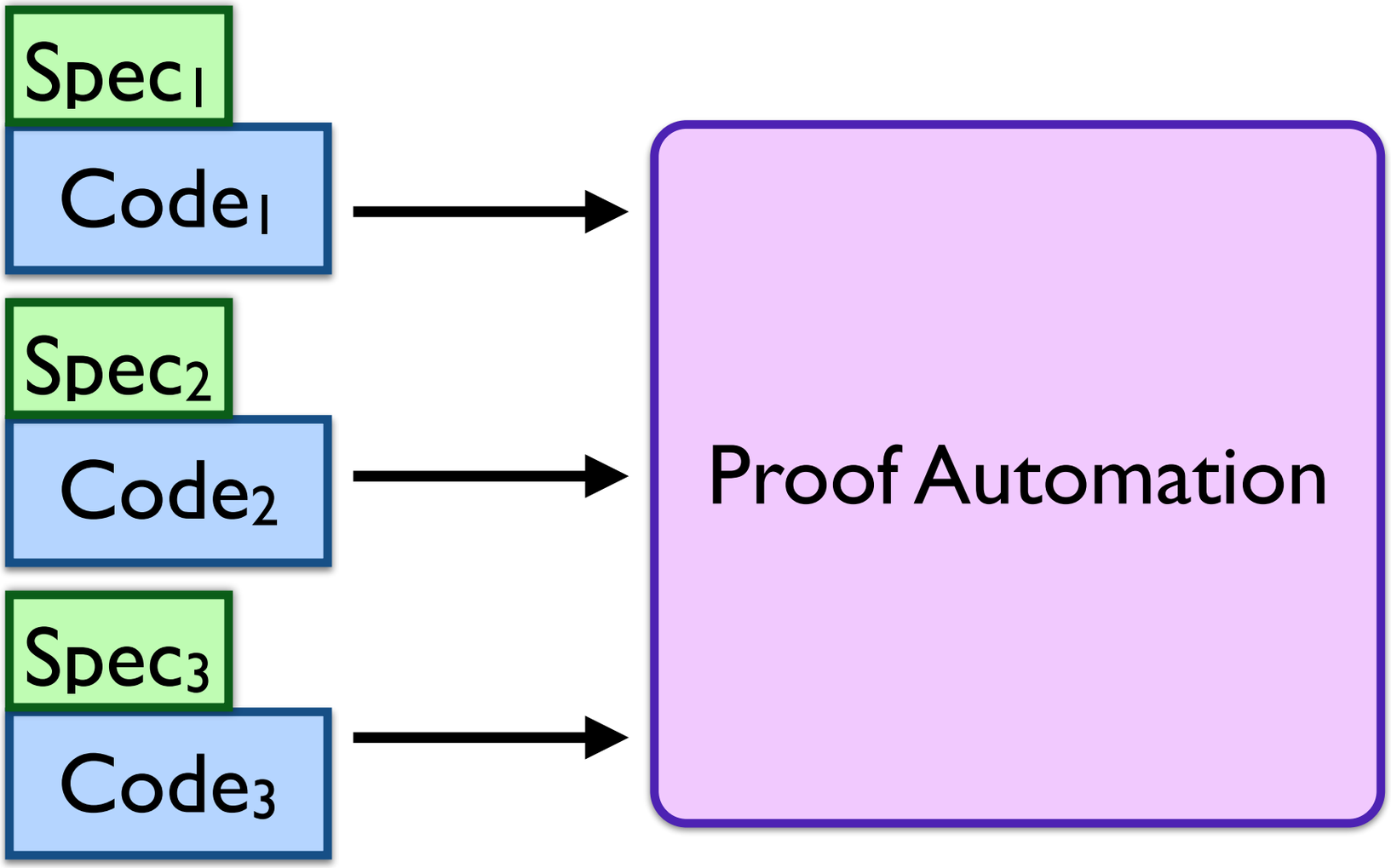
DSL for Code



Single application domain

DSL for Specs

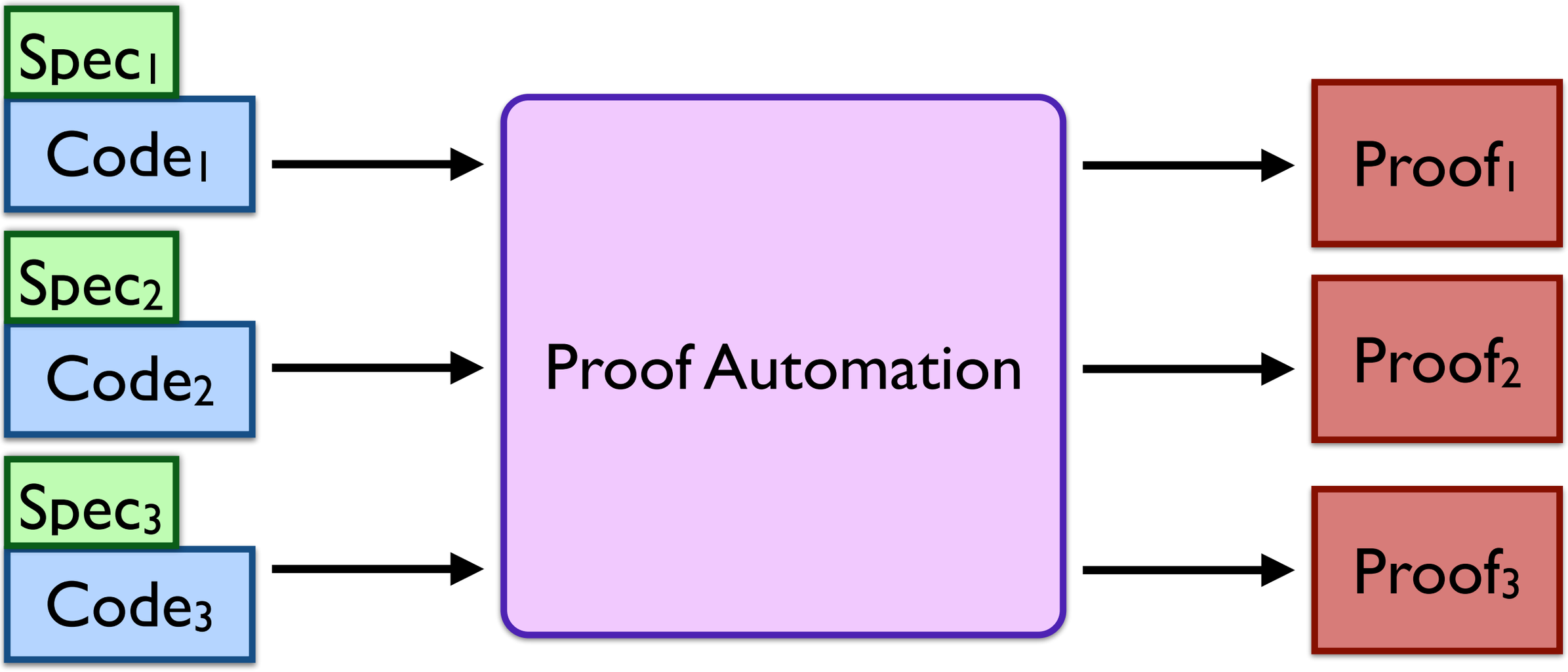
DSL for Code



Single application domain

DSL for Specs

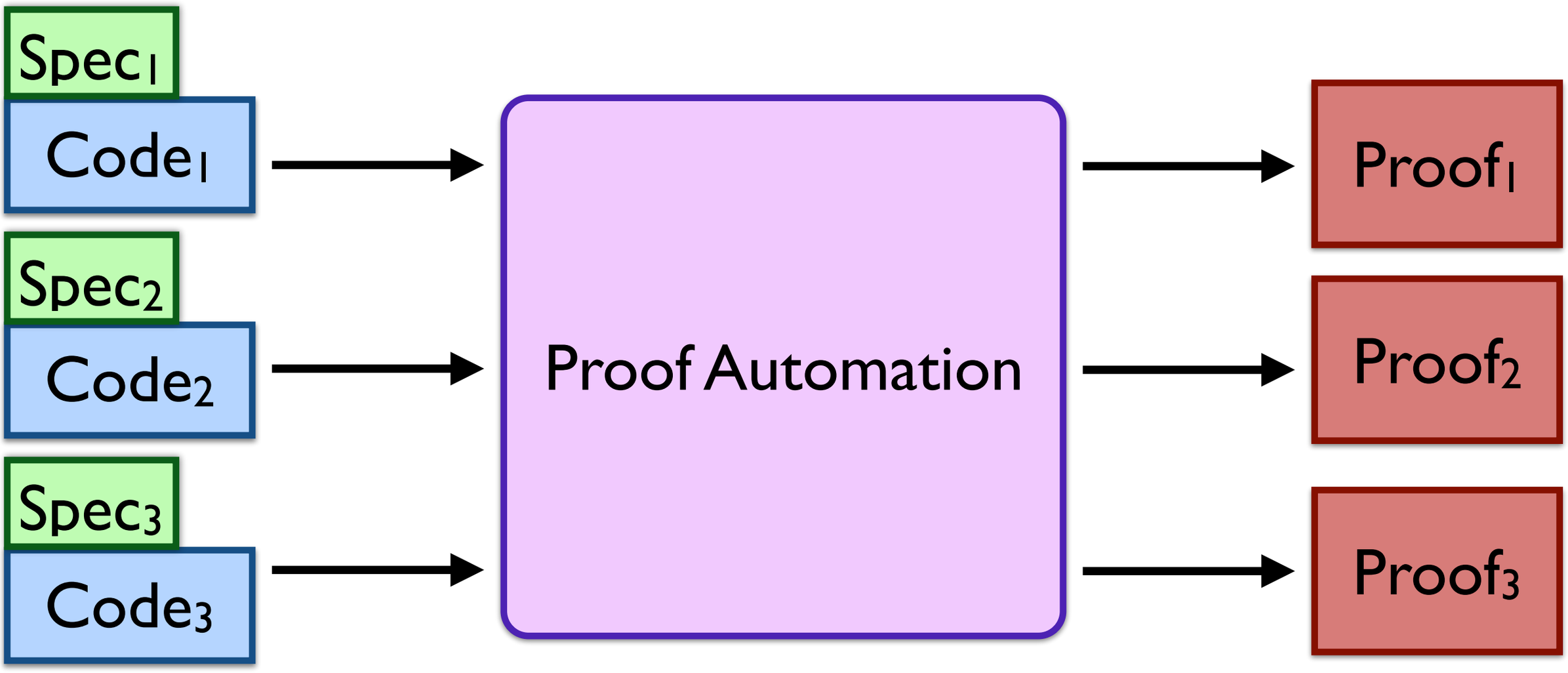
DSL for Code



Reactive systems

DSL for Specs

DSL for Code



Reactive systems

DSL for Specs

DSL for Code

Reactive systems

REFLEX

Reactive systems

REFLEX

No manual proofs,

Reactive systems

REFLEX

No manual proofs,

yet proof assistant guarantee.

Reactive systems

REFLEX

No manual proofs,

yet proof assistant guarantee.

Automation incomplete,

Reactive systems

REFLEX

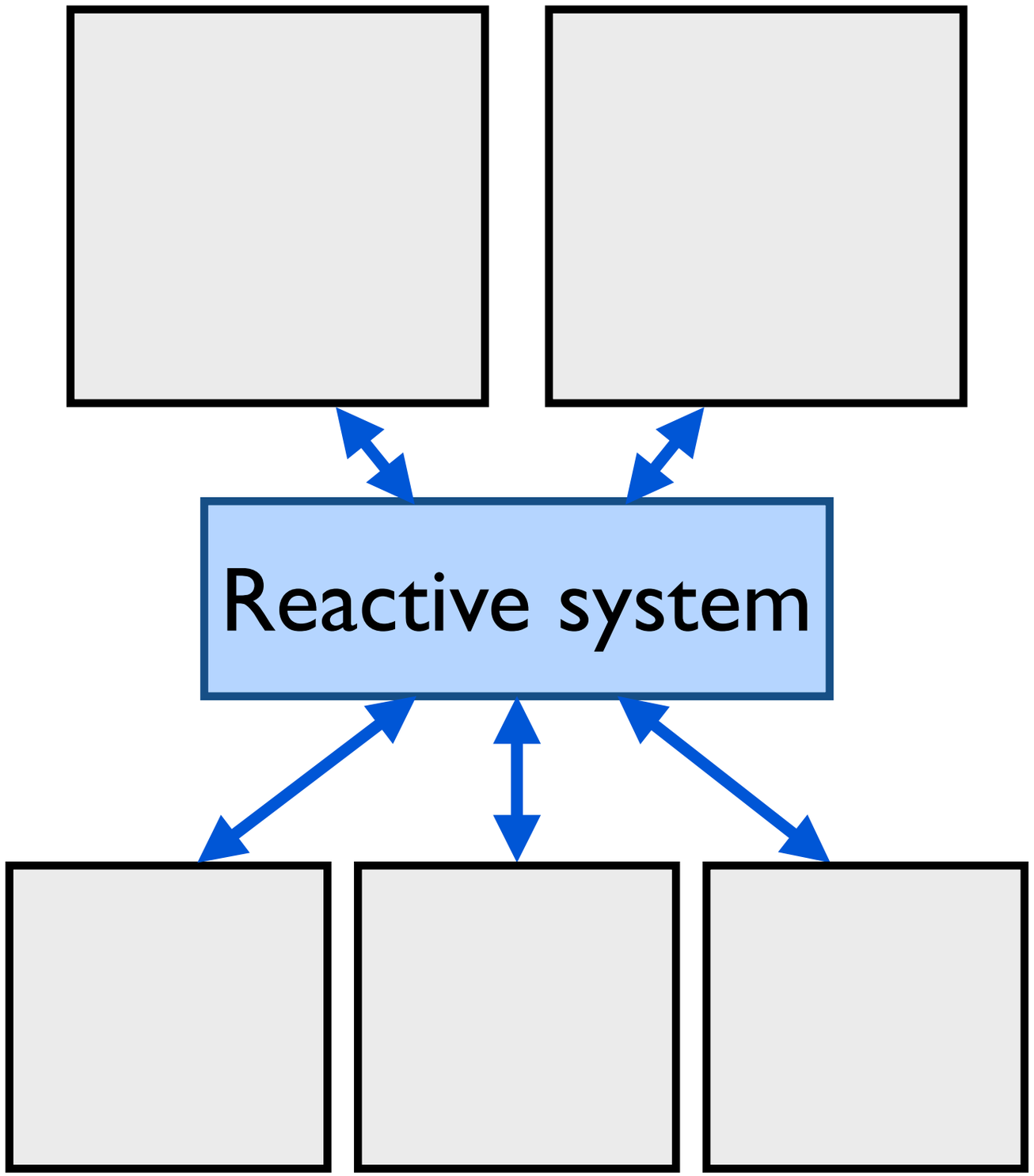
No manual proofs,

yet proof assistant guarantee.

Automation incomplete,

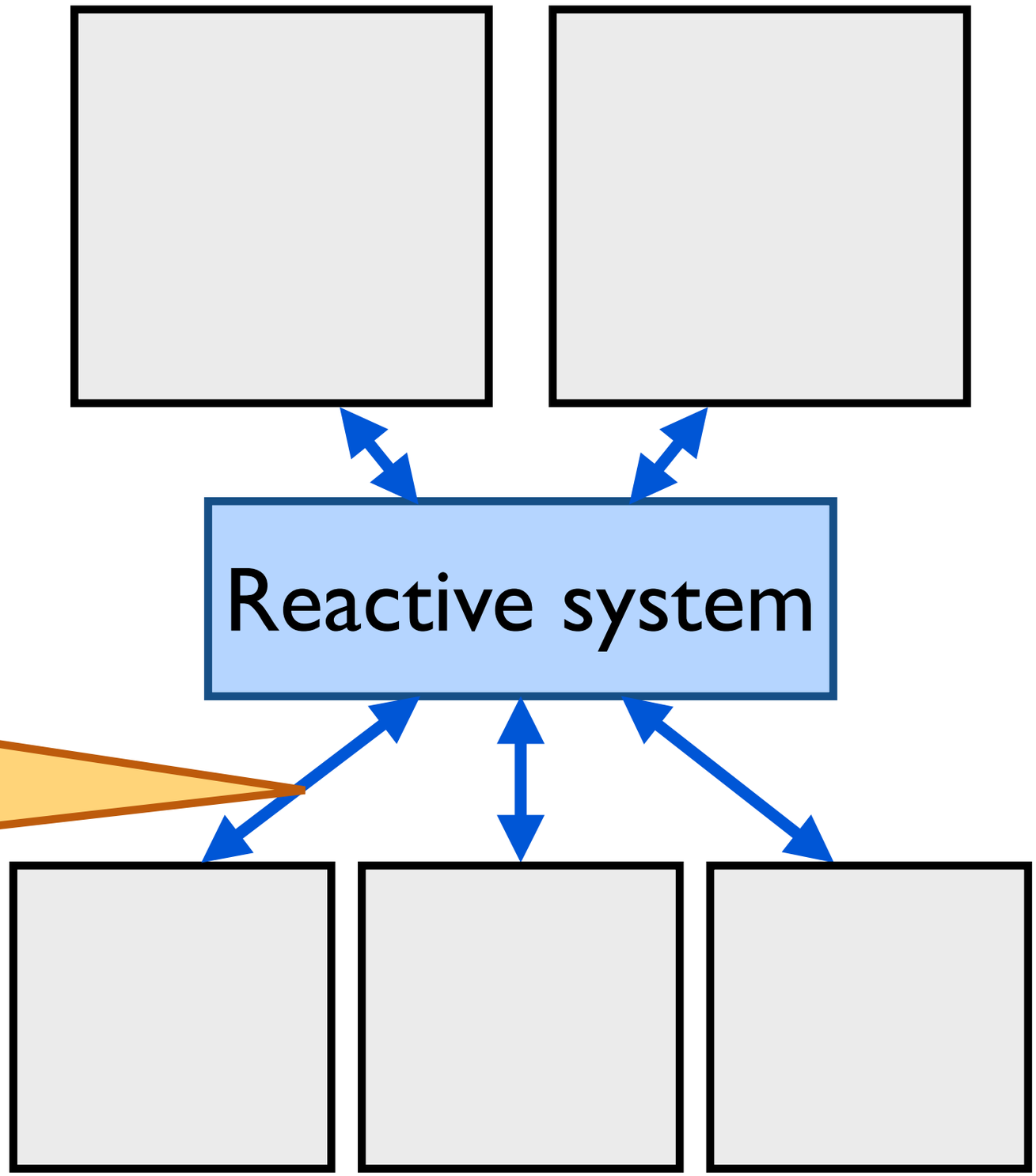
but verified browser, ssh, web server.

Reactive systems

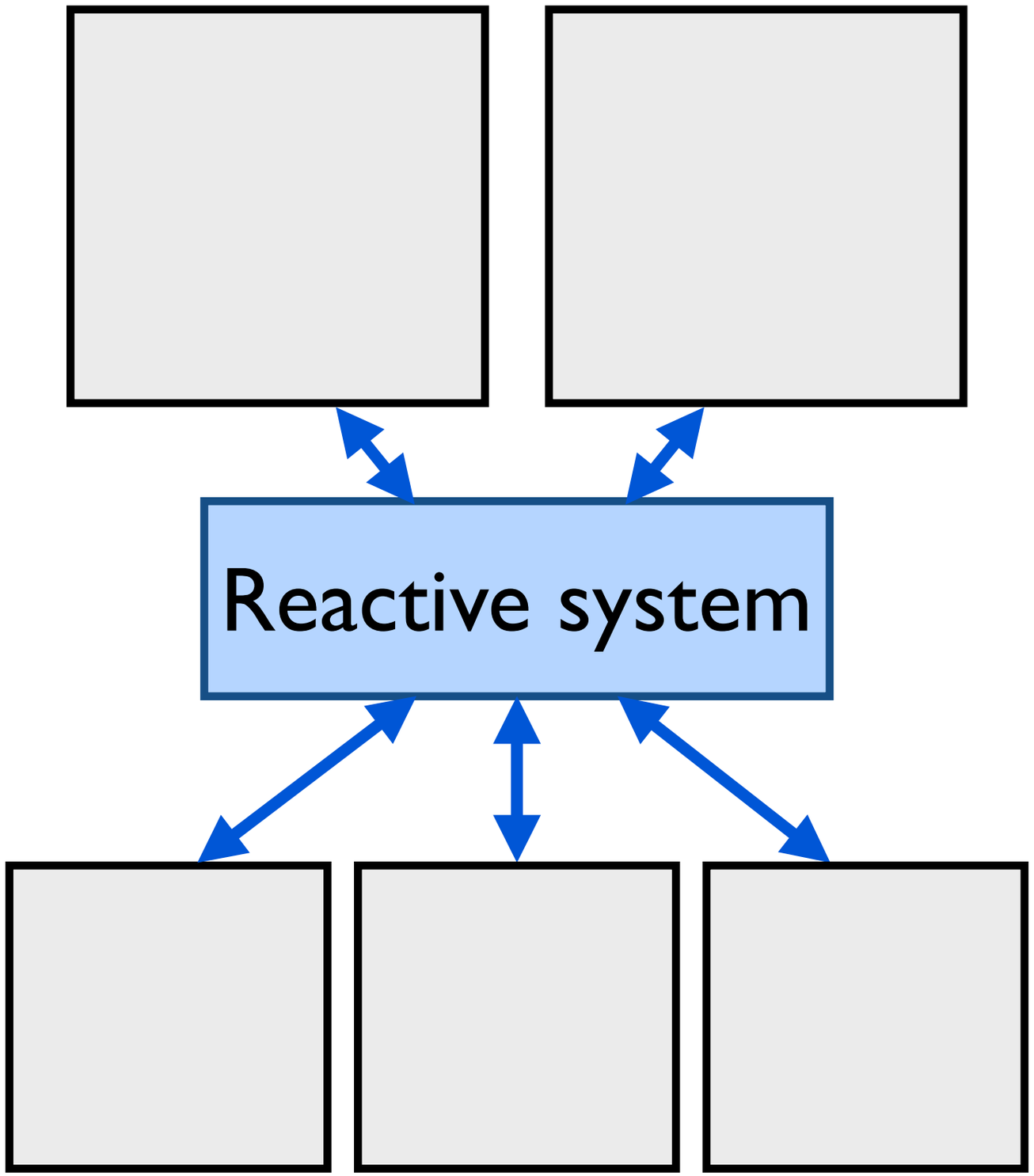


Reactive systems

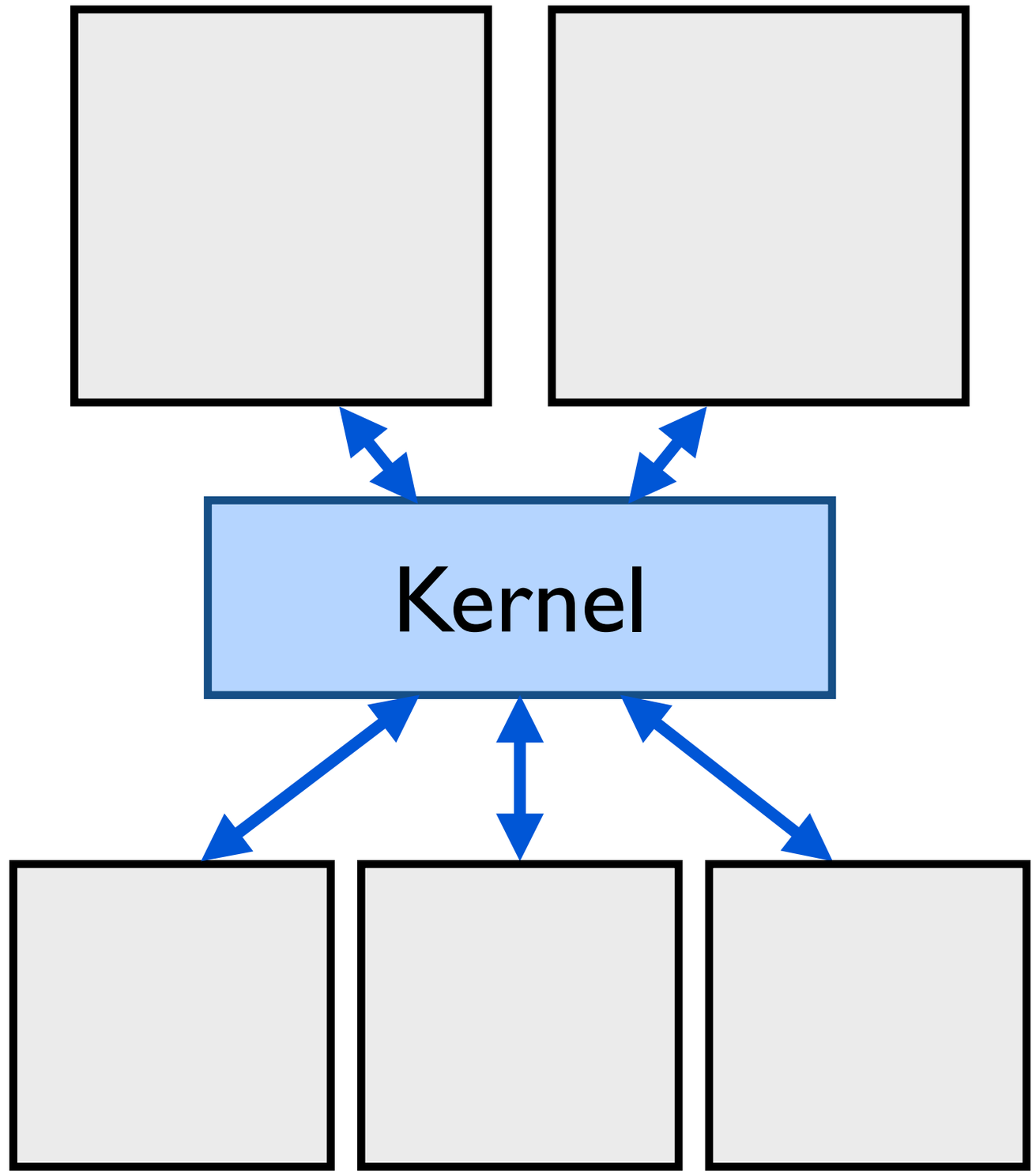
Continuously read messages from components and send messages to components



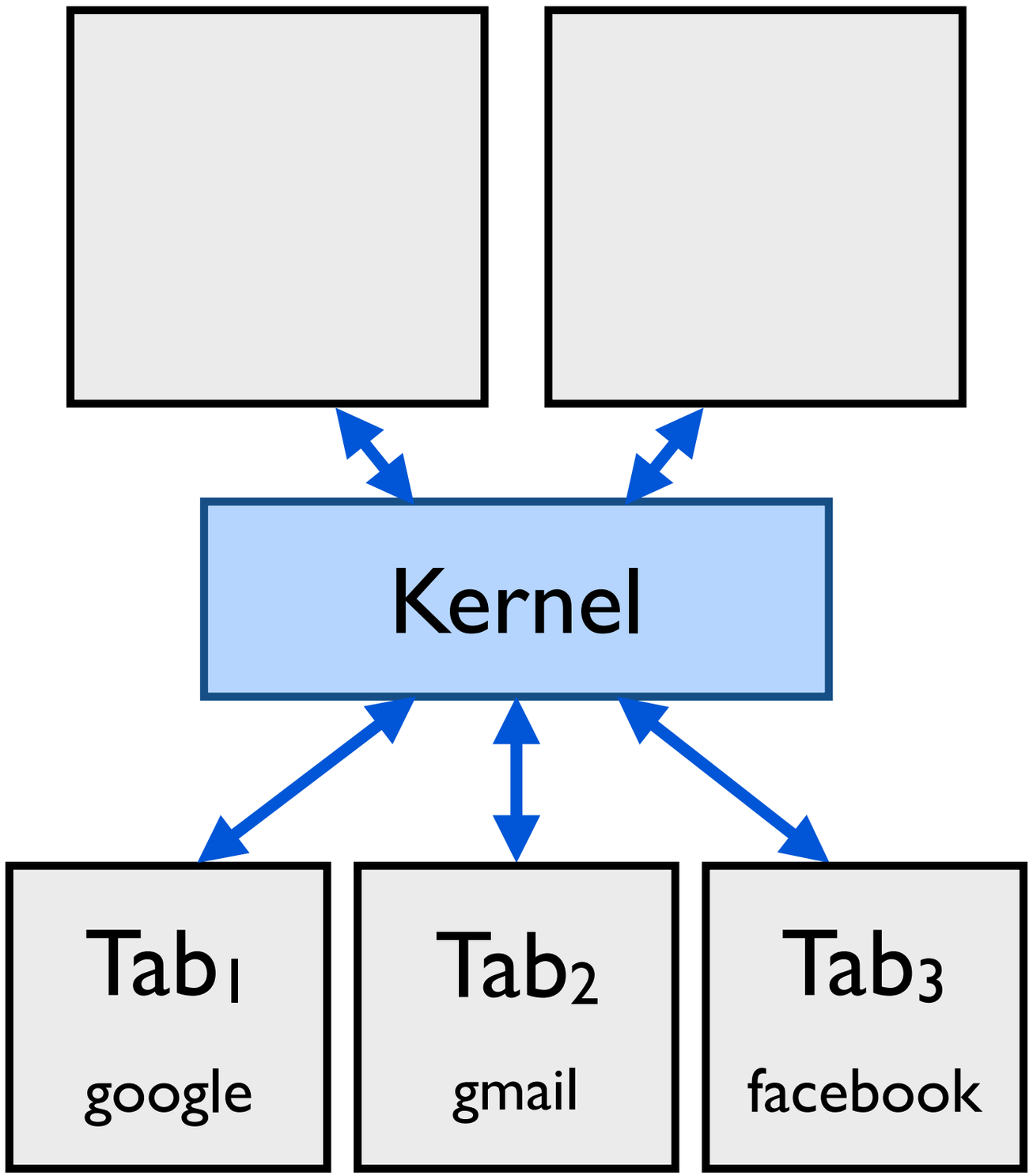
Reactive systems



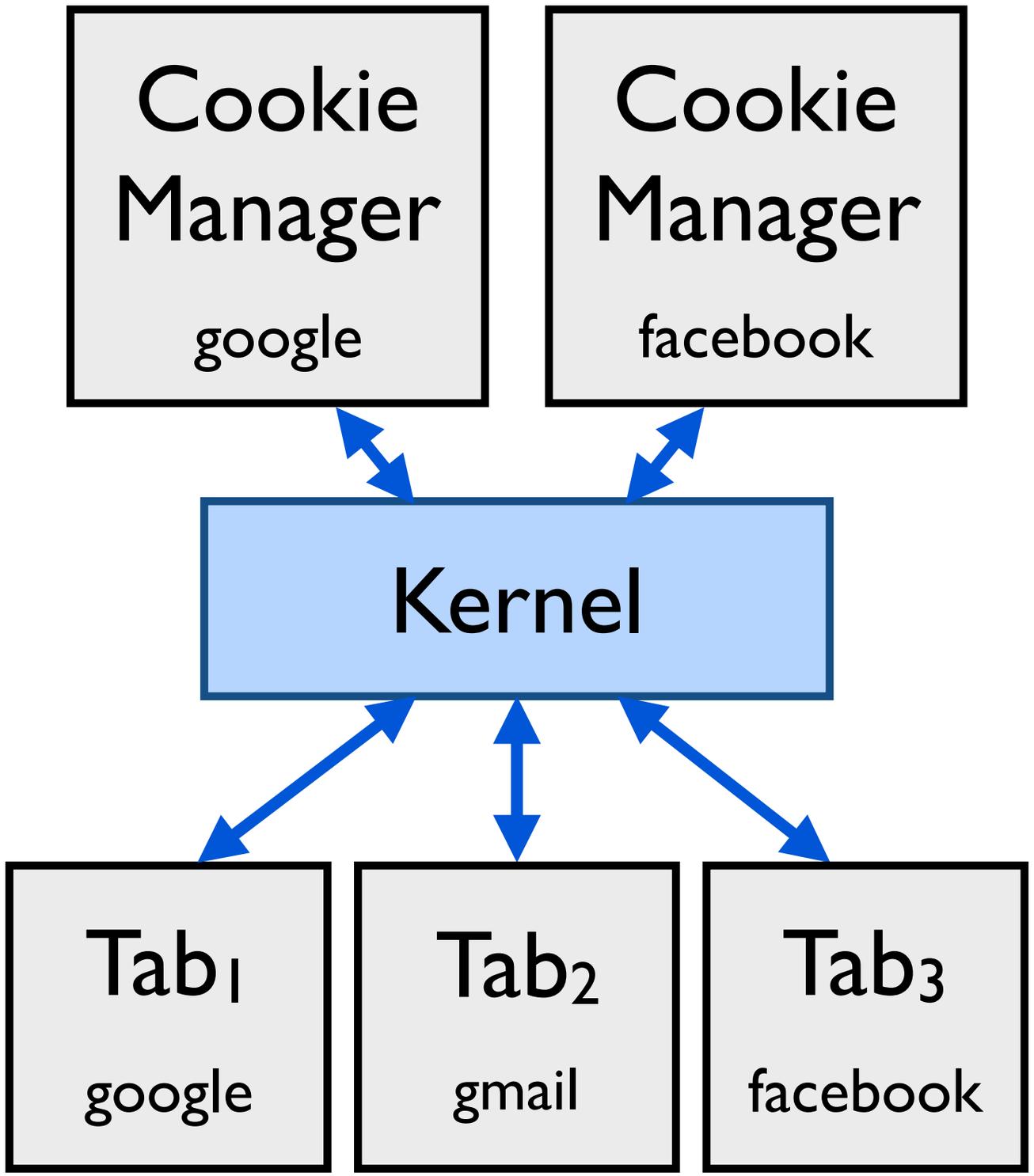
Example: Web browser kernel



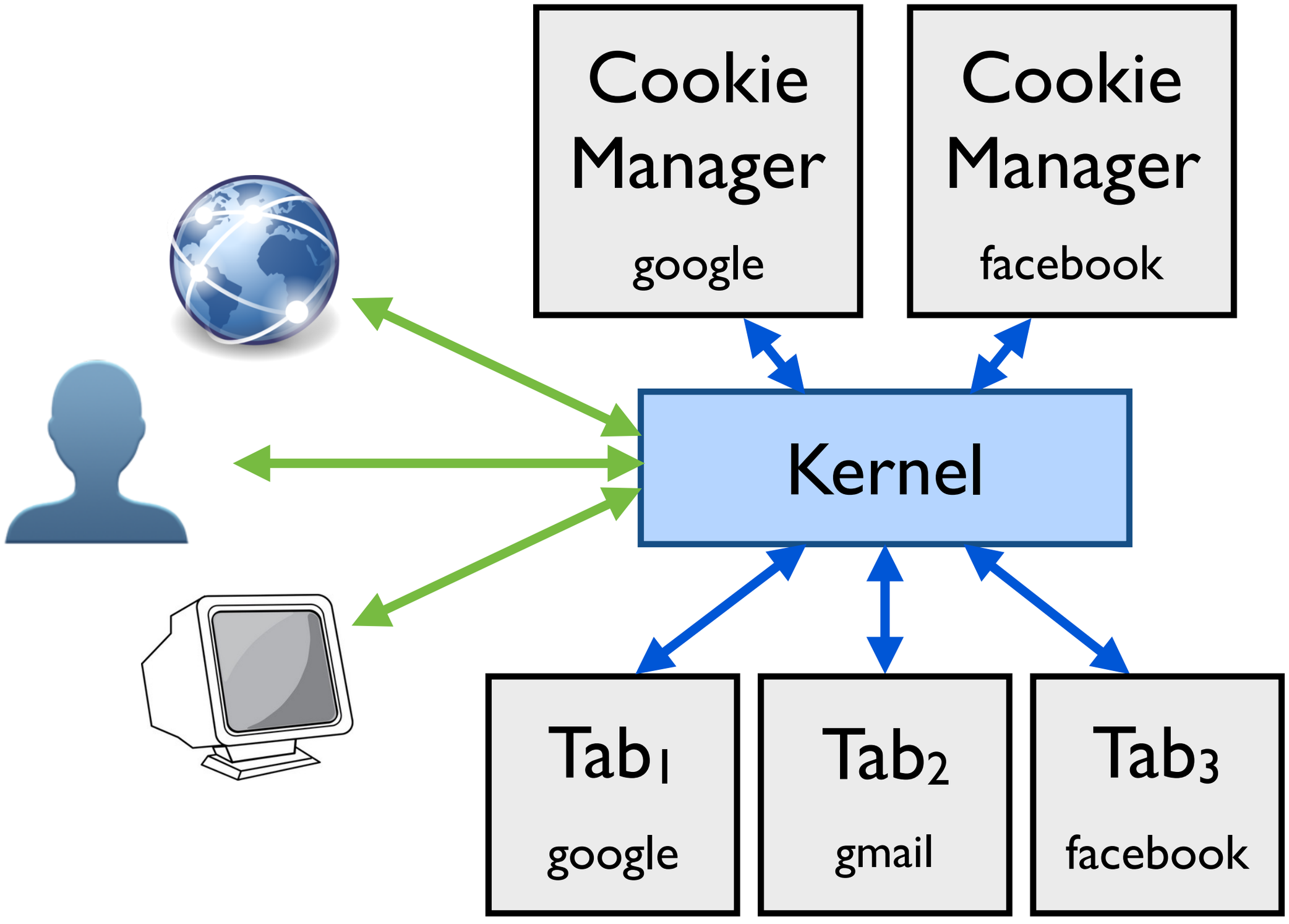
Example: Web browser kernel



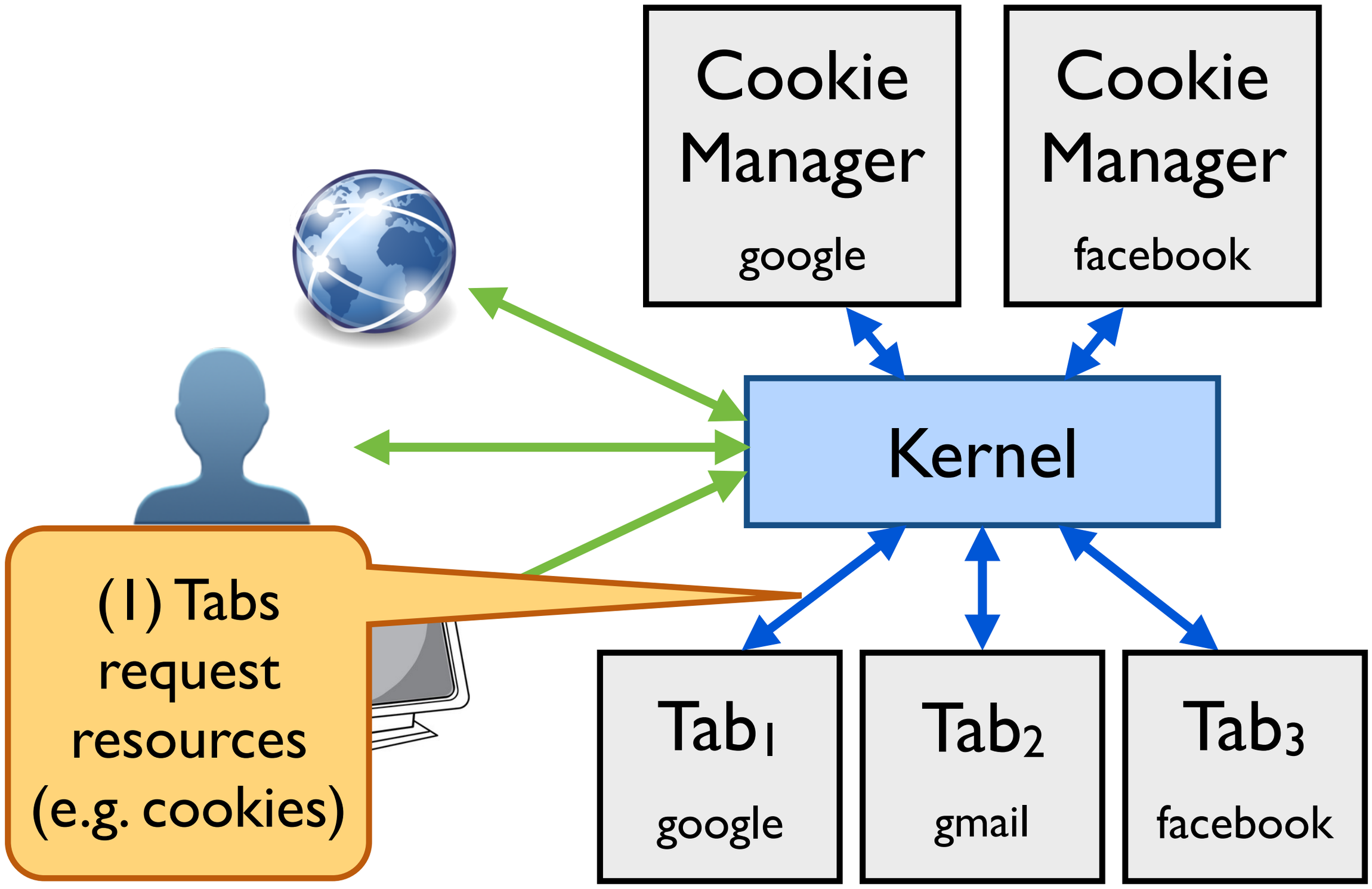
Example: Web browser kernel



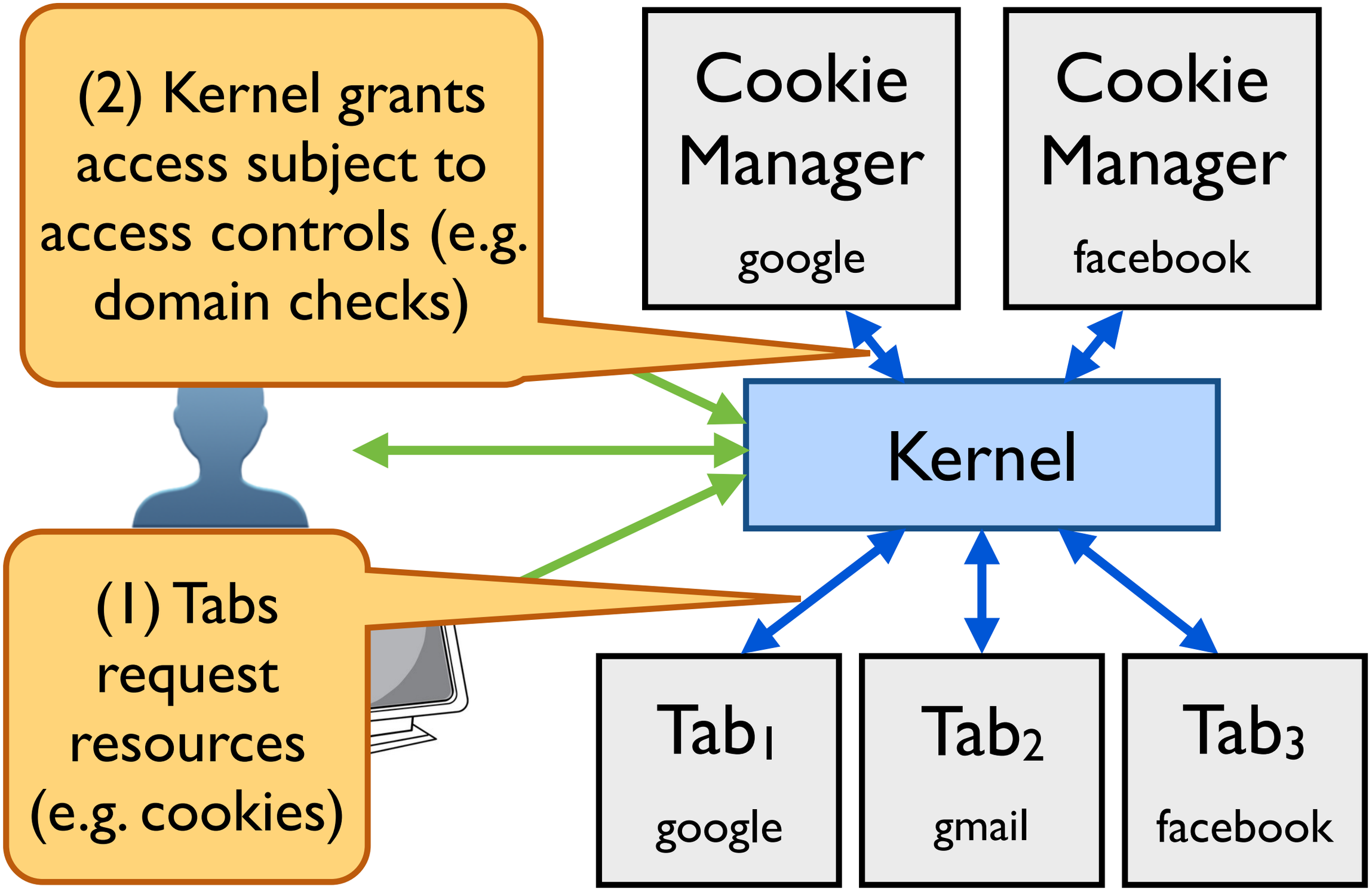
Example: Web browser kernel



Example: Web browser kernel

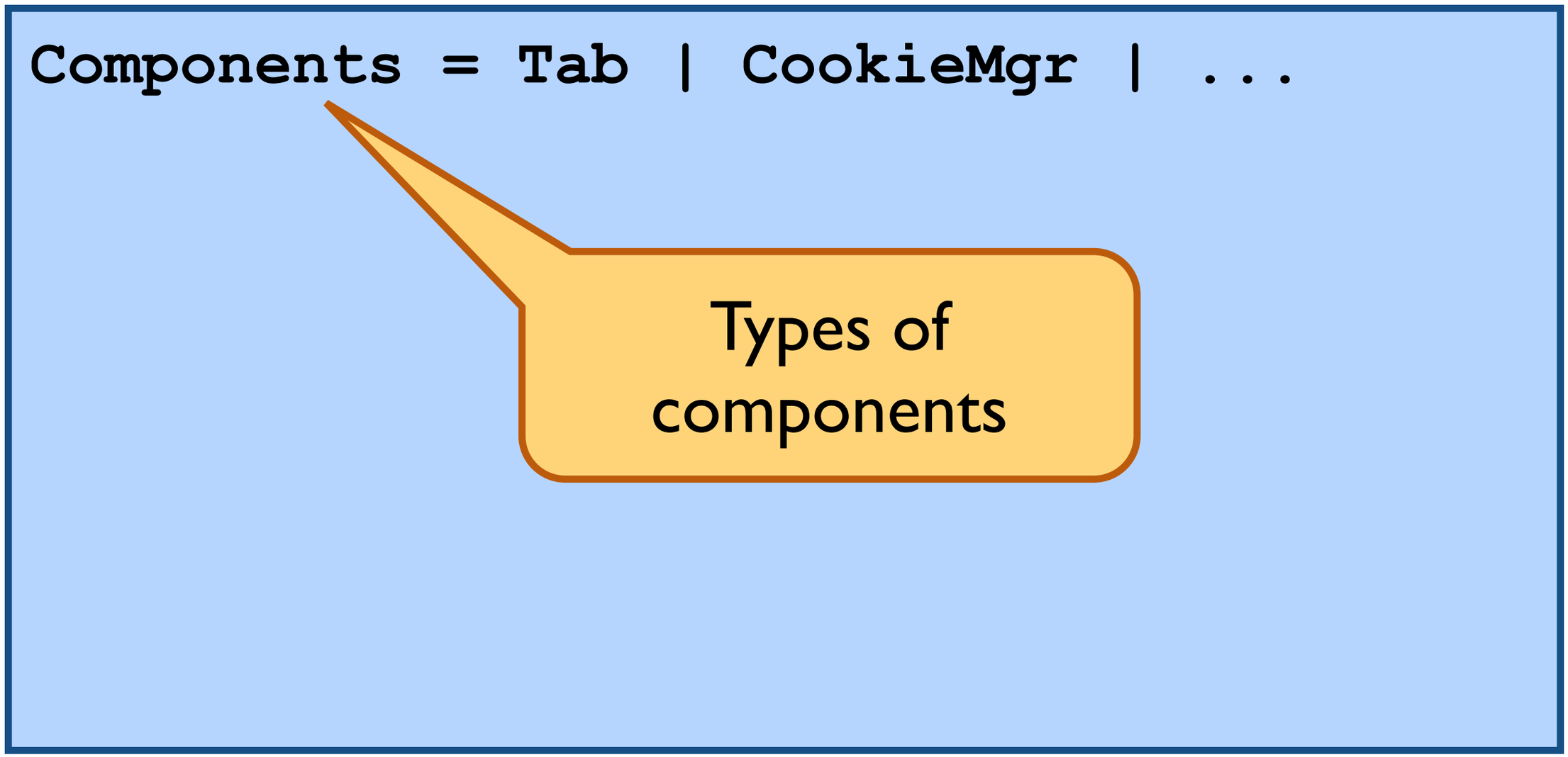


Example: Web browser kernel



Example: Web browser kernel

Example: Web browser kernel



Example: Web browser kernel

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...
```

Types of messages

Example: Web browser kernel

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
```

How the kernel responds to messages from components

Example: Web browser kernel

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
```

When tab **`t`** sends the kernel a **`CookieSet`** message with payload **`c`**

Example: Web browser kernel

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
     cp <- find CookieMgr(t.domain)
```

Get existing cookie manager with domain of `t` or spawn a new one

Example: Web browser kernel

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
      cp <- find CookieMgr(t.domain)
      send(cp, CookieSet(c))
```

Send the cookie **c** to the found cookie manager

Example: Web browser kernel

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
     cp <- find CookieMgr(t.domain)
     send(cp, CookieSet(c))

  When [Tab t] sends CookieGet(c):
     ...
```

More handlers

Properties

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
      cp <- find CookieMgr(t.domain)
      send(cp, CookieSet(c))

   When [Tab t] sends CookieGet(c):
      ...
```

Properties

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
     cp <- find CookieMgr(t.domain)
     send(cp, CookieSet(c))

When [Tab t] sends CookieGet(c):
     ...
```

Specify allowed behaviors

Properties

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
     cp <- find CookieMgr(t.domain)
     send(cp, CookieSet(c))

  When [Tab t] sends CookieGet(c):
     ...
```

Specify allowed behaviors wrt sequence of system calls

Properties

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
```

```
  When [Tab t] sends CookieSet(c):
    cp <- find CookieMgr(t.domain)
    send(cp, CookieSet(c))
```

```
  When [Tab t] sends CookieGet(c):
    ...
```

Specify allowed behaviors wrt sequence of system calls

Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

Specify allowed behaviors wrt sequence of system calls

Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

Specify allowed behaviors wrt sequence of system calls

Properties

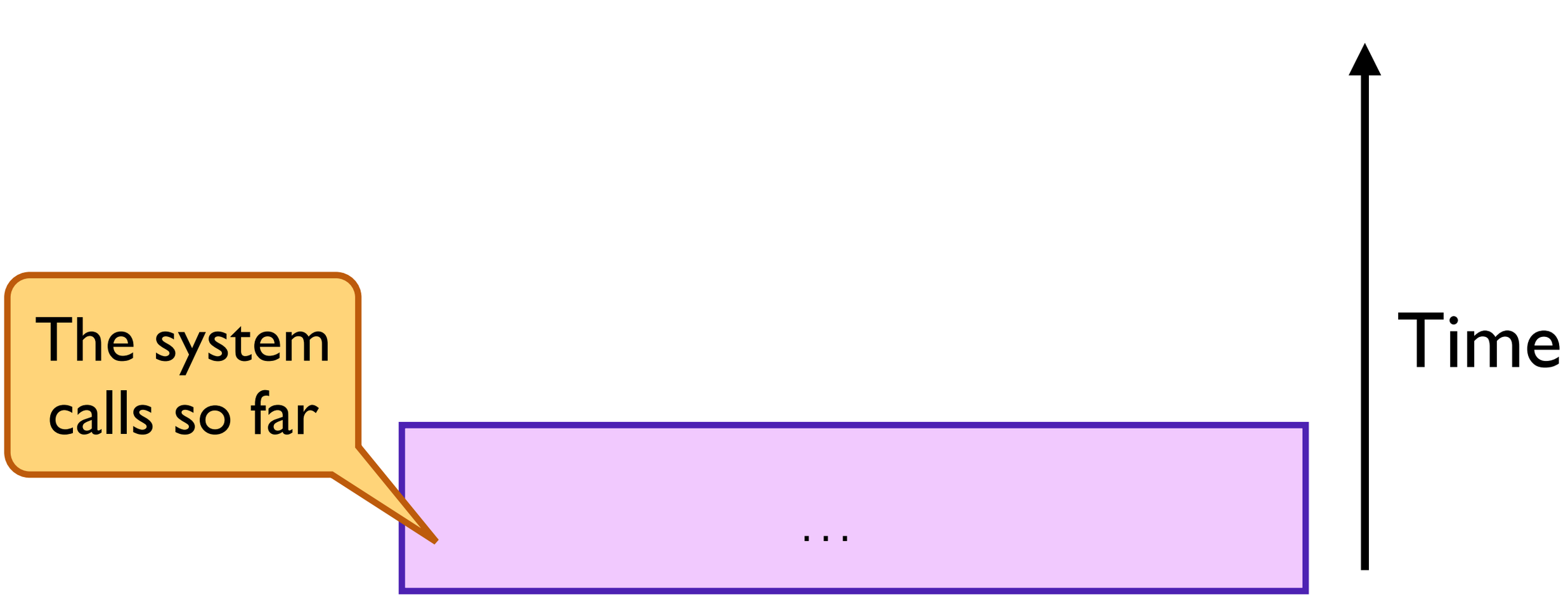
```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

Specify allowed behaviors wrt sequence of system calls

Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

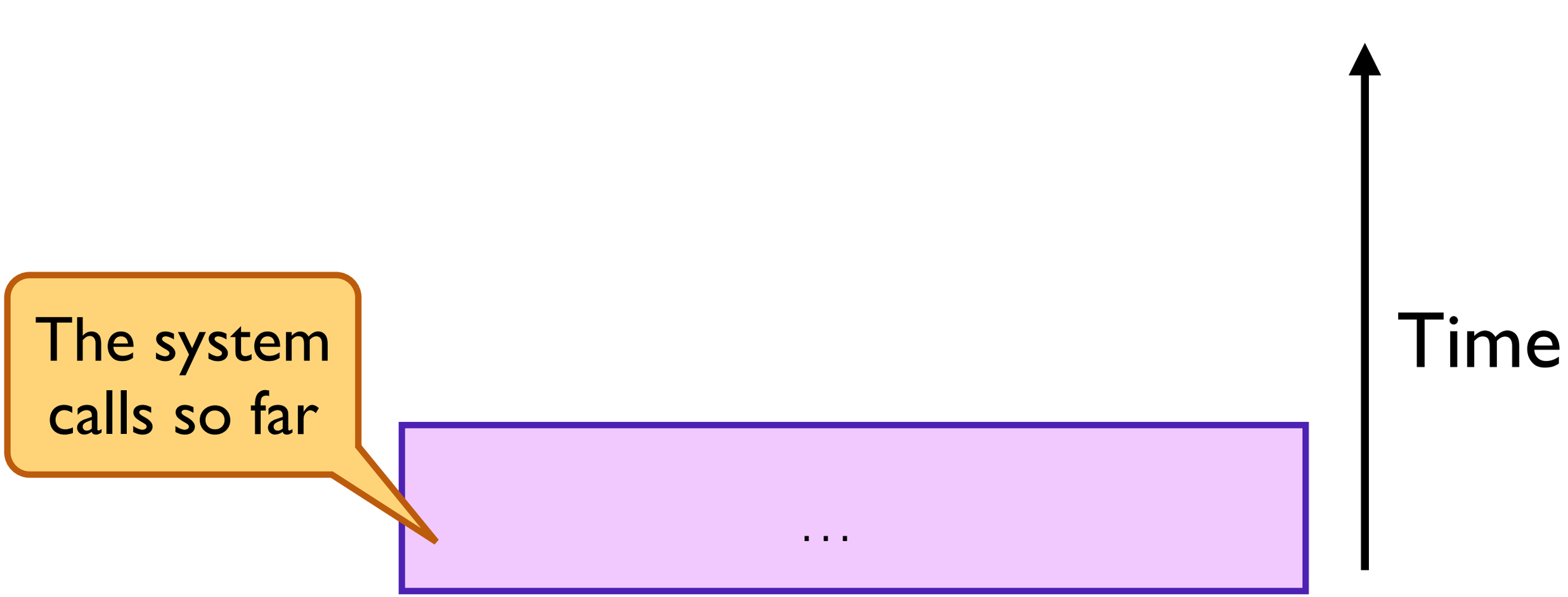
Specify allowed behaviors wrt sequence of system calls



Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

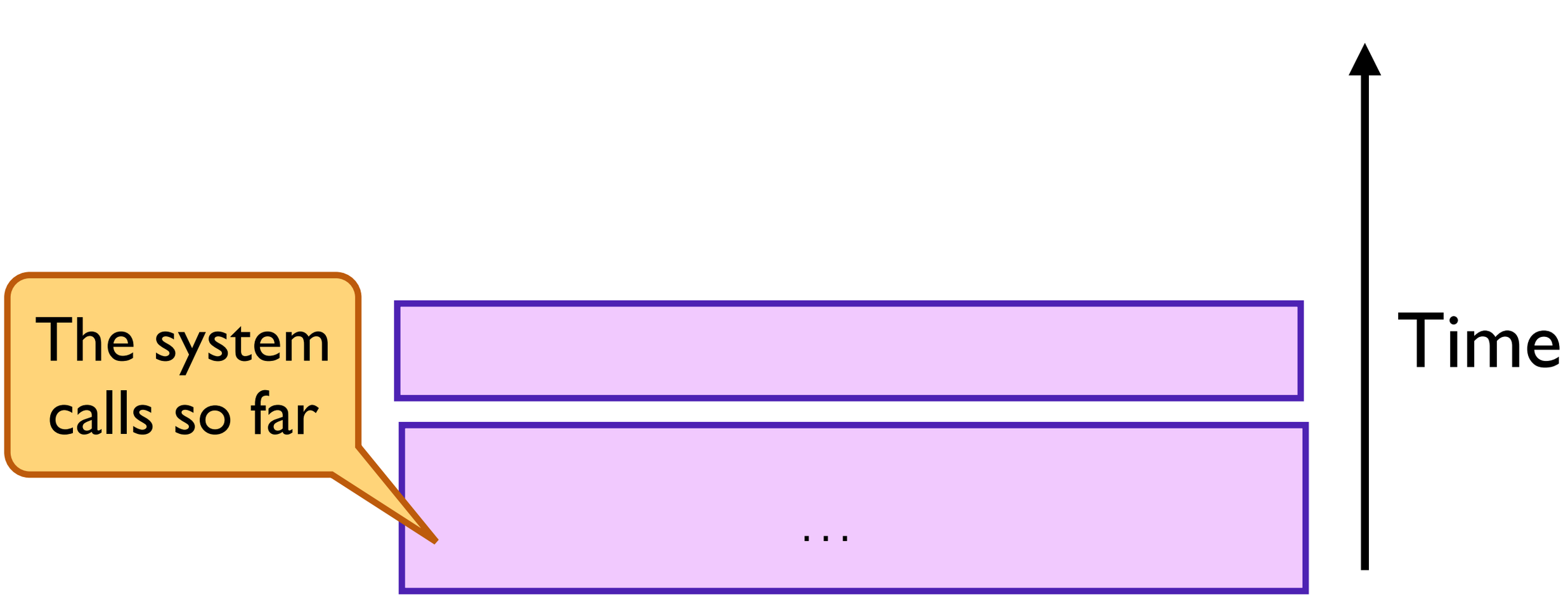
Specify allowed behaviors wrt sequence of system calls



Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

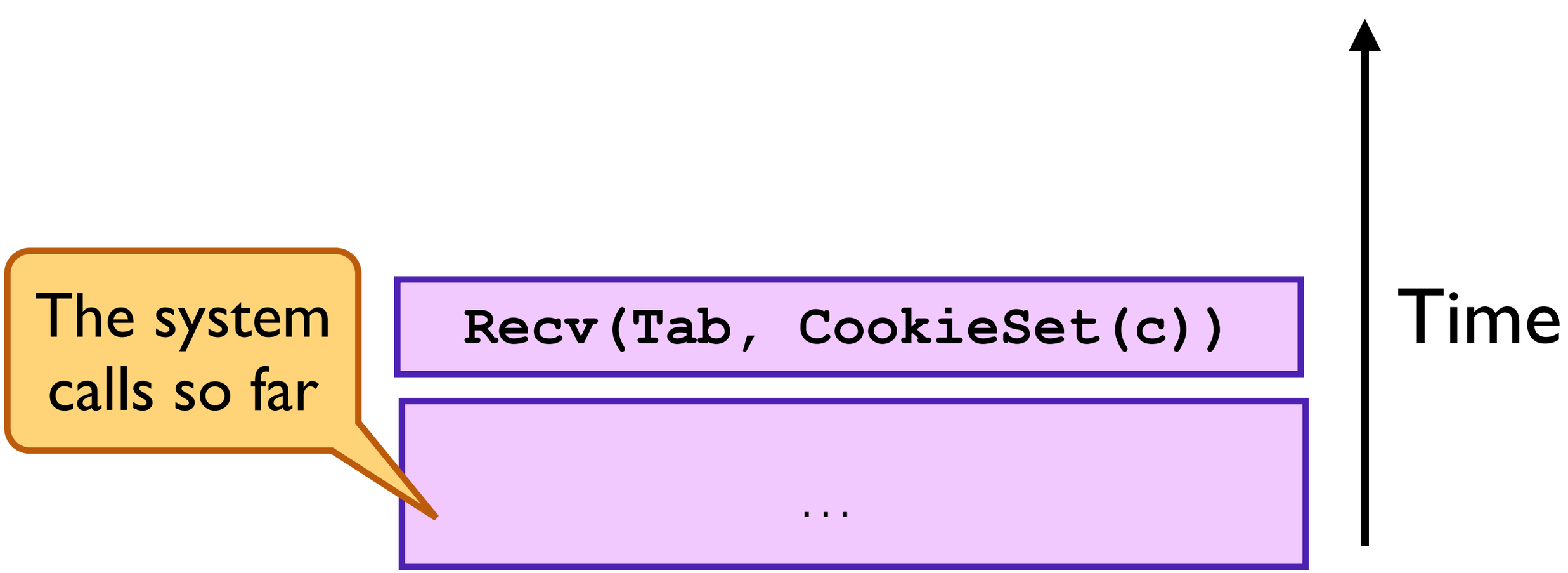
Specify allowed behaviors wrt sequence of system calls



Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

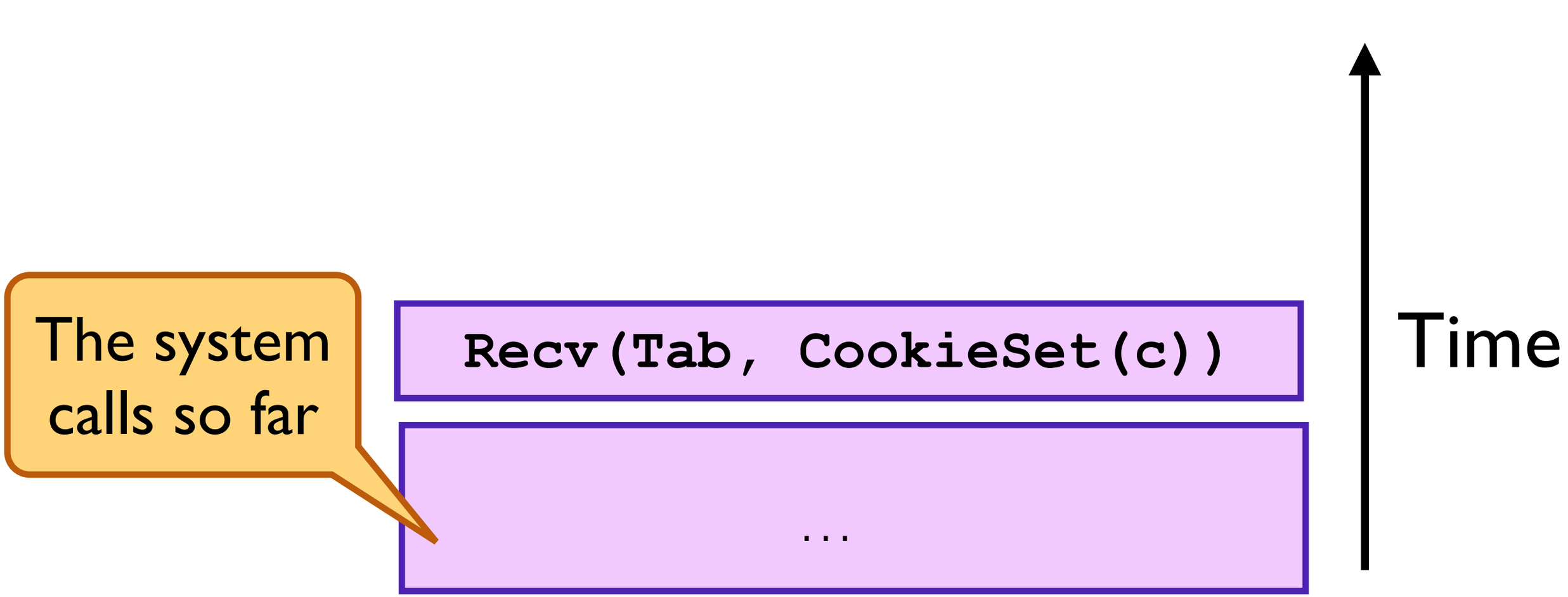
Specify allowed behaviors wrt sequence of system calls



Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

Specify allowed behaviors wrt sequence of system calls



Properties

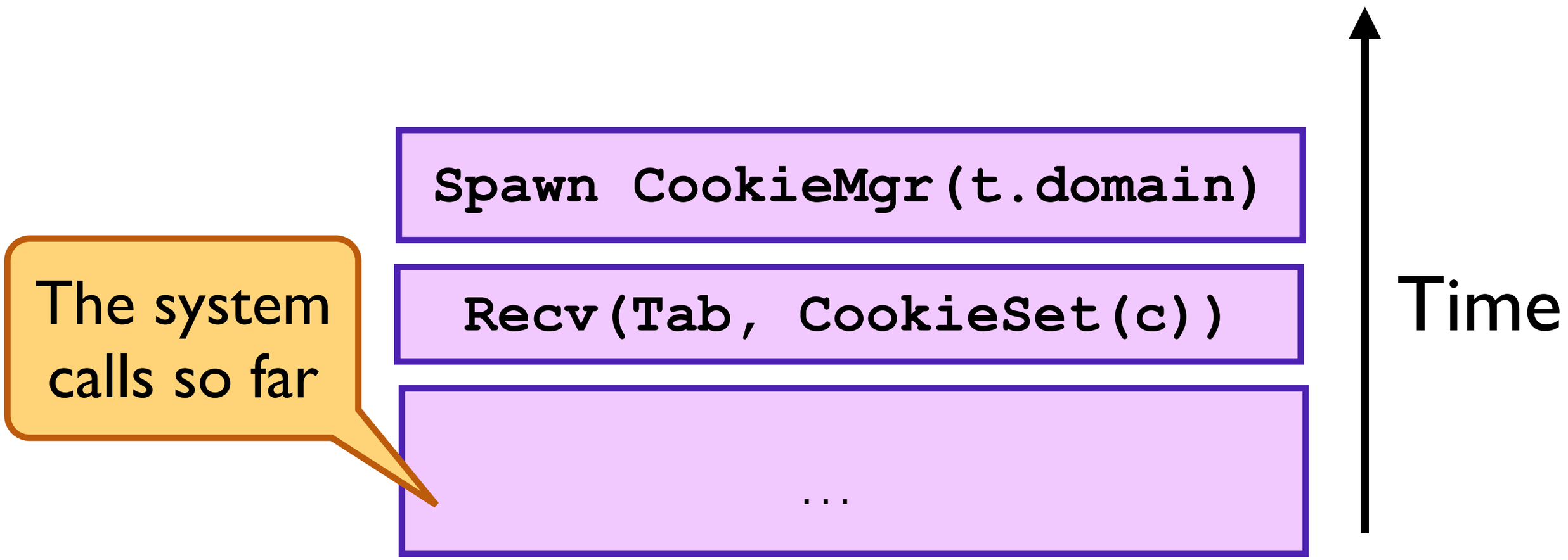
```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

Specify allowed behaviors wrt sequence of system calls

Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

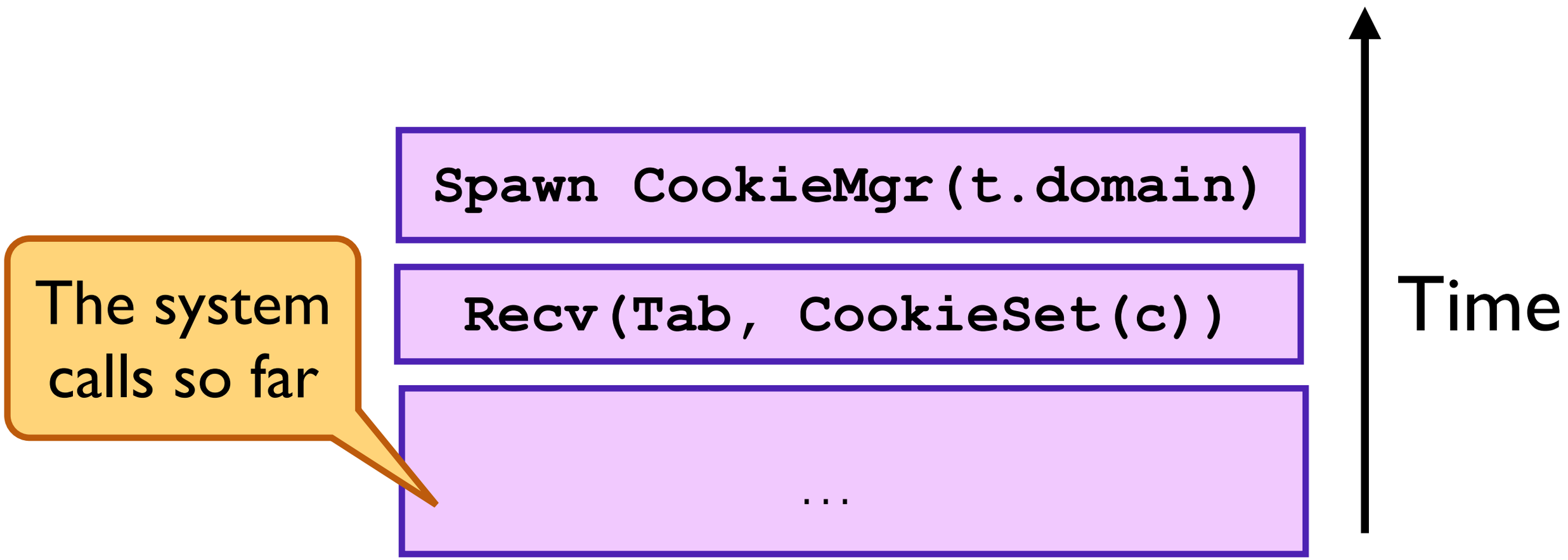
Specify allowed behaviors wrt sequence of system calls



Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

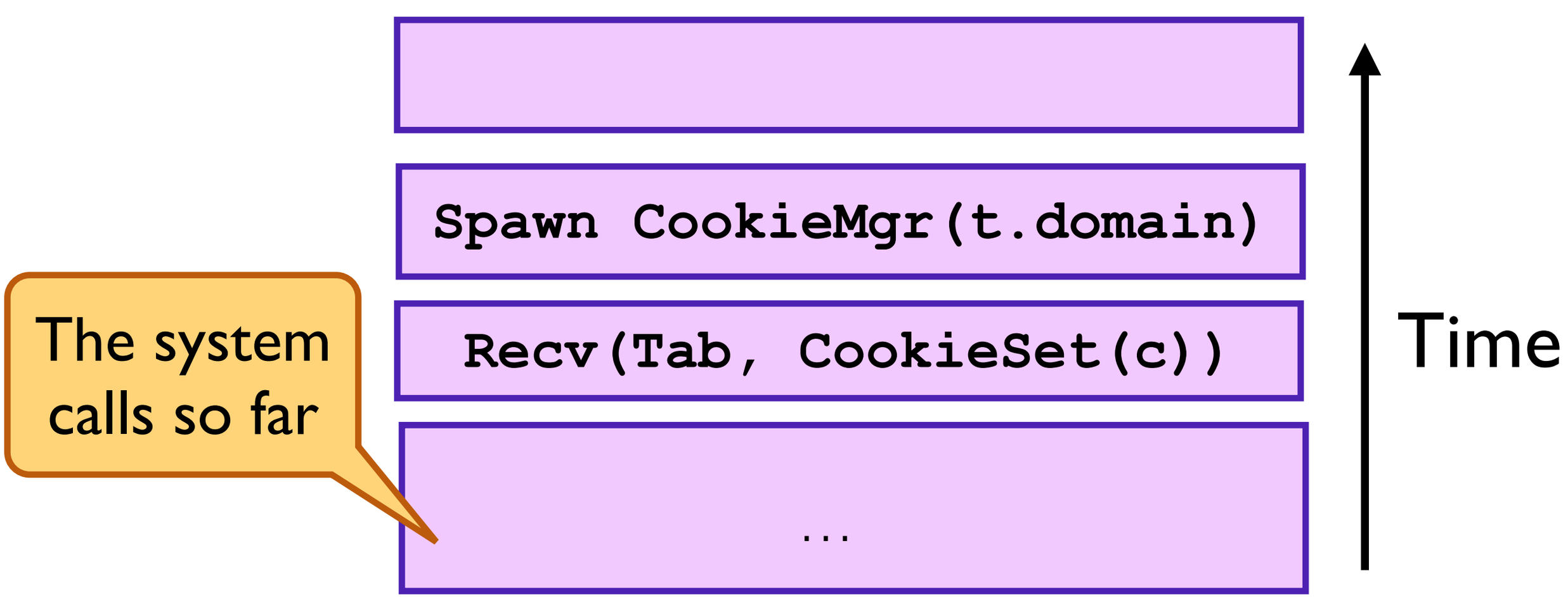
Specify allowed behaviors wrt sequence of system calls



Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

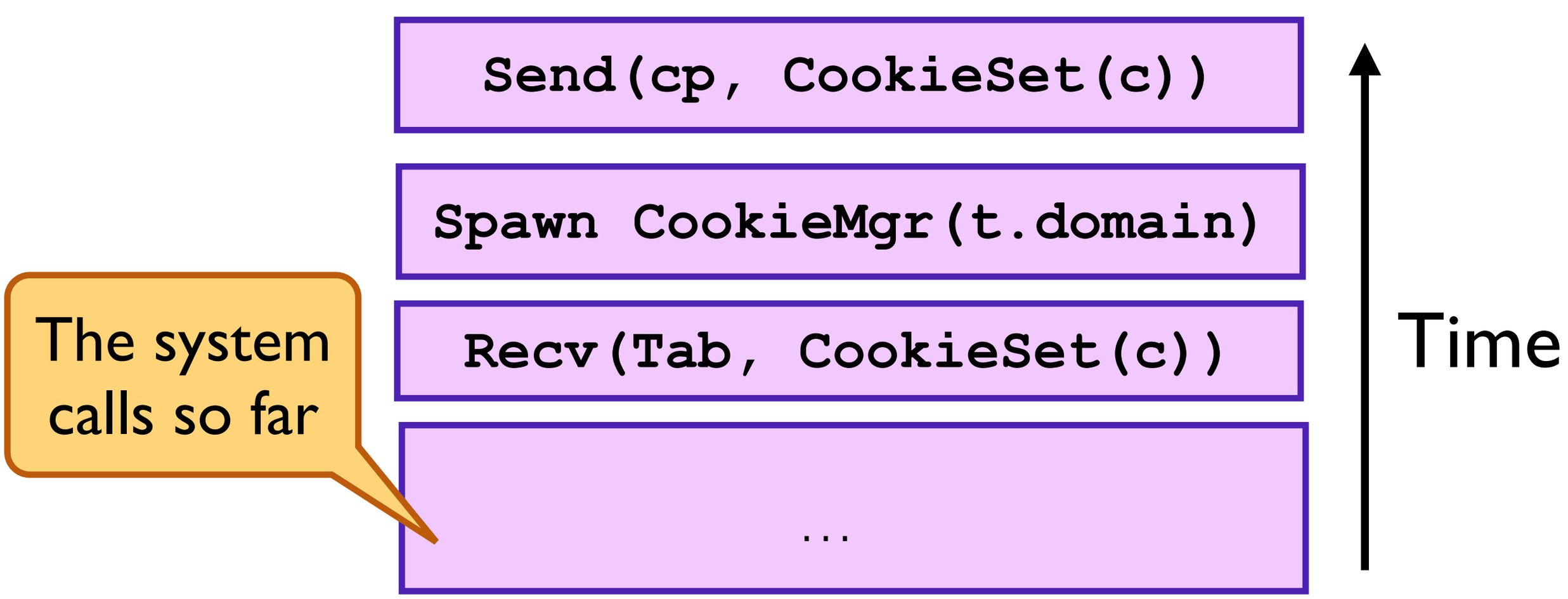
Specify allowed behaviors wrt sequence of system calls



Properties

```
When [Tab t] sends CookieSet(c):
  cp <- find CookieMgr(t.domain)
  send(cp, CookieSet(c))
```

Specify allowed behaviors wrt sequence of system calls



Example: Web browser kernel

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
  When [Tab t] sends CookieSet(c):
    cp <- find CookieMgr(t.domain)
    send(cp, CookieSet(c))

  When [Tab t] sends CookieGet(c):
    ...
```

Example: Web browser kernel

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
  When [Tab t] sends CookieSet(c):
    cp <- find CookieMgr(t.domain)
    send(cp, CookieSet(c))

  When [Tab t] sends CookieGet(c):
    ...
```

Example: Web browser kernel

Specify cookie integrity

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
  When [Tab t] sends CookieSet(c):
    cp <- find CookieMgr(t.domain)
    send(cp, CookieSet(c))

  When [Tab t] sends CookieGet(c):
    ...
```

Example: Web browser kernel

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
  When [Tab t] sends CookieSet(c):
    cp <- find CookieMgr(t.domain)
    send(cp, CookieSet(c))

  When [Tab t] sends CookieGet(c):
    ...
```

Example: Web browser kernel

```
forall d c,
```

For any domain **d** and cookie **c**

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
  When [Tab t] sends CookieSet(c):
    cp <- find CookieMgr(t.domain)
    send(cp, CookieSet(c))

  When [Tab t] sends CookieGet(c):
    ...
```

Example: Web browser kernel

```
forall d c,

   [Send(CookieMgr(d), CookieSet(c))]
```

The kernel sends the cookie manager for domain **d** a cookie **c**

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
      cp <- find CookieMgr(t.domain)
      send(cp, CookieSet(c))

   When [Tab t] sends CookieGet(c):
      ...
```

Example: Web browser kernel

```
forall d c,

   Enables
    [Send(CookieMgr(d), CookieSet(c))]
```

Only if

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
      cp <- find CookieMgr(t.domain)
      send(cp, CookieSet(c))

   When [Tab t] sends CookieGet(c):
      ...
```

Example: Web browser kernel

```
forall d c,
   [Recv(Tab(d), CookieSet(c))]
   Enables
   [Send(CookieMgr(d), CookieSet(c))]
```

```
Compon
Messag                                    ...

Handlers:
   When [Tab t] sends CookieSet(c):
      cp <- find CookieMgr(t.domain)
      send(cp, CookieSet(c))

   When [Tab t] sends CookieGet(c):
      ...
```

The kernel already received a cookie **c** from a tab of domain **d**

Example: Web browser kernel

```
forall d c,
  [Recv(Tab(d), CookieSet(c))]
  Enables
  [Send(CookieMgr(d), CookieSet(c))]
```

```
C
M                                                  ...

H
  When [Tab t] sends CookieSet(c):
    cp <- find CookieMgr(t.domain)
    send(cp, CookieSet(c))

  When [Tab t] sends CookieGet(c):
    ...
```

A Enables B

iff

every sys call **B** is preceded by sys call **A**

Example: Web browser kernel

```
forall d c,
   [Recv(Tab(d), CookieSet(c))]
   Enables
   [Send(CookieMgr(d), CookieSet(c))]
```

```
Components = Tab | CookieMgr | ...
Messages = CookieSet | CookieGet | ...

Handlers:
   When [Tab t] sends CookieSet(c):
      cp <- find CookieMgr(t.domain)
      send(cp, CookieSet(c))

   When [Tab t] sends CookieGet(c):
      ...
```

Example: Web browser kernel

Example: Web browser kernel

REFLEX Benefits:

Example: Web browser kernel

REFLEX Benefits:

Proofs fully automated

Example: Web browser kernel

REFLEX Benefits:

Proofs fully automated

No lemmas

Example: Web browser kernel

REFLEX Benefits:

Proofs fully automated

No lemmas

No invariants

Example: Web browser kernel

REFLEX Benefits:

Proofs fully automated

No lemmas

No invariants

No manual proofs

Proof Automation

Proof Automation

Prove kernel code satisfies properties

Proof Automation

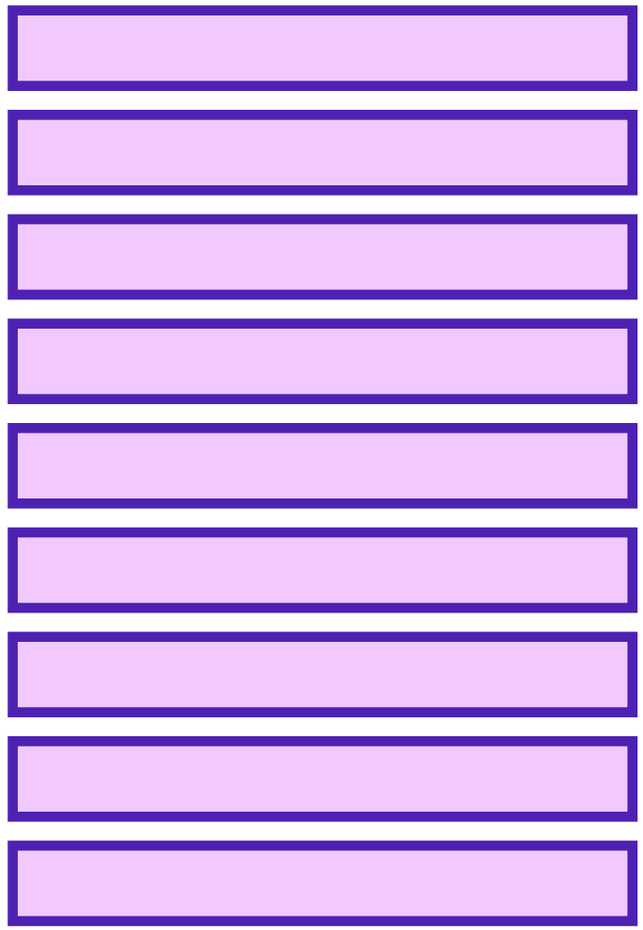
Prove kernel code satisfies properties

by induction on sys call sequences kernel can produce

Proof Automation

Prove kernel code satisfies properties

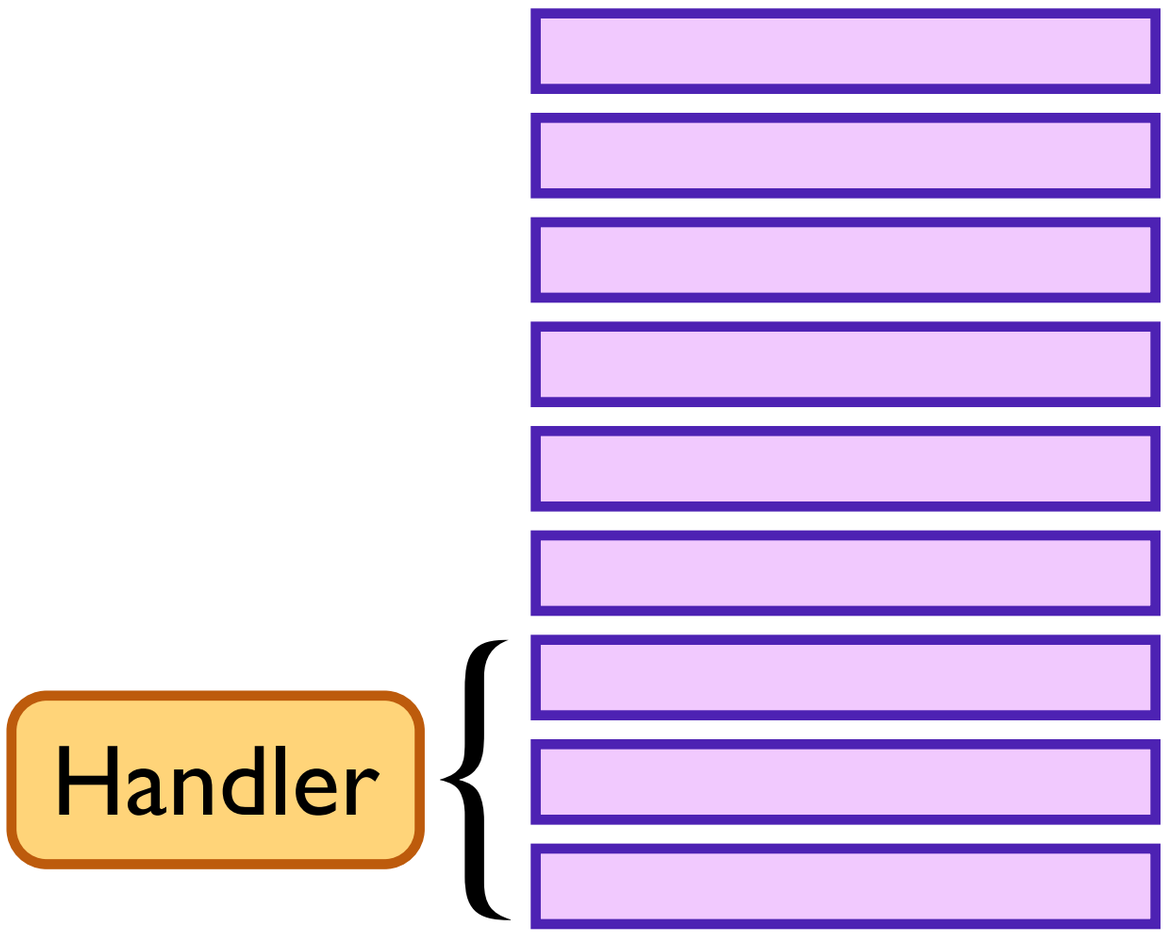
by induction on sys call sequences kernel can produce



Proof Automation

Prove kernel code satisfies properties

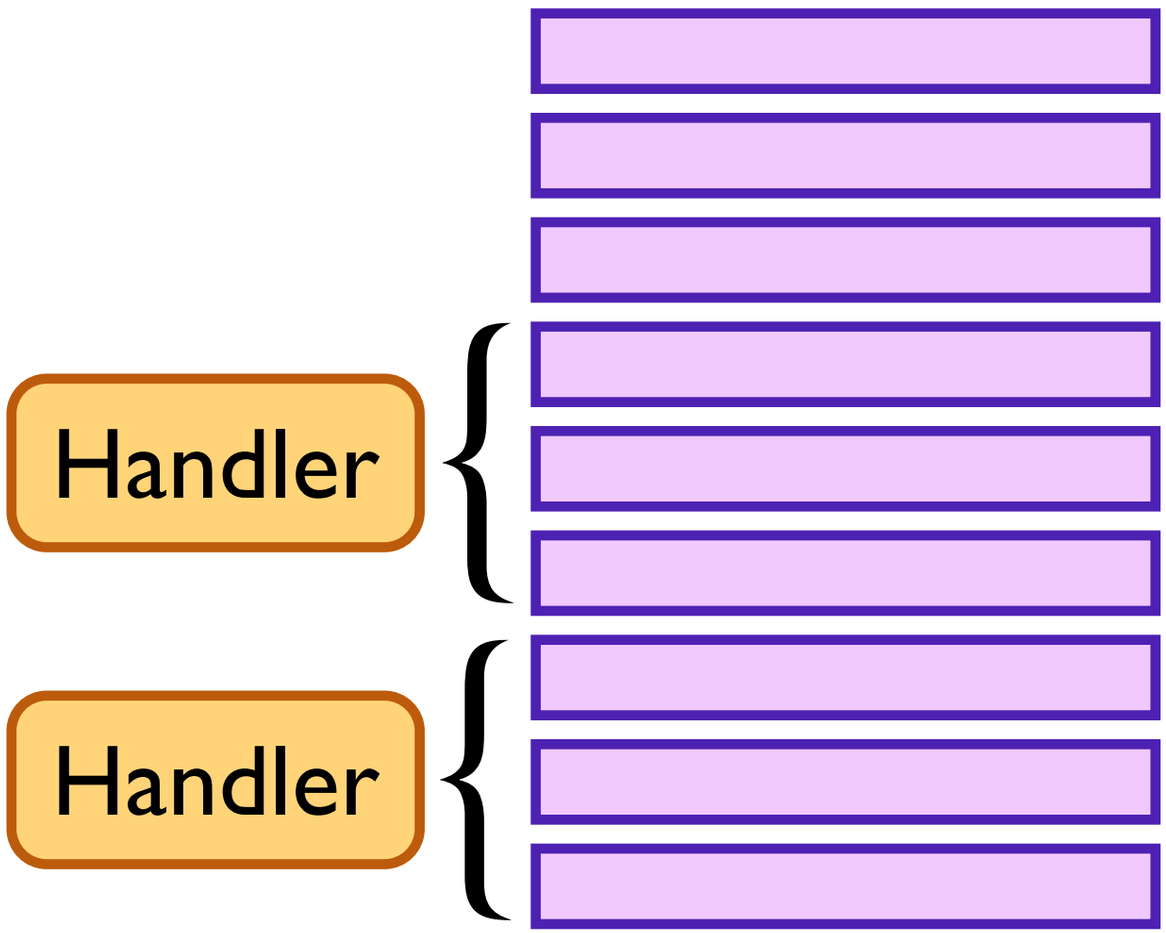
by induction on sys call sequences kernel can produce



Proof Automation

Prove kernel code satisfies properties

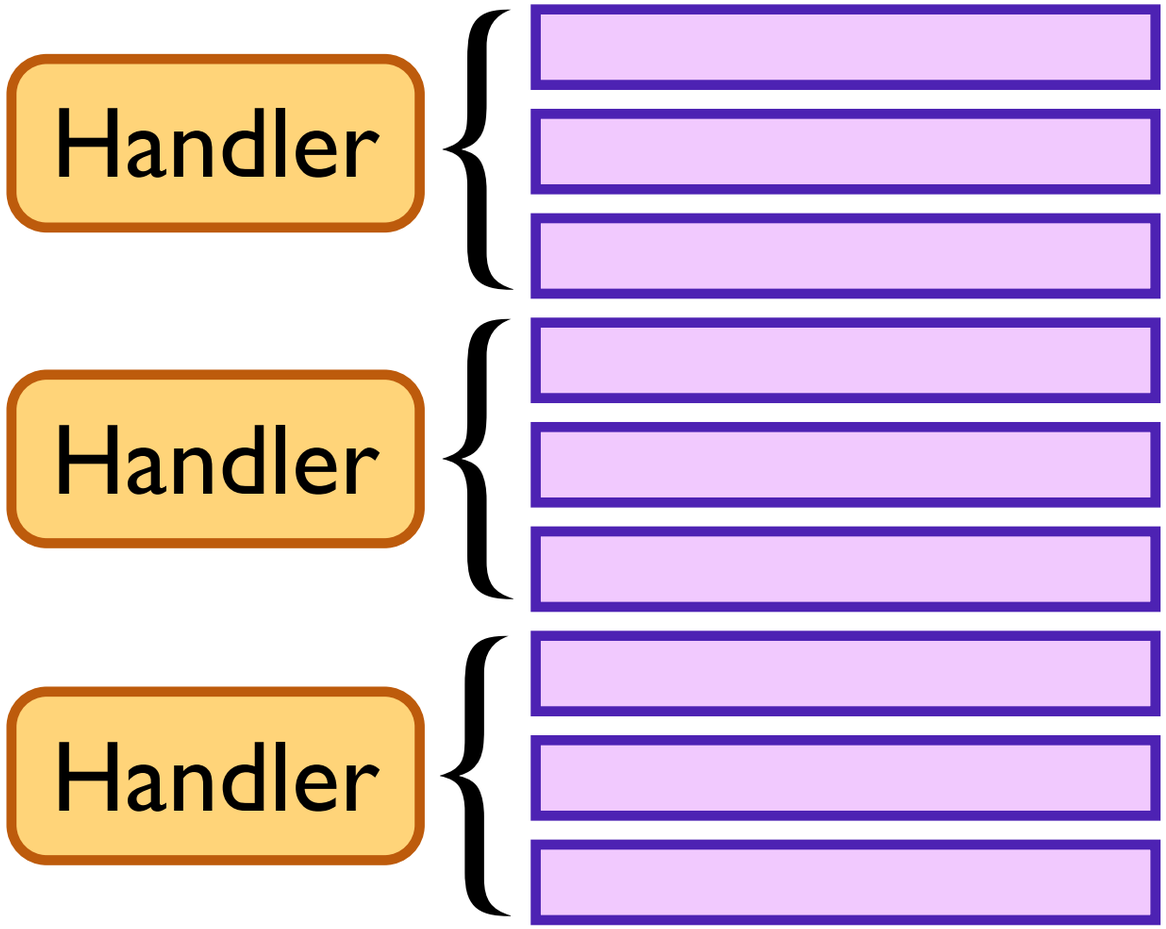
by induction on sys call sequences kernel can produce



Proof Automation

Prove kernel code satisfies properties

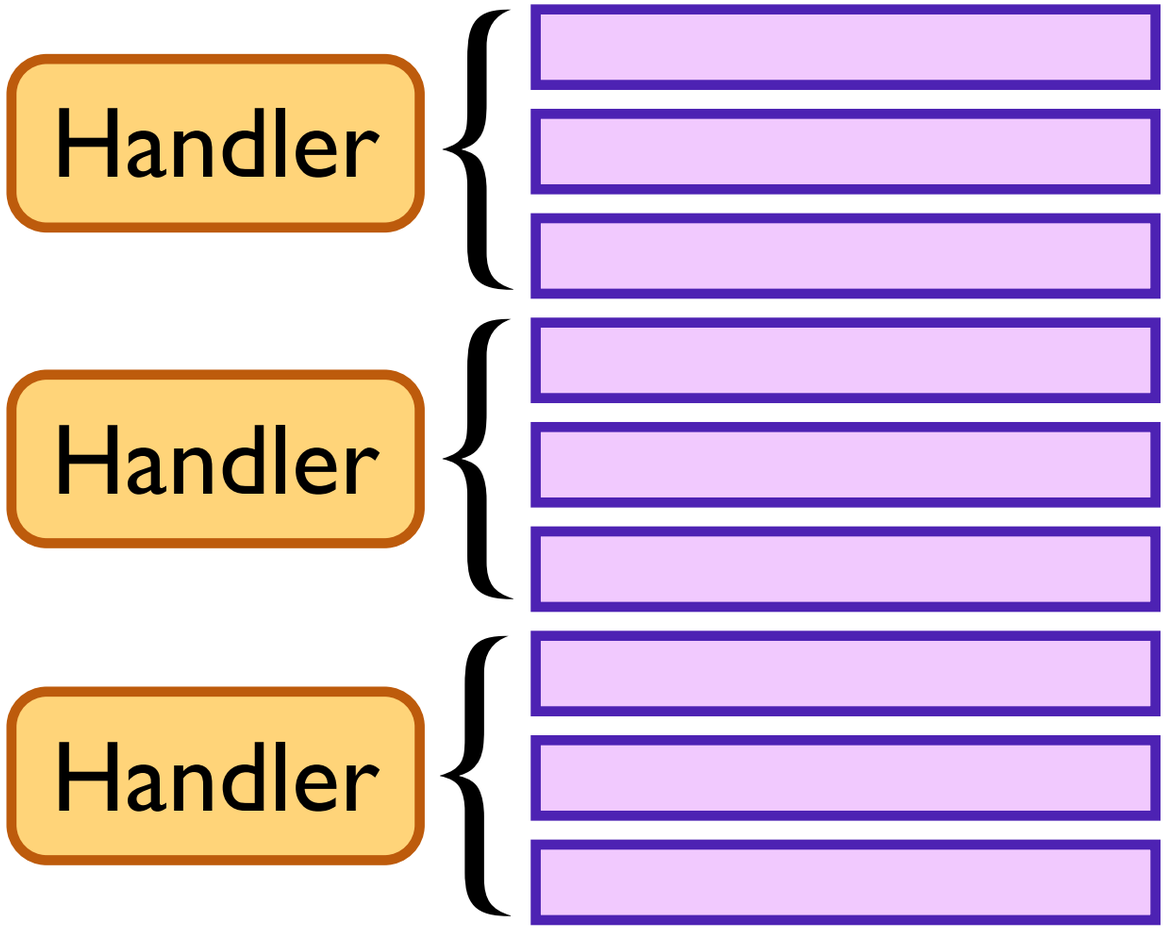
by induction on sys call sequences kernel can produce



Proof Automation

Prove kernel code satisfies properties

by induction on sys call sequences kernel can produce



Handlers create structure for induction

Proof Automation

Prove kernel code satisfies properties

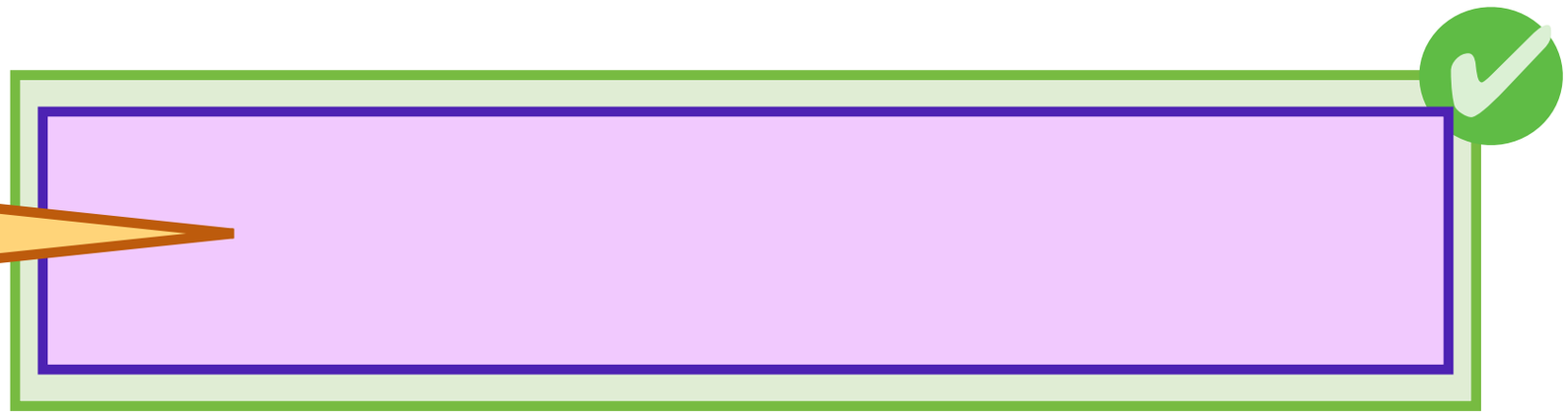
by induction on sys call sequences kernel can produce

Proof Automation

Prove kernel code satisfies properties

by induction on sys call sequences kernel can produce

Induction hypothesis: property holds up to this point

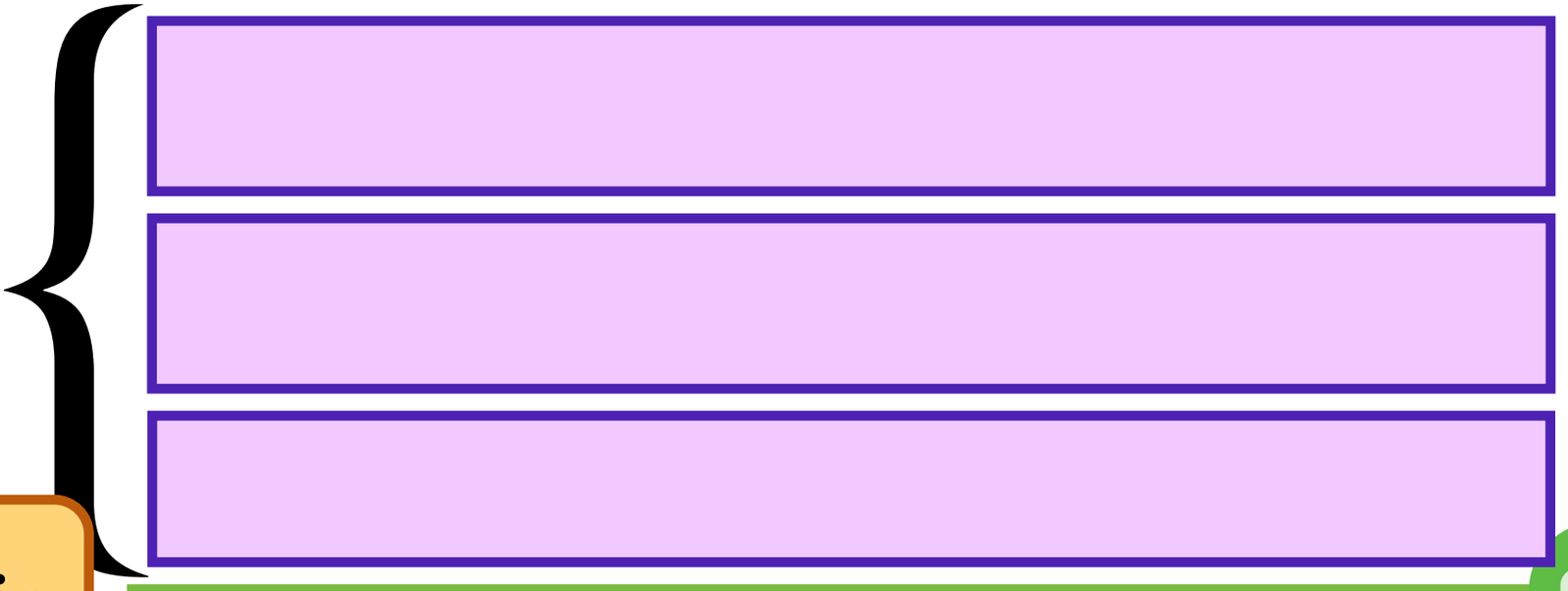


Proof Automation

Prove kernel code satisfies properties

by induction on sys call sequences kernel can produce

Run a single handler



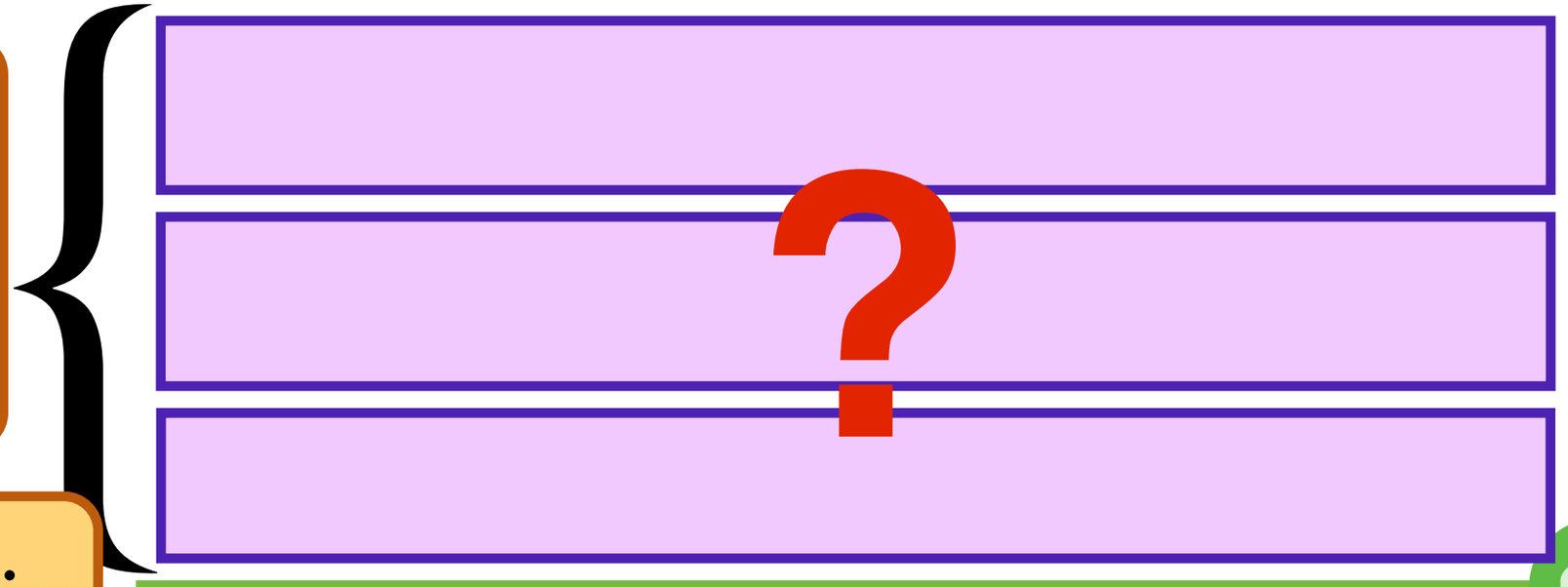
Induction hypothesis: property holds up to this point

Proof Automation

Prove kernel code satisfies properties

by induction on sys call sequences kernel can produce

Proof obligation: does property still hold?



Induction hypothesis: property holds up to this point

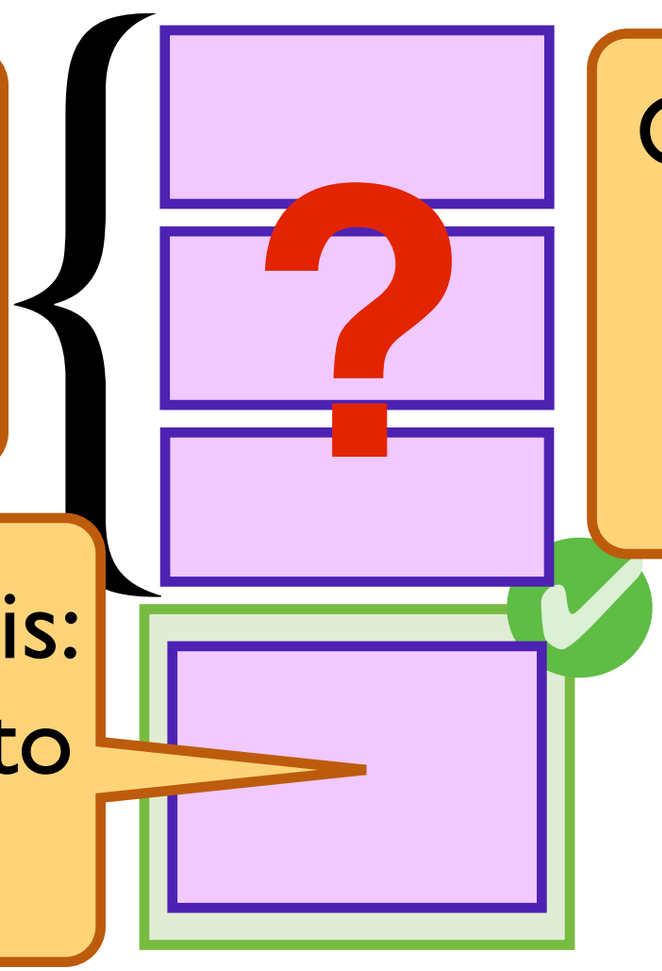
Proof Automation

Prove kernel code satisfies properties

by induction on sys call sequences kernel can produce

Proof obligation: does property still hold?

Induction hypothesis: property holds up to this point



Case analysis on handler

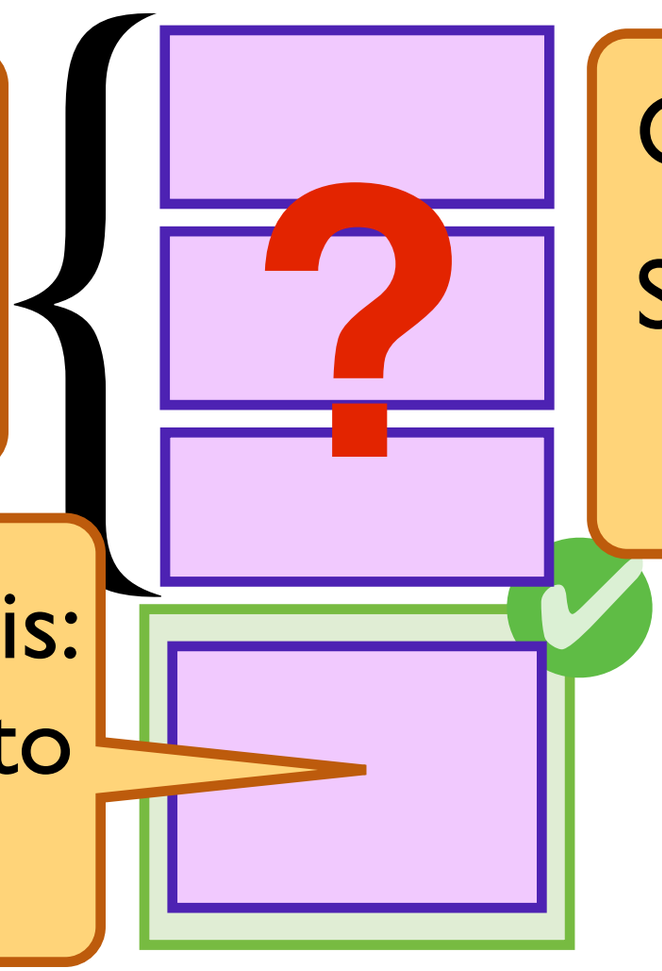
Proof Automation

Prove kernel code satisfies properties

by induction on sys call sequences kernel can produce

Proof obligation: does property still hold?

Induction hypothesis: property holds up to this point



Case analysis on handler

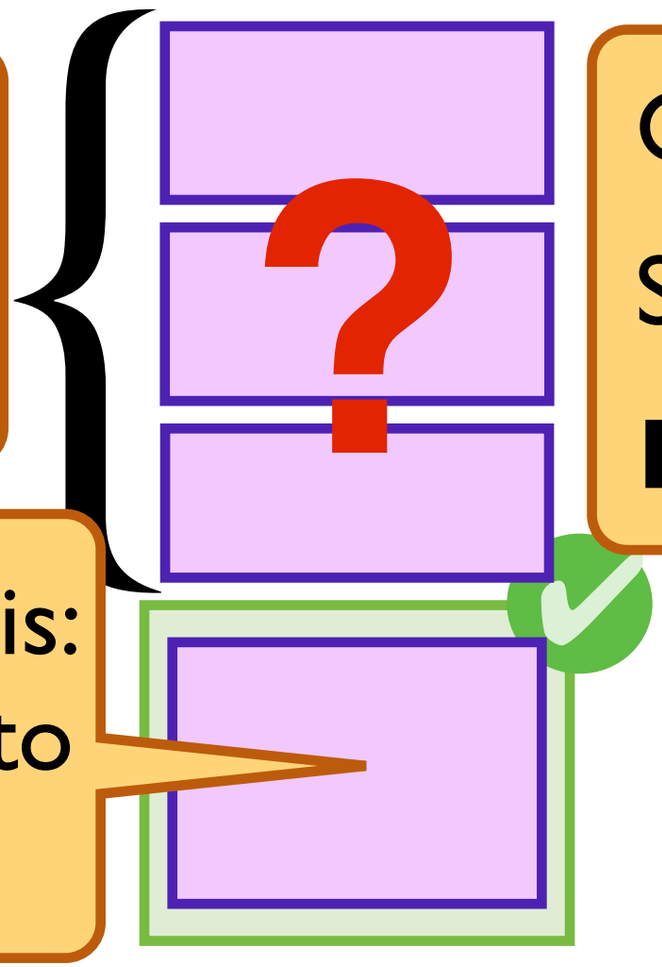
Symbolically run all paths

Proof Automation

Prove kernel code satisfies properties

by induction on sys call sequences kernel can produce

Proof obligation: does property still hold?



Case analysis on handler

Symbolically run all paths

Prove automatically

Induction hypothesis: property holds up to this point

Proof Automation

Proof Automation

Single domain insights:

Proof Automation

Single domain insights:

Small number of carefully designed property primitives

Proof Automation

Single domain insights:

Small number of carefully designed property primitives

Loop free handlers allowed symbolic eval of all paths

Proof Automation

Single domain insights:

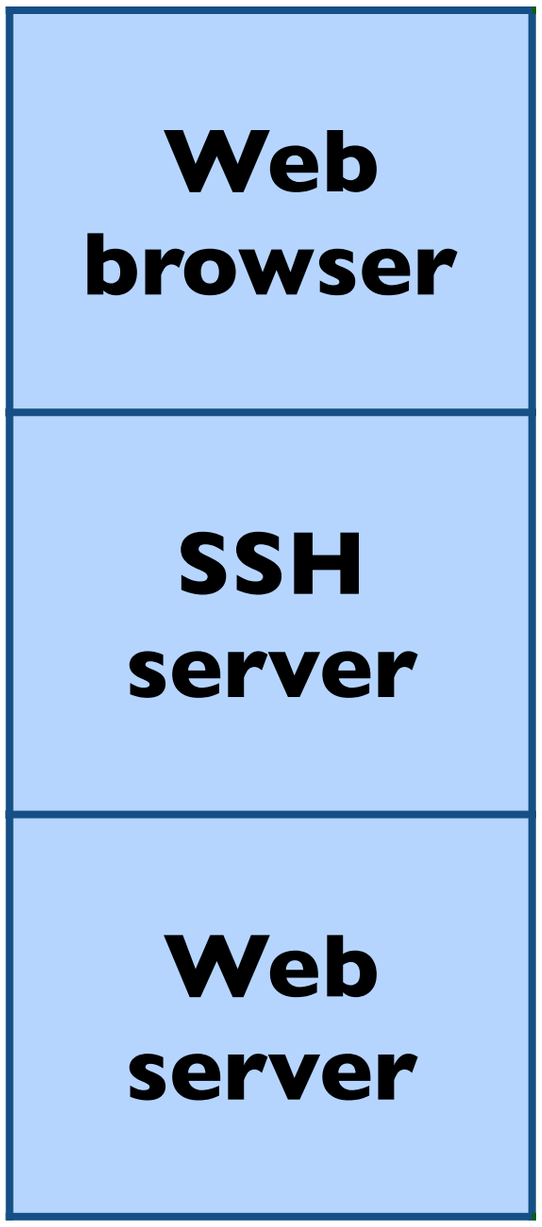
Small number of carefully designed property primitives

Loop free handlers allowed symbolic eval of all paths

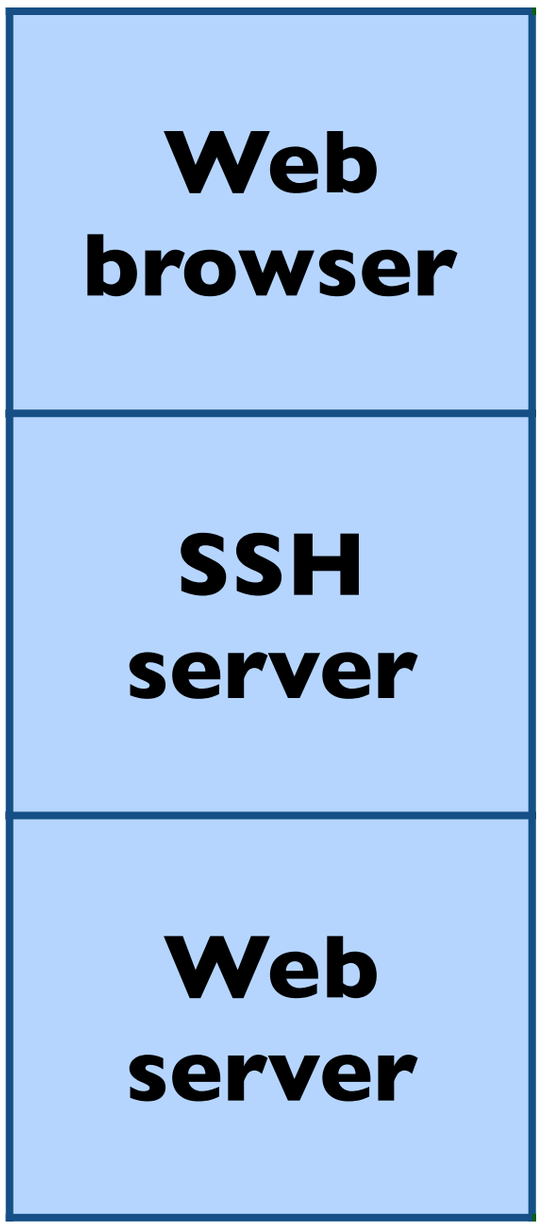
Domain-specific heuristics for non-local reasoning

Evaluation

Evaluation



Evaluation



Auto verified 33 properties (80% in < 2 minutes)

Evaluation

Web browser	**Domains do not interfere, Cookie integrity, …**
SSH server	**No PTY access before authentication, At most 3 authentication attempts, …**
Web server	**Clients only spawned after successful login, File requests guarded by access control, …**

Auto verified 33 properties (80% in < 2 minutes)

Evaluation

Web browser	**Domains do not interfere, Cookie integrity, …**
SSH server	**No PTY access before authentication, At most 3 authentication attempts, …**
	Only spawned after successful login, File requests guarded by access control, …

Able to automate proofs of non-interference

Auto verified 33 properties (80% in < 2 minutes)

Evaluation

Web browser	**Domains do not interfere, Cookie integrity, …**
SSH server	**No PTY access before authentication, At most 3 authentication attempts, …**
Web server	**Clients only spawned after successful login, File requests guarded by access control, …**

Auto verified 33 properties (80% in < 2 minutes)

Evaluation

Web browser	Domains do Cookie ir
SSH server	No PTY access before authentication, At most 3 authentication attempts, …
Web server	Clients only spawned after successful login, File requests guarded by access control, …

Able to automate proofs of non-local properties

Auto verified 33 properties (80% in < 2 minutes)

Development Effort: Framework

Development Effort: Framework

Reflex :

7500 lines of Coq

Development Effort: Framework

Reflex :

7500 lines of Coq

Web browser	**SSH server**	**Web server**

Development Effort: Framework

Reflex :

7500 lines of Coq

Web browser	**SSH server**	**Web server**

Quark Web browser :

5500 lines of Coq

Development Effort: Framework

Reflex :

7500 lines of Coq

Web browser	**SSH server**	**Web server**

Quark Web browser :

5500 lines of Coq

Single reactive system

Development Effort: Framework

Reflex : Many reactive systems

7500 lines of Coq

Web browser	**SSH server**	**Web server**

Quark Web browser :

5500 lines of Coq

Single reactive system

Development Effort: Systems

Development Effort: Systems

Benchmark	*Language*	*Lines of Code/Spec*	
Browser	REFLEX	81	37
	C++, Python	970,240	-
SSH server	REFLEX	64	22
	C, Python	89,567	-
Web server	REFLEX	56	29
	Python	386	-

Development Effort: Systems

Benchmark	*Language*	*Lines of Code/Spec*	
Browser	REFLEX	81	37
	C++, Python	970,240	
SSH server	REFLEX	64	22
	C, Python	89,567	-
Web server	REFLEX	56	29
	Python	386	-

Webkit

Development Effort: Systems

Benchmark	*Language*	*Lines of Code/Spec*	
Browser	REFLEX	81	37
	C++, Python	970,240	
SSH server	REFLEX	64	22
	C, Python	89,567	
Web server	REFLEX	56	29
	Python	386	-

Webkit

OpenSSH

Limitations

Limitations

Expressiveness:

Incompleteness:

Limitations

Expressiveness:

Strict subset of temporal logic + non-interference

Incompleteness:

Limitations

Expressiveness:

- **Strict subset of temporal logic + non-interference**
- **Loop free handlers**

Incompleteness:

Limitations

Expressiveness:

- **Strict subset of temporal logic + non-interference**
- **Loop free handlers**
- **No user-defined unbounded data structures**

Incompleteness:

Limitations

Expressiveness:

- **Strict subset of temporal logic + non-interference**
- **Loop free handlers**
- **No user-defined unbounded data structures**

Incompleteness:

- **Unable to infer some inductive invariants**

Limitations

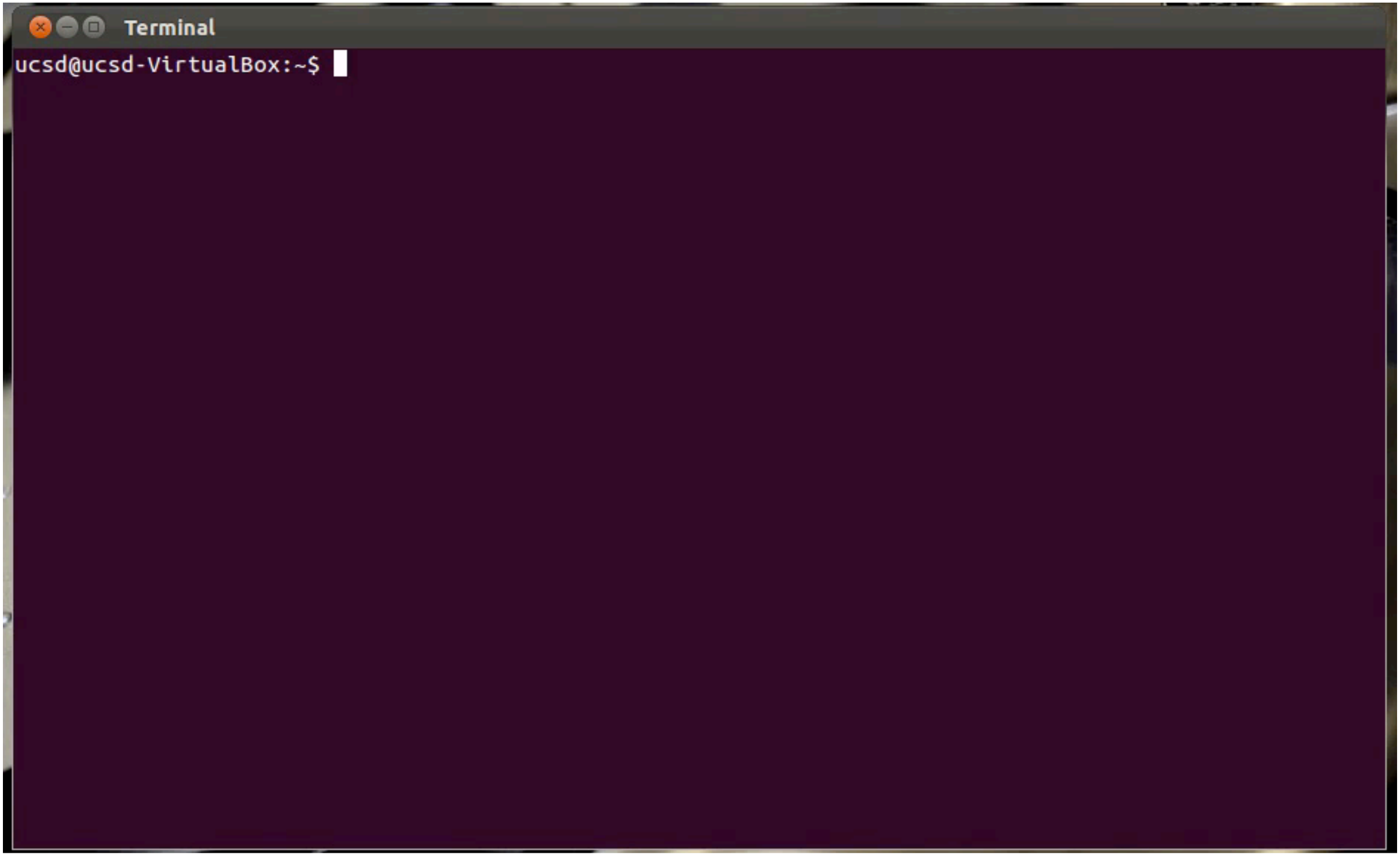
Expressiveness:

- **Strict subset of temporal logic + non-interference**
- **Loop free handlers**
- **No user-defined unbounded data structures**

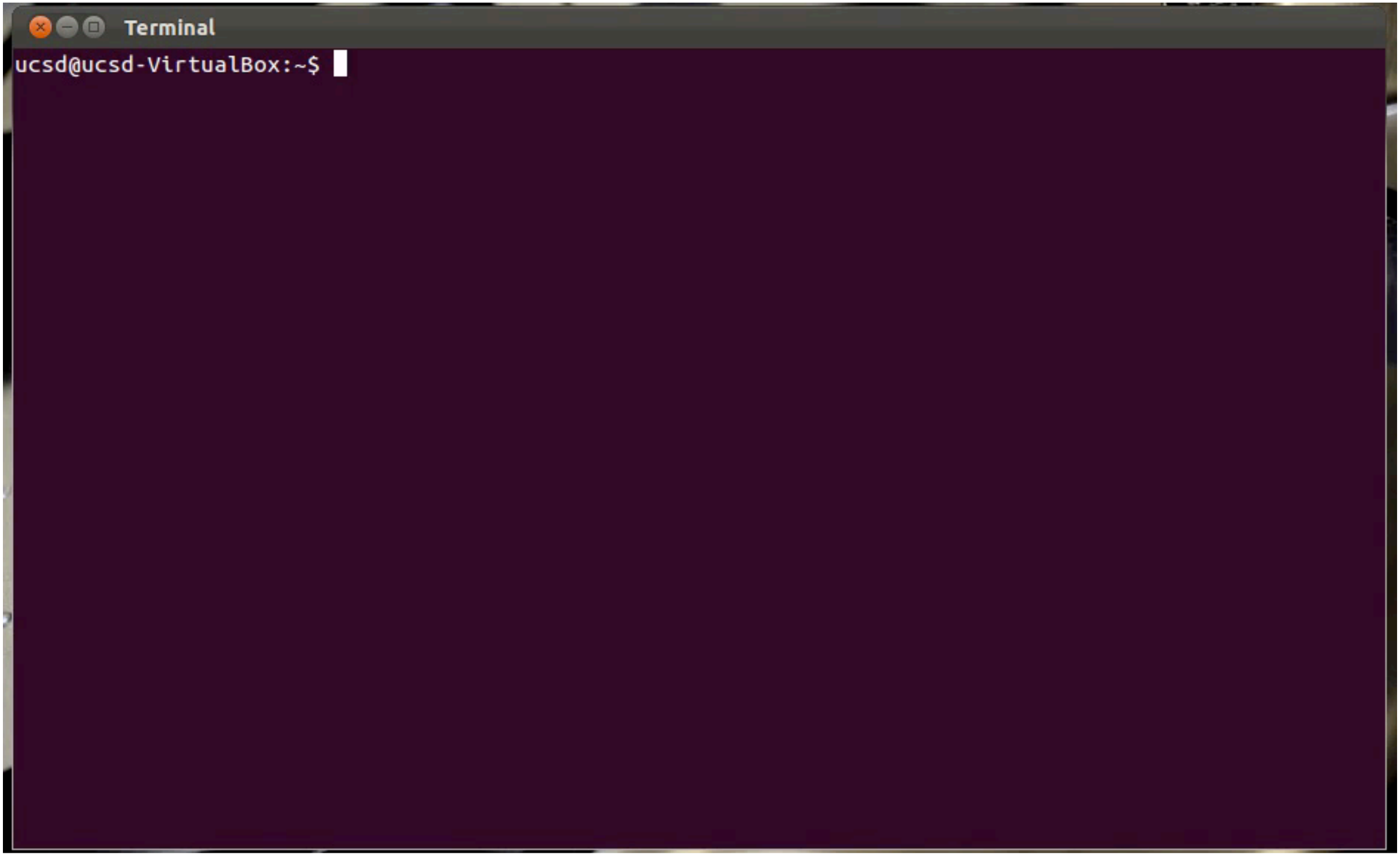
Incompleteness:

- **Unable to infer some inductive invariants**
- **Low level incompleteness in automation tactics**

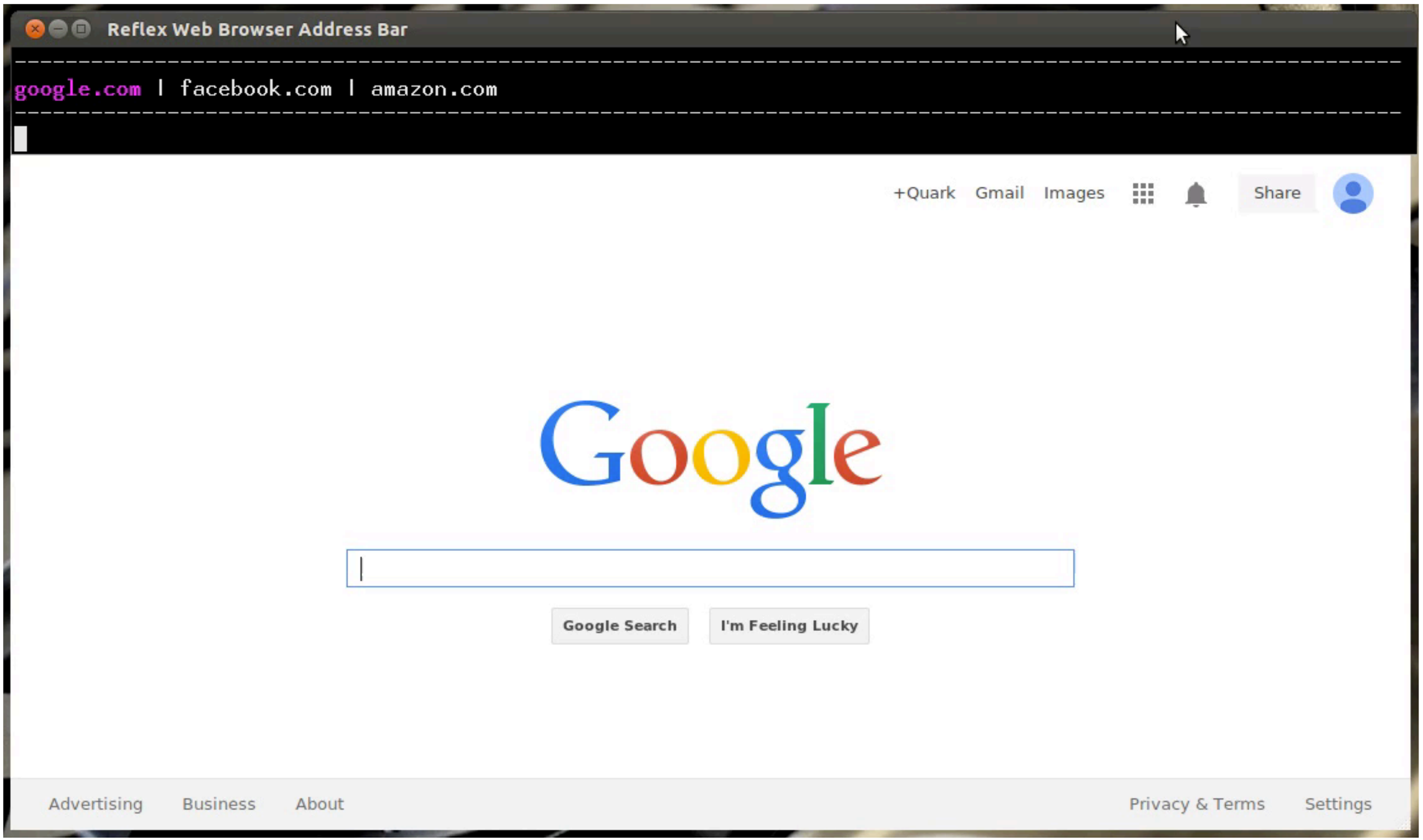
SSH Server Kernel in REFLEX



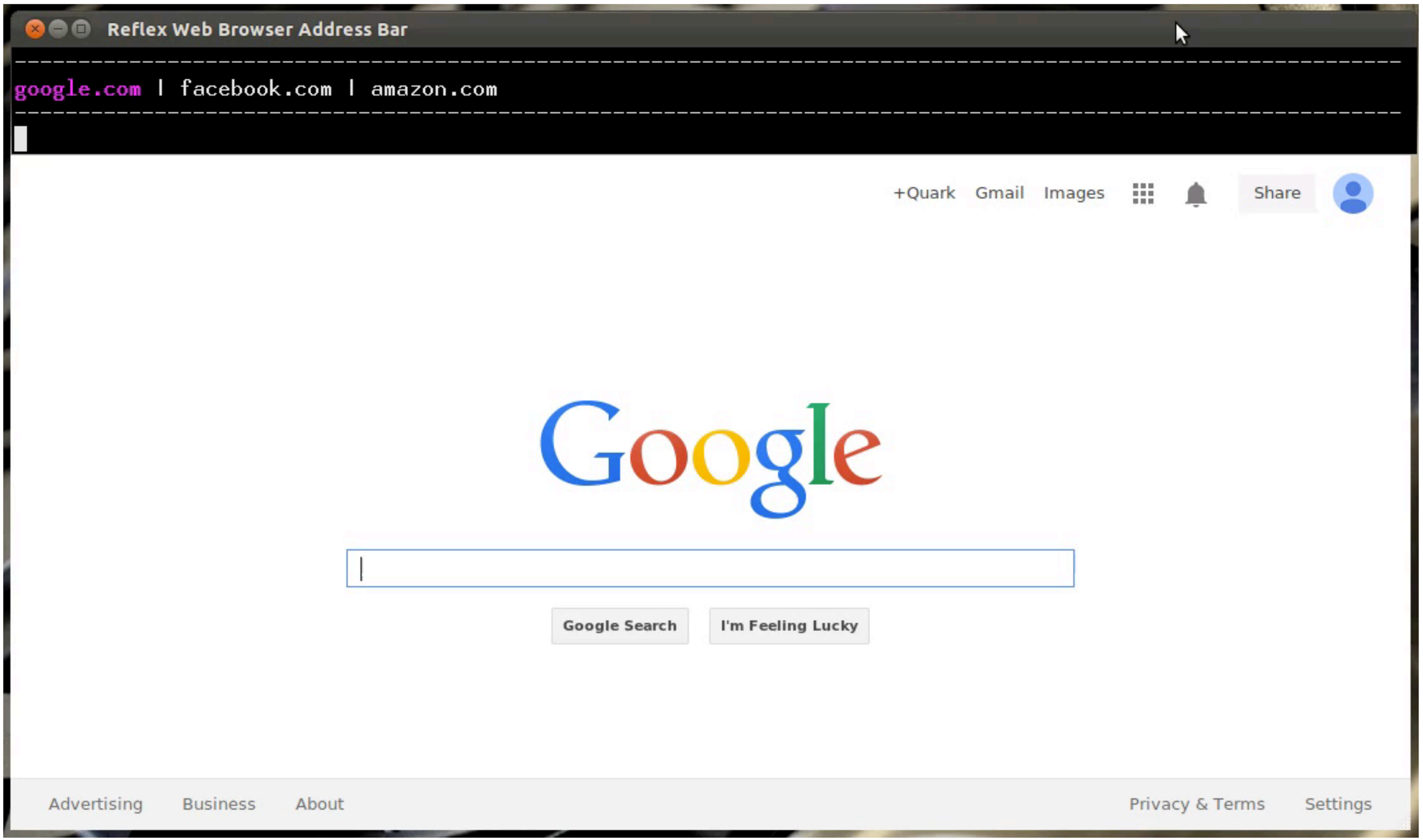
SSH Server Kernel in REFLEX



Web Browser Kernel in REFLEX



Web Browser Kernel in REFLEX



Conclusion

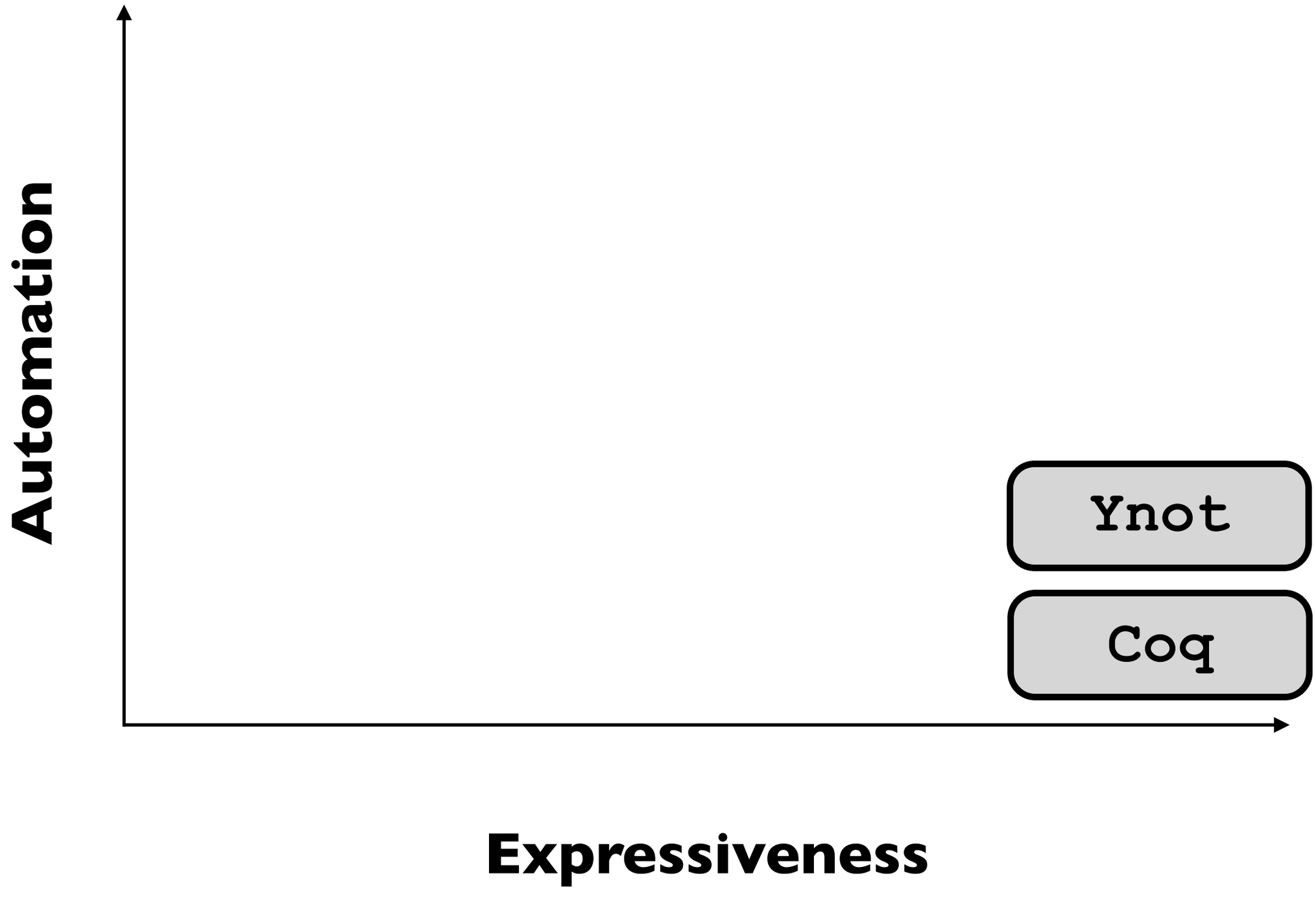
Proof assistant based verification

Conclusion

Proof assistant based verification

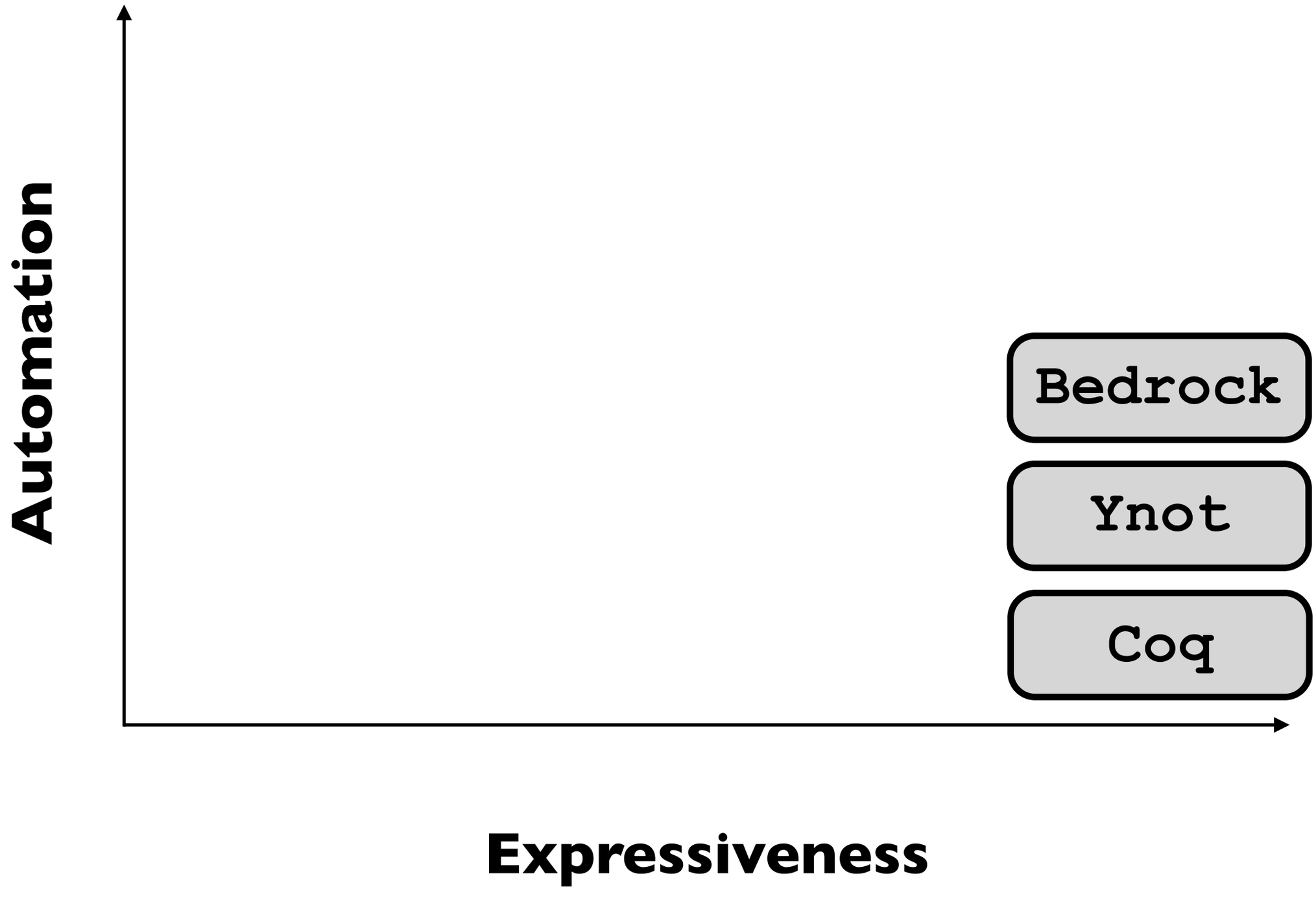
Conclusion

Proof assistant based verification



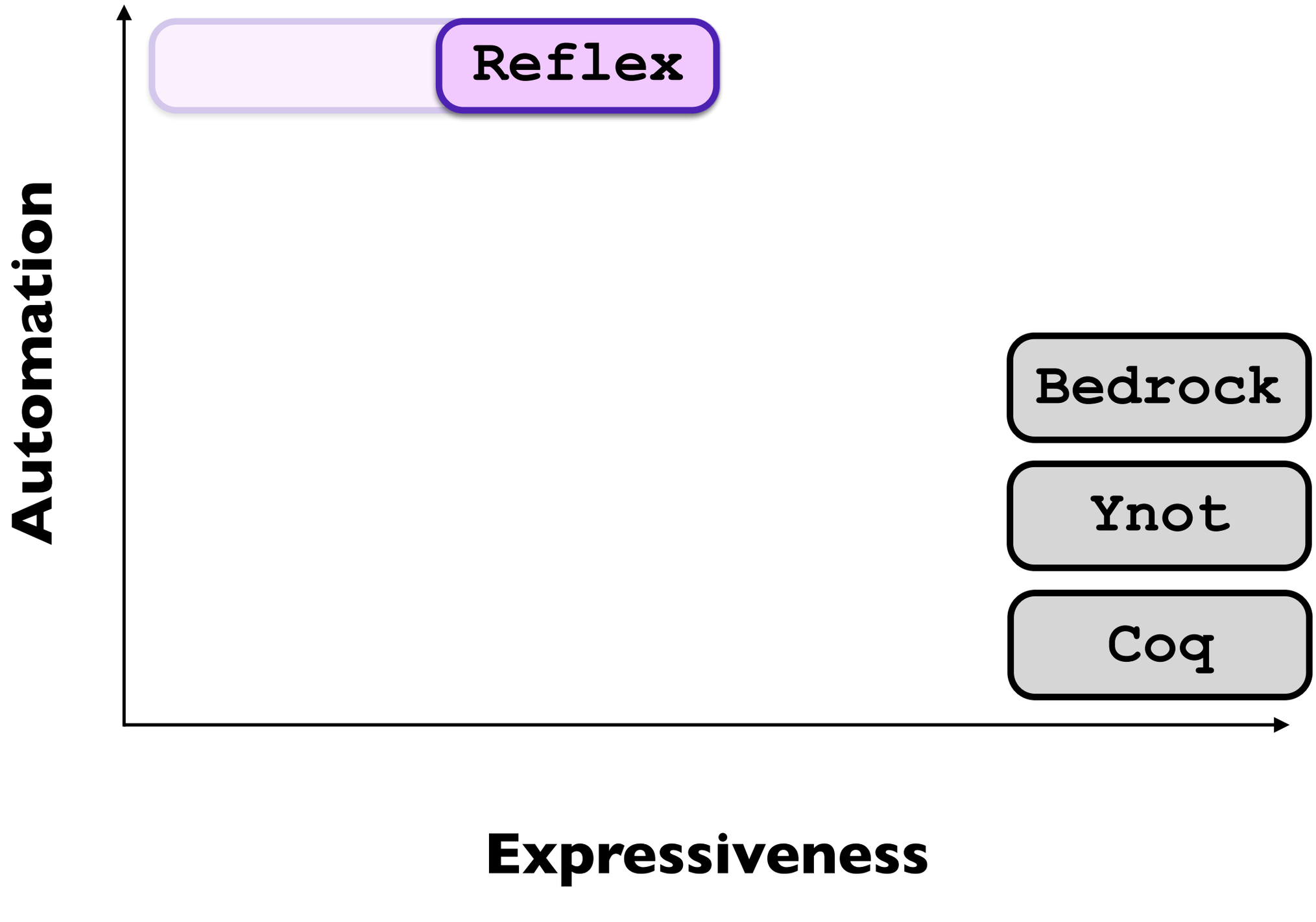
Conclusion

Proof assistant based verification



Conclusion

Proof assistant based verification

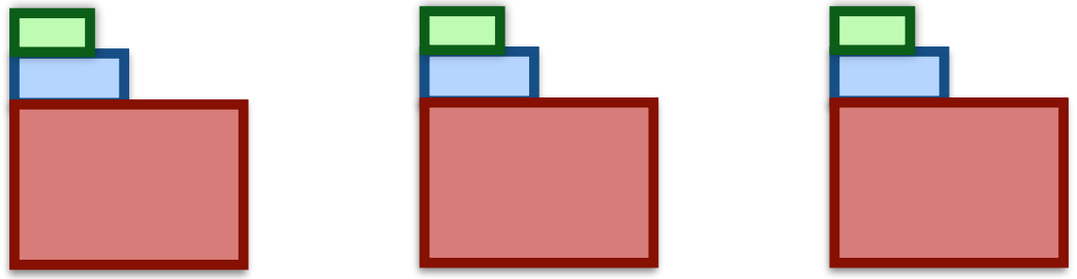


Conclusion

`Reflex`

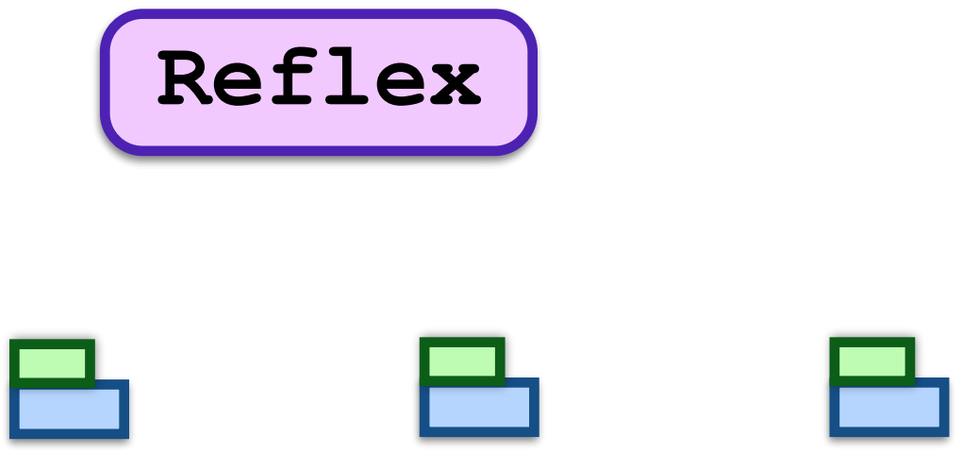
Conclusion

`Reflex`



DSL expressive enough for entire domain

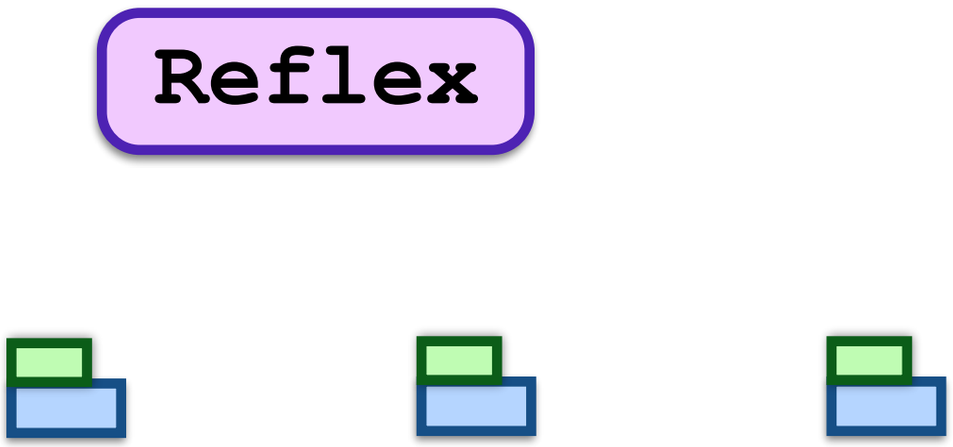
Conclusion



DSL expressive enough for entire domain

Automation eliminates manual proof burden

Conclusion

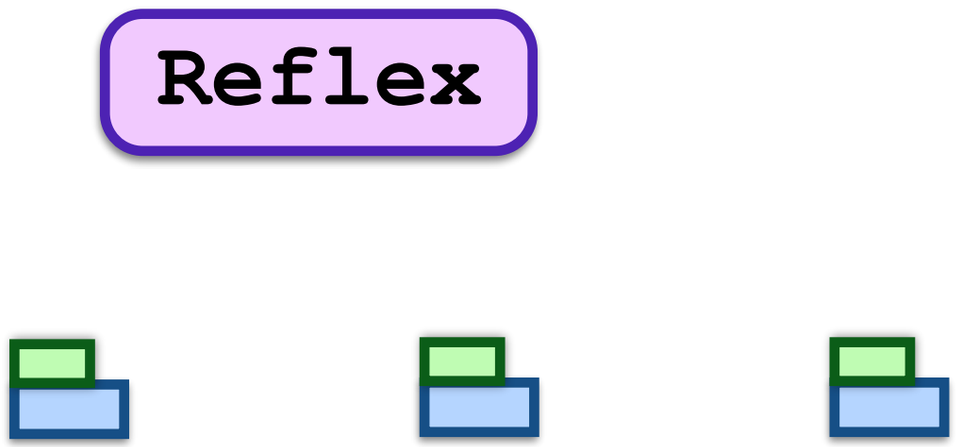


DSL expressive enough for entire domain

Automation eliminates manual proof burden

http://goto.ucsd.edu/reflex/

Thank You!



DSL expressive enough for entire domain

Automation eliminates manual proof burden

http://goto.ucsd.edu/reflex/