S²E

A Platform for In-Vivo Multi-Path Analysis of Software Systems

Vitaly Chipounov, Volodymyr Kuznetsov,
George Candea

School of Computer & Communication Sciences

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Bug Finding

Bug Finding

```
int main(argc, argv)
{
  if (argc == 2) {
    printf("%c", *argv[2]);
    return -1;
  }

  return 0;
}
```

Bug Finding

```
int main(argc, argv)
{
  if (argc == 2) {
    printf(“%c”, *argv[2]);
    return -1;
  }

  return 0;
}
```

```
$ ./prog
```

Bug Finding

```
int main(argc, argv)
{
  if (argc == 2) {
    printf("%c", *argv[2]);
    return -1;
  }

  return 0;
}
```

```
$ ./prog

$ ./prog p1
Segmentation fault
```

Bug Finding

```
int main(argc, argv)
{
  if (argc == 2) {
    printf("%c", *argv[2]);
    return -1;
  }

  return 0;
}
```

```
$ ./prog

$ ./prog p1
Segmentation fault
```

```
$ valgrind ./prog p1
Invalid read of size 1
main (prog.c:10)
```

Performance Profiling

Performance Profiling

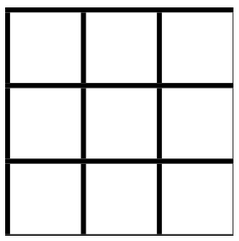
```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
    for(j = 0; j < m.h; j++)
      sum +=  m[i][j];

   return sum;
}
```

Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
    for(j = 0; j < m.h; j++)
      sum +=  m[i][j];

   return sum;
}
```

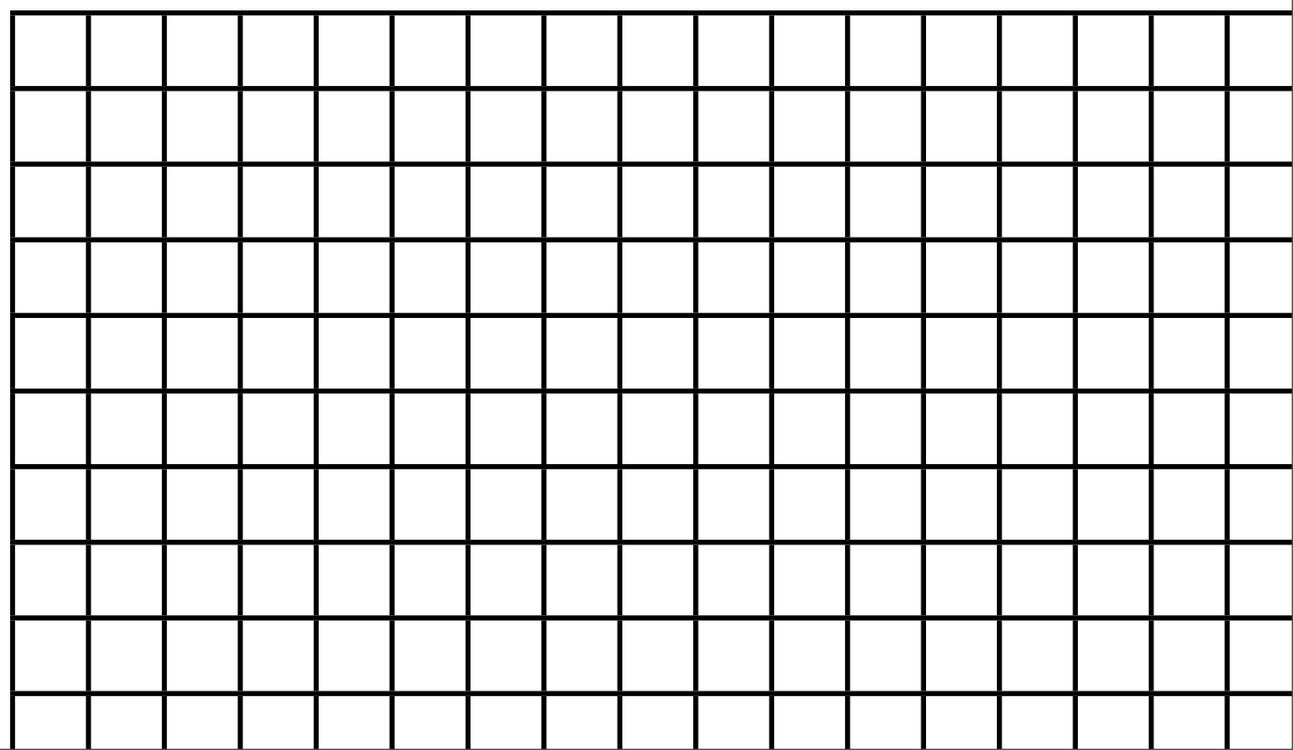
Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
    for(j = 0; j < m.h; j++)
      sum +=  m[i][j];

   return sum;
}
```



Performance Profiling

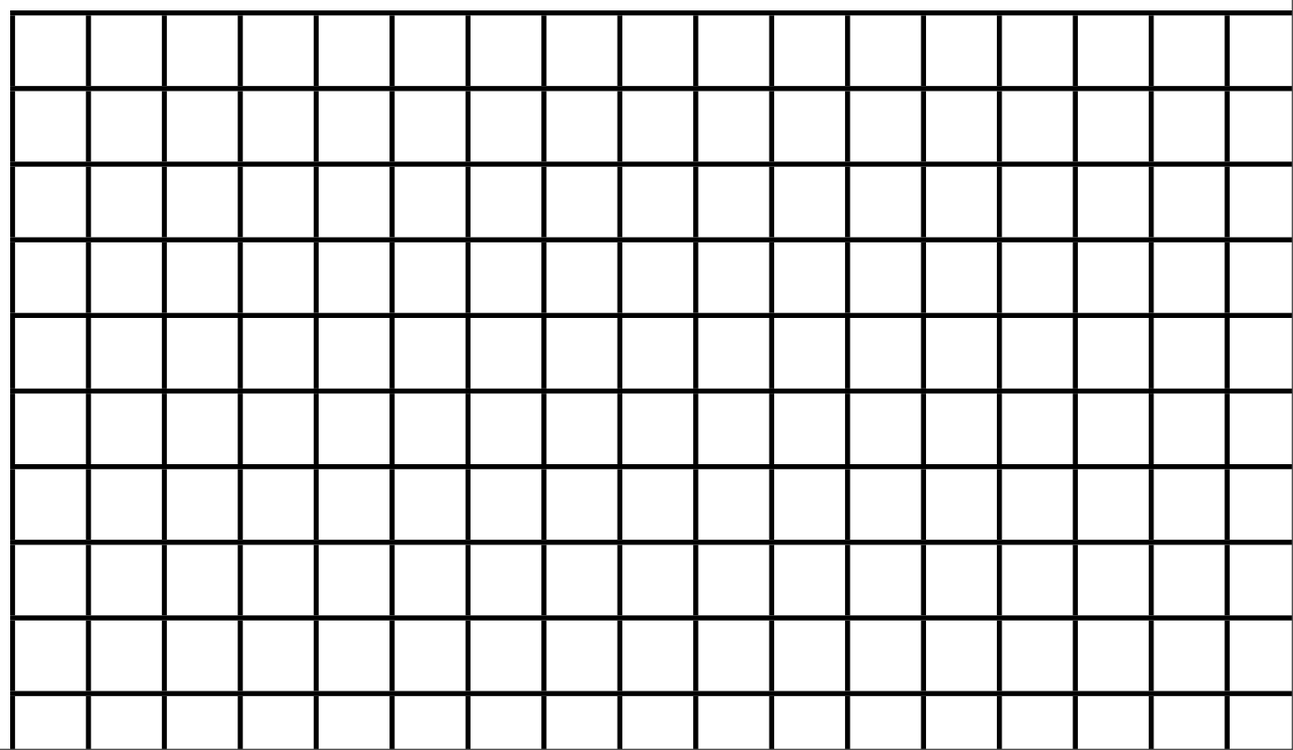
```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
    for(j = 0; j < m.h; j++)
      sum +=  m[i][j];

   return sum;
}
```



Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
    for(j = 0; j < m.h; j++)
      sum +=  m[i][j];

   return sum;
}
```

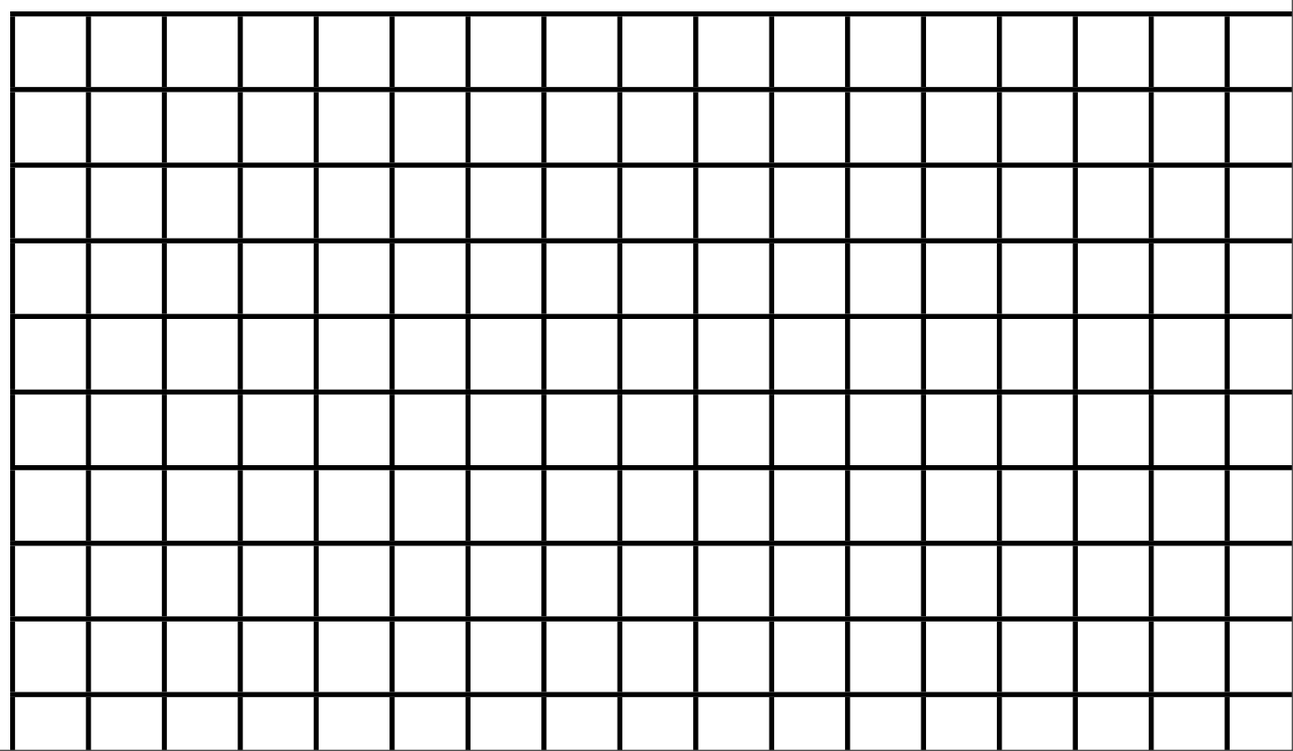
OProfile



Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
     for(j = 0; j < m.h; j++)
       sum += m[i][j];

   return sum;
}
```

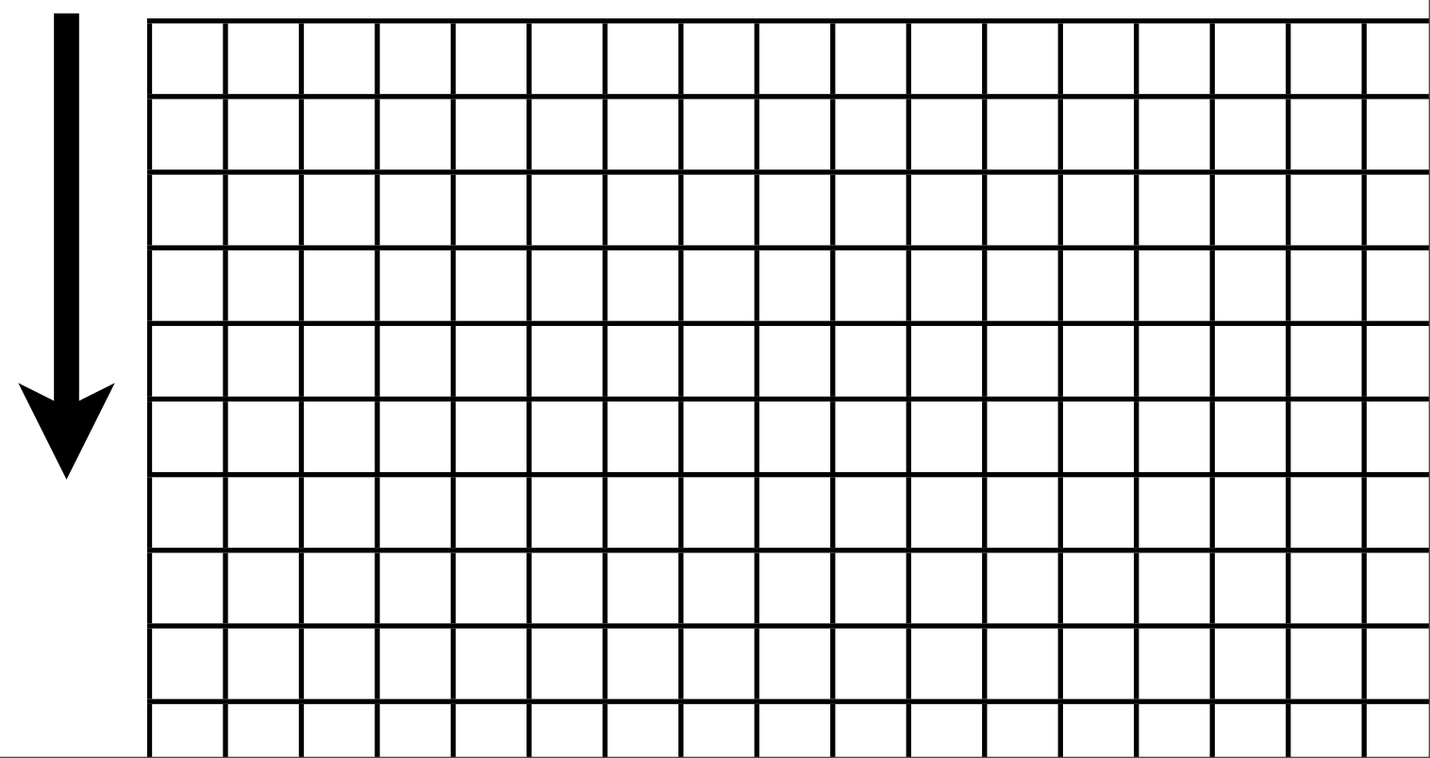
OProfile



Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
     for(j = 0; j < m.h; j++)
       sum += m[i][j];

   return sum;
}
```

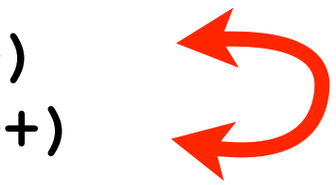
OProfile

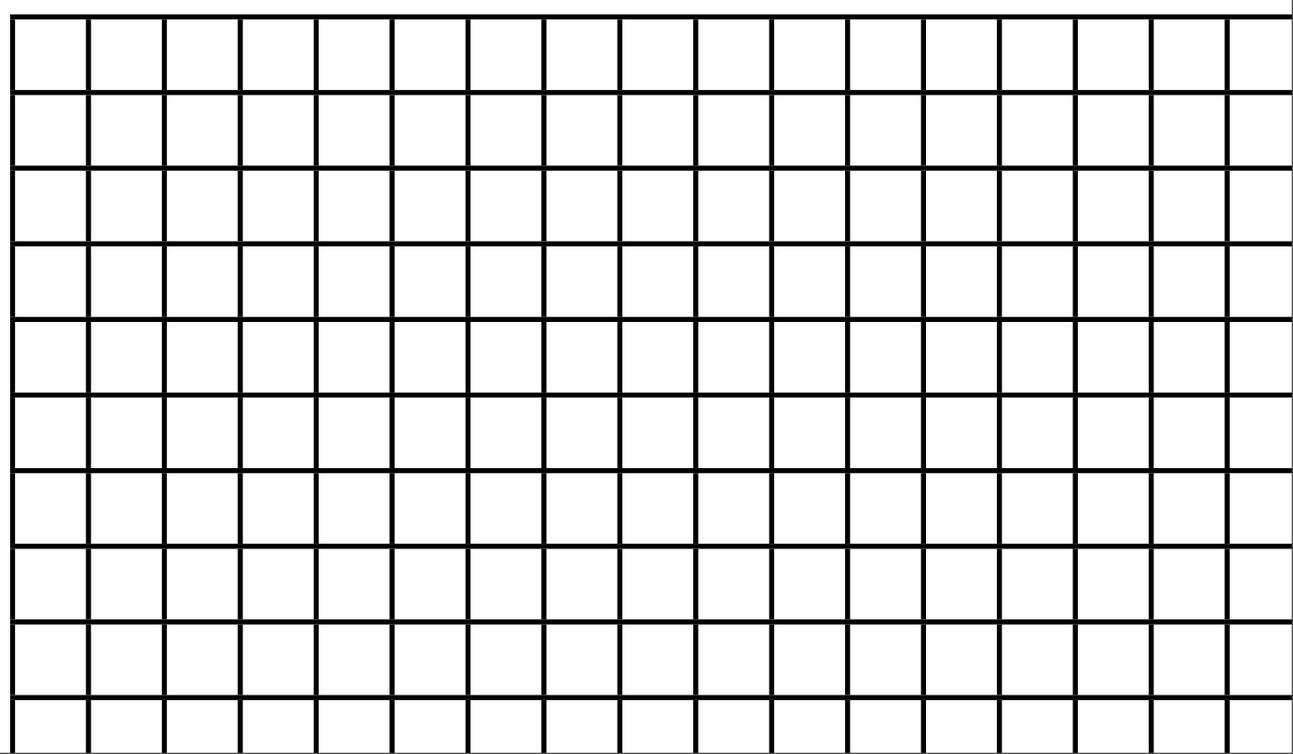


Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
     for(j = 0; j < m.h; j++)
       sum += m[i][j];

   return sum;
}
```

OProfile

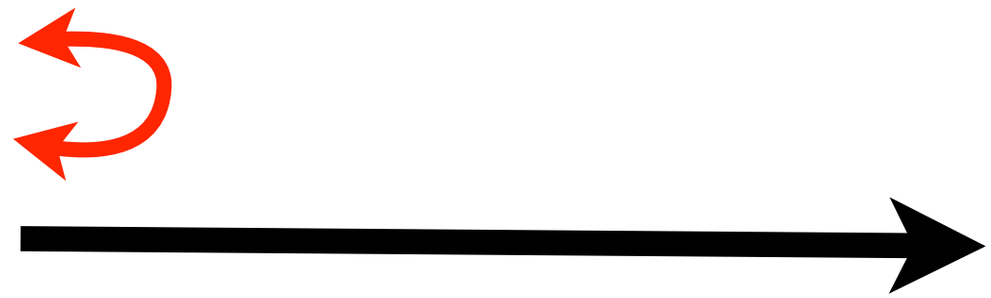


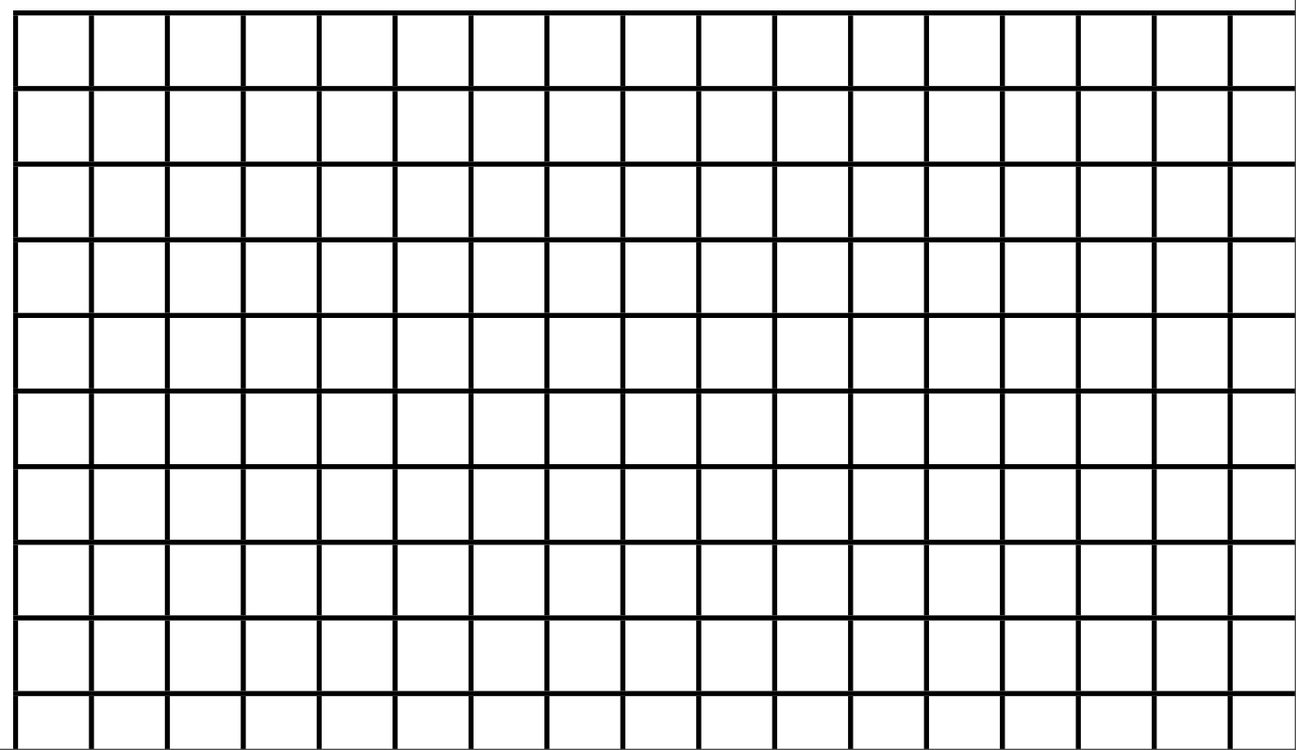


Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
     for(j = 0; j < m.h; j++)
       sum += m[i][j];

   return sum;
}
```

OProfile





Analyses

- Bug finding
- Performance profiling
- Verification/Certification
- Security analysis
- ...

Analyses

- Bug finding
- Performance profiling
- Verification/Certification
- Security analysis
- ...

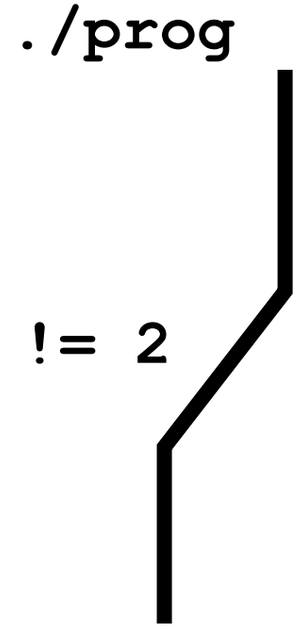
Check properties on execution paths

Bug Finding

```
int main(argc, argv)
{
  if (argc == 2) {
    printf("%c", *argv[2]);
    return -1;
  }

  return 0;
}
```

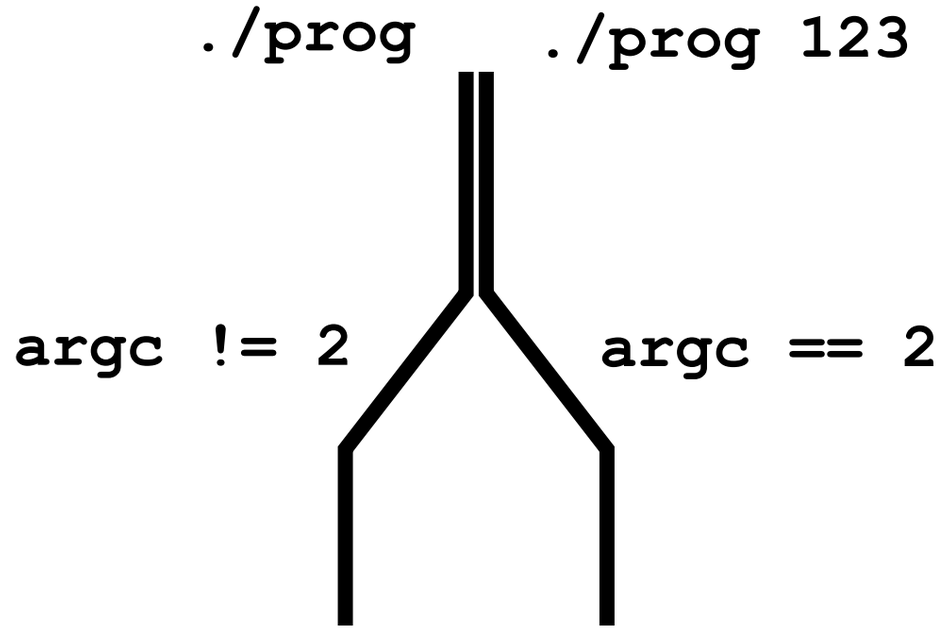
Bug Finding

```
int main(argc, argv)
{
  if (argc == 2) {
    printf(“%c”, *argv[2]);
    return -1;
  }

  return 0;
}
```



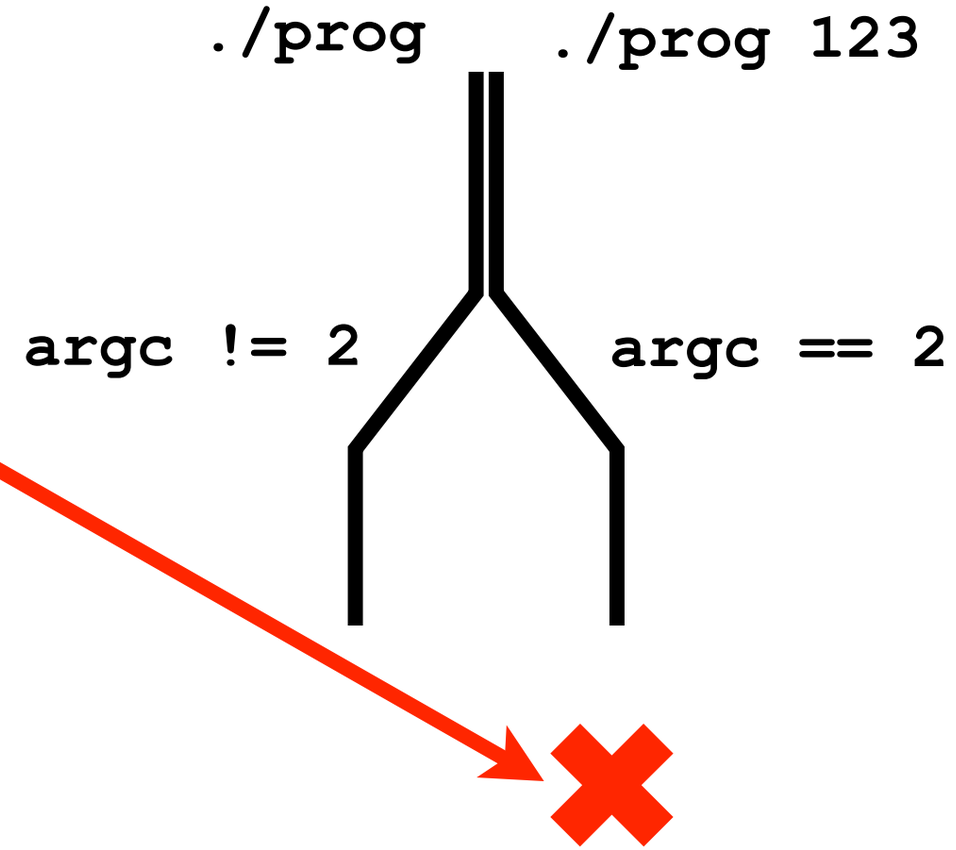
Bug Finding

```
int main(argc, argv)
{
  if (argc == 2) {
    printf("%c", *argv[2]);
    return -1;
  }

  return 0;
}
```



Bug Finding

```
int main(argc, argv)
{
  if (argc == 2) {
    printf("%c", *argv[2]);
    return -1;
  }

  return 0;
}
```

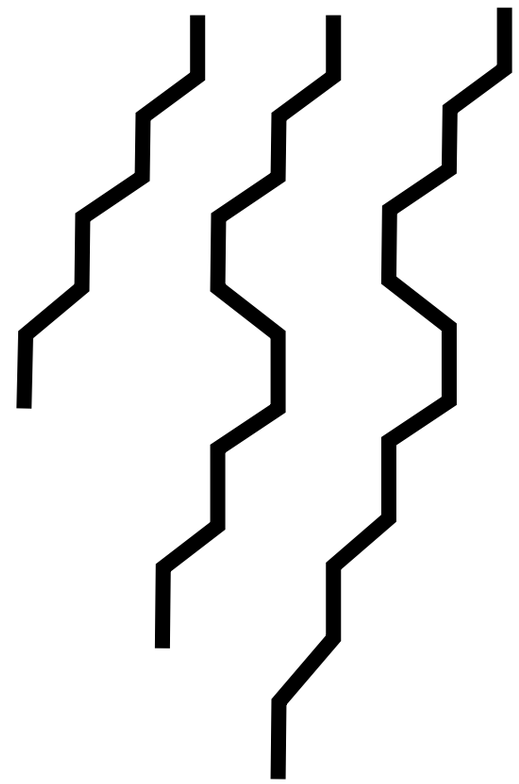


Performance Profiling

```
int matrixSum(matrix_t m)
{
  int sum=0;

  for(i = 0; i < m.w; i++)
   for(j = 0; j < m.h; j++)
     sum +=  m[i][j];

  return sum;
}
```

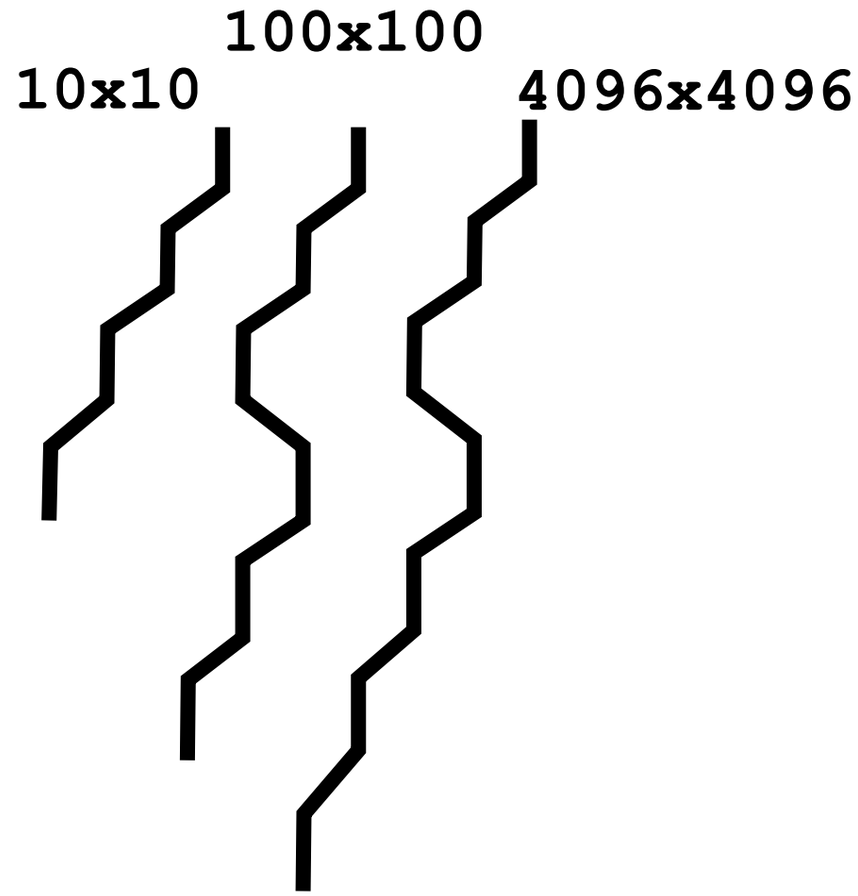
Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
    for(j = 0; j < m.h; j++)
      sum +=  m[i][j];

   return sum;
}
```



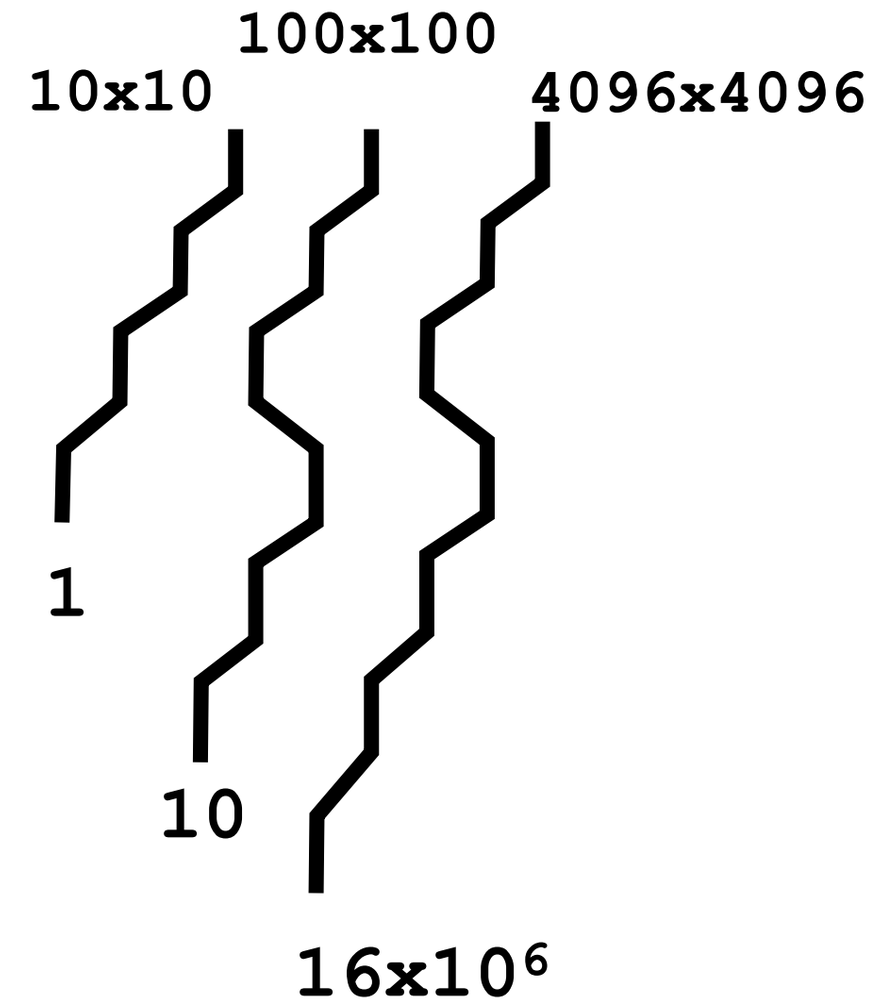
Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
    for(j = 0; j < m.h; j++)
      sum +=  m[i][j];

   return sum;
}
```



Performance Profiling

```
int matrixSum(matrix_t m)
{
   int sum=0;

   for(i = 0; i < m.w; i++)
    for(j = 0; j < m.h; j++)
      sum +=  m[i][j];

   return sum;
}
```



Cache misses

Systematic Path Enumeration

- Automatically finding the right paths
  To detect bugs
  To expose performance issues
  To ...

In-Vivo Multi-Path Analysis

Analyze a *living* system, for maximum realism

http://www.wtcc.wur.nl/NR/rdonlyres/E8C658AD-8303-41D9-A141-E5413DB9E3BD/41107/invitro.jpg
http://static.howstuffworks.com/gif/cure-for-cancer-virus-3.jpg

In-Vivo Multi-Path Analysis

Analyze a *living* system, for maximum realism

In Vivo

http://www.wtcc.wur.nl/NR/rdonlyres/E8C658AD-8303-41D9-A141-E5413DB9E3BD/41107/invitro.jpg
http://static.howstuffworks.com/gif/cure-for-cancer-virus-3.jpg

In-Vivo Multi-Path Analysis

Analyze a *living* system, for maximum realism

In Vitro

In Vivo

http://www.wtcc.wur.nl/NR/rdonlyres/E8C658AD-8303-41D9-A141-E5413DB9E3BD/41107/invitro.jpg
http://static.howstuffworks.com/gif/cure-for-cancer-virus-3.jpg

Challenge

Challenge

$2^{\text{system size}}$ paths

Today’s Approaches

Today's Approaches

Analyze only some of the paths

Introduces false negatives (FNs)

Today's Approaches

Analyze only some of the paths
Introduces false negatives (FNs)

Abstract away parts of the paths
Introduces false positives (FPs)

Outline

- Theory
  Execution consistency models
- System
  S²E: Platform for in-vivo multi-path analysis
- Results
  Using S²E in practice

http://s2e.epfl.ch

Outline

- Theory
  Execution consistency models
- System
  S²E: Platform for in-vivo multi-path analysis
- Results
  Using S²E in practice

http://s2e.epfl.ch

Execution Consistency Models

Execution Consistency Models

- Specify the set of paths to be analyzed

Execution Consistency Models

- Specify the set of paths to be analyzed
- Principled FPs/FNs trade-offs

Execution Consistency Models

- Specify the set of paths to be analyzed
- Principled FPs/FNs trade-offs
- Remember memory consistency models ?

Consistency Models in S2E

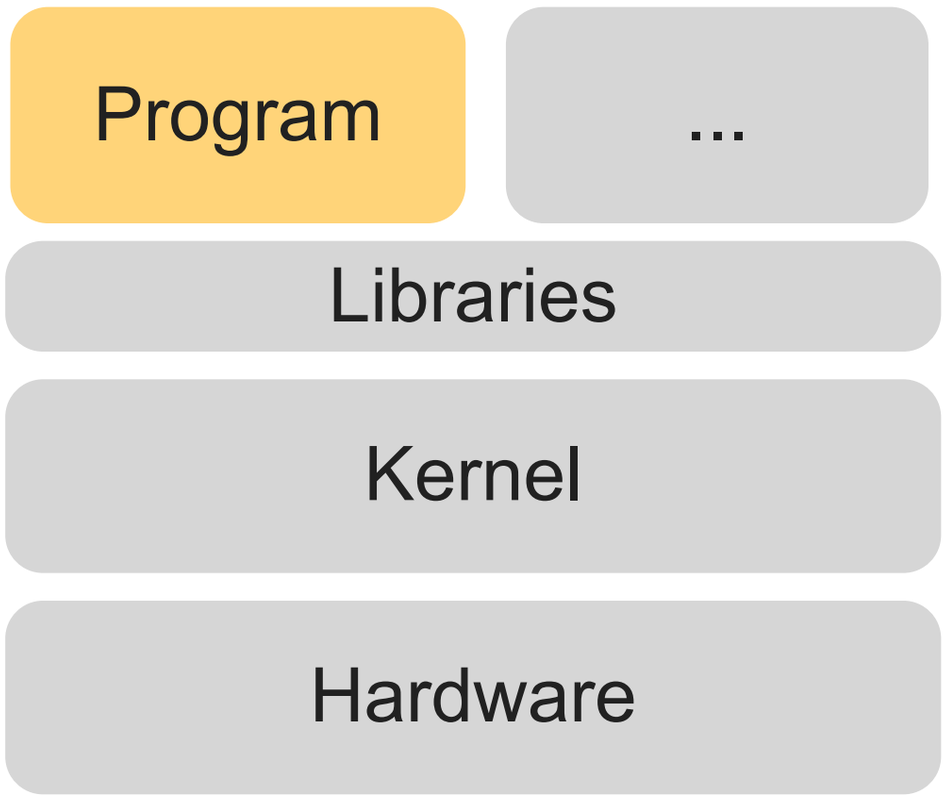
Consistency Models in S2E

```
int main(argc, argv) {

  if (argc == 0) {
     ...
  }

  p = malloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```

Program

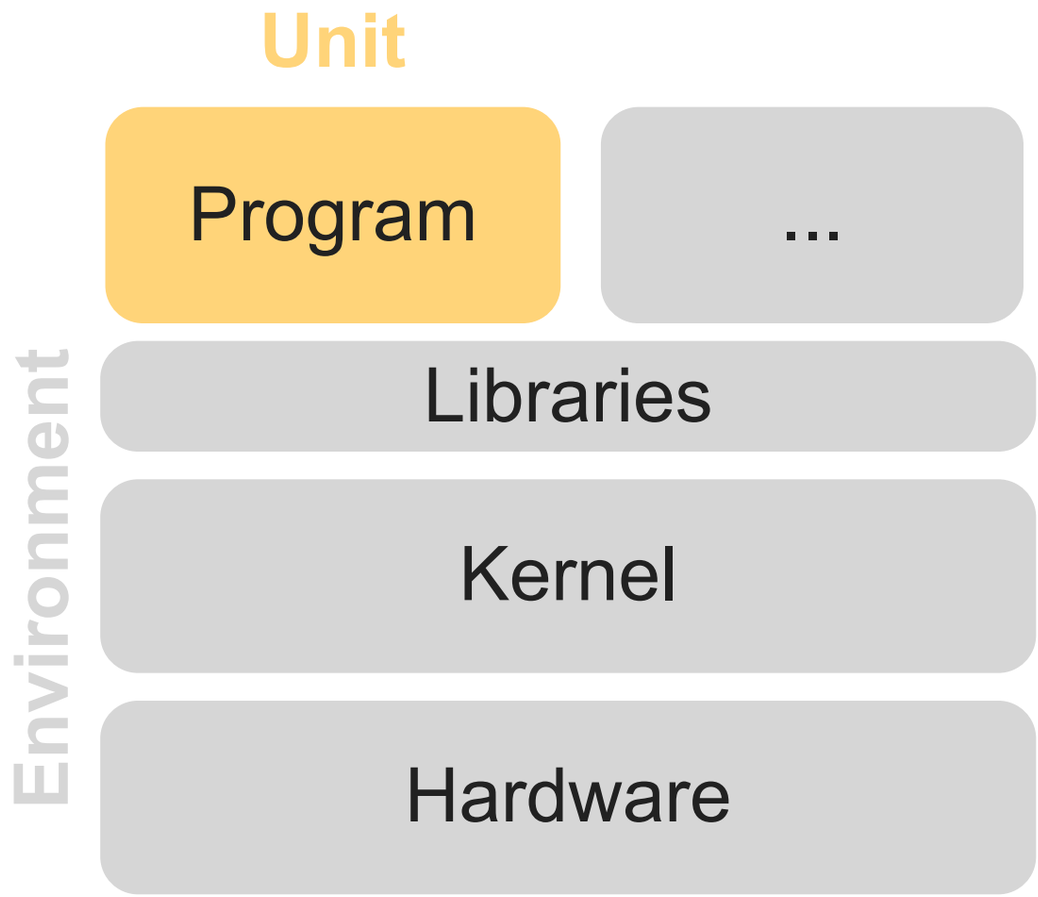
Consistency Models in S2E

```
int main(argc, argv) {

  if (argc == 0) {
     ...
  }

  p = malloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```



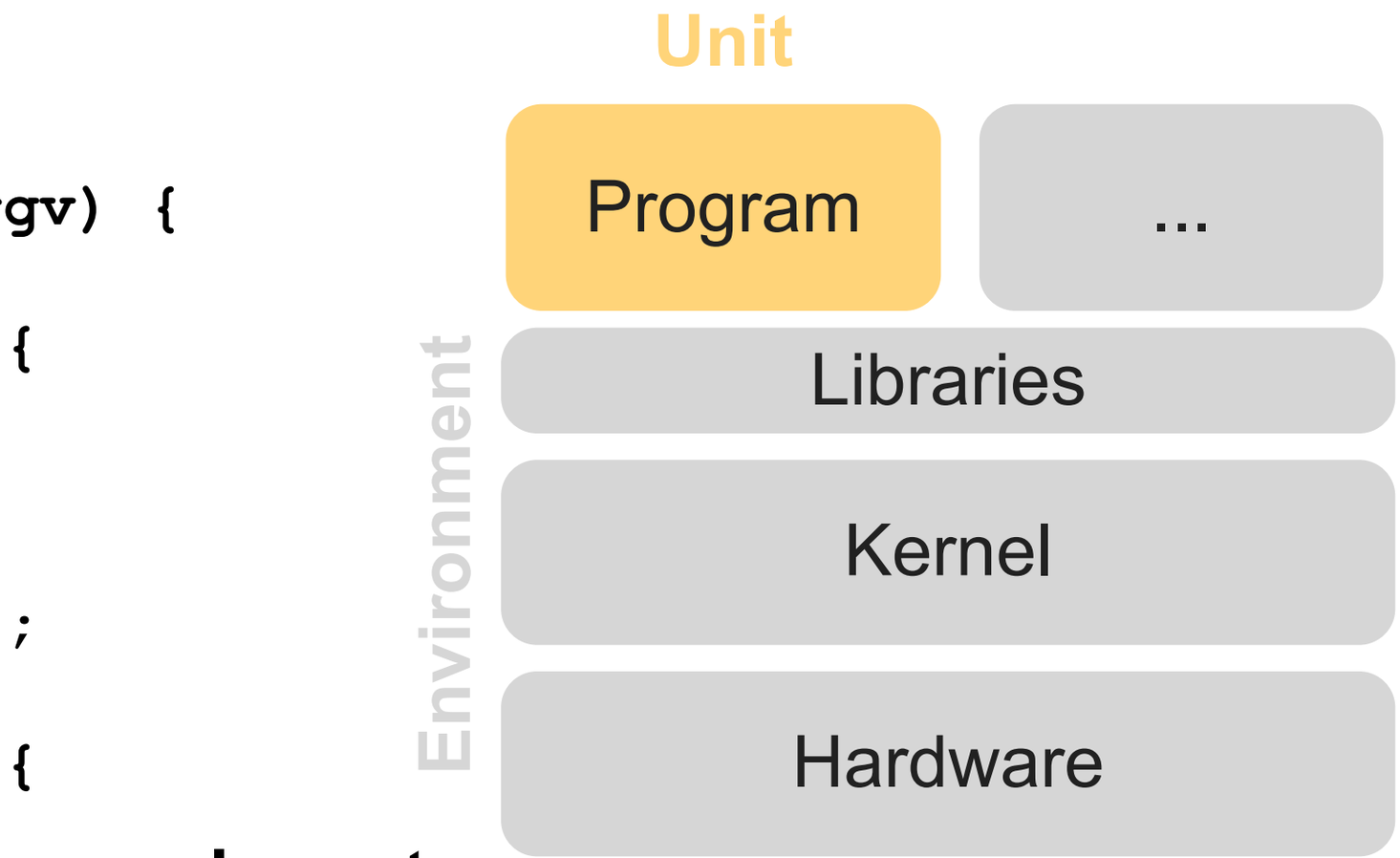
Consistency Models in S2E

```
int main(argc, argv) {

  if (argc == 0) {
     ...
  }

  p = malloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```



Consistency Models in S2E

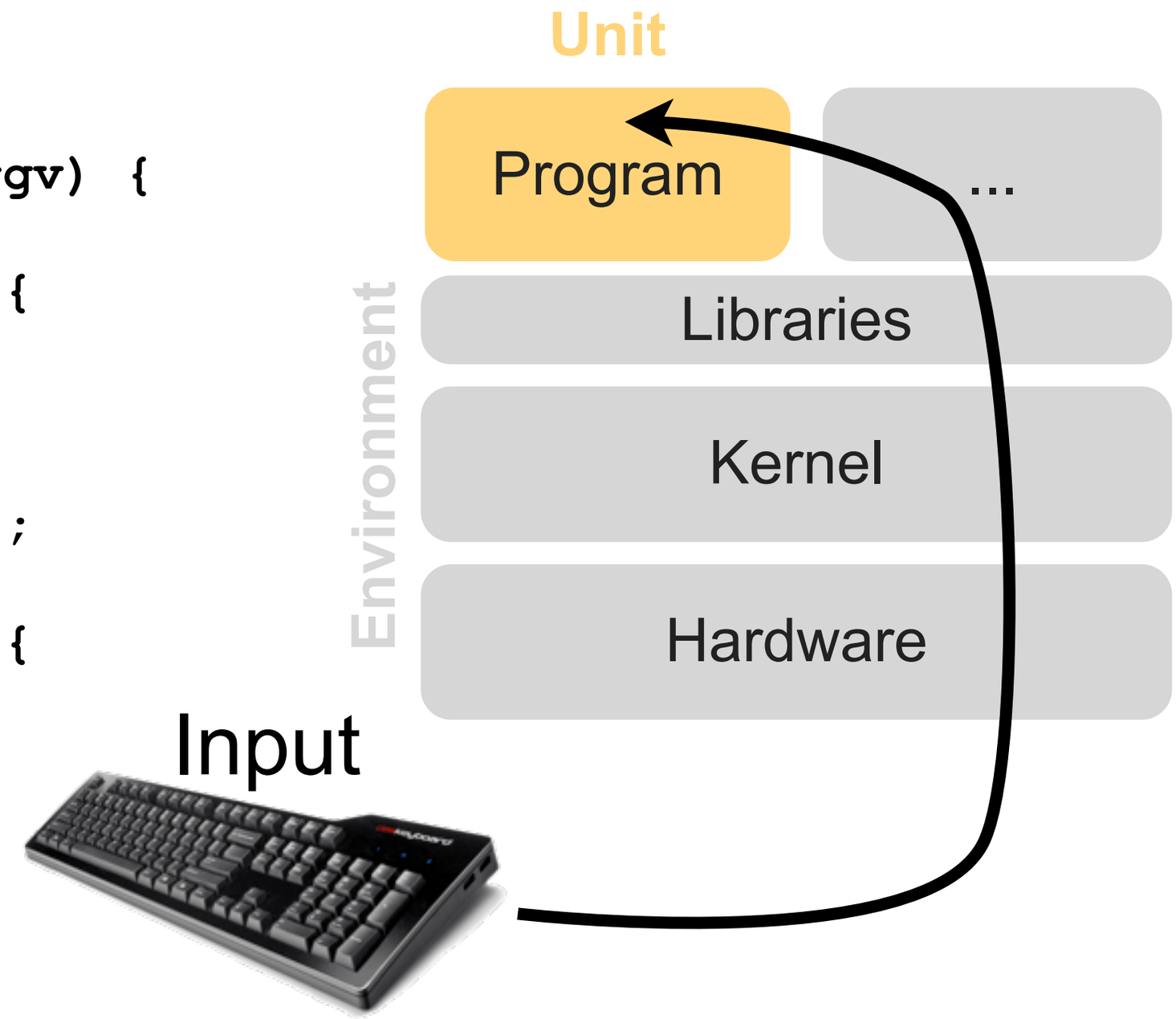
```
int main(argc, argv) {

  if (argc == 0) {
     ...
  }

  p = malloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```



Input

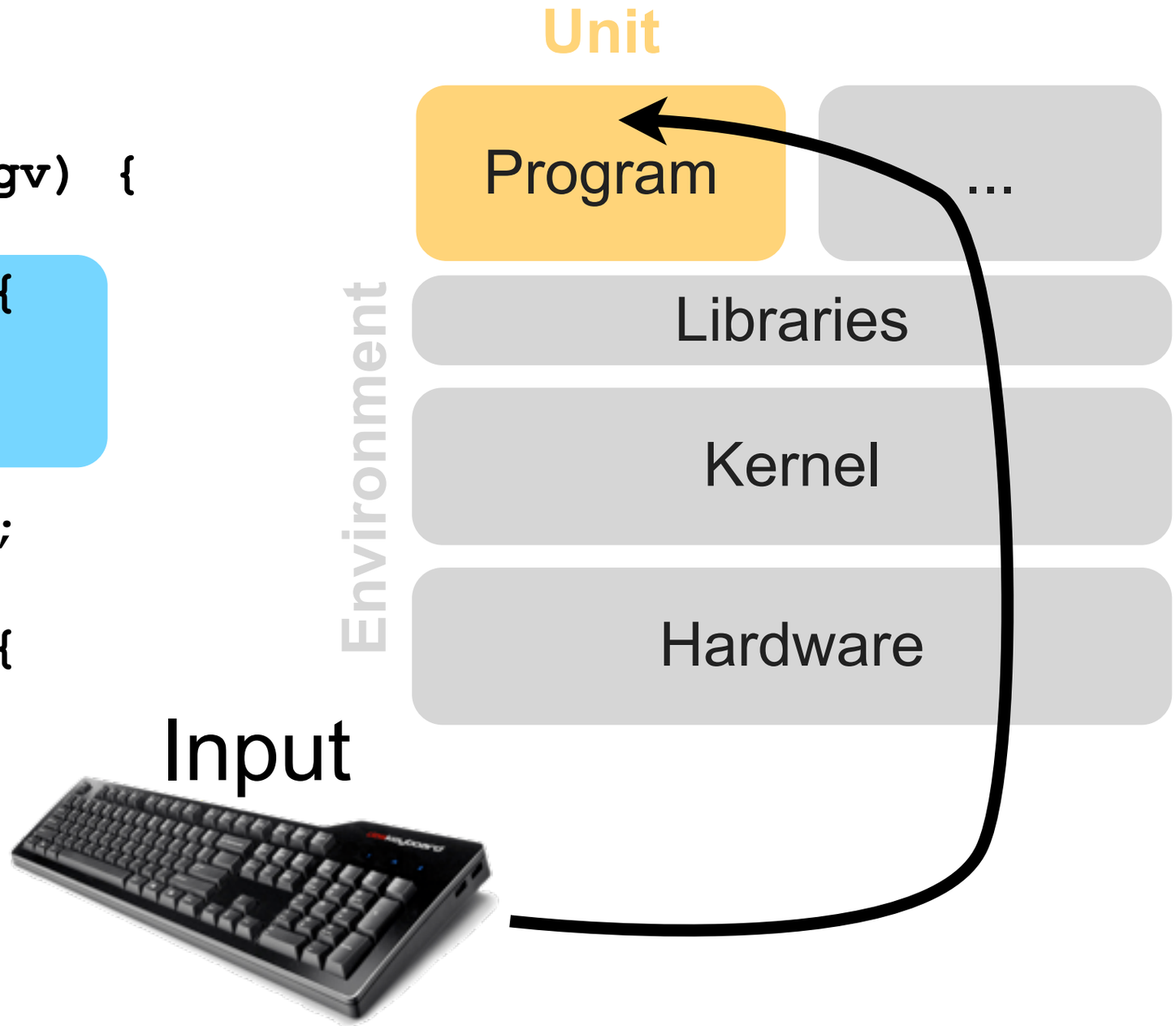
Consistency Models in S2E

```
int main(argc, argv) {

  if (argc == 0) {
     ...
  }

  p = malloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```



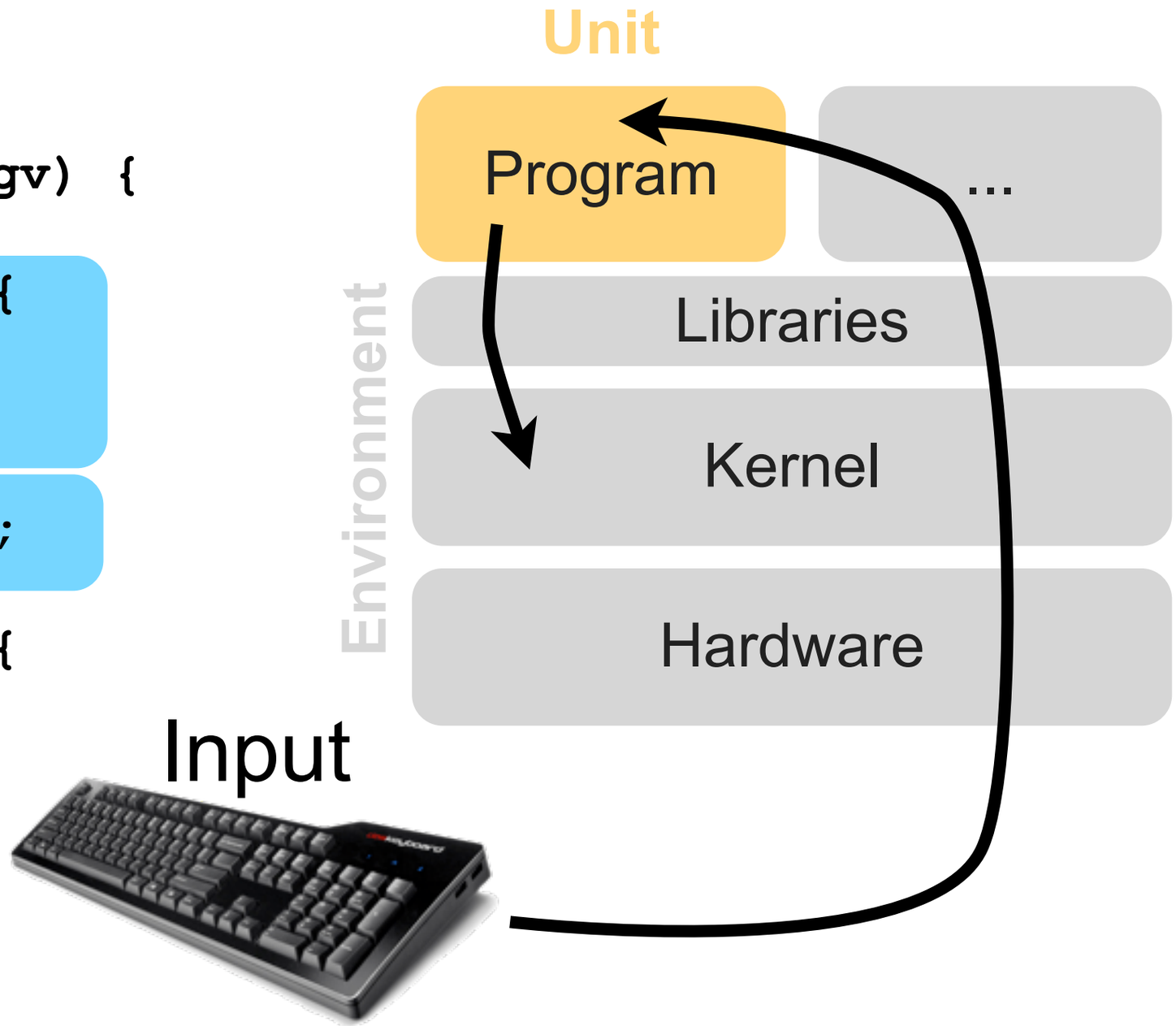
Consistency Models in S2E

```
int main(argc, argv) {
  if (argc == 0) {
      ...
  }
  p = malloc(...);

  if (p == NULL) {
      ...
  }
  ...
}
```



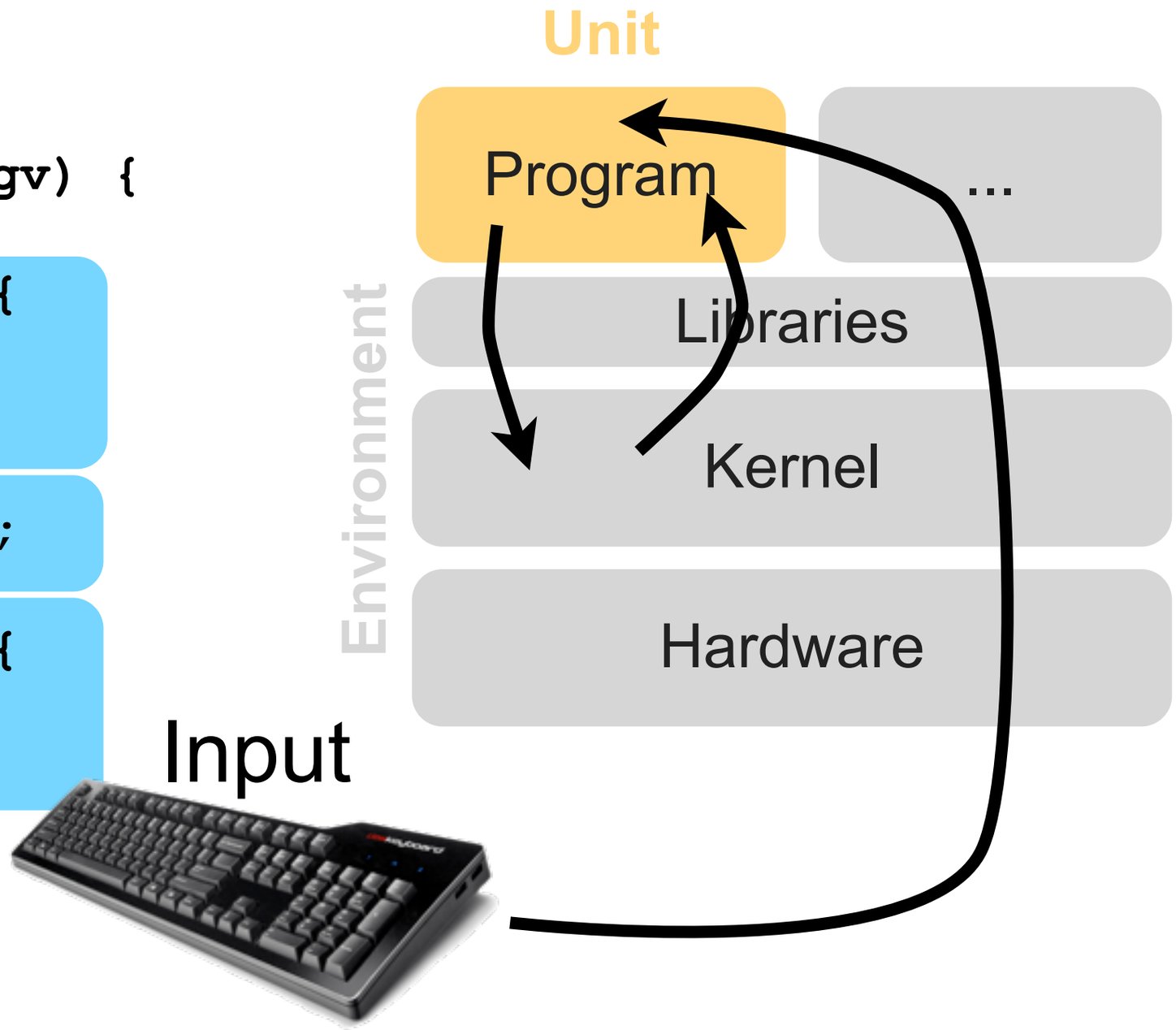
Consistency Models in S2E

```
int main(argc, argv) {
  if (argc == 0) {
      ...
  }
  p = malloc(...);
  if (p == NULL) {
      ...
  }
  ...
}
```



Consistency Models in S2E

```
int main(argc, argv) {
  if (argc == 0) {
      ...
  }
  p = malloc(...);
  if (p == NULL) {
      ...
  }
  ...
}
```



SC-SE

Strictly Consistent *System*-Level Execution

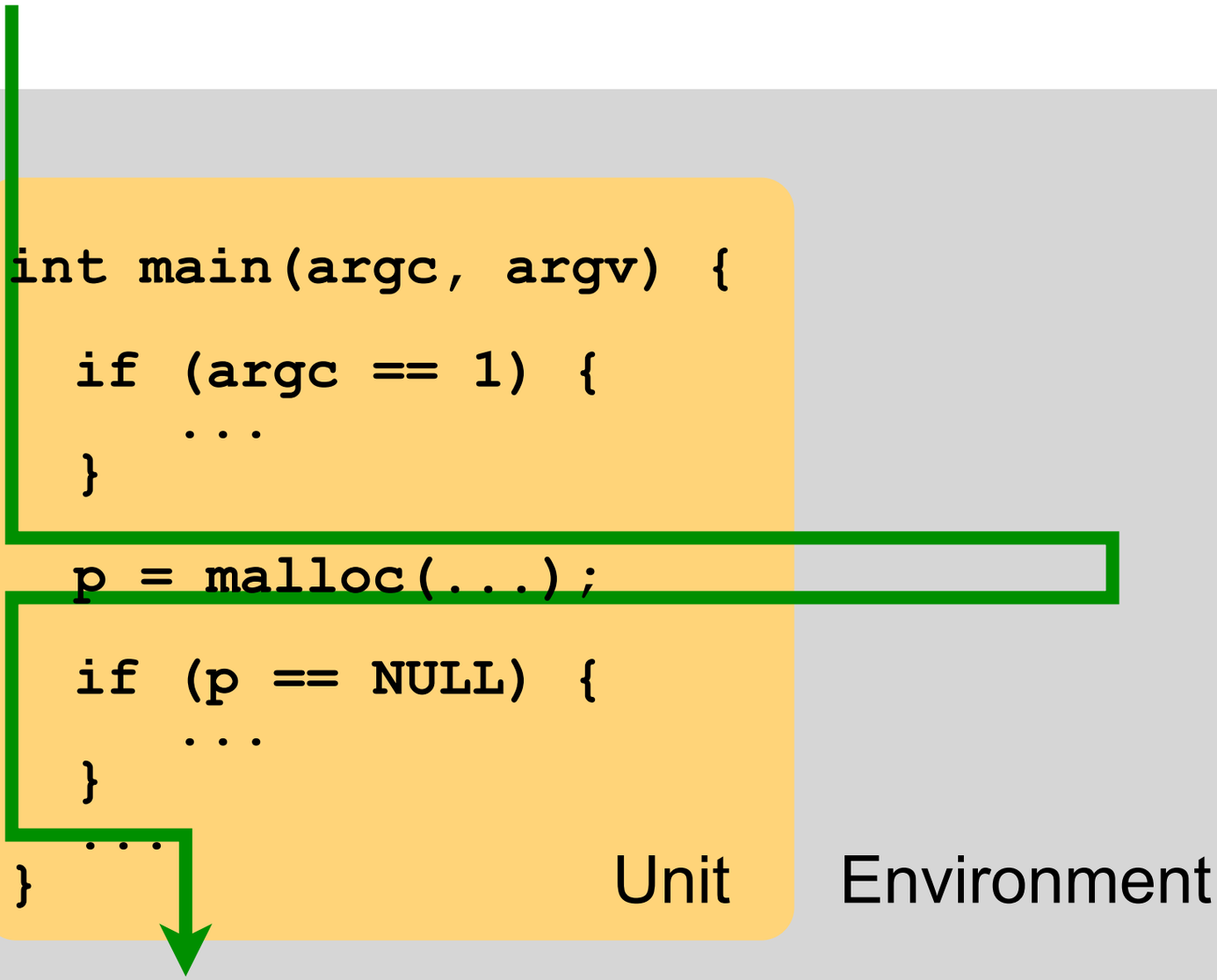
```
int main(argc, argv) {
  if (argc == 1) {
     ...
  }
  p = malloc(...);
  if (p == NULL) {
     ...
  }
  ...
}
```

Unit

Environment

SC-SE

Strictly Consistent *System*-Level Execution



SC-SE

Strictly Consistent *System*-Level Execution

System Input

```
int main(argc, argv) {
  if (argc == 1) {
     ...
  }
  p = malloc(...);
  if (p == NULL) {
     ...
  }
  ...
}
```

Unit

Environment

SC-SE

Strictly Consistent *System*-Level Execution

System Input

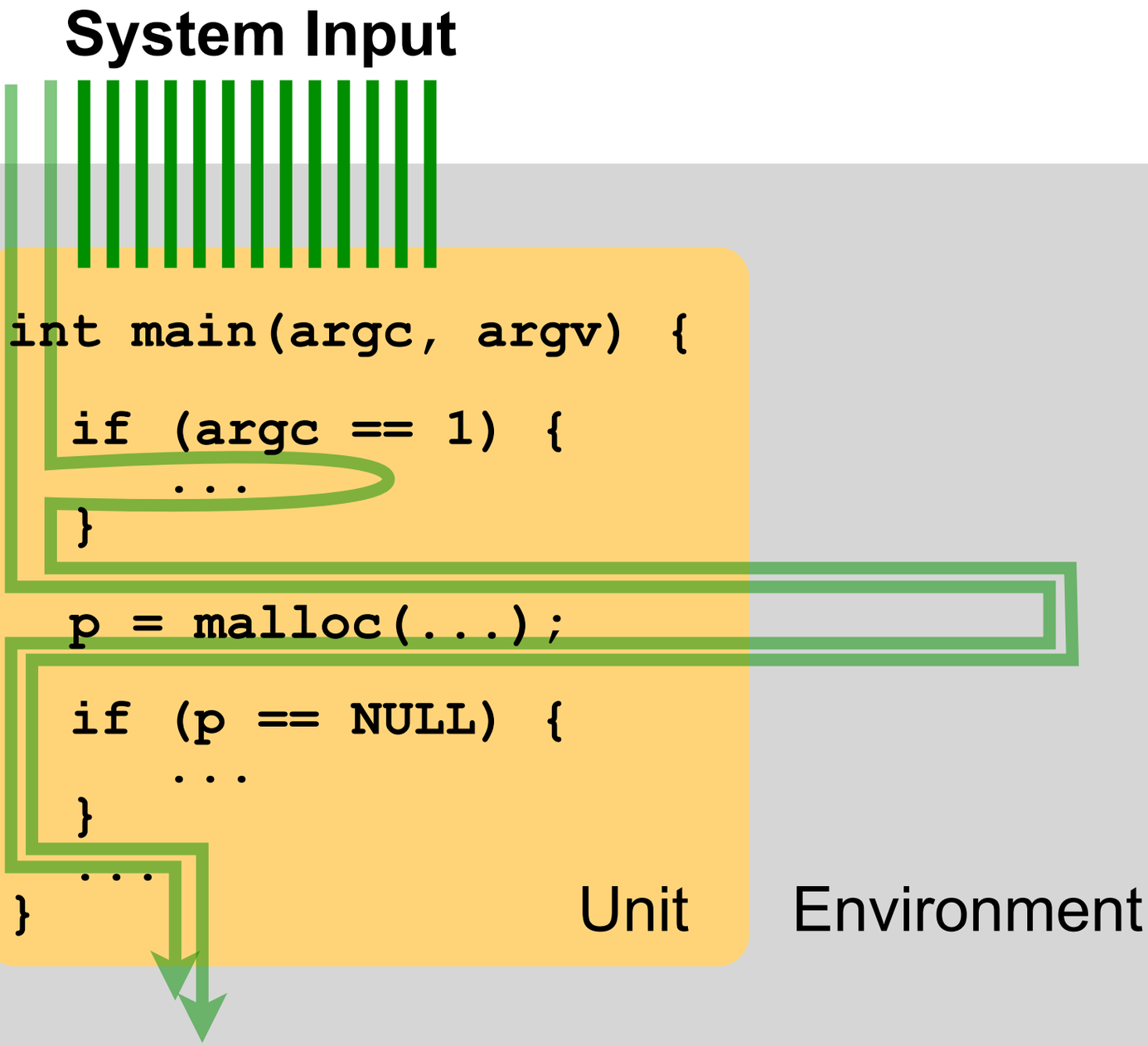
```
int main(argc, argv) {
  if (argc == 1) {
     ...
  }
  p = malloc(...);
  if (p == NULL) {
     ...
  }
  ...
}
```

Unit

Environment

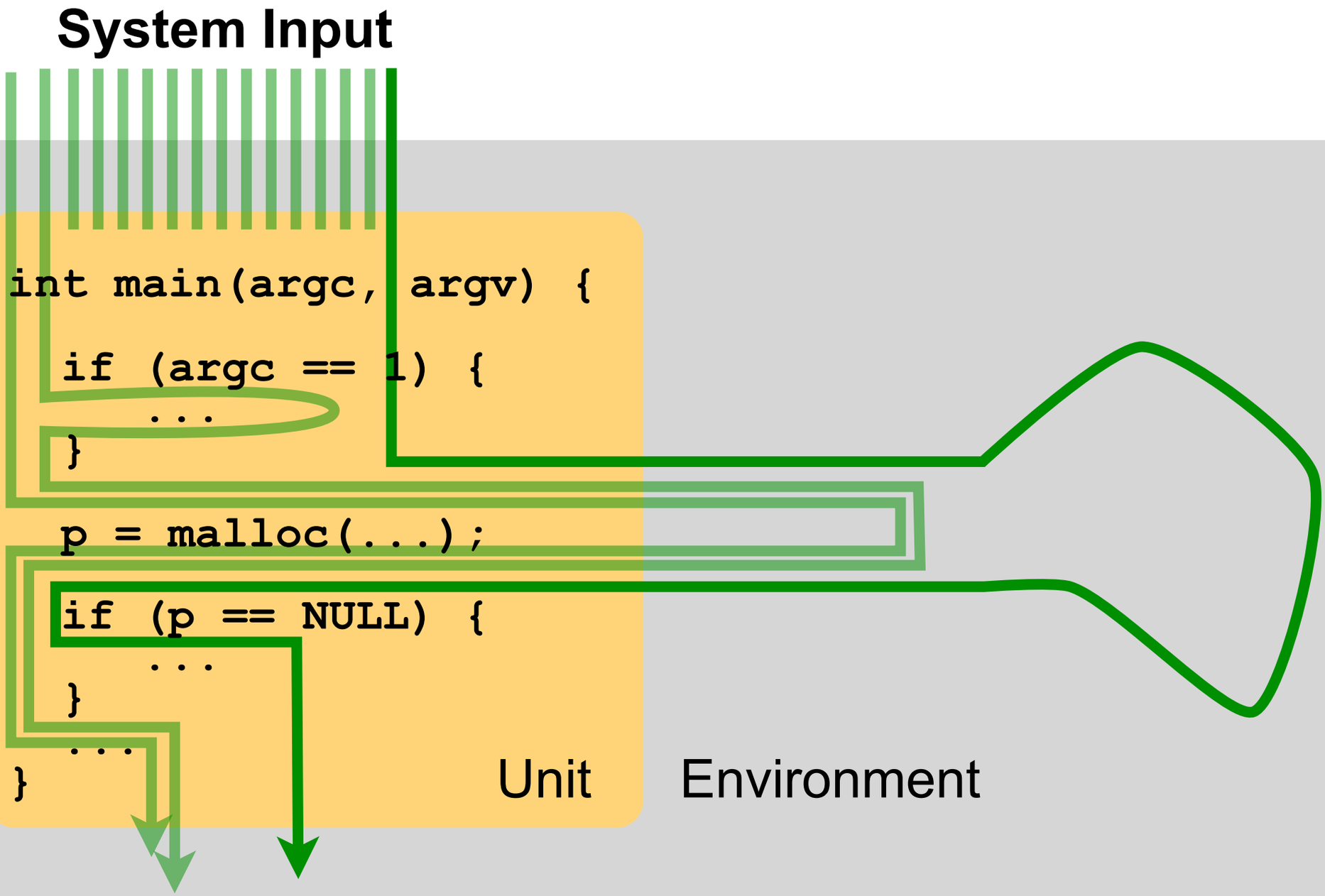
SC-SE

Strictly Consistent *System*-Level Execution



SC-SE

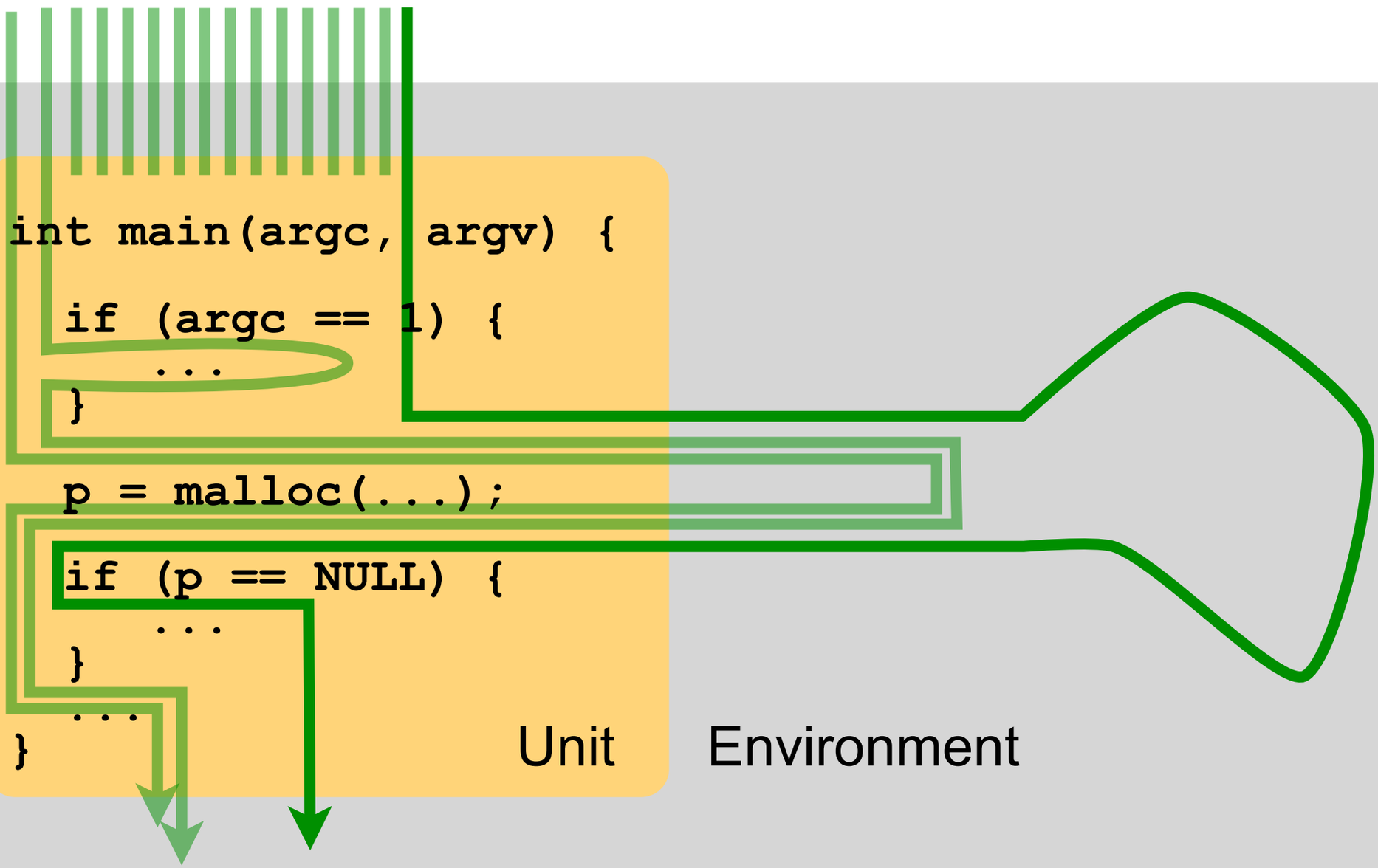
Strictly Consistent *System*-Level Execution



SC-SE

Strictly Consistent *System*-Level Execution

Zero FNs
Zero FPs



SC-SE

Strictly Consistent *System*-Level Execution

Zero FNs
Zero FPs

System Input

```
int main(argc, argv) {
  if (argc == 1) {
     ...
  }
  p = malloc(...);
  if (p == NULL) {
     ...
  }
  ...
}
```

Unit

Environment

Huge Number of Paths

SC-UE

Strictly Consistent *Unit-Level* Execution

```
int main(argc, argv) {
  if (argc == 1) {
     ...
  }
  p = malloc(...);
  if (p == NULL) {
     ...
  }
  ...
}
```

Unit

Environment

SC-UE

Strictly Consistent *Unit-Level* Execution

Unit Input

```
int main(argc, argv) {
  if (argc == 1) {
     ...
  }
  p = malloc(...);
  if (p == NULL) {
     ...
  }
  ...
}
```

Unit

Environment

SC-UE

Strictly Consistent *Unit-Level* Execution

Unit Input

```
int main(argc, argv) {
  if (argc == 1) {
     ...
  }
  p = malloc(...);
  if (p == NULL) {
     ...
  }
  ...
}
```

Unit

Environment

SC-UE

Strictly Consistent *Unit-Level* Execution

Presence of FNs

Unit Input

```
int main(argc, argv) {
  if (argc == 1) {
    ...
  }
  p = malloc(...);
  if (p == NULL) {
    ...
  }
  ...
}
```

Unit

Environment

RC
Relaxed Consistency

Unit Input

```
int main(argc, argv) {

  if (argc == 1) {
     ...
  }

  p = malloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```

RC
Relaxed Consistency

Unit Input

```
int main(argc, argv) {

  if (argc == 1) {
     ...
  }

  p = malloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```

Relax returned values

$p' \in \{NULL, p\}$

RC
Relaxed Consistency

Unit Input

```
int main(argc, argv) {

  if (argc == 1) {
     ...
  }

  p = malloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```

Relax returned values

$p' \in \{NULL, p\}$

RC
Relaxed Consistency

Unit Input

```
int main(argc, argv) {
  if (argc == 1) {
    ...
  }
  p = malloc(...);
  if (p == NULL) {
    ...
  }
  ...
}
```

Relax returned values

$p' \in \{NULL, p\}$

RC
Relaxed Consistency

Unit Input

```
int main(argc, argv) {
  if (argc == 1) {
    ...
  }
  p = malloc(...);
  if (p == NULL) {
    ...
  }
  ...
}
```

Relax returned values

$p' \in \{NULL, p\}$

Introduces memory leak

RC
Relaxed Consistency

Unit Input

```
int main(argc, argv) {
  if (argc == 1) {
    ...
  }
  p = malloc(...);
  if (p == NULL) {
    ...
  }
  ...
}
```

Relax returned values

$p' \in \{NULL, p\}$

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths
Concrete			

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths
Concrete			
SC-SE			

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths
Concrete			
SC-SE			
SC-UE			

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths
Concrete			
SC-SE			
SC-UE			
RC			

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths
Concrete			
SC-SE			
SC-UE			
RC			

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths
Concrete			
SC-SE			
SC-UE			
RC			
CFG			

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths
Concrete			
SC-SE			
SC-UE			
RC			
CFG			
Local			

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths	Uses
Concrete				Valgrind
SC-SE				KLEE
SC-UE				DART
RC				RevNIC
CFG				Disassemblers
Local				DDT

Execution Consistency Models

Model	FNs w.r.t. unit	FPs w.r.t. unit	# system paths	Uses
Concrete				Valgrind
SC-SE				KLEE
SC-UE				DART
RC				RevNIC
CFG				Disassemblers
Local				DDT

Design your own models

Outline

- Theory
  Execution consistency models
- System
  S²E: Platform for in-vivo multi-path analysis
- Results
  Using S²E in practice

http://s2e.epfl.ch

Outline

- Theory
  Execution consistency models
- System
  S²E: Platform for in-vivo multi-path analysis
- Results
  Using S²E in practice

http://s2e.epfl.ch

Symbolic Execution

Symbolic Execution

```
int func(int a, int b)
{

   if (a > 0) {
       ...
   }

   if (b < 0) {
   ...
   }

}
```

Symbolic Execution

a=1 b=2 a=3 b=5 a=5 b=2 a=10 b=22

```
int func(int a, int b)
{

  if (a > 0) {
      ...
  }

  if (b < 0) {
  ...
  }

}
```

Symbolic Execution

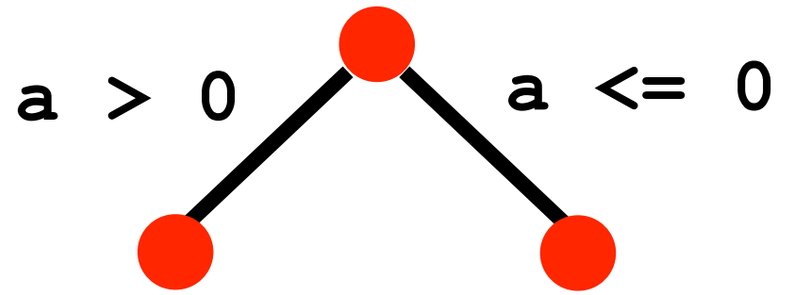
a=λ b=δ

```
int func(int a, int b)
{

  if (a > 0) {
      ...
  }

  if (b < 0) {
  ...
  }

}
```

Symbolic Execution

a=λ b=δ

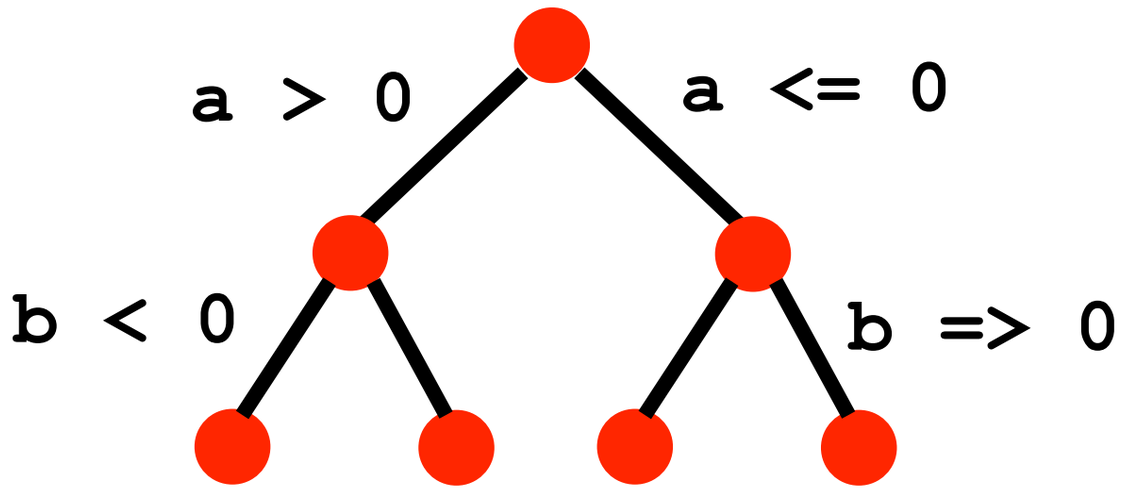
```
int func(int a, int b)
{

   if (a > 0) {
       ...
   }

   if (b < 0) {
   ...
   }

}
```



Symbolic Execution

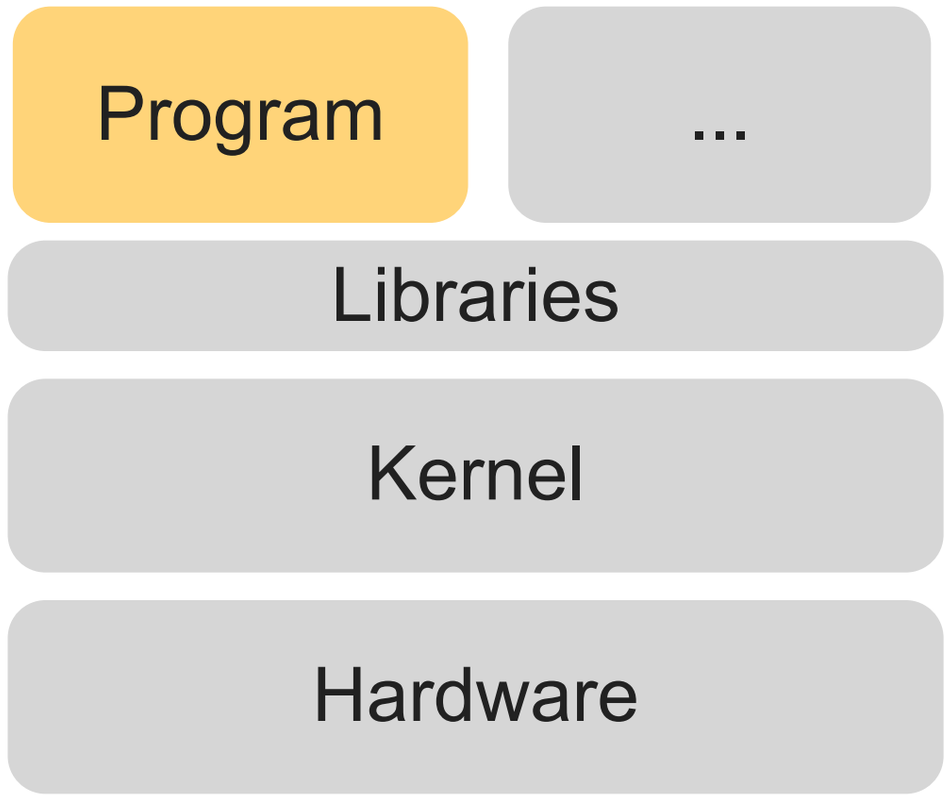
a=λ b=δ

```
int func(int a, int b)
{

   if (a > 0) {
       ...
   }

   if (b < 0) {
   ...
   }

}
```



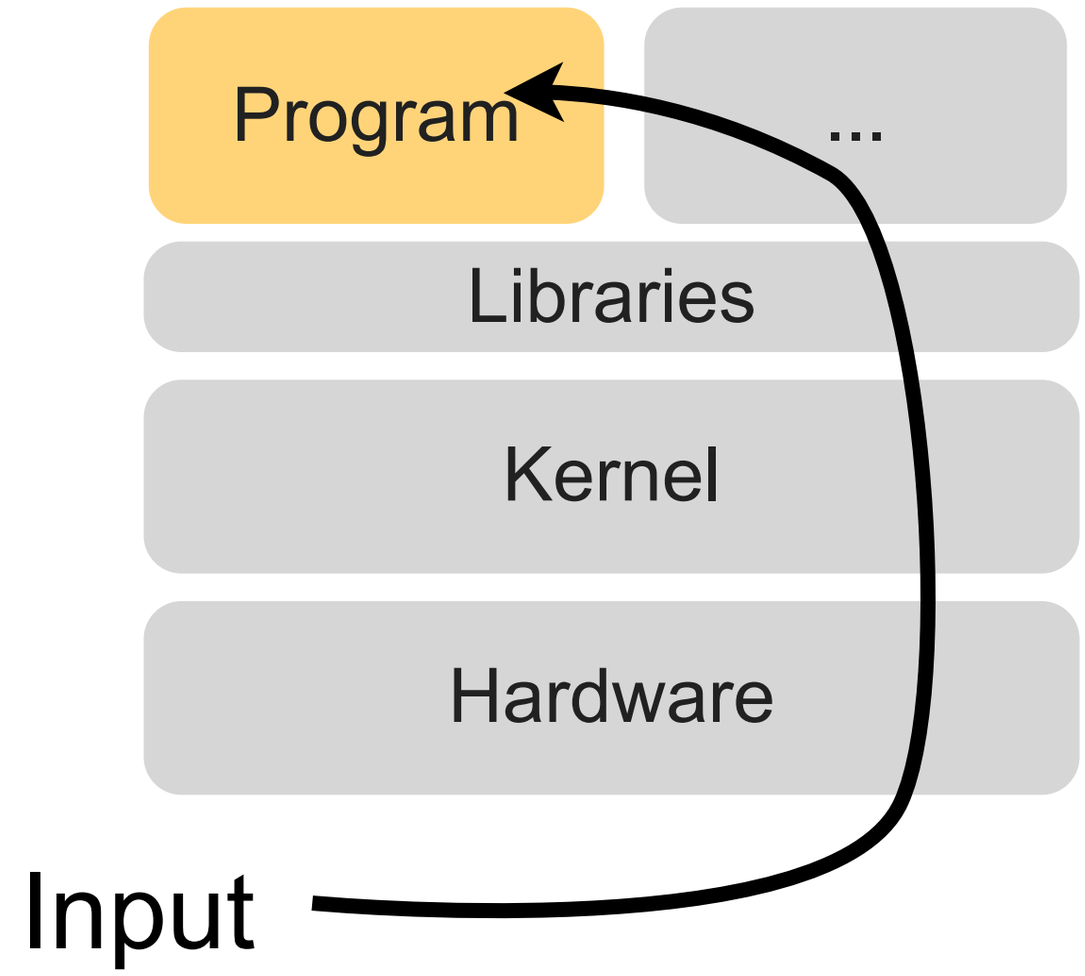
Concrete ➡ Symbolic

```
int main(argc, argv) {

  if (argc == 0) {
      ...
  }

  p = malloc(...);

  if (p == NULL) {
      ...
  }
  ...
}
```



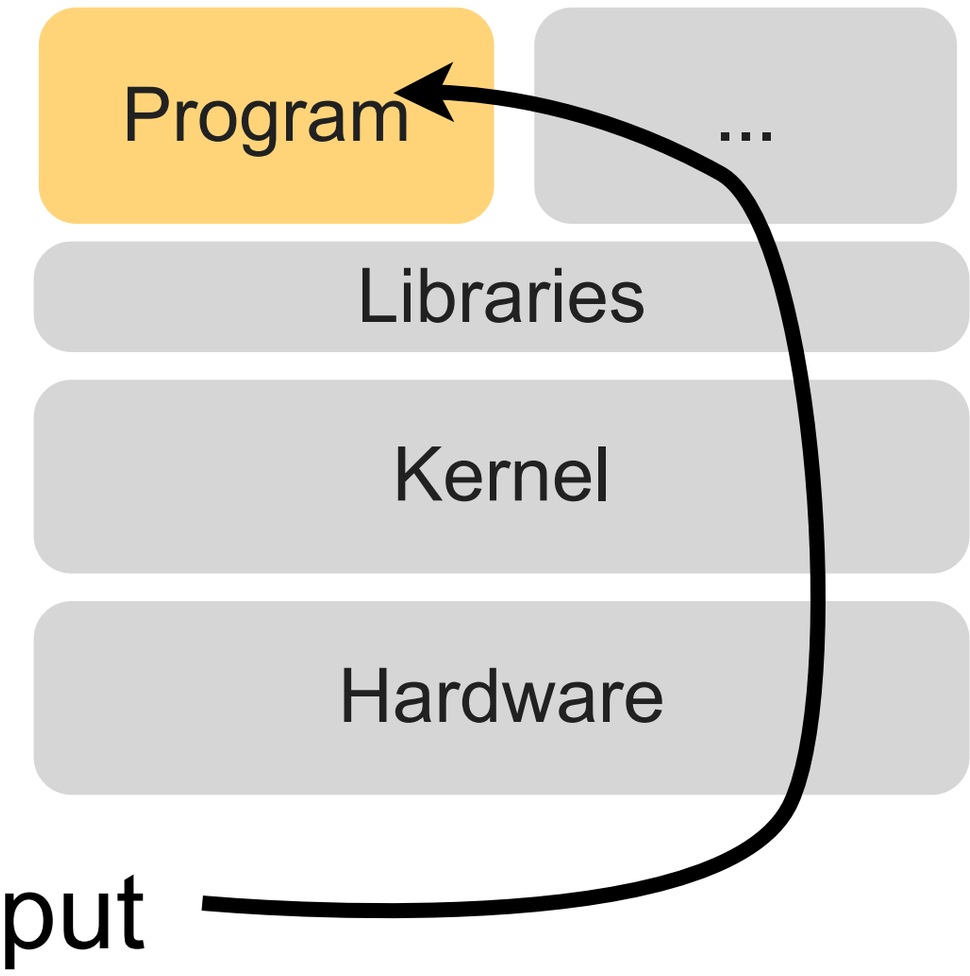
Concrete ➡ Symbolic

```
int main(argc, argv) {

  if (argc == 0) {
      ...
  }

  p = malloc(...);

  if (p == NULL) {
      ...
  }
  ...
}
```



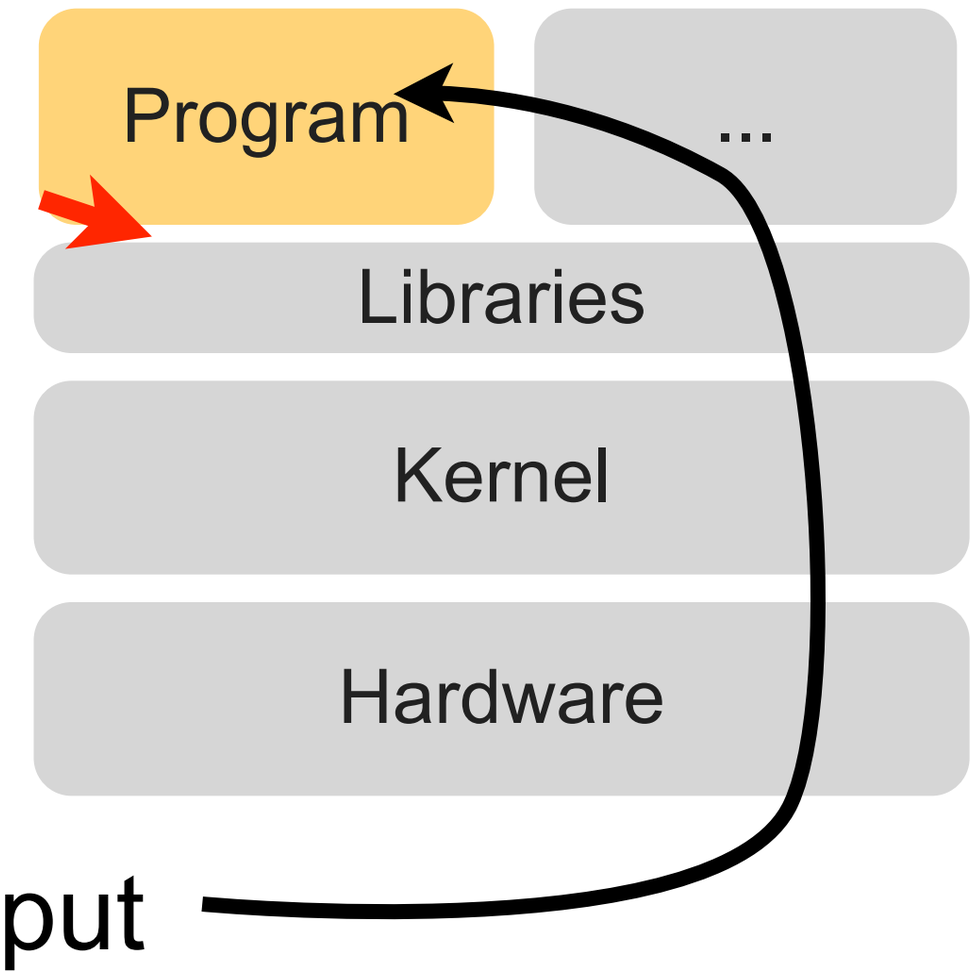
Concrete ➡ Symbolic

```
int main(argc, argv) {

  if (argc == 0) {
      ...
  }

  p = malloc(...);

  if (p == NULL) {
      ...
  }
  ...
}
```



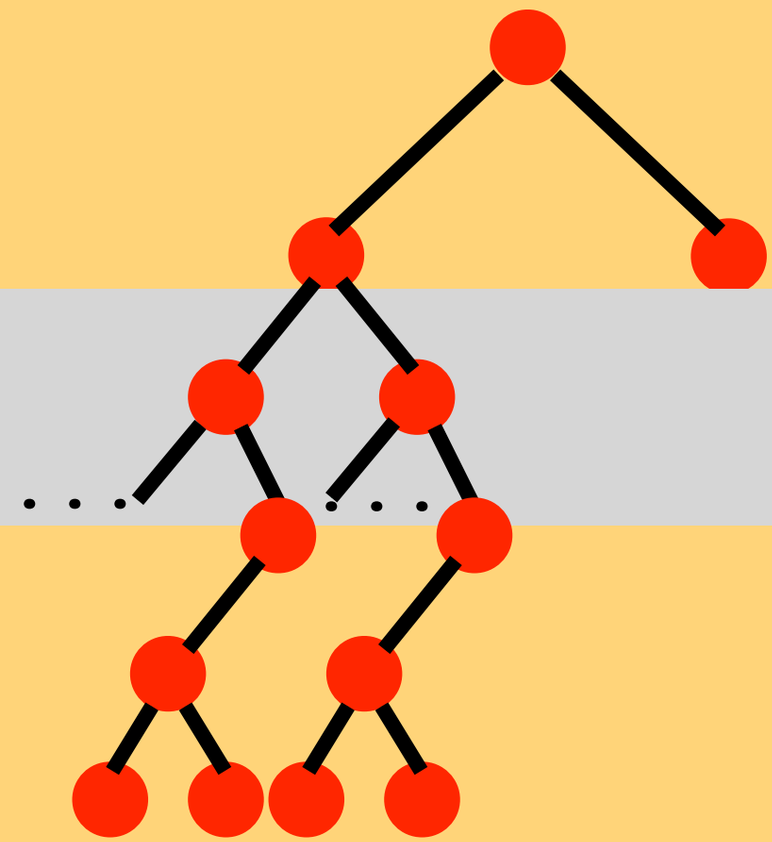
Concrete ➡ Symbolic

```
int main(argc, argv) {

  if (argc == 0) {
      ...
  }

  p = malloc(...);

  if (p == NULL) {
      ...
  }
  ...
}
```



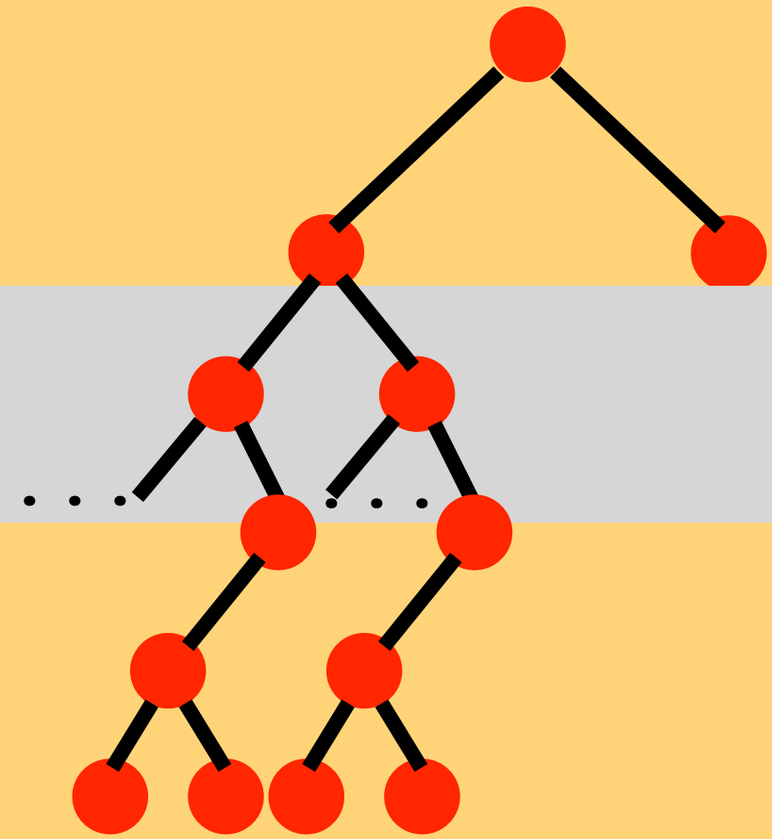
Concrete ➡ Symbolic

```
int main(argc, argv) {

    if (argc == 0) {
        ...
    }

    p = malloc(...);

    if (p == NULL) {
        ...
    }
    ...
}
```



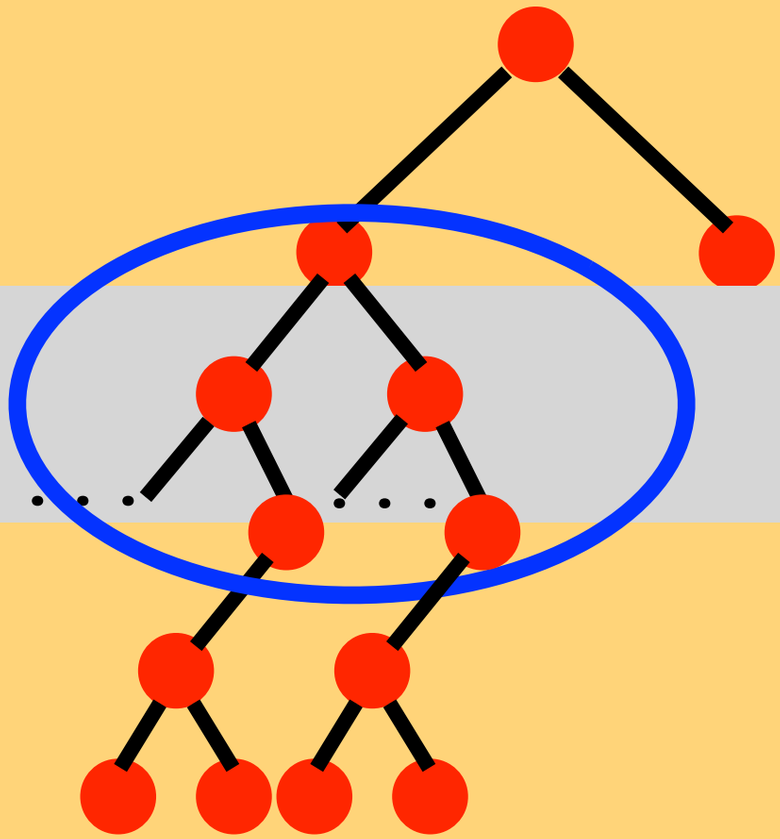
Concrete ➡ Symbolic

```
int main(argc, argv) {

    if (argc == 0) {
        ...
    }

    p = malloc(...);

    if (p == NULL) {
        ...
    }
    ...
}
```



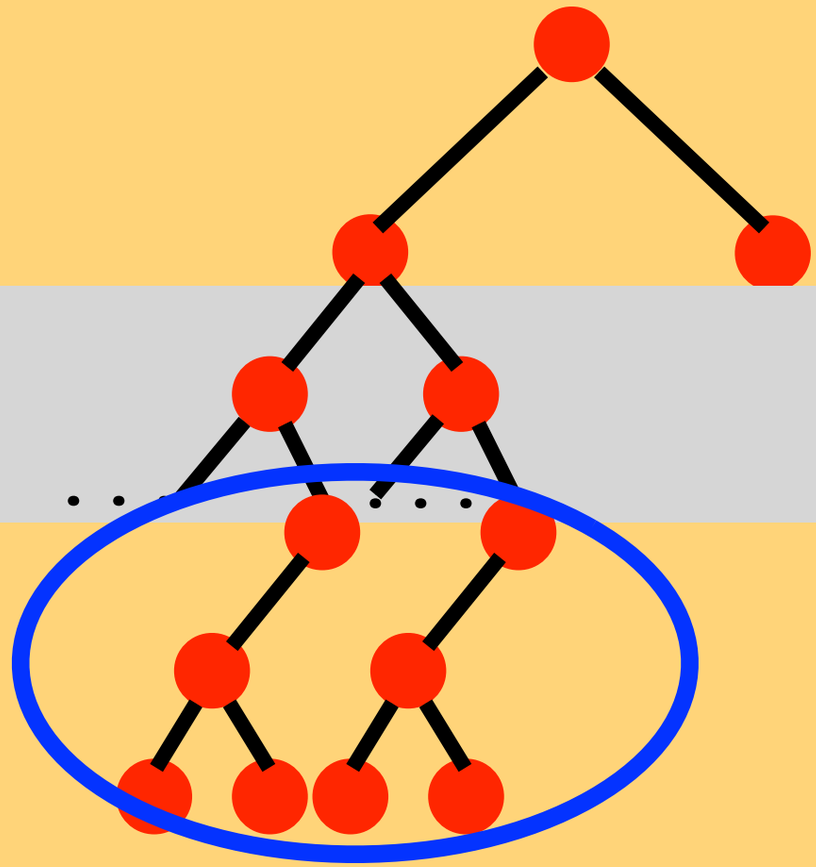
Concrete ➡ Symbolic

```
int main(argc, argv) {

    if (argc == 0) {
        ...
    }

    p = malloc(...);

    if (p == NULL) {
        ...
    }
    ...
}
```



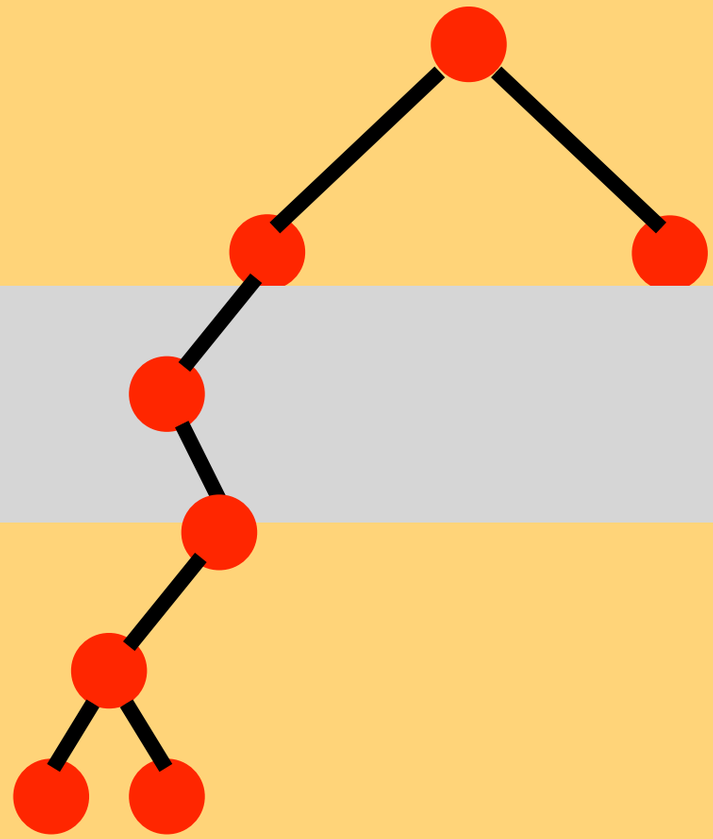
Concrete ➡ Symbolic

```
int main(argc, argv) {

    if (argc == 0) {
        ...
    }

    p = malloc(...);

    if (p == NULL) {
        ...
    }
    ...
}
```



Symbolic ➡ Concrete

```
int main(argc, argv) {

    if (argc == 0) {
        ...
    }

    p = malloc(...);

    if (p == NULL) {
        ...
    }
    ...
}
```

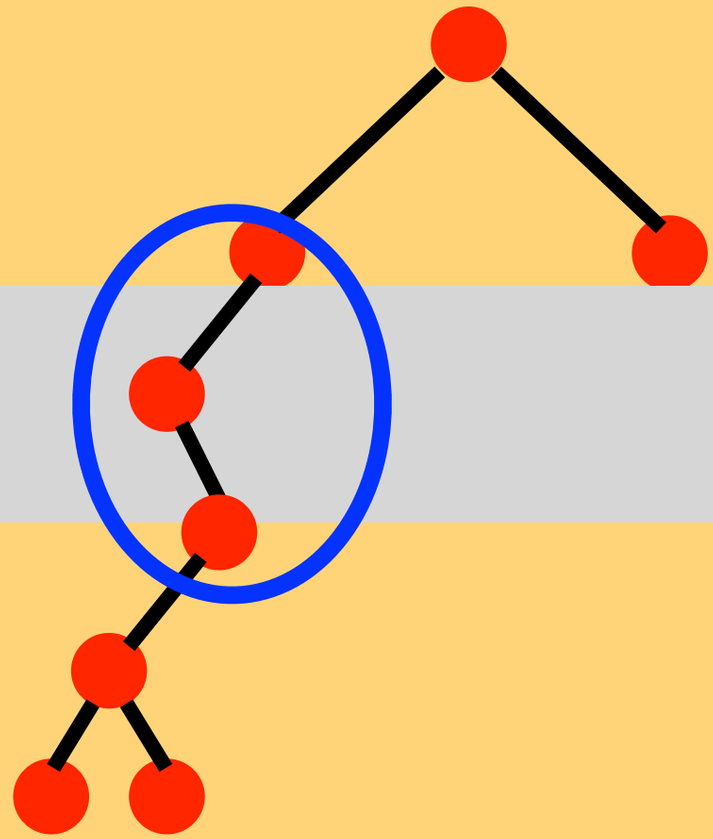


Unit

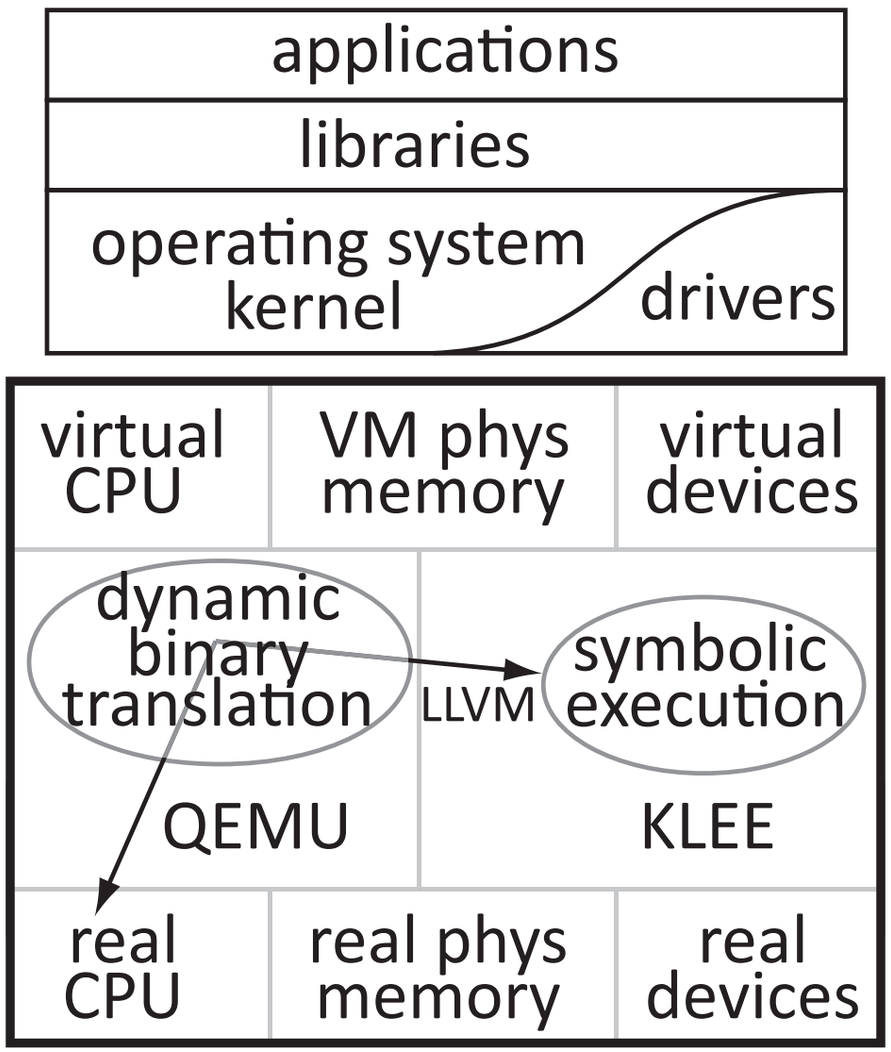
Env.

Symbolic ➡ Concrete

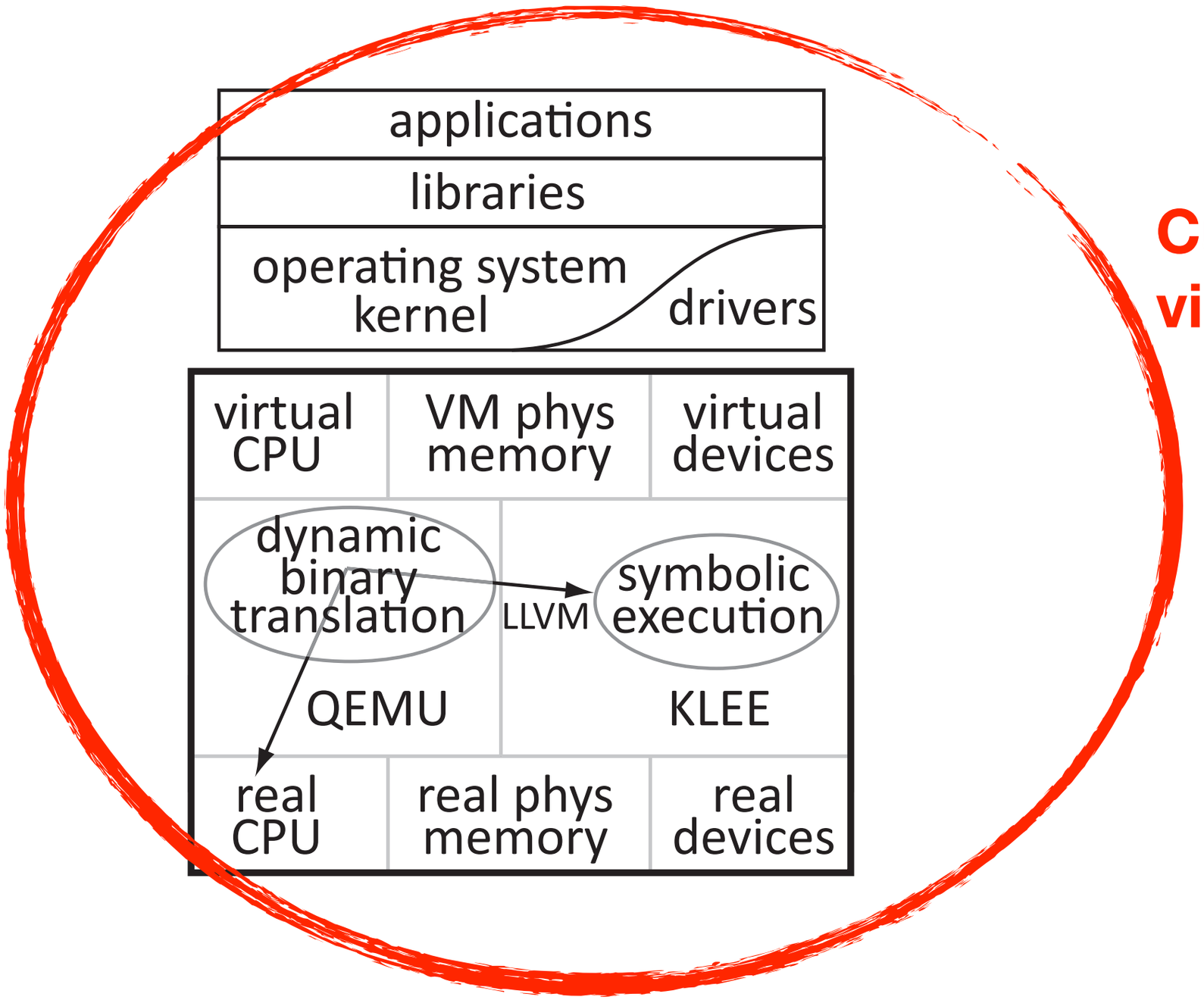
```
int main(argc, argv) {

    if (argc == 0) {
        ...
    }

    p = malloc(...);

    if (p == NULL) {
        ...
    }
    ...
}
```



S2E Is A Virtual Machine

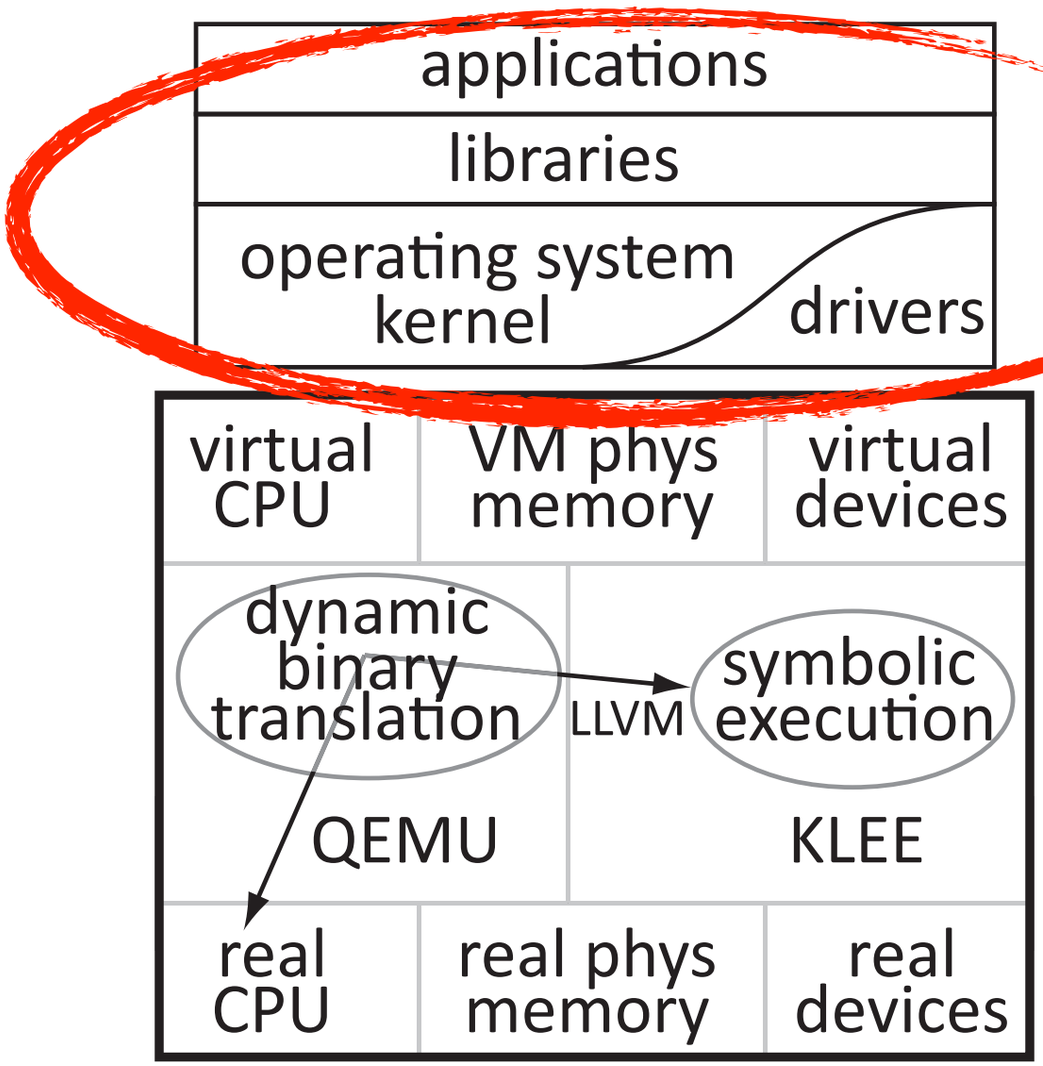


S2E Is A Virtual Machine



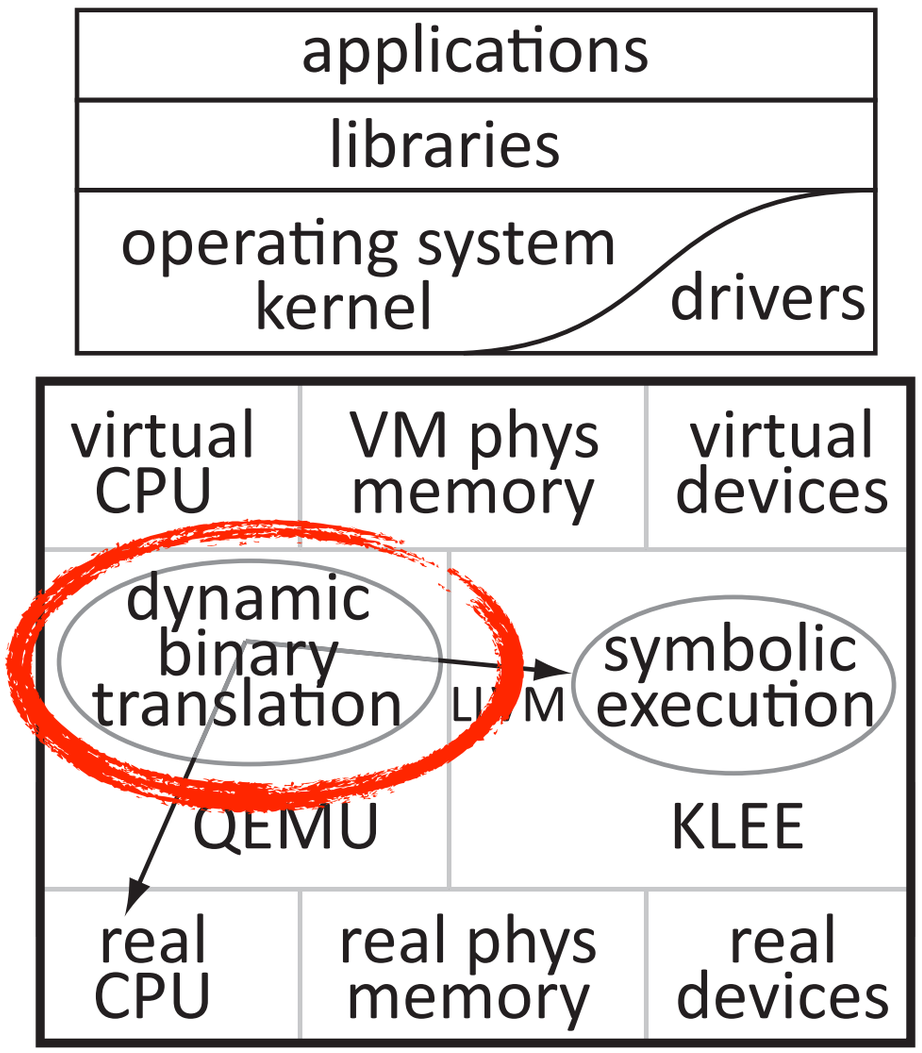
Customized virtual machine

S2E Is A Virtual Machine



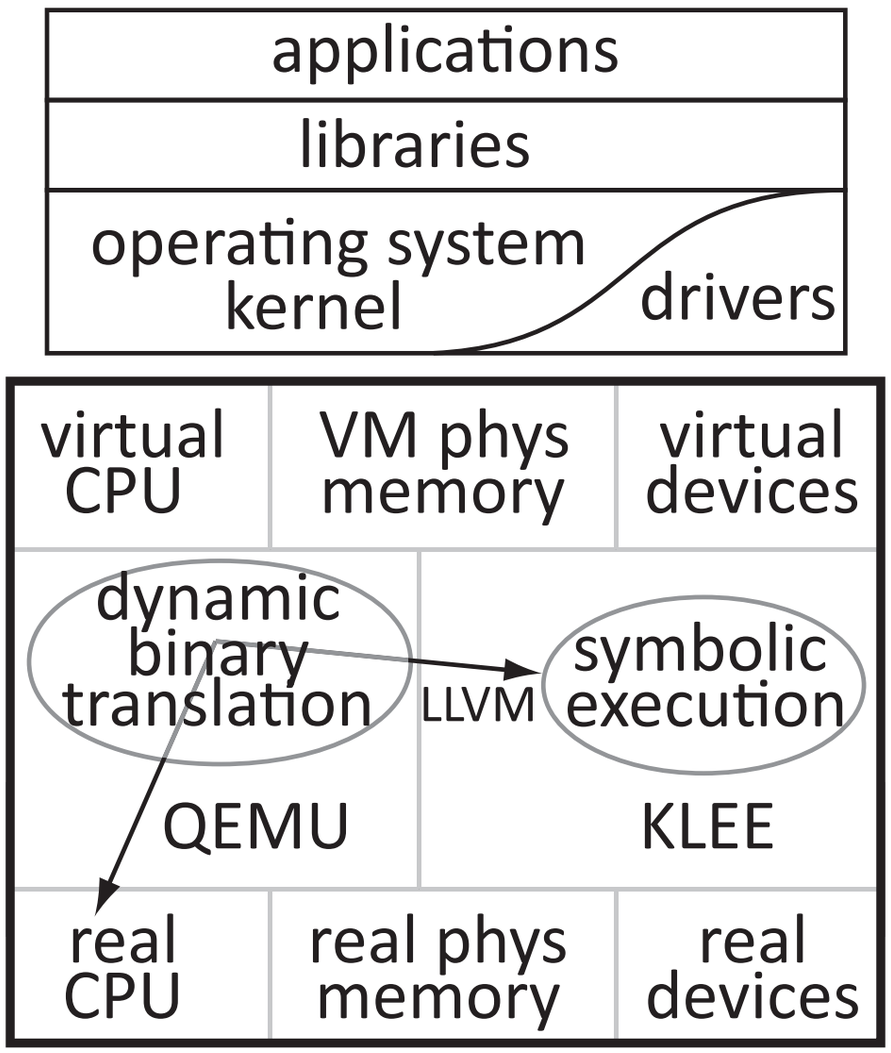
Runs unmodified x86 binaries (including proprietary/obfuscated/ encrypted binaries)

S2E Is A Virtual Machine



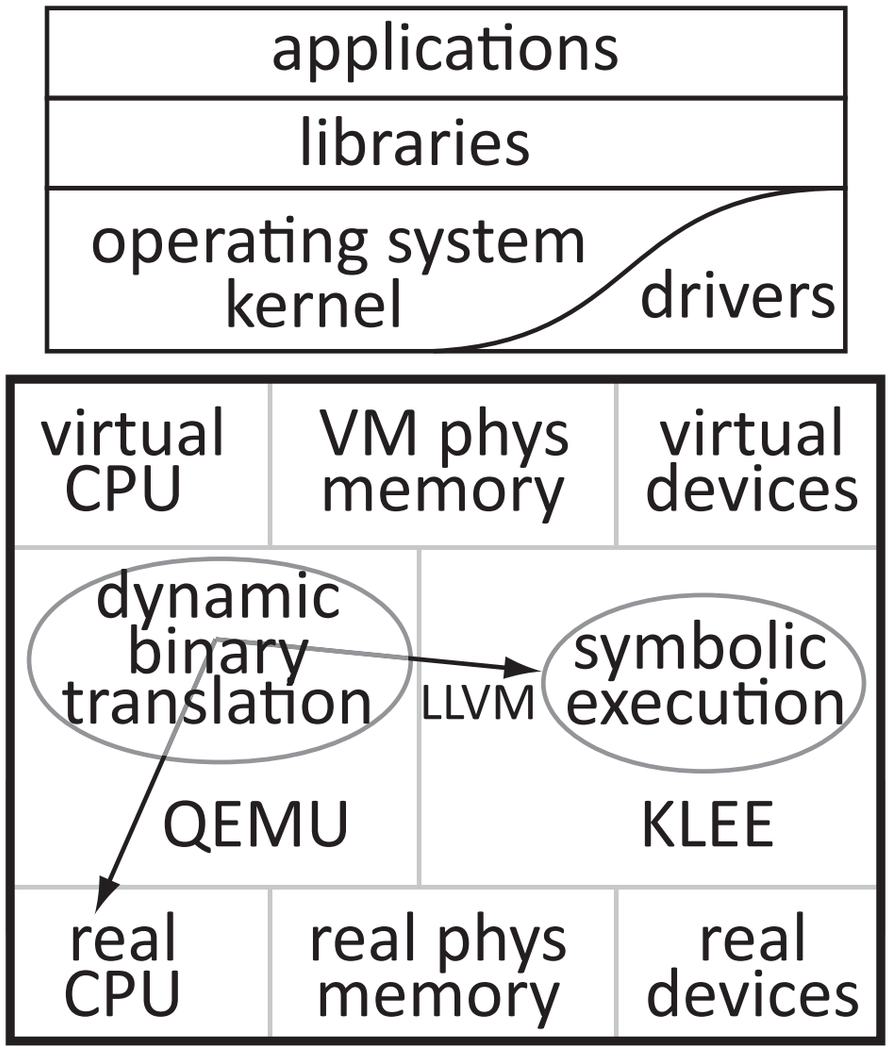
Selection done at runtime
Most code runs “natively”

S2E Is A Virtual Machine

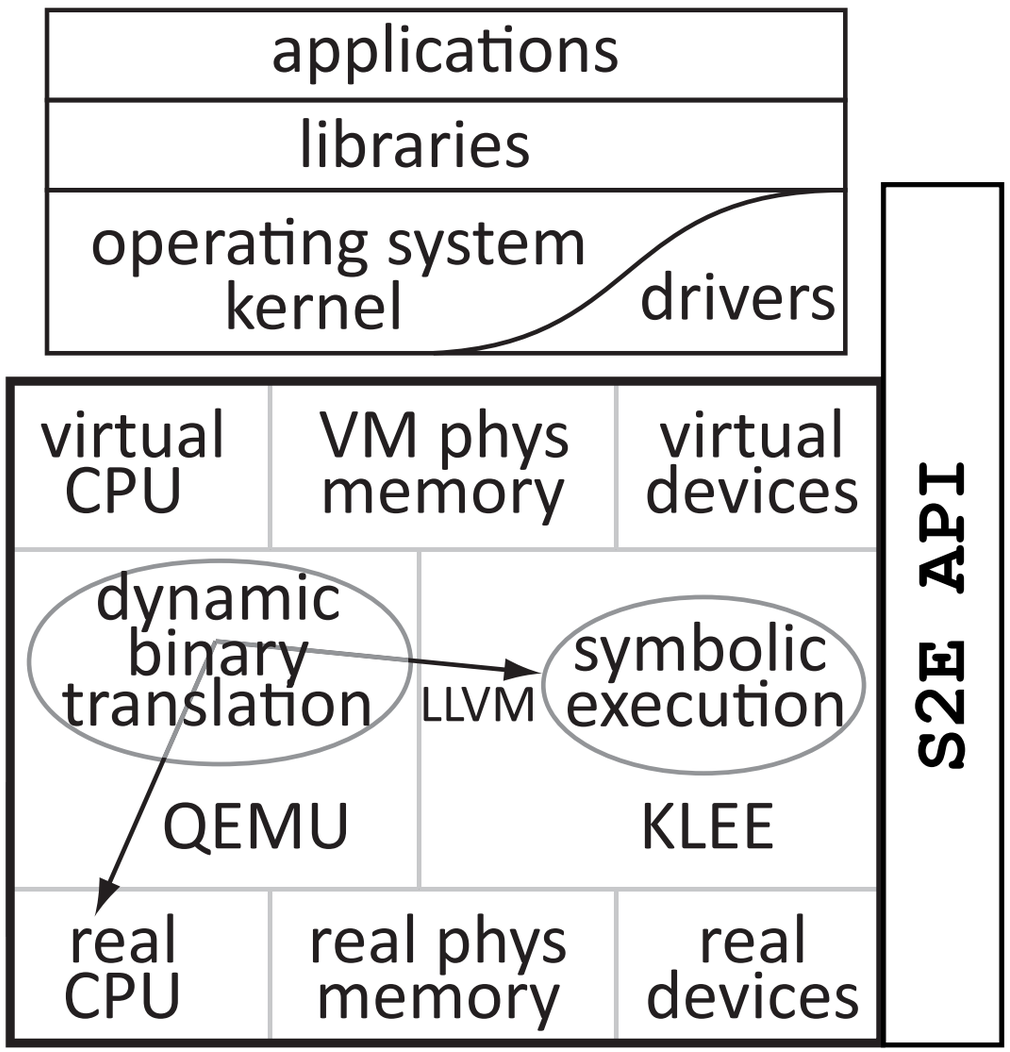


Shared concrete/symbolic state representation

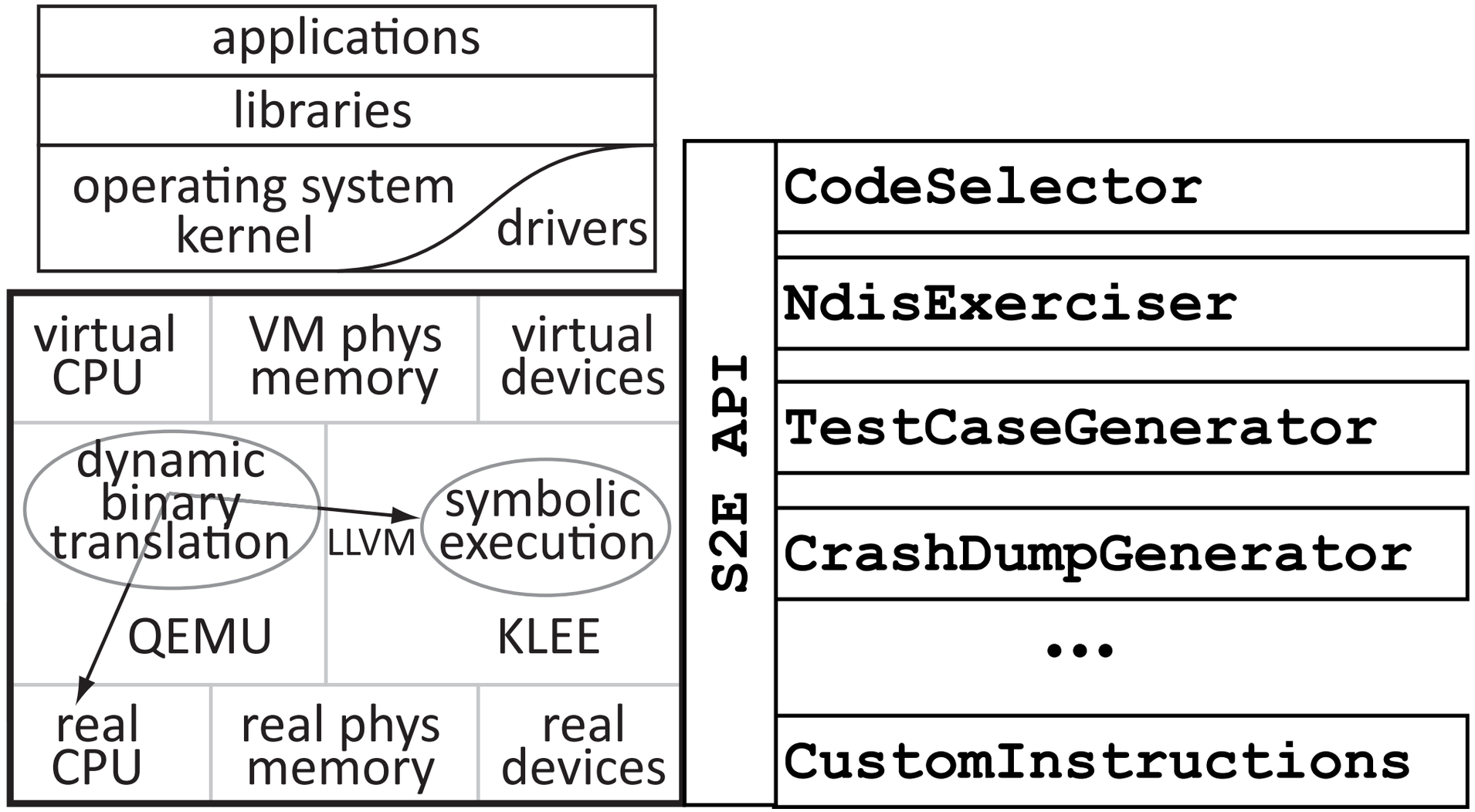
Plugin-based Architecture



Plugin-based Architecture



Plugin-based Architecture



Outline

- Theory
  Execution consistency models
- System
  S²E: Platform for in-vivo multi-path analysis
- Results
  Using S²E in practice

http://s2e.epfl.ch

Outline

- Theory
  Execution consistency models
- System
  S²E: Platform for in-vivo multi-path analysis
- Results
  Using S²E in practice

http://s2e.epfl.ch

Using S2E in Practice

- Automated Device Driver Testing
  DDT
- Automated Reverse Engineering
  RevNIC
- Multi-path Performance Profiling
  PROFs

Using S2E in Practice

- Automated Device Driver Testing
  DDT
- Automated Reverse Engineering
  RevNIC
- Multi-path Performance Profiling
  PROFs

A problem has been detected and windows has been shut down to prevent damage
to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

Driver Bugs are Critical

- Lead to system-level malfunction
  Crashes, security issues

- Driver bugs are ubiquitous
  85% of Windows crashes[1]
  3-7 times buggier than the kernel[2]

[1] V. Orgovan and M. Tricker. *An introduction to driver quality.* Microsoft Windows Hardware Engineering Conf., 2003.
[2] A. Chou, et al. *An empirical study of operating systems errors.* In SOSP 2001.

State of the Art in Driver Testing

State of the Art in Driver Testing

	Dynamic	**Static**
Closed-source drivers		
Unmodified environment		
Works without hardware		
Multi-path		

State of the Art in Driver Testing

	Dynamic	**Static**
Closed-source drivers	✔	
Unmodified environment	✔	
Works without hardware	✖	
Multi-path	✖	

State of the Art in Driver Testing

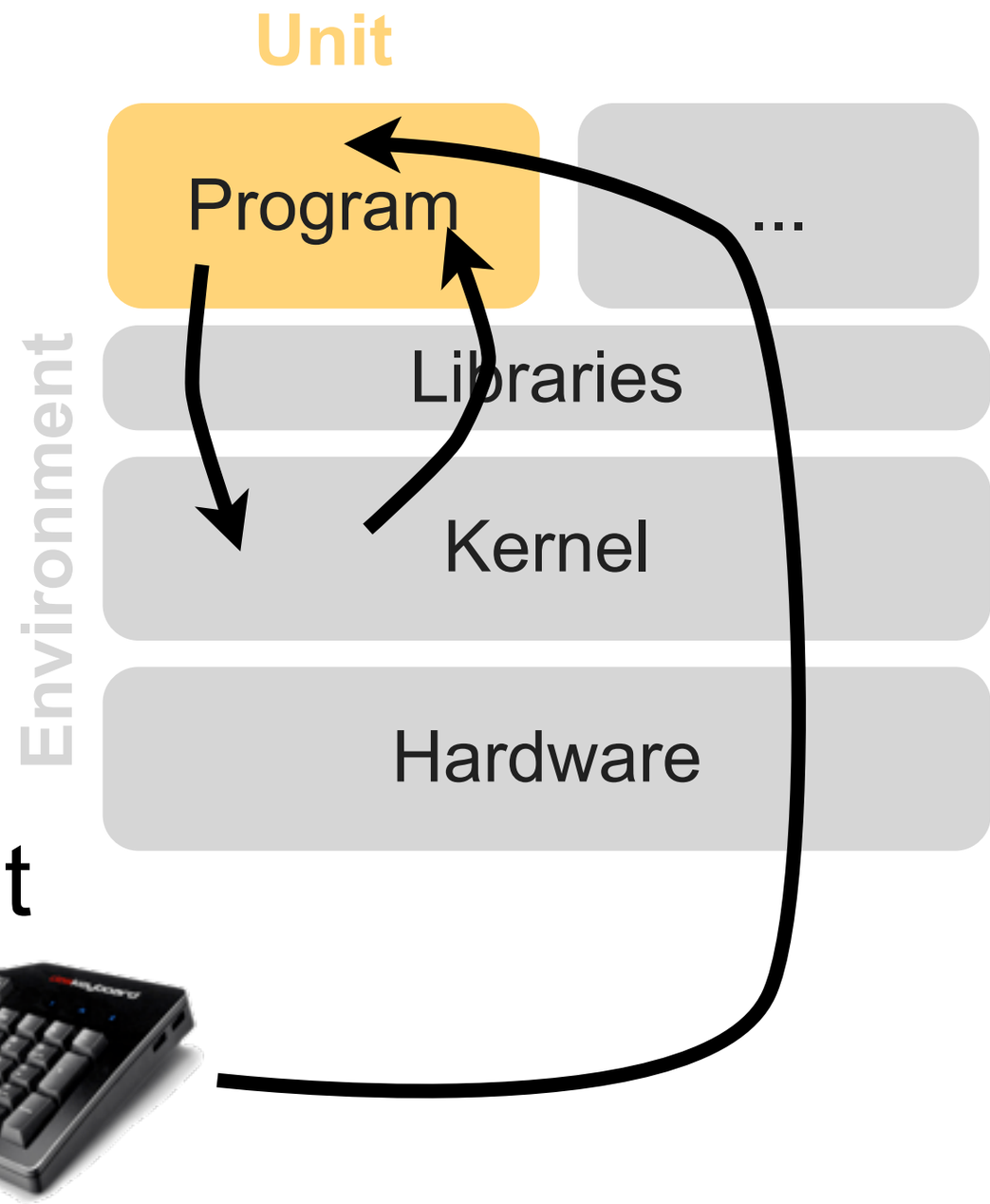
	Dynamic	**Static**
Closed-source drivers	✔	✖
Unmodified environment	✔	✖
Works without hardware	✖	✔
Multi-path	✖	✔

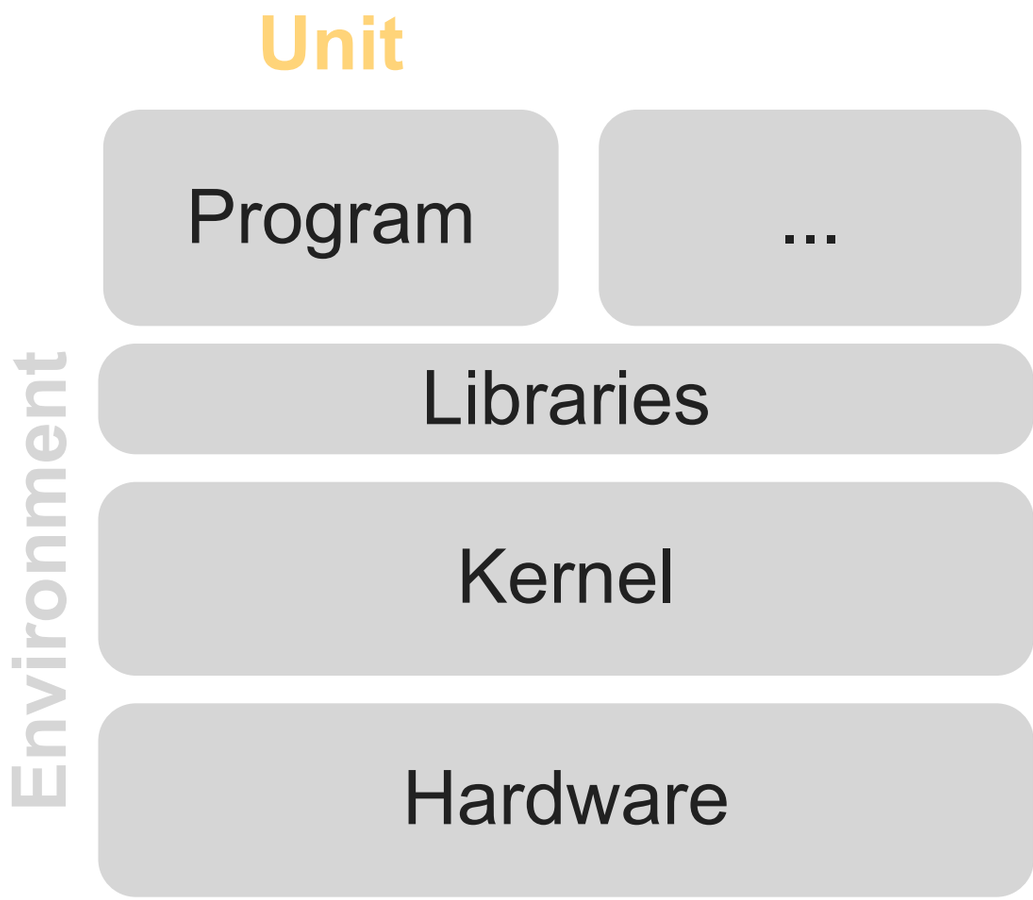
State of the Art in Driver Testing

	Dynamic	**Static**	**S2E/DDT**
Closed-source drivers	✔	✖	✔
Unmodified environment	✔	✖	✔
Works without hardware	✖	✔	✔
Multi-path	✖	✔	✔

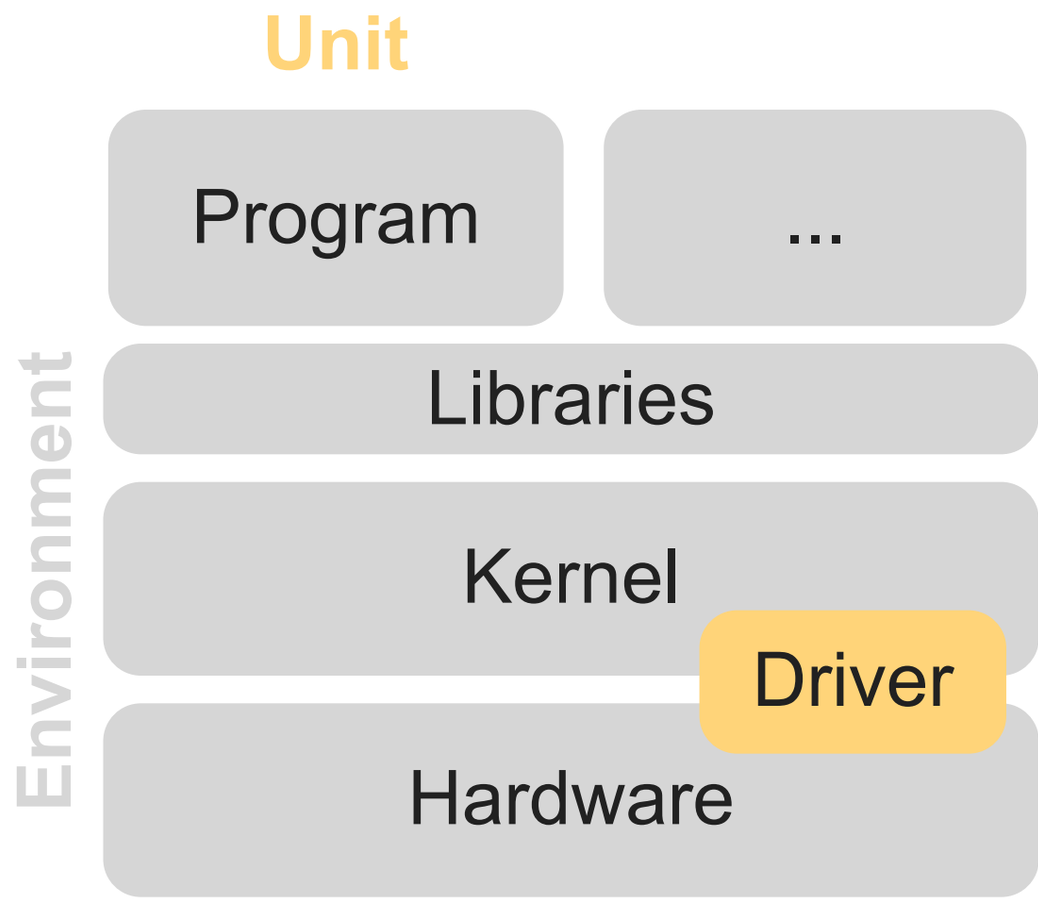
Consistency Models in S2E

```
int main(argc, argv) {

  if (argc == 0) {
     ...
  }

  p = malloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```

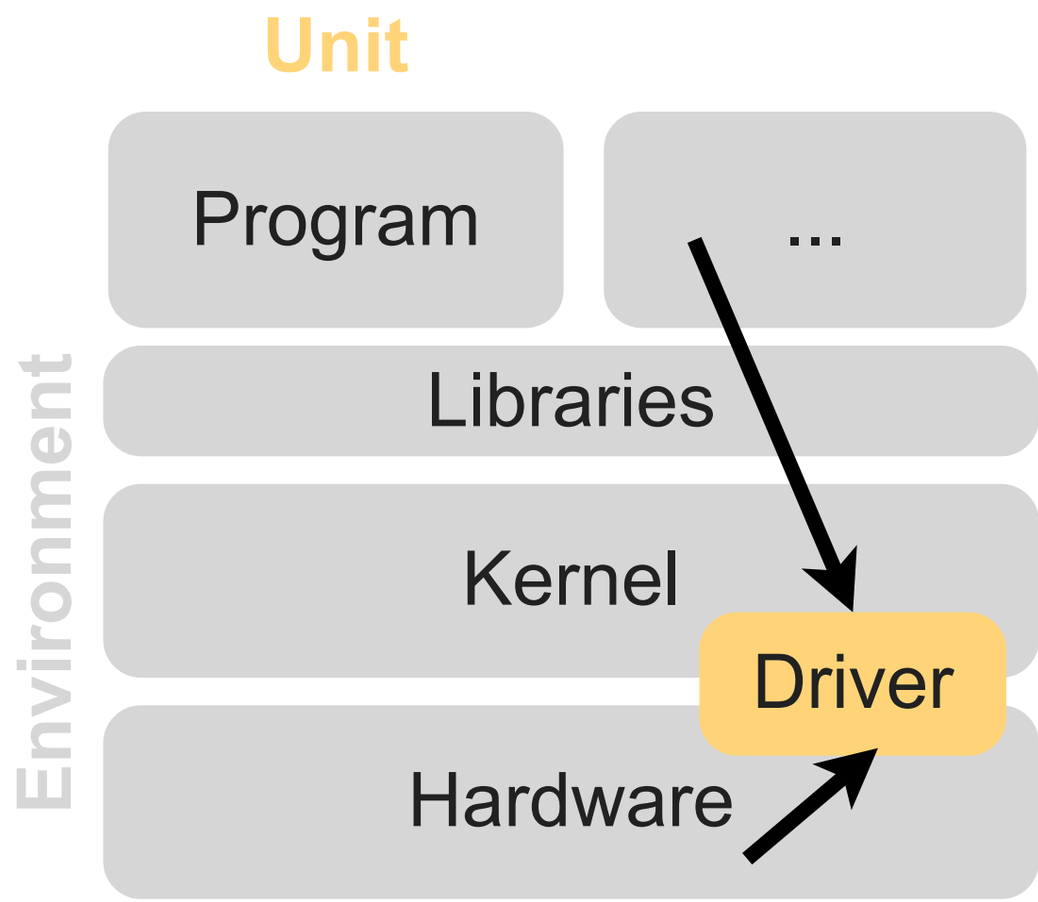



Unit
Program
...
Libraries
Kernel
Hardware
Environment

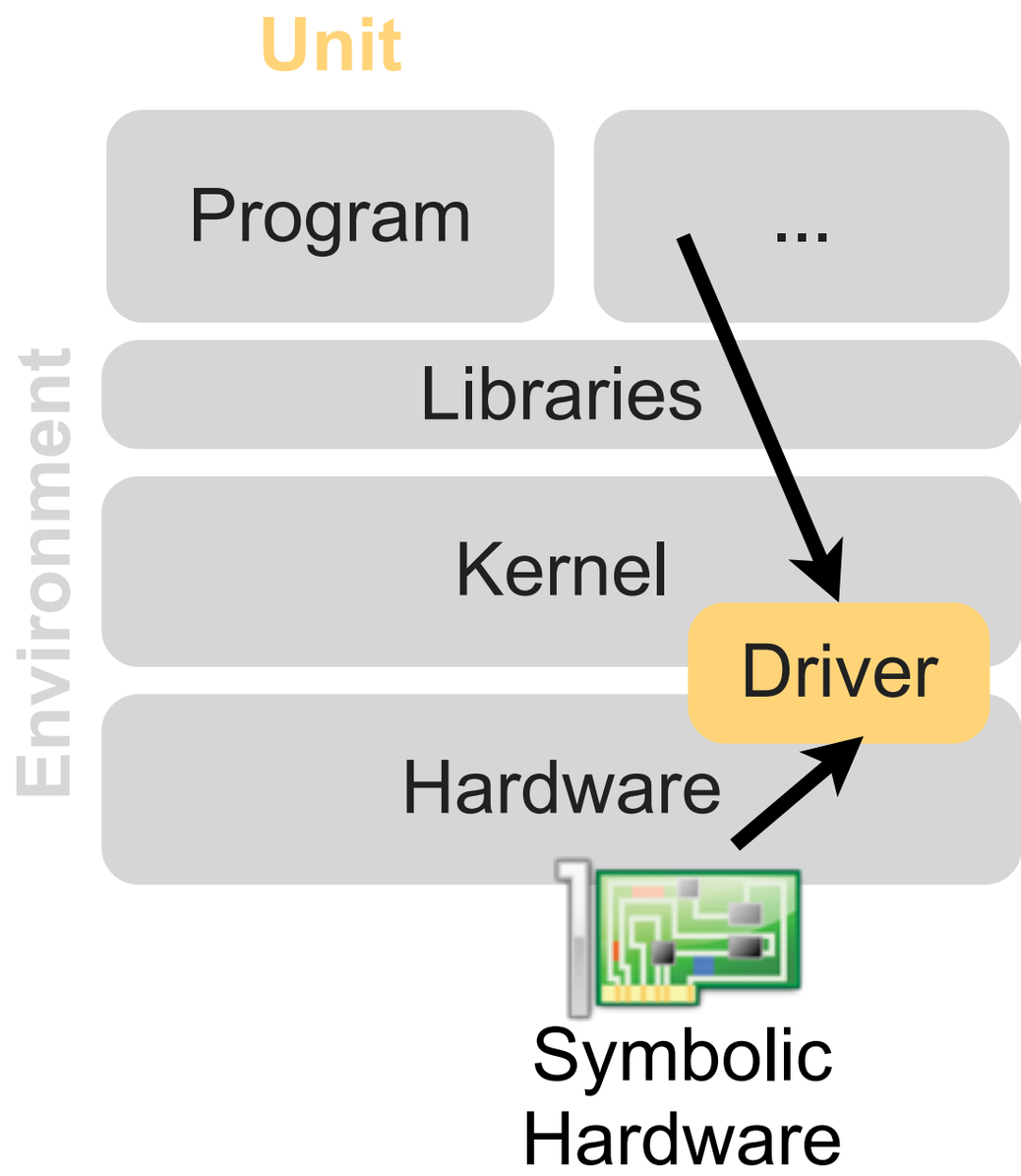
```
int open(...) {

   if (fd == 0) {
       ...
   }

   p = kmalloc(...);

   if (p == NULL) {
       ...
   }
   ...
}
```

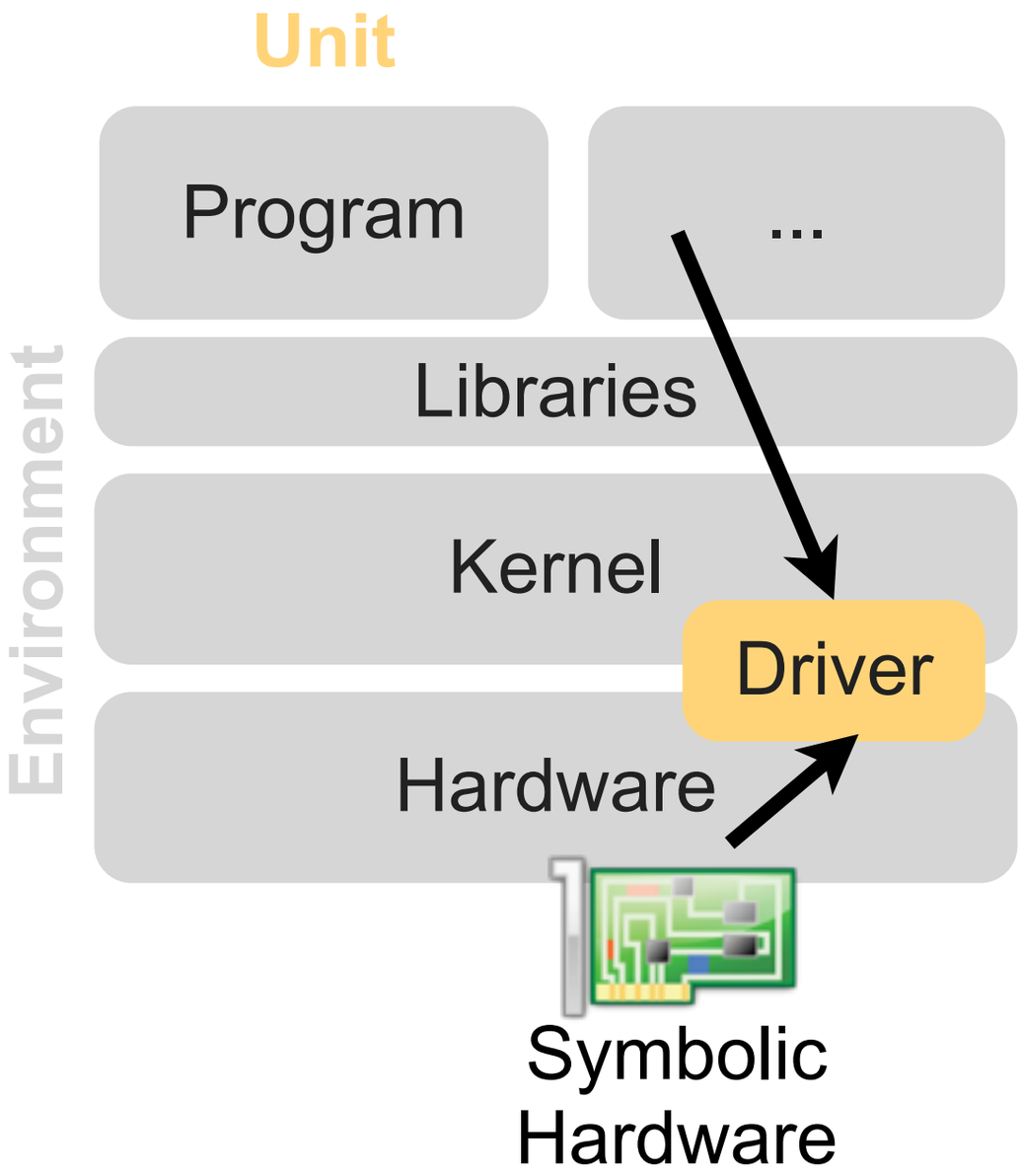


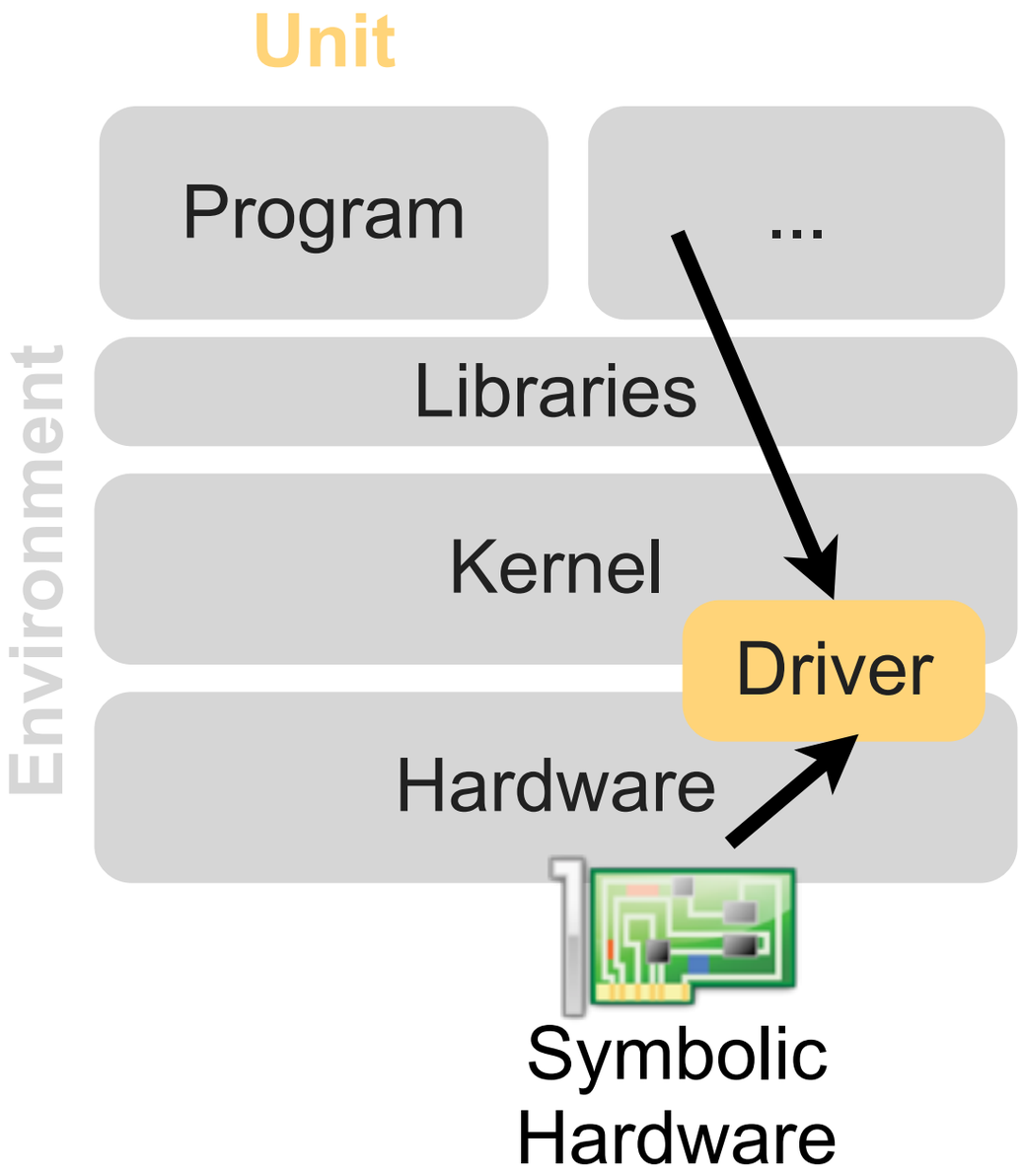
```
int open(...) {

  if (fd == 0) {
      ...
  }

  p = kmalloc(...);

  if (p == NULL) {
      ...
  }
  ...
}
```

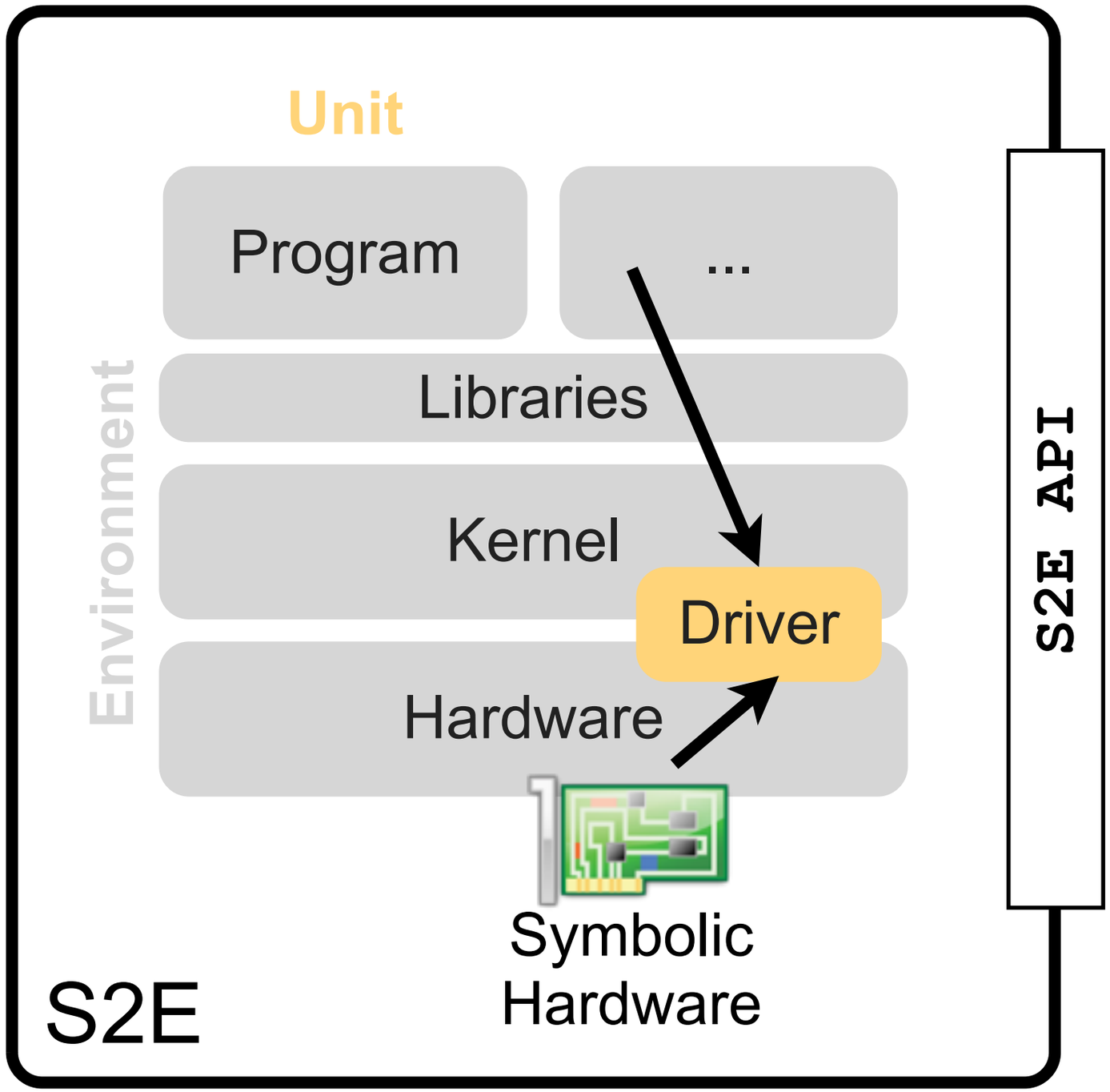


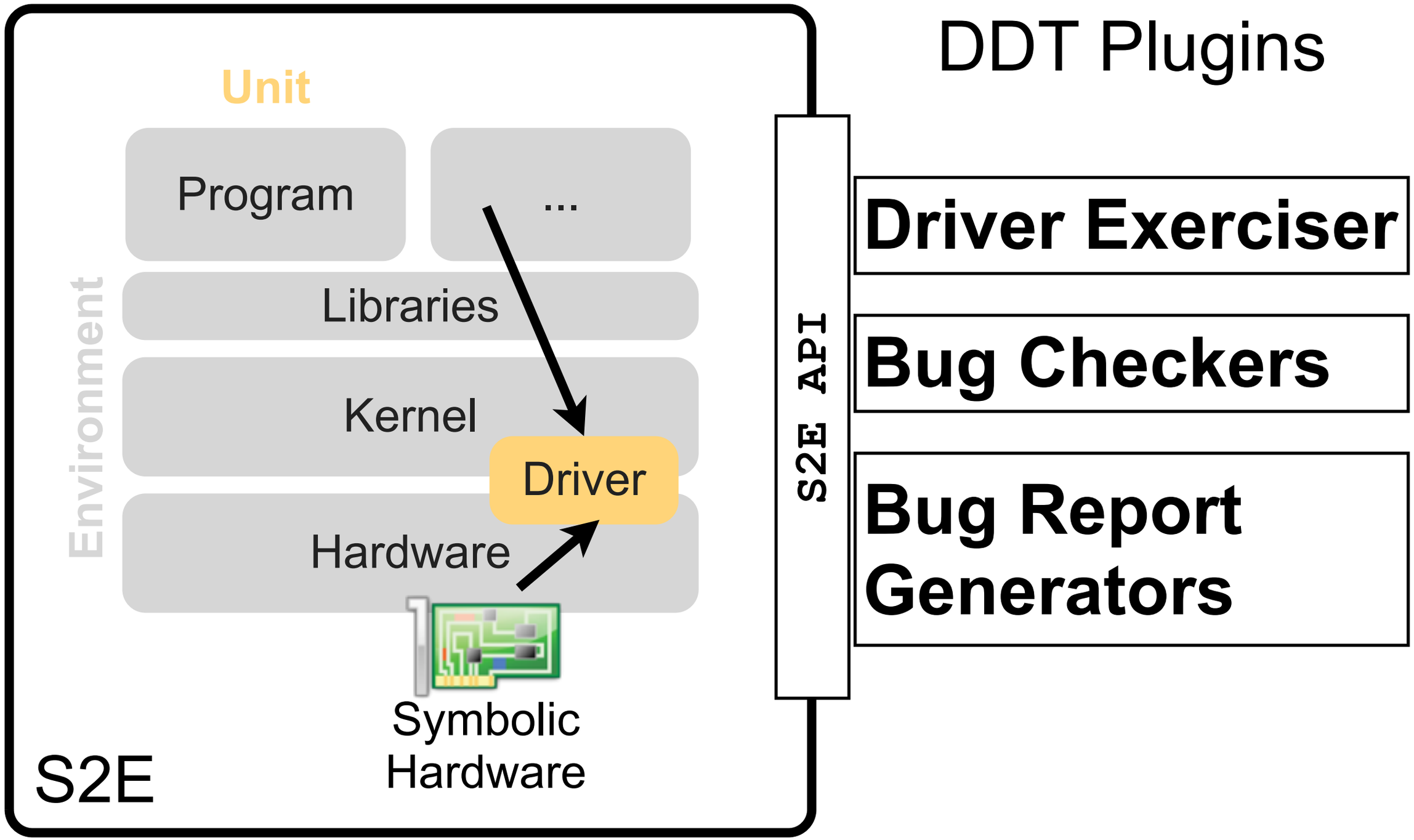
```
int open(...) {

  if (fd == 0) {
     ...
  }

  p = kmalloc(...);

  if (p == NULL) {
     ...
  }
  ...
}
```

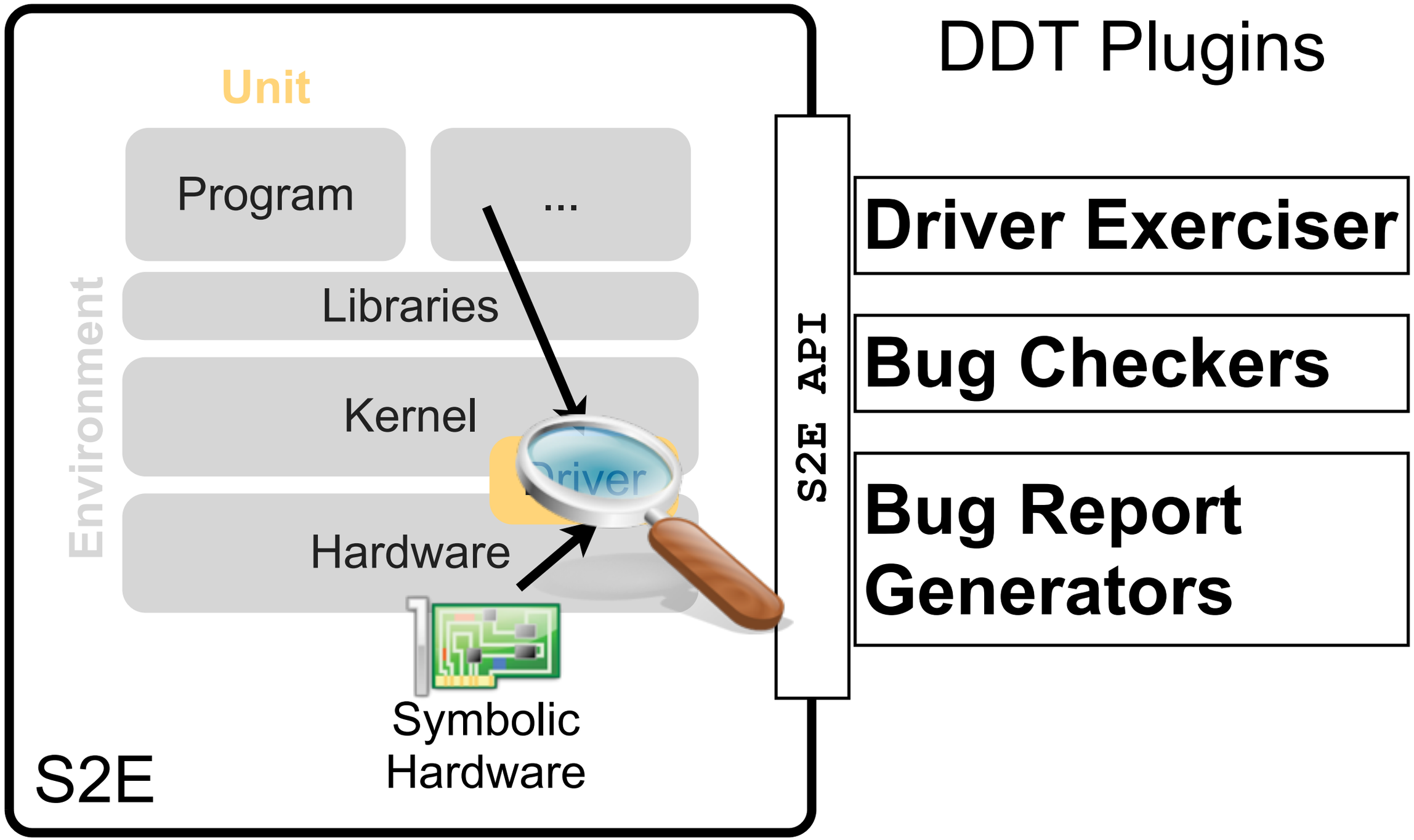



Unit
Program
...
Libraries
Kernel
Driver
Hardware
Environment
Symbolic
Hardware


Unit
Program
...
Libraries
Kernel
Driver
Hardware
Environment
Symbolic
Hardware


Unit
Program
...
Libraries
Kernel
Driver
Hardware
Environment
Symbolic
Hardware
S2E
S2E API


Unit
Program
...
Libraries
Kernel
Driver
Hardware
Symbolic Hardware
Environment
S2E
S2E API
DDT Plugins
Driver Exerciser
Bug Checkers
Bug Report Generators

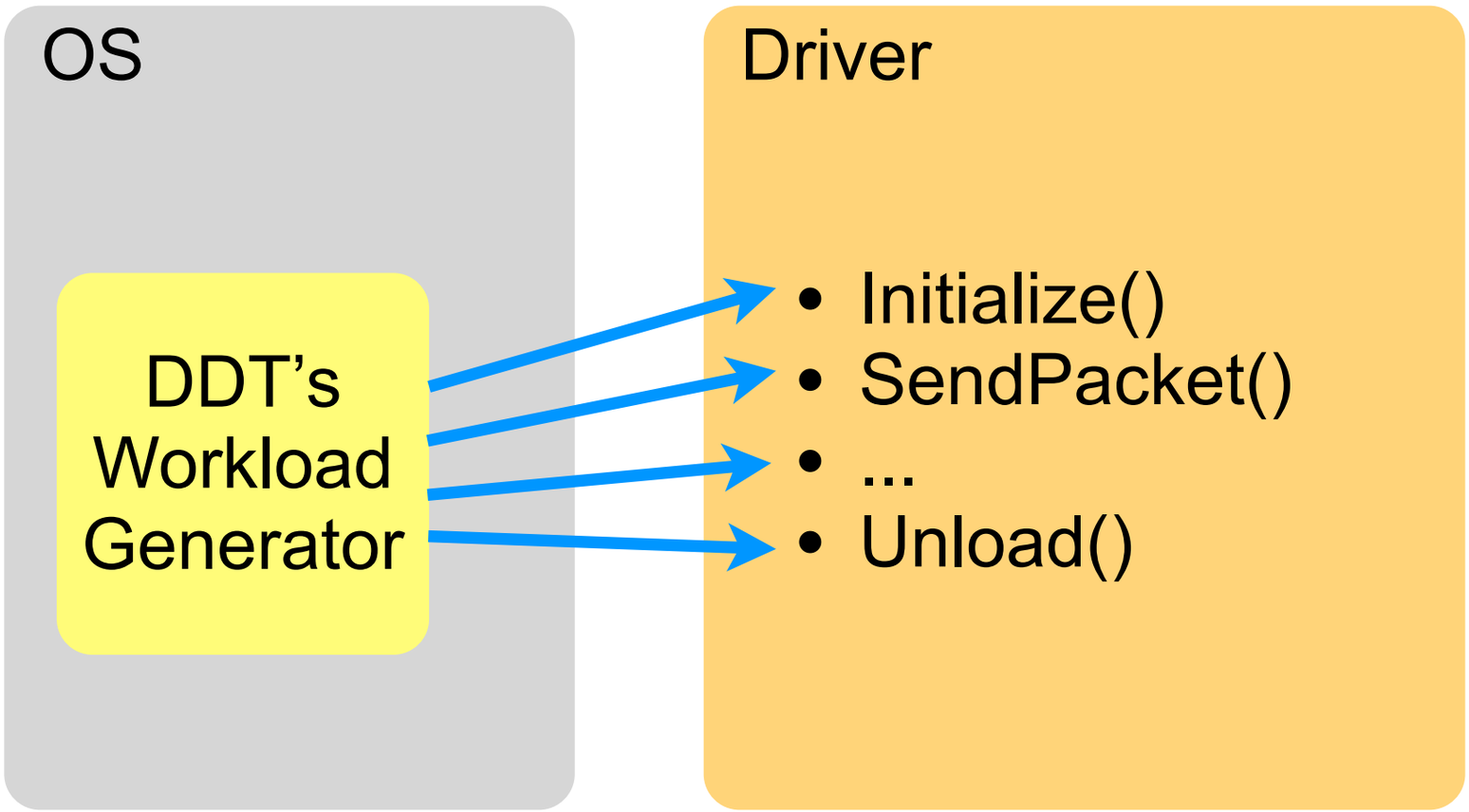

DDT Plugins
Unit
Program
...
Libraries
Kernel
Driver
Hardware
Environment
Symbolic
Hardware
S2E
API
S2E Plugins
Driver Exerciser
Bug Checkers
Bug Report
Generators

Drivers

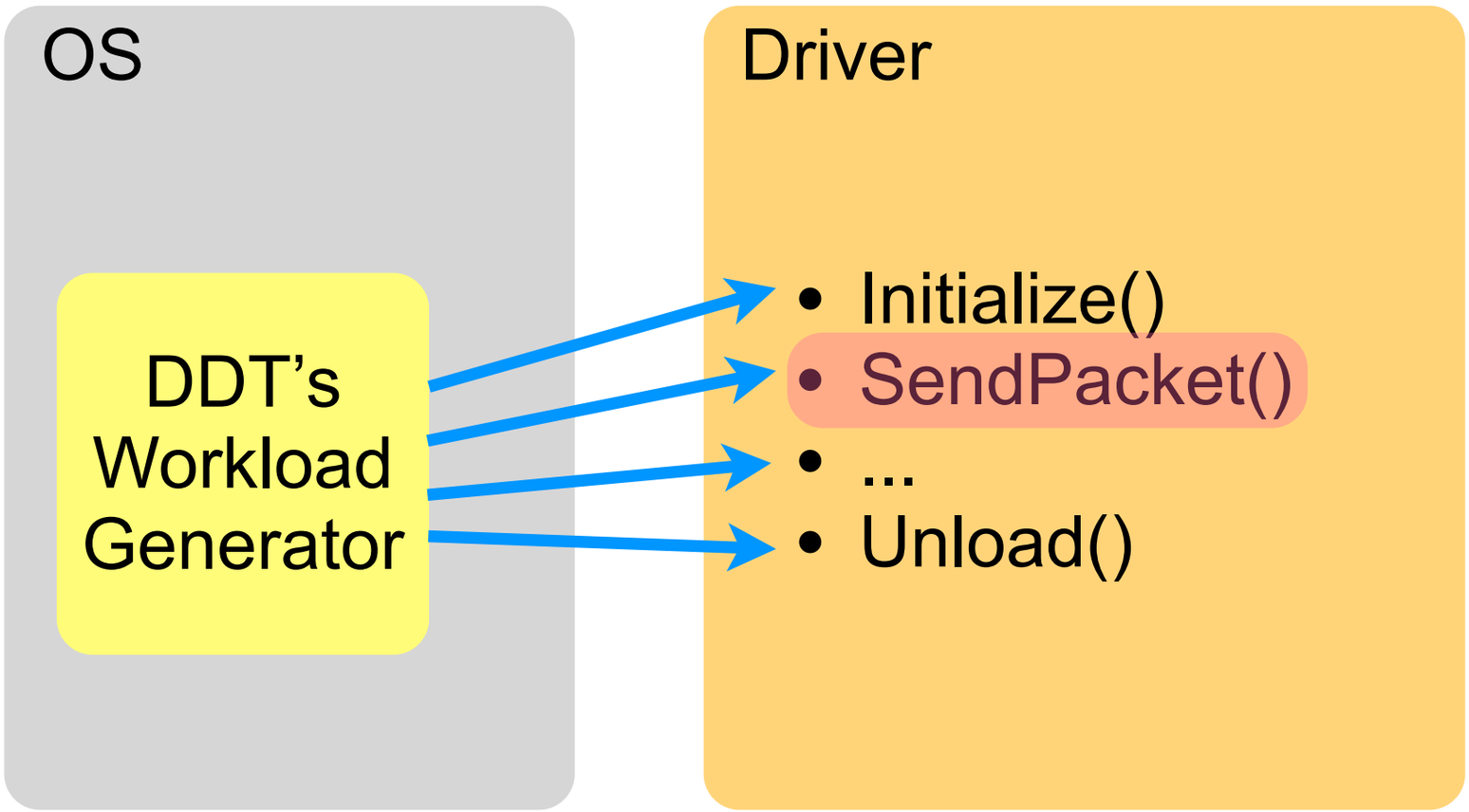
Driver

- Initialize()
- SendPacket()
- ...
- Unload()

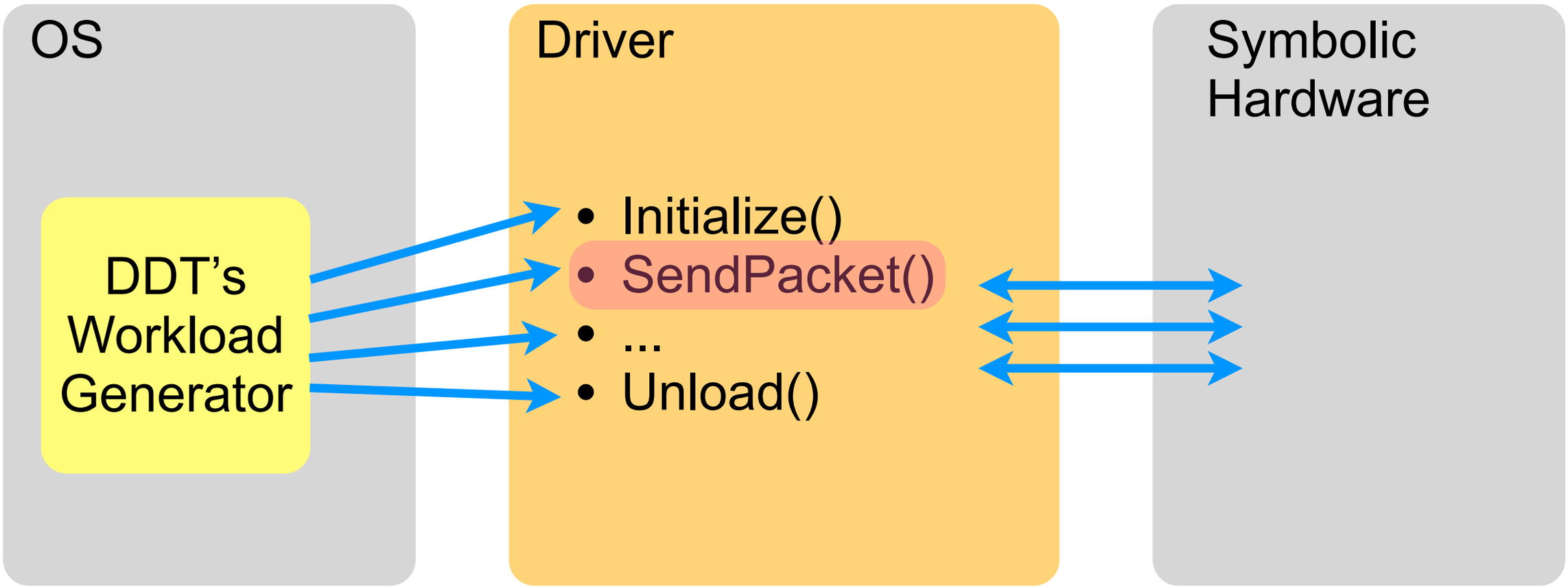
Drivers


OS
DDT's Workload Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()

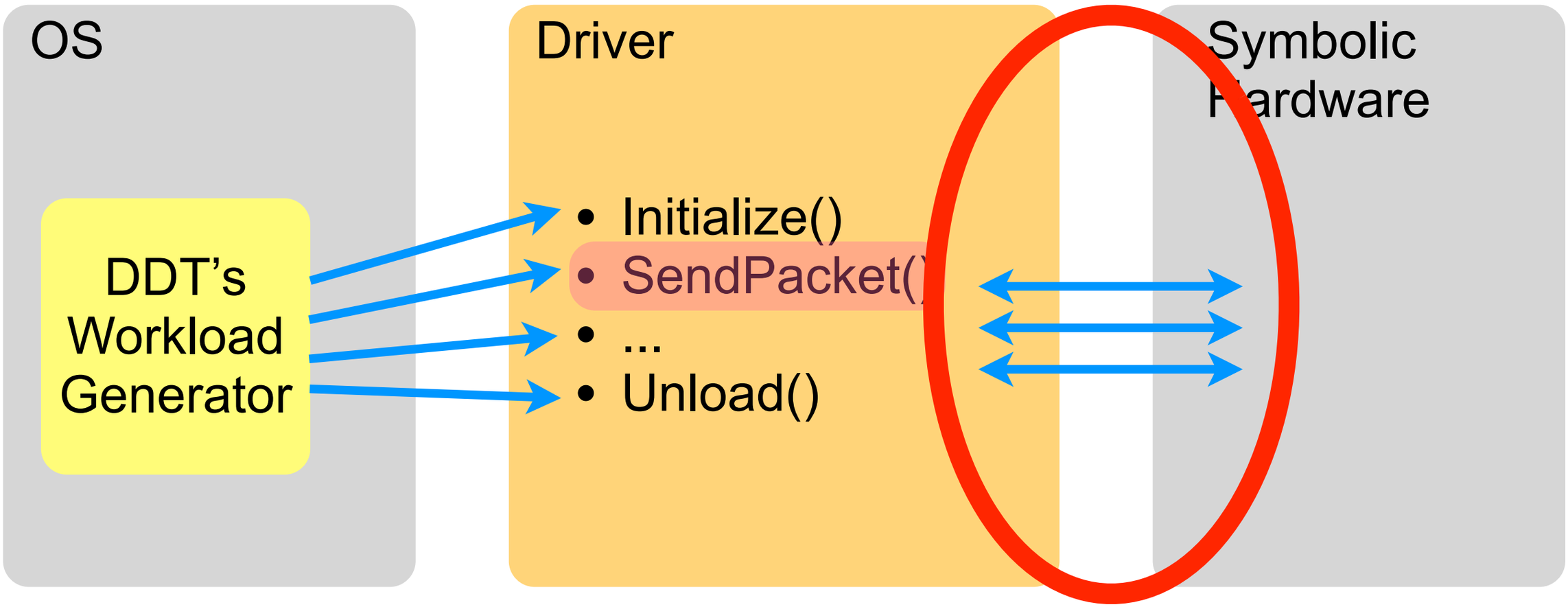
Drivers


OS
DDT's Workload Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()

Drivers


OS
DDT's Workload Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()
Symbolic Hardware

Drivers

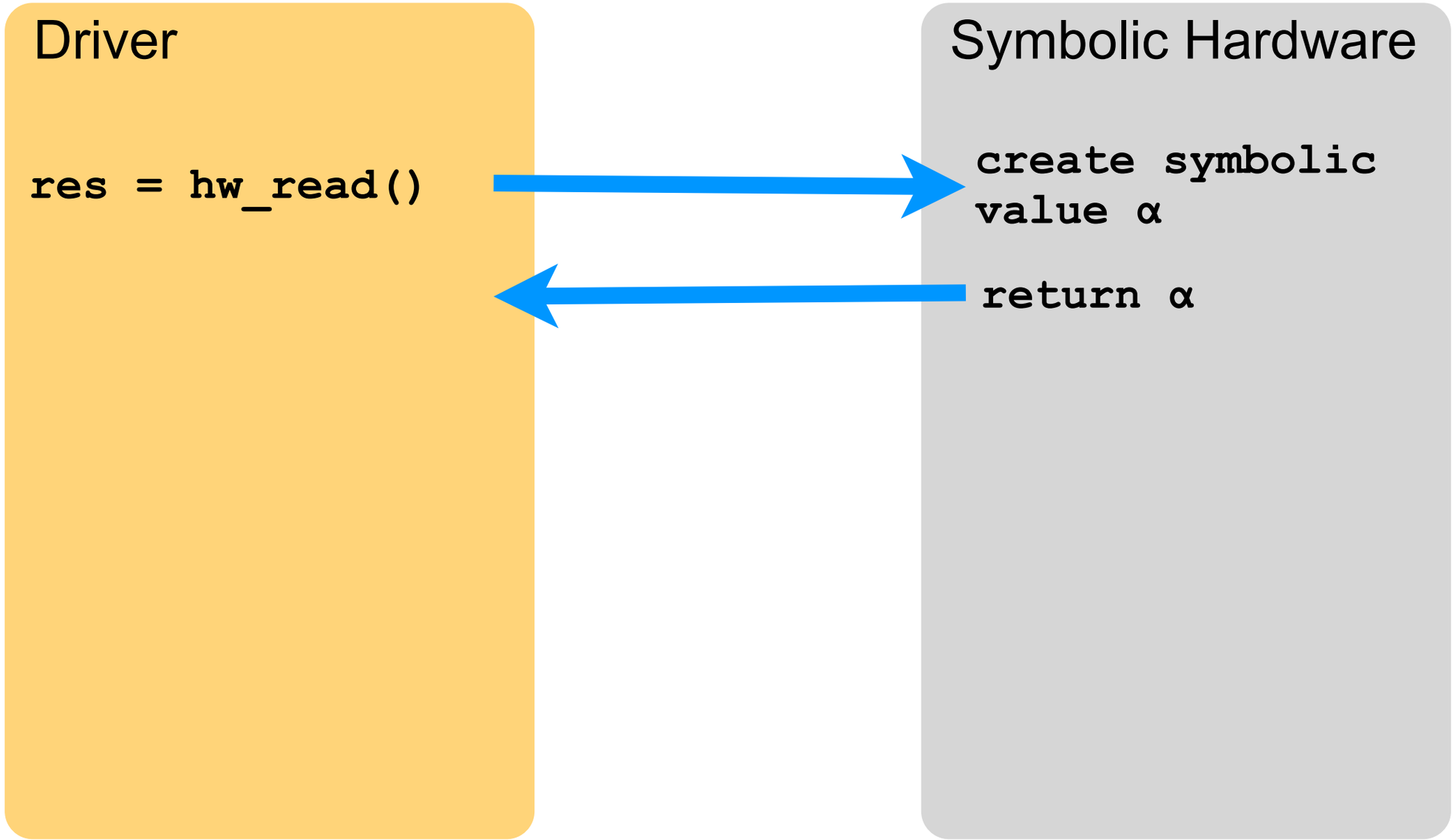

OS
DDT's Workload Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()
Symbolic Hardware

Driver-Hardware Interface

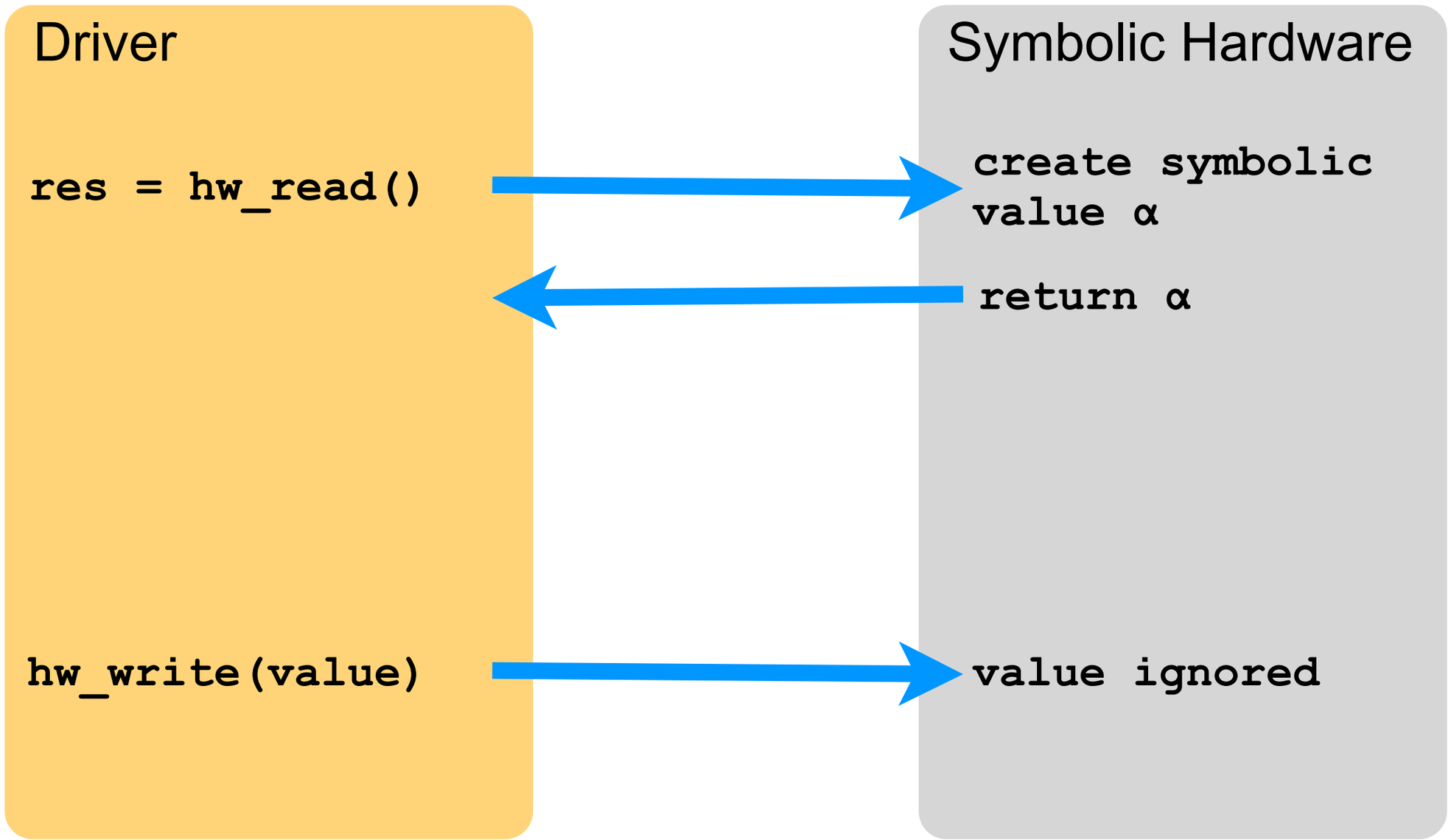
Driver

Symbolic Hardware

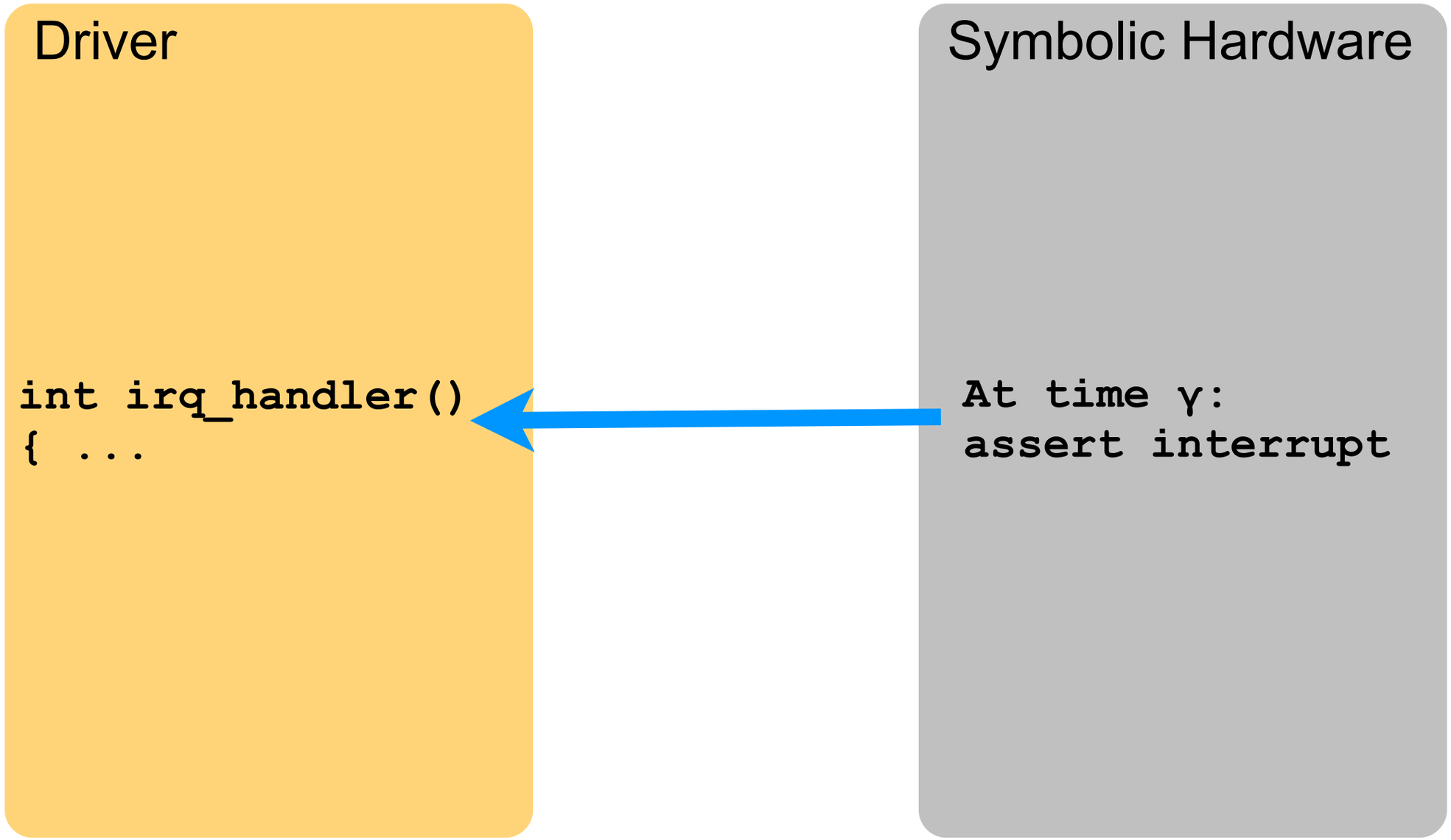
Driver-Hardware Interface



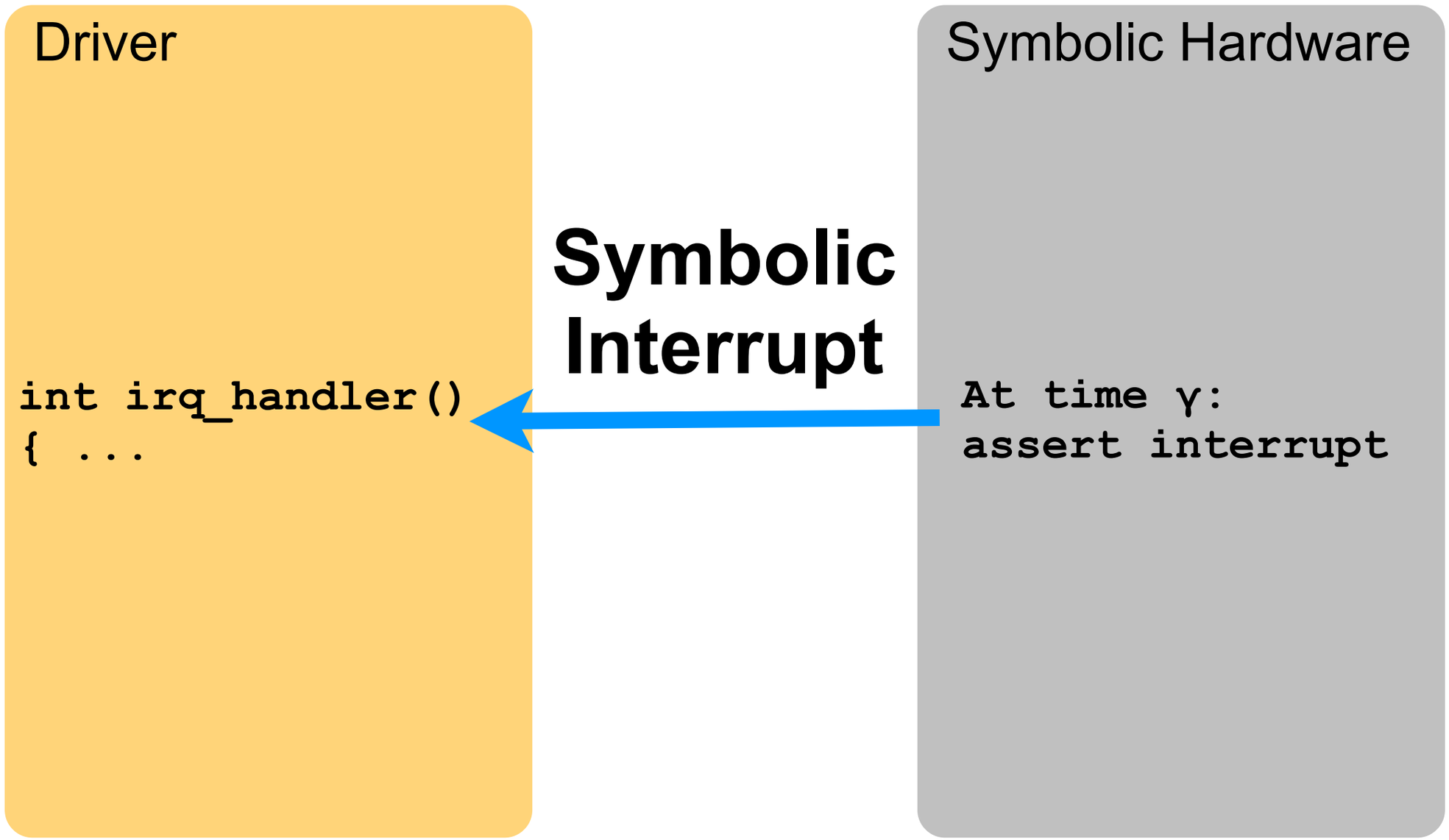
Driver-Hardware Interface



Driver-Hardware Interface

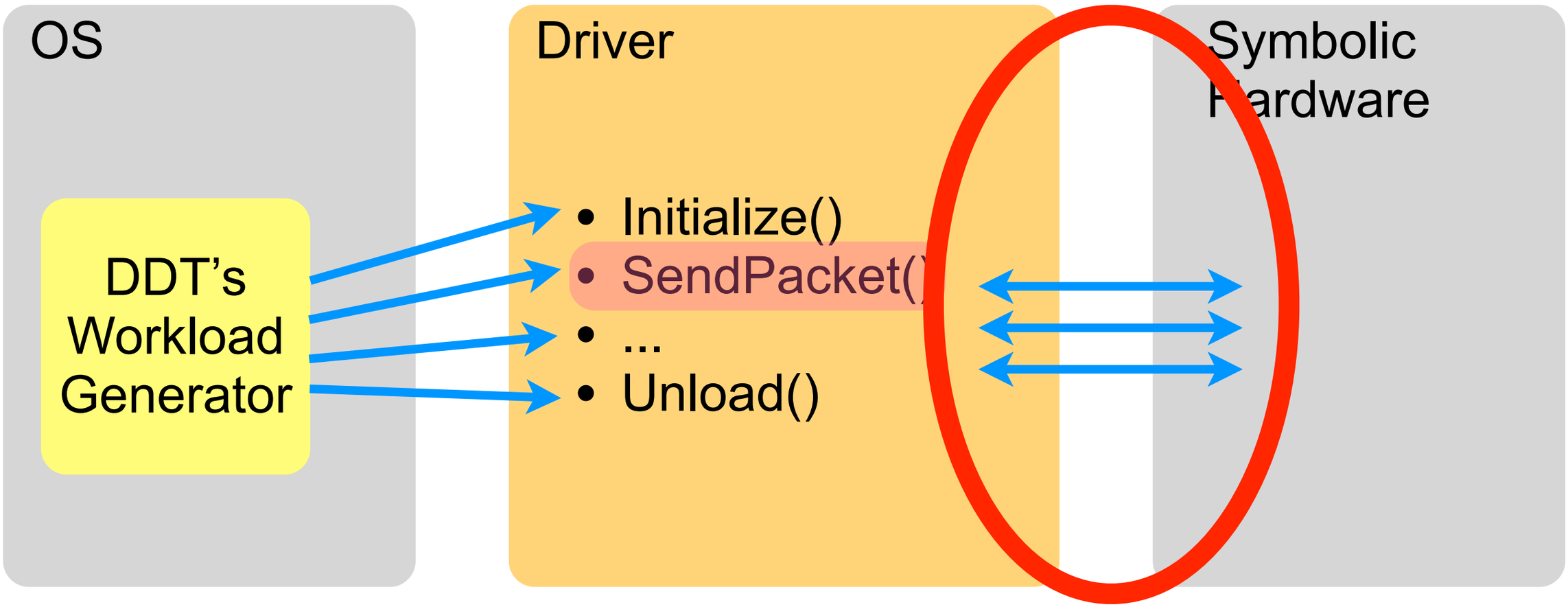


Driver-Hardware Interface

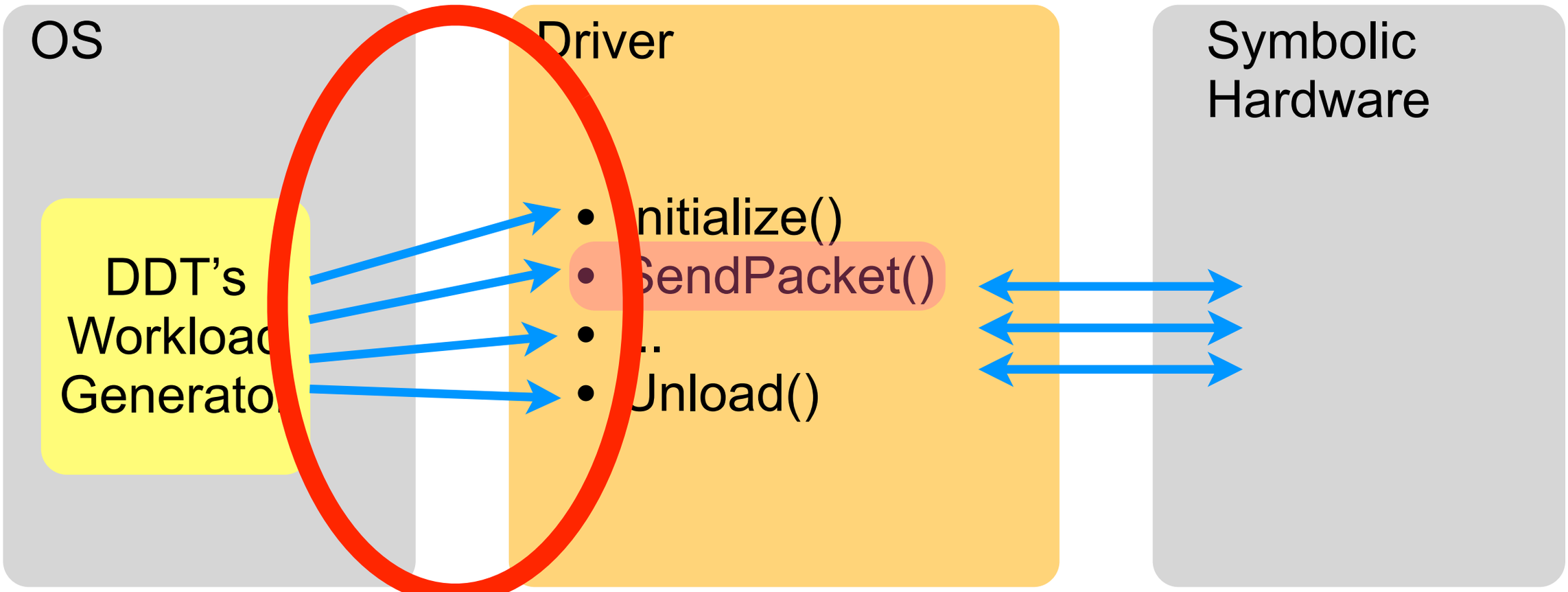


Driver-Hardware Interface

Drivers


OS
DDT's Workload Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()
Symbolic Hardware

Drivers


OS
DDT's
Workload
Generator
Driver
• Initialize()
• SendPacket()
• ...
• Unload()
Symbolic
Hardware

Kernel-Driver Interface

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if (status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```

Kernel-Driver Interface

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if (status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```

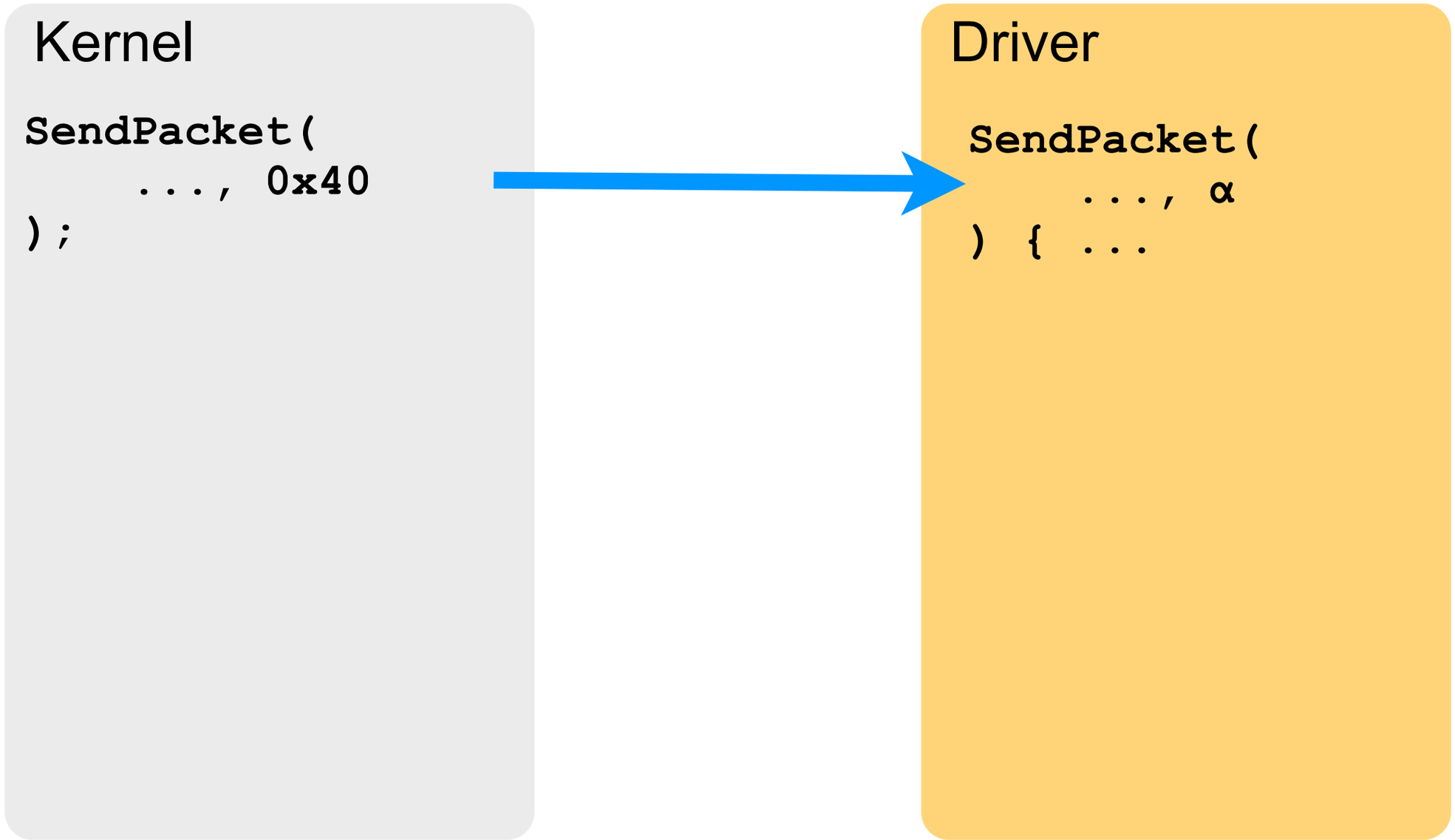
Kernel-Driver Interface

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if (status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```

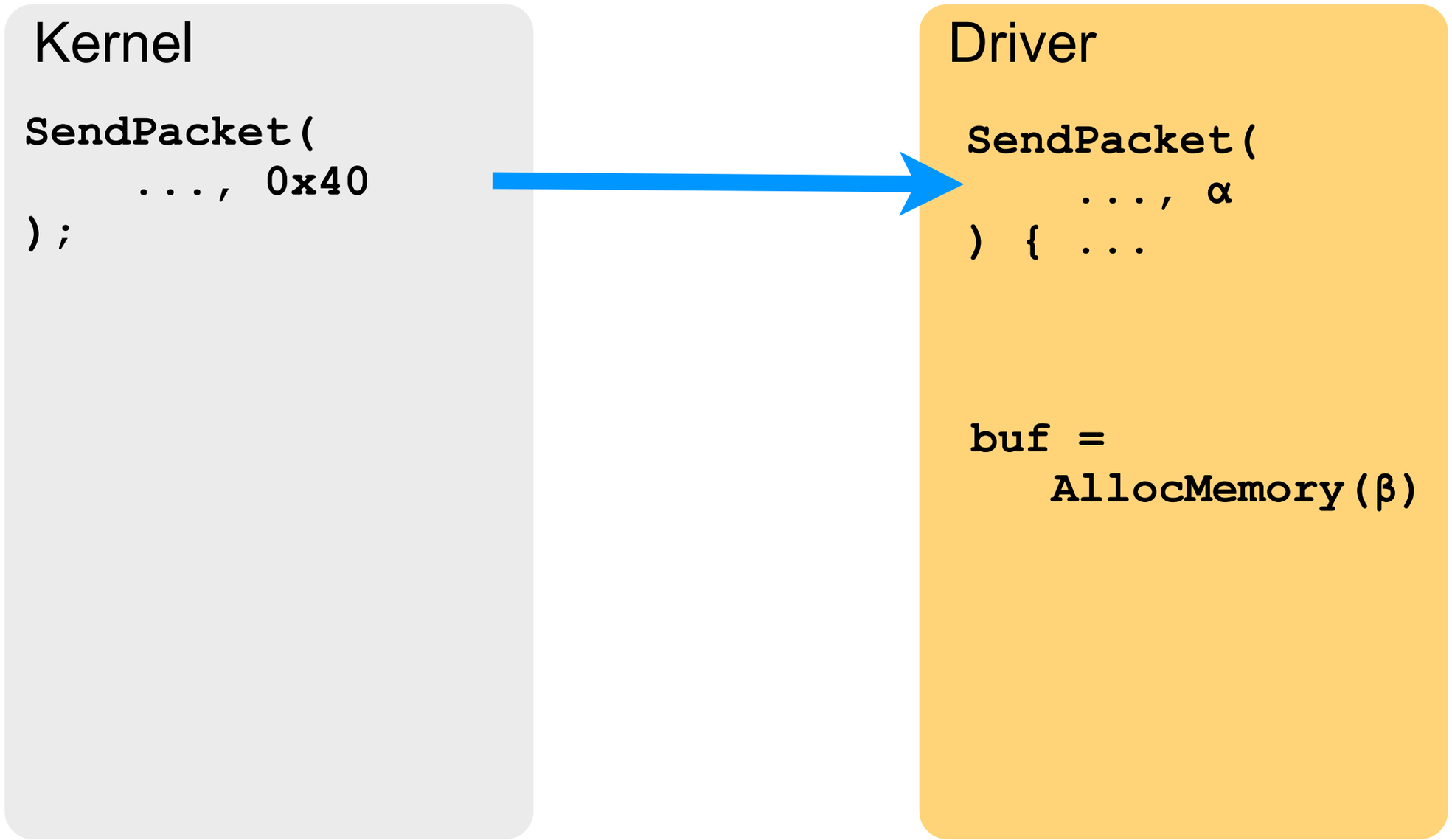
Kernel-Driver Interface

```
void SendPacket(char* packet,
        int size, int flags)
{
  int status = hw_read();

  if (status == READY) {
    send(packet);

  } else {
    if(flags & URGENT) {
      send_urgent(packet);
    } else {
      queue(packet);
    }
  }
}
```

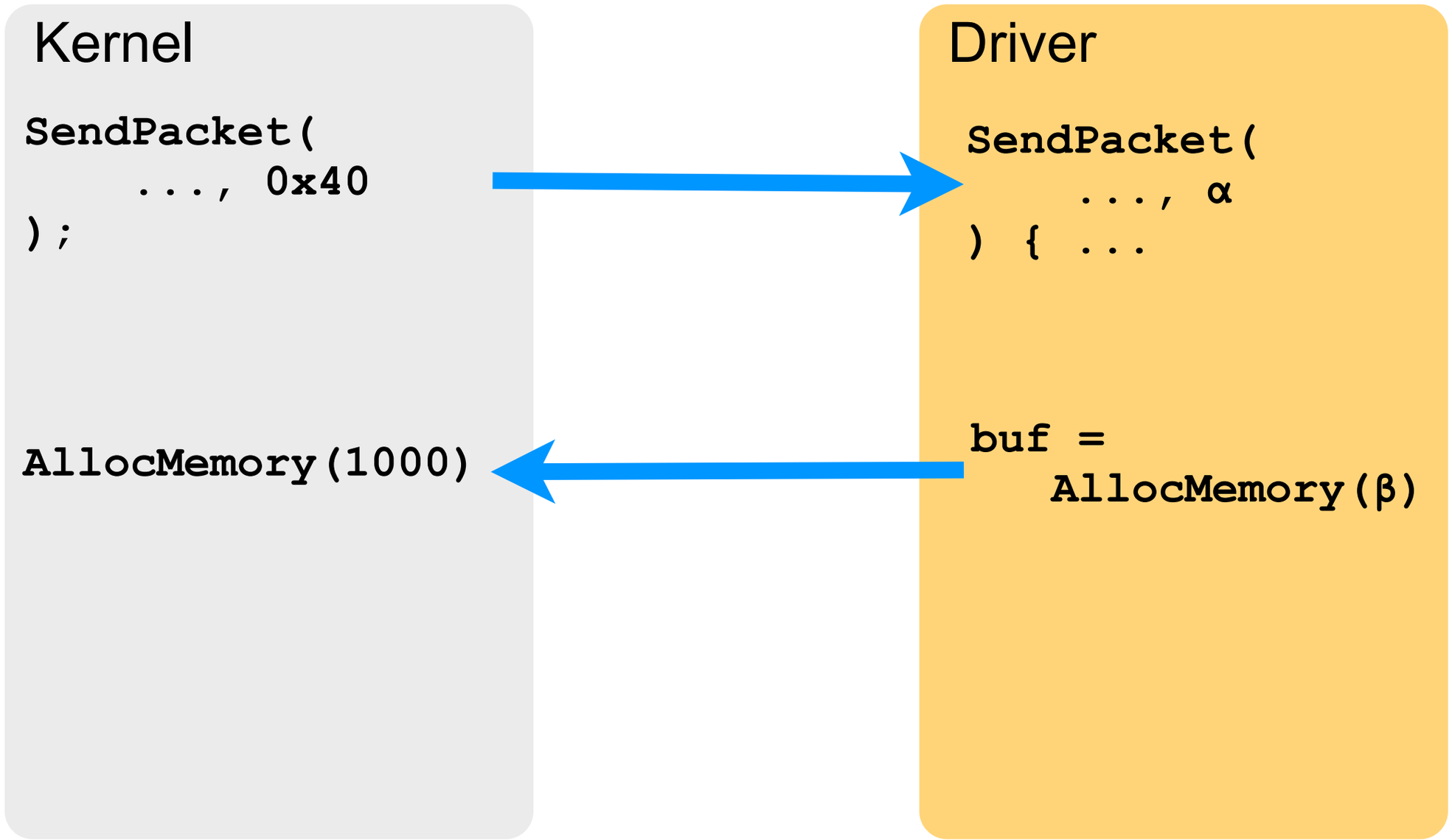
Kernel-Driver Interface



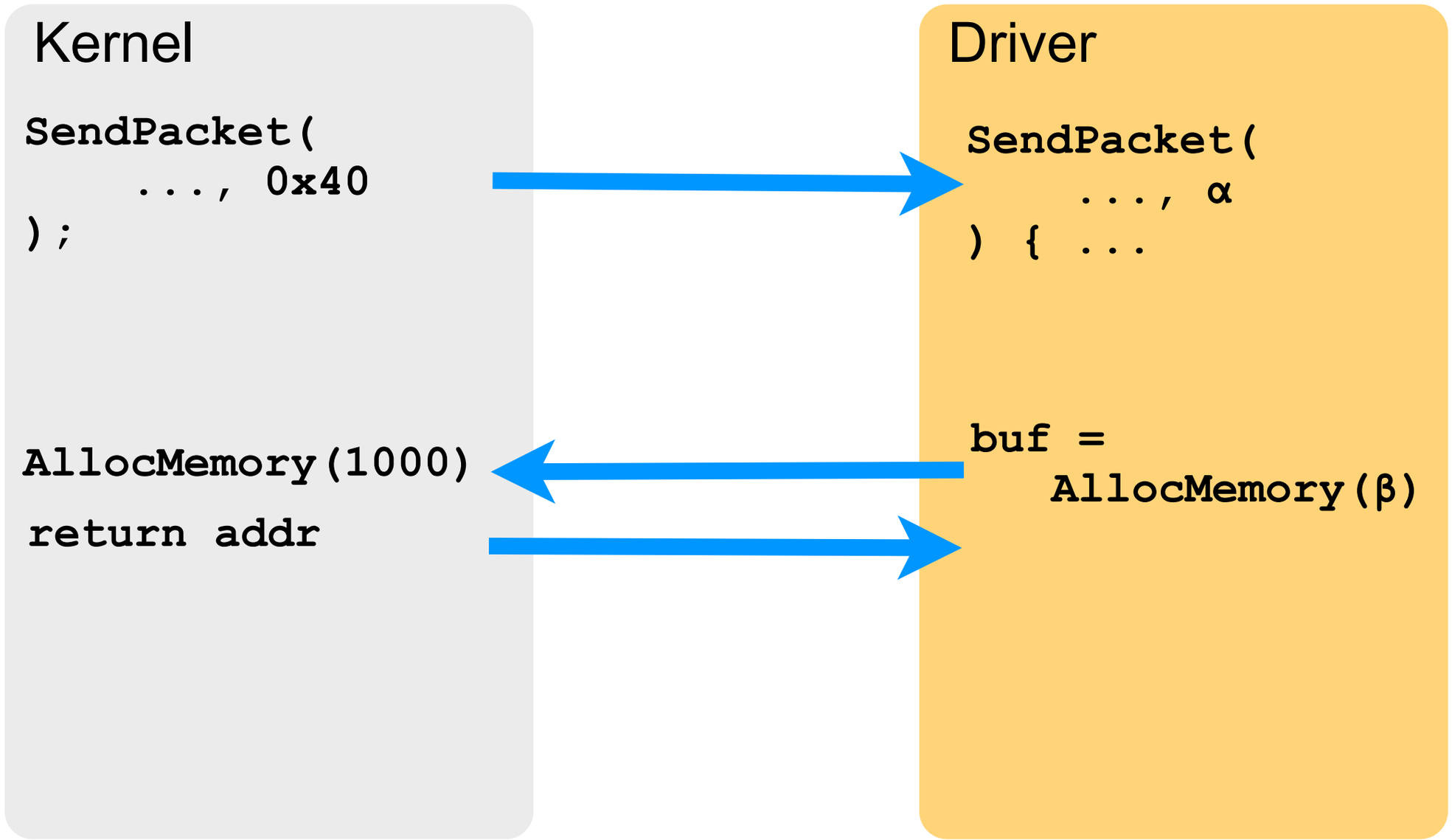
Kernel-Driver Interface



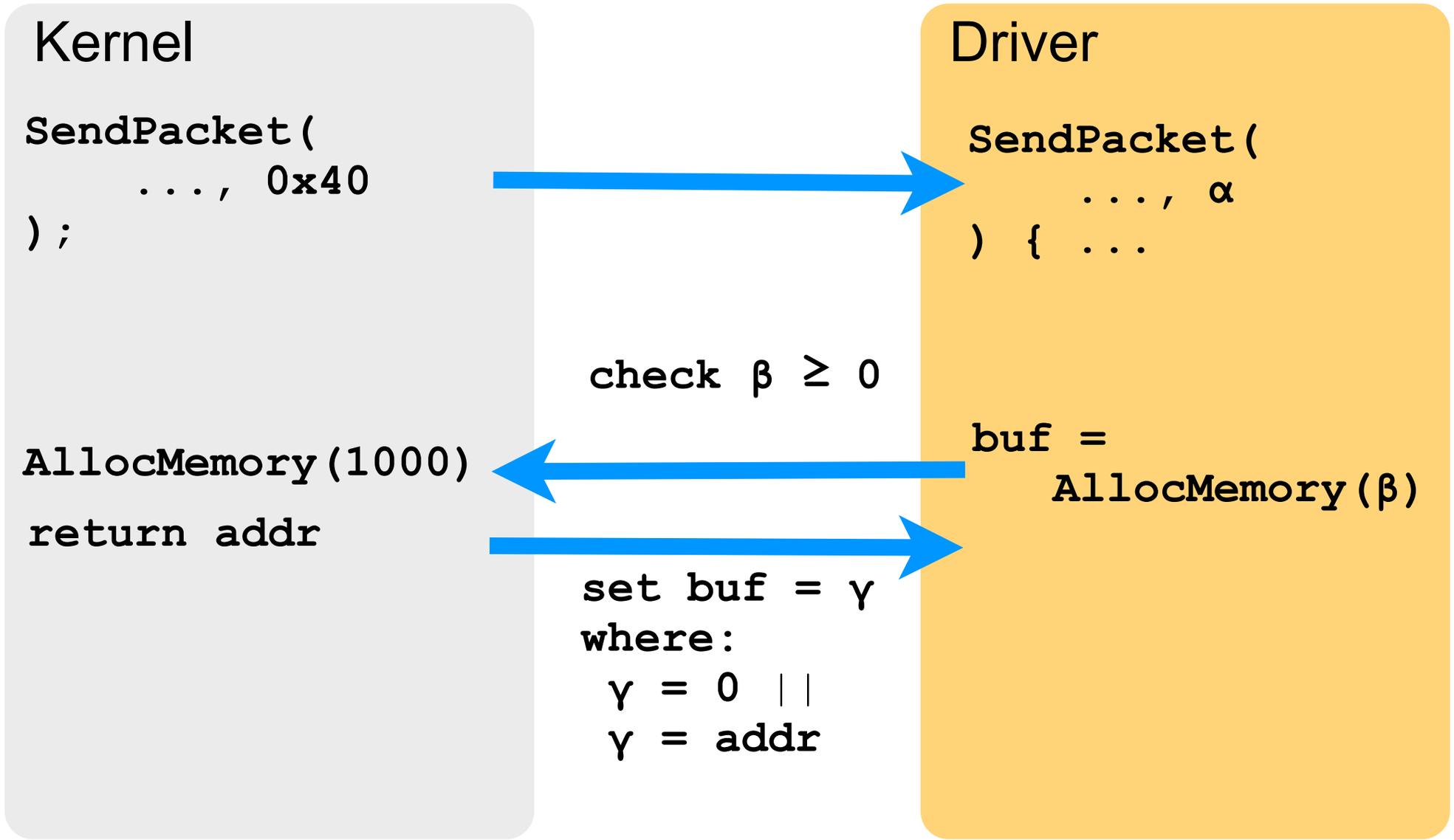
Kernel-Driver Interface



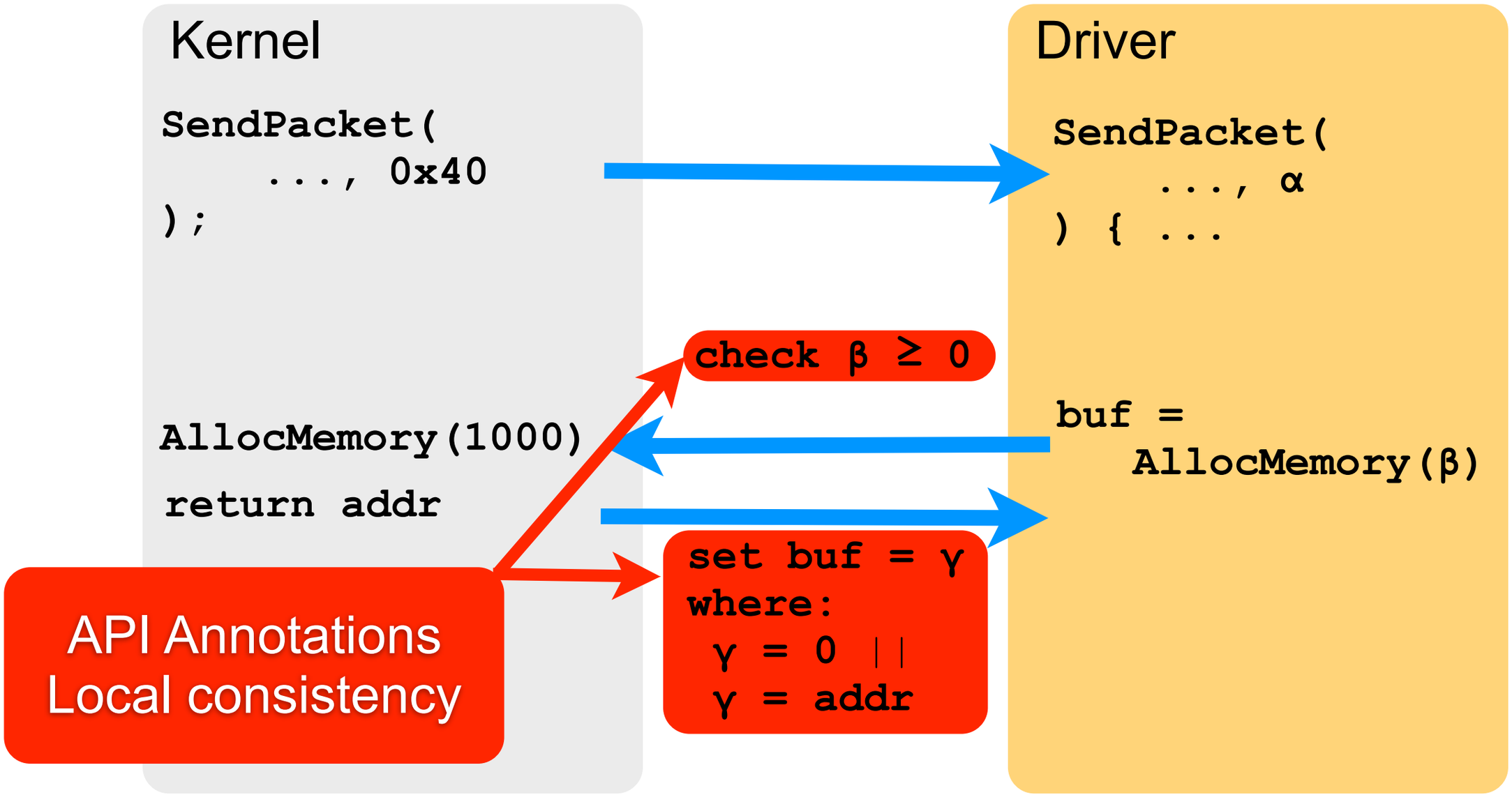
Kernel-Driver Interface

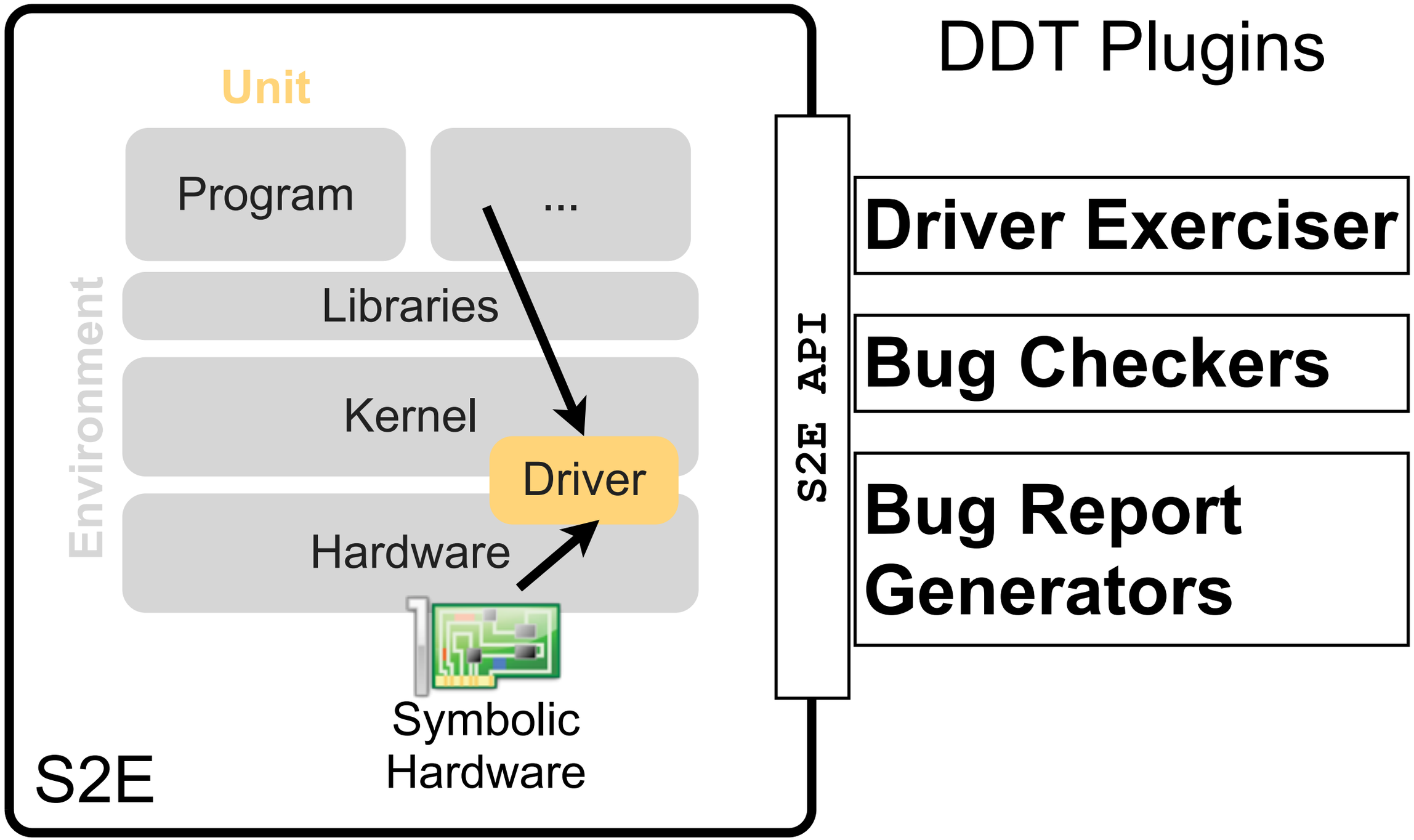


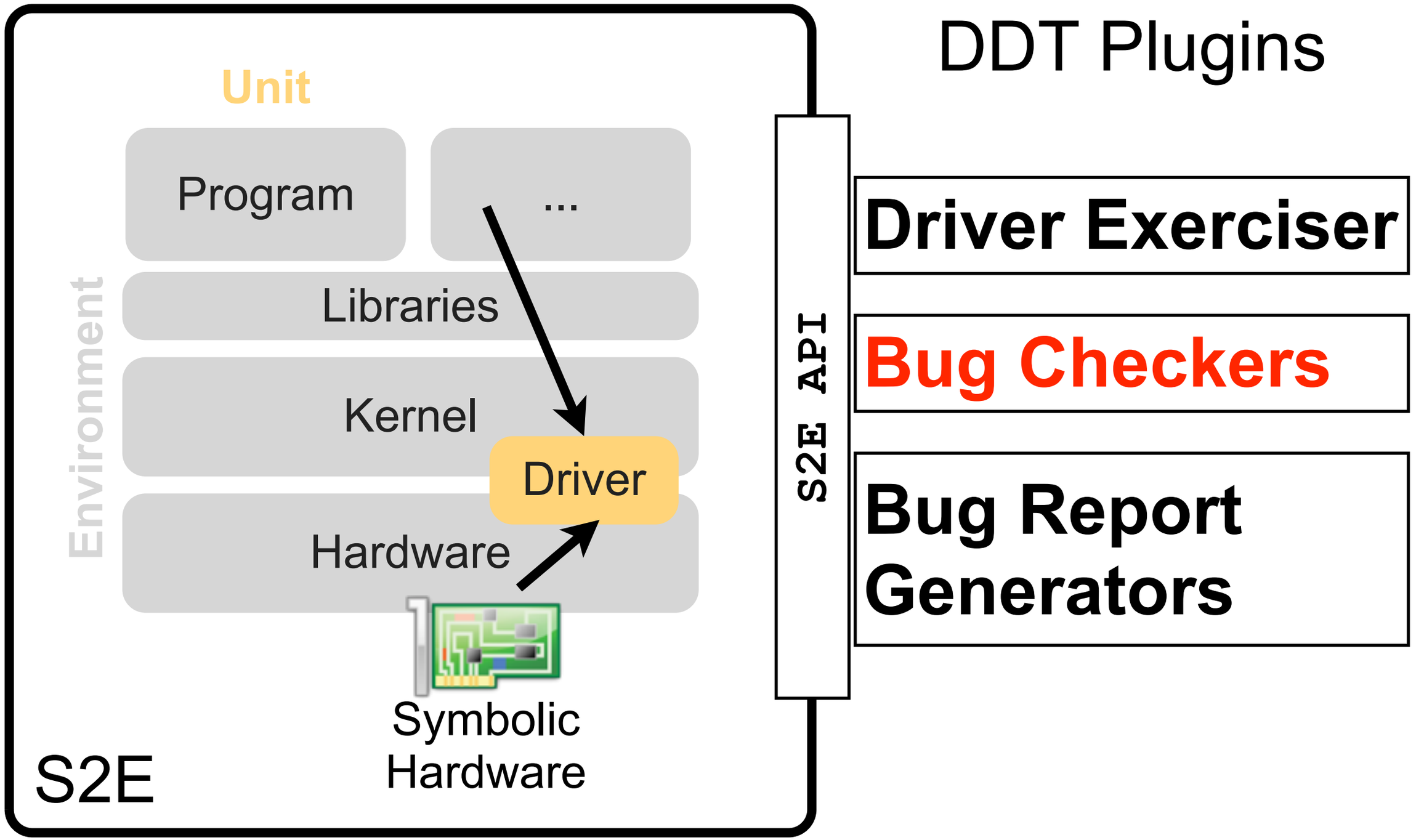
Kernel-Driver Interface



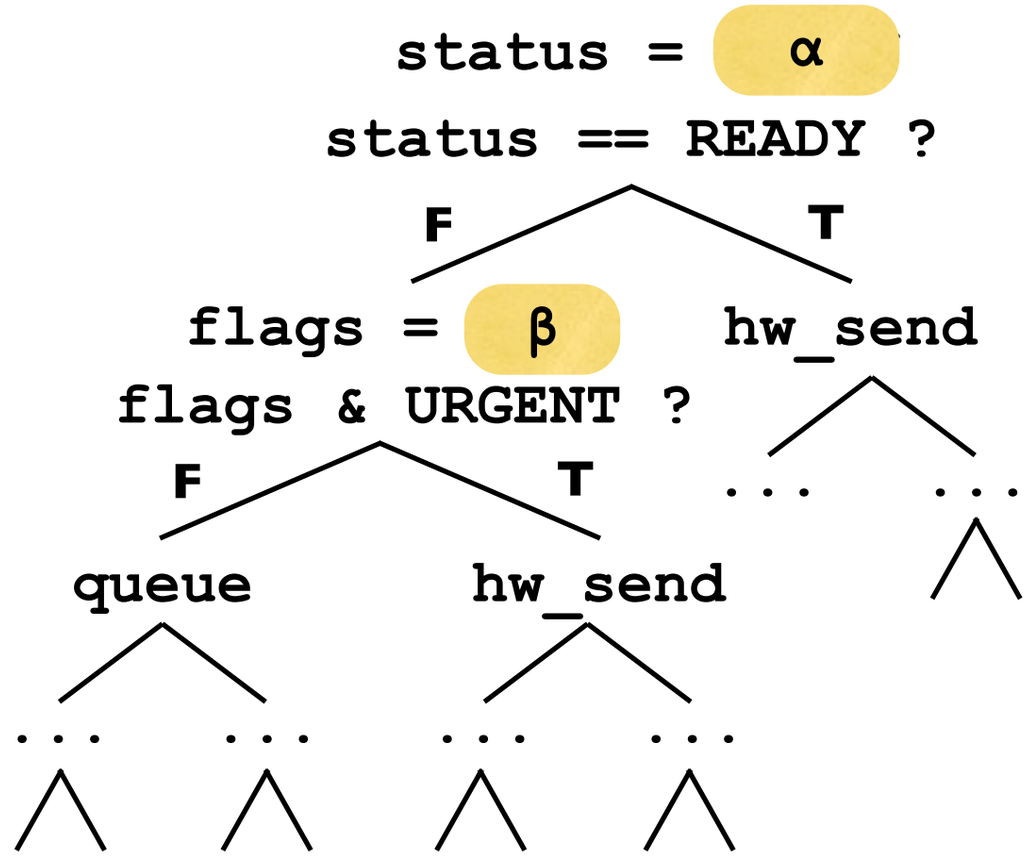
Kernel-Driver Interface




Unit
Program
...
Libraries
Kernel
Driver
Hardware
Symbolic Hardware
Environment
S2E
S2E API
DDT Plugins
Driver Exerciser
Bug Checkers
Bug Report Generators

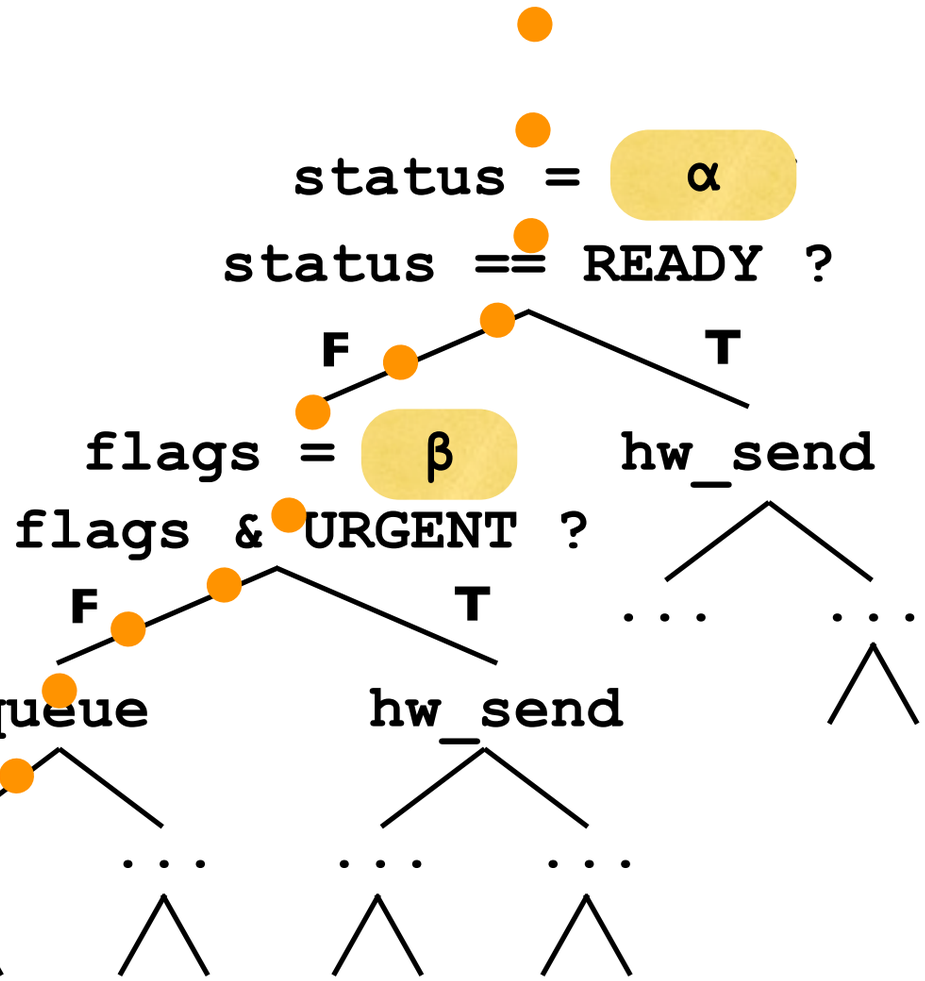

Unit
Program
...
Libraries
Kernel
Driver
Hardware
Symbolic Hardware
Environment
S2E
S2E API
DDT Plugins
Driver Exerciser
Bug Checkers
Bug Report Generators

Bug Checker Plugins



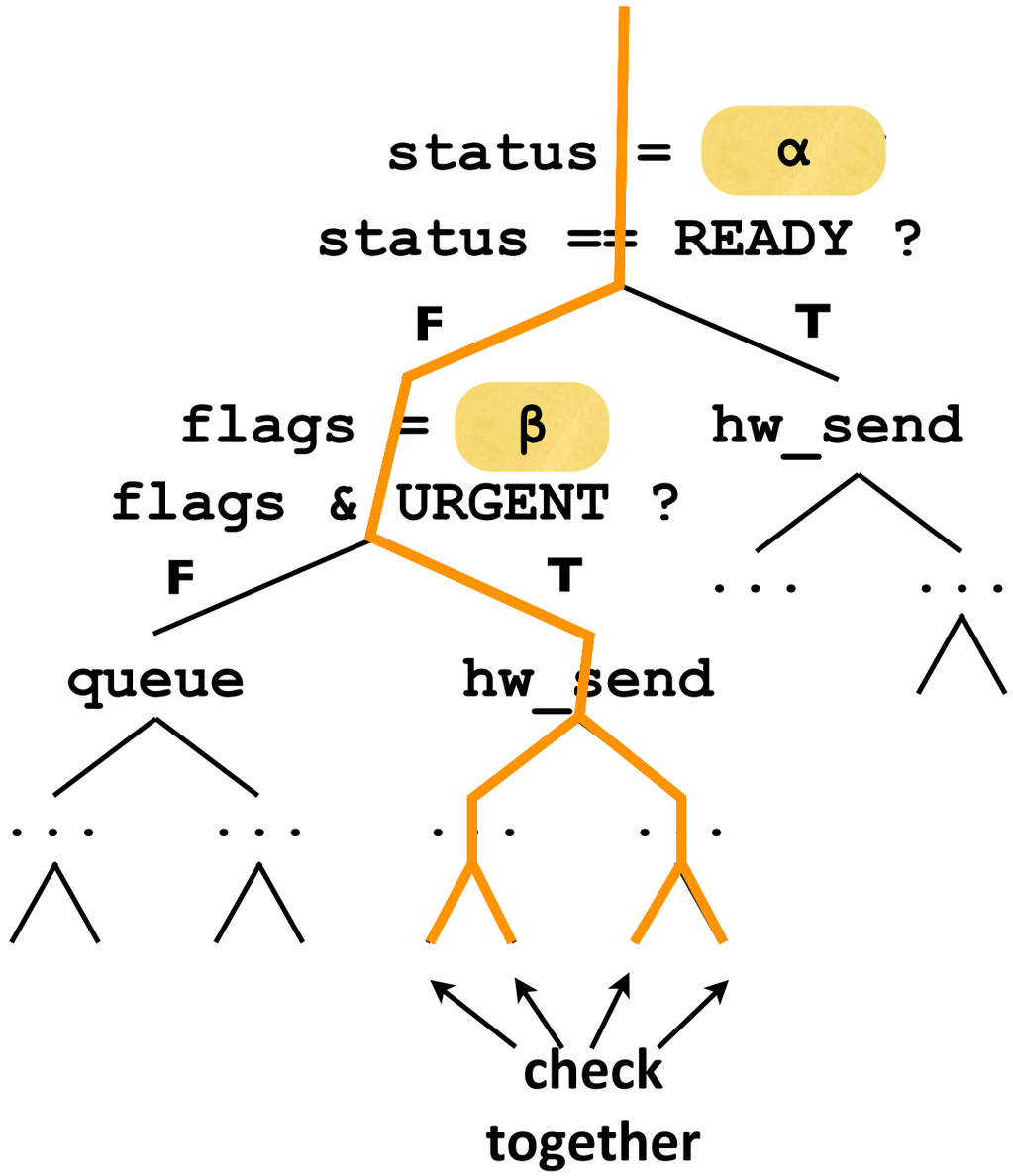
Bug Checker Plugins

- Instruction-granularity instrumentation



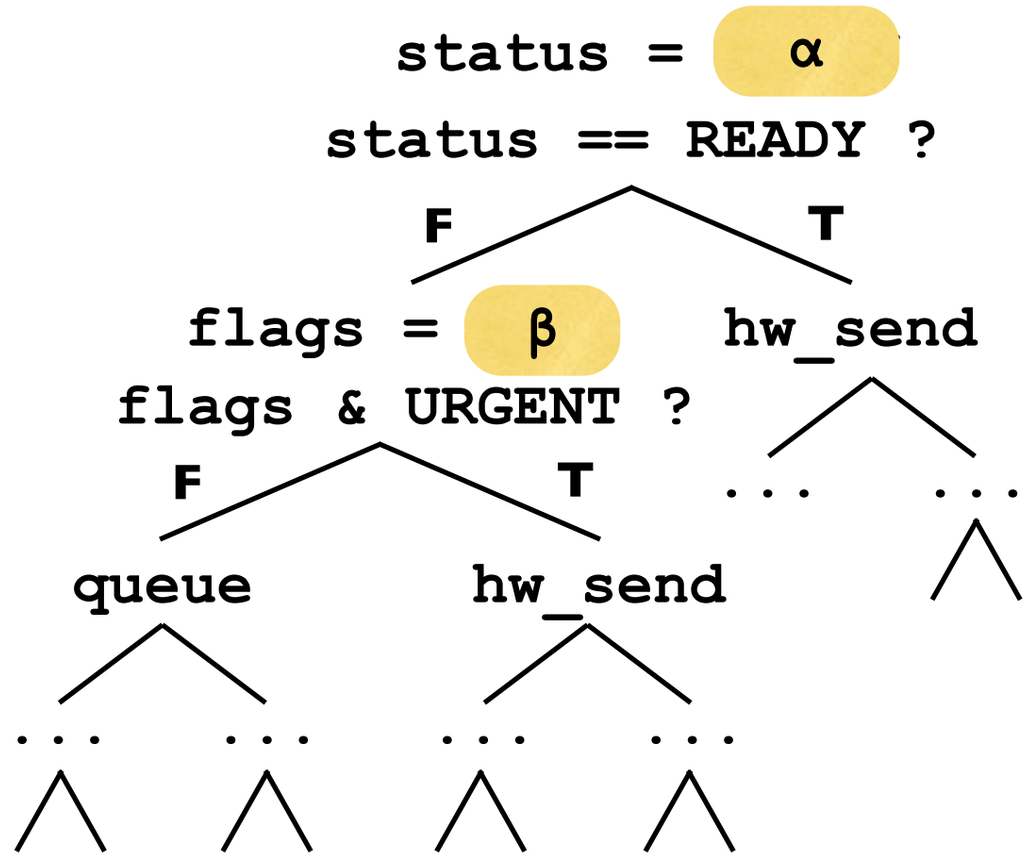
Bug Checker Plugins

- Instruction-granularity instrumentation
- Multi-path analysis



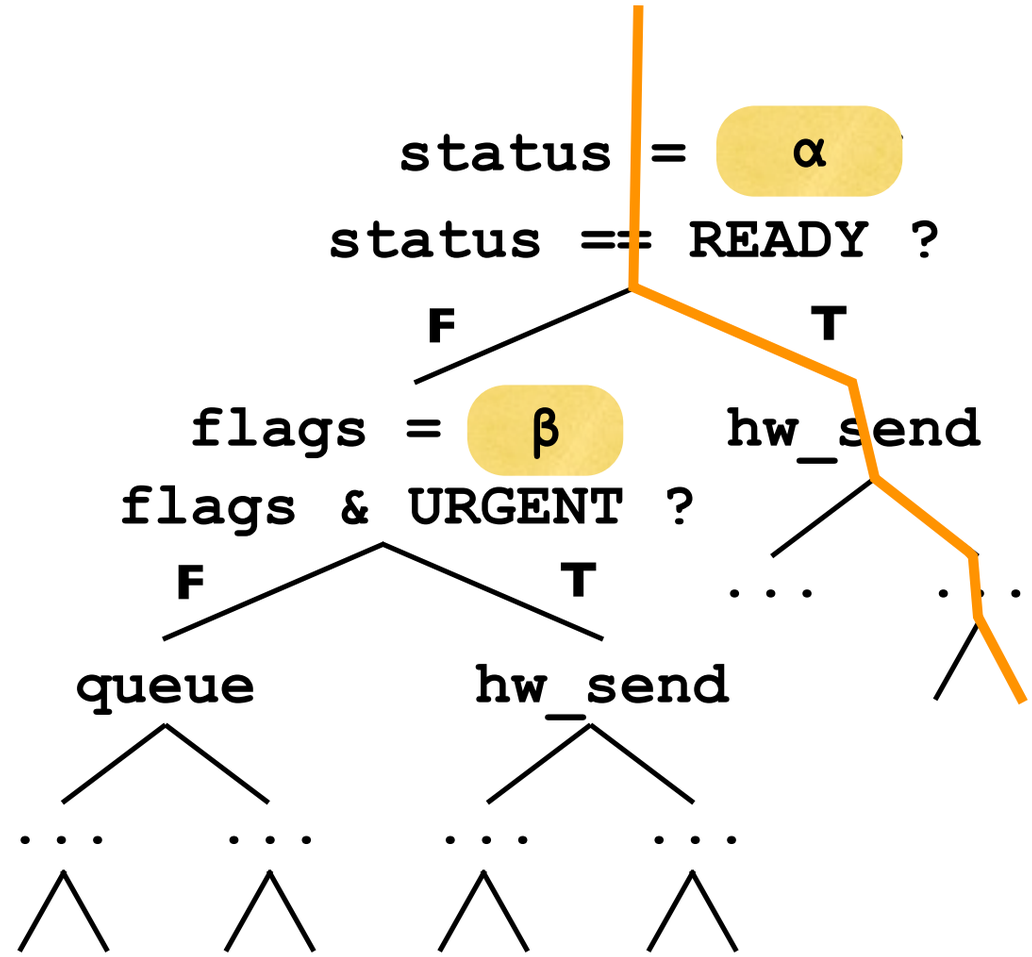
Guest Bug Checkers

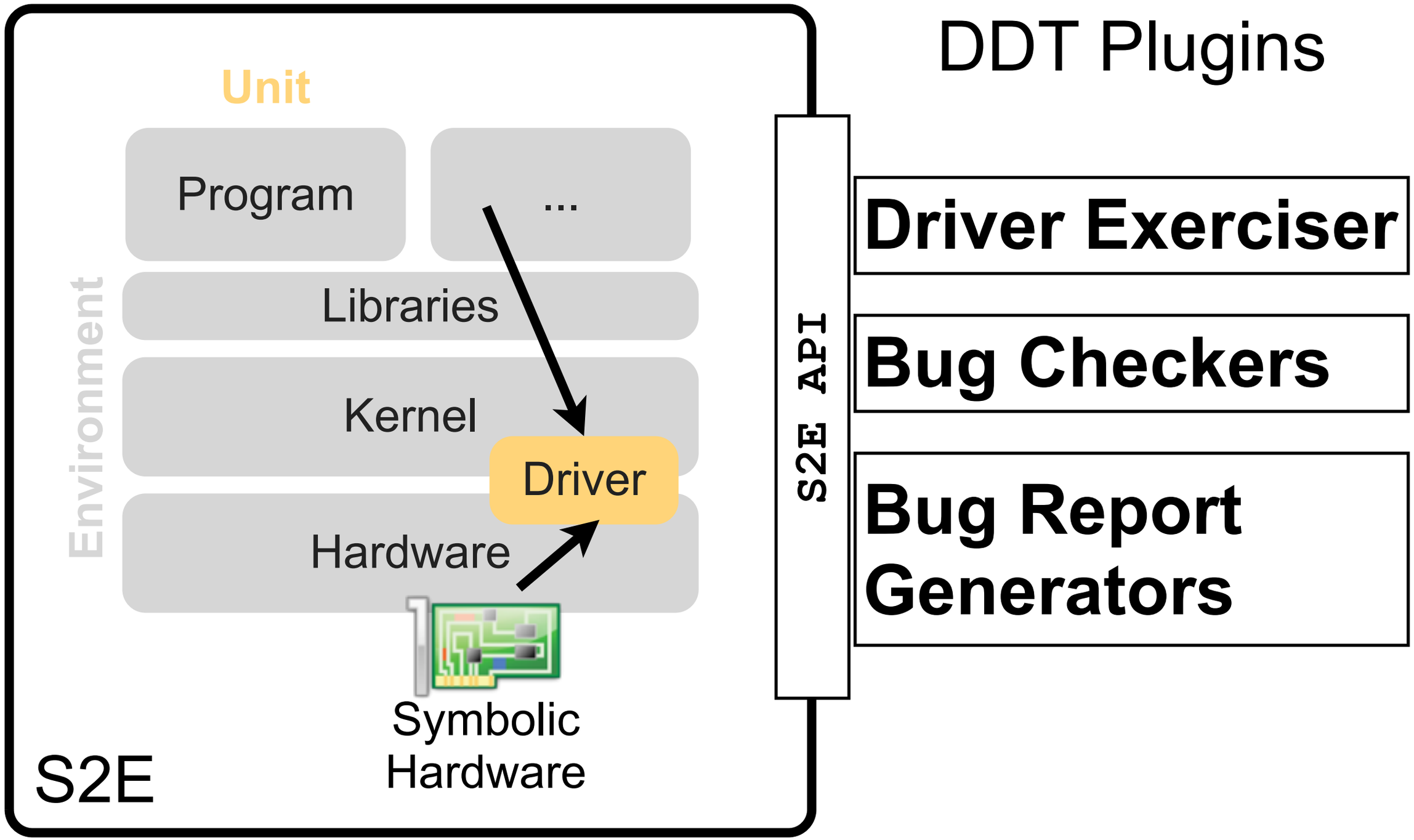
- They run in ***guest OS***
- You can ***reuse*** existing single-path dynamic bug-finding tools

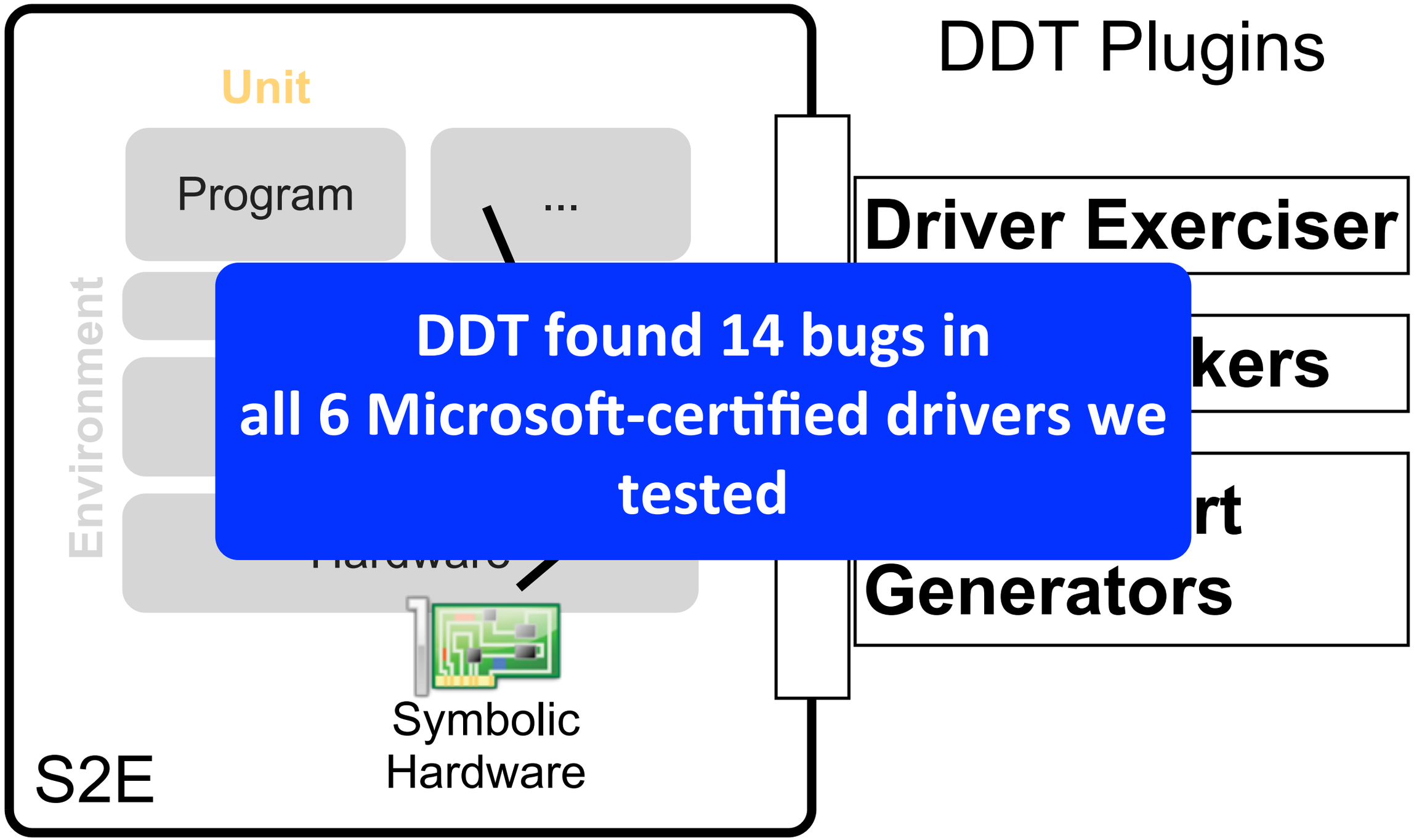


Guest Bug Checkers

- They run in ***guest OS***
- You can ***reuse*** existing single-path dynamic bug-finding tools




Unit
Program
...
Libraries
Kernel
Driver
Hardware
Symbolic Hardware
Environment
S2E
S2E API
DDT Plugins
Driver Exerciser
Bug Checkers
Bug Report Generators


Unit
Program
...
Environment
Hardware
Symbolic Hardware
S2E
DDT Plugins
Driver Exerciser
kers
rt
Generators
DDT found 14 bugs in all 6 Microsoft-certified drivers we tested

Using S2E in Practice

- Automated Device Driver Testing
  DDT
- Automated Reverse Engineering
  RevNIC
- Multi-path Performance Profiling
  PROFs

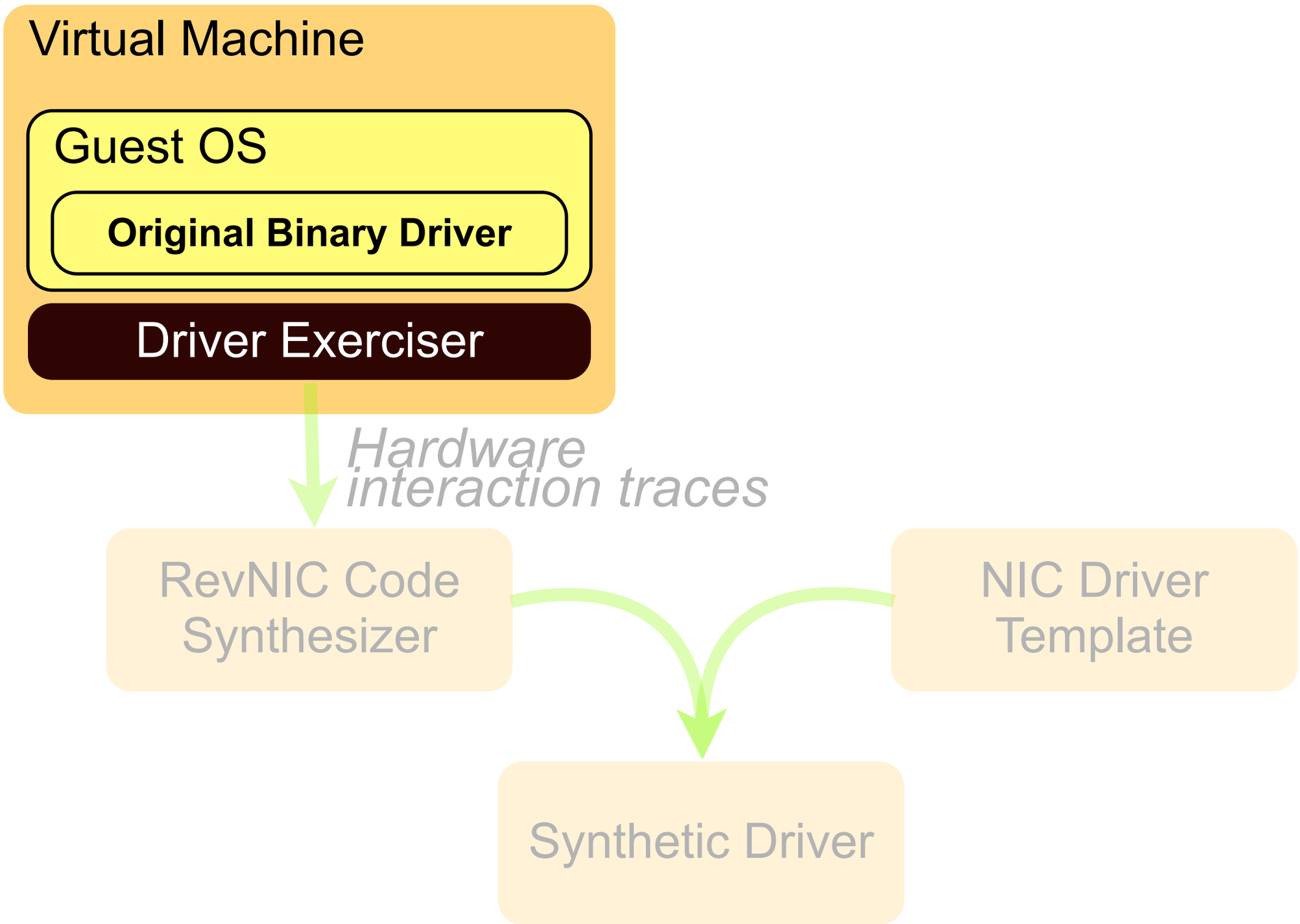
Using S2E in Practice

- Automated Device Driver Testing
  DDT
- Automated Reverse Engineering
  RevNIC
- Multi-path Performance Profiling
  PROFs

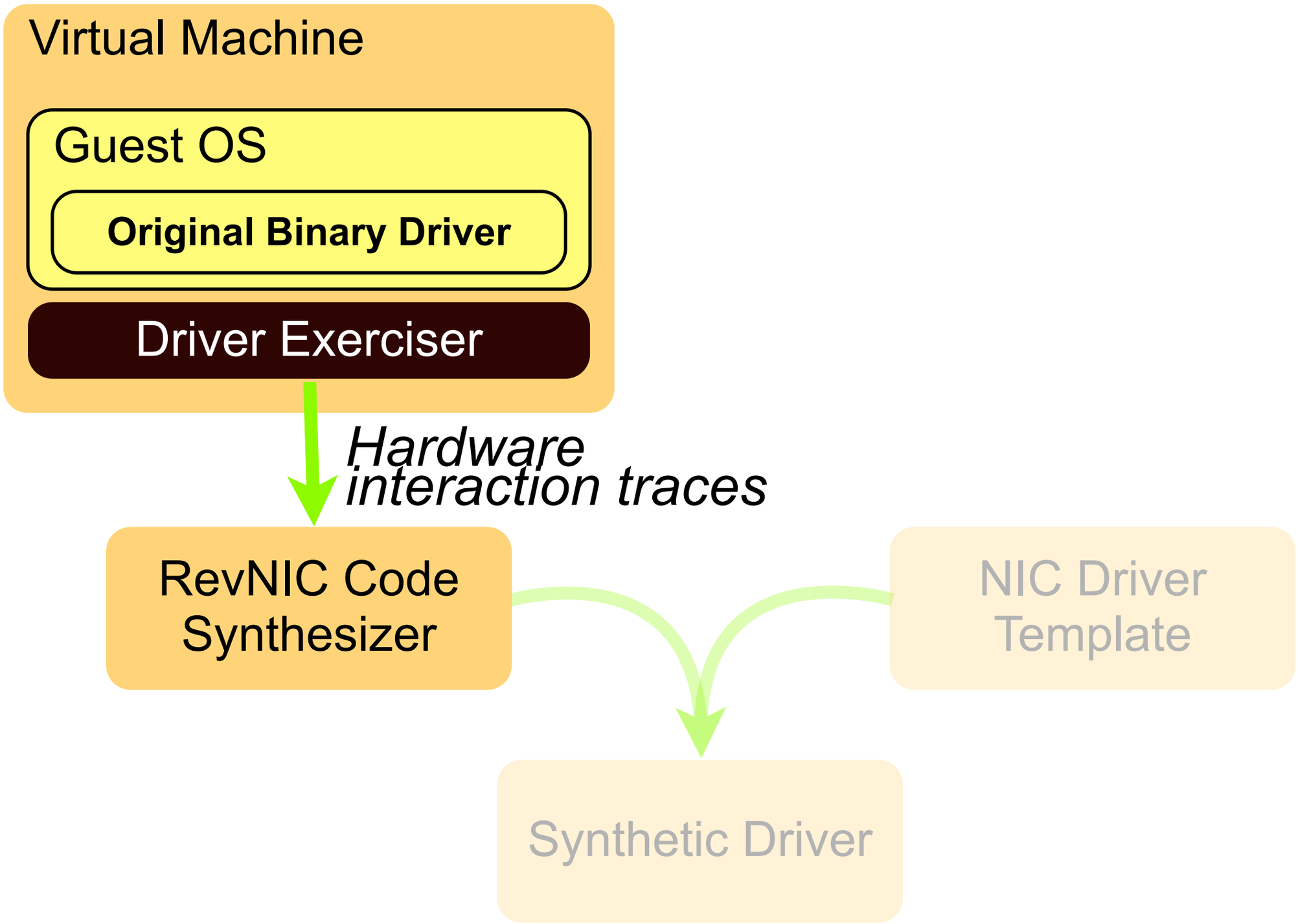
Drivers: Hard to Write and Hard to Port

- Drivers are often closed source
  Porting from existing drivers is difficult
- Devices rarely come with an interface specification
  Hard to write a driver from scratch
- Specifications are often incomplete and buggy
  Buggy driver implementation

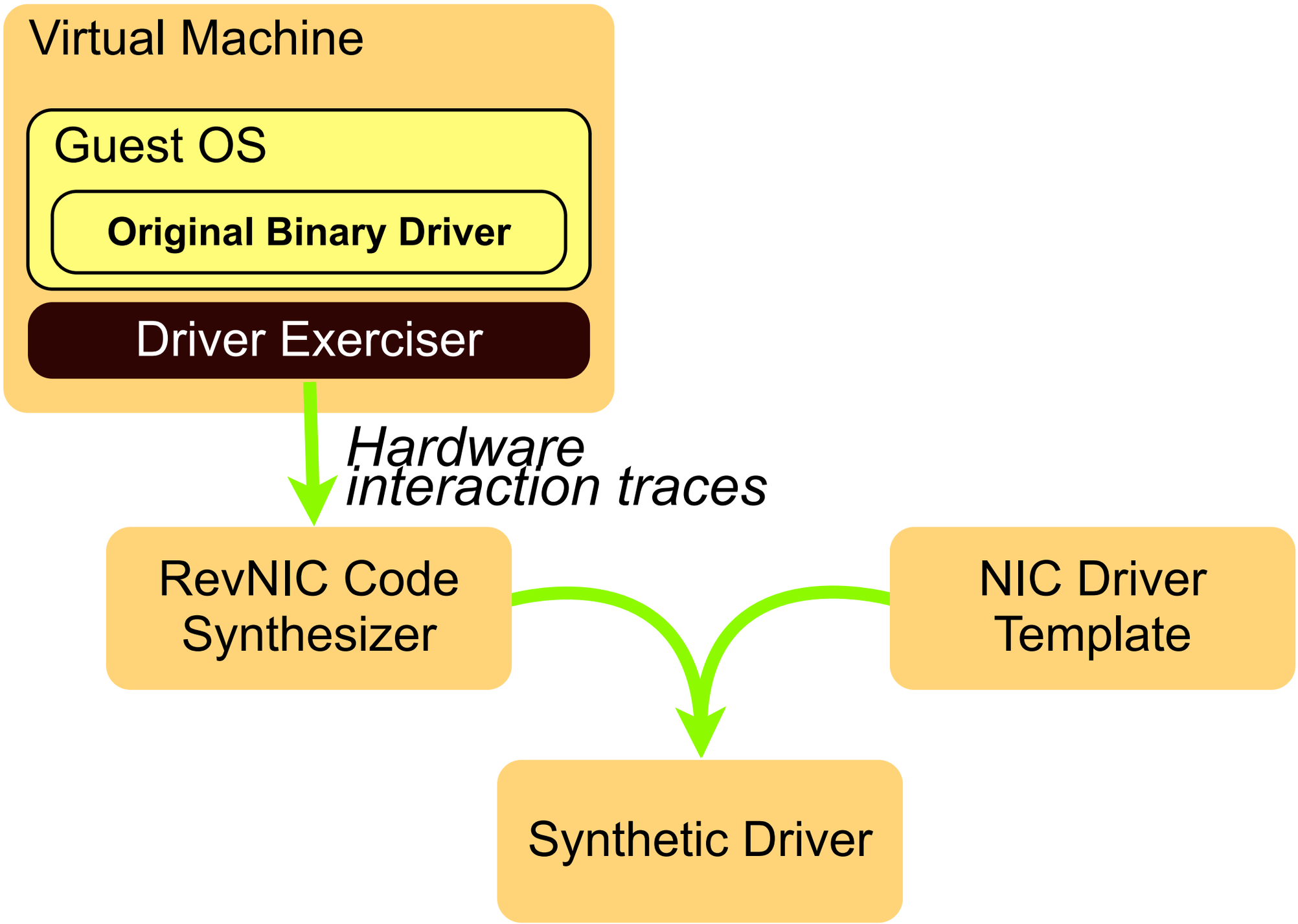
RevNIC


Virtual Machine
Guest OS
Original Binary Driver
Driver Exerciser
Hardware interaction traces
RevNIC Code Synthesizer
NIC Driver Template
Synthetic Driver

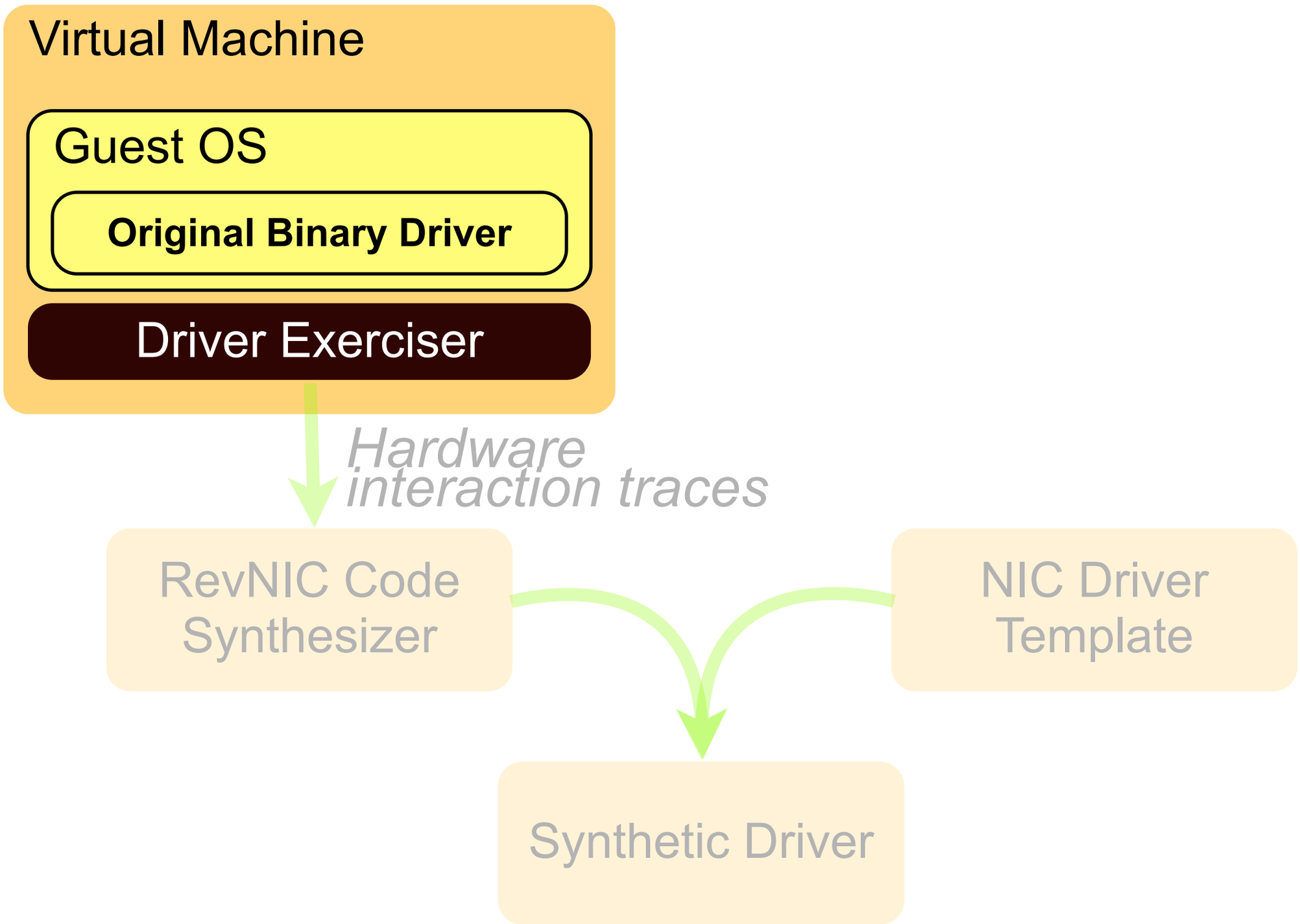
RevNIC


Virtual Machine
Guest OS
Original Binary Driver
Driver Exerciser
Hardware interaction traces
RevNIC Code Synthesizer
NIC Driver Template
Synthetic Driver

RevNIC

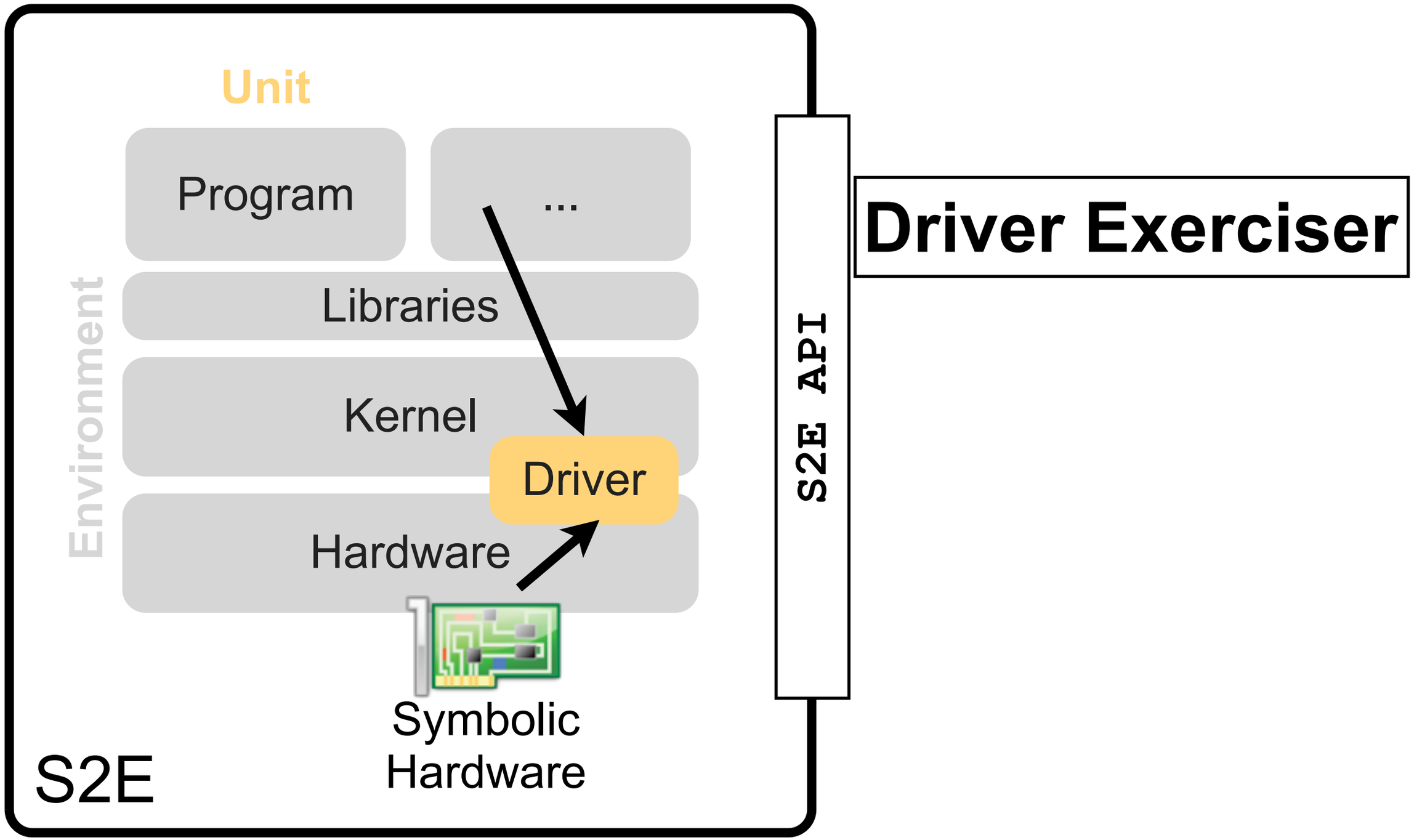

Virtual Machine
Guest OS
Original Binary Driver
Driver Exerciser
Hardware interaction traces
RevNIC Code Synthesizer
NIC Driver Template
Synthetic Driver

RevNIC


Virtual Machine
Guest OS
Original Binary Driver
Driver Exerciser
Hardware interaction traces
RevNIC Code Synthesizer
NIC Driver Template
Synthetic Driver

High Coverage Driver Exerciser

- Hand-crafted workload is not enough


Unit
Program
...
Libraries
Kernel
Driver
Hardware
Symbolic Hardware
Environment
S2E
S2E API
Driver Exerciser

Unit

Program

...

Libraries

Kernel

Driver

Hardware

Environment

Symbolic Hardware

S2E

S2E API

Driver Exerciser

Execution Tracer

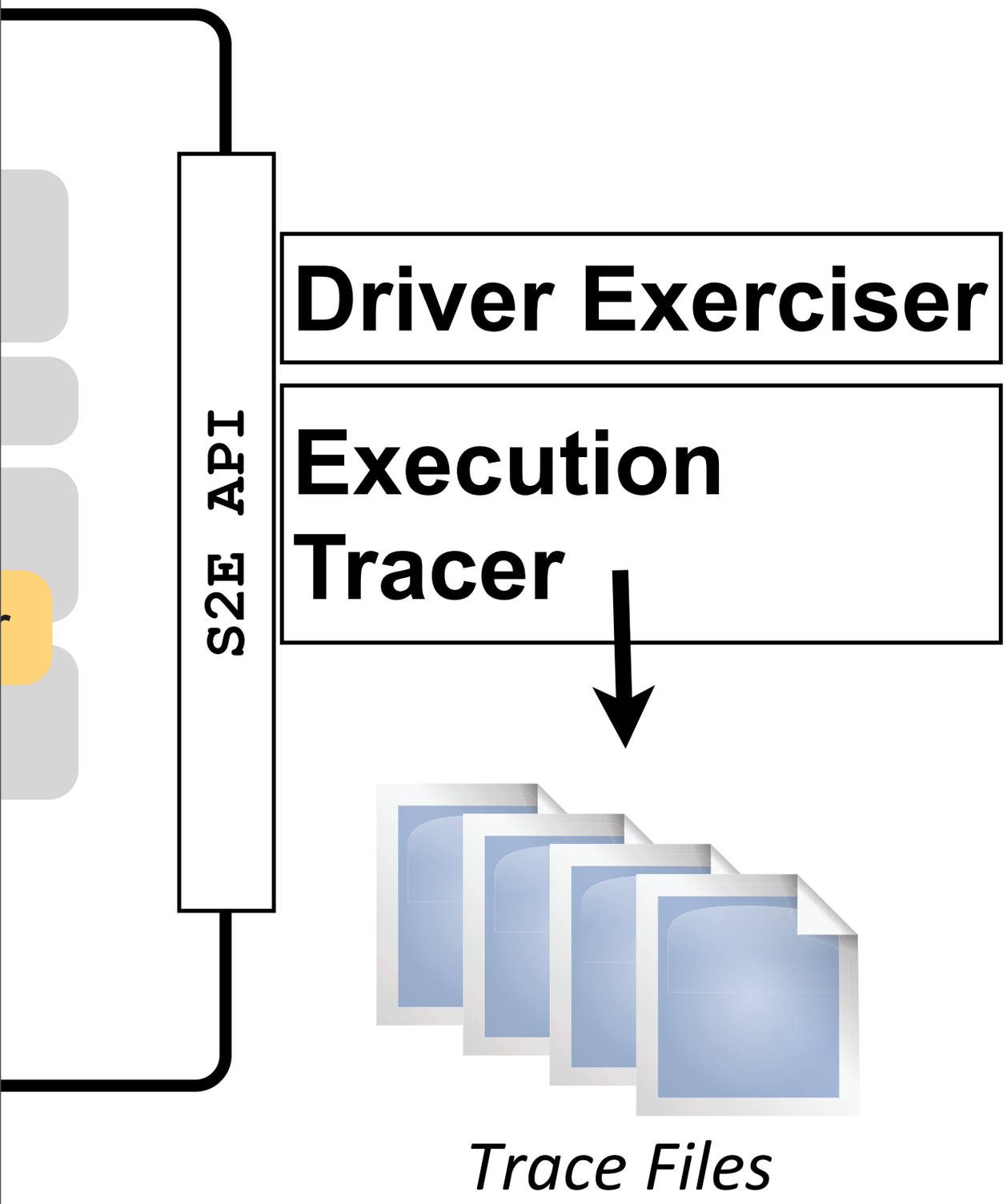
Trace Files

S2E API

Driver Exerciser

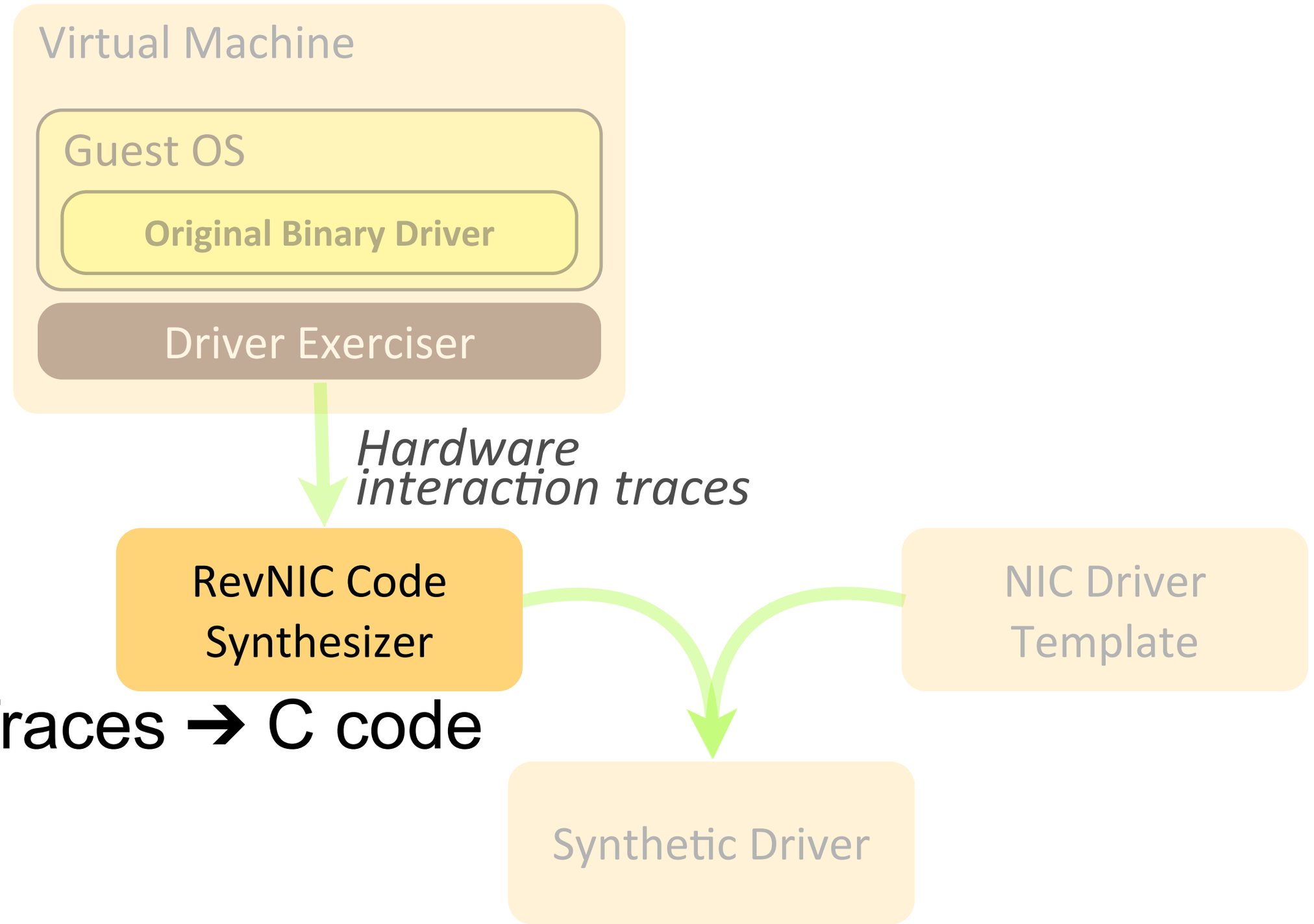
Execution Tracer

Trace Files



- Execution tree
- Machine instructions
- Memory accesses
- Register values
- (Memory-Mapped) I/O

RevNIC



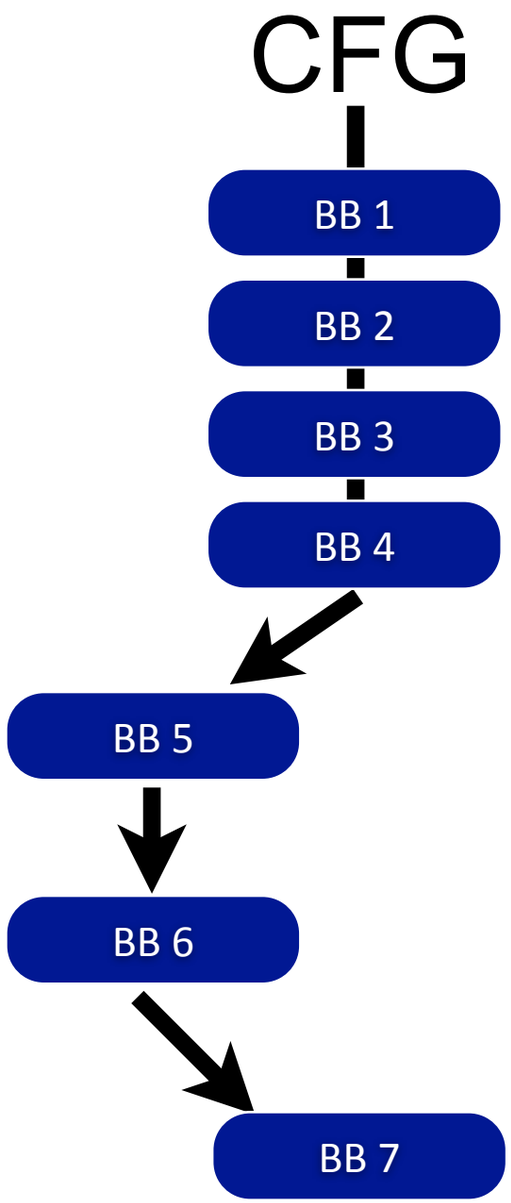
Traces → C code

Multi-Path Dynamic Disassembly

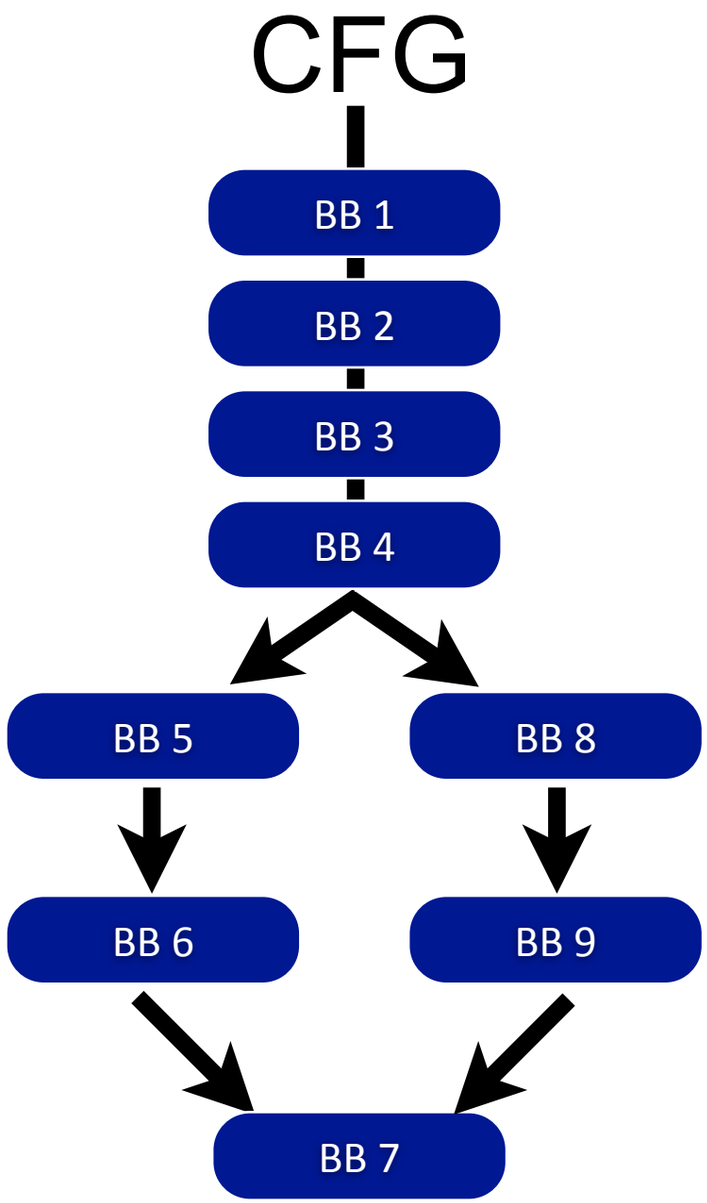
Multi-Path Dynamic Disassembly

CFG

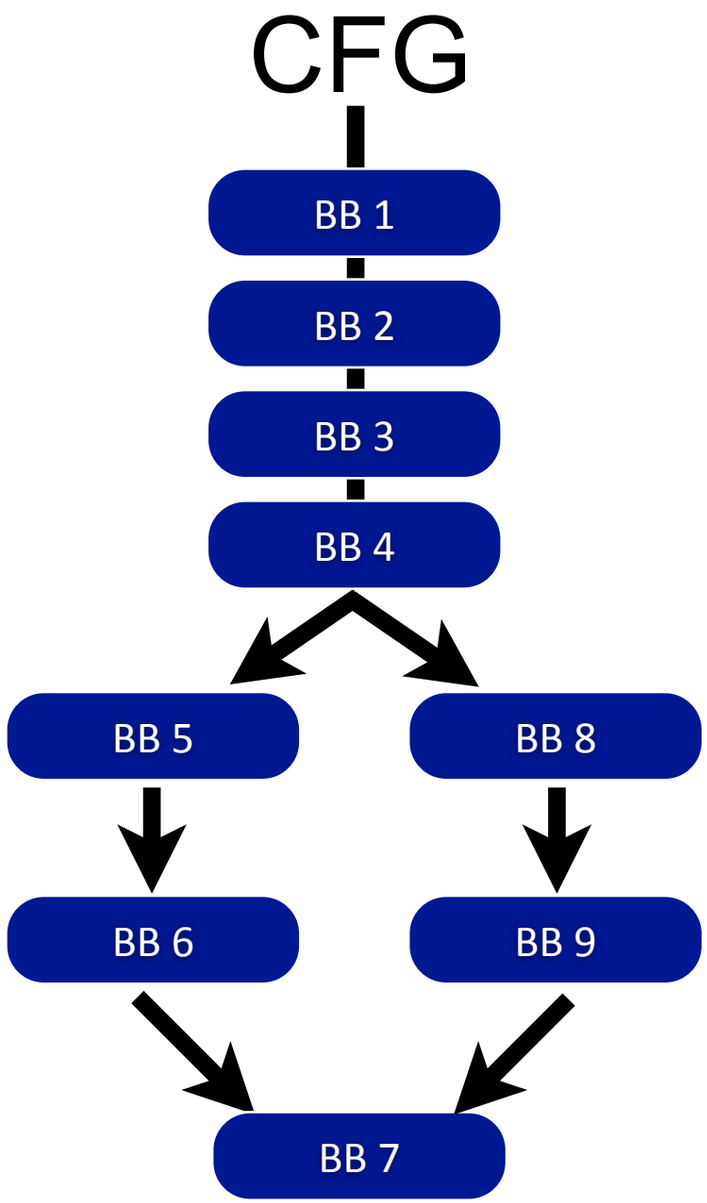
Multi-Path Dynamic Disassembly



Multi-Path Dynamic Disassembly

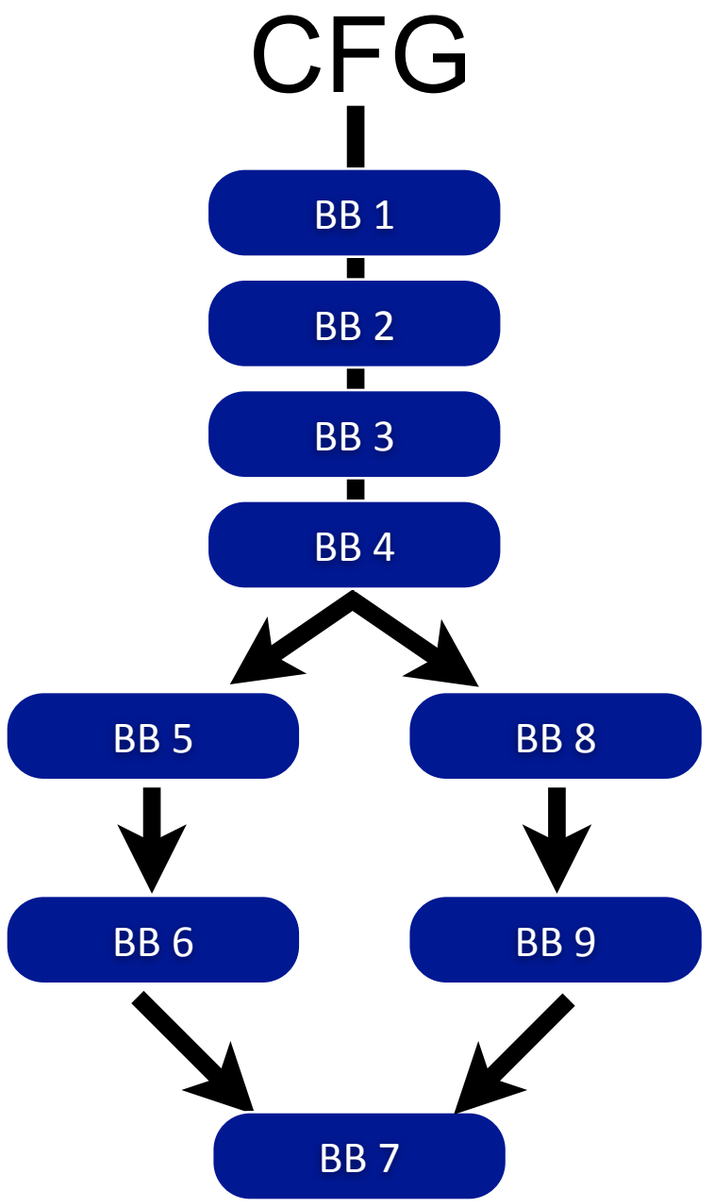


Multi-Path Dynamic Disassembly



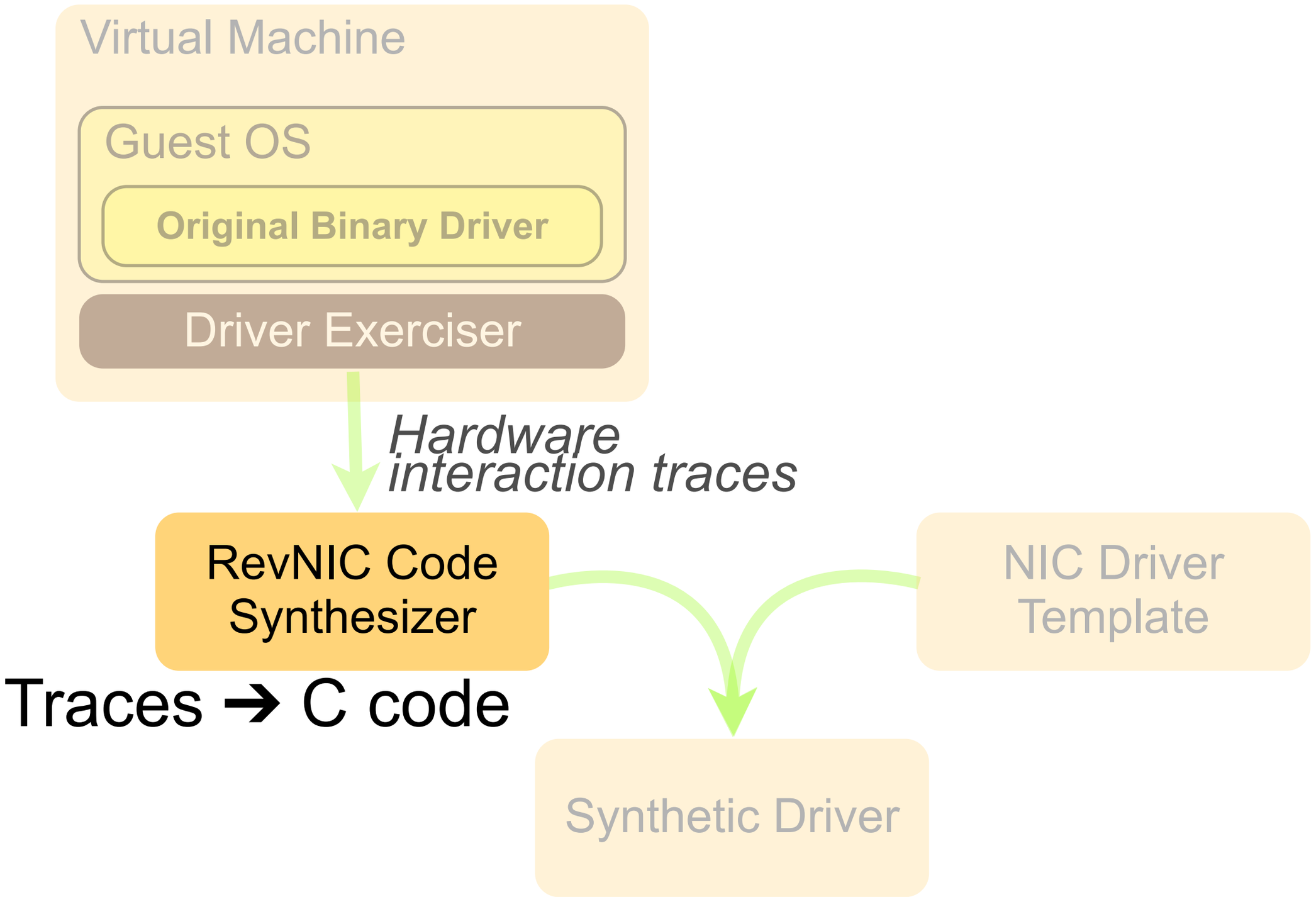

```
uint32_t function_0001(uint32_t param1,
                       uint32_t param2)
{  /* ... */
BB1:
   goto BB2;
BB2:
   v1 = read_port(param1);
BB3:
   v2 = read_port(param2);
BB4:
   if (v1 & 0x21) goto BB8;
BB5:
   write_port(param2, 0x1234);
BB6:
   goto BB7;
BB8:
   write_port(param1, 0x4567);
BB9:
   goto BB7;
BB7:
}
```

Multi-Path Dynamic Disassembly



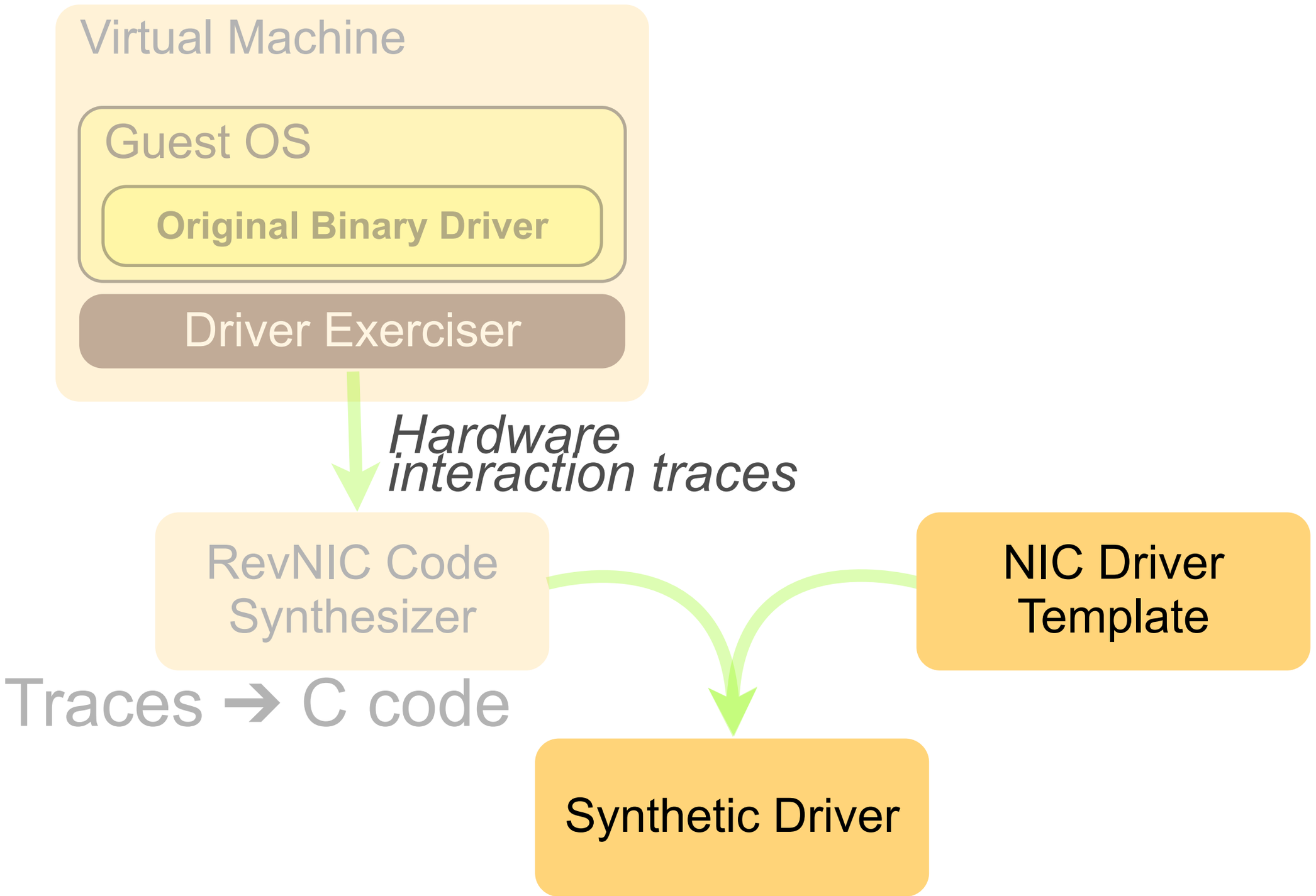

```
uint32_t function_0001(uint32_t param1,
                       uint32_t param2)
{  /* ... */
BB1:
   goto BB2;
BB2:
   v1 = read_port(param1);
BB3:
   v2 = read_port(param2);
BB4:
   if (v1 & 0x21) goto BB8;
BB5:
   write_port(param2, 0x1234);
BB6:
   goto BB7;
BB8:
   write_port(param1, 0x4567);
BB9:
   goto BB7;
BB7:
}
```

RevNIC

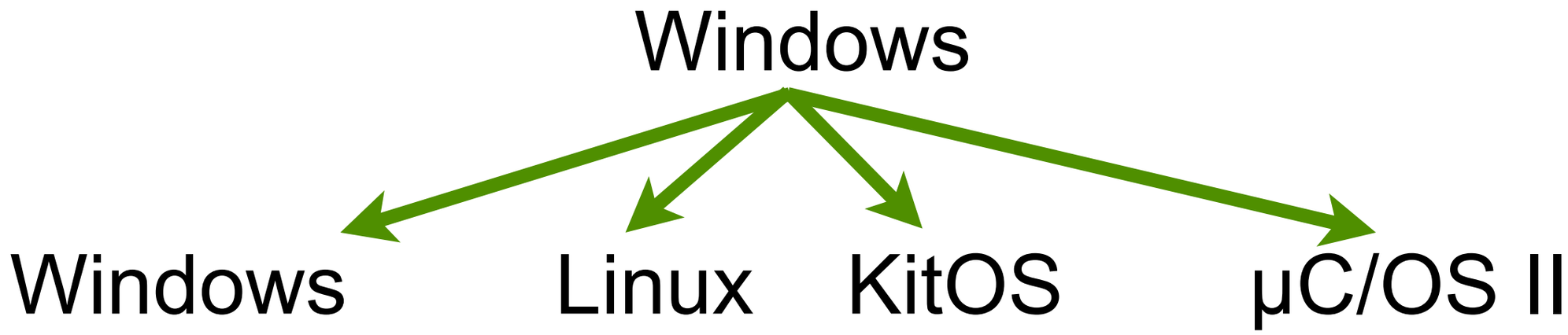


Traces → C code

RevNIC

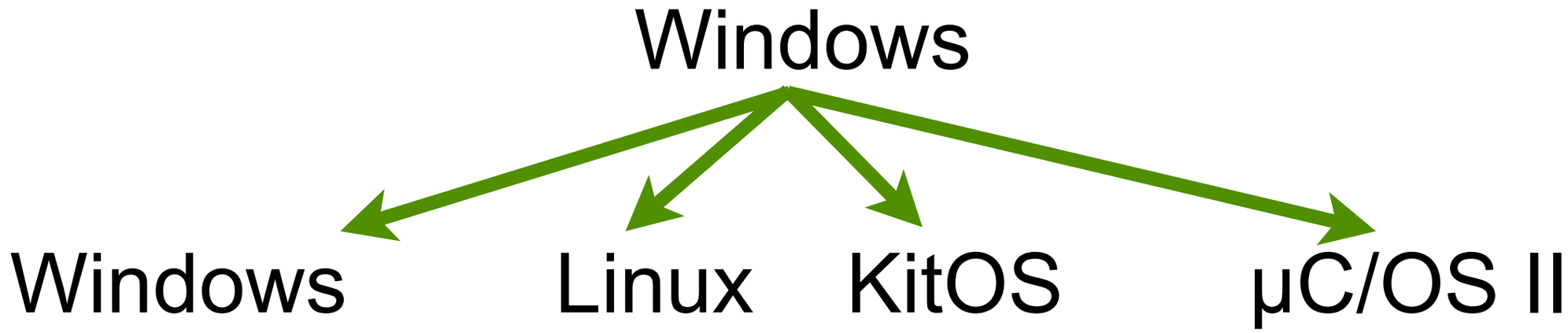

Virtual Machine
Guest OS
Original Binary Driver
Driver Exerciser
Hardware interaction traces
RevNIC Code Synthesizer
Traces → C code
NIC Driver Template
Synthetic Driver

Target Platforms



http://bplteensofwa.files.wordpress.com/2009/07/computer.jpg

Target Platforms

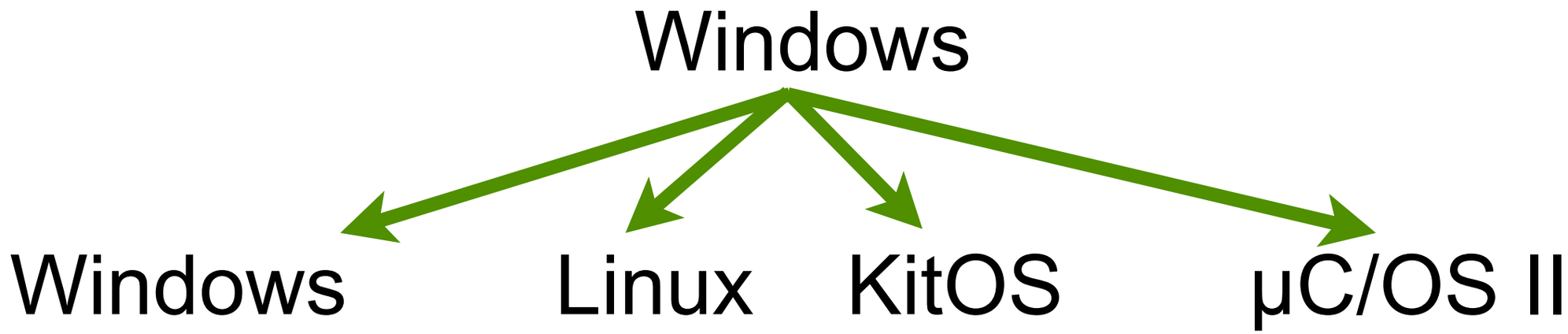


x86 PC

RTL8139

http://bplteensofwa.files.wordpress.com/2009/07/computer.jpg

Target Platforms



x86 PC

VMware
QEMU

RTL8139

PCnet, NE2000

http://bplteensofwa.files.wordpress.com/2009/07/computer.jpg

Target Platforms

Windows

Windows | Linux | KitOS | µC/OS II

x86 PC

VMware
QEMU

FPGA4U

RTL8139 PCnet, NE2000 SMSC 91C111

http://bplteensofwa.files.wordpress.com/2009/07/computer.jpg

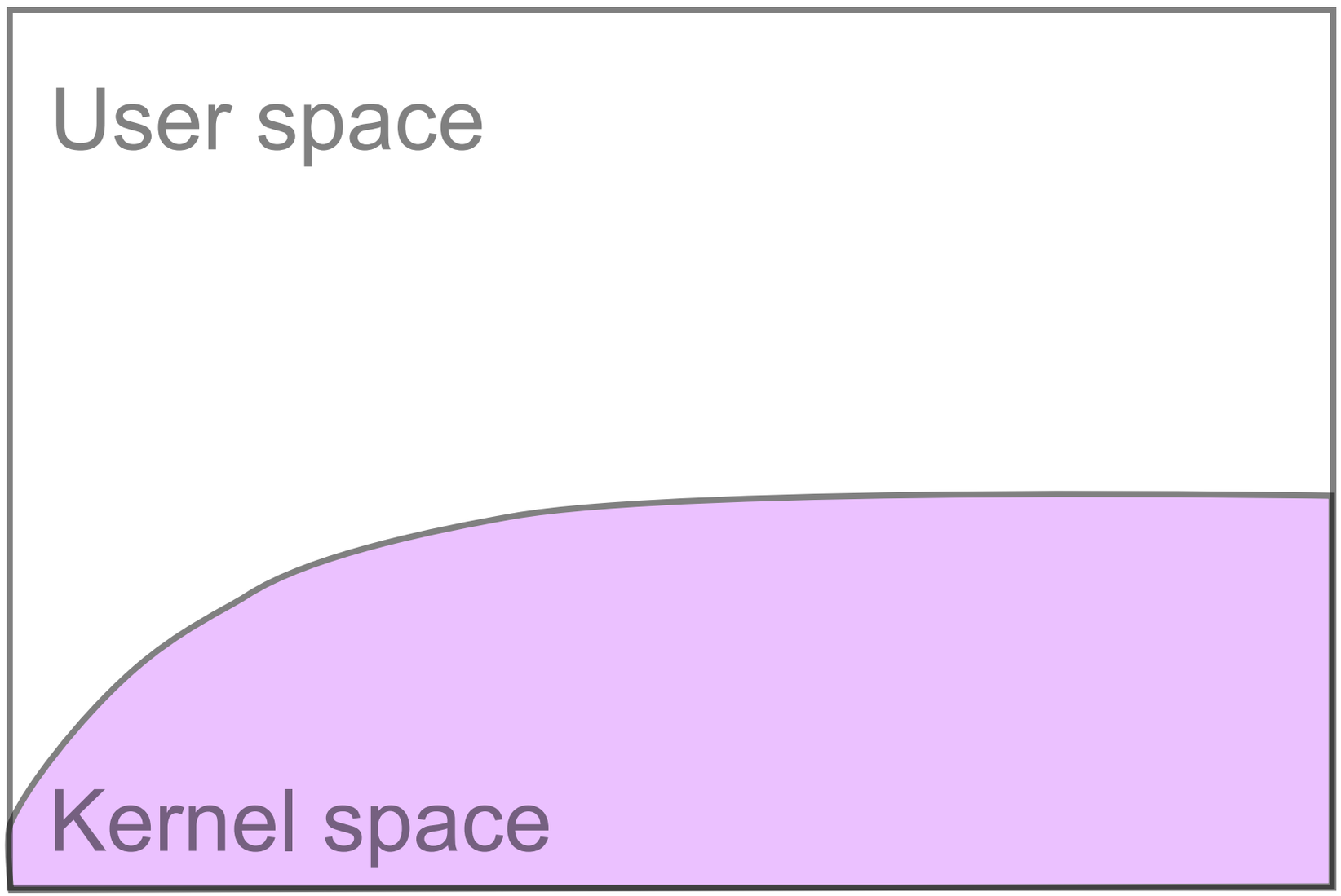
Using S2E in Practice

- Automated Device Driver Testing
  DDT
- Automated Reverse Engineering
  RevNIC
- Multi-path Performance Profiling
  PROFs

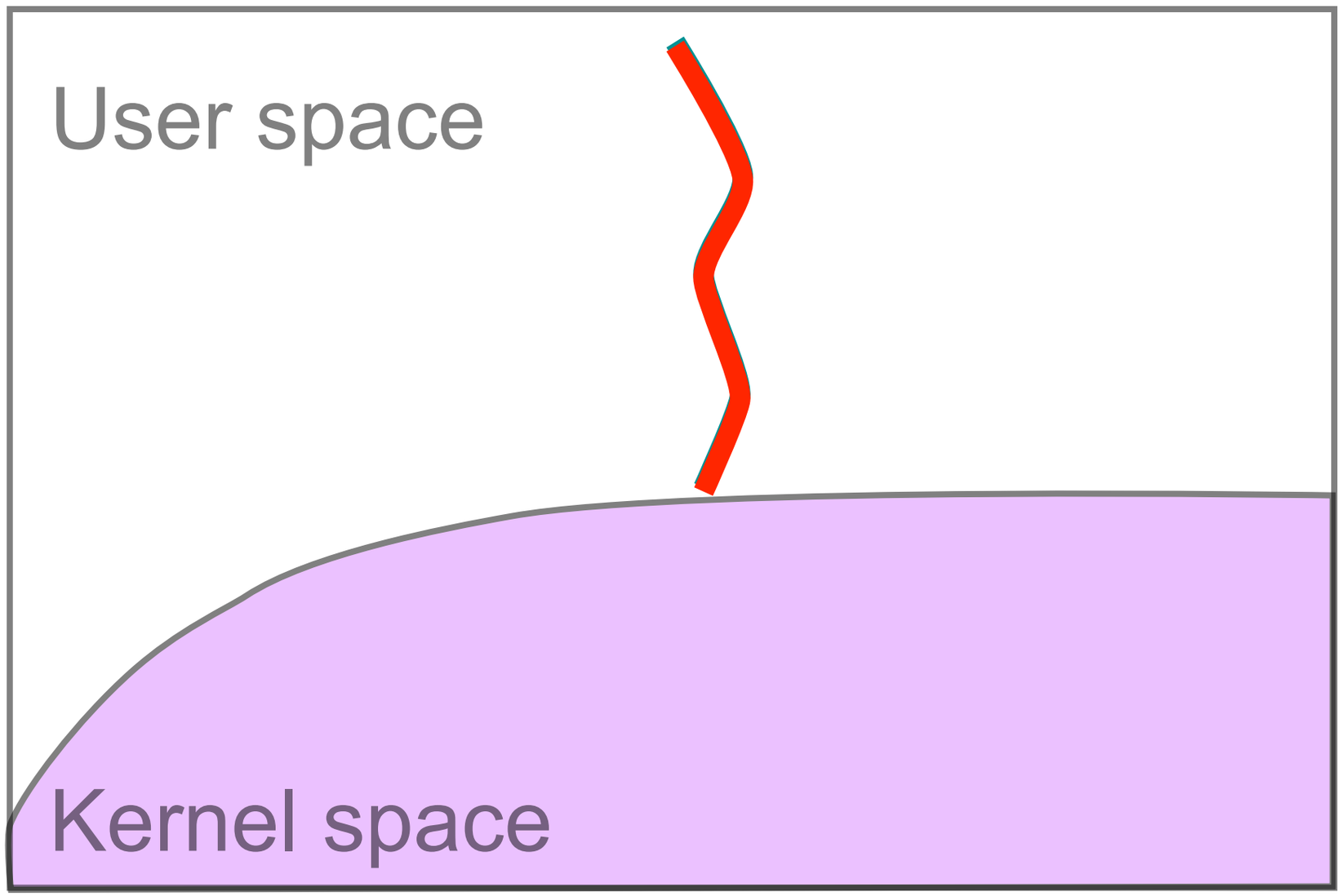
Using S2E in Practice

- Automated Device Driver Testing
  DDT
- Automated Reverse Engineering
  RevNIC
- Multi-path Performance Profiling
  PROFs

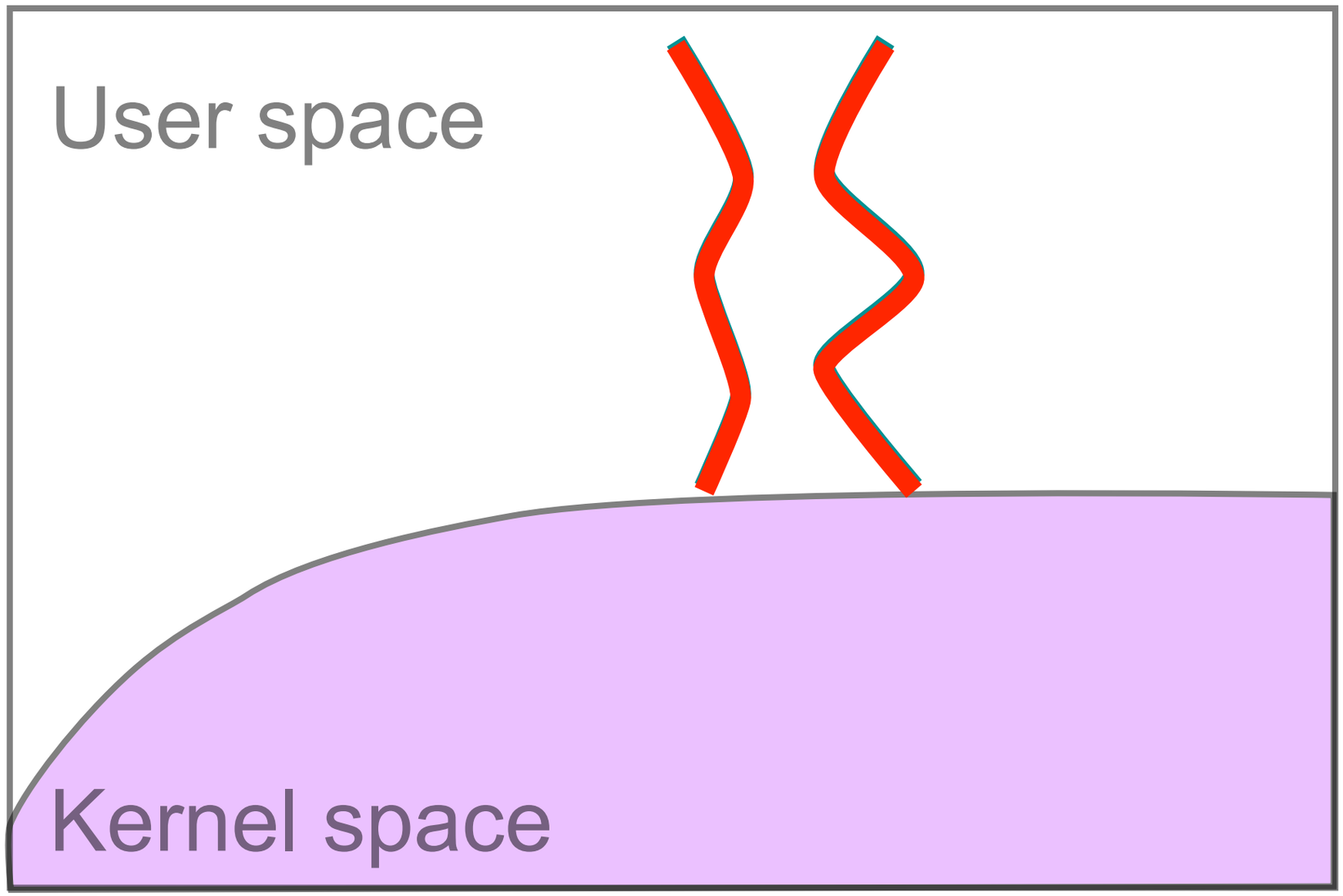
Single-Path Performance Profiling



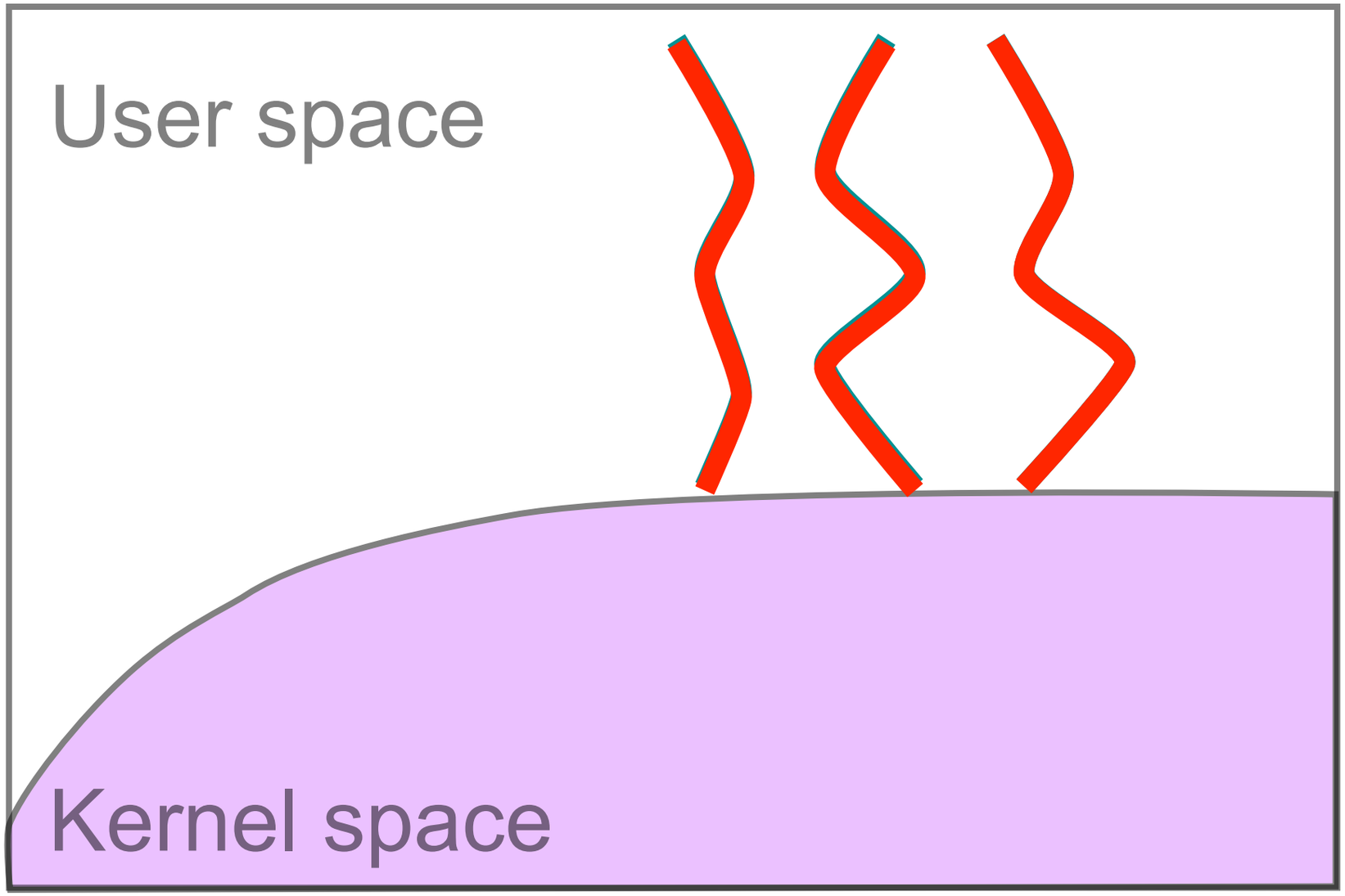
Single-Path Performance Profiling



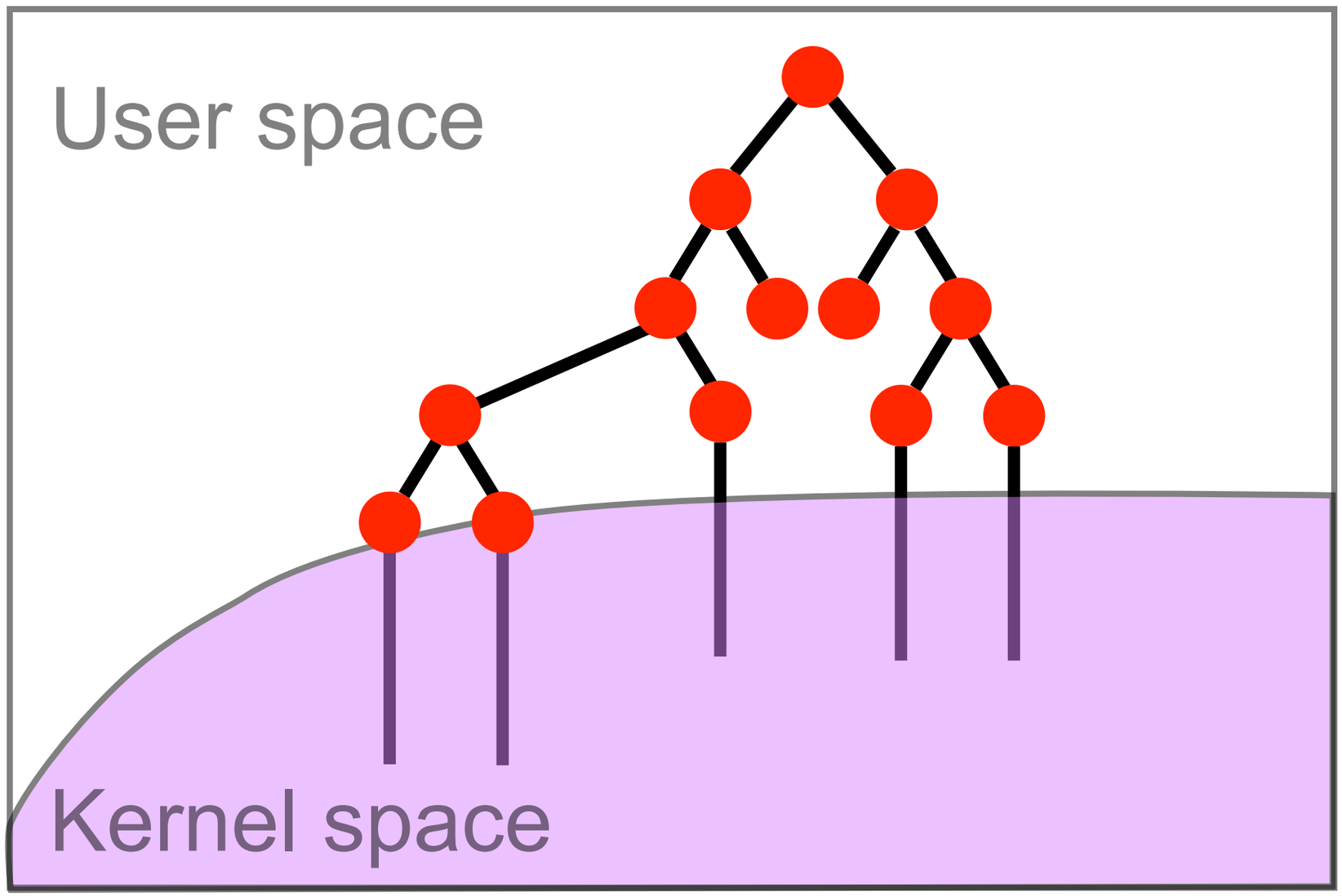
Single-Path Performance Profiling



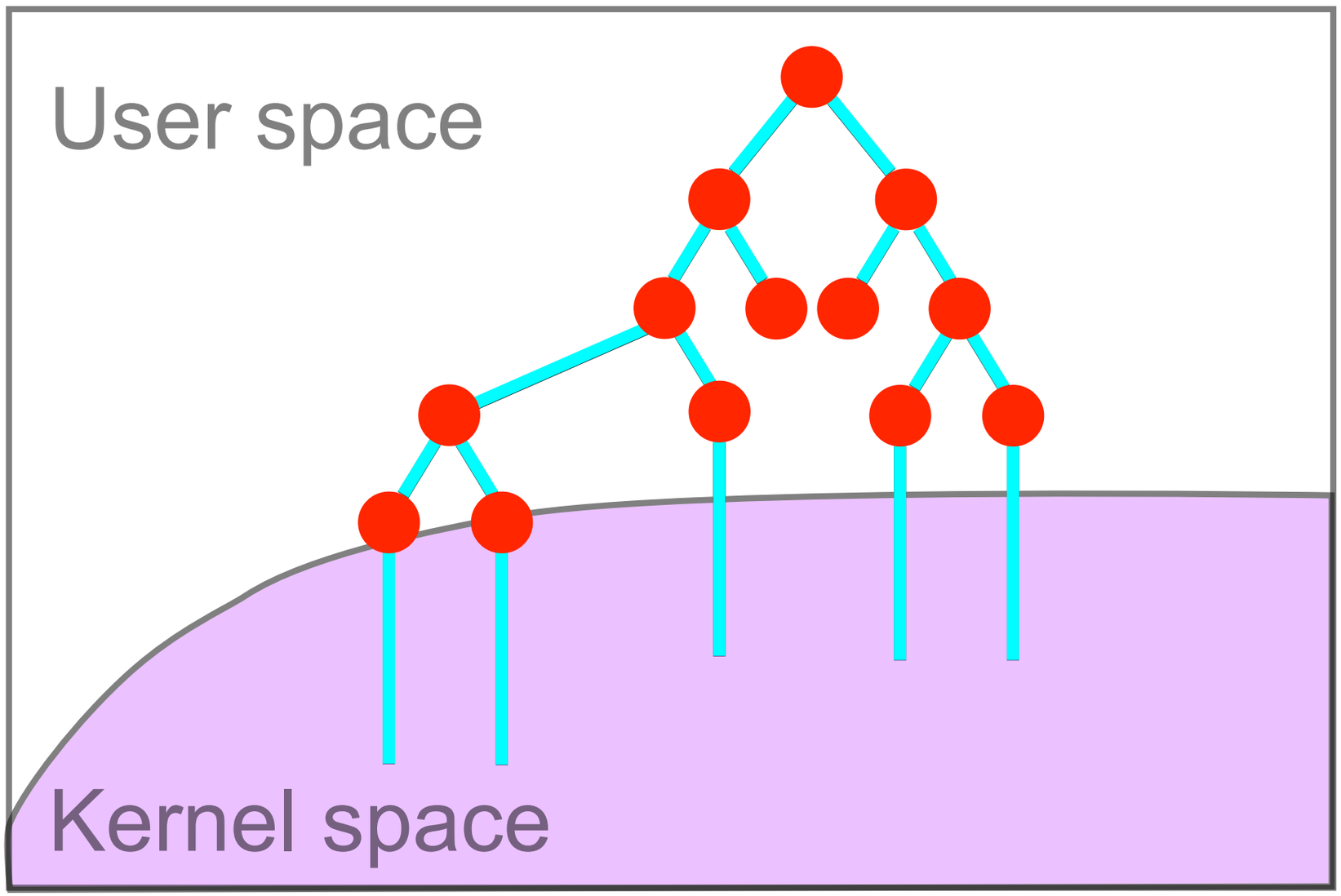
Single-Path Performance Profiling



Multi-Path In-Vivo Profiling



Multi-Path In-Vivo Profiling



PROF$_S$

- Cache Simulator
  Models arbitrary cache hierarchies
- Instruction Counter
  Machine instructions
- MMU Monitor
  Tracks TLB misses and page faults

Finding Performance Envelopes

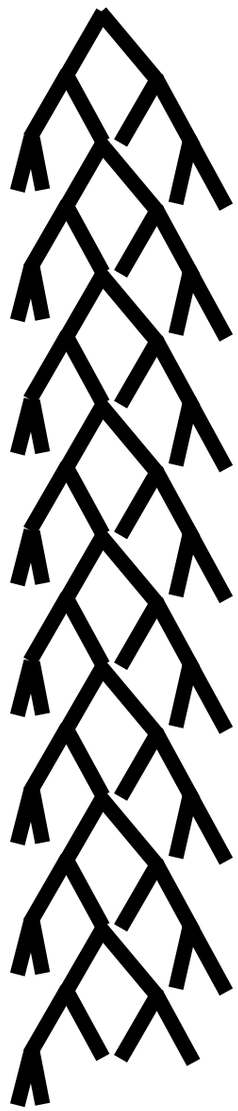
- Upper and lower bound on performance
- Fastest and slowest execution path
- Metrics?
  - # instructions, cache misses, page faults, ...

Finding Performance Envelopes

ping

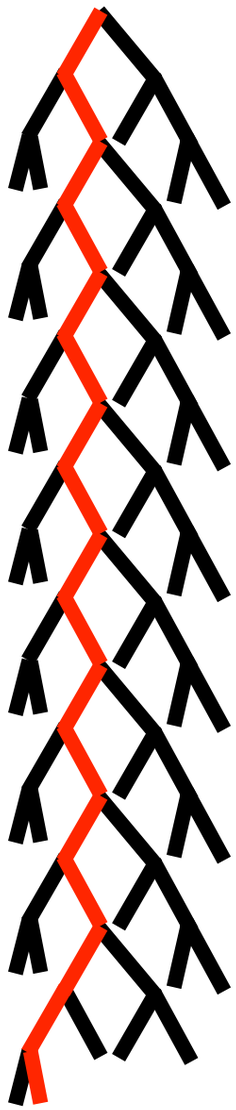
Finding Performance Envelopes

ping



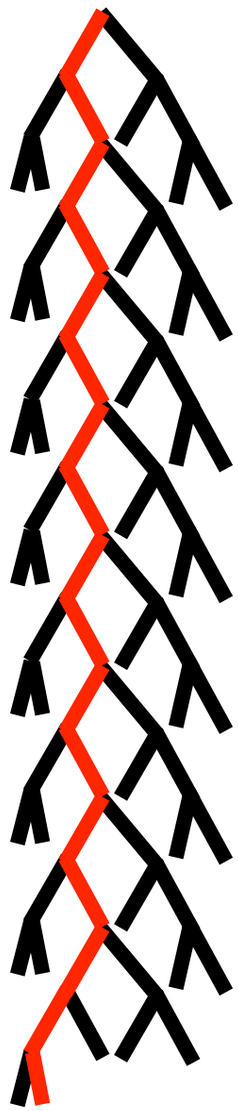
Finding Performance Envelopes

ping



Finding Performance Envelopes

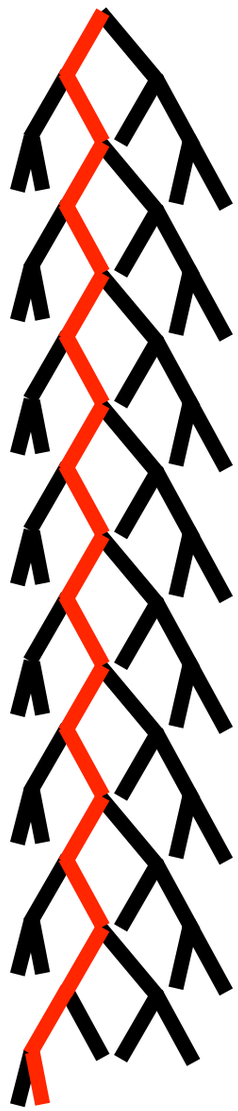
ping



>1.5 million instructions

Finding Performance Envelopes

ping



- Unbounded instruction count
- Infinite loop bug

>1.5 million instructions

Infinite Loop in Ping

```
void process_options(optptr...) {
  …
  while (totlen > 0) {
    …
    opt = optptr;
    …
    switch (*opt) {
      case OPTION_ROUTE_RECORD:
        length = *++opt;

        if (length < 4)
          continue;
    }
    …
  }
}
```

Infinite Loop in Ping

```
void process_options(optptr...) {
  …
  while (totlen > 0) {
    …
    opt = optptr;
    …
    switch (*opt) {
      case OPTION_ROUTE_RECORD:
        length = *++opt;

        if (length < 4)
          continue;
    }
    …
  }
}
```

Infinite Loop in Ping

```
void process_options(optptr...) {
  …
  while (totlen > 0) {
    …
    opt = optptr;
    …
    switch (*opt) {
      case OPTION_ROUTE_RECORD:
        length = *++opt;

        if (length < 4)
          continue;
    }
    …
  }
}
```

Infinite Loop in Ping

```
void process_options(optptr...) {
  …
  while (totlen > 0) {
    …
    opt = optptr;
    …
    switch (*opt) {
      case OPTION_ROUTE_RECORD:
        length = *++opt;

        if (length < 4)
          continue;
    }
    …
  }
}
```

Infinite Loop in Ping

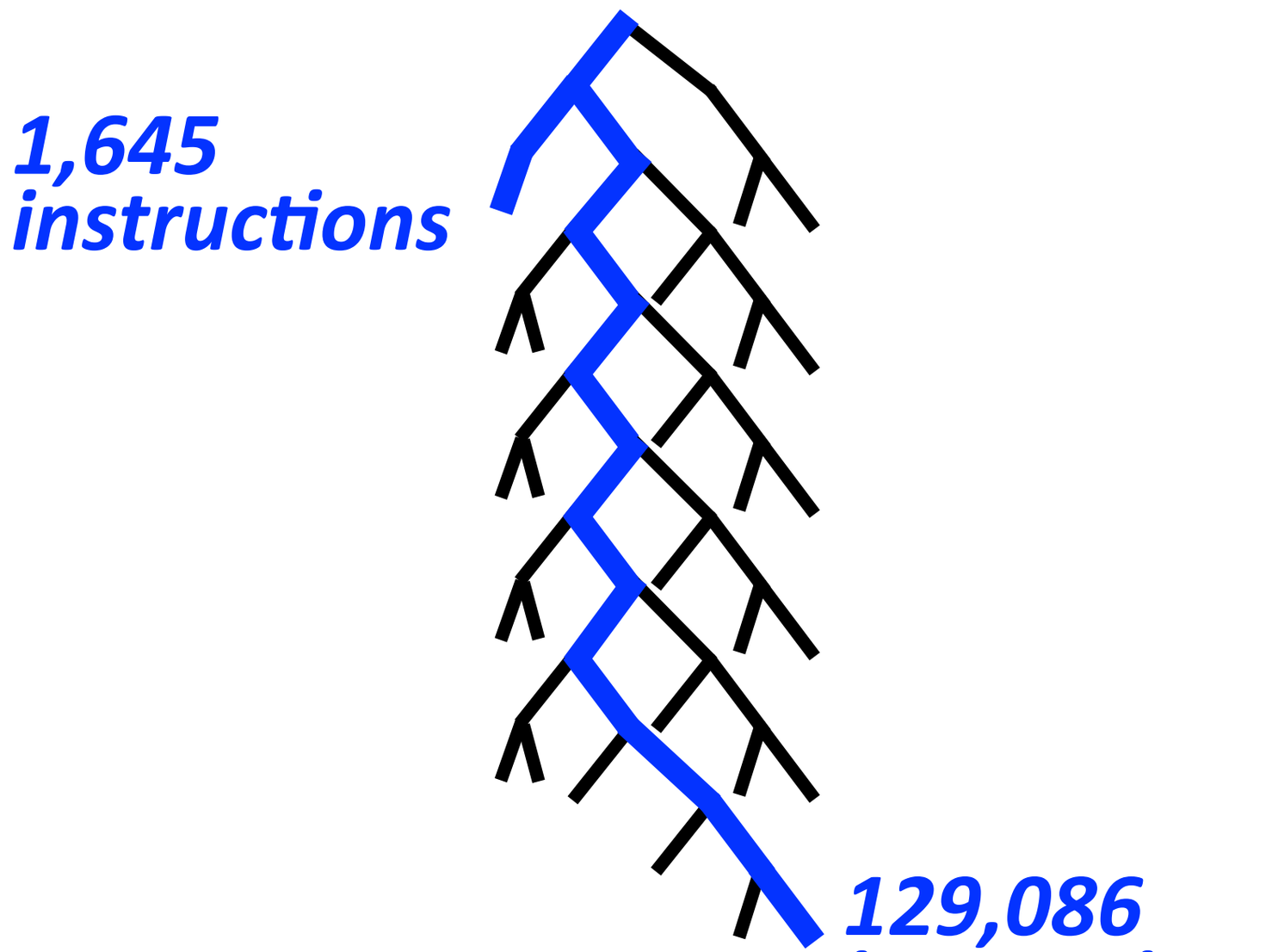
```
void process_options(optptr...) {
  …
  while (totlen > 0) {
    …
    opt = optptr;
    …
    switch (*opt) {
      case OPTION_ROUTE_RECORD:
        length = *++opt;

        if (length < 4)
          continue;
    }
    …
  }
}
```

Infinite Loop in Ping

```
void process_options(optptr...) {
  …
  while (totlen > 0) {
    …
    opt = optptr;
    …
    switch (*opt) {
      case OPTION_ROUTE_RECORD:
        length = *++opt;

        if (length < 4)
          continue;
    }
    …
  }
}
```

Perf. Envelope for Patched Ping

Perf. Envelope for Patched Ping



Conclusion

- Execution consistency models
- Platform for in-vivo multi-path analysis
- Use of symbolic execution in bug finding, reverse engineering, and performance analysis

http://s2e.epfl.ch

Ready-for-use VM, demos, tutorials, source code, documentation