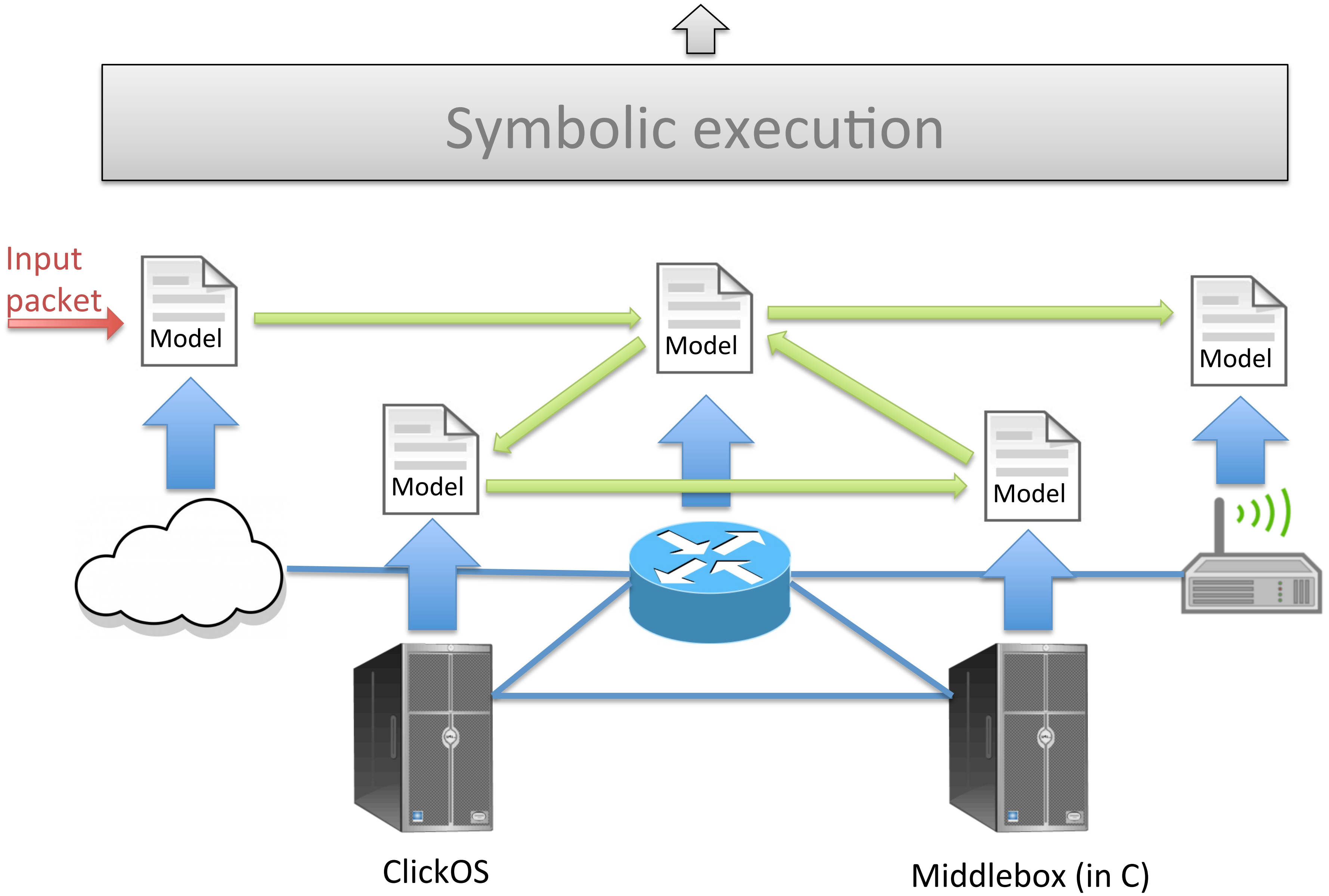
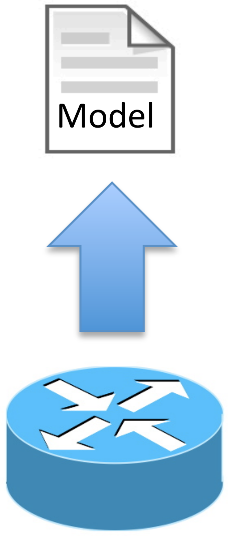
Symbolic execution model equivalence & applications

Matei Popovici, Radu Stoenescu,
Lorina Negreanu, Costin Raiciu

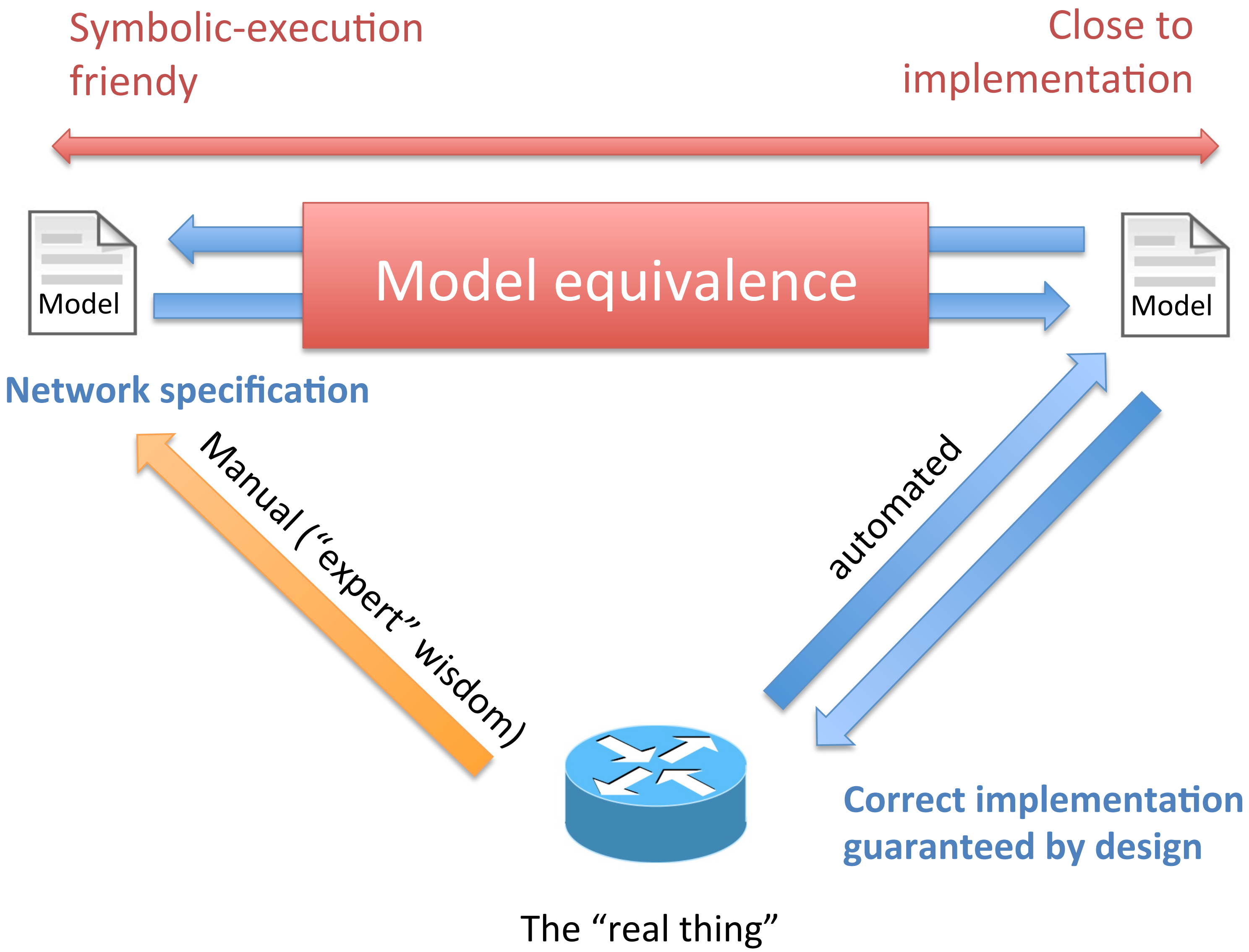
Network behavior



Network behavior



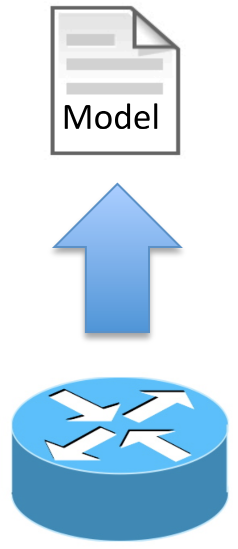
Network verification is as good as the model !


Symbolic-execution friendly
Close to implementation
Model
Model equivalence
Model
Network specification
Manual (“expert” wisdom)
automated
Correct implementation guaranteed by design
The “real thing”

Model equivalence

Symbolic execution

SYMNET – Symbolic execution Engine[1]



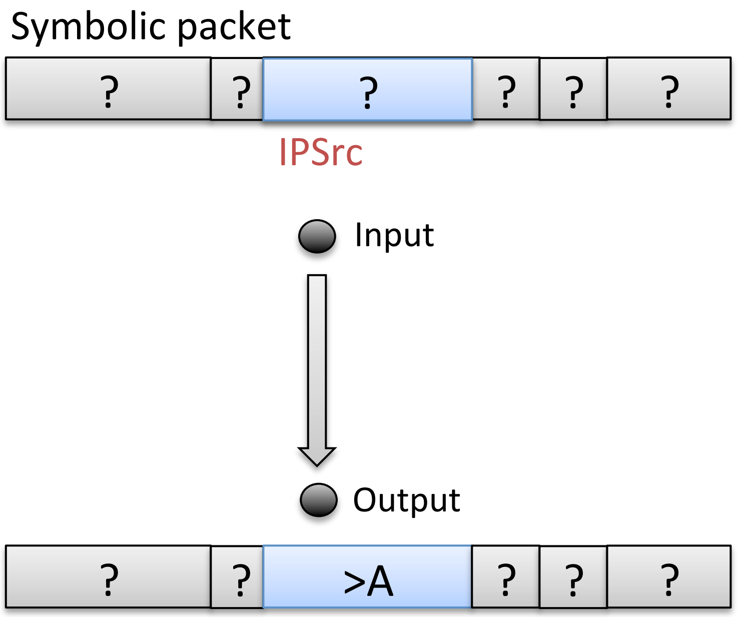
SEFL (Symbolic-Execution Friendly Language)[1]

1 – SYMNET: Scalable Symbolic Execution for modern networks (SIGCOMM 2016)

SEFL models

Constrain

`Constrain(IPSrc,>A)`

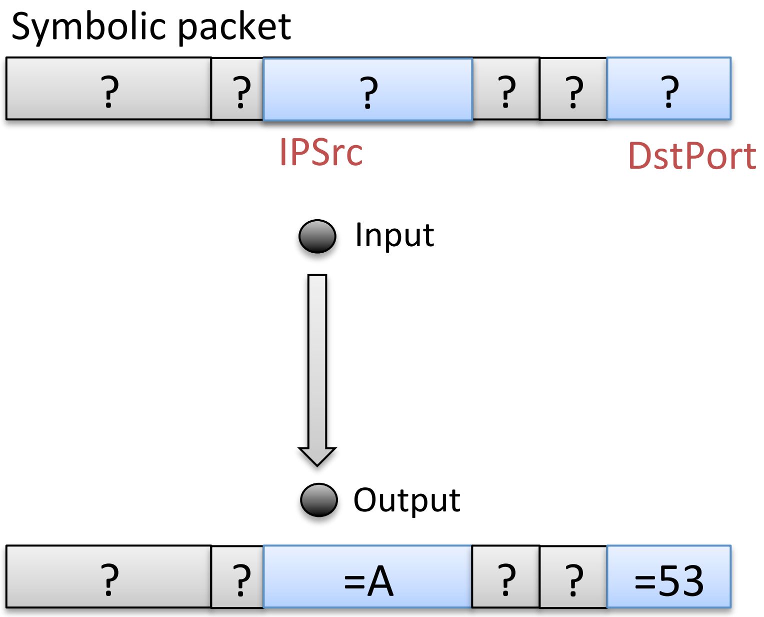


SEFL models

`InstructionBlock`

```
InstructionBlock(
   Constrain(IPSrc,=A),
   Constrain(DstPort,=53)
   )
```

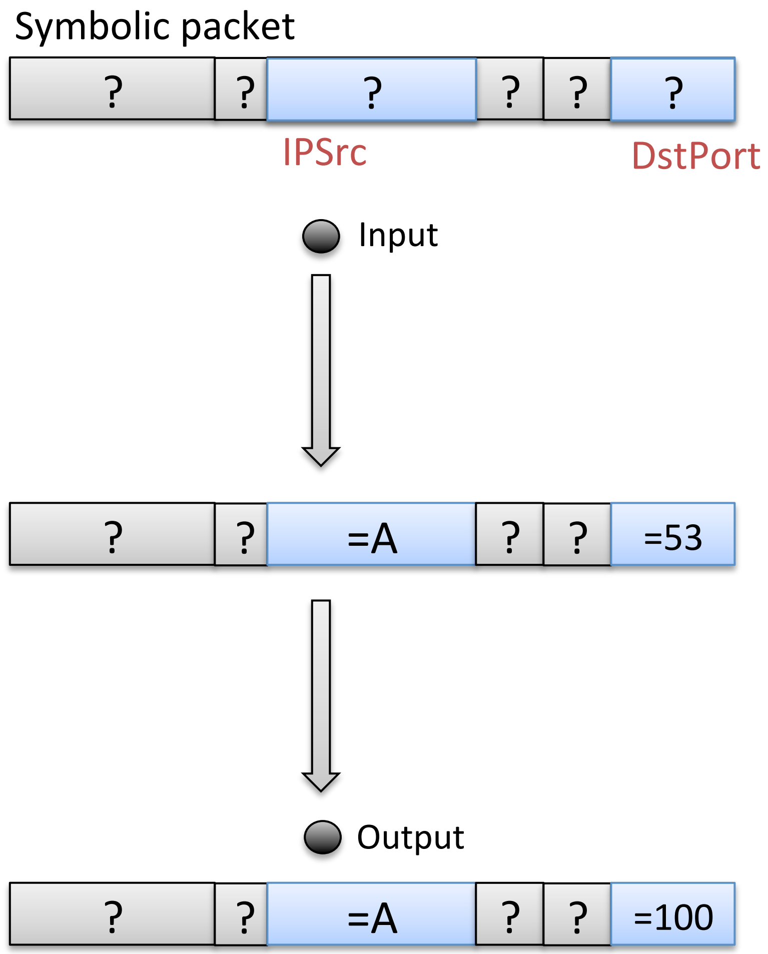
```
Constrain(IPSrc,=A);
Constrain(DstPort,=53)
```



SEFL models

`Assign`

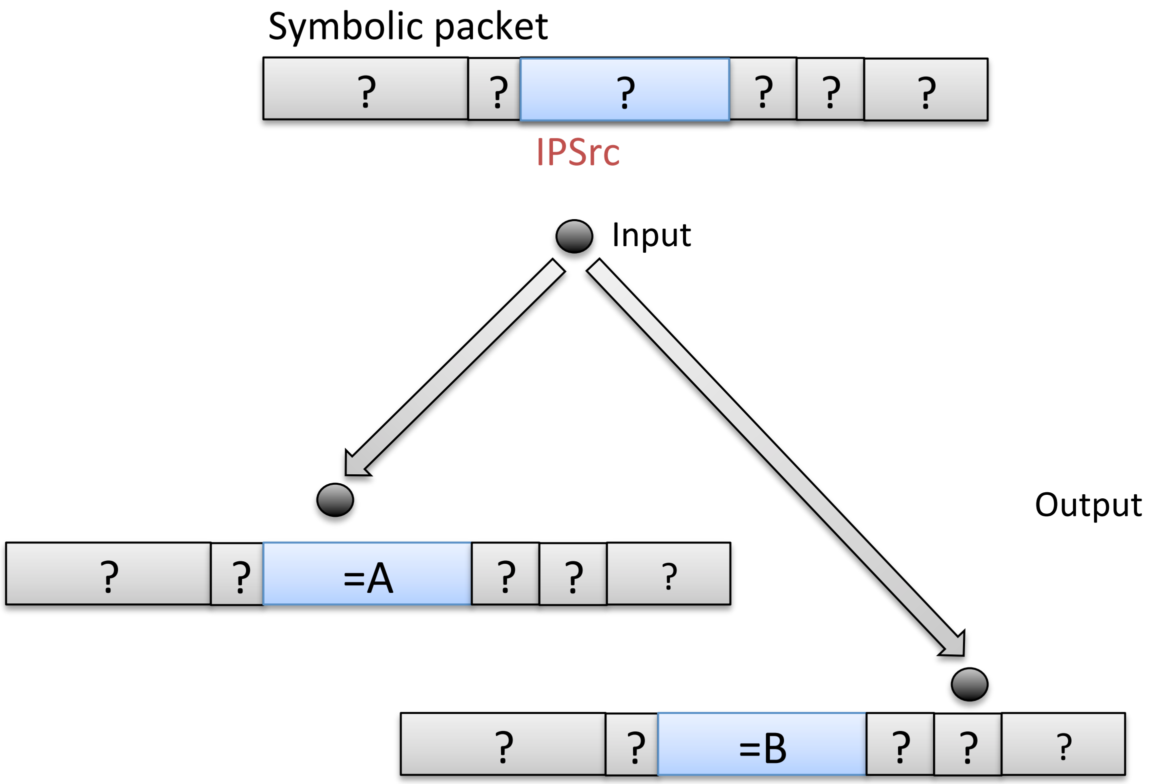
```
Constrain(IPSrc,=A);
Constrain(DstPort,=53);
Assign(DstPort,100)
```



SEFL models

Fork

```
Fork(
    Constrain(IPSrc,=A),
    Constrain(IPSrc,=B)
)
```

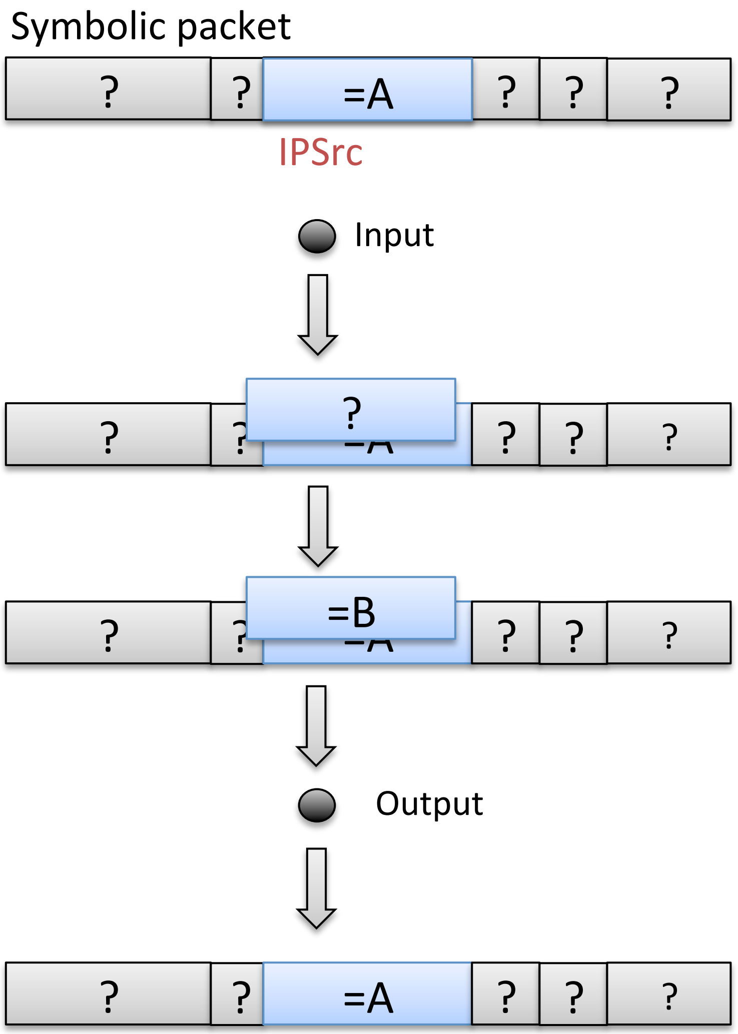


SEFL models

Allocate

```
Allocate(IPSrc);
Constrain(IPSrc,=B)
```

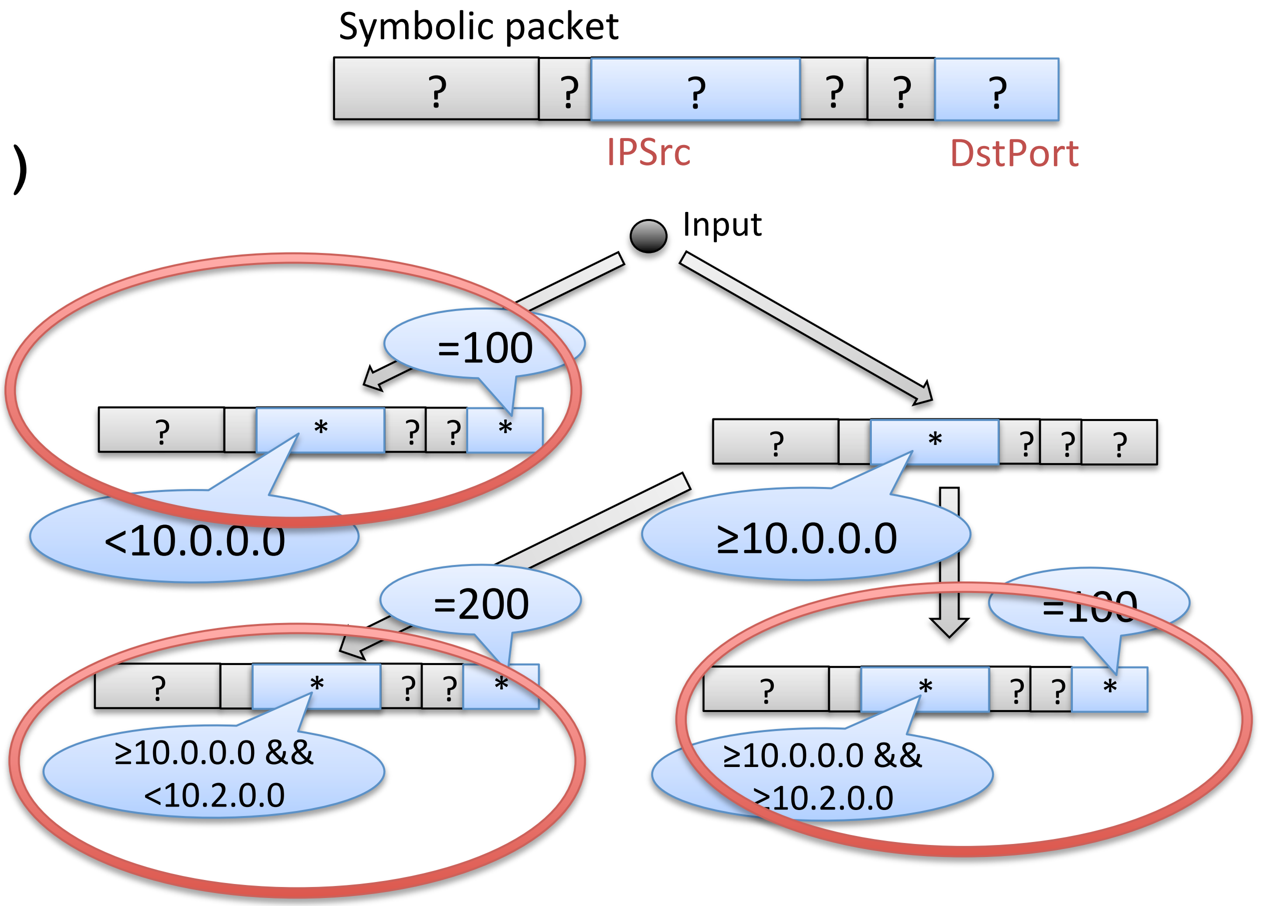
```
Deallocate(IPSrc)
```



SEFL models

`if (conditional)`

```
if (IPSrc,<10.0.0.0)
  Assign(DstPort,100)
  if (IPSrc,<10.2.0.0)
     Assign(DstPort,200)
     Assign(DstPort,100)
```



```
Fork(
    Constrain(IPSrc,<10.0.0.0 || ≥10.2.0.0);
    Assign(DstPort,100),

    Constrain(IPSrc,≥10.0.0.0 && <10.2.0.0);
    Assign(DstPort,200),

)
```

Model equivalence – Step 1. syntactic tr.

p

```
if (IPSrc,<10.0.0.0)
  Assign(DstPort,100)
  if (IPSrc,<10.2.0.0)
     Assign(DstPort,200)
       sign(DstPort,100)
```

```
      if (x,=e) e1 e2

 Fork(Constrain(x, =e); e1,
      Constrain(x,!=e); e2)
```

q

```
Fork(
    Constrain(IPSrc,<10.0.0.0 || ≥10.2.0.0);
    Assign(DstPort,100),

    Constrain(IPSrc,≥10.0.0.0 && <10.2.0.0);
    Assign(DstPort,200),

)
```

Model equivalence – rewriting if's

p

```
Fork(
 Constrain(IPSrc,<10.0.0.0);
 Assign(DstPort,100),

 Constrain(IPSrc,≥10.0.0.0);
 Fork(
   Constrain(IPSrc,<10.2.0.0);
   Assign(DstPort,200),

   Constrain(IPSrc,≥10.2.0.0);
   Assign(DstPort,100)
  )
)
```

q

```
Fork(
    Constrain(IPSrc,<10.0.0.0 || ≥10.2.0.0);
    Assign(DstPort,100),

    Constrain(IPSrc,≥10.0.0.0 && <10.2.0.0);
    Assign(DstPort,200),

)
```

Model equivalence – flattening Fork

p

```
Fork(
 Constrain(IPSrc,<10.0.0.0);
 Assign(DstPort,100),

 Constrain(IPSrc,≥10.0.0.0);
 Constrain(IPSrc,<10.2.0.0);
 Assign(DstPort,200),

 Constrain(IPSrc,≥10.0.0.0);
 Constrain(IPSrc,≥10.2.0.0);
 Assign(DstPort,100)

)
```

q

```
Fork(
    Constrain(IPSrc,<10.0.0.0 || ≥10.2.0.0);
    Assign(DstPort,100),

    Constrain(IPSrc,≥10.0.0.0 && <10.2.0.0);
    Assign(DstPort,200),

)
```

Model equivalence – removing Assign

p

```
Fork(
 Constrain(IPSrc,<10.0.0.0);
 Constrain(DstPort,=100),

 Constrain(IPSrc,≥10.0.0.0);
 Constrain(IPSrc,<10.2.0.0);
 Constrain(DstPort,=200),

 Constrain(IPSrc,≥10.0.0.0);
 Constrain(IPSrc,≥10.2.0.0);
 Constrain(DstPort,=100)

)
```

q

```
Fork(
    Constrain(IPSrc,<10.0.0.0 || ≥10.2.0.0);
    Constrain(DstPort,=100),

    Constrain(IPSrc,≥10.0.0.0 && <10.2.0.0);
    Constrain(DstPort,=200),

)
```

Model equivalence - Step 2. complement

p

```
Fork(
 Constrain(IPSrc,<10.0.0.0);
 Constrain(DstPort,=100),

 Constrain(IPSrc,≥10.0.0.0 &&
                 <10.2.0.0);
 Constrain(DstPort,=200),

 Constrain(IPSrc,≥10.2.0.0);
 Constrain(DstPort,=100)

)
```

q

```
Fork(
    Constrain(IPSrc,<10.0.0.0 || ≥10.2.0.0);
    Constrain(DstPort,=100),

    Constrain(IPSrc,≥10.0.0.0 && <10.2.0.0);
    Constrain(DstPort,=200),

)
```

Model equivalence – Step 2. complement

p

```
Fork(
 Constrain(IPSrc,<10.0.0.0);
 Constrain(DstPort,=100),

 Constrain(IPSrc,≥10.0.0.0 &&
                 <10.2.0.0);
 Constrain(DstPort,=200),

 Constrain(IPSrc,≥10.2.0.0);
 Constrain(DstPort,=100)

)
```

~q

```
Fork(
    Constrain(IPSrc,≥10.0.0.0 && <10.2.0.0);
    Constrain(IPSrc,<10.0.0.0 || ≥10.2.0.0),

    Constrain(IPSrc,≥10.0.0.0 && <10.2.0.0);
    Constrain(DstPort,!=200),

    Constrain(DstPort,!=100);
    Constrain(IPSrc,<10.0.0.0 || ≥10.2.0.0),

    Constrain(DstPort,!=100);
    Constrain(DstPort,!=200)

)
```

Model equivalence – Step 3. SE

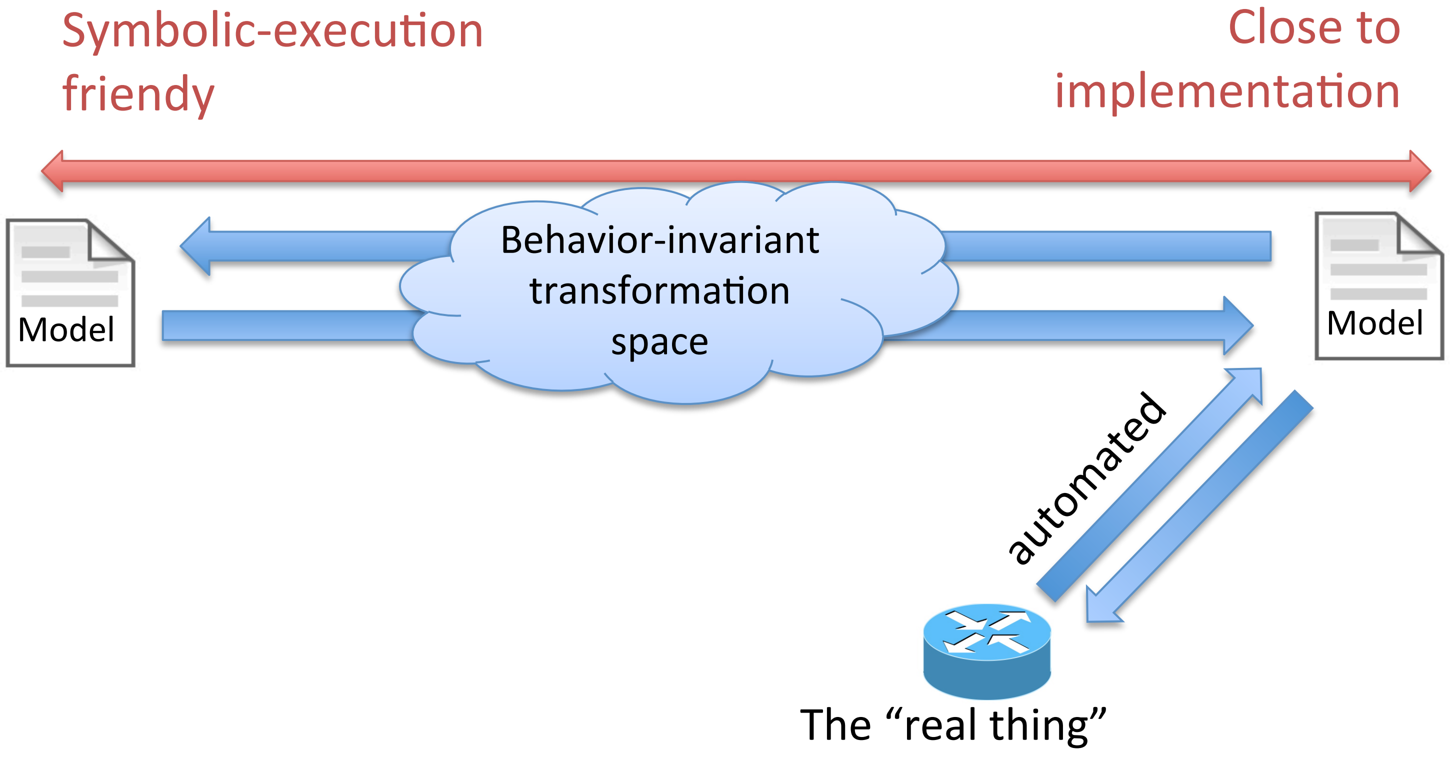
```
p ;~q
```

```
Fork(
 Constrain(IPSrc,<10.0.0.0);
 Constrain(DstPort,=100),

 Constrain(IPSrc,≥10.0.0.0 &&
                 <10.2.0.0);
 Constrain(DstPort,=200),

 Constrain(IPSrc,≥10.2.0.0);
 Constrain(DstPort,=100)

)
```

```
Fork(
    Constrain(IPSrc,≥1[illegible] && <10.2.0.0);
    Constrain(IPSrc,<1[illegible]0 || ≥10.2.0.0),

    Constrain(IPSrc,≥10.0.0.0 && <10.2.0.0);
    Constrain(DstPort,!=200),

    Constrain(DstPort,!=100);
    Constrain(IPSrc,<10.0.0.0 || ≥10.2.0.0),

    Constrain(DstPort,!=100);
    Constrain(DstPort,!=200)

)
```

Conclusion

- Model equivalence for SEFL
- Idea in short:
  1. Syntactic transformations to expose SEFL structure
  2. Symbolic execution


Symbolic-execution friendy
Close to implementation
Model
Behavior-invariant transformation space
Model
automated
The "real thing"

Model equivalence

- Cannot rely **solely** on symbolic execution
- Context (in)dependence

$$p = q$$

$$p_1;p;p_2 = p_1;q;p_2 \quad \text{for all } p_1,p_2$$

```
Allocate(x)            Allocate(x)
Constrain(x,=1)        Constrain(x,=2)
Allocate(x)
Constrain(x,=2)
```