

Securing Programs by Information Flow Control

Anoud Alshnakat
KTH Royal Institute of Technology



What is a "leak"?

A leak occurs when confidential data flows insecurely to a public output

Information flow control (IFC) checks that no secret can flow to a public output.

Noninterference:

A program is secure when ***secrets don't affect what the attacker sees.***

So, if you change a secret in the input, then the attacker sees no difference in the output!

Explicit flow

```
h.ipv4.ttl = secret;
```

+ Covert Channel

implicit flow

```
if (std_meta.ingress_port == 1 )  
  { std_meta.egress_spec = 5; }  
else  
  { std_meta.egress_spec = 6; }
```

Motivational example

$lbl \in \{H, L\}$

Input policy

If the ethernet type is IP_V4,
then the packet is public,
otherwise it is secret.
The **queue depth** is always secret.

Output policy

Program's output through **port 2** is
observable by the attacker.

Input port

```
if (h.eth.ethType == IP_V4) {  
    if (h.ipv4.dst [31:24] == 192 &&  
        h.ipv4.dst [23:16] == 168) {  
        if (q_depth >= 10)  
            h.ipv4.ecn = CONGESTION;  
        } else {  
            h.ipv4.ecn = UNSUPPORTED;  
        }  
    }  
    if (h.ipv4.dst[31:24] == 192) {  
        port = 1; //internal  
    } else {  
        port = 2; //external  
    }  
} else {  
    port = 1; //internal  
}
```

Port 1
Internal network
192.*.*.*

Port 2
External network



Solution overview

Interval
analysis

abstract program's
input and output states

Security
types

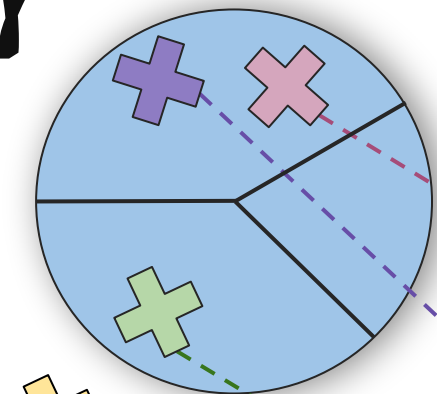
check how information
flows in a program

$\tau ::= \langle I, \text{lbl} \rangle$

- $I = [a, b]$
- $\text{lbl} \in \{\text{H}, \text{L}\}$

γ

State types



q_depth	ethType	rest of fields
$\ast, H$	$[0x0801, 0xFFFF], L$	$\ast, H$

q_depth	ethType	rest of fields
1	0x86DD	0000000000...

q_depth	ethType	rest of fields
2	0x86DD	1111111111...

q_depth	ethType	rest of fields
3	0xFFFF	0000000000...

q_depth	ethType	rest of fields
3	0x0000	0000000000...

$$p_1 \sim_{\gamma} p_2$$

Input policy cases

Input policy

If the ethernet type is IP_V4,
then the packet is public,
otherwise it is secret.
The **queue depth** is always secret.

 γ_i^1

q_depth	ethType	dst	ecn
*,H	0x0800,L	*,L	*,L

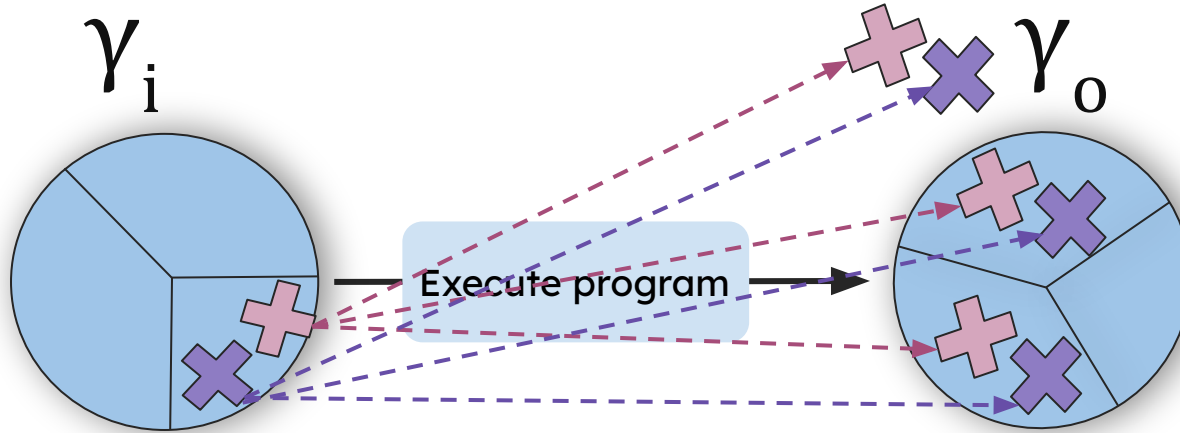
 γ_i^2

q_depth	ethType	dst	ecn
*,H	[0x0801,0xFFFF],L	*,H	*,H

 γ_i^3

q_depth	ethType	dst	ecn
*,H	[0x0000,0x0799],L	*,H	*,H

Noninterference



q_depth	ethType	rest of fields
*,H	[0x0801,0xFFFF],L	*,H

port	q_depth	rest of fields
1,H	*,H	*,H

q_depth	ethType	rest of packet
12	0x86DD	0000000000...

Program

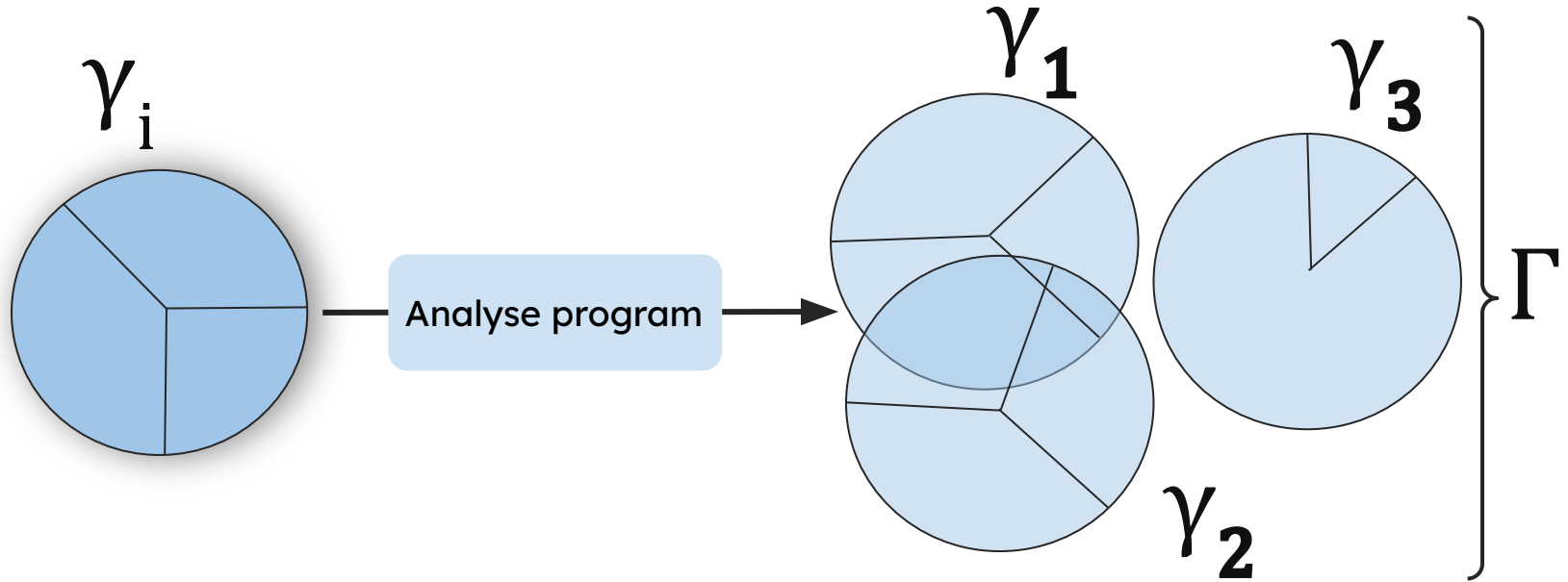
port	q_depth	rest of packet
1	1	0000000000...

q_depth	ethType	rest of packet
13	0x86DD	1111111111...

Program

port	q_depth	rest of packet
1	2	1111111111...

Enforcement overview



```

if (h.ipv4.dst[31:24] == 192 &&
    h.ipv4.dst[23:16] == 168) {
  if (q_depth >= 10)
    h.ipv4.ecn = 3;
} else {
  h.ipv4.ecn = 0;
}

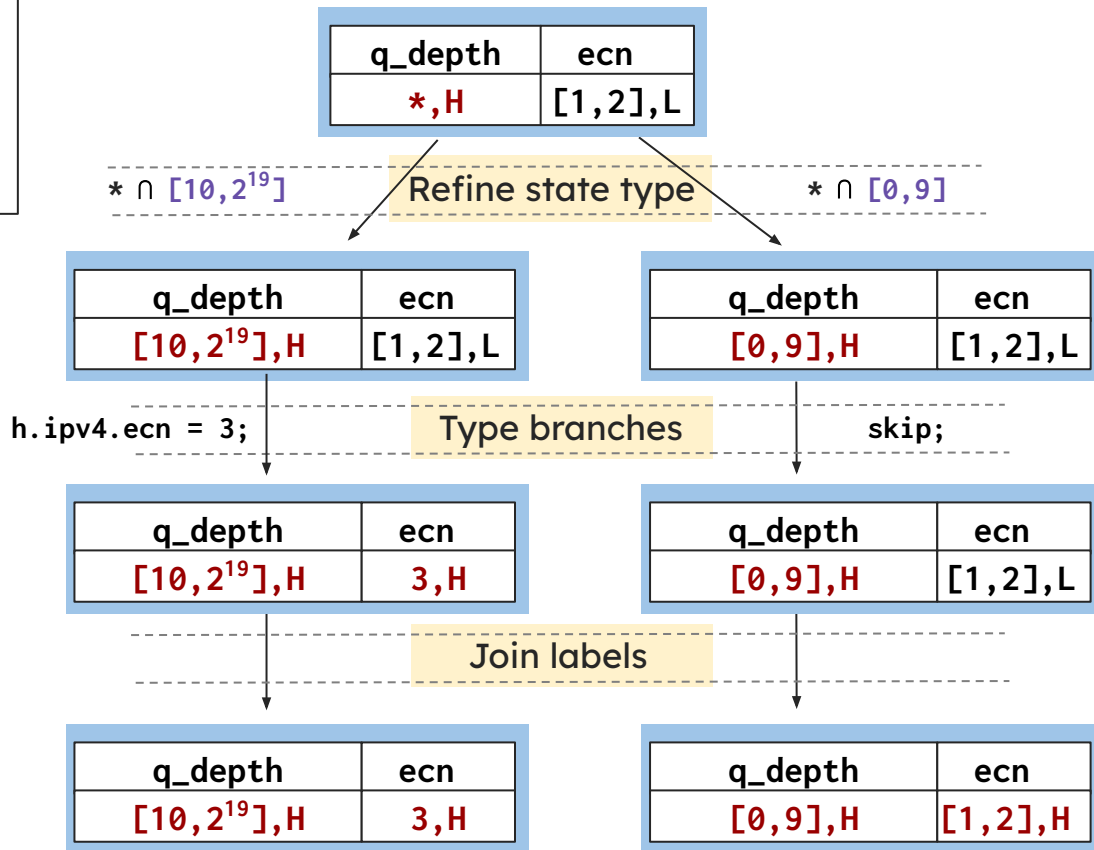
```

T-Cond

$\gamma \vdash e : \tau$
 $lbl = \text{getLabel}(\tau)$
 $pc' = pc \sqcap lbl$
 $pc', \text{refine}(\gamma, e) \vdash s_1 : \Gamma_1$
 $pc', \text{refine}(\gamma, \neg e) \vdash s_2 : \Gamma_2$
 $\Gamma_3 = \text{joinOnHigh}(\Gamma_1 \cup \Gamma_2, lbl)$

 $pc, \gamma \vdash \text{if } e \text{ then } s_1 \text{ else } s_2 : \Gamma_3$

Example



Example

```
if (h.ipv4.dst[31:24] == 192 &&
    h.ipv4.dst[23:16] == 168) {
    if (q_depth >= 10)
        h.ipv4.ecn = 3;
} else {
    h.ipv4.ecn = 0;
}
```

q_depth	ecn	dst
,H	[1,2],L	192..*.*,L

Refine state type

q_depth	ecn	dst
,H	[1,2],L	192.168..*,L

q_depth	ecn	dst
,H	[1,2],L	192.[169,255]..* ,L

if (q_depth >= 10) h.ipv4.ecn = 3;

Type branches

h.ipv4.ecn = 0;

q_depth	ecn	dst
[10,2 ¹⁹],H	3,H	192.168.*.*,L

q_depth	ecn	dst
[0,9],H	[1,2],H	192.168.*.*,L

q_depth	ecn	dst
,H	0,L	192.[169,255]..* ,L

Example

```

if (h.ipv4.dst[31:24] == 192 &&
    h.ipv4.dst[23:16] == 168) {
  if (q_depth >= 10)
    h.ipv4.ecn = 3;
} else {
  h.ipv4.ecn = 0;
}

```

T-Empty

$L, \bullet \vdash s : \Gamma$

q_depth	ecn	dst
,H	[1,2],L	192.168..*,L

Refine state type

q_depth	ecn	dst
,H	[1,2],L	192.168..*,L

if (q_depth >= 10) h.ipv4.ecn = 3;

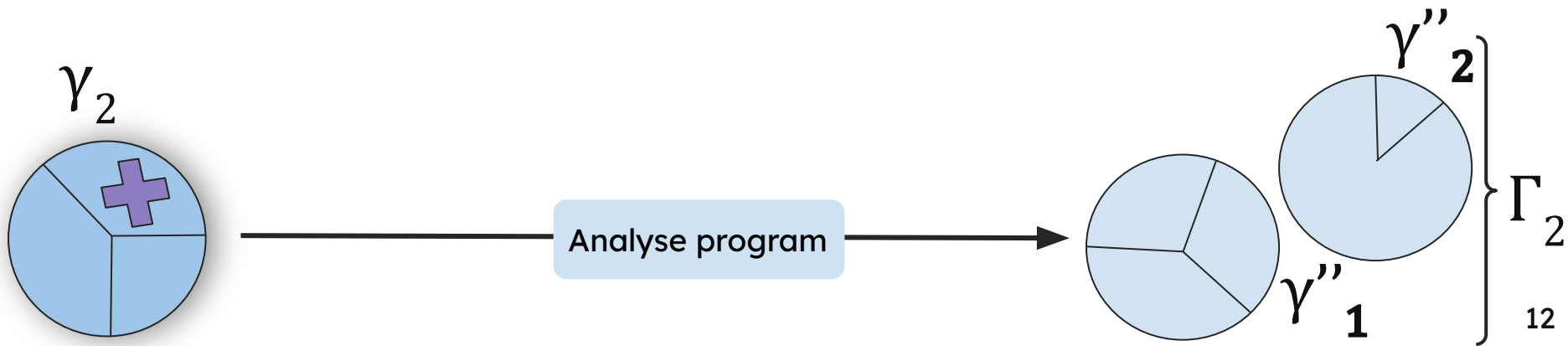
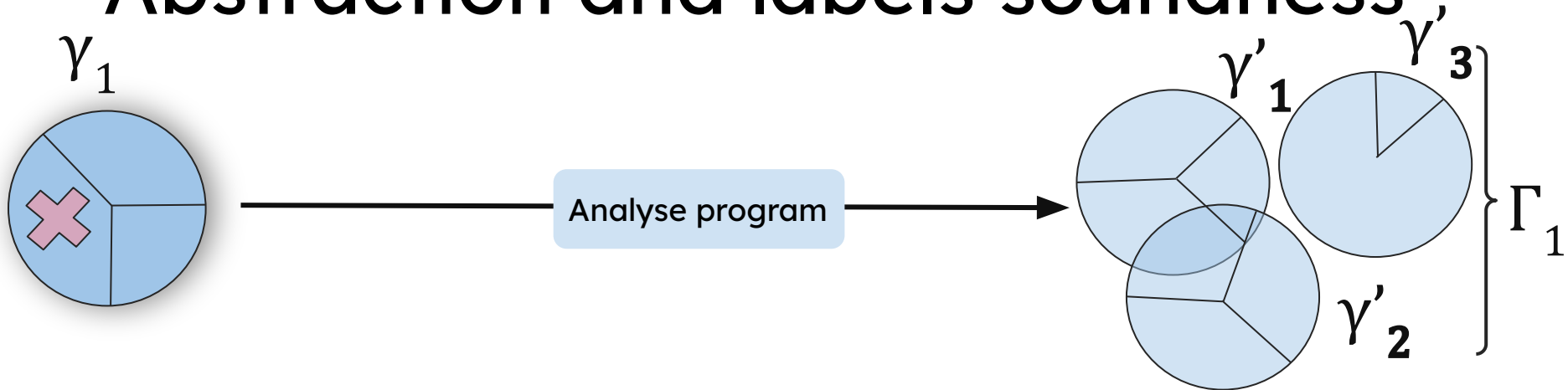
Type branches

h.ipv4.ecn = 0;

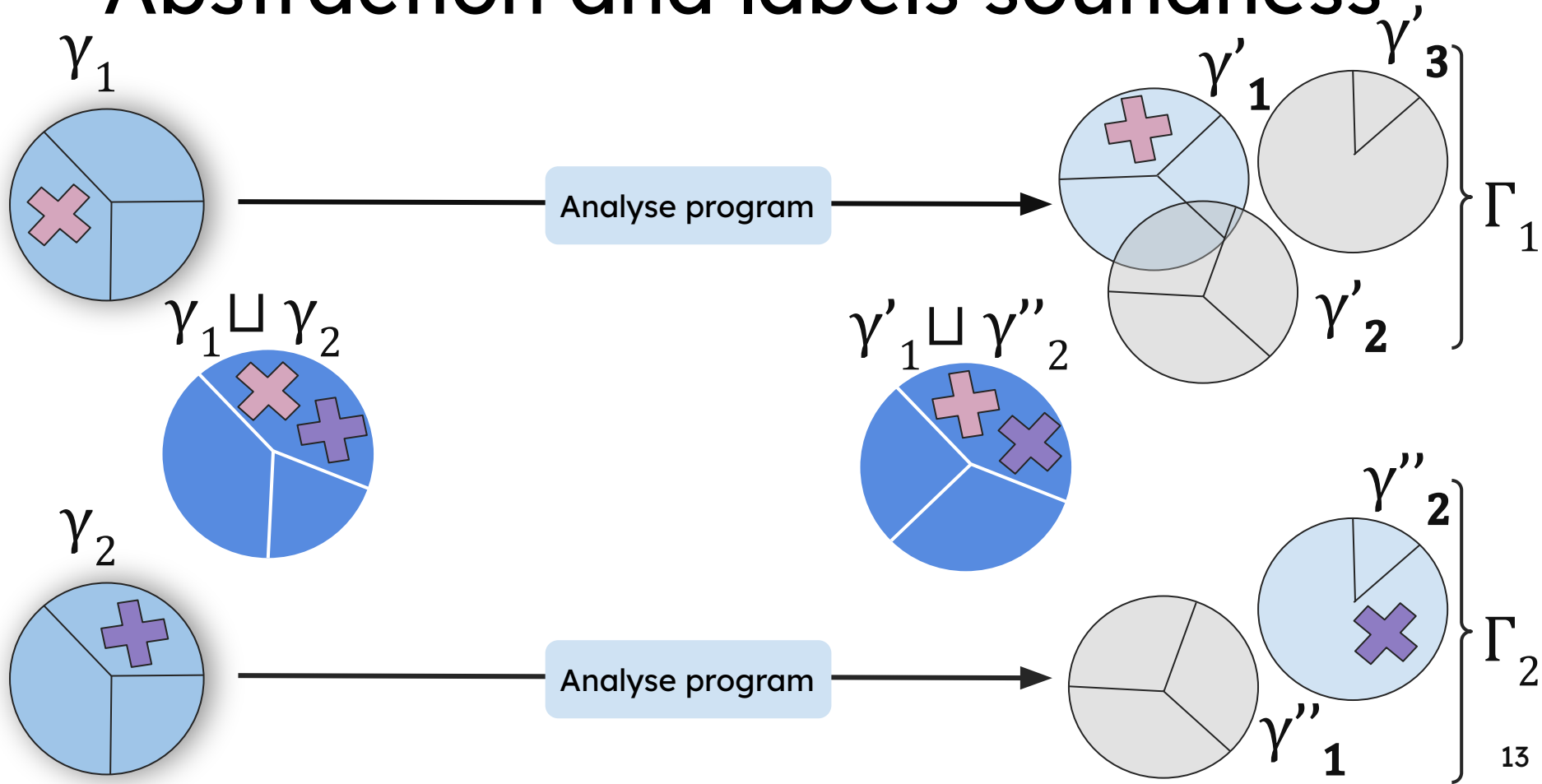
q_depth	ecn	dst
[10,2¹⁹],H	3,H	192.168.*.*,L

q_depth	ecn	dst
[0,9],H	[1,2],H	192.168.*.*,L

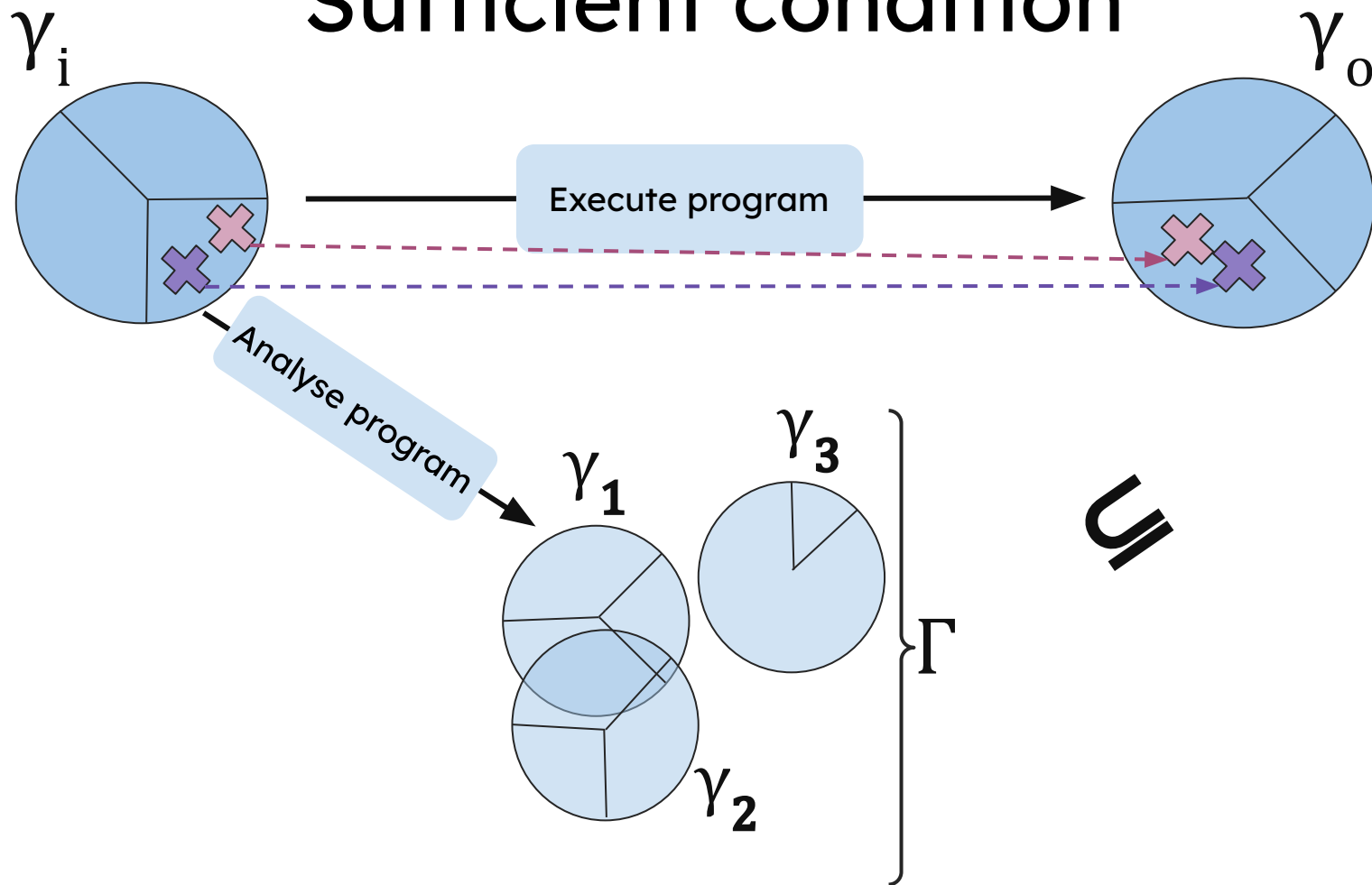
Abstraction and labels soundness



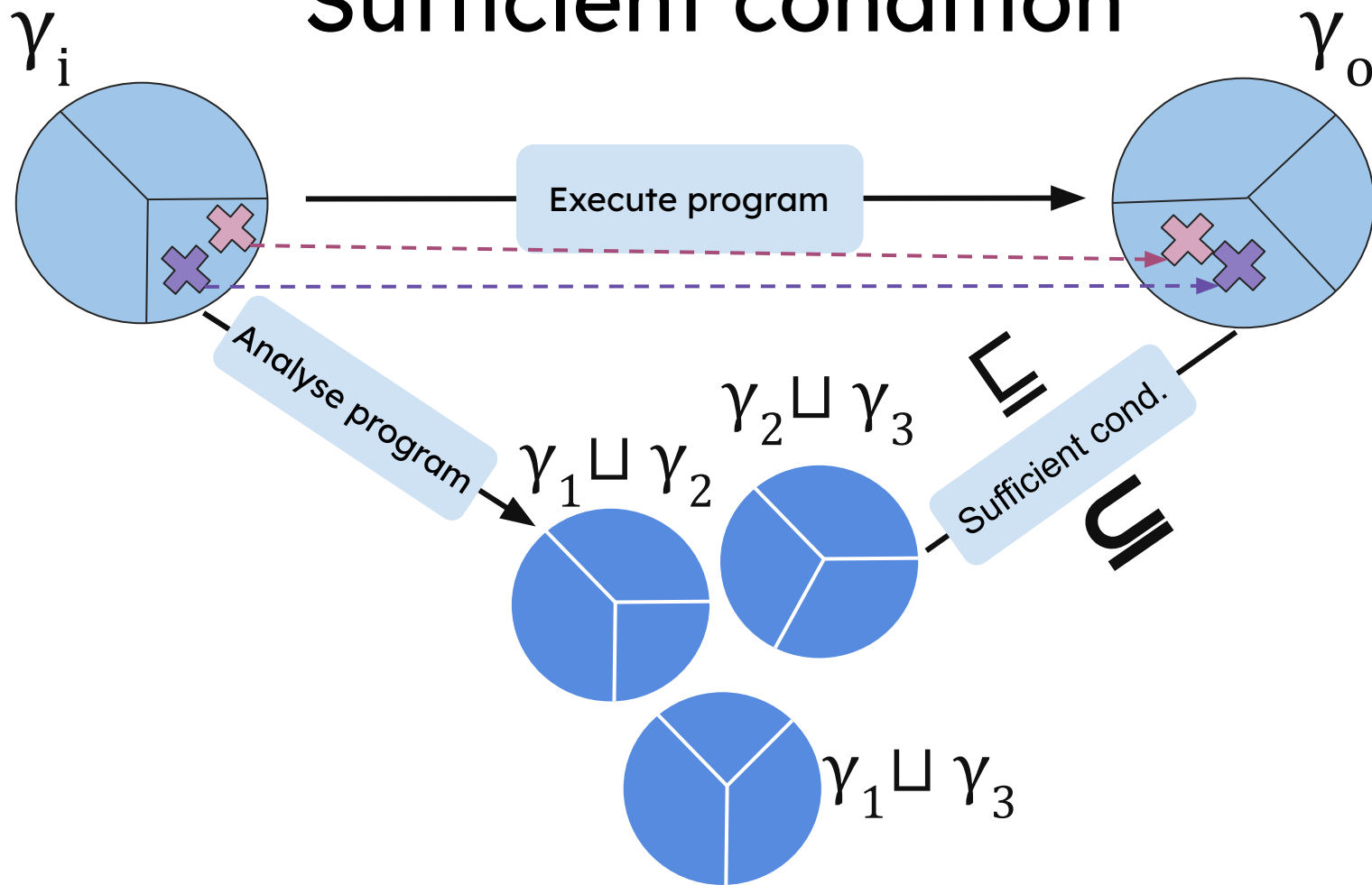
Abstraction and labels soundness



Sufficient condition



Sufficient condition



Evaluation of TAP4S

IFC for P4

P4

P4BID [Grewal et. al.]

- Does not handle data-dependent policies.
- Path *insensitive*
- Termination *insensitive*
- Requires code modifications.

P4BID

Network Isolation - Alice

243

27

62

3

Network Isolation - Top

242

23

63

3

Topology

202

40

4

3

Total time
[ms]

Typing time
[ms]

Security check
[ms]

Number of
final γ s

Evaluation of TAP4S

		Total time [ms]	Typing time [ms]	Security check [ms]	Number of final γ s
P4 language tutorials	Basic Congestion	5930	966	4794	97
	Basic Tunneling	610	157	290	15
	Multicast	199	16	23	6
	Firewall	4560	1015	3378	44
	MRI	7646	523	5957	23
P4BID	Data-plane Routing	247	109	8	12
	In-network Caching	261	91	14	6
	Resource Allocation	256	87	10	9
	Network Isolation - Alice	243	27	62	3
	Network Isolation - Top	242	23	63	3
	Topology	202	40	4	3

Conclusion

- Many realistic systems require **data-dependent policies**.
- Combining **abstract domains** and **security type systems** provides *expressive and precise analysis*.



Scan for full paper