

Prognosis: Black-Box Analysis of Network Protocol Implementations

Alexandra Silva

PROGNOSIS: Black-Box Analysis of Network Protocol Implementations

Tiago Ferreira
University College London

Harrison Brewton
University of Wisconsin–Madison

Loris D’Antoni
University of Wisconsin–Madison

Alexandra Silva
University College London

To appear at SIGCOMM’21

Semantics and Verification

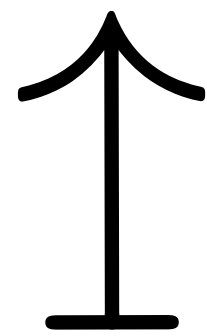
What does this program do?



Semantics and Verification



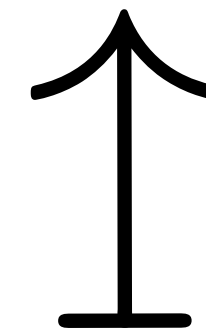
Semantics

 Σ 

High-level
language



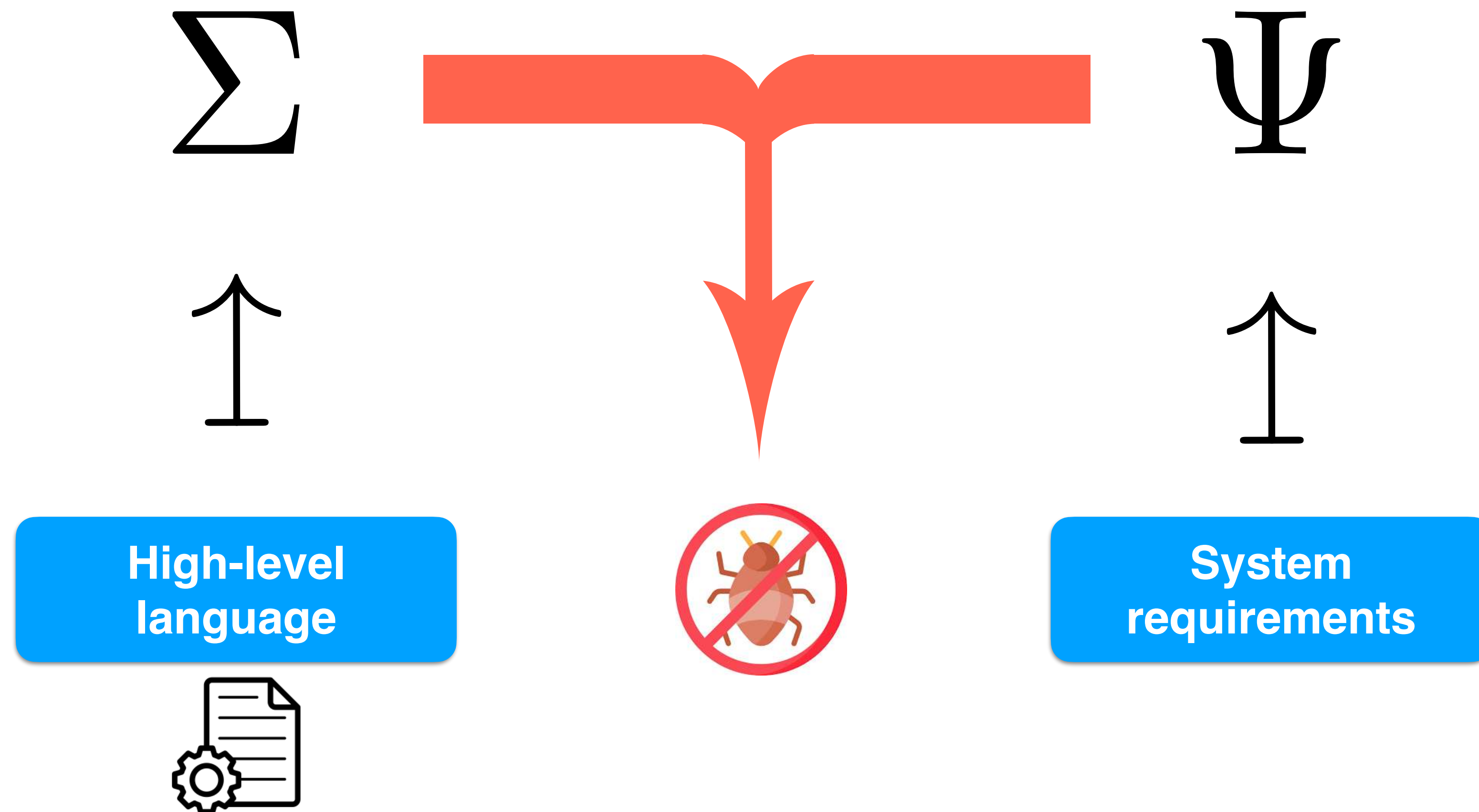
Verification

 Ψ 

System
requirements

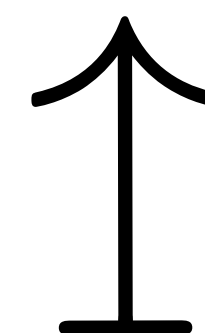
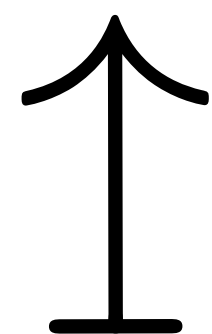
Semantics

Verification



Semantics

Verification

 Σ  Ψ 

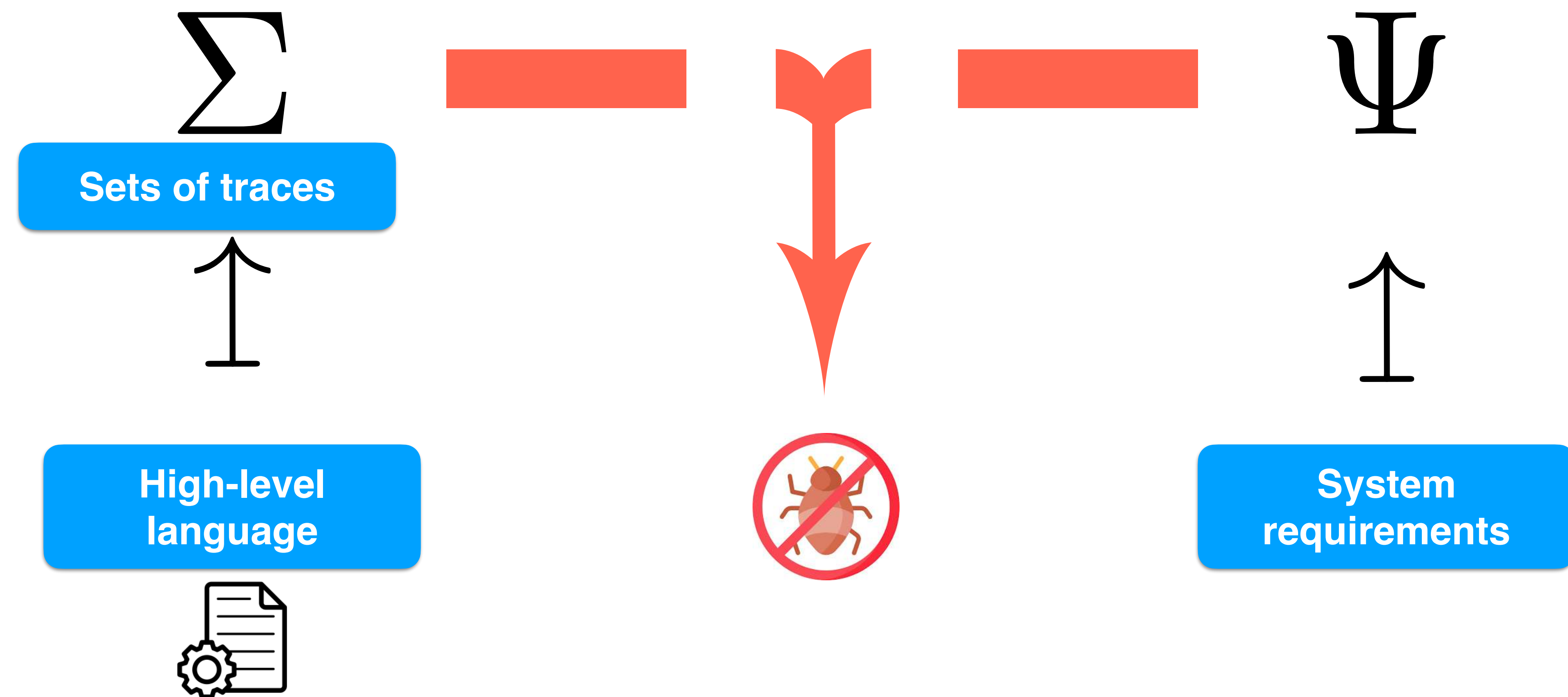
High-level
language



System
requirements

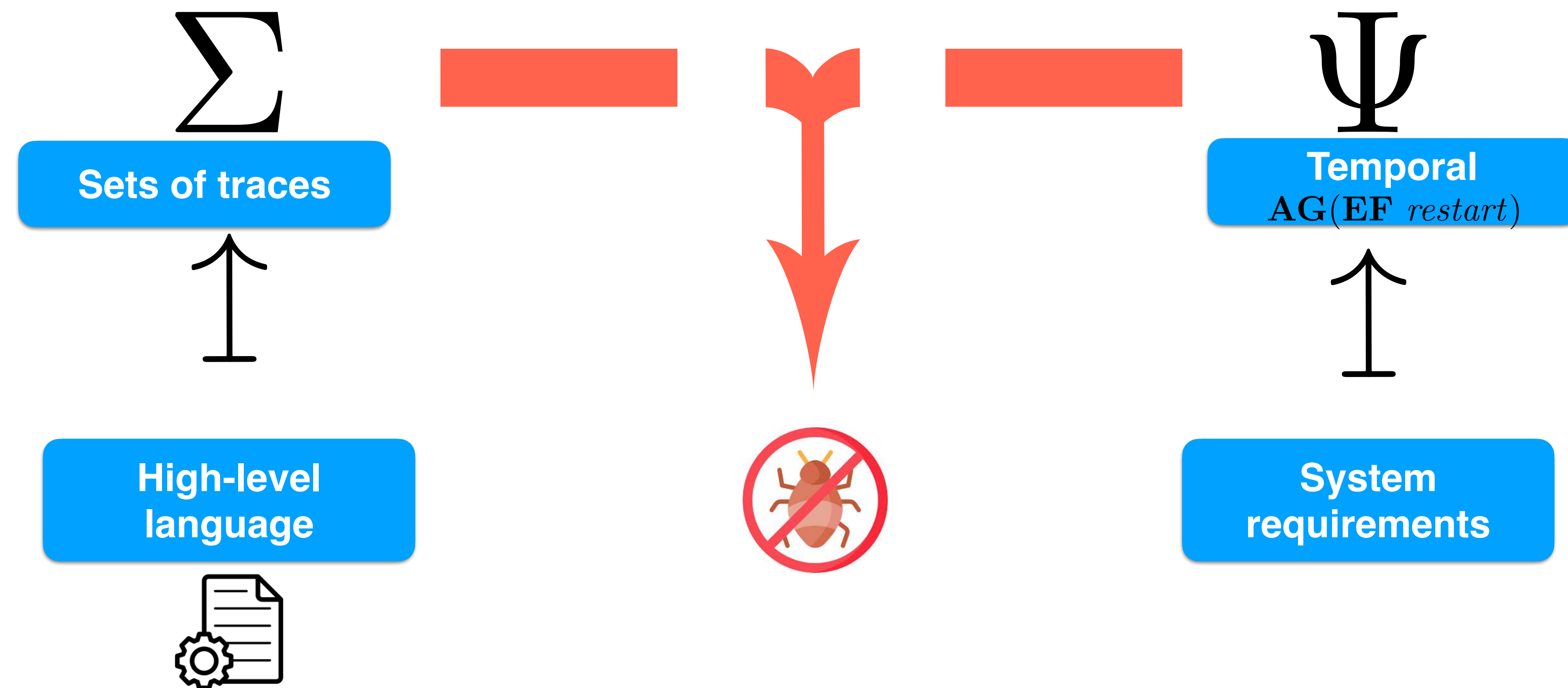
Semantics

Verification

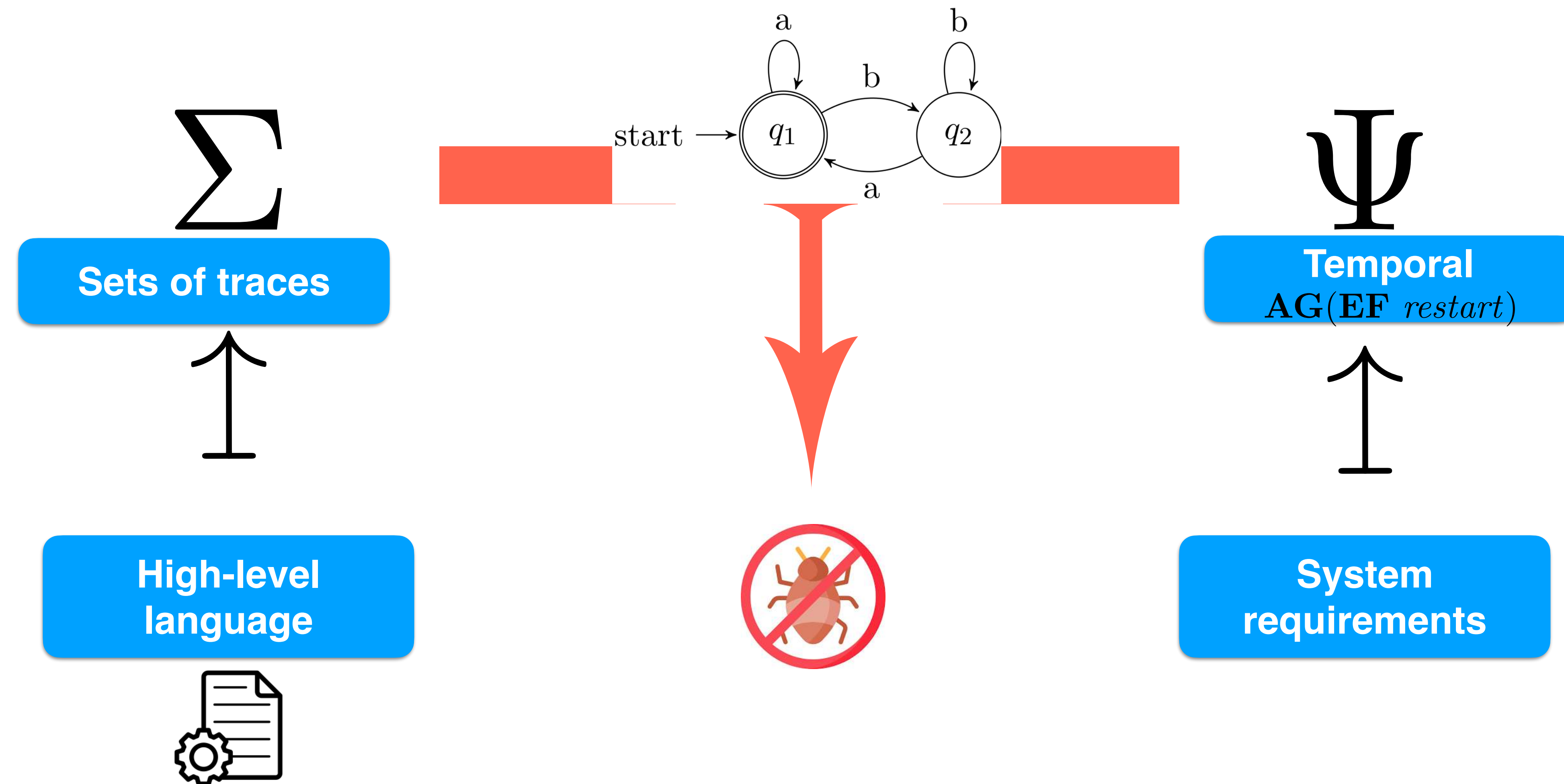


Semantics

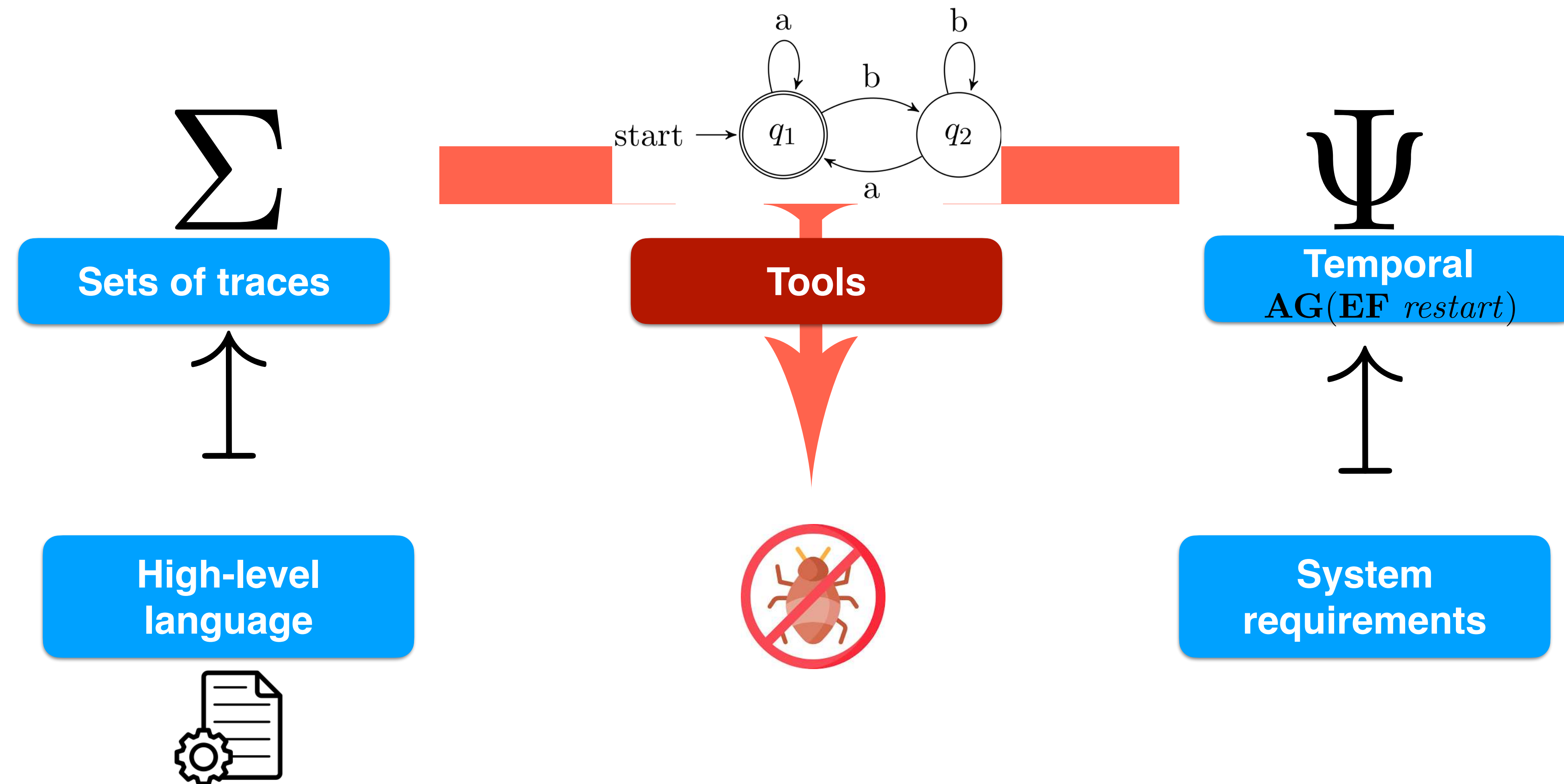
Verification



Semantics Automata Verification

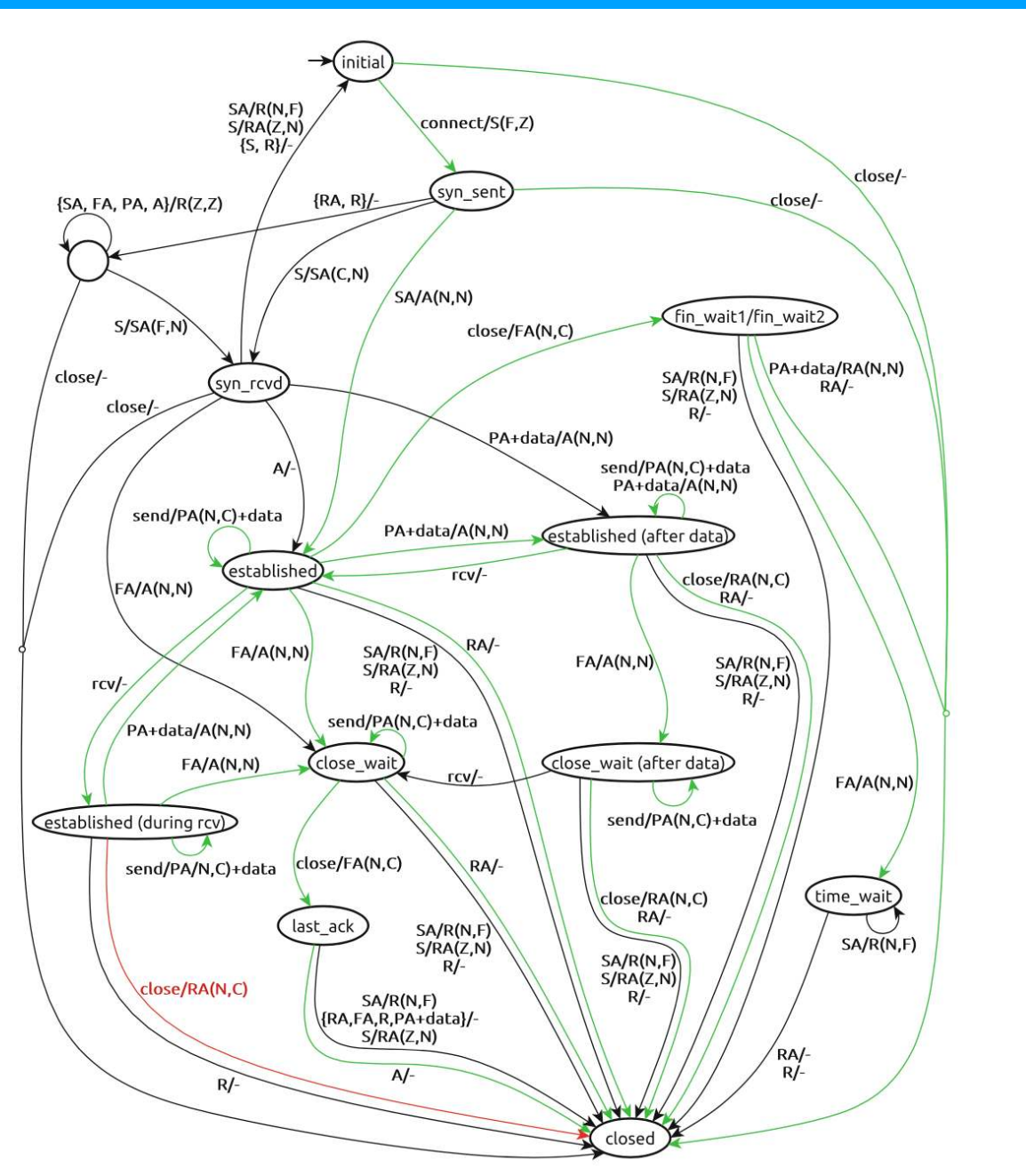


Semantics Automata Verification



Model Checking

Model of a system component / protocol



Model checker

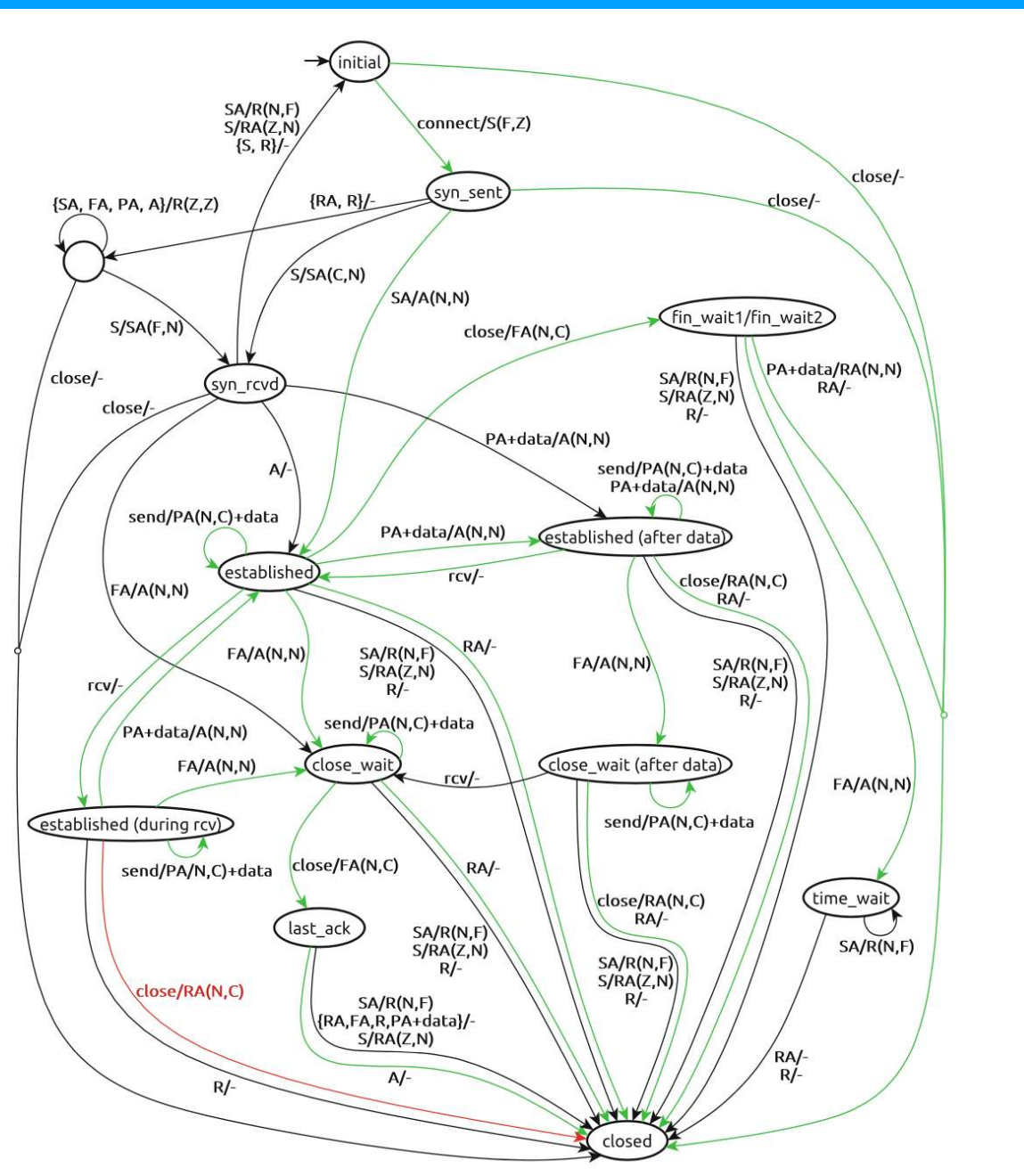
Property of interest

System requirements

Ψ

Model Checking

Model of a system component / protocol


$$\Psi$$

Property of interest

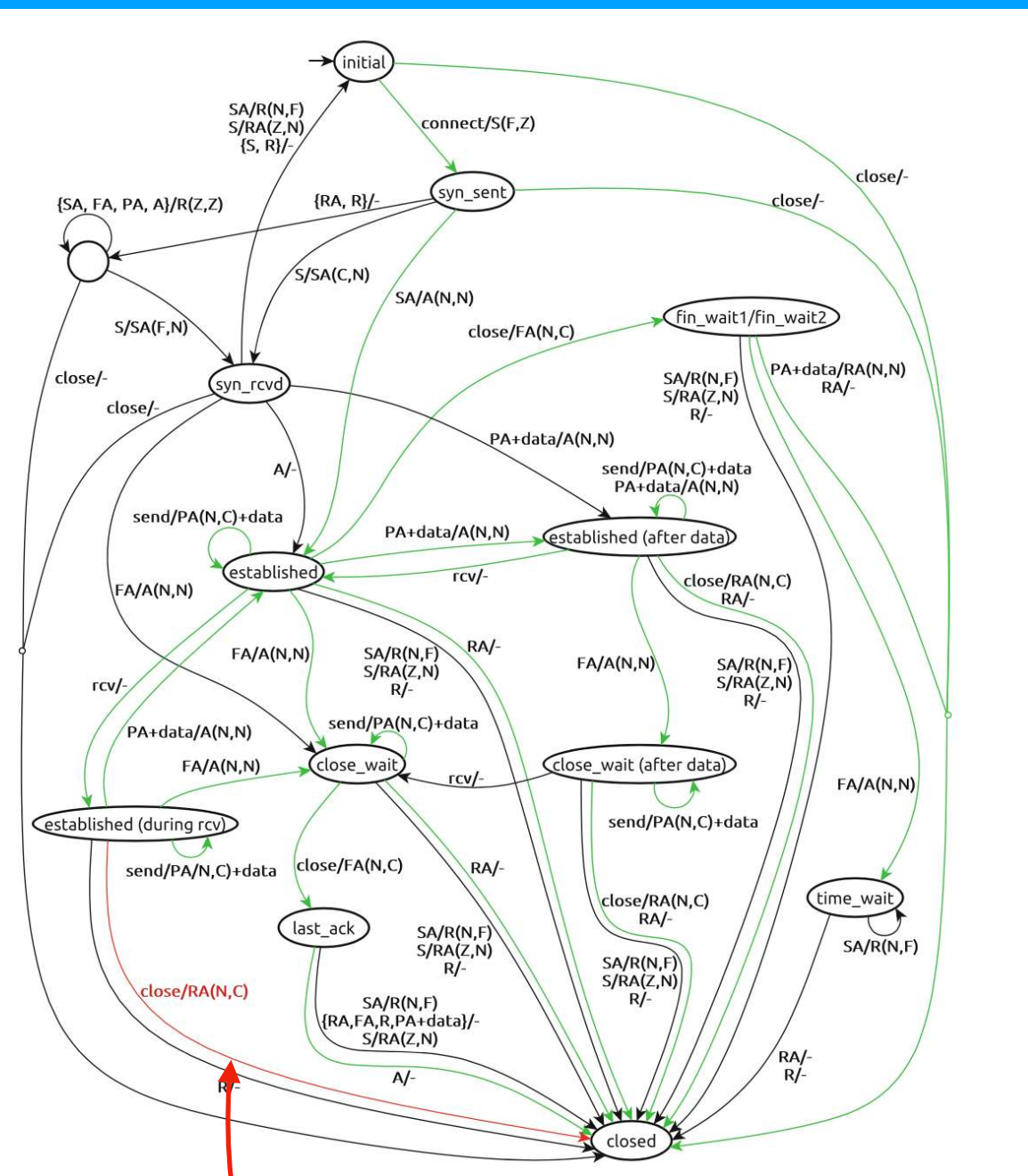
Model checker

System requirements



Model Checking

Model of a system component / protocol



+ Counter-example
trace

 Ψ

Property of interest

Model checker

System
requirements

Model Checking

Model of a system component / protocol

How do we design good models?
How do we maintain them?

Property of interest

Model checker

System
requirements

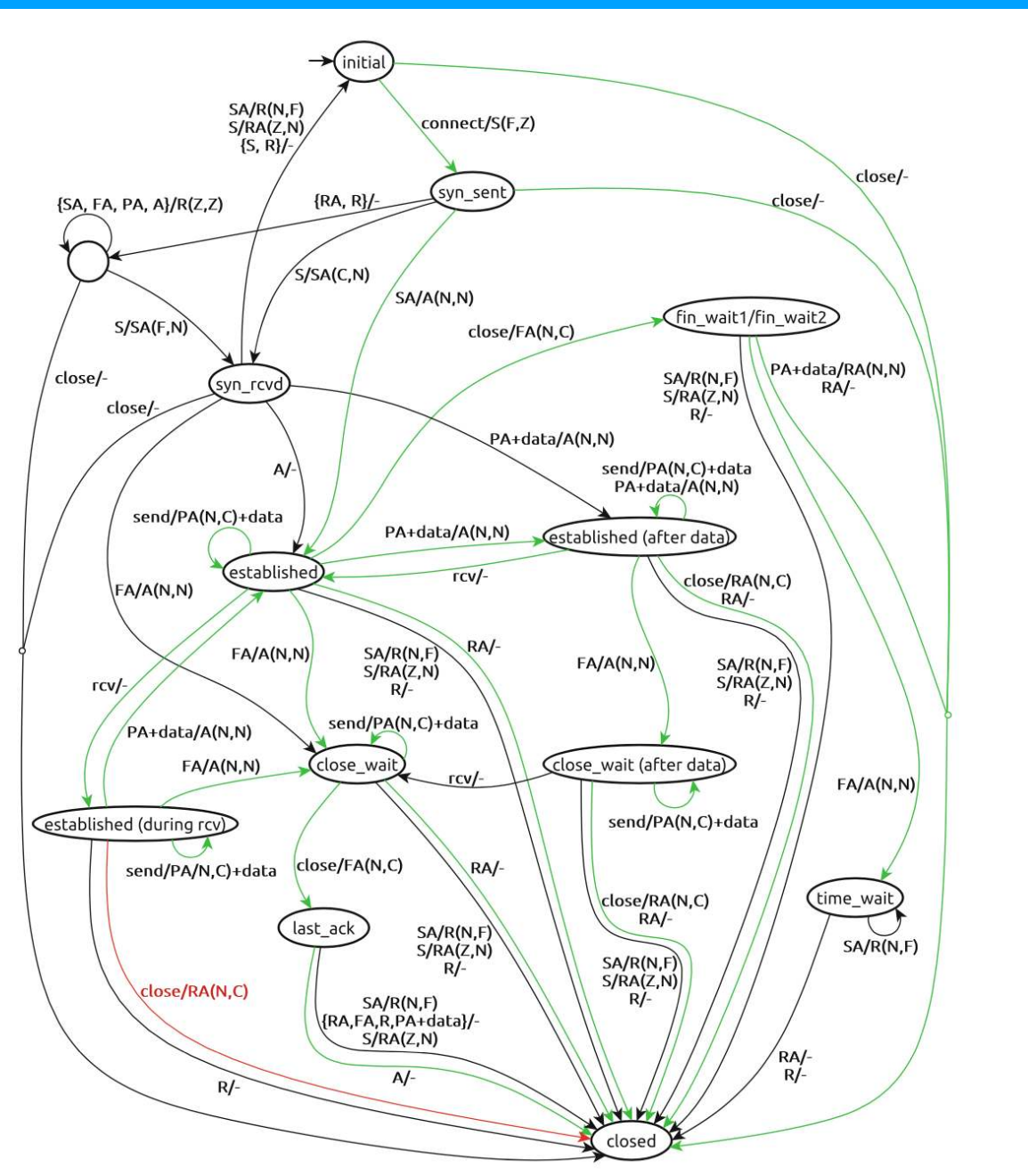


+ Counter-example
trace

Model Checking

Model of a system component / protocol

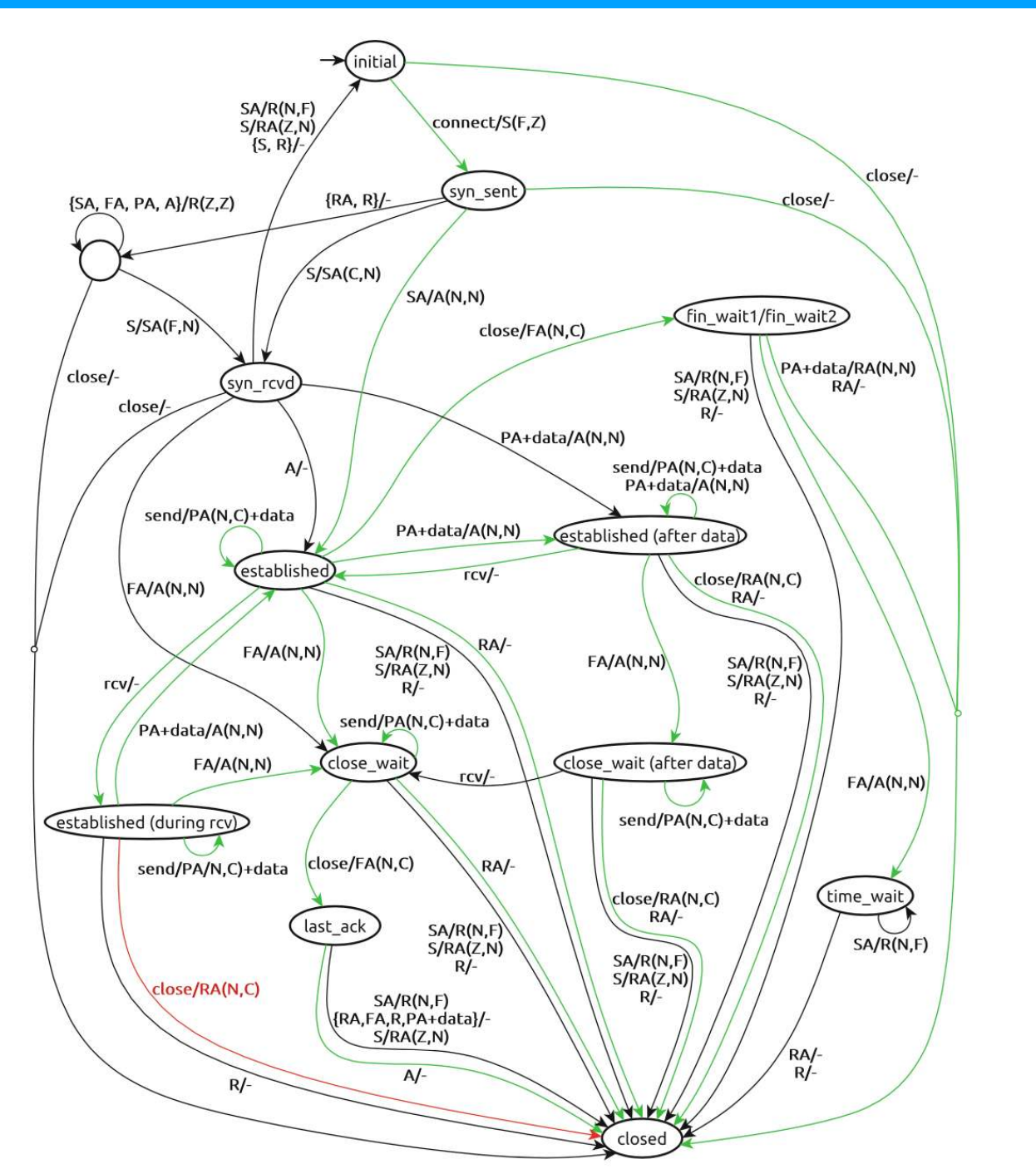
How do we design good models?
How do we maintain them?



+ Counter-example
trace

Model Learning

Model of a system component / protocol



build a model



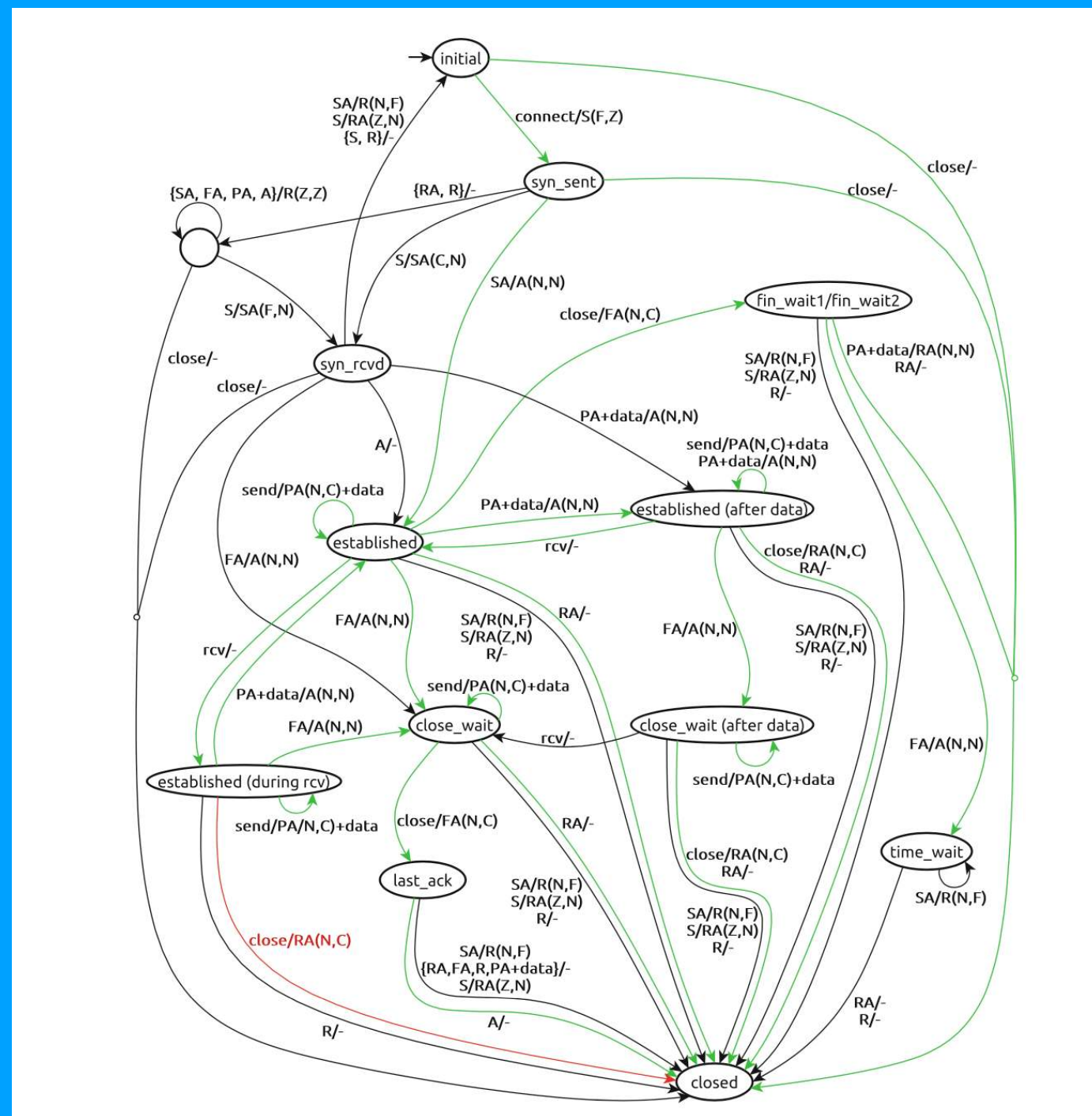
interact

Refine & update model

System
Implementation

Model Learning

Model of a system component / protocol



build a model



Properties of Ψ

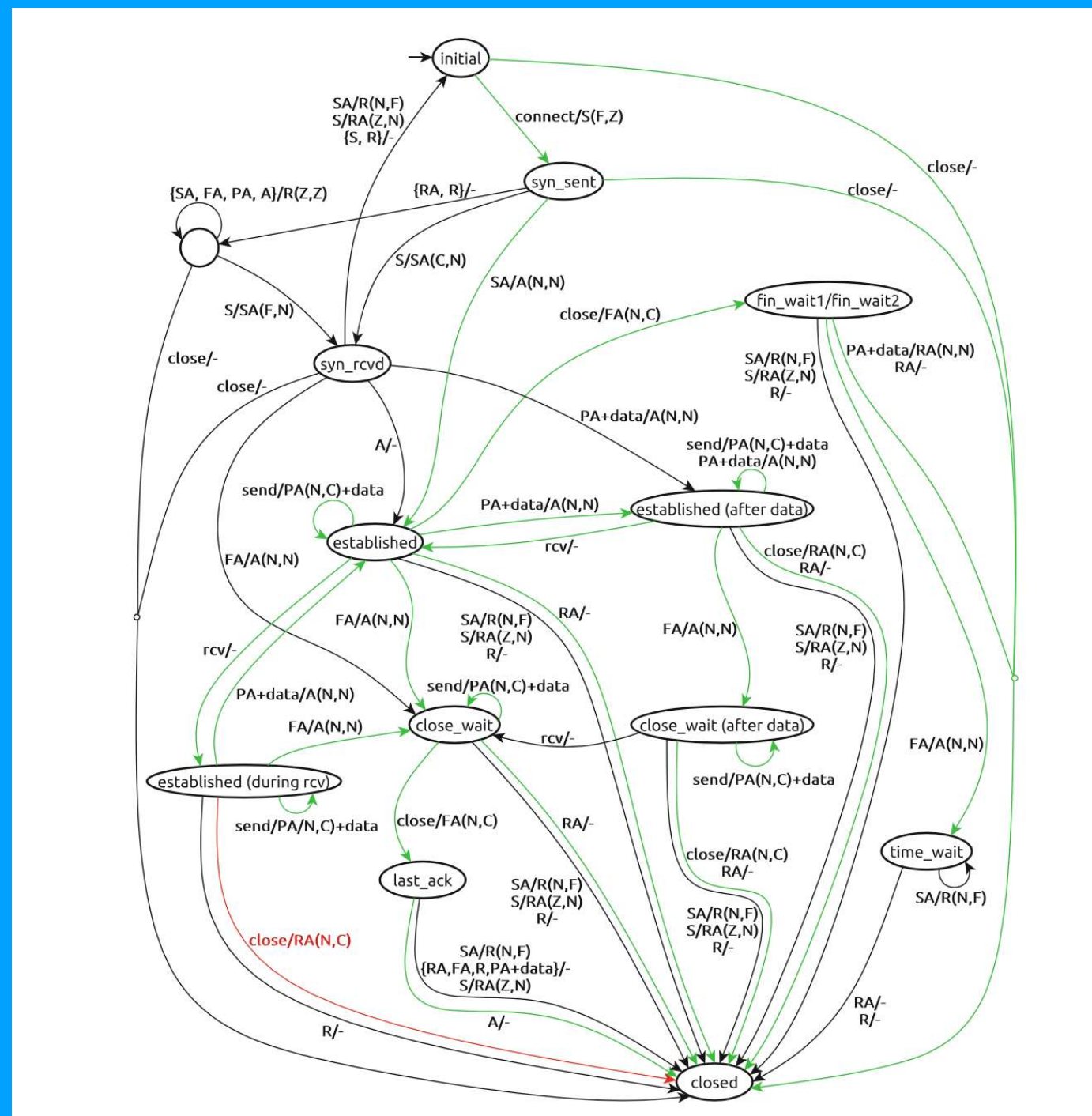
interact

Refine & update model

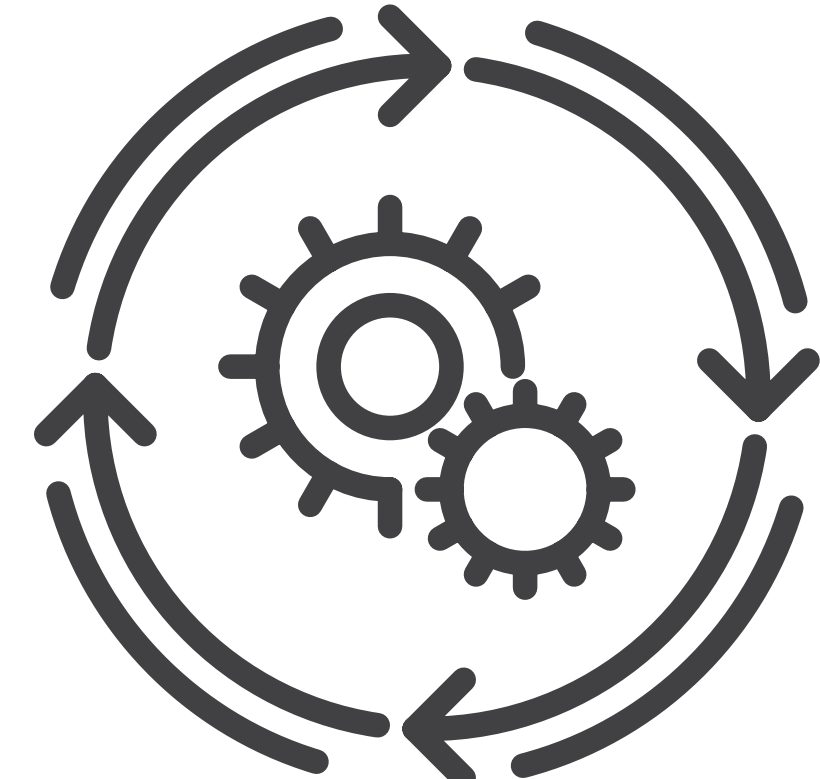
System Implementation

Model Learning

Model of a system component / protocol



build a model



Automated

Refine & update model

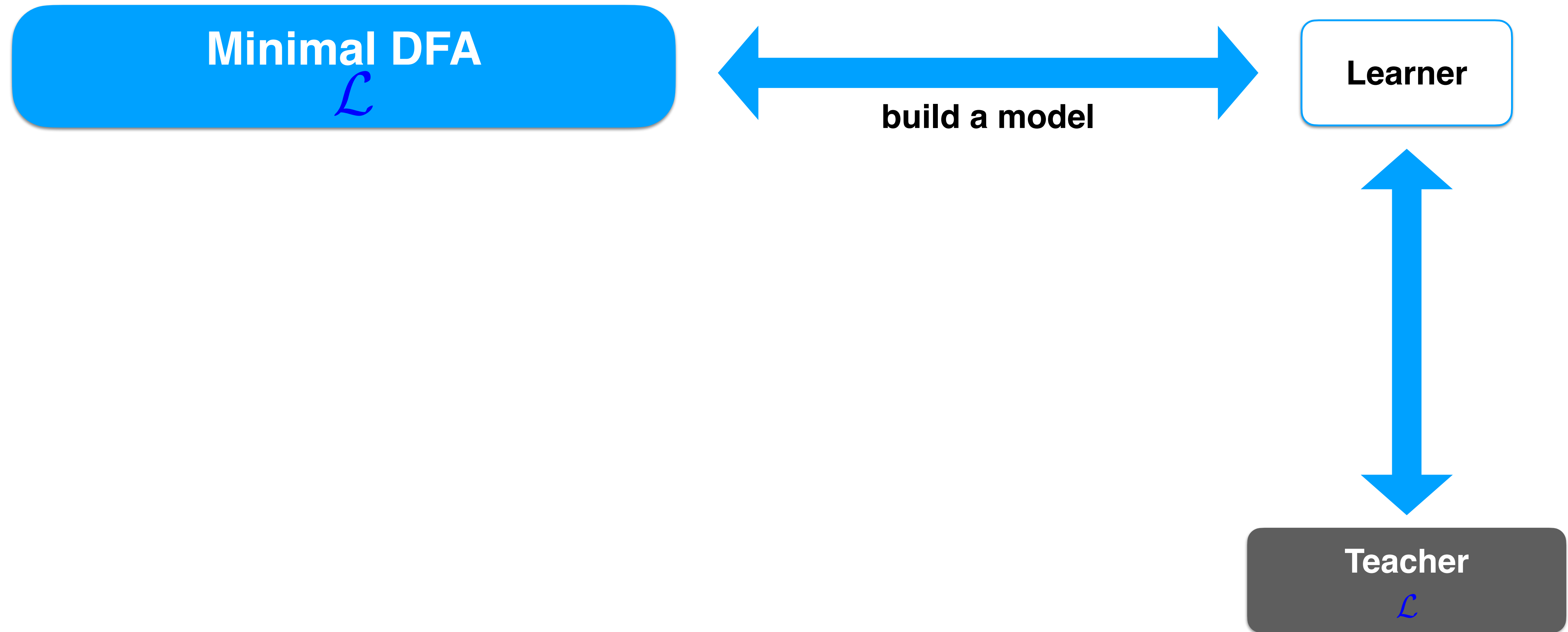


properties of

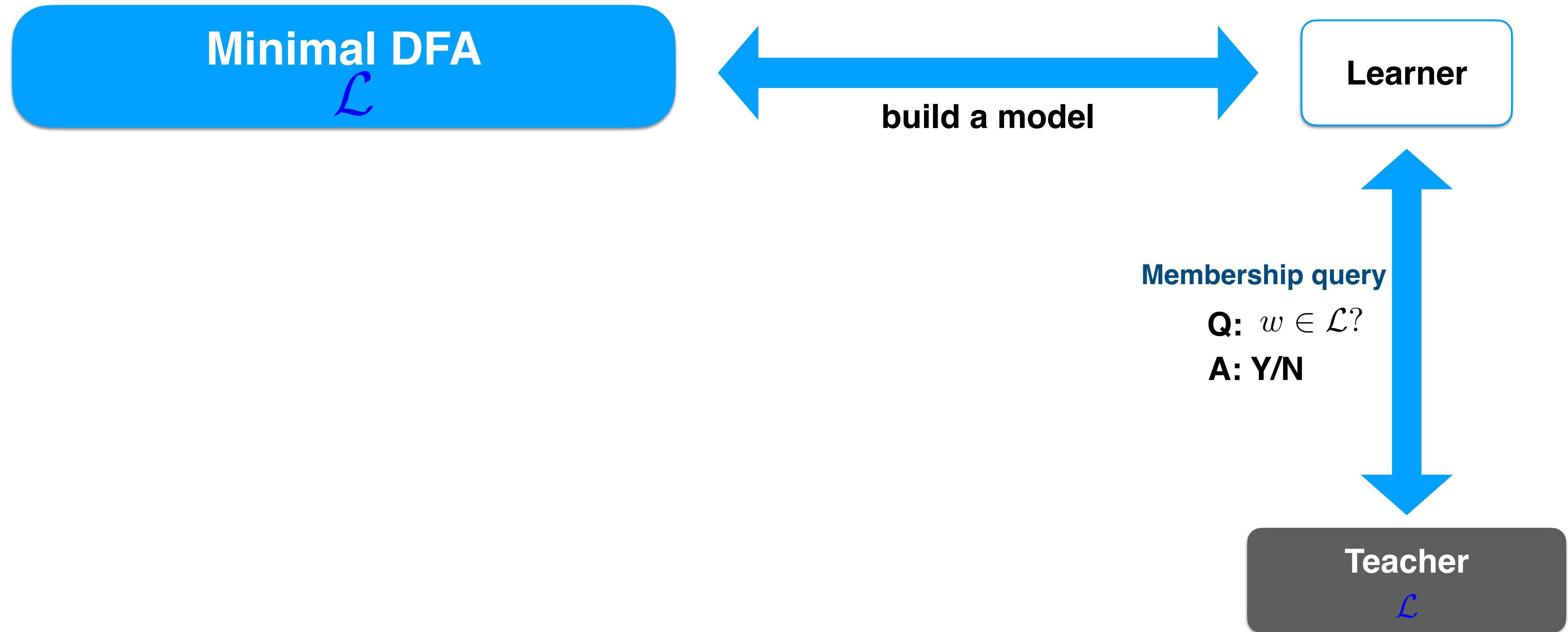
interact

System Implementation

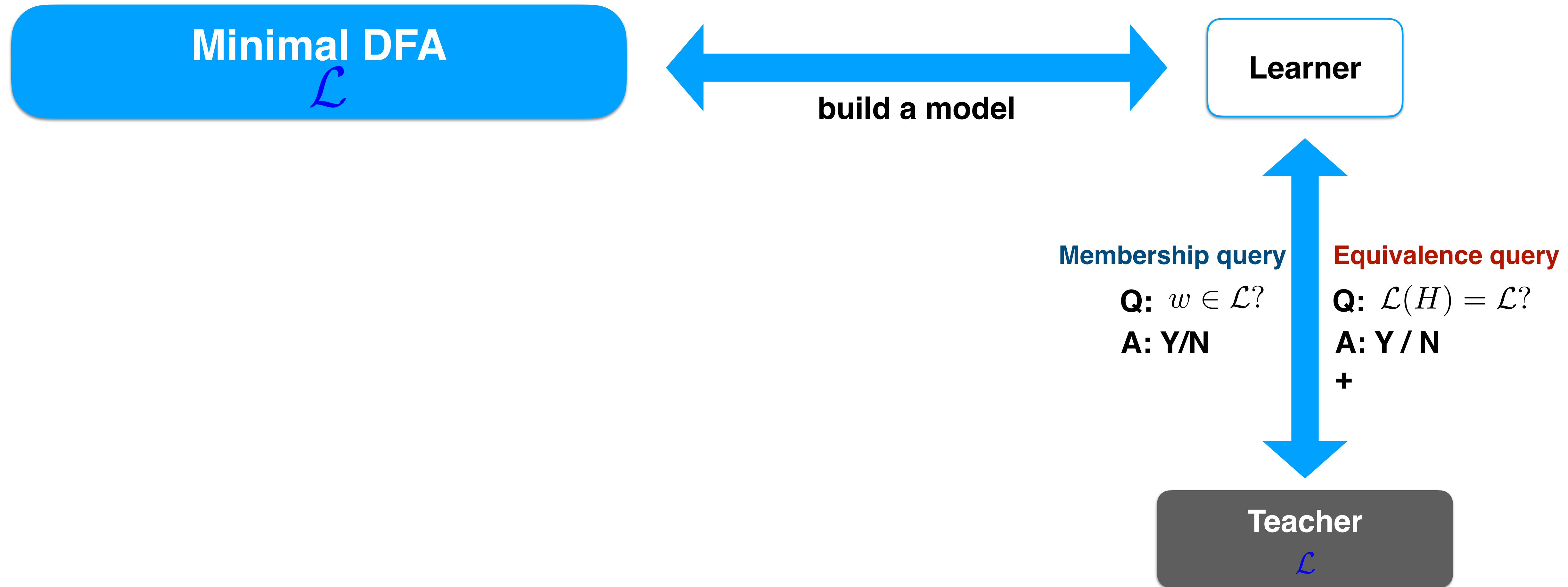
L^* - algorithm [Angluin'87]



L^* - algorithm [Angluin'87]



L^* - algorithm [Angluin'87]



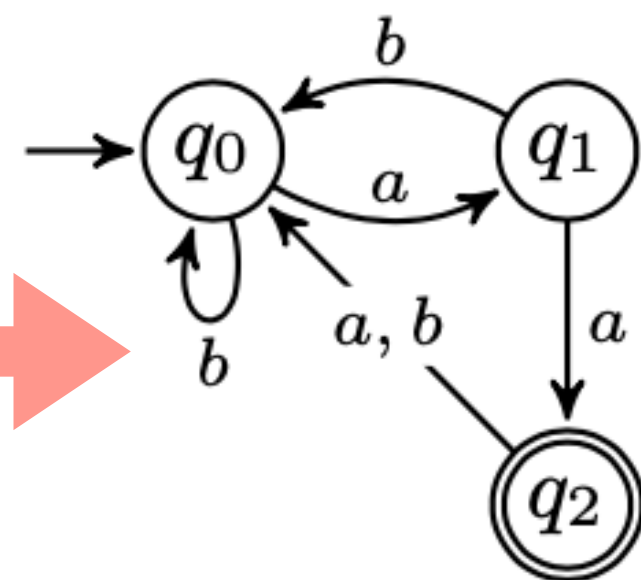
L^* - algorithm [Angluin'87]

Minimal DFA
 $\mathcal{L}$

build a model

Learner

	ϵ	a
ϵ	0	0
a	0	1
aa	1	0
b	0	0
ab	0	0
aaa	0	0
aab	0	0



Membership query

Q: $w \in \mathcal{L}$?
A: Y/N

Equivalence query

Q: $\mathcal{L}(H) = \mathcal{L}$?
A: Y / N
+

Teacher

Observation table

L^* - algorithm [Angluin'87]

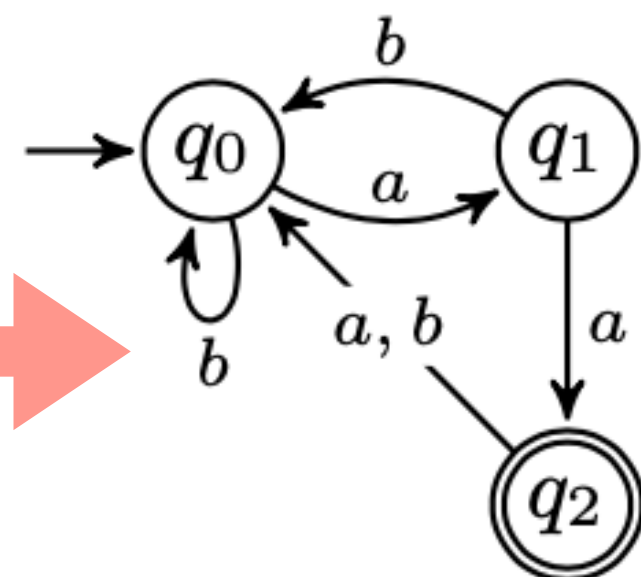
Minimal DFA

$\mathcal{L}$

Learner

build a model

	ϵ	a
ϵ	0	0
a	0	1
aa	1	0
b	0	0
ab	0	0
aaa	0	0
aab	0	0



Membership query

Q: $w \in \mathcal{L}$?

A: Y/N

Equivalence query

Q: $\mathcal{L}(H) = \mathcal{L}$?

A: Y / N

+

Teacher

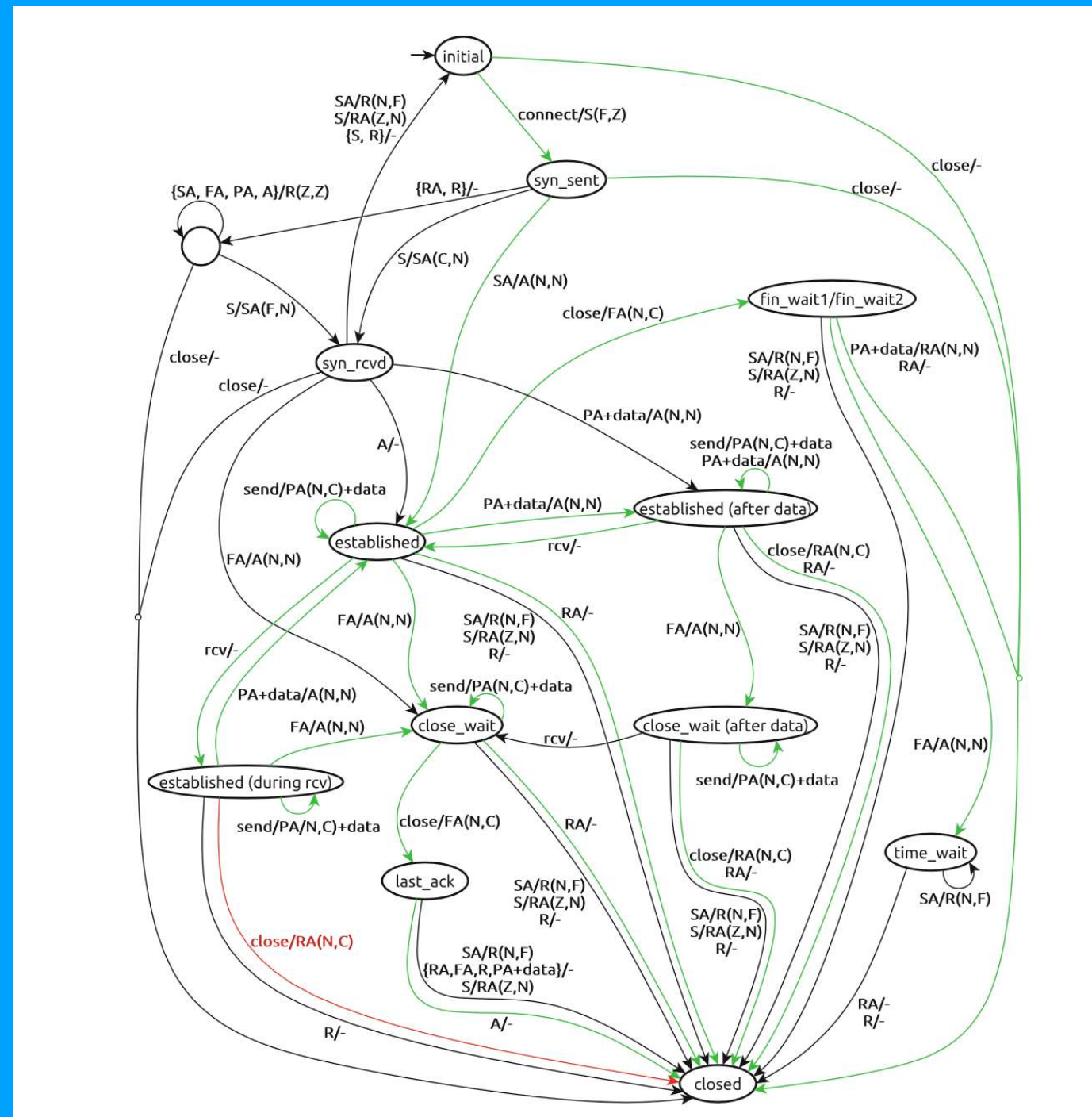
$\mathcal{L}$

Observation table

L^* at work

L* at work

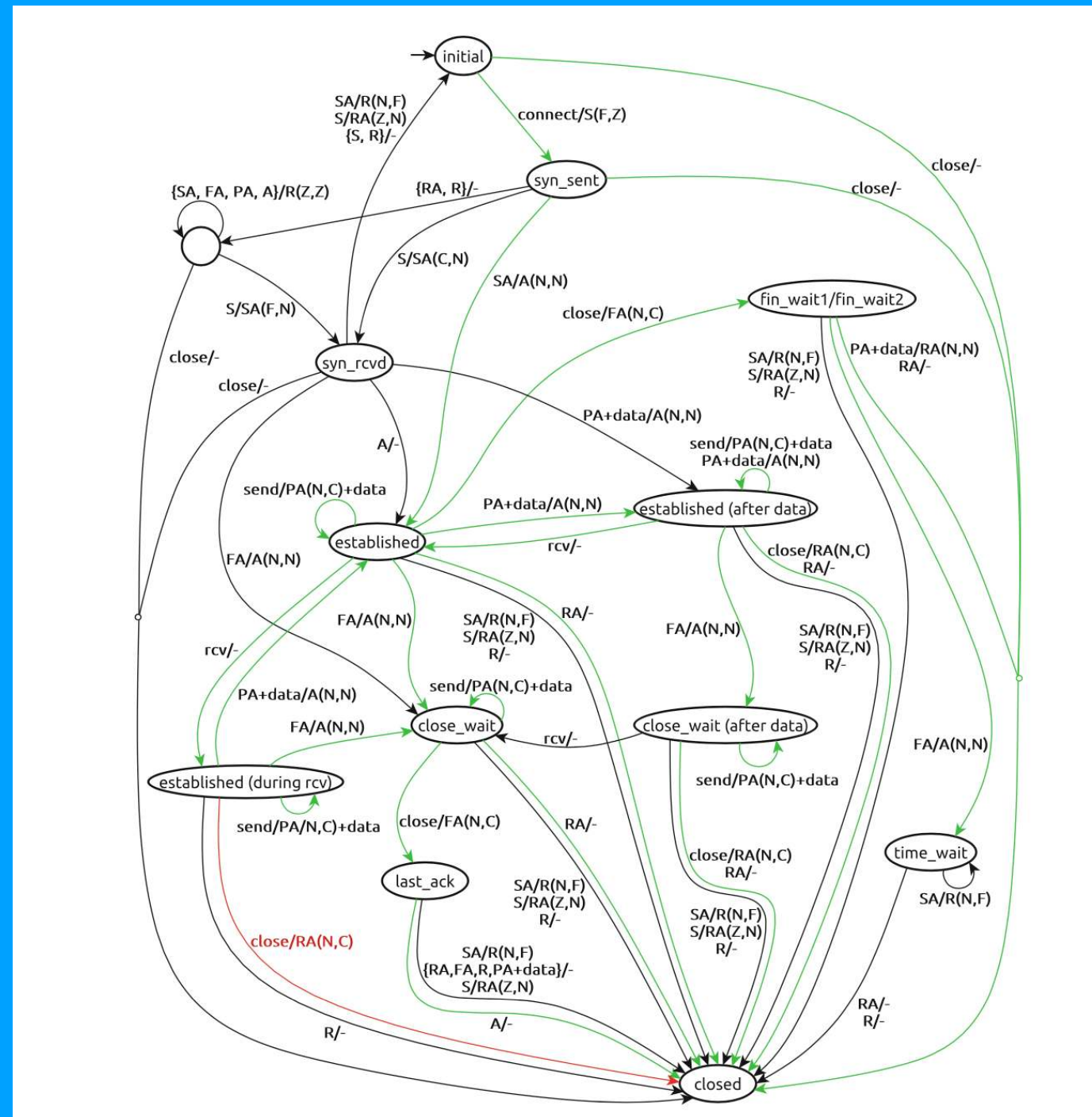
Model of Windows 8 TCP implementation



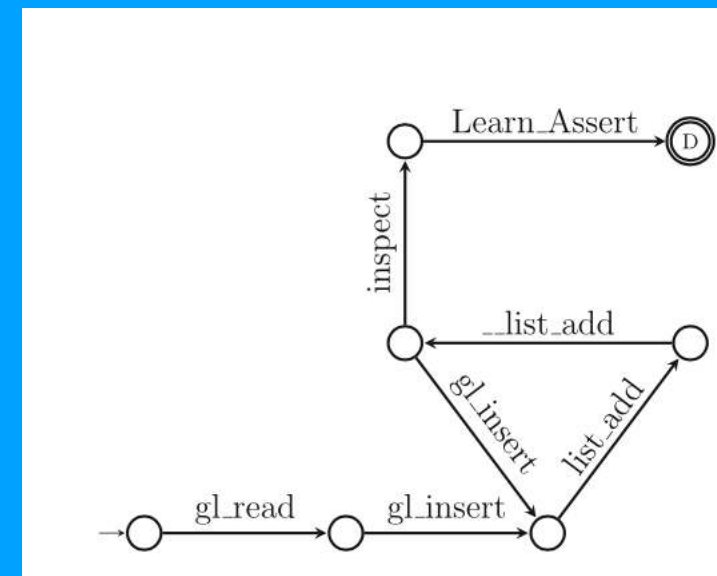
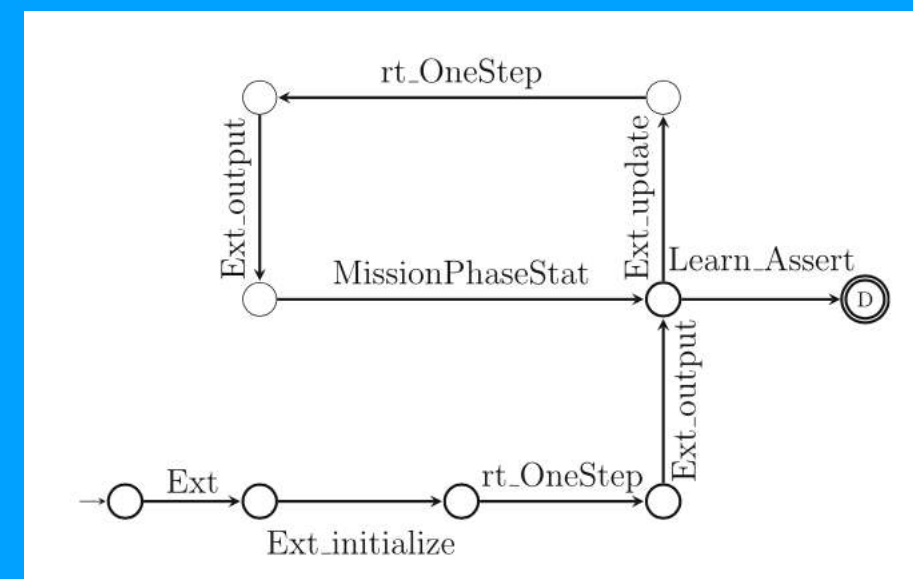
Combining Model Learning and Model Checking to Analyze TCP Implementations

L* at work

Model of Windows 8 TCP implementation



Learning Error Traces (function calls)



Combining Model Learning and Model Checking to Analyze TCP Implementations

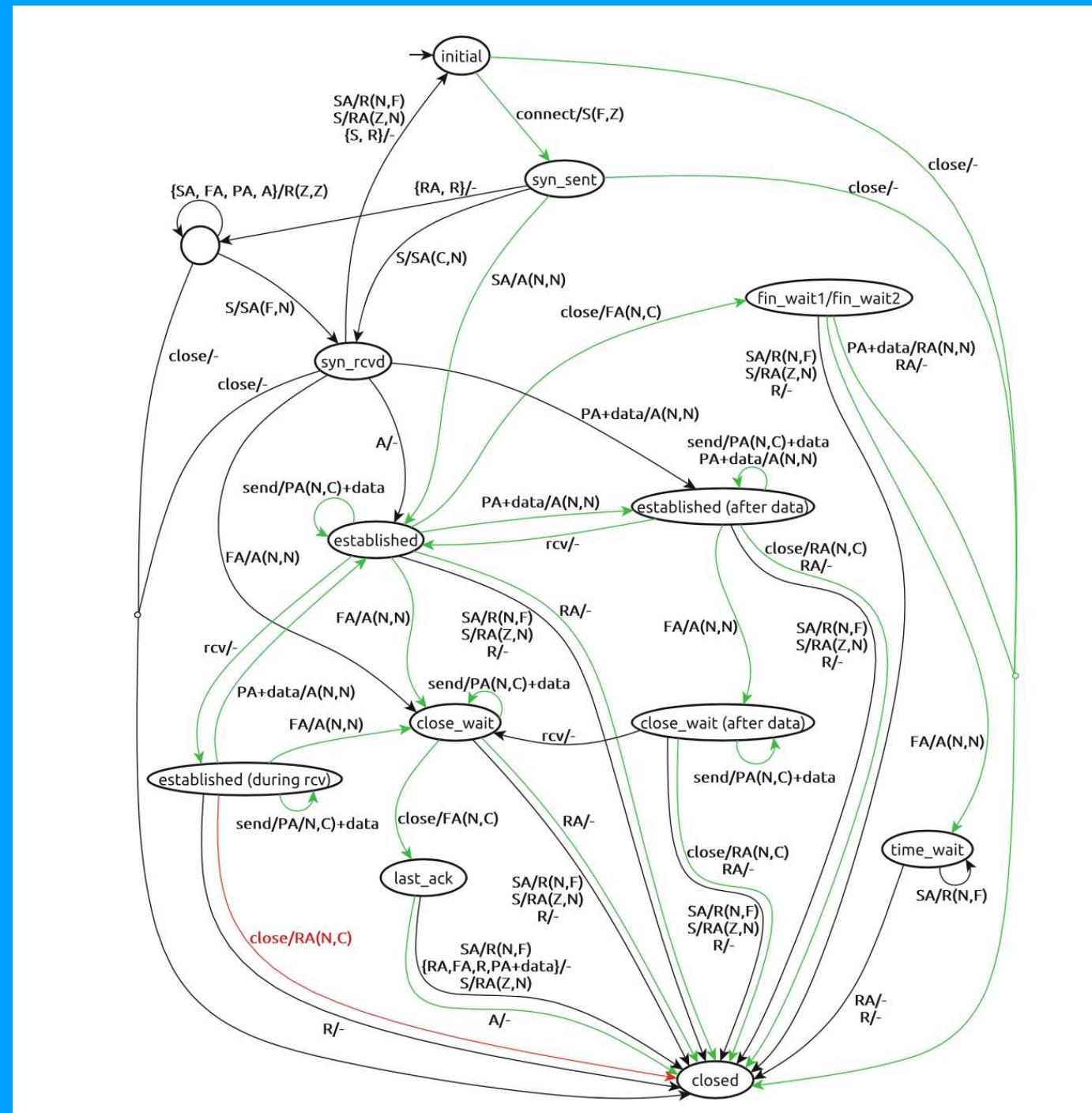
Paul Fiterău-Broștean, Ramon Janssen, and Frits Vaandrager^(✉)

Learning the Language of Error

Martin Chapman¹, Hana Chockler¹(✉), Pascal Kesseli², Daniel Kroening²,
Ofer Strichman³, and Michael Tautschnig⁴

L* at work

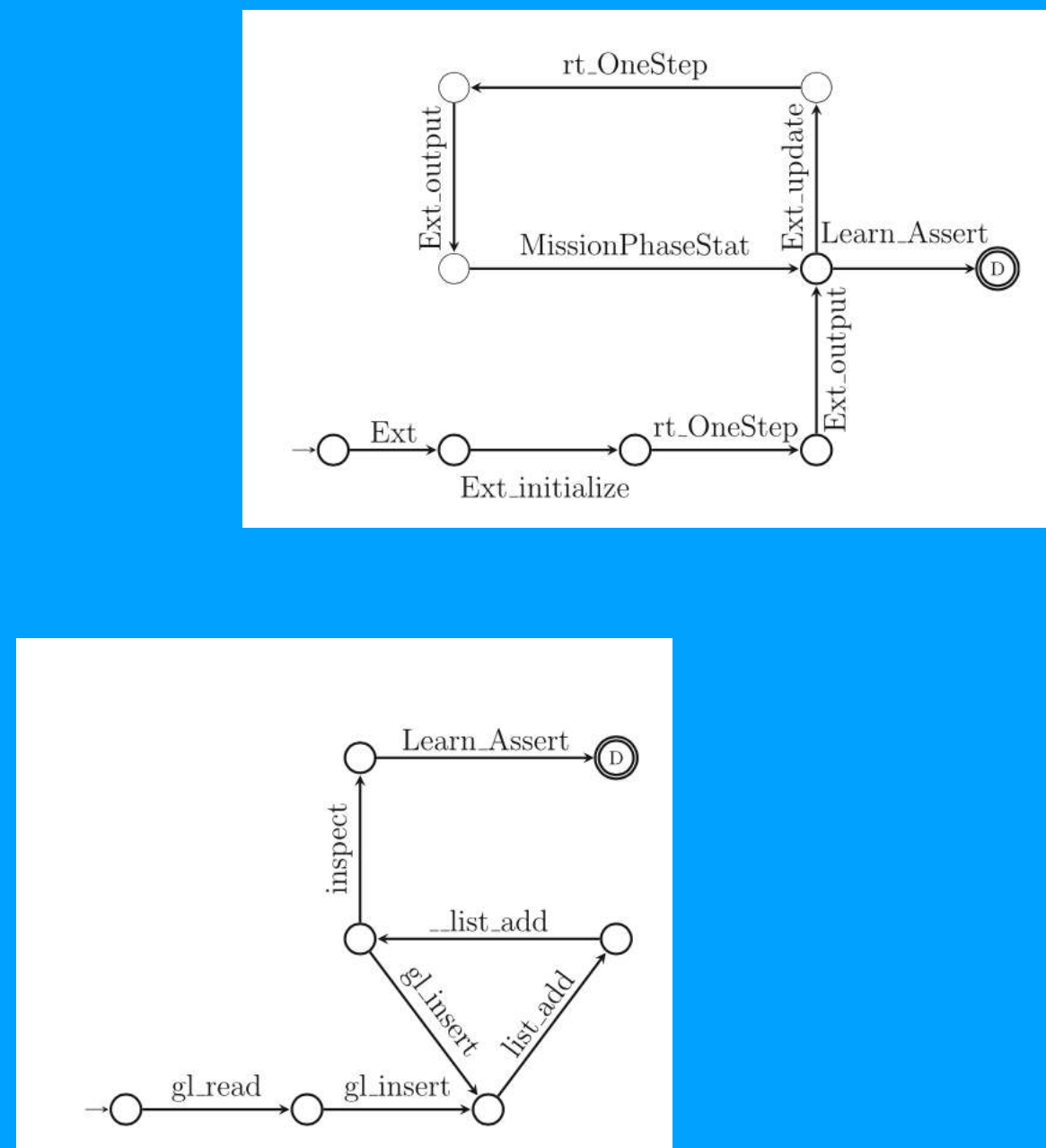
Model of Windows 8 TCP implementation



Combining Model Learning and Model Checking to Analyze TCP Implementations

Paul Fiterău-Broștean, Ramon Janssen, and Frits Vaandrager^(✉)

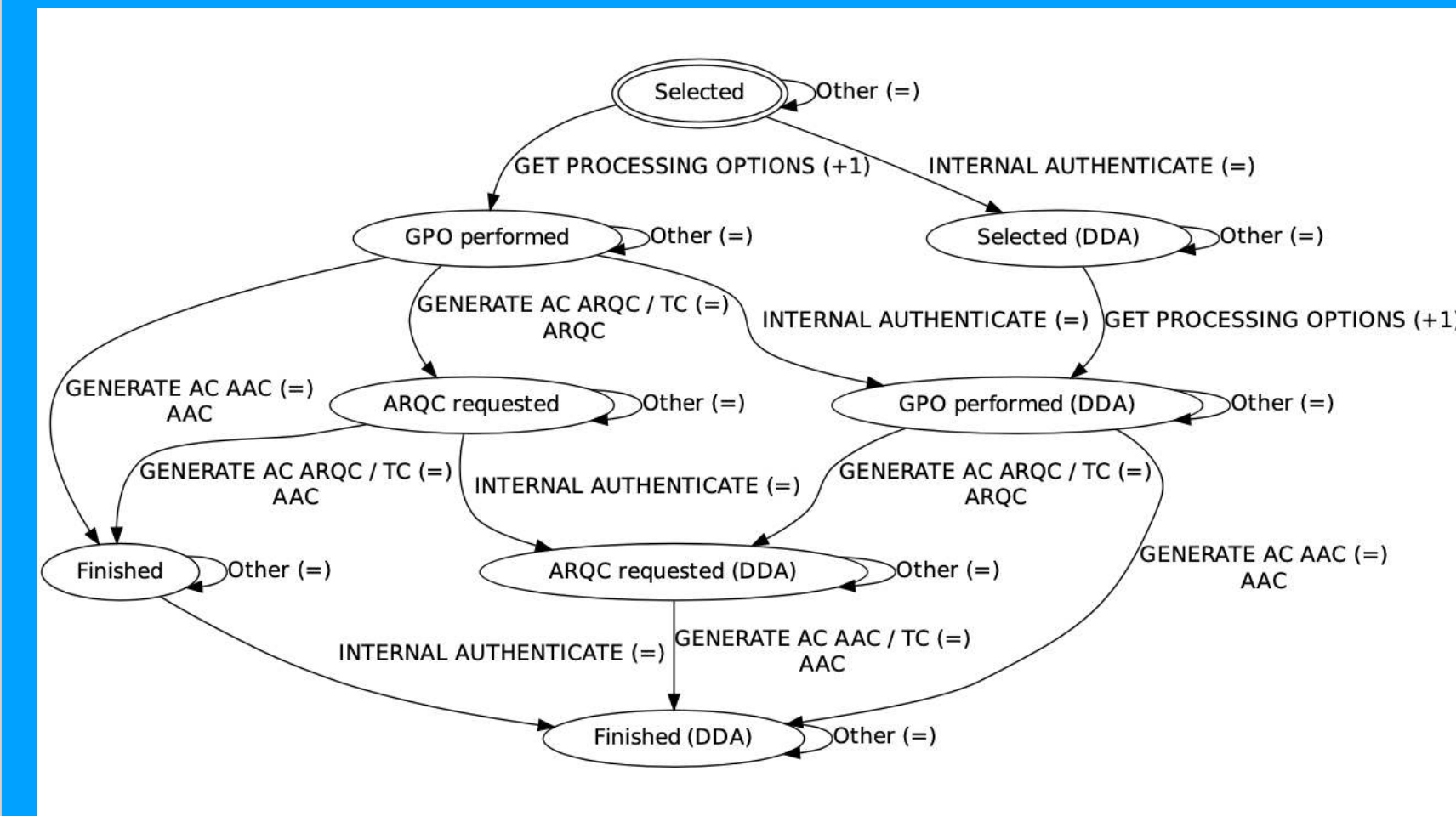
Learning Error Traces (function calls)



Learning the Language of Error

Martin Chapman¹, Hana Chockler^{1(✉)}, Pascal Kesseli², Daniel Kroening², Ofer Strichman³, and Michael Tautschnig⁴

Model of Visa Debit on Barclays card



Formal models of bank cards for free

Fides Aarts, Joeri de Ruiter, and Erik Poll

L* at work

Model of Windows 8 TCP implementation

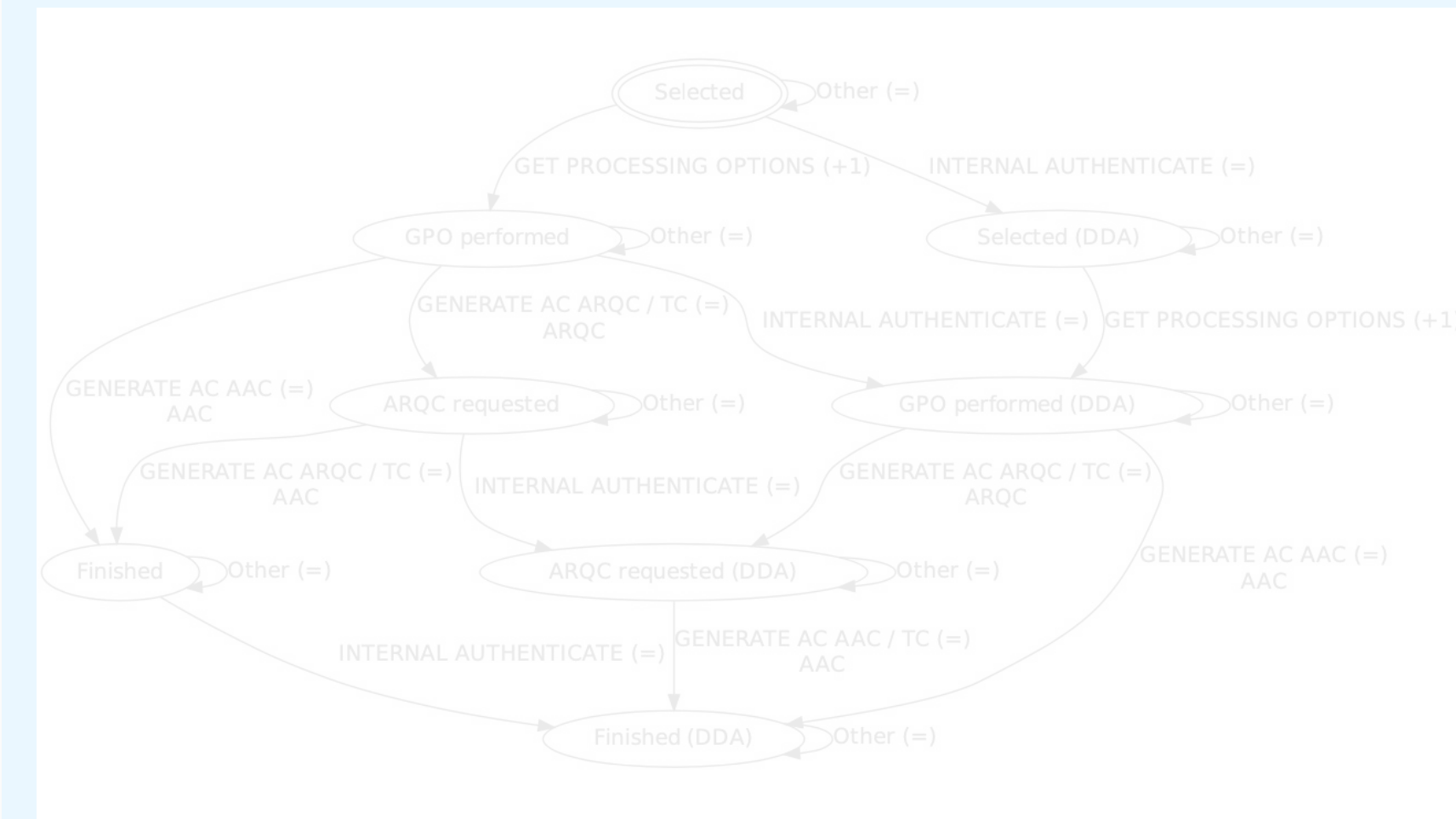
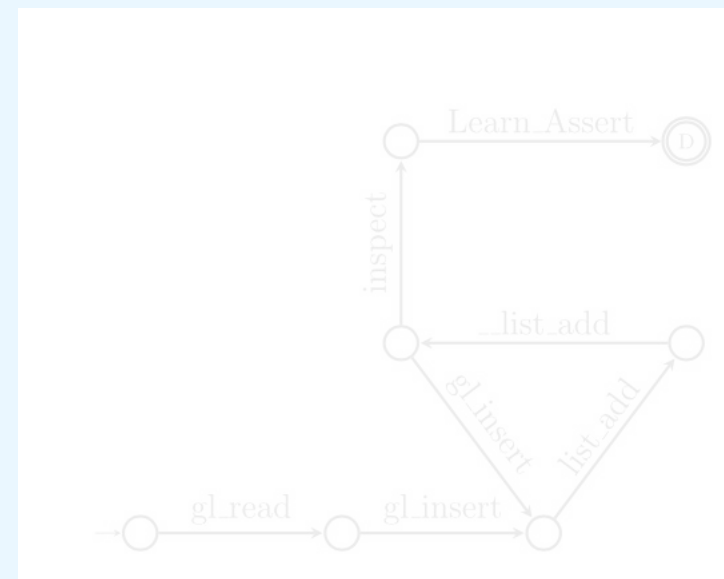
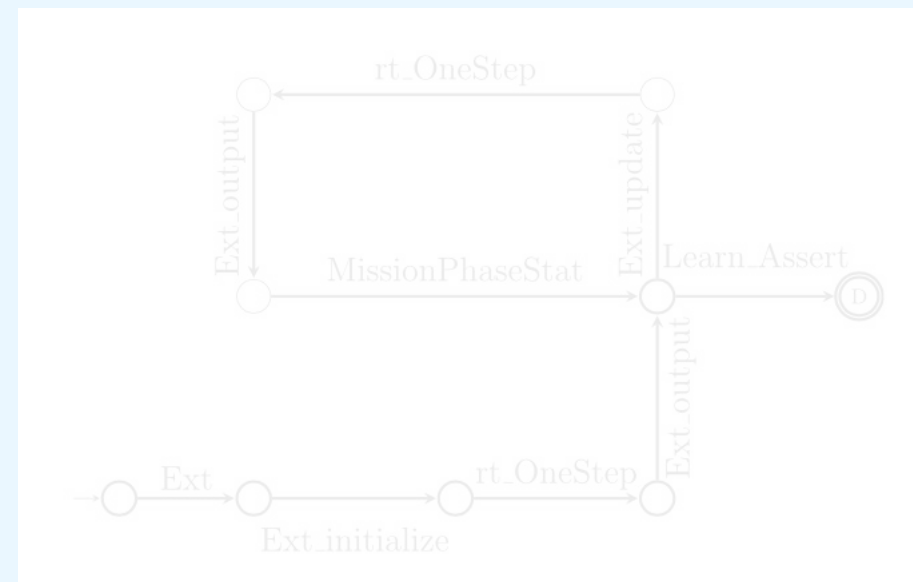
Learning Error Traces (function calls)

Model of Visa Debit on Barclays card

Other models?

Data-sensitive Quantitative Concurrency

...



Combining Model Learning and Model Checking
to Analyze TCP Implementations

Learning the Language of Error

Formal models of bank cards for free

Martin Chapman¹, Hana Chockler¹ (✉), Pascal Kesseli², Daniel Kroening²,
Ofer Strichman³, and Michael Tautschnig⁴

Paul Fiterău-Broștean, Ramon Janssen, and Frits Vaandrager (✉)

Fides Aarts, Joeri de Ruiter, and Erik Poll

L* at work

Model of Windows 8 TCP implementation

Other models?

Data-sensitive Quantitative Concurrency

Learning Error Traces (function calls)

Algorithm Design?

Model of Visa Debit on Barclays card

Correctness Termination Optimizations

Combining Model Learning and Model Checking
to Analyze TCP Implementations

Paul Fiterău-Broștean, Ramon Janssen, and Frits Vaandrager^(✉)

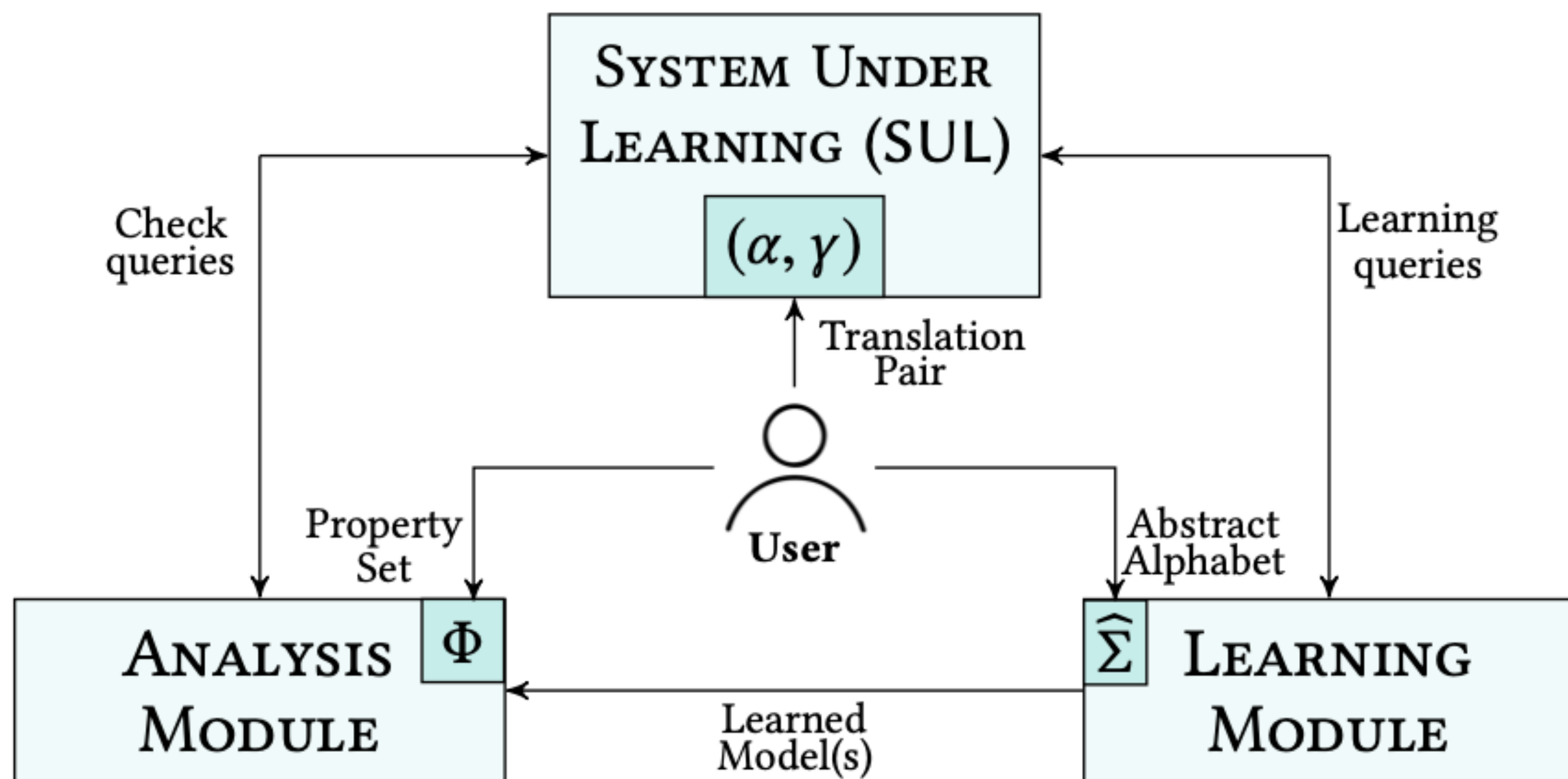
Learning the Language of Error

Martin Chapman¹, Hana Chockler^{1(✉)}, Pascal Kesseli², Daniel Kroening²,
Ofer Strichman³, and Michael Tautschnig⁴

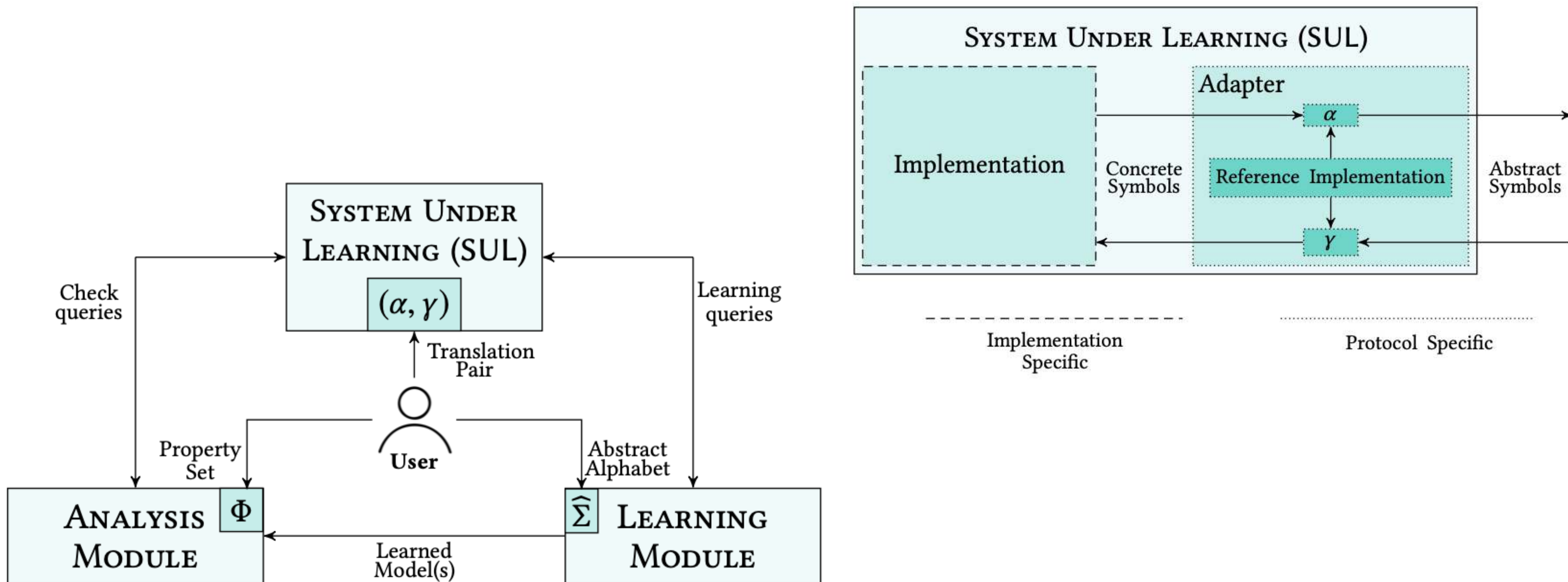
Formal models of bank cards for free

Fides Aarts, Joeri de Ruiter, and Erik Poll

Prognosis

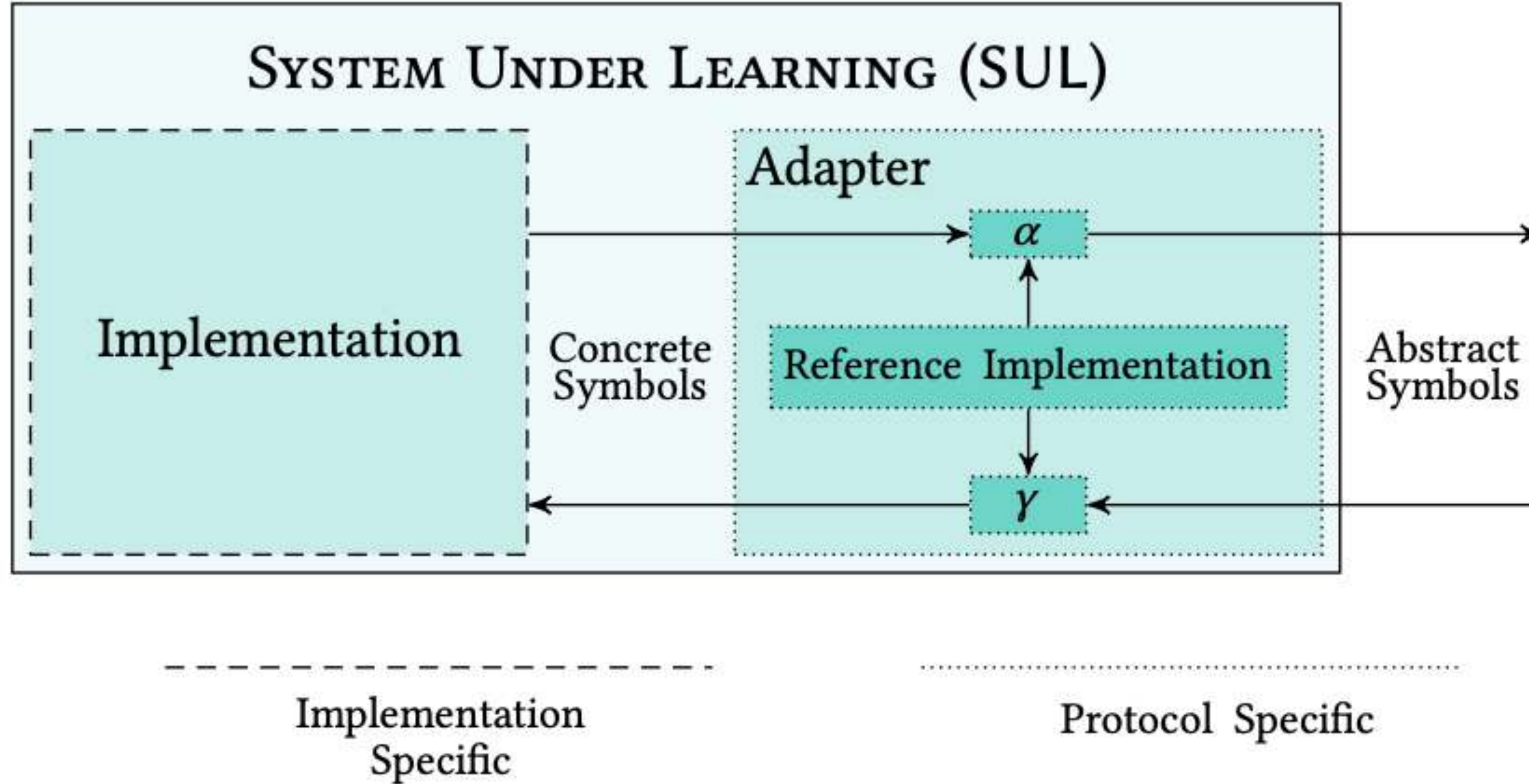


Prognosis

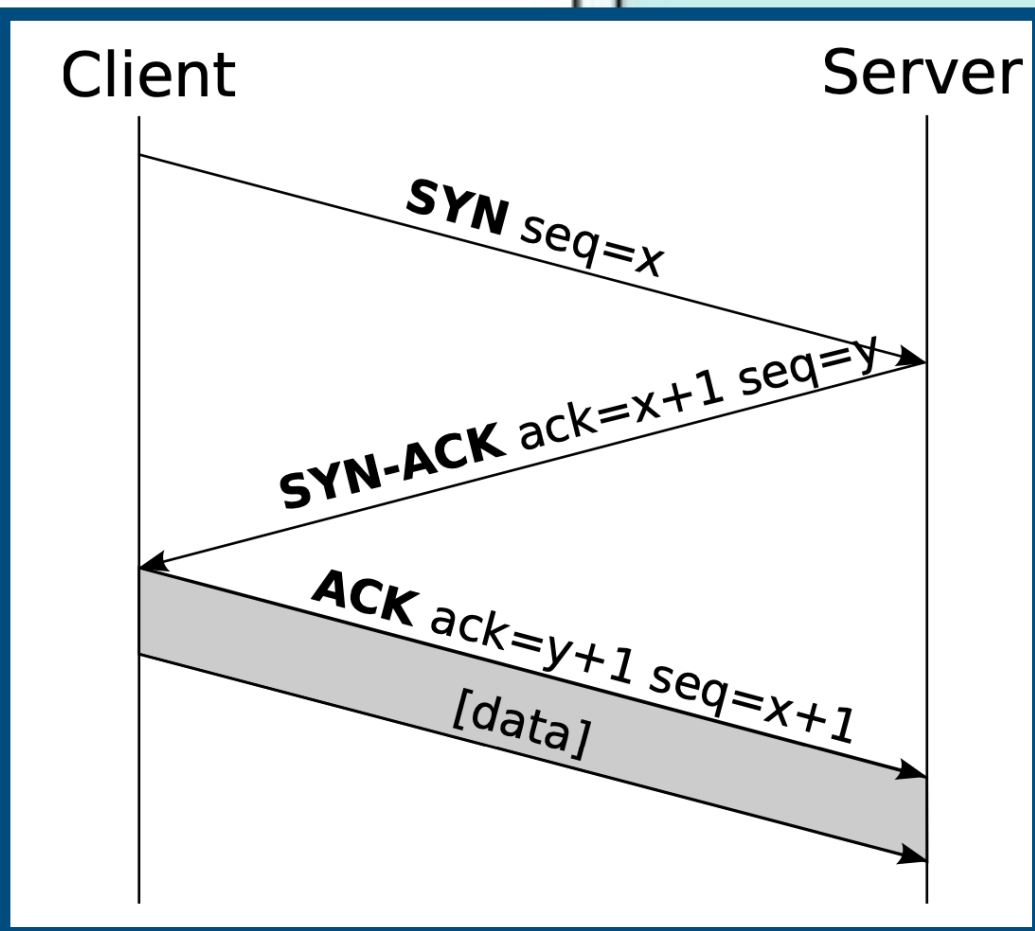
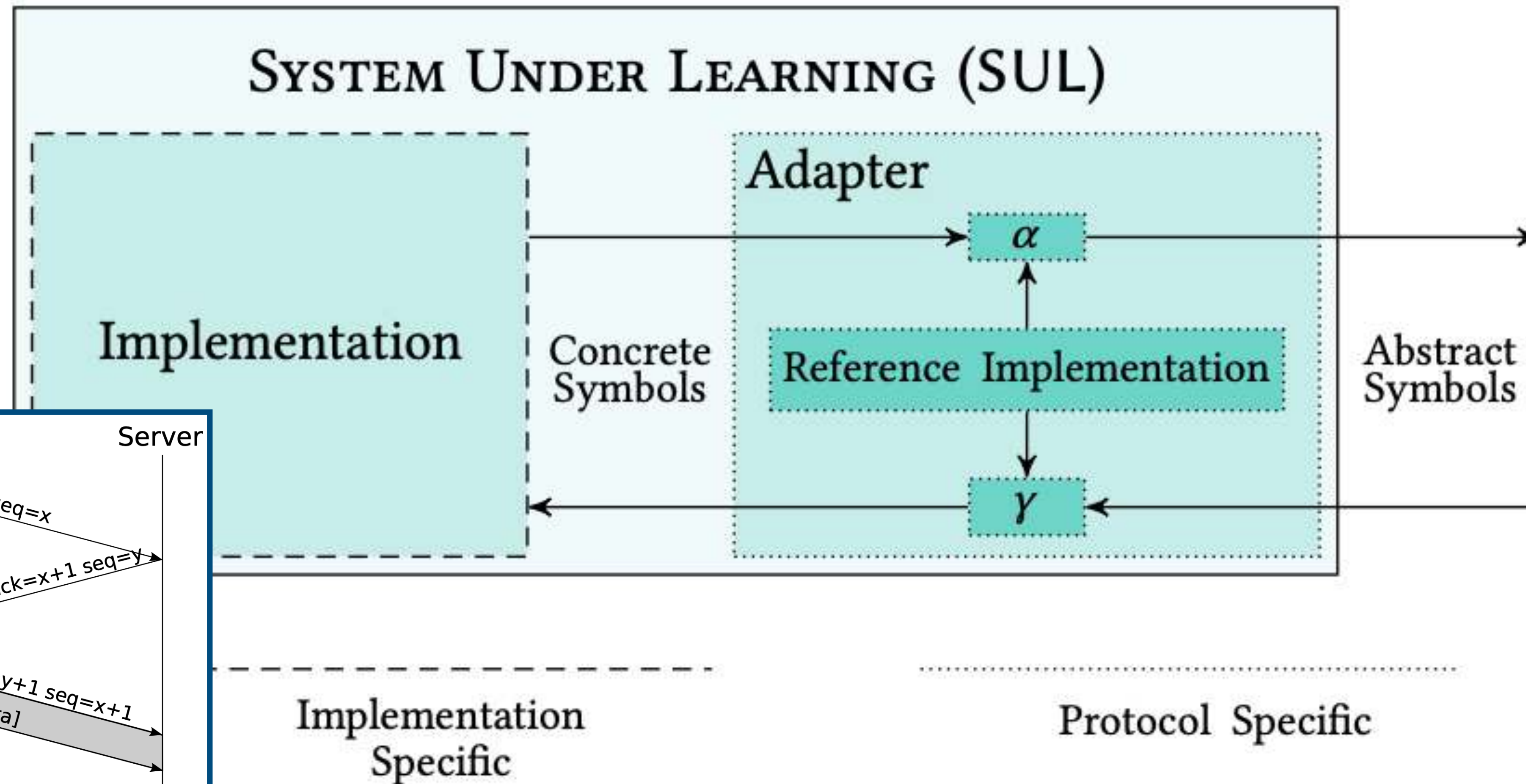


Prognosis

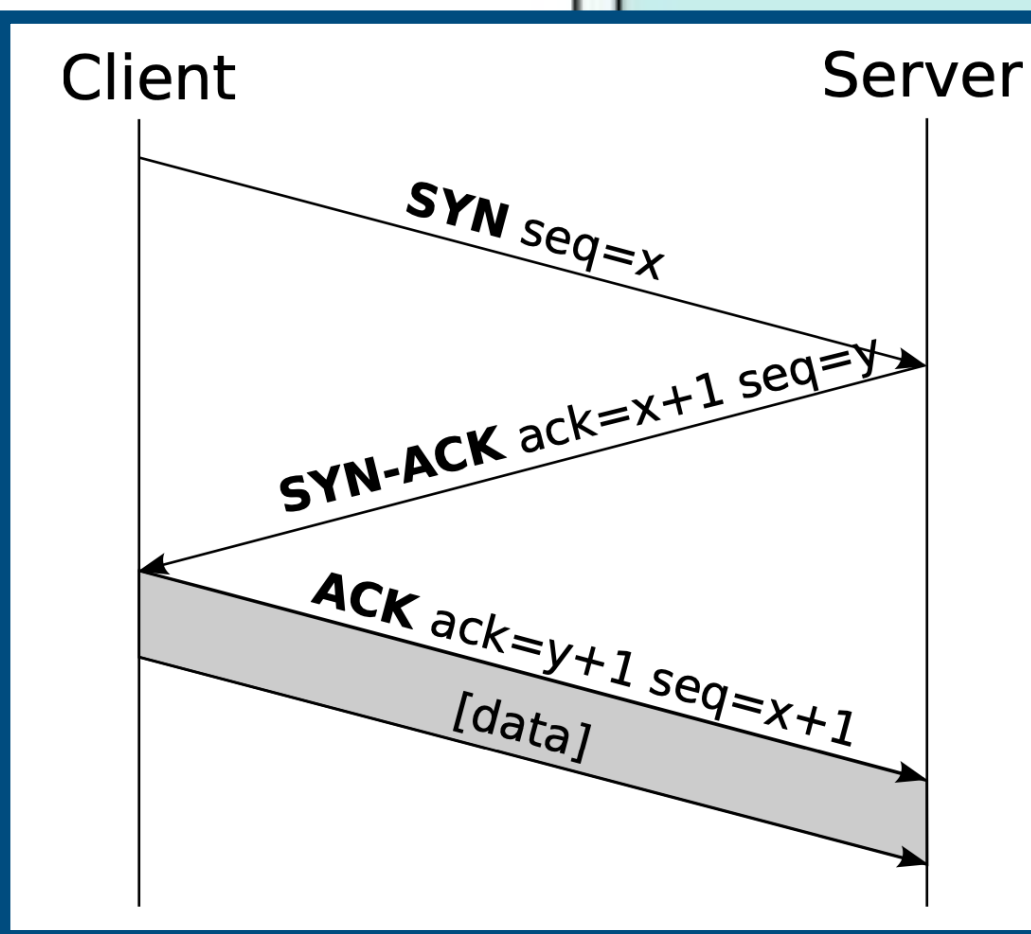
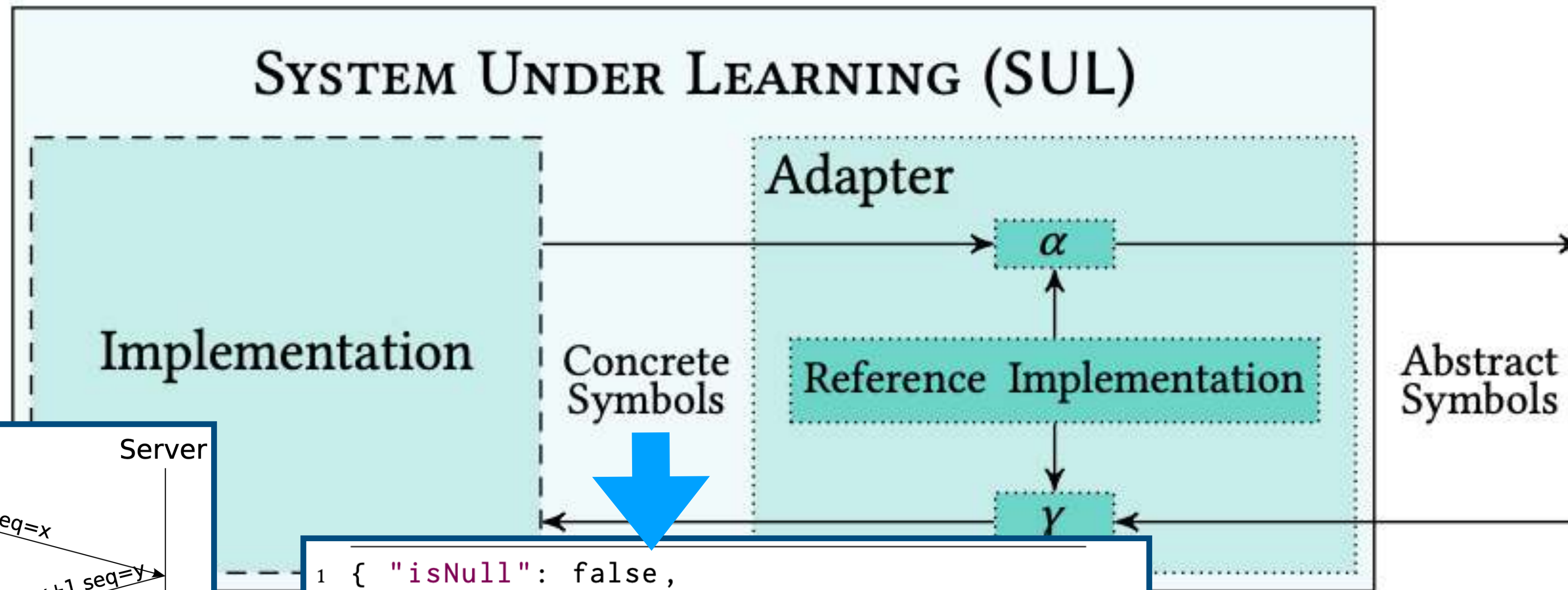
Prognosis



Prognosis



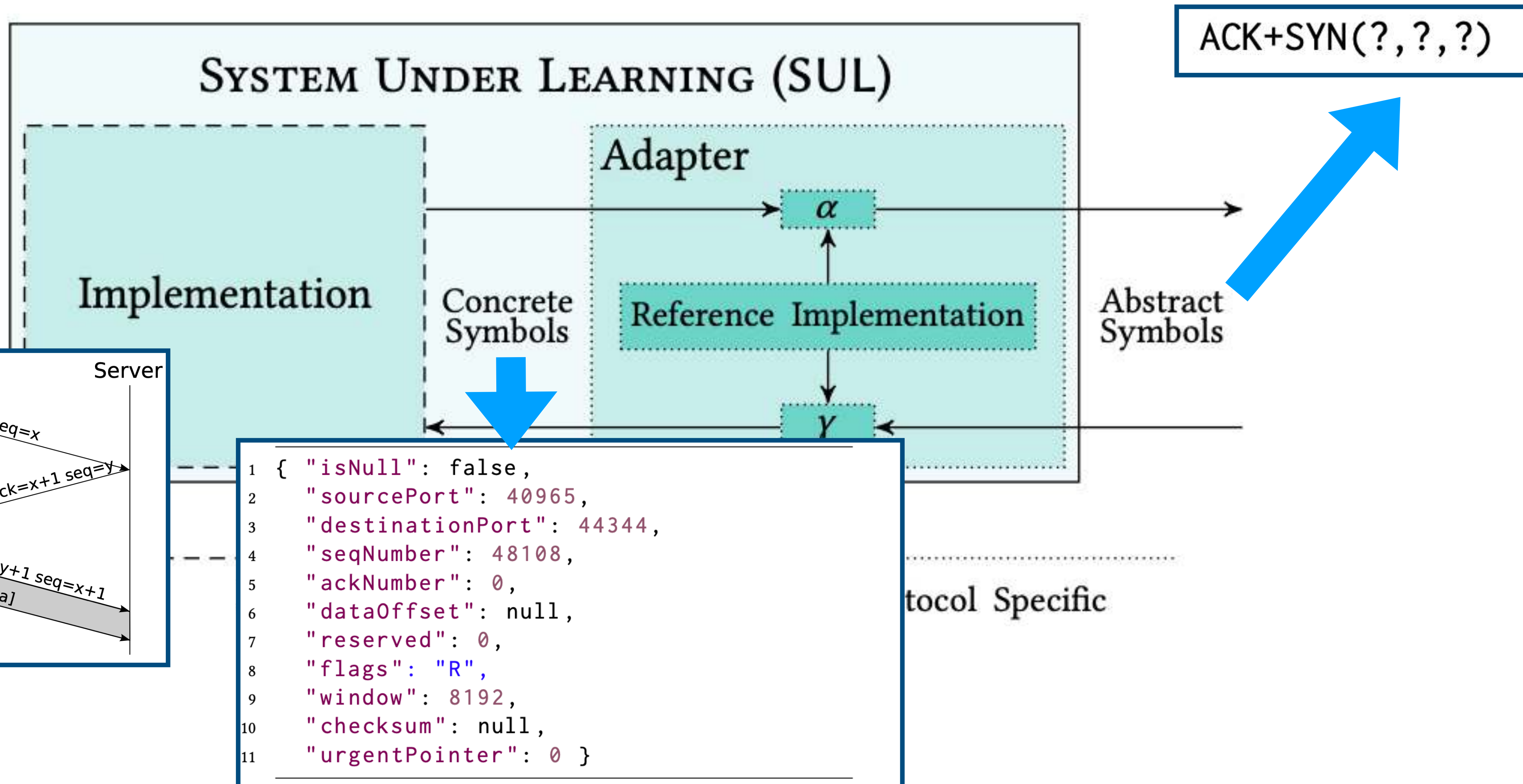
Prognosis



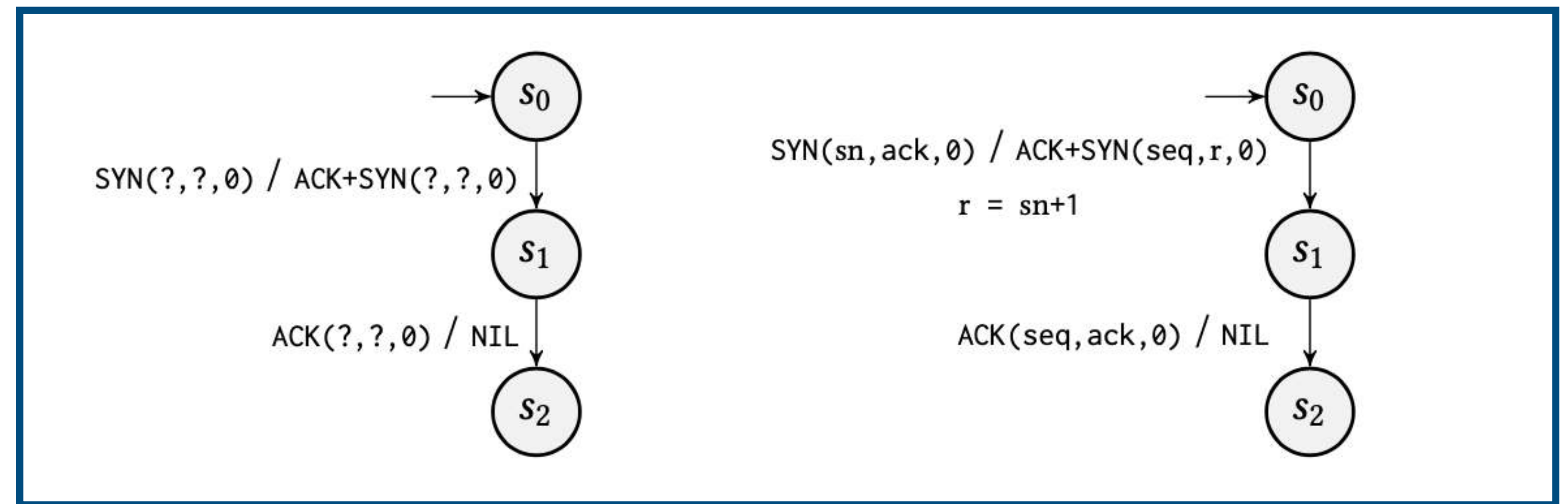
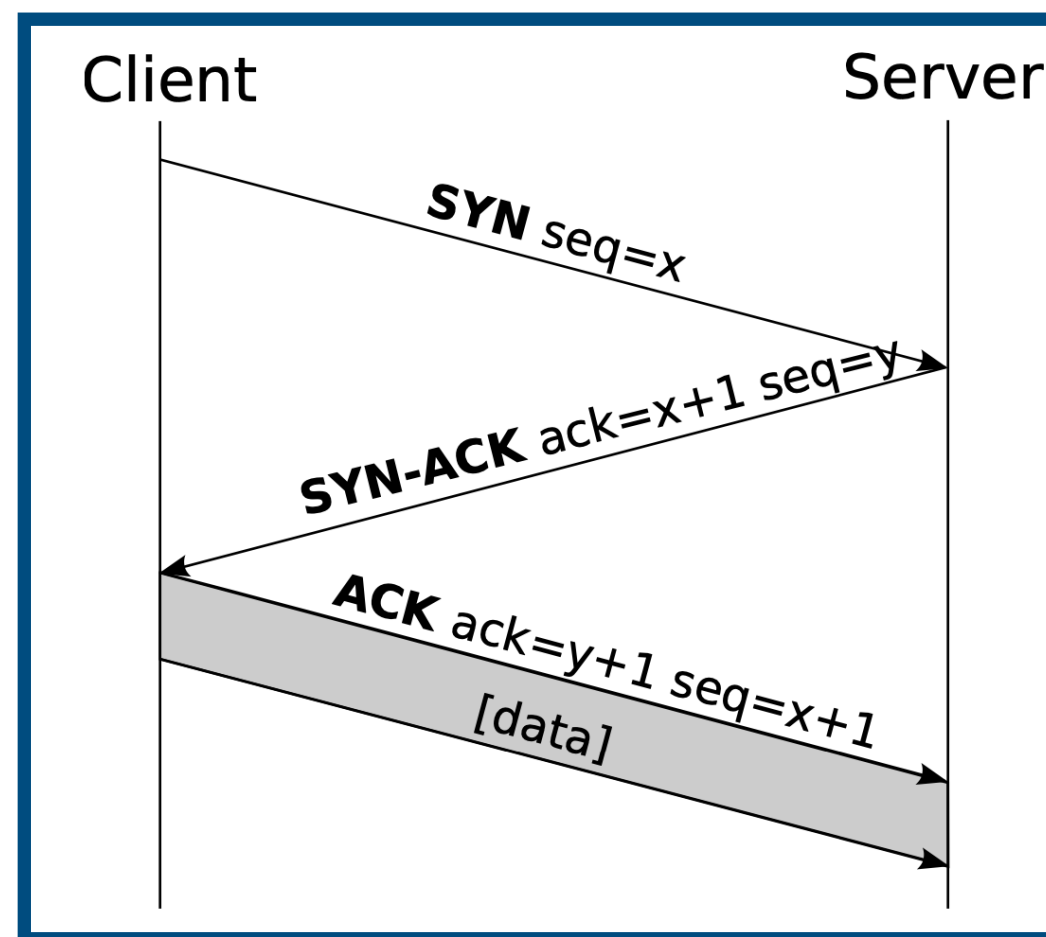
```
1 { "isNull": false,  
2   "sourcePort": 40965,  
3   "destinationPort": 44344,  
4   "seqNumber": 48108,  
5   "ackNumber": 0,  
6   "dataOffset": null,  
7   "reserved": 0,  
8   "flags": "R",  
9   "window": 8192,  
10  "checksum": null,  
11  "urgentPointer": 0 }
```

Protocol Specific

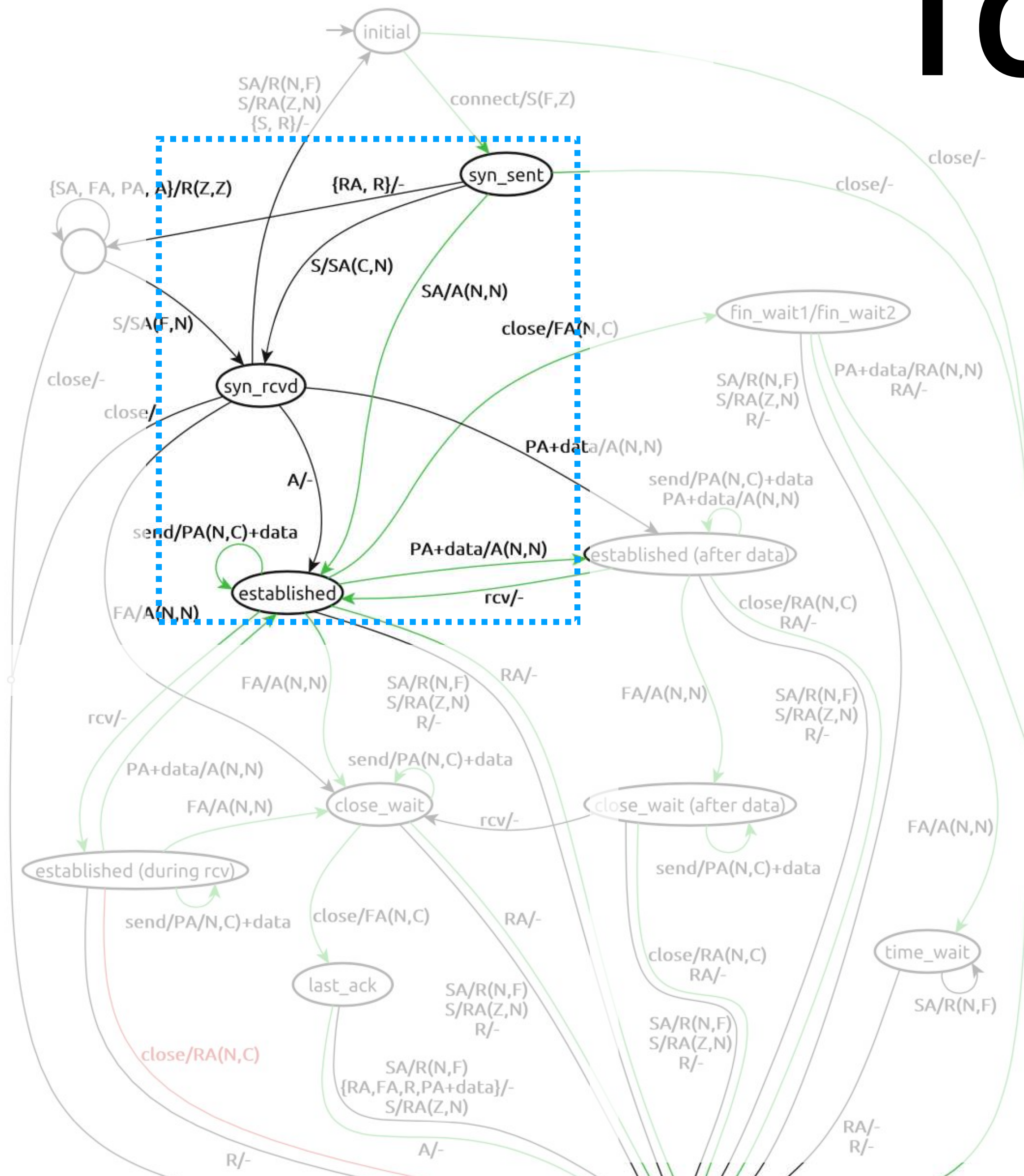
Prognosis



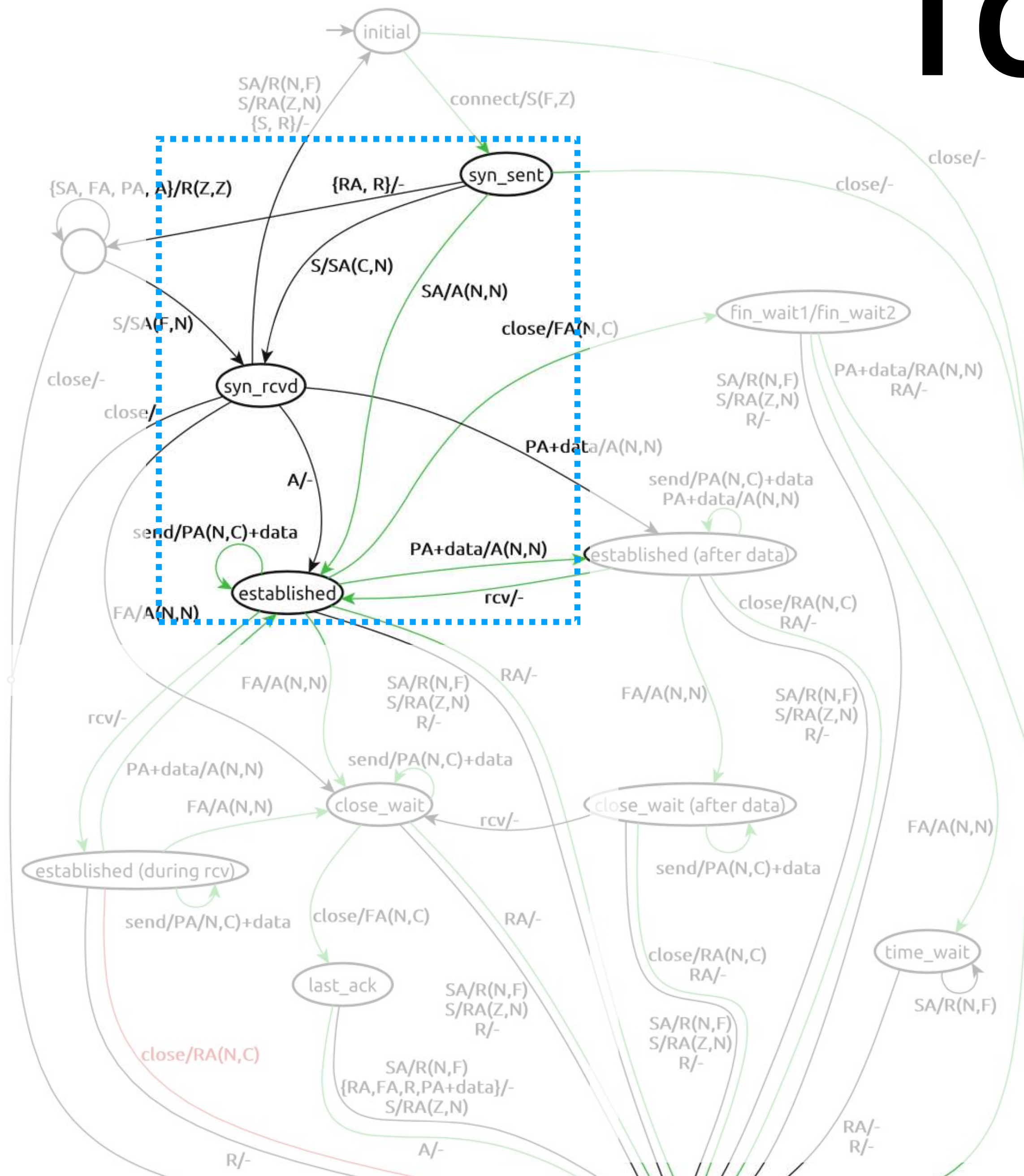
Output of the tool



TCP

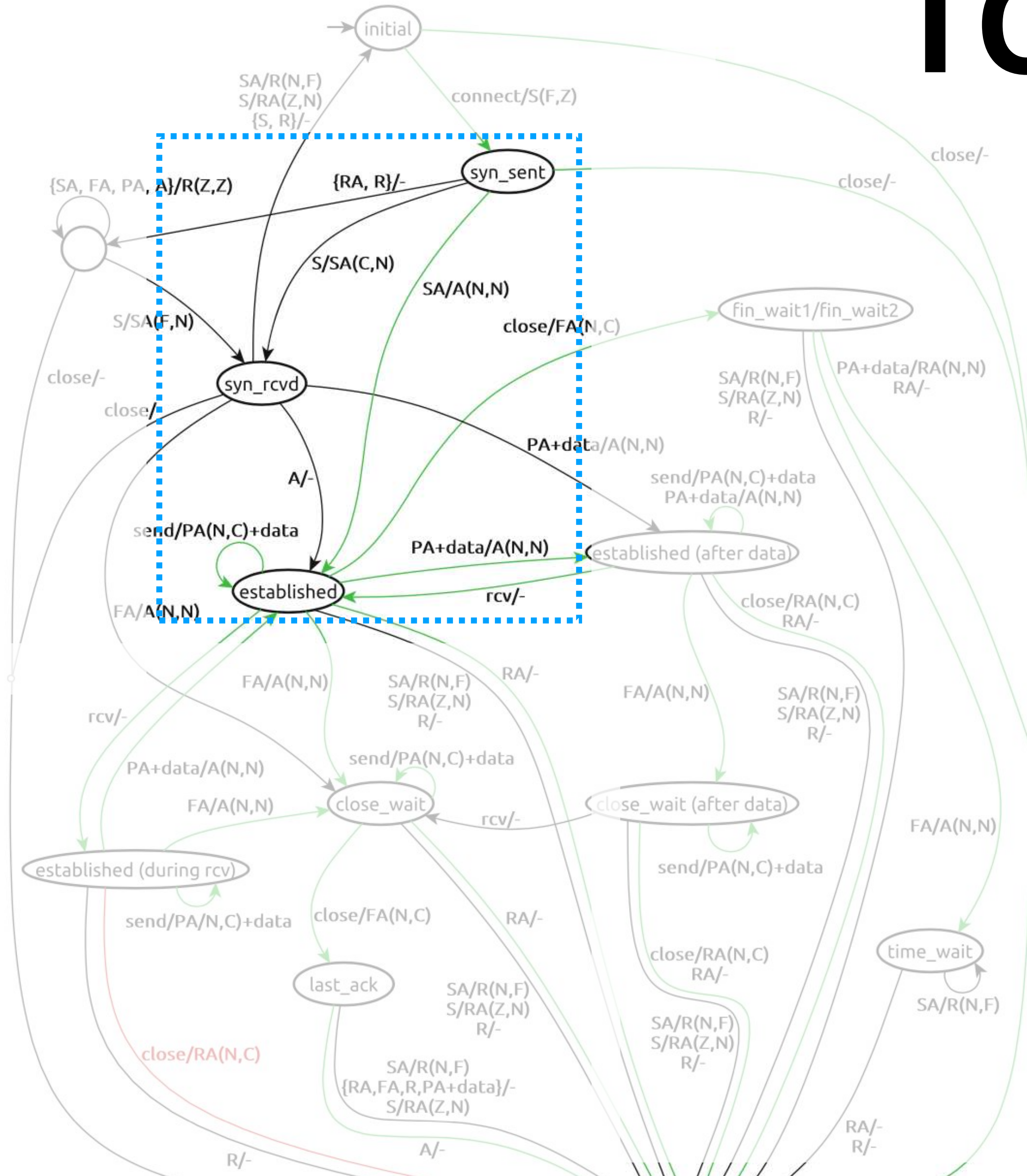


TCP



TCP is relatively simple and one can learn an (almost) complete model

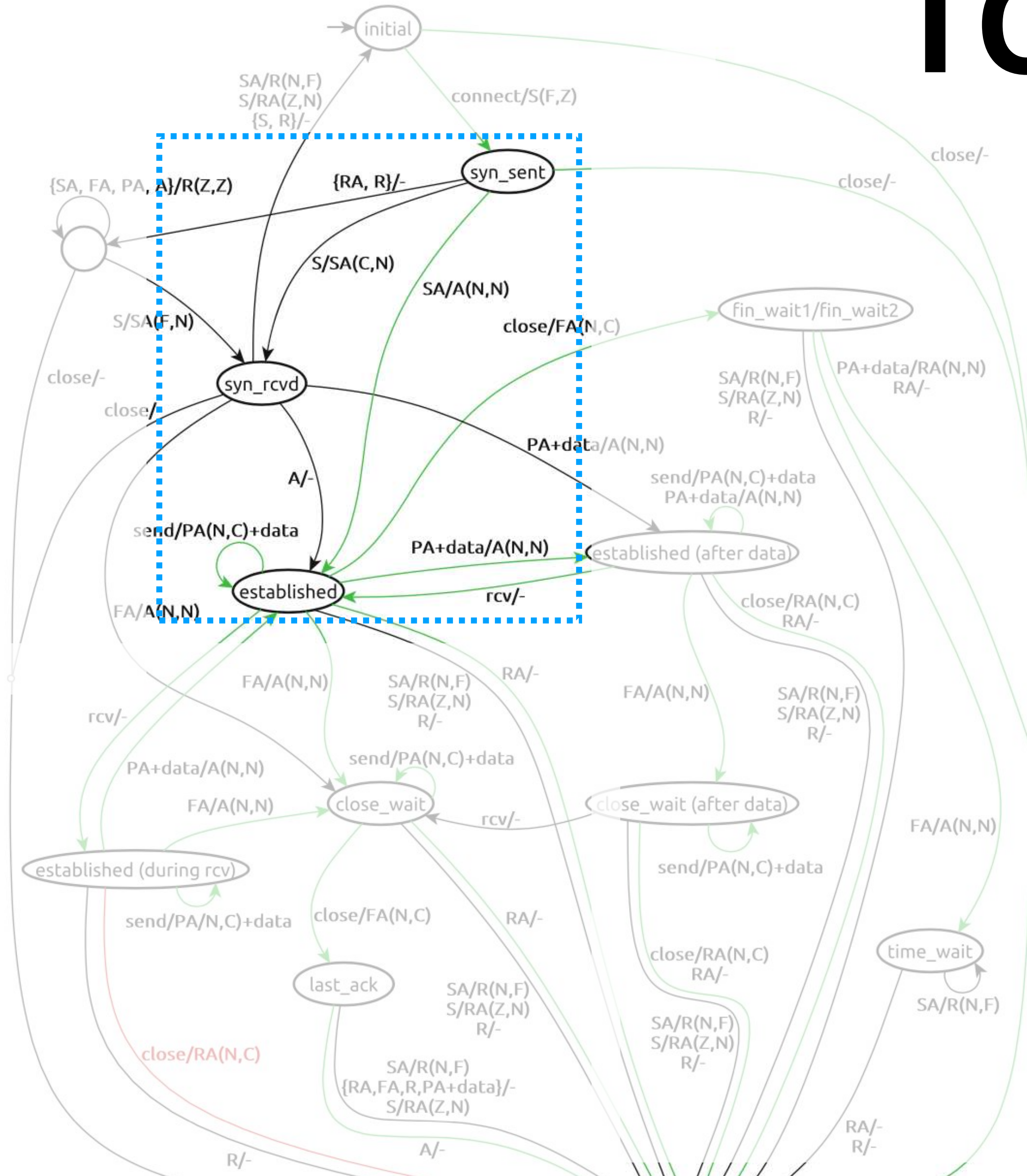
TCP



TCP is relatively simple and one can learn an (almost) complete model

How about more complex protocols?
How about focussed/partial behaviors?

TCP



TCP is relatively simple and one can learn an (almost) complete model

How about more complex protocols?
How about focussed/partial behaviors?

How effective are these models in helping developers finding bugs?

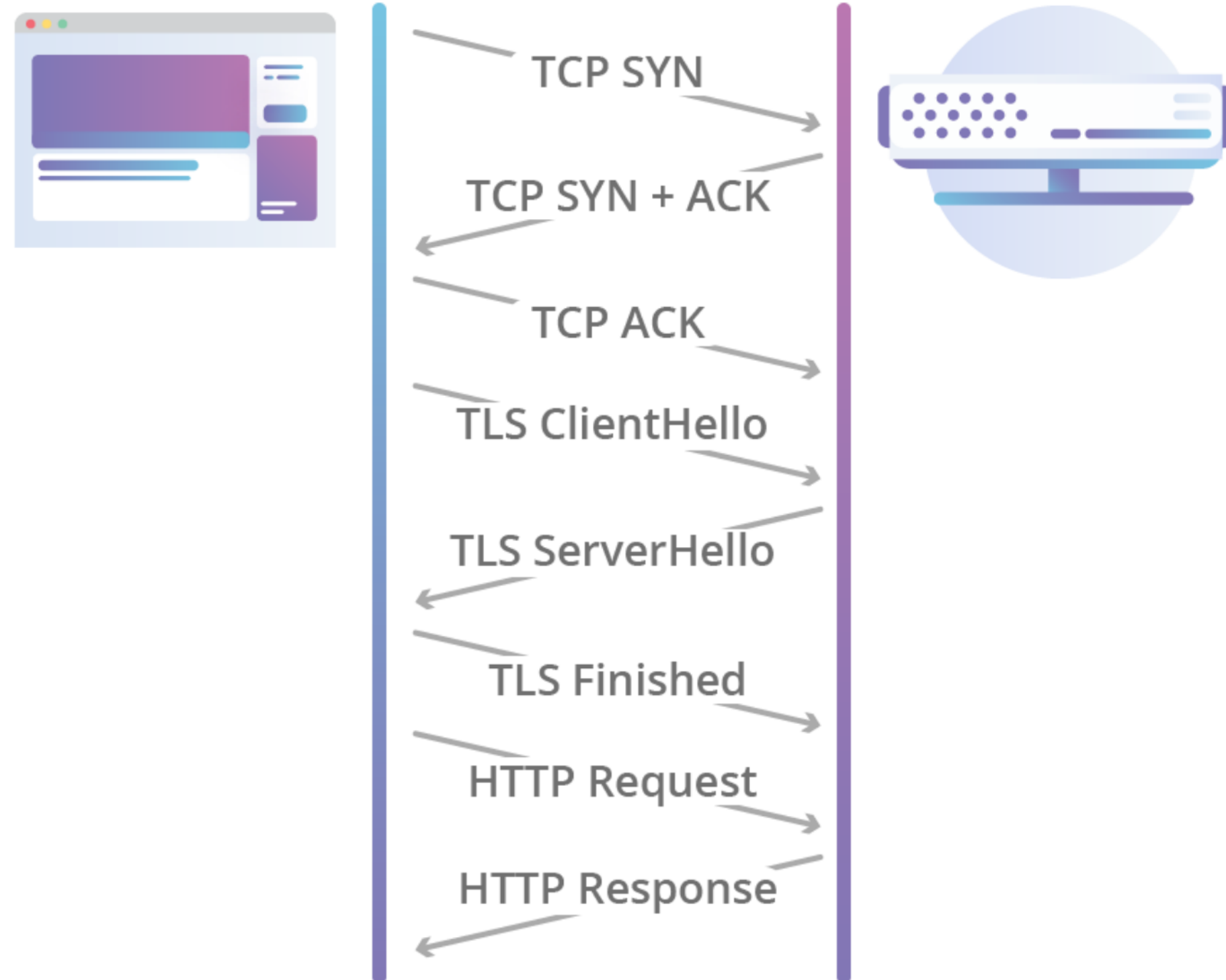
QUIC

QUIC

HTTP Request Over TCP + TLS

Client

Server

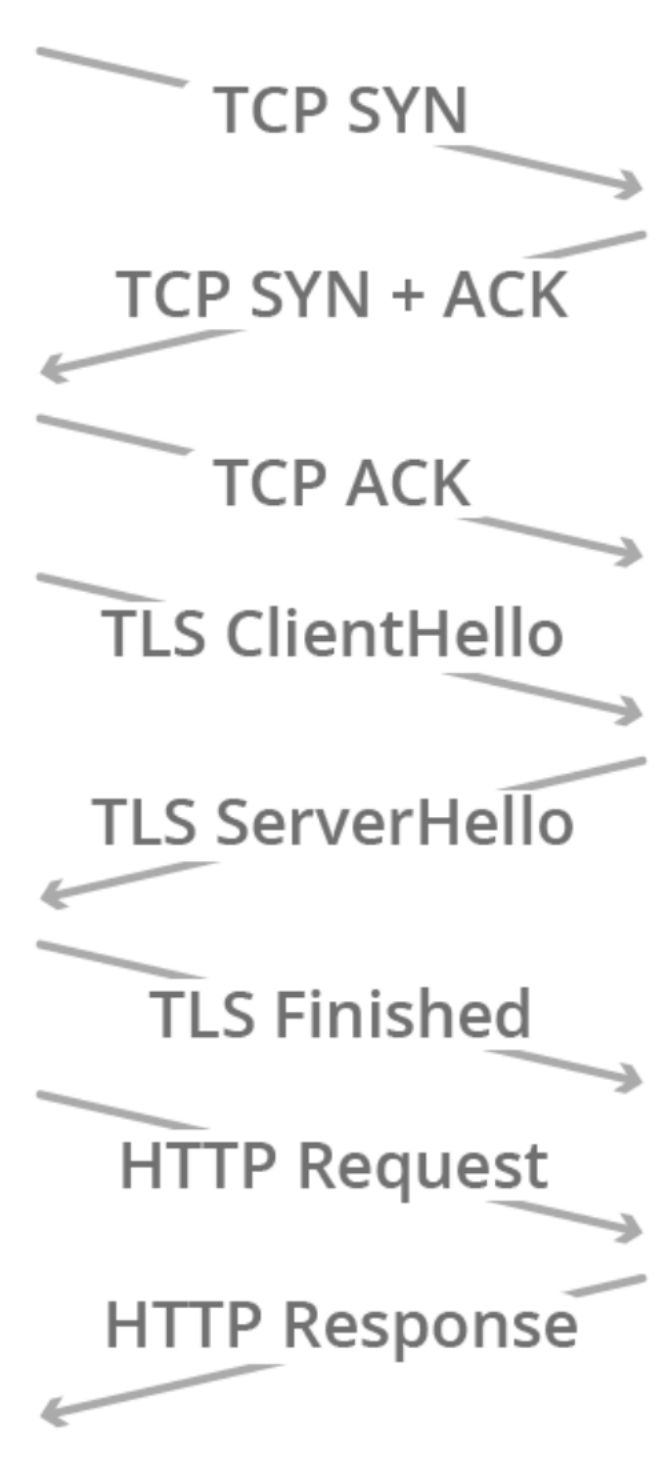


QUIC

HTTP Request Over TCP + TLS

Client

Server



HTTP/2

TLS1.2

TCP

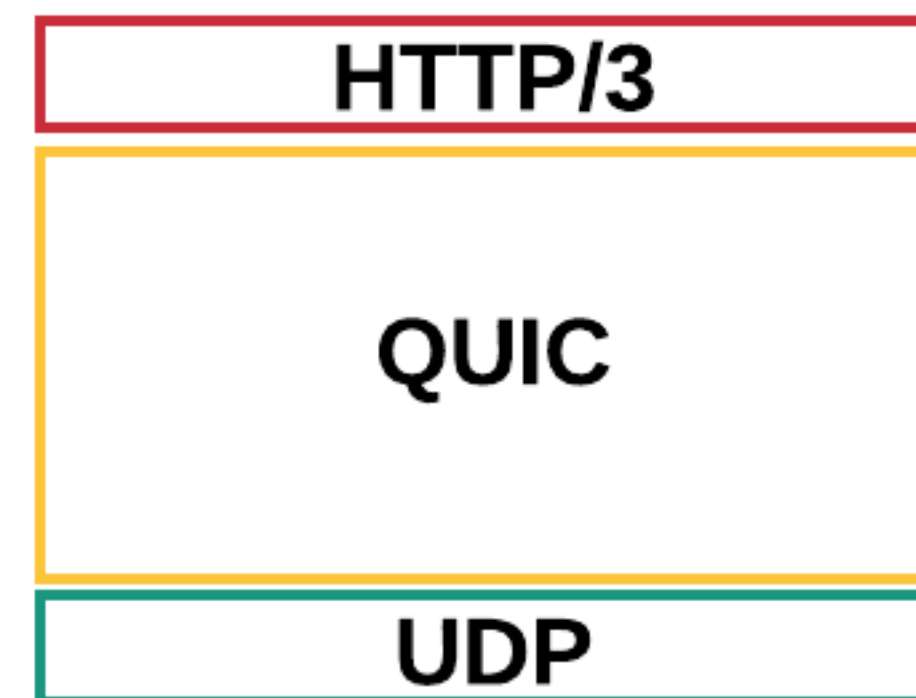
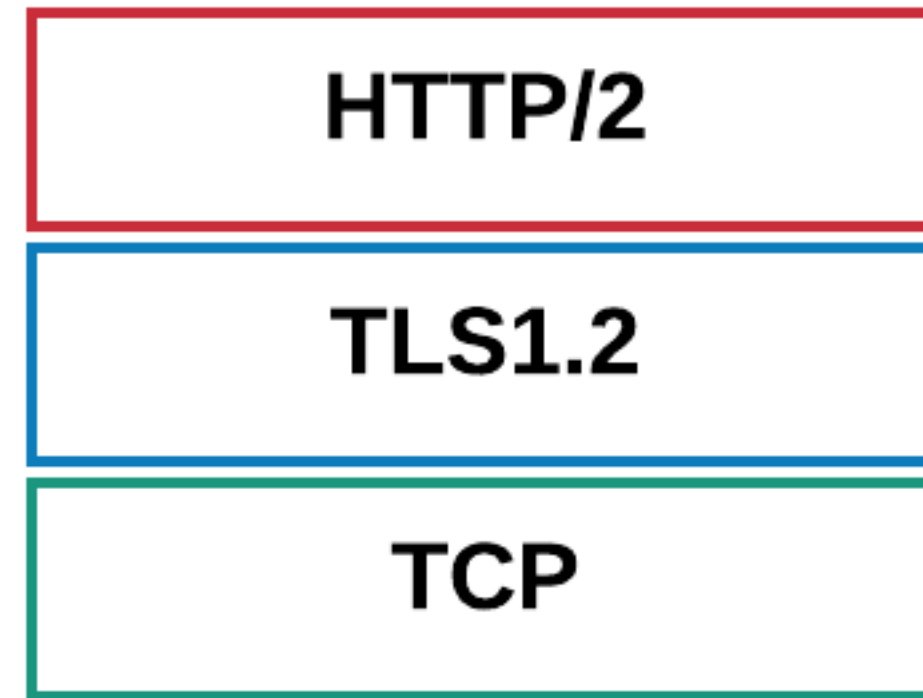
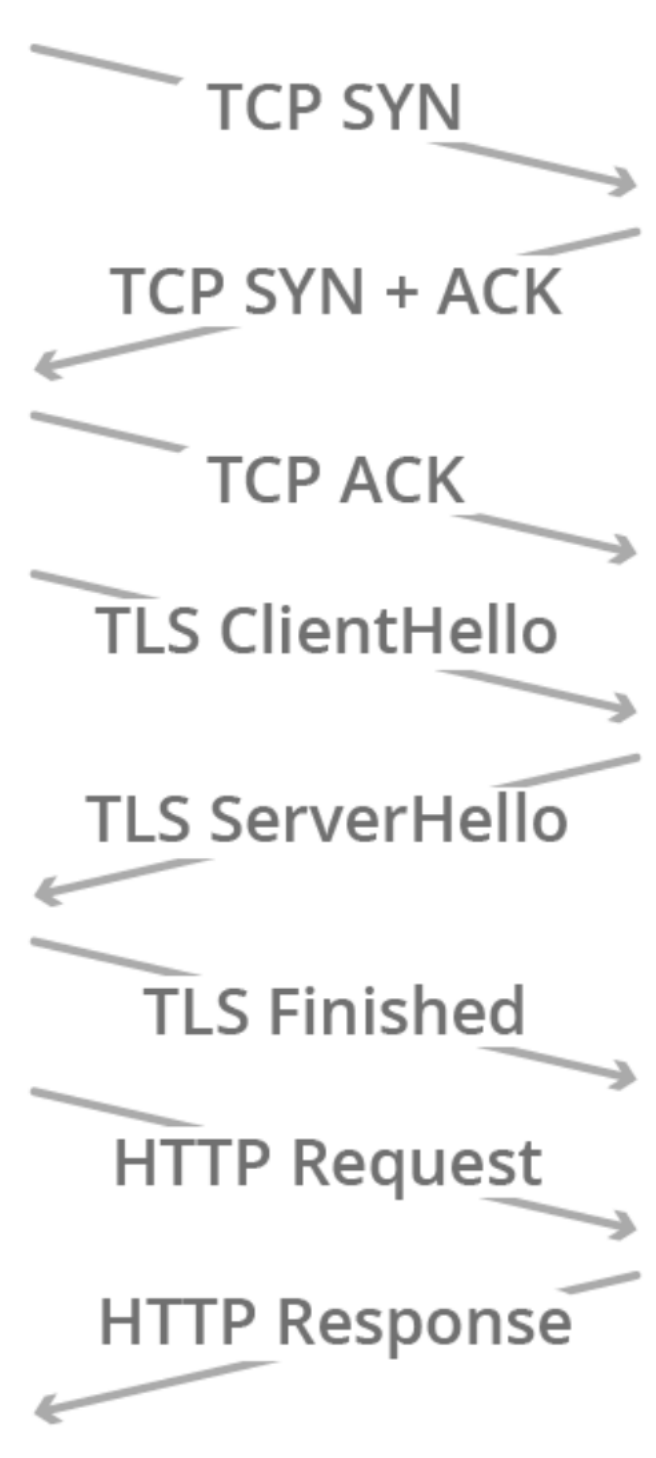
Internet Protocol (IP)

QUIC

HTTP Request Over TCP + TLS

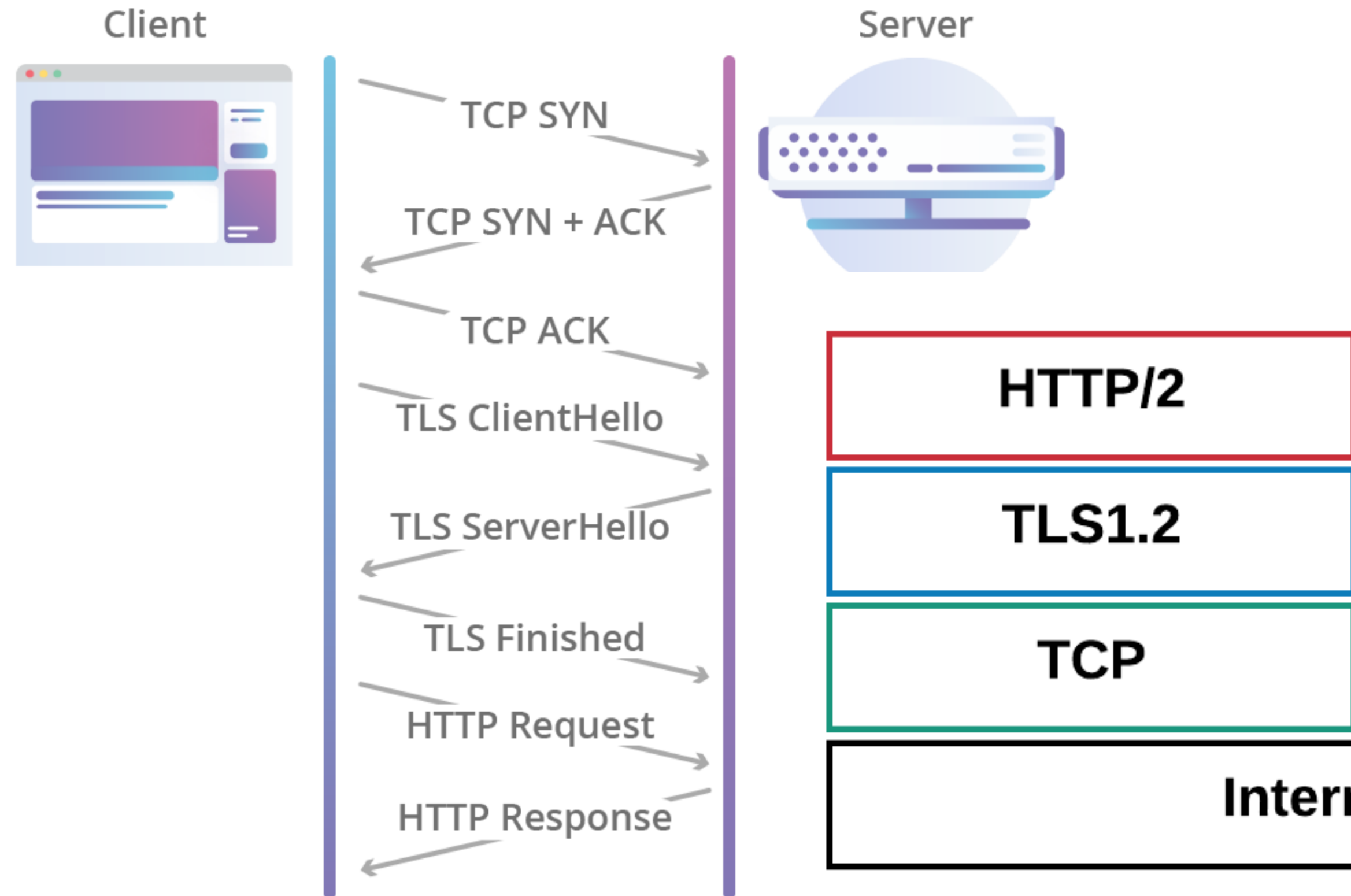
Client

Server

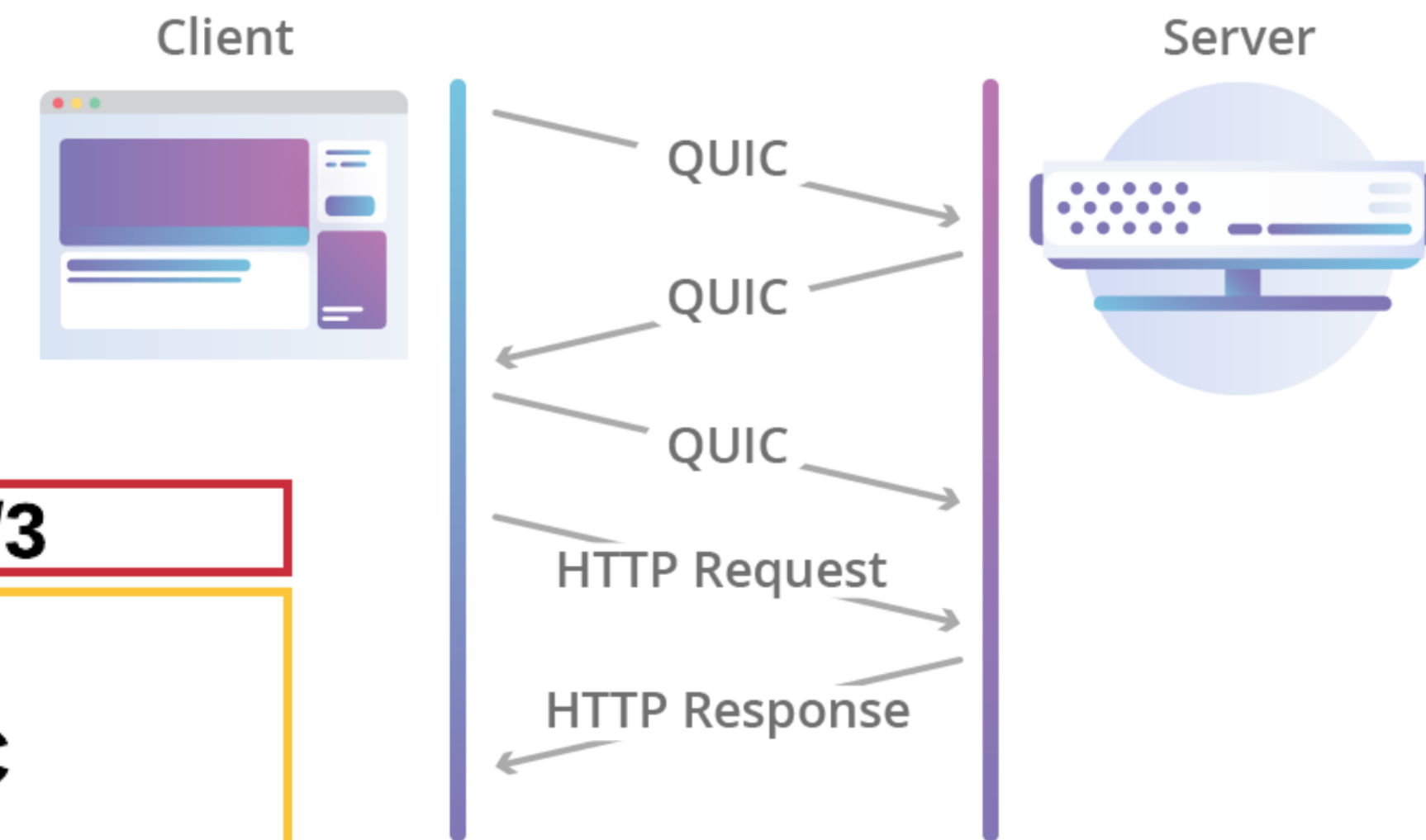


QUIC

HTTP Request Over TCP + TLS

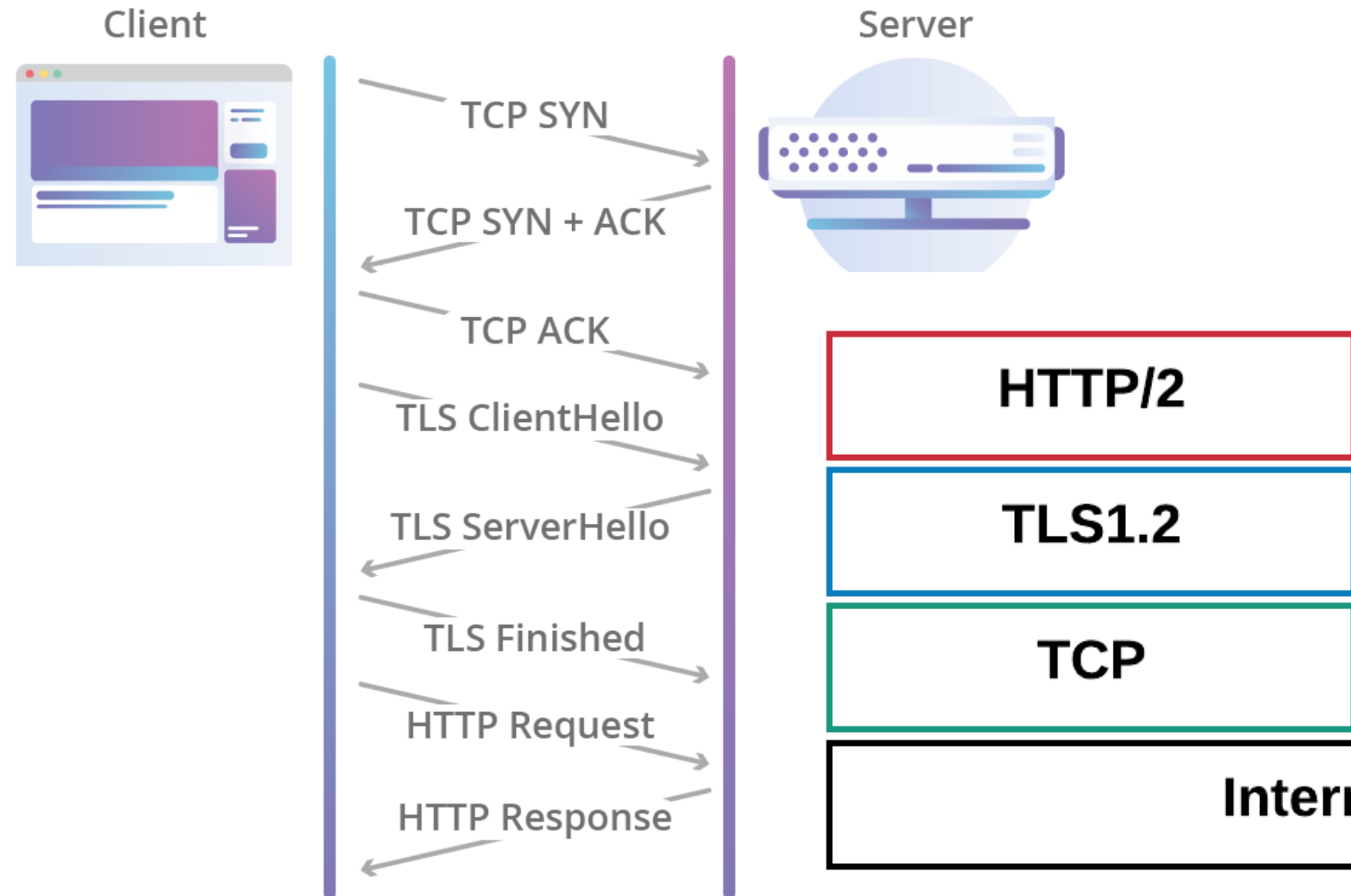


HTTP Request Over QUIC

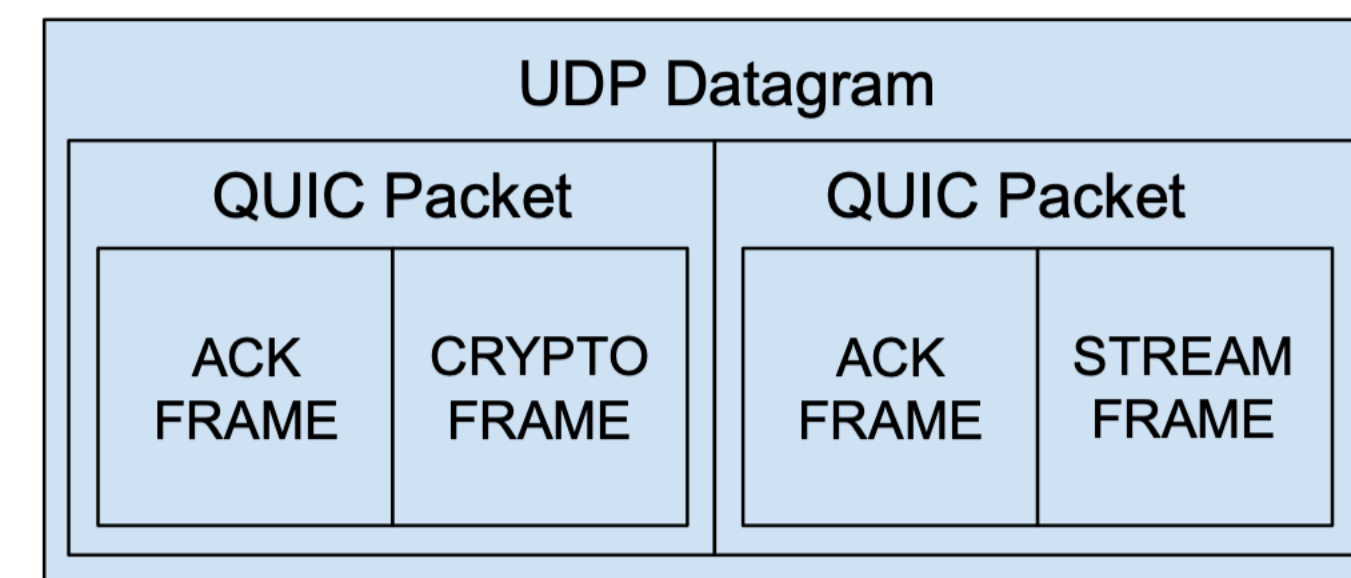
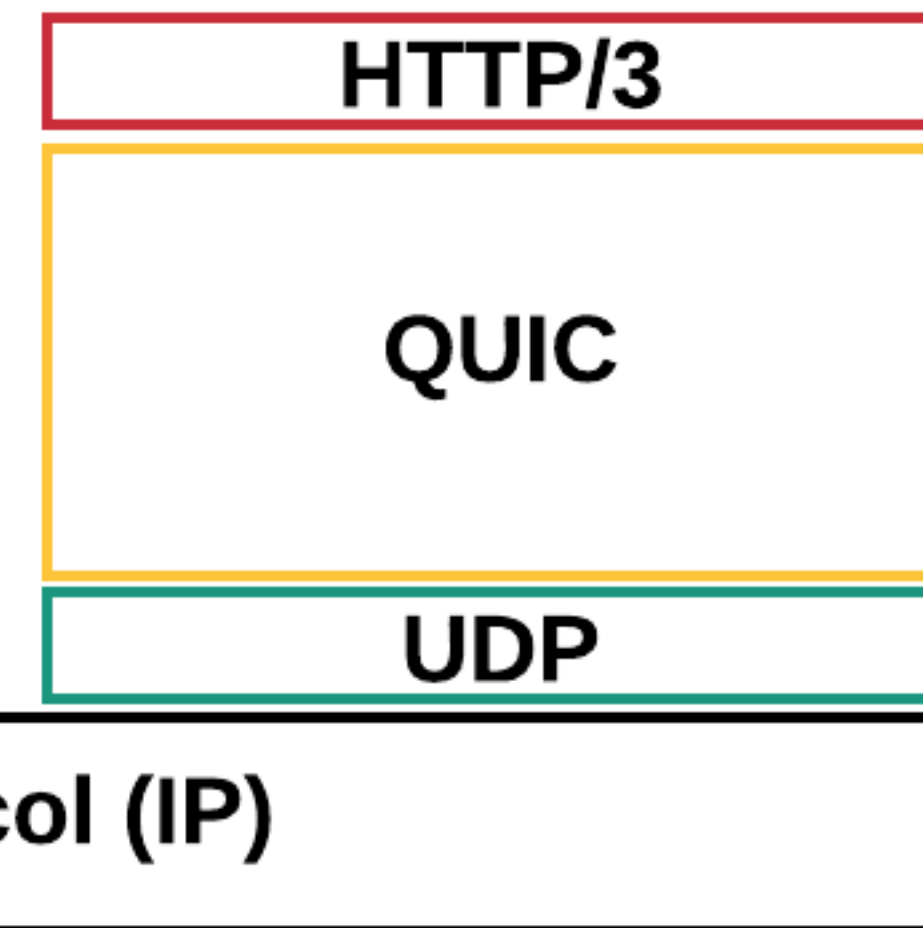
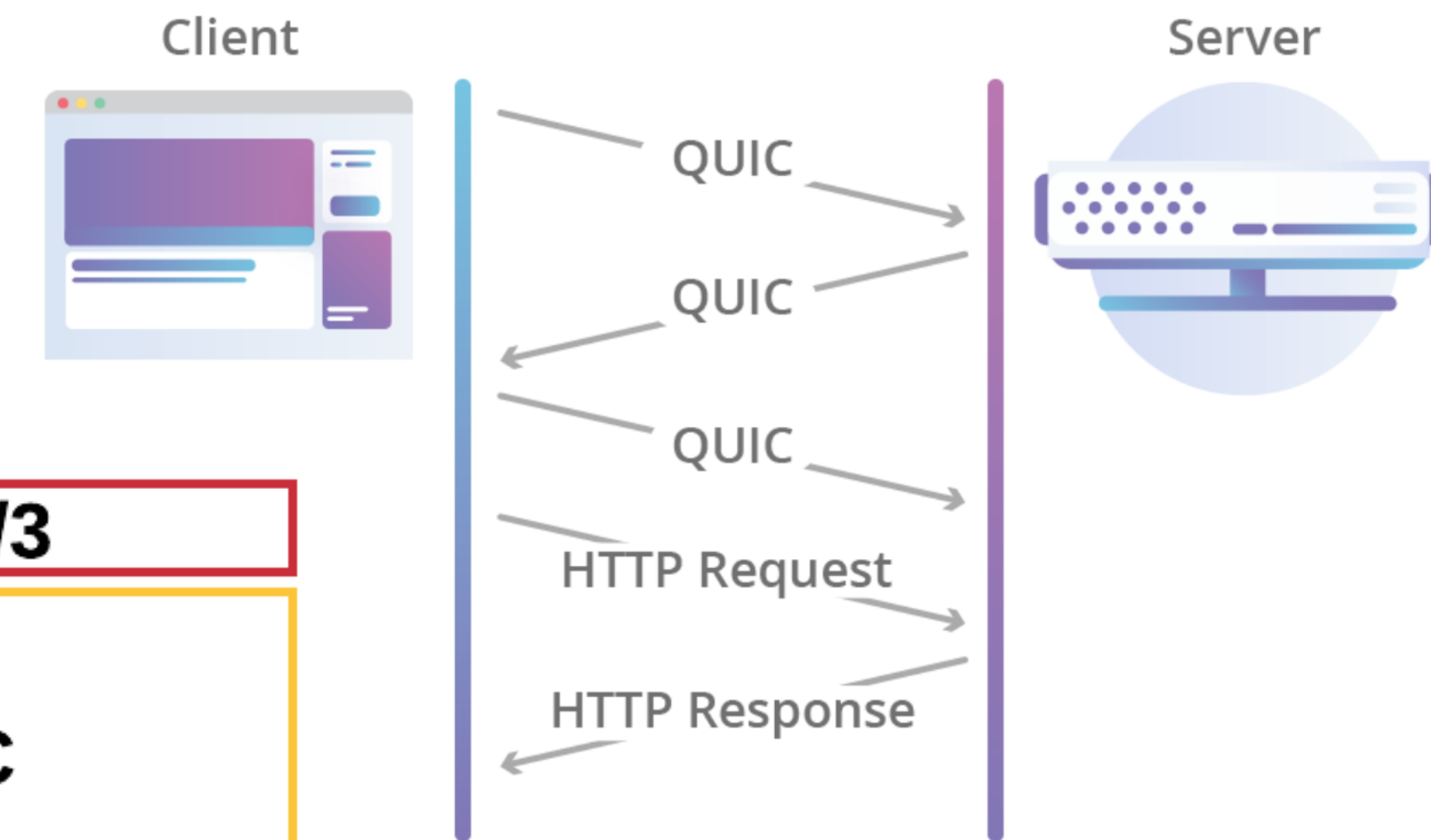


QUIC

HTTP Request Over TCP + TLS

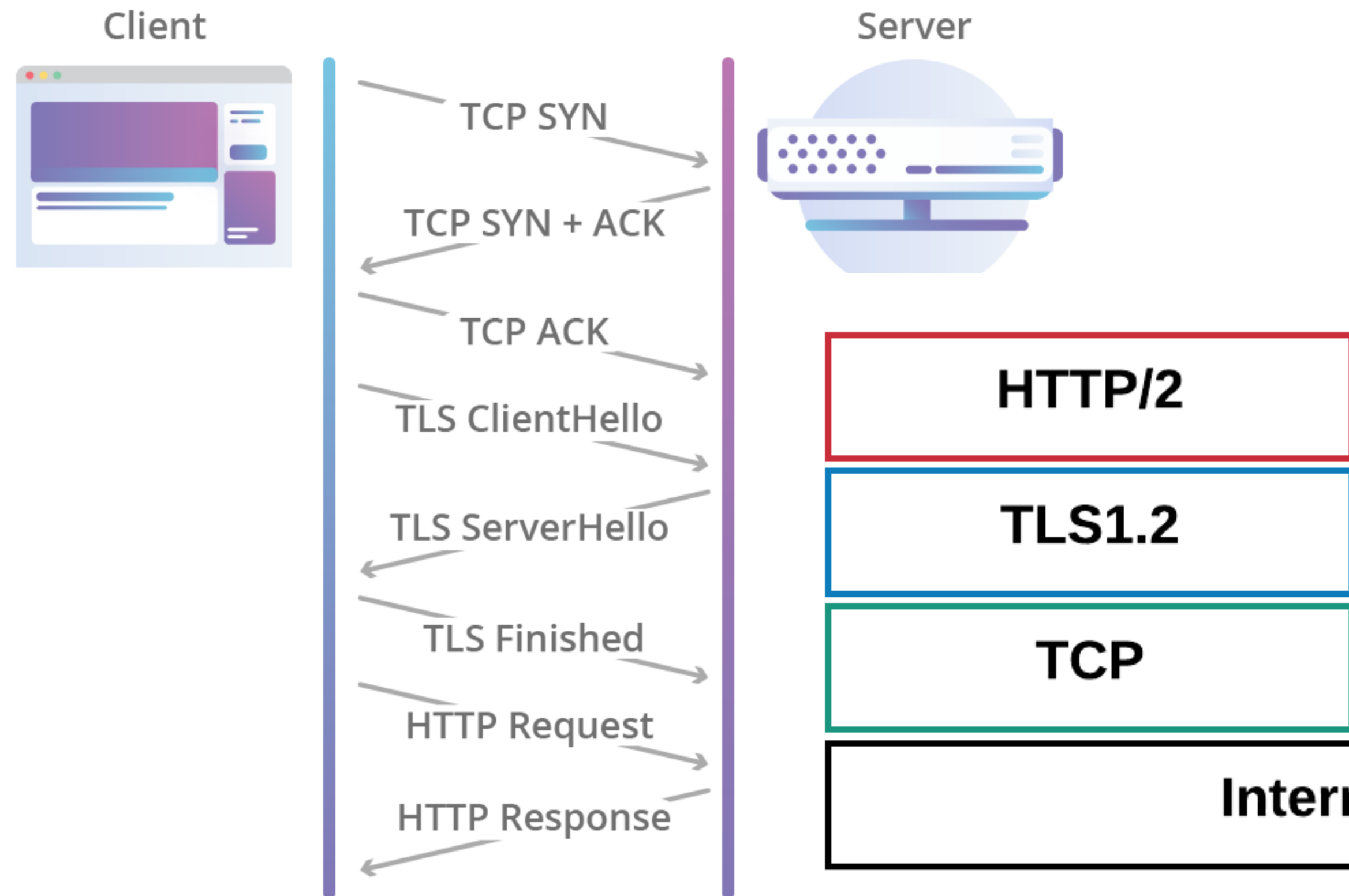


HTTP Request Over QUIC

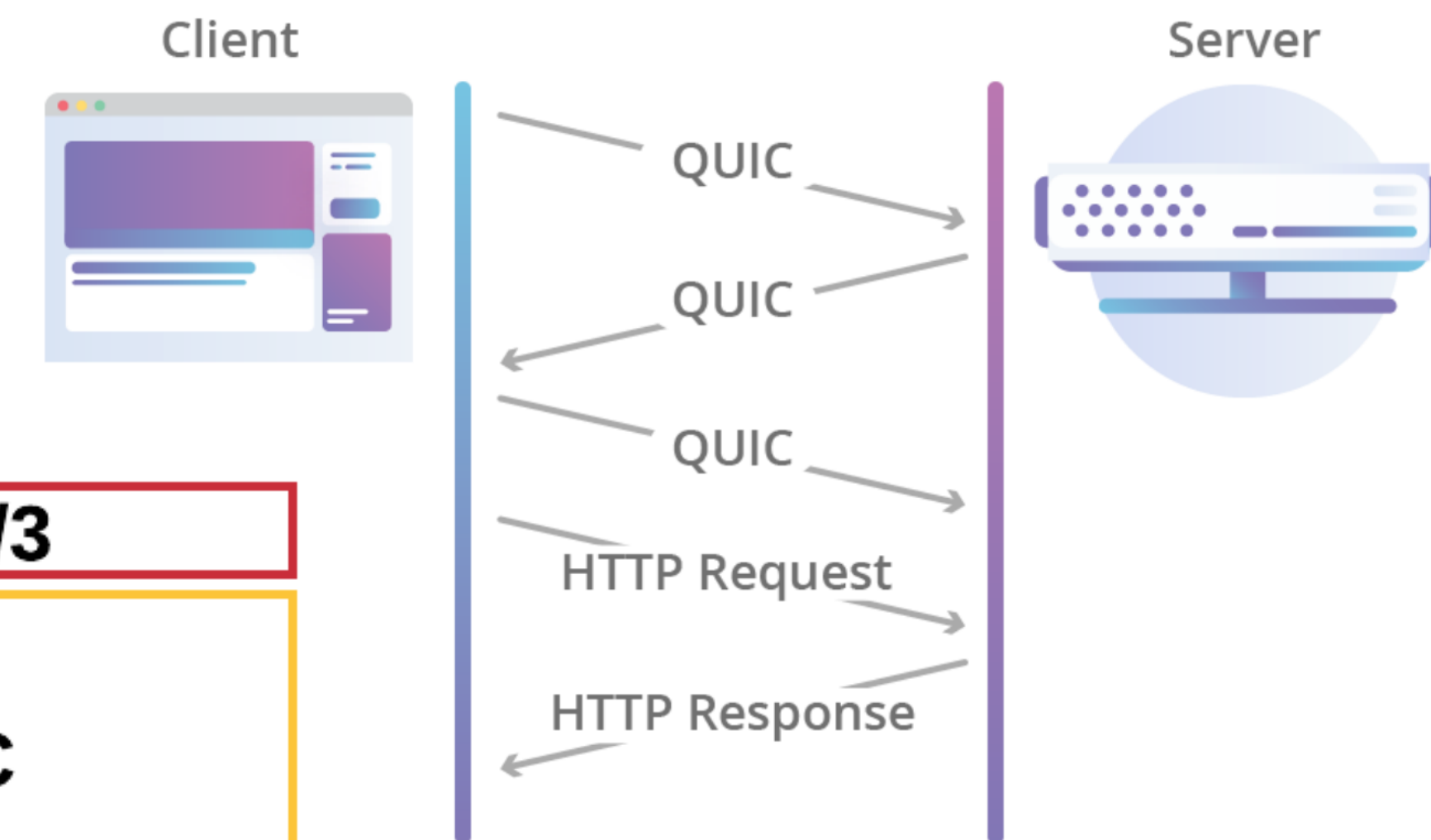


QUIC

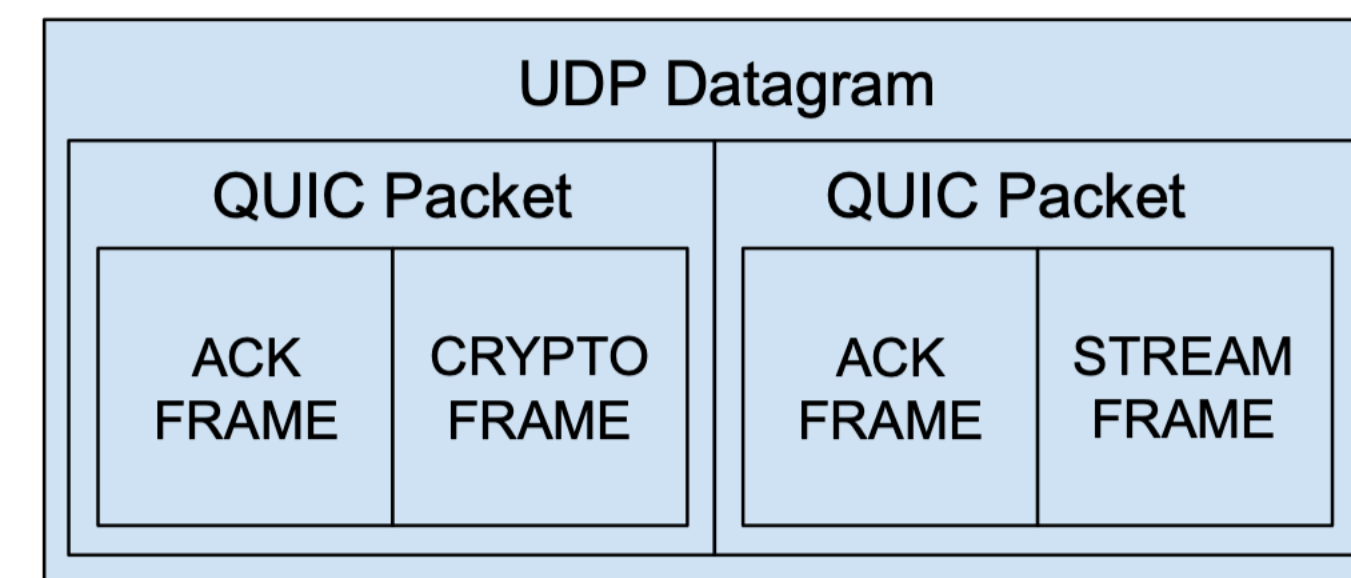
HTTP Request Over TCP + TLS



HTTP Request Over QUIC



INITIAL(?, ?)[CRYPTO], SHORT(?, ?)[ACK, STREAM]



Results

TCP

- Ubuntu 20.04.1 LTS TCP stack, kernel version 5.8.0-40-generic
- 6 states and 42 transitions; 4,726 membership queries to learn
- Replicated results of Fiterău-Broștean but 300 vs 3000 lines of extra code!

QUIC

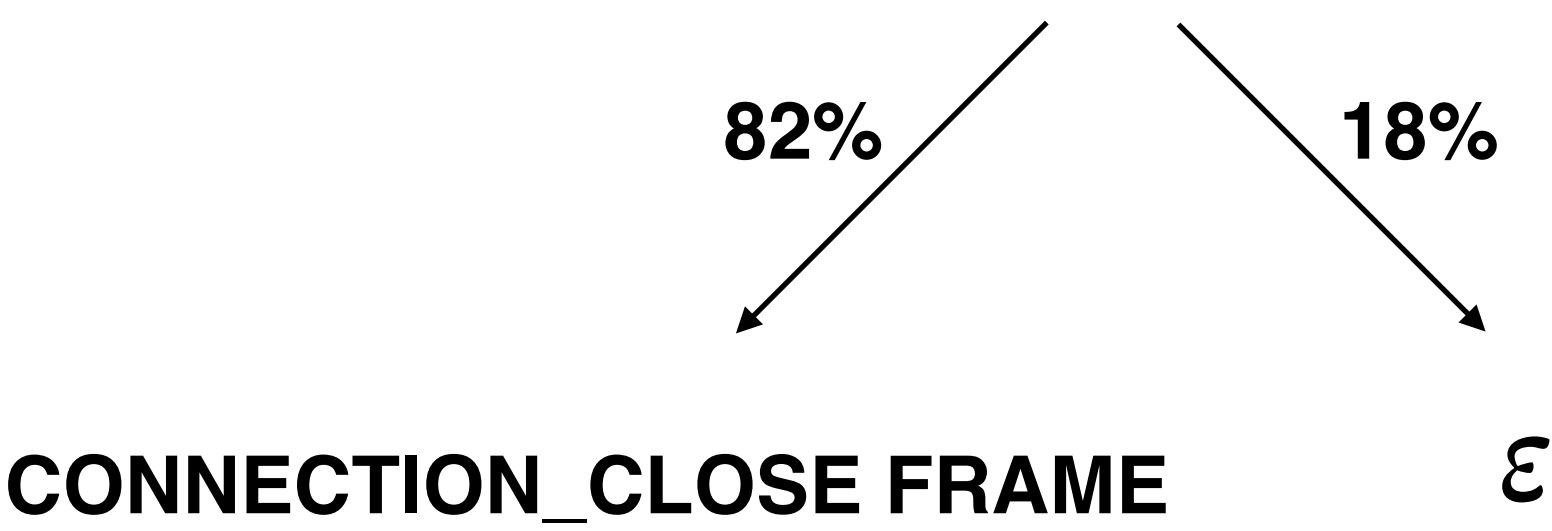
- 4 implementations: Cloudflare (quiche), Google, Facebook, Quick-tracker
- Model sizes varied (8-12 states and 56-84 transitions); ~10-25K membership queries to learn
- Discovered 3 bugs and filed an RFC change (2000 lines annotations)

Results

Facebook's Mvfst

Non-determinism Check

- INITIAL(?,?)[CRYPTO],
- HANDSHAKE(?,?)[ACK,HANDSHAKE_DONE],
- SHORT(?,?)[ACK]



Google's QUICServer

Register Synthesis

```
STREAM_DATA_BLOCKED Frame {
  Type (i) = 0x15,
  Stream ID (i),
  Maximum Stream Data (i),
}
```

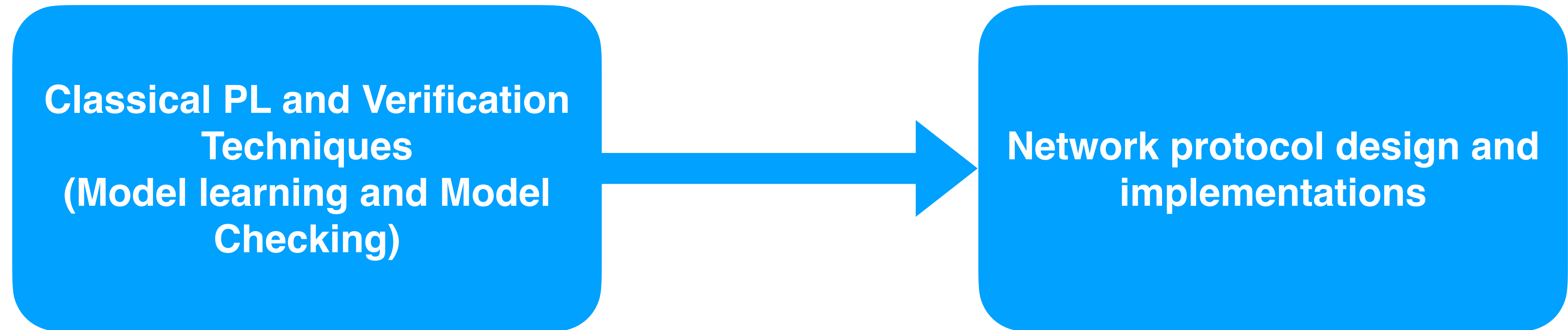
Figure 36: STREAM_DATA_BLOCKED Frame Format

STREAM_DATA_BLOCKED frames contain the following fields:

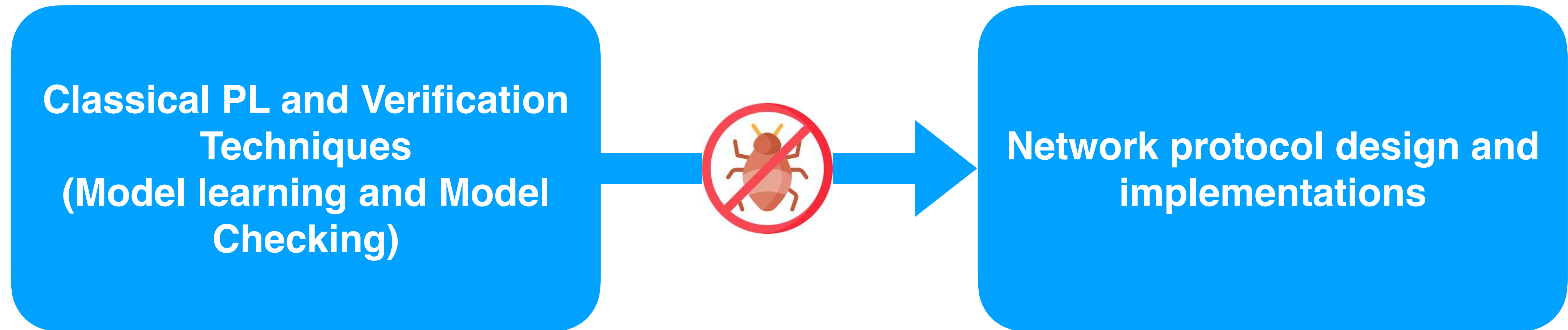
Stream ID: A variable-length integer indicating the stream which is flow control blocked.

Maximum Stream Data: A variable-length integer indicating the offset of the stream at which the blocking occurred.

Summary

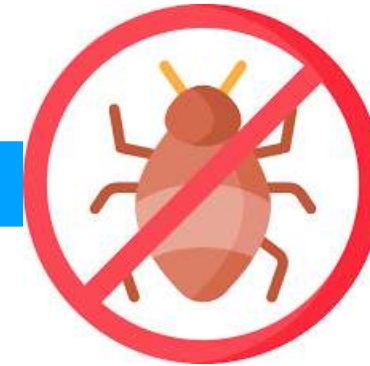


Summary



Summary

**Classical PL and Verification
Techniques
(Model learning and Model
Checking)**

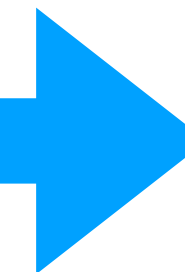
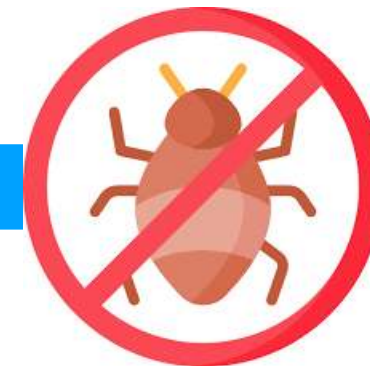


**Network protocol design and
implementations**

**Particularly fruitful in
new and complex
protocols like QUIC**

Summary

**Classical PL and Verification
Techniques
(Model learning and Model
Checking)**



**Network protocol design and
implementations**

**Particularly fruitful in
new and complex
protocols like QUIC**

**Prognosis is
modular enough it can
easily be applied to**

