The Frontiers of Active Learning in Network Verification

Alexandra Silva

Cornell Bowers CIS
Computer Science

Unifying Speaker

Unifying Speaker

ESOP
31st European Symposium on Programming

FASE
25th International Conference on Fundamental Approaches to Software Engineering

FoSSaCS
25th International Conference on Foundations of Software Science and Computation Structures

TACAS
28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems

Unifying Speaker

ESOP
31st European Symposium on Programming

FASE
25th International Conference on Fundamental Approaches to Software Engineering

FoSSaCS
25th International Conference on Foundations of Software Science and Computation Structures

TACAS
28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems

Unifying Speaker

ESOP
31st European Symposium on Programming

FASE
25th International Conference on Fundamental Approaches to Software Engineering

FoSSaCS
25th International Conference on Foundations of Software ~~Science~~ and Computation Structures

TACAS
28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems

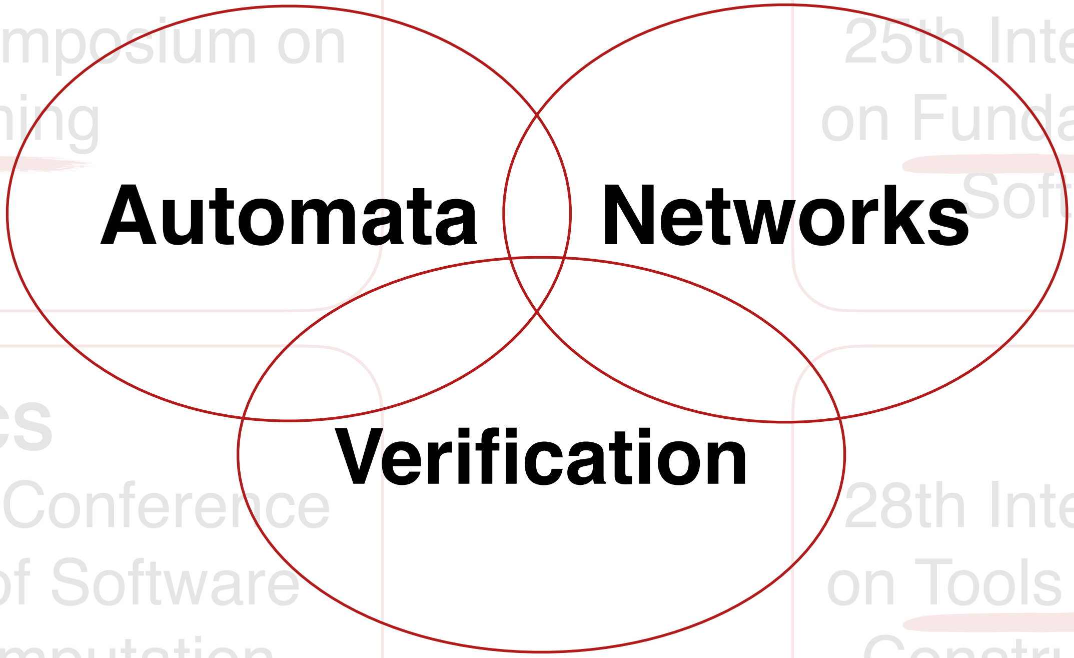
Unifying Speaker

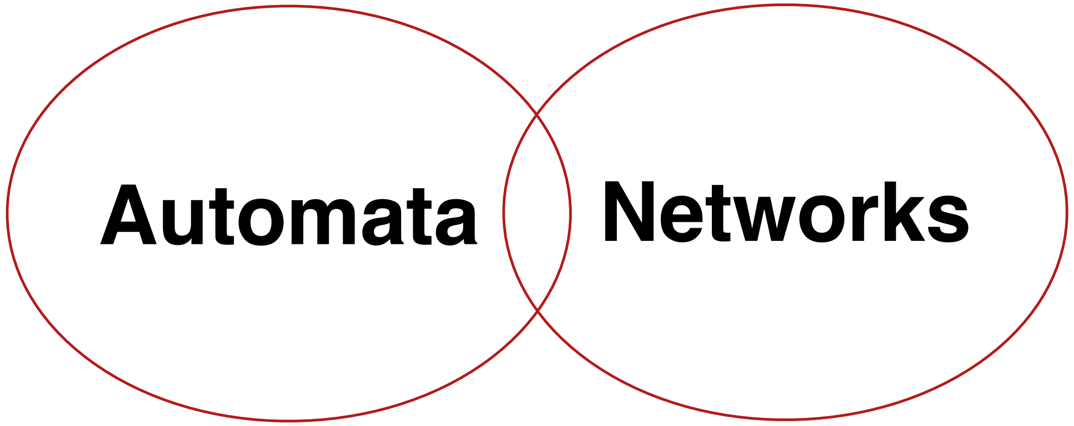
ESOP
31st European Symposium on Programming

FASE
25th International Conference on Fundamental Approaches to Software Engineering

FoSSaCS
25th International Conference on Foundations of Software Science and Computation Structures

TACAS
28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems




Automata
Networks

Automata
Networks

Automata

Networks

Long tradition

Model Checking

Model Learning

....

Automata

Networks

Long tradition

Model Checking

Model Learning

....

Long-ish tradition

Traditional networks

SDN

....

Automata

Networks

Programmable

Semantics

Tool design

....

Long tradition

Model Checking

Model Learning

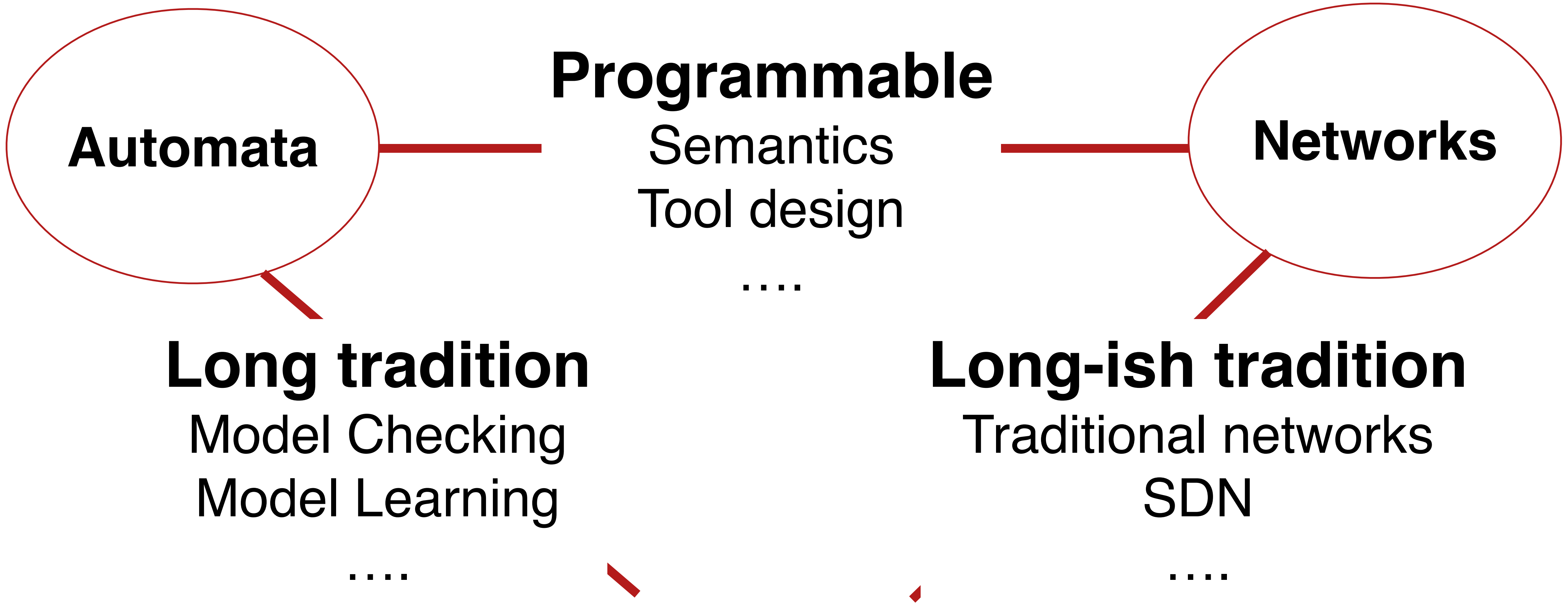
....

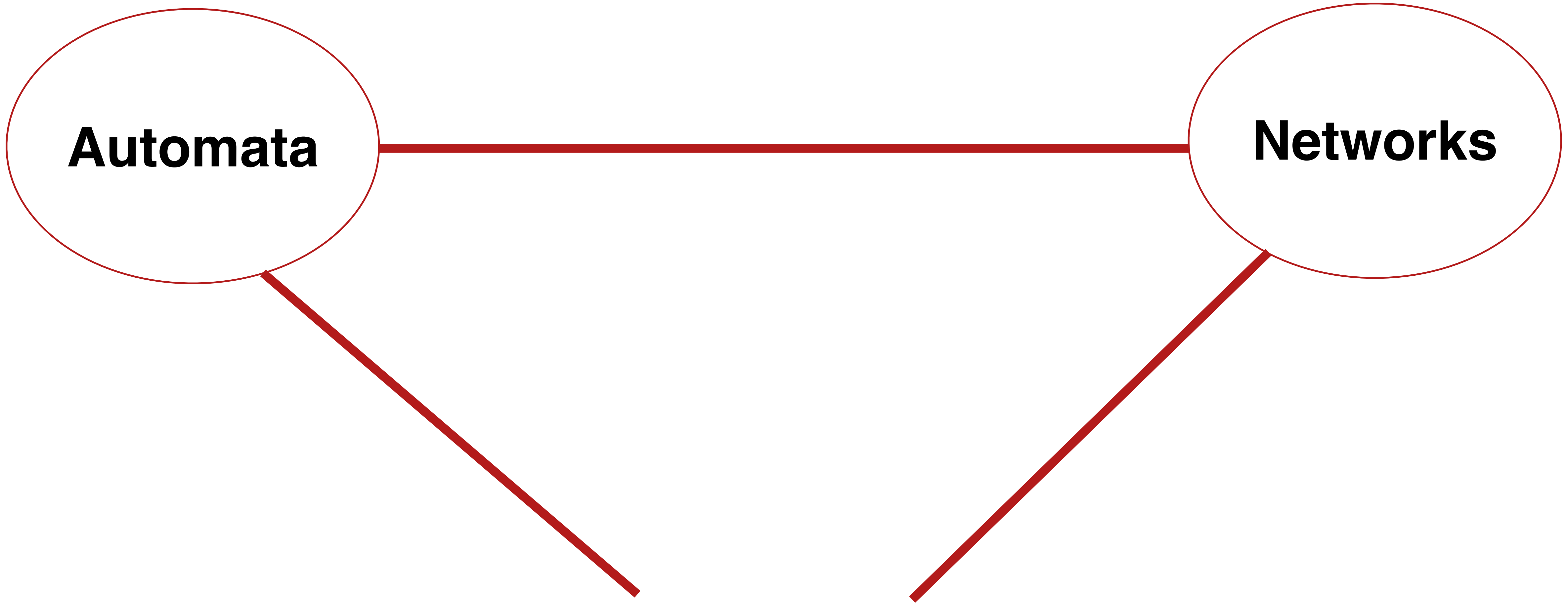
Long-ish tradition

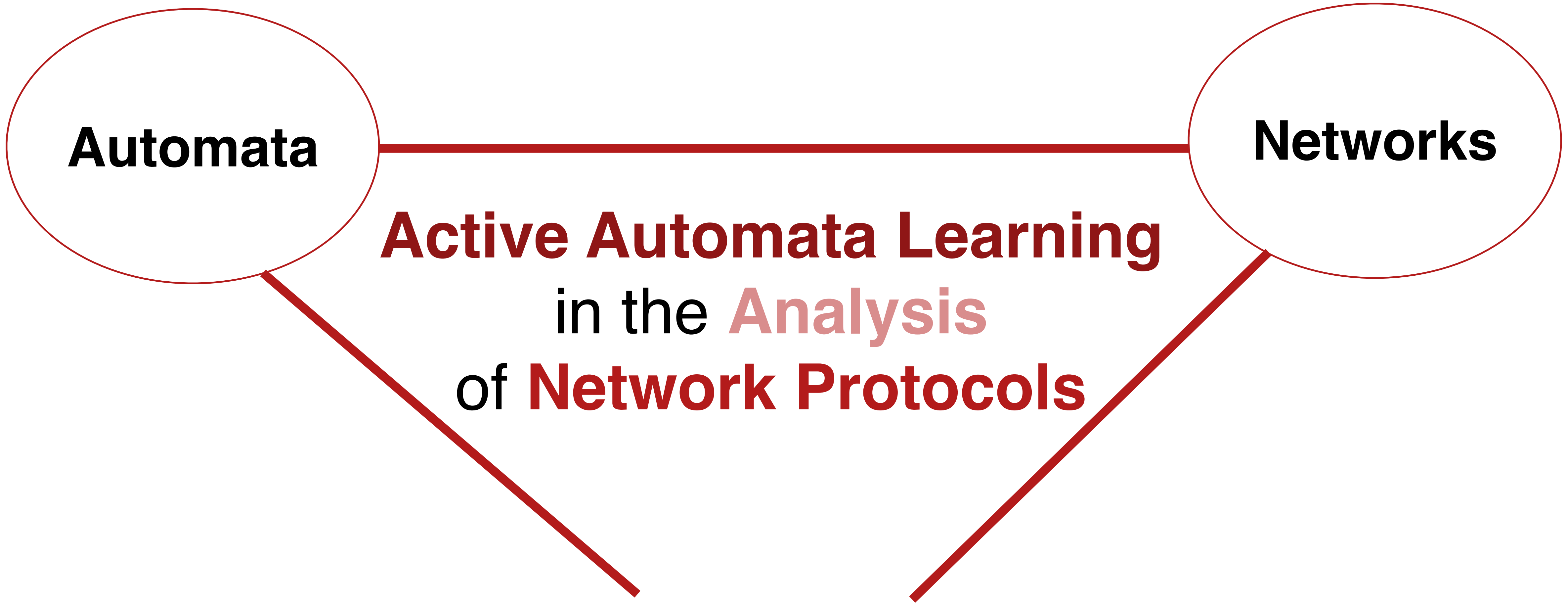
Traditional networks

SDN

....


Automata
Programmable
Semantics
Tool design
….
Networks
Long tradition
Model Checking
Model Learning
….
Long-ish tradition
Traditional networks
SDN
….

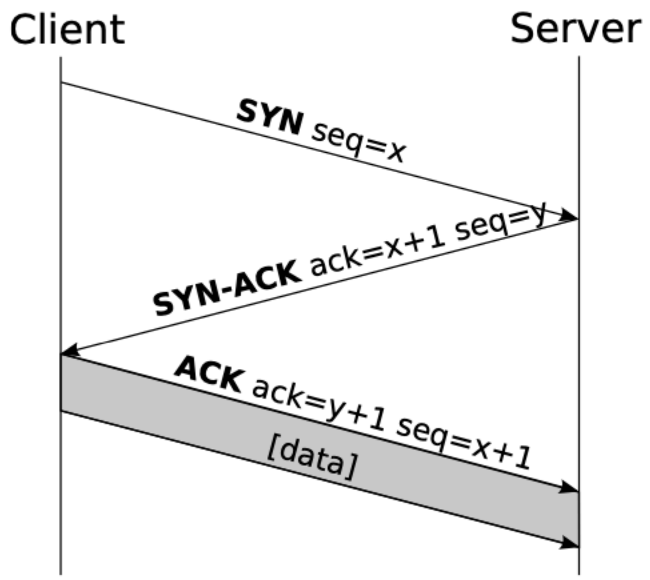

Automata
Networks



Active Automata Learning in the Analysis of Network Protocols

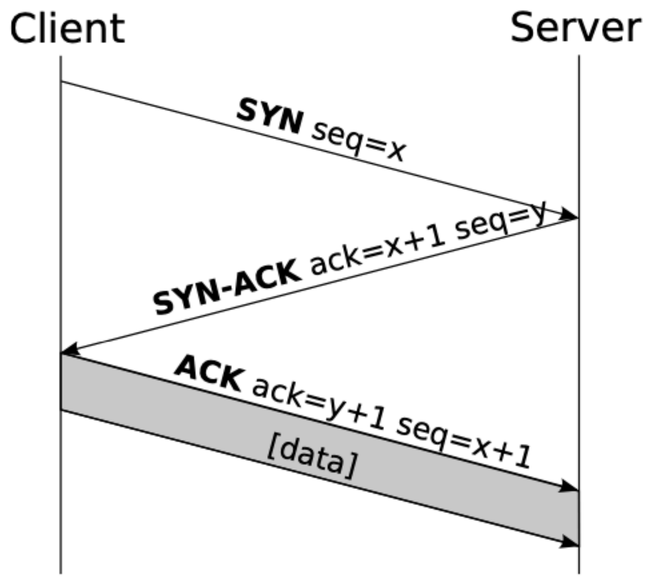
Why Network Protocols?

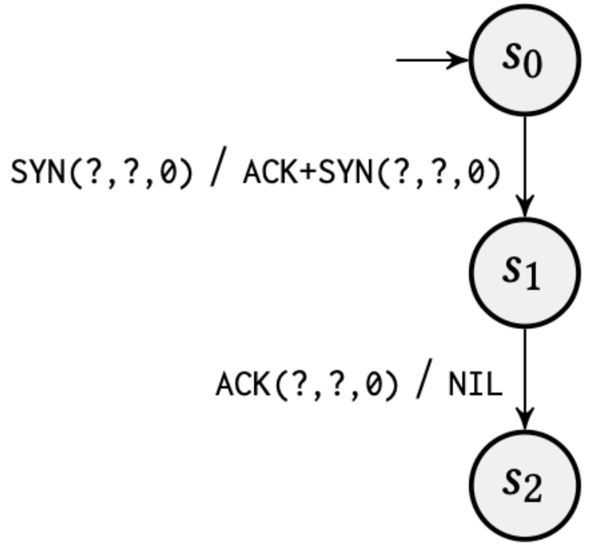
Why Network Protocols?



TCP/IP , QUIC, …
Simple fragments

Why Network Protocols?

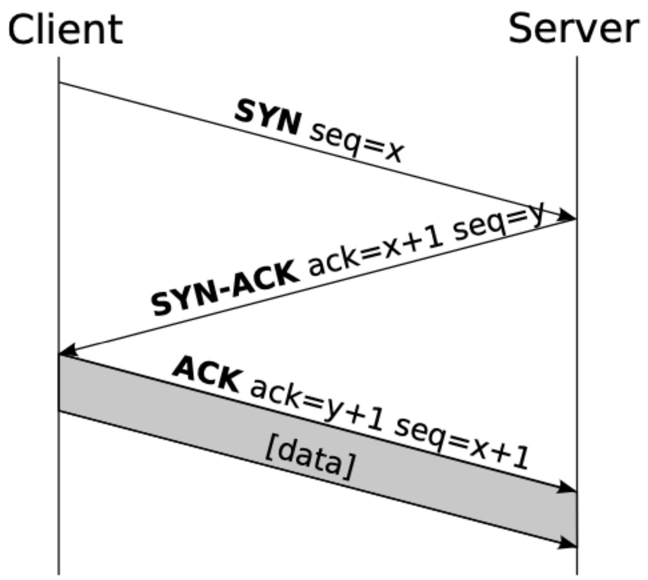


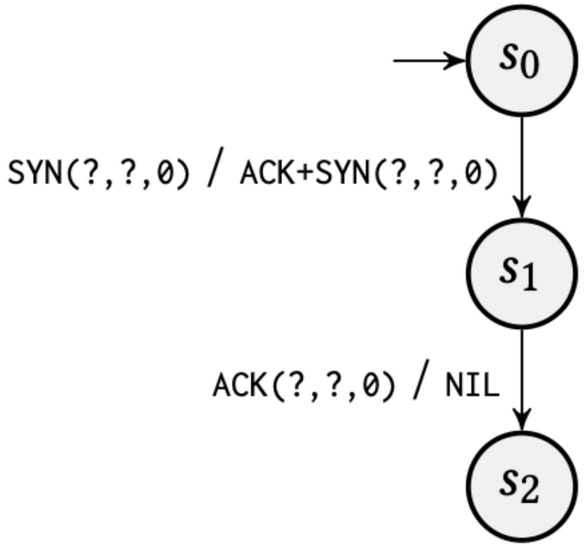


TCP/IP , QUIC, …
Simple fragments

Amenable to descriptions as finite state machines

Why Network Protocols?





TCP/IP , QUIC, …
Simple fragments

Amenable to descriptions as finite state machines

Many implementations!

Why Network Protocols?

Many implementations!

Why Network Protocols?

Many implementations!

Why Network Protocols?

Many implementations!

Implementations should follow a well-established behaviour

Why Network Protocols?

Many implementations!

Implementations should follow a well-established behaviour

Comparisons between implementations can be a powerful analysis tool

Why Network Protocols?

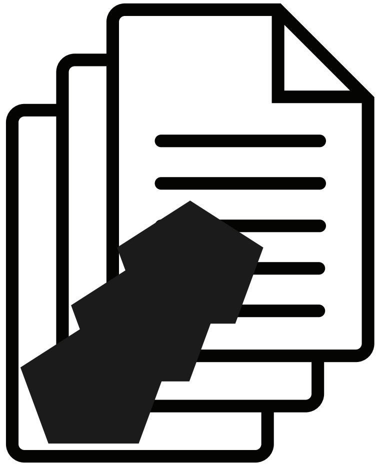
Many implementations!

Implementations should follow a well-established behaviour

Comparisons between implementations can be a powerful analysis tool

Different design choices might expose inconsistencies or underspecification in the spec

Active Automata Learning

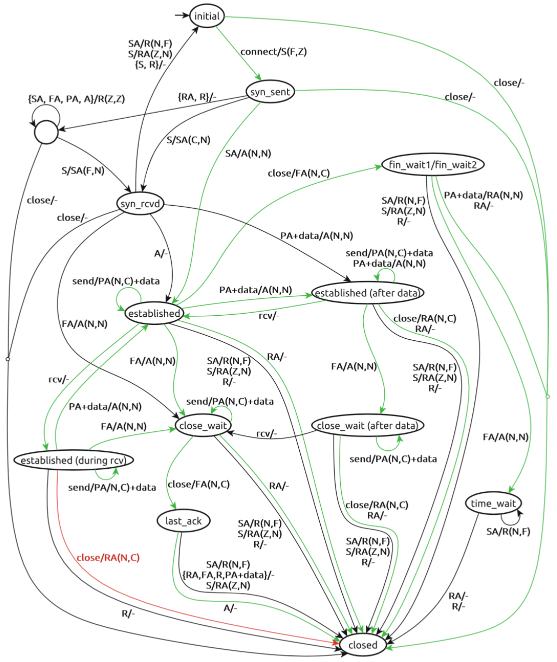


Black-box interactions

Zoom-in specific part of code

Incrementally build a model

Refine with properties of interest



Cornell Bowers C·IS
Computer Science

Active Automata Learning

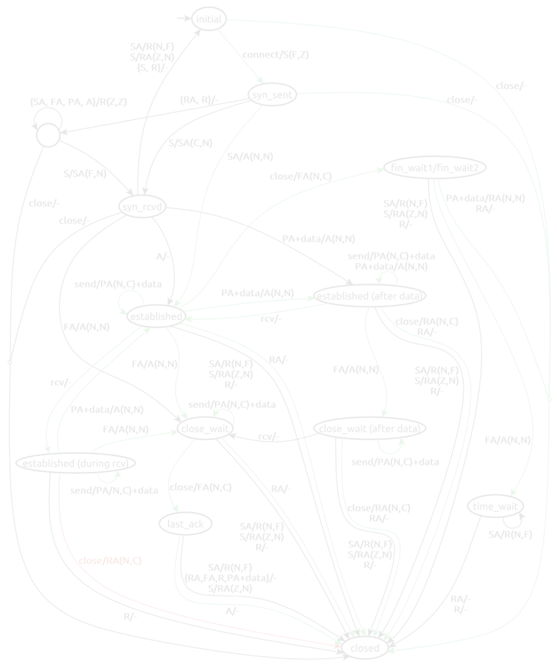
Black-box interactions

Zoom-in specific part of code

crementally uild a model

Refine with roperties of interest

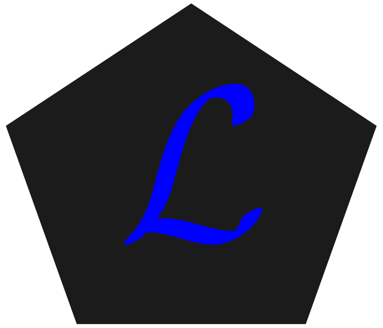
Automated



DFA Learning (L*, Angluin'87)

$\mathcal{L}$
Regular Language

DFA Learning (L*, Angluin'87)



Regular Language

Membership query

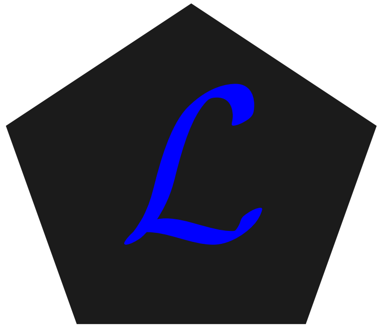
Q: $w \in \mathcal{L}?$

A: Y/N

Observation table

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

DFA Learning (L*, Angluin'87)



Regular Language

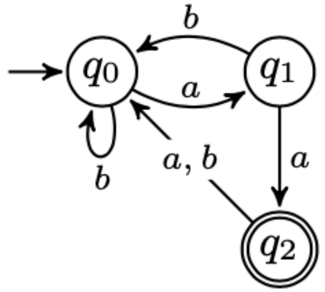
Membership query

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

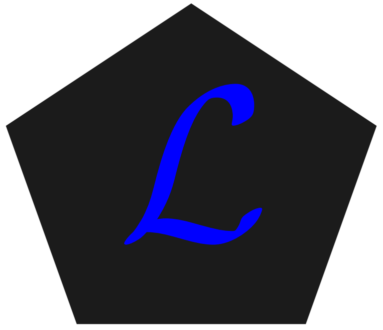
A: Y/N

Observation table

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


$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[loop below, "b"] \arrow[r, bend right, "a"'] & q_1 \arrow[l, bend right, "b"'] \arrow[d, "a"] \\
& & q_2 \arrow[ul, "{a, b}"]
\end{tikzcd}$$

DFA Learning (L*, Angluin'87)



Regular Language

Membership query

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

A: Y/N

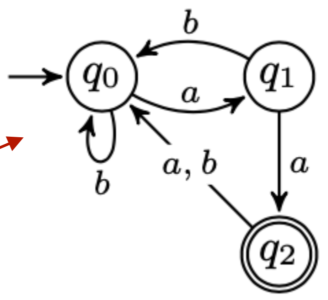
Equivalence query

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

A: Y / N
+ counterexample

Observation table

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



DFA Learning (L*, Angluin’87)

Membership query

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

A: Y/N

Regular Language

Equivalence q

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

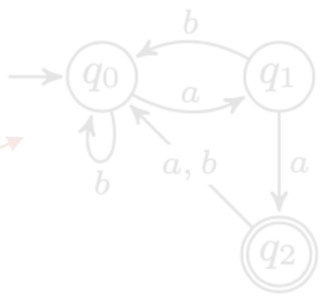
A: Y/N

+ counterexample

Learns the Minimal DFA

Observation table

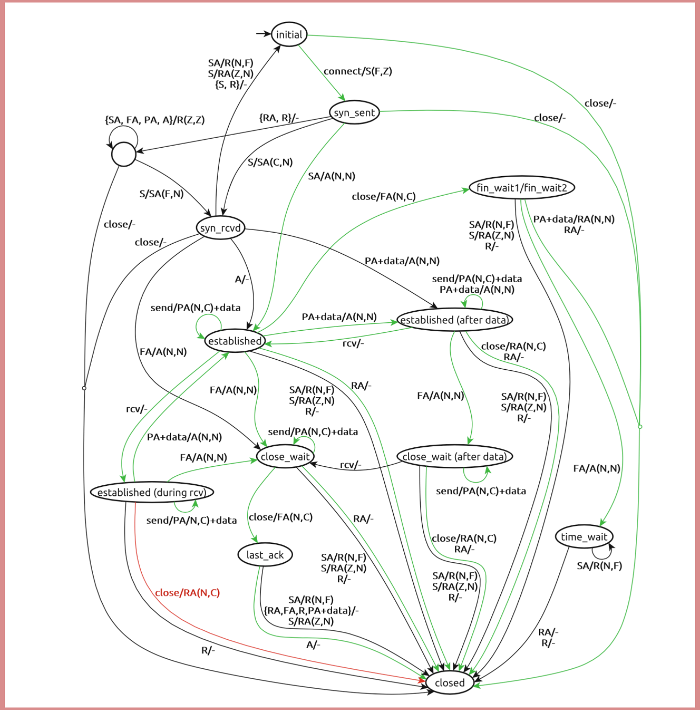
	ϵ	a
ϵ	0	0
a	0	1
aa	1	0
b	0	0
ab	0	0
aaa	0	0
aab	0	0



L* at work

L* at work

Model of Windows 8 TCP implementation

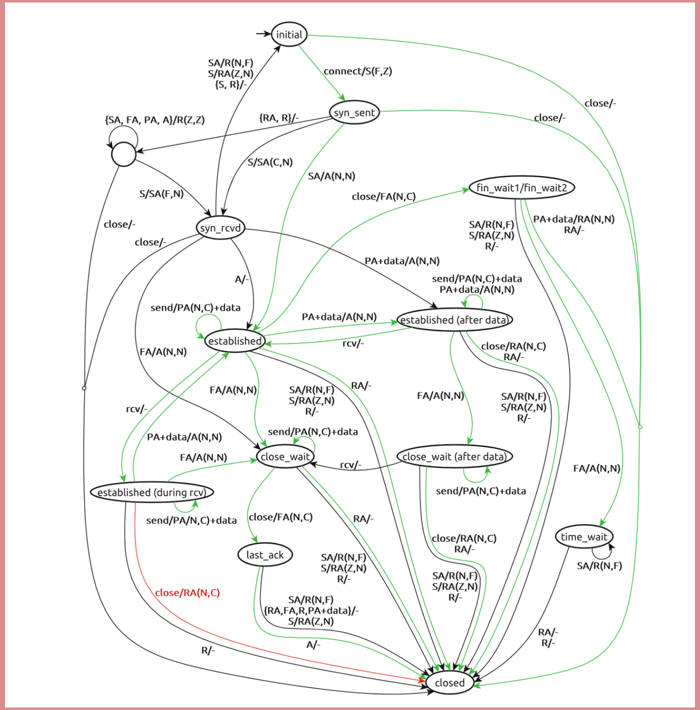


Combining Model Learning and Model Checking to Analyze TCP Implementations

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

L* at work

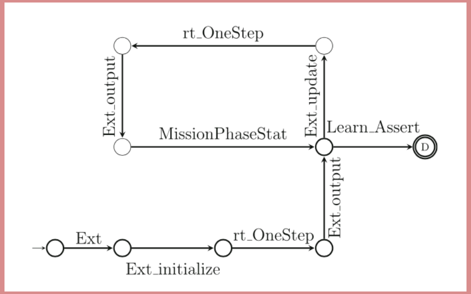
Model of Windows 8 TCP implementation

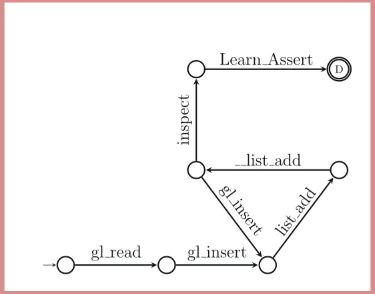


Combining Model Learning and Model Checking to Analyze TCP Implementations

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

Learning Error Traces (function calls)



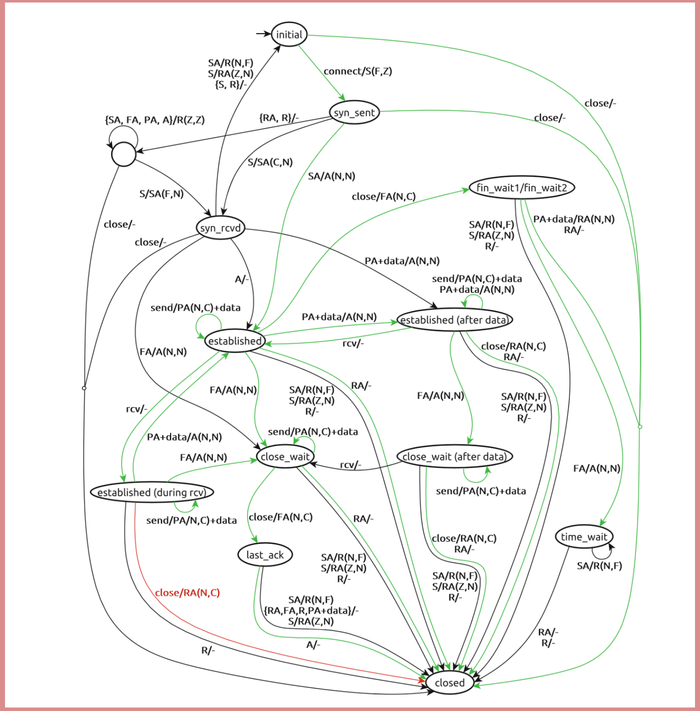


Learning the Language of Error

Martin Chapman[1], Hana Chockler[1](✉), Pascal Kesseli[2], Daniel Kroening[2], Ofer Strichman[3], and Michael Tautschnig[4]

L* at work

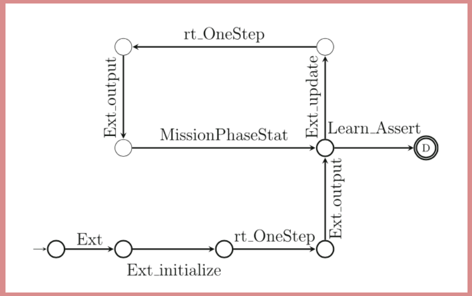
Model of Windows 8 TCP implementation

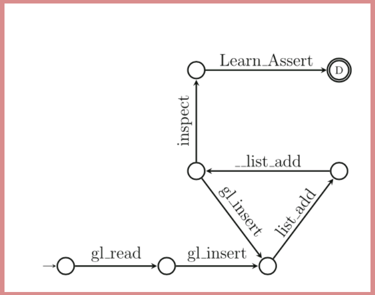


Combining Model Learning and Model Checking to Analyze TCP Implementations

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

Learning Error Traces (function calls)

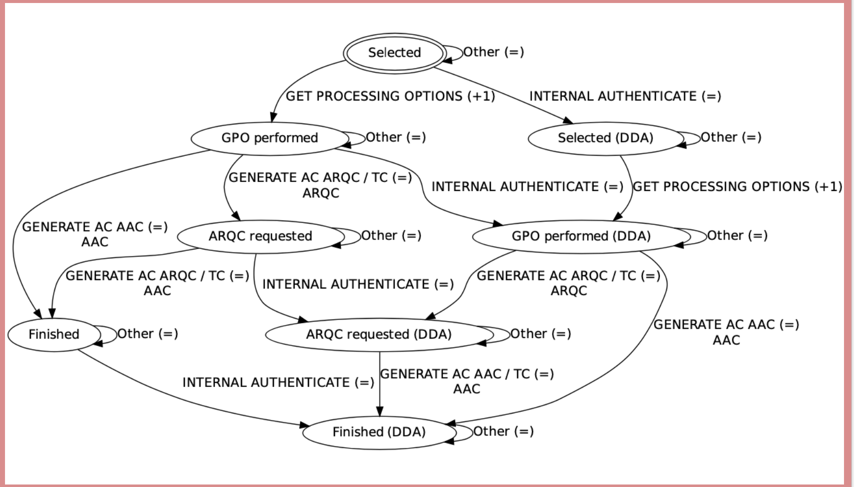




Learning the Language of Error

Martin Chapman[1], Hana Chockler[1](✉), Pascal Kesseli[2], Daniel Kroening[2], Ofer Strichman[3], and Michael Tautschnig[4]

Model of Visa Debit on Barclays card



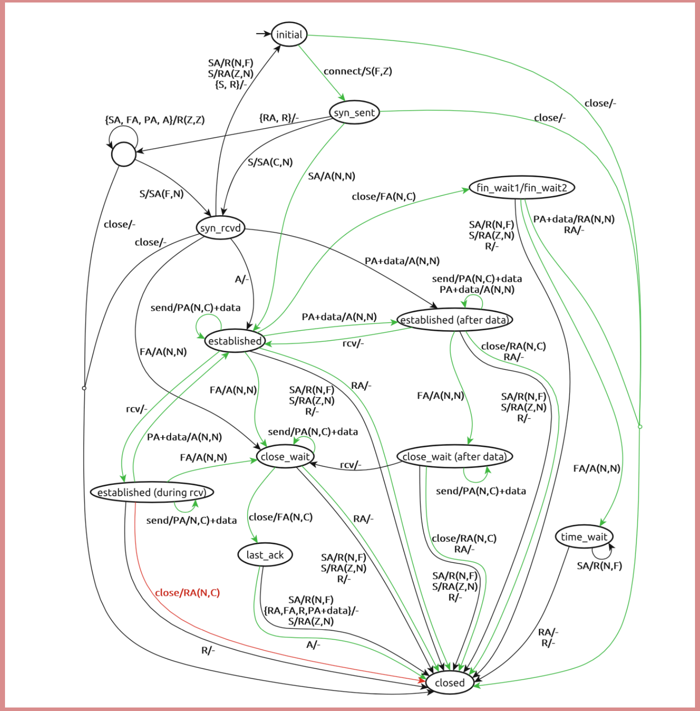
Selected
Other (=)
GET PROCESSING OPTIONS (+1)
INTERNAL AUTHENTICATE (=)
GPO performed
Selected (DDA)
GENERATE AC ARQC / TC (=) ARQC
GET PROCESSING OPTIONS (+1)
GENERATE AC AAC (=) AAC
ARQC requested
GPO performed (DDA)
GENERATE AC ARQC / TC (=) AAC
INTERNAL AUTHENTICATE (=)
GENERATE AC ARQC / TC (=) ARQC
GENERATE AC AAC (=) AAC
Finished
ARQC requested (DDA)
GENERATE AC AAC / TC (=) AAC
Finished (DDA)

Formal models of bank cards for free

Fides Aarts, Joeri de Ruiter, and Erik Poll

L* at work

Model of Windows 8 TCP implementation

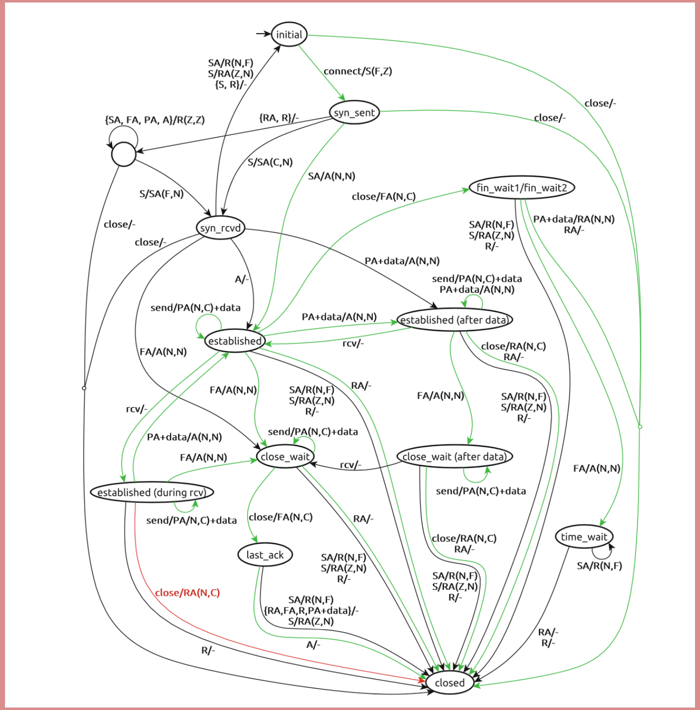


Combining Model Learning and Model Checking to Analyze TCP Implementations

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

L* at work

Model of Windows 8 TCP implementation

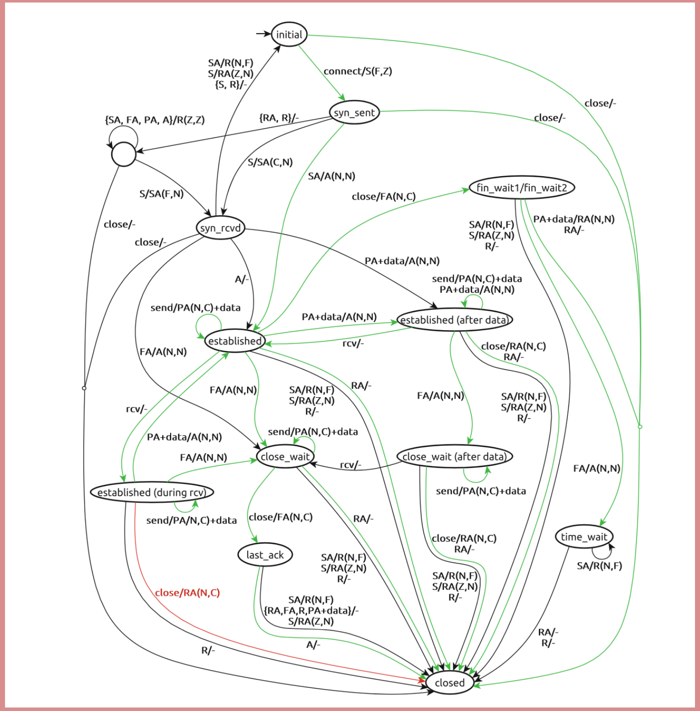


Combining Model Learning and Model Checking to Analyze TCP Implementations

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

L* at work

Model of Windows 8 TCP implementation

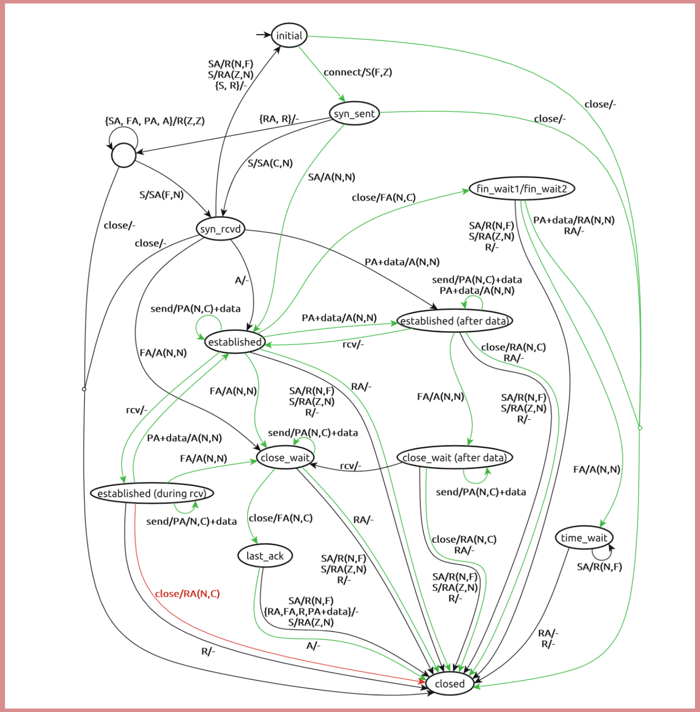


Combining Model Learning and Model Checking to Analyze TCP Implementations

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

L* at work

Model of Windows 8 TCP implementation



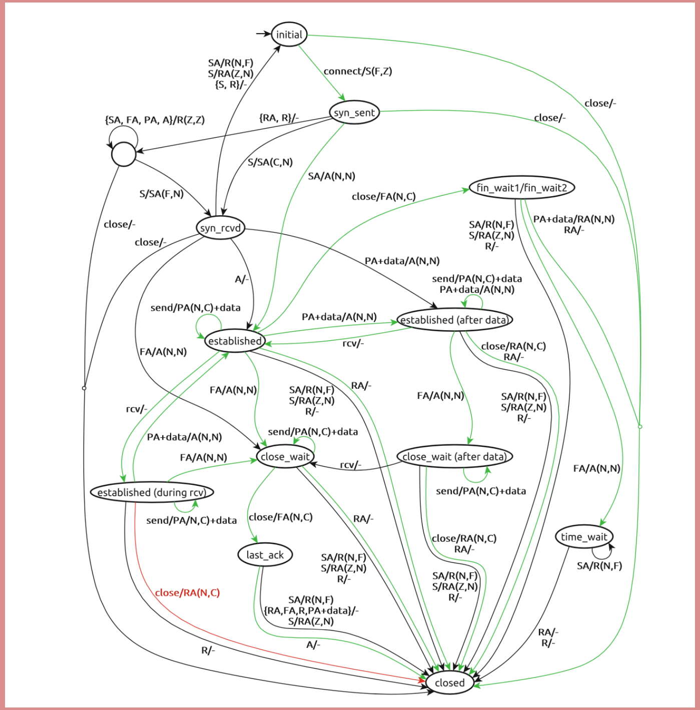
Combining Model Learning and Model Checking to Analyze TCP Implementations

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

Regular language and equivalence queries look like strong assumptions but ok in practice!

L* at work

Model of Windows 8 TCP implementation



Combining Model Learning and Model Checking to Analyze TCP Implementations

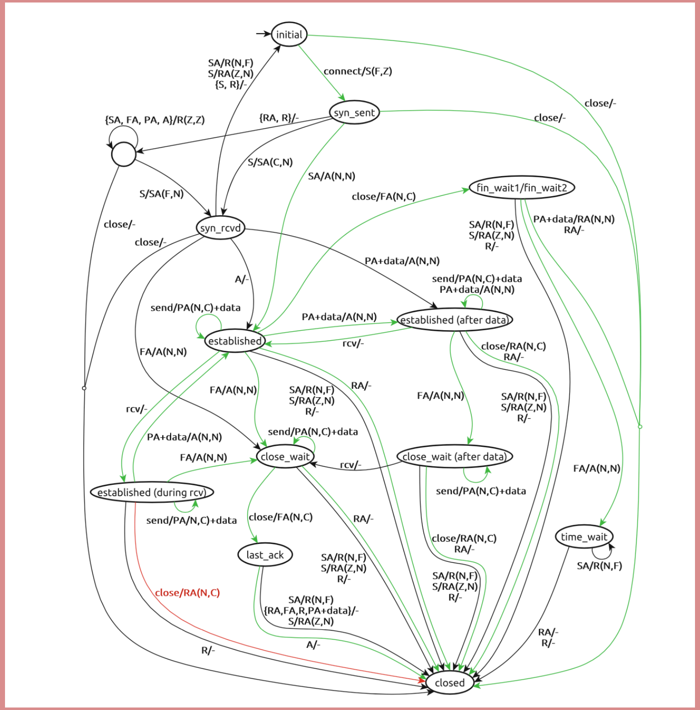
Paul Fiterău-Broştean, Ramon Janssen, and Frits Vaandrager(✉)

Regular language and equivalence queries look like strong assumptions but ok in practice!

Setting up the learning process can be tricky…

L* at work

Model of Windows 8 TCP implementation



Combining Model Learning and Model Checking to Analyze TCP Implementations

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

Regular language and equivalence queries look like strong assumptions but ok in practice!

Setting up the learning process can be tricky…

… domain knowledge required is heavy

Prognosis

T. Ferreira, H. Brewton, L. D’Antoni, AS. ***Prognosis: Closed-Box Analysis of Network Protocol Implementations.*** SIGCOMM’21

Prognosis

T. Ferreira, H. Brewton, L. D'Antoni, AS. ***Prognosis: Closed-Box Analysis of Network Protocol Implementations.*** SIGCOMM'21

Prognosis

Modular & *Reusable* framework: different protocols and protocol implementations

T. Ferreira, H. Brewton, L. D'Antoni, AS. ***Prognosis: Closed-Box Analysis of Network Protocol Implementations.*** SIGCOMM'21

Prognosis

Modular & *Reusable* framework: different protocols and protocol implementations

No need for manually programming the logic of protocol

T. Ferreira, H. Brewton, L. D'Antoni, AS. ***Prognosis: Closed-Box Analysis of Network Protocol Implementations.*** SIGCOMM'21

Prognosis

Modular & *Reusable* framework: different protocols and protocol implementations

No need for manually programming the logic of protocol

Configurable levels of abstraction of learned models and exploration of synthesis to enrich models

T. Ferreira, H. Brewton, L. D'Antoni, AS. ***Prognosis: Closed-Box Analysis of Network Protocol Implementations.*** SIGCOMM'21

Prognosis

Many
implementations!

Prognosis

Many implementations!

Prognosis

Many implementations!

Reference implementation

Prognosis

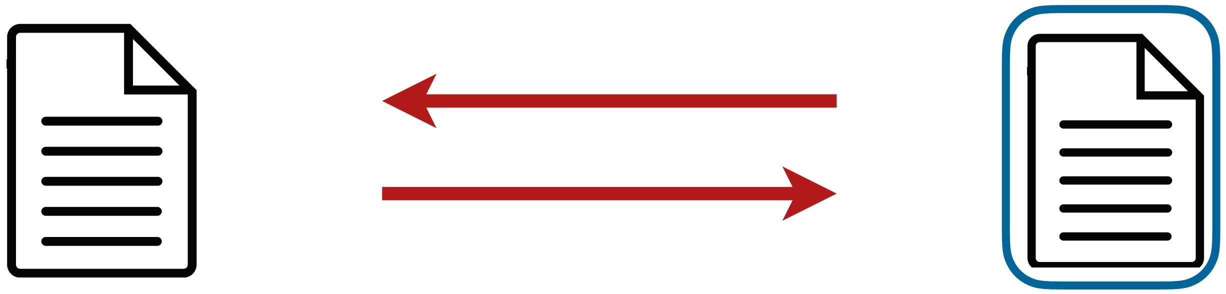
Reference implementation

Prognosis

Analysis implementation

Reference implementation

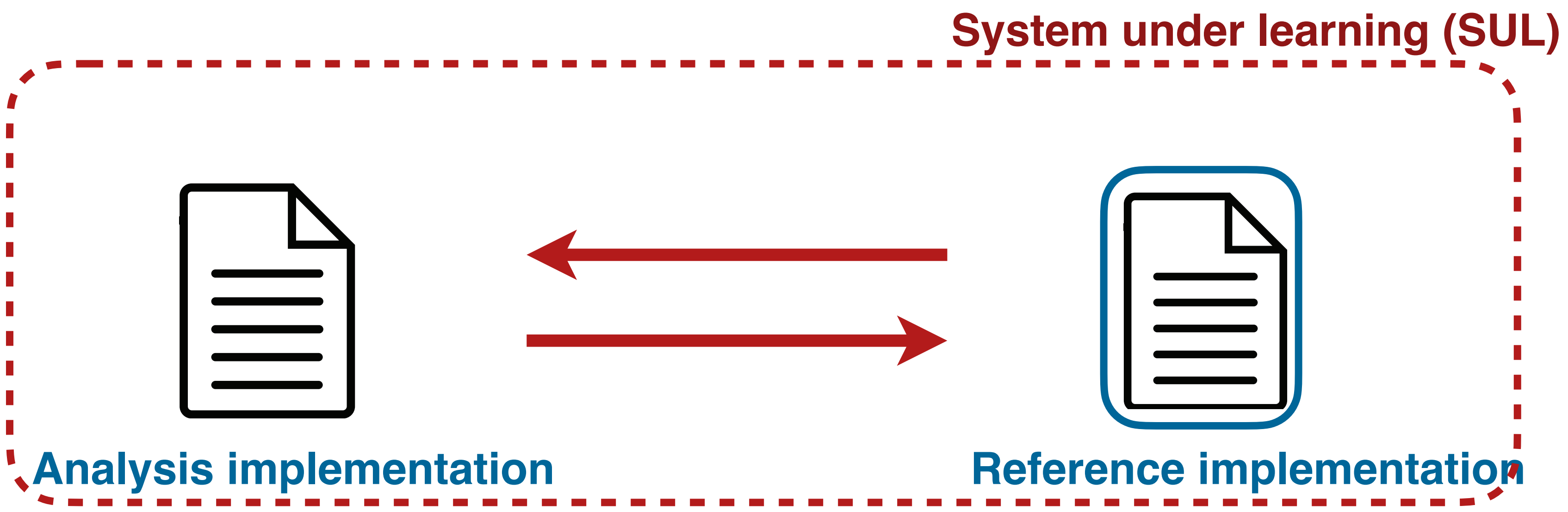
Prognosis



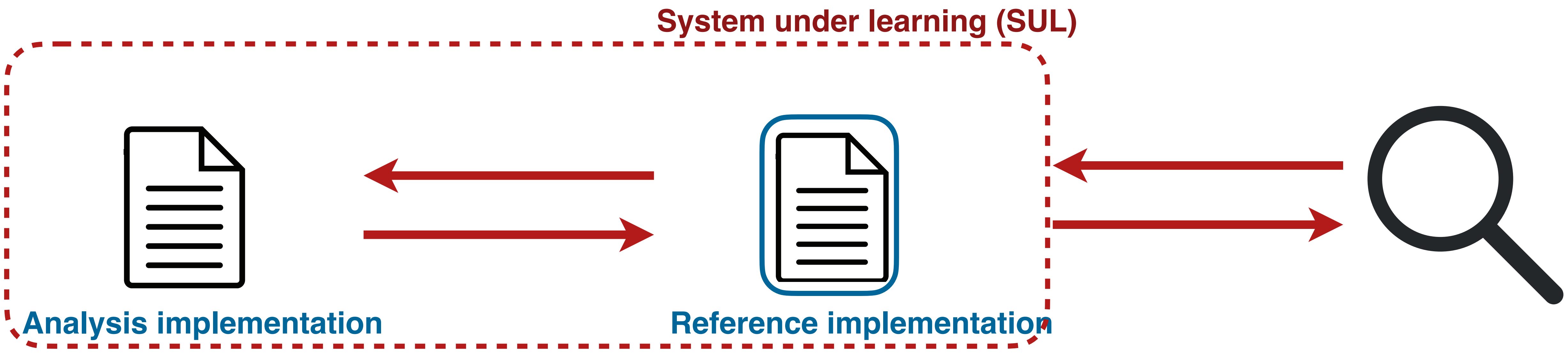
Analysis implementation

Reference implementation

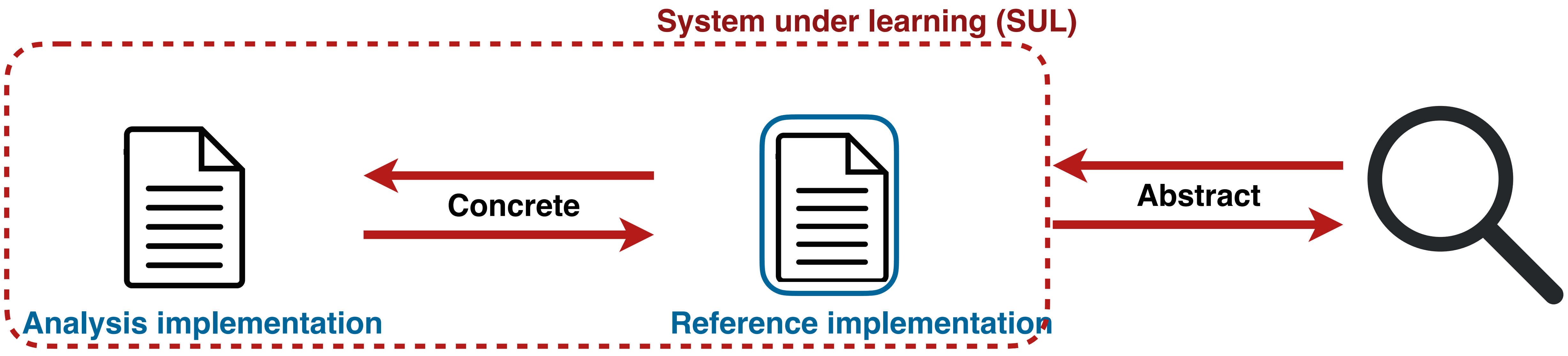
Prognosis


System under learning (SUL)
Analysis implementation
Reference implementation

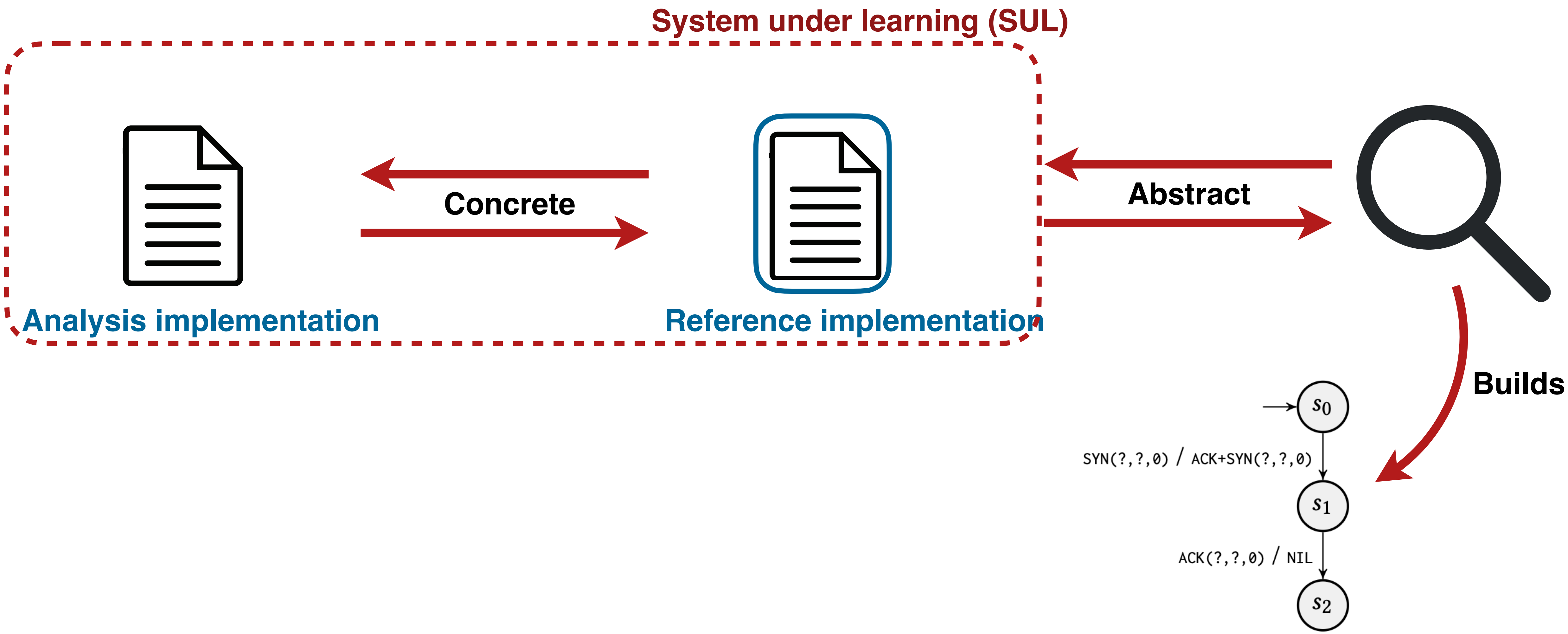
Prognosis


System under learning (SUL)
Analysis implementation
Reference implementation

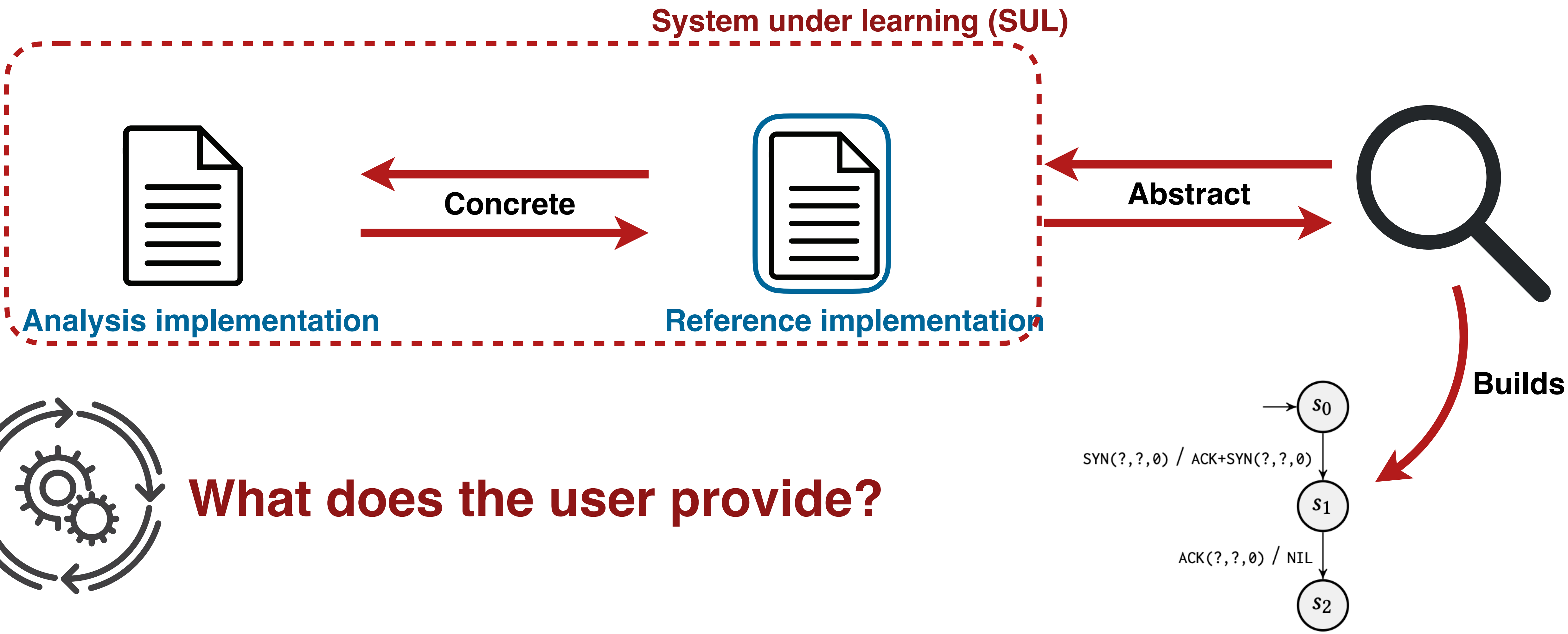
Prognosis


System under learning (SUL)
Concrete
Abstract
Analysis implementation
Reference implementation

Prognosis

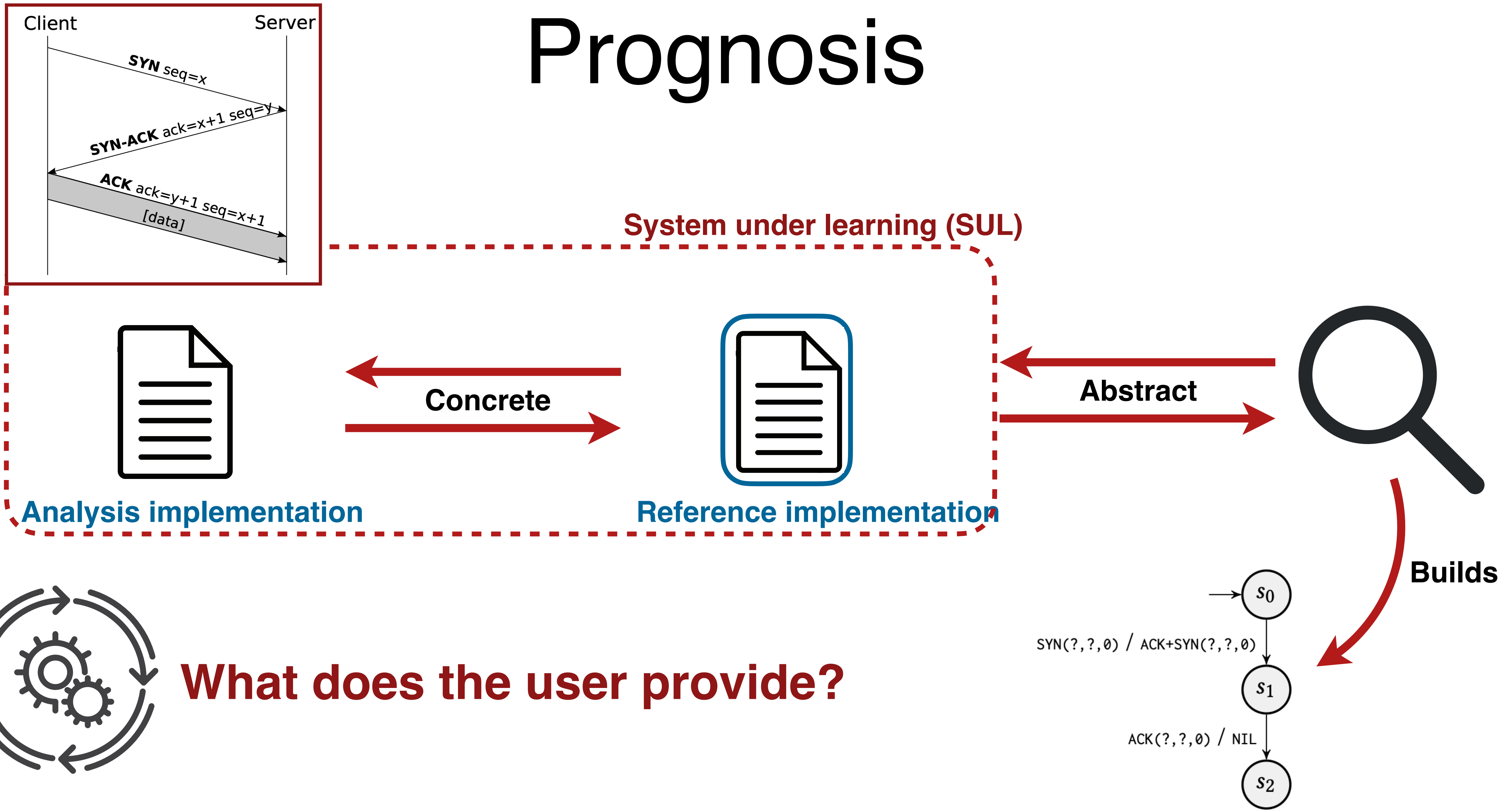

System under learning (SUL)
Concrete
Abstract
Analysis implementation
Reference implementation
Builds
s_0
SYN(?,?,0) / ACK+SYN(?,?,0)
s_1
ACK(?,?,0) / NIL
s_2

Prognosis



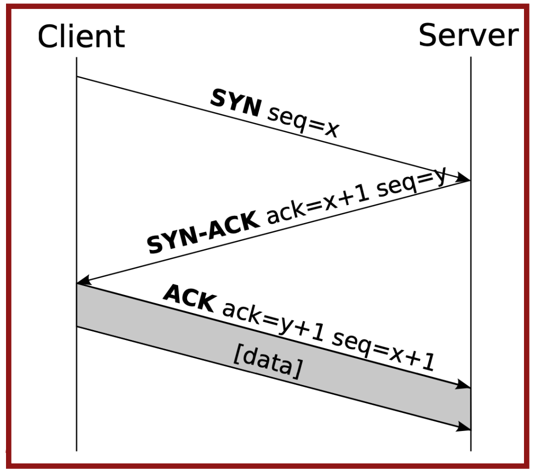
What does the user provide?

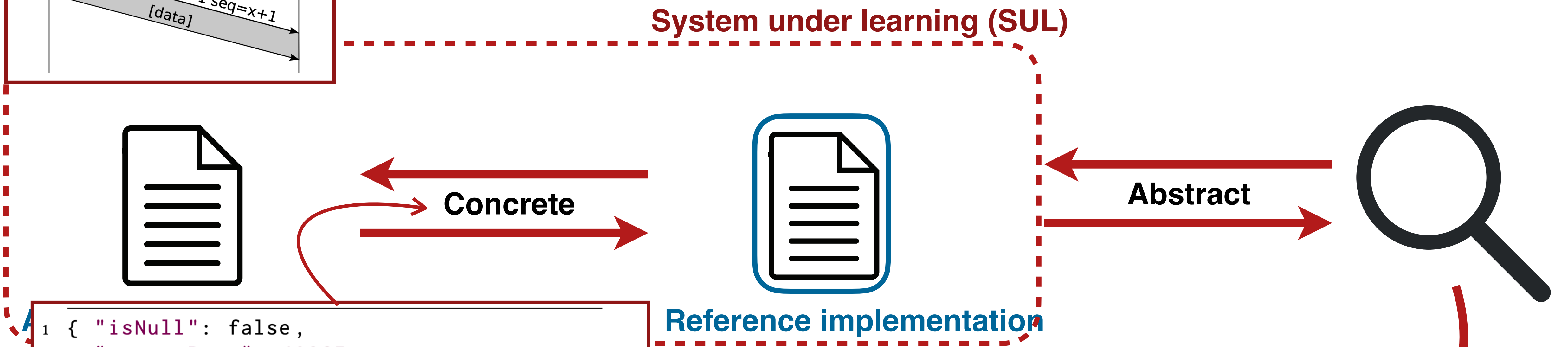
Prognosis

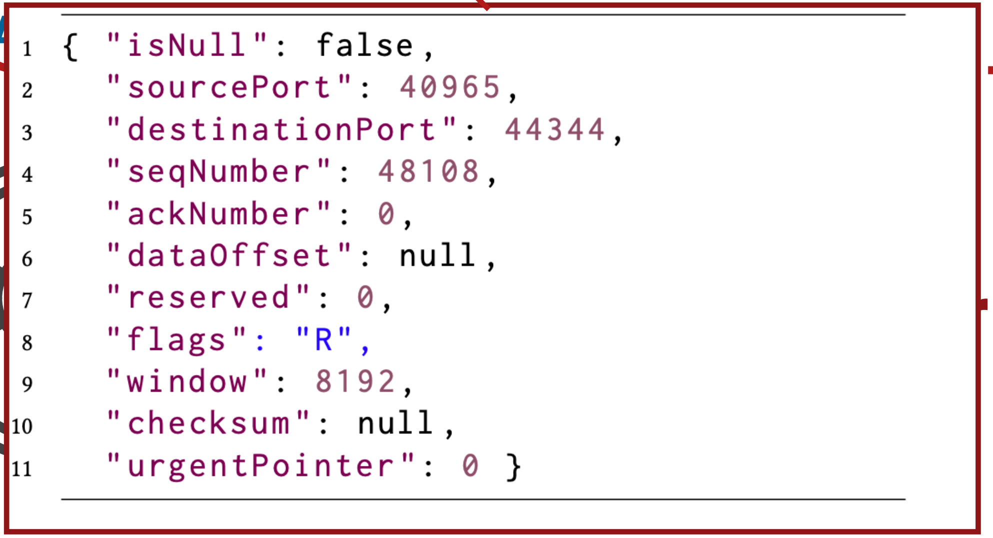


What does the user provide?

Prognosis

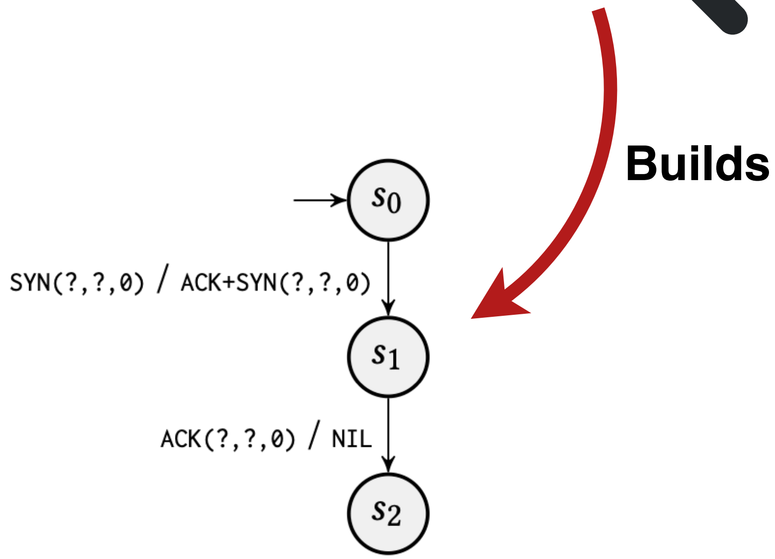




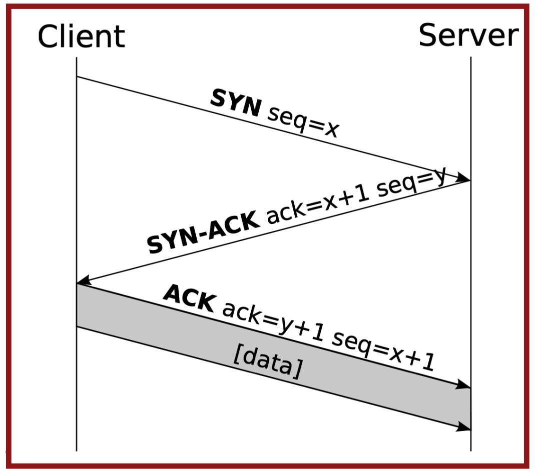


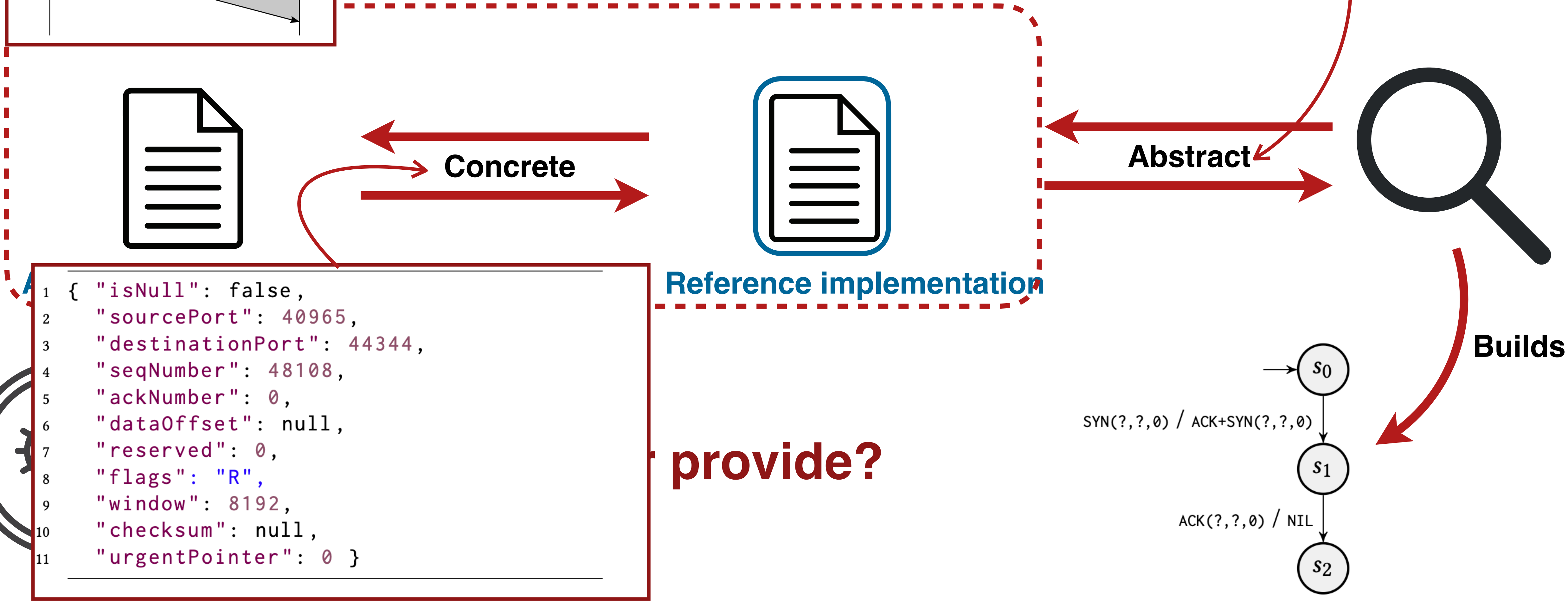
```
{ "isNull": false,
  "sourcePort": 40965,
  "destinationPort": 44344,
  "seqNumber": 48108,
  "ackNumber": 0,
  "dataOffset": null,
  "reserved": 0,
  "flags": "R",
  "window": 8192,
  "checksum": null,
  "urgentPointer": 0 }
```

provide?



Prognosis

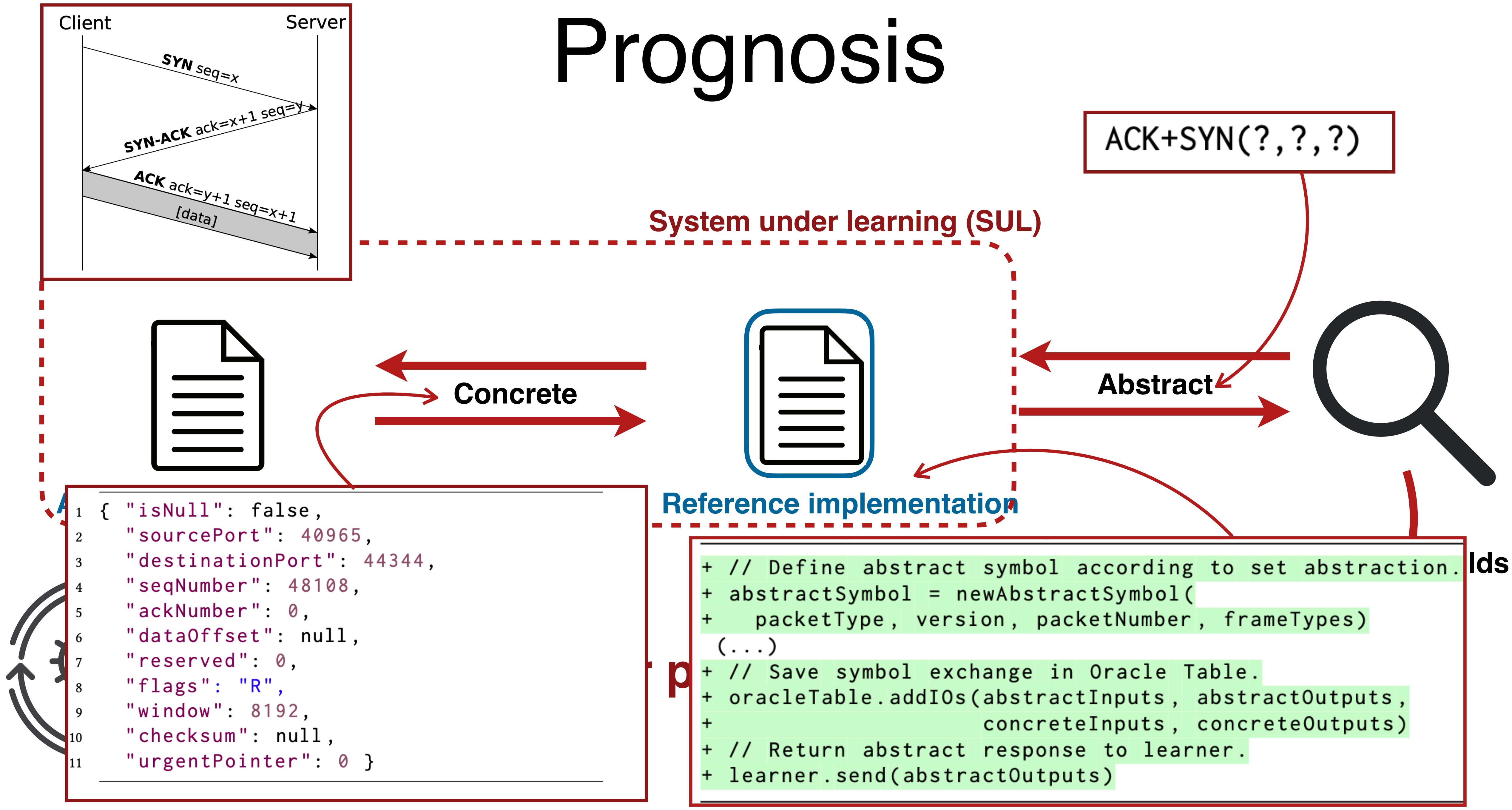




```
{ "isNull": false,
  "sourcePort": 40965,
  "destinationPort": 44344,
  "seqNumber": 48108,
  "ackNumber": 0,
  "dataOffset": null,
  "reserved": 0,
  "flags": "R",
  "window": 8192,
  "checksum": null,
  "urgentPointer": 0 }
```

provide?

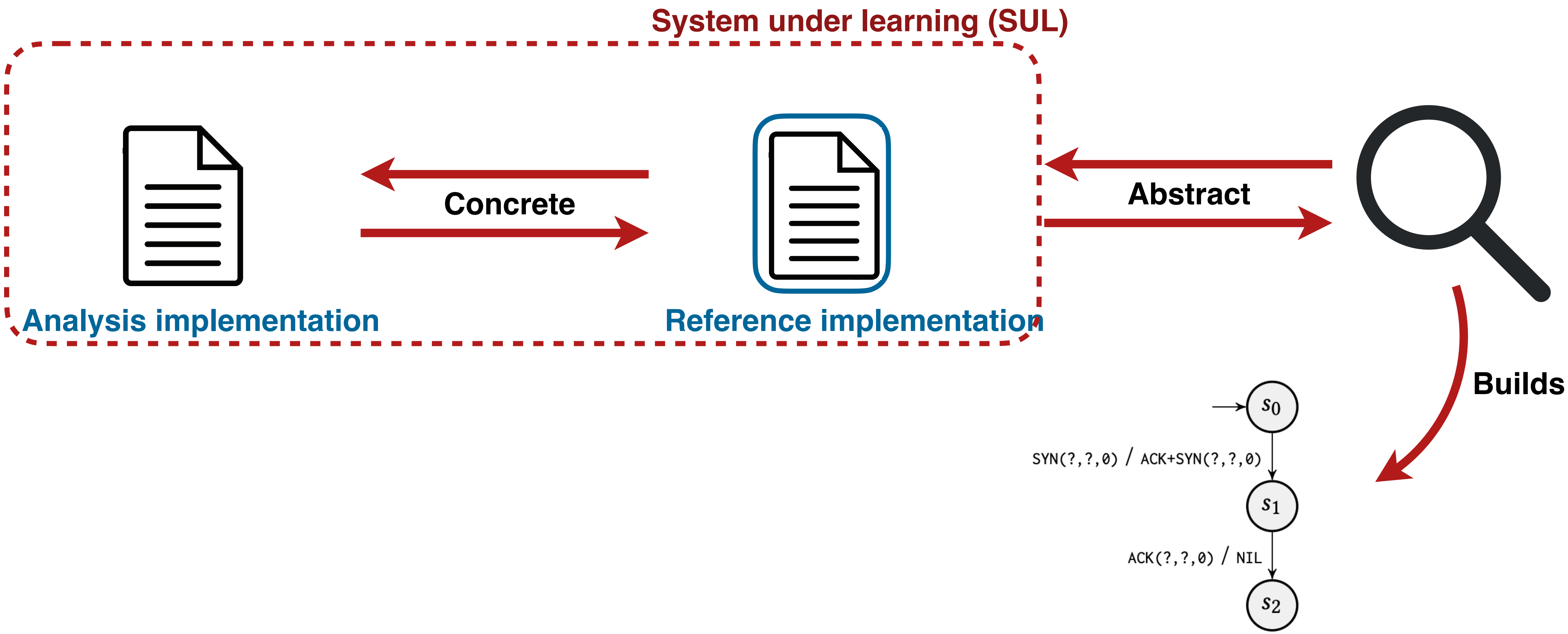
Prognosis



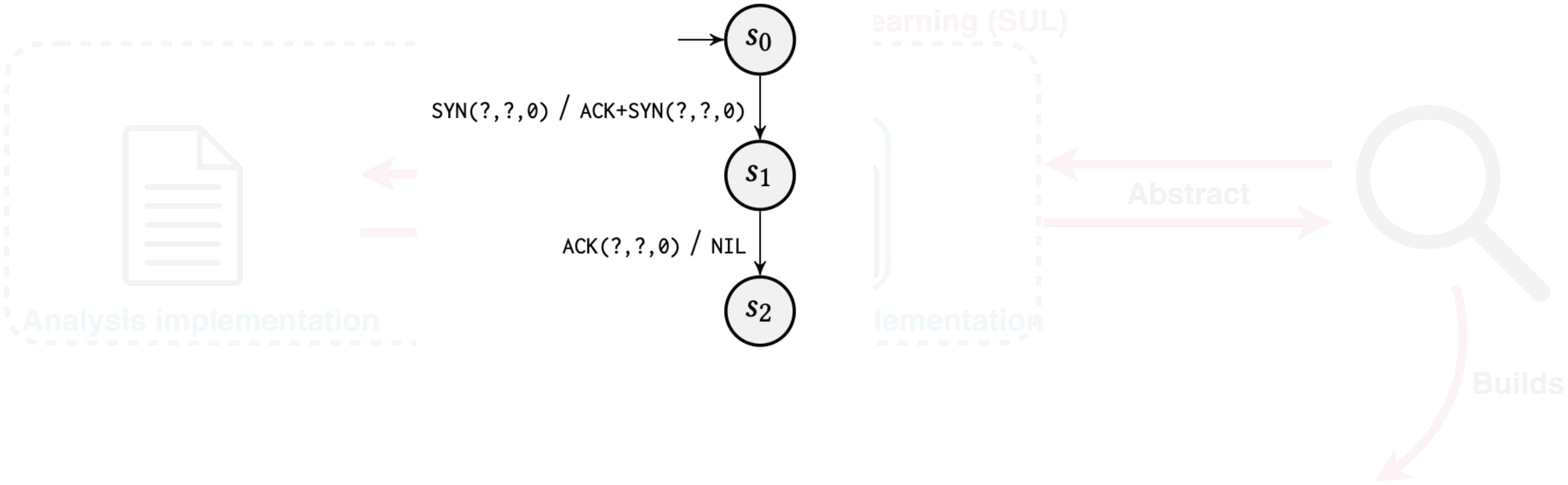
```
{ "isNull": false,
  "sourcePort": 40965,
  "destinationPort": 44344,
  "seqNumber": 48108,
  "ackNumber": 0,
  "dataOffset": null,
  "reserved": 0,
  "flags": "R",
  "window": 8192,
  "checksum": null,
  "urgentPointer": 0 }
```

```
+ // Define abstract symbol according to set abstraction.
+ abstractSymbol = newAbstractSymbol(
+   packetType, version, packetNumber, frameTypes)
 (...)
+ // Save symbol exchange in Oracle Table.
+ oracleTable.addIOs(abstractInputs, abstractOutputs,
+                    concreteInputs, concreteOutputs)
+ // Return abstract response to learner.
+ learner.send(abstractOutputs)
```

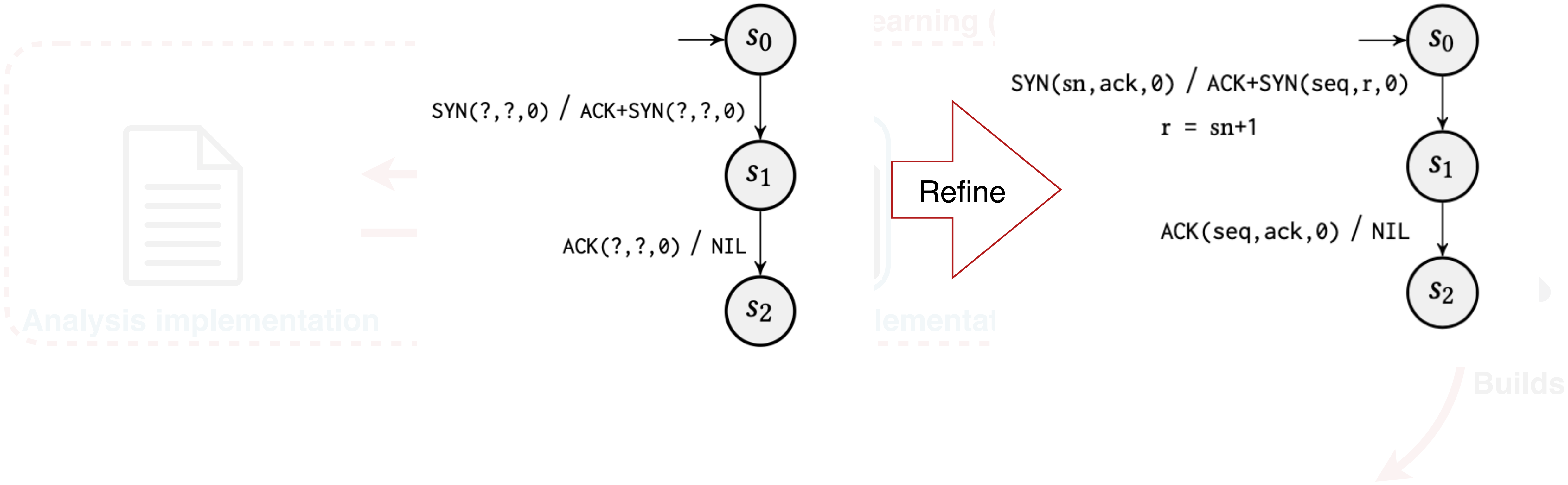
Prognosis


System under learning (SUL)
Concrete
Abstract
Analysis implementation
Reference implementation
Builds
s_0
SYN(?,?,0) / ACK+SYN(?,?,0)
s_1
ACK(?,?,0) / NIL
s_2

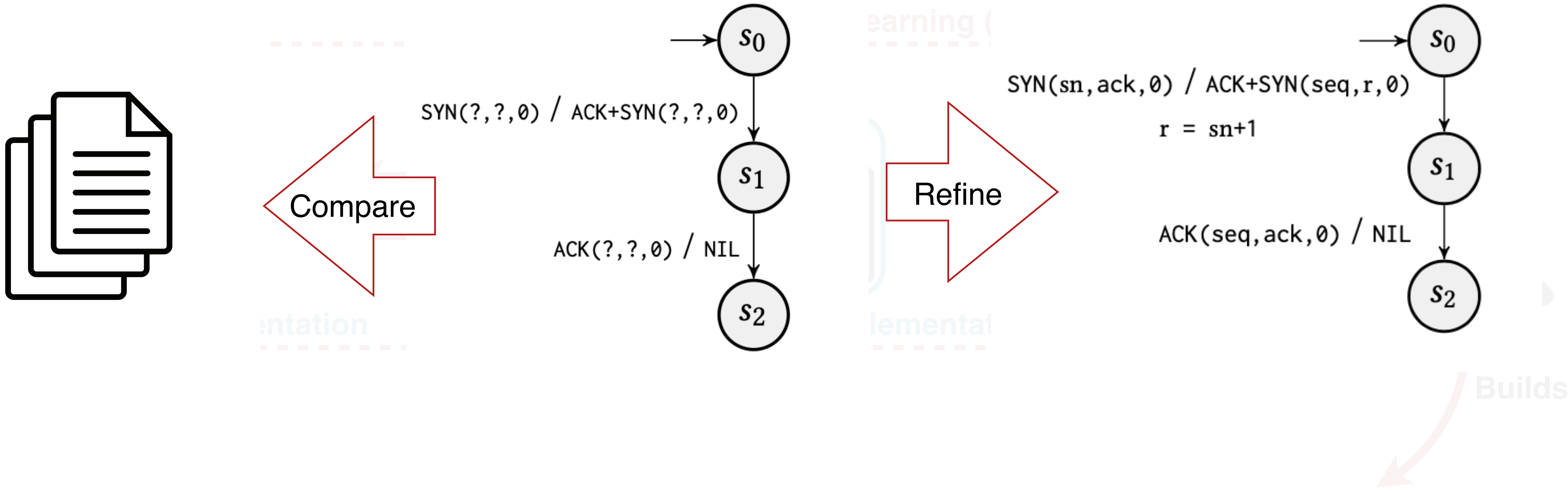
Prognosis


s_0
SYN(?,?,0) / ACK+SYN(?,?,0)
s_1
ACK(?,?,0) / NIL
s_2
earning (SUL)
Analysis implementation
lementation
Abstract
Builds

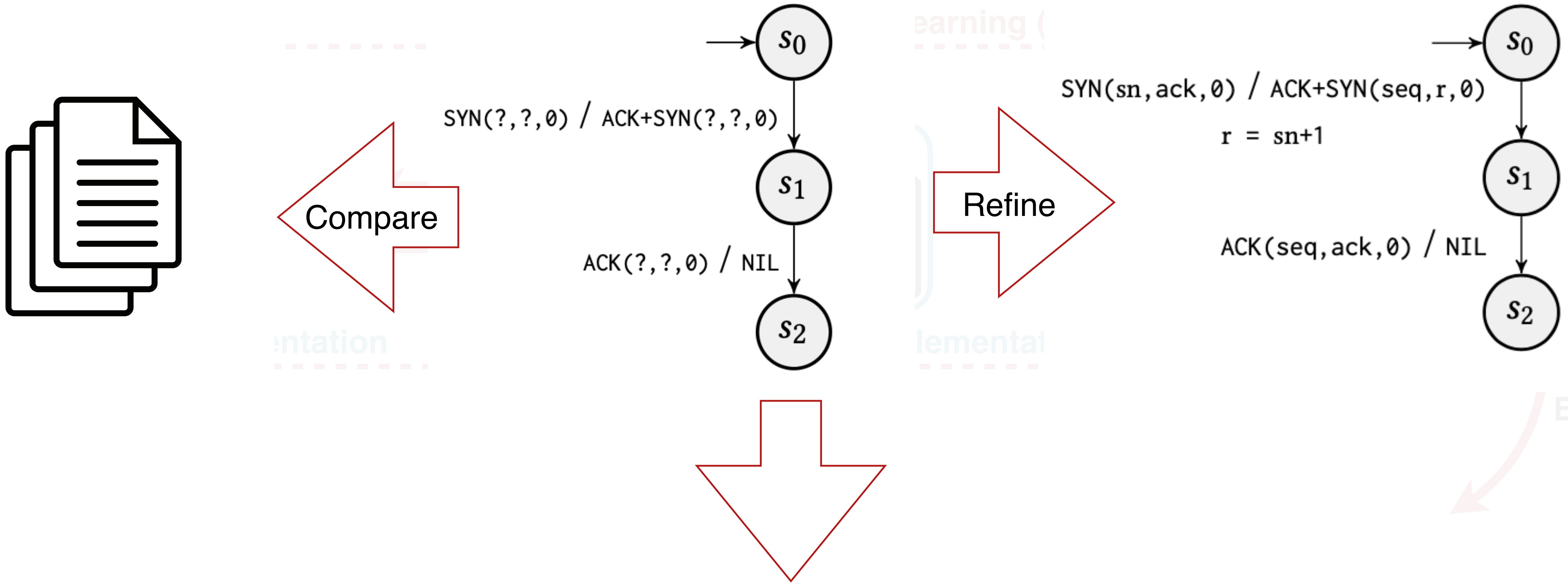
Prognosis


Analysis implementation
earning (
lementat
Builds
s_0
SYN(?,?,0) / ACK+SYN(?,?,0)
s_1
ACK(?,?,0) / NIL
s_2
Refine
s_0
SYN(sn,ack,0) / ACK+SYN(seq,r,0)
r = sn+1
s_1
ACK(seq,ack,0) / NIL
s_2

Prognosis


Compare
SYN(?,?,0) / ACK+SYN(?,?,0)
ACK(?,?,0) / NIL
s_0
s_1
s_2
Refine
SYN(sn,ack,0) / ACK+SYN(seq,r,0)
r = sn+1
ACK(seq,ack,0) / NIL
s_0
s_1
s_2

Prognosis



Your favourite analysis tool!

Prognosis in practice

TCP

QUI

Prognosis in practice

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-Broştean but 300 vs 3000 lines of extra code!

QUI

Prognosis in practice

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-Broştean but 300 vs 3000 lines of extra code!

QUI

4 implementations: Cloudflare, Google, Facebook, Quick-tracker

Model sizes varied (8-12 states an 56-84 transitions); ~10-25K membership queries to learn

Discovered 3 bugs and filed an RFC change (2000 lines annotations)

Prognosis in practice

TCP

Ubuntu 20.04.1 LT
stack, kernel versi
generic

6 states and 42 transitions; 4,726 membership queries to learn

Replicated results of Fiterău-Broştean but 300 vs 3000 lines of extra code!

QUI

tions: Cloudflare,
book, Quick-tracker

Model sizes varied (8-12 states an
56-84 transitions); ~10-25K membership queries to learn

Discovered 3 bugs and filed an RFC change (2000 lines annotations)

Prognosis in practice

TCP

Ubuntu 20.04.1 LT

stack, kernel versi

generic

6 states and 42 transitions;

4,726 membership queries to learn

Replicated results of Fiterău-Broştean but 300 vs 3000 lines of extra code!

QUI

tions: Cloudflare,

book, Quick-tracker

Model sizes varied (6 - 12 states an

56-84 transitions); ~10-25K membership queries to learn

Discovered 3 bugs and filed an RFC change (2000 lines annotations)

Finding bugs does not require a complete model — zooming the analysis in specific parts of the code is fruitful!

Prognosis in practice

TCP

Ubuntu 20.04.1 LT
stack, kernel versi
generic

6 states and 42 transitions;
4,726 membership queries to learn

Replicated results of Fiterău-Broştean but 300 vs 3000 lines of extra code!

QUI

tions: Cloudflare,
book, Quick-tracker

Model sizes varied (5-12 states and 56-84 transitions); ~10-25K membership queries to learn

Discovered 3 bugs and filed an RFC change (2000 lines annotations)

Finding bugs does not require a complete model — zooming the analysis in specific parts of the code is fruitful!

Ability to produce concrete traces key for communication with developers

Prognosis in practice

TCP

Ubuntu 20.04.1 LT
stack, kernel versi
generic

6 states and 42 transitions;

4,726 membership queries to learn

Replicated results of Fiterău-Broştean but 300 vs 3000 lines of extra code!

QUI

tions: Cloudflare,
book, Quick-tracker

Model sizes varied (5-12 states an
56-84 transitions); ~10-25K
membership queries to learn

Discovered 3 bugs and filed an
RFC change (2000 lines
annotations)

Finding bugs does not require a complete model — zooming the analysis in specific parts of the code is fruitful!

Ability to produce concrete traces key for communication with developers

Design choices give useful insights on potential bugs and underspecified RFC

Other approaches

TCP

QUI

Ubuntu 20.04.1 LTS TCP

stack, kernel version 5.8.0-40-

4 implementations: Cloudflare,

S.Bishop, M.Fairbairn, H.Mehnert, M. Norrish, T. Ridge, P. Sewell, M. Smith, and K. Wansbrough. *Engineering with Logic: Rigorous Test-Oracle Specification and Validation for TCP/IP and the Sockets API*. J. ACM 66
1 (2019)

4,726 membership queries to

50-64 transitions), ~10-25K

membership queries to learn

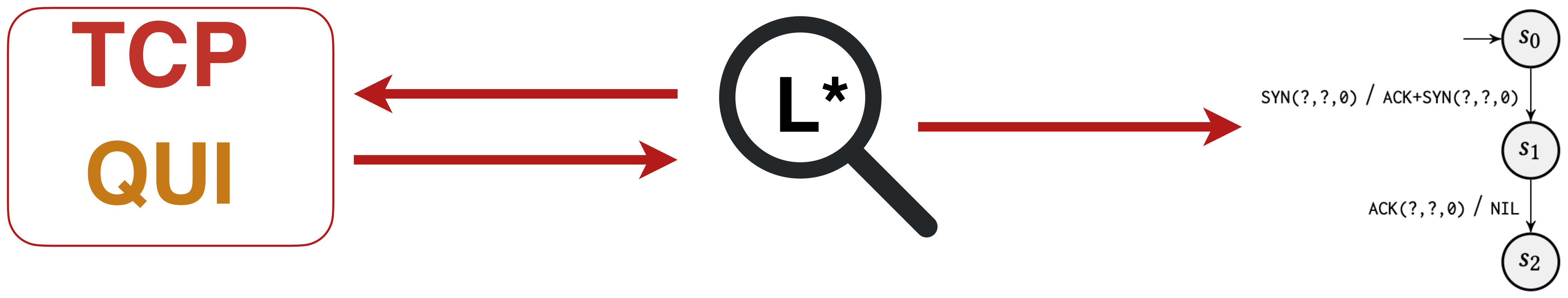
Kenneth L. McMillan and Lenore D. Zuck. *Formal specification and testing of QUIC*. SIGCOMM '19

Build formal specification of the protocol, and then use it for test generation to find bugs

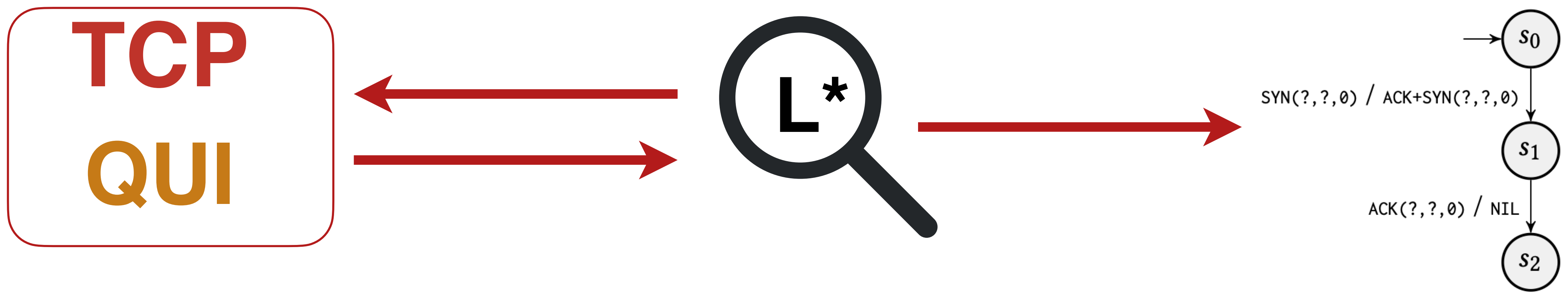
Replicated results of Fiterău-Broştean but 300 vs 3000 lines of extra code!

Discovered 3 bugs and filed an RFC change (2000 lines annotations)

Active Learning in Network Protocols

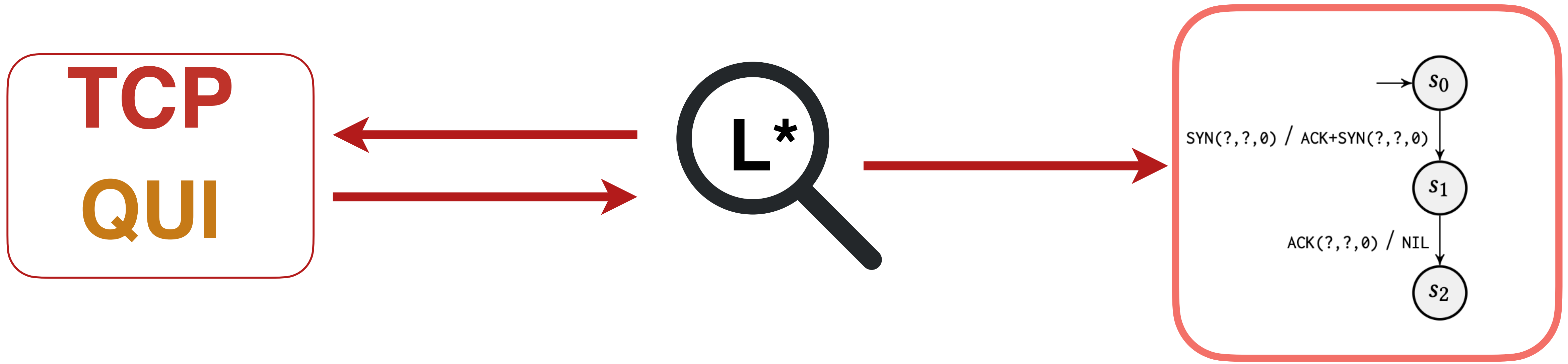


Active Learning in Network Protocols



Larger fragments?
Other protocols?
Quantitative analysis?

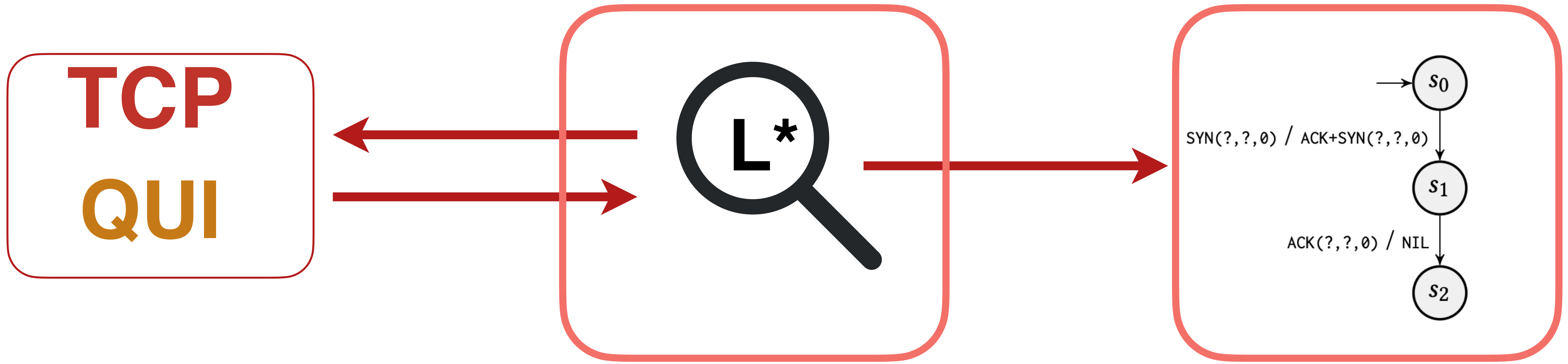
Active Learning in Network Protocols



Richer automata models

Larger fragments?
Other protocols?
Quantitative analysis?

Active Learning in Network Protocols



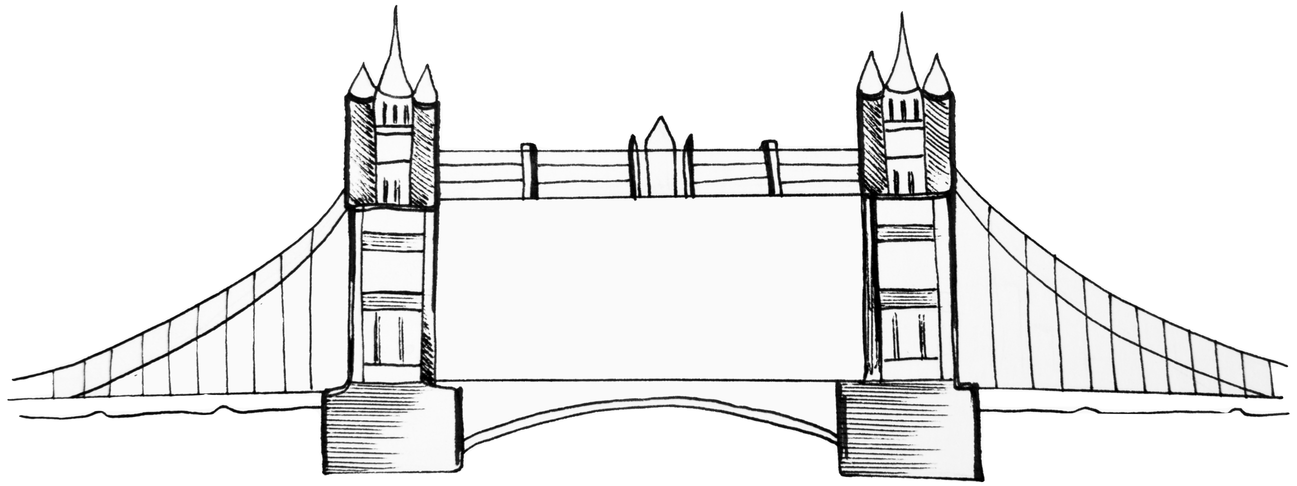
Richer learning algorithms

Richer automata models

Larger fragments?
Other protocols?
Quantitative analysis?

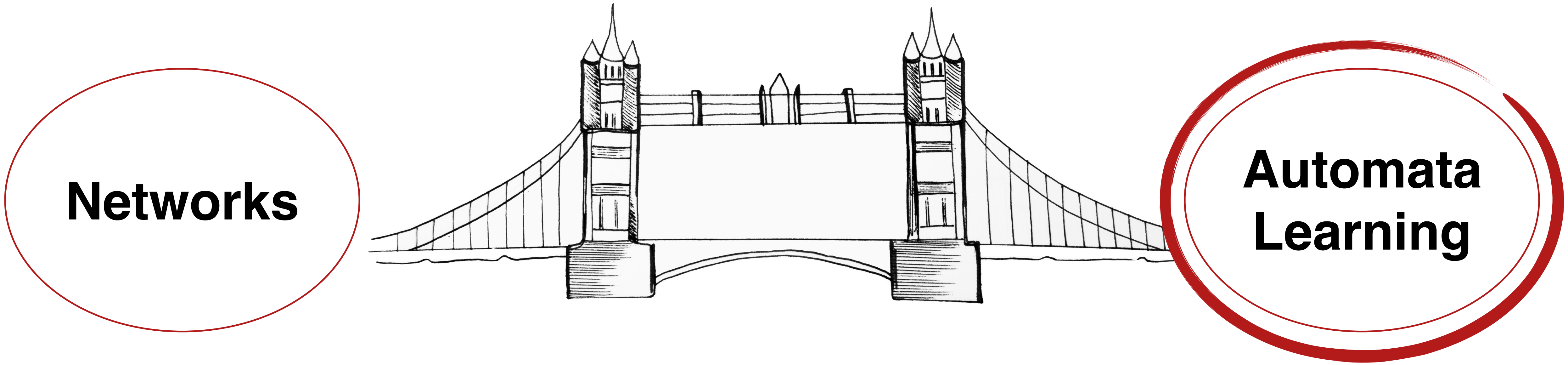
Active Learning in Network Protocols

Networks

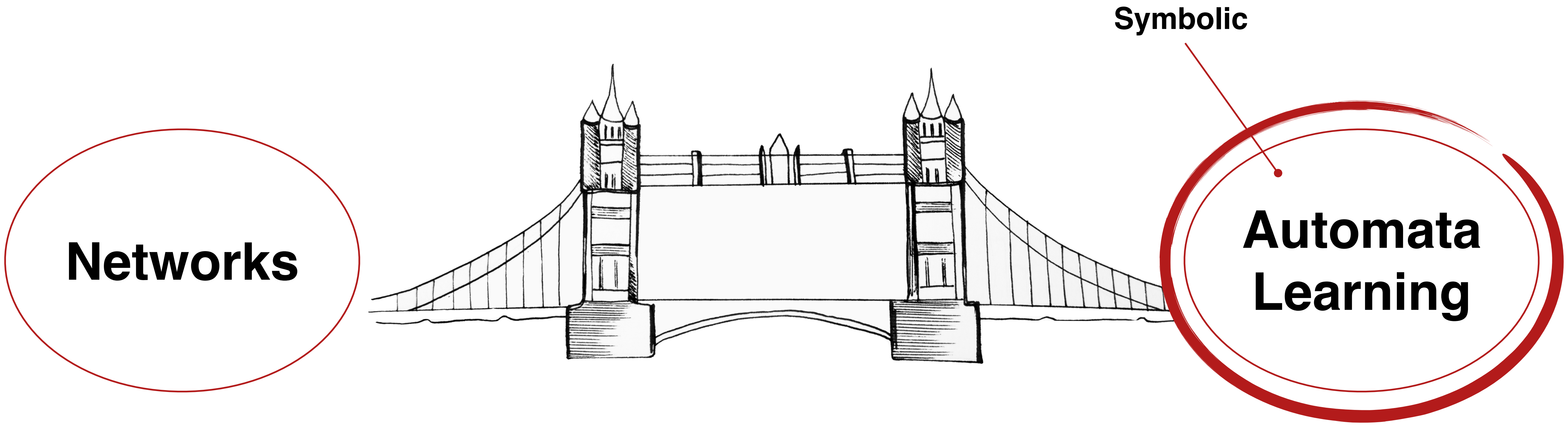


Automata Learning

Active Learning in Network Protocols

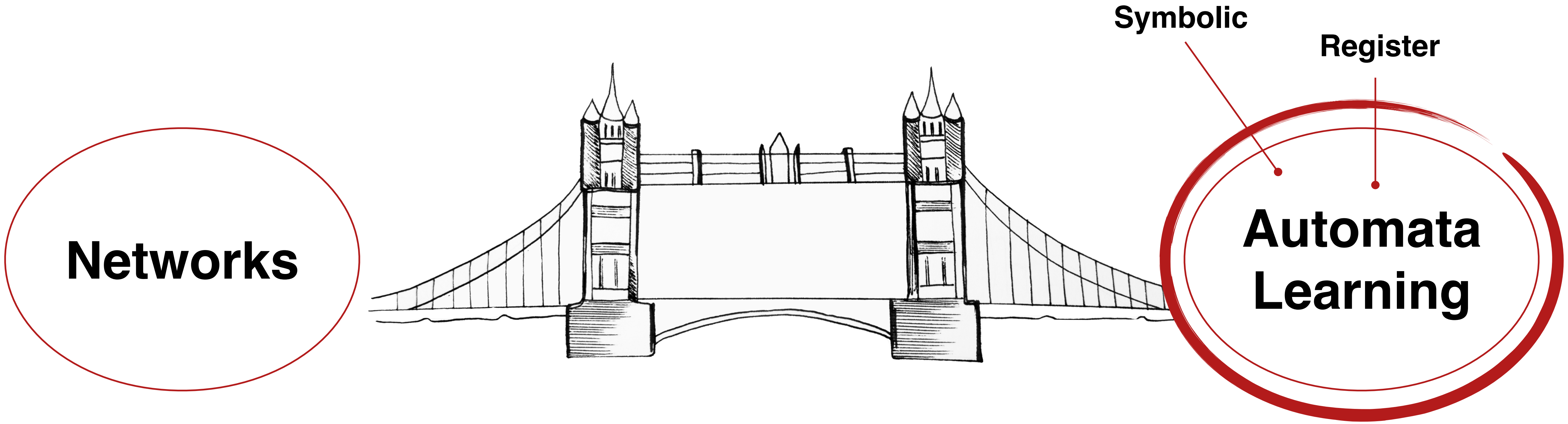


Active Learning in Network Protocols



S Drews, L D'Antoni:
Learning Symbolic Automata. **TACAS** (1) 2017: 173-189

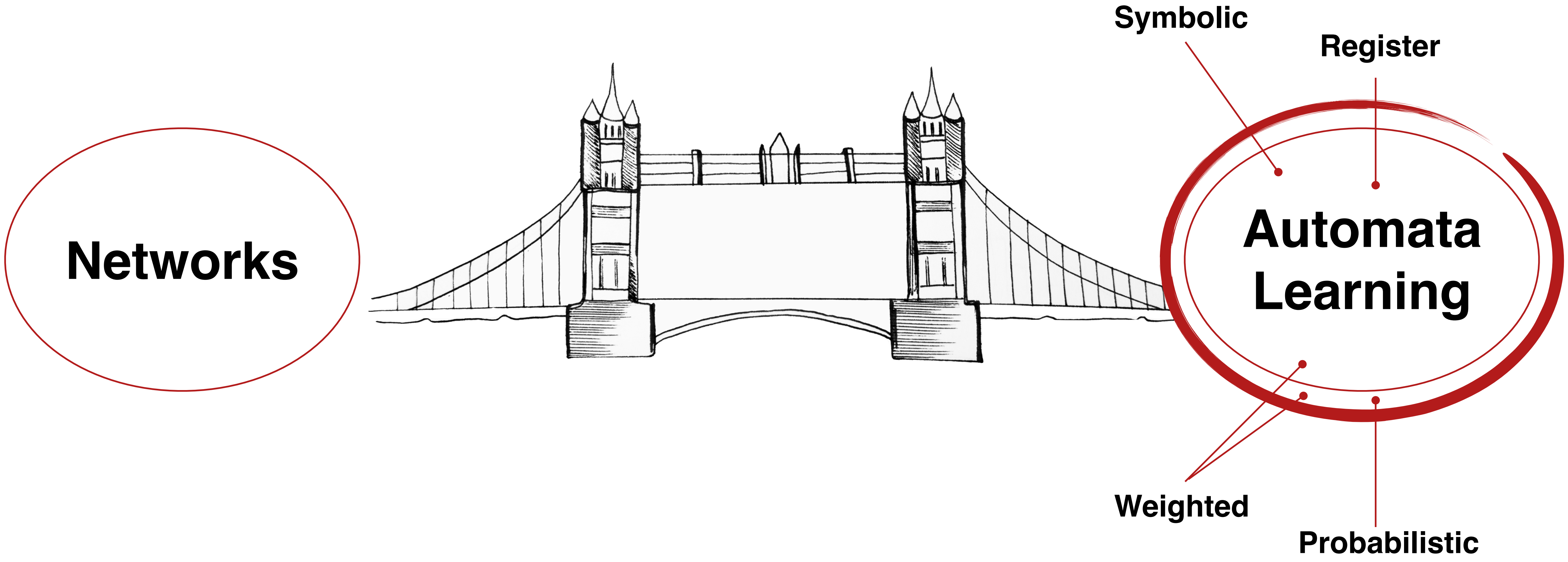
Active Learning in Network Protocols



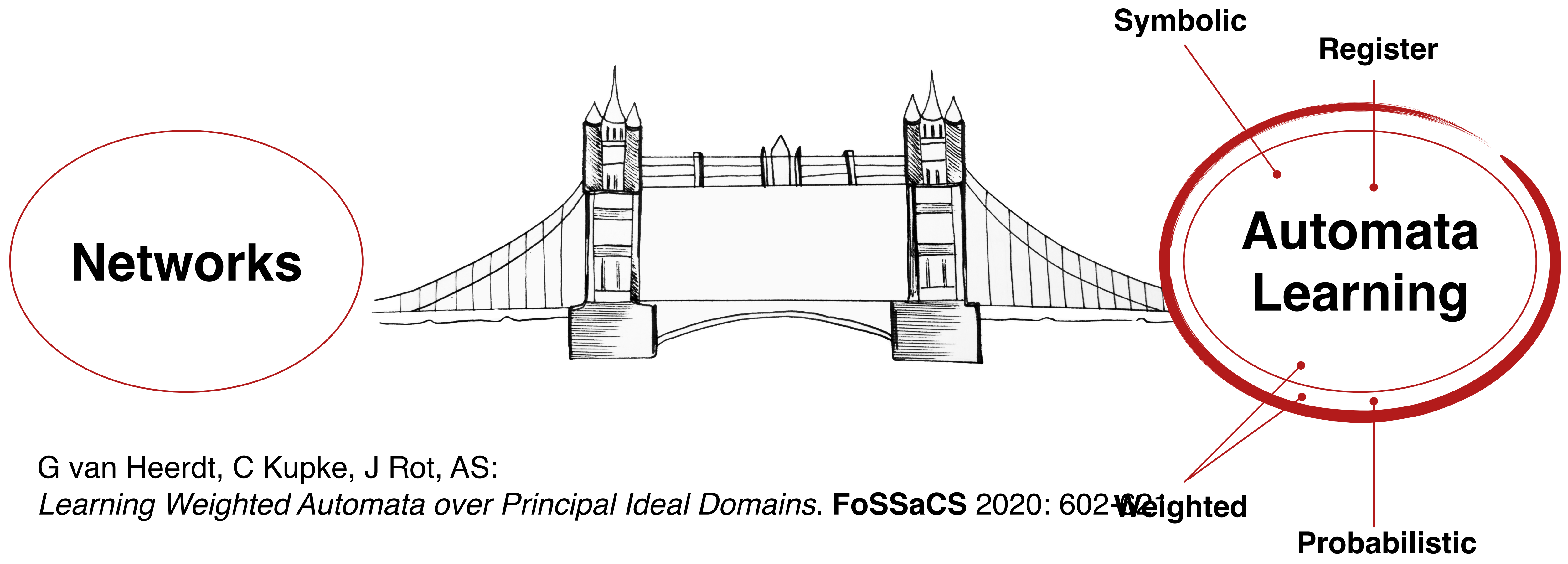
S Drews, L D'Antoni:
Learning Symbolic Automata. **TACAS** (1) 2017: 173-189

M Merten, F Howar, B Steffen, S Cassel, B Jonsson:
Demonstrating Learning of Register Automata. **TACAS** 2012: 466-471

Active Learning in Network Protocols

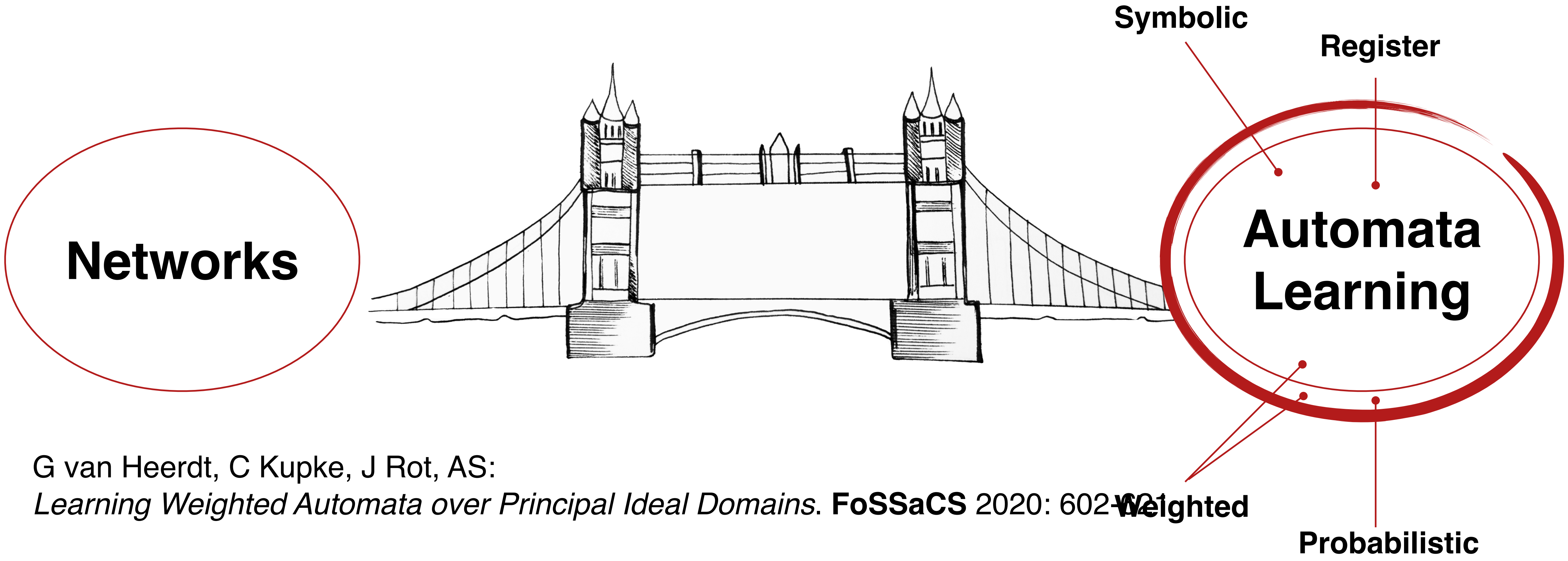


Active Learning in Network Protocols



G van Heerdt, C Kupke, J Rot, AS:
Learning Weighted Automata over Principal Ideal Domains. **FoSSaCS** 2020: 602-621

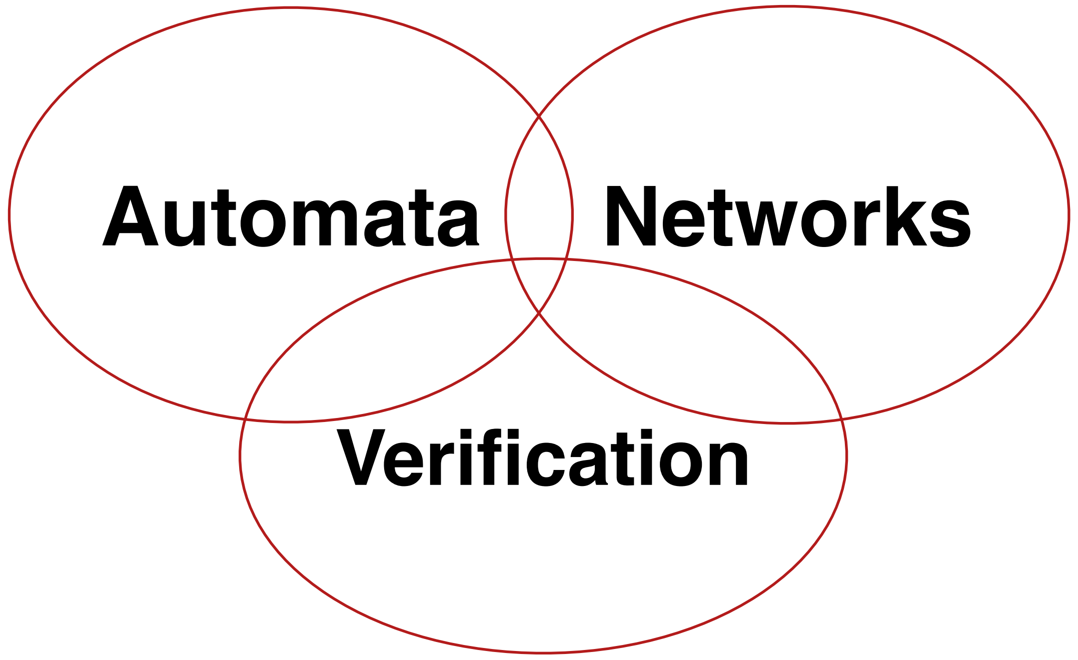
Active Learning in Network Protocols



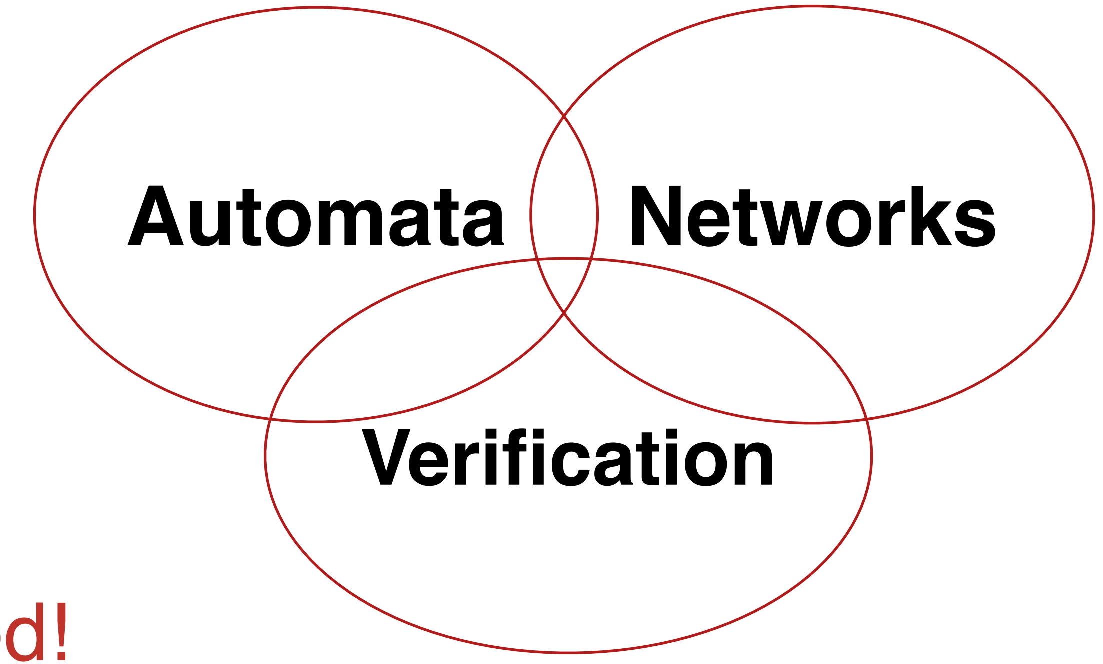
G van Heerdt, C Kupke, J Rot, AS:
Learning Weighted Automata over Principal Ideal Domains. **FoSSaCS** 2020: 602-621

Little known about L* for probabilistic automata!

Conclusion

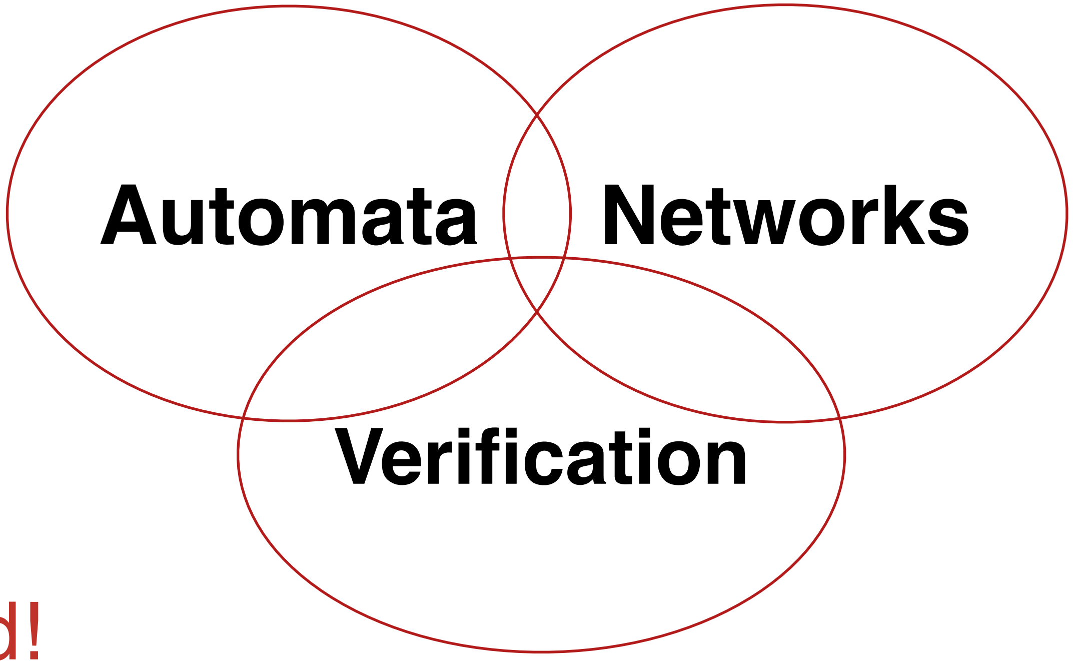


Conclusion



Simple is good!

Conclusion

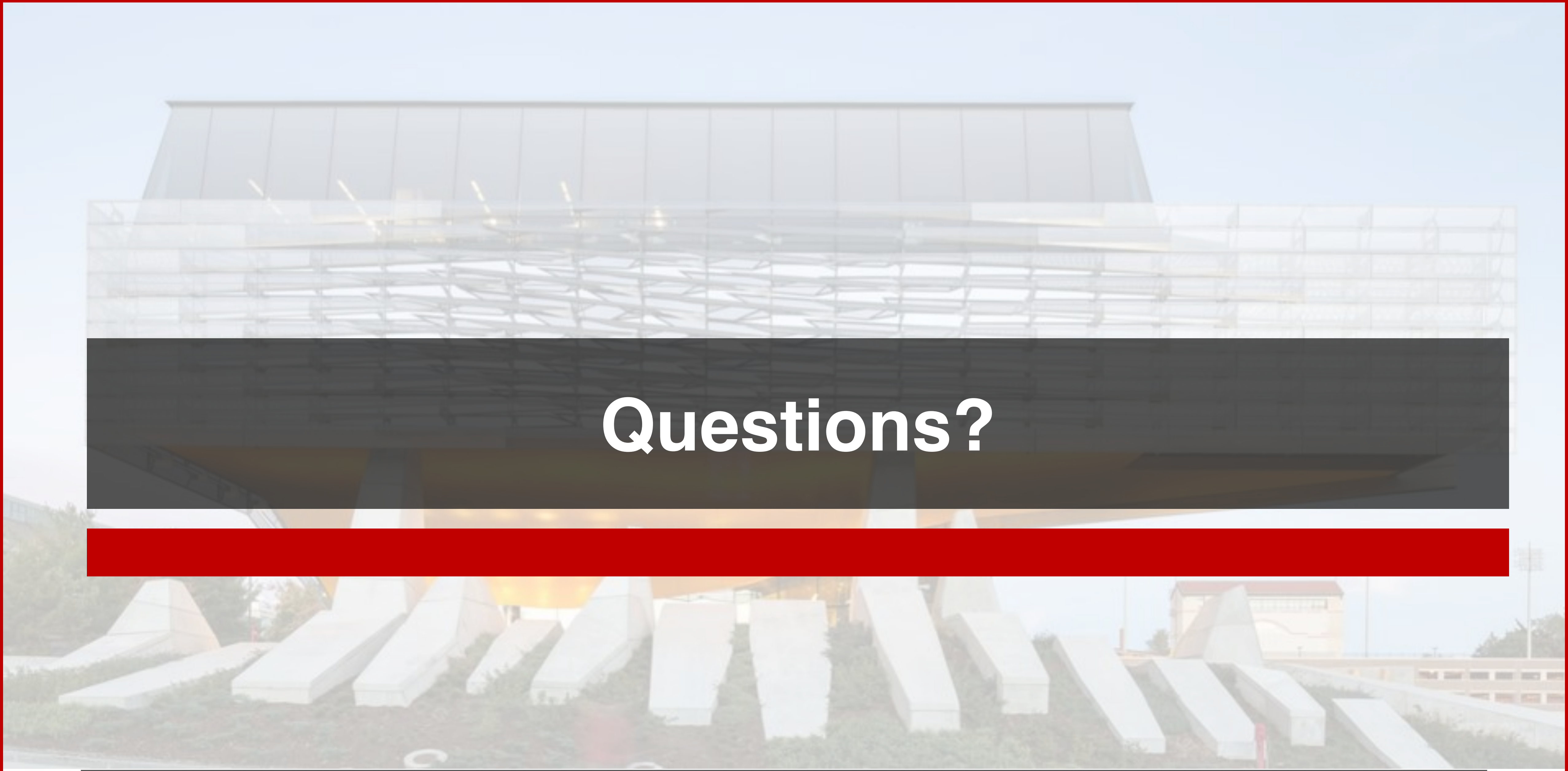


Simple is good!

Bridges are good!

Research is a conversation.

- Stephanie Weirich, PLMW'22



Questions?

Cornell Bowers CIS
Computer Science