A Case Study on Extracting the Characteristics of the Reachable States of a State Machine Formalizing a Communication Protocol with Inductive Logic Programming

Dung Tuan Ho*, Min Zhang**, Kazuhiro Ogata*

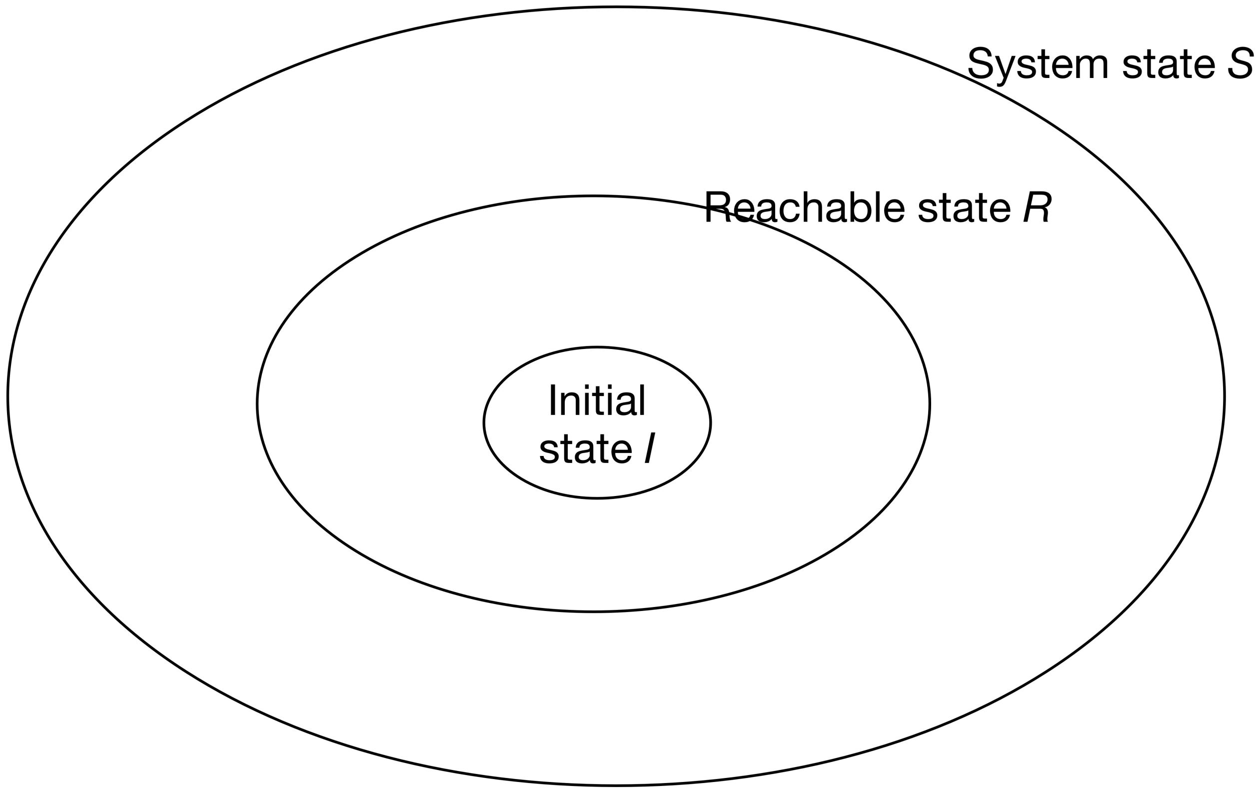
*Japan Advanced Institute of Science and Technology (JAIST)

**East China Normal University (ECNU)

Outline

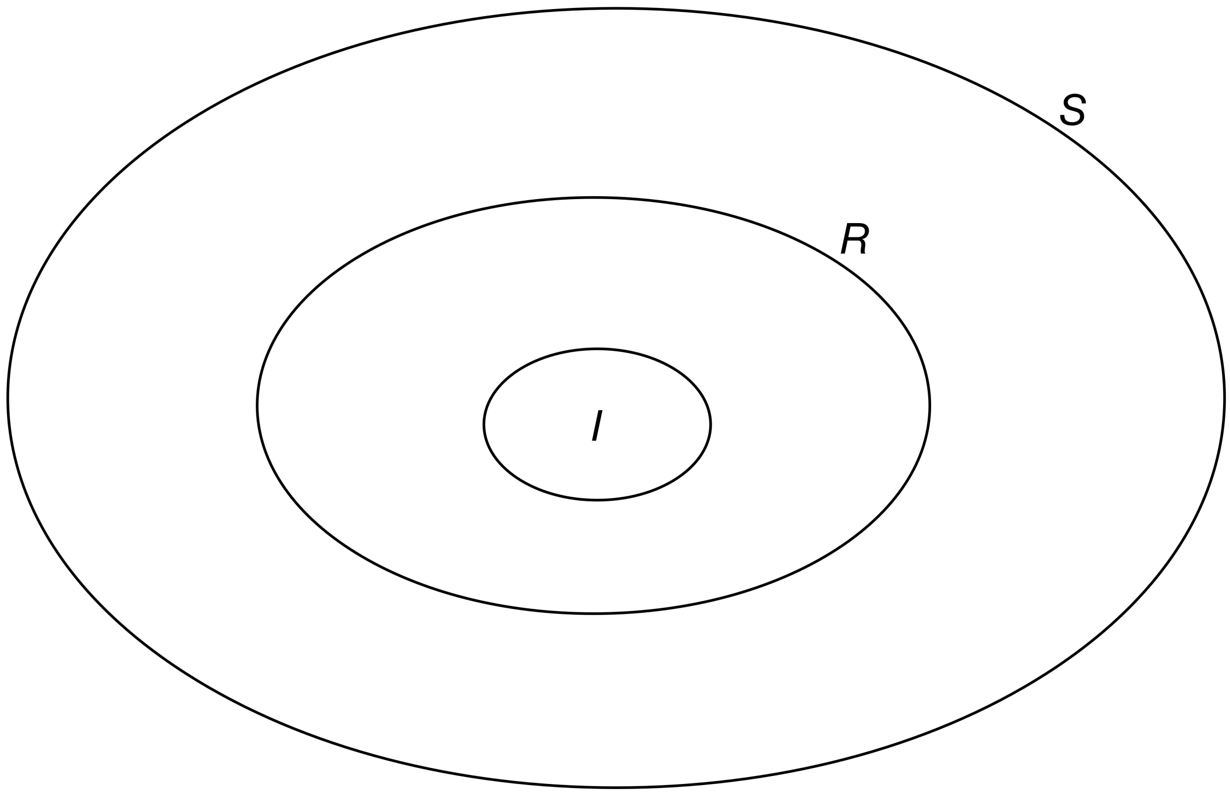
- Interactive Theorem Proving
- Communication Protocol
- Method & Architecture of Tool Used
- Case Studies
- Conclusion and Future Work

Interactive Theorem Proving



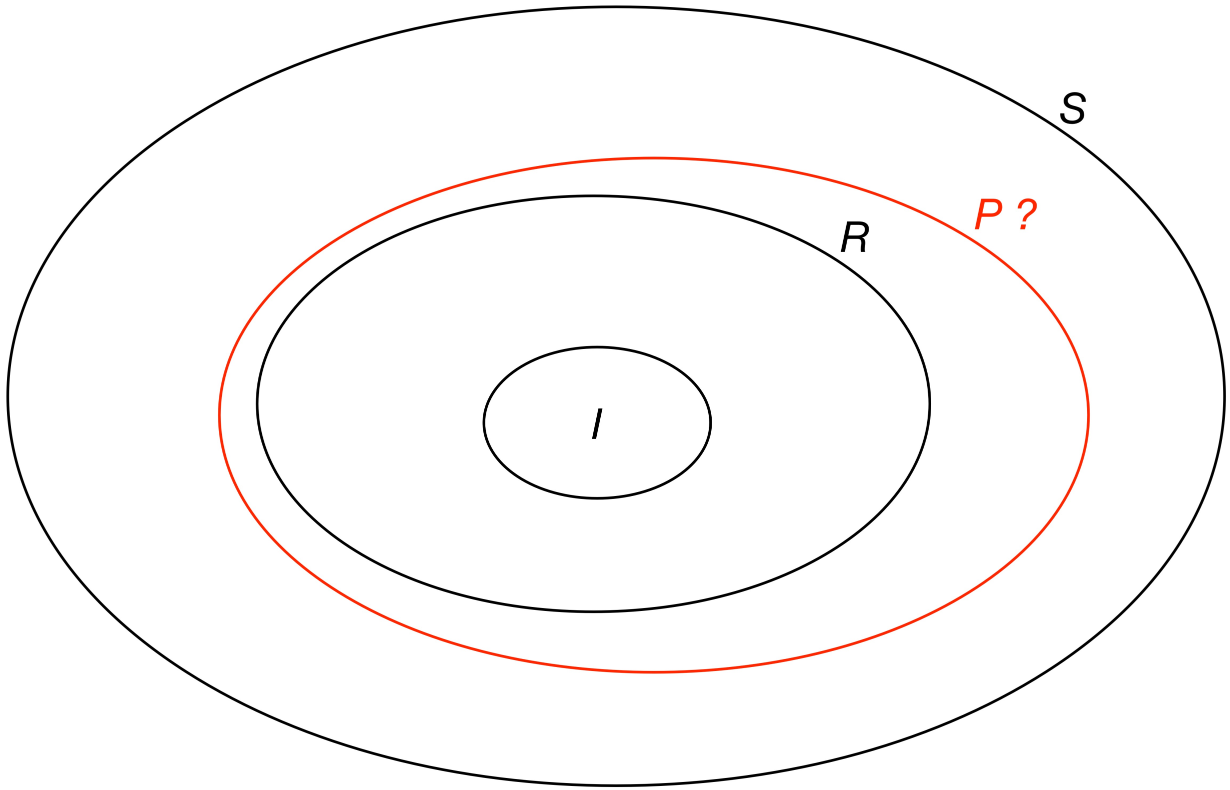
a System ⟶ a State Machine $M \triangleq$ < *S*, *I*, *T*>

Interactive Theorem Proving



a System ⟶ a State Machine $M \triangleq \langle S, I, T \rangle$ State predicate p ⟵ a Property

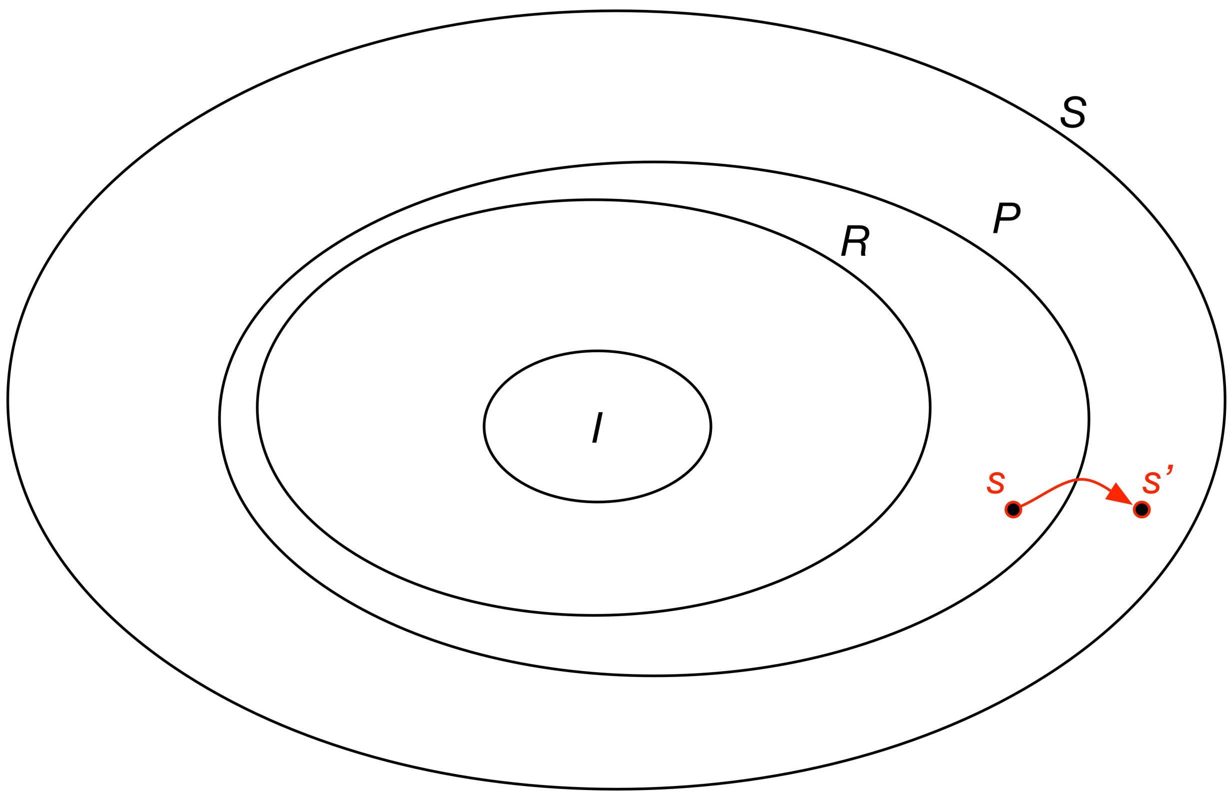
Interactive Theorem Proving



a System ⟶ a State Machine $M \triangleq < S, I, T>$ $\models$ State predicate p ⟵ a Property

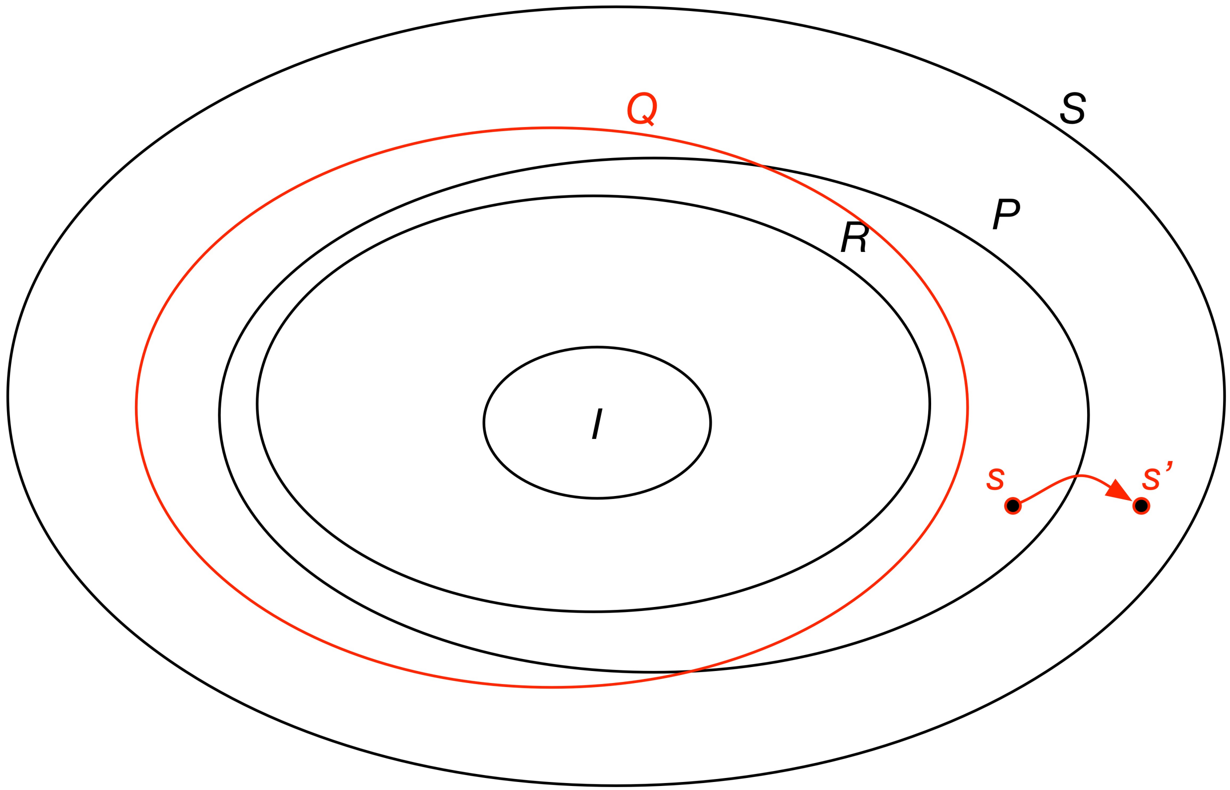
p is an invariant ?

Interactive Theorem Proving



a System ⟶ a State Machine $M \triangleq \langle S, I, T \rangle$ $\models$ State predicate p ⟵ a Property

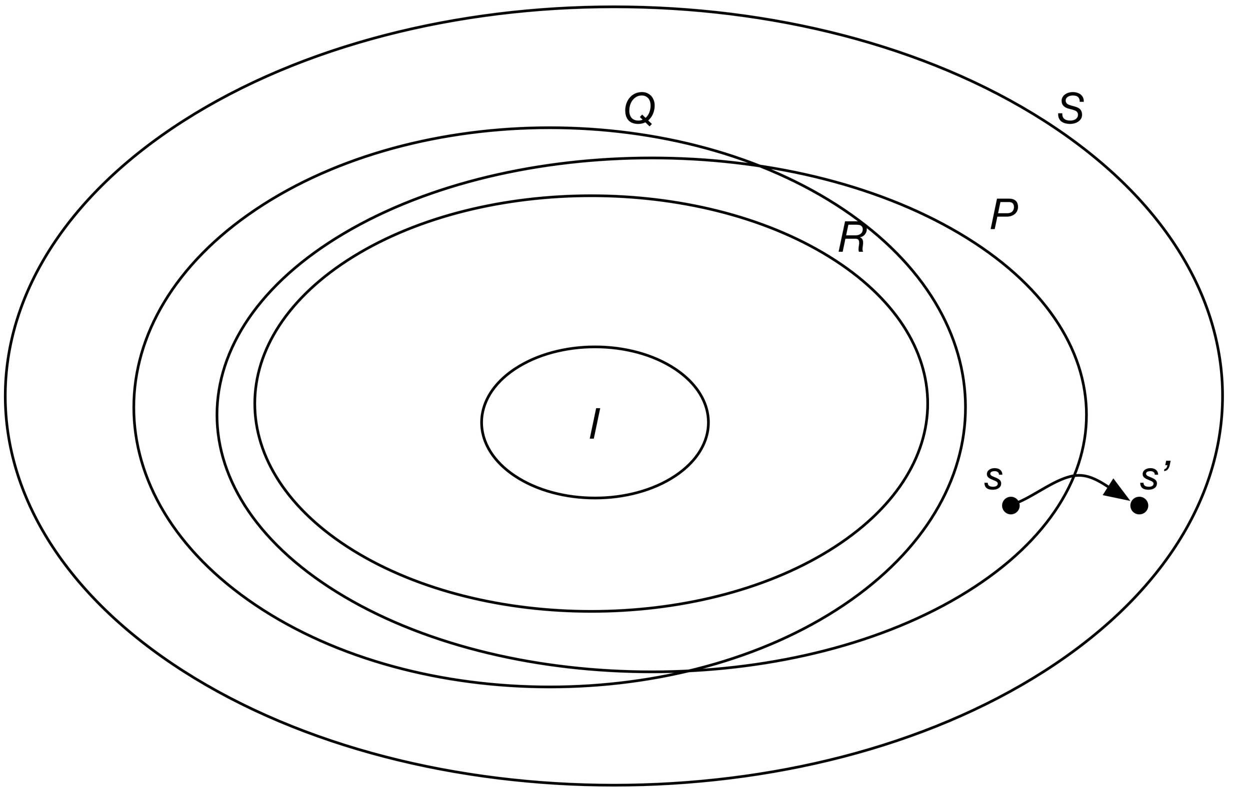
Interactive Theorem Proving



a System $\longrightarrow$ a State Machine $M \triangleq < S, I, T>$ $\models$ State predicate p $\longleftarrow$ a Property

Conjecture a lemma q

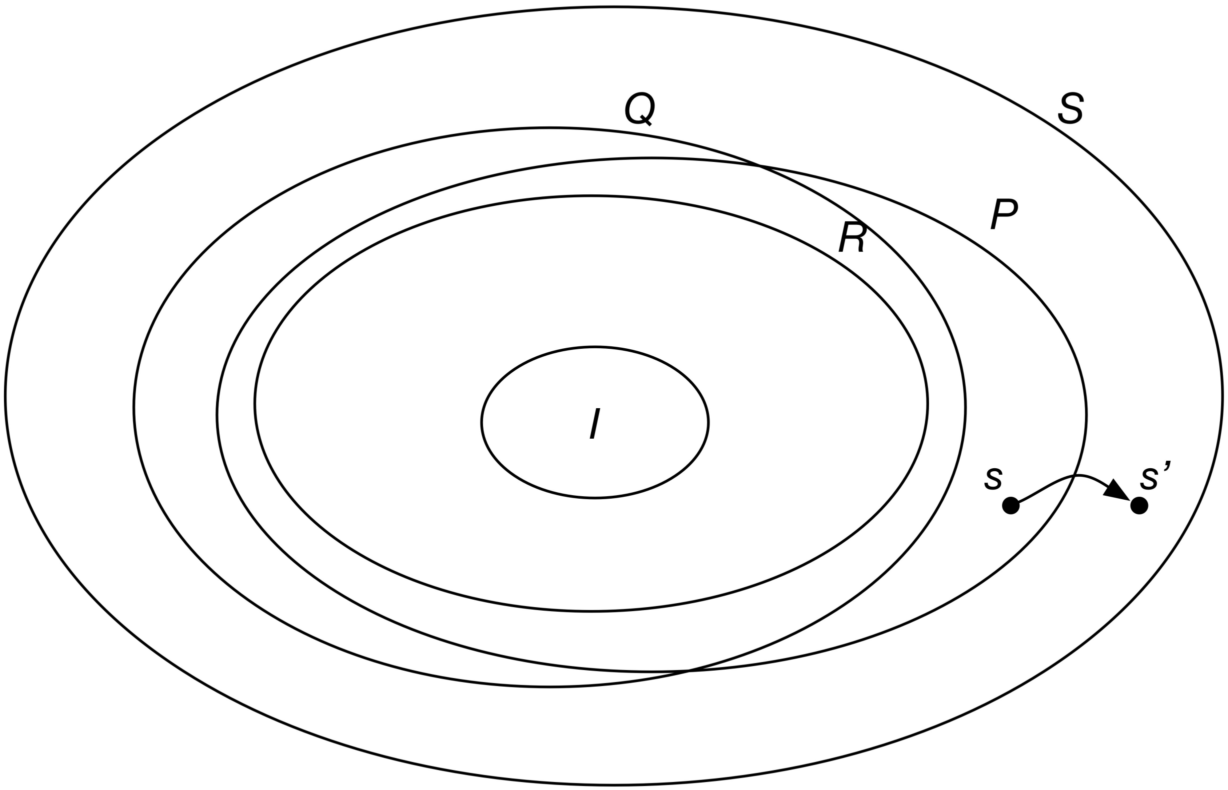
Interactive Theorem Proving



Issue!!!

How to conjecture such lemma q ?

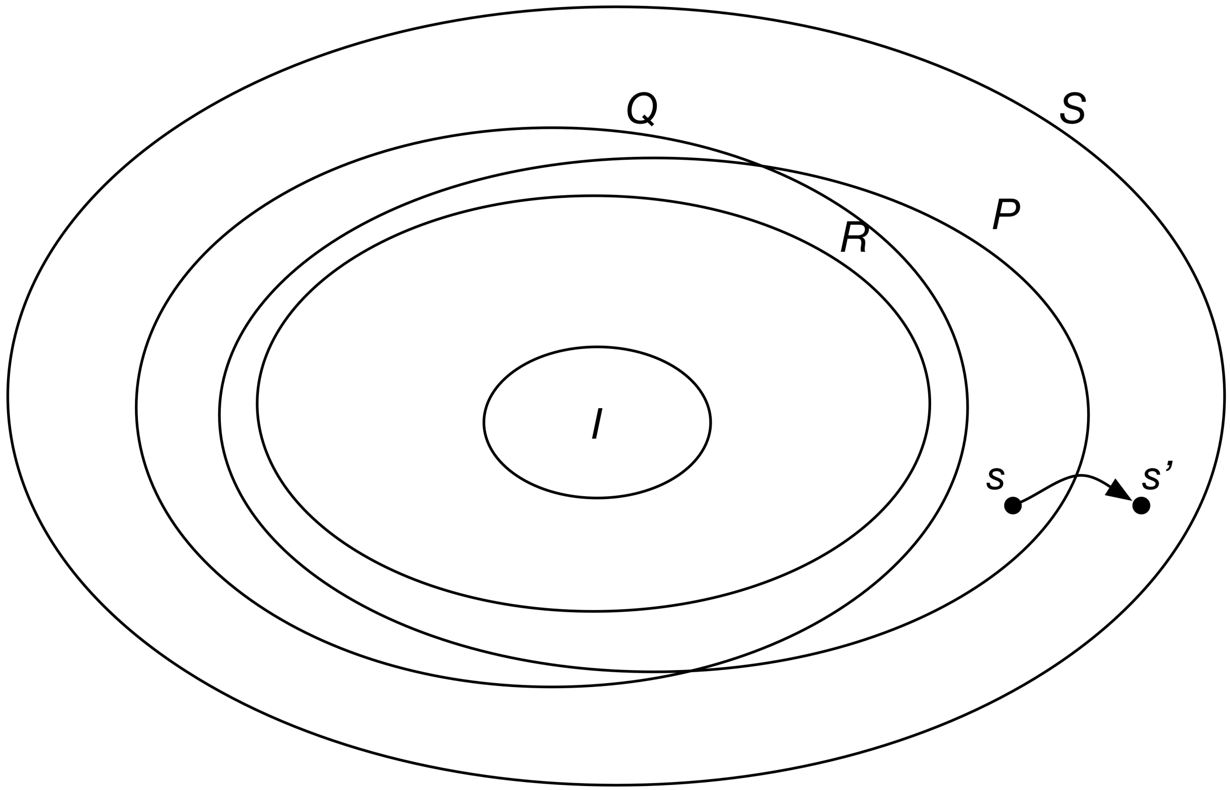
Interactive Theorem Proving



Issue!!!

How to conjecture such lemma q ? Get better understanding

Interactive Theorem Proving

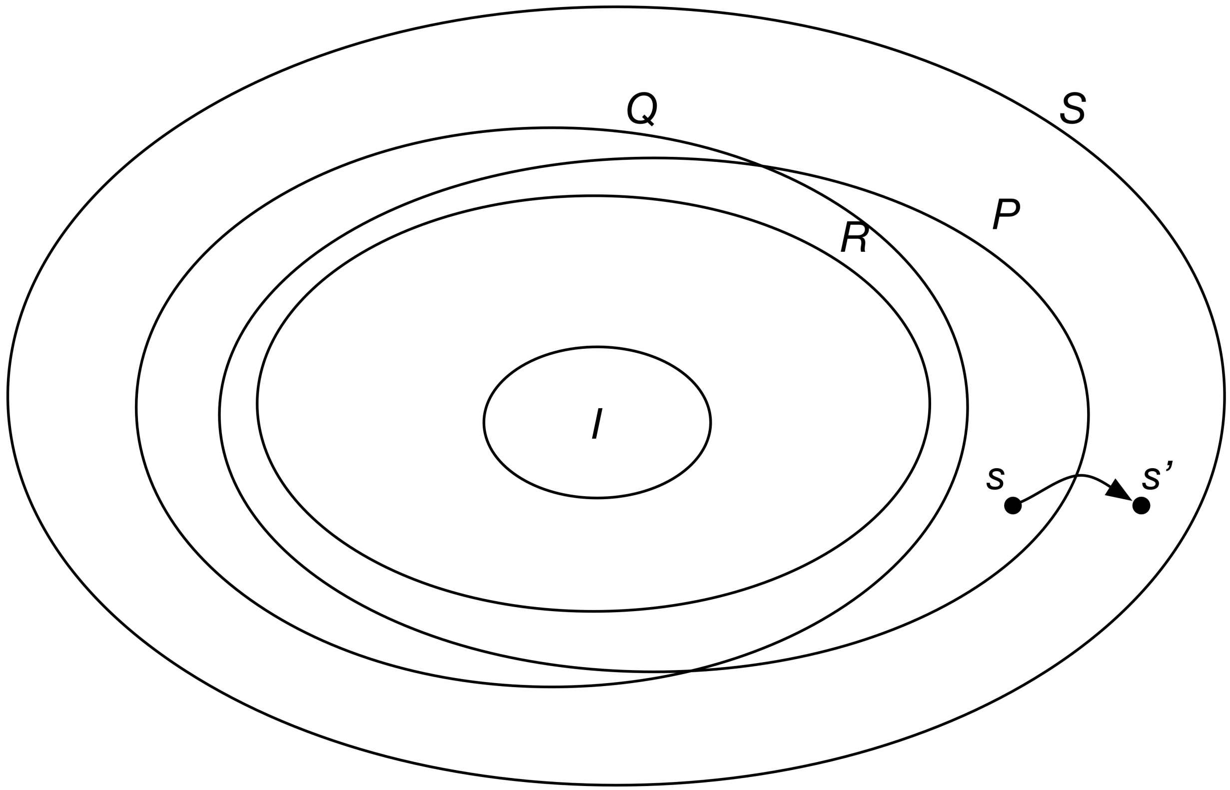


Issue!!!

How to conjecture such lemma q ? Get better understanding

Which is the reliable source ?

Interactive Theorem Proving

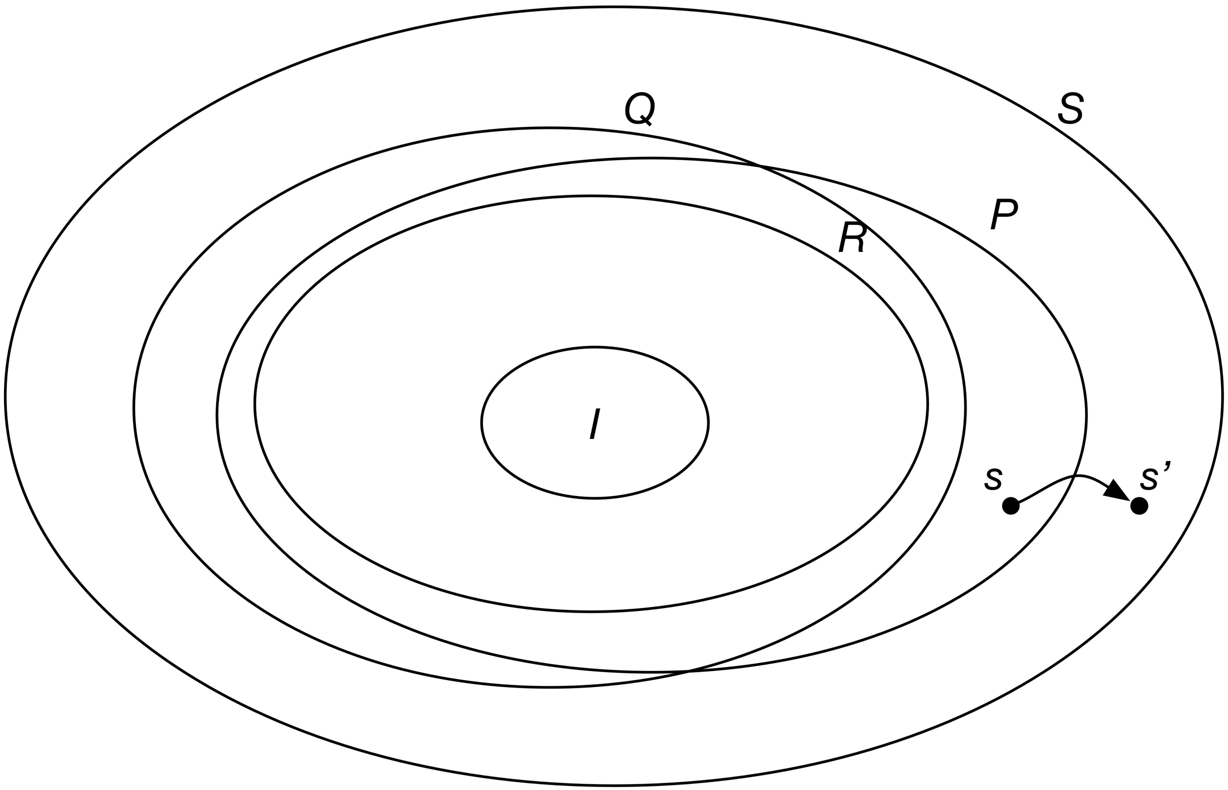


Issue!!!

How to conjecture such lemma q ? Get better understanding

Which is the reliable source ? Reachable States

Interactive Theorem Proving

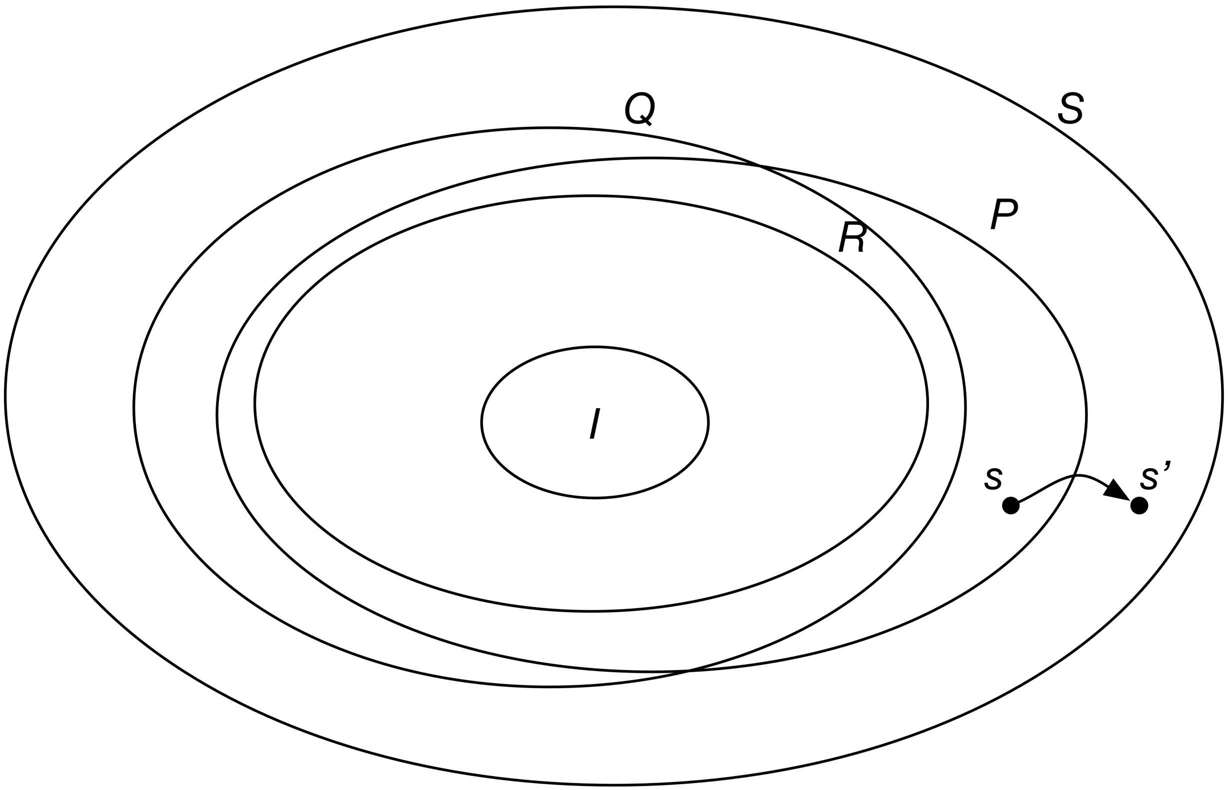


Issue!!!

How to conjecture such lemma q ? Get better understanding

Which is the reliable source ? Reachable States Machine Learning

Interactive Theorem Proving



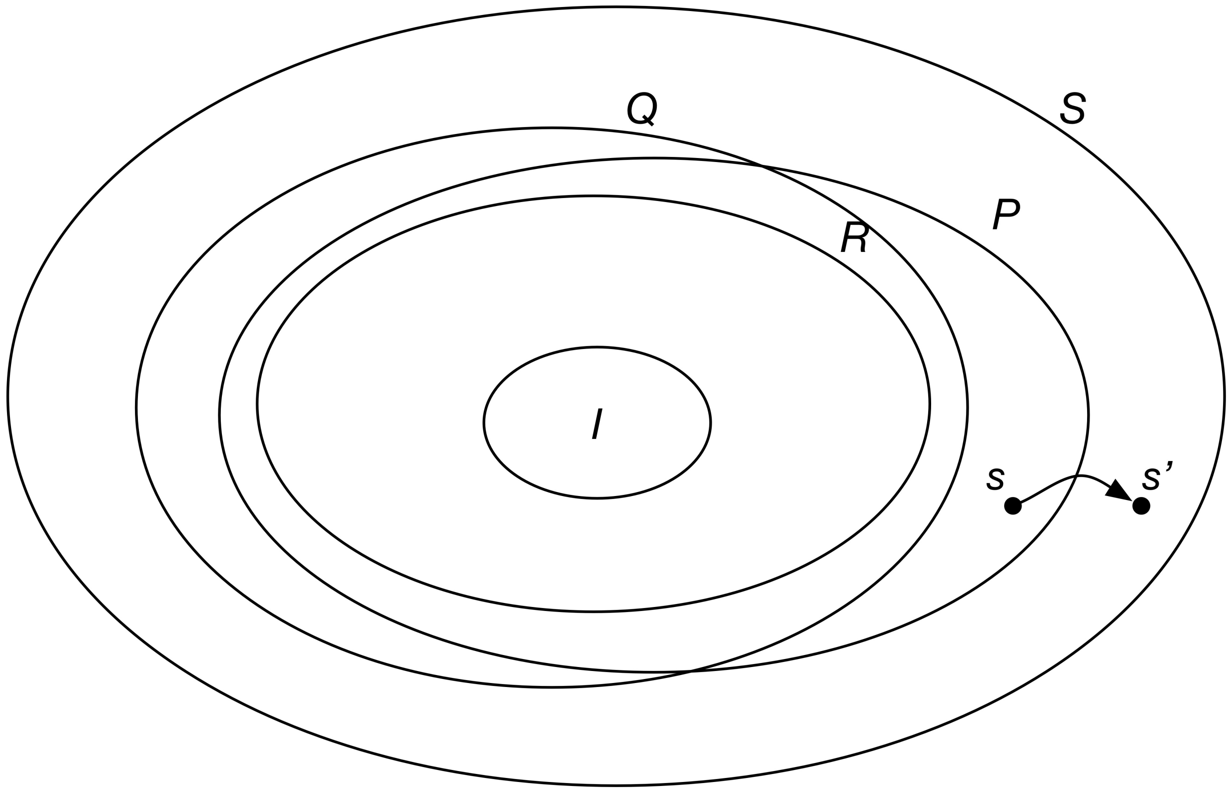
Issue!!!

How to conjecture such lemma q ? Get better understanding

Which is the reliable source ? Reachable States   Machine Learning

Description of Reachable states is first-order logic formulas !!!

Interactive Theorem Proving



Issue!!!

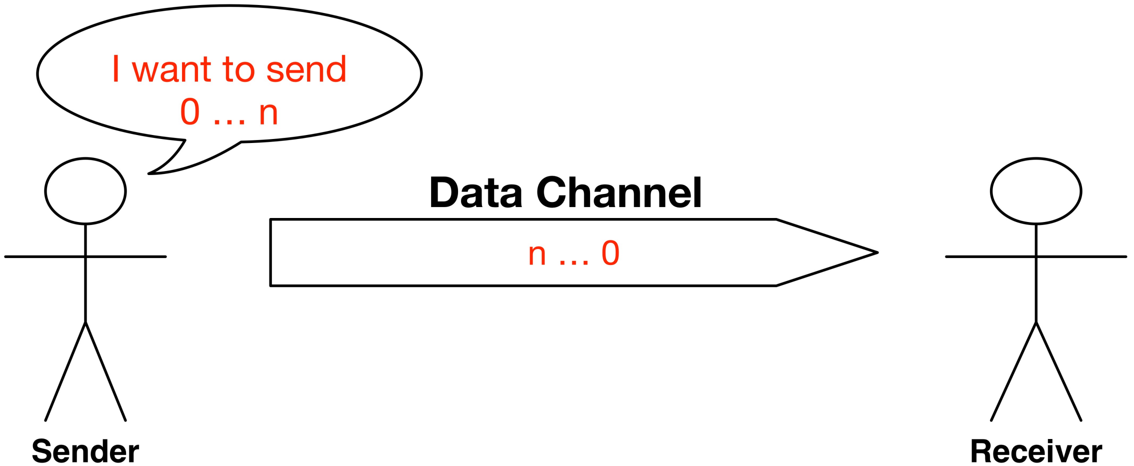
How to conjecture such lemma q ? Get better understanding

Which is the reliable source ? Reachable States   Machine Learning

Description of Reachable states is first-order logic formulas !!!

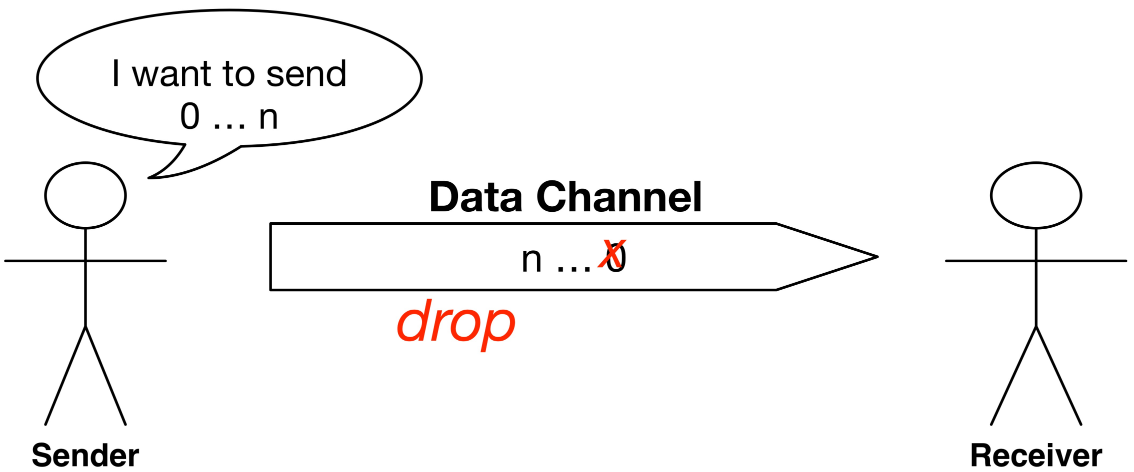
Inductive Logic Programming

Communication Protocol



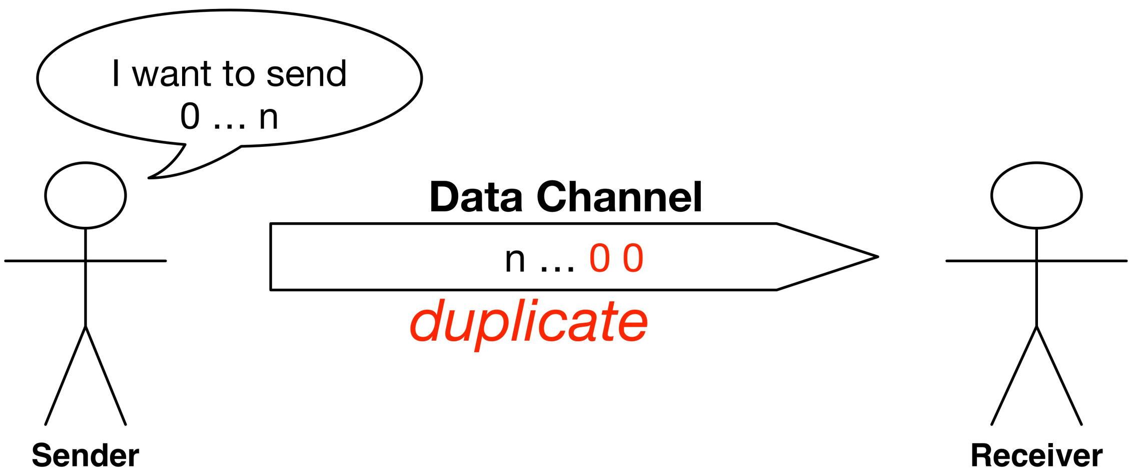
TCP – a Communication Protocol

Communication Protocol



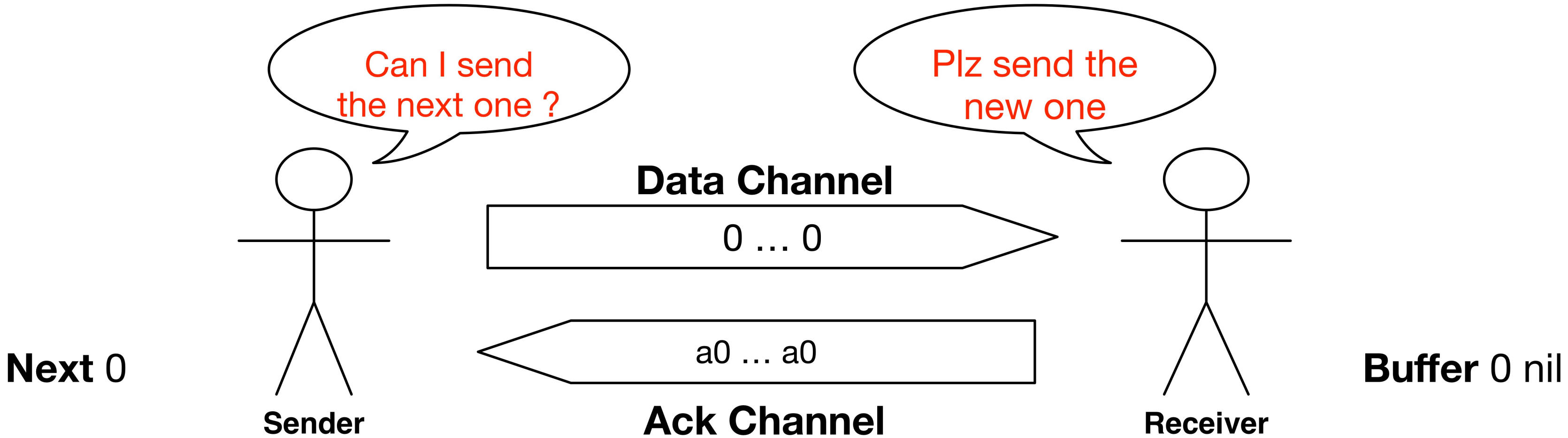
TCP – a Communication Protocol

Communication Protocol

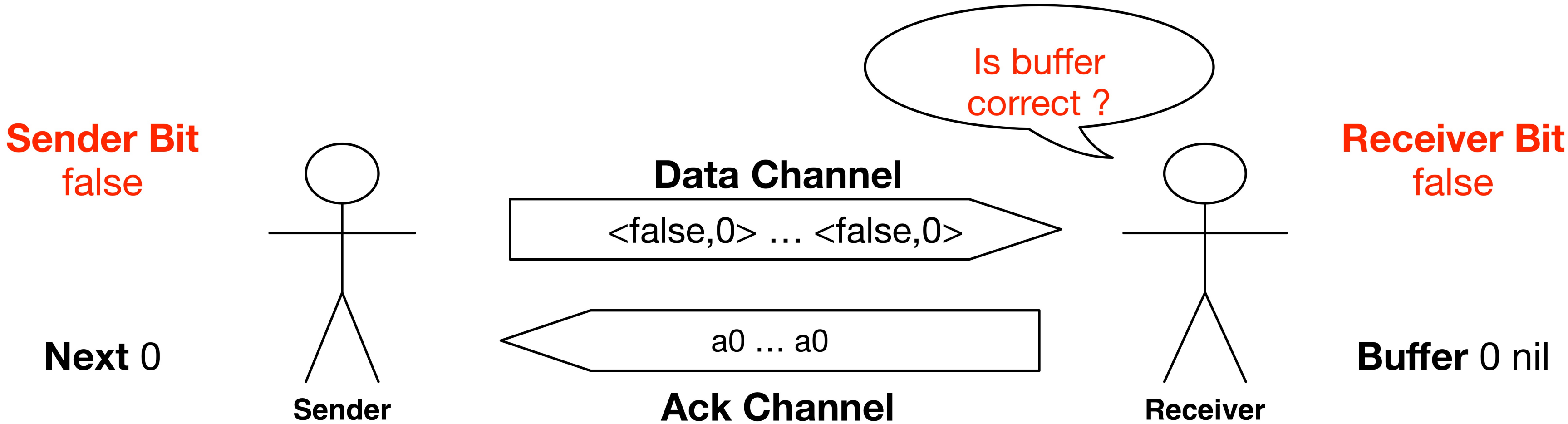


TCP – a Communication Protocol

Communication Protocol



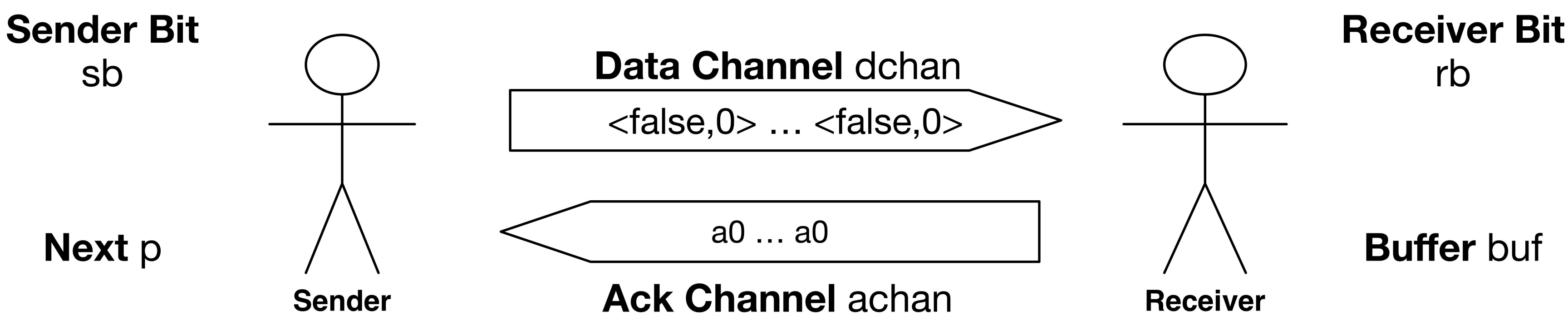
Communication Protocol



Two simplified versions of TCP

- Alternating Bit Protocol (ABP): unbounded channels
- Simple Communication Protocol (SCP) : bounded channels – capacity 1

Communication Protocol



A system state s = < *sb, b, rb, buf, dchan, achan* >

Communication Protocol

Reliable Communication Property The sequence of numbers sent by Sender is successfully received by Receiver, no drop or duplicated ones

$(sb(S) = rb(S) \rightarrow mk(p(S)) = (p(S)\ buf(S))) \wedge (sb(S) \neq rb(S) \rightarrow mk(p(S)) = buf(S))$

Lemmas for SCP verification

$achan(S) = c(b) \rightarrow (sb(S) = b \vee rb(S) = b)$.

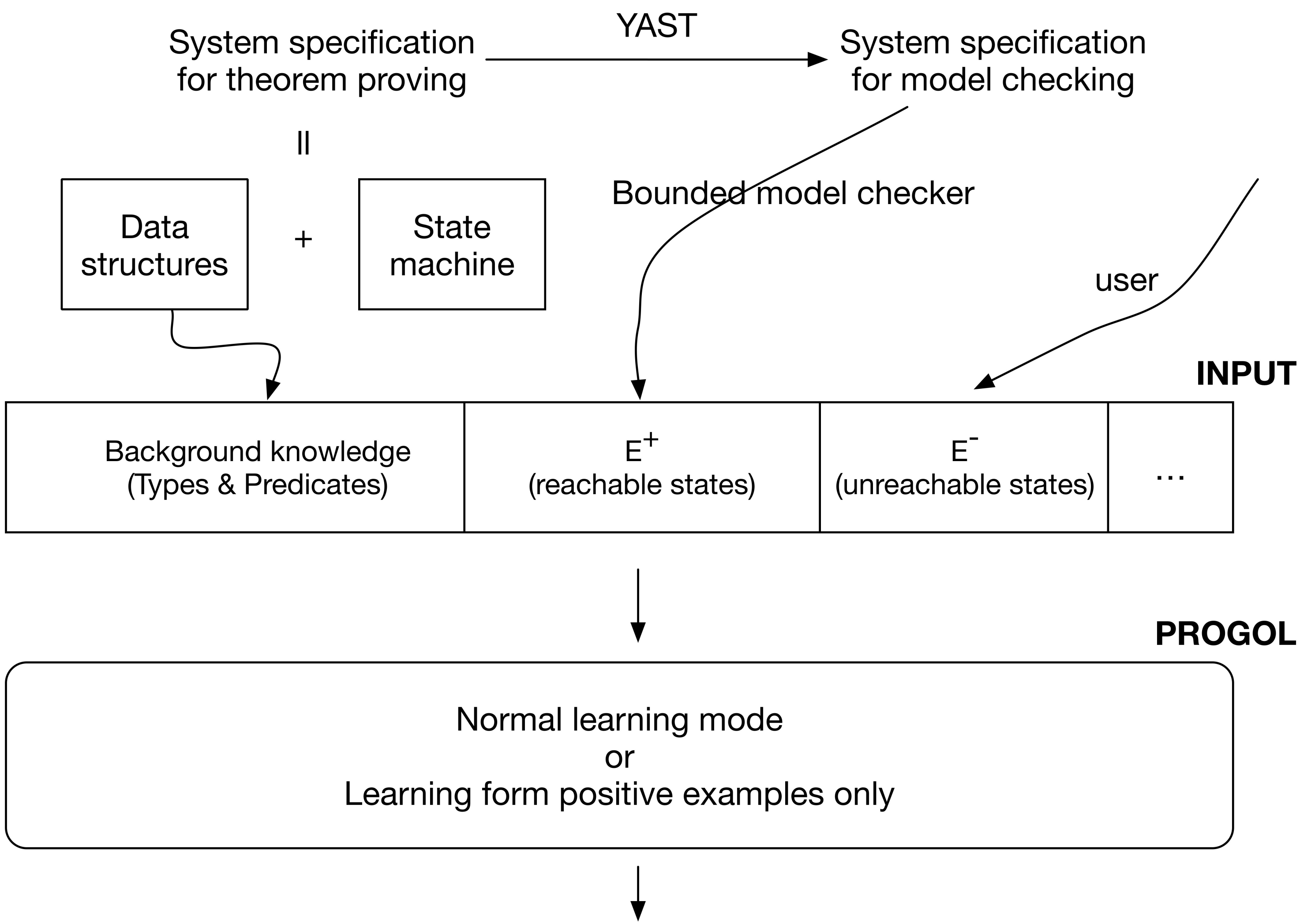
$(dchan(S) = \mathrm{c}(< \mathrm{b}, \mathrm{n} >) \wedge rb(S) = \mathrm{b}) \rightarrow p(S) = \mathrm{n}$.

$(dchan(S) = \mathrm{c}(< \mathrm{b}, \mathrm{n} >) \wedge rb(S) = \mathrm{b}) \rightarrow sb(S) = \mathrm{b}$.

$(dchan(S) = \mathrm{c}(< \mathrm{b}, \mathrm{n} >) \wedge achan(S) = \mathrm{c}(\mathrm{b}')) \rightarrow sb(S) = \mathrm{b}' \vee rb(S) \neq b$

*S : a variable of System states

Method & Architecture of Tool Used



Method & Architecture of Tool Used

Examples

```
state(f,0,t,[0],c(p(f,0)),c(f)).
...
:- state(f,s(0),f,[0],c(p(f,s(s(s(0))),c(t)).
...
```

Background knowledge

```
pnat(0).
pnat(s(X)) :- pnat(X).
...
mk(0,[0]):-!.
mk(s(N),[s(N)|L1]) :- pnat(N), mk(N,L1).
...
```

Mode declaration

```
%scp
:- modeh(1,state(+bool,+pnat,+bool,+nlist,c(p(+bool,+pnat)),c(+bool)))?
%abp
:- modeh(1,state(+bool,+pnat,+bool,+nlist,[p(+bool,+pnat)|+pqueue],[+bool|+bqueue]))?


:- modeb(1,neg(+bool,+bool))?
:- modeb(1,mk(+pnat,+nlist))?
...
```

* < sb, p, rb, buf, dchan, achan>

Case Studies

	Input				Output
SCP	Background knowledge B	Learning mode	#States E	Constraints	
	- Boolean - Natural numbers - List of natural numbers	- Original learning mode - <u>Learning form positive examples only</u>	100 – 5000 <u>1000</u>	- Size of p is fixed 1 - 10 - Only non-empty lists - Several initial states	Every characteristic
ABP	- (SCP's B) - Queue of boolean values - Queue of < b, n > - User-defined predicates	- Original learning mode - <u>Learning form positive examples only</u>	500 – 10000 <u>1500</u>	- Size of p is fixed 1 – 10 - Lists/Queues contain at least 2 elements - Several initial states	-Some characteristics - Non-trivial characteristics need some non-trivial B

SCP Case Study

- Set 1
  - state(A,B,C,D,c(p(A,B)),c(E)) :- mk(B,D).
  - state(A,B,A,C,c(p(D,E)),c(A)) :- neg(A,D), mk(B,[B|C]).
  - state(A,B,A,C,c(p(A,B)),c(A)) :- mk(B,[B|C]).
- Set 2
  - state(A,B,C,D,c(p(A,B)),c(A)) :- neg(A,C).
  - state(A,B,C,D,c(p(A,B)),c(C)) :- neg(A,C), mk(B,D).
  - state(A,B,A,C,c(p(D,E)),c(A)) :- neg(A,D), mk(B,[B|C]).

* < sb, p, rb, buf, dchan, achan>

SCP Case Study

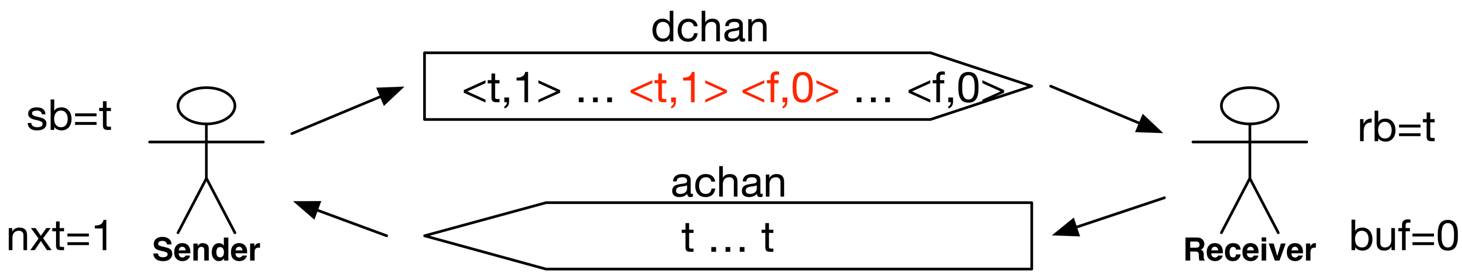
- Set 1
  - state(A,B,C,D,c(p(A,B)),c(E)) :- mk(B,D).
  - state(A,B,A,C,c(p(D,E)),c(A)) :- neg(A,D), mk(B,[B|C]).
  - state(A,B,A,C,c(p(A,B)),c(A)) :- mk(B,[B|C]).
- Set 2
  - state(A,B,C,D,c(p(A,B)),c(A)) :- neg(A,C).
  - state(A,B,C,D,c(p(A,B)),c(C)) :- neg(A,C), mk(B,D).
  - state(A,B,A,C,c(p(D,E)),c(A)) :- neg(A,D), mk(B,[B|C]).

Manually conjecture

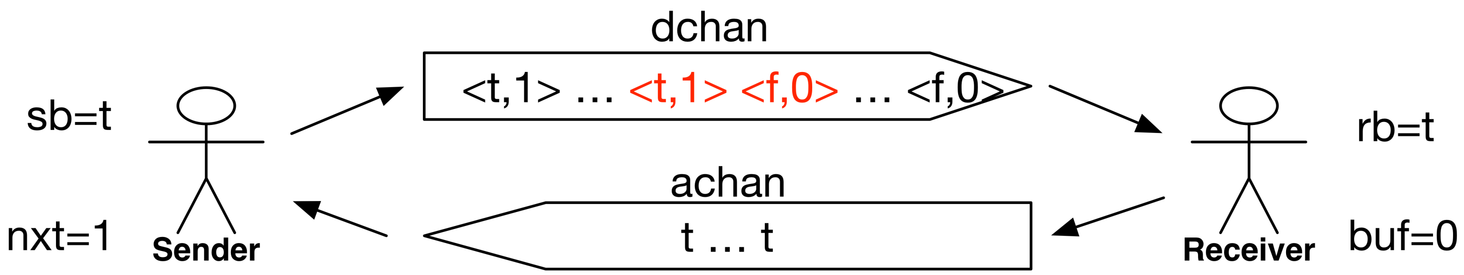
$(achan(S) = c(b) \wedge sb(S) \neq b) \rightarrow rb(S) = b$

* < sb, p, rb, buf, dchan, achan>

ABP Case Study



ABP Case Study




```
state(A,B,C,D,[p(A,B)|E],[A|F]) :- neg(A,C), gap0(p(A,B),E).
state(A,B,C,D,[p(A,B)|E],[C|F]) :- mk(B,D), gap0(p(A,B),E).
state(A,B,A,C,[p(D,E)|F],[A|G]) :- neg(A,D), succ(E,B), mk(B,[B|C]), gap1(p(A,B),F).
state(A,B,A,C,[p(A,B)|D],[A|E]) :- mk(B,[B|C]), gap0(p(A,B),D).
state(A,B,A,C,[p(A,B)|D],[E|F]) :- neg(A,E), mk(B,C), gap0(p(A,B),D).
```

```
gap0(P,[]) :- bnpair(P).
gap0(P,[P|T]) :-  bnpair(P), gap0(P,T).

gap1(P,[]) :- bnpair(P).
gap1(P1,[P2|T]) :- bnpair(P1), bnpair(P2), ((P1 \== P2, next(P2,P1), gap1(P1,T)); gap0(P1,T)).
```

Conclusion

- ILP can characterize reachable states from a system specification
- Some useful lemmas are manually conjectured
- Some non-trivial characteristics need a non-trivial background knowledge

Future work

- Systematically find non-trivial predicates
  - Predicate Invention
- Systematically conjecture useful lemmas from the characteristics