max planck institut
informatik

Automated Reasoning

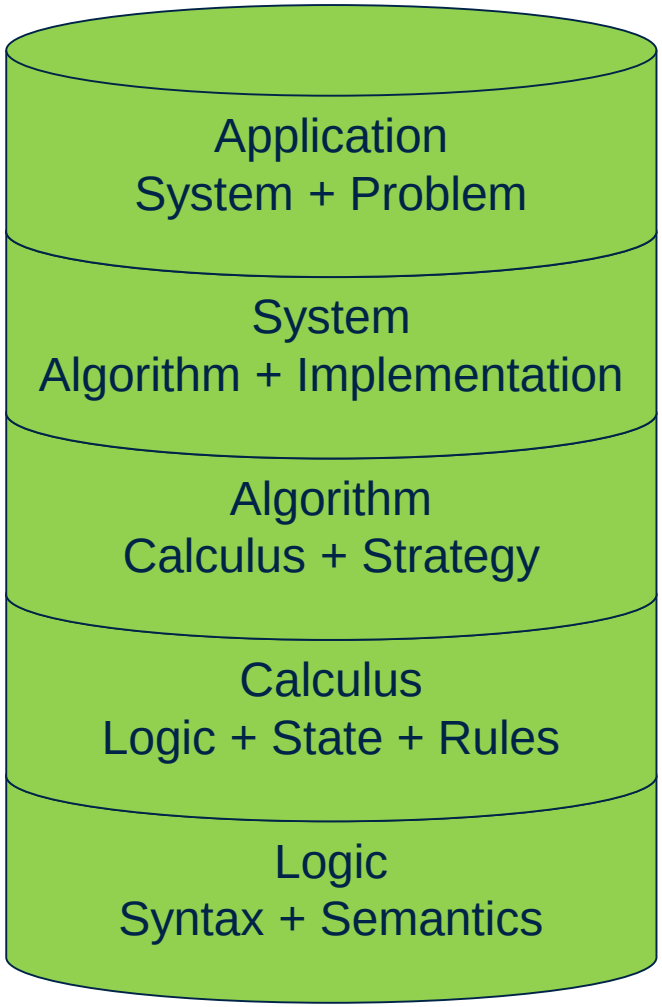
Practical Session

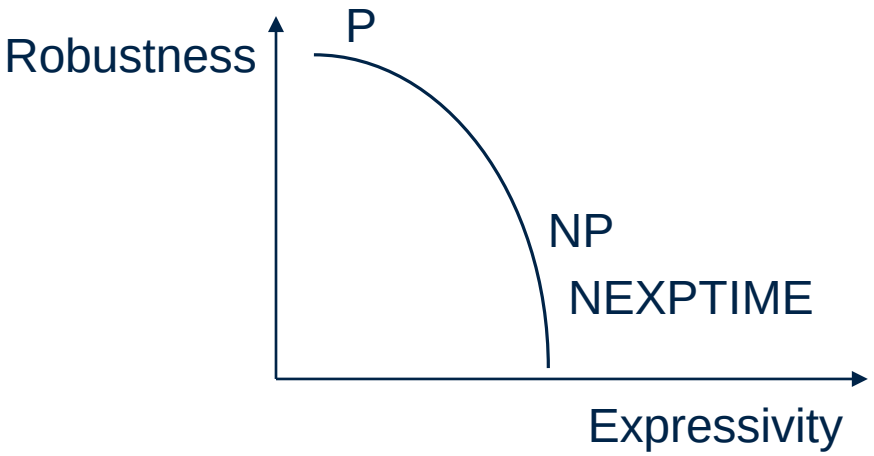
Christoph Weidenbach

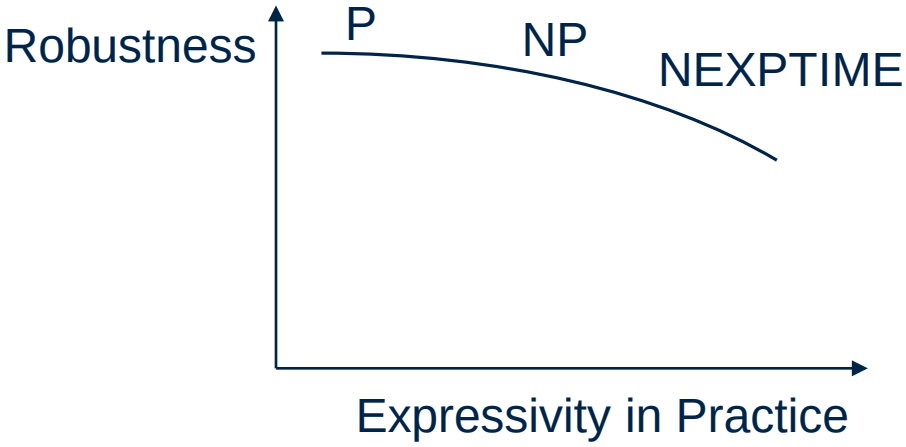
The Practical AR World

max planck institut informatik

The AR Tower







AR System Properties

max planck institut informatik

Soundness:

if the AR system says “yes” there is a solution

Completeness:

if there is a solution the AR system says “yes”

Termination:

the AR system always terminates saying “yes” or “no”

Redundancy:

the AR system eliminates redundant clauses

I don’t bash Soundness, Completeness, Termination, consider Redundancy.

AR System Properties

max planck institut informatik

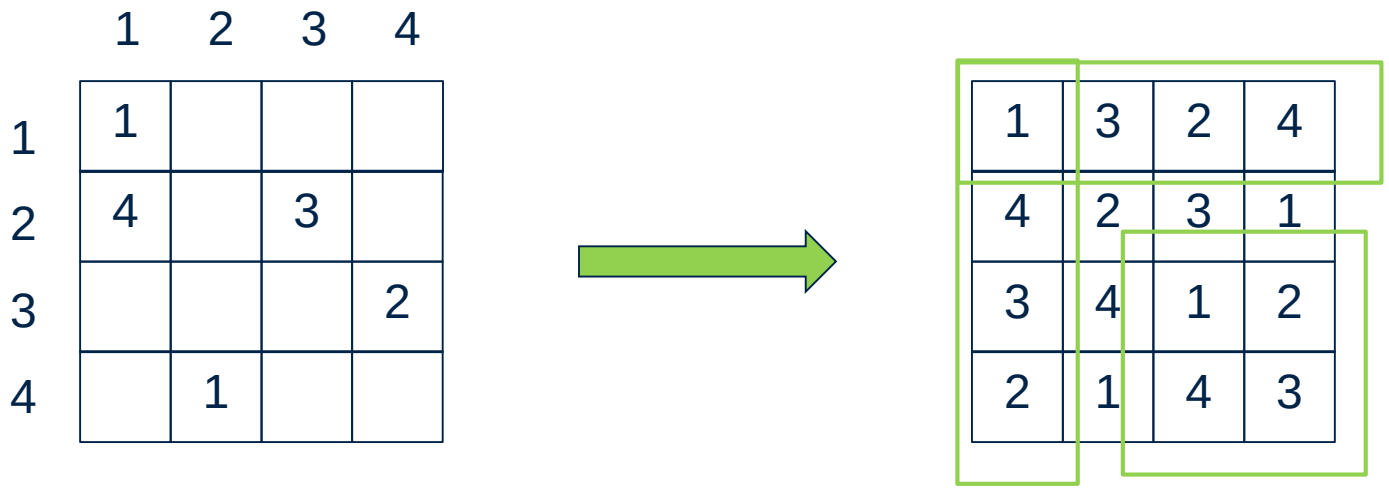
Propositional SAT Solver:

- satisfiability: difficult
- unsatisfiability: more difficult

First-Order Theorem Prover:

- satisfiability: undecidable (very more difficult)
- unsatisfiability: semi-decidable (very difficult)

Sudoku (Propositional Logic)



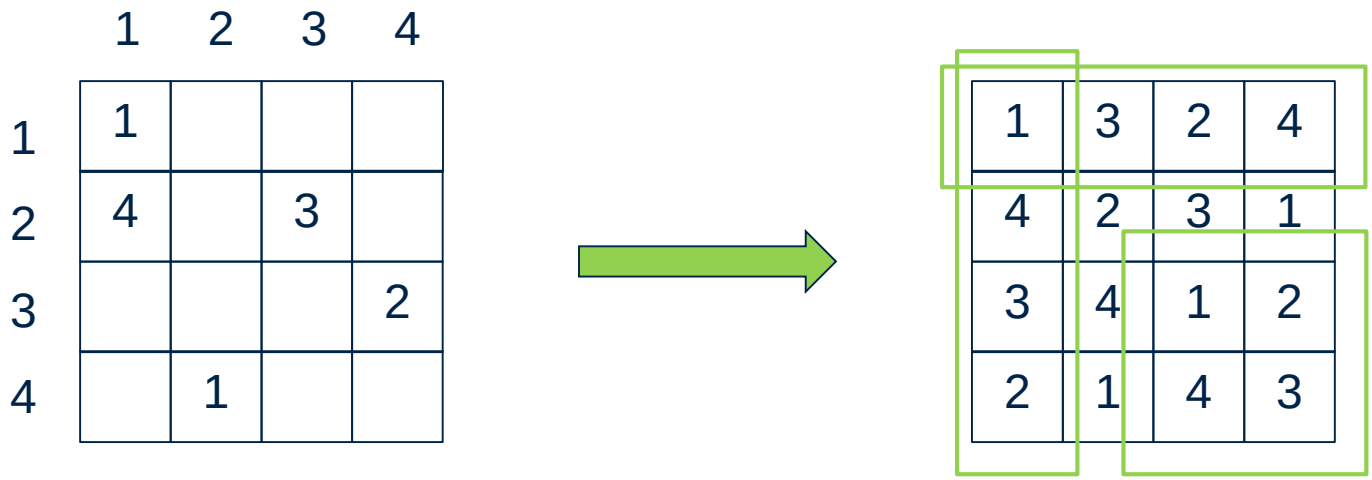
Pxyz is true if square (x,y) holds value z

P111 and P214 and P233 and P342 and P421

SPASS connectives: and(), or(), implies(), not()

Size of the encoding: $O(n^4)$

Sudoku (First-Order Logic)



Function value(x,y) = z is true if square (x,y) holds value z

All variables are restricted to the domain 1, 2, 3, 4

Basically, encoding is in BSR, a NEXPTIME-complete class

Size of the encoding: $O(n^2)$

Redundancy

max planck institut informatik

$$\text{Number}(u)\ \text{Number}(v)\ \text{Number}(w)\ \text{equal}(\text{value}(u,w),2) \rightarrow$$
$$\text{equal}(\text{value}(v,w),1)\ \text{Number}(4)\ \text{Number}(3)\ \text{equal}(v,u)$$

$$\Upsilon$$

$$\rightarrow \text{Number34}(3)$$

$$\rightarrow \text{Number34}(4)$$

$$\text{Number34}(u) \rightarrow \text{Number}(u)$$

$$\rightarrow \text{Number}(3)$$

Redundancy Continued

$$\{V \to \neg S; Q \to R, V\} \Rightarrow_{\mathsf{Res}} \{V \to \neg S; Q \to R, V; Q \to R, \neg S\}$$

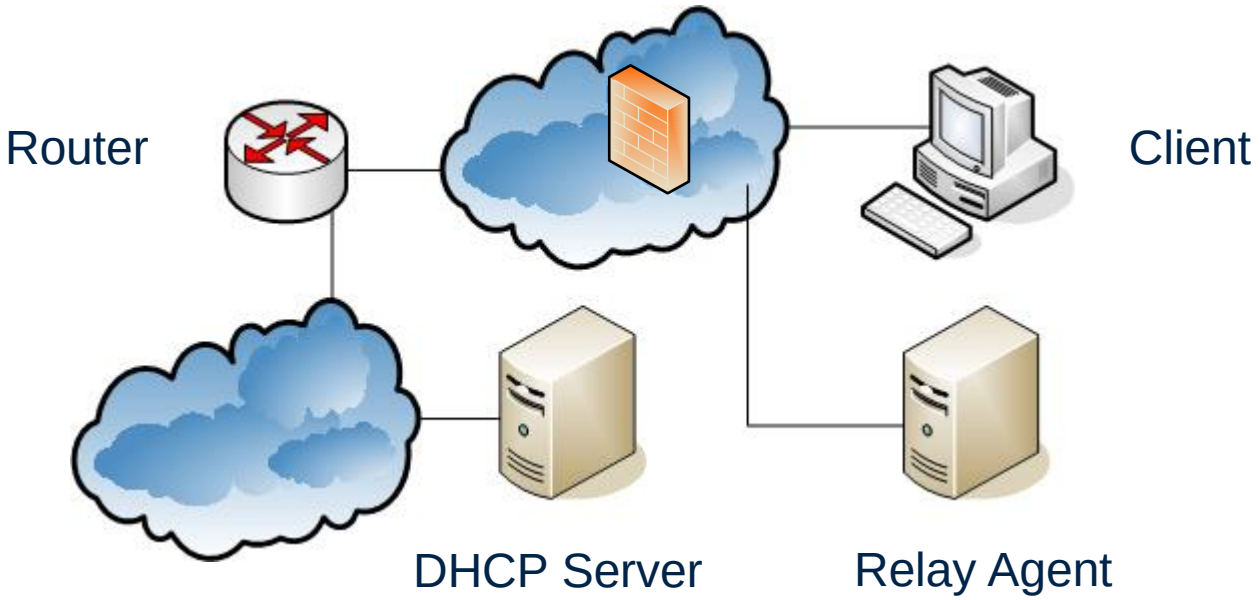
$$P \prec Q \prec T \prec R \prec S \prec U \prec V$$

$$V \to \neg S; Q \to R, V \quad \succ \quad Q \to R, \neg S$$

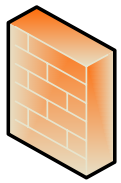
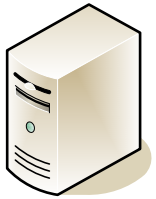
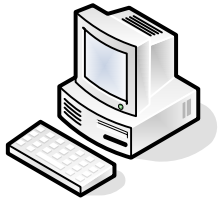
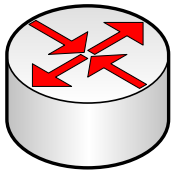
Analysis of LAN Infrastructures 2006 Simon Hirth, Carsten Karl

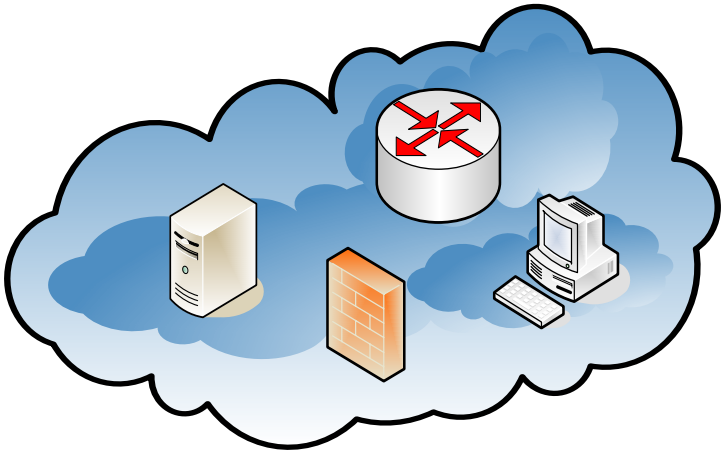
max planck institut informatik

- LAN Infrastructure: all LAN services supporting a typical user client-server application (DHCP, Firewalling, TCP/IP, Forwarding/Routing)
- Analysis: Functionality, Changes, Errors, Security
- Example: DHCP



The Approach

First-Order Model

+

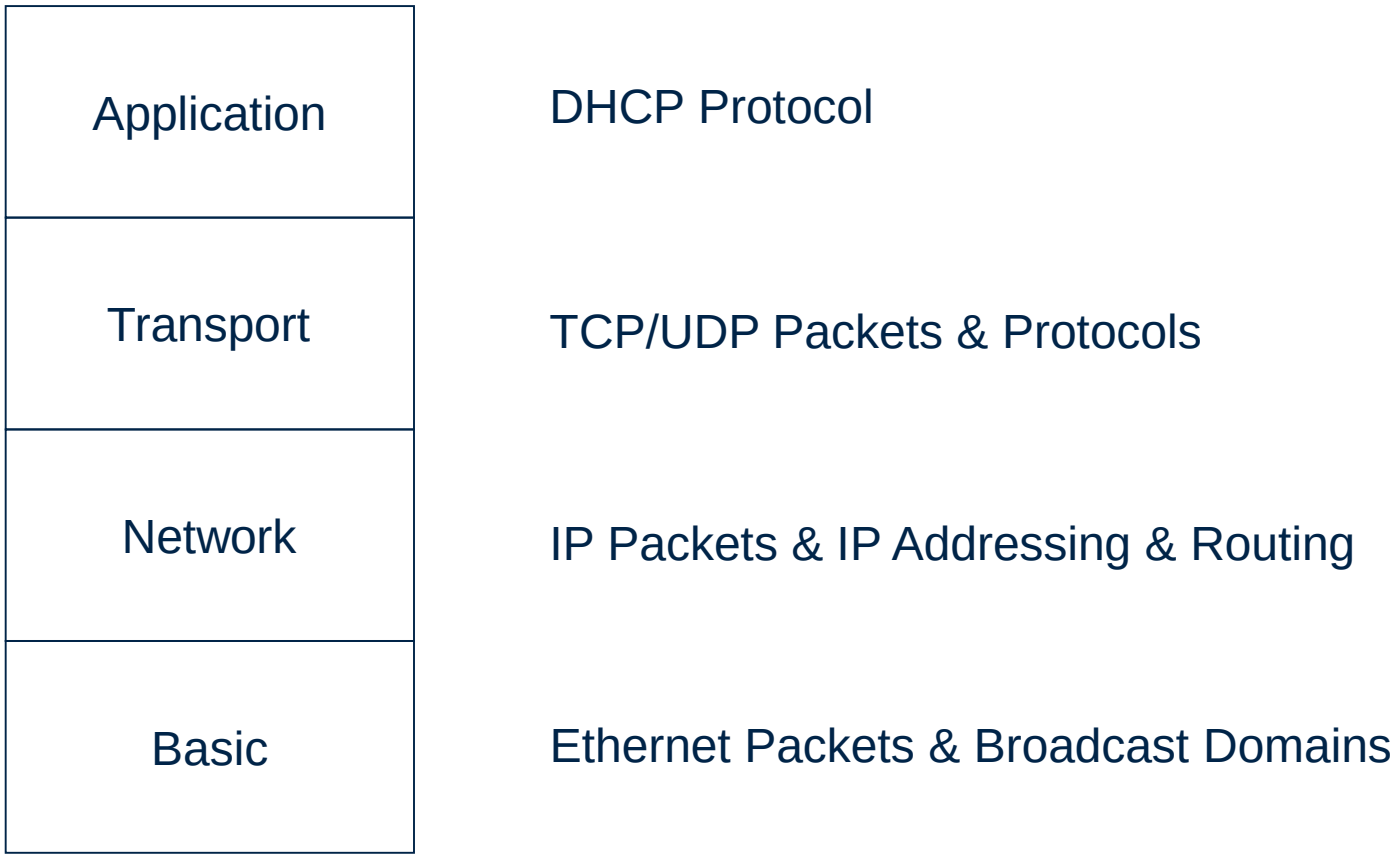
Configuration

Properties

Network Model

- Perfect Channels: Unlimited Bandwidth, Zero Latency, No Errors
- Time: Sequence of Sent Packets
- Broadcast Domains/Ethernet Segements: Set of Sent Packets
- Packets: First-Order Terms

TCP/IP Stack

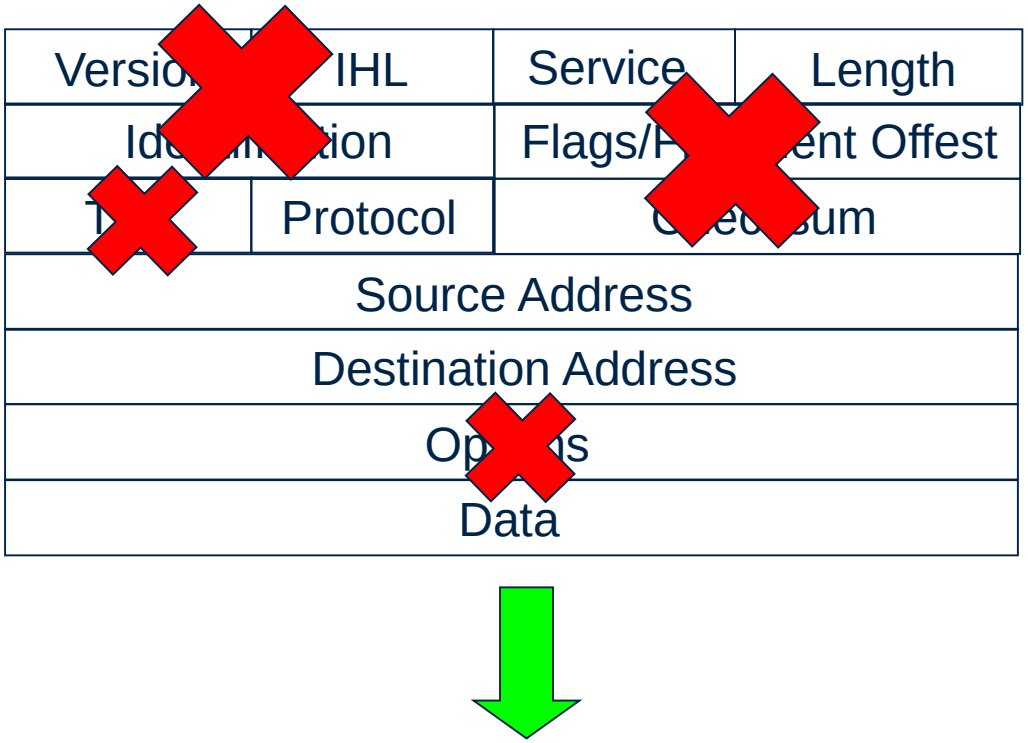

Application
Transport
Network
Basic
DHCP Protocol
TCP/UDP Packets & Protocols
IP Packets & IP Addressing & Routing
Ethernet Packets & Broadcast Domains

Basic

epacket(network, dest_address, source_address, type, data)

used to distinguish ethernet broadcast domains

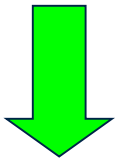
Network: IP Packets



ippacket(ip_source_address, ip_dest_address, proto, data)

Network: IP Addressing

IP addresses are 32-Bit Numbers: 10.5.6.7



ip(0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1,1)

ip(0,0,0,0,1,0,1,0,
0,0,0,0,0,1,0,1,
0,0,0,0,0,1,1,0,
0,0,0,0,0,1,1,1)

Network: IP Network Calculation

max planck institut informatik

ipand(ip(x0, …,x31), ip(y0, …,y31)) = ip(land(x0,y0), …., land(x31,y31)))

land(x,y) = 0 then x = 0 or y = 0

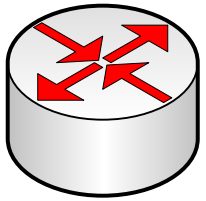
land(x,y) = 1 then x = 1 and y = 1

land(0,x) = 0

land(1,x) = x

Network: IP Routing

max planck institut informatik

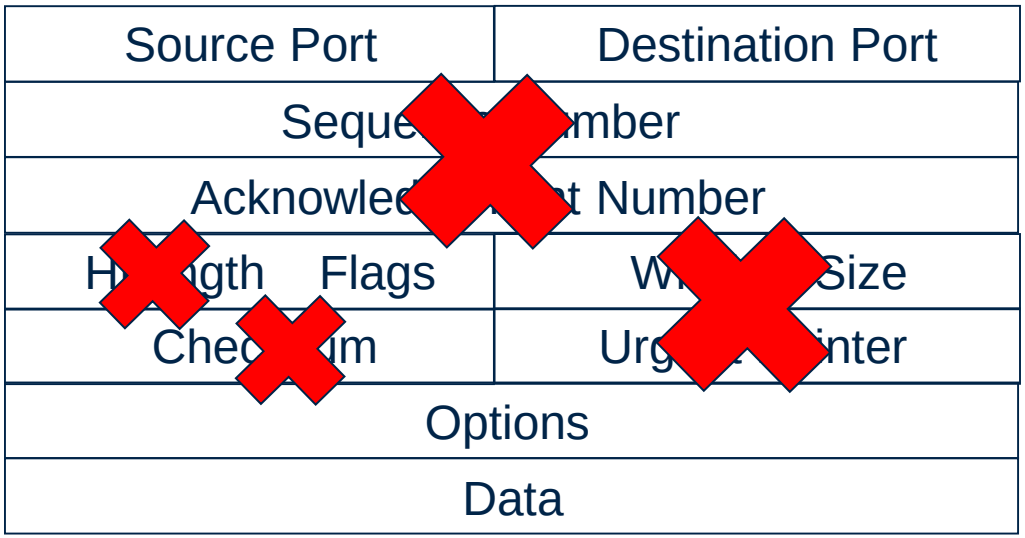


- Determine Routes (not modeled)
- Receives Ethernet frames
- Extracts the Destination IP Address
- Checks for a Route Entry for the Destination Address
- Forwards the Packet to the Destination or Next Router
- Longest Prefix Matches

Router: Forwarding

$$
\begin{aligned}
&\text{RouteIPPacket}(\text{route\_ip\_packet}(\text{v\_host},\\
&\qquad \text{ippacket}(\text{v\_ip\_src}, \text{v\_ip\_dst}, \text{v\_ip\_proto}, \text{v\_ip\_payload}\ ))) \wedge\\
&\text{Interface}(\text{interface}(\text{v\_host},\text{v\_ip\_host},\text{v\_netmask},\text{v\_mac\_src},\text{v\_segment})) \wedge\\
&\text{RouteEntry}(\text{route}(\text{v\_host},\text{v\_route\_mask},\text{v\_dst\_net\_addr},\text{local},\text{first\_route})) \wedge\\
&\text{equal}(\ \text{ipand}(\text{v\_ip\_dst},\text{v\_route\_mask}),\ \text{v\_dst\_net\_addr}) \wedge\\
&\text{equal}(\ \text{ipand}(\text{v\_ip\_host},\text{v\_netmask}),\ \text{ipand}(\text{v\_ip\_dst},\text{v\_netmask}))\\
&\qquad\qquad\qquad\qquad \Rightarrow\\
&\text{SendIPPacket}(\text{send\_ip\_packet}(\text{v\_host},\\
&\qquad \text{v\_ip\_dst},\text{ippacket}(\text{v\_ip\_src},\text{v\_ip\_dst},\text{v\_ip\_proto},\text{v\_ip\_payload})))
\end{aligned}
$$

Transport: TCP Packets



tcppacket(source_port, dest_port, flags, data)

Encapsulated TCP Packet

max planck institut informatik

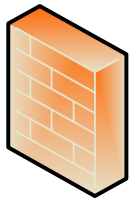
epacket(seg1, r1_mac_net1, c1_mac, e_ip,

ippacket(c1_ip, s1_ip, tcp

tcppacket(c1_port, s1_port, syn, 0)))

Transport: Firewall

max planck institut informatik



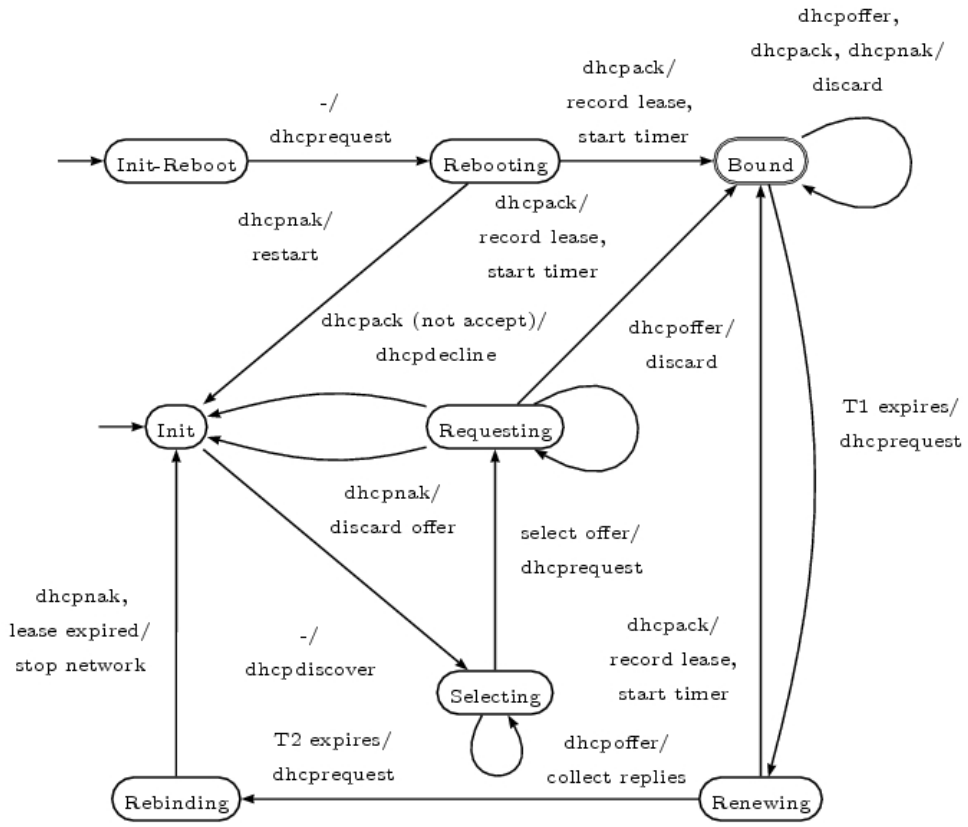
- Receives IP Packets
- Tests Source, Destination addresses and Ports
- Tests correct Use of Protocols: TCP, UDP, …
- If Test succeeds: Forwards the Packet
- Test is build out of a Rule List

Firewall: One Case

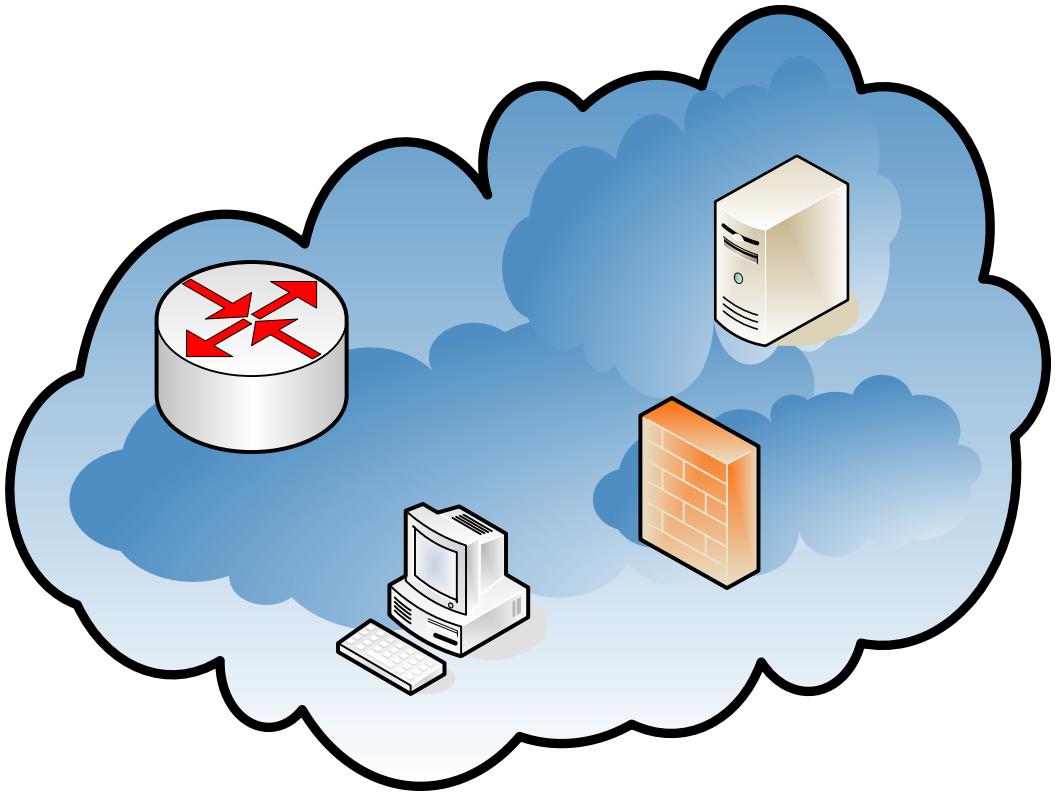
max planck institut informatik

$$
\begin{aligned}
&\text{Sent}(\text{epacket}(\text{incoming\_net}, \text{dst\_mac}, \text{src\_mac}, \mathbf{e\_ip},\\
&\qquad \text{ippacket}(\text{ip\_src}, \text{ip\_dst}, \mathbf{tcp},\\
&\qquad\quad \text{tcppacket}(\text{rule\_tcp\_src\_port}, \text{rule\_tcp\_dst\_port}, \mathbf{syn}, \text{tcp\_data}\ )\ )\ )\ )\ \wedge\\
&\text{Firewall}(\text{firewall}(\text{fw},\text{incoming\_net},\text{outgoing\_net}))\ \wedge\\
&\text{TCP\_FirewallRule}(\text{tcp\_fwrule}(\text{fw}, \mathbf{first\_rule}, \mathbf{permit}, \text{rule\_ip\_src\_addr},\\
&\qquad \text{rule\_ip\_src\_mask}, \text{rule\_ip\_dst\_addr}, \text{rule\_ip\_dst\_mask},\\
&\qquad \text{rule\_tcp\_src\_port}, \text{rule\_tcp\_dst\_port})\ )\ \wedge\\
&\text{equal}(\text{ipand}(\text{ip\_src},\text{rule\_ip\_src\_mask}), \text{rule\_ip\_src\_addr}\ )\ \wedge\\
&\text{equal}(\text{ipand}(\text{ip\_dst},\text{rule\_ip\_dst\_mask}), \text{rule\_ip\_dst\_addr}\ )\\
&\qquad\qquad\qquad\qquad\qquad\qquad \Rightarrow\\
&\text{Sent}(\text{epacket}(\ \text{outgoing\_net}, \text{dst\_mac}, \text{src\_mac}, \mathbf{e\_ip},\\
&\qquad \text{ippacket}(\text{ip\_src}, \text{ip\_dst}, \mathbf{tcp},\\
&\qquad\quad \text{tcppacket}(\text{rule\_tcp\_src\_port}, \text{rule\_tcp\_dst\_port}, \mathbf{syn}, \text{tcp\_data}\ )\ )\ )\ )\ \wedge\\
&\text{TCP\_ClientSynSent}(\text{tcp\_clientsynsent}(\text{fw}, \text{ip\_src}, \text{ip\_dst},\\
&\qquad \text{rule\_tcp\_src\_port}, \text{rule\_tcp\_dst\_port}\ )\ )\ )
\end{aligned}
$$

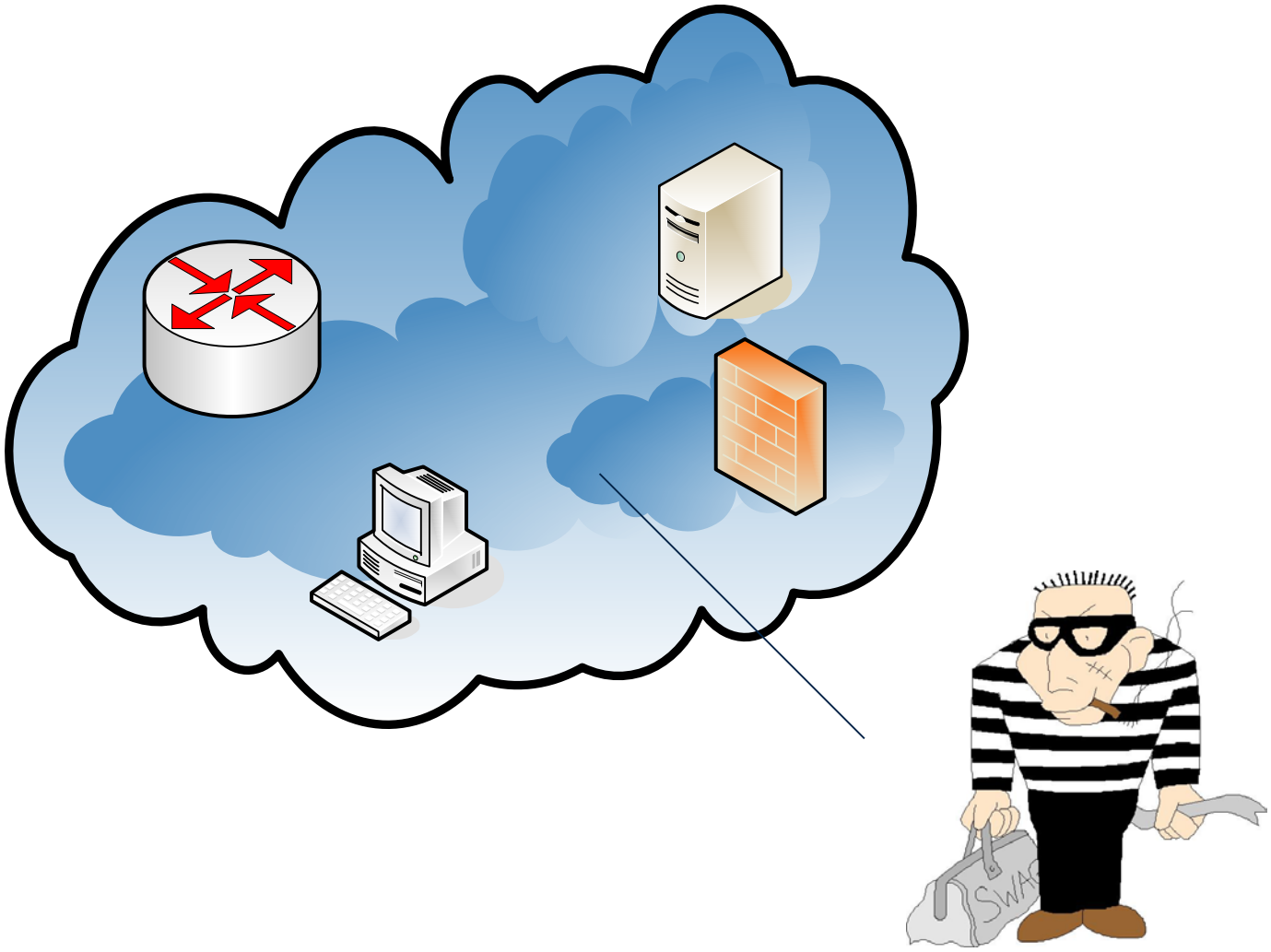
DHCP: Client State Automaton



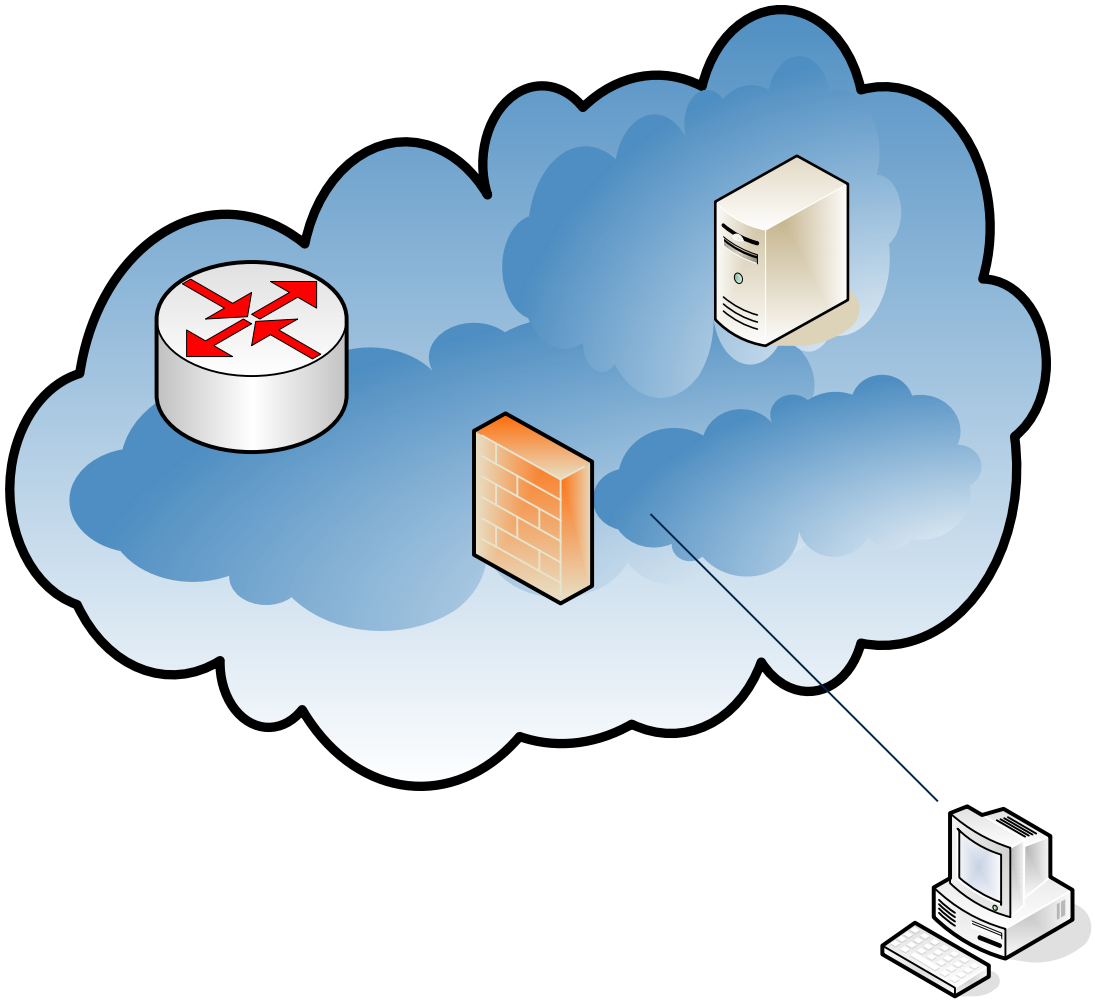
Properties: Fault Analysis



Properties: Intruder



Properties: Client



Results of the Case Study

max planck institut informatik

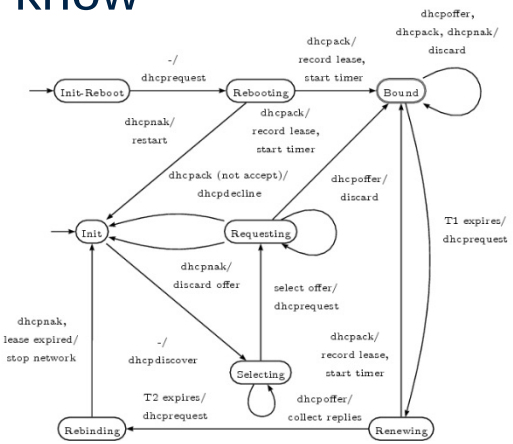
- Overall Theory ~150kB
- DHCP correct
- Firewalling
- Model Scales: We Finitely Saturated the Overall MPI-INF & MPI-SWS Network Configuration
- Example:

  2 Router (40 interfaces, 500 ARP, 150 Routes) +

  2 Firewalls (3000 rules) = 30 min saturation

Why does it work?

- message sequences are finite
- beautiful fixed ordering
- sufficient to consider redundancy at variable level
- no recursive redundancy needed
- I know



470[0:Res:55.0,157.1] || Leaseoffered(leaseofferedtupel(r1,u,v)) ReceivedUDPPacket(recv_udp_packet(net2,r1,w,ip(0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,0),udppacket(port67,port67,dhcpdata(broadcast,one,x,ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),w,v,dhcpoptions(DHCPdiscover,ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),null)))))* Hostentry(hostentrytupel(r1,y,v,u,z,x1))* -> SkP1(r1_mac_net2,ip(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0),x1,z,u,y,v,x,ip(0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,0),w,r1,net2)*.

subsumes

642[0:Res:303.0,157.1] || Leaseoffered(leaseofferedtupel(r1,u,v)) ReceivedUDPPacket(recv_udp_packet(net2,r1,w,ip(0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,0),udppacket(port67,port67,dhcpdata(broadcast,one,x,ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),w,v,dhcpoptions(DHCPdiscover,ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),ip(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),null)))))* Hostentry(hostentrytupel(r1,y,v,u,z,x1))* -> SkP1(r1_mac_net2,ip(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0),x1,z,u,y,v,x,ip(0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,0),w,r1,net2)*.

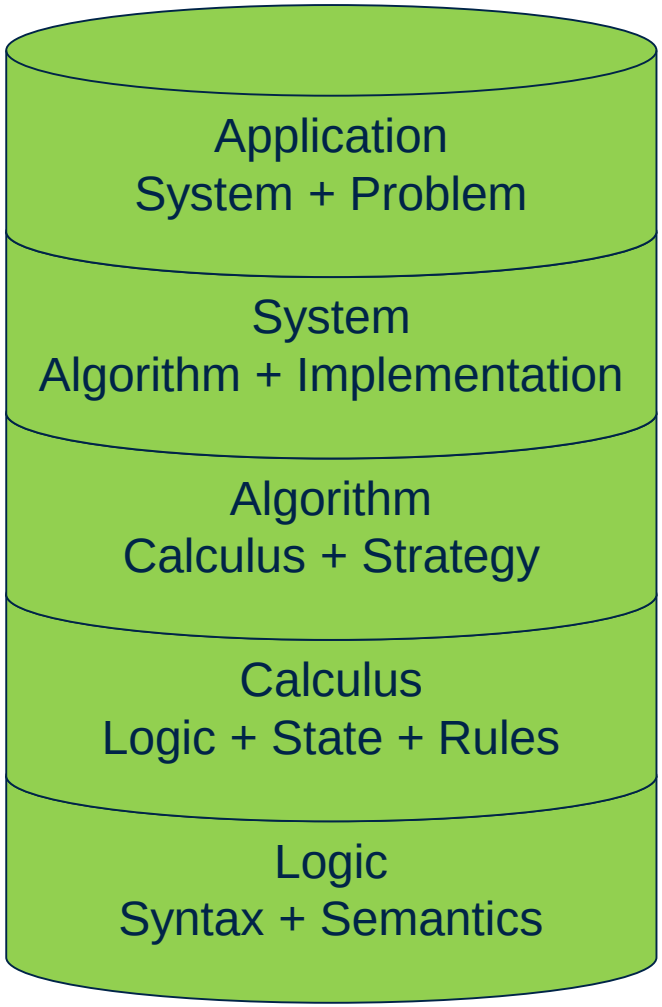
Extending the Model

- Time/Metrics - Arithmetic
- Buffering – Data Structure
- Finite Sets – Finite Domains
- Inductive Properties – Induction by Saturation
- Changes – Abduction
- 64-Bit IP addresses: linear overhead

The Practical AR World

max planck institut informatik

The AR Tower



The END