PART TWO:

CONNECTION PATHS

AND INTEROPERATION

*a formal model in Alloy, with all the details*

*this lecture does not use DFC*
CONNECTION PATHS

This is a connection path between two clients.

Which clients, servers, and addresses are in it? Why?
AS A USER, WHAT I LIKE ABOUT ALLOY

**LANGUAGE: RELATIONAL + PREDICATE LOGIC**

there is no distinction among relations, sets, and scalars

nominally first-order, effectively second order (this says that all translation relations are partial functions)

fact \{ all \(a: \text{Address}, f: \text{InteropFeature} \mid \)

lone \(a \cdot (f \cdot \text{interTrans}) \)

\}

the \text{interTrans} attribute of \(f \)

image of a under this relation

contains one or zero elements

very few operators, used uniformly to be powerful and flexible

quantifiers \text{no}, \text{one}, \text{lone}, \text{some}

by the way, this could also have been declared

\text{interTrans}: \text{Address} \rightarrow \text{lone Address}

**ANALYSIS**

within a limited scope (number of objects of each basic type):

1) find an instance of a formula

- this is essential because it is very easy to write a logic model that is inconsistent

2) check that there are no counterexamples to an assertion

- if it works, you have a lazy person's proof

- if it fails, the counterexample shows you what is wrong

- in all cases, the instances are very illuminating, even essential to getting the model right

**THE STREAMLINED SYNTAX MAKES WRITING LOGIC A PLEASURE**
What phenomena are being constrained?

They are environment-controlled

They are network-controlled

What is the status of the constraints?

They are indicative ("knowledge", "assumptions")

They are optative ("desired", "hypothetical")

Shared (visible on both sides)

World

Architecture

Requirements

Specification

How the formal models are divided into parts
CONNECTIONS (INDICATIVE)

agent (capable of participating in connections)

port (belongs to one agent)

hop (a port participates in at most one role in at most one hop)

link (a port participates in at most one role in at most one link)

all these ports are connected to each other

there is no notion of insufficient ports or other reasons for unavailability
THE ENVIRONMENT AND THE NETWORK

ENVIRONMENT-CONTROLLED AND SHARED

NETWORK-CONTROLLED AND UNSHARED

client

external port

server

internal port

these ports are observably connected to each other (they can observe the connection directly by sending data/media to each other)
the ports and hops are partitioned into administrative domains

**DOMAINS (INDICATIVE)**

sig Domain { 
  eports: set ExternalPort, network-controlled, shared
  space: set Address, network-controlled, unshared
  iports: set InternalPort, network-controlled, unshared
  map: space -> Agent
}


domains and hops (indicative)

sig Domain {
    ports: set Port,
    space: set Address,
    map: space -> Agent
}

• a hop is routed to an agent that its target address maps to:

  fact {  all h: Hop, g: Agent  |
      h.acceptor in g.ports =>
      g in (h.target).(h.domain.map)  }

• if the initiator port of a hop belongs to a client, the source address of the hop maps to the client in the hop's domain

• a client has ports in a domain (one or more) if and only if addresses (one or more) map to that client in that domain

• if an address maps to a client in a domain, it does not map to any other agent in that domain
**INTEROPERATION FEATURES (OPTATIVE)**

```
sig InteropFeature {
    servers: set Server,
    domain, toDomain: Domain,
    exported, imported,
        returning, returned: set Address,
    interTrans:
        exported some -> some imported
} { domain != toDomain &&
    exported in domain.space &&
    imported in toDomain.space &&
    returning in exported &&
    returned in imported &&
    returning.interTrans = returned
}
```

- Each server implements one interoperation feature.
- A foreign address in a domain must map to one or more servers of one or more interoperation features to whose `returning` set it belongs.
- If a server is the acceptor of a hop in its `domain` with target address in its `returning` set, then the hop is linked to a corresponding hop in its `toDomain`.
- `returned` is the image of `returning` under `interTrans`. 
ADDRESS EXAMPLES

BoxOS Internet Overlay Network

pstn:11234567890

boxos:bob@white

sip:alice@black

SIP Internet Overlay Network

sip:11234567890?user=phone

Public Switched Telephone Network

sip:alice@black

11234567890
INTEROPERATION REQUIREMENTS (OPTATIVE)

REACHABILITY

If an external port is requesting a connection to an address in a domain, and if the address in the domain is known as a way of reaching the client, then that port is observably connected to a port of that client.

```
sig Client extends Agent
{ knownAt: Address -> Domain }
```

NON-AMBIGUITY

If two external ports are both requesting a connection to the same address in the same domain, then either neither is observably connected, or both are observably connected to ports of the same client.

RETURNABILITY

If two external ports are observably connected (one is an initiator and one is an acceptor), and if in the acceptor's domain an agent is requesting a connection to the source of the first connection, then the second connection reaches the same client that initiated the first connection.
Satisfying Non-Ambiguity and Reachability

**Non-Ambiguity**

- **Constraint 1**
  
  \[
  \text{fact \{ all } a : \text{Address}, f : \text{InteropFeature} | \text{lone } a.(f.\text{interTrans}) \}\]

  An address translation is a partial function.

- **Constraint 2**
  
  \[
  \text{fact \{ all } a : \text{Address}, d : \text{Domain} | \text{lone } f : \text{InteropFeature} | \text{f.domain} = d \&\& \text{a in f.returning} \Rightarrow \text{(a.(f.\text{interTrans}) -> f.toDomain) in c.reachedBy} \}\]

  An address triggers at most one interoperation feature.

**Reachability**

- **Constraint 3**
  
  \[
  \text{fact \{ all } c : \text{Client} | \text{c.knownAt in c.reachedBy} \}\]

  ReachedBy contains the ways to reach a client.

\[(a->d) \text{ in c.reachedBy iff (some (d.ports & c.ports) \&\& c in a.(d.map)) || (Foreign(a,d) \&\& (all f : \text{InteropFeature} | f.domain = d \&\& \text{a in f.returning} \Rightarrow \text{(a.(f.\text{interTrans}) -> f.toDomain) in c.reachedBy}) )}\]

Real constraint has mechanism to ensure a least fixed point.

Evidence from extensive use of the Alloy analyzer suggests that:

**World \& ARCHITECTURE \& SPECIFICATION**

\[\Rightarrow \text{Requirement}\]

Includes: there are only interoperation servers, performing only interoperation functions.

Includes only the constraints needed for the particular requirement.
Satisfying Returnability

Source addresses are preserved

-- Constraint 4
fact { all f: InteropFeature |
    f.domain.space in f.exported}

translation is invertible

-- Constraint 6
fact { all f1, f2: InteropFeature |
    PartnerTo(f1,f2) =>
    (f1.interTrans).(f2.interTrans) in iden
}

Pred PartnerTo (f1,f2: InteropFeature)
{  f2.domain = f1.toDomain &&
    f2.toDomain = f1.domain &&
    (f1.imported - f1.returned) in f2.returning)

-- Constraint 5
fact { all f1: InteropFeature |
    some f2: InteropFeature |
    PartnerTo(f1,f2) }

Every interoperation feature has a partner


GLOBAL UNIQUENESS OF ADDRESSES
IS NOT NECESSARY

c0.reachedBy = 
{ (a0->d0),
  (a1->d1),
  (a2->d2) }
c1.reachedBy = 
{ (a2->d0),
  (a0->d1),
  (a1->d2) }
c2.reachedBy = 
{ (a1->d0),
  (a2->d1),
  (a0->d2) }
if \( ax[a2] \) is an address, then interoperation translation is a function, not one-to-many. However, it is not invertible (Constraint 6), so there is no guarantee of returnability.

SOLUTION TO THE PROBLEM:

- Note that funny addresses are never used as sources.
- Rewrite Constraints 5 and 6 to allow for a special class of targetOnly addresses.
- Use Analyzer to establish that Constraints 5' and 6' also guarantee returnability.
OTHER POSSIBLE APPLICATIONS OF THIS APPROACH

CONNECTION SERVICES

- features that manipulate abstract addresses (as opposed to concrete addresses, which map to clients)
- connection graphs (the generalization of connection paths)
- infrastructure for mobility
- features for directory lookup
- address-related aspects of security

NETWORK ROUTING

*Internet routing is "staggeringly complex"

reachability must take into account (at least):

- dynamic distribution of routing information by BGP and others
- multiple hops within a domain
- packet filters, packet transformers

current work includes:

- high-level, informal proofs
- static analysis of configuration files, including recursive computation of reachability
- simulation

there is room for formal modeling of principles and their consequences