ALLOY AND THE FUTURE OF NETWORKING

(MAYBE)

Pamela Zave

Princeton University

Princeton, New Jersey

THIS IS
OUR MANIFESTO

The compositional architecture of the Internet

Pamela Zave
Princeton University
Princeton, New Jersey
pamela@pamelazave.com

Jennifer Rexford
Princeton University
Princeton, New Jersey
jrex@cs.princeton.edu

ABSTRACT

Contrary to the "classic" Internet architecture familiar to most people, today's Internet is a composition of a wide variety of networks. The IP protocol suite offers a general-purpose network design with a widely available implementation; as such, it is re-used to design and implement networks with many different purposes. Compositional architecture explains how, despite the fact that IP has not changed significantly since 1993, the Internet has evolved to meet many new requirements and challenges since then. In this paper we introduce a new and principled model for describing Internet architecture, and give many examples of its validity. We also explain how the model can help us facilitate innovation through interoperation and evolution, re-use successful solution patterns, and verify trustworthy network services.

ACM Reference format:
Pamela Zave and Jennifer Rexford. 2018. The compositional architecture of the Internet. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 8 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In 1992, the explosive growth of the World Wide Web began. The architecture of the Internet was commonly described as having four layers above the physical media, each providing a distinct function: a "link" layer providing local packet delivery over heterogeneous physical networks, a "network" layer providing best-effort global packet delivery across autonomous networks all using the Internet Protocol (IP), a "transport" layer providing communication services such as reliable byte streams (TCP) and datagram service (UDP), and an "application" layer. In 1993 the last major change was made to this "classic" Internet architecture [10]; since then the scale and economics of the Internet have precluded further changes to IP [11].

A lot has happened in the world since 1993. The overwhelming success of the Internet has created many new uses and challenges that were not anticipated by its original architecture:

- Today, most networked devices are mobile.
- There has been an explosion of security threats.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

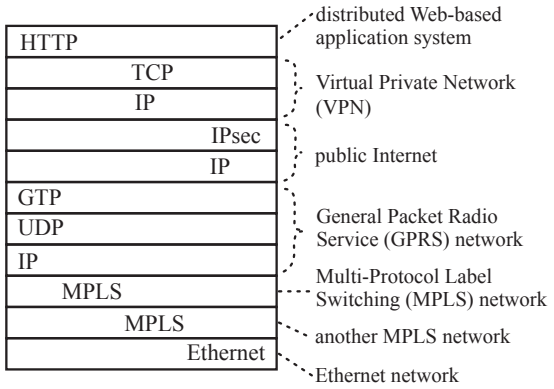


Figure 1: Headers of a typical packet in the AT&T backbone network. Headers lower in the diagram are outermost in the actual packet.

- Most of the world's telecommunication infrastructure and entertainment distribution has moved to the Internet.
- Cloud computing was invented to help enterprises manage the massive computing resources they now need.
- The IPv4 32-bit address space has been exhausted, but IPv6 has not yet taken over the bulk of Internet traffic.
- In a deregulated, competitive world, network providers control costs by allocating resources dynamically, rather than provisioning networks with static resources for peak loads.

Here is a conundrum. The Internet is meeting these new challenges fairly well, yet neither the IP protocol suite nor the way that experts describe the Internet has changed significantly since 1993. Figure 1 shows the headers of a typical packet in the AT&T backbone [19], giving us clear evidence that the challenges have been met by mechanisms well outside the limits of the classic Internet architecture. In the classic description, no headers below HTTP except those labeled "public Internet" and "Ethernet network" would exist.

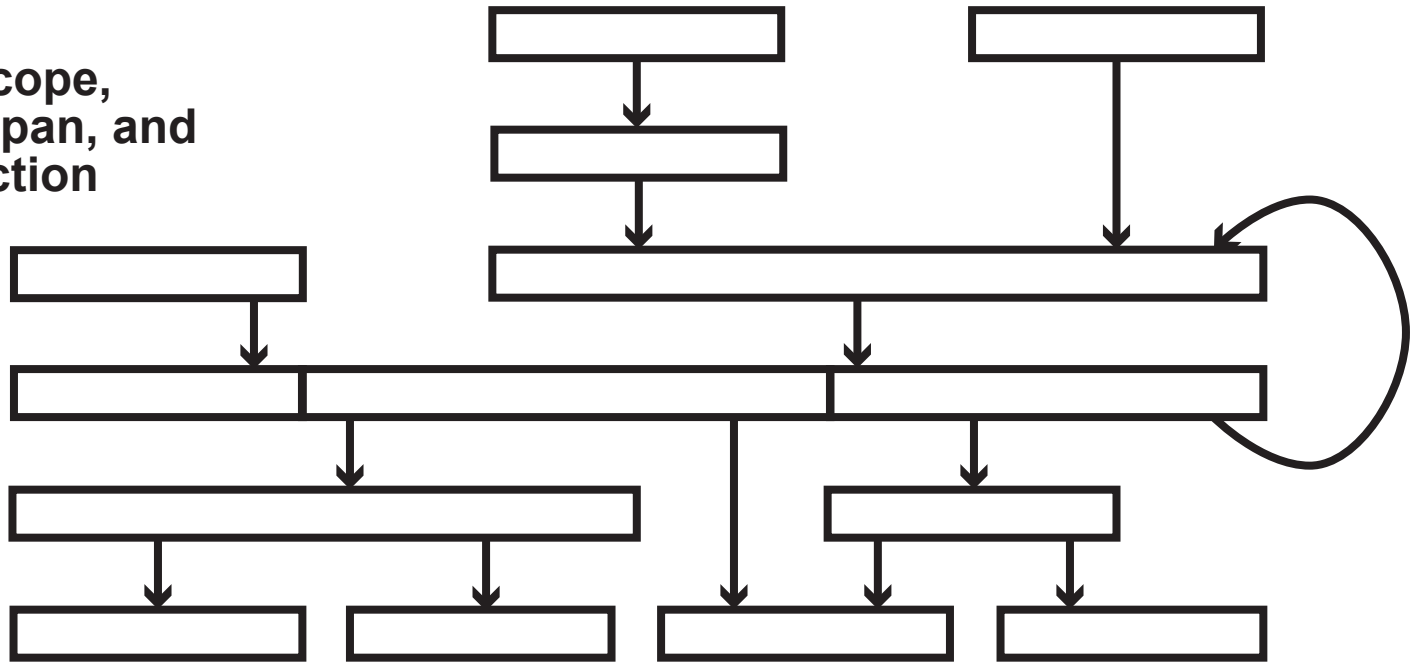
In this paper we will present a new way of describing the Internet, better attuned to the realities of networking today, and to meeting the challenges of the future. Its central idea is that the architecture of the Internet is a flexible *composition* of many networks—not just the networks acknowledged in the classic Internet architecture, but many other networks both above and below the public Internet in a hierarchy of abstraction. For example, the headers in Figure 1 indicate that the packet is being transmitted through six networks below the application system. Our model is as principled as the one it replaces, so that we are not reduced to grappling with masses of

THE INTERNET IS A FLEXIBLE COMPOSITION OF MANY NETWORKS

each network has all the same basic mechanisms, . . .

. . . but in each network they are specialized for a particular . . .
. . . purpose,
. . . membership scope,
. . . geographical span, and
. . . level of abstraction

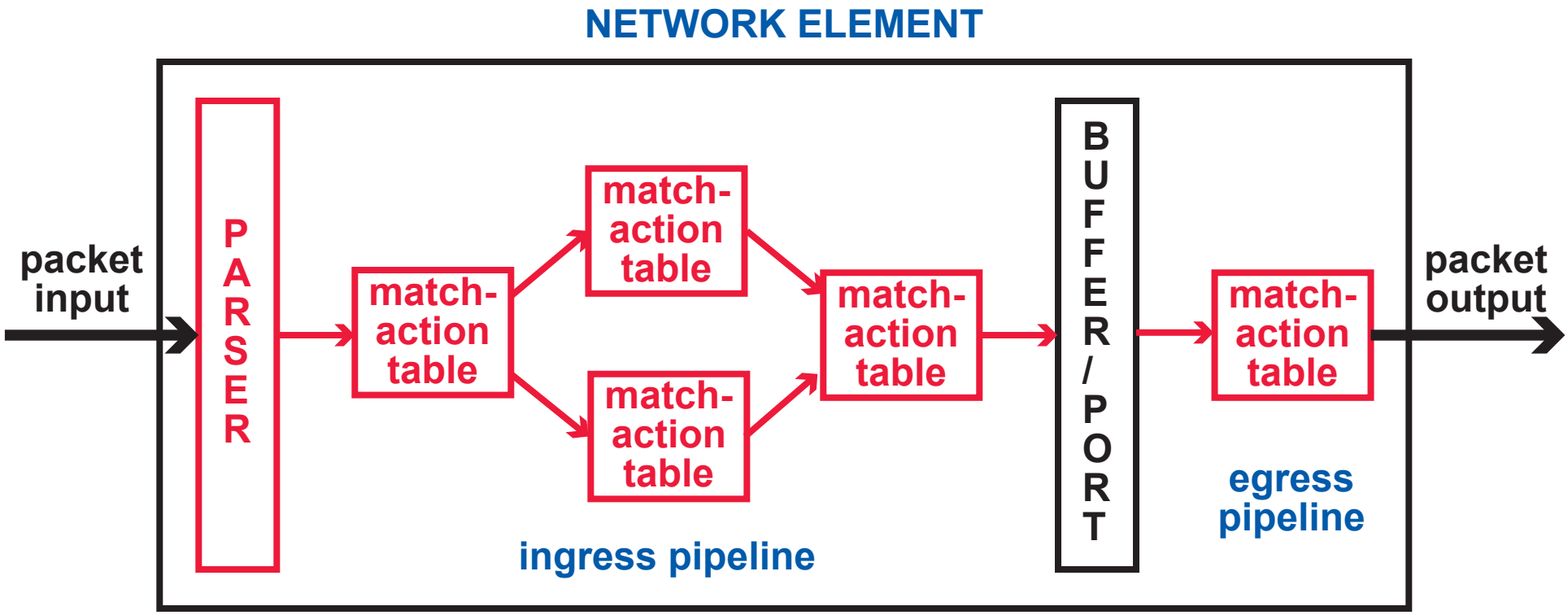
because all networks have fundamental similarity, they all have common interfaces for composition



in other words, networking can be made modular, and the module is a network

our need:
useful formal models to support this work
(definition, design, analysis, verification, transformation, code generation, etc.)

P4: PROGRAMMING PROTOCOL-INDEPENDENT PACKET PROCESSORS



the red parts are coded in P4, compiled to software or hardware

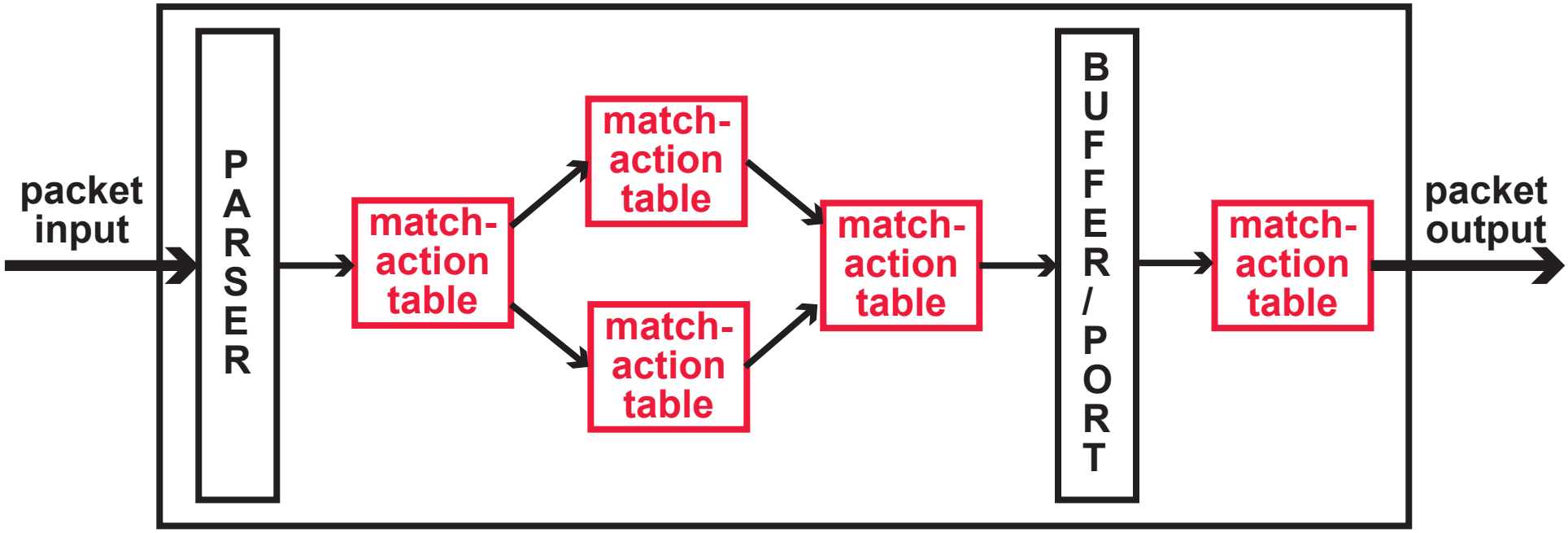
actions generate meta-data for a packet, modify packets, choose output port for a packet

Why do we like P4?

- by networking standards it is a high-level language
- P4 is cutting-edge research with an active community
- plausibly, we can use P4 to describe generic packet processing for the compositional model

A P4 PROGRAM IS NOT COMPLETE . . .

BECAUSE IT DOES NOT CREATE OR MAINTAIN MATCH-ACTION TABLES



forwardingTable: headers -> inPort -> actions -> outPort

reaches: headers -> nodes -> nodes

generalized P4 programs as operational specifications of network semantics

+

Alloy models of static tables, their properties, and their semantic consequences

=

plausible formal models of composable and composed networks

HAVE I MENTIONED THAT NETWORKING IS REALLY, REALLY, COMPLEX?

CAN ONLY ANALYZE A SMALL PIECE OF IT IN ONE MODEL . . . SO I WANT FAMILIES OF MODELS, AND SOME AUTOMATED HELP WITH DERIVATION AND CONSISTENCY

not an original thought, but . . .
. . . I have great examples
. . . I am willing to use a restricted style

auth.als

tree.als

basic.als ..... **a very general, basic model of forwarding**

unique.als

HAVE I MENTIONED THAT NETWORKING IS REALLY, REALLY, COMPLEX?

CAN ONLY ANALYZE A SMALL PIECE OF IT IN ONE MODEL . . . SO I WANT FAMILIES OF MODELS, AND SOME AUTOMATED HELP WITH DERIVATION AND CONSISTENCY

not an original thought, but . . .
. . . I have great examples
. . . I am willing to use a restricted style

adds source names in packets and authentication of source names

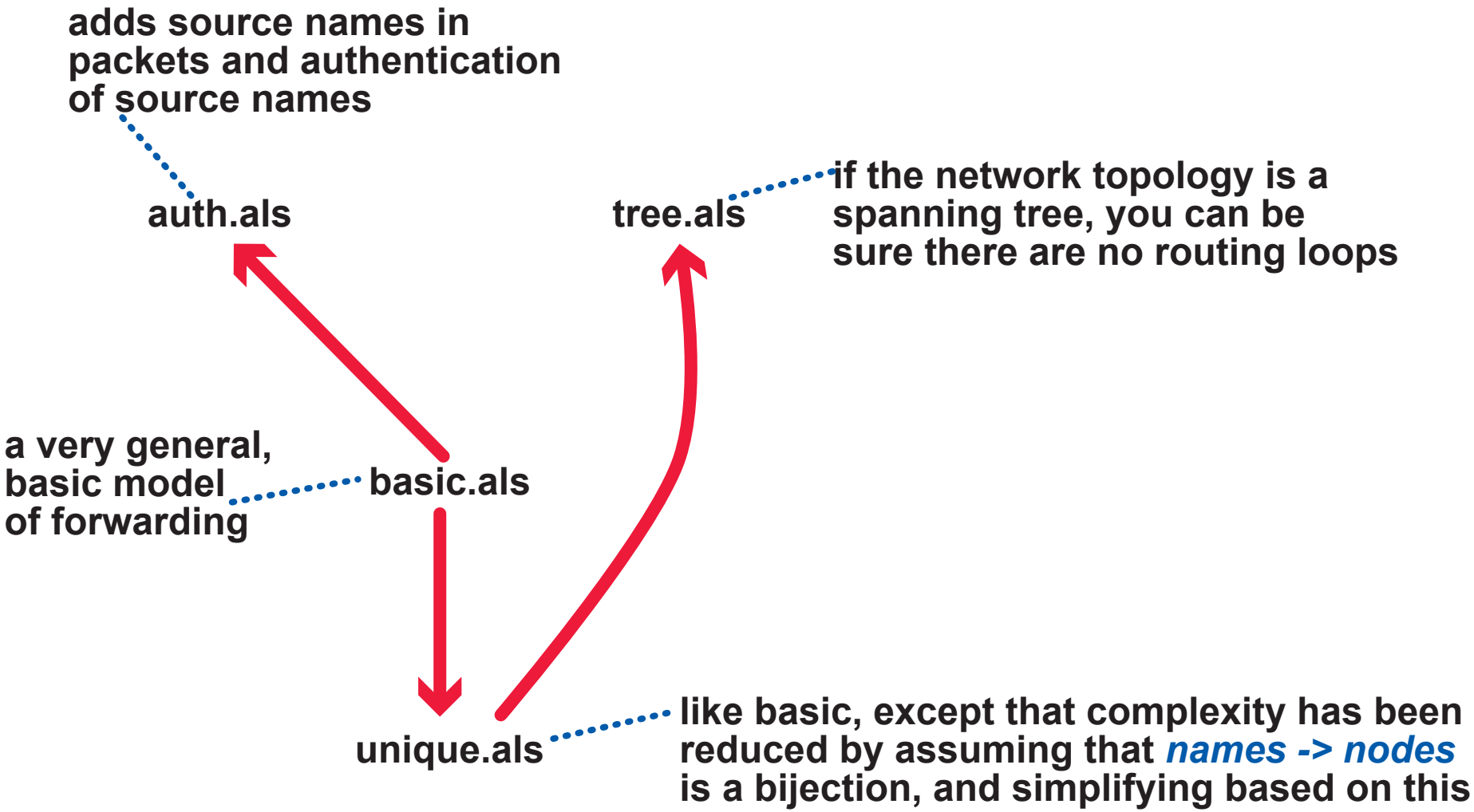
auth.als

tree.als — **if the network topology is a spanning tree, you can be sure there are no routing loops**

basic.als — **a very general, basic model of forwarding**

unique.als — **like basic, except that complexity has been reduced by assuming that *names -> nodes* is a bijection, and simplifying based on this**

THIS IS WHAT I HAVE DONE

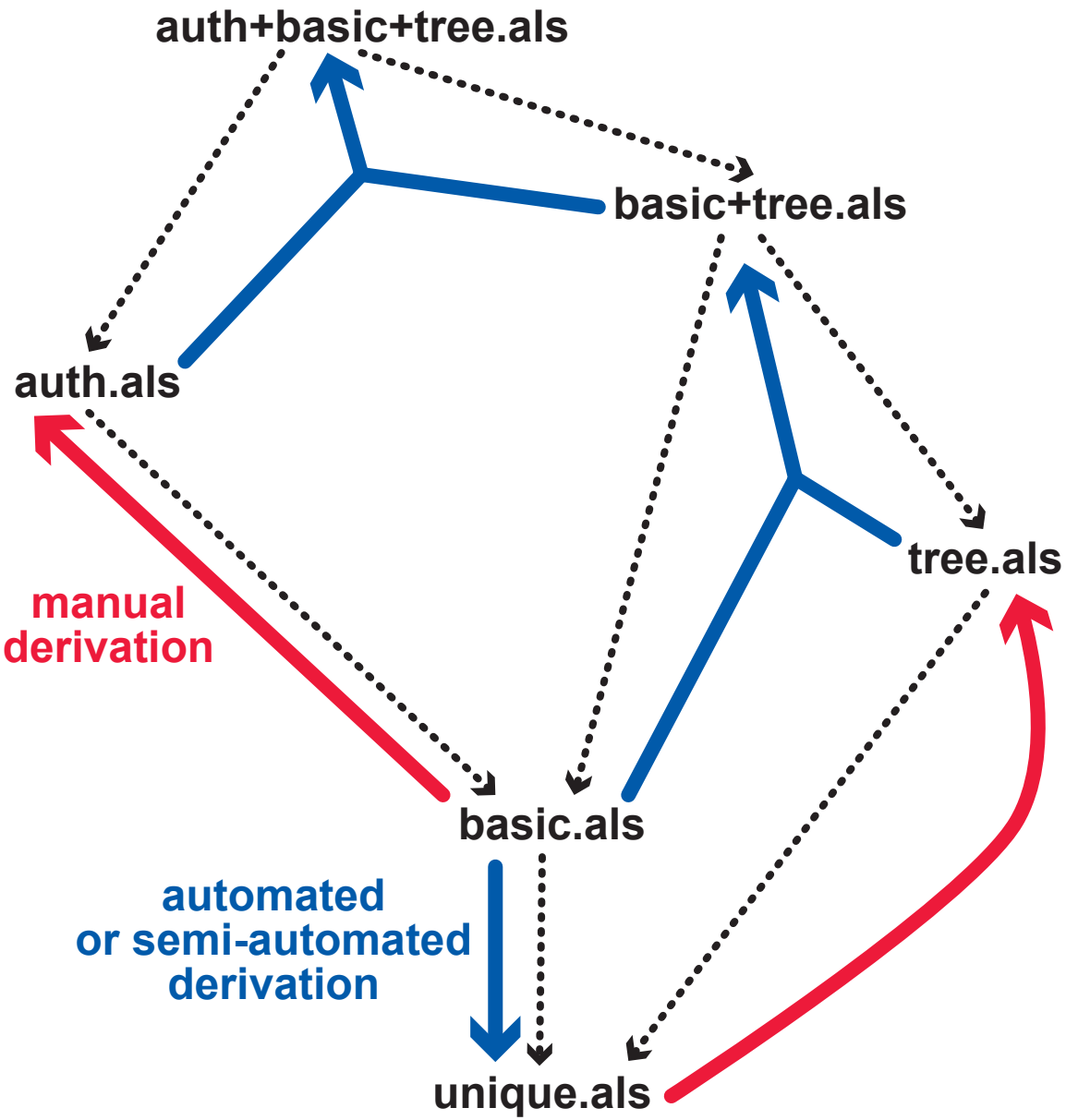


the models may not look complex, but debugging them has been really hard

frequently, I find an improvement by working on one model, then have to go back and apply the change to all the other models

THIS IS WHAT I DREAM OF:

A LATTICE WHOSE PARTIAL ORDER IS "IS AN EXTENSION OF"



DERIVATION OF UNIQUE.ALS FROM BASIC.ALS

basic.als

```
sig Name, Node { }

sig Network {
    nodes: set Node,
    names: set Name,
    name: nodes -> names,
    . . .
}
```

name is an indirect way of referring to a node

unique.als

```
sig Node { }

sig Network {
    nodes: set Node,
    . . .
}
```

TO DERIVE:

- drop “sig Name,” “names” and “name” fields
- make substitutions:
  - Name → Node
  - names → nodes
  - name → identity function on Node
- simplify expressions
- remove tautologies

DERIVATION OF BASIC+TREE.ALS

tree.als extends unique.als

basic and minor sigs, facts

sig Network {

many fields }

{ *constraints on fields* }

predicates for named, interesting properties like
ls_spanning_tree_network [w: Network]

take the union;

where named entities have different declarations or definitions, use the version from basic.als because it is more general than unique.als

pred Good_network_exists [w: Network]

{ *all named predicates* }

run Good_network_exists

assertions and checks of theorems

the types of Header.destination, positiveReach, negativeReach have changed;

when theorems from tree.als using these relations, substitute names for nodes

and the theorems are still valid!

ASK ME ABOUT . . .

ALLOY:

THE MISSING MANUALS

ALLOY: THE MISSING MANUALS

ALL ABOUT SOLVERS

- which solver should I use?
- how should I structure my model —which has a big invariant—to get useful unsat cores?

ALL ABOUT SEQUENCES

- where do I find the libraries for the three (I think) ways of modeling sequences?
- how do the three compare with respect to notational simplicity, contraindications, equality tests, and other gotchas?
- is it true that if a model manipulates sequences, it must instantiate all sequences?
- could I please have a short but complete example (include files, sigs, etc.) for each style, so I don't have to figure it all out again each time?