
Adam D. Bradley
with Azer Bestavros and Assaf J. Kfoury
iBench Initiative
Protocol Verification

- How do we know that a protocol is correct?
  - It will not misbehave, e.g. deadlock?

- Popular Option: Finite-State Model Checking
  - Model agents as FSAs
  - Possibilistic analysis of model (inter)actions, searching for violations of stated invariants
  - Explores all corner cases
  - Intuitively useful results: Viewable models, auditable event sequences leading to violations
FSM Checking: Models

/* Promela model of HTTP/1.0 (per RFC1945) with RFC2068 keepalive
   w.r.t. Expectation and Continuation
   (C) 2002 Adam D. Bradley, All Rights Reserved
*/

proctype server_rfc1945(chan upstream, downstream) {
  /* HTTP/1.0 server implementing 1.1 keepalive */
  xr upstream;
  xs downstream;

  reg reqdata;
  resp respdata;
  respdata.version = RESP_HTTP_10;
  respdata.hasentity = true;
  do
    :: upstream?(request, reqdata) ->
      if
        :: reqdata.hasentity ->
          do
            :: upstream?(entitypiece, reqsink) -> skip;
            :: upstream?(entityend, reqsink) -> break;
            :: upstream?(eof, reqsink) -> /* client tanked */
              goto shutdown;
            :: BAD_CLIENT_EVENT(upstream, response);
            :: BAD_CLIENT_EVENT(upstream, continue);
          // leave upstream?request enqueued
          od
        :: else -> skip;
      fi;
    send_general_server_response();
    if
      :: respdata.close -> break;
      :: else ->
        fi;
    :: upstream?(eof, reqsink) -> break; /* client tanked */
    :: BAD_CLIENT_EVENT(upstream, entitypiece);
    :: BAD_CLIENT_EVENT(upstream, entityend);
    :: BAD_CLIENT_EVENT(upstream, continue);
    :: BAD_CLIENT_EVENT(upstream, response);
  od;

  shutdown:
    downstream!eof(0);
}
FSM Checking: Composition

2 Agents

3 Agents

4 Agents
Every failure of the model maps directly to an “Exemplar” of model execution.

What Happened?
Where was the failure?
What events preceded it?
Problem: How do we generalize to “For all $\mathbb{N}$”?
Consider an indirection-based protocol (i3, an HTTP transaction, etc)
- 1 client, 0 or more intermediaries, 1 server
- How do we ensure that all cases are correct?

Conventional FSM: Can’t do it.
- Check 0, 1, 2, ..., N intermediaries, then stop.
- Are there emergent behaviors at N+1?
- Do failure “modes” recur in larger arrangements?
Infinite-State Techniques

- Translate problem into a “regular” $\infty$-space
  
  [Kucera and Jancar 2002], [Chechik et al 2001],
  [Leuschel and Massart 2000], [Dingel and Filkorn 1995],
  [Jackson 1994]

- Failure cases can be difficult to translate back into concrete examplars

- Techniques not yet “accessible” to protocol/systems hackers
The **CHAIN** System

- **Canonical Homomorphic Abstraction of Infinite Network** protocol compositions

  - Leverage “Term Rewriting” to extend the reach of finite-system/finite-state verification tools to some infinite problem spaces
Algebra represents sequences of agents (A)

- Infinite set of strings, defined however you like
- “Regular sets” preferable - easier to manipulate (regular expressions, grammars)
CHAIN Anatomy

- Algebra represents sequences of agents \( (A) \)
- Property of interest \( (\pi) \)
  - e.g.: Deadlock-safe, schedulable, convergence...
  - Can be computed for any single composition
  - Usually a decision algorithm \( \text{(true/false)} \)
  - CHAIN is agnostic to the computation method...

Model Checking
Type Checking \([\text{Chaki+Rajamani 2002}]\)
Worst-Case Performance Values
Control-Theoretic Analysis
Algebra represents sequences of agents (A)

Property of interest ($\pi$)

Rewrite rules (R) over A which preserve $\pi$

- "If you connect two agents of type $X$, then their peers will be unable to distinguish them from a single $X$.": $a \ X \ X \ b \triangleright a \ X \ b$ (for all $a, b$)

- Rewriting to multiple strings also allowed:
  $a \ X \ b \triangleright \{ a \ Y, \ Z \ b \}$

  which means: $\pi(a \ X \ b) = \pi(a \ Y) \&\& \pi(Z \ b)$

  (greater flexibility in devising rewrite rules, greater difficulty formulating a complexity bound)
- Algebra represents sequences of agents (A)
- Property of interest (π)

- Rewrite rules (R) over A which preserve π
  - Any rule deduction technique is acceptable

Any technique establishing invarianct w.r.t. π
*(depends upon proof method of π)*

I/O Equivalence (NP-Complete)
Type Checking [Chaki+Rajamani 2002]
Algebra represents sequences of agents (A)
Property of interest ($\pi$)
Rewrite rules (R) over A which preserve $\pi$

Each $R_i$ “reduces” the search space to $A_i \subset A$

- $A_i$ is a “Homomorphic image”: The “shape” of the behavior of all strings in A are represented by strings in $A_i$
- e.g., if $R_n=\text{“a } \text{X X b }\Rightarrow\text{ a X b”}$,
  then $A_n=\text{“all A with no } \text{X X’ substring”}$
- if $R_n=\text{“a } \text{X b }\Rightarrow\{\text{a Y, Z b}\}$”,
  then $A_n=\text{“all A with no } \text{X’ substring”}$
Algebra represents sequences of agents \( A \)
- Property of interest \( (\pi) \)
- Rewrite rules \( (R) \) over \( A \) which preserve \( \pi \)
- Each \( R_i \) “reduces” the search space to \( A_i \subset A \)
- With a little luck, \( \cap A_i \) defines a finite set
  - If \( \cap A_i \) is not finite, its cycles may identify “target spaces” which could be searched for additional valid reductions
**CHAIN Anatomy**

- Algebra represents sequences of agents ($A$)
- Property of interest ($\pi$)
- Rewrite rules ($R$) over $A$ which preserve $\pi$
- Each $R_i$ "reduces" the search space to $A_i \subset A$
- With a little luck, $\bigcap A_i$ defines a finite set
- Algebra represents sequences of agents (A)
- Property of interest (\( \pi \))
- Rewrite rules (R) over A which preserve \( \pi \)
- Each \( R_i \) “reduces” the search space to \( A_i \subset A \)
- With a little luck, \( \bigcap A_i \) defines a finite set

\[ \bigcap A_i \text{ (aka } A_\perp) \]: Finite Homomorphic Image of A

- Compute \( \pi \) for every member of that set
- This captures the “shape” of the behaviors of every one of the (infinitely many) members of A
CHAIN Anatomy

- Algebra represents sequences of agents (A)
- Property of interest (\(\pi\))
- Rewrite rules (R) over A which preserve \(\pi\)
- Each \(R_i\) “reduces” the search space to \(A_i \subset A\)
- With a little luck, \(\bigcap A_i\) defines a finite set
- \(\bigcap A_i\) (aka \(A_\perp\)): Finite Homomorphic Image of A

- Prove that R \textit{terminates} and is \textit{confluent}
  - Every member of A has a single “canonical form” which represents it behaviors w.r.t. \(\pi\)
Algebra represents sequences of agents \((A)\)

Property of interest \((\pi)\)

Rewrite rules \((R)\) over \(A\) which preserve \(\pi\)

Each \(R_i\) “reduces” the search space to \(A_i \subseteq A\)

With a little luck, \(\bigcap A_i\) defines a finite set

\(\bigcap A_i\) (aka \(A_\perp\)): Finite Homomorphic Image of \(A\)

Prove that \(R\) terminates and is confluent

Why? Closing “gaps” in confluence may reveal additional rewrite rules which further shrink \(A_\perp\)
Example: HTTP Deadlocks

- **RFC2068 Flaw: 100 Continue Deadlock**
  - RFC2616 fixes most cases \([BBK'02]\)

- **A = Set of “Arrangements” of HTTP Agents**
  
  \[ A = CP^* S \]

- **\(\pi\) = Property: Deadlock-Safe or Not**
  
  \[ \forall a \in A, \pi(a) \in \{\text{true}, \text{false}\} \quad \text{(computed using SPIN)} \]

- **R = Rewrite Rules: 7 in [BBK'02], 3 in [BBK'03]**
  
  \[ R_1 : x \ P0 \ y \triangleright \{x \ S0, C0 \ y\} \]
  
  \[ R_5 : x \ P2 \ P2 \ y \triangleright \{x \ P2 \ y\} \]
Finding the Finite Set

- Each Rewrite Rule “Removes” a Subset of $A$:
  \[ R_1 \mapsto A_1 = A - C \ P* \ P0 \ P* \ S \]
  \[ R_5 \mapsto A_5 = A - C \ P* \ P2 \ P2 \ P* \ S \]

- “Sufficient Subset”: intersection of $A_1 \ldots A_{10}$:
  \[ A_\perp = A_1 \cap \ldots \cap A_{10} = \]
  \[ |A_\perp| = 29 \]

- $R$ terminates, confluent
- Rewriting is canonical
MPEG Overlay Routing

- Active drop algorithm: discard MPEG frames you know are worthless [He, Muller, Lawall 2002]
- Composing such agents w/ each other and w/ normal network behaviors (drop, reordering) causes non-worthless packets to be dropped.
MPEG Overlay Routing

- **Rewrites:**
  - **Primitive Normalizations**
    - “Any sequence of reord and drop nodes” $\Rightarrow$ “drop reord”
    - “mrouter drop” $\Rightarrow$ “mrouter”
    - Not sufficient to produce finite image
  - **Other reductions involving mrouter:**
    - As specified, no others were readily apparent
    - Defined rewrite rules as “Goals” for alternative algorithms:
      - e.g., “mrouter mrouter” $\Rightarrow$ “mrouter”

- Gives rise to $|A_\perp|=6$
Intra-Cache Consistency (BTC) \([BB02]\)

- Completely Non-FSM approach
  - Rewrite proofs follow directly from caching and clock-vector def’n
- 20 rewrite rules based upon purely “local” properties produces \(A_\perp\) with 2 members: one “consistent”, one “may be inconsistent”
- Local proofs give rise to a simple global pattern rule to determine whether an arrangement is inconsistency-prone \(A_\perp\)
What about more involved structures?

- We have only addressed linear compositions
- What about trees?
  - Content Distribution Networks/Cache networks
  - Multicast routing
- What about more general graphs?
  - P2P “clouds”? Collections of routers?
Non-Linear CHAIN

- There are well-developed notions of “tree/graph rewriting” (the “algebraic” component)

- Can sufficient meaningful “rewrite rules” be deduced for interesting applications?
Conclusions

- **CHAIN** supports characterizing infinitely large protocol compositions with finite computation
  - Intuitively accessible tool
  - Agnostic w.r.t. particular verification technologies
    - Any technique can be used to verify members of the minimal set
    - Any technique can be used to establish equivalence/reductions
  - Failure modes provided in “native” FSM form
  - Confluence of rewrite rules ensures every arrangement reduces to a single *canonical form* which is behaviorally representative