Invariant Discovery for Networked Systems

Hongyu Hè*
Princeton University

Alexander Krentsel*
UC Berkeley / Google

Sylvia Ratnasamy
UC Berkeley / Google

Maria Apostolaki
Princeton University

Abstract

Invariants, the relations expected to hold among measured signals of a network, underpin applications from verification to traffic generation, telemetry imputation, and input validation. Yet, writing them by hand demands rare expertise in both formal logic and networking. Testing all possible combinations of telemetry variables statistically is intractable at network scale, while existing automatic tools still require a human-authored grammar of admissible invariants and learn only exact, "hard" rules that break under real-world noise. Our key idea is to use an LLM to inspect variable metadata, classify counter data types, and predict candidate syntactic relationships that form a compact grammar, followed by a deterministic statistical search within that grammar. If the initial grammar fails to recover trusted known invariants on held-out telemetry, Autogram instructs the LLM to iteratively widen the grammar until recovery succeeds. By confining the LLM to grammar induction and re-induction while delegating candidate verification to logical and statistical checks, Autogram turns non-deterministic AI proposals into auditable invariants with formal guarantees within the induced grammar. We design and prototype Autogram and evaluate it on both public and production WAN telemetry, recovering expert-derived invariants with full recall and low false positives. We close with a discussion of open problems on the path toward fully open-ended invariant discovery.

CCS Concepts

• **Networks** → **Network manageability**; *Network monitoring*; • **Computing methodologies** → *Machine learning*; • **Theory of computation** → *Automated reasoning*.

*Equal contribution.



This work is licensed under a Creative Commons Attribution 4.0 International License.
HotNets '26, Salt Lake City, UT, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3029-0/2026/11
https://doi.org/10.1145/3847290.3859952

Keywords

Network invariants, invariant discovery, network telemetry, large language models, grammar induction, automated reasoning

ACM Reference Format:
Hongyu Hè, Alexander Krentsel, Sylvia Ratnasamy, and Maria Apostolaki. 2026. Invariant Discovery for Networked Systems. In *The 25th ACM Workshop on Hot Topics in Networks (HotNets '26), November 16-17, 2026, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3847290.3859952

1 Introduction

Network operators rely on relationships that hold between measurements in a network snapshot at any given time, known as *network invariants*, for a variety of applications. These invariants capture operational rules such as packet conservation across a router, where a node's total output equals the sum of its outgoing links. Formal verifiers check invariants to find misconfigurations or faults [5, 7, 19, 27, 48, 58], traffic and workload generators follow them to maintain realism [13, 28, 33–35, 54], telemetry imputers and runtime guards use them to fill missing values and protect learning-based controllers [21, 22, 26, 29], and data-pipeline validators check them to catch corrupt inputs before they spread [39, 40, 57]. Making these relationships machine-checkable has caught corrupted controller inputs behind major WAN outages [39], corrected drifting agricultural and IoT sensors [52, 55], and flagged industrial process or capacity anomalies before equipment failed [18, 36].

Deriving usable invariants for a production network remains difficult and labor-intensive. Writing them by hand demands expertise in both formal logic and the target system. Even when experts know the expected invariants, real measurements carry collection errors, systematic biases, and implementation bugs that require manual data analysis to characterize [39]. In production, the invariants that matter are usually *soft*, meaning they hold within a tolerance across most observations.

Automatic invariant mining promises to learn these relationships directly from telemetry [23, 28, 59, 60], yet a fundamental scaling barrier stands in the way. A real network tracks too many variables to search through every possible combination of counters using statistical techniques alone.

To make the search tractable, existing approaches require a human expert to first author a *grammar* that restricts which variable types can be grouped and compared. Even when an expert writes that grammar by hand, existing approaches look for exact rules and miss relationships that are masked by measurement noise (§2).

A grammar shrinks the search space by grouping variables into types and specifying which syntactic relationships can connect those types. Consider a flow conservation rule I_1 at a router X, which states that total output approximately equals the sum of traffic on its outgoing links, $\text{out}_X \approx \texttt{egress}(X \to Y) + \texttt{egress}(X \to Z)$. Expressing this rule requires four building blocks in the grammar: (1) *types* so packet counts never mix with byte counts; (2) *locality groups* that collect all links leaving router X; (3) an n-ary *sum* whose size adapts to the router's degree; and (4) a *soft comparison* that permits a bounded tolerance. Raw counter values do not expose these building blocks, so prior systems rely on human experts to define the grammar for each dataset.

Defining a grammar requires domain intuition because raw variable names often omit structural context. For instance, standard OpenConfig interface counters do not encode topological roles directly, so systems such as CrossCheck rely on human annotations to map counters to network structure [39]. Across tabular knowledge bases and databases [31, 41], software systems [16], and networks [28], a human has always defined the space of candidate rules in advance before an algorithm searches within that space. We observe that LLMs can now supply this domain intuition from variable names and basic metadata to classify data types and predict which types relate to one another. Relying on an LLM, however, raises an immediate question: since LLMs are stochastic and can hallucinate, how can invariants discovered with their help remain reproducible and trustworthy?

Our key idea is to use the LLM exclusively to construct and refine the *grammar*, while delegating all candidate verification to deterministic logical and statistical checks. Asking an LLM to propose individual invariants directly produces isolated guesses with no defined hypothesis space; even if a downstream solver and statistical verifier checked each guessed rule for soundness, the search would still lack exhaustiveness and completeness, and its recall would degrade as the network grows. Having the LLM define a grammar quantified over routers and links instead restricts the search space in a single step, which allows downstream checks to search that smaller space exhaustively across all network entities. Because the LLM never proposes or filters individual candidate rules, three guarantees hold for whatever grammar is active (§3): the downstream search is *deterministic and exhaustive*, evaluating every candidate in the grammar identically on every run; *statistically sound*, accepting only rules that a solver proves non-trivial and that exceed a strict hold-rate threshold on the data; and *monotone complete*, guaranteed to recover any expressible invariant that holds at least as often, on at least as much data, as a known invariant the operator already trusts.

We instantiate this design in Autogram, where an LLM induces a bounded, typed grammar from variable names and operator metadata such as topology, role, and vendor annotations, and iteratively widens that grammar if known baseline invariants are missed. A deterministic engine then enumerates the candidate rules in the grammar, a logical solver prunes trivial and contradictory forms, and a statistical verifier tests how consistently each surviving rule holds on telemetry. To prevent false positives when testing thousands of candidates, Autogram fits each rule's tolerance on a separate data split, demands high agreement across snapshots, and sets its minimum hold-rate threshold high enough to reject coincidental relationships on randomly shuffled data.

Running Autogram on public WAN benchmarks and a large production backbone recovers every invariant previously written by experts and uncovers additional conservation and symmetry invariants that boost downstream validation (§4). Two open boundaries frame our broader research agenda (§5): composing discovered invariants into multi-hop relational structures [27], and extending verification beyond the decidability limits of linear solvers into non-linear queueing and buffer-sharing invariants [3, 48, 66].

2 Background and Motivation

What a soft invariant is. A *soft invariant* is a typed relationship over named variables that holds across most snapshots of a network's data within a bounded tolerance. Each soft invariant therefore carries a hold-rate between zero and one. The relationships span numeric conservation invariants where a router's output equals the sum of its outgoing links, magnitude bounds where a queue length stays within its dynamic switch buffer threshold [1–3, 48, 66], and logical presence pairings where a route is advertised or selected only when its next-hop interface and policy path are active [6, 9, 11]. A soft invariant acts as an executable runtime check that returns a pass-or-fail verdict, a numeric residual, and a calibrated hold-rate on any snapshot. Hard invariants correspond to the special case where the hold-rate equals one at zero tolerance. Every soft invariant remains a readable symbolic formula that operators can inspect and monitors, validators, or runtime controller shields can evaluate on live telemetry [26, 29].

Why soft invariants are worth discovering. Soft invariants serve as checkable contracts that keep production systems reliable, and discovering more of them directly strengthens downstream data analysis. In a wide-area backbone, an

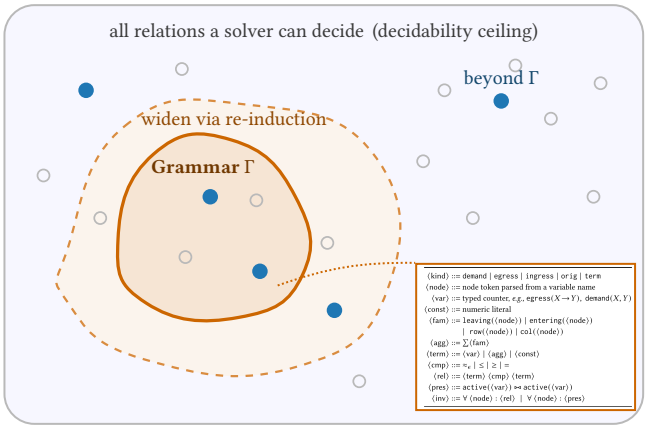


Figure 1: The grammar defines a *space*. An LLM proposes a bounded region Γ inside the universe of all relations a solver can decide, and AUTOGRAM searches only *within* Γ, so a grammar *confines* the hypothesis space rather than enlarging it; the inset previews the concrete grammar detailed in Table 1. The data decides which points hold (• invariants, ∘ candidates), and when known invariants fall outside Γ, re-induction widens it (dashed) but never past the decidability ceiling.

SDN controller computes traffic engineering allocations from an aggregated demand matrix, and validating that matrix against local router counters catches corrupted inputs that cause major outages [39]. Downstream validators such as CrossCheck grow more effective as they check more independent invariants, because each additional conservation or symmetry rule tightens the constraints on noisy measurements and sharpens fault localization [39]. In a multi-tenant telemetry pipeline, the conservation invariant input = output + backlog + loss separates persistent packet loss from transient incast bursts caused by datacenter and distributed machine-learning workloads [24, 64, 65]. Similarly, correlated sensor invariants catch drifting agricultural and IoT sensors before they corrupt downstream models [52, 55], and operational invariants flag routing, capacity, or control anomalies before service degrades [10, 18, 30, 36]. Production invariants are soft for two distinct physical reasons. First, zero-mean measurement noise scatters residuals symmetrically around zero, which reflects aleatoric uncertainty [15, 32, 37]. Second, systematic bias shifts a conservation invariant by a few percent at every router, for example when interface counters include packet headers that demand matrices omit. Both behaviors provide operational value: a near-exact rule acts as a tight correctness check, while a consistent offset diagnoses an accounting discrepancy across platforms.

The grammar is the real bottleneck. Defining the *grammar* of candidate rules is the central bottleneck in invariant discovery because the grammar determines which relationships can be expressed and keeps the search space small enough to check (Fig. 1). A real network exposes too many variables to test all possible combinations of counters without structural guidance. A useful grammar shrinks that search space by identifying what each variable measures, grouping related variables into *locality groups* such as all links leaving a router, allowing n-ary sums over those groups, and specifying which data types can be compared. Constructing those groupings requires domain intuition about what the variable names mean. Raw counter values do not reveal that two egress counters belong to the same outgoing link group, and hand-crafting a new grammar for every telemetry schema and network does not scale.

Where prior methods fall short. Existing tools require a human to define the search space by hand and struggle when measurements contain noise. Network verifiers check whether a network satisfies a *given* property [7, 19, 38, 58, 62], so they rely on someone else to supply the invariant first. Classical rule-learning approaches fix the candidate templates in advance: database dependency approaches search predetermined relational forms [8, 17, 31, 47], software tools such as Daikon test a static library of patterns [16], and protocol inference approaches infer boolean predicates over discrete state against an exact oracle [59, 60]. The closest network invariant approach, NetNomos [28], requires an expert-written grammar for each counter set and accepts only rules that hold on 100% of snapshots without supporting group sums or soft tolerances. Other statistical paradigms solve different problems: symbolic regression fits a function for a single target variable [12], association-rule mining links discrete items [4, 53], and Markov logic learns weights over fixed predicate schemas without per-rule tolerances [50]. Recent LLM-guided search systems such as FunSearch and AlphaEvolve ask an LLM to edit candidate programs inside an evolutionary loop scored by a known objective function [42, 44, 51]. Those systems query the LLM for every candidate and emit opaque code, whereas invariant discovery lacks a ground-truth scoring function and needs readable formulas that a solver can verify. Having the LLM construct the grammar defines the entire space of readable candidates in a handful of calls while keeping the LLM out of candidate verification.

Why symbolic invariants complement predictive models. Discovering explicit symbolic invariants serves a different purpose than training a black-box model to predict telemetry values. An invariant paired with a hold-rate forms an auditable contract that operators can read and reuse across verification [27, 48], input validation [39], runtime safety enforcement [26, 29], and synthetic trace generation [28, 34].

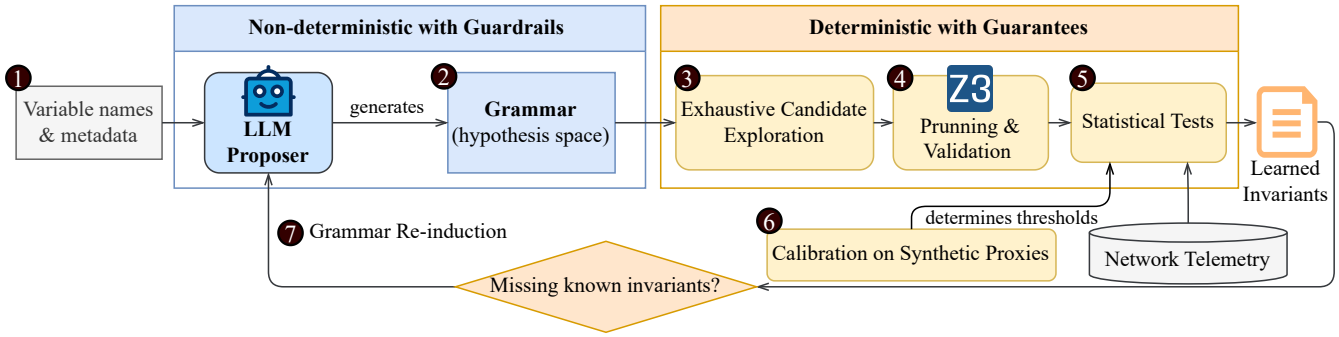


Figure 2: AUTOGRAM separates invariant discovery into an LLM stage that proposes a *grammar* of candidate relationship templates from variable names and metadata (❶, ❷), and a deterministic verification pipeline driven by formal logic and statistics. Downstream, AUTOGRAM enumerates all candidates in the grammar (❸), prunes trivial or redundant rules with a solver (❹), and accepts soft invariants that exceed a data-calibrated hold-rate threshold (❺). Generic thresholds are calibrated on synthetic proxies and shuffled null controls (❻). If recall on a held-out split of known invariants stalls, AUTOGRAM asks the LLM for a wider grammar (❼). Raw telemetry reaches only the evaluator.

A black-box predictor absorbs measurement noise and systematic offsets into learned weights, which hides accounting bugs inside the model. A symbolic soft invariant surfaces both the relationship and its residual offset, turning a two-percent gap between demand matrices and router counters into an explicit diagnostic. Achieving that clarity requires controlling false positives without ground-truth labels while isolating the LLM's stochastic behavior so that downstream guarantees hold (§3).

3 Network Invariant Discovery

AUTOGRAM separates the single step that requires domain intuition from the downstream computation that requires formal and statistical rigor, as illustrated in Fig. 2.

Problem formulation. AUTOGRAM takes a dataset of network counters and measurements observed across many snapshots, paired with a small set of baseline invariants that the network operator already trusts. The system outputs a portfolio of verified soft invariants over those variables, where each invariant carries an empirical hold-rate and a tolerance band. Combining grammar induction with per-candidate tolerance fitting and synthetic proxy calibration allows AUTOGRAM to discover soft invariants directly from noisy telemetry, where prior approaches retain only rules that hold on every snapshot [23, 28, 59, 60].

Only one step needs world knowledge. Deciding which variable types can be grouped and compared is the only step that requires domain intuition, and that decision defines the grammar (❷). The LLM reads the variable names and metadata ❶ and outputs a typed grammar of candidate templates without ever asserting that a specific invariant holds, as shown by the induced BNF grammar Γ in Table 1. For example, given counter names for router origination traffic, link egress traffic, and demand pairs, the LLM classifies variable kinds ($\langle$kind$\rangle$), groups variables by router X into locality groups ($\langle$group$\rangle$) such as outgoing links `leaving`(X) and demand rows `row`(X), and permits comparing router totals against group sums ($\langle$agg$\rangle$). Because the LLM only shapes the search space by translating metadata into candidate templates [27], a model hallucination cannot inject an unverified invariant into the final output.

$\langle$kind$\rangle$	::= `demand` | `egress` | `ingress` | `orig` | `term`
$\langle$node$\rangle$	::= node token parsed from a variable name
$\langle$var$\rangle$	::= typed counter, *e.g.*, `egress`$(X \to Y)$, `demand`(X, Y)
$\langle$const$\rangle$	::= numeric literal
$\langle$group$\rangle$	::= `leaving`($\langle$node$\rangle$) | `entering`($\langle$node$\rangle$)
	| `row`($\langle$node$\rangle$) | `col`($\langle$node$\rangle$)
$\langle$agg$\rangle$	::= $\sum\langle$group$\rangle$
$\langle$term$\rangle$	::= $\langle$var$\rangle$ | $\langle$agg$\rangle$ | $\langle$const$\rangle$
$\langle$cmp$\rangle$	::= $\approx_\epsilon$ | $\leq$ | $\geq$ | $=$
$\langle$rel$\rangle$	::= $\langle$term$\rangle$ $\langle$cmp$\rangle$ $\langle$term$\rangle$
$\langle$pres$\rangle$	::= `active`($\langle$var$\rangle$) $\bowtie$ `active`($\langle$var$\rangle$)
$\langle$inv$\rangle$	::= $\forall\,\langle$node$\rangle : \langle$rel$\rangle$ | $\forall\,\langle$node$\rangle : \langle$pres$\rangle$

Table 1: Grammar Γ of the phrase structure for admissible invariants that AUTOGRAM induces for the Cross-Check telemetry data [39]. Making locality groups (`leaving`, `row`), n-ary aggregation ($\sum$), soft equality ($\approx_\epsilon$ within a tolerance ϵ read from the data), and presence pairing ($\bowtie$) first-class productions adaptively is what lets AUTOGRAM express complex invariants that existing approaches like NetNomos [28] cannot.

Metadata is a rich, shareable signal. Operator metadata such as variable names, topological roles, and vendor labels provides a compact signal for constructing the grammar ❶ [41]. Relying on declarative metadata interfaces brings two practical advantages for production deployments [25]. First, raw telemetry values never leave the operator's premises during LLM queries, avoiding the source-level privacy leakage that affects even synthesized network traces [35]. Second, because the induced grammar depends on variable names rather than measurement values, the same grammar transfers across deployments that share a telemetry schema and can be re-evaluated across multiple threshold settings after a single LLM query ❷.

Balancing grammar quality through induction and re-induction. A high-quality grammar must balance two competing goals: it must stay constrained enough to keep the search space small and fast to enumerate, while remaining expressive enough to capture the true relationships that hold in the data. Widening the grammar captures more possible invariants, yet that expansion also increases the number of candidates to check. To manage this tension, AUTOGRAM uses a single fixed prompt template across all datasets to request a compact initial grammar ❷, and validates that every proposed group and type in Table 1 maps cleanly to actual dataset columns before touching the measurements. We then evaluate grammar expressiveness empirically by checking whether enumeration recovers known baseline invariants on held-out data; if recovery stalls, AUTOGRAM prompts the LLM to widen the grammar with additional aggregations or operators ❼.

An invariant's tolerance is written in its residuals. Genuine soft invariants exhibit different noise levels and systematic offsets (§2), so AUTOGRAM estimates each candidate's tolerance directly from its empirical residuals. For a candidate relation $A \approx_\epsilon B$ such as $\forall X : \texttt{orig}(X) \approx_\epsilon \sum \texttt{row}(X)$ from Table 1, AUTOGRAM computes the dimensionless relative residual on each snapshot t as

$$v_t = \frac{|A_t - B_t|}{\max(|A_t|, |B_t|, s_0)},$$

where $s_0 > 0$ is a small scale floor, with analogous one-sided violation residuals for $\lesssim_\epsilon$ and $\gtrsim_\epsilon$. When the candidate relationship holds on the network, the relative residuals v_t on most snapshots stay near zero, followed by a sharp jump to a few noisy outliers. To avoid overfitting the error cutoff ϵ to the test data [43, 45, 56], AUTOGRAM splits snapshots into a 70% calibration set and a 30% held-out test set. On the 70% calibration set, AUTOGRAM sorts the residuals $v_{(1)} \le \ldots \le v_{(m)}$, finds the sharpest jump $i^\star = \arg\max_i (v_{(i+1)} - v_{(i)})$ between the tight cluster and the outliers, and sets the error cutoff to $\epsilon = \min(v_{(i^\star)}, \epsilon_{\max})$ before measuring the hold-rate on the 30% test set as the fraction of snapshots where $v_t \le \epsilon$ ❺. Fitting ϵ from the data captures noisy invariants such as origination demand conservation at a 68.6% hold-rate in Table 2, while the global cap $\epsilon_{\max} = 0.05$ prevents unrelated variables from choosing a wide cutoff to manufacture agreement.

Logical pruning and statistical validation. After enumerating all candidates in the grammar ❸, AUTOGRAM applies formal logic, calibrated statistics, and algebraic dominance pruning to filter the candidate pool without relying on an LLM. First, an SMT solver [14] screens out trivially true candidates such as $\sum \texttt{leaving}(X) \le \sum \texttt{leaving}(X)$, self-contradictory formulas, and rules entailed by a simpler accepted invariant ❹. Second, the statistical verifier checks whether each surviving candidate agrees within its tolerance band across a large majority of held-out snapshots ❺. Third, to suppress uninteresting algebraic variations among statistically valid candidates, a Pareto archive collapses Z3-equivalent rearrangements into a single canonical representative, discards any rule logically entailed by a shorter accepted rule, and prunes weakened derivatives such as scaled-slack inequalities $A \le c \cdot B$ with $c > 1$ or multi-term one-sided bounds whenever a tighter approximate equality $A \approx_\epsilon B$ or atomic sign bound already holds. Using an LLM to propose or filter individual candidates would forfeit exhaustiveness and monotone completeness because the LLM could omit valid forms or non-deterministically drop true invariants before verification. Exhaustive search over a grammar keeps false-positive risk low across M candidates because the verifier requires a large hold-rate separation rather than testing a zero-effect null at $p < 0.05$, where M independent tests would produce $0.05M$ false positives on average. Given k passes out of n held-out observations ($\hat{p} = k/n$), AUTOGRAM computes the one-sided $(1-\alpha)$ Wilson lower confidence bound $L_\alpha(k, n)$ at $\alpha = 0.05$ and requires it to clear a high hold-rate threshold $\theta \ge 0.62$ ($z_{0.05} = 1.645$):

$$L_\alpha(k,n) = \frac{\hat{p} + \frac{z_\alpha^2}{2n} - z_\alpha \sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z_\alpha^2}{4n^2}}}{1 + \frac{z_\alpha^2}{n}} \ge \theta.$$

For unrelated noisy counters, the baseline probability p_{null} of falling within a capped 5% band is only a few percent ($p_{\text{null}} \ll 0.10$). By Hoeffding's inequality and a union bound over all M candidates, the probability of even one false positive is at most $M \exp(-2n(\theta - p_{\text{null}})^2)$, which stays tiny for $M \sim 10^5$ and $n \ge 1{,}000$ snapshots. Fitting tolerances on the 70% split prevents in-sample overfitting on the 30% test split, and threshold calibration ❻ runs the full grammar against synthetic and time-shuffled null controls to reject any setting that admits a false rule on noise.

Grammar-conditioned guarantees. Because stochasticity is confined to grammar construction, three guarantees hold relative to whatever grammar is active. Determinism

and exhaustiveness: given a fixed grammar and dataset, Autogram enumerates every candidate template identically on every run ❸ and retains a canonical representative of each accepted equivalence class, making every run reproducible and auditable. Statistical soundness: every reported invariant is proven non-trivial by the solver, evaluated on out-of-sample data splits, and required to stay within the calibrated hold-rate confidence bound across all entity groups. Monotone completeness: the statistical verifier applies the same hold-rate threshold to every candidate and remains blind to which rules are already known to operators. Consequently, when the acceptance threshold is calibrated so that a trusted known invariant passes, any other rule in the grammar that holds at least as often across at least as many snapshots is guaranteed to pass as well.

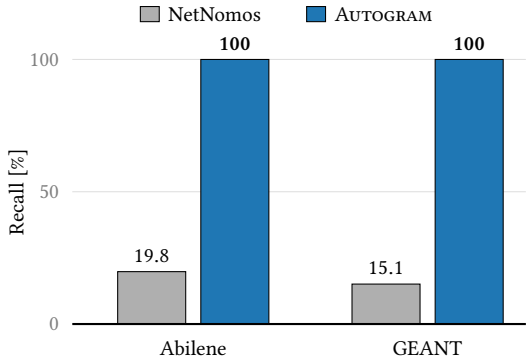


Figure 3: Autogram recovers every invariant on both networks, whereas NetNomos recovers barely a fifth because it keeps only rules that hold exactly.

4 Preliminary Results

An SDN controller can reject a corrupted traffic-engineering input only when it knows which relationships a valid snapshot must satisfy, and discovering more of those invariants strengthens validation across the network [39]. We use Autogram to discover these invariants directly from WAN telemetry, using our experiments to test how grammar induction and soft statistical validation perform on noisy operational data.

Setup. We evaluate Autogram on two public WAN telemetry datasets, Abilene and GEANT, packaged as CrossCheck snapshots [39], and on a large production backbone we call "WAN prod." In this CrossCheck case study, each snapshot includes an aggregated traffic-engineering *demand matrix* alongside per-router *interface counters* measured in the dataplane. A demand matrix is specific to this cross-layer validation case study and is not required by Autogram: when given only standard router interface counters, the exact same prompt template and pipeline still discover two-end link agreement, router flow conservation, counter non-negativity, and link presence symmetry. Our evaluation ground truth is an expert-compiled catalog of invariant categories, where CrossCheck originally used only two-end agreement and flow conservation while leaving non-negativity, zero self-demand, topology presence, and demand conservation unstated. We compare against NetNomos [28], a state-of-the-art approach that we supply with a hand-written grammar requiring over an hour of expert effort. Autogram uses the same fixed starting prompt across all three networks, taking only the dataset's counter names to induce the grammar once per network.

Insight 1: Variable names and annotations carry the grammar. From counter names alone, Autogram recovers 100% of the invariant catalog on both Abilene and GEANT, whereas NetNomos recovers under 20% (Fig. 3), finding only basic non-negativity and zero-self-demand rules. NetNomos misses the core CrossCheck invariants because two-end link agreement holds exactly on too few noisy snapshots and flow conservation cannot be written in its grammar. Even in this benchmark, the counter names reflect structural annotations prepared for CrossCheck rather than raw router MIB strings, which are often cryptic in schemas such as OpenConfig [39]. Combining cryptic variable names with the topology, role, and vendor metadata that operators already track gives the LLM the context needed to classify data types automatically.

Insight 2: Exact approaches miss soft and aggregated invariants. Two barriers prevent an exact approach from recovering production network invariants. First, measurement noise and systematic bias cause genuine invariants to fail an exact 100% hold-rate check: two-end link agreement holds exactly on barely one-sixth of snapshots, and flow conservation carries a consistent offset at every router. Second, an exact approach's binary addition operator cannot express sums over variable-sized locality groups, which span 11 links per router on Abilene and 21 on GEANT. Autogram overcomes both barriers by inducing group-sum templates and evaluating them within data-calibrated tolerances, while pointing toward richer non-linear queueing and buffer relationships for future work [3, 48, 66] (§5).

Insight 3: Soft invariants diagnose systematic accounting offsets. Autogram discovers that the traffic a router reports originating approximately equals the row-sum of the demand matrix, and terminating traffic approximately equals the column-sum, capturing a routing-free conservation invariant directly from the counter names. Both relationships run roughly 2% short at every router because local interface counters and end-to-end demand aggregates account for packet headers and sampling intervals differently. Rather than discarding these imperfect relationships or burying the offset inside learned neural weights, Autogram reports the

Invariant	Hold-rate	Origin
Two-end counter agreement	96.5%	CrossCheck
Router & network flow conservation	92.3–97.5%	CrossCheck
Non-negativity, zero self-demand	100%	AUTOGRAM
Topology & demand presence symmetry	92.4–97.3%	AUTOGRAM
Demand conservation (termination)	**73.2%**	**AUTOGRAM**
Demand conservation (origination)	**68.6%**	**AUTOGRAM**

Table 2: On a production network, AUTOGRAM recovers CrossCheck's invariants and discovers deeper ones it never reported (bold).

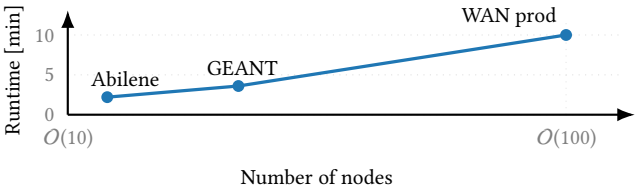


Figure 4: AUTOGRAM discovers a network's invariants in minutes across the evaluated networks.

conservation invariant together with its 2% residual band as an auditable diagnostic.

Insight 4: Quantifying over routers and links keeps exhaustive search compact and readable. Because AUTOGRAM quantifies candidate templates over routers and links (∀ node), the number of enumerated templates depends on the schema roles rather than the node count, generating thousands of entity-level templates rather than millions of individual variable pairs. Under the sum-only grammar in Table 1, enumeration at ❸ generates 6,332 raw candidates on Abilene and 2,972 on GEANT and `WAN prod`, of which 3,923 on Abilene and 1,595 on GEANT and `WAN prod` survive complexity, admissibility, and Z3 screening at ❹; adding `AVG`, `MIN`, and `MAX` aggregations and scaling coefficients yields 122,130 → 44,183 candidates on Abilene and 91,908 → 25,925 on GEANT. Statistical validation and Pareto pruning at ❺ distill those surviving candidates into 116 accepted rules on Abilene, 101 on GEANT, and 41 on `WAN prod` (Table 2). On GEANT, NetNomos outputs 1,645 individual rules, including 946 vacuous disjunctions and only 27 matches to known invariants, whereas AUTOGRAM achieves full recall in a compact portfolio that directly strengthens downstream CrossCheck validation.

Insight 5: Production telemetry validates both scalability and precision. To evaluate real-world scalability, we run AUTOGRAM on a live production WAN with $O(100)$ routers and $O(1000)$ links. Using the same prompt template and counter names alone, AUTOGRAM enumerates 2,972 candidates at ❸, retains 1,595 after Z3 screening at ❹, and completes in ten minutes, discovering 41 invariants across all benchmark categories including demand conservation and its 2% offset (Table 2). Runtime grows gently across the evaluated networks (Fig. 4). When our network operator inspected the 41 rules discovered on `WAN prod`, the operator saw no spurious invariants empirically in practice: while the public benchmarks admit a few incidental inequalities from correlated traffic, every rule accepted on the production WAN captured a genuine topological or traffic invariant, and the compact 41-rule output lets an operator verify the entire portfolio in minutes.

5 Research Agenda

Discover more by annotating more. Enriching the operator metadata that AUTOGRAM reads directly expands the invariants it can discover. Adding attributes that operators already track, such as router vendor or hardware generation, allows the grammar to express relationships that variable names alone cannot capture, such as an accounting offset that appears only on links between different router vendors.

When the annotations are wrong. AUTOGRAM treats metadata as a starting signal, yet variable names and annotations often drift, conflict across teams, or contain human errors [20]. Inaccurate labels can skew the induced grammar, yet discovered invariants also help diagnose those very labeling errors: when a conservation invariant holds across every router except one interface, that outlier provides strong evidence that the interface is mislabeled [49]. Characterizing how much label noise the pipeline can tolerate before grammar quality degrades remains an open problem.

Let the data shape the grammar. Summary statistics from the telemetry itself can guide grammar construction before full enumeration begins. A lightweight statistical pass that clusters variables with correlated time series can narrow which variable groups belong in the grammar, turning grammar induction into data-guided refinement [16, 63].

Compose invariants into new ones. Discovered invariants can serve as building blocks for multi-step deduction. Chaining router-level invariants across adjacent hops can derive network-wide relationships that no single template expresses in isolation [11, 27].

Proxies that do not presuppose the answer. Evaluating grammar recall against a catalog of known invariants still depends on having a trusted seed set. A broader calibration signal can score candidate grammars directly on downstream tasks such as telemetry imputation [21, 22], traffic forecasting and control [2, 30, 46], or trace and workload synthesis [28, 35, 54, 57, 61], rewarding grammars that capture the joint structure of the telemetry without referencing known invariants.

Acknowledgments

We thank Lillian Tsai and the anonymous reviewers for their insightful feedback and comments, which meaningfully improved this paper. This work was supported by the National Science Foundation (NSF) through Grants CNS-2442625 and CNS-2319442 and Office of Naval Research (ONR) through Grant N000142612552.

References

[1] Vamsi Addanki, Maria Apostolaki, Manya Ghobadi, Stefan Schmid, and Laurent Vanbever. 2022. ABM: Active Buffer Management in Datacenters. In *Proceedings of the ACM SIGCOMM 2022 Conference*.

[2] Vamsi Addanki, Maria Apostolaki, Manya Ghobadi, Stefan Schmid, and Laurent Vanbever. 2024. Credence: Augmenting Datacenter Switch Buffer Sharing with ML Predictions. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*.

[3] Vamsi Addanki, Wei Bai, Stefan Schmid, and Maria Apostolaki. 2024. Reverie: Low Pass Filter-Based Switch Buffer Sharing for Datacenters with RDMA and TCP Traffic. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*.

[4] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*. 487–499.

[5] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. 2014. NetKAT: Semantic Foundations for Networks. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '14)*. ACM, 113–126. doi:10.1145/2535838.2535862

[6] Maria Apostolaki, Ankit Singla, and Laurent Vanbever. 2021. Performance-Driven Internet Path Selection. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*. 41–53.

[7] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. 2017. A General Approach to Network Configuration Verification. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, 155–168. doi:10.1145/3098822.3098834

[8] Nicolas Beldiceanu and Helmut Simonis. 2012. A Model Seeker: Extracting Global Constraint Models from Positive Examples. In *Principles and Practice of Constraint Programming*, Michela Milano (Ed.). Springer, Berlin, Heidelberg, 141–157. doi:10.1007/978-3-642-33558-7_13

[9] Henry Birge-Lee, Maria Apostolaki, and Jennifer Rexford. 2022. It takes two to tango: cooperative edge-to-edge routing. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. 174–180.

[10] Henry Birge-Lee, Maria Apostolaki, and Jennifer Rexford. 2025. Global BGP Attacks that Evade Route Monitoring. In *Proceedings of the 26th Passive and Active Measurement Conference (PAM 25)*.

[11] Henry Birge-Lee, Sophia Yoo, Benjamin Herber, Jennifer Rexford, and Maria Apostolaki. 2024. TANGO: Secure Collaborative Route Control across the Public Internet. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*.

[12] Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. 2020. Discovering symbolic models from deep learning with inductive biases. *Advances in neural information processing systems* 33 (2020), 17429–17442.

[13] Joscha Cüppers, Adrien Schoen, Gregory Blanc, and Pierre-Francois Gimenez. 2024. FlowChronicle: Synthetic Network Flow Generation through Pattern Set Mining. *Proceedings of the ACM on Networking* 2, CoNEXT4 (2024), 1–20.

[14] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings 14*. Springer, 337–340.

[15] Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? Does it matter? *Structural Safety* 31, 2 (2009), 105–112. doi:10.1016/j.strusafe.2008.06.020

[16] Michael D. Ernst, Jeff H. Perkins, Philip J. Guo, Stephen McCamant, Carlos Pacheco, Matthew S. Tschantz, and Chen Xiao. 2007. The Daikon System for Dynamic Detection of Likely Invariants. *Science of Computer Programming* 69, 1-3 (2007), 35–45. doi:10.1016/j.scico.2007.01.015

[17] Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. 2011. Discovering Conditional Functional Dependencies. *IEEE Transactions on Knowledge and Data Engineering* 23, 5 (May 2011), 683–698. doi:10.1109/TKDE.2010.154 Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[18] Cheng Feng, Venkata Reddy Palleti, Aditya Mathur, and Deeph Chana. 2019. A Systematic Framework to Generate Invariants for Anomaly Detection in Industrial Control Systems. In *Network and Distributed System Security Symposium (NDSS)*.

[19] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. 2015. A General Approach to Network Configuration Analysis. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. USENIX Association, Oakland, CA, 469–483. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/fogel

[20] Benoît Frénay and Michel Verleysen. 2014. Classification in the Presence of Label Noise: A Survey. *IEEE Transactions on Neural Networks and Learning Systems* 25, 5 (2014), 845–869.

[21] Fengchen Gong, Divya Raghunathan, Aarti Gupta, and Maria Apostolaki. 2023. Towards Integrating Formal Methods into ML-Based Systems for Networking. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks (HotNets '23)*. Association for Computing Machinery, New York, NY, USA, 48–55. doi:10.1145/3626111.3628188

[22] Fengchen Gong, Divya Raghunathan, Aarti Gupta, and Maria Apostolaki. 2024. Zoom2net: Constrained network telemetry imputation. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 764–777.

[23] Travis Hance, Marijn Heule, Ruben Martins, and Bryan Parno. 2021. Finding invariants of distributed systems: It's a small (enough) world after all. In *18th USENIX symposium on networked systems design and implementation (NSDI 21)*. 115–131.

[24] Hongyu Hè. 2024. *FPGA-based SmartNIC for Distributed Machine Learning*. Master's thesis. ETH Zurich.

[25] Hongyu Hè. 2024. PerOS: Personalized Self-Adapting Operating Systems in the Cloud. *arXiv preprint arXiv:2404.00057* (2024).

[26] Hongyu Hè and Maria Apostolaki. 2025. Just-in-Time Logic Enforcement: A new paradigm of combining statistical and symbolic reasoning for network management. In *Proceedings of the 24th ACM Workshop on Hot Topics in Networks (HotNets 25)*. 184–192.

[27] Hongyu Hè and Maria Apostolaki. 2026. Let AI Agents Translate Networks, Not Reason About Them: Typographical Network Modeling with Automated and Verifiable Axiomatization. *arXiv preprint arXiv:2607.22947* (2026).

[28] Hongyu Hè, Minhao Jin, and Maria Apostolaki. 2026. Making Logic a First-Class Citizen in Generative ML for Networking. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*.

[29] Hongyu Hè, Minhao Jin, and Maria Apostolaki. 2027. Worst-Case Discovery and Runtime Protection for RL-Based Network Controllers. In *24th USENIX Symposium on Networked Systems Design and Implementation (NSDI 27)*.

[30] Mehdi Hosseinzadeh, Karthick Shankar, Maria Apostolaki, Jay Ramachandran, Steven E Adams, Vyas Sekar, and Bruno Sinopoli. 2023. CANE: A Cascade Control Approach for Network-Assisted Video QoE Management. *IEEE Transactions on Control Systems Technology* (2023).

[31] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. Tane: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *Comput. J.* 42, 2 (Feb. 1999), 100–111. doi:10.1093/comjnl/42.2.100

[32] Eyke Hüllermeier and Willem Waegeman. 2021. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning* 110, 3 (2021), 457–506. doi:10.1007/s10994-021-05946-3

[33] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. 2024. Netdiffusion: Network data augmentation through protocol-constrained traffic generation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8, 1 (2024), 1–32.

[34] Minhao Jin and Maria Apostolaki. 2025. Robustifying ML-powered Network Classifiers with PANTS. In *34th USENIX Security Symposium (USENIX Security 25)*.

[35] Minhao Jin, Hongyu Hè, and Maria Apostolaki. 2027. Cross-Flow Correlations Survive Synthesis: Measuring Source-Level Privacy Leakage in Synthetic Network Traces. In *Network and Distributed System Security Symposium (NDSS 27)*.

[36] Aqsa Kashaf, Aidan Walsh, Maria Apostolaki, Vyas Sekar, and Yuvraj Agarwal. 2024. Network Function Capacity Reconnaissance by Remote Adversaries. *arXiv preprint arXiv:2405.09442* (2024).

[37] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in Bayesian deep learning for computer vision?. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 30.

[38] Ahmed Khurshid, Wenxuan Zhou, Matthew Caesar, and P. Brighten Godfrey. 2012. VeriFlow: Verifying Network-Wide Invariants in Real Time. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*. ACM, 49–54. doi:10.1145/2342441.2342452

[39] Alexander Krentsel, Rishabh Iyer, Isaac Keslassy, Bharath Modhipalli, Sylvia Ratnasamy, Anees Shaikh, and Rob Shakir. 2026. CrossCheck: Input Validation for WAN Control Systems. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. USENIX Association, 647–667. https://www.usenix.org/conference/nsdi26/presentation/krentsel

[40] Alexander Krentsel, Rishabh R. Iyer, Isaac Keslassy, Sylvia Ratnasamy, Anees Shaikh, and Rob Shakir. 2024. The Case for Validating Inputs in Software-Defined WANs. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks (HotNets 24)*. ACM, 246–254. doi:10.1145/3696348.3696874

[41] Benno Kruit, Hongyu Hè, and Jacopo Urbani. 2020. Tab2Know: Building a Knowledge Base from Tables in Scientific Papers. In *Proceedings of the 19th International Semantic Web Conference (ISWC 20)*. Springer, 349–365.

[42] Algorithmic SuperIntelligence Labs. [n. d.]. Open-source implementation of AlphaEvolve. https://github.com/algorithmicsuperintelligence/openevolve.

[43] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. 2018. Distribution-Free Predictive Inference for Regression. *J. Amer. Statist. Assoc.* 113, 523 (2018), 1094–1111.

[44] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. 2025. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131* (2025).

[45] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alexander Gammerman. 2002. Inductive Confidence Machines for Regression. In *Machine Learning: ECML 2002*. Springer, 345–356.

[46] Konstantina Papagiannaki, Nina Taft, Zhi-Li Zhang, and Christophe Diot. 2005. Long-Term Forecasting of Internet Backbone Traffic. *IEEE Transactions on Neural Networks* 16, 5 (2005), 1110–1124.

[47] Thorsten Papenbrock and Felix Naumann. 2016. A Hybrid Approach to Functional Dependency Discovery. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 821–833. doi:10.1145/2882903.2915203

[48] Divya Raghunathan, Maria Apostolaki, and Aarti Gupta. 2025. A Layered Formal Methods Approach to Answering Queue-related Queries. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. USENIX Association, 1289–1304. https://www.usenix.org/conference/nsdi25/presentation/raghunathan

[49] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. Holistic Data Repairs with Probabilistic Inference. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1190–1201.

[50] Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning* 62, 1–2 (2006), 107–136.

[51] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. 2024. Mathematical Discoveries from Program Search with Large Language Models. *Nature* 625, 7995 (2024), 468–475.

[52] Rahul Anand Sharma, Ishan Sabane, Maria Apostolaki, Anthony Rowe, and Vyas Sekar. 2022. Lumen: a framework for developing and evaluating ML-based IoT network anomaly detection. In *Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies*. 59–71.

[53] Žiga Stupan and Iztok Fister Jr. 2022. Niaarm: a minimalistic framework for numerical association rule mining. *Journal of Open Source Software* 7, 77 (2022), 4448.

[54] Dmitrii Ustiugov, Dohyun Park, Lazar Cvetković, Mihajlo Djokic, Hongyu Hè, Boris Grot, and Ana Klimovic. 2023. Enabling In-Vitro Serverless Systems Research. In *Proceedings of the 4th Workshop on Resource Disaggregation and Serverless (WORDS 23)*. ACM.

[55] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. 2017. FarmBeats: An IoT Platform for Data-Driven Agriculture. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 515–529.

[56] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic Learning in a Random World*. Springer.

[57] Xizheng Wang, Libin Liu, Li Chen, Dan Li, Yukai Miao, and Yu Bai. 2025. Resolving Packets from Counters: Enabling Multi-scale Network Traffic Super Resolution via Composable Large Traffic Model. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. USENIX Association, 1541–1561. https://www.usenix.org/conference/nsdi25/presentation/wang-xizheng-resolving

[58] Xieyang Xu, Yifei Yuan, Zachary Kincaid, Arvind Krishnamurthy, Ratul Mahajan, David Walker, and Ennan Zhai. 2024. Relational Network Verification. In *Proceedings of the ACM SIGCOMM 2024 Conference*. ACM, 213–227. doi:10.1145/3651890.3672238

[59] Jianan Yao, Runzhou Tao, Ronghui Gu, and Jason Nieh. 2022. {DuoAI}: Fast, automated inference of inductive invariants for verifying distributed protocols. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. 485–501.

[60] Jianan Yao, Runzhou Tao, Ronghui Gu, Jason Nieh, Suman Jana, and Gabriel Ryan. 2021. {DistAI}:{Data-Driven} automated invariant

learning for distributed protocols. In *15th USENIX symposium on operating systems design and implementation (OSDI 21)*. 405–421.

[61] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. 2022. Practical gan-based synthetic ip header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 458–472.

[62] Yifei Yuan, Soo-Jin Moon, Sahil Uppal, Limin Jia, and Vyas Sekar. 2020. {NetSMC}: A Custom Symbolic Model Checker for Stateful Network Verification. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 181–200.

[63] Chengqi Zhang and Shichao Zhang. 2002. *Association rule mining: models and algorithms*. Springer.

[64] Anchengcheng Zhou, Carter Costic, Hongyu Hè, Ahmad Ghalayini, Abdul Kabbani, and Maria Apostolaki. 2025. Mitigating Inter-datacenter Incast with a Proxy. In *Proceedings of the 24th ACM Workshop on Hot Topics in Networks (HotNets 25)*.

[65] Anchengcheng Zhou, Vipul Harsh, Sangeetha Abdu Jyothi, Maria Apostolaki, and P. Brighten Godfrey. 2026. Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*.

[66] Anchengcheng Zhou, Joshua Lau, P. Brighten Godfrey, and Maria Apostolaki. 2026. Controlling Arbitrary Internet Queues with Titrate. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*.