

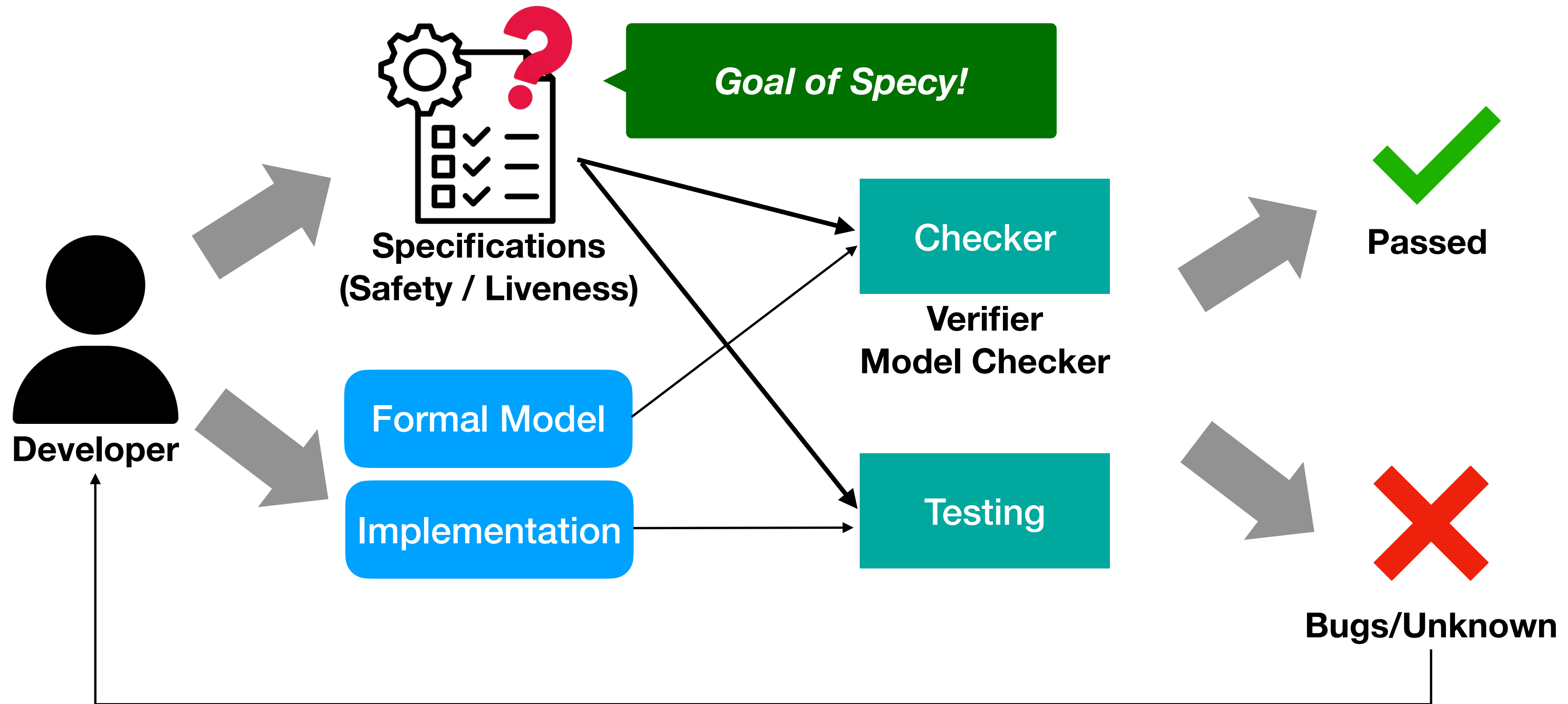


Specy: Learning Specifications for Distributed Systems from Event Traces

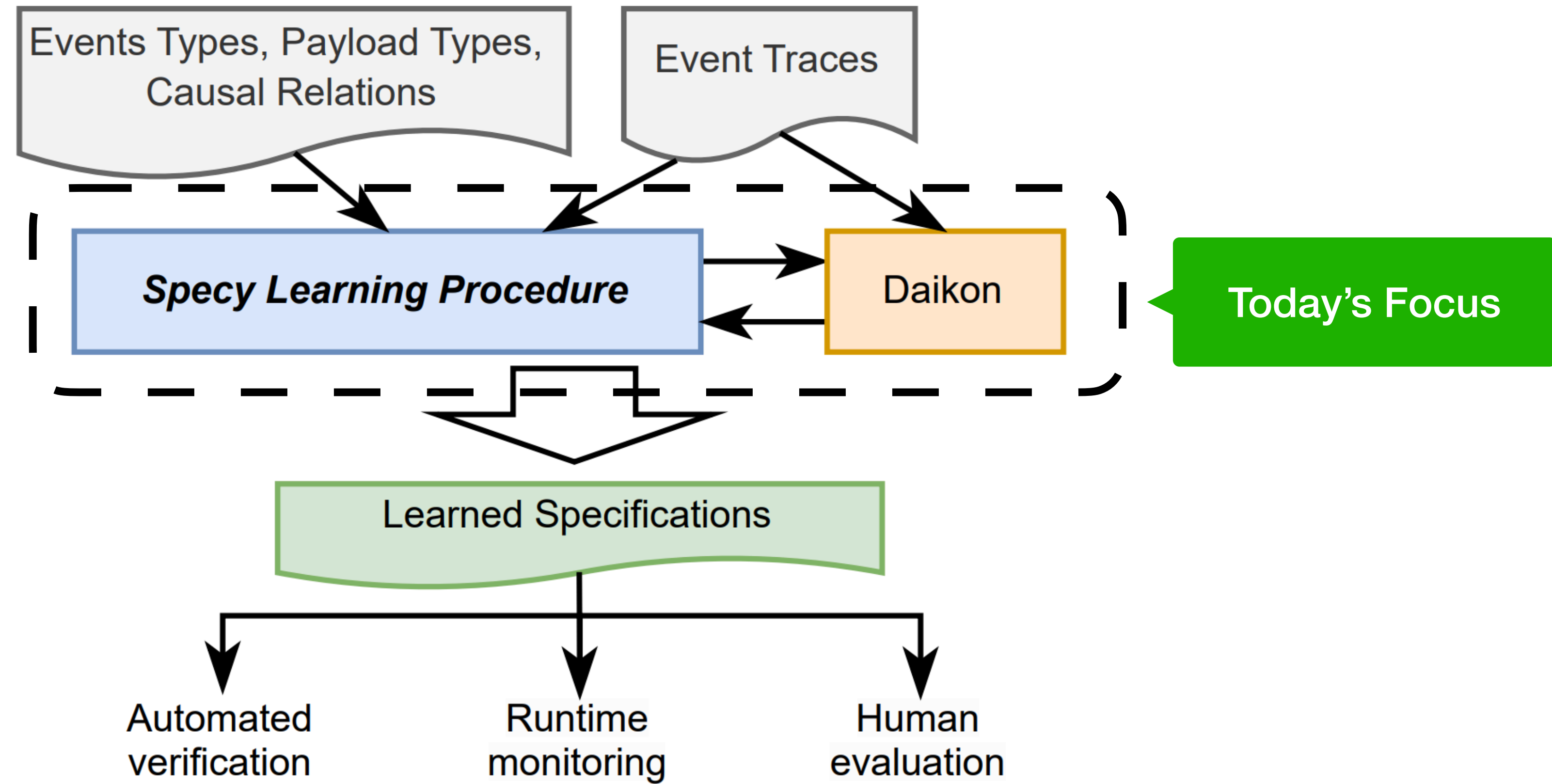
Mike He, Ankush Desai, Aishwarya Jagarapu, Doug Terry, Sharad Malik, Aarti Gupta



Guaranteeing correctness of distributed systems in development



Specy Prototype Workflow





Insight 1

Most specifications can be expressed as (restricted) first-order logic (FOL) formulas over events

- Events are *visible* at inter-module interfaces
- Event-base specifications are *reusable* across formal models and implementations with the same interfaces

Examples

External Consistency (Google Spanner)

If a transaction T1 commits before T2 starts, then the timestamp of T1 is less than that of T2

$$\forall e_0, e_1 : eCommit.e_0.commit < e_1.start \rightarrow e_0.ts < e_1.ts$$

Whitelist Safety (Firewall)

For each event that is allowed by the firewall, there exists an earlier event that grants the permission to the sender

$$\forall e_0 : eRecv.e_0.allowed \rightarrow \exists e_1 : eGrant.e_1 \prec e_0 \wedge e_0.sender = e_1.host$$



Happens-before relation

Quorum Votes (Consensus)

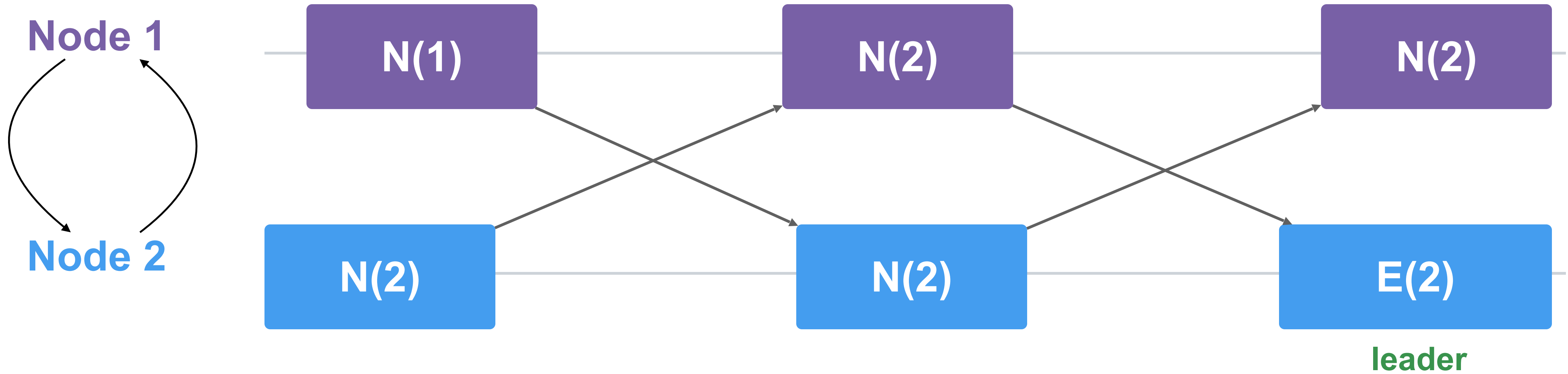
Every decision must have received a majority of votes

$$\forall e_0 : eDecide . \exists_{\geq quorum} e_1 : eVote.e_1 \prec e_0 \wedge e_0.ballot = e_1.vote$$

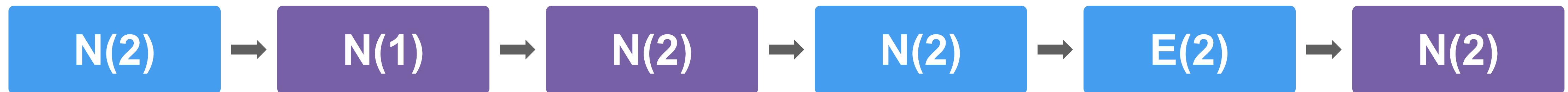


At least a quorum number of eVote event

Motivating Example: Ring Leader Election (RLE) Protocol

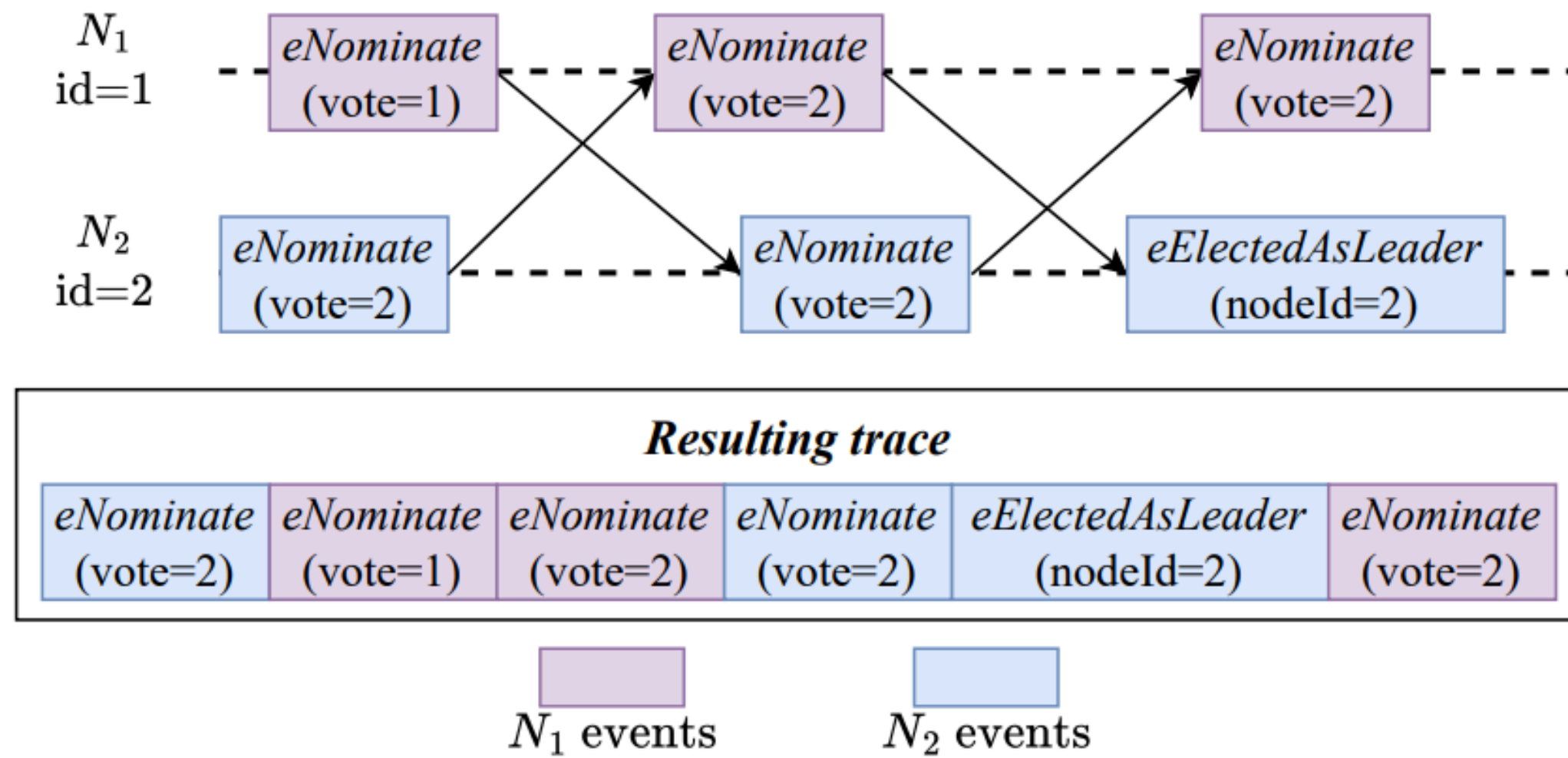


Observed trace



N(v): nomination vote E(v): elected node

Motivating Example: Goal Specifications for RLE



(b) An example event trace with 2 nodes N_1 and N_2

Unique Leader:

$$\forall e_0, e_1 : eElectedAsLeader . e_0 . nodeId = e_1 . nodeId$$

Leader has the highest id

$$\forall e_0 : eElectedAsLeader, e_1 : eNominate . e_0 . nodeId \geq e_1 . vote$$

Leader was voted

$$\forall e_0 : eElectedAsLeader . \exists e_1 : eNominate . e_1 < e_0 \wedge e_0 . nodeId = e_1 . vote$$

How to search for such specifications from event traces?



Insight 2

Specifications expressed as FOL formulas over events can be decomposed into the following components

$$\forall (e_i : \tau_i)^+ . \boxed{G} \rightarrow \exists (e_j : \tau_j)^* . \boxed{W} \wedge \boxed{H}$$

GuardsWitnesses (optional)

Control condition**Existence of certain events**

Hypotheses
Property holds under Guards and Witnesses

Whitelist Safety (Firewall)

$$\forall e_0 : eRecv . \boxed{e_0 . allowed} \rightarrow \exists e_1 : eGrant . \boxed{e_1 \prec e_0} \wedge \boxed{e_0 . sender = e_1 . host}$$

Grammar of Specification Formulas

$$\begin{aligned}\phi_{\forall} &::= (\forall e_i : \tau_i)^+ . \boxed{\bigwedge \vec{P}} \rightarrow \bigwedge \vec{P} \\ \phi_{\forall\exists} &::= (\forall e_i : \tau_i)^+ . \boxed{\bigwedge \vec{P}} \rightarrow (\exists_{SC} e_j : \tau_j)^+ . \boxed{\bigwedge \vec{P}} \bigwedge \vec{P}\end{aligned}$$

Top-level rules

Preds

Event Preds

Field Preds

SetCardinality

$$P ::= P_E \mid P_F \mid \neg P_F$$

$$P_E ::= T_E = T_E \mid T_E < T_E \mid T_E > T_E$$

$$P_F ::= T_F = T_F \mid T_F < T_F$$

$$\mid T_F \leq T_F \mid uP(uf(T_F^+)^+)$$

$$SC(m) ::= m \geq 1 \mid m = C \mid m \leq C \mid m \geq C$$

Vars

Constants

Event Vars

Event Types

Terms

Event Terms

Field Terms

$$v \in V$$

$$c \in C$$

$$e \in E$$

$$\tau_i$$

$$T ::= T_E \mid T_F$$

$$T_E ::= \text{indexof}(E)$$

$$T_F ::= C \mid E.v \mid f(T_F^+)$$

Predicates

Terms

temporal predicates & relational predicates

Allow user-provided predicates and functions

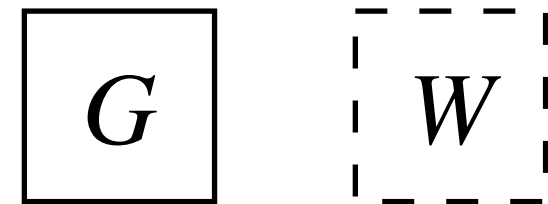
Core of Learning Procedure

Combining static enumerative search and dynamic learning

$$\forall(e_i : \tau_i)^+ . \boxed{G} \rightarrow \exists(e_j : \tau_j)^* . \boxed{W} \wedge \boxed{H}$$

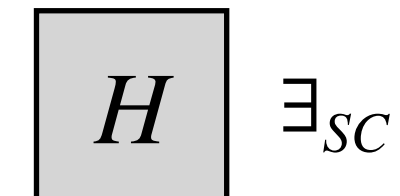
Static Enumeration

Guards
(optionally Witnesses)



Dynamic Learning

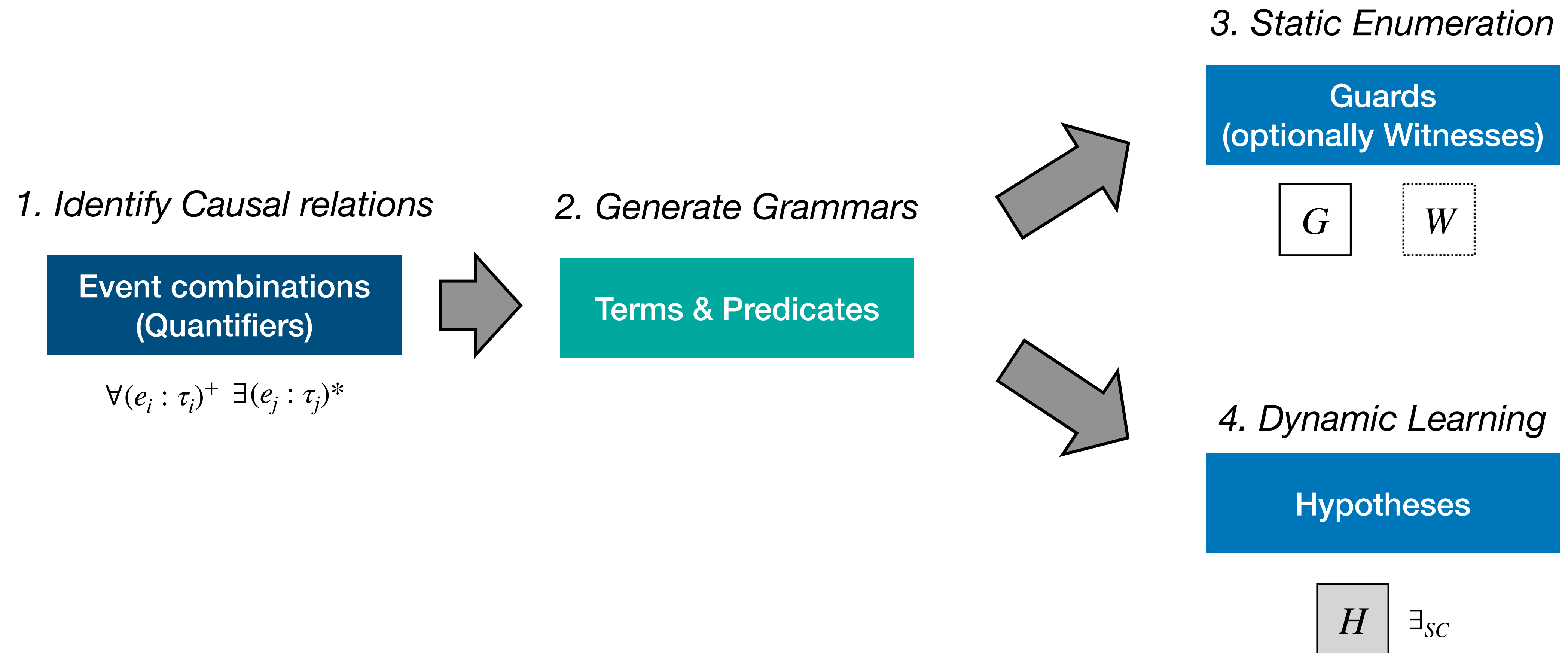
Hypotheses



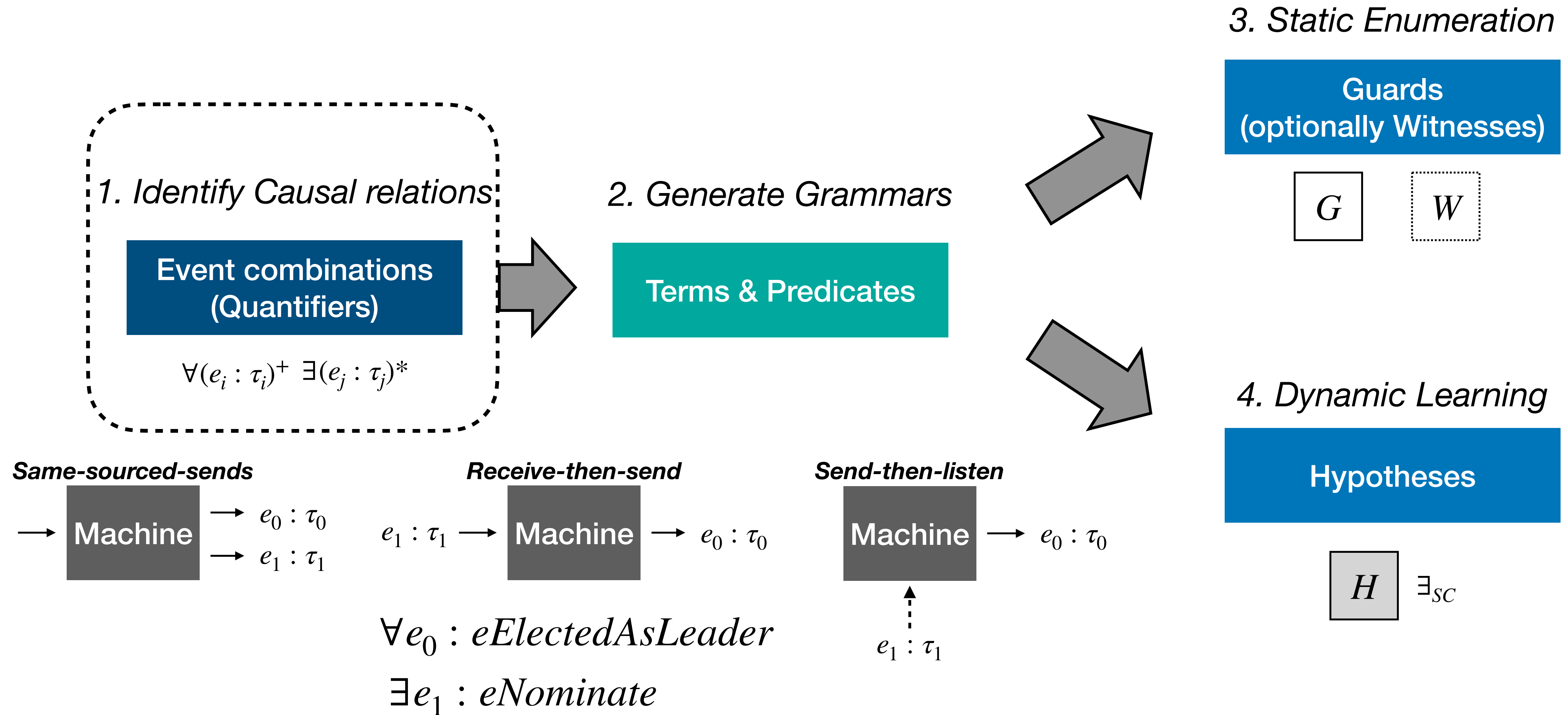
controls the scope of dynamic learning

Search problem decomposed based on the grammar

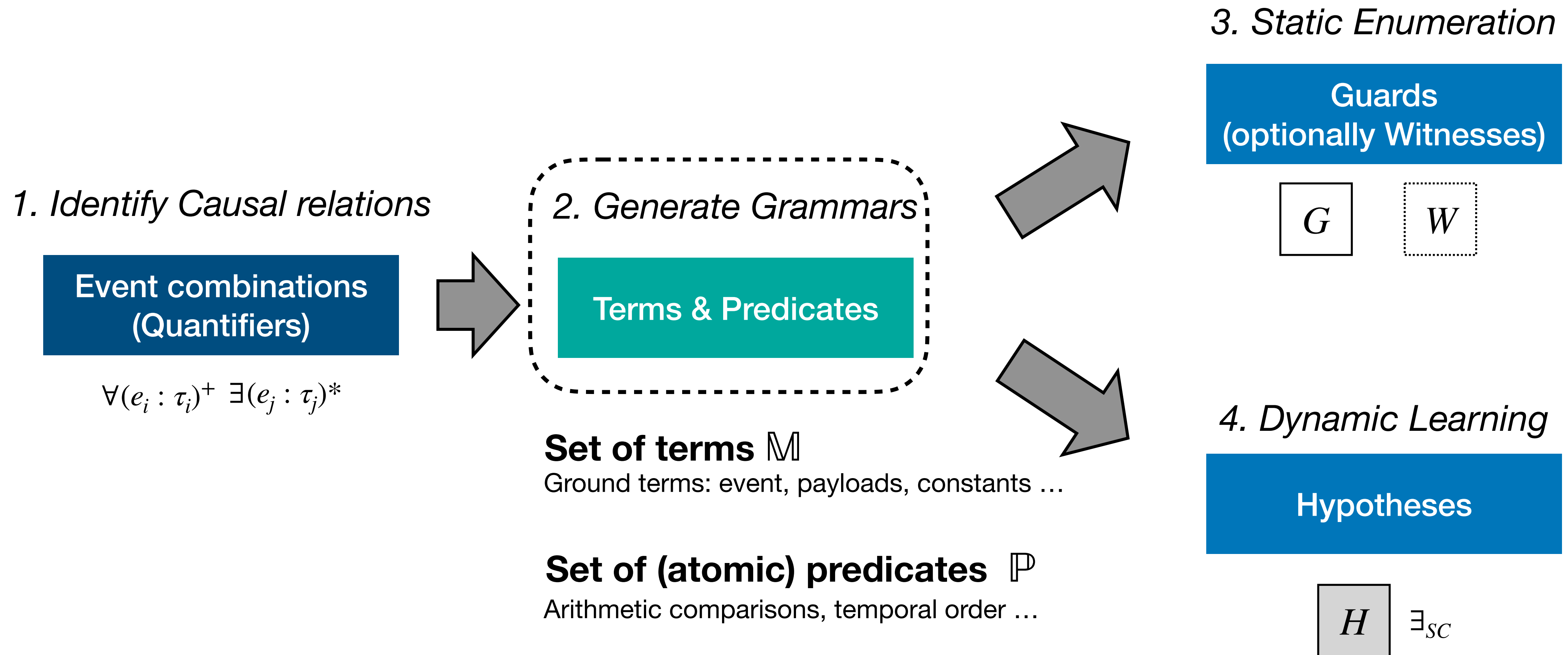
Learning Procedure



Learning Procedure



Learning Procedure



Learning Procedure

1. Identify Causal relations

Event combinations
(Quantifiers)

$\forall(e_i : \tau_i)^+ \exists(e_j : \tau_j)^*$

$\forall e_0 : eElectedAsLeader$
 $\exists e_1 : eNominate$

2. Generate Grammars

Terms & Predicates

Set of terms $\mathbb{M}$

$e_0 \quad e_1$
 $e_0.nodeId$
 $e_1.vote$

Set of predicates $\mathbb{P}$

$e_0 < e_1 \quad e_1 < e_0$
 $e_0.nodeId = e_1.vote$
 $e_0.nodeId < e_1.vote$
 $e_0.nodeId > e_1.vote$

3. Static Enumeration

Guards
(optionally Witnesses)

G

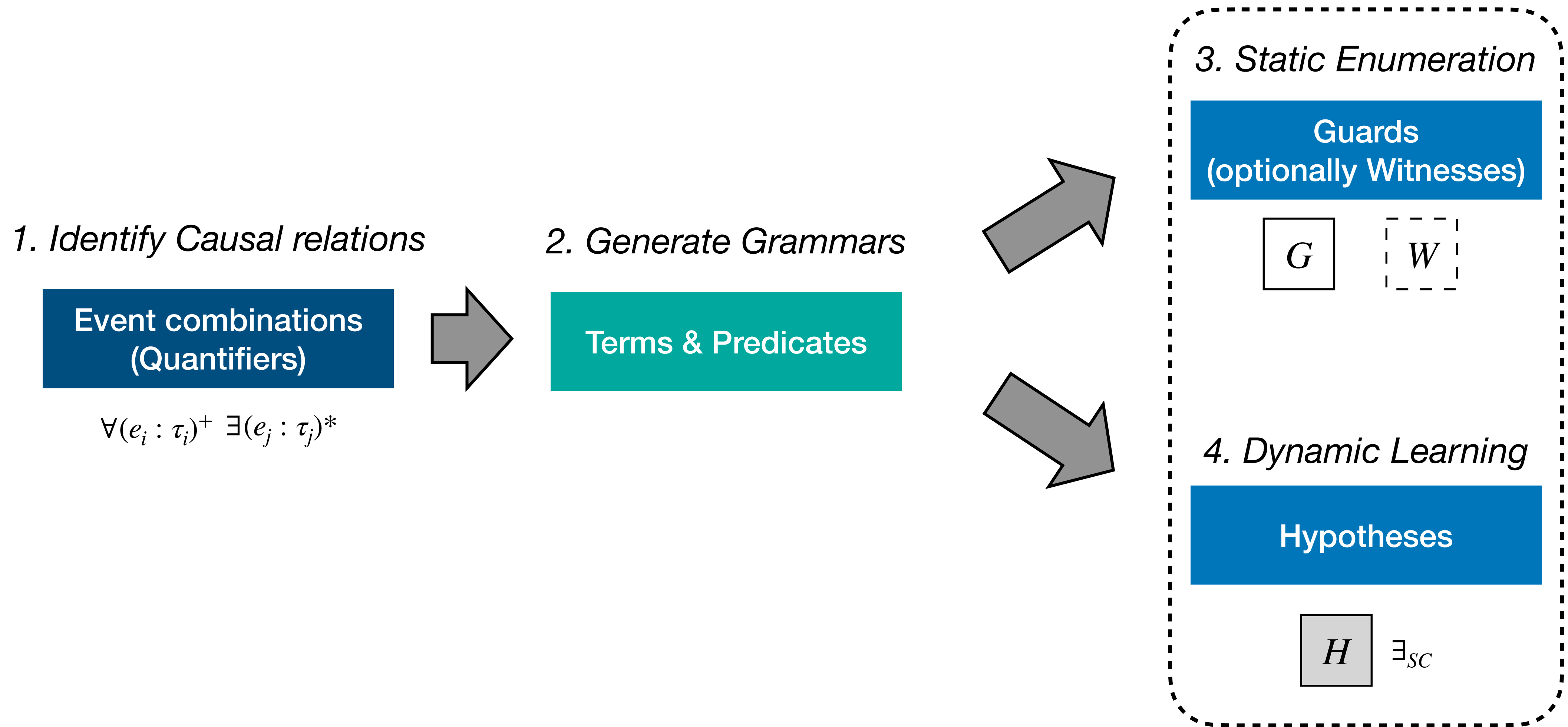
W

4. Dynamic Learning

Hypotheses

$H \quad \exists_{sc}$

Learning Procedure



3. Static Enumeration

Grid Search for

Guards
(optionally Witnesses)

Given: A set of terms $\mathbb{M}$
A set of predicates $\mathbb{P}$

At a grid point (g, w, r) , Specy enumerates

G

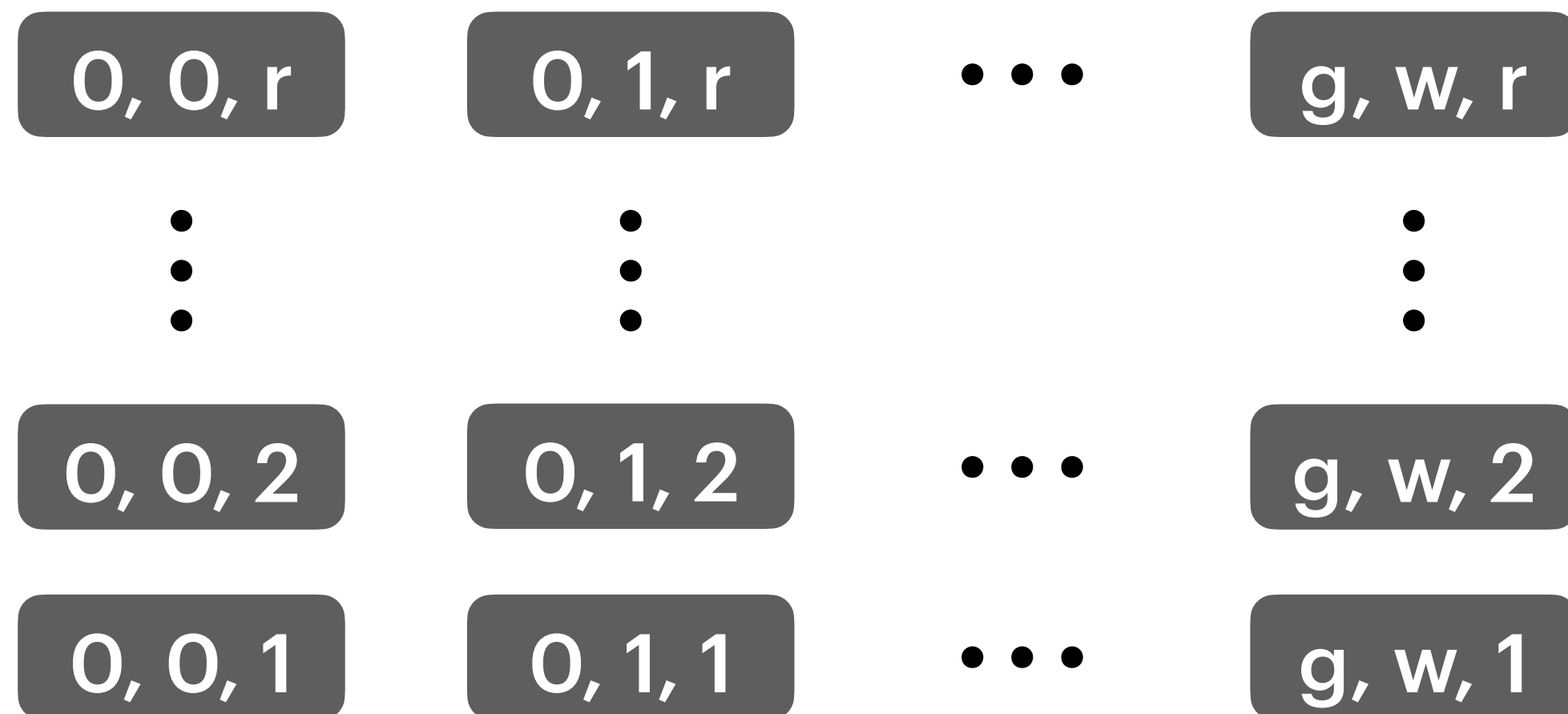
Conjunctions of g number predicates from $\mathbb{P}$

W

Conjunctions of w number predicates from $\mathbb{P}$

$\mathbb{R}$

Subsets of $\mathbb{M}$ with size r , called Relate Sets
over which the dynamic learner learns predicates in H



G conjoins g number of predicates
W conjoins w number of predicates
H relates 2 terms

4. *Dynamic Learning*

Learn

Hypotheses	H
------------	-----

 from traces

Given: $\forall(e_i : \tau_i)^+$
 $\exists(e_j : \tau_j)^*$, enumerated

G

W

 $\mathbb{R}$ and event traces

1. Compute instantiations Σ that satisfy

G

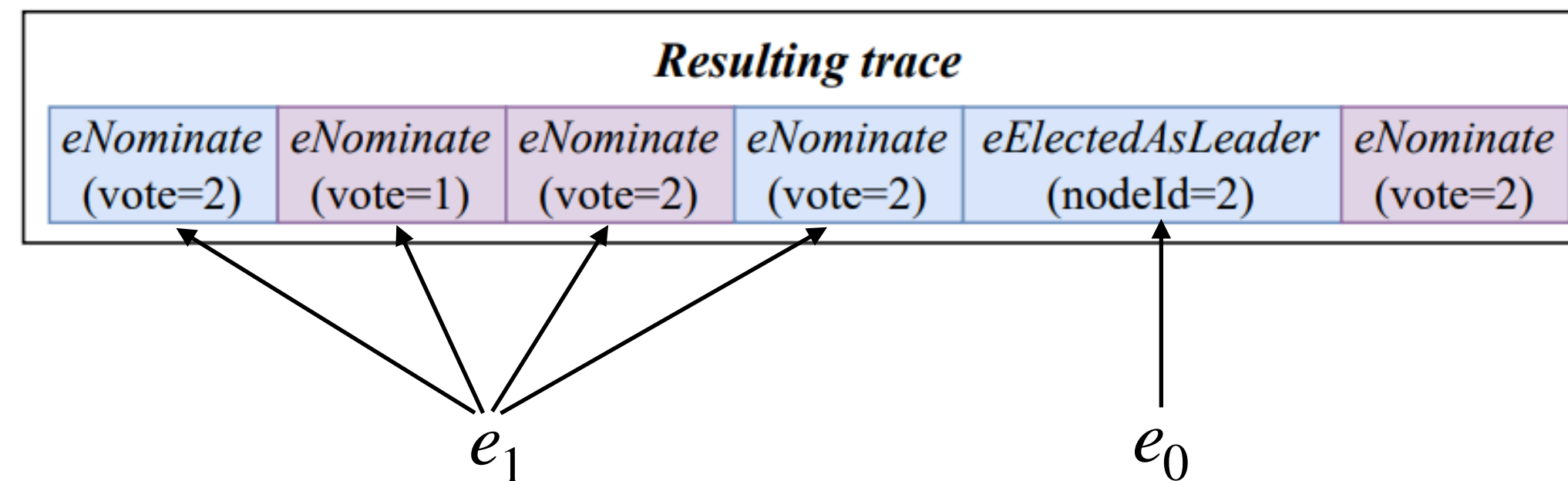
W

4. Dynamic Learning

Learn **Hypotheses** H from traces

Given: $\forall (e_i : \tau_i)^+$, enumerated G W $\mathbb{R}$ and event traces

1. Compute instantiations Σ that satisfy $e_1 < e_0$ T
 $\forall e_0 : eElectedAsLeader, e_1 : eNominate$

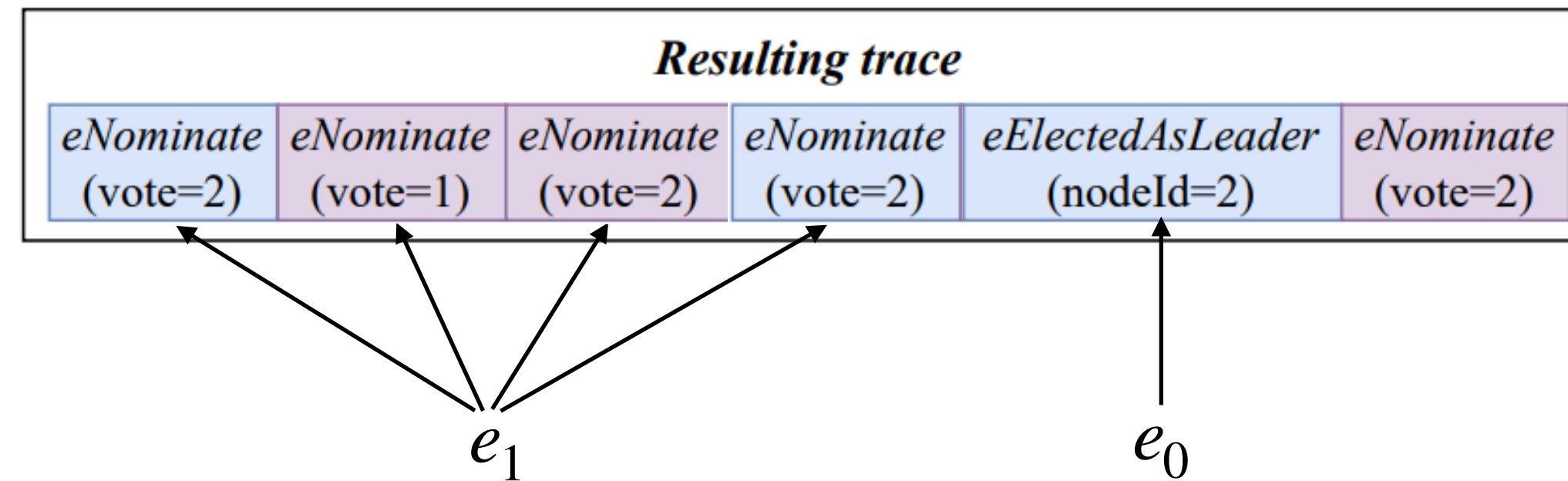


4. Dynamic Learning

Learn **Hypotheses** H from traces

Given: $\forall(e_i : \tau_i)^+$, enumerated G W $\mathbb{R}$ and event traces

1. Compute instantiations Σ that satisfy $e_1 < e_0$ T
 $\forall e_0 : eElectedAsLeader, e_1 : eNominate$



2. Compute *Observations* of $\mathbb{R}$, $\mathcal{O}_{\mathbb{R}}$, under each instantiation

$$\mathbb{R} = \{e_0 . nodeId, e_1 . vote\}$$

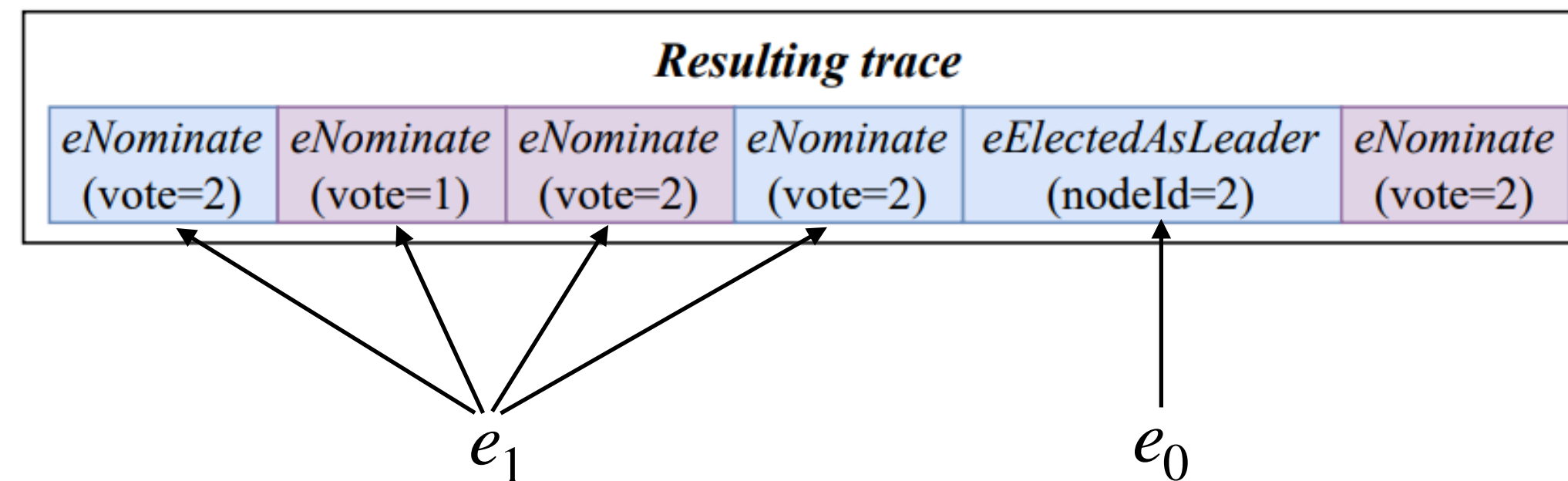
$$\mathcal{O}_{\mathbb{R}} = \{(2, 2), (2, 1), (2, 2), (2, 2)\}$$

4. Dynamic Learning

Learn **Hypotheses** H from traces

Given: $\forall(e_i : \tau_i)^+$, enumerated G W $\mathbb{R}$ and event traces

1. Compute instantiations Σ that satisfy $e_1 < e_0$ T
 $\forall e_0 : eElectedAsLeader, e_1 : eNominate$



2. Compute Observations of $\mathbb{R}$, $\mathcal{O}_{\mathbb{R}}$, under each instantiation

$$\mathbb{R} = \{e_0 . nodeId, e_1 . vote\}$$

$$\mathcal{O}_{\mathbb{R}} = \{(2, 2), (2, 1), (2, 2), (2, 2)\}$$

3. Learn predicates for H that hold for *all* observations in $\mathcal{O}_{\mathbb{R}}$ $e_0 . nodeId \geq e_1 . vote$

4. *Dynamic Learning*

Putting them together

$$\forall e_0 : eElectedAsLeader, e_1 : eNominate \quad \boxed{e_1 < e_0} \quad \longrightarrow \quad \boxed{e_0 . nodeId \geq e_1 . vote}$$

Evaluations

[RQ1] Learning specifications identified by previous literatures and developers

[RQ2] Learning inductive invariant for deductive verification

[RQ3] Benefit of Specy in practice

RQ1: Learning Known Specifications

Well-known
distributed protocols

Protocols
developed by AWS

Benchmark	LoC	S_{goals}	Description of Specifications	Time (s)
Ring Leader Election [9]	47	3 ✓	Unique leader, Leader Highest ID, Leader Voted	352
Consensus ^T [26]	61	1 ✓	Safety (Unique Decision)	925
2PC [3]	238	2 ✓	Atomicity ($\forall \exists_n$), Safety (Commit or Abort)	3935
Sharded KV ^{T*} [26]	48	1 ✓	Safety (Unique Shard Owner)	481
Paxos [31]	164	2 ✓	Safety (Unique Decision), B3($\mathcal{B}$) [†]	1155
Distributed Lock ^T [37]	73	1 ✓	Safety (Mutual Exclusion)	902
Vertical Paxos [33]	241	2 ✓	Safety (Unique Decision), B3($\mathcal{B}$) [†]	2589
Firewall [38]	80	1 ✓	Whitelisted Safety ($\forall \exists$)	753
Lock Server ^T [37]	102	1 ✓	Safety (Mutual Exclusion)	620
Chain Replication [49]	452	6 ✓	Update Propagation, ⁺ Inprocess Requests, ⁺ Linearizability ($\forall \exists$) ^{†#}	984
Raft*[40]	1191	5 ✓	Election Safety, State Machine Safety, Log Matching ⁺ Leader Append-Only, Leader Completeness ⁺	4223
GlobalClock	177	3 ✓	Clock Monotonicity, Real-time ordering [#]	805
DBLeaderElection	1178	5 ✓	Monotonic Commits, Unique Leader, Read Consistency ^{†#}	697
MVCC-2PC	1135	10 ✓	Atomicity ($\forall \exists_n$) [#] , Snapshot Isolation ($\forall \exists$) ^{‡#}	6642

Learned all specifications from well-known and proprietary protocols identified by previous works

All learned specifications are listed in Supplementary Material of our paper

RQ2: Learning inductive invariant

Table 6. Results for learning inductive invariants using SPECY (RQ3). We use a subset of benchmarks with proofs in PVerifer [38]. I_e and I_s columns show the numbers of event-based and state-based inductive invariants, respectively.

Benchmark	I_e	I_s	All learned
Ring Leader Election*	1	2	✓
Consensus*	1	5	✓
Distributed Lock*	4	4	✓
Lock Server*	4	4	✓
Firewall	1	0	✓
Sharded KV*	1	1	✓

All the event-based inductive invariants are learned automatically
State-based inductive invariants are learned with user guidance

RQ3: Impact of Learned Specifications in Practice

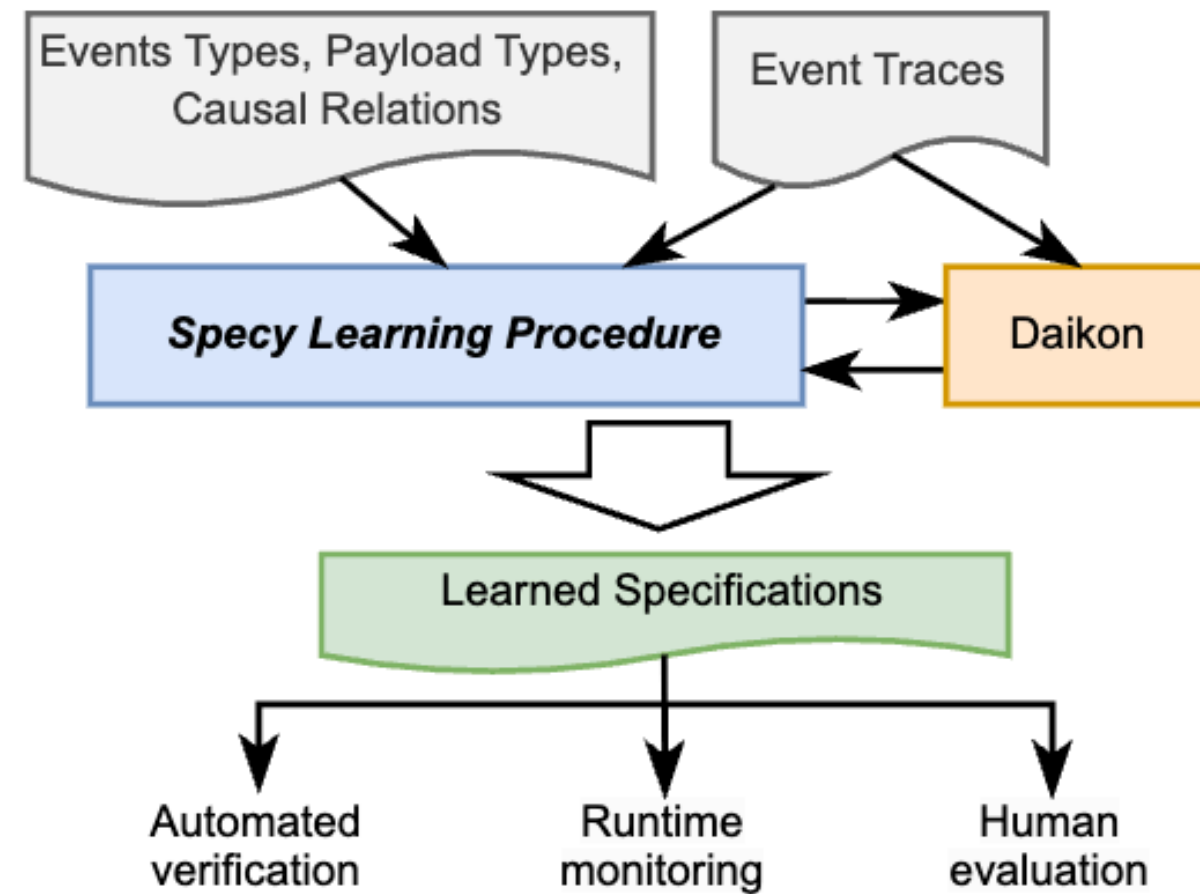
Table 3. Comparing a set of golden ($\varphi_U \in S_U$) and SPECY-learned ($\varphi_L \in S_L$) specifications

Relationship	What it indicates	Development actions
$S_U \subset S_L$	SPECY learns missing specifications	Add missing specifications to tests
$\varphi_L \Rightarrow \varphi_U$	Inadequate behavioral coverage in testing	Improve tests to relax/weaken φ_L
Other cases	Inadequate traces or search space Potential bug in the implementation	Diversify traces with more tests Rigorously debug with $S_U \setminus S_L$

Specy identified 2 atomicity specifications of a proprietary protocol overlooked by developers

Specy discovered an over-constrained behavior in the formal model

Summary



Learning specifications from event traces for both testing and verification

Search problem decomposed by the grammar of FOL formulas quantifying over events

Learned known specifications, inductive invariants
Helped uncover missing specifications and testing gap

$$\forall(e_i : \tau_i)^+ . \boxed{G} \rightarrow \exists(e_j : \tau_j)^* . \boxed{W} \wedge \boxed{H}$$



Paper: <https://dl.acm.org/doi/epdf/10.1145/3798209>



Code: <https://github.com/AD1024/PIInfer-OOPSLA-Artifact>

Backup Slides

Evaluations

Table 5. Number of specifications remaining after each pruning step. S_{raw} and S_{sem} columns show the number of specifications before and after all the pruning procedures, where the reduction ratio ($S_{\text{raw}}/S_{\text{sem}}$) is shown in RR column.

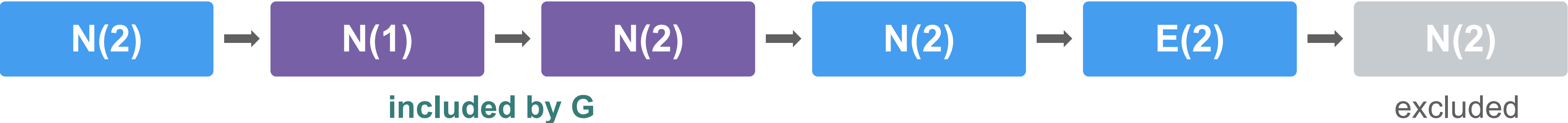
Benchmark	S_{raw}	S_{syn}	S_{sem}	RR	S_{false}
Ring Leader Elect.	479	52	30	15×	0
Consensus	361	28	28	13×	1
2PC	887	60	46	19×	9
Sharded KV	289	23	19	15×	0
Paxos	850	52	49	17×	5
Distributed Lock	740	91	77	9×	0
Vertical Paxos	1650	92	76	21×	5
Firewall	606	46	40	19×	4
Lock Server	399	44	35	11×	0
Chain Replication	533	40	37	14×	2
Raft	1080	73	58	18×	10
GlobalClock	368	37	37	9×	0
DBLeaderElect.	577	50	47	12×	9
MVCC-2PC	5789	334	236	24×	0

Pruning steps reduced by 15x on average while keeping the important specifications

For the proprietary benchmark MVCC-2PC, Specy learned two specifications that are missed by the developers

Universal: every matching pair

Enumerate G: $n < e$, and terms R: {e.nodeId, n.vote}



e.nodeId	n.vote
2	2
2	1
2	2
2	2



Daikon learns H

$$e.\text{nodeId} \geq n.\text{vote}$$

Equality fails on (2,1)

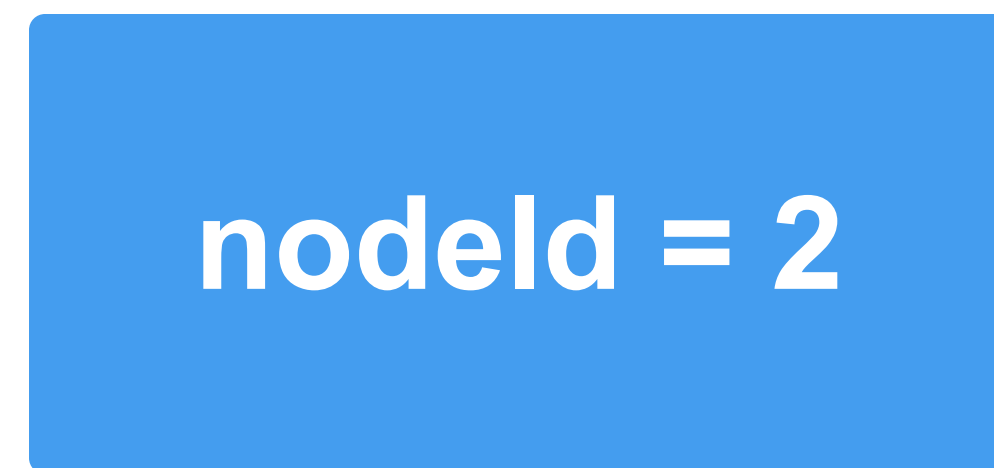
$$\forall e:E, n:N. \quad n < e \Rightarrow e.\text{nodeId} \geq n.\text{vote}$$

One trace shown; learning uses many runs

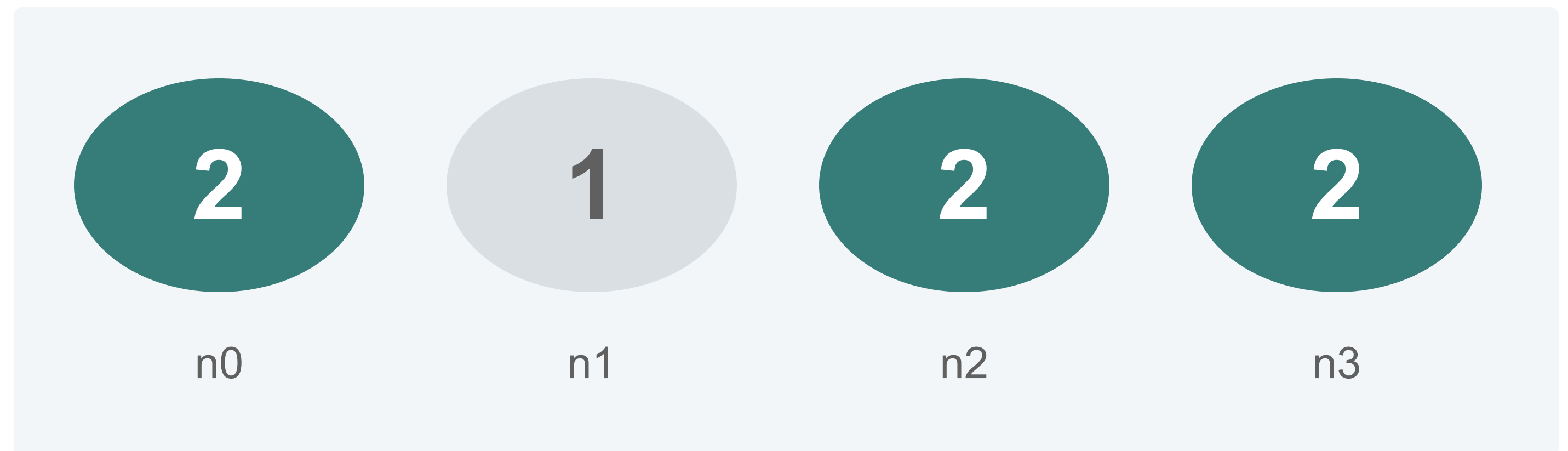
Existential: a witness per election

Enumerate G : true, W : $n < e$, and terms $\{e.\text{nodeId}, n.\text{vote}\}$

Election e



Witness set for this election



Daikon: membership

$2 \in \{2, 1, 2, 2\}$

$\forall e:E. \exists n:N. n < e \wedge e.\text{nodeId} = n.\text{vote}$

Separate witness sets for every election in every trace