Verifiable Hierarchical Protocols with Network Invariants on Parametric Systems

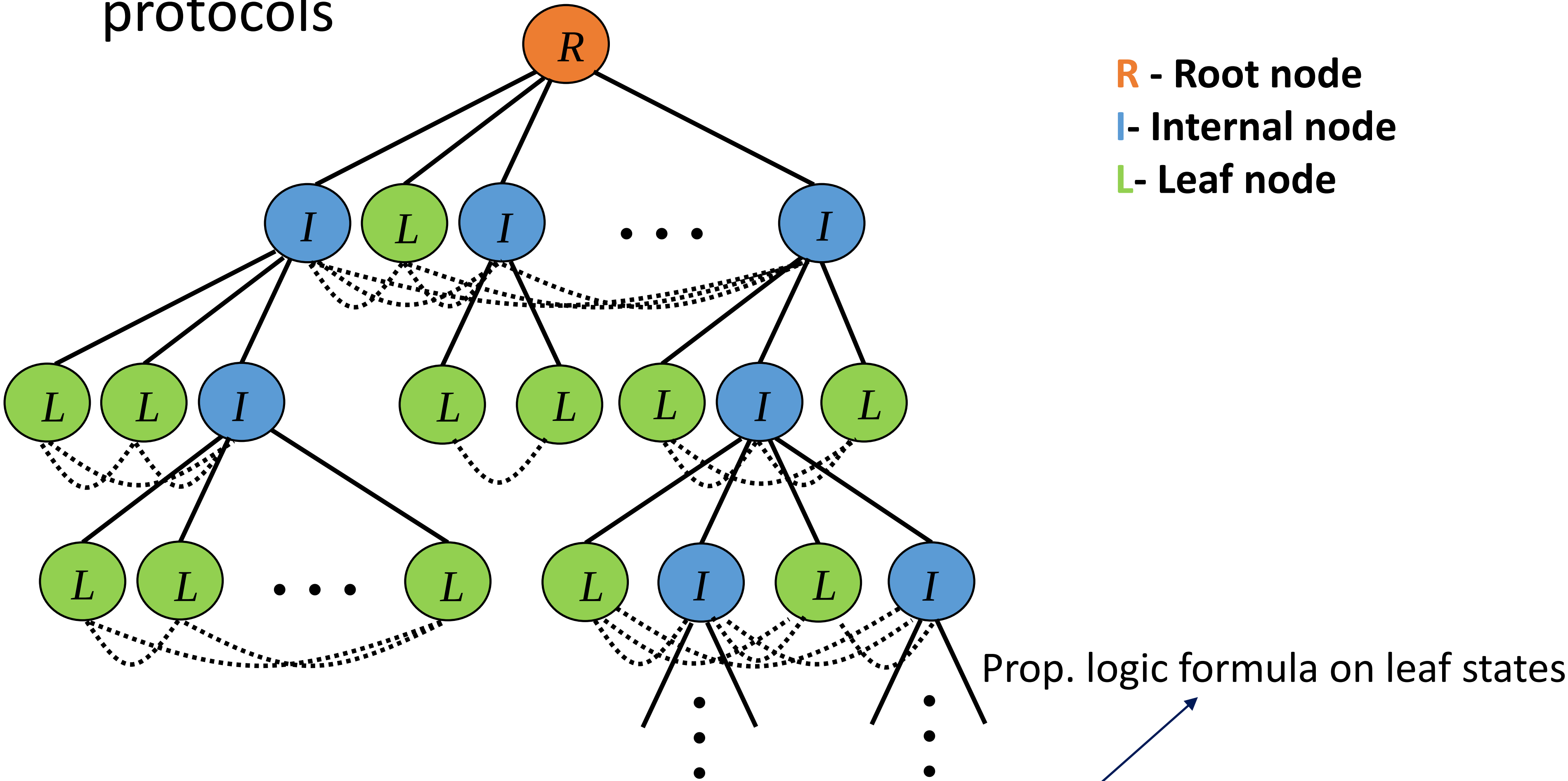
Opeo**luwa** Matthews, Jesse Bingham, Daniel Sorin

http://people.duke.edu/~om26/

FMCAD 2016 - Mountain View, CA

Problem Statement

- Goal: design and automated verification of hierarchical protocols



Safety property: $\forall L_1, \ldots, L_k . Distinct(L_1, \ldots, L_k) \Rightarrow P(L_1, \ldots, L_k)$

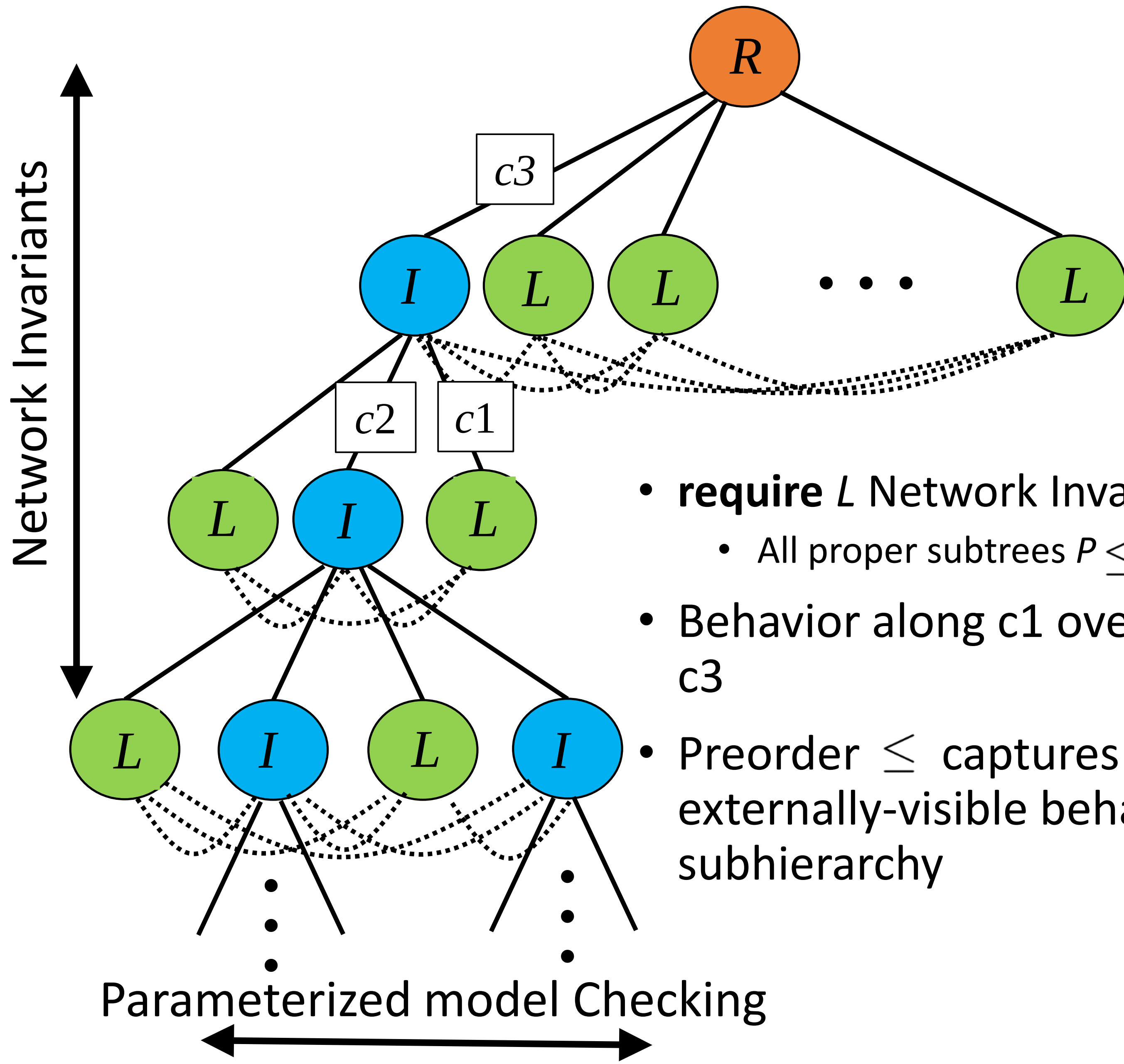
Problem Statement

- Parametric model checkers fall short
  - Suitable for flat protocols
  - Can’t handle asymmetry in hierarchical protocols

- Solution: Design specifically to fit automated techniques

- Formally specify class of transition systems – Neo
  - Require properties that enable automated safety verification
  - Key: Network invariants + parameterized verification

Illustration of our Approach

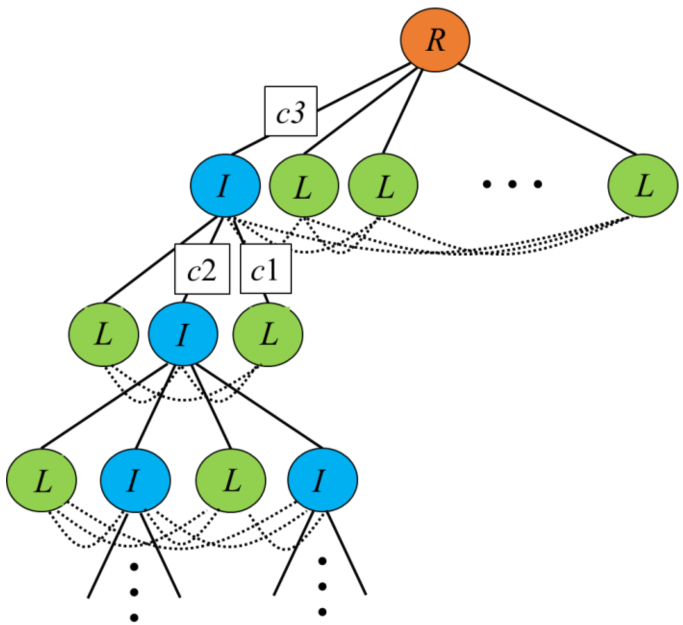


R - **Root node**
I- **Internal node**
L- **Leaf node**

- **require** *L* Network Invariant
  - All proper subtrees $P \leq L$
- Behavior along c1 over-approximates c2, c3
- Preorder $\leq$ captures states and externally-visible behaviors of subhierarchy

Neo Framework

- Neo formalized on I/O Automata (IOA) process theory
- Neo system is an IOA with specific properties for actions, composition, and executions
- 3 classes of IOA
  - **Internal node**
  - **Leaf node**
  - **Root node**



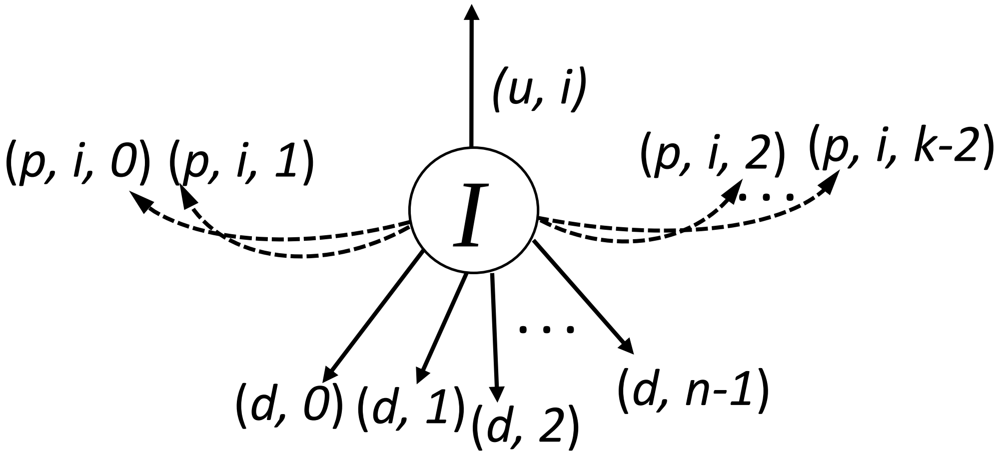
- Define 3 sets of actions
  - Upward actions – *U*
  - Downward actions – *D*
  - Peer-to-peer actions – *P*

Internal Node

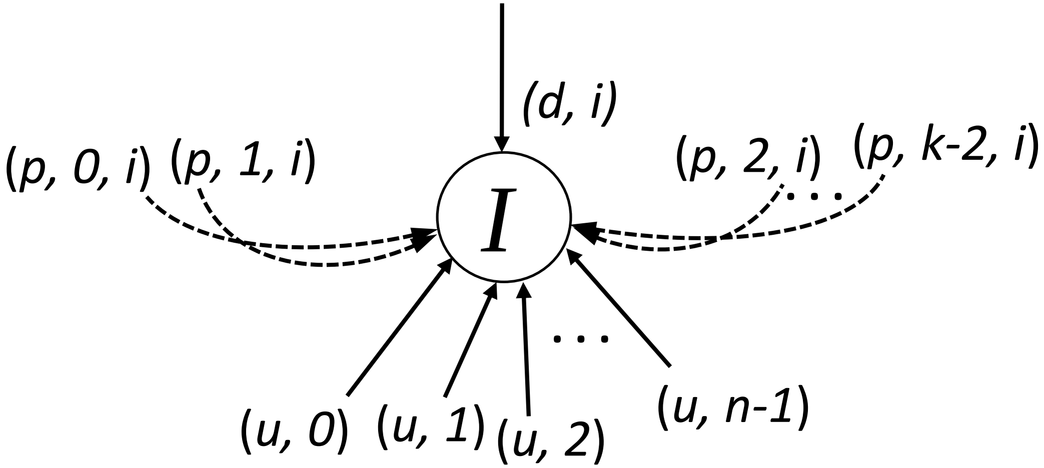
n-child *k*-peer ***Internal Node*** *I* is IOA that:

- Communicates with 1 parent, n children, k-1 peers, with index *i*

$u \in U, p \in P, d \in D$



output actions



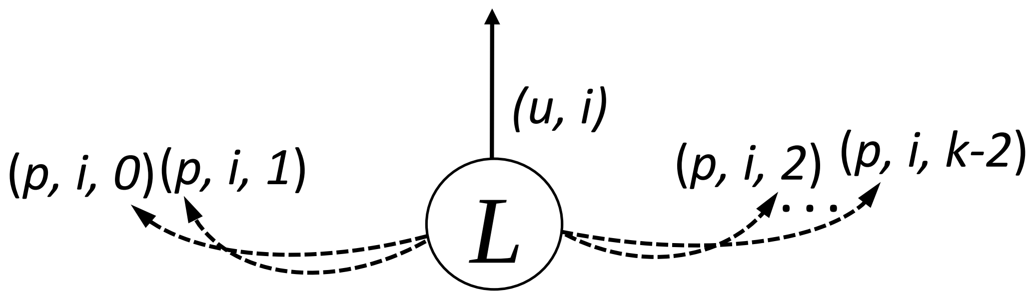
input actions

Leaf Node

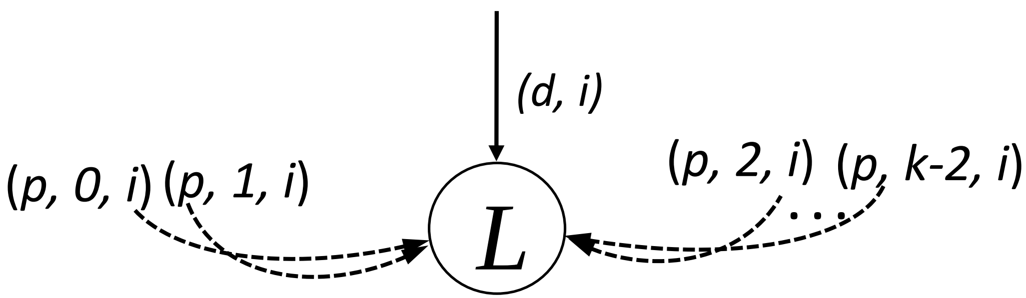
Leaf node *L* is 0-child, *k*-peer internal node:

- Communicates with 1 parent and k-1 peers, with index *i*

$u \in U, p \in P, d \in D$



output actions

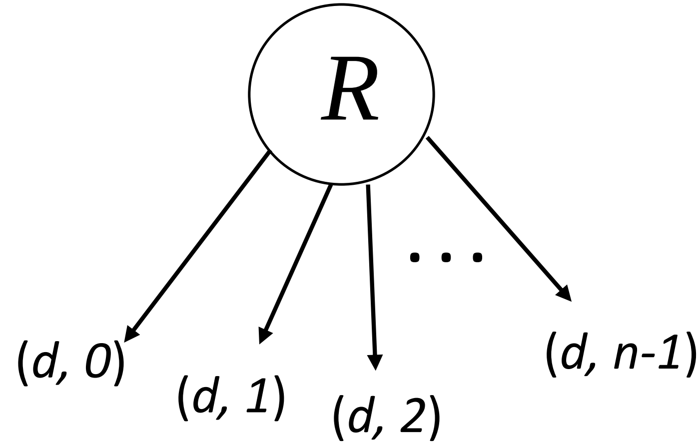


input actions

Root Node

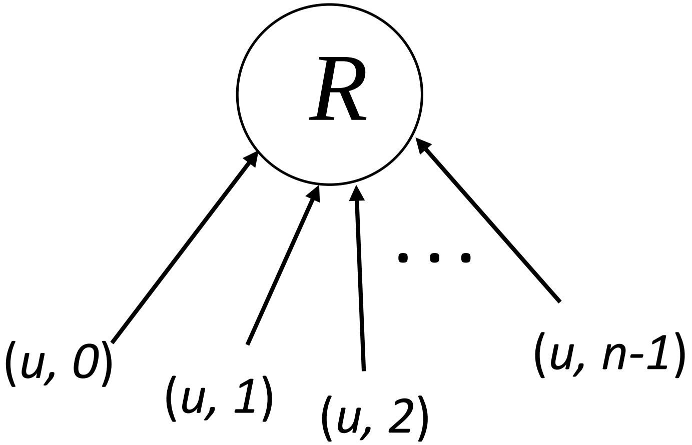
n-child **Root Node** *R* is IOA that:

- Communicates with n children



$d \in D, u \in U$

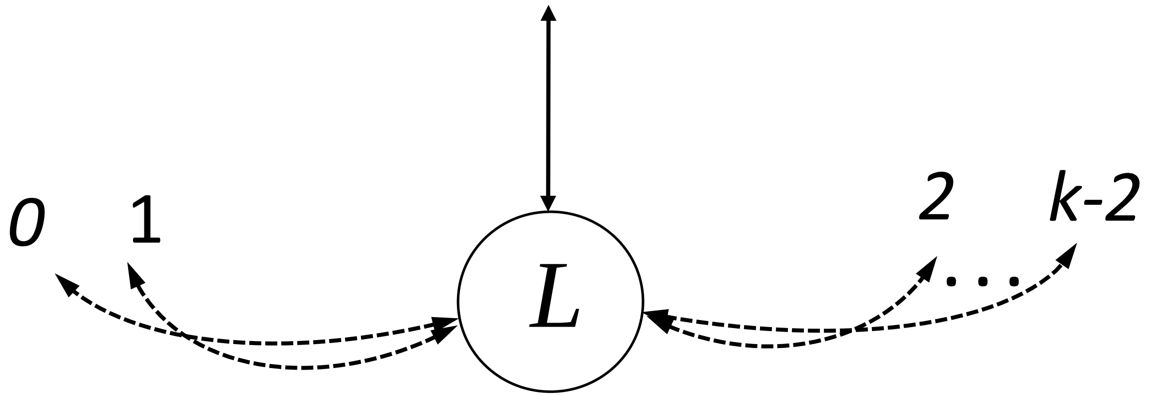
output actions



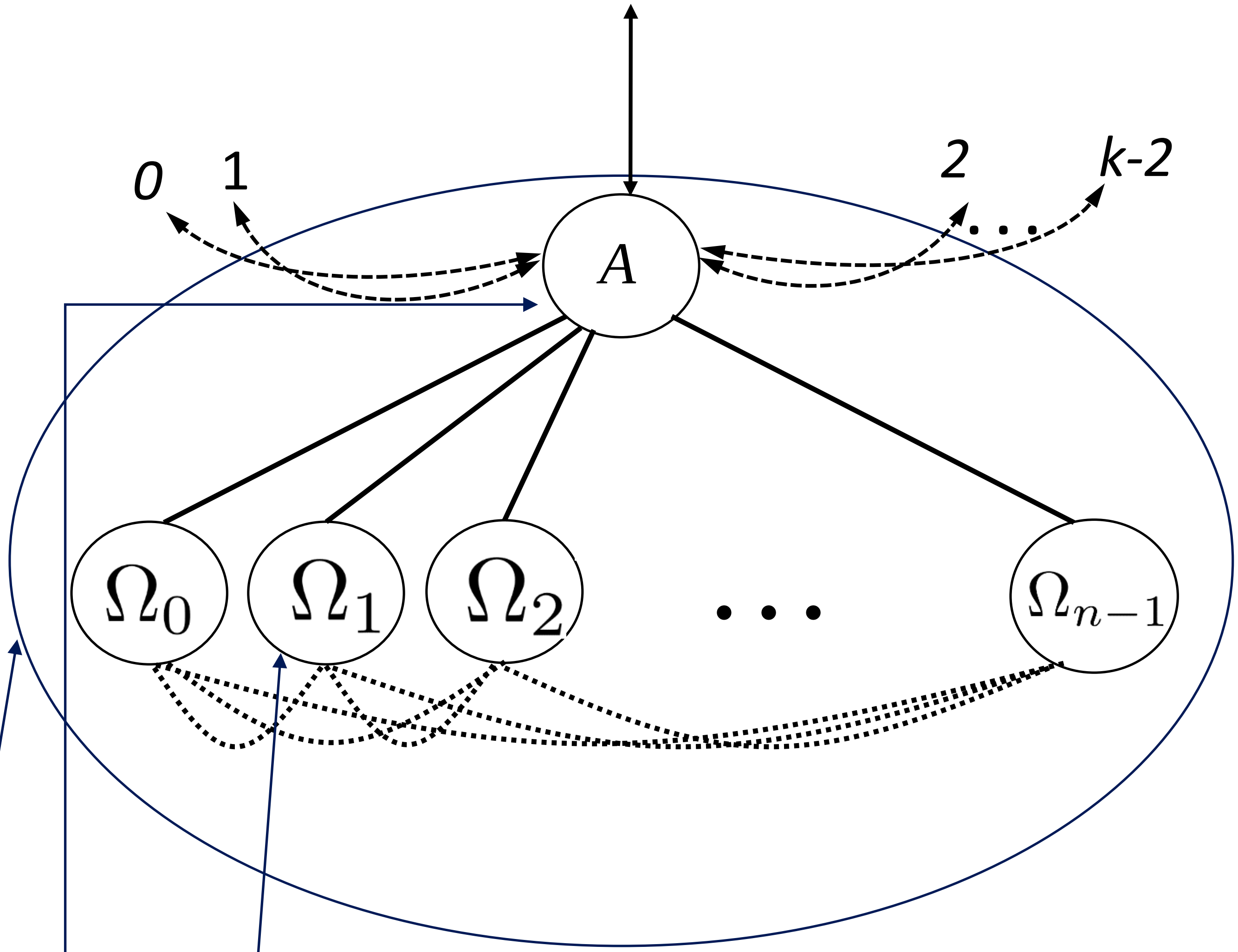
input actions

Defining Neo Systems

- *k*-peer Leaf *L* is *Open Neo System*, communicates with *k-1* peers



Defining Neo Systems

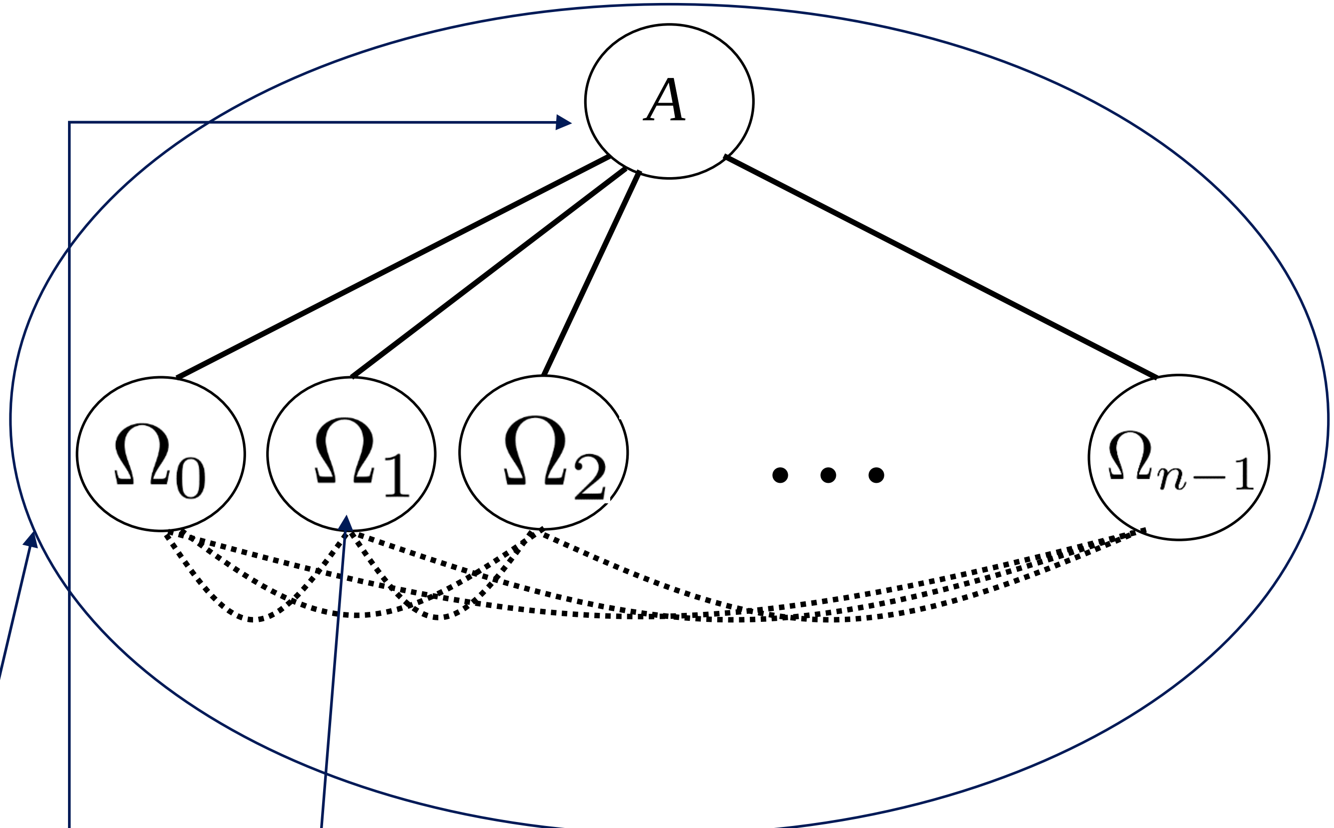


$$\Omega = A \cdot \prod_{i=0}^{n-1} \Omega_i$$

k-peer internal node *A* →

k-peer *Open* Neo System

Defining Neo Systems



$$\Omega = A \cdot \prod_{i=0}^{n-1} \Omega_i$$

A is root node →

Closed Neo System

Network Invariants on Neo Systems

- *Network Invariants* captures behavior of subhierarchies (open Neo systems)
  - Require: Every open Neo system must implement leaf wrt $\preceq$

- $\preceq$ captures summaries of states and executions
  - *Summary states*
  - *Summary functions*
  - *Summary sequences of executions*

Summarizing States – Nodes

- *Sum* is set of *summary states*, with special element *bad*
- Have sum_* functions for every Neo system to capture summary state of each subhierarchy
- For leaf *L*, $sum_L : states(L) \to Sum$
- For each n-child root or internal node *A*,

$$sum_A : states(A) \times Sum^n \to Sum$$

$bad \in \{s_0, \ldots, s_{n-1}\}$ implies $sum_A(s, s_0, \ldots, s_{n-1}) = bad$

Summarizing States – Neo systems

- For Neo system $\Omega = A \cdot \prod_{i=0}^{n-1} \Omega_i$

define $sum_\Omega : states(\Omega) \to Sum$ as

$$sum_\Omega(s_a, s_0, \ldots, s_{n-1}) = \\ sum_A(s_a, sum_{\Omega_0}(s_0), \ldots, sum_{\Omega_{n-1}}(s_{n-1}))$$

Neo Safety

$s \in states(\Omega)$ **safe** if $sum_{\Omega}(s) \neq bad$

Ω **safe** if all reachable states are safe

Summarizing Executions

- Generate *summary sequence* of exec *e* of Ω as follows:

state

action

$$e = s_0, \alpha_1, \ldots, \alpha_k, s_k$$

summarize states

$$sum_\Omega(s_0), \alpha_1, \ldots, \alpha_k, sum_\Omega(s_k)$$

Remove "silent" terms that don't affect safety

Delete all $\alpha_i, sum_\Omega(s_i)$ such that

$\alpha_i \in int(\Omega)$ and $sum_\Omega(s_i) = sum_\Omega(s_{i-1})$

Neo Preorder Definition

- Need preorder for network invariants
- Given 2 open Neo systems Ω_1, Ω_2

$\Omega_1 \preceq \Omega_2$ implies for all executions e_1 of Ω_1

there exists execution e_2 of Ω_2

such that $sum(e_1) = sum(e_2)$

Theoretical Result

Theorem 1. (Every Neo system is safe.) *Suppose that for each n-child internal or root node A, $\Omega_L = A \cdot \prod_{i=0}^{n-1} \phi_i(L)$ is safe. Furthermore, suppose that if A is an internal node, then $\Omega_L \preceq L$. Then all Neo systems are safe.*

Antecedents:

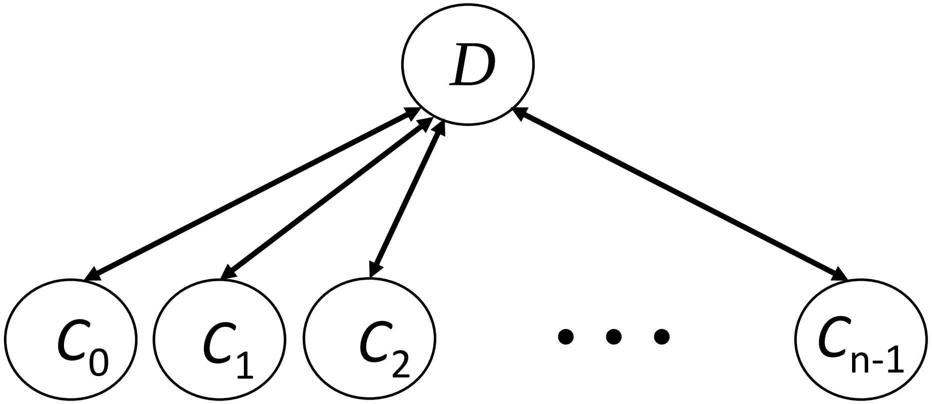
1. Every 1-level (all-leaf) open or closed neo system safe
2. Every 1-level (all-leaf) open neo system implements leaf

- If 1. and 2. can be performed in parametric model checker

<u>Implication:</u> Reduced 2-dimensional verification problem to 1 dimension

Case Study

- We design and verify hierarchical coherence protocol *NeoGerman*
- Modify (originally flat) German protocol into Neo hierarchy
- Coherence defined on predicates {*E*,*S*,*I*} on cache states
- 2 private caches in (E, E) or (E, S) prohibited



$$\Omega_R$$

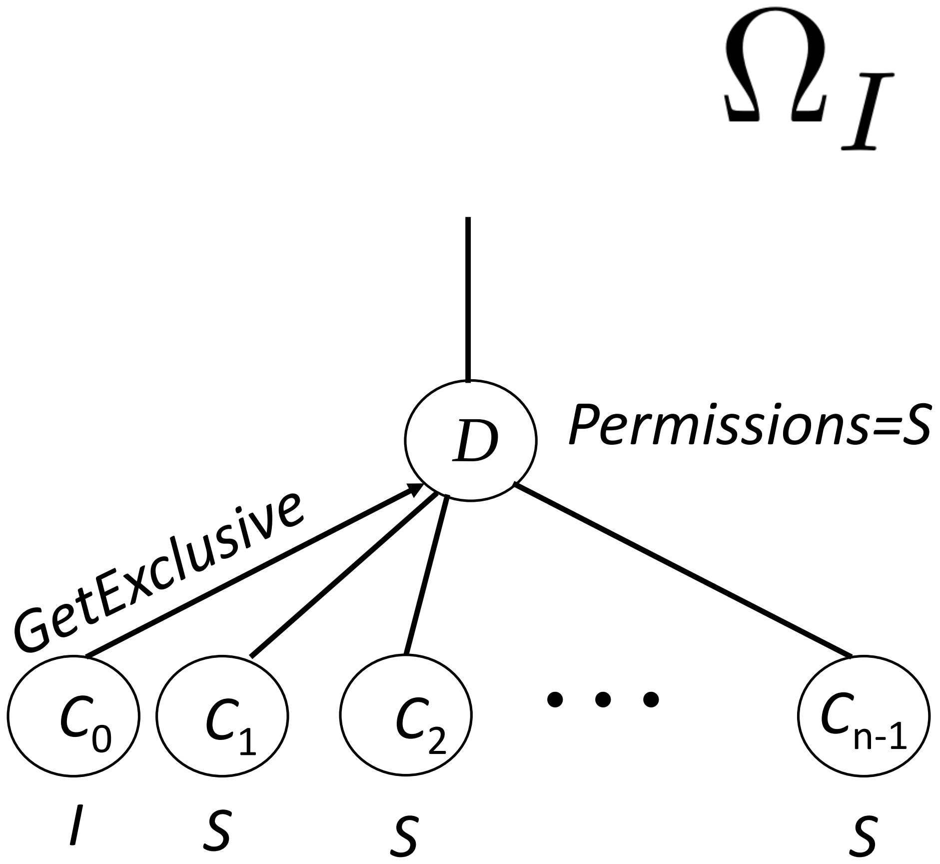
NeoGerman Protocol

- Root node is same as directory of German protocol
  - Ω_R is closed Neo system

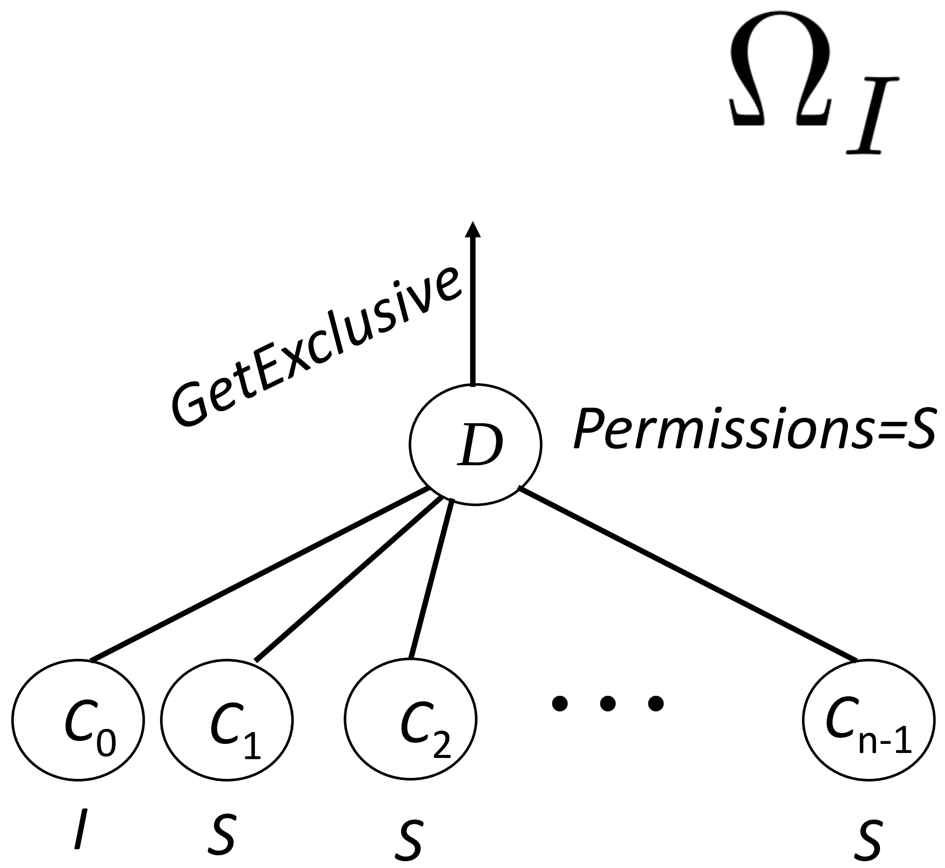
- To get open Neo system Ω_I, modify directory to be internal node (talk to parent)

- Internal node has state variable *Permissions*, captures summary of subhierarchy

NeoGerman Protocol Illustration

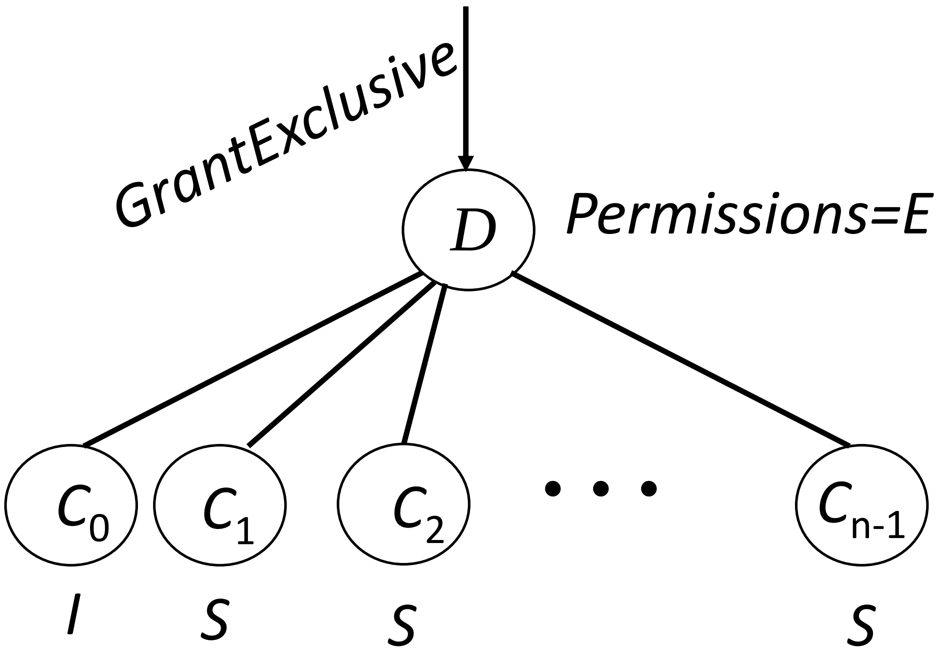


NeoGerman Protocol Illustration

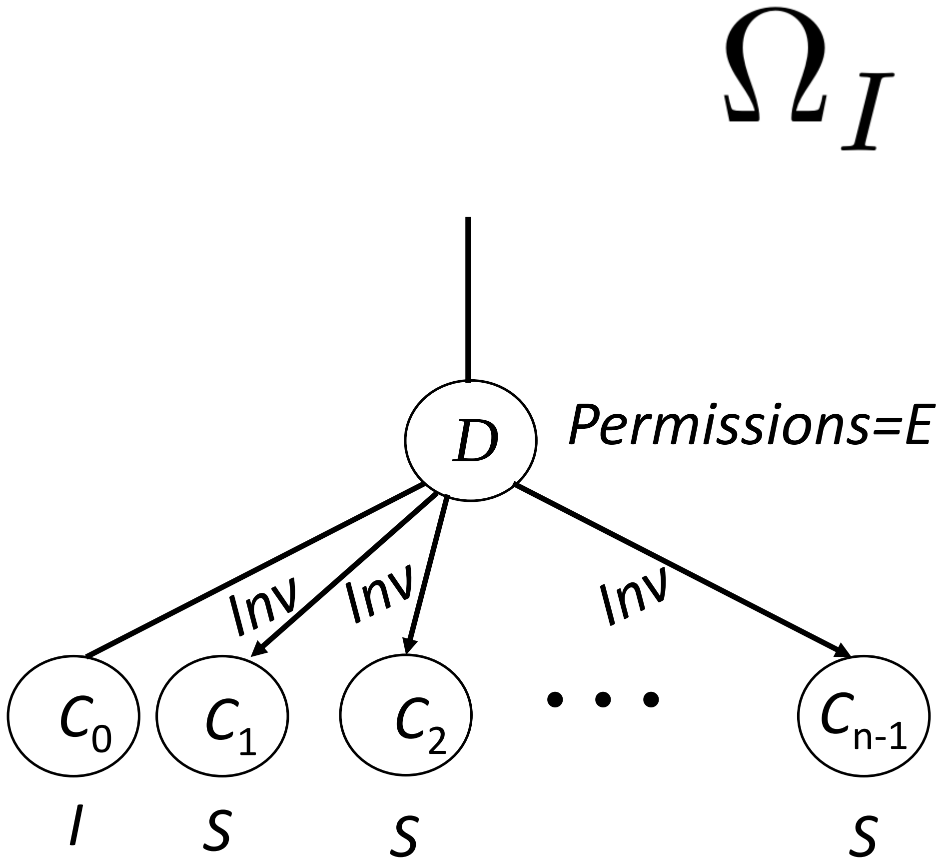


NeoGerman Protocol Illustration

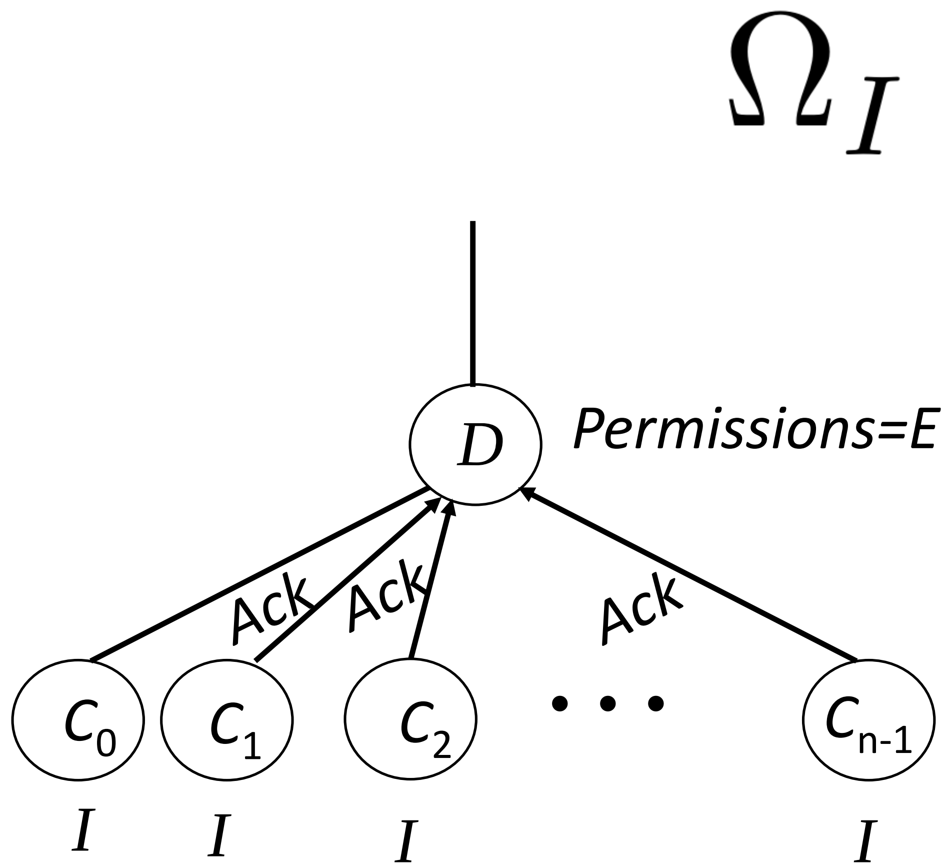
$$\Omega_I$$



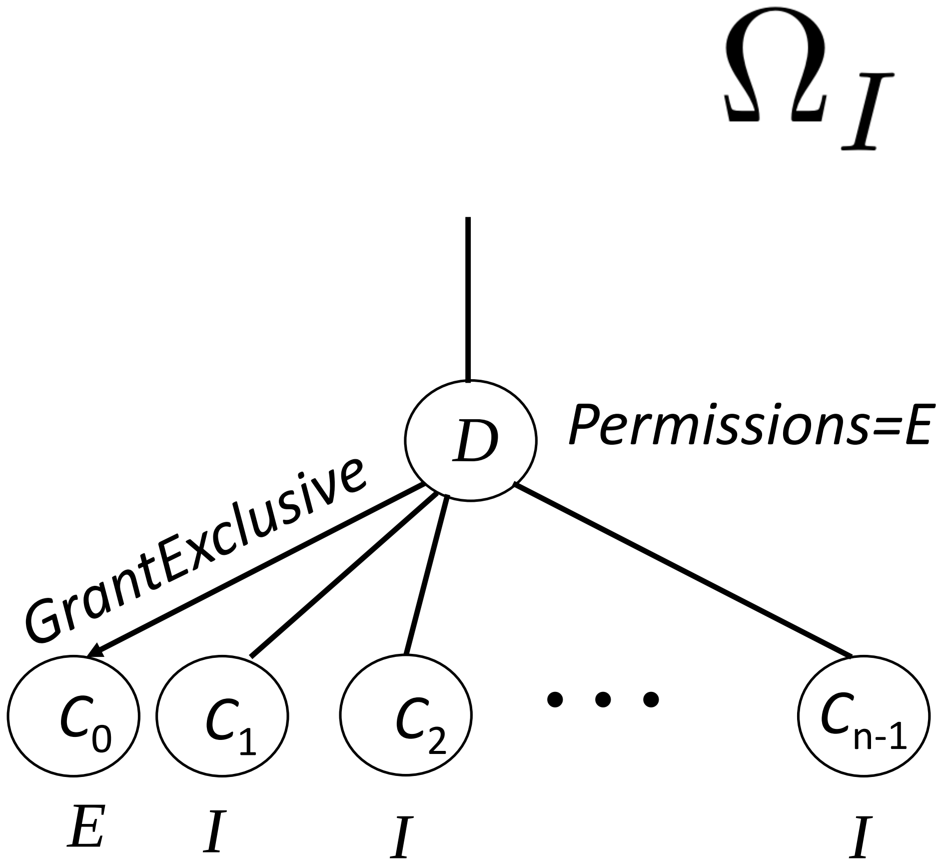
NeoGerman Protocol Illustration



NeoGerman Protocol Illustration



NeoGerman Protocol Illustration



NeoGerman Summary Functions

- Preorder, safety defined w.r.t summary functions
- Need: if safety violated → function returns *bad*

- Create ordering < on *Sum*: I < S < E < *bad*
- 2 constraints on sum_A:

1) $sum_A(s_a, s_0, \ldots, s_{n-1}) = bad$ if $s_i = E$ and $s_j \neq I$
2) $s_i \leq sum_A(s_a, s_0, \ldots, s_{n-1})$

- Output of sum_A always returns value of *Permissions*

Verification Methodology

- All verification done automated in Cubicle parametric model checker
  - SMT-based, backward reachability
  - Similar syntax to Murϕ, guard/action semantics
  - Clean, promising results, great support!

- Must prove antecedents of Theorem 1
  1. Ω_R and Ω_I safe – express in Cubicle
  2. $\Omega_I \preceq L$ (preorder) trickier

Preorder Proof

- Model both Ω_I and *L* in same Cubicle program
- Force Ω_I and *L* to transition in lockstep, starting with Ω_I
- Have variables *O_action* and *L_action*, represent IOA *action*, updated after each transition, internal actions updated to λ (silent)
- One each transition, there needs to exist *L* step that "matches" Ω_I step
    - To reveal witness step, conjunct expression to *L* guards, forcing *L* take "right" step w.r.t Ω_I step.
    - Note: conjunction can only restrict *L* behavior

Preorder Proof

After each Ω_I step, Cubicle checks:

- There exists *L* action that can fire
  - Cubicle safety prop: Disjunction of all *L* guards is true

After each pair of Ω_I and *L* steps, Cubicle checks:

- *O_action=L_action,* summary state outputs match

What Safety Properties can Neo Verify?

- Define class of FOL formulas we can verify are invariant

Given set $LP = \{p_1, \ldots, p_m\}$ of predicates on leaf states and proposition logic formula $P(L_1, \ldots, L_k)$ over atoms of form $p_j(L_i)$

- We can verify all safety properties of the form:

$$\forall L_1, \ldots, L_k . Distinct(L_1, \ldots, L_k) \Rightarrow P(L_1, \ldots, L_k)$$

- E.g., *LP={E,S,I}* $\forall L_1, L_2 . Distinct(L_1, L_2) \Rightarrow (E(L_1) \Rightarrow I(L_2))$

- We provide summary function guaranteed to verify all such safety properties

Future Work

- Industrial-strength hierarchical coherence protocol
  - Request forwarding
  - MESI coherence permissions
  - Support for unordered networks
- Distributed lock management
  - Richer permissions (NL, CR, CW, PR, PW, EX)
- Dynamic power management
  - Natural hierarchy in datacenters

Conclusions

- Neo framework enables design and automated verification of hierarchical protocols safe for arbitrary configurations
- Case study: Design and verify hierarchical coherence protocol
  - Correct for arbitrary size, depth, branching degrees per node
  - Proof completely automated in parametric model checker
- Prove observational preorder in parametric setting