https://refinery.tools

Refinery: Reasoning about modeling problems with partial models

Kristóf Marussy, *Oszkár Semeráth,*
Attila Ficsor, Dániel Varró

MŰEGYETEM 1782 ftsrg Critical Systems Research Group

McGill UNIVERSITY

LINKÖPING UNIVERSITY

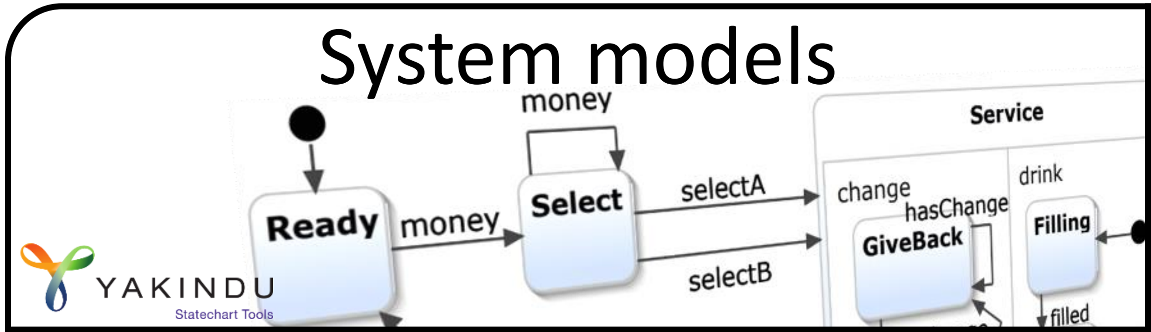
WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

amazon research awards

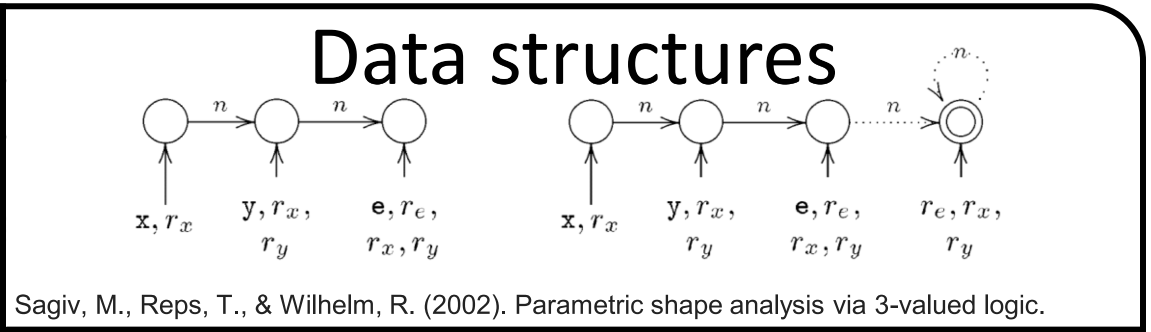
Office of Naval Research Science & Technology

Modeling with Graphs

- Graph based models are widely used in software engineering

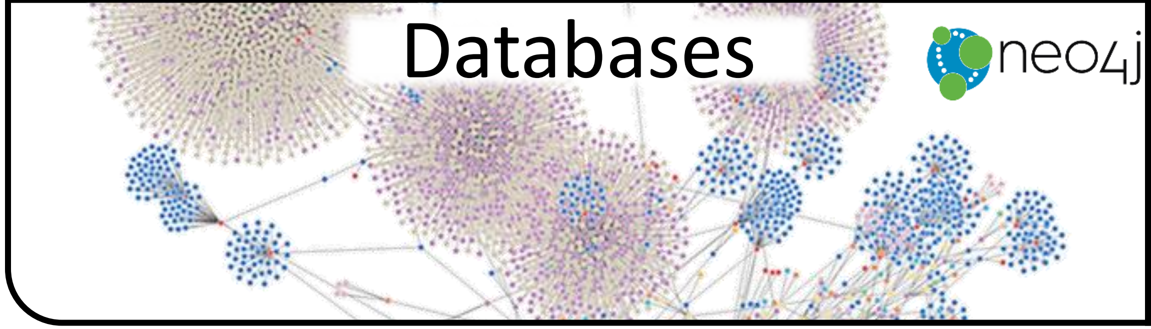


System models

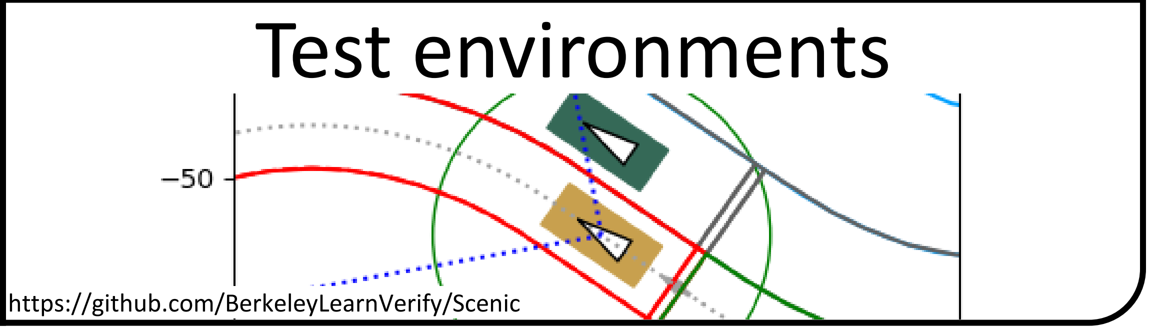


Data structures

Sagiv, M., Reps, T., & Wilhelm, R. (2002). Parametric shape analysis via 3-valued logic.



Databases



Test environments

https://github.com/BerkeleyLearnVerify/Scenic

- Testing, benchmarking or design space exploration scenarios

Generating **(consistent | realistic | diverse | scalable)** models

Hands-on demo

- Code examples available at https://refinery.tools/learn/tutorials/dlt/
- Watch out for numbered code examples!

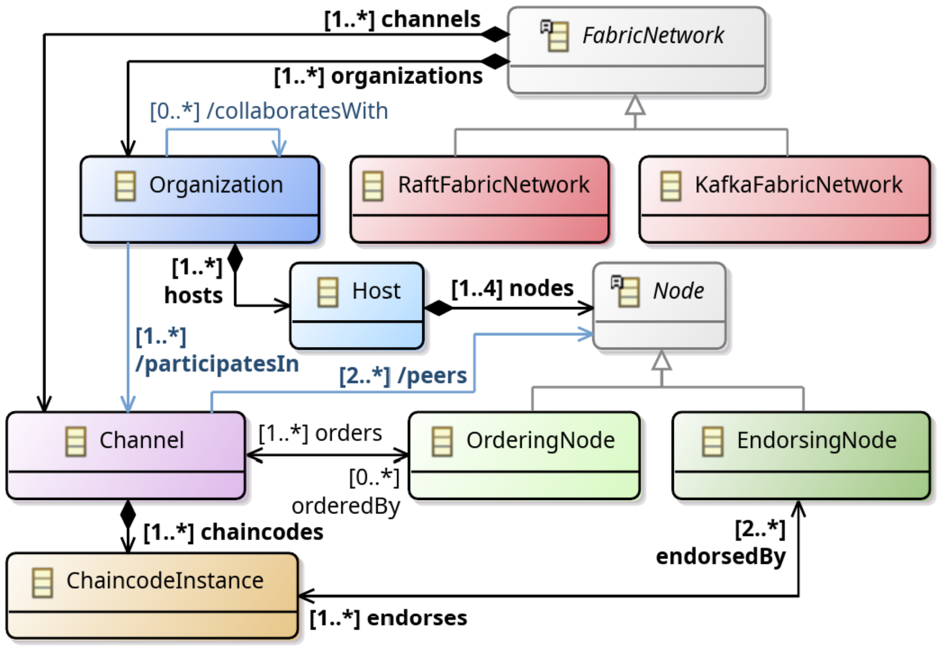
→ Example 1

Graph Structure: Hyperledger Architectures

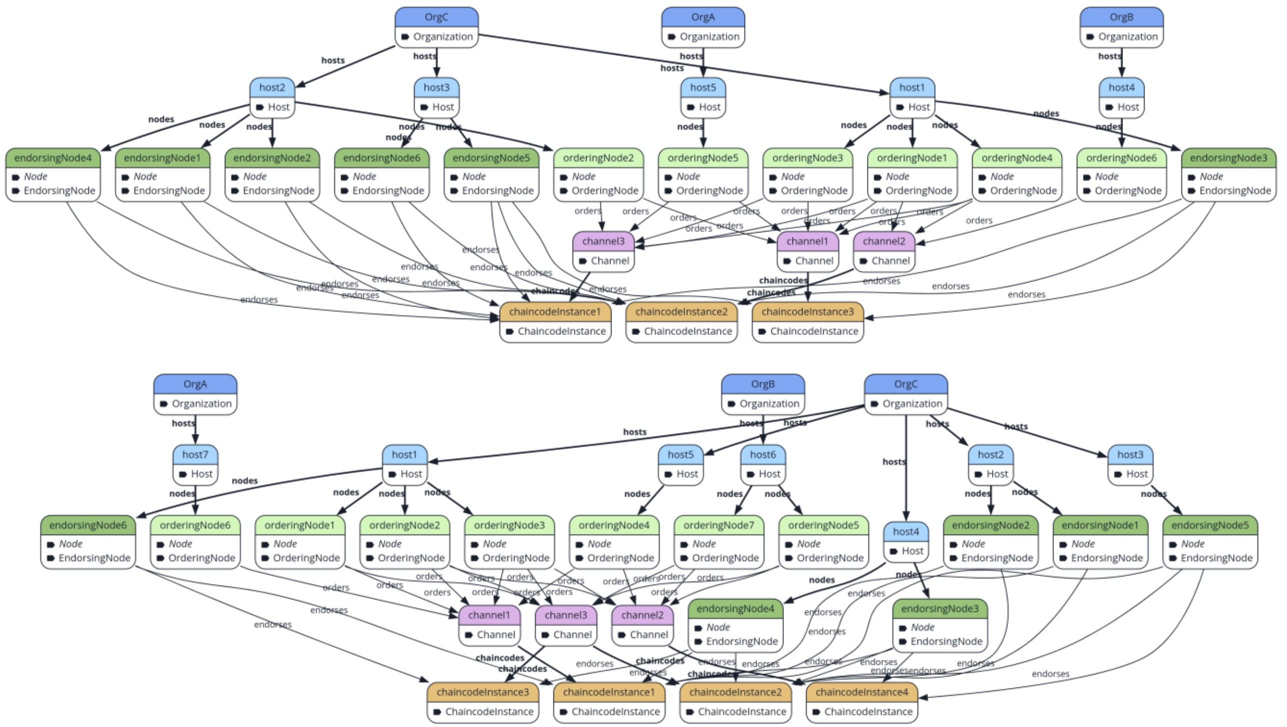
- Typical modeling workflow: **metamodel → instance model**
- **Example:** Organizations, nodes, and channels

→ Example 2

Metamodel



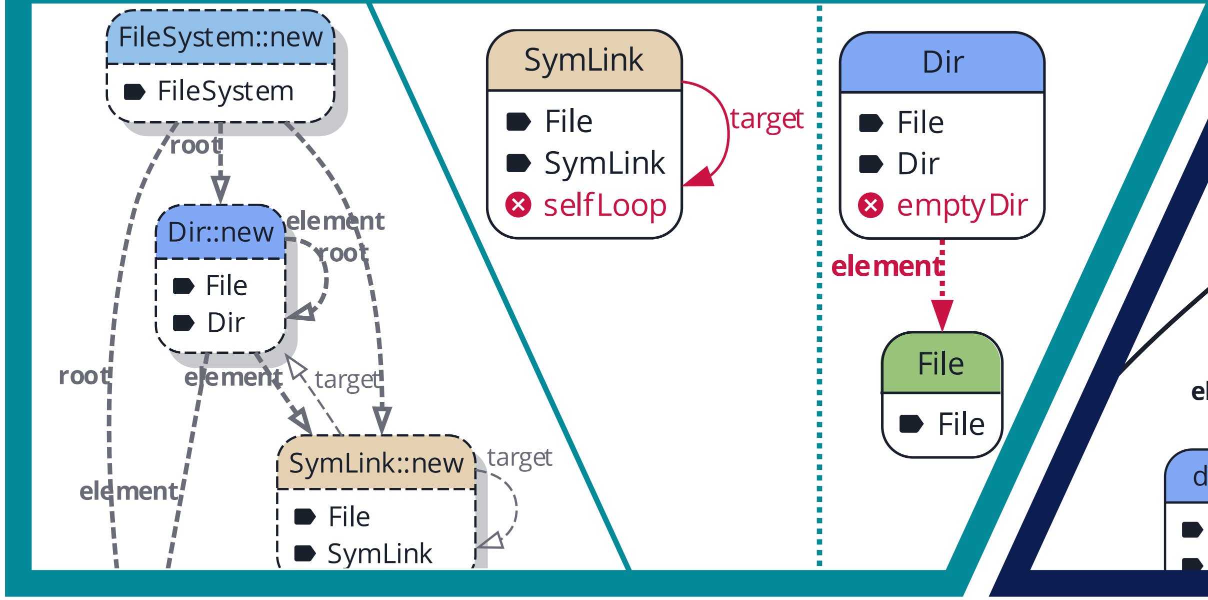
Instance Model



Models and Partial Models

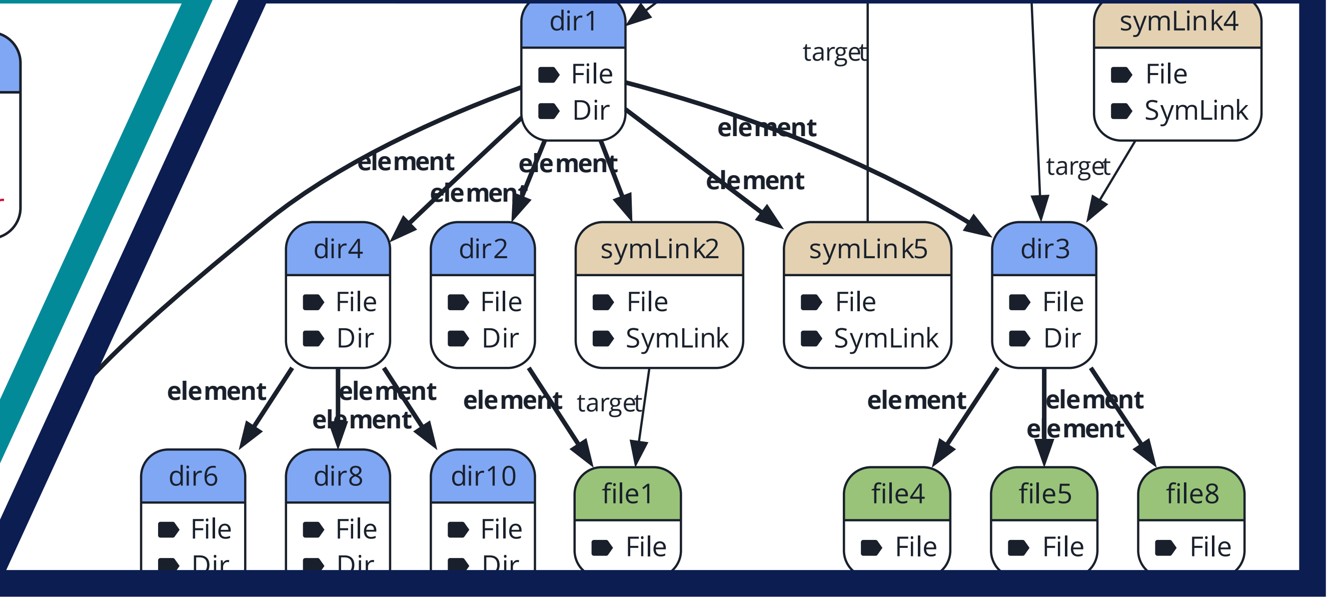
Input

Abstract models
(Metamodel + Constraints)



Output

Concrete models
(Labelled graphs)



Model generation: exploration process that gradually reduces uncertainty

Overview of the tutorial

Domain specification (metamodel)

- Define classes (nodes) and relations (edges)

4-valued partial model specification

- Seed partial model to extend (with reasoning)

Constraint specification

- Graph query language (inspired by Datalog / VIATRA Query)

Reasoning with propagation rules

- Derive new facts from the state of the knowledge base with custom rules

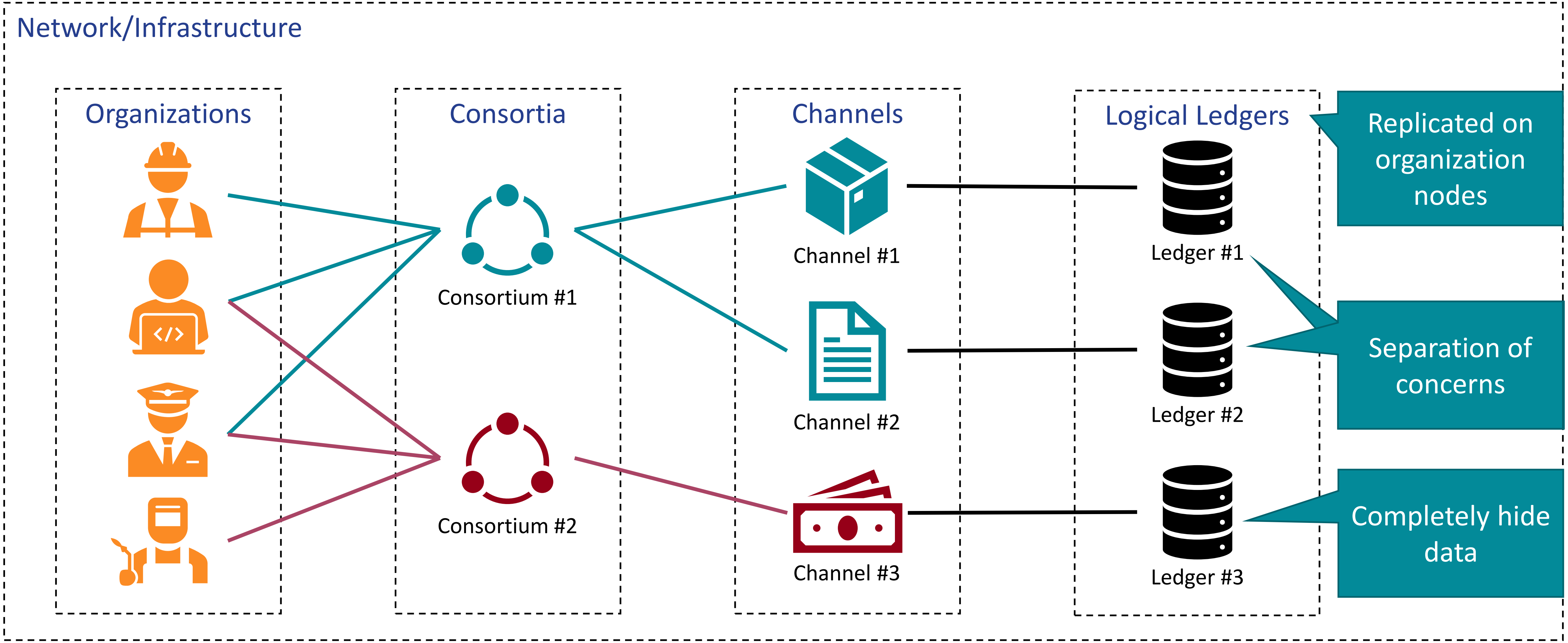
Case study

Architecture design for consortial Distributed Ledger Technology

One Network, Multiple Ledgers – Consortial DLT

HYPERLEDGER FABRIC

A Fabric network is a set of independently maintained ledgers.

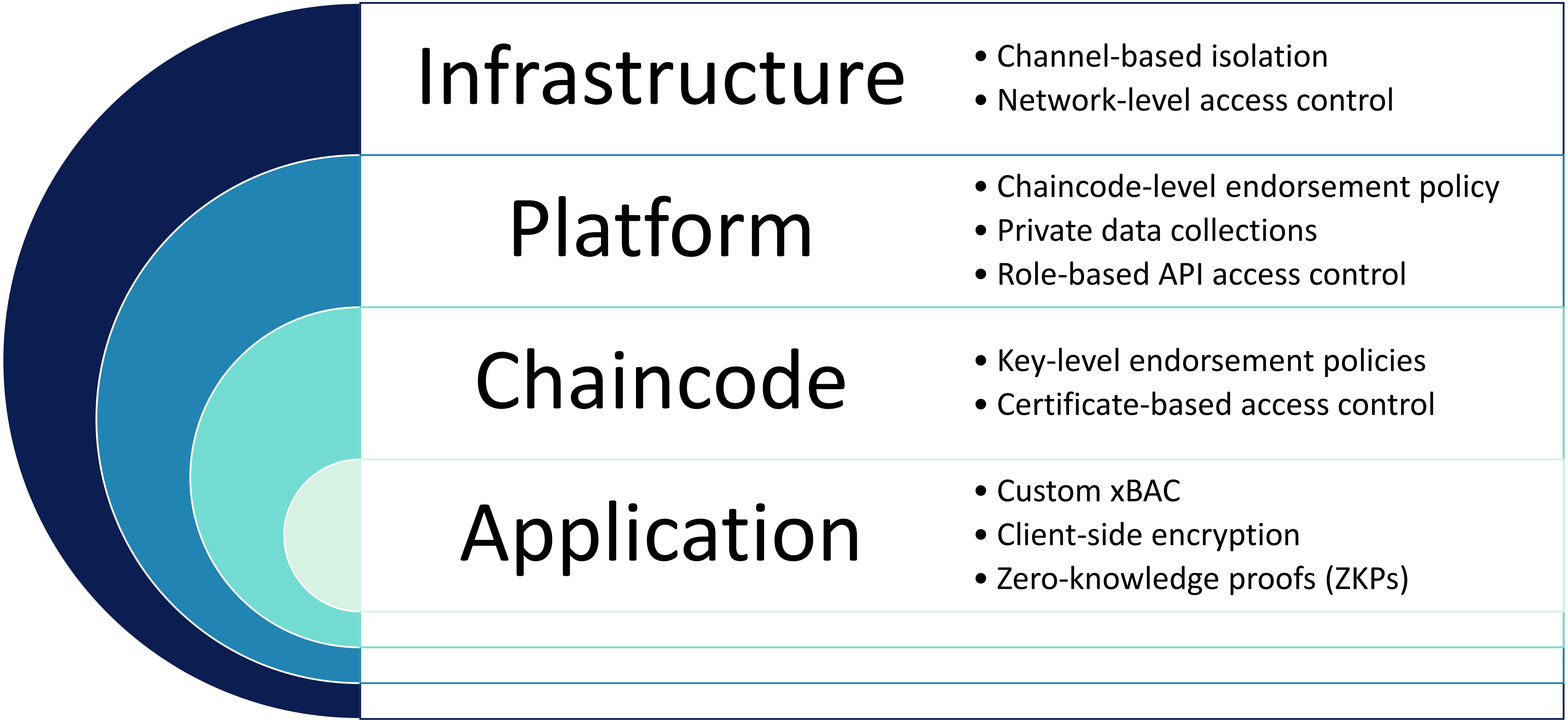


Replicated on organization nodes

Separation of concerns

Completely hide data

Engineered Privacy/Confidentiality in Fabric



Infrastructure

- Channel-based isolation
- Network-level access control

Platform

- Chaincode-level endorsement policy
- Private data collections
- Role-based API access control

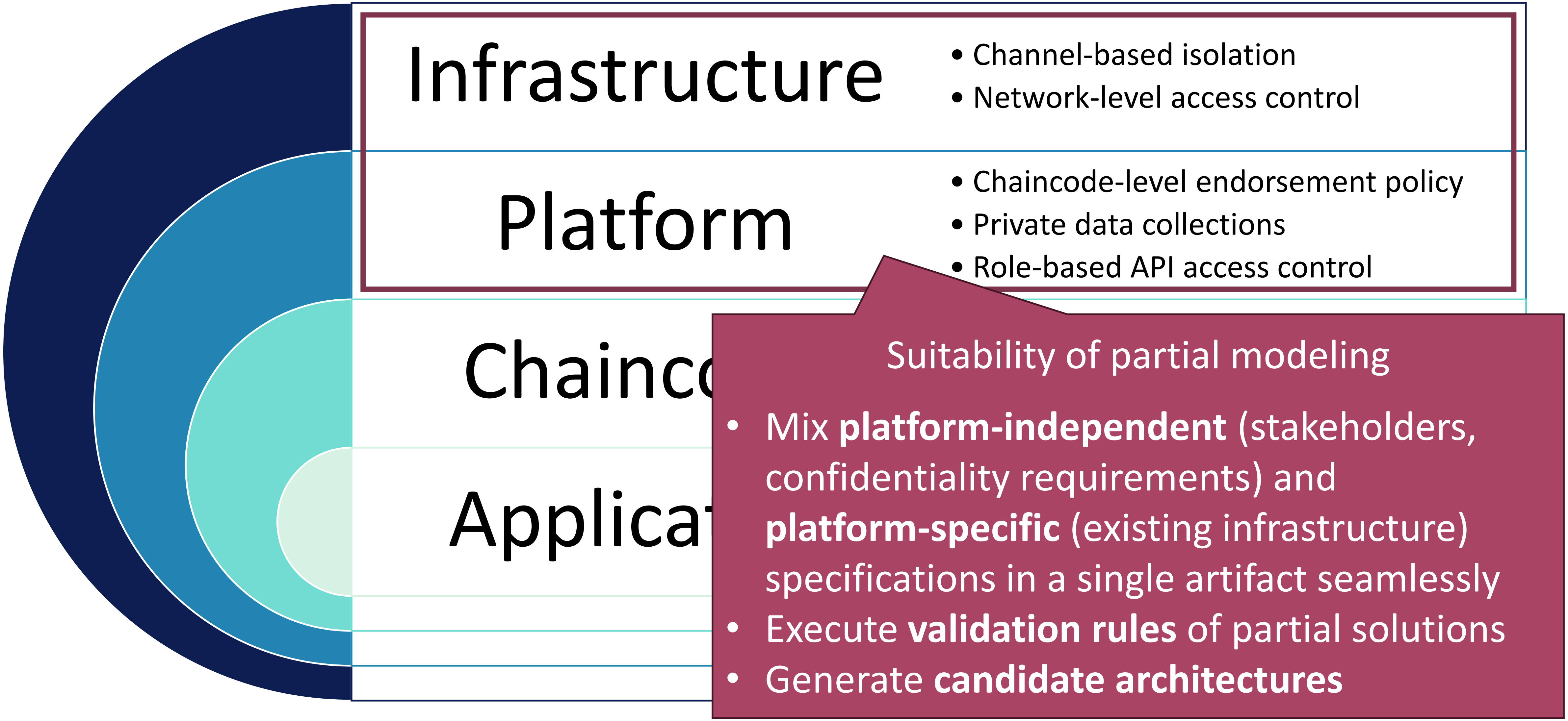
Chaincode

- Key-level endorsement policies
- Certificate-based access control

Application

- Custom xBAC
- Client-side encryption
- Zero-knowledge proofs (ZKPs)

Engineered Privacy/Confidentiality in Fabric



Infrastructure

- Channel-based isolation
- Network-level access control

Platform

- Chaincode-level endorsement policy
- Private data collections
- Role-based API access control

Chainco

Applicat

Suitability of partial modeling

- Mix **platform-independent** (stakeholders, confidentiality requirements) and **platform-specific** (existing infrastructure) specifications in a single artifact seamlessly
- Execute **validation rules** of partial solutions
- Generate **candidate architectures**

Domain-specific partial models

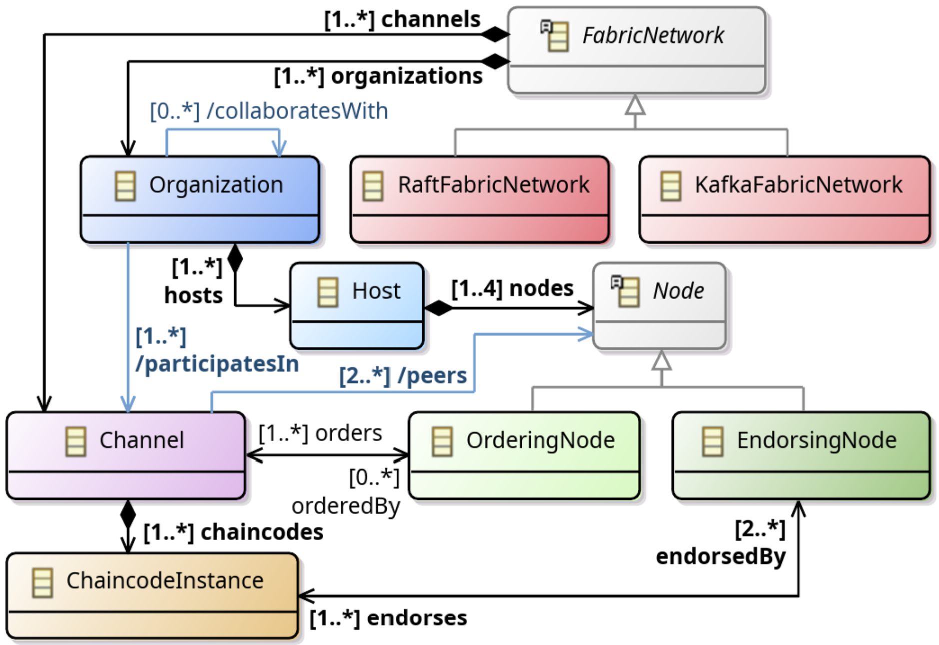
4-valued logic for reasoning about graph models

Graph Structure: Hyperledger Architectures

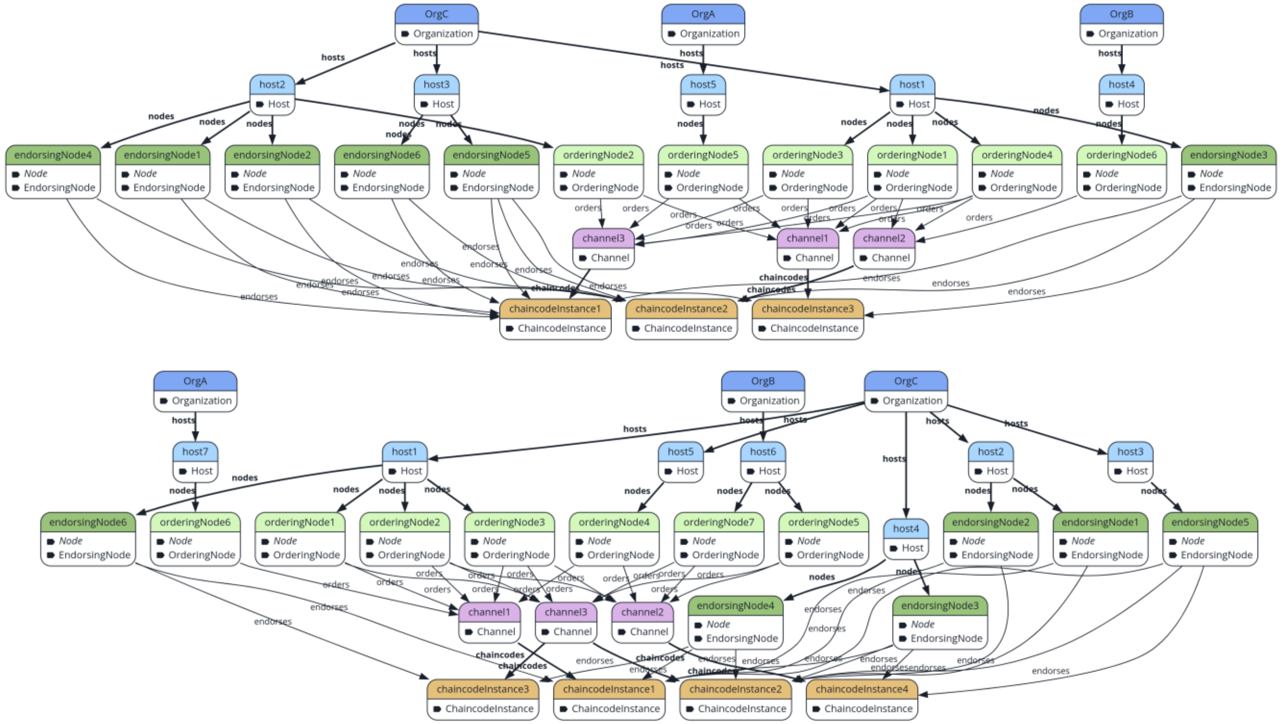
- Typical modeling workflow: **metamodel → instance model**
- **Example:** Organizations, nodes, and channels

→ Example 2

Metamodel



Instance Model



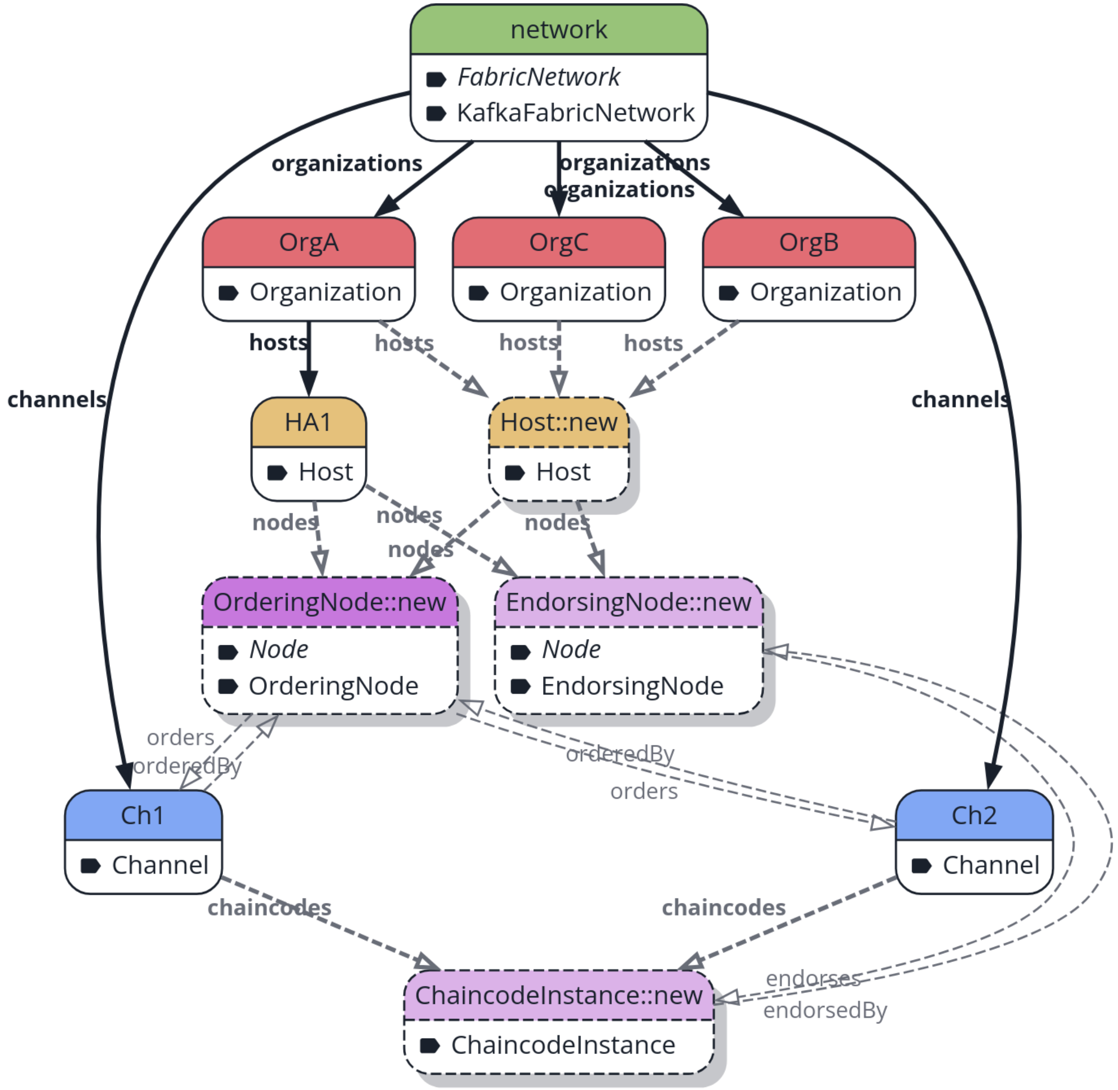
Partial Modeling with 4-valued Logic

- Represent all potential extensions with **uncertainty**
- Logic abstraction:

  ■TRUE | False |
  □Unknown | ⊗Error

  - 4-valued **exists**: added or removed
  - 4-valued **equals**: merging or splitting
- **Refinement:** reduces uncertainty → concrete models

→ Example 3



Partial Modeling with 4-valued Logic

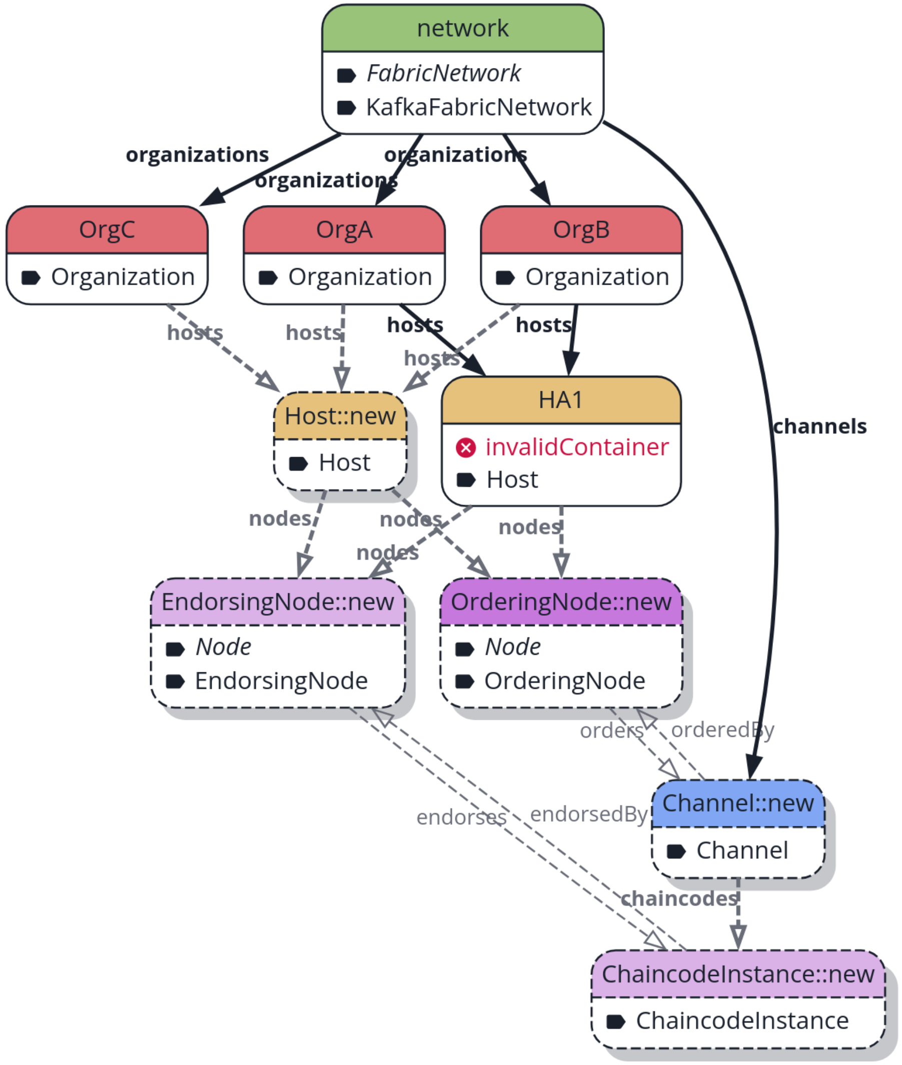
- Represent all potential extensions with **uncertainty**
- Logic abstraction:

  ➡TRUE | False | ⇨Unknown | ⊗Error

  – 4-valued **`exists`**: added or removed

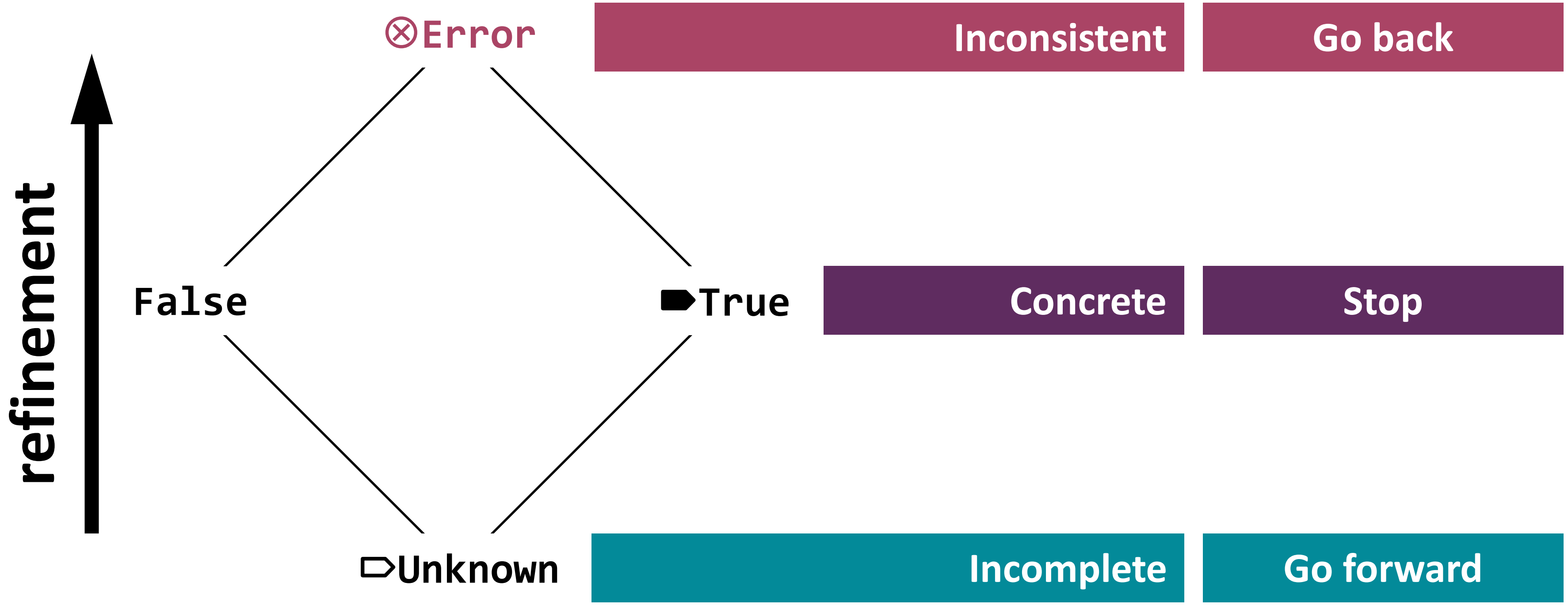
  – 4-valued **`equals`**: merging or splitting
- **Refinement:** reduces uncertainty → concrete models

→ Example 4



Refinement: 4-valued Logic

- Model generation is executed with respect to model refinement



refinement

⊗Error

Inconsistent | Go back

False

■True

Concrete | Stop

□Unknown

Incomplete | Go forward

E.g.: $\texttt{person(\_,\_):}\mathbf{unknown} \xrightarrow{\texttt{+true}} \texttt{person(\_,\_):}\mathbf{true}$

$\texttt{person(\_,\_):}\mathbf{true} \xrightarrow{\texttt{+false}} \texttt{person(\_,\_):}\mathbf{error}$

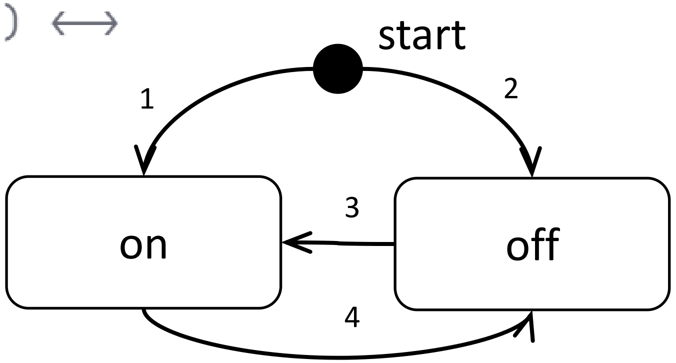
Constraint specification

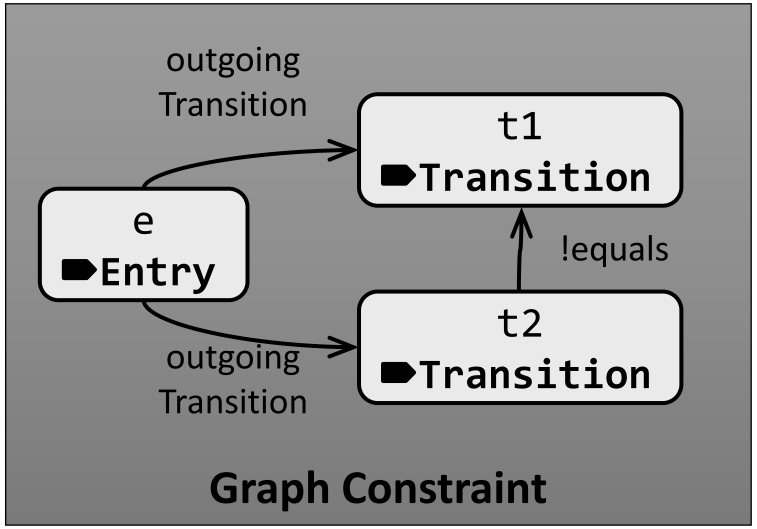
Using graph queries

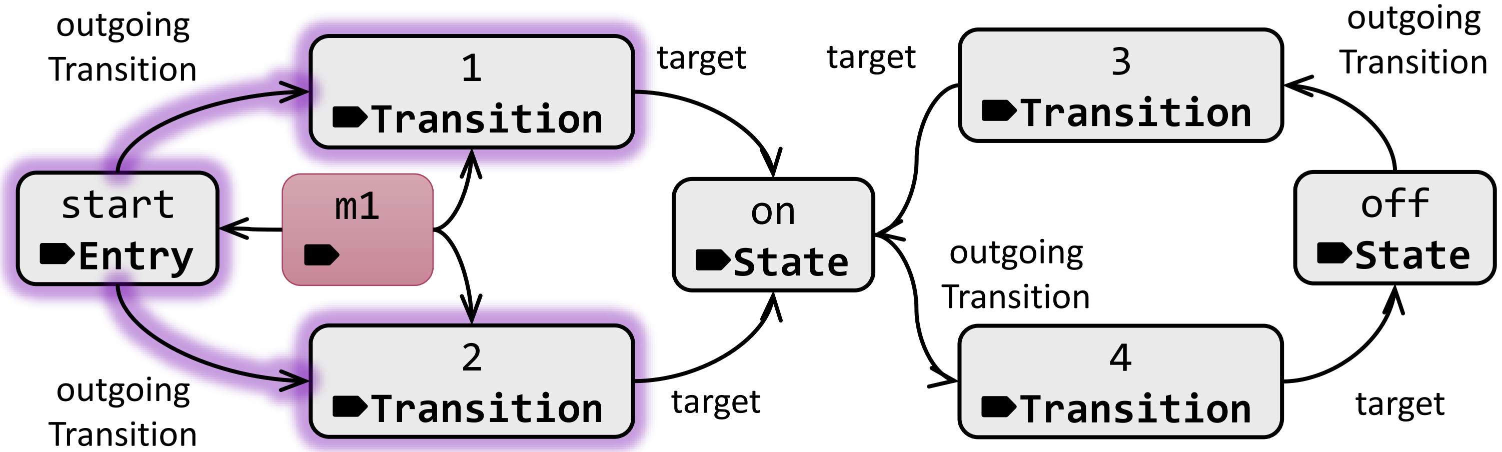
Graph Constraint Evaluation

→ Examples 5-7

```
error multipleTransitionFromEntry(Entry e, Transition t1, Transition t2) ⟷
    outgoingTransition(e, t1),
    outgoingTransition(e, t2),
    t1 ≠ t2.
```



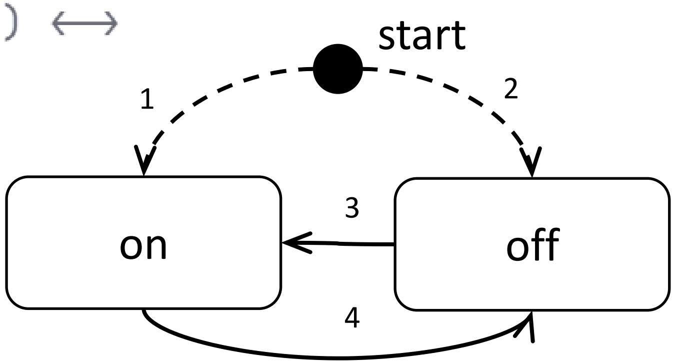


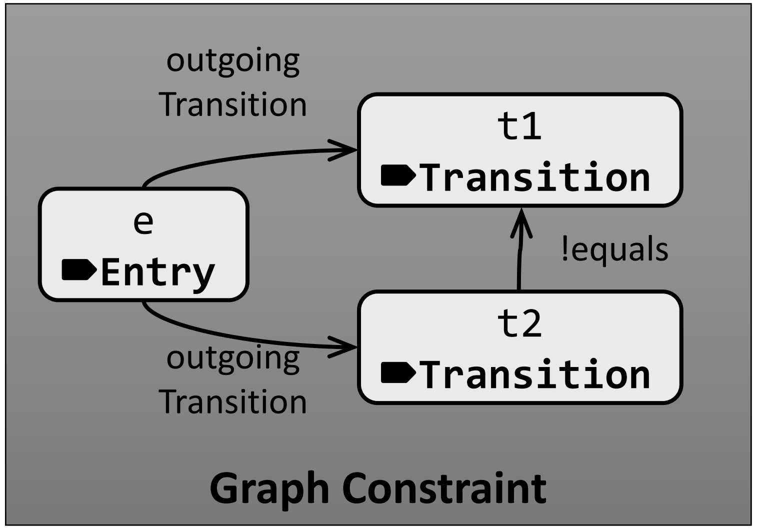


Each match of the query is a certain constraint violation (an error)

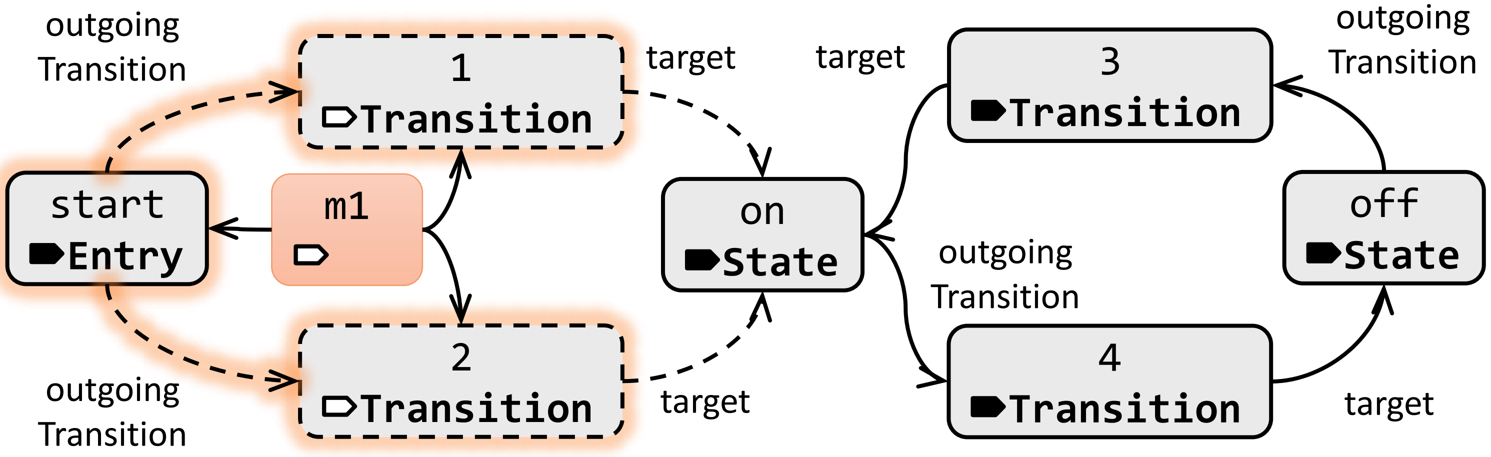
Partial Graph Constraint Evaluation

```
error multipleTransitionFromEntry(Entry e, Transition t1, Transition t2) ⟷
    outgoingTransition(e, t1),
    outgoingTransition(e, t2),
    t1 ≠ t2.
```





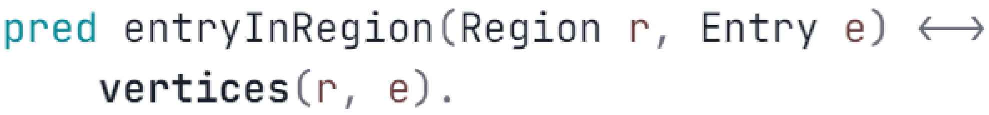
Graph Constraint



A may-match of a query is a *potential* error (which may disappear)

Predicates vs Constraints

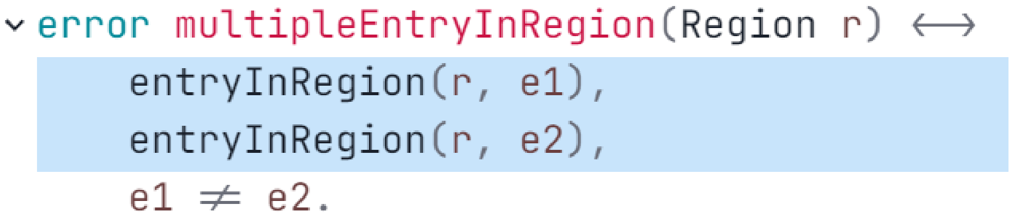
→ Example 8



```
pred entryInRegion(Region r, Entry e) ⟷
    vertices(r, e).
```

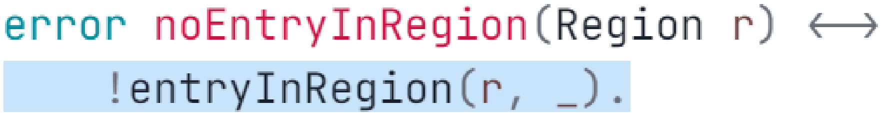
Predicates

- A graph query / predicate
- Composable: Reusable in other predicates or constraints
- Positive condition

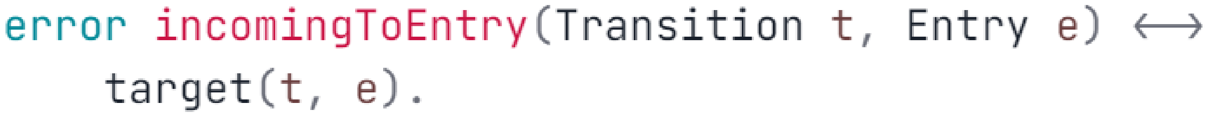


```
error multipleEntryInRegion(Region r) ⟷
    entryInRegion(r, e1),
    entryInRegion(r, e2),
    e1 ≠ e2.
```

- Negative condition



```
error noEntryInRegion(Region r) ⟷
    !entryInRegion(r, _).
```



```
error incomingToEntry(Transition t, Entry e) ⟷
    target(t, e).
```

Constraints (Error patterns)

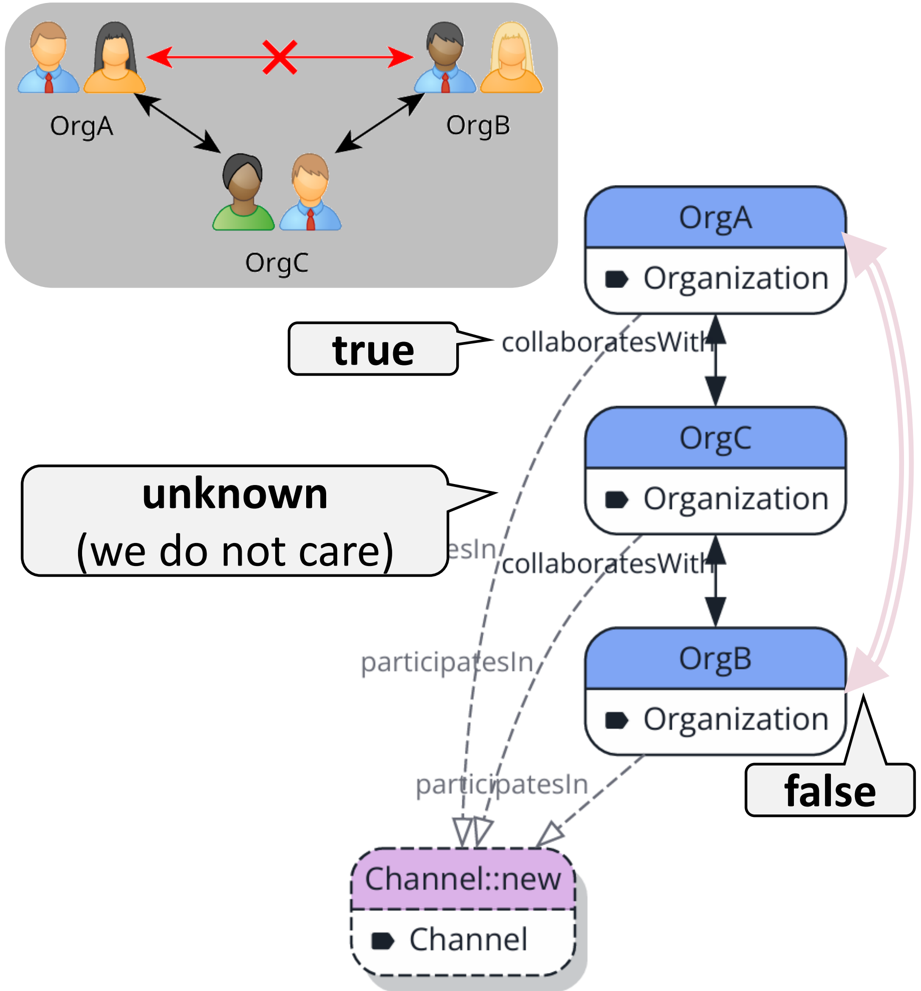
- Capture the violating cases of a domain constraint
- Each match is an error (inconsistency)
- Constraints vs. Types
  - 1-parameter constraint: special *node* type
  - 2-parameter constraint: special *edge* type

Capturing functional requirements as Partial Models

- Requirements = assertions about the architecture

> *"**C** must communicate with **A** and **B**,*
> *but **A** and **B** cannot communicate"*

- Encode **families** of requirements by graph predicates
  - Requirements that should hold everywhere *(characteristics of the DLT platform):* **error** patterns
  - Requirements for specific model elements *(functional requirements raised by stakeholders):* logical **assertions** about predicates

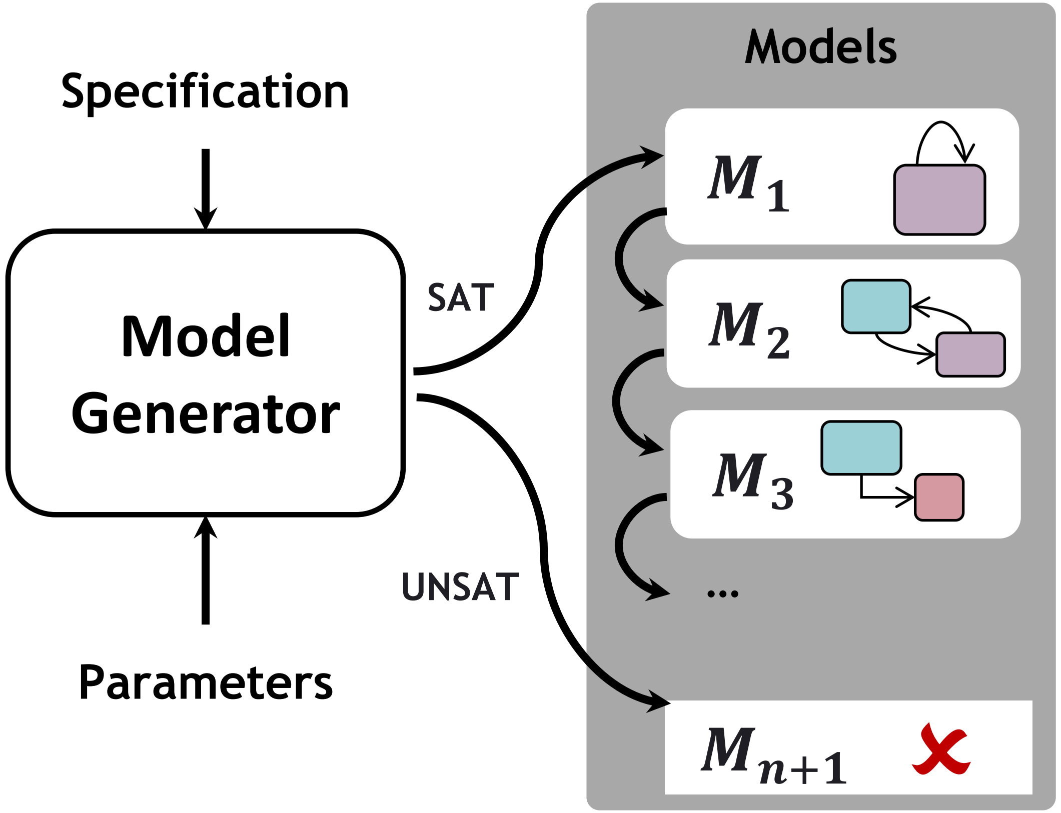


Consistent Graph Generation

→ Example 9

Architecture of a generator:

- **Input:** Problem Specification
  Defines the structure of the models
  Defines the consistency constraints
  Defines an initial model fragment
- **Input:** Search Parameters
  Configures the generation process
- **Output:** Models
  Sequence of consistent models
- **Output:** Inconsistency
  Proving that there are no such consistent model



Search Parameters for Model Generation

- Constraints are continuously reevaluated
- Automatically searching for valid models by applying refinements
- Search is parametrized
  - Number of different solutions
  - Difference between the solutions (non-isomorphic)
  - Random seed
- Scope: *”size of the models”*

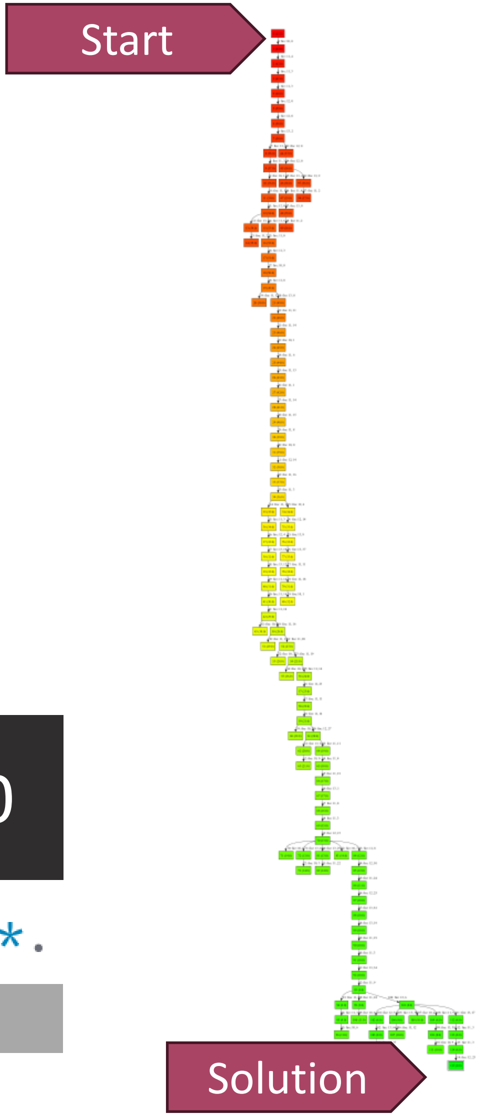
→ Example 10

```
scope node = 15..60, Node = 8..30, FabricNetwork = 1, Channel += 3..*.
```

# of nodes

nodes by type

# of new objects



Propagation rules

Reasoning with partial models

Propagation rules

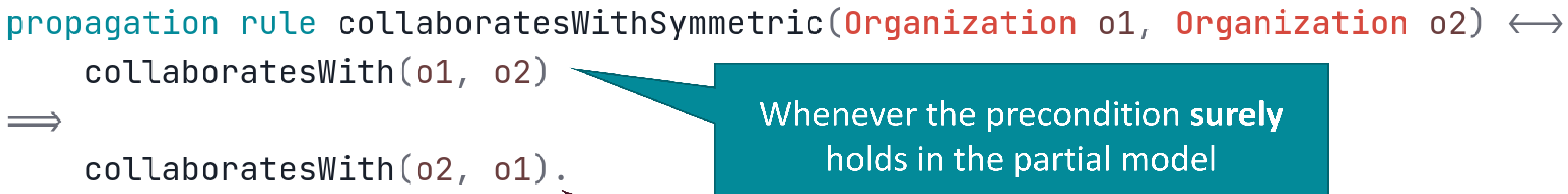
- Deduce **new facts** from the existing knowledge base
- **Preserves** all possible **consistent refinements**
- Fired until **fixed point**
- Firing a single activation **disables** it

Decision rules

- **Branching points** in the search for consistent refinements
- **Reduces uncertainty** by excluding some refinements
- Single decision activation is fired before executing propagation
- Not always self-disabling

Propagation rules

→ Examples 11-13



```
propagation rule collaboratesWithSymmetric(Organization o1, Organization o2) ⟷
    collaboratesWith(o1, o2)
⟹
    collaboratesWith(o2, o1).
```

Whenever the precondition **surely** holds in the partial model

And the postcondition is **not yet** part of the partial model

Add the postcondition to the partial model

- Only valid if the precondition is **logically implied** by the postcondition
- Add implications not automatically discovered by Refinery as propagation rules to improve **reasoning power**

Advanced propagation

→ Examples 14-15

- **Shadow predicates:** lightweight graph patterns with “forward-only” reasoning

```
shadow pred endorsesChaincode(EndorsingNode n, Channel c, ChaincodeInstance i) ⟷
    chaincodes(c, i), endorses(n, i).

shadow pred endorsesMultipleChaincodes(EndorsingNode n, Channel c) ⟷
    endorsesChaincode(n, c, i1), endorsesChaincode(n, c, i2), i1 ≠ i2.
```

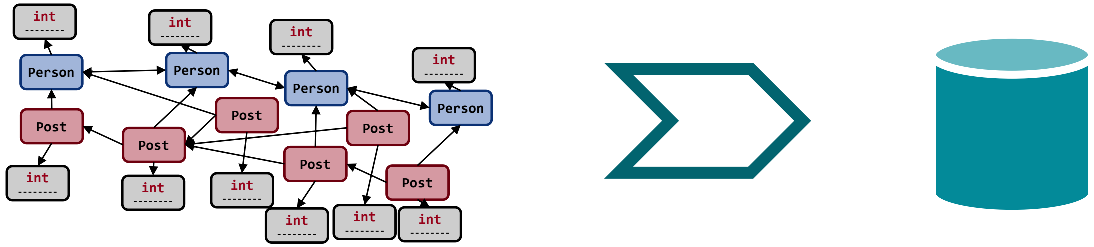
- **Modal operators** in preconditions and shadow predicates

```
propagation rule mustEndorse(EndorsingNode n, ChaincodeInstance i) ⟷
    peer(n, c), !endorsesMultipleChaincodes(n, c), chaincodes(c, i),
    may endorses(n, i)
⟹
    endorses(n, i).
```

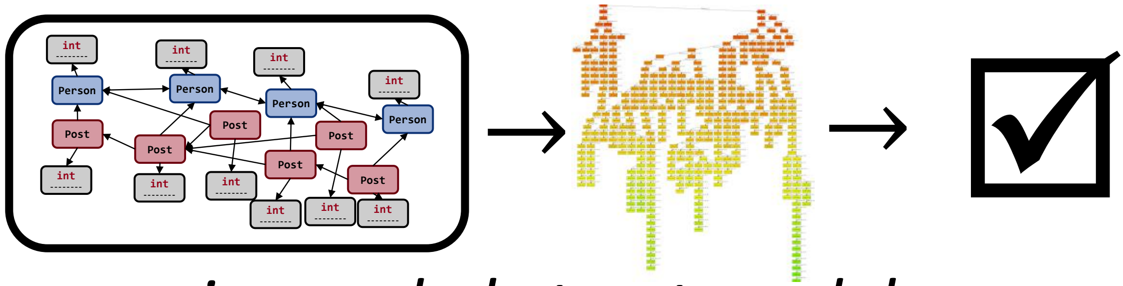
Matches if `endorses(n, i)` is either **true** or **unknown**

The Refinery framework

- **Management:** *store/compare multiple versions of (abstract) models*



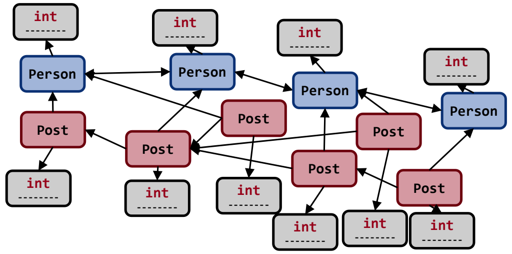
- **Exploration:** *transformation and optimization of models*



- **Reasoning:** *logic reasoning and abstract model concretization*

invalidTime(*p1*,*p2*)↔
 target(*p2*,*p1*)∧target(*p1*,*t1*)
 ∧ created(*p2*,*t2*)∧ (*t2*≤*t1*)

→

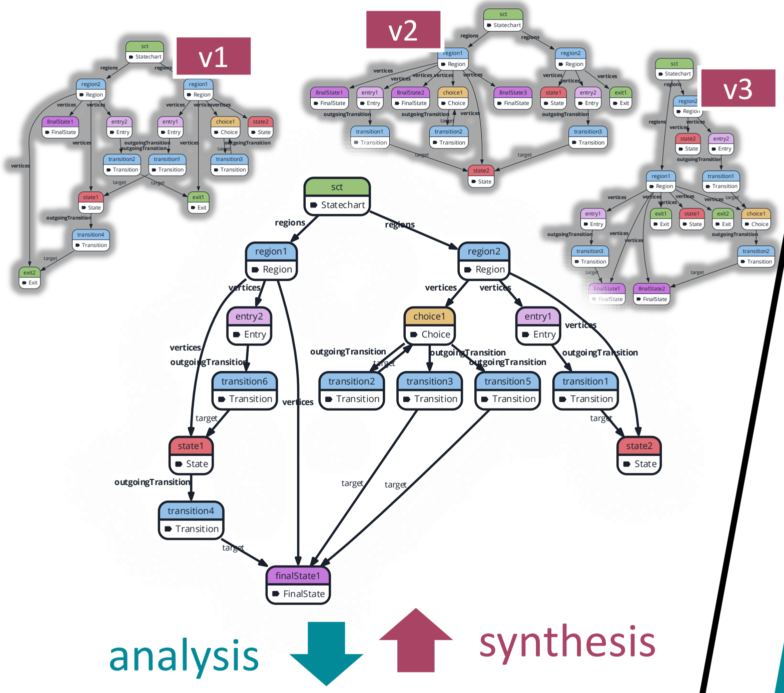


Refinery elsewhere

Applications & appearances

Graph analysis and synthesis

- Powerful mathematical **analysis techniques** for models
- Novel graph-based logic solver for the **automated synthesis** of design alternatives
- **Precision + Scalability**
- **Goal:** solve problems with complex structure



☒ ☑ validation

$$$ cost

◢◢ performance

%% coverage

Recent Results

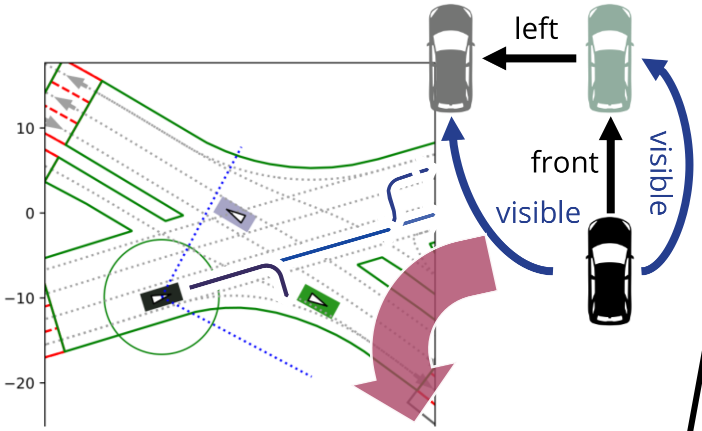
- Research project

VERIFIABLE AI/ML TECHNIQUES FOR PNT APPLICATIONS

esa

Verification/Testing of AI/ML Applications

- AI applications are **data-oriented** systems
- **Complex, dynamic** environment
- Novel **generation** + Advanced **simulators**
  → Diverse **tests**
- Systematic testing of **AI applications**



Recent Results

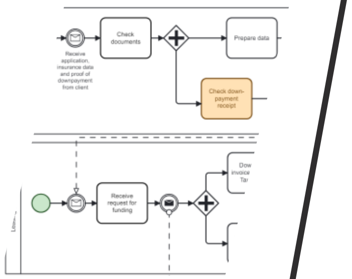
- R&D project with Knorr-Bremse
- Research project with USA Navy

Office of Naval Research Global

Advancing DLT applications

- **Hungarian Blockchain Coalition**
  - Prof. Pataricza – member of the board
  - I. Kocsis: Education WG lead, L. Gönczy: FinTech WG
- **Supporting the EMAP project (PM/NAV)**
  - "Even-based Data-sharing Platform" pilot
  - Employer data provisions: event-based, single-channel
  - Blockchain-based implementation in preparation
- **CBDC research cooperation with MNB**
  - Mapping out: blockchain ↔ Central Bank Digital Currency
  - Payment, car leasing, energy support, industrial cooperation
  - Currently: "ecosystem" research
- **EDGE-Skills: data veracity in EU data spaces**
  - Blockchain-backed Verifiable Credentials

BLOCKCHAIN Koalíció


Receive application, insurance data and proof of downpayment from client
Check documents
Prepare data
Check down-payment receipt
Receive request for funding
Dow invoice Ta

Recent results

Energy price support CBDC prototype: BIS Rosalind finalist

Fabric ↔ Ethereum CBDC bridge in Hyperledger Cacti

Smart gas meters and readings – in production

Refinery@MODELS2024

- Friday 14:30 (FAME) – *Refinery hands-on session*
- Sunday 16:00 (Super Mario Bros)
  T9: Refinery: Logic-based partial modeling
- Wednesday 15:24 (HS7 – Applications 1)
  Ulf Kargén, Dániel Varró. Towards Automated Test Scenario Generation for Assuring COLREGs Compliance of Autonomous Surface Vehicles
  - Find inconsistencies in maritime traffic rules with partial modeling
- Thursday 15:45 (HS1 – MDE&AI)
  José Antonio Hernández López, Máté Földiák, Dániel Varró. Text2VQL: Teaching a Model Query Language to Open-Source Language Models with ChatGPT
  - Generate graph models to verify graph queries generated by Large Language Models
- Thursday 15:45 (HS7 – Applications 2)
  Noor Al-Gburi, András Földvári, Kristóf Marussy, Oszkár Semeráth, Imre Kocsis. Requirement-Driven Generation of Distributed Ledger Architectures
  - Generate architectures for consortial blockchain systems

Further Information

Specification language

- K. Marussy, O. Semeráth, A. Babikian, D. Varró: A Specification Language for Consistent Model Generation based on Partial Models. J. Object Technol. 19(3): 3:1-22 (2020)

Consistent graph generation techniques

- O. Semeráth, A. Nagy, D. Varró: A graph solver for the automated generation of consistent domain-specific models. ICSE 2018: 969-980
- K. Marussy, O. Semeráth, D. Varró: Automated Generation of Consistent Graph Models With Multiplicity Reasoning. IEEE Trans. Software Eng. 48(5): 1610-1629 (2022)
- A.. Babikian, O. Semeráth, A. Li, K. Marussy, D. Varró: Automated generation of consistent models using qualitative abstractions and exploration strategies. Softw. Syst. Model. 21(5): 1763-1787 (2022)

Diverse and realistic graph generation

- O. Semeráth, R. Farkas, G. Bergmann, D. Varró: Diversity of graph models and graph generators in mutation testing. Int. J. Softw. Tools Technol. Transf. 22(1): 57-78 (2020)
- O. Semeráth, A. Babikian, B. Chen, C. Li, K. Marussy, G. Szárnyas, D. Varró: Automated generation of consistent, diverse and structurally realistic graph models. Softw. Syst. Model. 20(5): 1713-1734 (2021)

Correctness proofs

- D. Varró, O. Semeráth, G. Szárnyas, Á. Horváth: Towards the Automated Generation of Consistent, Diverse, Scalable and Realistic Graph Models. Graph Transformation, Specifications, and Nets 2018: 285-312

Summary

- **Logic reasoning** and **model generation** over graphs
- **Web-based editor:**
  - **Live editing** and **feedback**
  - Support for partial models and graph constraints
- **Containerized execution:**
  - Continuously deployed at https://refinery.services
  - Available as **Docker image**: https://refinery.tools/learn/docker/
- **Open-source project:** https://refinery.tools

MŰEGYETEM 1782
ftsrg Critical Systems Research Group

McGill UNIVERSITY

LiU LINKÖPING UNIVERSITY
WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

amazon research awards

Office of Naval Research ONR Science & Technology USN