Applied Category Theory for Scientists: compositional knowledge representation

Kristopher Brown, Evan Patterson, Andrew Baas, Tyler Hanks, James Fairbanks

Category theory is a field of math that studies *structure*. When we model real-world domains using categories, basic constructions of CT very often correspond to useful operations with elegant implementations.

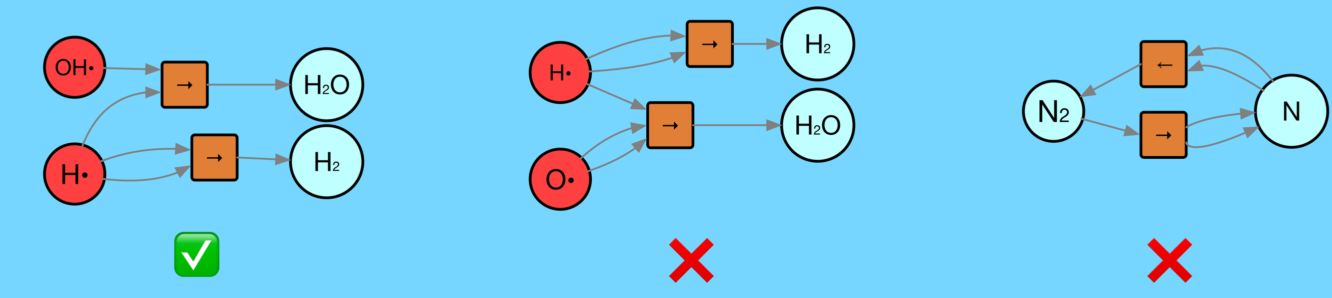
Knowledge syntax/medium	Reasoning engine
Mathematical equations	Computer algebra
Vectors/embeddings	Deep learning
Formal logic	Auto. theorem proving
Scientific papers	Human intuition
Scripts/software ("the model *is* the code")	Running the code

❌ Inference not automatable, doesn't scale well, or error-prone.

✅ CT facilitates separating a simple graphical syntax (easily reasoned about) + giving it semantics in a richer mathematical framework.

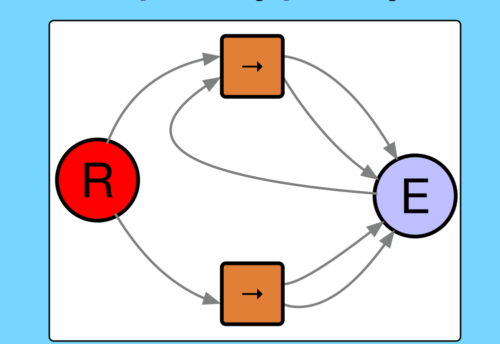
Example of inference: constrained Rxn networks

We can represent chemical reaction networks via an alternative graphical syntax called *Petri nets*. This allows for declaring and enforcing certain high-level constraints using very basic tools, which would not be straightforward with traditional syntax.



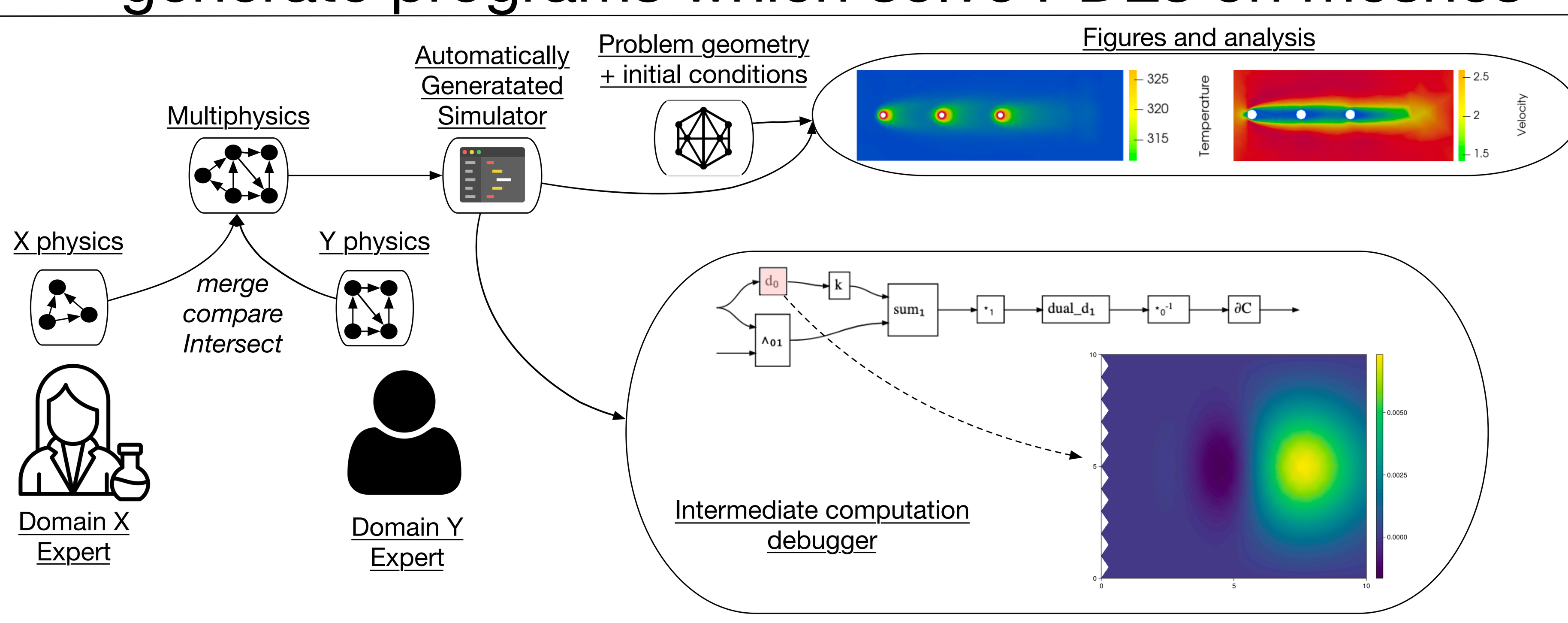
A very general algorithm (*homomorphism search*) can be implemented just once and applied to this kind of graph to check/enforce these constraints. It also has many other applications.

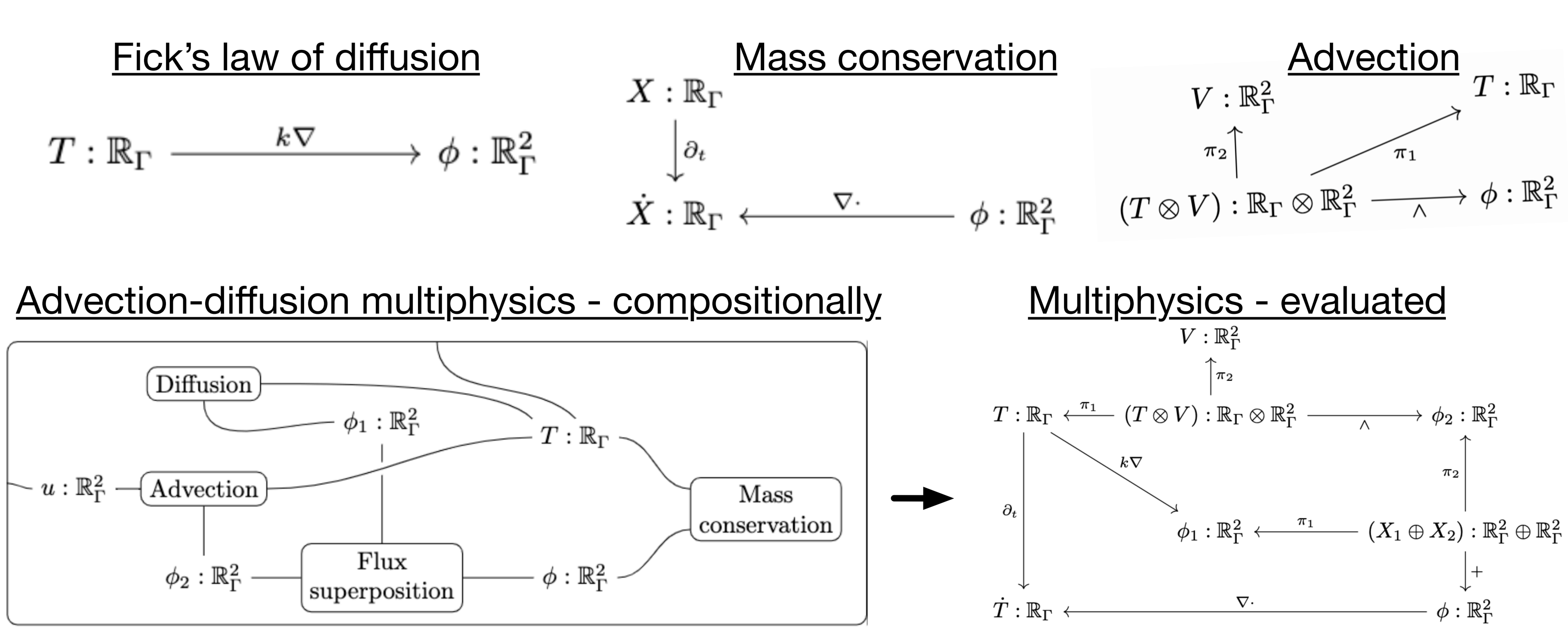
Example Type system



"Two types of reactions, bimolecular or unimolecular. Each reaction requires a 'reactive' species"

Multiphysics solvers specified via diagrams, generate programs which solve PDEs on meshes





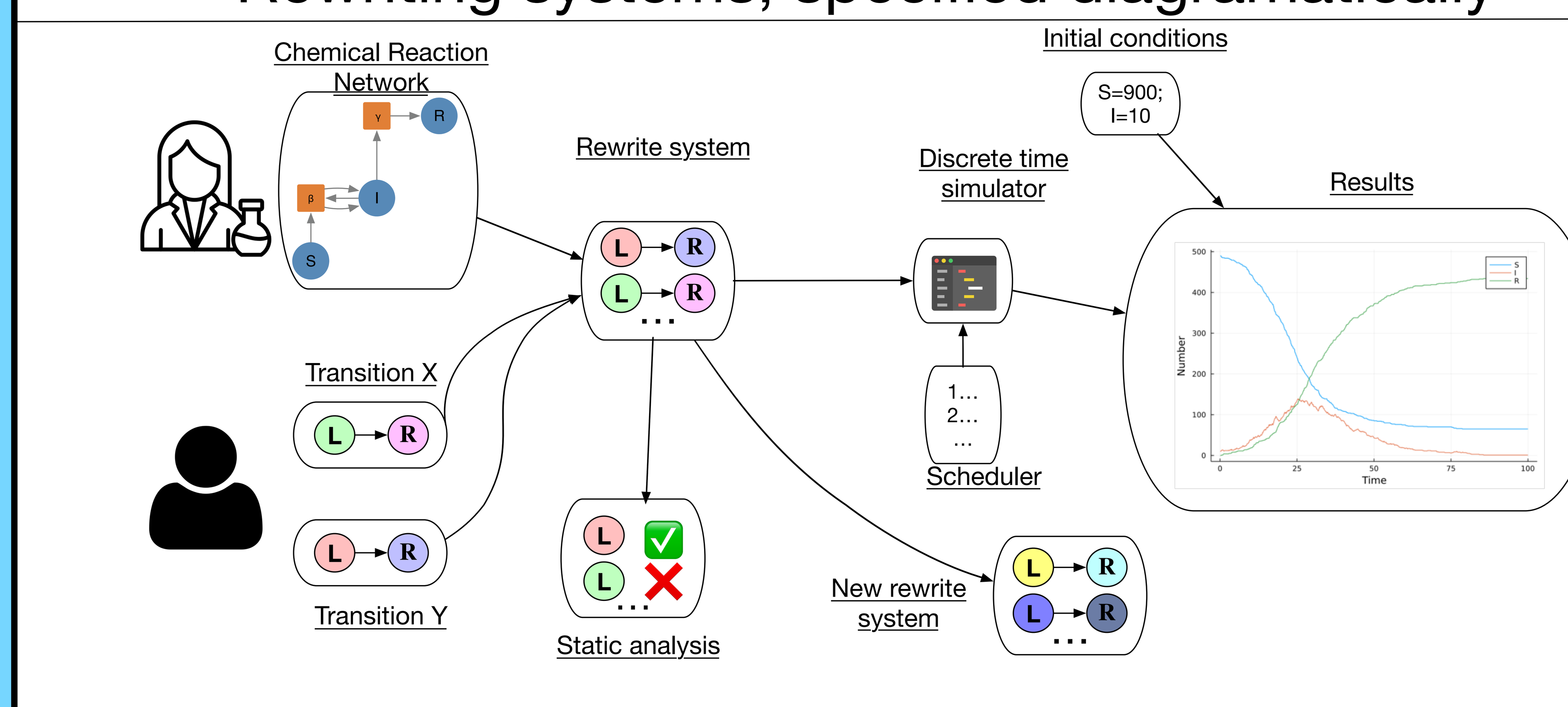
✓Avoid introducing errors when redoing one's mathematical model, in code

✓Governing equations and boundary conditions can be depicted diagrammatically

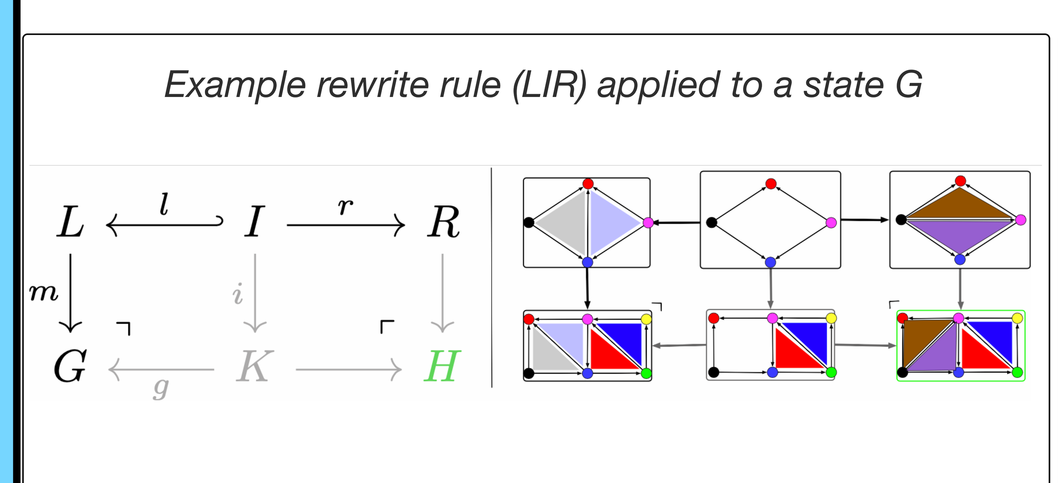
✓This allows sophisticated manipulation / construction of multiphysics

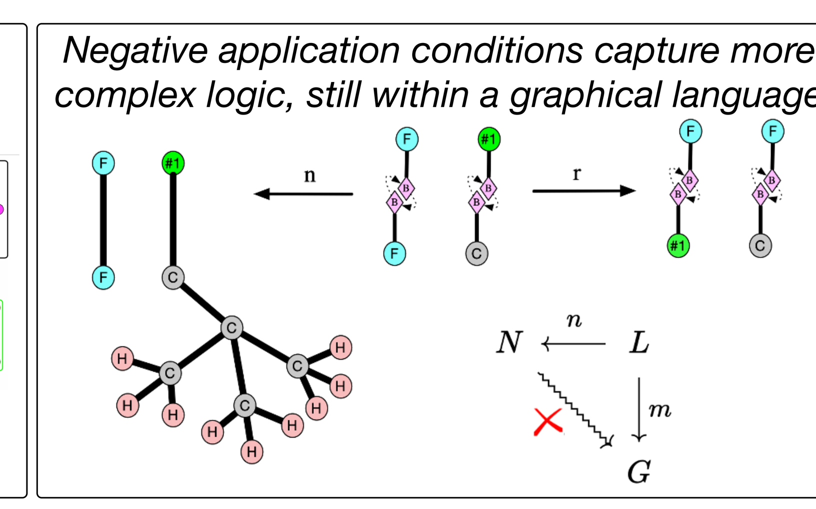
✓We can check when one diagram has the same solutions as another, symbolically

Rewriting systems, specified diagramatically



Solvable tasks: • Perform simulation • Generate terms via a grammar • Equational reasoning





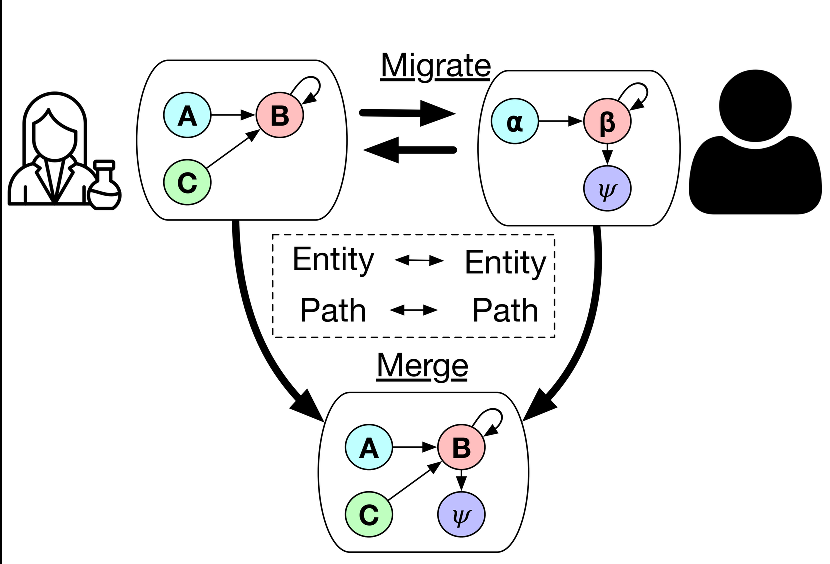
✓Declarative language for a very broad type of program. No code needs to be written!

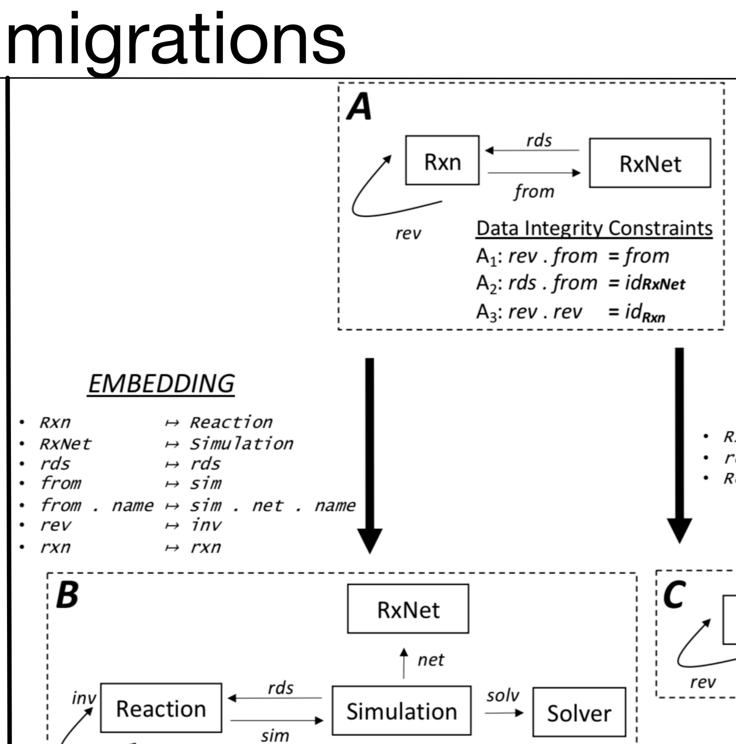
✓Rewrite systems easier to debug than OOP code - rules are just graphs that can be analyzed independently.

✓CT rewriting naturally gives us provenance and one implementation works over a broad class of data structures (data structure is an input *parameter*).

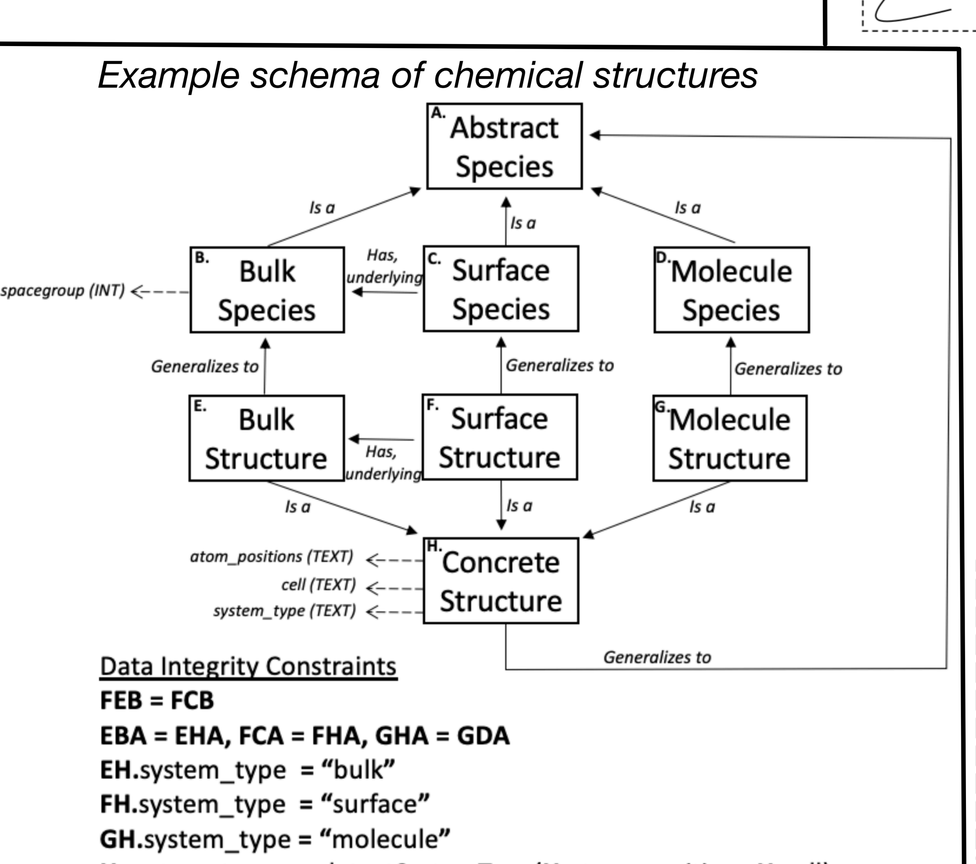
✓We can perform migrations on our rules when our assumptions change.

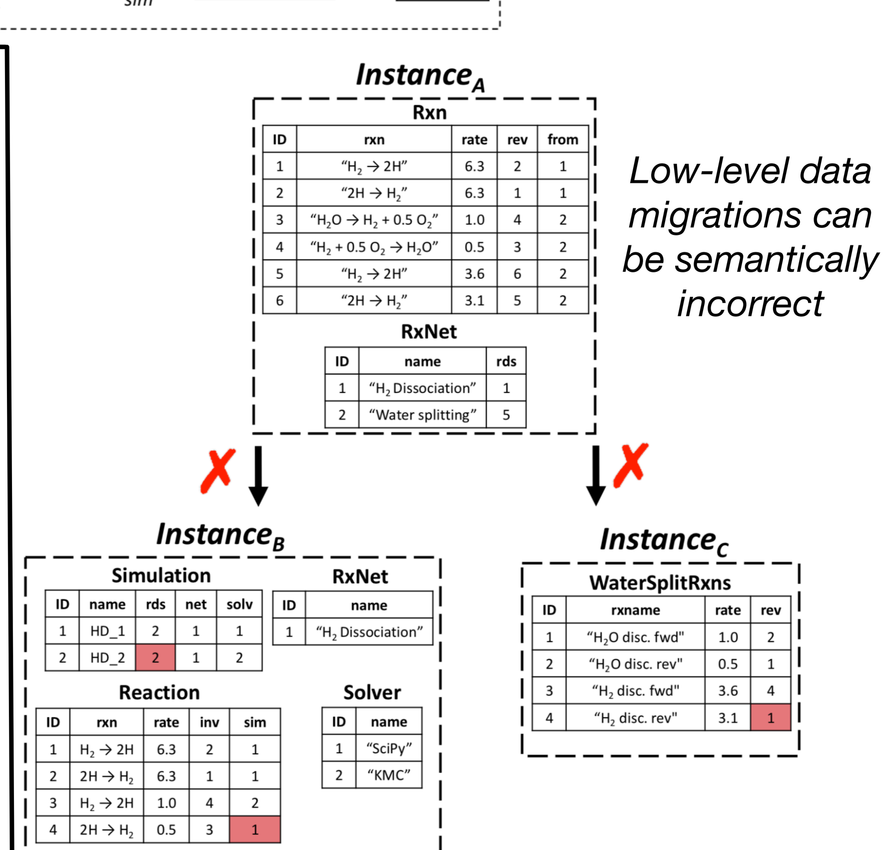
Declarative language for meaning-preserving data migrations





Maps between schemas which induce data migration programs.





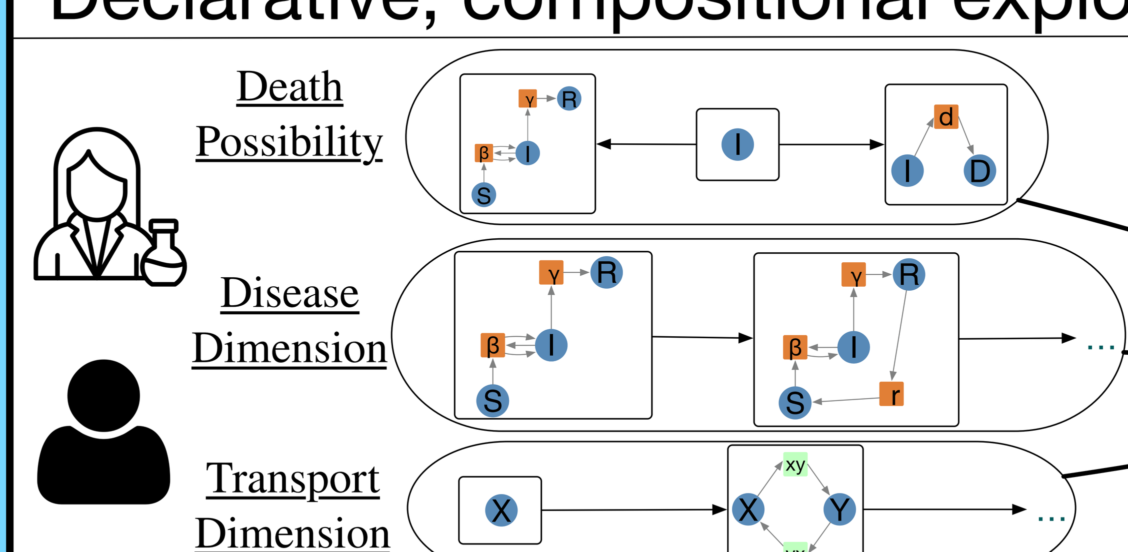
Low-level data migrations can be semantically incorrect

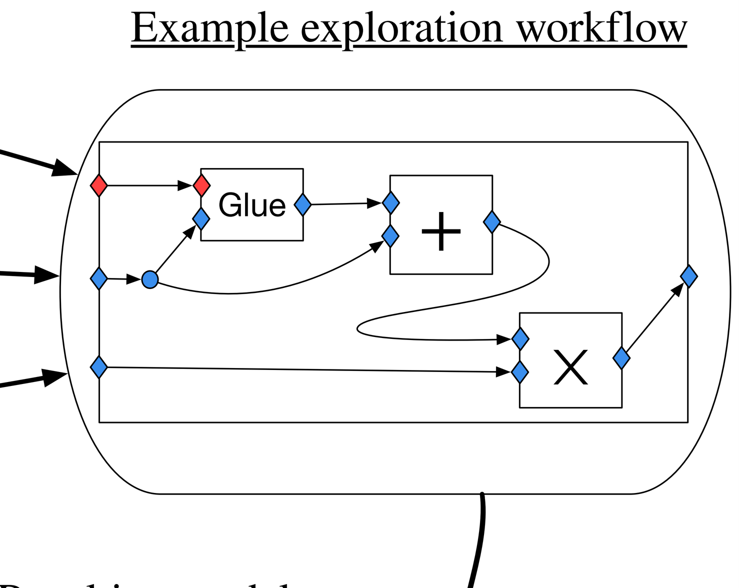
✓Mathematically-rigorous data migration induced by high-level schema maps.

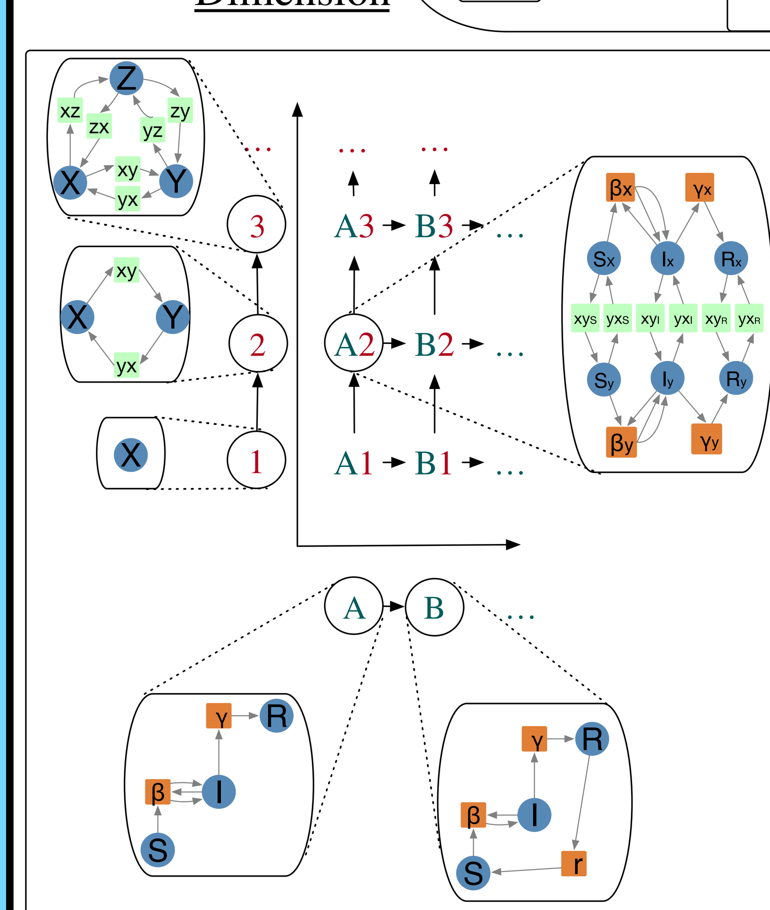
✓Data sharers can equip data with constraints; restrict misuse by data receivers.

✓Constraints from data receivers filter data that doesn't meet this specification.

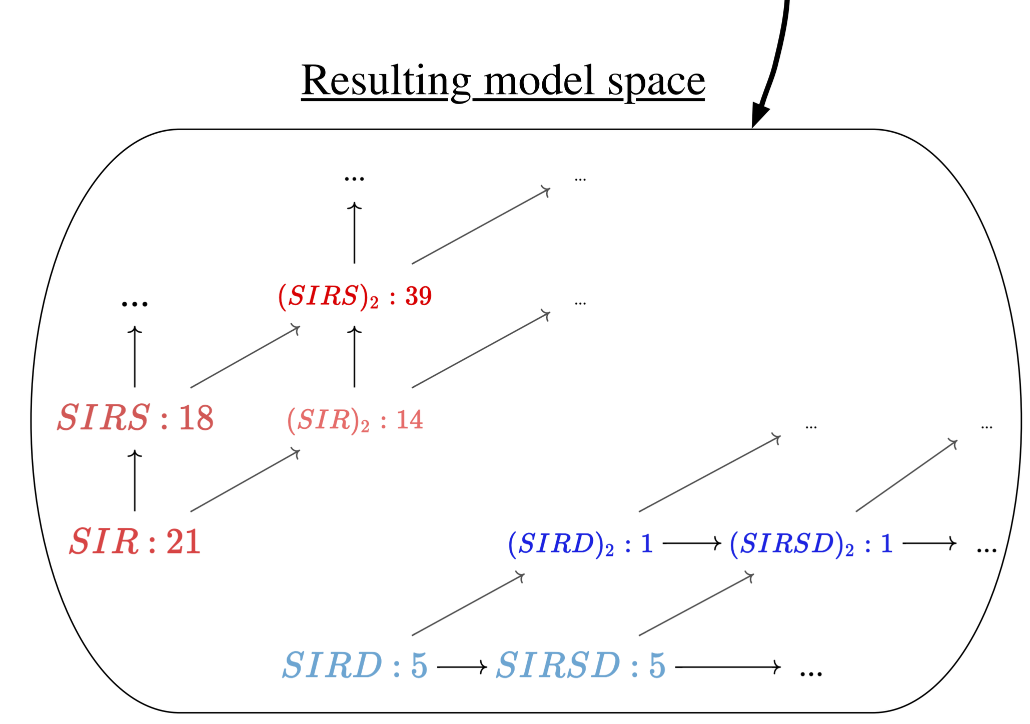
Declarative, compositional exploration of model space

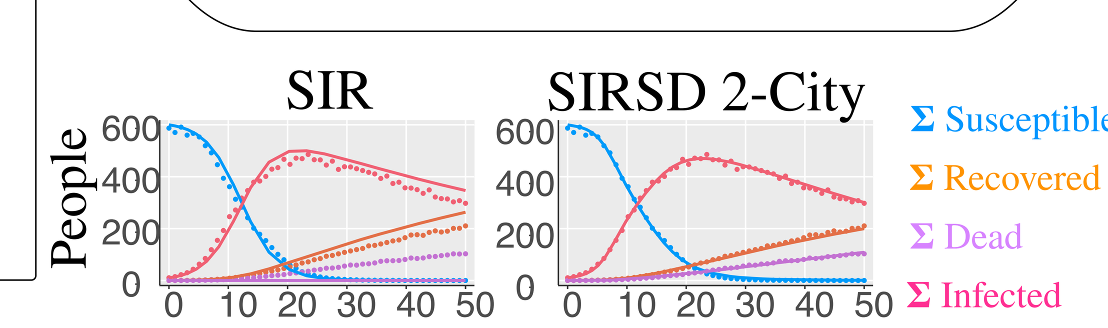






Example product of "disease" and "transportation" model spaces





✓Declarative language for specifying and executing the exploration of model spaces

✓Provenance of model components aids analysis and comparison of resulting models

✓Flexible, generic tools for combining smaller model spaces into larger ones