Breakout Session Meeting

16 May 2019 / 10:00 AM / CONFERENCE ROOM

# Attendees

GTRI: James Fairbanks, Christine Herlihy, Erica Briscoe

Pitt: Paul Cohen, Tomek

GE: Alfredo Gabaldon

Arizona: Souratosh Khan

# Agenda

## Topics

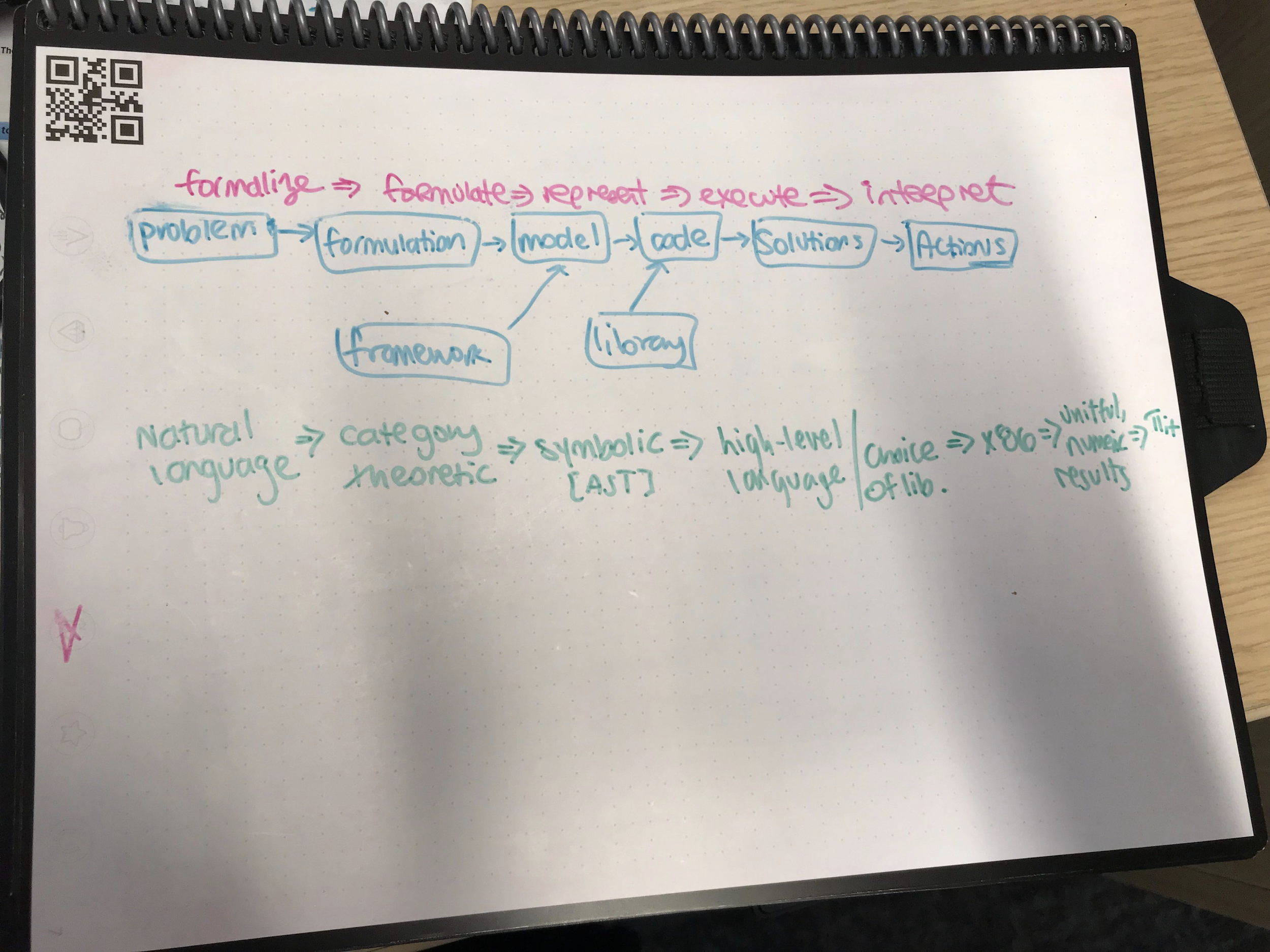
1. **MM1: Meta Modeling:** Defining sets and types of scientific meta-modeling procedures for ASKE (top half). What are the key meta-modeling procedures scientists use/need at the structure and abstract knowledge levels (and crucially processes for translating up and down)?
2. How do we collect, characterize, catalog, and formalize these procedures?
3. Are they obvious or do we need to find a social scientist who studies scientists?
   1. Yes we need to watch scientists do the first step problem -> model
   2. Survey-based /mixed-methods + structured data extraction about how scientists engage in modeling process (e.g., diffs of model cycles)
4. Several groups are thinking about how to use abstract representations of knowledge to explore, discover, create, etc. models in various ways and for various domains. And at least one is looking at how we can learn new abstractions and grammars from model representations. What are the generalizable principles?
   1. Problem -> formulation -> model -> code
   2. Framework ^
   3. Classes of domain problems have attributes that can be expressed as presence/absence eg:
      1. All problems that we try to model have some notion of quantity/amount
      2. Relations between things (are estimated)
      3. Decisions and processes
         1. Control flow; also in some cases optimization
         2. Difference between markov chain and sequential decision process == action
         3. Agent/agency (evaluation/information that affects the decision)
      4. Processes: sequence of states/decisions
         1. Is the model deterministic or non-deterministic?
         2. SVM: separate the “hyperplane” as a model from the algorithm that implements
         3. Separate the “model” from the algorithm that estimates/approximates

# Notes

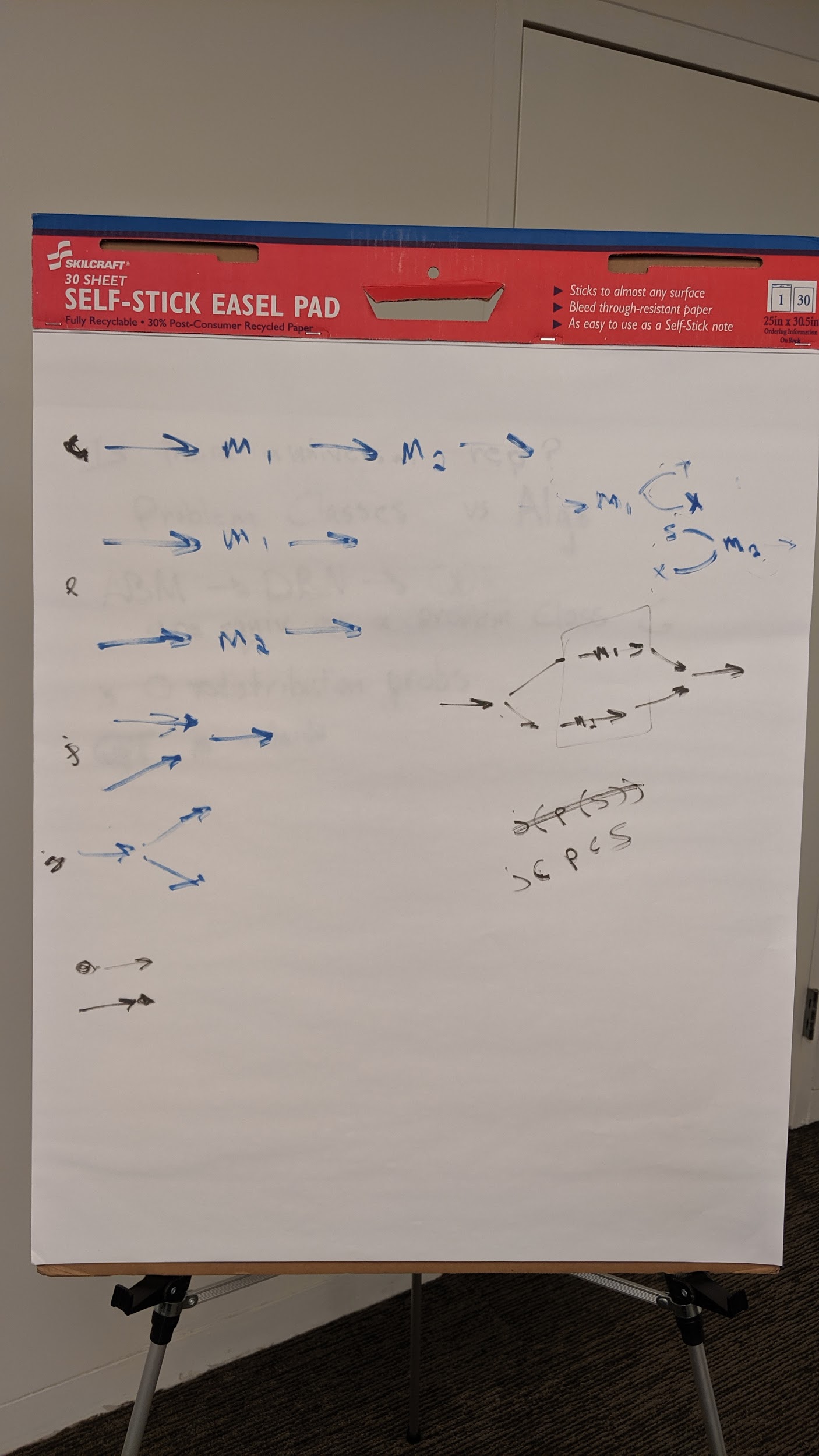
* Is the use case of constructing new models from existing models (with KG as base) an example of something we can formalize with category theory?
* Model synthesis: in parallel or in series; combinations with these operations as primitives can yield “wiring diagram” of model synthesis
* Class of primitive models and a framework for composing them in different ways (split; join; parallel; compose) can build any model by composing the primitives in some way
* Do we need the notion that the composition of two models can be done in parallel?
  + Can be an indicator of independence
  + If you have variables you don’t need parallel b/c you have assignment statements; for closed form though, parallel needed
* If we treat models as expressions and the composition of models as expressions, the framework can work at the base and meta level; framework is internally consistent
* These primitives can represent tupling/un-tupling; can also contain mathematical operations on the output(s) of model(s)
  + Can push data through or start out with a result and try to go back through and see what operations are needed
  + Diagrams are an indication of sparsity b/c of relative frequency of 0s in the edges
* Abstract grammar: the set of all possible models I can make by using primitives and transformations to compose models
* Convergence toward an IR (Galois: syntax tree; how things relate to each other; formulas that include arithmetic formulas and boolean predicates)
* Given rules, composition: how to incorporate stochasticity?
  + Need a random number generator; entropy source, etc. can only be pseudo-random
  + Can use “coin flips” to “flip bits” in model circuits
  + How to incorporate/recover sampling?
  + Adding a “randomness” primitive changes the types of models we can make given a class of input models
* Need to be able to encode domain validity checks (e.g., re: sequential operations that cannot occur given physics, etc.)
  + Type checking can get you so far, but semantic validity when chaining ops?
* What does Turing completeness get us; it may not be the correct intermediary layer (e.g., should probably be domain specific rather than universal)
* PRAM project: any model can be represented as a PRAM -> is this the most compact form of model?
  + Bayes Net -> subdivide domain into variables, some of which are conditionally independent; graphical model more natural for humans; rules -> conditional logic
  + Equivalence relationship between graphical models and control flow logic; should be possible to specify as a PRAM and run as a bayes net b/c more understandable/explainable and then move back to PRAM and then maybe move to ODE. Universal over ODEs, ABMs, DBMs.
  + Could we for example represent SVMs in this same framework?
  + We have intuition that DBM, ABM, ODEs are syntactically different but roughly semantically equivalent
    - Analogy b/w problem classes and algorithms that we see in the theory of computation
      * Reduction re: complexity classes and associated costs (space, time)
      * For a particular problem class, AMBs, DBMs, ODEs look semantically equivalent; the question is: what is that problem class?
      * Redistribution problems (per PRAM team): for this problem class, these are roughly equiv.
      * Game theoretic/decision-making models?
        + Seeking to reason about/make inferences about equilibrium
        + Axel work is about the emergence of strategies
        + Which strategies (and time horizons) yield (un)stable equilibria?
      * Should we think about models or should we be thinking about problem classes (with computational complexity classes as the implicit example of “classes”)
    - Can category theory provide a framework to help us understand what functors may exist between these modeling frameworks for a given class of problems?
      * Functors must preserve structure of the categories they bridge
    - PRAM can solve any problem that involves redistributing mass
      * Mass can be introduced; is allowed to change.
    - ABMs w/ vital dynamics?
      * 2 groups, one with infinite size that produces new mass but doesn’t count toward mass of simulation and same for dead people “reservoirs outside for dead”
    - If we can frame a problem as a PRAM?
      * FRED collaborators !like the PRAM formulation; want people as agents to be unique
      * How to turn ABM into a PRAM? -> those people are not unique; they form groups and when you have groups you have mass
      * Assumption of compartmental models: uniform mixing within compartments; ABMs a reaction to this.
      * Within a compartment there’s uniformity.
      * Group can split into other groups.
      * Degenerate case => an ABM (e.g, where everyone is unique and groups are all of size 1).
      * Identity of the agent and state of the agent at a particular point in time /in the simulation are not the same thing; e.g., in ABM agents are children and states evolve over time
      * Every group has a feature vector; hash the feature vector and anything that matches the hash contributes mass to that group
      * Group is composed of all agents whose feature vectors hash to the same hash
      * The infectivity of that group is a function of the group interaction parameter. Can also condition on relations to other objects (which are components of the feature vector).
      * The rules related to the disease tell you which “features” matter for infectivity; in some sense, we can think about this as masking certain bits of the feature vector.
      * The features can be dynamic over time (e.g., opposes vaccination and then gets removed from the school)
* What are the features of the problem classes?
  + Empirically observe models and the objects they deal in
  + Configuration -> discrete buckets or continuous “feature” space of these abstract classes of problems?
    - All problems that we try to model have some notion of quantity/amount
    - Relations between things (are estimated)
    - Decisions and processes
      * Control flow; also in some cases optimization
      * Difference between markov chain and sequential decision process == action
      * Agent/agency (evaluation/information that affects the decision)
      * Processes: sequence of states/decisions
      * Is the model deterministic or non-deterministic?
    - SVM: separate the “hyperplane” as a model from the algorithm that implements
    - Separate the “model” from the algorithm that estimates/approximates
  + “Parts that solve it”
    - SDP ; code that implements it; how to represent the actions and
      * MDPs; POMDPs; Markov chains;
        + Do you have information; will you ever get the information; is there an actor, etc.
        + Forgetful functors to “forget” which elements are only partially observed; can go up in complexity by specifying things to hide
  + Problem classes are defined as equivalence classes of a relation between problems
  + Classes defined should be determined by the class of problem not the computational methods or mathematical methods used to achieve the estimation
  + Reducibility is a helpful framework
* Problems Identified:
  + Equivalence classes of frameworks
  + Transformation between frameworks
  + (which) structures (are) preserved
  + Metamodeling => Model augmentation, synthesis, validation
* To get to formal representation of a real-world phenomenon, you have to make choices about how to represent the elements of the phenomenon
  + No mathematical formulation of a problem … but a grammatical one?
  + Problem; modeling choices yield a formulation; pick a framework
  + Static analysis happens AFTER you’ve made the mathematical formulation
  + Are there subclasses of problems
* Instance of a domain problem -modeling choices yield a formulation (can be many but pick one) ->given the formulation, pick a framework (can be many valid but pick one) -> having picked a framework, pick a model (can be many, pick one) -> pick an implementation/library -yields-> code; here ML comes in re: hyperparameter selection; optimization, etc.
  + DET and NON\_DET redistribution problems
  + Redistribution problems as a class of problems; how to distinguish between the members => those which can be reduced
  + RDP that can be implemented as DBM and those that cannot
  + Seems to suggest the class of redistribution problems is hierarchical
  + We know going in that there exist constrained processes: PRAM can solve anything that can be formulated as a DBN but there are PRAMs that cannot be formulated by DBNs
  + DBN = framework; formulation = mathematical representation of domain problem; symbolic representation of the domain problem (can be a process diagram; if represented by petri net, petri net’s the framework)
  + SIR diagram => formulation (eg, diagram + semantic knowledge that the boxes are states)
  + There’s an implicit connection b/w formulation and framework
  + Can we define classes of problems, and classes of constraints; given a framework and a formulation, you can decide whether there’s a model that can capture the two sets of info?
  + Sample the states of models: we want to be able to constrain it bc otherwise state space is intractable
  + Language of formulation
  + “Can I do this with model of class x?” => “do this” == represent this formulation within the choice of framework?
    - Language of formulation is key to algorithmic decidability (of the “language”)
    - Are there cycles within => to check properties of the problem and see whether a framework can handle these properties?
    - Every formulation has a set of primitives and rules for composition to allow you to make more complex models out of the primitives
  + There are PRAMs that can be implemented as DBNs
    - “If you’re rich you stay home” -> means that you need an infinite number of conditional probabilities
    - During formulation if a loop exists, DBN will be precluded as a framework choice
    - PRAMs as a language ?
    - Once you write down the rules you can tell how you can implement it? It’s expressive enough to check it can be implemented as a DBN
    - How to enumerate the rules?
      * You write the ontology for the domain in expressive language; can prune this and then chose the least expressive parser necessary (eg. minimize complexity to the extent possible)
      * Expressing problem in the formulas; can this formula be represented using a DBN? Can this DBN be implemented in STAN; PRAM.py, etc. (for example)
  + The shift that changes at each point in time is what tells you it can’t be implemented with a DBN; DBN doesn’t let you capture a relationship
    - Can have a random variable stand for number of people sick at a particular school
    - Would have to duplicate dbn structure at every location
    - Alpha represents arbitrary relations; conditional probability tables will change at each time step and thus DBN ! well-suited
    - There does exist a formulation but when you go to the actual framework (DBN), it shouldn’t work (system should be able to tell you this is a bad idea)
    - Cpt with subscripts ! allowable and :. Can’t be modeled using the framework
    - You can build a dbn but with subscripts (it’s about the number which in turn is about the extent of state space expansion implied)
    - You don’t know what the cpt is until runtime because it depends on each previous time step. Pram implements an “illegal” broader notion of cpt. PRAM ~ DBN ++
      * An example of how machines could help people validate their choice of framework given a formulation; you need to change one or the other to continue modeling
    - Want a language in which we can express the problem we’re trying to solve; syntax tells us which frameworks it can be represented in
  + Can compile into delayed differential; eg. can’t calculate values until know state of system
  + The example system implies that time moves in discrete time steps b/c we don’t know what happens in between t and t + 1
  + Model augmentation => can we perturb the example model?
  + Objective: transformations in metamodeling language directly (which correspond to changes in the models which correspond to changes in the code)
  + The formulation is decontextualized; the semantic information comes from the domain description of the problem
    - How can we inject the context into the metamodeling language?
    - The context in some cases does impact your choice of framework (e.g. geospatial; disease vectors)
    - Use cases also matter: if you want to couple one system with another, this also impacts the (possible? optimal?) choices of frameworks?
    - Metamodeling language should be expressive enough to capture context, coupling between models, etc.
    - This type of info can be captured via the type system (floats with units; geographic area which has lat/long or some other spatial information)
      * Values, types, syntax
    - How to express coupling and context
    - PRAM uses relational tables and databases (if variable ! in database, or if there’s ambiguity <- partial matching)
      * Categories -> can be rep. As database schemas
      * Going from schema to model and vice versa
    - Can take DB and turn into equivalent KG

# Action Items

# Readout

?

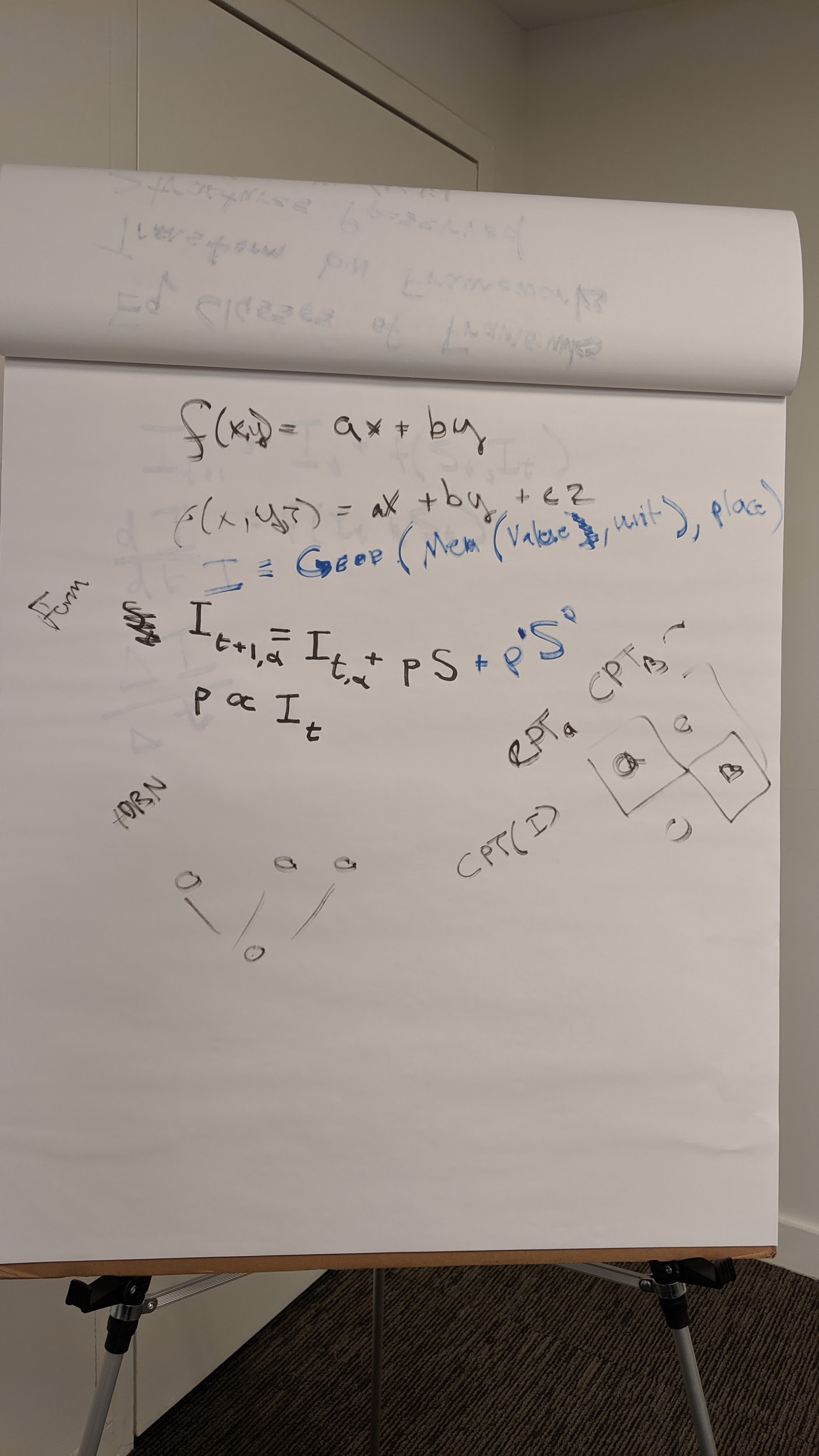
1. Workflow
2. Equivalence Classes of Frameworks / Models
3. Abstract formulations, Domain specific primitives and grammars



1. Transformation between Frameworks
   1. What Structures are preserved?
      1. Objects & Morphisms,
      2. Variables & Expressions,
      3. Agents & Transitions
      4. Compartments & Flux

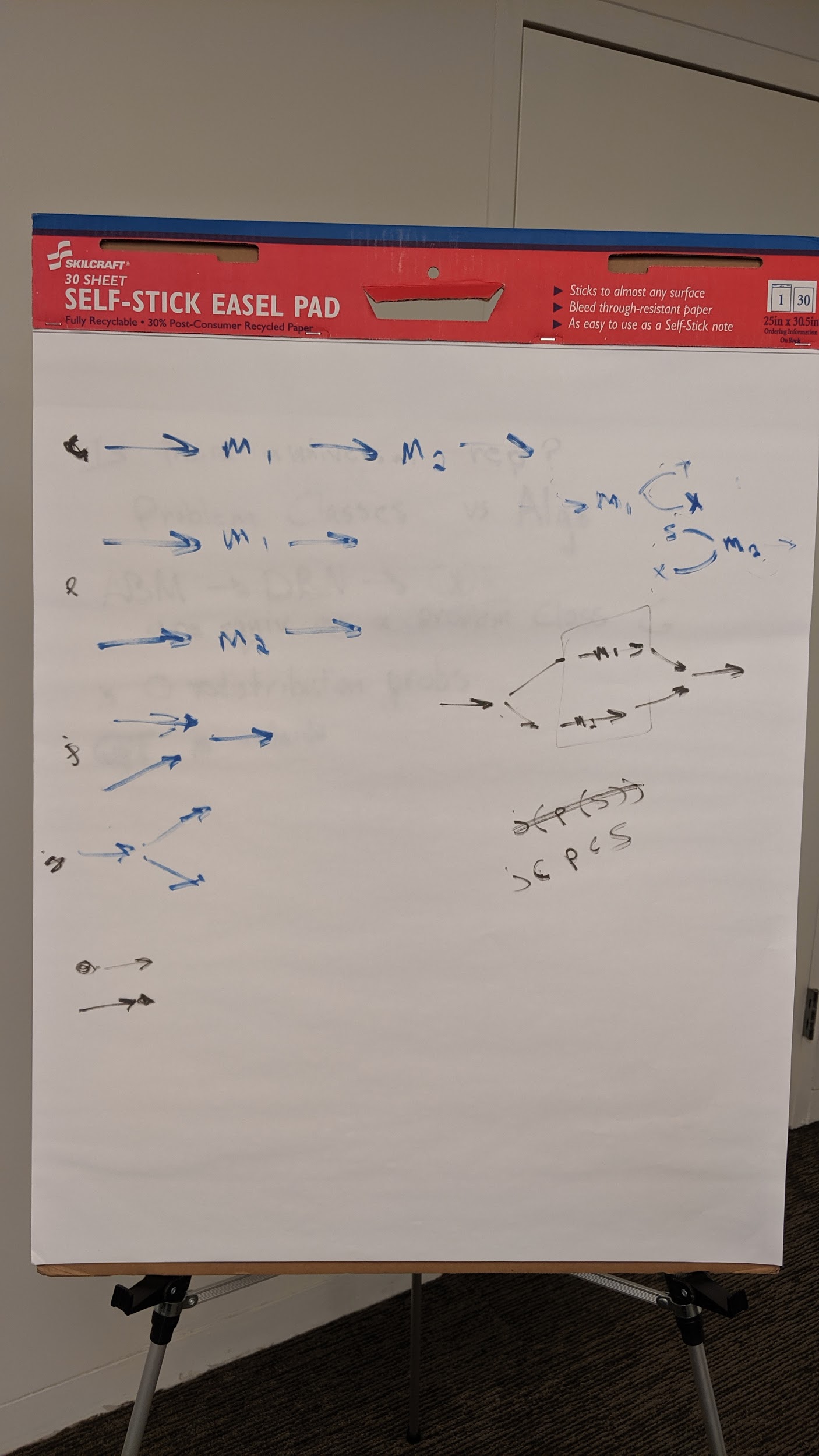


1. Tasks: Augmentation, Synthesis, and Validation



1. Quantitative, Relational, Agency, Observation/Information
2. The formulation step needs Ethnographic Methods
3. **MM1: Meta Modeling:** Defining sets and types of scientific meta-modeling procedures for ASKE (top half). What are the key meta-modeling procedures scientists use/need at the structure and abstract knowledge levels (and crucially processes for translating up and down)?
4. How do we collect, characterize, catalog, and formalize these procedures?
   1. See Diagram
5. Are they obvious or do we need to find a social scientist who studies scientists?
   1. Yes we need to watch scientists do the first step problem -> model
   2. Survey-based /mixed-methods + structured data extraction about how scientists engage in modeling process (e.g., diffs of model cycles) can help us design a system that “learns” when erroneous and/or domain invalid configs are present.
6. Several groups are thinking about how to use abstract representations of knowledge to explore, discover, create, etc. models in various ways and for various domains. And at least one is looking at how we can learn new abstractions and grammars from model representations. What are the generalizable principles?
   1. Problem -> formulation -> model -> code
   2. Framework ^
   3. Classes of domain problems have attributes that can be expressed as presence/absence, e.g.:
      1. All problems that we try to model have some notion of quantity/amount
      2. Relations between things
      3. Decisions and processes
         1. Control flow; also in some cases optimization
         2. Difference between markov chain and sequential decision process == action
         3. Agent/agency (evaluation/information that affects the decision)
      4. Processes: sequence of states/decisions
         1. Is the model deterministic or non-deterministic?
         2. SVM: separate the “hyperplane” as a model from the algorithm that implements
         3. Separate the “model” from the algorithm that estimates/approximates
7. Transformations between modeling frameworks / IRs are they invertible? Just because you can represent a convert between them, doesn’t mean you get the output a domain expert would expect.
   1. Emergent effects

# Visual Notes

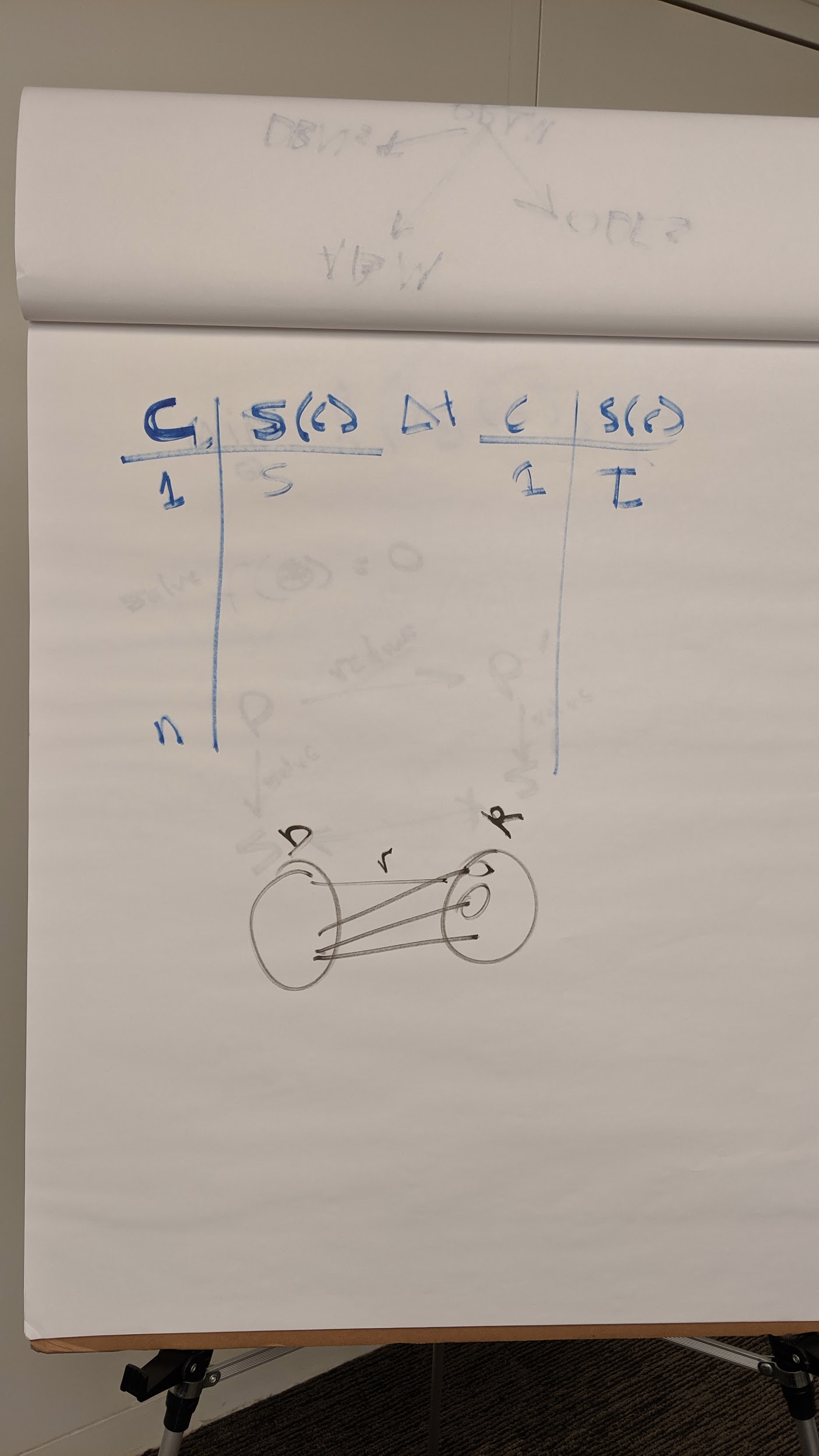
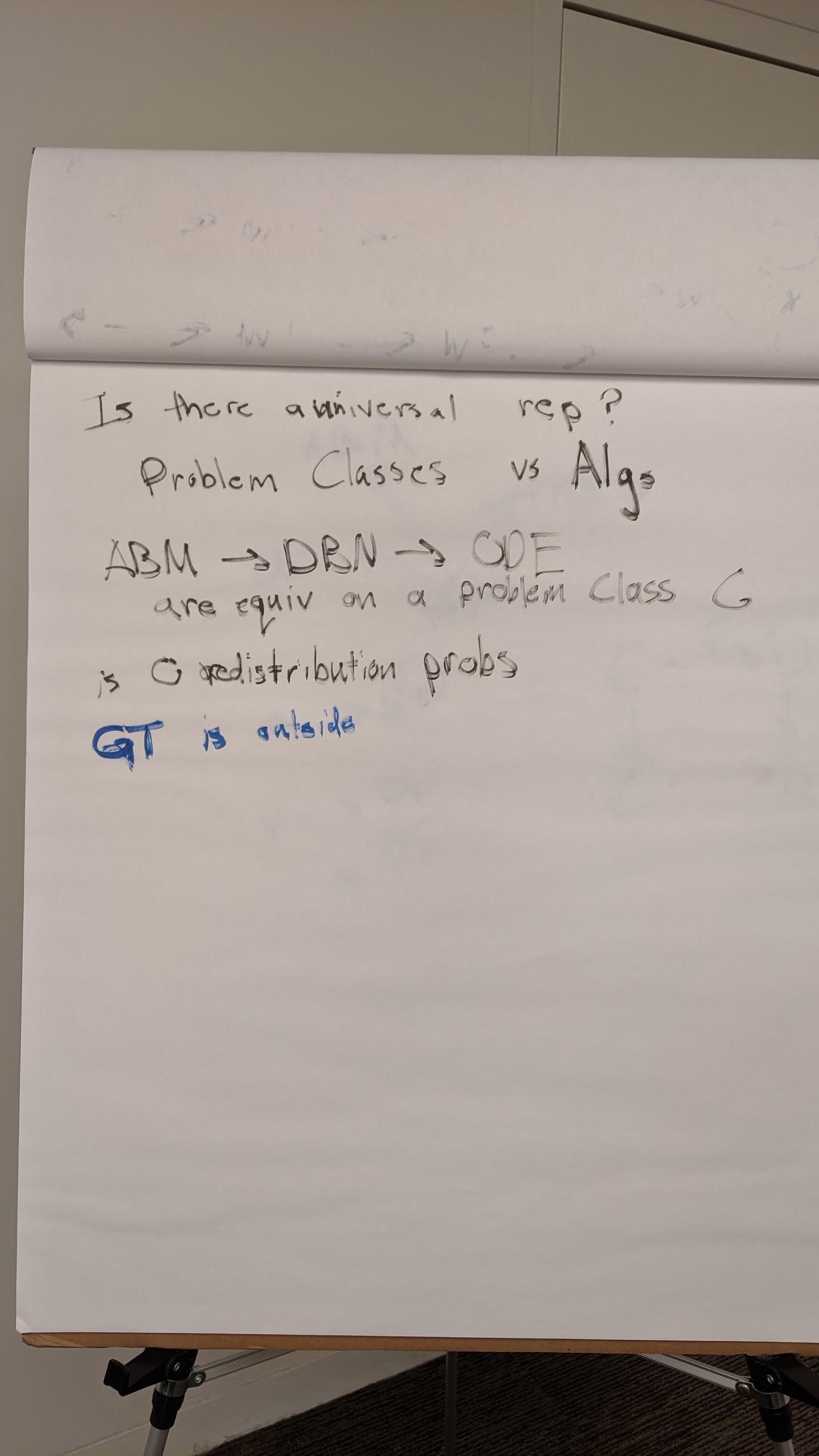


One example of how the Formulation block could be implemented

A set of primitive operations of model composition

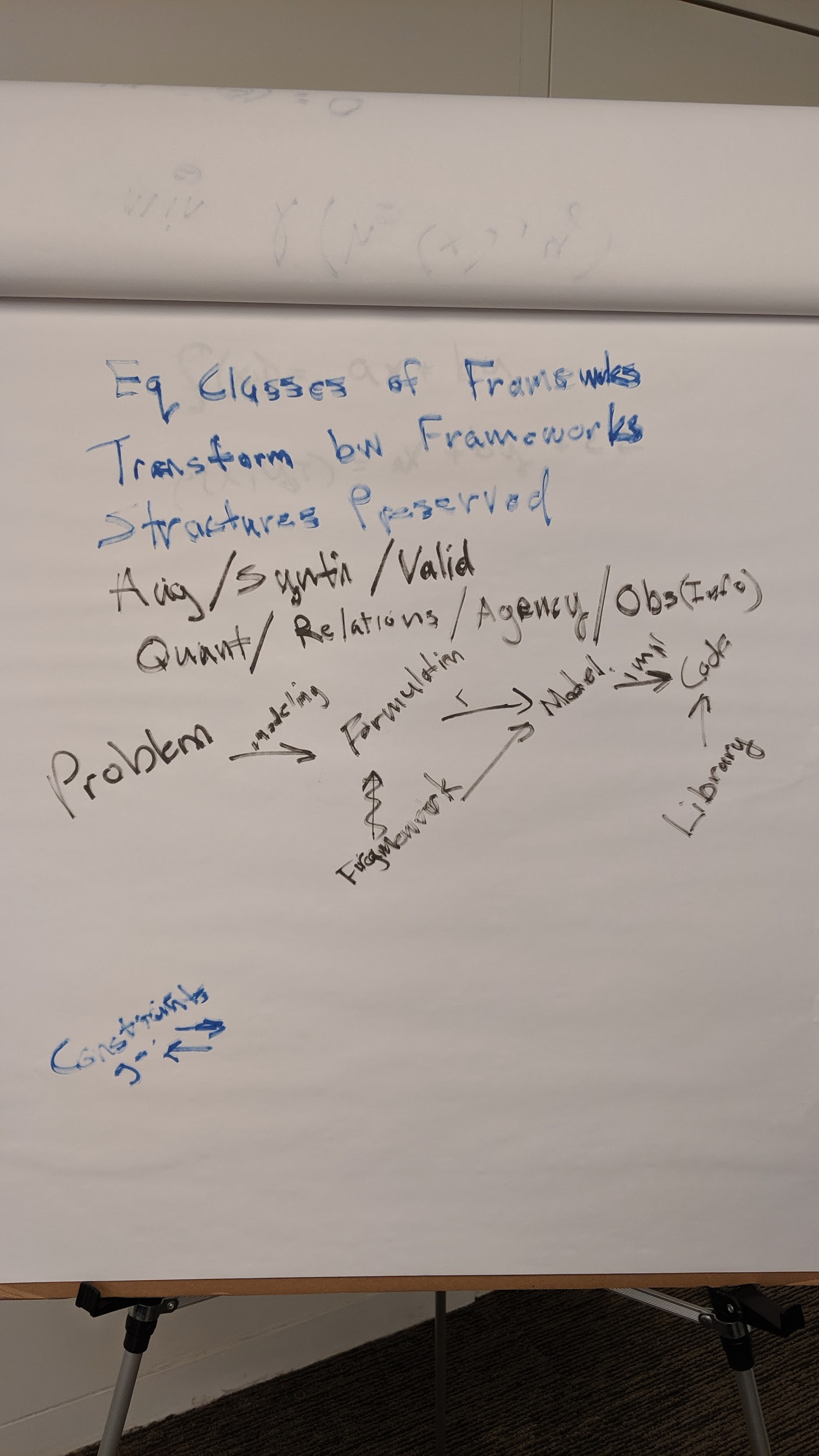
Ideally maximally expressive (i.e., encompassing the universe of all relevant models)

Can be analyzed to identify mapping to models (i.e., the Model block) thus excluding some classes of models



ABMs can be result-equivalent to other models (e.g., PRAMs) even if internal “wiring” is different





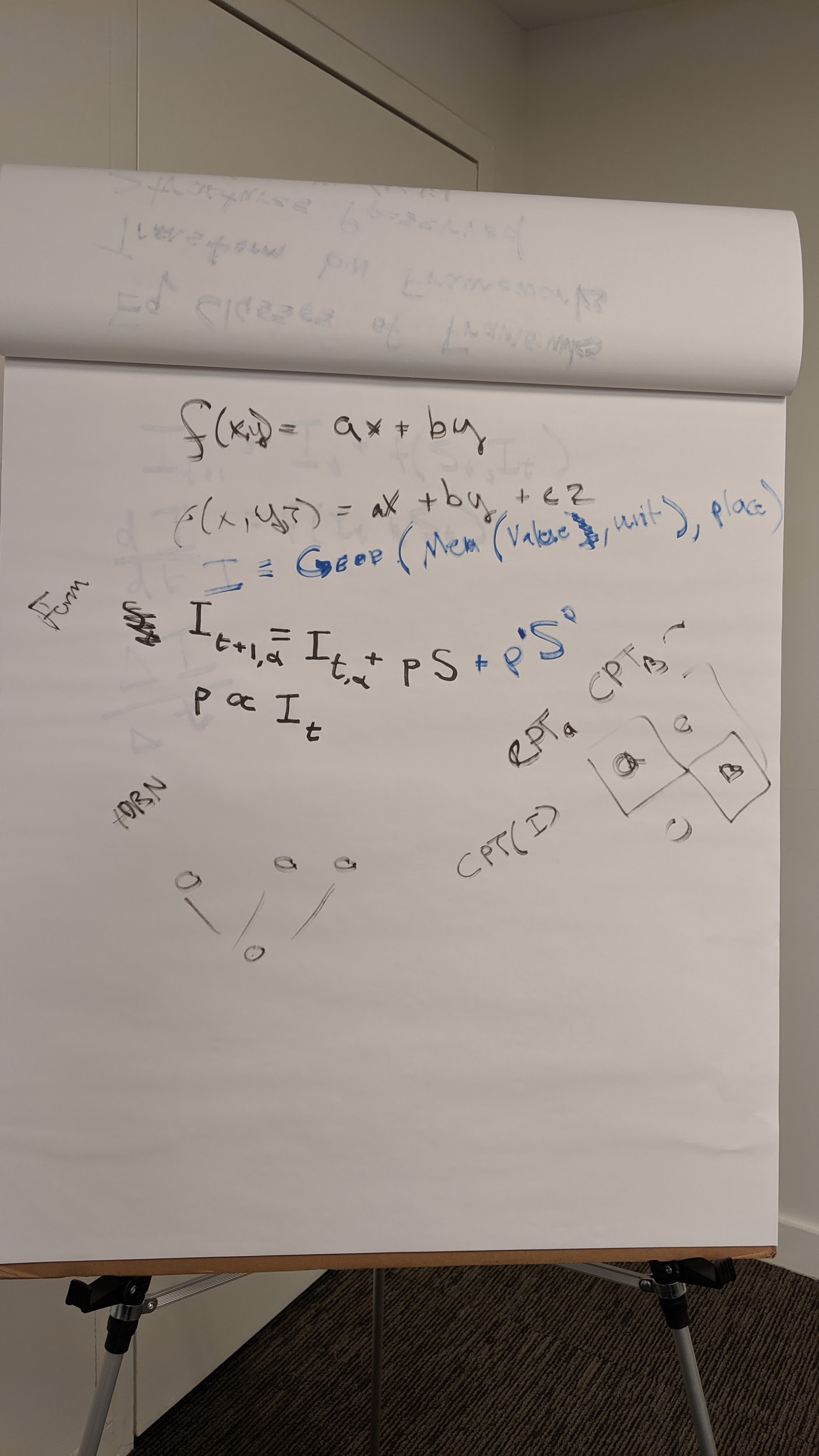
Equivalence Classes of Frameworks / Models

Transformation between Frameworks

What Structures are preserved

Augmentation, Synthesis, and Validation

Quantitative, Relational, Agency, Observation/Information



Another stab at a formulation language

DBNs vs more expressive DNB-like models (i.e., those with a changing transition model)

