Wrapup: More on connecting theory and practice

Arnon Rosenthal

(Generalizing Paolo, Georg, me) Aim for *mixed* solutions

General solution
- Automatic generation
- Principled
- Domain info as m'data

Ad hoc programs
- Manually generated
- Principled?
- Domain-specific

Hand solution
- Unprincipled
  (semantics of result?)
- Slow and costly

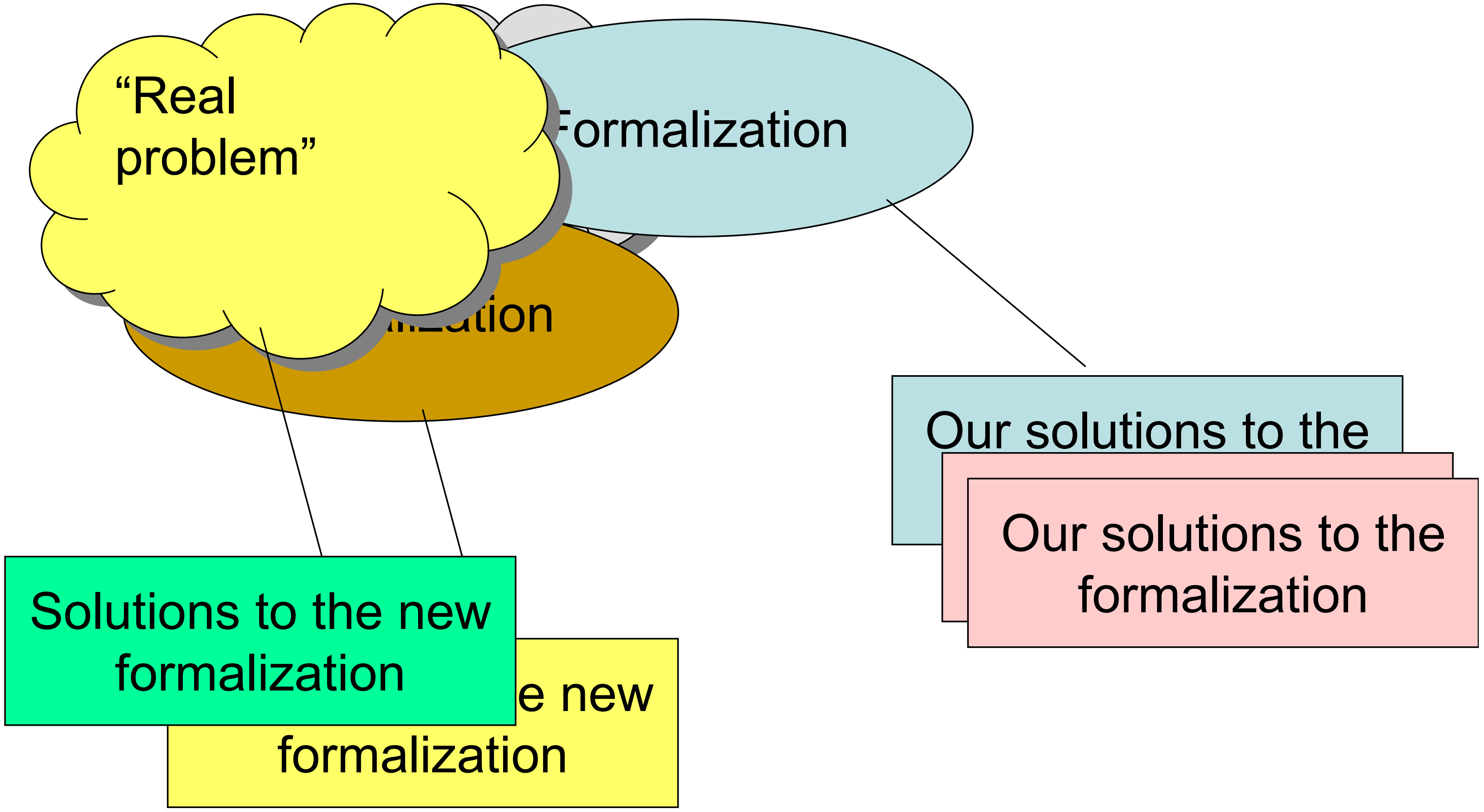
- You don't need to automate fully
  - Position yourself as part of the big picture

Workaround on hard problems --1

- A testbed will be hard (too hard ?), so …

  help couple tools, *loosely* (able to exchange data, but no common database and UI)
- Approach: Create specifications for exchanging the metadata they capture, e.g., TGDs, mappings
  - Bad behavior: After 20 years, still no "standard" data structure for Datalog programs
  - Advantage: Needs little coordination
- Step 1: Create a straightforward XML encoding, web service API, etc.
- Step 2: *Encourage adoption*
  - Provide some services *that others want*, that work off your specification (e.g., displays, analyses)

Data exchange theory involves hard problems. *How to get unstuck?*



Data exchange theory involves hard problems. *How to get unstuck?*

- Understand the range of *real* data exchange problems
  - From vendors?
  - From bio projects?
- Where do target schemas come from (different scenarios may have different requirements)?

- Half baked ideas follow
  - With possible research problems

Getting unstuck

- Ask for a *small* representation, not minimum
  - Algorithms that guarantee small solution?
  - Average performance?
- If you’re at a design stage, maybe you can change the target schema
  - Either structure or constraints
  - Algorithms to suggest something tractable (e.g., chase, to generate tractable constraints)
- Maybe the dependencies aren’t right?
  - Do the difficult cases indicate likely mistakes?

Binary vs. Tuple models

- Tuple representation can be a source of problems (Cartesian products). Would a binary (object, property, value) model be better?
  - Are there Cartesian product effects that could be removed by going to a binary or hierarchical model?
- If so
  - Important intellectually
  - We can’t drop support for relational
  - Perhaps could move *some* new capabilities

Getting unstuck

- Instead of Exists join value, define a new relationship in the target schema, and assert values directly
  - Business databases try to predefine the sorts of assertions to be accepted
  - Investigative dbs often allow new classes to be defined on the fly
    - The same tactic seems relevant to importing info from foreign sources
    - But training and keeping query sets gets tougher

Getting unstuck

- What if we split the constraint set (e.g., to break cycles), and "solved" a tractable case?
  - Can we decompose, incorporating the remaining constraints later?
  - Suggest where to break? Perhaps identify constraints that we can do without?

Distinction many years earlier

- *We should be interested in whether the transform is in some sense semantically right.*
- Some steps had a rigorous notion of info preservation
- Others changed the info. Most useful with a first cut specification, which one might not accept as correct.