Agentics Tutorial

Lecture 5: *Logical Transduction Algebra*

Guest lectures at Columbia University Class on Agentic AI, Fintech, and the Data Economy. - Prof. Agostino Capponi

Alfio Massimiliano Gliozzo

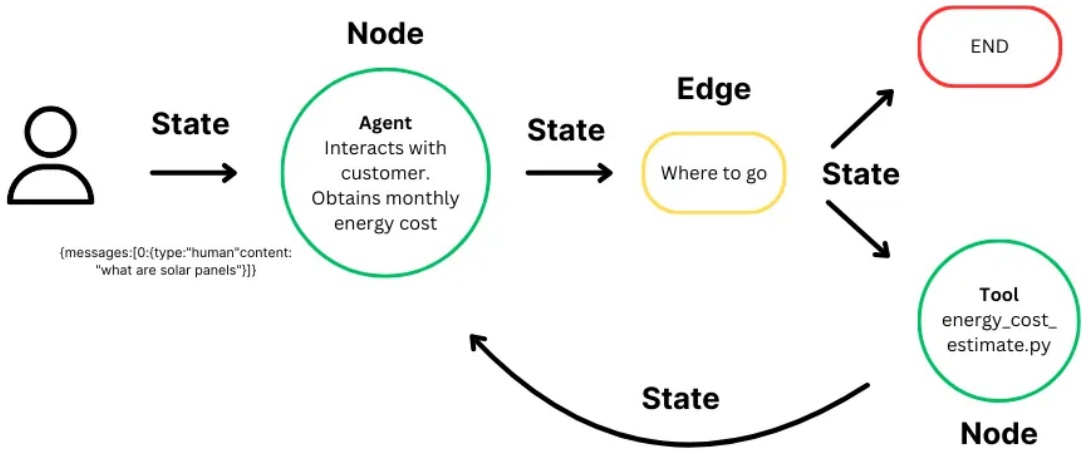
IBM research

gliozzo@us.ibm.com

Outline

- Agentics 1.0: Recap
- Logical Transduction Algebra
- Agentics 2.0 Design Patterns
  - Decision Making
  - Relation Extraction
  - Discovery
- Future Directions
  - No code interfaces
  - Self Code Generation
  - Sthochastic functions
- Discussion

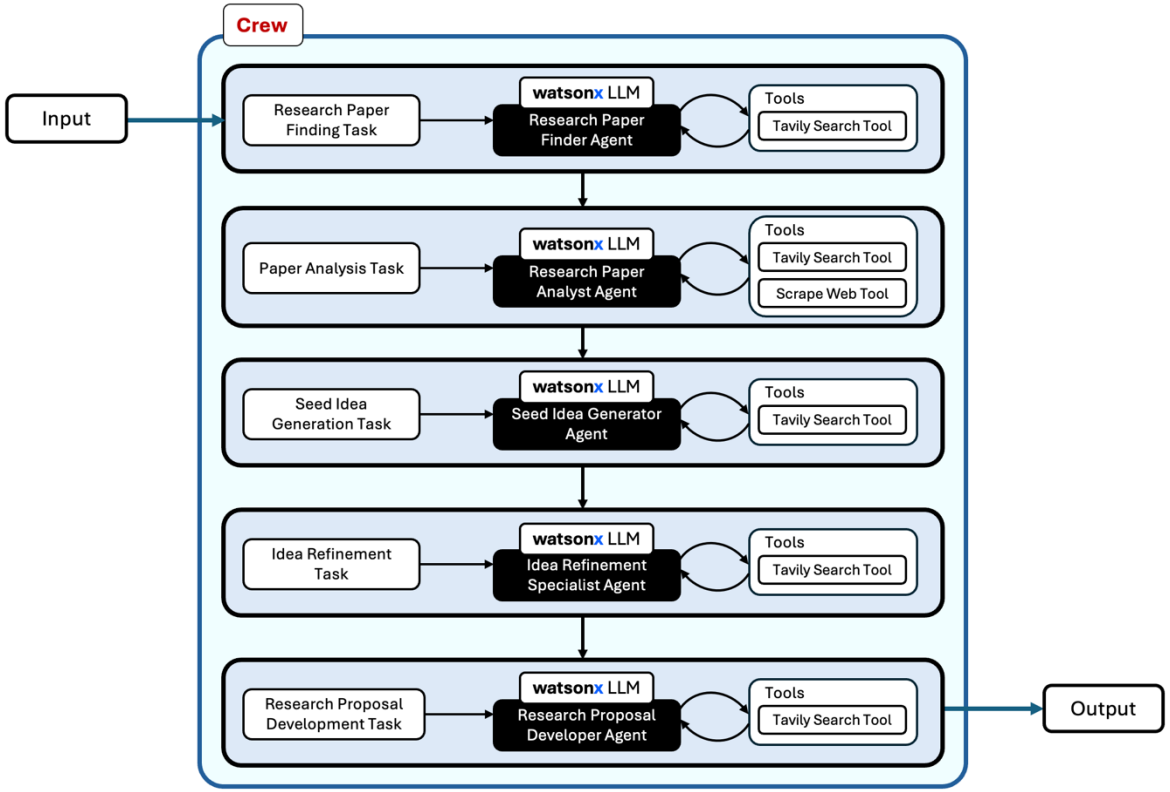
Low-Level Control-Flow Graphs (LangGraph)



- Allows explicit definition of state graphs for agent workflows.
- Pydantic objects flow through states, giving more deterministic control.
- Supports conditional routing, cycles, retries, and branches.
- Offers more structure, but still relies on agent-centric semantics.

Agentic Workflows: CrewAI

- Workflow structure defined implicitly through:
  - agent roles
  - task sequencing
  - tool invocation
- Easy to prototype, but workflow logic remains loosely specified and difficult to formalize.



Logical Transduction

The inference-driven transformation of an object *x* of type *X* into an object *y* of type *Y*, such that for all predicted slot values of *y* it is possible to provide a logical explanation supported by the slot values of x

$$\mathbf{y} = Y \ll \mathbf{x}$$

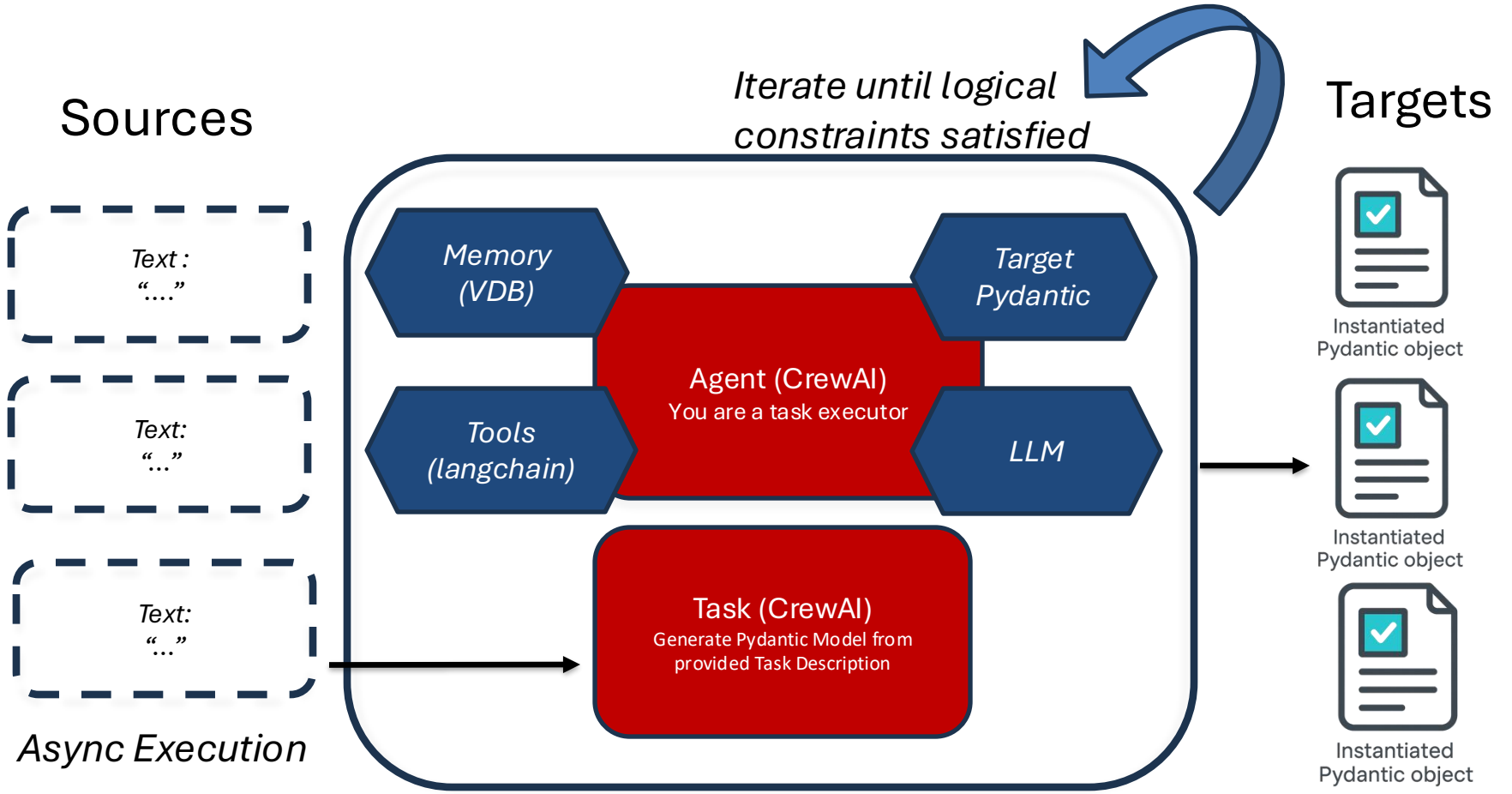
```
class SentimentSummary(BaseModel):
    sentiment: Literal["positive",
                       "neutral", "negative"]
    reason: str
```

```
class ProductReview(BaseModel):
    reviewer: str
    text: str
    stars: int
```

```
{"sentiment": "positive",
"reason": "Excellent quality and fast
    delivery"},
{"sentiment": "neutral", "reason": "Okay
    product, but package issues"},
{"sentiment": "negative",
"reason": "Broke after one use"
}
```

```
{"reviewer": "Alice", "text": "Excellent
    product quality and fast delivery!", "
    stars": 5},
{"reviewer": "Bob", "text": "It's okay, but
    the package was damaged", "stars": 3},
{"reviewer": "Carol", "text": "Terrible
    experience, broken after one use!", "
    stars": 1}
```

Logical Transduction is imp
stateless agents wit

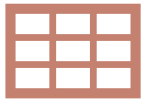


Pydantic Transducers are implemented by **stateless REACT agents with tools** whose goal is to generate an object of the **target pydantic type** which is logically inferred by their source and additional observations from tools, None otherwise. Reasoning and planning can be optionally used.

Scaling out GenAI workflows

Problems:

LLM calls are often executed in the order of seconds

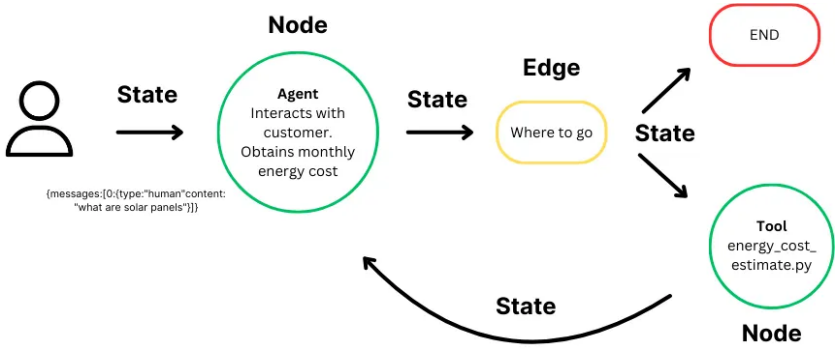


It takes 1 h to execute 1000 llm calls, which is a small size for financial analytics

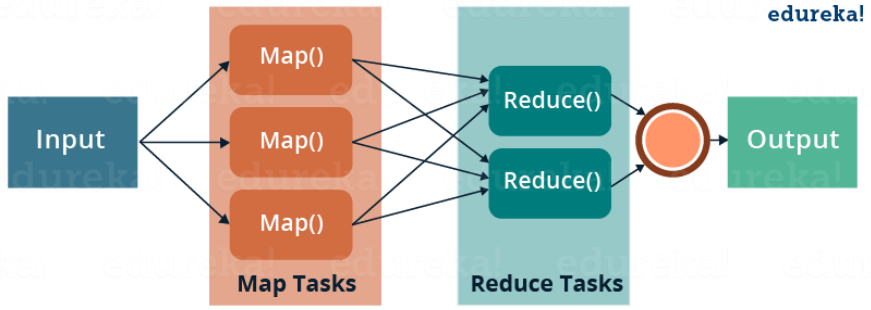
Natural Language based IO will trigger dozens of exceptions

Solutions:

State Graphs



Map Reduce[1]



[1]Source: https://www.edureka.co/blog/mapreduce-tutorial/

The Agentics Community

Git Repo:
https://github.com/IBM/agentics/

Lectures:
https://github.com/IBM/Agentics/tree/main/agentics_lectures

Tutorials:
https://github.com/IBM/Agentics/tree/main/tutorials

Student Projects:
https://github.com/orgs/AgenticsFintekColumbia/repositories

Demos:
https://d28ay7eykuuini.cloudfront.net/

Be an Agentics Ambassador (only if you truly enjoyed using it)

Feedback From Early Adopters:
https://docs.google.com/forms/d/1UNPJWANG2xbGb55j9XG_DfIQ_4lvXexW8Y6vJTw1wF0/edit#responses

Papers: Agentics:
https://arxiv.org/abs/2508.15610

https://arxiv.org/abs/2510.21117

Outline

- Agentics 1.0: Recap
- Logical Transduction Algebra
- Agentics 2.0 Design Patterns
  - Decision Making
  - Relation Extraction
  - Discovery
- Future Directions
  - No code interfaces
  - Self Code Generation
  - Sthochastic functions
- Discussion

Types and States

Definition — Type. A *Type* is defined recursively by:

$$T ::= b \mid [\, s_1 : T_1,\ \ldots,\ s_n : T_n \,],$$

where $b \in \mathsf{Basic}$ and

$$\mathsf{Basic} = \{\texttt{str}, \texttt{int}, \texttt{float}, \texttt{bool}, \ldots\}.$$

Thus a Type is either a basic type or a finite record of named *slots*, each slot s_i paired with a Type T_i.

Composition. Given two types

$$X = [\, s_i : X_i \,], \qquad Y = [\, t_j : Y_j \,],$$

their *composition* is the type

$$X \otimes Y = [\, \texttt{source} : X,\ \texttt{target} : Y \,].$$

```python
class Movie(BaseModel):
    movie_name: Optional[str] = None
    description: Optional[str] = None
    year: Optional[int] = None

class Genre(BaseModel):
    genre:Optional[str] = Field(
        None,
        description="return one category,   "
                    "e.g. comedy, drama, ...")
```

Type Composition

```python
ComposedType = Genre @ Movie
print(ComposedType.model_fields)
```

```
{'source': FieldInfo(annotation=Movie),
'target': FieldInfo(annotation=Genre)}
```

State Composition

```python
comp_inst = movie @ Genre(genre="Crime")
comp_inst.model_dump()
```

✓ 0.0s

```
{'source': {'genre': 'Crime'},
 'target': {'movie_name': 'The Godfather',
  'description': 'Spanning the years 1945 to
  'year': None}}
```

Transducible Functions

A function

$$f : X \to Y$$

is called a *transducible function* if it satisfies:

1. **Locality.** The output y is computed from a specific, non-empty subset of input slots:
$$\mathcal{E} \subseteq \text{slots}(X).$$

2. **Explainability.** The function provides an explanation e for the output y, derived strictly from the evidence subset:
$$e = \phi(x \upharpoonright X_{\mathcal{E}}, y),$$
where ϕ is an *explainability function* and $x \upharpoonright X_{\mathcal{E}}$ is the projection operation that select the slot in $\mathcal{E}$ from x.

3. **Provenance.** For each output object, the backward mapping to the input slots is explicit:
$$\mathcal{T}(y) = \mathcal{E},$$
forming a graph that records how the output is justified by the input.

Can be implemented using a mix of LLM and Python code

—

```python
@transducible(provide_explanation=True)
async def classify_genre(state:Movie)-> Genre:
    """Classify the genre of the source Movie """
    return Transduce(state)

genre, explanation  = await classify_genre(movie)
```


```
{
  "movie_name": "The Godfather",
  "description": "Spanning the years 1945 to 1955, a chronicle of the fictional
Corleone barely survives an attempt on his life, his youngest son, Michael steps
  "release_date": "3-24-1972"
}
{
  "genre": "Crime"
}
{
  "explanation": "The target genre 'Crime' was deduced from the source data. The
\"organized crime\", which are strong indicators of the crime genre. Additionall
reinforcing the inference. The 'release_date' does not contribute to genre deter
  "relevant_source_attributes": [
    "movie_name",
    "description"
  ],
  "confidence": 0.96
}
```

The Transduction Operator (<<)

The expression

$$\mathcal{F} = [\, Y \ll X \,]$$

denotes the class of all transducible functions that an LLM may instantiate

The LLM selects, among the class $[\, Y \ll X \,]$, the most likely function conditioned on the provided instruction string I. We denote this Maximum-Likelihood Estimator (MLE) by

$$(Y \ll X)^*_I.$$

The *canonical* transduction, used when no instructions are given, corresponds to the empty instruction string $I = \varepsilon$:

$$f := Y \ll X \;\; := \;\; (Y \ll X)^*_\varepsilon.$$

In the remainder of the paper, we treat transductions and their induced functions interchangeably, since each admissible pair of types (X, Y) is assumed to uniquely specify such a function. When applied to an instance $x \in X$

$$y = f(x) = (Y \ll X)(x) = Y \ll x$$

if $x \in X$, undefined otherwise

Dynamic Functions are Generated

```
classify_genre= Genre << Movie
genre = await classify_genre(movie)
print((genre @ movie).model_dump_json(indent=2))
```

✓ 1.1s


```
{
  "source": {
    "movie_name": "The Godfather",
    "description": "Spanning the years 1945 to 1955,
    "year": null
  },
  "target": {
    "genre": "Crime"
  }
}
```


```
await (Genre << movie)
```

✓ 1.0s

```
Genre(genre='Crime')
```

Composition

Let

$$f_1 = Y \ll X, \qquad f_2 = Z \ll Y$$

be two transducible functions. Their sequential composition is defined in the usual way:

$$f_2 \circ f_1 = Z \ll (Y \ll X), \qquad (f_2 \circ f_1)(x) = f_2(f_1(x)).$$

Monoid and Categorical Structure. Let

$$\Theta = \{\, f = Y \ll X \mid X, Y \text{ types} \,\}$$

be the set of all transducible functions. Because identities exist for every type and composition is associative,

$$(\Theta, \circ)$$

forms a monoid on the space of arrows.

```
input = GenericInput(content=
    "Write news story on Zoran Mandani won the Election"
    "in NYC and send it to Alfio")

write_mail = Email<<GenericInput
mail = await write_mail(input)
print(mail.model_dump_json(indent=2))

summary_composite_1 = Summary << write_mail
mail = await summary_composite_1(input)
print(mail.model_dump_json(indent=2))

summary_composite_2 = Summary <<(Email<<GenericInput)
mail = await summary_composite_2(input)
print(mail.model_dump_json(indent=2))
```

✓ 13.8s

```
{
  "to": "Alfio",
  "subject": "Zoran Mandani Wins NYC Election – News Story",
  "body": "In a historic turn of events, Zoran Mandani has emer
}
{
  "summary_text": "Zoran Mandani won the New York City mayoral
}
{
  "summary_text": "Zoran Mandani won the New York City mayoral
}
```

Map Reduce

5.1 The Map Operator

Let

$$f = X \ll Y$$

be a transducible function. Given a finite collection

$$[x_1, \ldots, x_N], \quad x_i \in [\![X]\!],$$

the *map operator* applies f pointwise:

$$f^{\leftarrow}([x_1, \ldots, x_N]) = [\, f(x_1), \ldots, f(x_N) \,].$$

5.2 The Reduce Operator

Let

$$r = Z \ll Y^N$$

be a transducible function that consolidates a collection of Y-typed instances into a single output of type Z. The *reduce operator* is defined as

$$r([y_1, \ldots, y_N]) = r(y_1, \ldots, y_N).$$

```
to_roman_number = RomanNumber << Number
roman_numbers = await to_roman_number([Number(number=i) for i in range(1,5)])
for roman in roman_numbers: print(roman)
```

```
roman_number='I'
roman_number='II'
roman_number='III'
roman_number='IV'
```

```
map = Number << With(RomanNumber,
    instructions="return the integer number corresponding to the roman numeral")

reduce = Number << With(Number,
    areduce=True,
    instructions="return the sum of the input numbers' number fields")

await reduce(await map(roman))
```

```
Number(number=10)
```

Outline

- Agentics 1.0: Recap
- Logical Transduction Algebra
- Agentics 2.0 Design Patterns
  - Decision Making
  - Relation Extraction
  - Discovery
- Future Directions
  - No code interfaces
  - Self Code Generation
  - Sthochastic functions
- Discussion

Multiple Choice Decision Pattern

Agentics Design Patter to perform multiple choice decision making.

```python
class DecisionMakingTask(BaseModel):
    task_description: str
    options: list[str]

class DecisionMakingCase(BaseModel):
    case_description: Optional[str] = None

class DecisionOutcome(BaseModel):
    chosen_option: str

decision_making= DecisionOutcome \
    << With(DecisionMakingTask & DecisionMakingCase,
            provide_explanation=True)
```

Returns a chosen option with explanation and confidence.

```python
task= DecisionMakingTask(
    task_description="Choose the best mode of transportation",
    options=["Car", "Bicycle", "Public Transit", "Walking"]
)
case = DecisionMakingCase(
    case_description="The destination is 5 miles away in a"
                     "busy urban area with moderate traffic.")

decision, explanation = asyncio.run(decision_making(task & case ))
```

Decentralized Collective Decision-Making

thousands of proposals across major DeFi protocols (Aave, Uniswap, Arbitrum etc.).

Human voters face **information overload**, **low participation**, and **power concentration**

No existing system provides **structured, scalable, and reasoning-capable** evaluation of proposals.

Question: ***Can an AI agent reliably evaluate proposals and align with collective human outcomes?***

Solution: Agentic Multiple Choice decision pattern
Input: proposal context, historical data, Voting Options (MCP tools)
Output: vote, justification

Evaluation
Dataset: 3,383 proposals from 8 major DAOs
Metric: decision agreement with final majority/economic outcomes
AI: 92.5%
Humans: (76.6%)

Agostino Capponi, Alfio Gliozzo, Chunghyun Han, Junkyu Lee, **DAO-AI: Evaluating Decentralized Collective Decision-Making through Agentic AI, accepted at AAAI 26 workshop on Agentic AI and Finance** https://arxiv.org/pdf/2510.21117

Outline

- Open Challenges in Agentic Workflows
- Agentics 2.0
- Logical Transduction Algebra
- Applications
  - Decision Making
  - Relation Induction
  - Discovery

Relation Extraction Pattern

Agentics Design Patter to identify relation among large collections of objects
Returns a list of entities and relations among them.

Types

```python
class InputData(BaseModel):
    data: list[str] = None

class Relation(BaseModel):
    subject: Optional[str] = None
    object: Optional[str] = None
    relation_type: Optional[str] = None

class Ontology(BaseModel):
    entities: Optional[list[str]] = None
    relations: Optional[list[Relation]] = None
```

Example Output on Financial Terms

Treasury **issues** Treasury Note
SEC **regulates** Treasury Note
Sortino Ratio **measures** Risk

Single LLM call Implementation

```python
extract_relations = Ontology<< With(InputData,
    instructions="Derive an Ontology from the input"
        "objects by identifying relations among them",
        verbose_transduction=False,
        batch_size=20)
```

Map Reduce Implementation

```python
async def relation_extracton_map_reduce(
        input_objects: InputData,
        n_clusters:int = 10) -> Ontology:

    terms=AG(states=input_objects.data)
    clusters = terms.cluster(n_partitions=n_clusters)
    ontologies = await extract_relations(clusters)
    return merge_ontologies(ontologies)
```

Relation Extraction Evaluation

Financial Terminology (Generated)

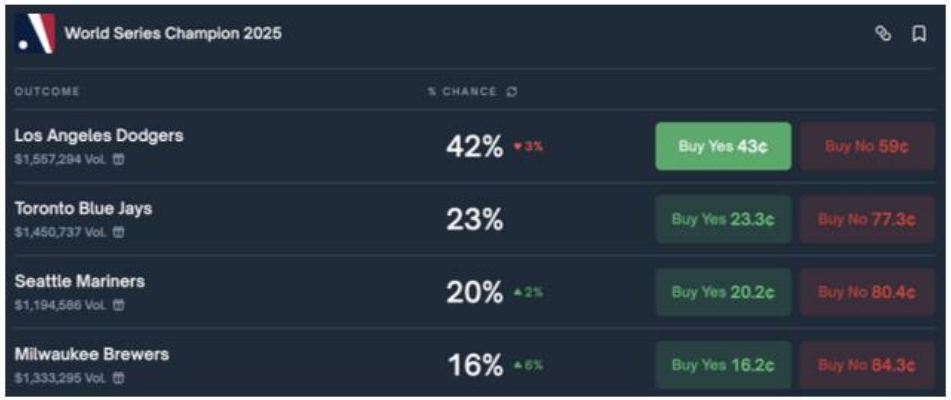
N terms	Term Generation Time	RE time	# RE	RE-MR time	RE-MR #
10	1.84	5.43	6	2.56	0
20	3.04	7.50	19	4.15	3
30	3.22	12.06	23	6.14	10
40	4.02	18.81	33	10.47	30
50	**4.36**	**19.09**	**49**	**5.94**	**40**
60	8.09	18.20	47	7.39	42
70	7.29	18.65	75	6.14	30
80	9.53	NA	NA	8.31	63
90	5.84	NA	NA	8.52	62
100	10.54	NA	NA	10.54	82
200	18.56	NA	NA	26.58	163
300	19.55	NA	NA	16.74	207

Open Domain (Wordnet)

N terms	RE-MR time	RE-MR relations
400	15.48	21
500	16.59	28
600	27.06	74
700	28.84	101
900	37.77	197
1000	61.20	387
1500	90.58	413
2000	143.69	706
3000	140.89	1060
4000	215.33	1474
5000	234.51	1689
10000	575.53	3963
20000	1170	8856
30000	2114.68	14528

Case Study: Polymarket

- Largest prediction by volume (~$200 billion cumulative volume), operating on Polygon blockchain
- Known for "predicting" notable events (e.g. 2024 US Presidential election, 2025 Nobel Peace Prize winner)
- Market makers deposit $1 of collateral for 1 YES and 1 NO token, post quotes on YES and NO order books



Co-authors: Brian Zi Qi Zhu, Agostino Capponi

Clustering Polymarket Bets

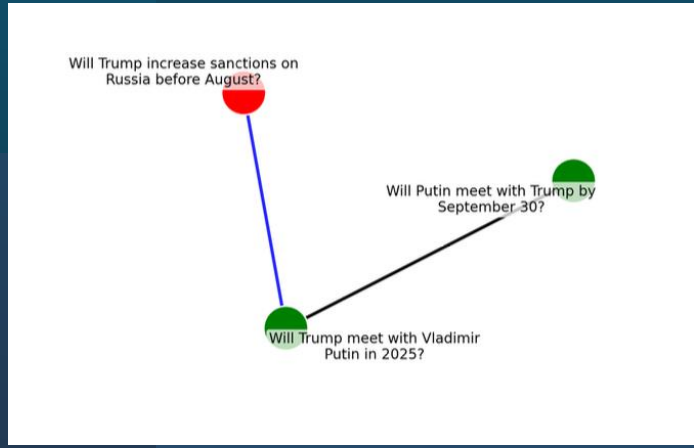
- Numerically encodes each market question → cluster markets using k-means algorithm (pre-set number of clusters) → categorize clusters (sports, politics, etc.)
- Very fast: can cluster thousands of markets on the order of seconds

```
cluster 8: politics
    - Will Trump impose large tariffs in 2025?
    - Trump ends taxes on tips in 2025?
    - Will Trump lower tariffs on Canada by July 31?
    - Will Trump remove 10% blanket tariff before July?
    - Trump ends taxes on tips before August?
    - 50% tariff goes into effect on EU by July 9?
    - Will Trump remove majority of reciprocal tariffs before 90 day deadline?
    - Trump cuts taxes in 2025?
    - Will anyone be charged with insider trading Trump tariff announcement?
    - Will Trump impose large tariffs in his first 6 months?
```

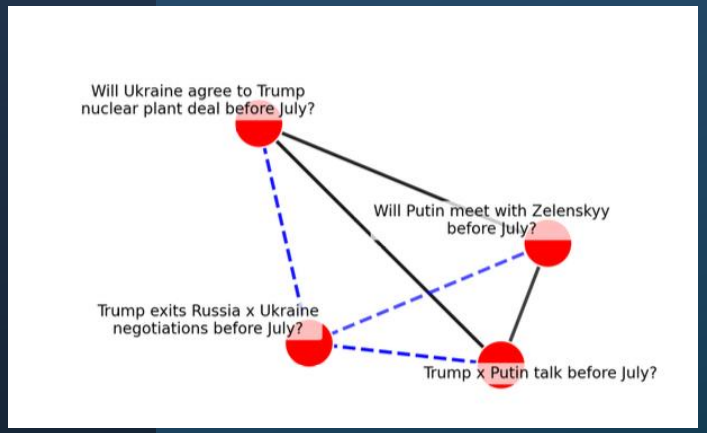
```
cluster 32: geopolitics
    - Will Netanyahu meet with Ahmed al-Sharaa before August?
    - Will Putin meet with Trump by September 30?
    - Will Putin meet with Zelenskyy before July?
    - Will Putin meet with Zelenskyy before August?
    - Trump exits Russia x Ukraine negotiations before July?
    - Will Ukraine agree to Trump nuclear plant deal before July?
    - Will Trump meet Pope Leo XIV before August?
    - Will Trump meet with Lula da Silva in 2025?
    - Will the Vatican host Russia-Ukraine peace talks before September?
    - Will Trump increase sanctions on Russia before August?
    - Trump x Putin talk before July?
    - Will Trump meet with Vladimir Putin in 2025?
    - Will Trump meet with Netanyahu before August?
    - Will Trump meet with Zelenskyy before August?
    - Will Zelenskyy wear a suit before July?
```

Causal Outcome-Equivalence Relation

A relation induced between two events i and j expressing whether outcomes are expected to be the same or different due to an inferred causal interaction.



- We use structured 'Atypes' in the Agentics framework
- Input data:
  - Event corresponding to prediction market, phrased as question
  - Market start time: when users can start trading on the market
  - Market end time: when market closes for trading
- Output data – a list of market relations, described by the following:
  - Questions *i* and *j* (names of markets that agent believes to be related)
  - Does the agent believe these markets will have the same or different outcomes?
  - Confidence score (between 0 and 1) and rationale: indicating agent's reasoning and confidence in answers



Evaluation

Cause: Israel x Iran ceasefire before July?

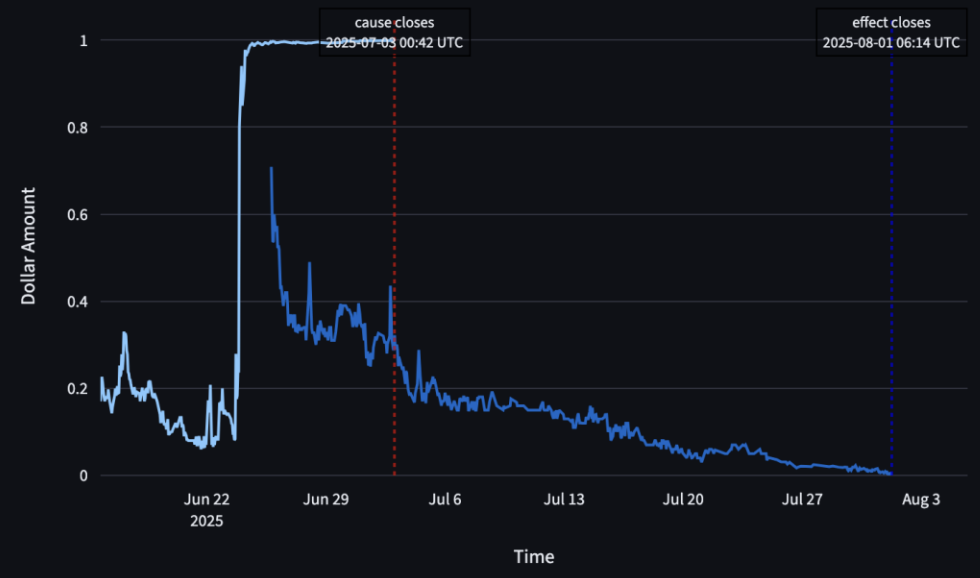
Effect: Israel strikes Iran before August?

System Prediction (is same outcome?): False

Rationale: Similar to military action, if an Israel x Iran ceasefire is in place before July (Q_i=Yes), it would make Israel striking Iran before August (Q_j=Yes) less probable (Q_j=No). The absence of a ceasefire (Q_i=No) would increase the likelihood of strikes (Q_j=Yes). This suggests an inverse relationship.

Gain for this bet: 0.3217

Bet Values Over Time



29 Bets Selected

ROI 12% (2 weeks)

Cause: Will Pump.fun perform an airdrop by July 31?

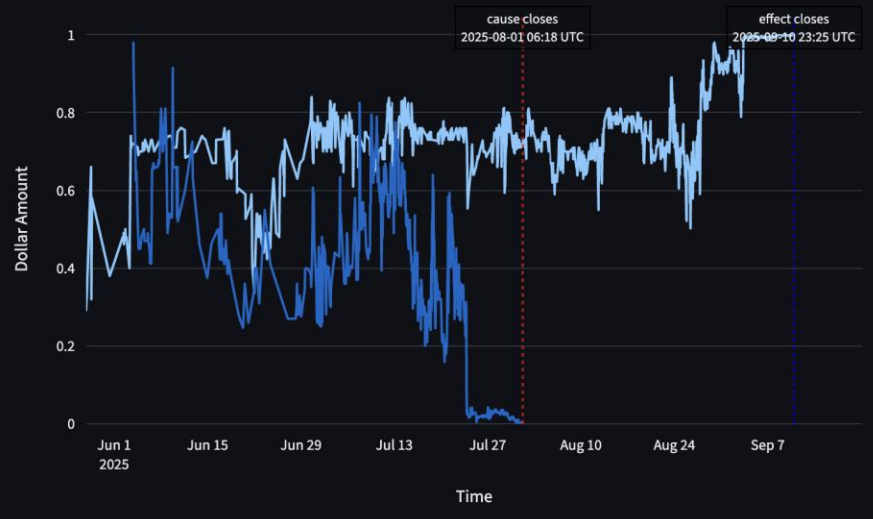
Effect: Linea airdrop in Q3 2025?

System Prediction (is same outcome?): True

Rationale: Both questions relate to airdrops expected around the beginning of Q3 2025 (July). While Pump.fun is Solana-based and Linea is an Ethereum L2, the decision to conduct an airdrop can still be influenced by overarching crypto market trends, investor sentiment, and the perceived success or timing of other significant airdrops. If the early Q3 period proves to be a strong 'airdrop season,' both projects might seize the opportunity. Therefore, there's a reasonable likelihood of their outcomes being correlated in the same direction.

Gain for this bet: -0.2776

Bet Values Over Time

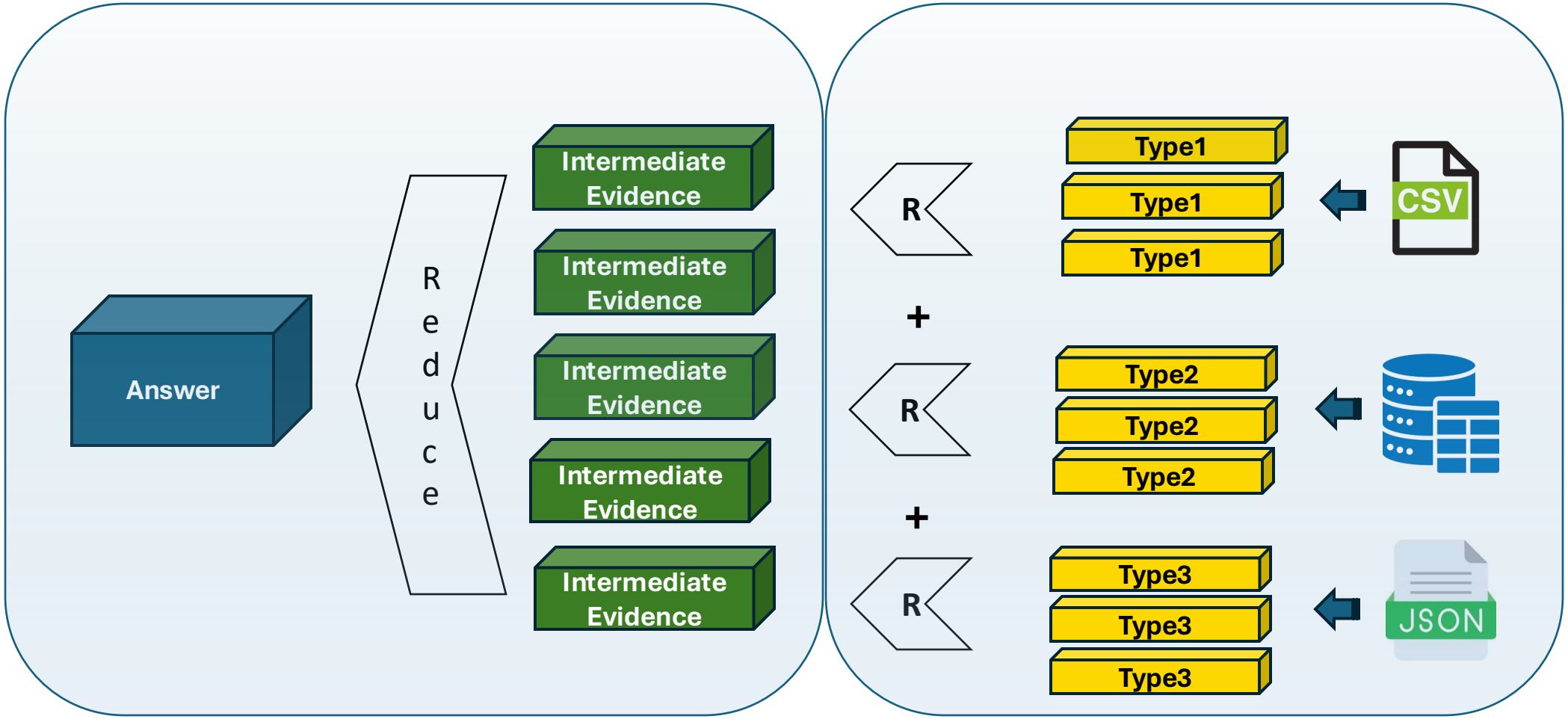


Outline

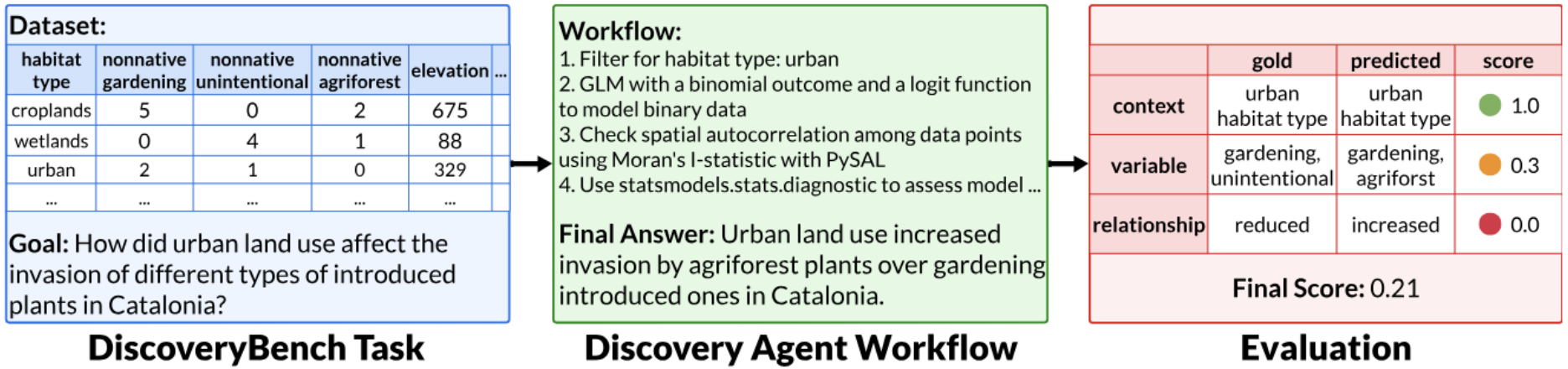
- Open Challenges in Agentic Workflows
- Agentics 2.0
- Logical Transduction Algebra
- Applications
  - Decision Making
  - Relation Induction
  - Discovery

Discovery Pattern

Agentics Design Pattern to answer questions by performing extensive scan of given source data
Async and massively parallel transduction of heterogeneous data into common intermediate Evidence
Reduction of intermediate evidence into final answer



Discovery Bench Task



	Train	Test
# tasks	25	239
# unique hypotheses	14	144
# tasks need $>$ 1 dataset	4	110
# domains	3	6

Table 1: Statistics for DB-REAL

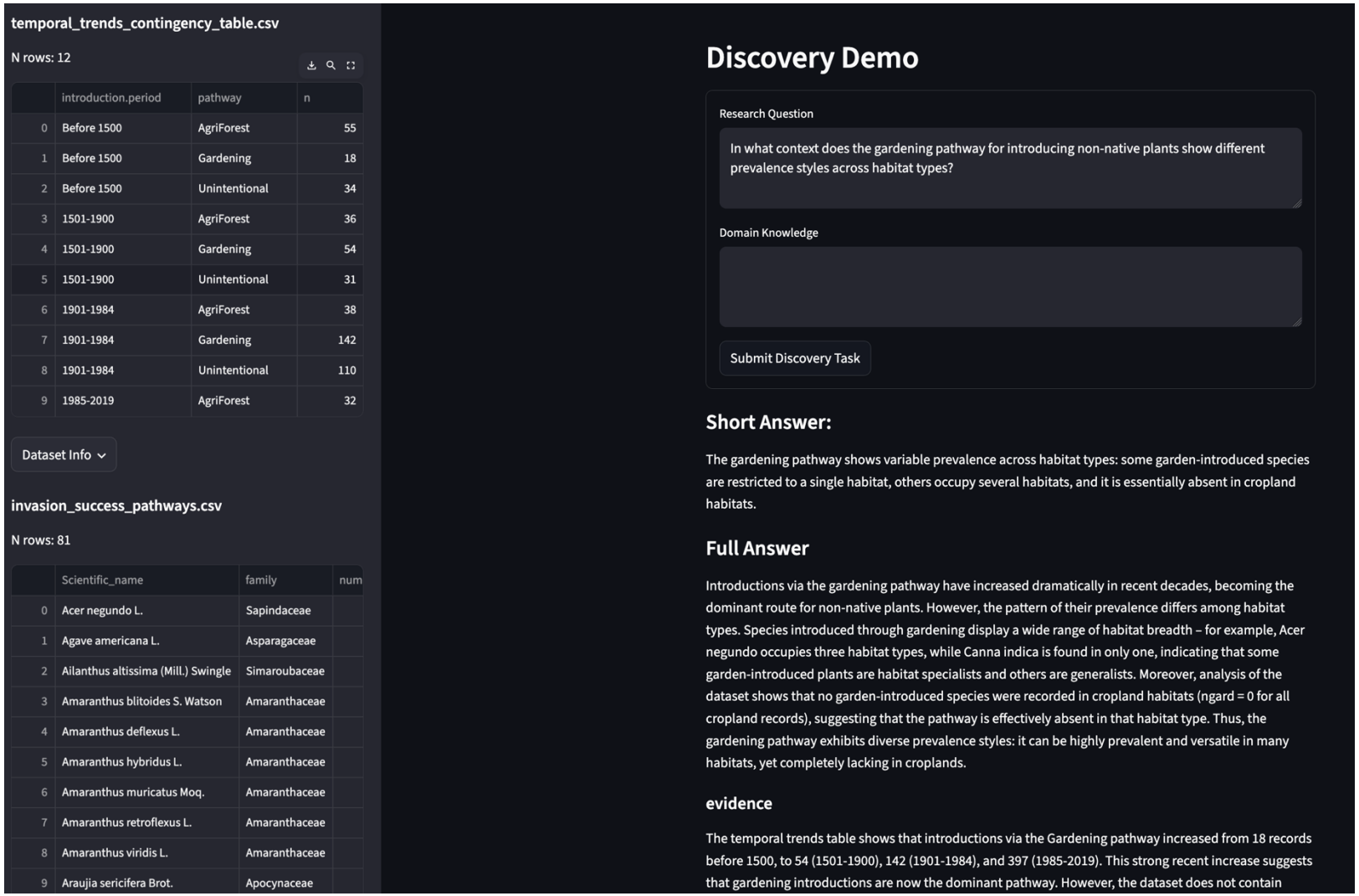
	GPT-4o	GPT-4p	Llama-3
DB-REAL			
NoDataGuess	0.0	4.7	11.5
CodeGen	15.5	16.3	12.1
React	15.4	15.6	13.5
DataVoyager	15.4	13.9	11.5
Reflexion (Oracle)	**24.5**	**19.5**	**22.5**

Data Voyager: uses planner, code generator, data analysis, and critic, that orchestrate the discovery process.

Reflection: uses HMS from ground truth, which: measures the degree to which answer and GT align

Data Voyager: B. P. Majumder, H. Surana, D. Agarwal, S. Hazra, A. Sabharwal, and P. Clark. Data-driven discovery with large generative models. ICML, 2024.

Discovery Bench Experiments and Demo



	GPT-4o	GPT-4p	Llama-3
DB-REAL			
NoDataGuess	0.0	4.7	11.5
CodeGen	15.5	16.3	12.1
React	15.4	15.6	13.5
DataVoyager	15.4	13.9	11.5
Reflexion (Oracle)	**24.5**	**19.5**	**22.5**

Agentics Discovery : 0.214

OSS LLM : gpt-oss-120

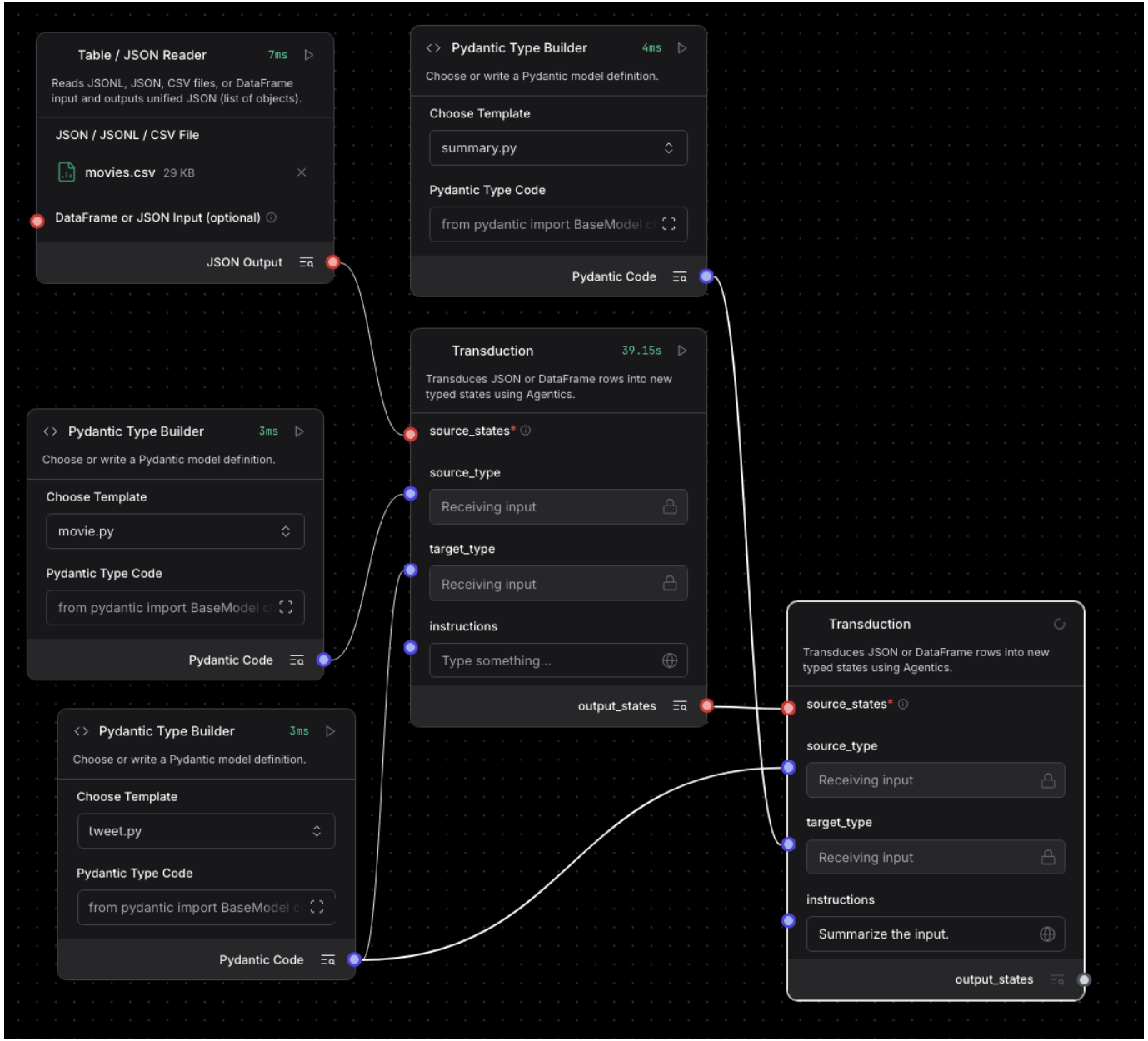
Demo http://9.59.195.168:8501/
(Need IBM VPN)

Majumder, B., Surana, H., Agarwal, D., Dalvi, B., Meena, A., Prakhar, A., Vora, T., Khot, T., Sabharwal, A., & Clark, P. (2024). DiscoveryBench: Towards Data-Driven Discovery with Large Language Models. ArXiv, abs/2407.01725.

Outline

- Agentics 1.0: Recap
- Logical Transduction Algebra
- Agentics 2.0 Design Patterns
  - Decision Making
  - Relation Extraction
  - Discovery
- Future Directions
  - No code interfaces
  - Self Code Generation
  - Sthochastic functions
- Discussion

No Code Interfaces: Langflow



Stochastic Functions

A **mathematical function** acts on deterministic instances:

$$f : X \to Y, \qquad \forall x \in X,\ \exists!\, y = f(x).$$

Such mappings are unique and well-defined, but contain no inherent model of uncertainty.

A **stochastic function** extends this notion by associating each deterministic input $x \in X$ with a probability distribution over Y:

$$F : X \to \mathcal{S}(Y), \qquad y \sim F(x).$$

4.1 Composition

Let

$$T_1 : \mathcal{S}(X) \to \mathcal{S}(Y), \qquad T_2 : \mathcal{S}(Y) \to \mathcal{S}(Z)$$

be two valid transducible functions mapping structured stochastic types. In the continuous case, their composition follows the law of total probability:

$$(T_2 \circ T_1)(z \mid x) = \int_y P_{T_2}(z \mid y)\, P_{T_1}(y \mid x)\, dy.$$

Why it matters:

LLM generates multiple hypothesis for each single transduction and propagates them over the transduction flow. Ambiguity is then resolved at the end taking all options into consideration.

Outline

- Agentics 1.0: Recap
- Logical Transduction Algebra
- Agentics 2.0 Design Patterns
  - Decision Making
  - Relation Extraction
  - Discovery
- Future Directions
  - No code interfaces
  - Self Code Generation
  - Sthochastic functions
- Discussion

Was it fun?

Share your experience

moment of pleasure and frustration

Aha ha moments

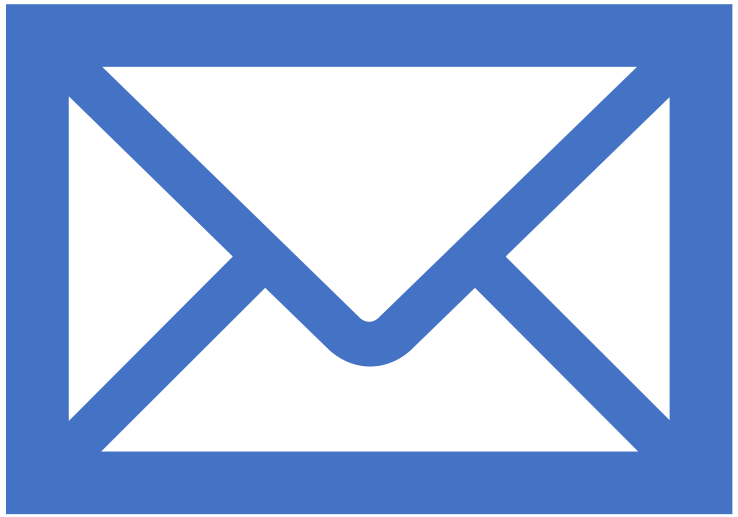
Why Agentics?

What new classes of applications can be built that would be difficult or impossible with traditional pipelines or LLM APIs?

What’s Next?

If you were leading the next iteration of Agentics, what would you prioritize improving or adding?

Contacts



Mail: gliozzo@us.ibm.com

Linkedin: https://www.linkedin.com/in/gliozzo/

Offices:

- One Madison Ave, NY, NY
- IBM Research center, Yorktown Heights