Generative Ontologies

Ryan Wisnesky | Conexus AI | ryan@conexus.com | Presented at Texas Data Day 2025

- This is really a talk about “***generative symbolic AI***”, which I will motivate using ontologies (part 1) and data-driven expert systems (part 2).

- What is an ontology?
- I claim an ontology is a sparse “deductive database”.

- What is a deductive database?
- I claim a deductive database is a regular database modulo logical rules.

- That is, a deductive database contains not just a finite set of data, but all the (possibly infinite) data deducible from that data using a set of logical rules.

- This is an old idea – you don’t even need a computer to have an ontology (e.g. the Dewey Decimal System for Libraries).

Ontologies

© Conexus AI, Inc. All Rights Reserved.

BFO 2020 Participation Axioms

Participates in and has participant are inverse relations [xjr-1]

$$\forall t,a,b\,(\mathrm{participatesIn}(a,b,t) \leftrightarrow \mathrm{hasParticipant}(b,a,t))$$

At every time a process exists it has a participant [trl-1]

$$\forall p,t\,(\mathrm{instanceOf}(p,\mathrm{process},t) \rightarrow \exists c\, \mathrm{participatesIn}(c,p,t))$$

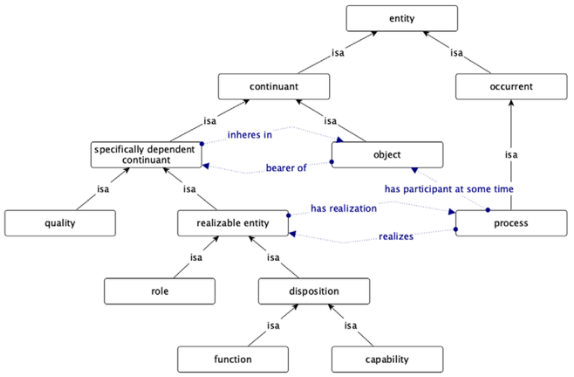
Participates in is dissective on third argument, a temporal region [yjm-1]

$$\forall p,q,r,s\,(\mathrm{participatesIn}(p,q,r) \wedge \mathrm{temporalPartOf}(s,r) \rightarrow \mathrm{participatesIn}(p,q,s))$$

If c participates in p at t and p occupies temporal region r then t is part of r [kxe-1]

$$\forall c,p,r,t\,(\mathrm{occupiesTemporalRegion}(p,r) \wedge \mathrm{participatesIn}(c,p,t) \rightarrow \mathrm{temporalPartOf}(t,r))$$

…



BFO – Basic Formal Ontology

© Conexus AI, Inc. All Rights Reserved.

1.1 A data exchange example 5

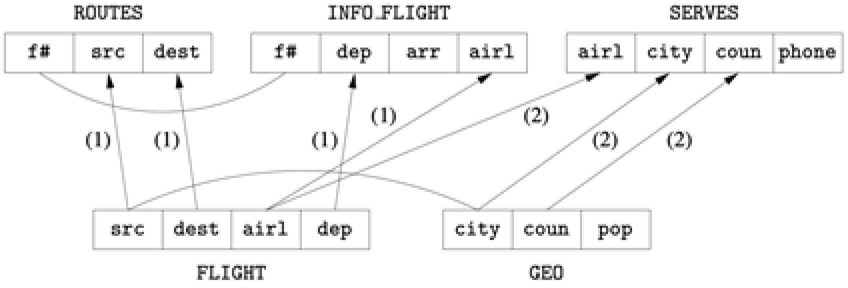
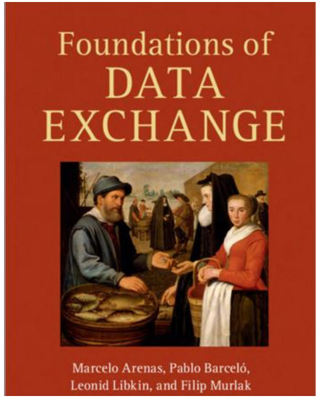


Figure 1.2 Schema mapping: a proper graphical representation

1.1 A data exchange example

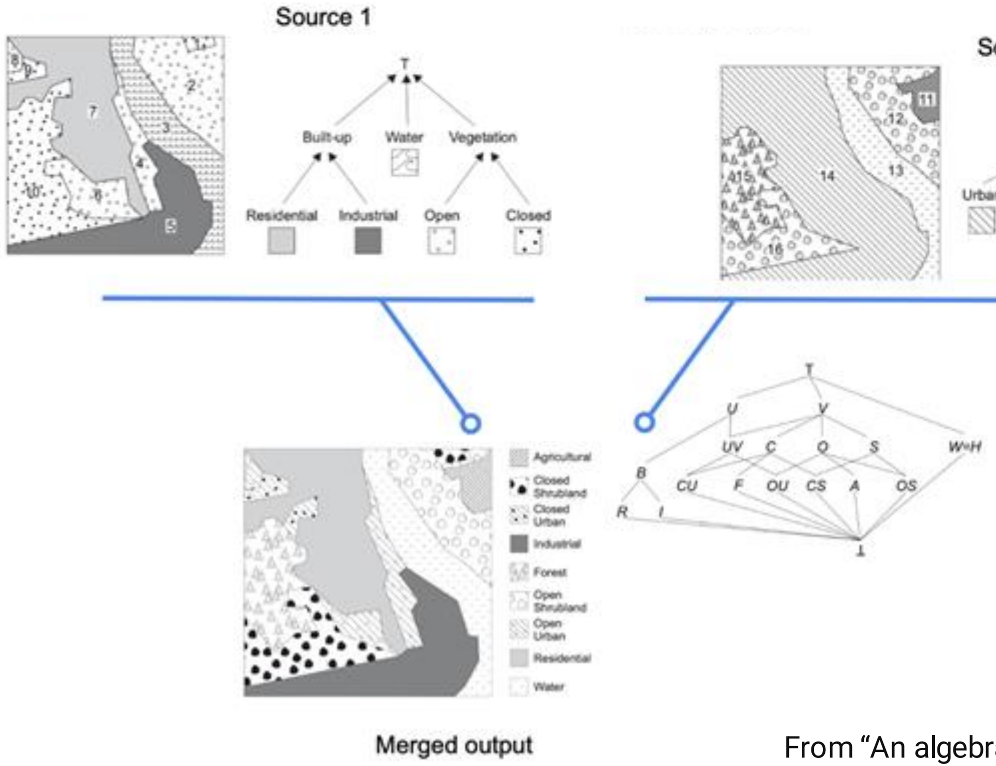
$$
\begin{array}{ll}
(1) & \texttt{FLIGHT(src,dest,airl,dep)} \longrightarrow \\
 & \quad \exists \texttt{f\#}\, \exists \texttt{arr}\, \big(\texttt{ROUTES(f\#,src,dest)} \\
 & \qquad \wedge\ \texttt{INFO\_FLIGHT(f\#,dep,arr,airl)} \big) \\
 & \\
(2) & \texttt{FLIGHT(city,dest,airl,dep)} \wedge \texttt{GEO(city,country,popul)} \\
 & \quad \longrightarrow \exists \texttt{phone}\ \texttt{SERVES(airl,city,country,phone)} \\
 & \\
(3) & \texttt{FLIGHT(src,city,airl,dep)} \wedge \texttt{GEO(city,country,popul)} \\
 & \quad \longrightarrow \exists \texttt{phone}\ \texttt{SERVES(airl,city,country,phone)}
\end{array}
$$

Figure 1.3 A schema mapping



Ontologies for data migration

© Conexus AI, Inc. All Rights Reserved.

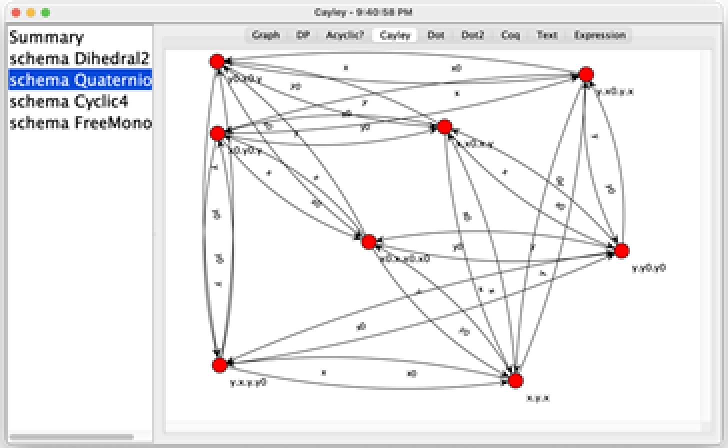


From "An algebraic approach to automated information fusion"

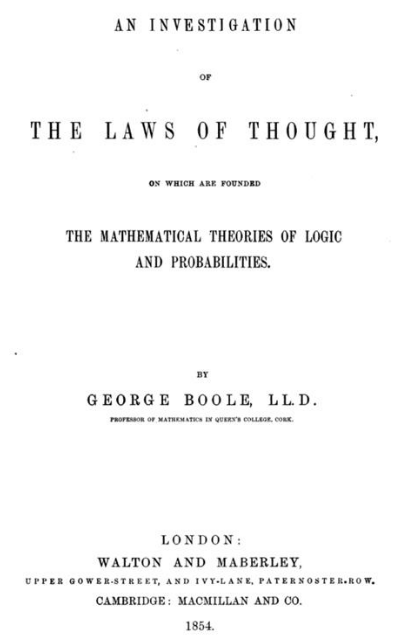
Satellite Image Ontology Merge

© Conexus AI, Inc. All Rights Reserved.

```
schema Quaternions = literal : empty {
	entities
		G
	foreign_keys
		x y x0 y0 : G -> G
	path_equations
		G.x.x0=G
		G.x0.x=G
		G.y.y0=G
		G.y0.y=G
		G.x.x.x.x = G
		G.x.x = G.y.y
		G.y0.x.y = G.x0
}
```



“Presentations by generators and relations”



AN INVESTIGATION

OF

THE LAWS OF THOUGHT,

ON WHICH ARE FOUNDED

THE MATHEMATICAL THEORIES OF LOGIC AND PROBABILITIES.

BY

GEORGE BOOLE, LL.D.

PROFESSOR OF MATHEMATICS IN QUEEN'S COLLEGE, CORK.

LONDON:
WALTON AND MABERLEY,
UPPER GOWER-STREET, AND IVY-LANE, PATERNOSTER-ROW.
CAMBRIDGE: MACMILLAN AND CO.
1854.

Ontologies for Math

© Conexus AI, Inc. All Rights Reserved.

patient

ID	first_name	last_name	birthdate	create_date
937189	john	doe	340465020	1187438212
937190	amrit	kumar	246222505	1187444008
937191	alexandra	grant	121408849	1187445155

visit

ID	patient_id	visit_date
12378727	937189	1187438212
12378728	937190	1187444008
12378729	937191	1187445155

observation

ID	visit_id	clinician_id	obs_type	observation
487298329	12378727	562	HR	114
487298330	12378727	562	WT	180
487298331	12378727	562	BP	130/82

+

patient

ID	fname	lname	dob
25234	alexandra	grant	121408849
25235	vincent	von hoff	409235232
25236	brian	tsai	380665171

prescription

ID	patient_id	date	details
675345	25234	1639676732	Enalapril Ma...
675346	25234	1639696544	chlorthalid...
675347	25235	1639704522	Lisinopril 5...

observation

ID	patient_id	clinician_id	obs_type	observation
154298449	25234	132	HR	116
154298450	25234	132	WT	220
154298451	25234	132	BP	132/82

=

patient

ID1	ID2	fname	lname	dob	create_date
937191	25234	alexandra	grant	121408849	1187445155
	25235	vincent	von hoff	409235232	
	25236	brian	tsai	380665171	
937189	?1	john	doe	340465020	1187438212
937190		amrit	kumar	380665171	1187444008

visit

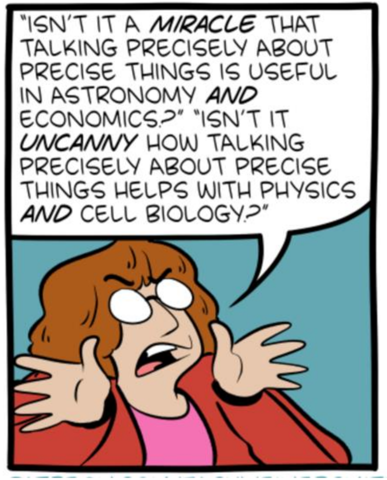
ID	patient_id	visit_date
12378727	937189	1187438212
12378728	937190	1187444008
12378729	937191	1187445155
?0	937191	1639676732

prescription

ID	patient_id	date	details
675345	25234	1639676732	Enalapril Ma...
675346	25234	1639696544	chlorthalid...
675347	25235	1639704522	Lisinopril 5...

observation

ID	visit_id	clinician_id	obs_type	observation	patient_id
487298329	12378727	562	HR	114	?1
487298330	12378727	562	WT	180	?1
487298331	12378727	562	BP	132/82	?1
154298449	?0	132	HR	116	25234
154298450	?0	132	WT	220	25234
154298451	?0	132	BP	132/82	25234



Ontologies in data integration

© Conexus AI, Inc. All Rights Reserved.

Lots of Ontology Systems

OLOGS

Ologs are a way to define ontologies using the branch of math called applied category theory. A number of systems implement ologs, including Algebraic Julia, Statebox CQL, Conexus CQL, and EASIK. One key differentiator between ologs and ontologies is that ologs allow data migration from one olog to another and so enable data integration and migration. Another is their level of expressive power, with arbitrary Excel spreadsheets being ontologies.

RDF/OWL

The Web Ontology Language is a way to define ontologies using subject-predicate-object triples. Highly effective in the hands of an expert and widely available, its foundations require care to be taken by users querying it. Data.world provides RAG examples where ontologies improve LLM performance.

DATALOG/PROLOG

Datalog one of the original ways to define simple ontologies. Widely deployed in defense applications, it has many implementations, including on GPUs. Prolog extends datalog with additional expressive power.

CYC

Cyc Is a long-term artificial intelligence project that aims to assemble a comprehensive ontology and knowledge base that spans basic concepts and rules about how the world works. Hoping to capture common sense knowledge, Cyc focuses on implicit knowledge. The project began in July 1984 at MCC and was developed later by the Cycorp company. It has many clients today.

APACHE TINKERPOP

Tinker pop is commonly used to construct knowledge graphs, which, when coupled with business rules written in Gremlin/java, allow the definition of ontologies. This line of thinking has been taken up by Tinkerpop's creator in the mm-adt project.

MICROSOFT LAMBDA GRAPH (HYDRA)

A successor to Uber's Dragon project, Microsoft Lambda graph focuses on representing computational knowledge graphs using the lambda calculus, enabling new features such as type inference for graphs and active graphs that evolve on their own over time. Like Ologs, Lambda graph allows for data migration between ontologies.

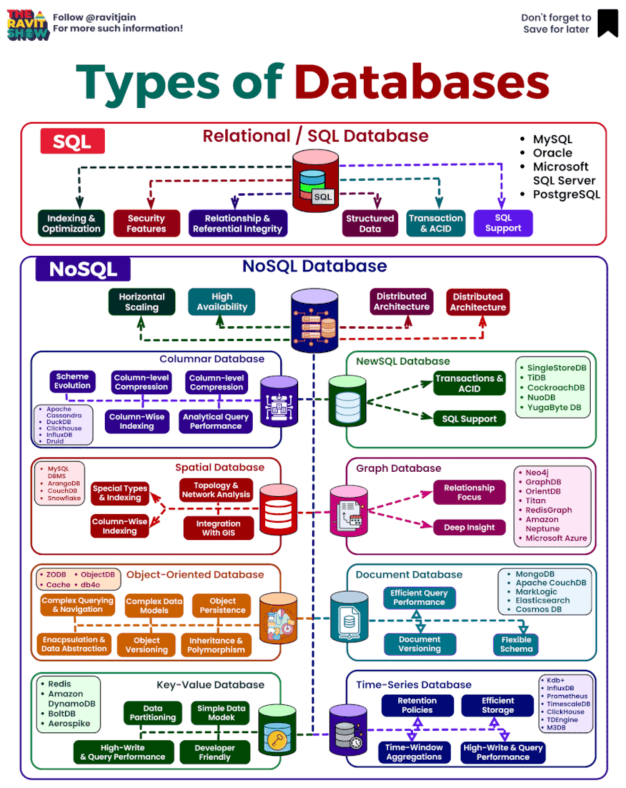
List available at http://conexus.com/ontology

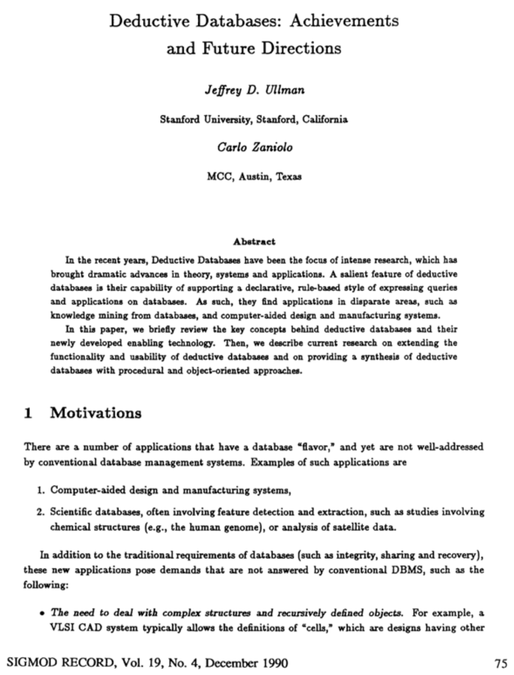
Ologs: categoricaldata.net

Hydra: github.com/CategoricalData/hydra

© Conexus AI, Inc. All Rights Reserved.

- Deductive databases were heavily studied in the 1980s
- Ontologies formed the basis of RDF/OWL
- So why are deductive databases and/or ontologies not popular?





Deductive Databases: Achievements and Future Directions

Jeffrey D. Ullman

Stanford University, Stanford, California

Carlo Zaniolo

MCC, Austin, Texas

Abstract

In the recent years, Deductive Databases have been the focus of intense research, which has brought dramatic advances in theory, systems and applications. A salient feature of deductive databases is their capability of supporting a declarative, rule-based style of expressing queries and applications on databases. As such, they find applications in disparate areas, such as knowledge mining from databases, and computer-aided design and manufacturing systems.

In this paper, we briefly review the key concepts behind deductive databases and their newly developed enabling technology. Then, we describe current research on extending the functionality and usability of deductive databases and on providing a synthesis of deductive databases with procedural and object-oriented approaches.

1 Motivations

There are a number of applications that have a database "flavor," and yet are not well-addressed by conventional database management systems. Examples of such applications are

1. Computer-aided design and manufacturing systems,
2. Scientific databases, often involving feature detection and extraction, such as studies involving chemical structures (e.g., the human genome), or analysis of satellite data.

In addition to the traditional requirements of databases (such as integrity, sharing and recovery), these new applications pose demands that are not answered by conventional DBMS, such as the following:

- *The need to deal with complex structures and recursively defined objects.* For example, a VLSI CAD system typically allows the definitions of "cells," which are designs having other

SIGMOD RECORD, Vol. 19, No. 4, December 1990 75

Deductive Databases

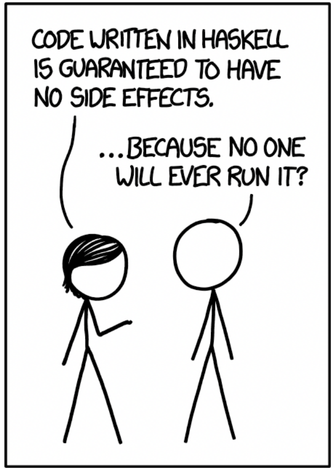
© Conexus AI, Inc. All Rights Reserved.

- I claim deductive databases are not popular for the same reason logic programming is not popular.

- A deduction engine is like a genie, it gives the programmer whatever they ask for, and can be hard to control.
  - Asking for long life? Don't forget to specify long health…

- Logic programming is "structured editing" for data.

- Also, some logics do not specify a unique result, or even a unique row count.
  - One reason RDF/OWL can be tough to use to integrate data



Deductive Databases – the bad

© Conexus AI, Inc. All Rights Reserved.

- The same traits that make logic programming tough in general make it good for manipulating ontologies.
    - if there were any place to want a genie, it would be to establish a "theory of being"
    - if there were any place to want structured editing, it would be to "preserve being"
    - We must still choose our logic carefully, which I'll talk about next

- Ontologies are often sparse, making them suited to graph databases

- Ontologies often overlap, making them suited to category-theoretic formal methods (ologs), including bi-directional transformation

Going forward, an ontology is an "expert system" and logic programming is "generative AI". I will show how to generate ontologies using logic.

https://github.com/categoricaldata/hydra

http://ontologica.org

Ontologies – the good

© Conexus AI, Inc. All Rights Reserved.

Symbolic Generative AI

Ryan Wisnesky | Conexus AI | ryan@conexus.com | Presented at Texas Data Day 2024

Recall: ontologies are expert systems

Outline

Claim 1

Data integration, properly (i.e., rigorously, mathematically) understood, is (deterministically and universally) generative.

Claim 2

Many expert systems/ontologies (collections of logical rules) are data integration systems in disguise.

Therefore: Many expert systems/ontologies are generative AIs.

Speculation: Generative symbolic AI could be more useful than generative stochastic AI

© Conexus AI, Inc. All Rights Reserved.

In **artificial intelligence**, an **expert system** is a computer system emulating the decision-making ability of a human expert.[1]

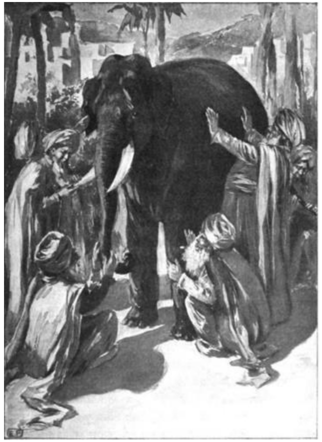
Expert systems are designed to solve complex problems by **reasoning** through bodies of knowledge, represented mainly as if-then rules rather than through conventional **procedural code.**

The first expert systems were created in the 1970s and then proliferated in the 1980s. Expert systems were among the first truly successful forms of **artificial intelligence (AI)** software.

An expert system is divided into two subsystems: the **inference engine** and the **knowledge base**.

- The knowledge base represents facts and rules.
- The inference engine applies the rules to the known facts to deduce new facts.

Recall: ontologies are expert systems



Expert Systems

© Conexus AI, Inc. All Rights Reserved.

There are mainly two modes for an inference engine: ***forward chaining*** and ***backward chaining***.

They differ by whether the inference engine is driven by the antecedent (left hand side) or the consequent (right hand side) of the rule.

In forward chaining the antecedent fires and asserts the consequent. For example, consider the following rule:

$$Man(x) \rightarrow Mortal(x)$$

In forward chaining, if $Man(Socrates)$ is added to the knowledge base, the rule fires and adds $Mortal(Socrates)$ to the knowledge base.

Forward Chaining

© Conexus AI, Inc. All Rights Reserved.

Forward chaining only determines a unique model for certain logics. cf "why it is mathematically impossible to use RDF/OWL for data integration"

```
Actor(x) and USGovernor(x)  →  Bodybuilder(x) or Austrian(x)
```

If you have an actor and US governor who is neither a bodybuilder nor Austrian, there is no canonical choice for whether to make them a body builder, Austrian, or both!

This is one reason why RDF/OWL can perform poorly at data integration. See https://arxiv.org/abs/2407.19095 for more

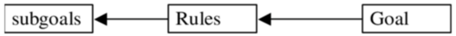
This is one reason spreadsheets are so useful in data integration. See https://arxiv.org/abs/2209.14457 for more

Forward Chaining Limitations

© Conexus AI, Inc. All Rights Reserved.

Backward Chaining

Backward chaining gives yes/no result



There are mainly two modes for an inference engine: forward chaining and backward chaining.

They differ by whether the inference engine is driven by the antecedent (left hand side) or the consequent (right hand side) of the rule.

In ***backward chaining*** the consequent fires and asserts the antecedent. For example, consider the following rule:

$$Man(x) \rightarrow Mortal(x)$$

In backward chaining, if $Mortal(Socrates)?$ is asked, then the inference engine asks if $Man(Socrates)$ is in the knowledge base.

© Conexus AI, Inc. All Rights Reserved.

1.1 A data exchange example 5

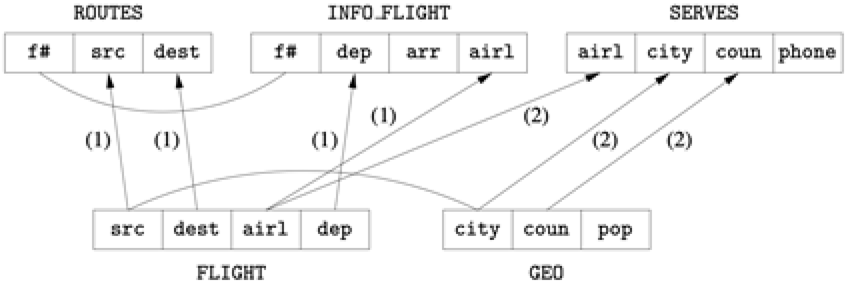


Figure 1.2 Schema mapping: a proper graphical representation

1.1 A data exchange example

(1) $\mathtt{FLIGHT(src,dest,airl,dep)} \longrightarrow$
$\exists \mathtt{f\#}\, \exists \mathtt{arr}\, \big(\ \mathtt{ROUTES(f\#,src,dest)}$
$\wedge\ \mathtt{INFO\_FLIGHT(f\#,dep,arr,airl)}\big)$

(2) $\mathtt{FLIGHT(city,dest,airl,dep)} \wedge \mathtt{GEO(city,country,popul)}$
$\longrightarrow \exists \mathtt{phone}\ \mathtt{SERVES(airl,city,country,phone)}$

(3) $\mathtt{FLIGHT(src,city,airl,dep)} \wedge \mathtt{GEO(city,country,popul)}$
$\longrightarrow \exists \mathtt{phone}\ \mathtt{SERVES(airl,city,country,phone)}$

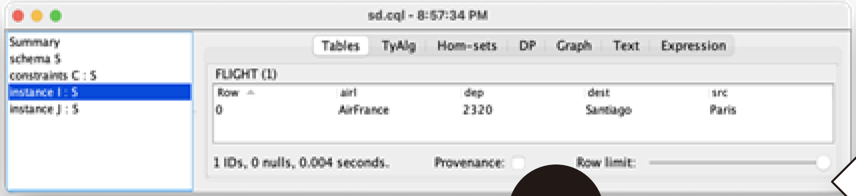
Figure 1.3 A schema mapping

Claim 1

Data integration, properly (i.e., rigorously, mathematically) understood, is (deterministically and universally) generative.

The “existential horn clauses” shown at left define a unique way to generate missing information from known information using forward chaining.

© Conexus AI, Inc. All Rights Reserved.



"Training Data"

\+

```
constraints C = literal : S {
    forall f:FLIGHT -> exists r:ROUTES i:INFO_FLIGHT
     where f.src=r.src f.dest=r.dest f.dep=i.dep f.airl=i.airl r."f#"=i."f#"

    forall f:FLIGHT g:GEO where f.src=g.city ->
    exists s:SERVES where s.airl=f.airl g.city=s.city g.country=s.country

    forall f:FLIGHT g:GEO where f.dest=g.city ->
    exists s:SERVES where s.airl=f.airl g.city=s.city g.country=s.country
}
```

"prompt"

1.1 A data exchange example 5

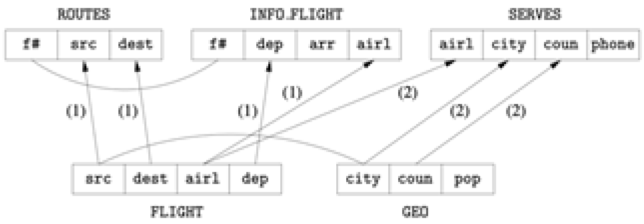
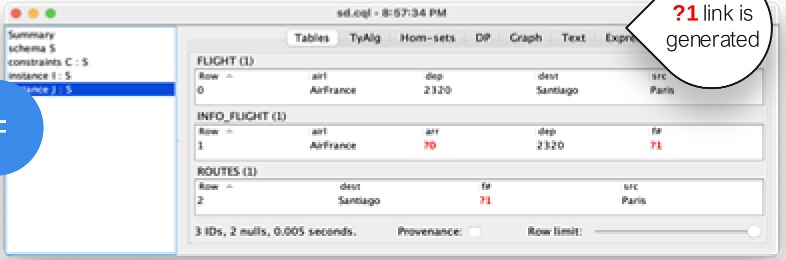


Figure 1.2 Schema mapping: a proper graphical representation

Claim 1: data integration, properly (i.e., rigorously, mathematically) understood, is (deterministically and universally) generative.

=



?1 link is generated

Example in Conexus CQL

© Conexus AI, Inc. All Rights Reserved.

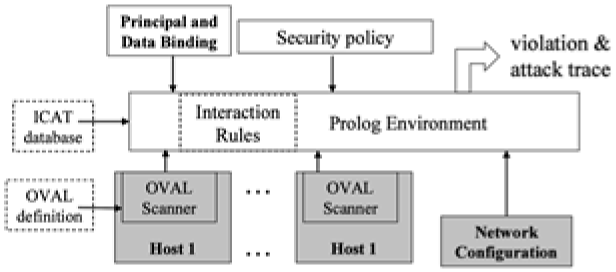
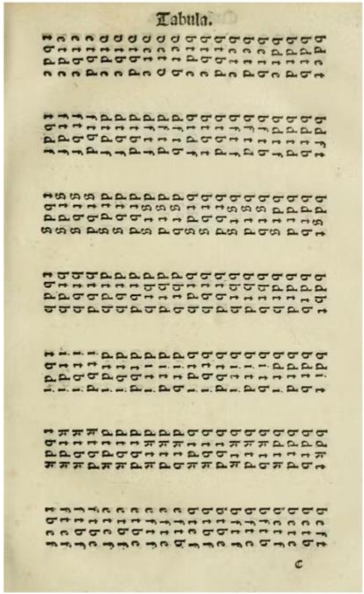


Figure 1: The MulVAL framework

```
execCode(Attacker, Host, Priv) :-
  vulExists(Host, VulID, Program),
  vulProperty(VulID, remoteExploit,
              privEscalation),
  networkService(Host, Program,
                 Protocol, Port, Priv),
  netAccess(Attacker, Host, Protocol, Port),
  malicious(Attacker).
```

Many expert systems/ontologies (collections of logical rules) are data integration systems in disguise.

“proof by example/definition”: Many expert systems/ontologies can be expressed in the language of ‘existential horn clauses’, the largest logic that generates unique forward chains. This is also the logic upon which modern data integration is based. In fact, modern data integration is based on this logic because it is the largest logic that generates unique forward chains.



Page of letter combinations from 16th-century edition of Ramon Llull’s *Ars Magna* (1517)

Claim 2

© Conexus AI, Inc. All Rights Reserved.

An aside on applied category theory

- Expert systems became popular in the 80s and data integration has been understood as logic since the 2010s, so why is “symbolic generativity” new?

- From a logic point of view, it is natural to “minimize generativity”.
- But from an algebraic view, it is natural to “maximize generativity”.

- In other words, realizing that data integration is “symbolically generative” requires a viewpoint change from one aspect of the “computational trinity” to another (logic to algebra).

https://en.wikipedia.org/wiki/Applied_category_theory

© Conexus AI, Inc. All Rights Reserved.

Ramifications of Generative Symbolic AI

- Generative symbolic AI is deterministic, but not predictable- arbitrarily complex behavior can be encoded using existential horn clauses.
- The future is formal - expert systems can be made even more useful thanks to discoveries in categorical algebra.
- AI systems will be composed of social-statistical-symbolic components, all generative in their own way.

Bonus claim: Knowledge graph merge and ontology merge are generative by definition

© Conexus AI, Inc. All Rights Reserved.

Thank you

Ryan Wisnesky
ryan@conexus.com
http://wisnesky.net

© Conexus AI, Inc. All Rights Reserved.