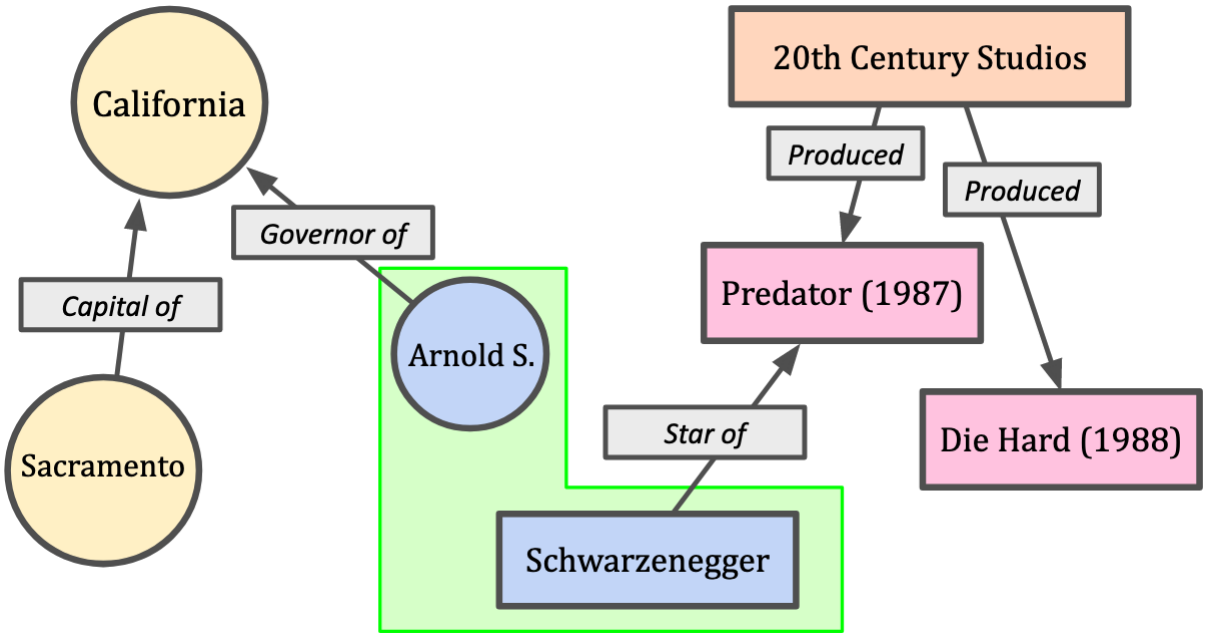
On Knowledge Graphs, Herbrand Universes and Reasoning

Logic has generally two strands, called propositional logic which have been historically employed by the Jains and Stoics and the more general predicate logic of Aristotle which incorporates semantics in general. Propositional logic merely considers sentence symbols and their truth values. For Predicate logic we also consider semantics. First let us delve into knowledge graphs and then we can consider extensions to it using Predicate logic.

A knowledge graph incorporates ontologies as it's vertex set and predicates in it's edge set which are used to represent the predicate(ontology_1, ontology_2) relationships. A diagram to illlustrate this is as follows.



These knowledge graphs can be constructed by parsing copula of texts using NLP dependency parsing and from each sentence extracting the {predicate, subject, object} triple out. Such an approach is described here https://arxiv.org/abs/1811.05768.

Knowledge graphs are restricted to binary predicates. There are a number of ways that it can be used to perform reasoning, and a quick chatGPT query yields the following.

🧠 1. Symbolic (Logic-Based) Reasoning

These methods rely on formal logic and ontologies to derive new facts based on rules and axioms.

A. Ontology-Based Reasoning (using OWL, RDF(S))

- **Inferred triples** are derived using **description logic**.
- Tools like **OWL reasoners** (e.g., HermiT, Pellet, Fact++) can infer:
  - Class hierarchies (e.g., If `Cat ⊆ Mammal` and `Tom` is a `Cat`, then `Tom` is a `Mammal`)
  - Property inheritance
  - Consistency checks (e.g., conflicting types)
  - Type inference (instance classification)

B. Rule-Based Reasoning (using SWRL, SPARQL rules)

- You define **if-then rules** like:

  mathematica
  CopyEdit
  `If person X hasParent Y and Y hasSibling Z, then X hasUncleOrAunt Z`

- Useful for business logic or domain-specific inference.

🔢 2. Statistical (Machine Learning-Based) Reasoning

These methods infer new links or facts by learning patterns from existing data.

A. Link Prediction / Knowledge Graph Embeddings

- Use models like **TransE, DistMult, ComplEx, RotatE** to embed entities and relations into vector spaces.
- The model learns to score triplets `(head, relation, tail)` — if the score is high, the triplet is likely true.
- Enables discovery of **missing links**: e.g., if `(Albert_Einstein, bornIn, ?)` is missing, embeddings may suggest `Ulm`.

B. Graph Neural Networks (GNNs)

- GNNs can propagate and aggregate information over the graph structure.
- Useful for entity classification, link prediction, and reasoning with **context-aware** embeddings.

🔄 3. Hybrid Reasoning (Symbolic + Statistical)

- Combines logical rules with learned patterns.

- Example: **Neuro-Symbolic AI** where a neural network suggests candidates and logic validates them.
- Tools: **DeepProbLog**, **Logic Tensor Networks**, **Probabilistic Soft Logic (PSL)**.

🧪 4. Probabilistic Reasoning

- Adds uncertainty to the graph, allowing reasoning under noise or partial information.
- Example tools:
  - **Markov Logic Networks (MLNs)**: merge first-order logic with probabilistic graphical models.
  - **Bayesian networks**: can model conditional dependencies between entities.

📊 5. Query-Based Reasoning

- Languages like **SPARQL** allow for complex queries that express reasoning over paths, filters, and constraints.
- For example:

  sparql
  CopyEdit

  ```
  SELECT ?ancestor WHERE {
    ?person :hasParent+ ?ancestor .
  }
  ```

  This finds all ancestors using **transitive closure** over `hasParent`.

Summary Table

Method	Type	Tools/Models	Use Cases
Ontology reasoning	Symbolic	OWL, RDF(S), Pellet	Inference, validation
Rule-based reasoning	Symbolic	SWRL, SPARQL Rules	Business logic
Embeddings	Statistical	TransE, RotatE, ComplEx	Link prediction
GNNs	Statistical	R-GCN, GAT	Contextual reasoning
Probabilistic	Hybrid	MLNs, PSL	Uncertainty modeling
Query reasoning	Declarative	SPARQL	Graph traversal

Next we move onto a defining what a Herbrand Universe is and how it may be more advantageous over Knowledge Graphs.

In Predicate Logic, we have something called first order logic which incorporates Predicates, Function symbols, Quantifier symbols, Constant symbols and Variables, along with the sentence connnective symbols from Propositional Logic. You have to define a model on a language of first order logic to define if a sentence is valid or not with respect to that model. A **Herbrand structure** is a special kind of model used to reason about logic formulas **syntactically** rather than semantically. Instead of interpreting functions and constants with real-world meanings (like numbers or people), it interprets them **using the symbols themselves**. Also we have the following theorem - Let Σ be a set of first-order sentences (usually in Skolem form). If Σ is satisfiable, then it has a **Herbrand model**, i.e., a model whose domain is the Herbrand universe of Σ, and where function symbols are interpreted syntactically. This means that we can under certain assumptions use Herbrand Structures in a valid way in first order logic.

Now for our application, we don't need function symbols at all.. There only will be variables, constants, prediacates, quantifiers and the = symbol. We may also incorporate $\subseteq$ which is just a binary predicate. Since we have only a finite number of constant symbols, we need to consider only finite models. This greatly simplifies the process of making deductions. There will be a set of axioms and the data structure for the Herbrand Universe itself from which we can make deductions by applying syntactic rules from proof theory or natural deduction. For instance if we have proven the theorem ∀x ∀y P_1(x, y) -> P_2 y and also ∀ x Px, then we have ∀y P_2 y.

Knowledge Graphs are Herbrand Structures restricted to binary predicate instead of general n-ary Predicates. It also insists on a specific data structure, a generalization on graphs but we have not yet imposed a specific data structure for storing Herbrand Universes in general, but a tree and a disjoint set data structure to store the Predicate trees and as a symbol lookup table for equivalence classes respectively should suffice.

In terms of computational complexity, Herbrand universes should be decidable and run in polynomial time if we restrict ourselves to Horn clauses, that is, clauses that contain the ∀ quantifier and of the form $A1 \land A2 \land \cdots \land An \rightarrow B$ or equivalently $\neg A1 \lor \neg A2 \lor \cdots \lor \neg An \lor B$. This is indeed employed in Datalog (a subset of Prolog with no function symbols and finite domains) and deductive databases such as Flix.