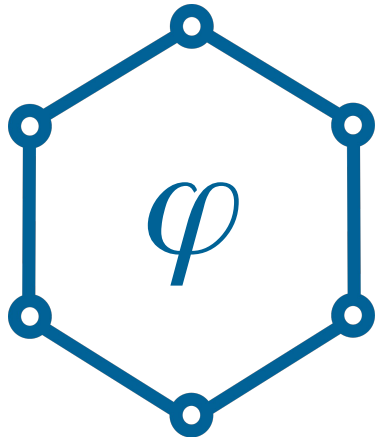
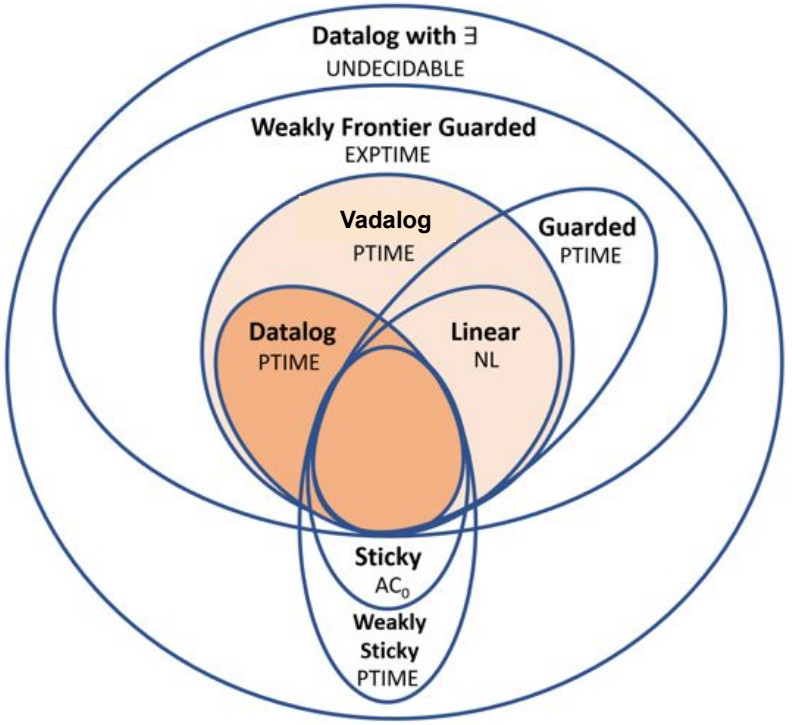


Logical Knowledge in KGs

Vadalog – Syntax

Vadalog Datalog +/– Ontological Reasoning Languages

- **Declarative** logic language
  - Describe what
  - Not how
- **Concise** logical statements
  - Domain knowledge (what the data does not say)
  - Queries to be answered
- **Trade-off** between
  - **Expressivity**
  - **Computational complexity**
  - At the price of very mild syntactic restrictions it guarantees PTIME query answering

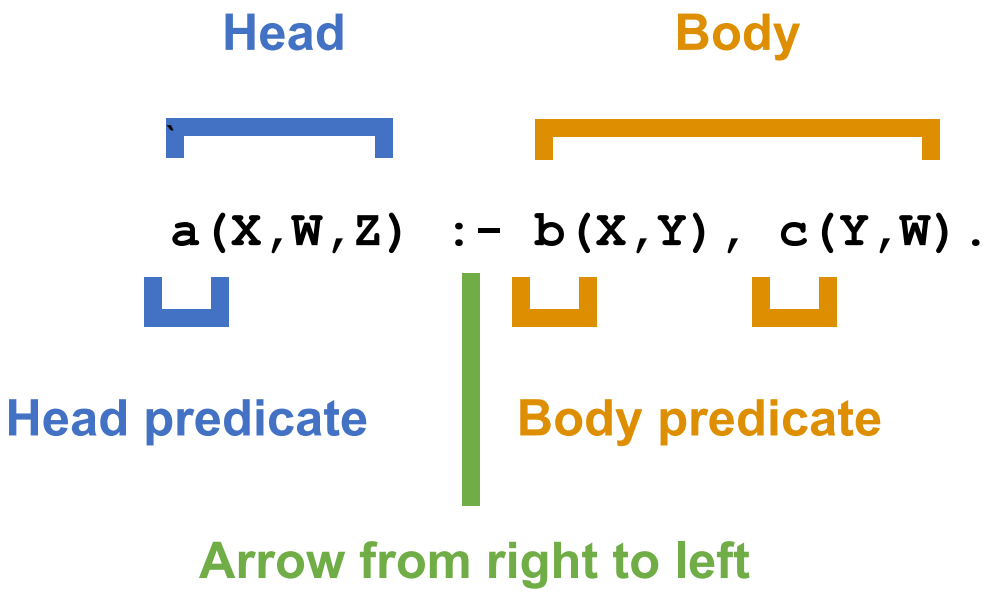


Syntactic containment of Datalog± languages

Bellomarini, L., Gottlob, G., & Sallinger, E. (2020). The Vadalog System: Datalog-based Reasoning for Knowledge Graphs. *PVLDB* 12, 9 (2018), 975-987.

Rules

Head and Body Atoms



- **Generates facts** for atom **a**, given facts for atom b and c
- **:-** can be read as “if” or “given”
- Predicates are lower case
- Don’t forget the period at the end of each rule!

Variables

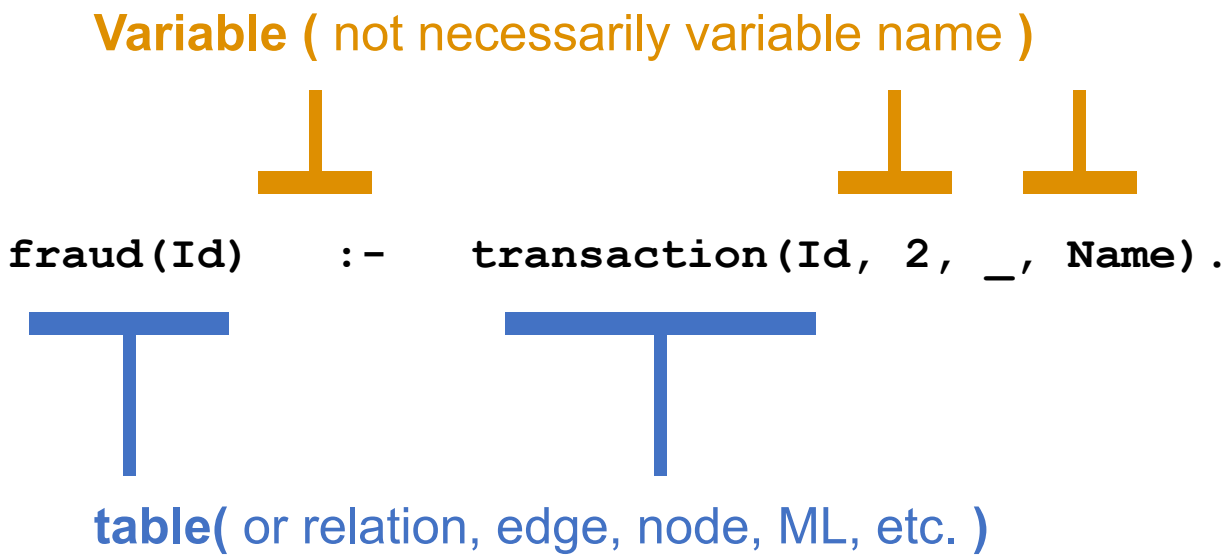
Different from Imperative Languages

- **Imperative languages**
  - Essentially memory locations.
  - May change over time.
- **Algebra and physics**
  - Represent a concrete value
  - Once we replace variables with concrete actual values, we have relations between concrete arithmetic expressions.
  - Similar to pronouns in natural language
- **Vadalog**
  - Like variables in algebra and in first-order logic

Variables

Different from Imperative Languages

- Variables represent values of the specific position in the predicate
- Use the anonymous variable “_” for arguments that don’t matter
- Variables live and die within each rule
- Variables must be capital



Rules

Example

```
fraud(Id) :- transaction(Id,   _  )
```

Rules

Example

Rules

Example

Rules

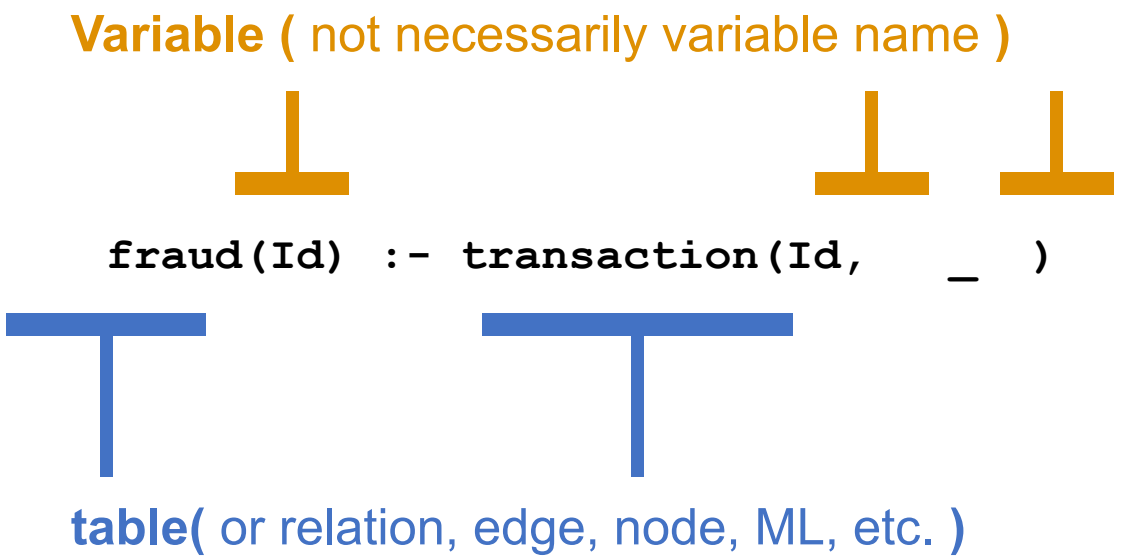
Example

```
fraud(Id) :- transaction(Id,   _   )
```

table(or relation, edge, node, ML, etc. **)**

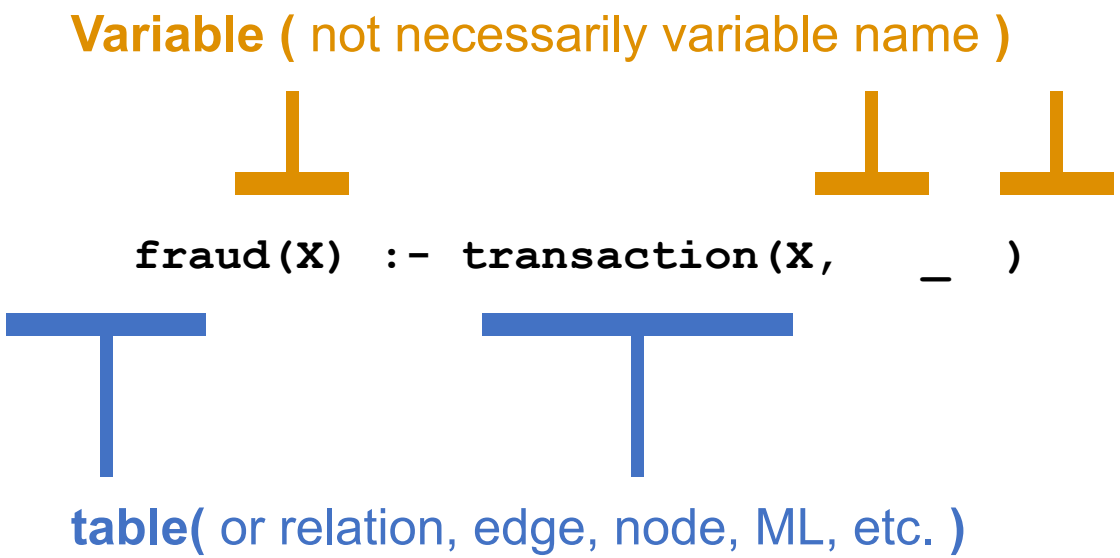
Rules

Example



Rules

Example



Rules

Linear Rules

- Rules with one single atom in the body

```
fraud(Id) :- transaction(Id,   _   )
```

Rules

Join Rules

- Join rules have multiple atoms in the body
- Joins occur when the same variable is used in multiple atoms

```
project(DepId,Id) :- employee(Id), department(DepId,Id).
```

Rules

Negation

- Negation is a prefix modifier that negates the truth value for an atom
- Not employee(X) holds if X is not an employee

```
safeProjects(X,P) :- project(X,P), not contractor(P).
```

- Here we define safe projects as those not run by contractors

Recursive Rules

The simplest form of recursion is that in which the head of a rule also appears in the body

```
path(X,Y) :- edge(X,Y).
path(X,Z) :- path(X,Y), edge(Y,Z).
```

Recursive Rules

The simplest form of recursion is that in which the head of a rule also appears in the body

```
own(X,Y) :- dir_own(X,Y).
own(X,Z) :- own(X,Y), dir_own(Y,Z).
```

Rules

Linear, Join and Recursion

Simple copy

```
city(City) :- airport(IATA,City).
```

Similar to SQL

```
SELECT City INTO city FROM airport
```

Joining Data

```
dest(D) :- flight(L,D,A), airport(L,"London").
```

Similar to SQL

```
SELECT f.Destination INTO dest
FROM flight f, airport a
WHERE f.Origin = a.IATA AND a.City = "London"
```

Becomes very lengthy...

Recursion

```
connection(X,Y) :- flight(X,Y,_).
connection(X,Z) :- flight(X,Y,_), connection(Y,Z).
```

Data Types

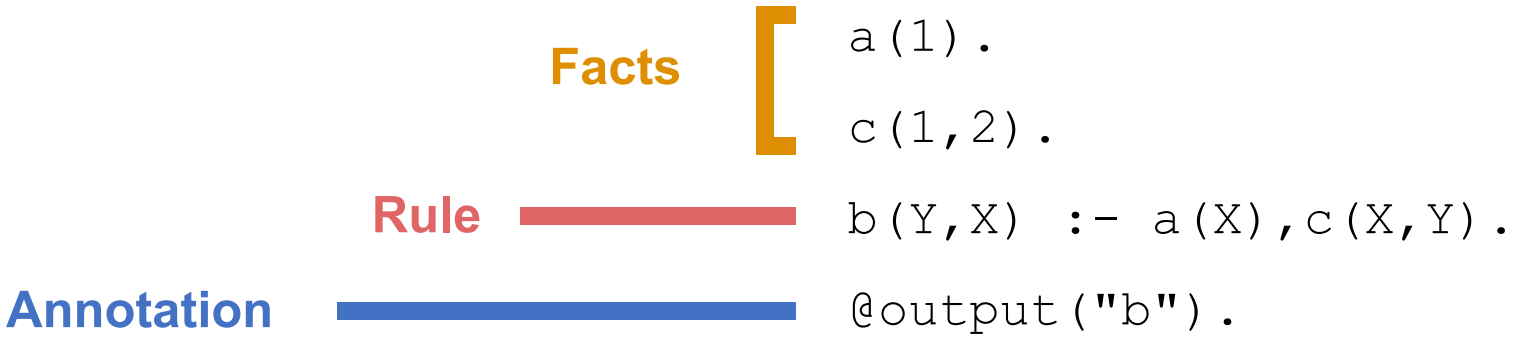
Single-fact operators

Data type	Operators
all	`==`, `>`, `<`, `>=` , `<=`, `<>`, `!=`
string / list	`substring`, `contains`, `starts_with`, `ends_with`, `concat`, `index_of` (string only), `string_length`
integer	(monadic) `-`, `*`, `/`, `+`, `-`, `( )`
double	(monadic) `-`, `*`, `/`, `+`, `-`, `( )`
Boolean	`&&` (and), `||` (or), `not`, `( )` for associativity
set	`|` (union), `&` (intersection), `( )` for associativity

- == : equals to
- > : greater than
- < : less than
- >= : greater or equal
- <= : less or equal
- <> : not equal
- != : not equal

Programs Or Ontologies

A Vadalog program (or ontology) is a set of rules and facts



It specifies that the facts for b must be returned in output.

Interacting with data — Annotations

Input Data as Facts

```
flight("VIE","LHR","BA").
```

Input data from source.

```
@bind("flight","csv","./disk/data","res.csv").
```

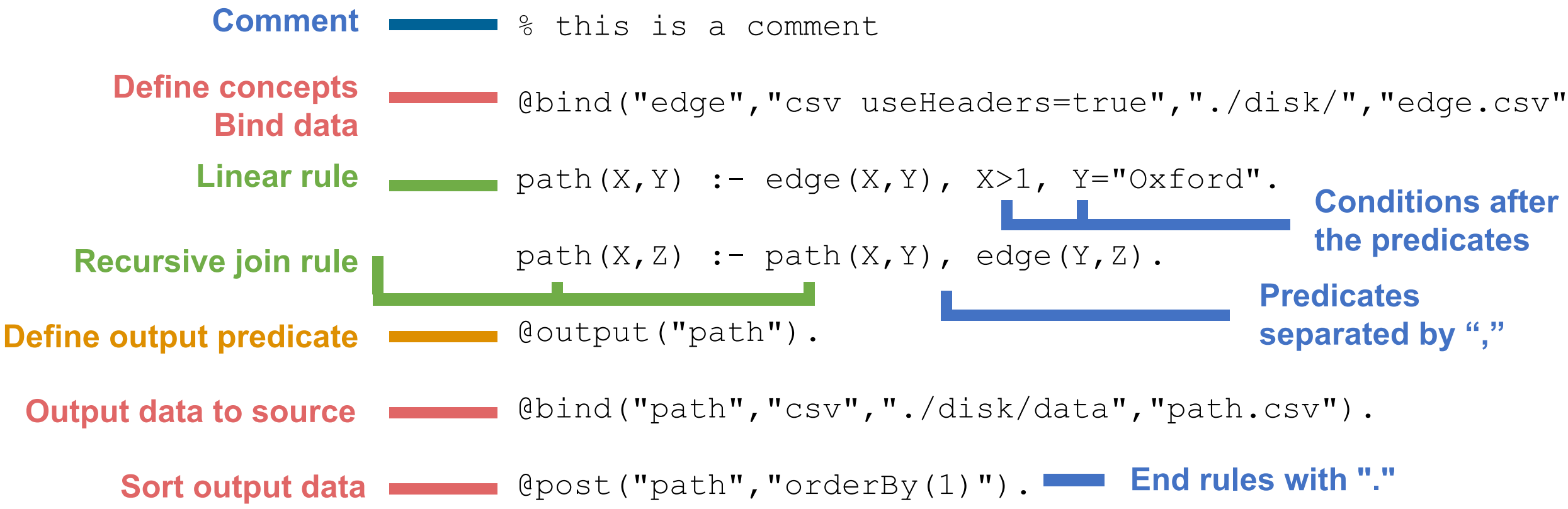
Output Data Directly

```
@output("connection").
```

Output data to source.

```
@output("connection").
@bind("connection","csv","./disk/data","res.csv").
```

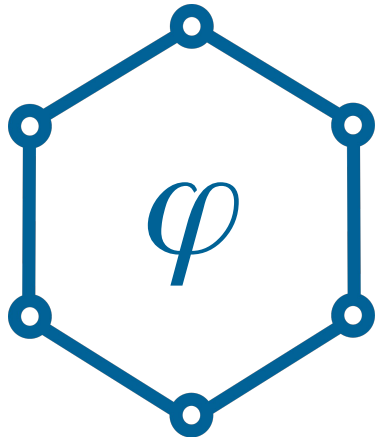
Quick Reference Syntax and Interacting with Data



```
5  %Bind concepts to multiple data sources
6  @bind("order", "csv", "s3a://prometheux-data/","orders.csv").
7  @bind("product","neo4j","neo4j", "(:Product)").
8  @bind("customer", "postgresql","postgres", "customer").
9
10 %Join customers, orders and products across different sources
11 customer_purchase(CustomerId, Name, Product) :-
12     customer(CustomerId, Name, Surname, Email),
13     order(CustomerId, OrderId, ProductId),
14     product(Id,ProductId,Product).
15
16 @output("customer_purchase").
```

customer_purchase

CustomerId	Name	Product
9	Luca	Apple Watch Ultra
9	Luca	Xbox Series X
9	Luca	DJI Mavic 3



Logical Knowledge in KGs

Vadalog – Syntax