Data Modeling and Integration Using the Open Source Tool AQL

Peter Gates

Categorical Informatics Inc.

$$C_i$$

$$\Sigma \dashv \Delta \dashv \Pi$$

Category Theory, Applied?

What constitutes a successful application of category theory?

Follow the Money: Top 10 Companies in Market Cap

1. Apple -> $807 billion
2. Alphabet (Google) -> $677 billion
3. Microsoft -> &608 billion
4. Facebook -> $497 billion
5. Amazon -> $467 billion
6. Berkshire Hathaway -> $433 billion
7. Johnson & Johnson -> $385 billion
8. Exxon Mobil -> $353
9. JPMorgan Chase -> $350 billion
10. Bank of America -> $286 billion

Aggregated by sector

1. Tech -> $3.6 trillion
2. Financial -> $0.6 trillion
3. Retail -> $0.5 trillion
4. Health Care -> $0.4 trillion

Categorical Semantics of Schemas and Instances: Naive

Syntax

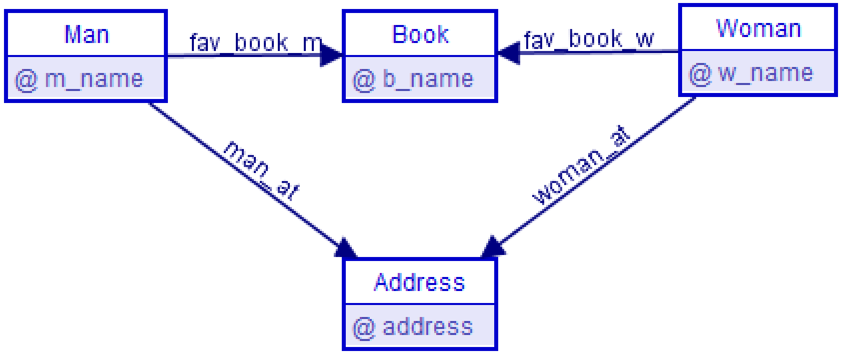
- A "graph" S we call a schema.
- A morphism of schemas S -> T.
- An S-instance
- A morphism of S-instances $I \rightarrow J$

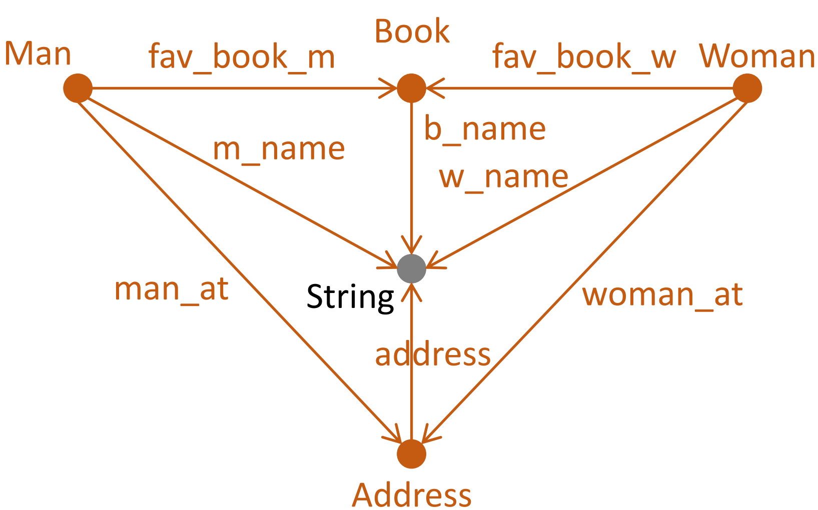
Semantics

- A presentation of a category $\mathcal{S}$
- A functor $\mathcal{S} \longrightarrow \mathcal{T}$.
- A functor $\mathcal{S} \longrightarrow \boldsymbol{Set}$
- A natural transformation $I \rightarrow J$

Intuitively one can think of an instance as a set of tables, one per node of S and each column an edge of S.

Schema Example






```
schema S = literal : Ty {
    entities
        Man Woman Address Book
    foreign_keys
        man_at : Man -> Address
        woman_at : Woman -> Address
        fav_book_m : Man -> Book
        fav_book_w : Woman -> Book
    attributes
        m_name : Man -> String
        w_name : Woman -> String
        b_name : Book -> String
        address : Address -> String}
```

Instance Example

Man

ID	m_name	fav_book_m	man_at
m0	**bob**	b0	m0.man_at
m1	**charlie**	b0	m1.man_at
m2	**frank**	b2	m2.man_at

Book

ID	b_name
b0	b0.b_name
b1	b1.b_name
b2	b2.b_name
b3	b3.b_name

Woman

ID	w_name	fav_book_w	woman_at
w0	**alice**	b1	w0.woman_at
w1	**doris**	b2	w1.woman_at
w2	**ellie**	b3	w2.woman_at

Address

ID	address
m0.man_at	m0.man_at.address
m1.man_at	m1.man_at.address
m2.man_at	m2.man_at.address
w0.woman_at	w0.woman_at.address
w1.woman_at	w1.woman_at.address
w2.woman_at	w2.woman_at.address

```
instance iSrc = literal : sSrc {
	generators
		m0 m1 m2 : Man
		w0 w1 w2 : Woman
		b0 b1 b2 b3 : Book
	multi_equations
		m_name -> {m0 bob, m1 charlie, m2 frank}
		w_name -> {w0 alice, w1 doris, w2 ellie}
		fav_book_m -> {m0 b0, m1 b0, m2 b2}
		fav_book_w -> {w0 b1, w1 b2, w2 b3}}
```

Immediate Insight:
Functorial Data Migration

A schema mapping $F: S \to T$ induces three data migration functors:

- $\Delta_F: T\text{-inst} \to S\text{-inst}$

$$\begin{tikzcd} S \arrow[r, "F"] \arrow[dr, swap, "\Delta_F(I) := I \circ F"] & T \arrow[d, "I"] \\ & \textbf{Set} \end{tikzcd}$$

- $\Pi_F: S\text{-inst} \to T\text{-inst}$ (right adjoint to Δ_F)

$$\forall I, J.\;\; S\text{-inst}\,(\Delta_F(I), J) \cong T\text{-inst}(I, \Pi_F(J))$$

- $\Sigma_F: S\text{-inst} \to T\text{-inst}$ (left adjoint to Δ_F)

$$\forall I, J.\;\; S\text{-inst}\,(J, \Delta_F(I)) \cong T\text{-inst}(\Sigma_F(J), I)$$

Two Challenges

1. There seem to be two different kinds of columns/edges in a schema:
   - Entity -> Entity (foreign keys).
   - Entity -> Data type (attributes).
   - Meaningless identifiers vs. **meaningful values**.
2. Although $\Sigma \dashv \Delta \dashv \Pi$ are central to the mathematics they don't always meet engineering requirements.

Solution to Challenge 1.
Schema as Extension of the Type Side

- Fix an arbitrary multi-sorted algebraic theory Ty to serve as an ambient *type-side* or "background theory".
- We say sorts of Ty are types and the morphisms are operations.
  - For Ty the theory of strings, one sort S, one binary operation concat: $S * S \to S$, "":1 → S and for all ascii characters x, "x":1 → S
  - Concat and "" satisfy the monoid laws.
- A schema is an algebraic theory that extends Ty
  - New sorts which we call entities.
  - Unary function symbols between entities which we call foreign keys.
  - Unary function symbols from entities to types which we call attributes.
- Categorically this can be interpreted as an "algebraic profunctor"*
  - $S_0 : S_e \nrightarrow Ty$ ($S_e^{op} \times Ty \to \boldsymbol{Set}$) where S_e is the *entity category*.
  - To be elaborated during our discussion of collages.

*http://math.mit.edu/~dspivak/informatics/CatData.pdf

Category Theory, Applied!
Challenge 1

- We have placed categorical semantics in a computational framework.
- We have anchored the abstract entity category to a meaningful type side.

Challenge 2.

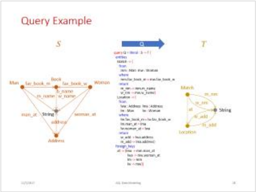
2. Although $\Sigma \dashv \Delta \dashv \Pi$ are central to the mathematics they don't always meet engineering requirements.

Partial Solution:
Constructions Useful to Database Engineers

- Data Migration -> Query; operation from a source schema S to a target schema T.
- Data Integration -> Merge; operation from a diagram $F\colon D \longrightarrow Sch$ to a target schema T.
- …

We have barely scratched the surface!

Queries are Profunctors



Given a query $Q: S \to T$ ($T^{op} \times S \to \boldsymbol{Set}$) define a "collage" schema as follows,

1. Define a schema that is the coproduct $S + T$.
2. For each target entity t create an new foreign key from t to each entity in its inverse image.
3. Add a "path equation" for each equational constraint between source entities in the inverse image of a target entity.
4. And similarly for foreign keys and attributes.

This defines a canonical cospan with the collage schema at the head and the source and target schemas on each arm.

Moving an instance from the source schema to the target schema can be implemented as a Π from source to collage followed by a Δ from the collage to the target.

Using Colimits for Data Integration

- Step 1: Integrate Schemas. E.g. given input schemas S_1, S_2, and overlap schema S, and mappings F_1, F_2:

$$S_1 \overset{F_1}{\leftarrow} S \overset{F_2}{\rightarrow} S_2$$

  we propose to use their pushout T as the integrated schema:

$$S_1 \overset{G_1}{\rightarrow} T \overset{G_2}{\leftarrow} S_2$$

- Step 2: Integrate Data. Given input S_1-instance I_1, S_2-instance I_2, overlap S-instance I, and row mappings $h_1 : \Sigma_{F_1}(I) \rightarrow I_1$ and $h_2 : \Sigma_{F_2}(I) \rightarrow I_2$, we propose to use the pushout of:

$$\Sigma_{G_1}(I_1) \xleftarrow{\Sigma_{G_1}(h_1)} (\Sigma_{G_1 \circ F_1}(I) = \Sigma_{G_2 \circ F_2}(I)) \xrightarrow{\Sigma_{G_2}(h_2)} \Sigma_{G_2}(I_2)$$

  as the integrated T-instance.

Brace Yourself

The following content contains material that may be distrubing!

Category Theory, Applied!

- Who might use your product?
- What is already available, i.e. your competition?
- How is what you are offering better?
- Prototype and validate with potential customers.
- Listen to feedback and invest in future product development accordingly.

Slogan: *If you can’t get people to part with their money it is just a hobby.*

Acknowledgements

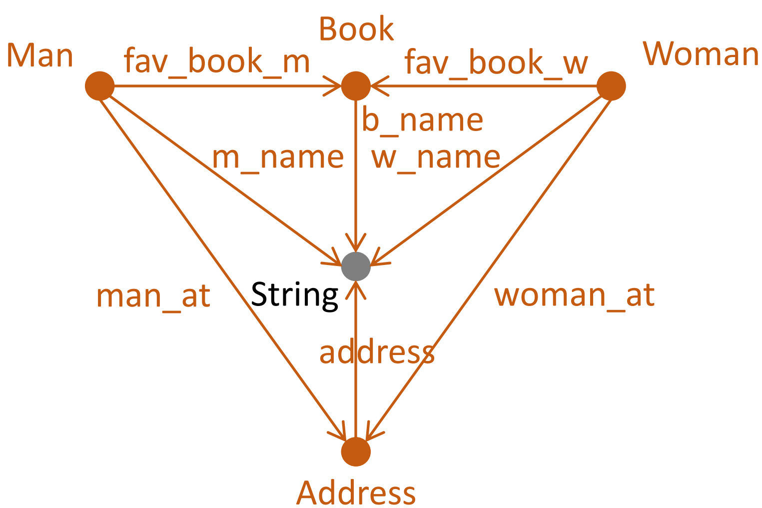
David Spivak
MIT Math Department

Ryan Wisnesky
Categorical Informatics Founder

Backup Slides

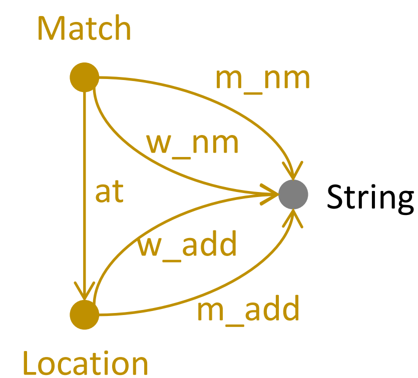
Query Example

S

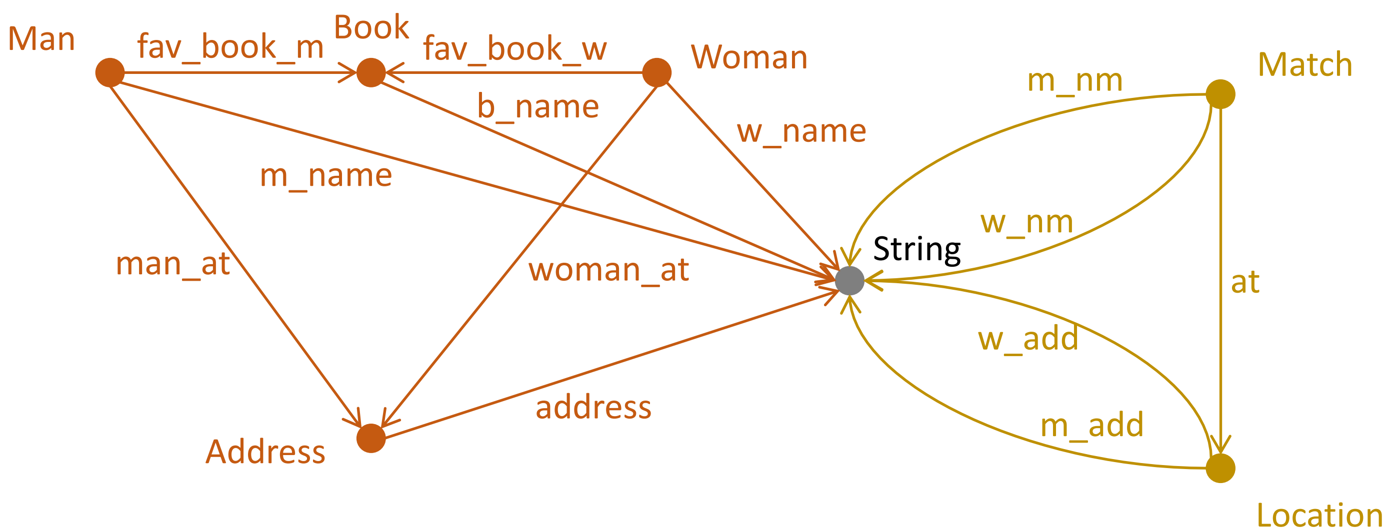


T

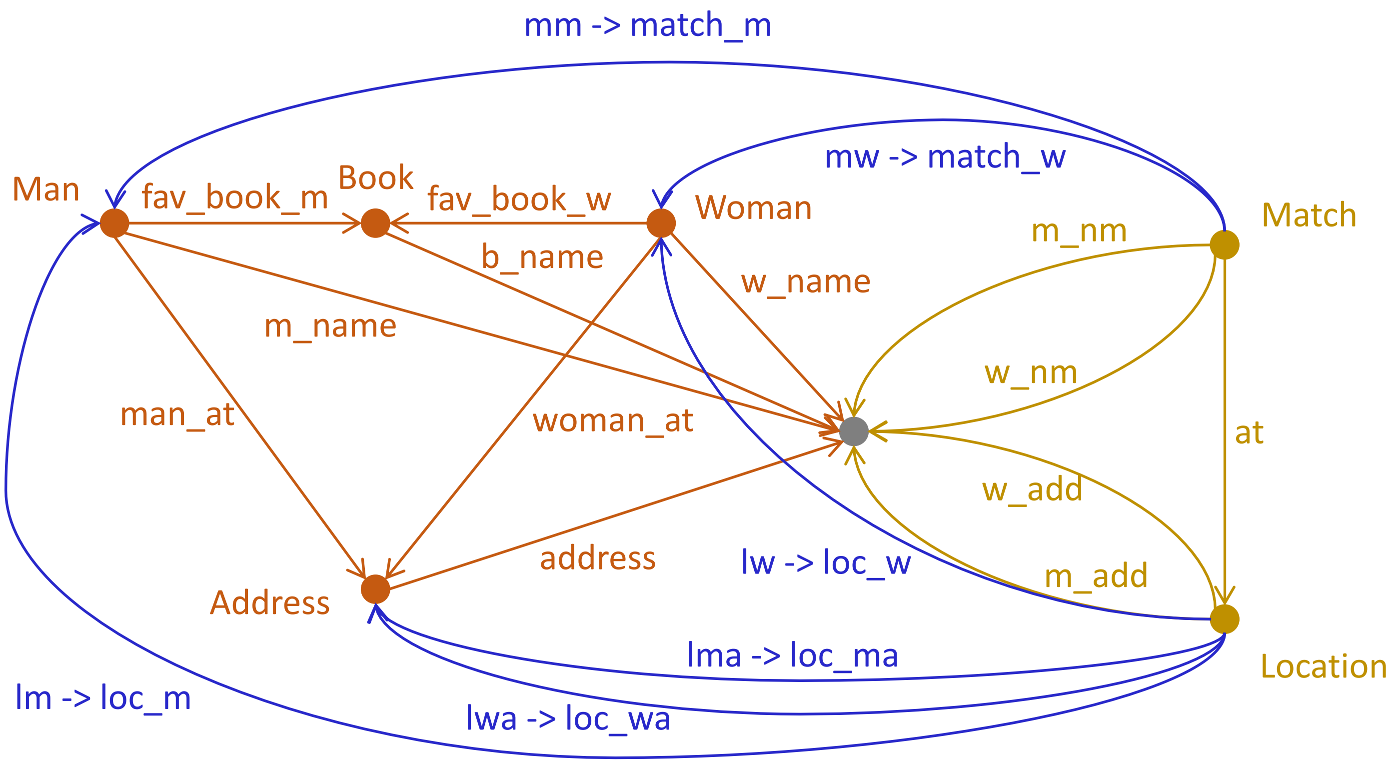
```
query Q = literal : S -> T {
 entities
  Match -> {
   from
    mm : Man  mw : Woman
   where
    mm.fav_book_m = mw.fav_book_w
   return
    m_nm -> mm.m_name
    w_nm -> mw.w_name}
  Location -> {
   from
    lwa : Address  lma : Address
    lm : Man         lw : Woman
   where
    lm.fav_book_m = lw.fav_book_w
    lm.man_at = lma
    lw.woman_at = lwa
   return
    w_add -> lwa.address
    m_add -> lma.address}
 foreign_keys
  at -> {lma -> mm.man_at
          lwa -> mw.woman_at
          lm -> mm
          lw -> mw}}
```



1. Take the Coproduct (Source + Target)/Type-side (String)

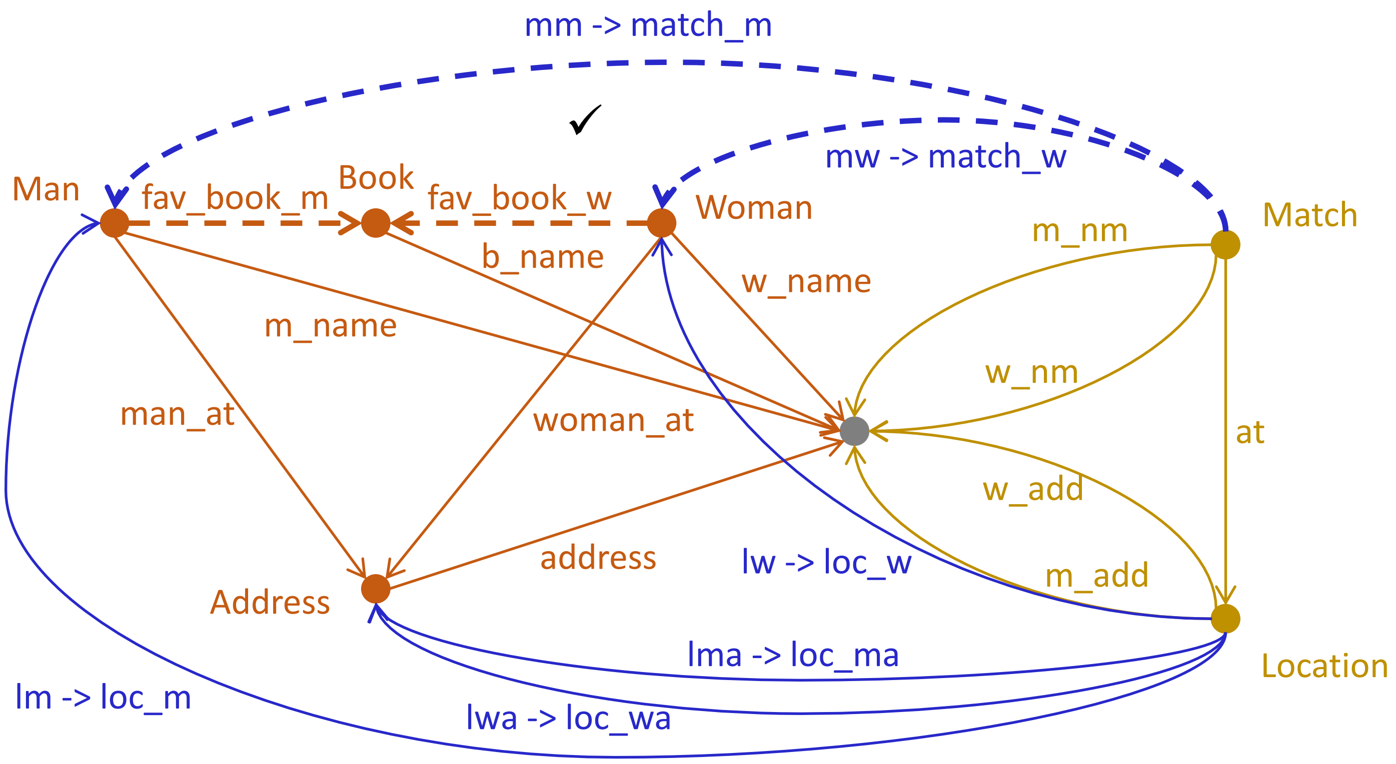


2. Add Foreign Key/Arrow for each Generator Variable



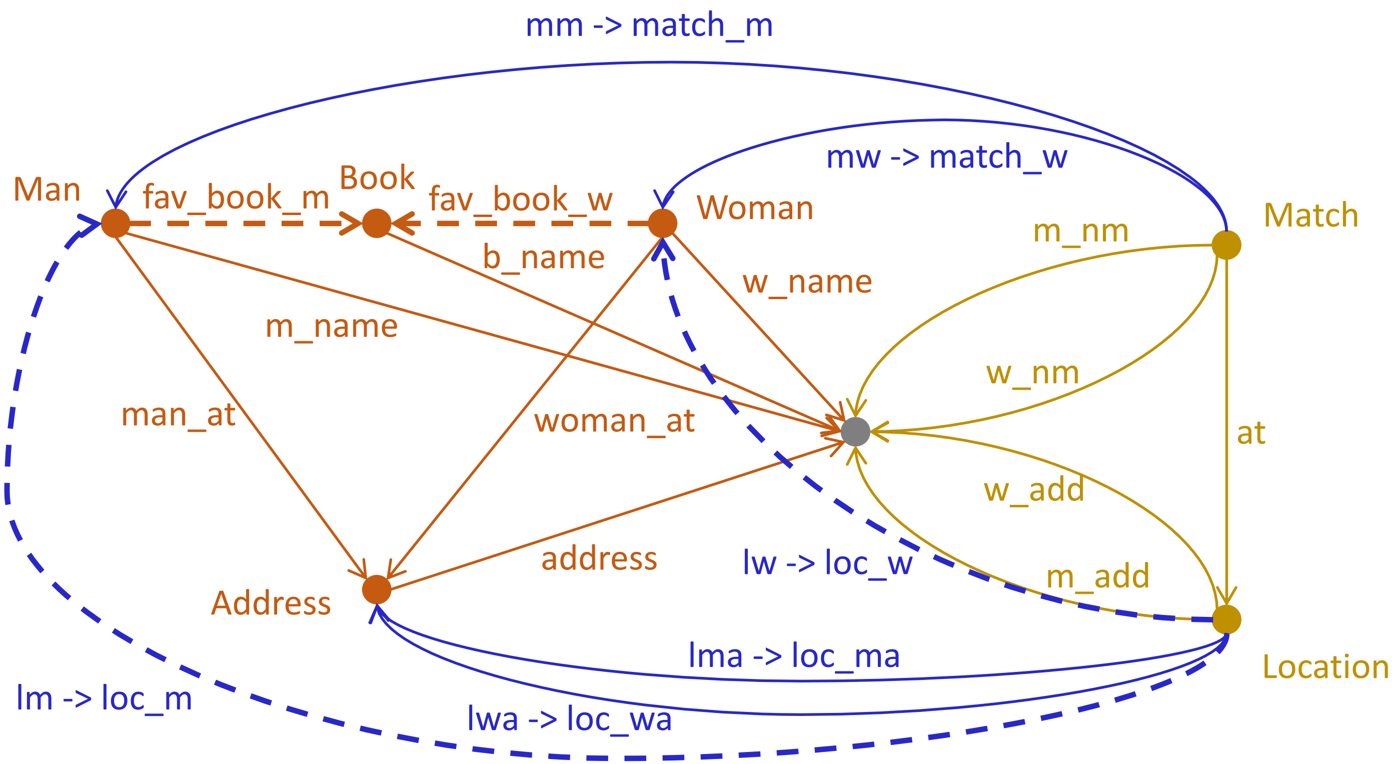
3. Add a Path Equation for each Equation in a Where Clause:

mm.fav_book_m = mw.fav_book_w



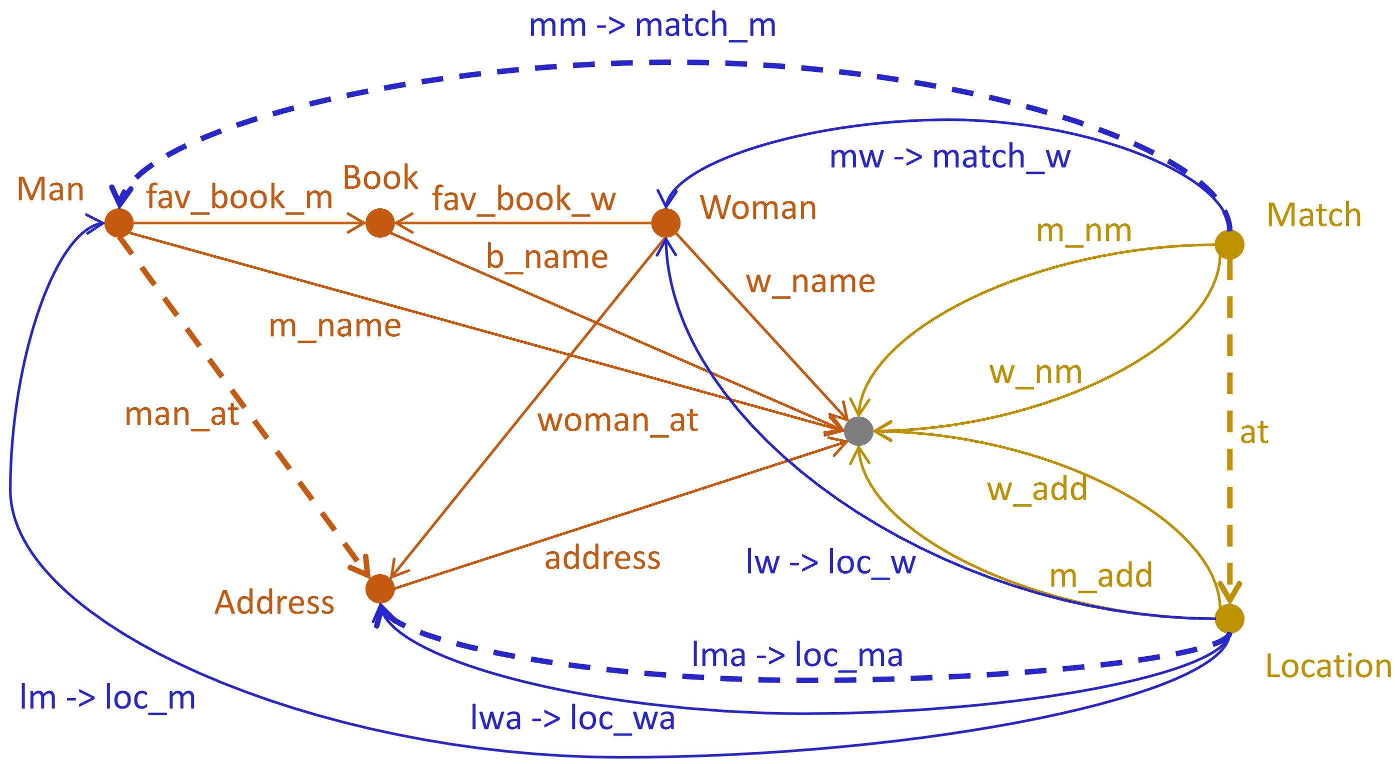
3. Add a Path Equation for each Equation in a Where Clause:

lm.fav_book_m = lw.fav_book_w



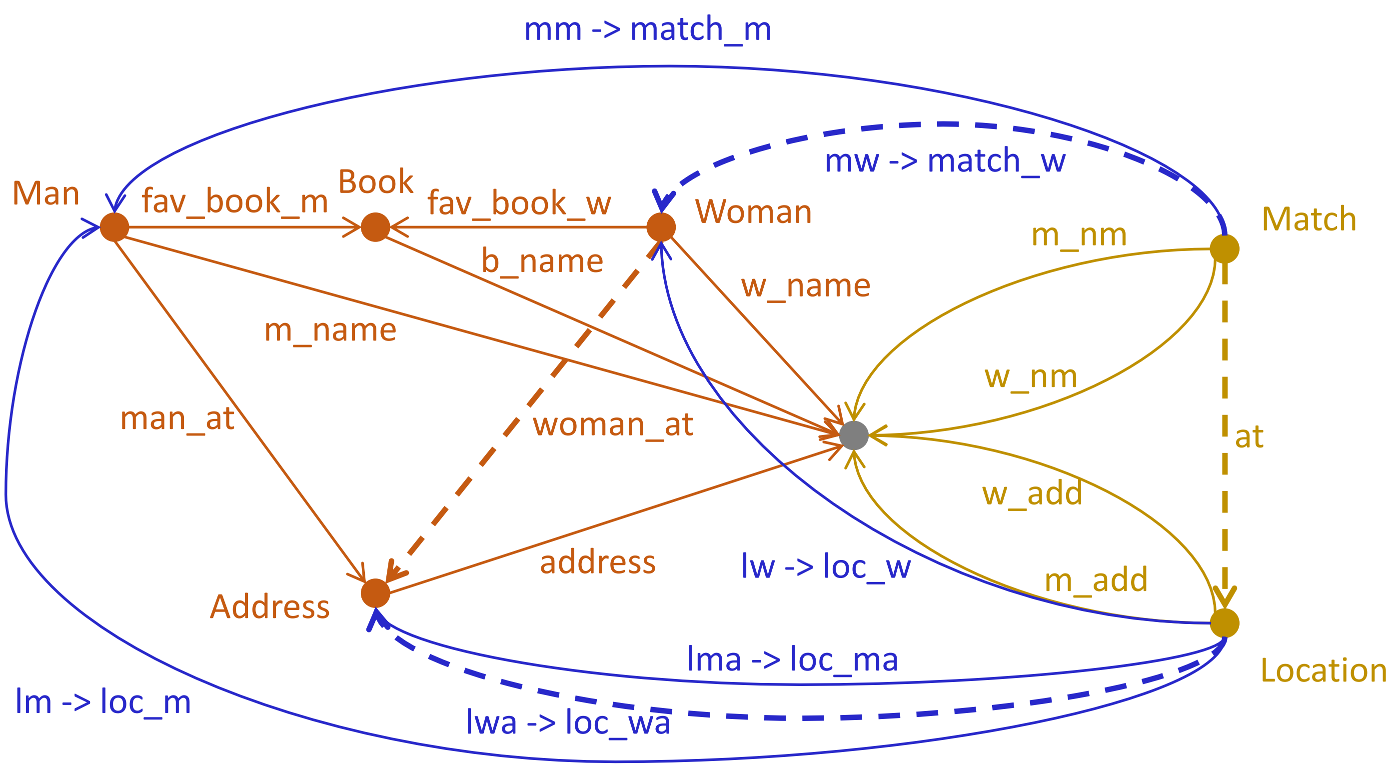
4. Add a Path Equation for each Foreign Key Assignment:

at.loc_ma = match_m.man_at



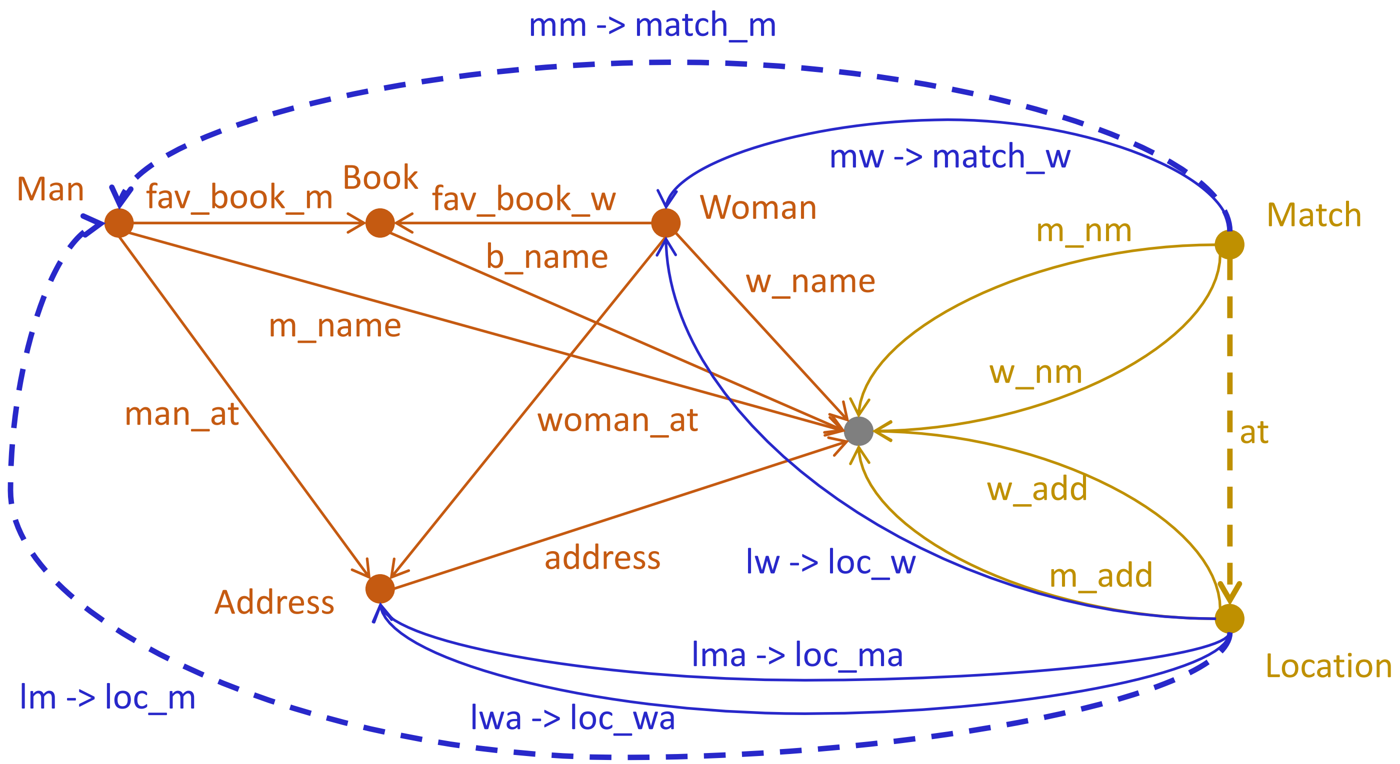
4. Add a Path Equation for each Foreign Key Assignment

at.loc_wa = match_w.woman_at



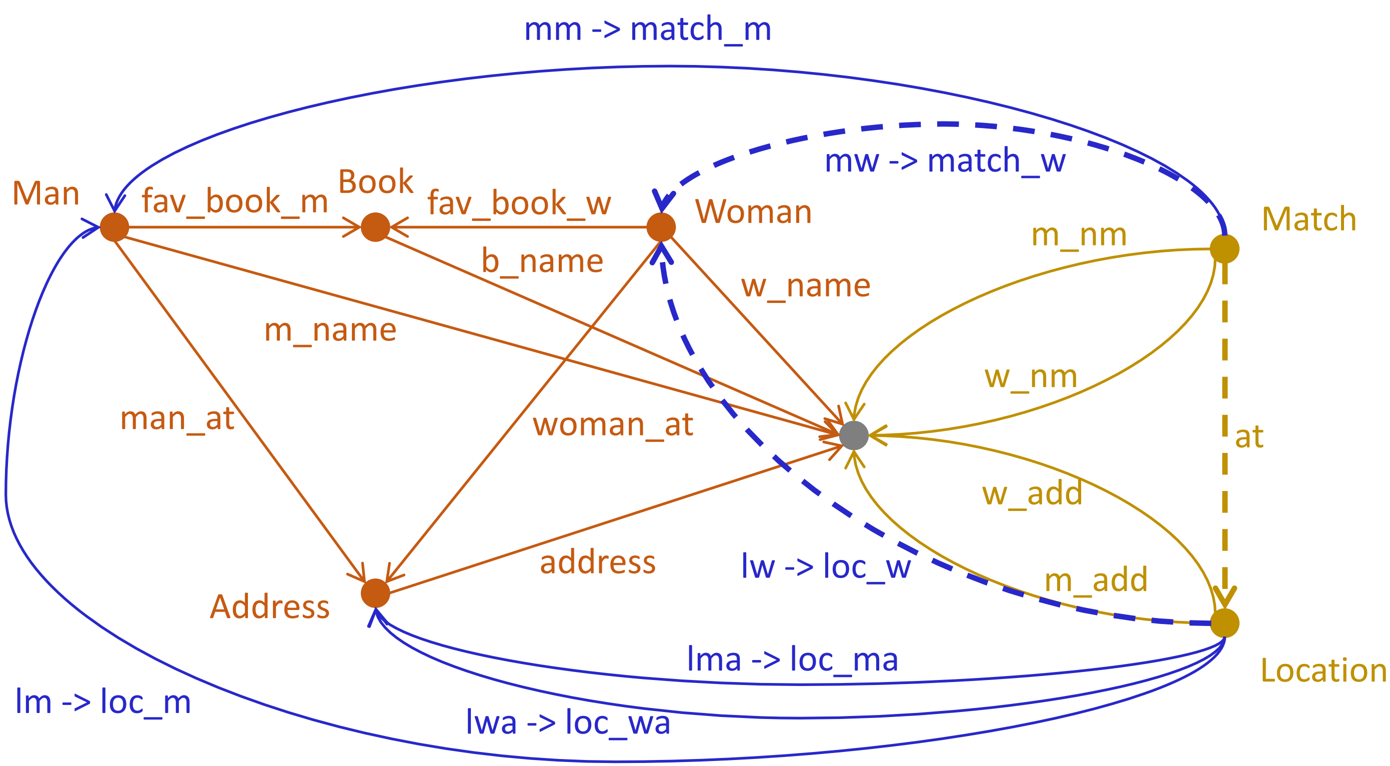
4. Add a Path Equation for each Foreign Key Assignment

at.loc_m = match_m



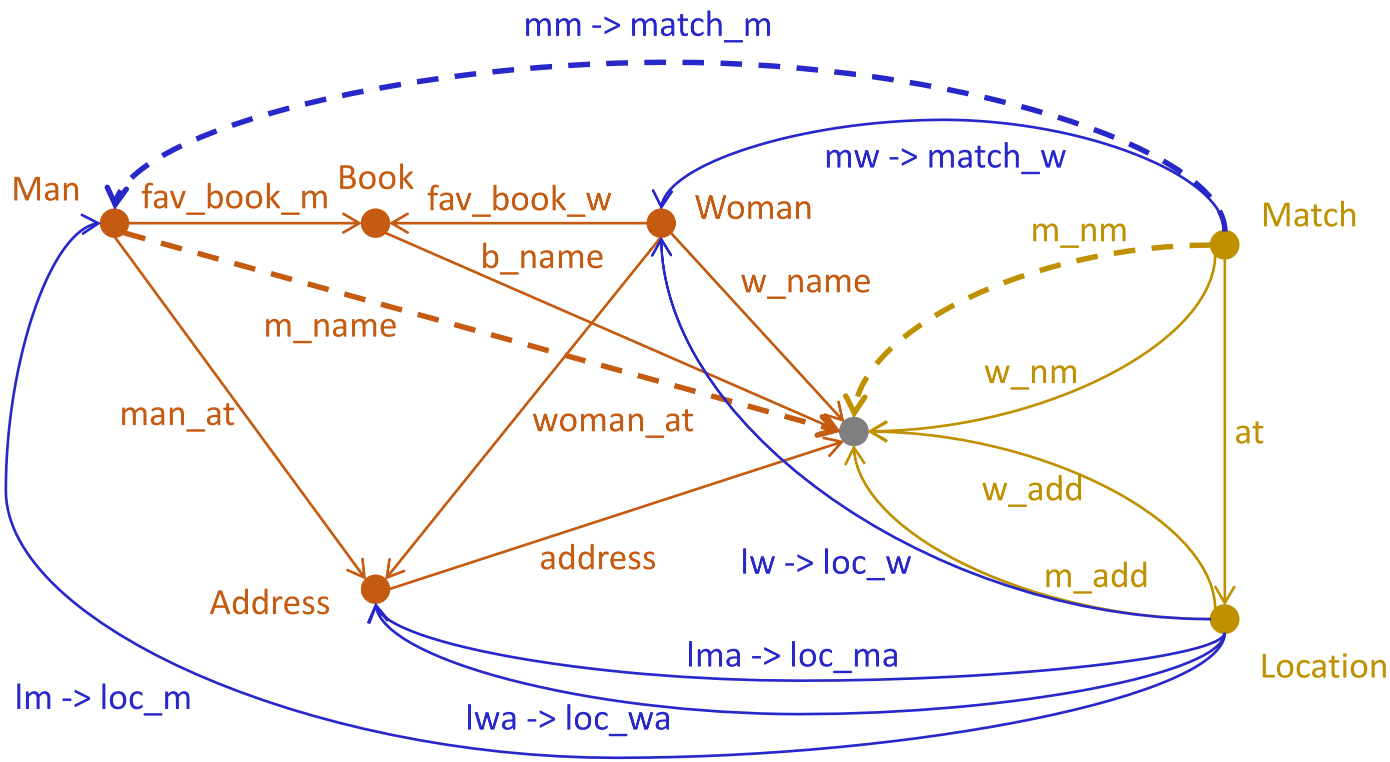
4. Add a Path Equation for each Foreign Key Assignment

at.loc_w = match_w



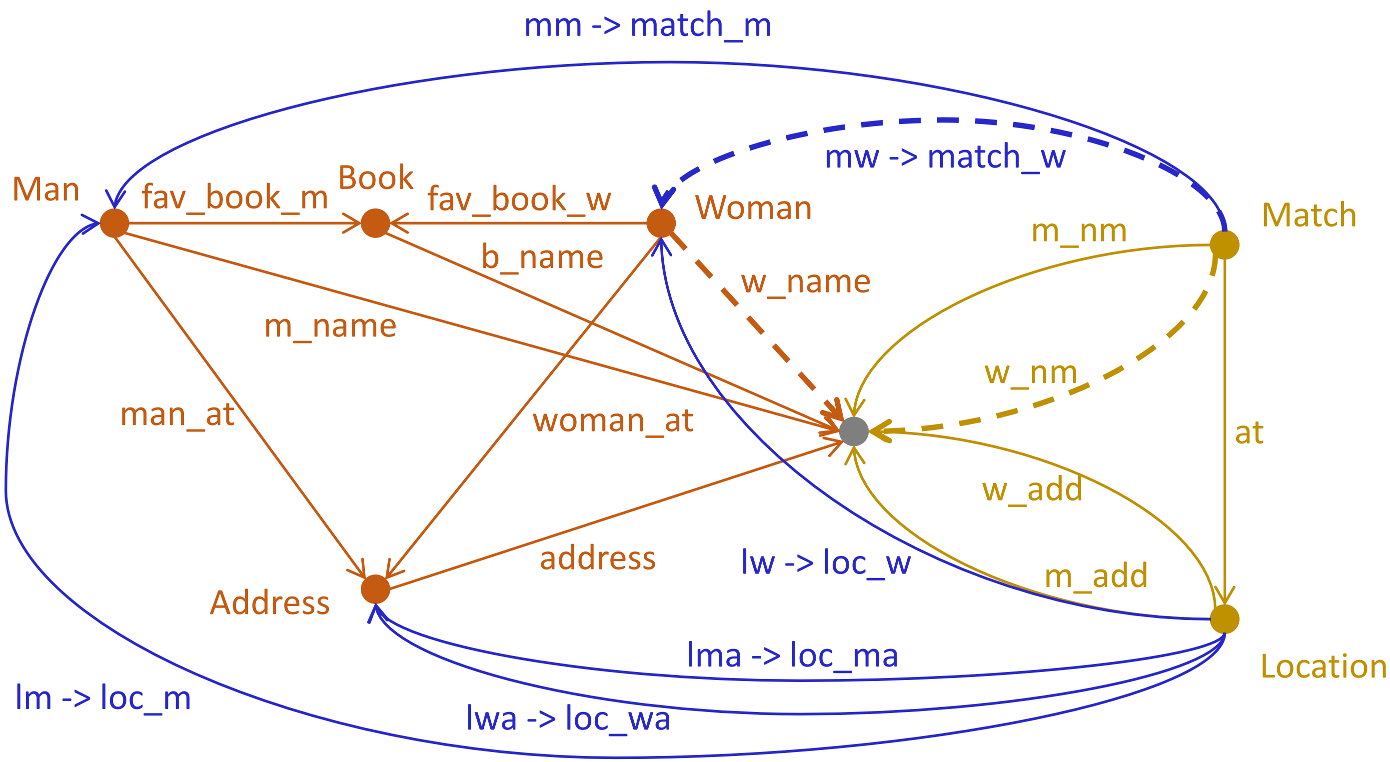
5. Add an Observation Equation for Target Attribute

m_nm = match_m.m_name



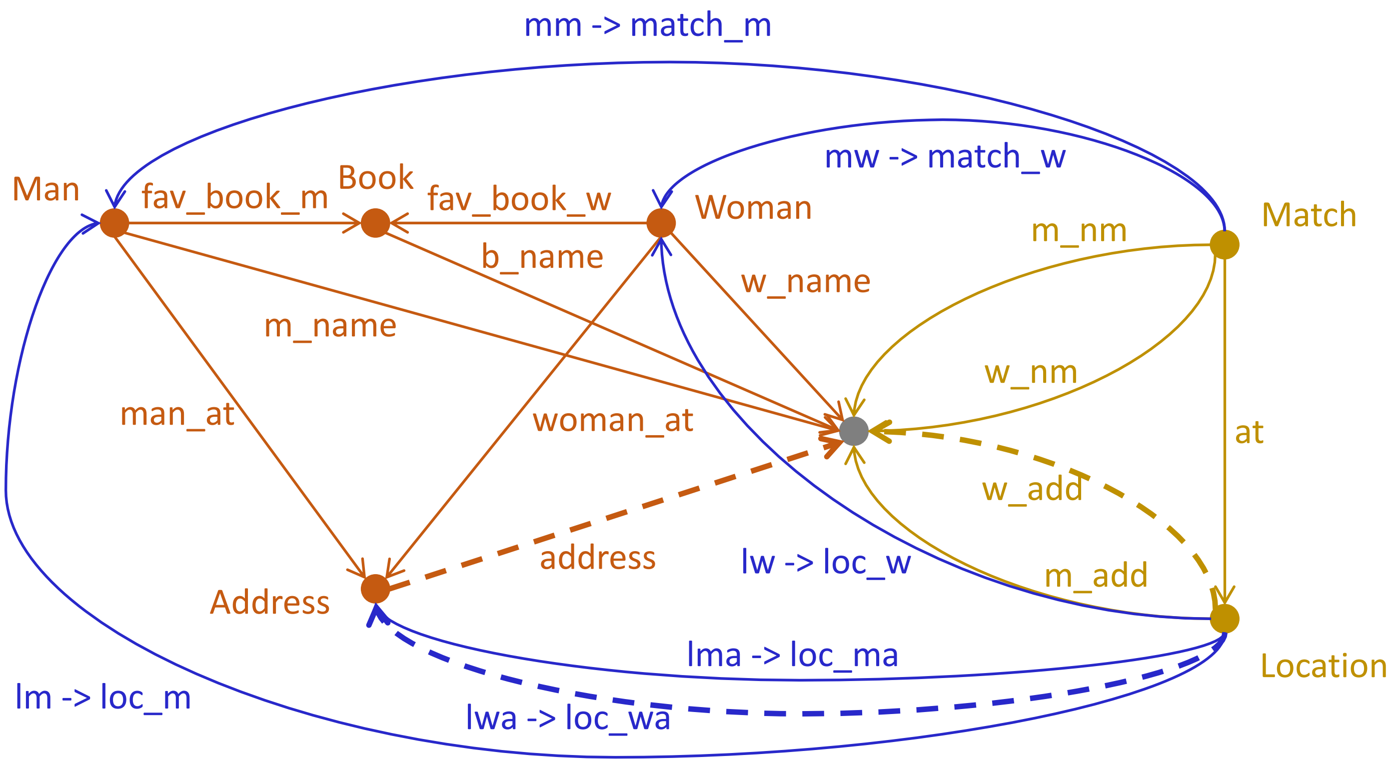
5. Add an Observation Equation for Target Attribute

w_nm = match_w.w_name



5. Add an Observation Equation for Target Attribute

w_add = loc_wa.address



5. Add an Observation Equation for Target Attribute

m_add = loc_ma.address