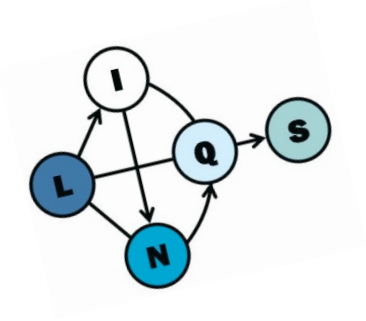
Exploiting Statistical and Relational Information on the Web and in Social Media

Lise Getoor & Lily Mihalkova

Tutorial at SDM-2011



Statistical Relational Learning and the Web

Challenges Addressed by SR Learning and Inference

- Multi-relational data
  - Entities can be of different types
  - Entities can participate in a variety of relationships
- Probabilistic reasoning under noise and/or uncertainty

Some

Challenges Arising in Web Applications

- Entities of different types
  - E.g., users, URLs, queries
- Entities participate in variety of relations
  - E.g., click-on, search-for, link-to, is-refinement-of
- Noisy, sparse observations

Tutorial Goals

- Understand the interactions between SRL and Web/ social media applications:
  - What are some sources of relational and statistical information on the Web/social media?
  - What are the basic SRL methods and techniques?
  - To what extent are existing SRL techniques a good fit for the challenges arising on the Web?
  - What future developments would make these areas more closely integrated?

Tutorial Road Map

- Introduction: Brief survey of statistical and relational info on the Web and in social media

- Main: Survey of SRL Models & Techniques
  - Relational Classifiers
  - Collective Classification
  - Advanced SRL models

- Conclusion: Looking Ahead

Disclaimer

- *Not* an attempt to provide a complete survey of the Web, social media, or SRL literatures
  - 3 hours is not enough for this!

- We provide a *biased* view, motivated by our goal of identifying the interesting intersection points of SRL and Web/social media applications

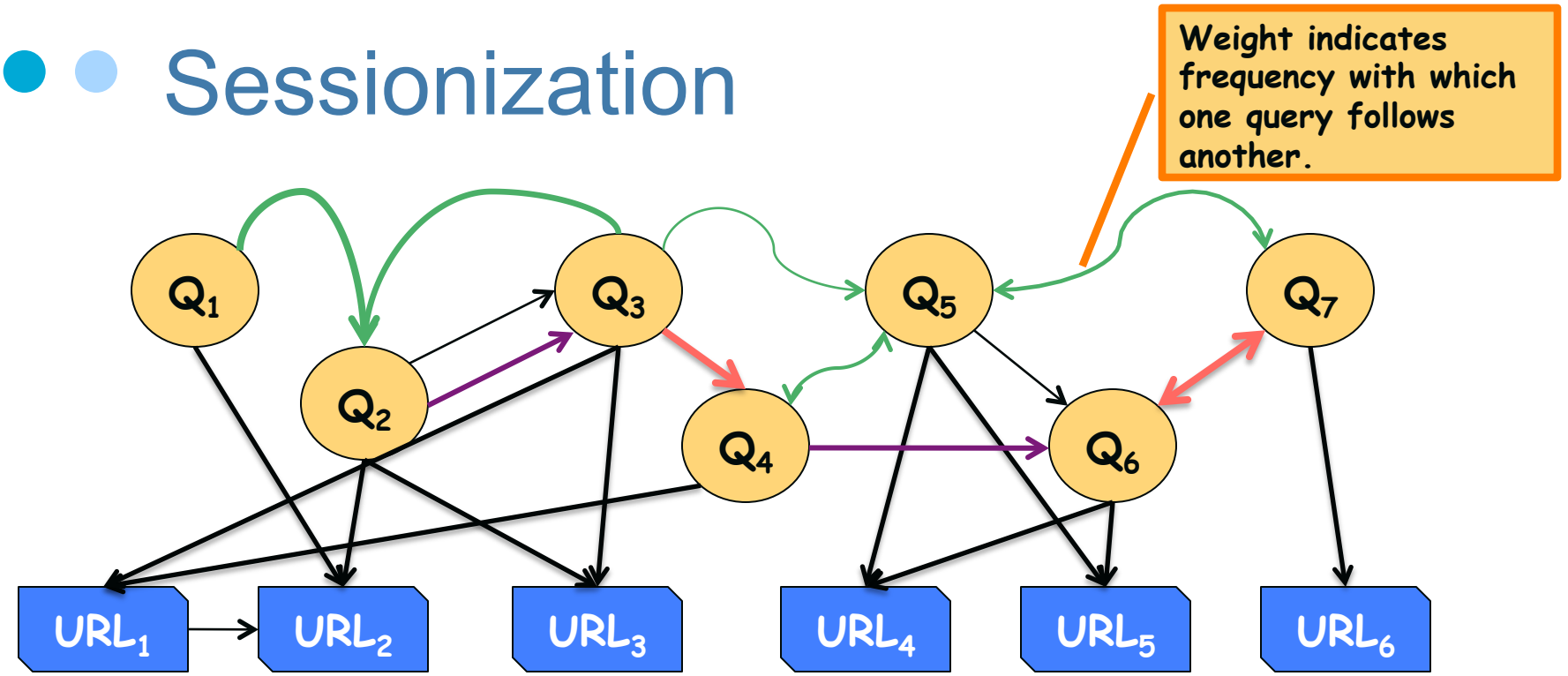
Relational Info on the Web

- Search engine log applications
  - Sessionization, clustering/refining queries, query personalization/disambiguation, click models, predicting commercial intent, query advertisement matching, many others
- Social networks/social media applications
  - Finding important nodes/influentials, understanding social roles/collaborative dynamics, viral marketing & information flow, link recommendation, community discovery

Sessionization

- Two kinds of sessions:
  - Search session
    - Determined using time-outs
  - Logical session
    - The same search session may contain queries for more than one information-seeking intent or search mission
    - Logical sessions may:
      - straddle search sessions
      - be intertwined
- Goal: Use query logs to determine whether two queries are part of the same logical session
- Following example is based on [Boldi et al., CIKM08] and [Jones & Klinkner, CIKM08]

Sessionization



Features Derived From:

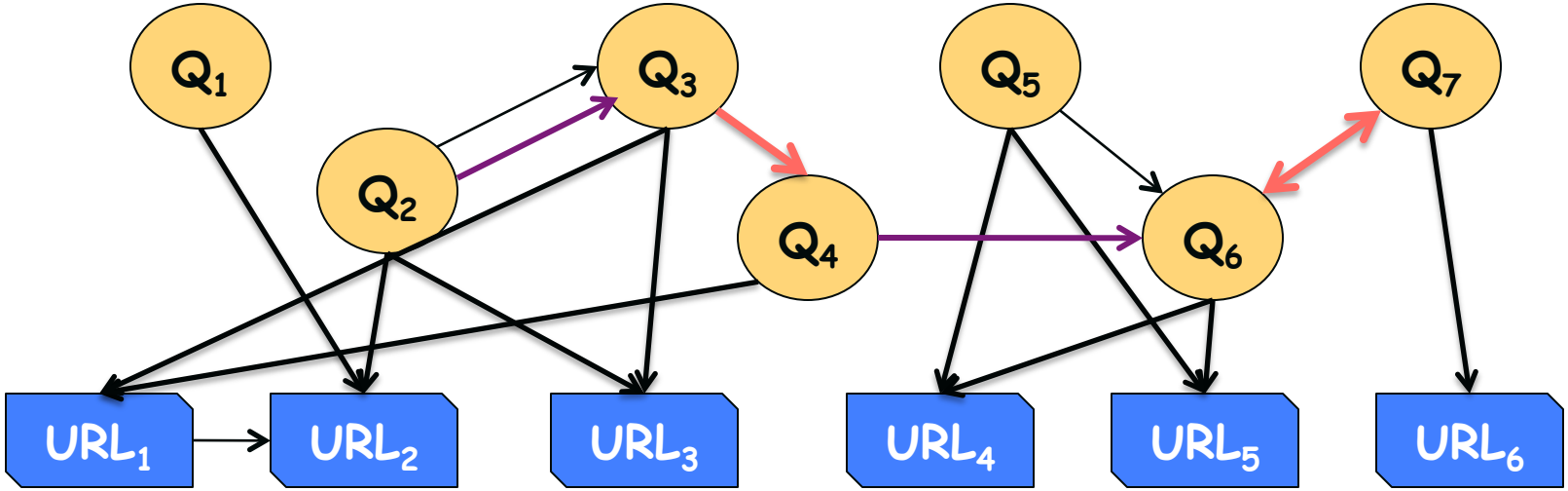
- Clicked-For
- Shares-Words
- Same-Session
- Precedes-In-Session
- Precedes-Temporally

Used to Learn to Predict:

- Precedes-In-Logical-Session
- Same-Logical-Session

Sessionization: Features

- Relations are typically not used directly; rather features are defined over them.



Clicked-For

Shares-Words

Same-Session

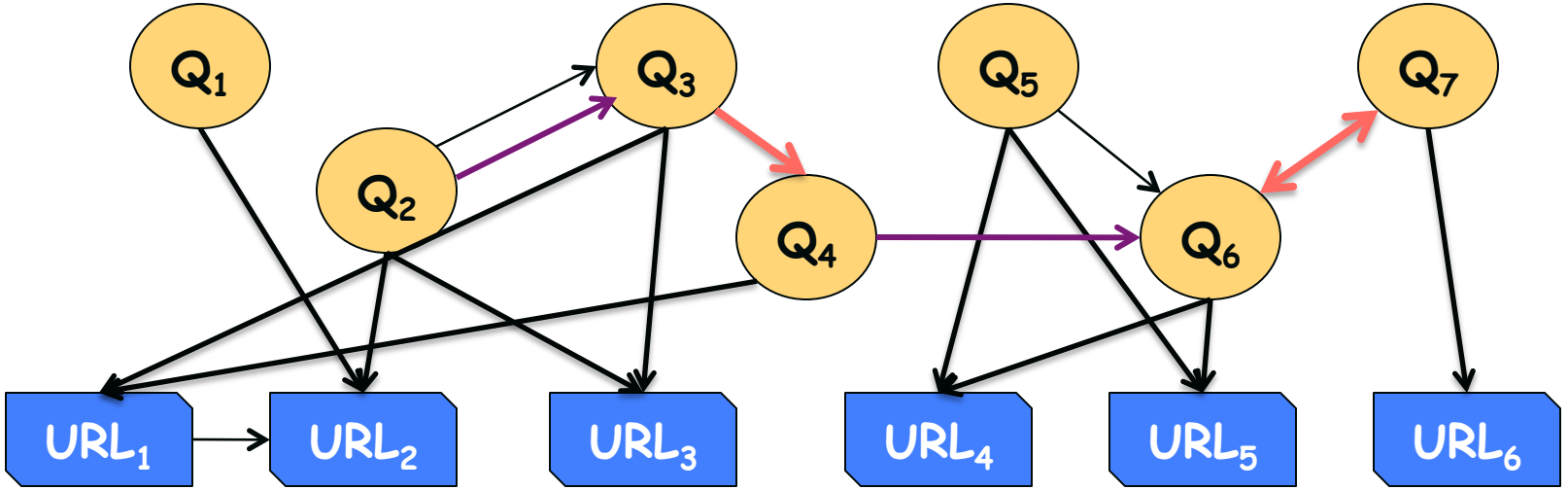
Precedes-In-Session

Precedes-Temporally

Word/character similarity, such as:
- Number of common words/characters
- Cosine, Jaccard similarity
- Character edit distance

Sessionization: Features

- Relations are typically not used directly; rather features are defined over them.



Clicked-For

Shares-Words

Same-Session

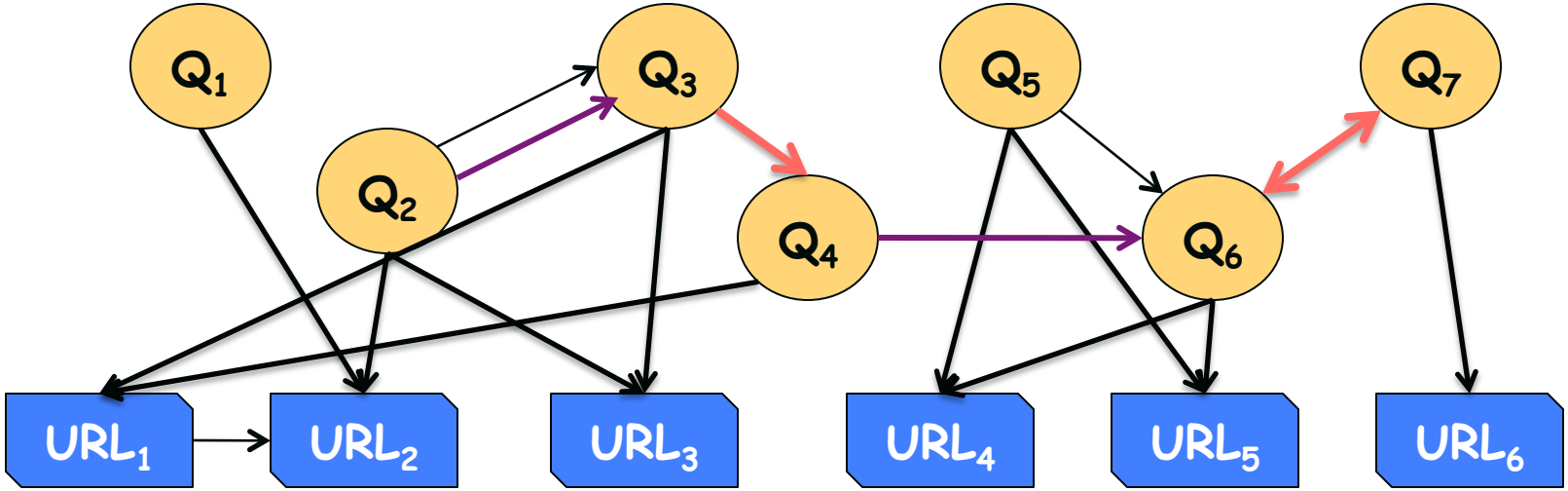
Precedes-In-Session

Precedes-Temporally

For example:
- Number of sessions in which co-occur
- Variety of stats over co-occurrence sessions, e.g. average length, average position of queries
- Statistical test indicating significance of co-occurrence

Sessionization: Features

- Relations are not used directly;
  rather, features are defined over them.

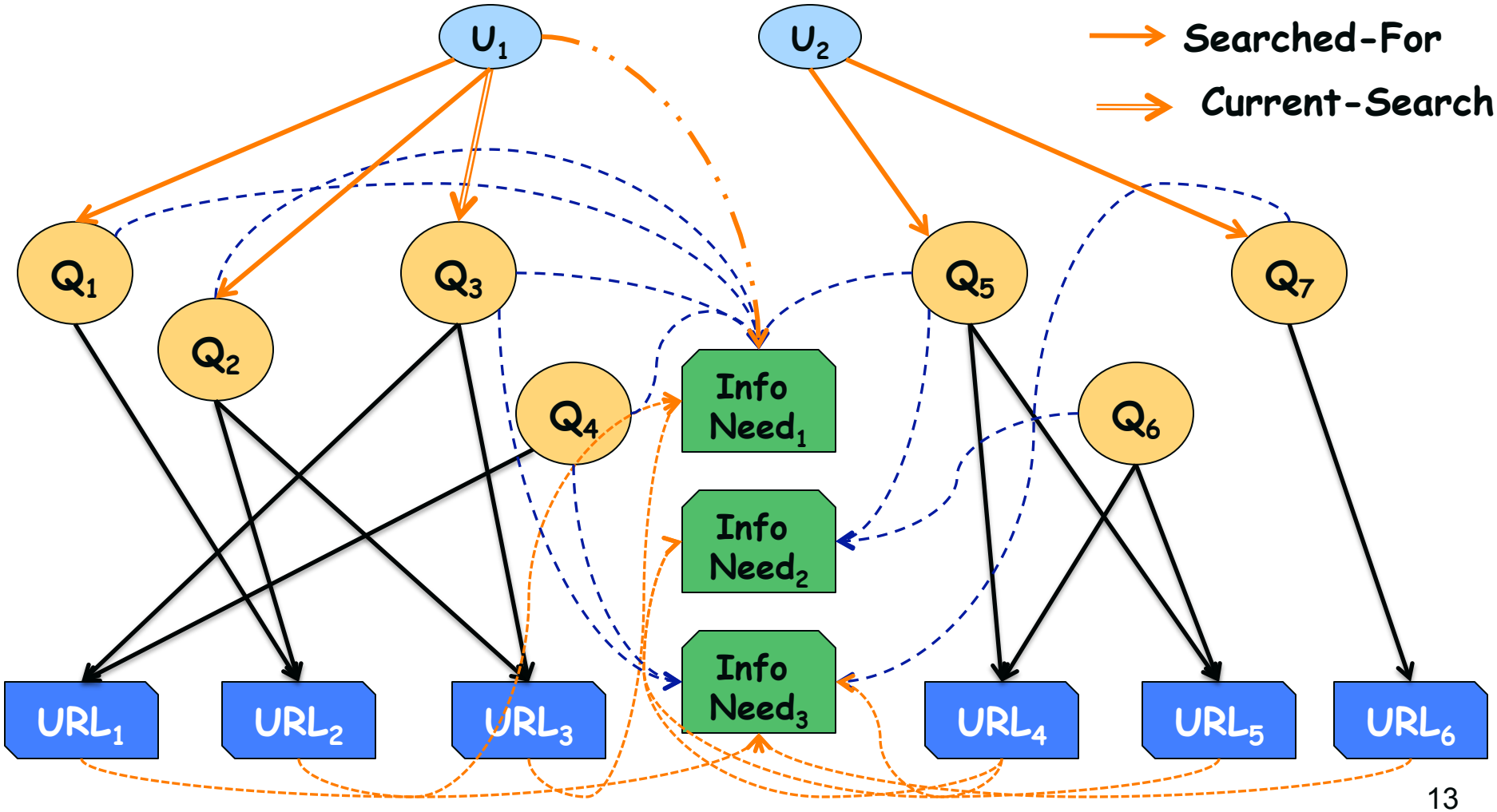


Clicked-For
Shares-Words
Same-Session
Precedes-In-Session
Precedes-Temporally

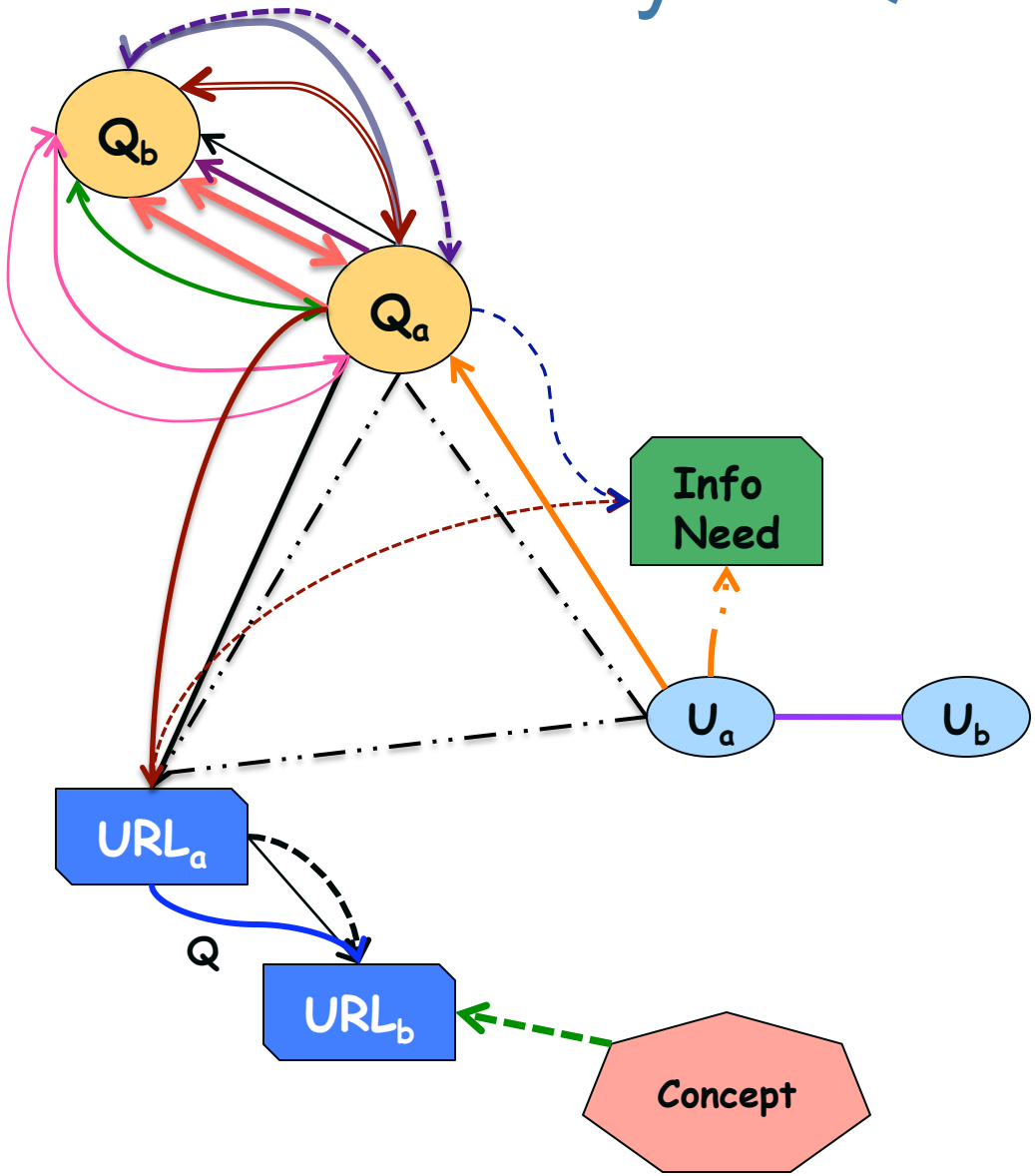
Examples:
- Average time between queries
- Time between queries > threshold

Personalized Search

- Can also include information about users, their searches and their information needs

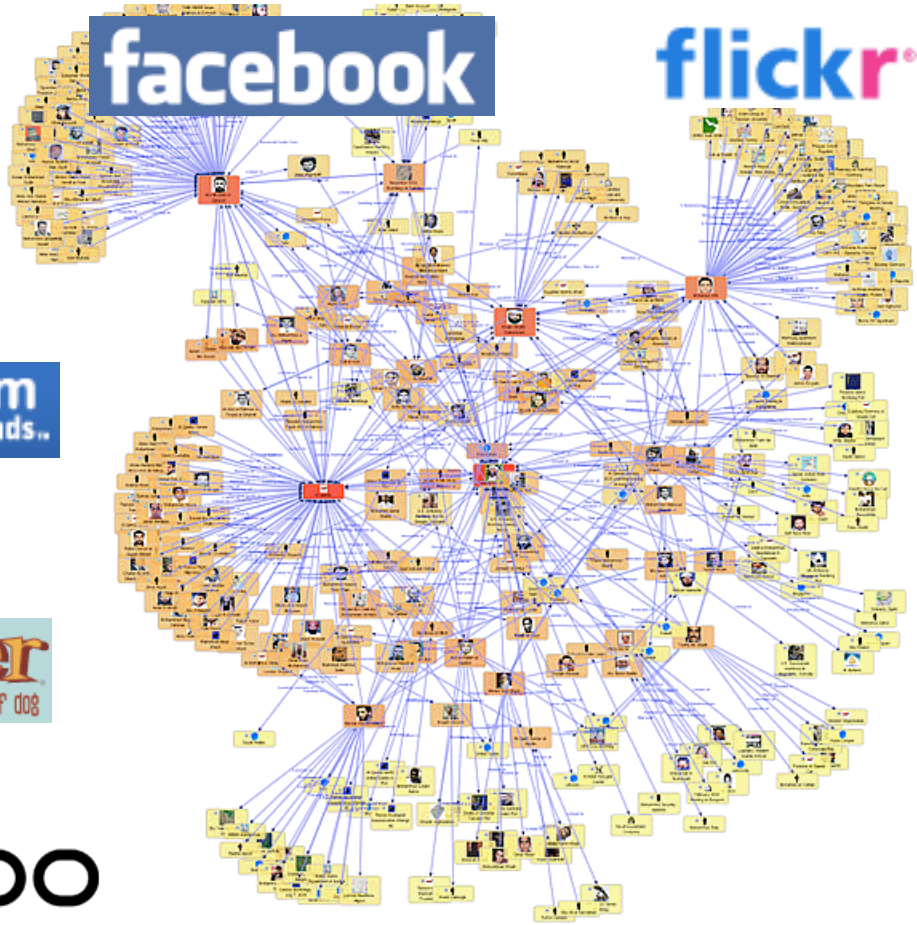


Summary of Query Logs Apps

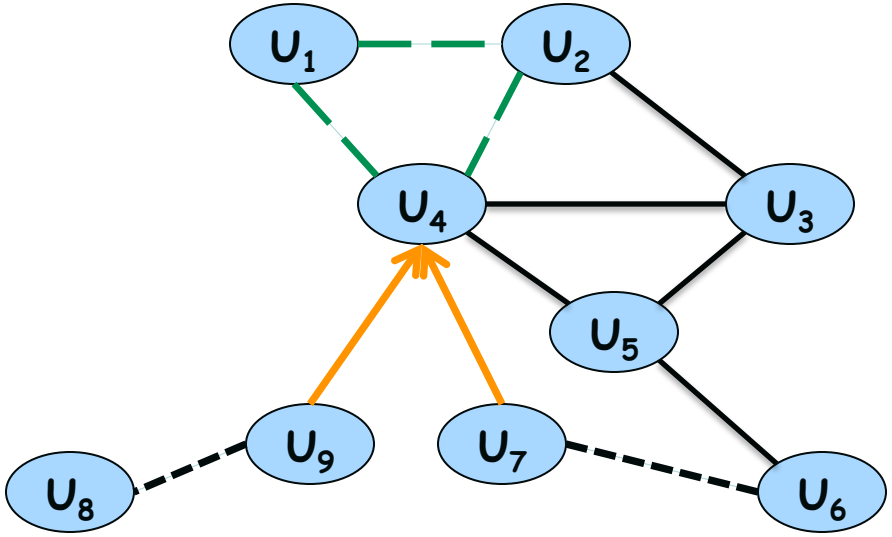


Clicked-For
Shares-Terms
Shares-Words
Same-Topic
Hyperlink
Same-Session
Precedes-In-Session
Subset-URLs
Identical-URLs
Partial-Overlap-URLs
Precedes-Temporally
Prec-In-Logical-Sess.
Same-Logical-Session
Is-Represented-By
Fulfills-Info-Need
Targets-Info-Need
Have-Info-Need
Search-For
Search-For-&-Click
Similar Users
Q More-Relevant-Than

Relational Info in Social Media



Online Social Networks



Friends

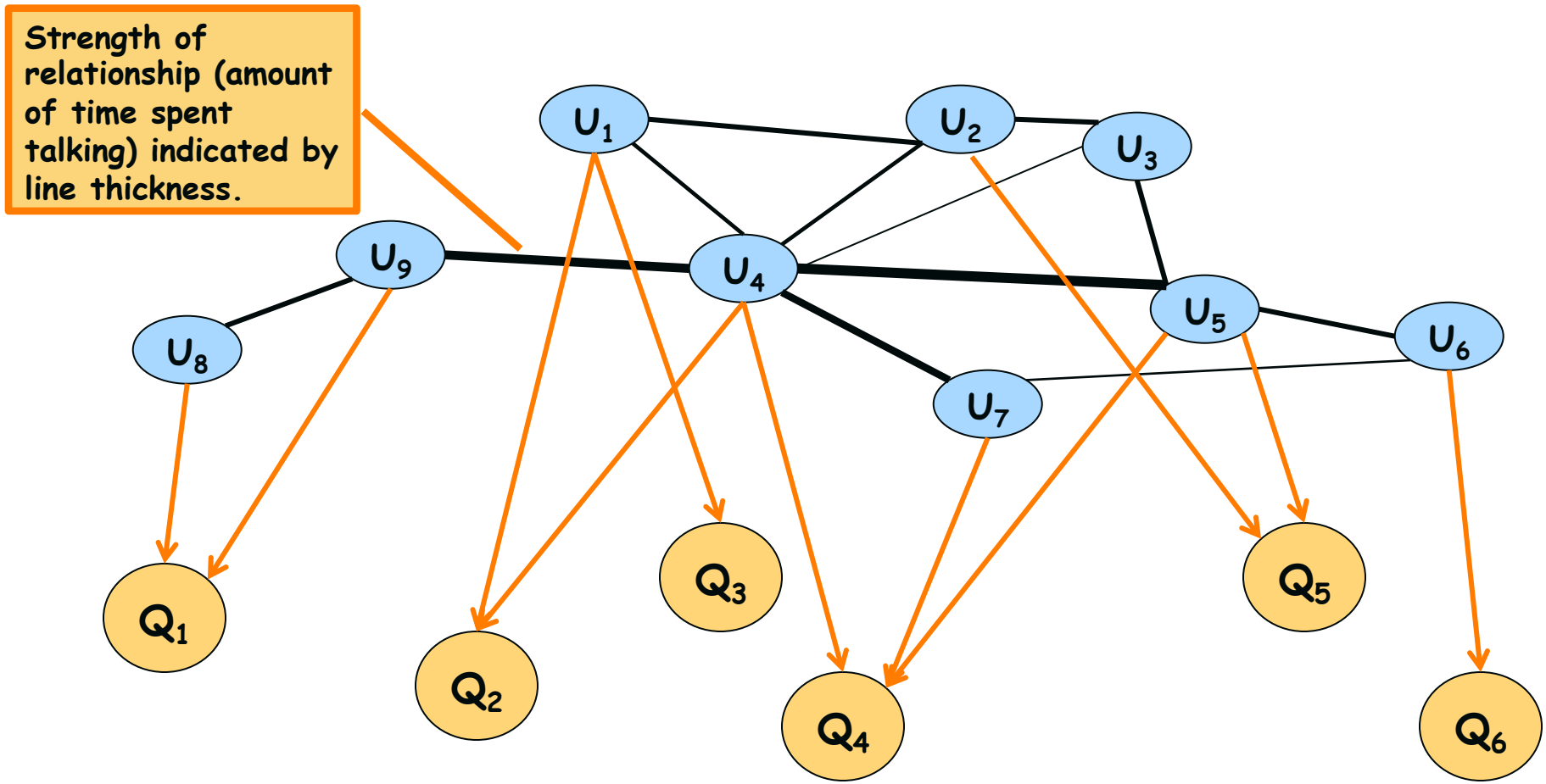
Collaborators

Family

Fan/Follower

Comments, Replies, Edits, Co-Edits, Co-Mentions, etc.

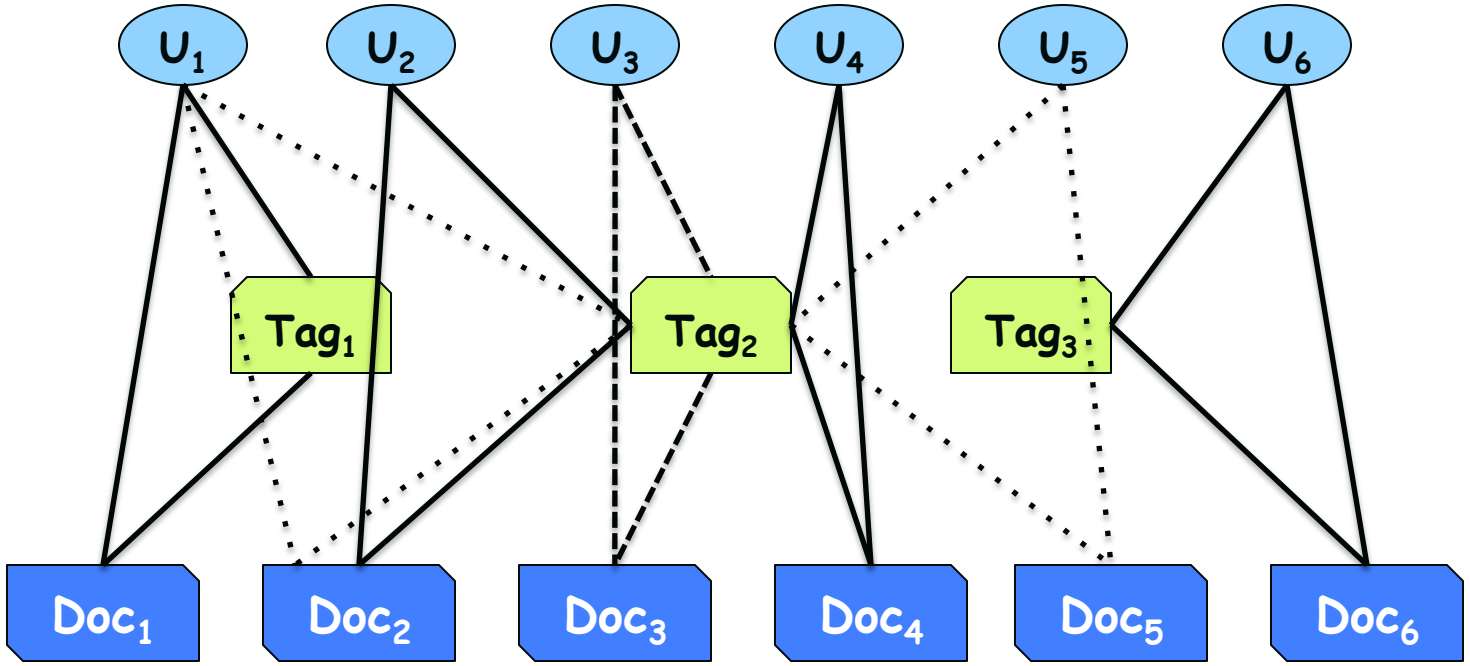
Social Networks & Query Logs



[Singla & Richardson WWW08]: Similarities between querying behavior and talking to each other or having friend in common.

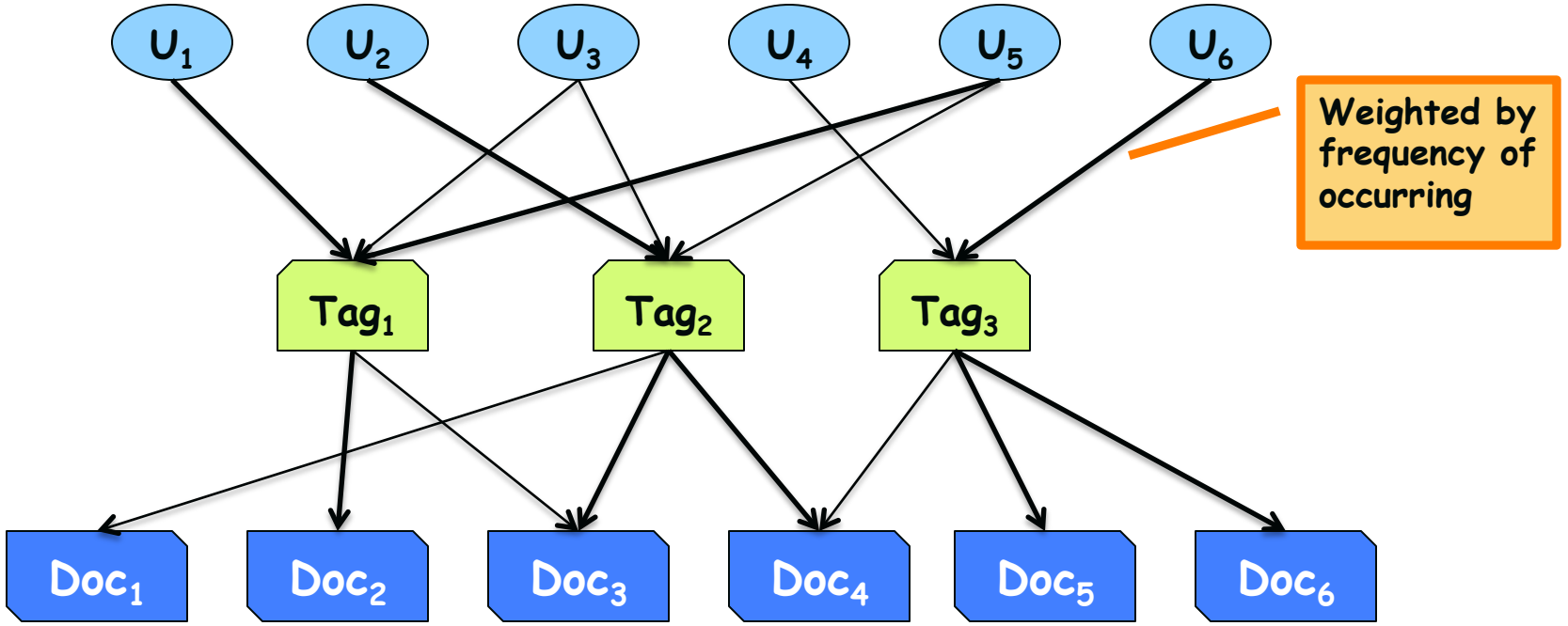
Social Tagging, View 1

- Ternary relationships between tags, users, documents



Social Tagging, View 2

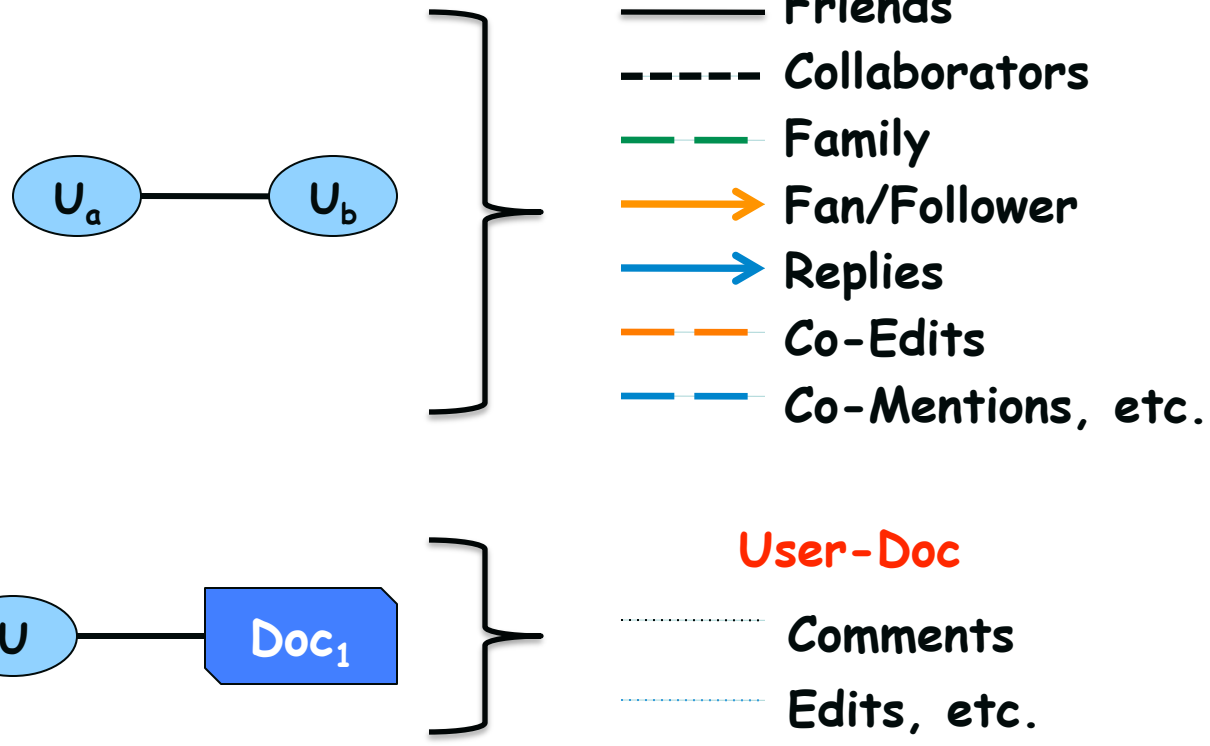
- Tri-partite graph
  - Aggregate over documents/tags



[Shepitsen et al., RS08]
[Guan et al., WWW10]

Document recommendations are based on not just preferences of similar users but also preferences for tags.

Summary of Social Media Relationships



SURVEY OF SRL MODELS & TECHNIQUES

Road Map

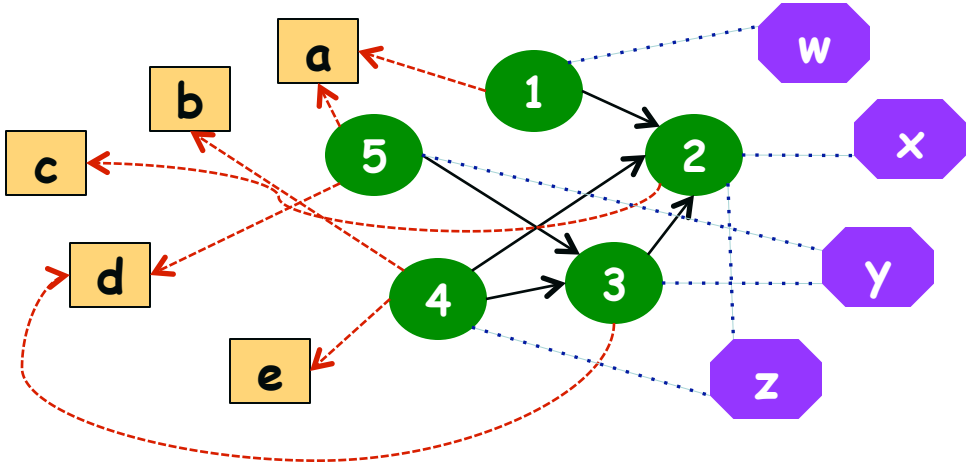
- Relational Classifiers 
- Collective Classification
- Advanced SRL Models

Road Map

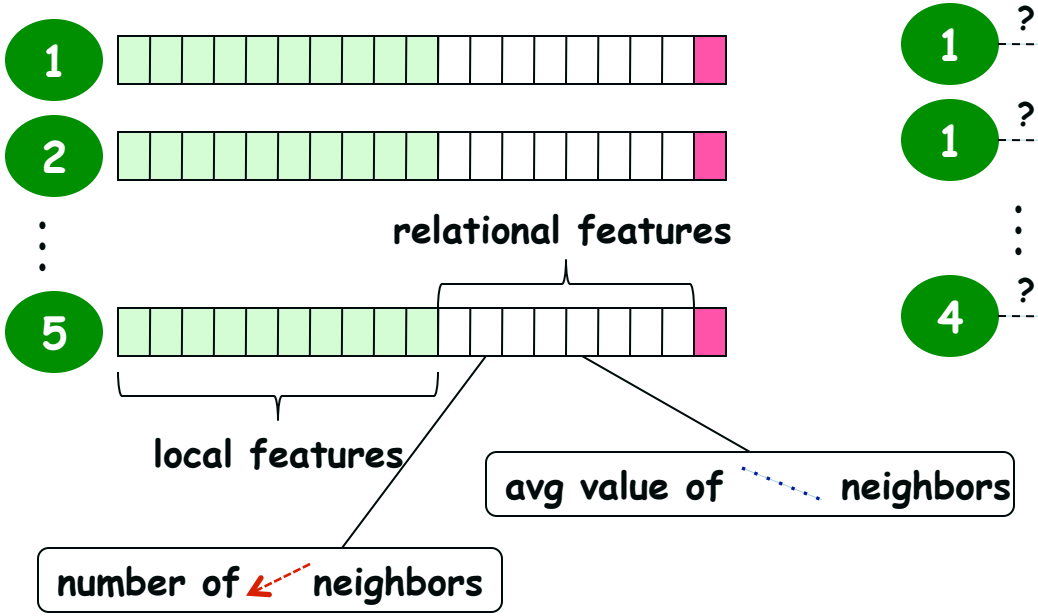
- Relational Classifiers
  - Definition
  - Case Studies
  - Key Idea: Relational Feature Construction
- Collective Classification
- Advanced SRL Models

Relational Classifiers

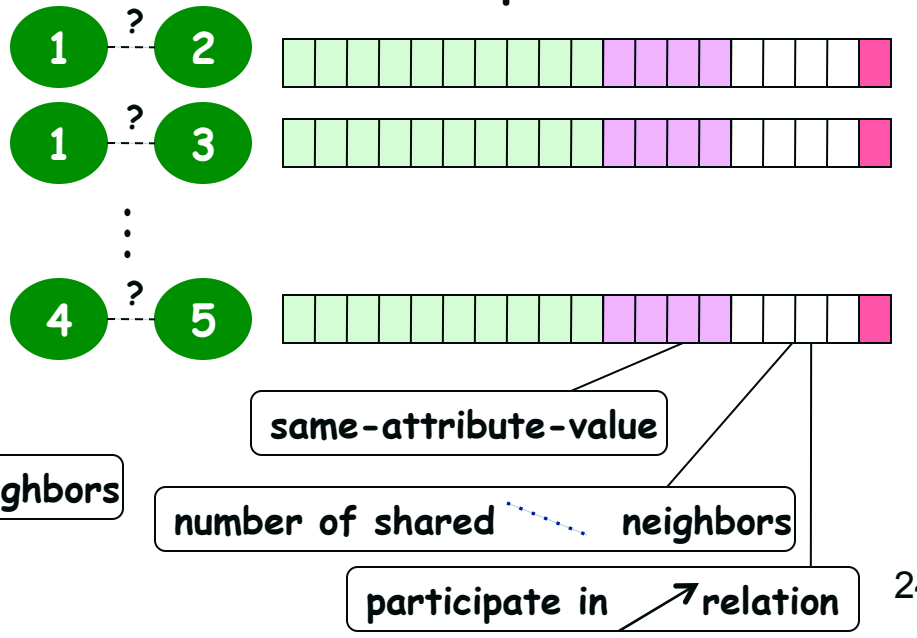
Given:



Task: Predict attribute of some of the entities



Alternate task: Predict existence of relationship between entities



Relational Classifiers

- Relational features are pre-computed by aggregating over related entities
- Values are represented as a fixed-length feature vector
- Instances are treated independently of each other
- Any classification or regression model can be used for learning and prediction

Application Case Studies

- Next we present two applications that use relational classifiers
  - Focus is on types of relational features used

- Case Study 1: Predicting click-through rate of search result ads
- Case Study 2: Predicting friendships in a social network

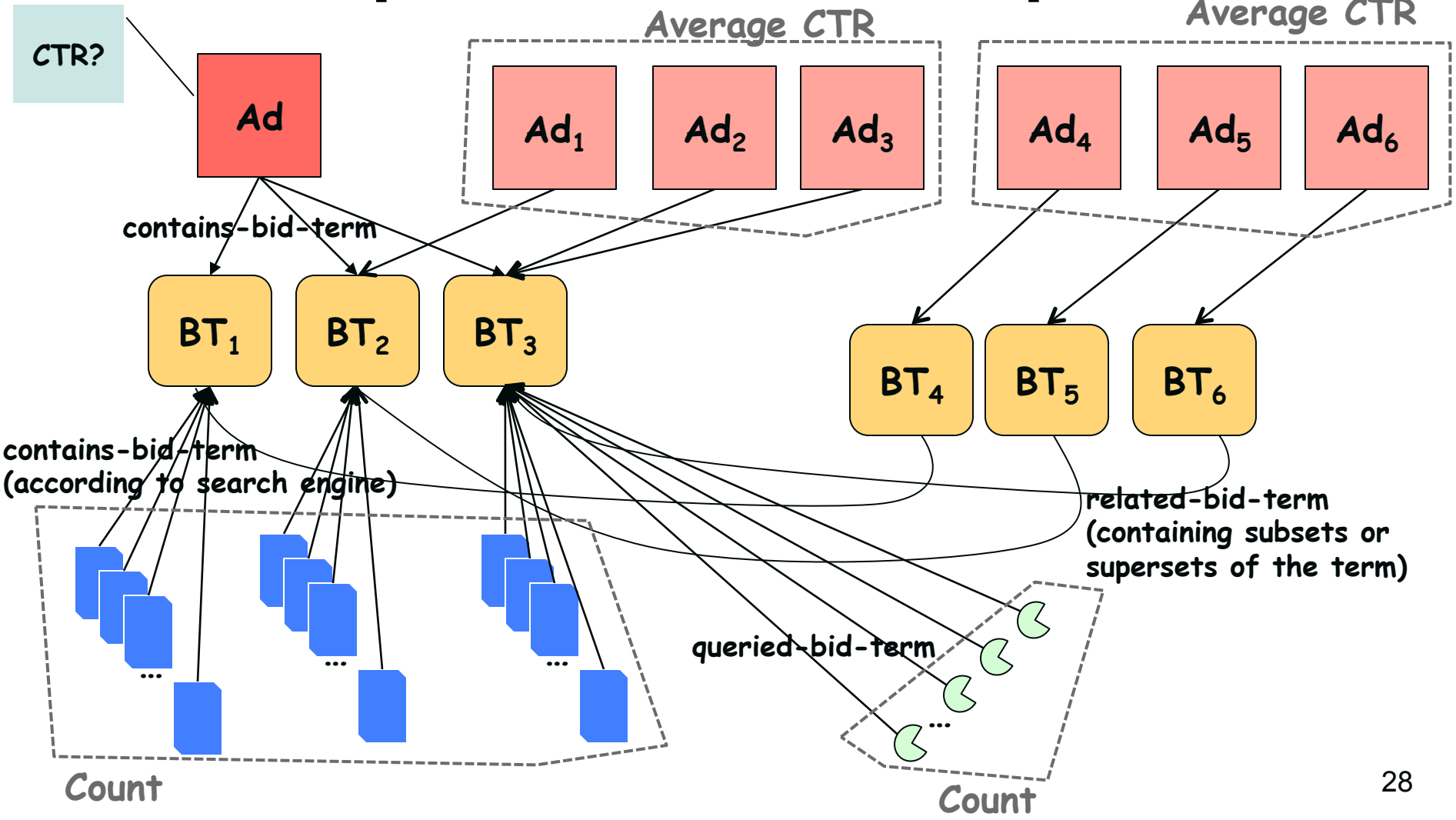
Case Study 1: Predicting Ad Click-Through Rate

- Task: Predict the click-through rate (CTR) of an online ad, given that it is seen by the user, where the ad is described by:
  - URL to which user is sent when clicking on ad
  - Bid terms used to determine when to display ad
  - Title and text of ad

- Our description is based on approach by
  - [Richardson et al., WWW07]

Relational Features Used

- Based on [Richardson et al., WWW07]



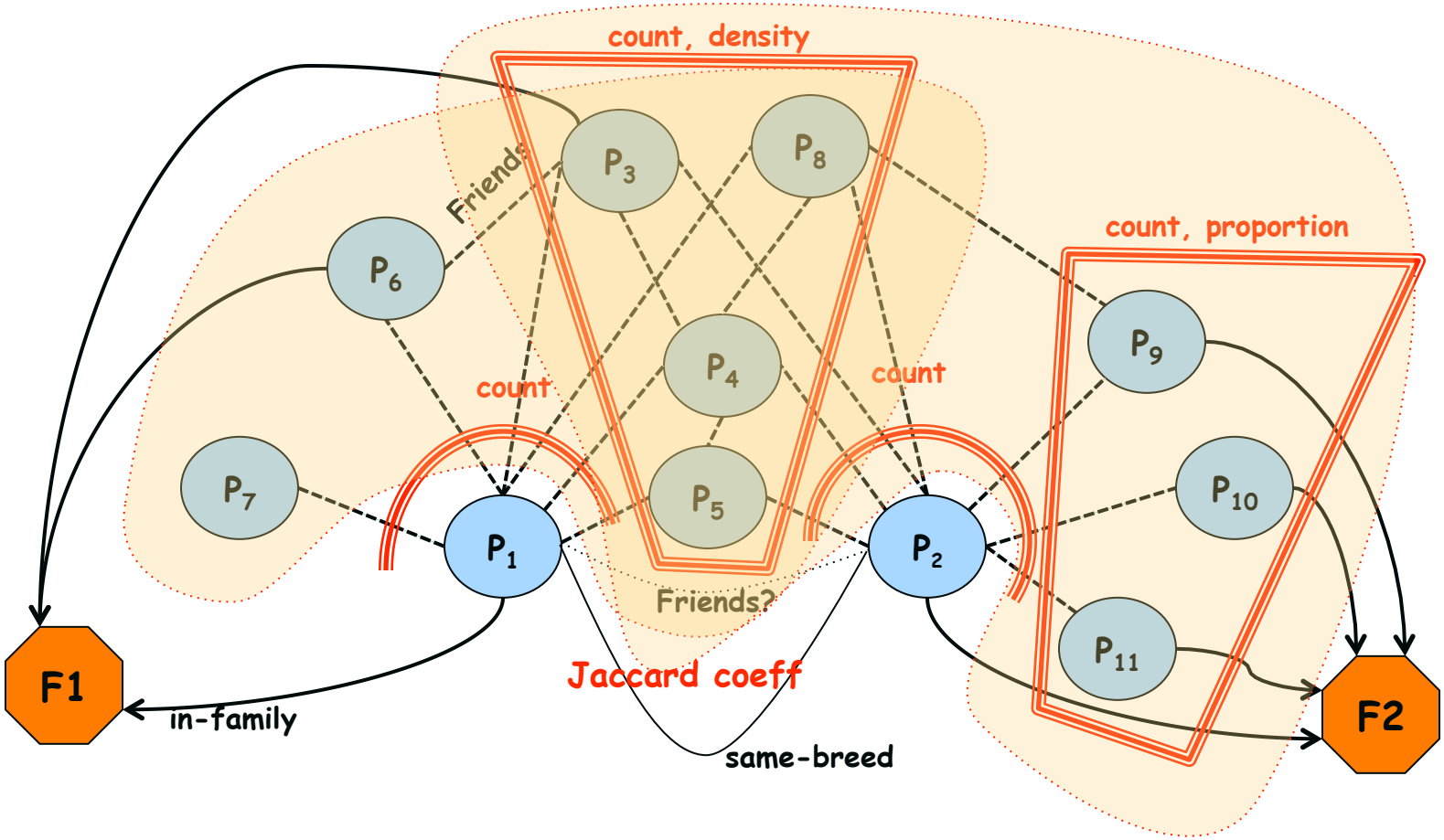
Case Study 2: Predicting Friendships

- Task: Predict new friendships among users, based on their descriptive attributes, their existing friendships, and their family ties.

- Our description is based on approach by
  - [Zheleva et al., SNAKDD08]

Relational Features Used

- "Petworks" - social networks of pets



Key Idea: Feature Construction

- Feature informativeness is key to the success of a relational classifier
- Next we provide a systematic review of relational feature construction
  - Global measures
  - Node-specific measures
  - Node pair measures
- These will be useful also for collective classifiers and other SRL models

Global Measures

- Summarize properties of entire graph (or subgraph)
- Next we discuss:
  - Graph Cohesion
  - Clustering coefficient
  - Bipolarity
- Many others possible…

Graph Cohesion

[Everett & Borgatti, 1999]

- Density (% of possible edges)
- Average Degree
- Average Tie Strength
- Max flow
- Size of largest clique
- Average geodesic distance
- Diameter (max distance)
- F Measure - proportion of pairs of nodes that are unreachable from each other

- Many others….

Clustering Coefficient

[Watts & Strogatz, Nature98]

- Measures cliquishness of an undirected, unweighted graph, or its tendency to form small clusters
- Computed as the proportion of all incident edge pairs that are completed by a third one to form a triangle

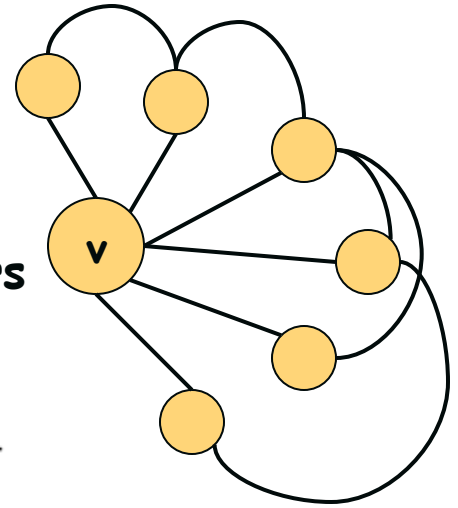
$$CC(G) = \frac{1}{|V|} \sum_{v \in V} CC(v)$$

$$CC(v) = \frac{|\{(i,j) \in E | i \in N_v \wedge j \in N_v\}|}{\frac{1}{2} k_v (k_v - 1)}$$

$$= \frac{\text{Num actual neighbor links}}{\text{Possible num neighbor links}}$$

Set of v's neighbors

Number of neighbors of v



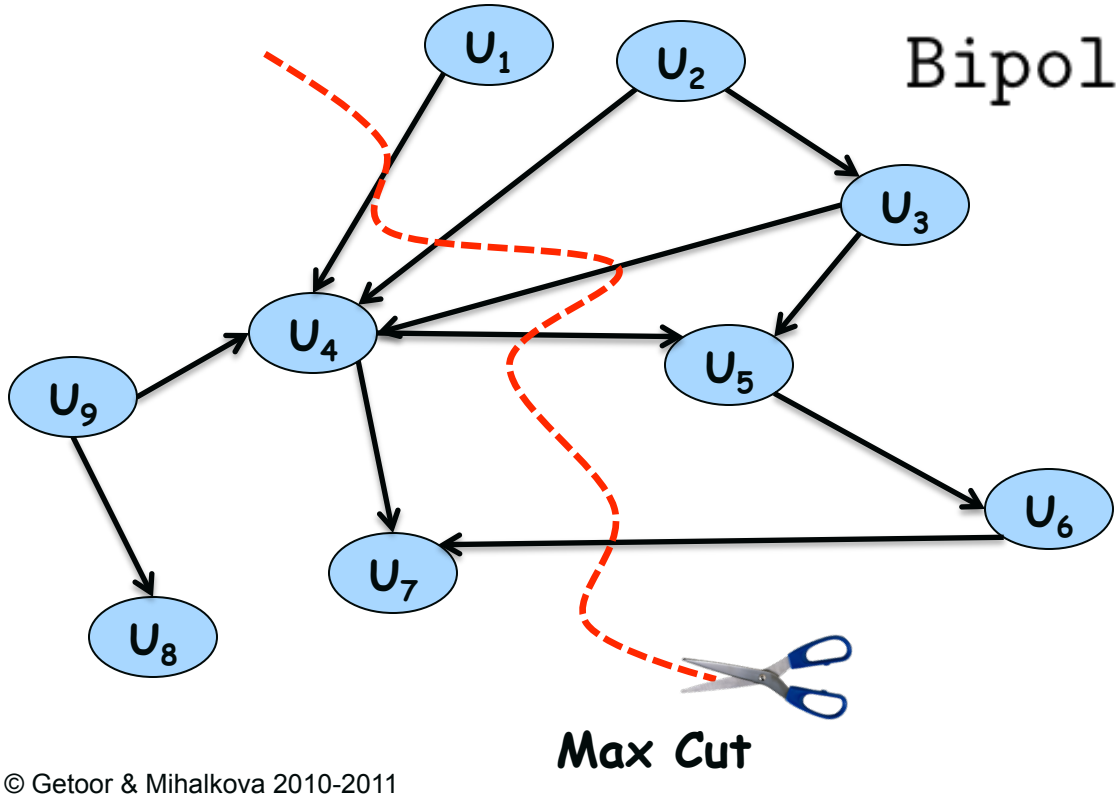
Clustering Coefficient Cont.

- Extensions exist for
  - Directed graphs [Kunegis et al. WWW09]
  - Graphs with weighted edges [Kalna & Higham, AICommunic07]
  - Graphs with signed edges [Kunegis et al. WWW09]

Bipolarity

[Brandes et al, WWW09]

- Defined on a weighted directed graph
- Measures to what extent the nodes in the graph are organized in two opposing camps
  - i.e., how close is the graph to being bipartite



$$\text{Bipolarity}(G) = \frac{w - c}{w + c}$$

max weight across the cut

weight on either side of the cut

Value between -1 and +1

Node-specific Measures

- Summarize properties of node

- Next we discuss:
  - Attribute aggregates
  - Structural measures

Attribute Aggregates: Level 1

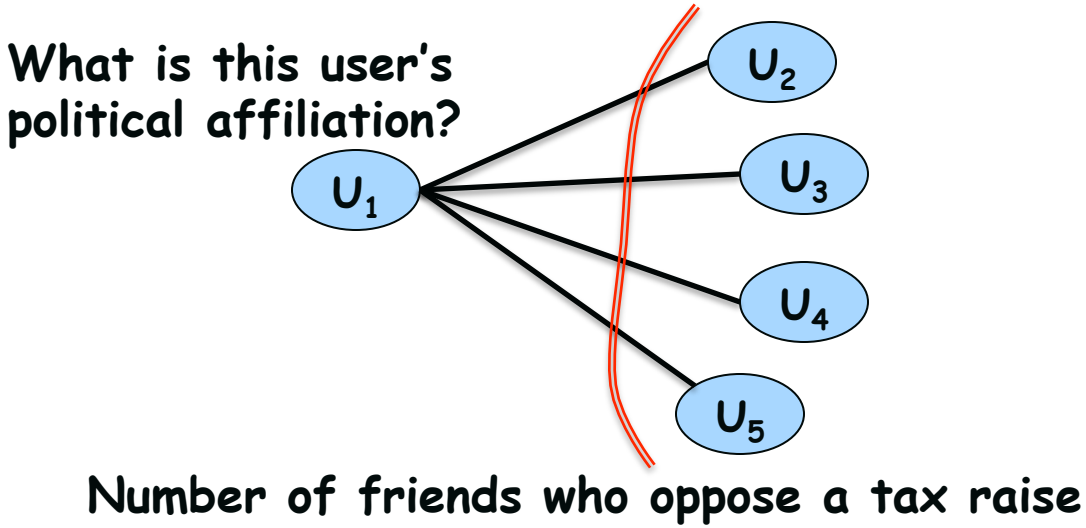
Based on [Perlich & Provost, KDD03]

- No aggregation necessary
  - Use an attribute of the entity about which a prediction is made
  - Relationships to other entities are not used
- Example: Predicting the political affiliation of a social network user can be based on whether user opposes a tax raise

Attribute Aggregates: Level 2

Based on [Perlich & Provost, KDD03]

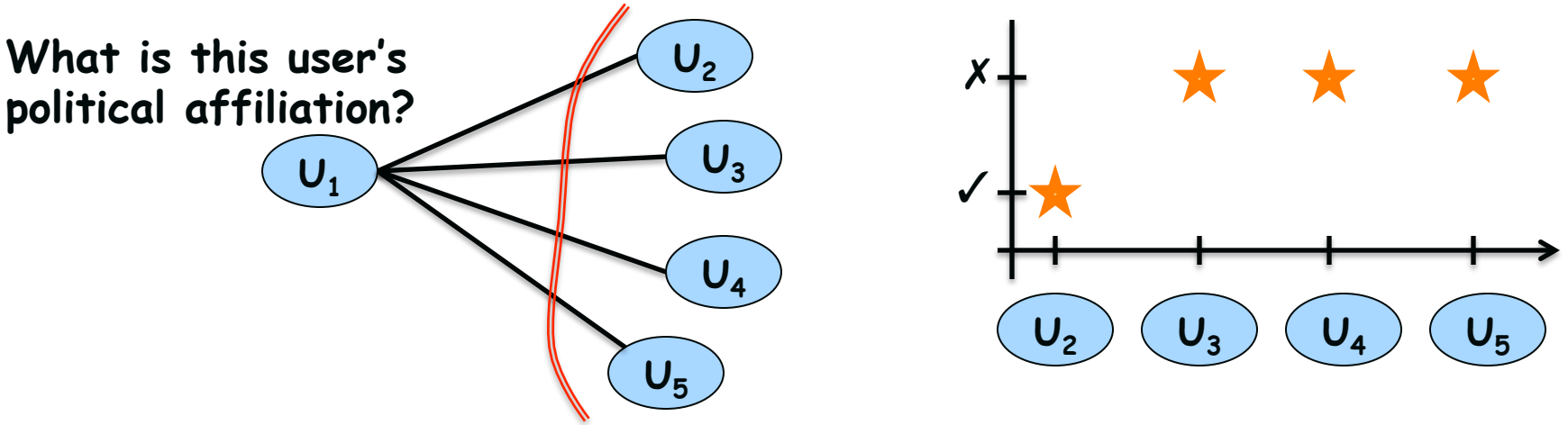
- Aggregation over independent attributes of related entities
  - Values at related entities are considered independently of one another
- Example:



Attribute Aggregates: Level 3

Based on [Perlich & Provost, KDD03]

- Aggregation over dependent attributes of related entities
  - Values at related entities need to be considered together as a set
- Example:



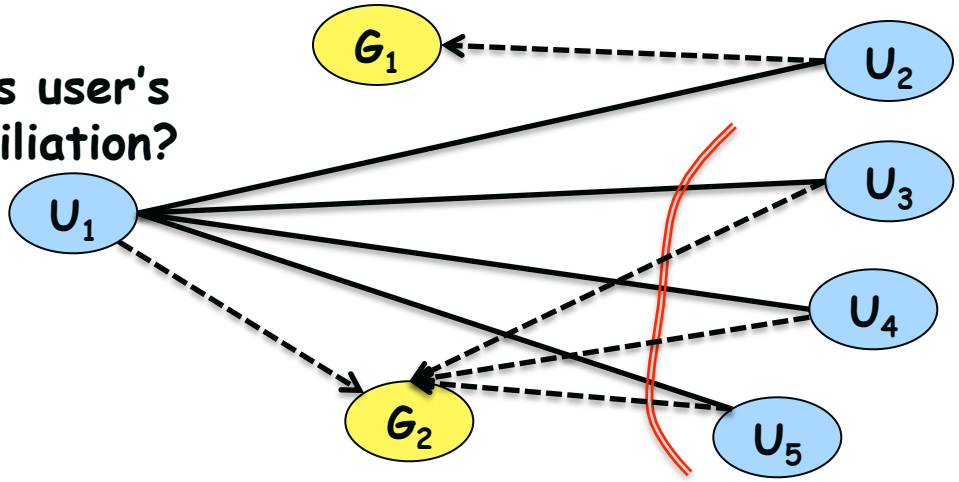
Trend of friendships to people who oppose a tax raise made over time

Attribute Aggregates: Level 4

Based on [Perlich & Provost, KDD03]

- Level 4: Aggregation over dependent attributes across multiple relations
  - Aggregate computed over multiple "hops" across relational graph
  - Values need to be considered together
- Example:

What is this user's political affiliation?



Trend of friendships made over time to liberal users that are members of the same groups as U_1

Representing Attribute Aggregates with First-Order Logic

Based on [Perlich & Provost, KDD03] and [Popescul & Ungar, MRDM03]

- Defining Boolean-valued features using FOL
  - A feature that checks if U_1 has a liberal friend who shares group membership:

$$\exists u: \text{friends}(U_1,u) \wedge \text{inGroup}(U_1,g) \wedge \text{inGroup}(u,g) \wedge \text{liberal}(u)$$

- Augmenting FOL with arbitrary aggregation functions
  - A feature that counts the number of such friends

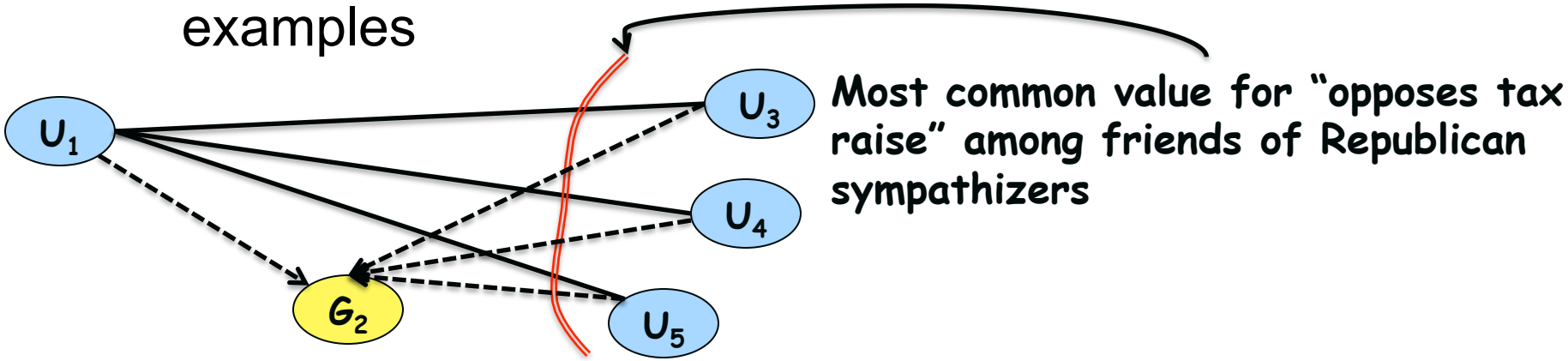
$$\text{Count}(u): \quad \text{friends}(U_1,u) \wedge \text{inGroup}(U_1,g) \wedge \text{inGroup}(u,g) \wedge \text{liberal}(u)$$

Advantage: Can represent arbitrary chains of relations
Disadvantage: Numerical values are cumbersome

Numeric Aggregations

Based on [Perlich & Provost, KDD03]

- Features based on frequently occurring values
  - Most common value
  - Most common value in positive/negative training examples



  - Value whose frequency of occurring differs the most in positive vs negative examples
- Features based on vector distances
  - Difference in distribution over values

Structural Measures

- Cohesion
  - CC(v) – clustering coefficient at a node
  - Stability - valence of triads: +++, --- are stable; +-+ instable
- Centrality
  - Degree centrality
  - Betweenness centrality
  - Eigenvalue centrality (a.k.a. PageRank)
- For more, see [Wasserman & Faust, 94]

Degree Centrality

- A very simple but useful aggregation:
- Degree centrality of a node = number of neighbors
- Sometimes normalized by the total number of nodes in the graph

Betweenness Centrality

- A node **a** is more central if paths between other nodes must go through it; i.e. more node pairs need **a** as a mediator

Number of shortest paths between j and k that go through a

$$BC(a) = \sum_{j<k} \frac{|SP^{\rightsquigarrow a}(j,k)|}{|SP(j,k)|}$$

Total number of shortest paths between j and k

Node-Pair Measures

- Summarize properties of (potential) edges

- Next we discuss:
  - Attribute-based measures
  - Edge-based measures
  - Neighborhood similarity measures

Attribute Similarity Measures

- Measures defined on pairs of nodes
- Attribute similarity measures to compare nodes based on their attributes'
  - String similarity
  - Hamming distance
  - Cosine
  - etc.
- Component similarities are features for relational classifier*

***or overall attribute similarity based on some weighted combination of components and simple threshold is applied**

Edge-Based Measures

- Edges can be of different types, corresponding to different kinds of relationships
  - Edges of one type can be predictive of edges of another type, e.g., working together is predictive of friendship
- Edges can be weighted or have other associated attributes to indicate the strength, or other qualities, of a relationship
  - E.g., the thickness of an edge between two users indicates frequency of exchanged emails

Structural Similarity Measures

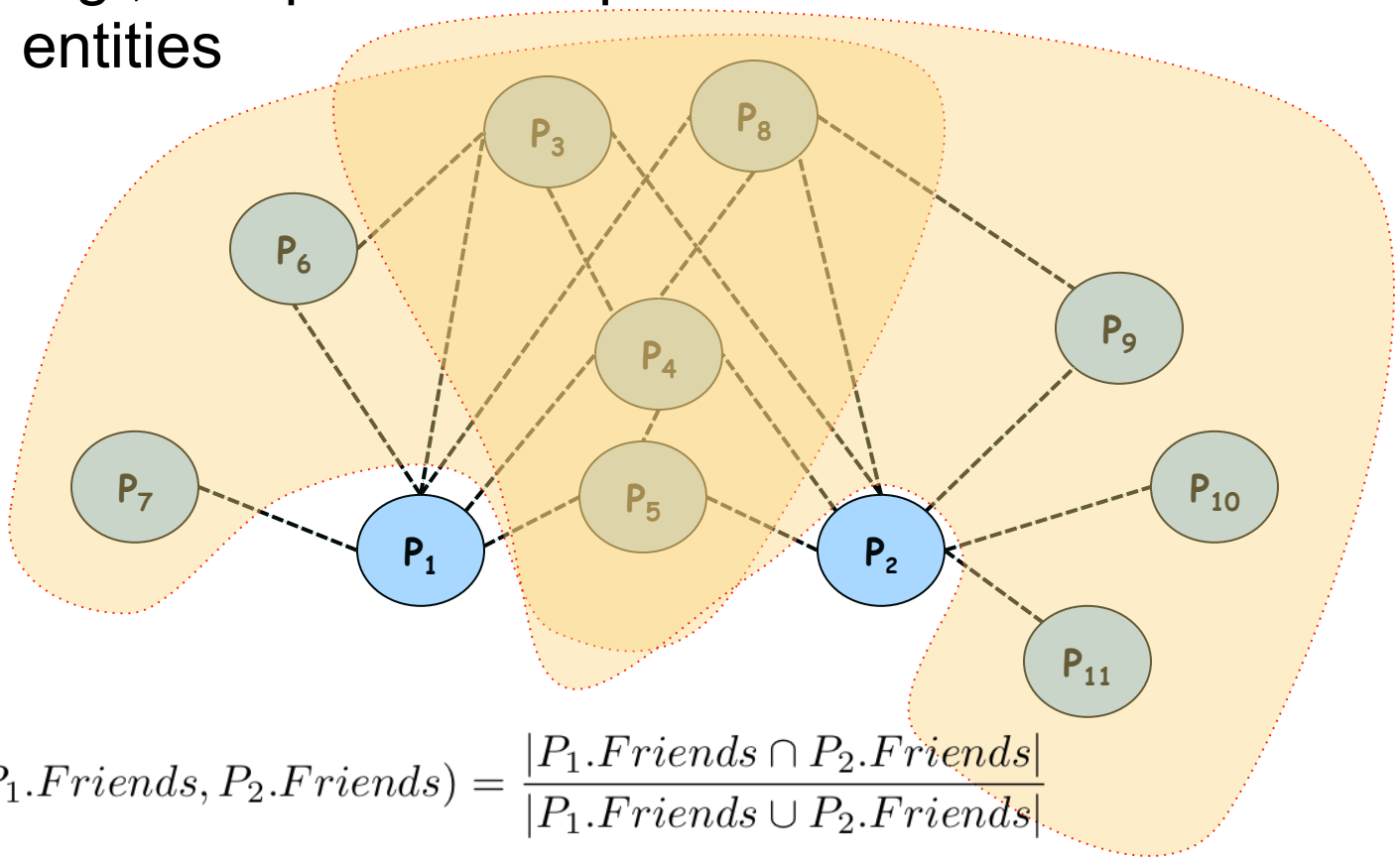
- Set similarity measures to compare nodes based on set of related nodes, e.g., compare neighborhoods

- Examples:
  - Average similarity between set members
  - Jaccard coefficient
  - Preferential attachment score
  - Adamic/Adar measure
  - SimRank
  - Katz score

- For more details, see [Liben-Nowell & Kleinberg, JASIST07]

Jaccard Coefficient

- Compute overlap between two sets
  - e.g., compute overlap between sets of friends of two entities



$$Jaccard(P_1.Friends, P_2.Friends) = \frac{|P_1.Friends \cap P_2.Friends|}{|P_1.Friends \cup P_2.Friends|}$$

Preferential Attachment Score

[Liben-Nowell & Kleinberg, JASIST07]

- Based on studies, e.g. [Newman, PRL01], showing that people with a larger number of existing relations are more likely to initiate new ones.

$$s(a,b) = |N_a| \cdot |N_b|$$

Set of a's neighbors

Adamic/Adar Measure

[Adamic & Adar, SN03]

- Two users are more similar if they share more items that are overall less frequent

$$s(a,b) = \sum_{i \in \text{Shared items}} \frac{1}{\log(\text{frequency(i)})}$$

Can be any kind of shared attributes or relationships to shared entities

Overall frequency in the data

SimRank

[Jeh & Widom, KDD02]

- "Two objects are similar if they are related to similar objects"
- Defined as the unique solution to:

Decay factor between 0 and 1

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$

Set of incoming edges into a

- Computed by iterating to convergence
- Initialization to s(a, b) = 1 if a=b and 0 otherwise

Katz Score

- Two objects are similar if they are connected by shorter paths

$$s(a,b) = \sum_{l=1}^{\infty} \beta^l \cdot |\underbrace{\text{paths}^{\langle l \rangle}(a,b)}|$$

Set of paths between a and b of length exactly ℓ

Decay factor between 0 and 1

- Since expensive to compute, often use approximate Katz, assuming some max path length of k

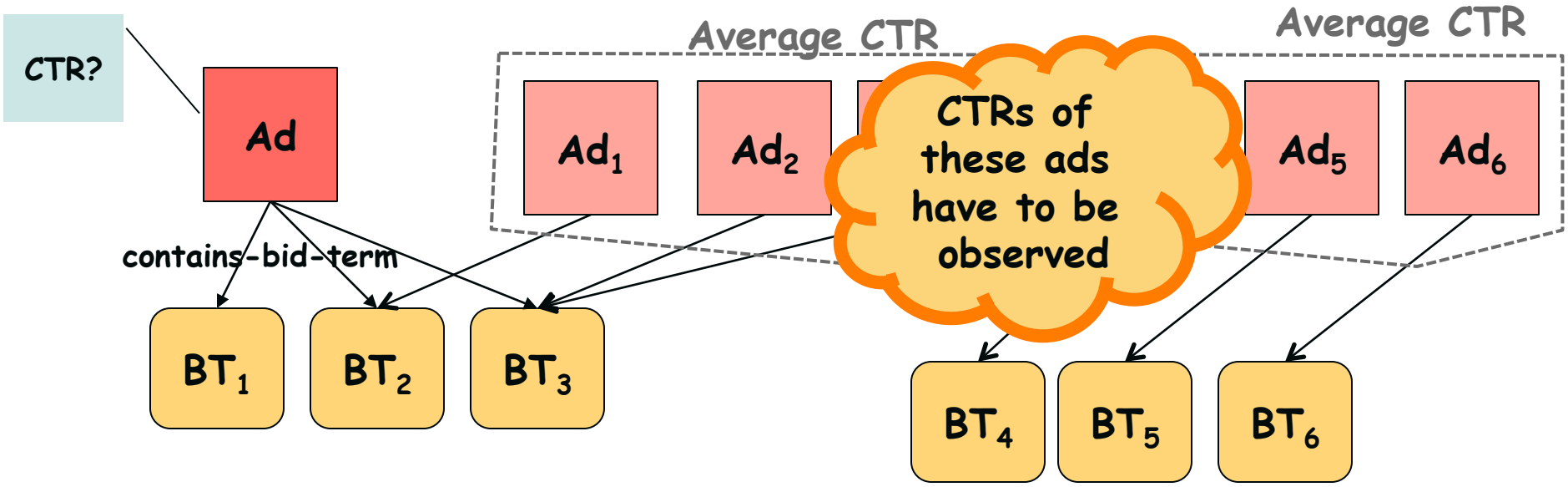
Relational Classifiers: Pros

- Efficient
  - Can handle large amounts of data
    - Features can often be pre-computed ahead of time
  - One of the most commonly-used ways of incorporating relational information
- Flexible
  - Can take advantage of well-understood classification/ regression algorithms

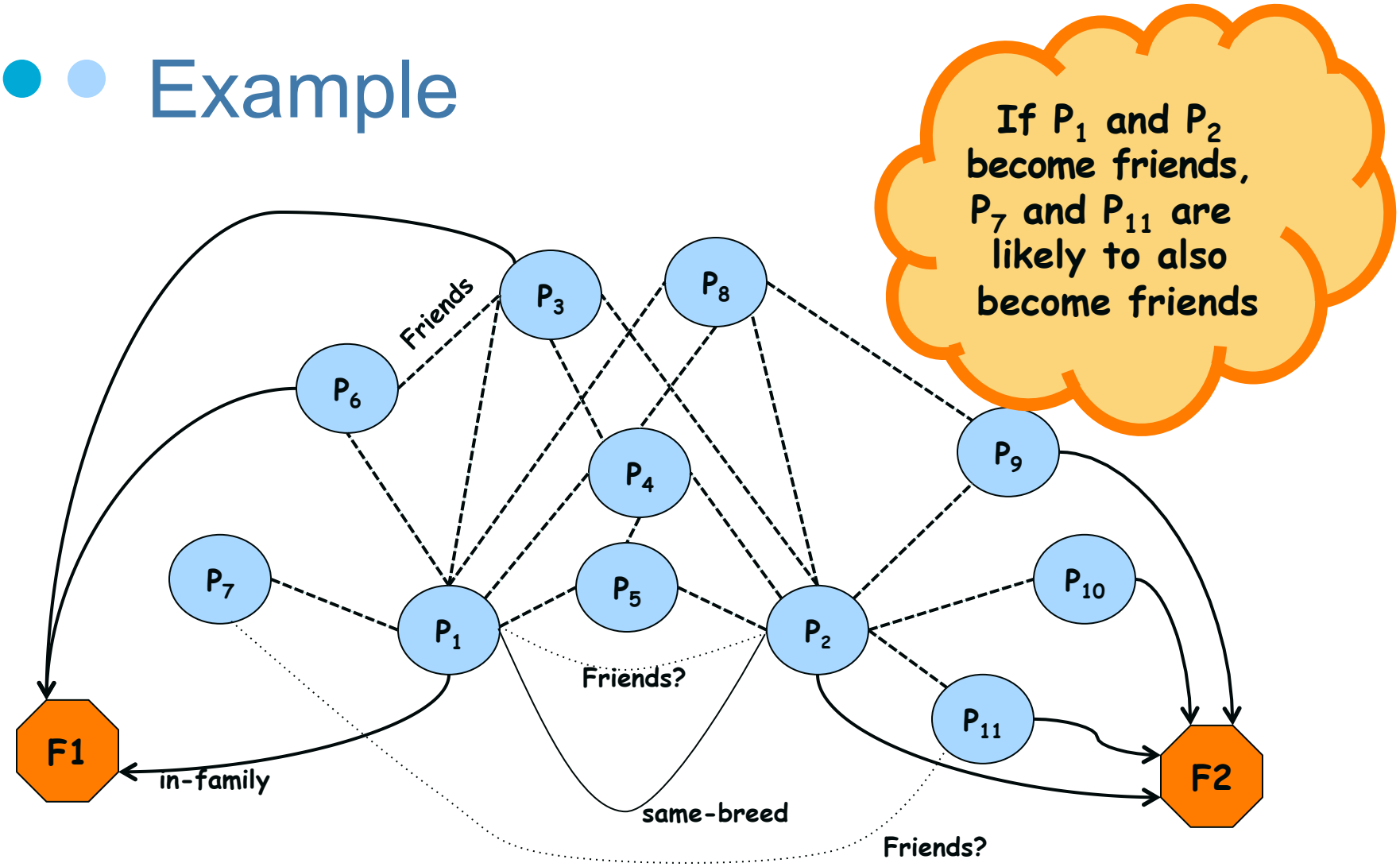
Relational Classifiers: Cons

- Relational features cannot be based on attributes or relations that are being predicted
  - For example :

Example



Example



Relational Classifiers: Cons

- Relational features cannot be based on attributes or relations that are being predicted

We'll see a general approach to doing this

... but a couple of caveats:

- This can be overcome by proceeding in two rounds:
  1. Make predictions using only observed features and relations
  2. Make predictions using observed features and relations and predictions of unobserved ones from round 1.
- Inductive Logic Programming techniques for learning "recursive" clauses exist that allow the model to prove further examples from previously proven ones

Relational Classifiers: Cons

- Relational features cannot be based on attributes or relations that are being predicted

- Cannot impose global constraints on joint assignments
  - For example, when inferring a hierarchy of individuals, we may want to enforce constraint that it is a tree

Road Map

- Relational Classifiers
- Collective Classification
- Advanced SRL Models

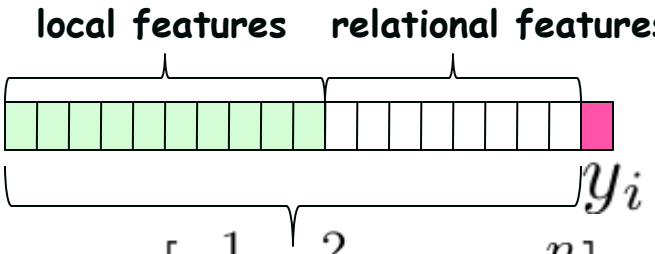
Road Map

- Relational Classifiers
- Collective Classification
  - Definition
  - Case Studies
  - Key Idea: Iteration / Propagation
- Advanced SRL Models

Collective Models

- Disadvantages of relational classifiers can be addressed by making collective predictions
  - Can help correct errors
  - Can coordinate assignments to satisfy constraints
- Collective models have been widely studied. Here we present a derivation based on extending flat relational representations

Towards Collective Models 1



$$x_i = [x_i^1, x_i^2, \ldots, x_i^n]$$

To simplify matters, suppose y_i is binary.

$$P(y_i = 1; \theta) = \frac{\exp(\theta \cdot x_i)}{\exp(\theta \cdot x_i) + 1}$$

If we use Logistic regression

$$f_i(y_i) = [x_i^1 \mathbf{1}[y_i = 1], x_i^2 \mathbf{1}[y_i = 1], \ldots, x_i^n \mathbf{1}[y_i = 1]]$$

Let's make the features be a function of y_i

$$P(y_i = 1; \theta) = \frac{\exp(\theta \cdot f_i(y_i))}{\exp(\theta \cdot f_i(y_i = 1) + \exp(\theta \cdot f_i(y_i = 0))}$$

$$= \frac{\exp(\theta \cdot f_i(y_i))}{\sum_{y_i' \in \{0,1\}} \exp(\theta \cdot f_i(y_i'))}$$

Same thing with new features

Towards Collective Models 2

This trivial transformation makes it easy to generalize to features that are functions of more than one y_i, thus forcing the model to make collective decisions…

$$P(y_i;\theta) = \frac{\exp(\theta \cdot f_i(y_i))}{\sum_{y_i' \in \{0,1\}} \exp\left(\theta \cdot f_i(y_i')\right)}$$

normalize

$y = [y_1, y_2, \ldots, y_m]$

becomes

$$P(y;\theta_L,\theta_G) = \frac{\exp\left(\sum_i \theta_L \cdot f_i(y_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i,y_j)\right)}{\sum_{y'} \exp\left(\sum_i \theta_L \cdot f_i(y_i') + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i',y_j')\right)}$$

local and global parameter vectors

sum local features, as before

compute probability over joint assignment to all instances

sum features that are functions of more than one y_i; in general, could be more than 2

Towards Collective Models 2

This trivial transformation makes it easy to generalize to features that are functions of more than one y_i, thus forcing the model to make collective decisions…

$$P(y_i;\theta) = \frac{\exp(\theta \cdot f_i(y_i))}{\sum_{y'_i \in \{0,1\}} \exp\left(\theta \cdot f_i(y'_i)\right)}$$

becomes $i, j \in E$

$$P(y;\theta_L,\theta_G) = \frac{\exp\left(\sum_i \theta_L \cdot f_i(y_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i,y_j)\right)}{\sum_{y'} \exp\left(\sum_i \theta_L \cdot f_i(y'_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y'_i,y'_j)\right)}$$

Often not all possible pairs are considered, but just ones that are related.

Towards Collective Models 3

- Good news:
  - Now we have a way of coordinating the assignments to the query attributes/relationships
- Bad news:
  - Looks like we have to enumerate over all possible joint assignments

$$P(y;\theta_L,\theta_G) = \frac{\exp\left(\sum_i \theta_L \cdot f_i(y_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i,y_j)\right)}{\sum_{y'} \exp\left(\sum_i \theta_L \cdot f_i(y'_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y'_i,y'_j)\right)}$$

- … but there are ways around this

Collective Classification

- Variety of algorithms
  - Iterated conditional modes [Besag 1986; …]
  - Relaxation labeling [Rosenfeld et al. 1976; …]

- Make coherent joint assignments by iterating over individual decision points, changing them based on current assignment to related decision points

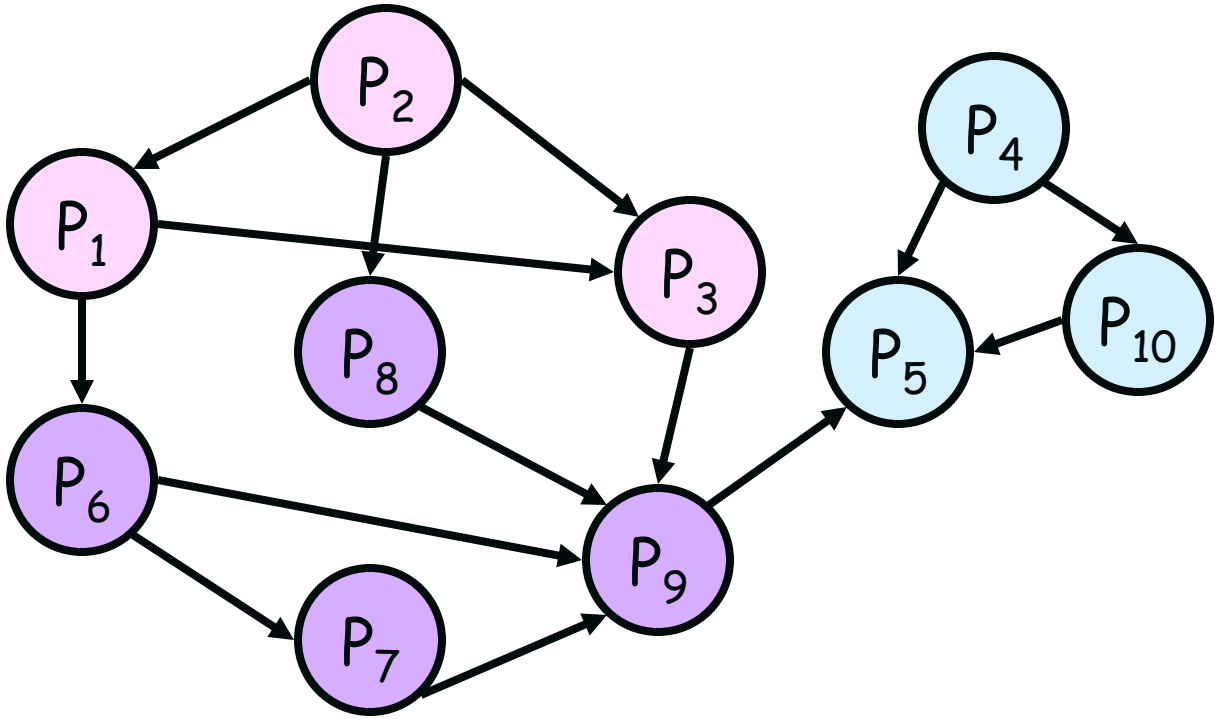
Iterative Classification Algo. (ICA)

[Neville & Jensen, SRL00; Lu & Getoor, ICML03]

- Extends flat relational models by allowing relational features to be functions of predicted attributes/ relations of neighbors
- At training time, these features are computed based on observed values in the training set
- At inference time, the algorithm iterates, computing relational features based on the current prediction for any unobserved attributes
  - In the first, bootstrap, iteration, only local features are used

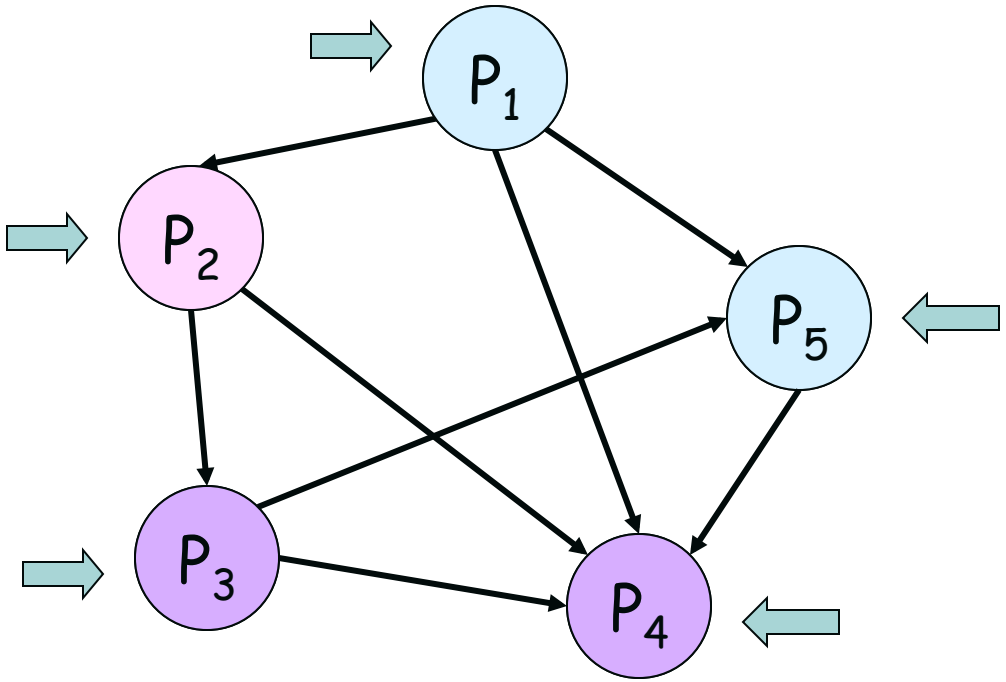
ICA: Learning

- label set:



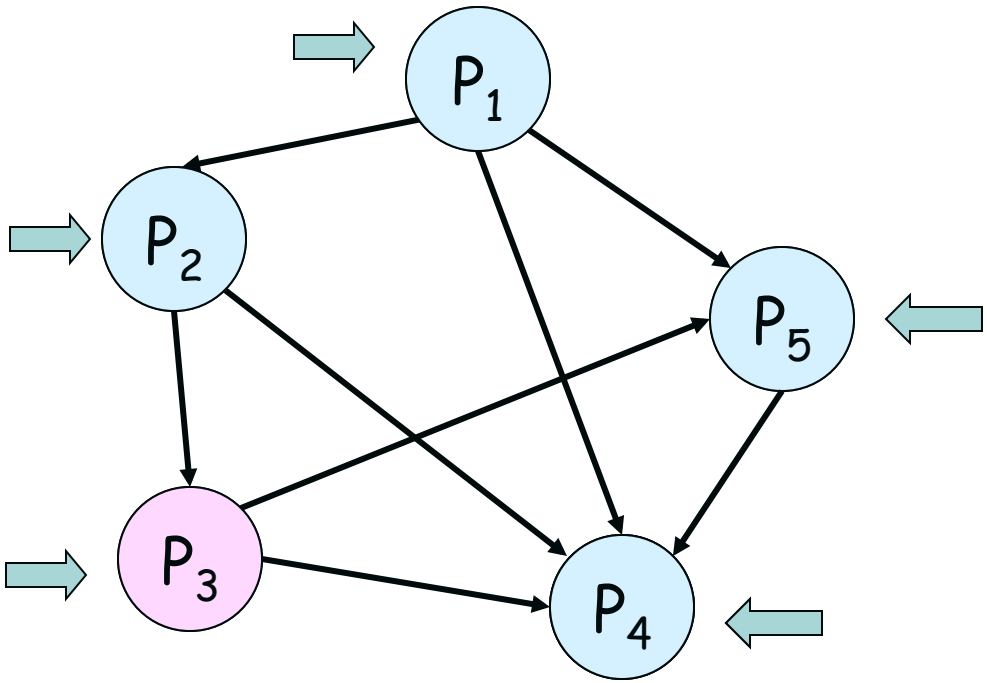
Learn models (local and relational) from fully labeled training set

ICA: Inference (1)



Step 1: Bootstrap using entity attributes only

ICA: Inference (2)



Step 2: Iteratively update the category of each entity, based on related entities' categories

ICA Summary

- Simple approach for collective classification
- Variations:
  - Propagate probabilities, rather than mode (see also Gibbs Sampling later)
  - Batch vs. Incremental updates
  - Ordering strategies
- Related Work:
  - Cautious Inference [McDowell et al., JMLR09]
  - Weighted neighbor [Macskassy, AAAI07]
  - Active Learning [Bilgic et al., TKDD09, ICML10]

Road Map

- Relational Classifiers
- Collective Classification
- Advanced SRL Models
  - Background: Graphical Models
  - Key Ideas: Par-factor graphs
  - Languages

Road Map

- Relational Classifiers
- Collective Classification
- Advanced SRL Models
  - Background: Graphical Models 
  - Key Ideas: Par-factor graphs
  - Languages

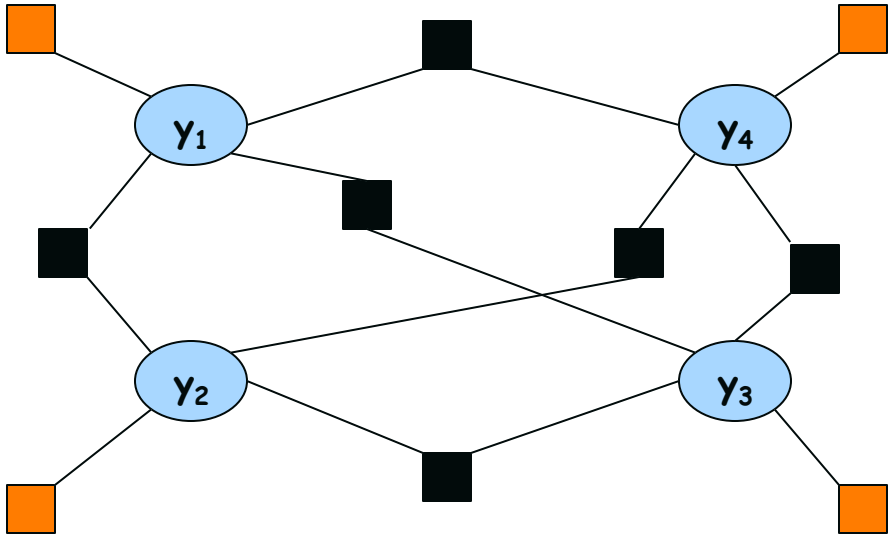
Factor Graphs

- Let's go back to our joint probability distribution:

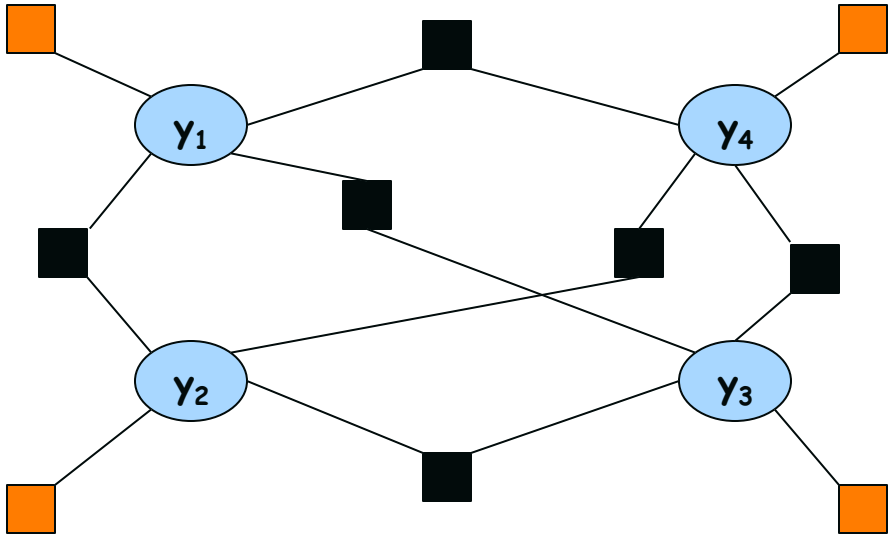
$$y = [y_1, y_2, \ldots, y_m]$$

$$P(y; \theta_L, \theta_G) = \frac{\exp\left(\sum_i \theta_L \cdot f_i(y_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y_i, y_j)\right)}{\sum_{y'} \exp\left(\sum_i \theta_L \cdot f_i(y'_i) + \sum_{i,j} \theta_G \cdot f_{i,j}(y'_i, y'_j)\right)}$$

- The factor graph representation of this is:



Factor Graphs



- Each ■ represents

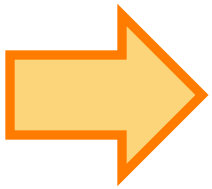
$$\exp(\theta_L \cdot f_i(y_i))$$

- Each ■ represents

$$\exp(\theta_G \cdot f_{i,j}(y_i, y_j))$$

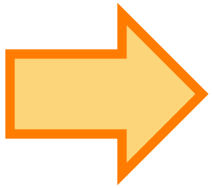
More Generally…

- Factors can be functions of any number of variables



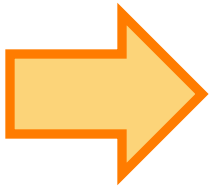
However, to keep the model compact, we want to keep factors small. In the worst case, the number of parameters needed by a factor is exponential in the number of variables of which it is a function.

- Not all pairs of variables have to share a factor



In fact, we want to avoid having variables share factors unless there truly is a dependence between them.

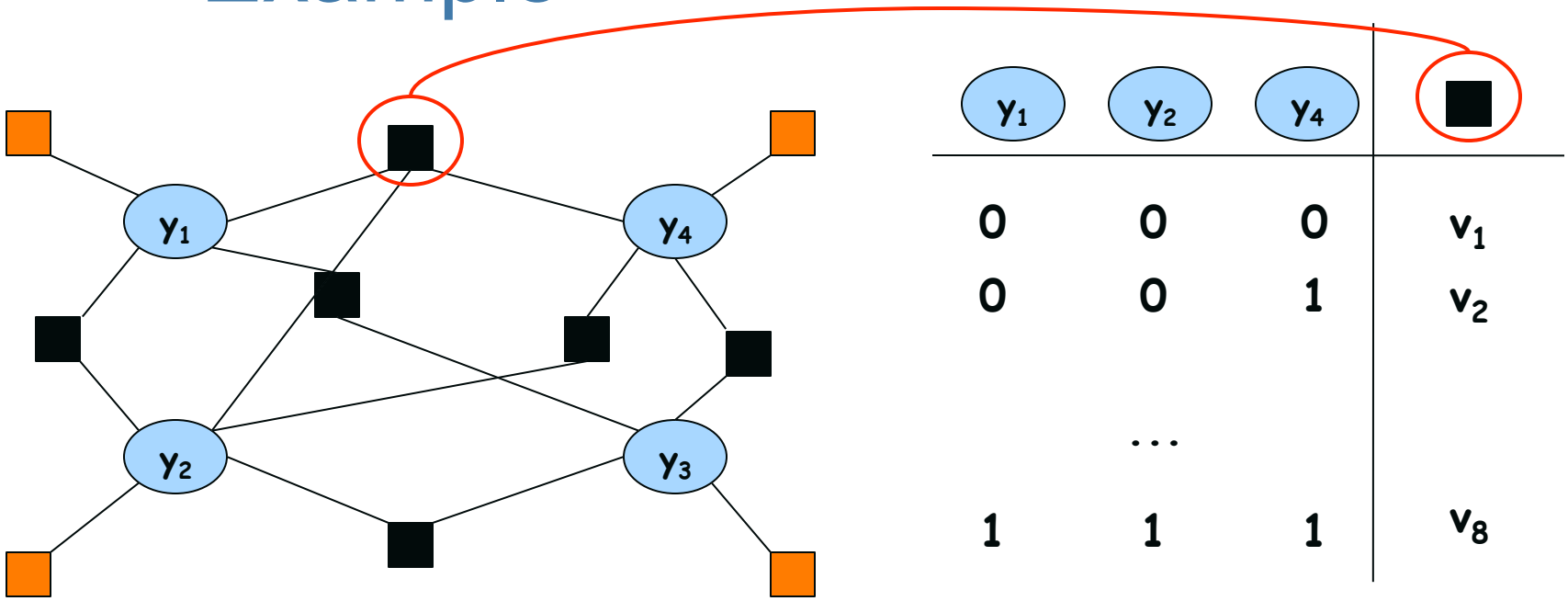
- Factors can be computed by any function that returns a strictly positive value



The log-linear representation is convenient and has nice properties.

Example

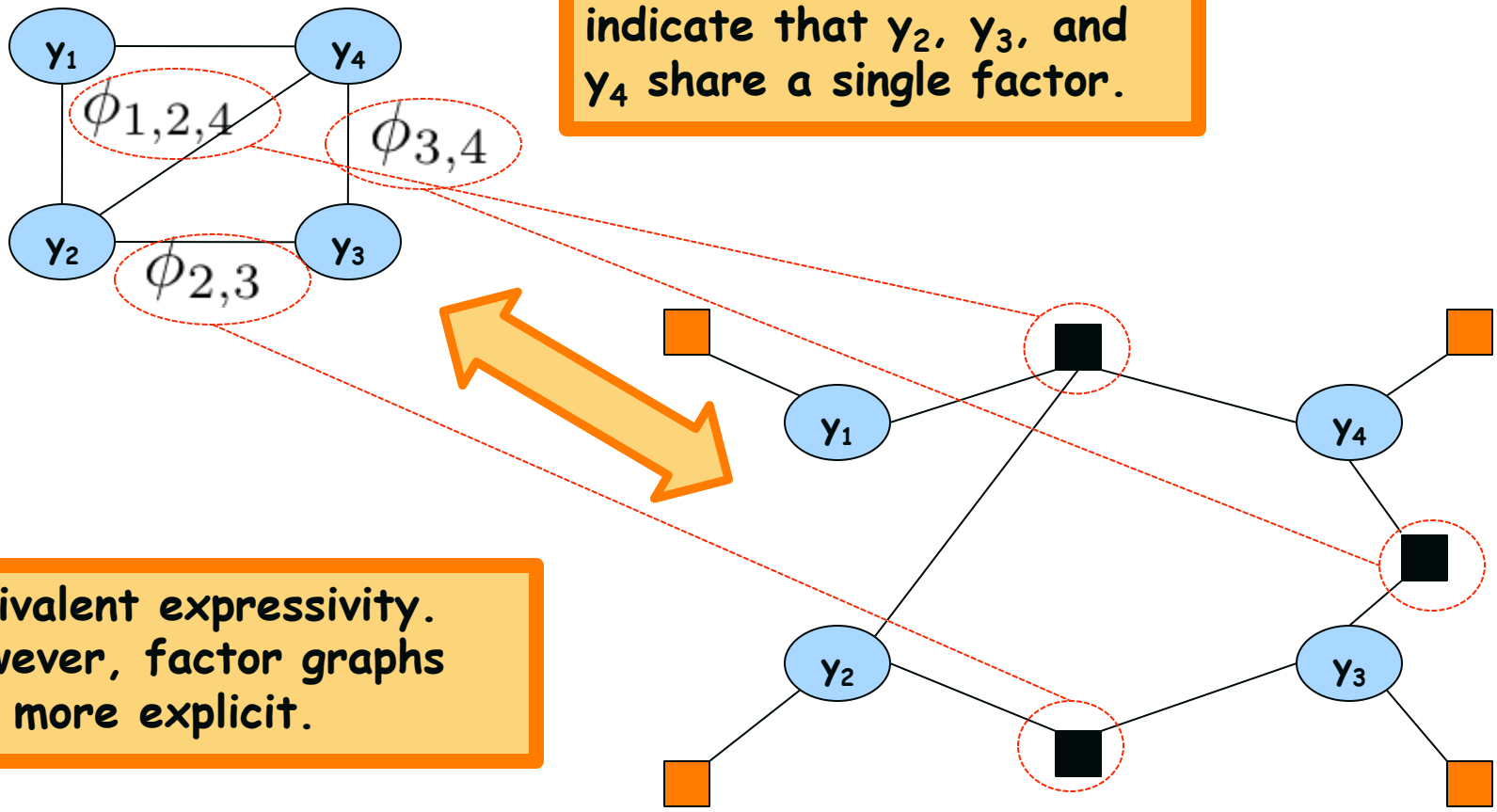
... in the most basic representation:



Now, y_3 is conditionally independent of y_1, given y_2 and y_4.

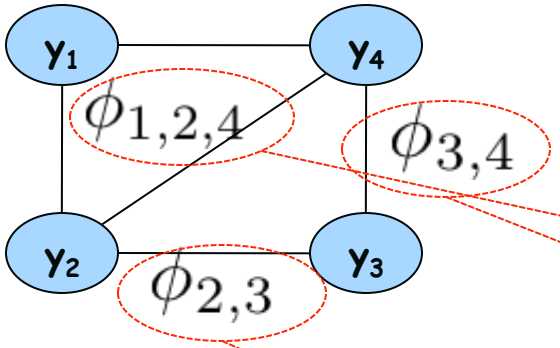
Markov Nets

- Markov networks (aka Markov random fields) can be viewed as special cases of factor graphs:



Markov Nets Continued

- Factors are called potential functions
- Viewed as functions that ensure compatibility between assignments to the nodes



Variables participating in shared potential functions form cliques in the graph.

For example, in the Ising Model the possible assignments are {-1, + 1}, and one has:

$$\phi_{i,j} = \exp(\theta_{i,j} y_i y_j)$$

Positive, or ferromagnetic, $\theta_{i,j}$ encourages neighboring nodes to have the same assignment.

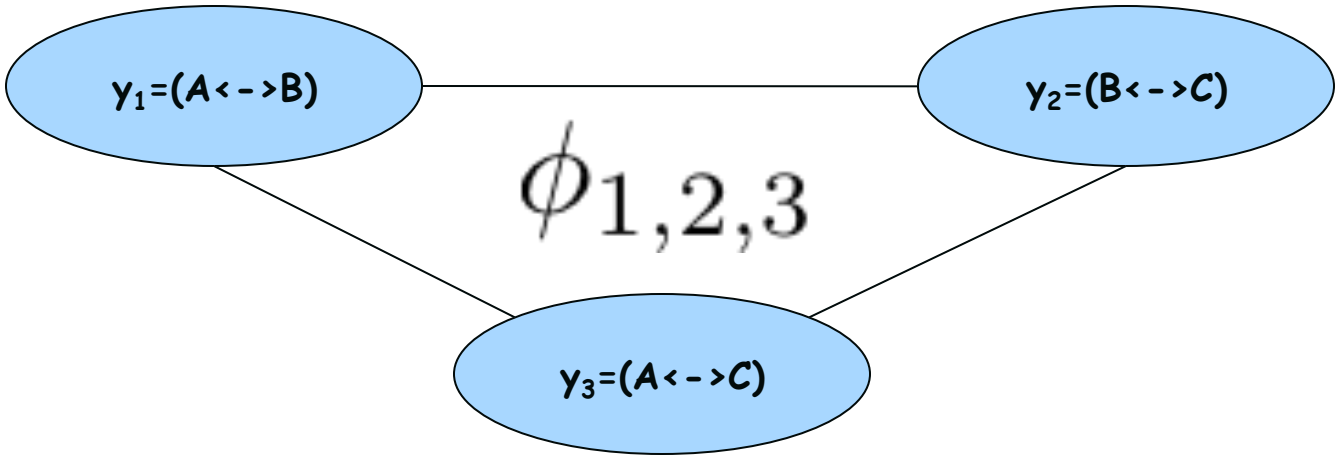
Negative, or anti-ferromagnetic, $\theta_{i,j}$ encourages contrasting assignment.

Markov Nets: Transitivity

- How to encode transitivity?

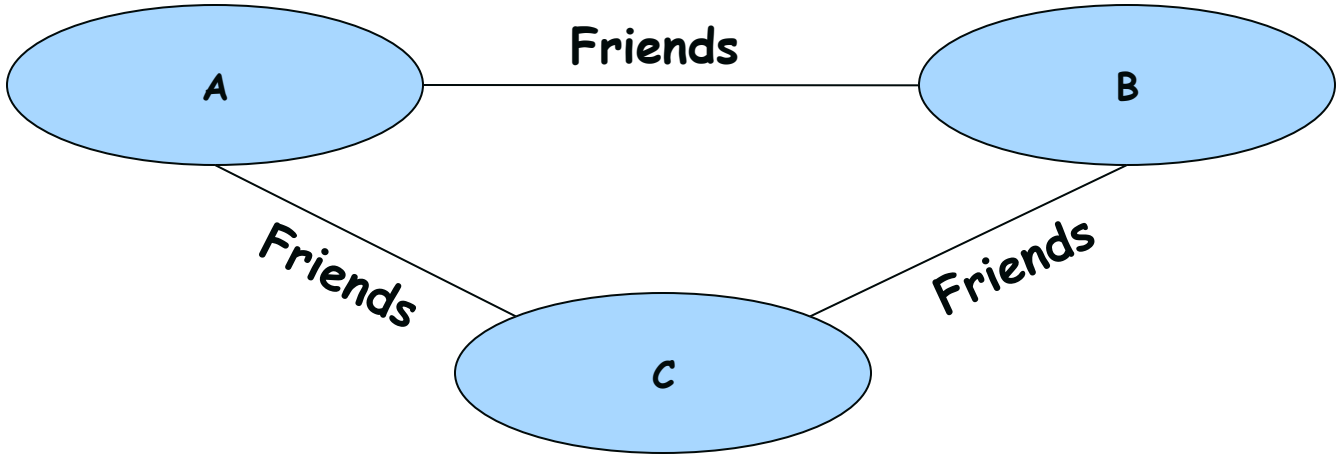
Want to say: If *A* is friends with B and
B is friends with *C*, then
A is friends with *C*.
For all permutations of the letters.

- Model as a Markov net with a node for each decision, connecting dependent decisions in cliques
- Possible assignments: 1 (friends), 0 (not friends)



Quick Aside: Two Kinds of Graphs

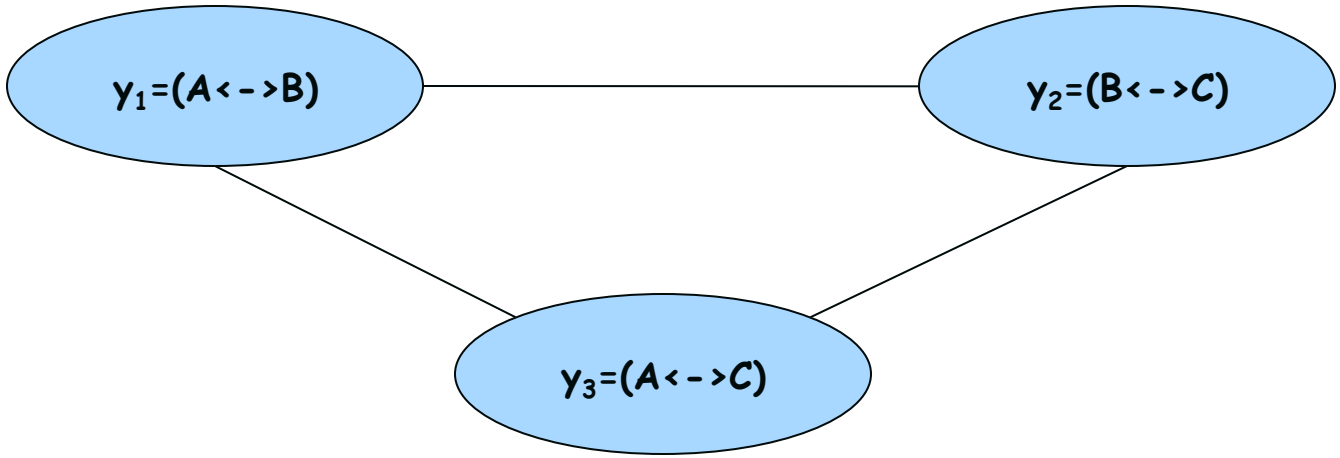
We often draw social networks like this:



Relational Graph:

- Nodes represent entities
- Edges represent relationships

... not to be confused with a Markov net:



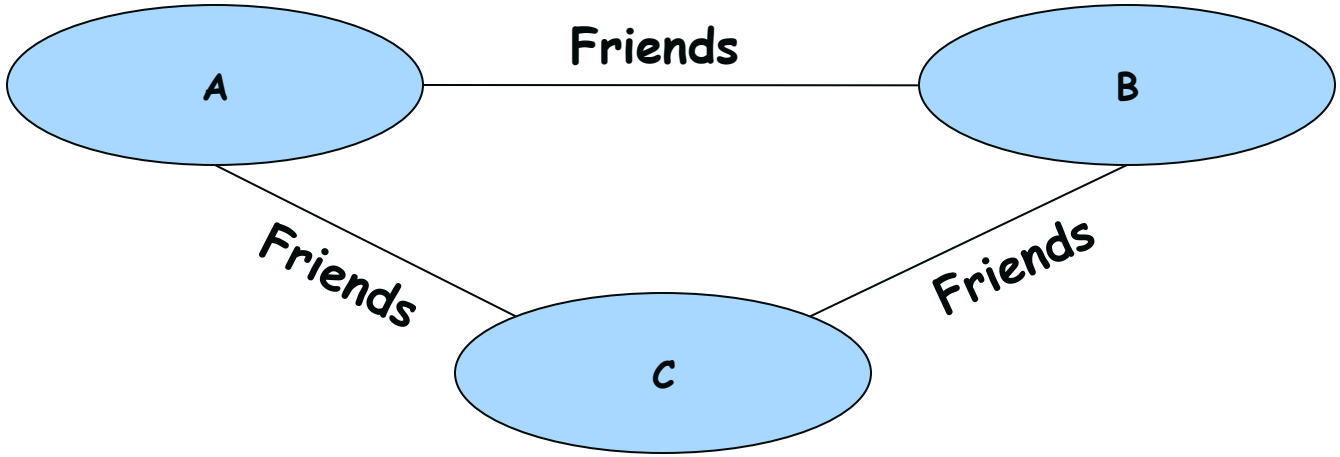
Markov Net:

- Nodes represent decisions
- Edges represent dependencies between decisions

Quick Aside: T

Since here we are trying to infer the presence of a relationship, our Markov Net has a node for each possible edge in the Relational graph.

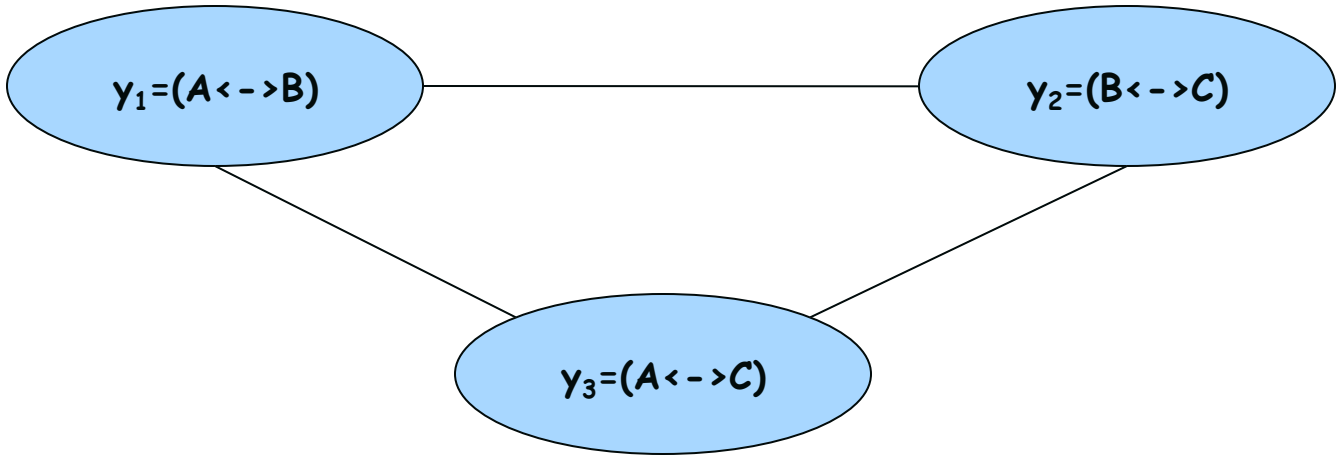
In Part I we were drawing social r



Relational Graph:

- Nodes represent entities
- Edges represent relationships

... not to be confused with a Markov net:



Markov Net:

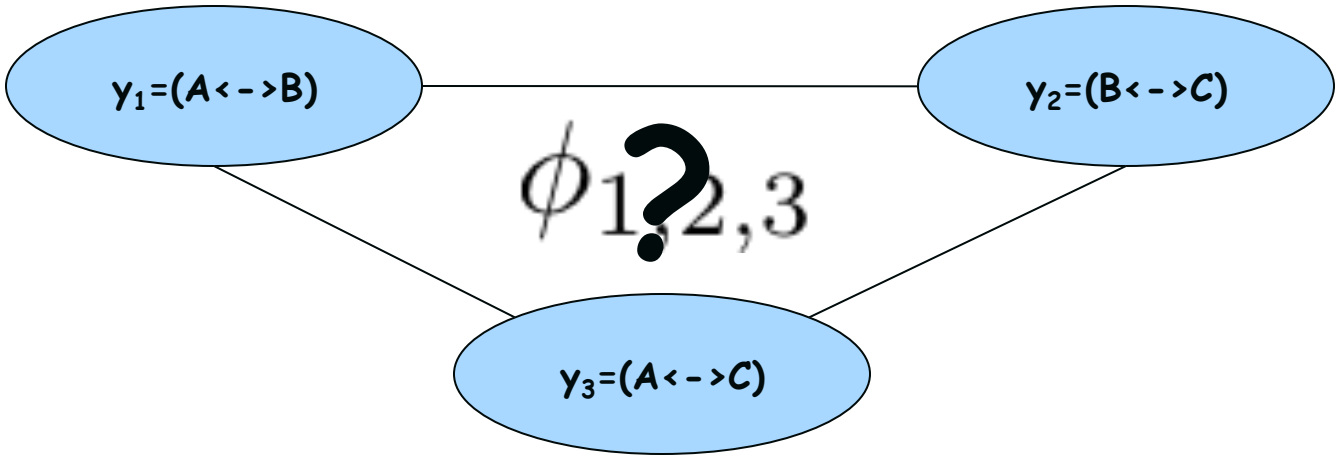
- Nodes represent decisions
- Edges represent dependencies between decisions

Markov Nets: Transitivity

- How to encode transitivity?

> **Want to say: If *A* is friends with B and**
> **B is friends with *C*, then**
> ***A* is friends with *C*.**
> **For all permutations of the letters.**

- Model as a Markov net with a node for each decision, connecting dependent decisions in cliques
- Possible assignments: 1 (friends), 0 (not friends)



Markov Nets: Transitivity

... one possibility

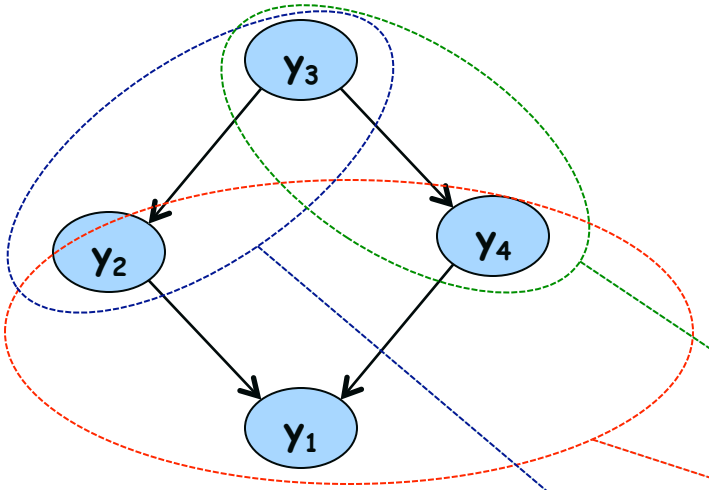
y_1=(A<->B)	y_2=(B<->C)	y_3=(A<->C)	$\phi_{1,2,3}$
0	0	0	 e^0
0	0	1	 e^0
0	1	0	 e^0
0	1	1	✗ $e^{-\theta}$
1	0	0	 e^0
1	0	1	✗ $e^{-\theta}$
1	1	0	✗ $e^{-\theta}$
1	1	1	✔ e^{θ}

Variants

If A and B are enemies and
B and C are enemies, then
A and C are friends.
For all permutations of the letters.

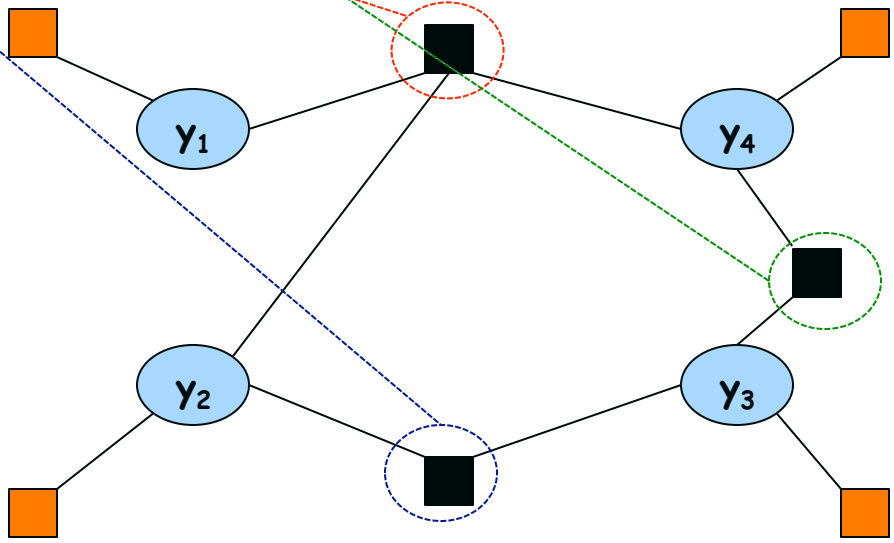
y_1=(A<->B)	y_2=(B<->C)	y_3=(A<->C)	$\phi_{1,2,3}$
0	0	0	✗ $e^{-\theta}$
0	0	1	✔ e^{θ}
0	1	0	✔ e^{θ}
0	1	1	≈ e^{0}
1	0	0	✔ e^{θ}
1	0	1	≈ e^{0}
1	1	0	≈ e^{0}
1	1	1	≈ e^{0}

Bayesian Nets

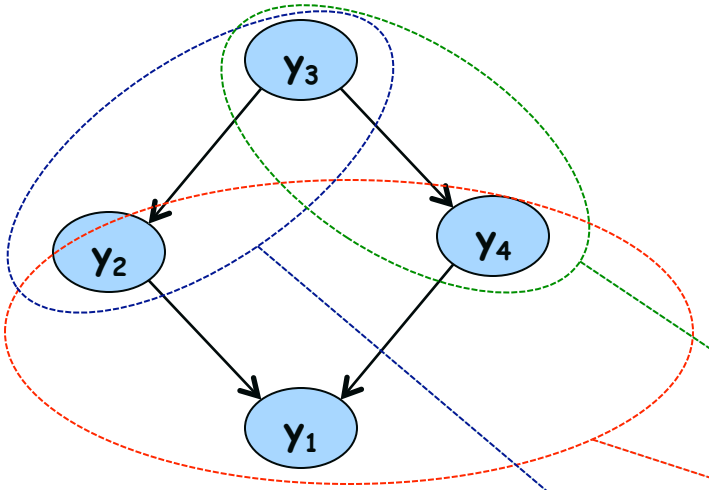


Going the other way requires ensuring acyclicity.

To cast a Bayesian net as a factor graph, include a factor as a function of each node and its parents.



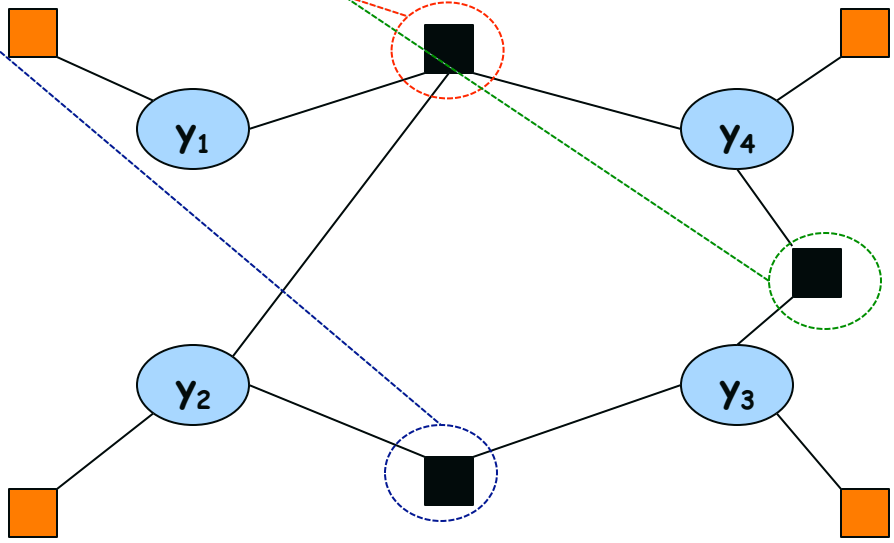
Bayesian Nets Continued



Here the factors take the shape of conditional probability tables, giving, for each configuration of assignments to the parents, the distribution over assignments to the child.

Automatically normalized!

To cast a Bayesian net as a factor graph, include a factor as a function of each node and its parents.

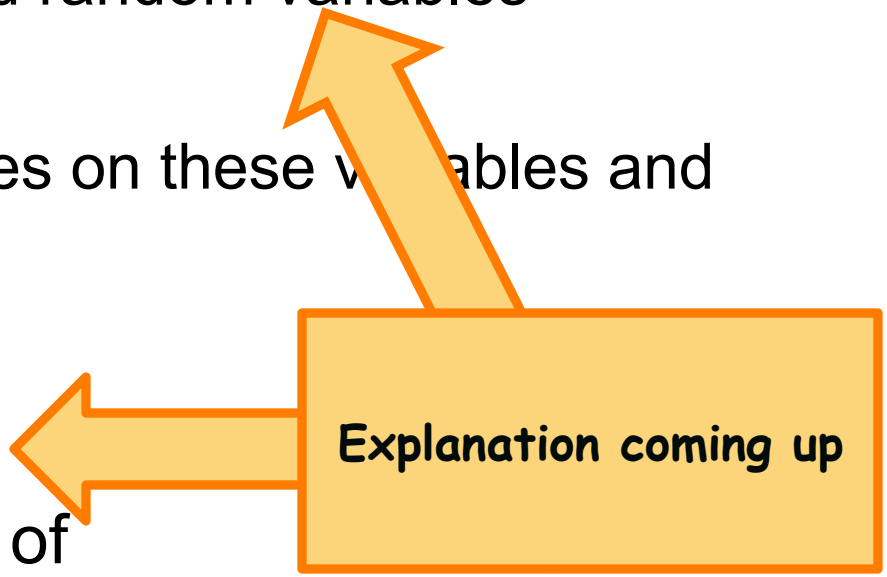


Road Map

- Relational Classifiers
- Collective Classification
- Advanced SRL Models
  - Background: Graphical Models
  - Key Ideas: Par-factor graphs
  - Languages

Par-factor Graphs

- Factor graphs with parameterized factors
  - Terminology introduced by [Poole, IJCAI03]
- A par-factor is defined as the triple
  - $\mathcal{A}$: set of parameterized random variables
  - f : function that operates on these variables and evaluates to > 0
  - $\mathcal{C}$: set of constraints
- A par-factor graph is a set of par-factors



Parameterized Random Vars

- Can be viewed as a blueprint for manufacturing random variables
- For example:
  - Let A and B be variables, then

    A<->B

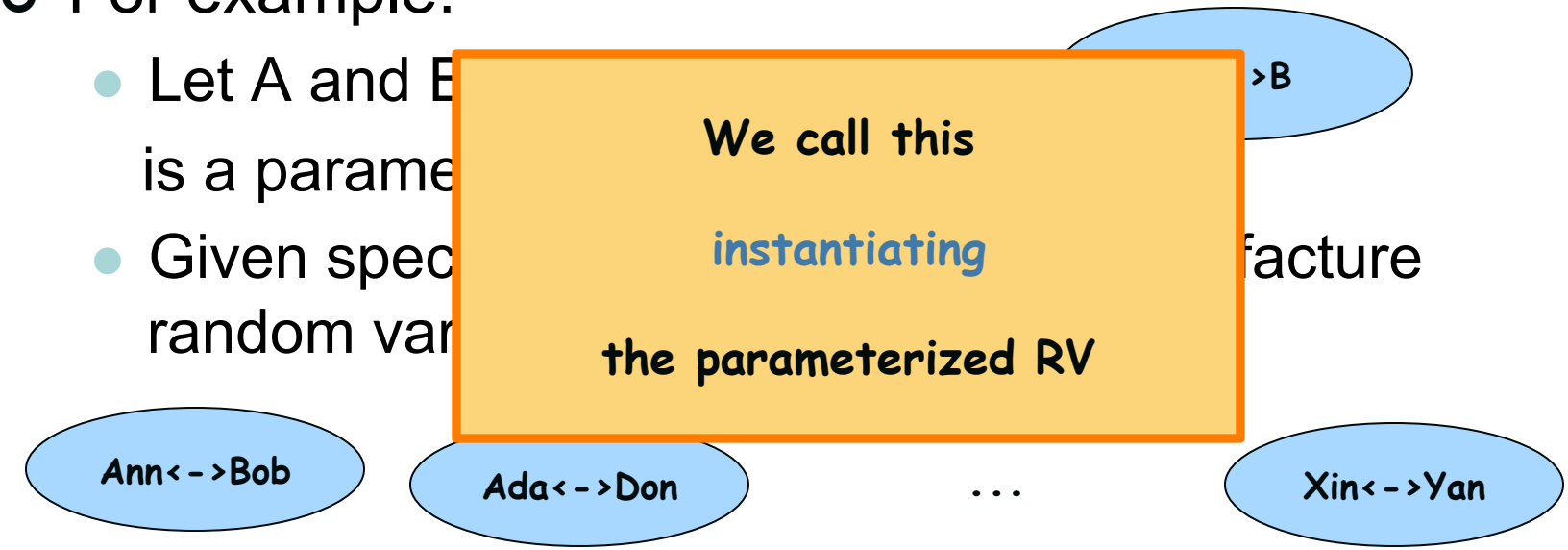
    is a parameterized random variable.
  - Given specific individuals, we can manufacture random variables from it:

Ann<->Bob Ada<->Don . . . Xin<->Yan

So far we are not assuming a particular language for expressing par-RVs.

Parameterized Random Vars

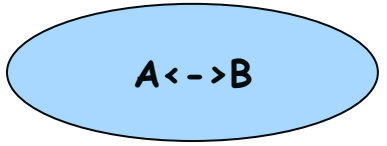
- Can be viewed as a blueprint for manufacturing random variables
- For example:
  - Let A and B
    is a parame
  - Given spec facture random var



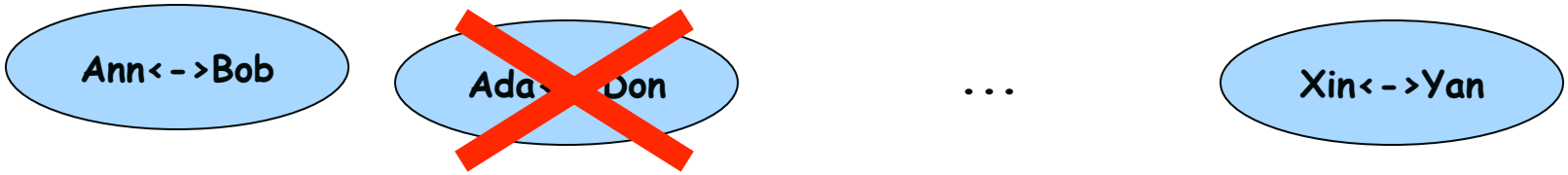
So far we are not assuming a particular language for expressing par-RVs.

Constraints

- The constraints in set $\mathcal{C}$ govern how par-RVs can be instantiated
- For example, one constraint for our par-RV could be that B ≠ Don

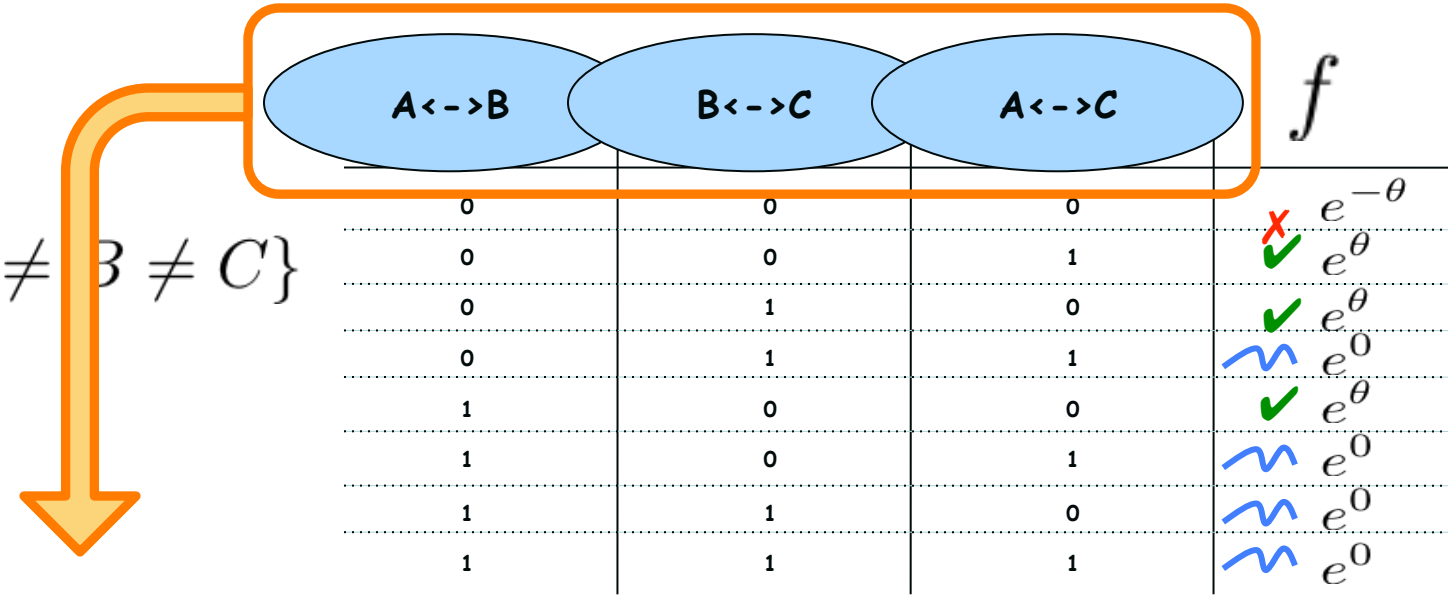


- With this constraint, the possible instantiations are



Transitivity Par-factor

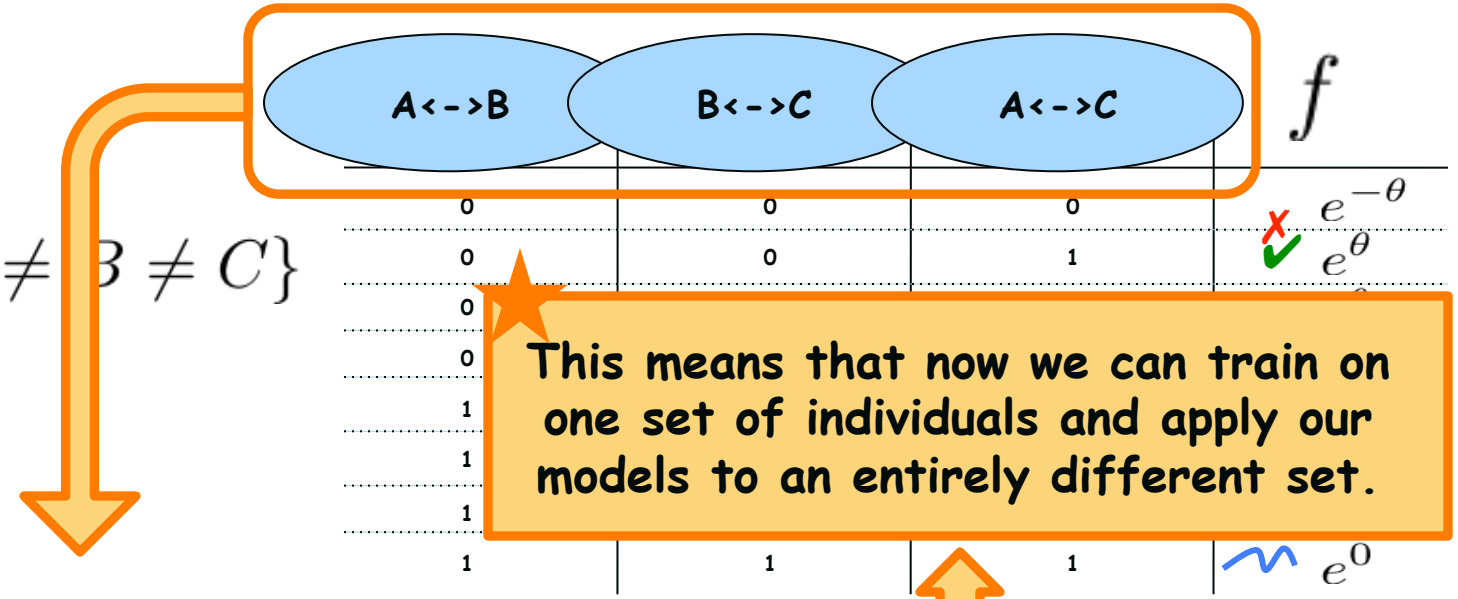
- $\mathcal{A} =$ {A<->B, B<->C, A<->C}
- f can be defined as before
- $\mathcal{C} = \{A \neq B \neq C\}$



However, whereas before these referred to the potential friendships of specific individuals, now they refer to variables, i.e. to people in general

Transitivity Par-factor

- $\mathcal{A} =$ {A<->B, B<->C, A<->C}
- f can be defined as before
- $\mathcal{C} = \{A \neq B \neq C\}$

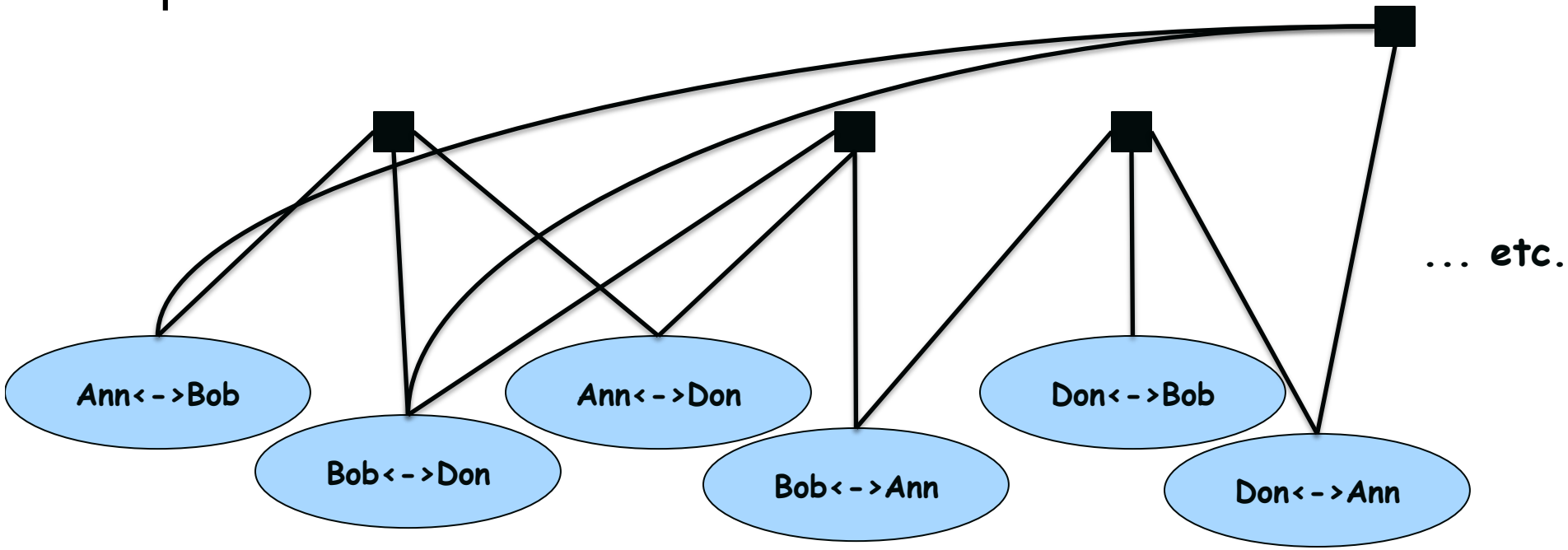


This means that now we can train on one set of individuals and apply our models to an entirely different set.

However, whereas before these referred to the potential friendships of specific individuals, now they refer to variables, i.e. to people in general

Transitivity Par-Factor Instantiated

- To instantiate a par-factor, we need a set of individuals: **Ann, Bob, Don**
- Then we consider all possible instantiations of the par-RVs with these individuals:



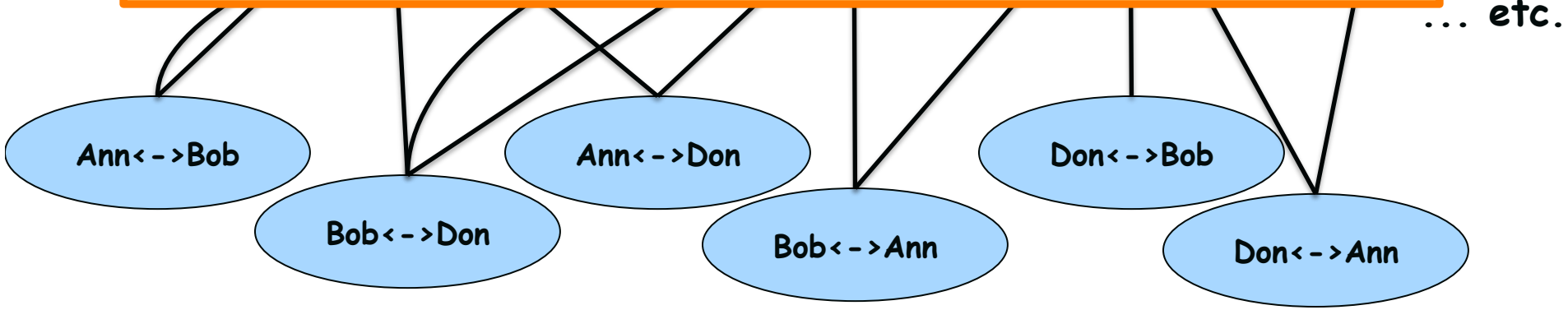
Transitivity Par-Factor Instantiated

- To instantiate a par-factor, we need a set of

Moral of the story:
So much power can be dangerous!

➔ Starting with just 3 individuals, we've ended up with a huge and densely connected graph (for n individuals, we would get $O(n^3)$ factors)

➔ Inference becomes very problematic



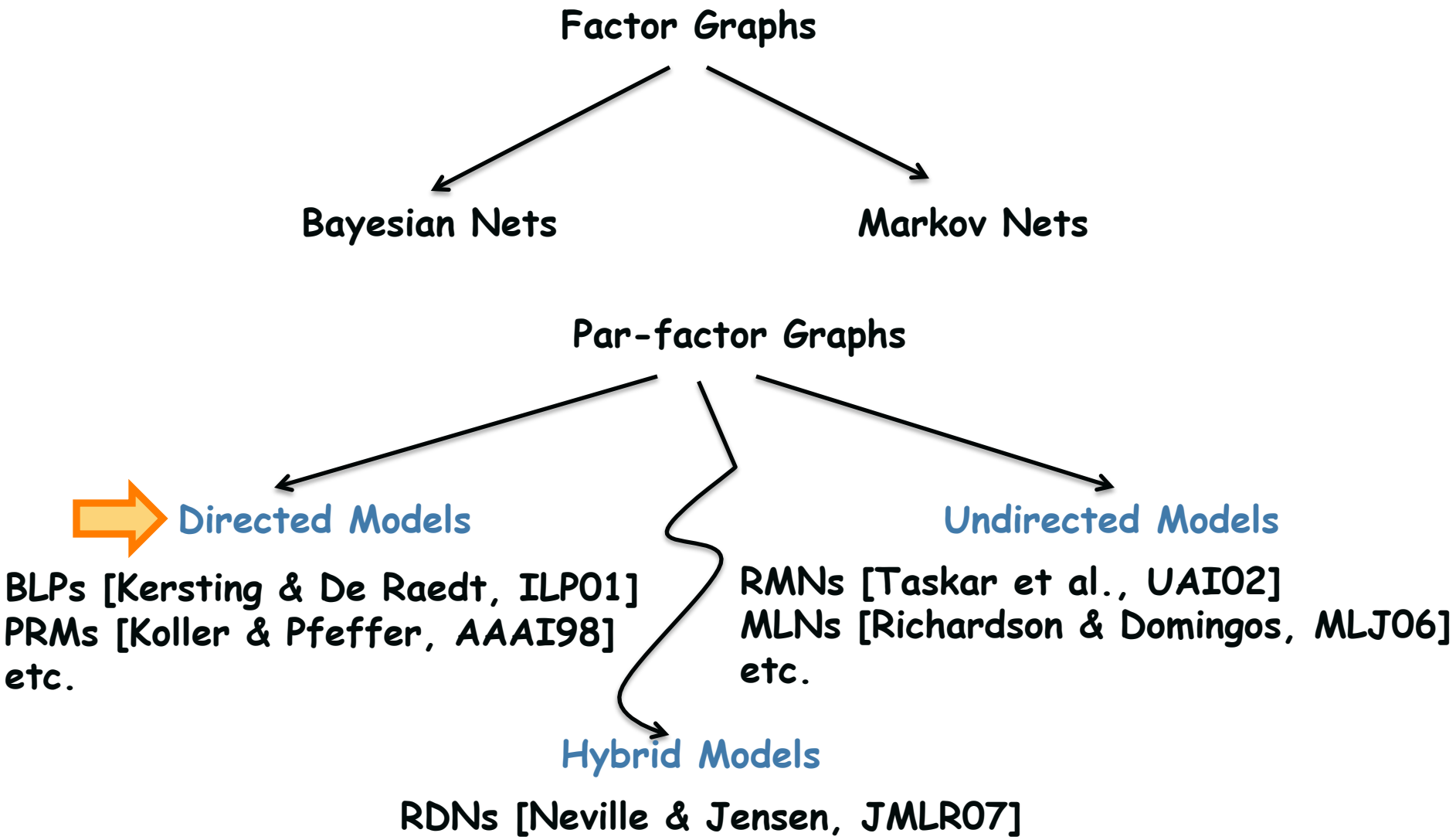
Managing our Power

- Constraints
  - One way of keeping the factor graph size manageable is by imposing appropriate constraints on permitted instantiations
- Par-factor size
  - More par-RVs per par-factor translate into more RVs per factor
- When defining a par-factor, it is important to think:
  - How many instantiations will this par-factor have?
  - How many RVs per instantiation?
- This is easier said than done
  - Will discuss more

Recap So Far

- Extended factor graphs to allow for convenient parameter tying
  - Parameter learning: an extension of parameter learning in Bayesian/Markov nets
  - Inference: instantiate the par-factors and perform inference as before
- Are we done?
  - We still do not have a convenient language for specifying the function part of a par-factor
  - A wide range of languages have been introduced and studied in the field of statistical relational learning (SRL). Here we review just a few

SRL Road Map

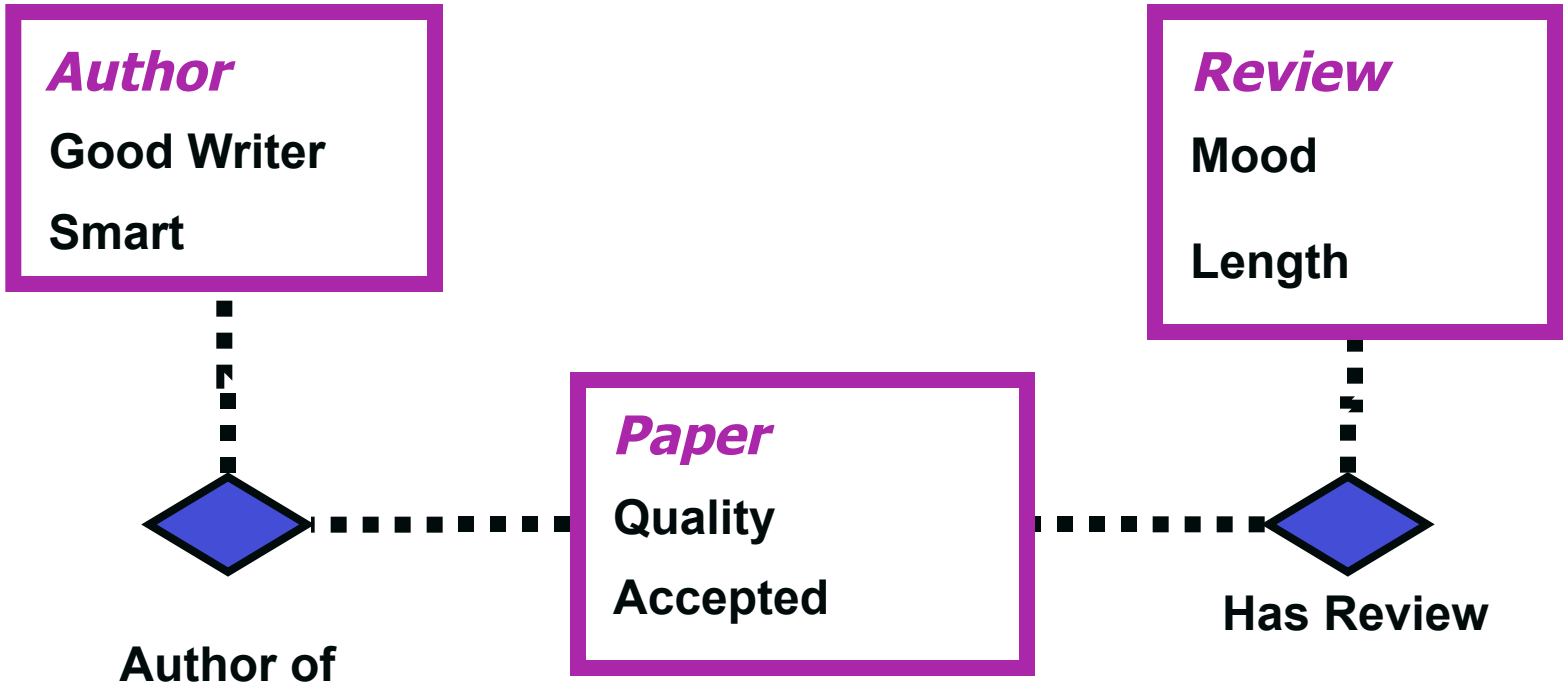


Directed Models

- Bayesian logic programs (BLPs)
  - Based on first-order logic
  - [Kersting & De Raedt, ILP01]

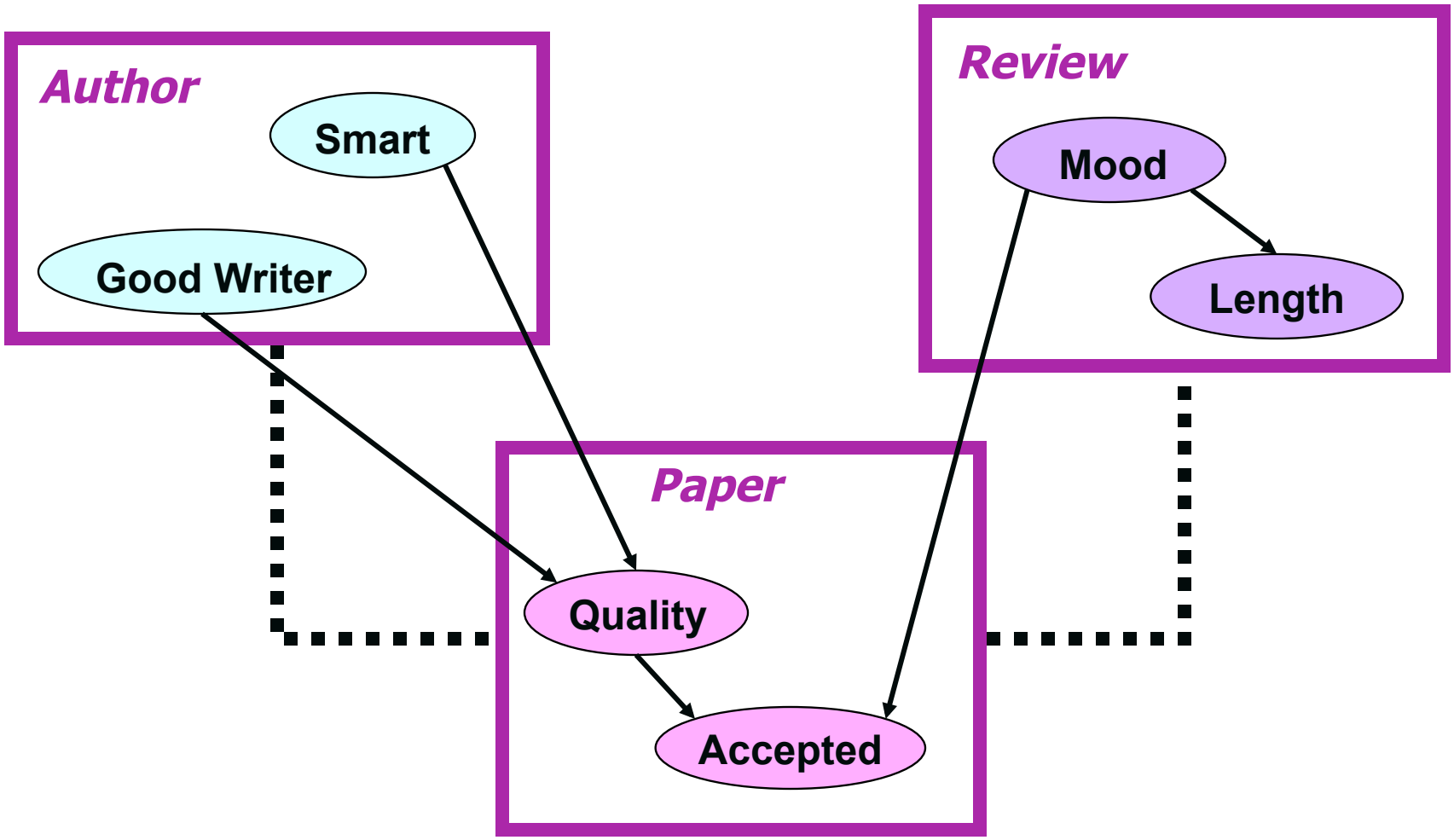
- Probabilistic relational models (PRMs)
  - Using an object-oriented, frame-based representation
  - [Koller & Pfeffer, AAAI98]

Relational Schema

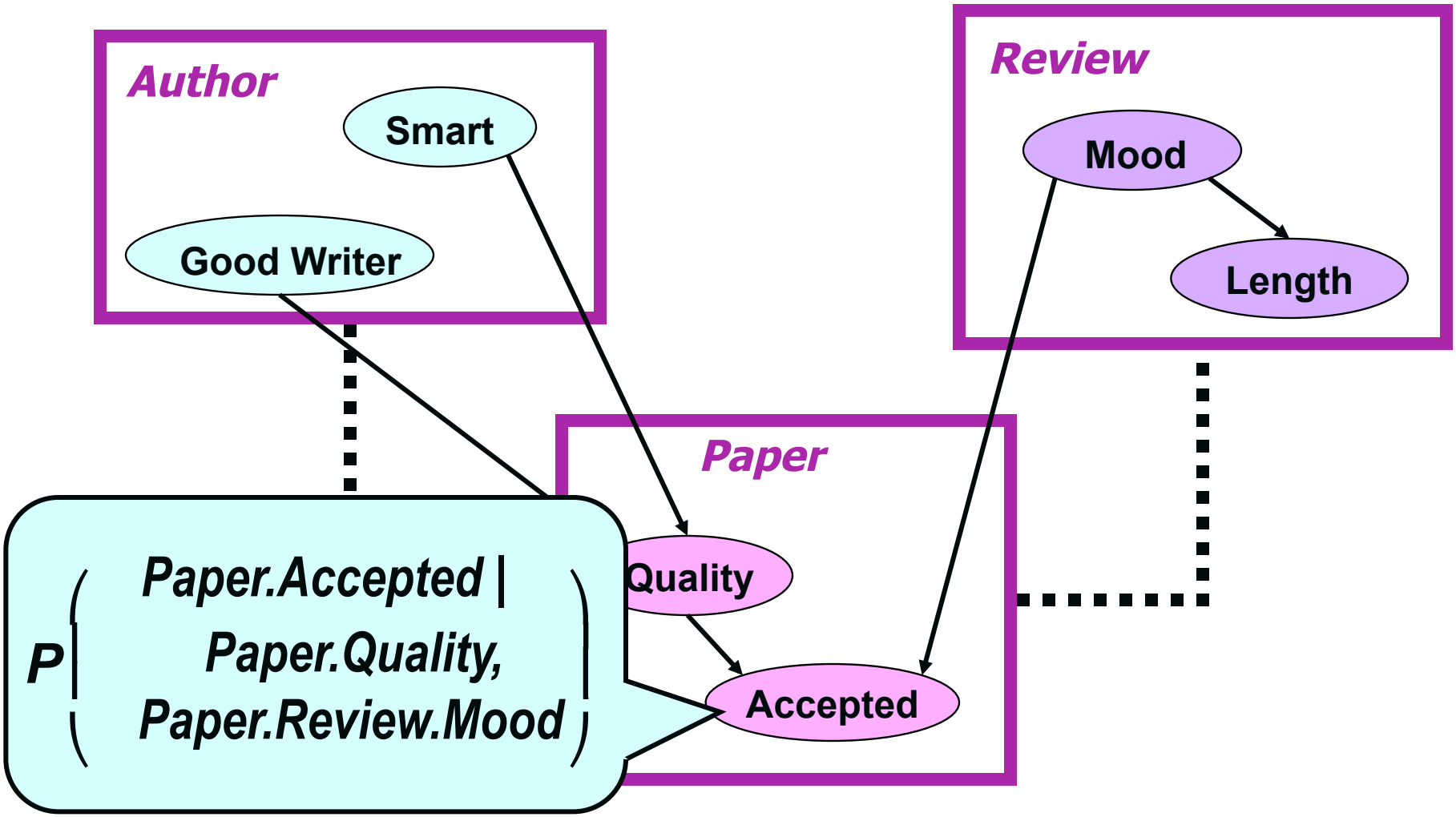


- **Describes the types of objects and relations in the database**

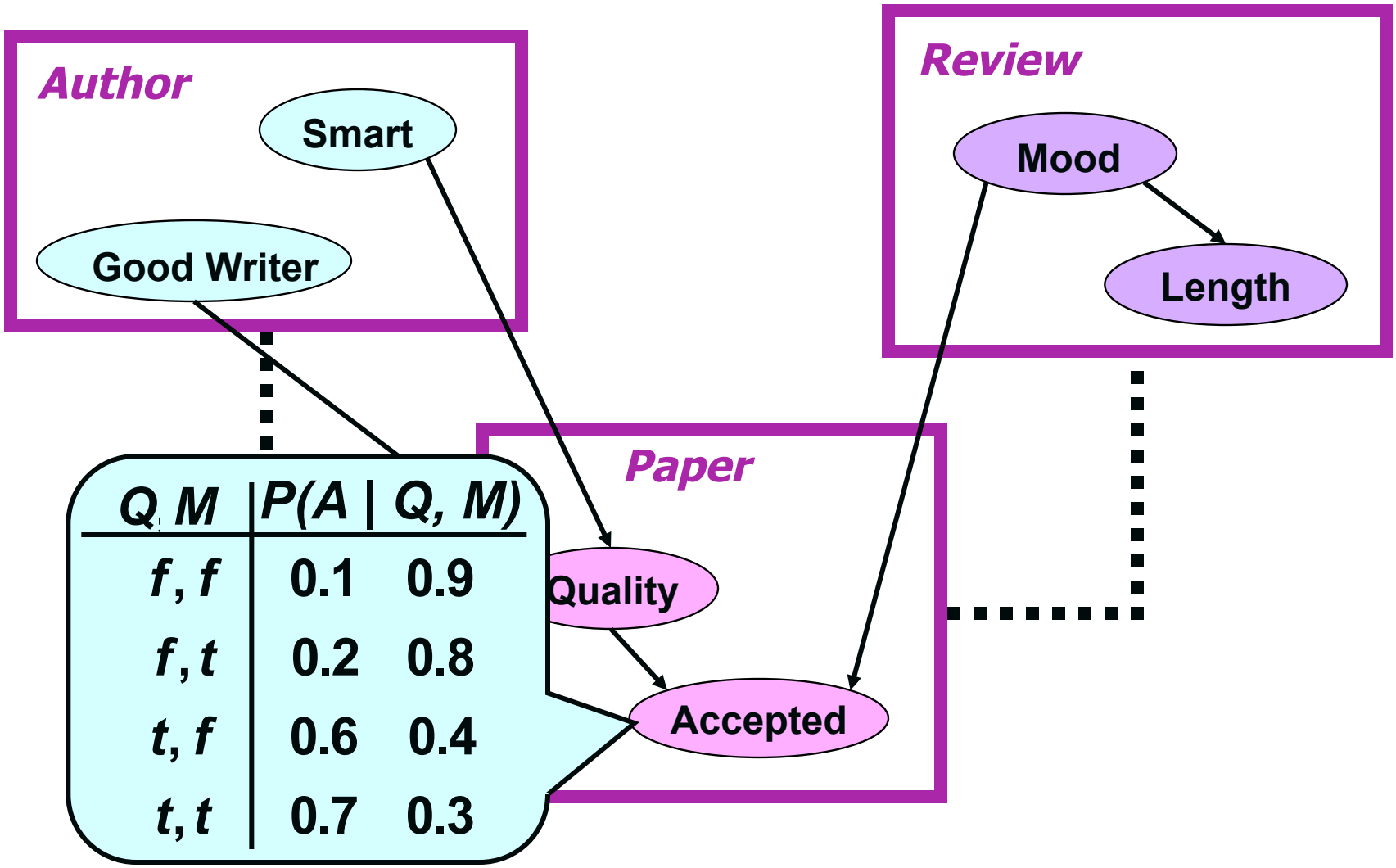
Probabilistic Relational Model



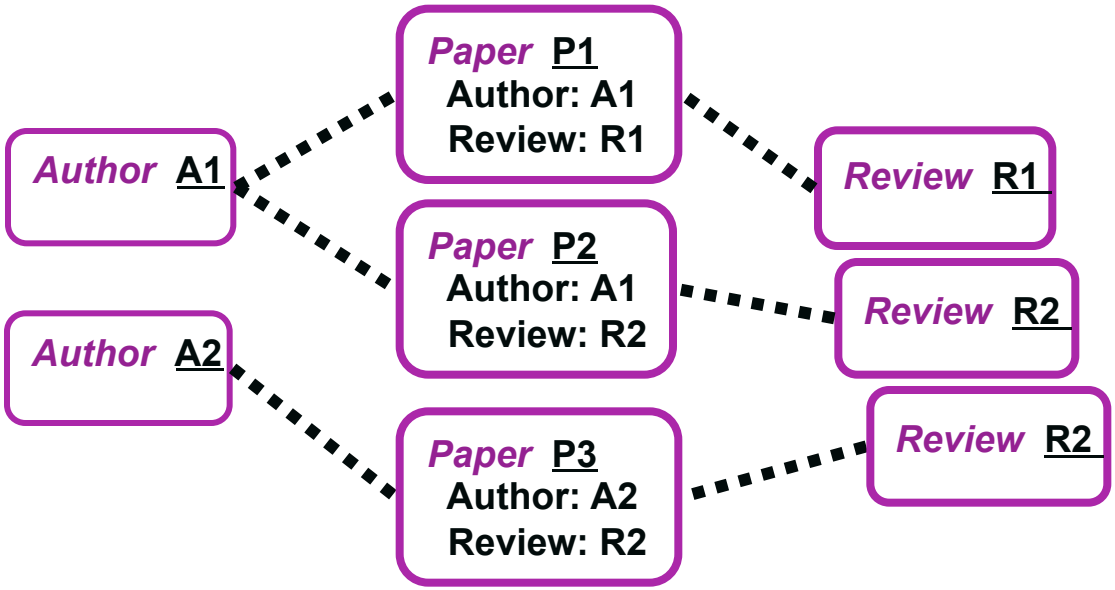
Probabilistic Relational Model



Probabilistic Relational Model



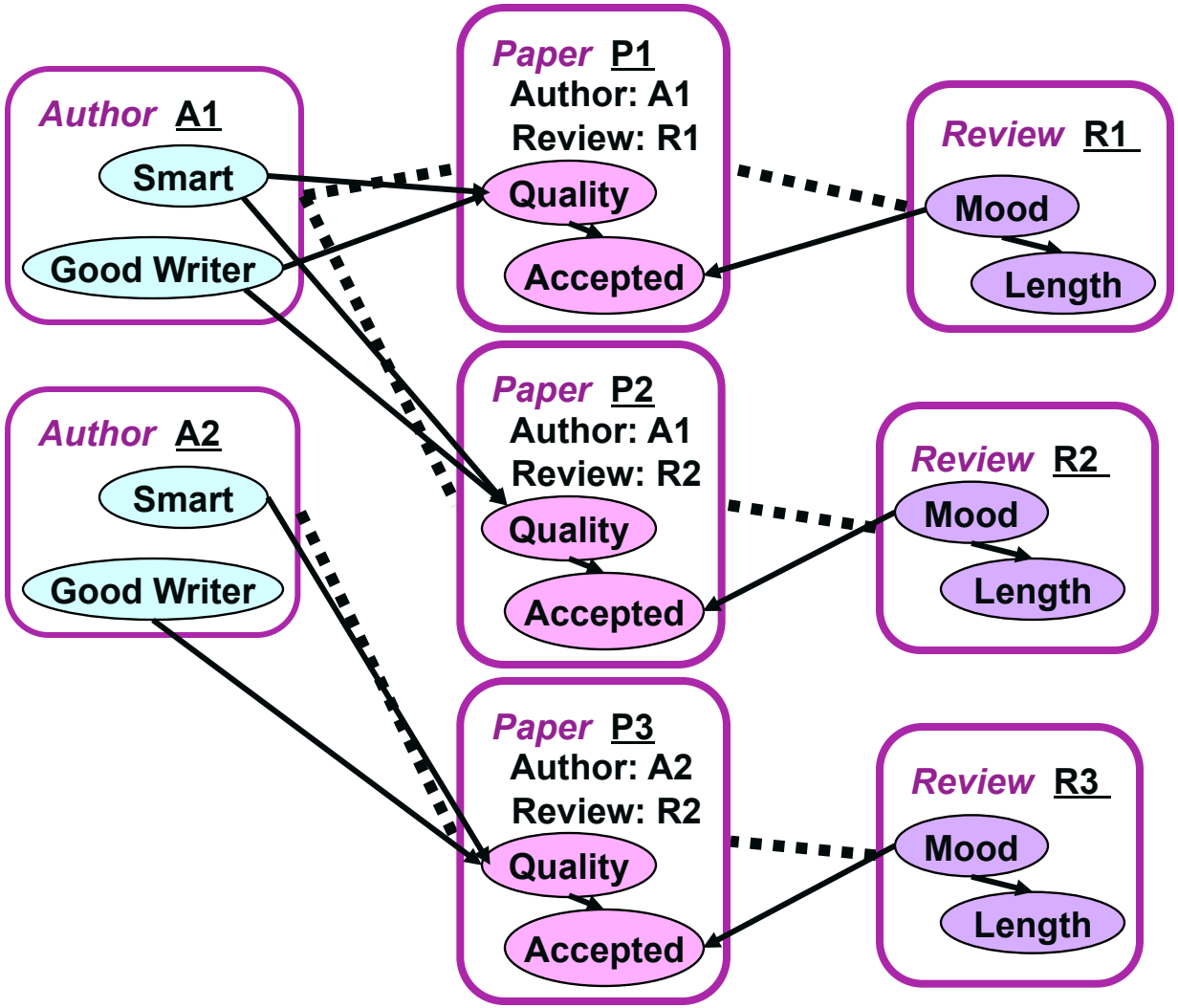
Relational Skeleton



Fixed relational skeleton σ:

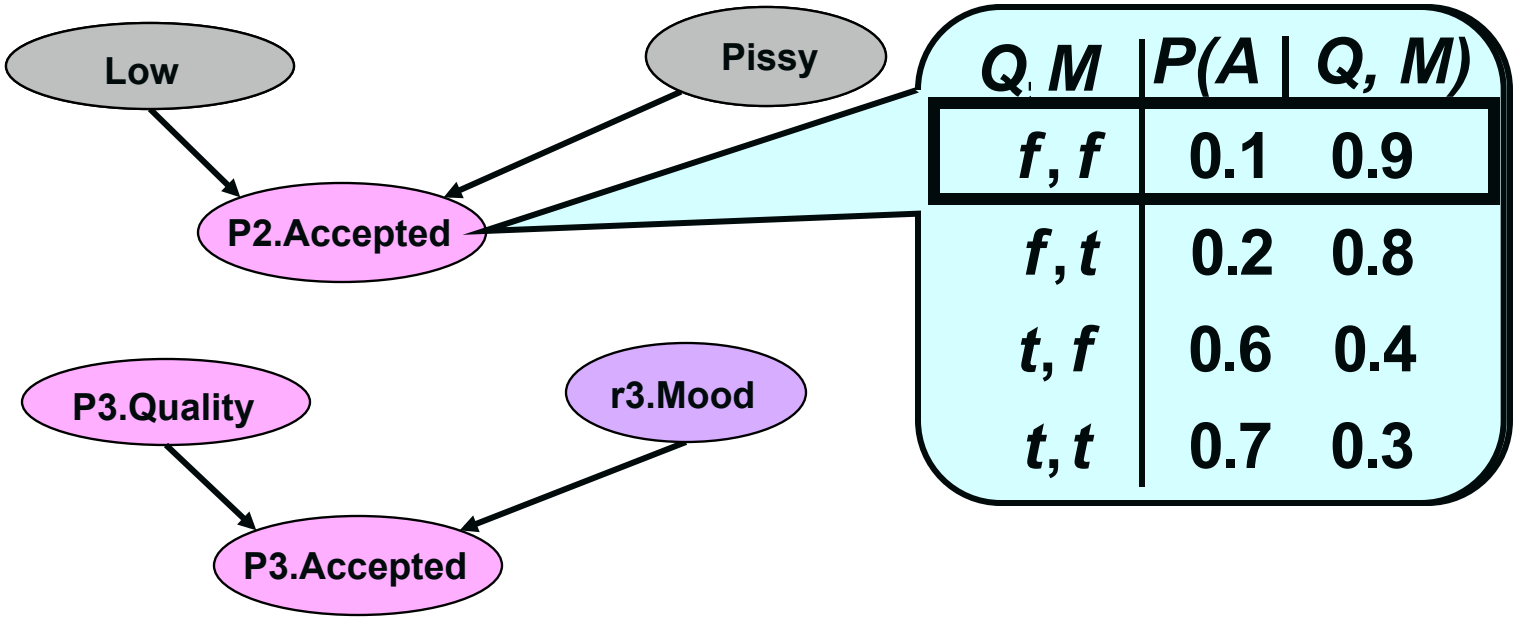
- **set of objects in each class**
- **relations between them**

PRM w/ Attribute Uncertainty

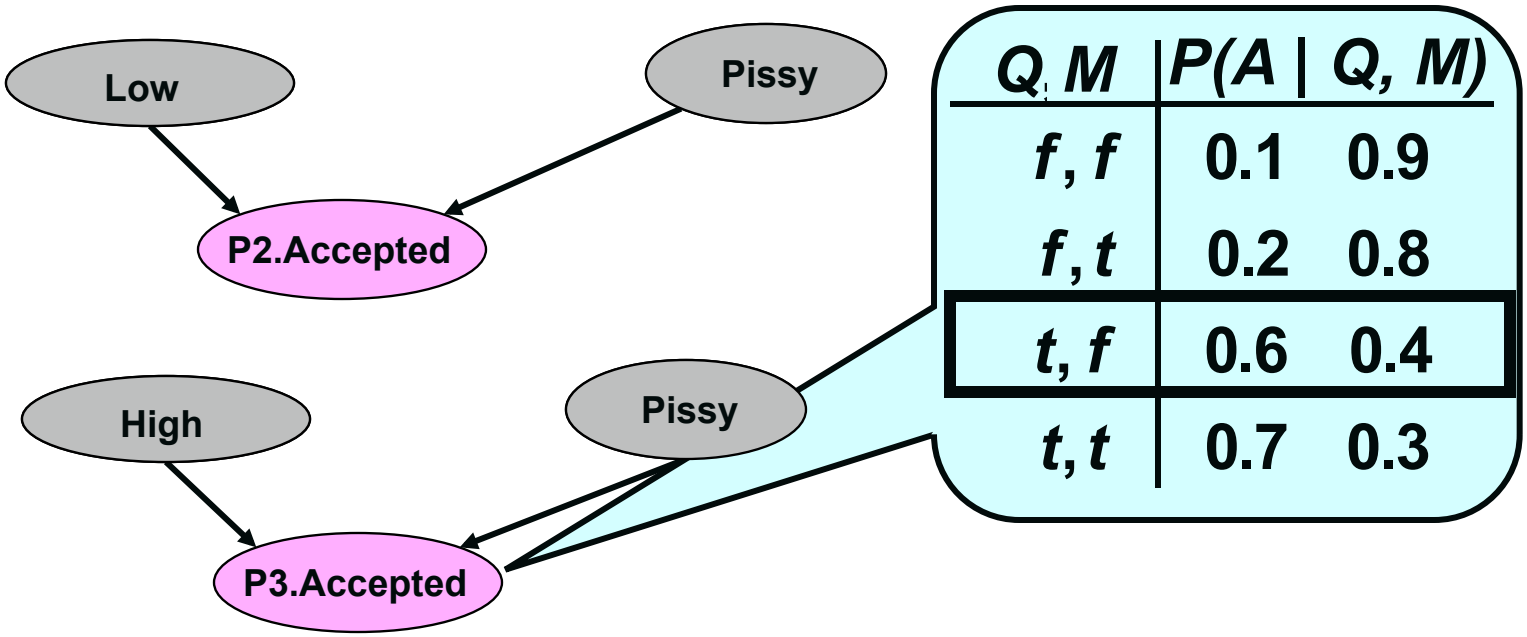


PRM defines distribution over instantiations of attributes

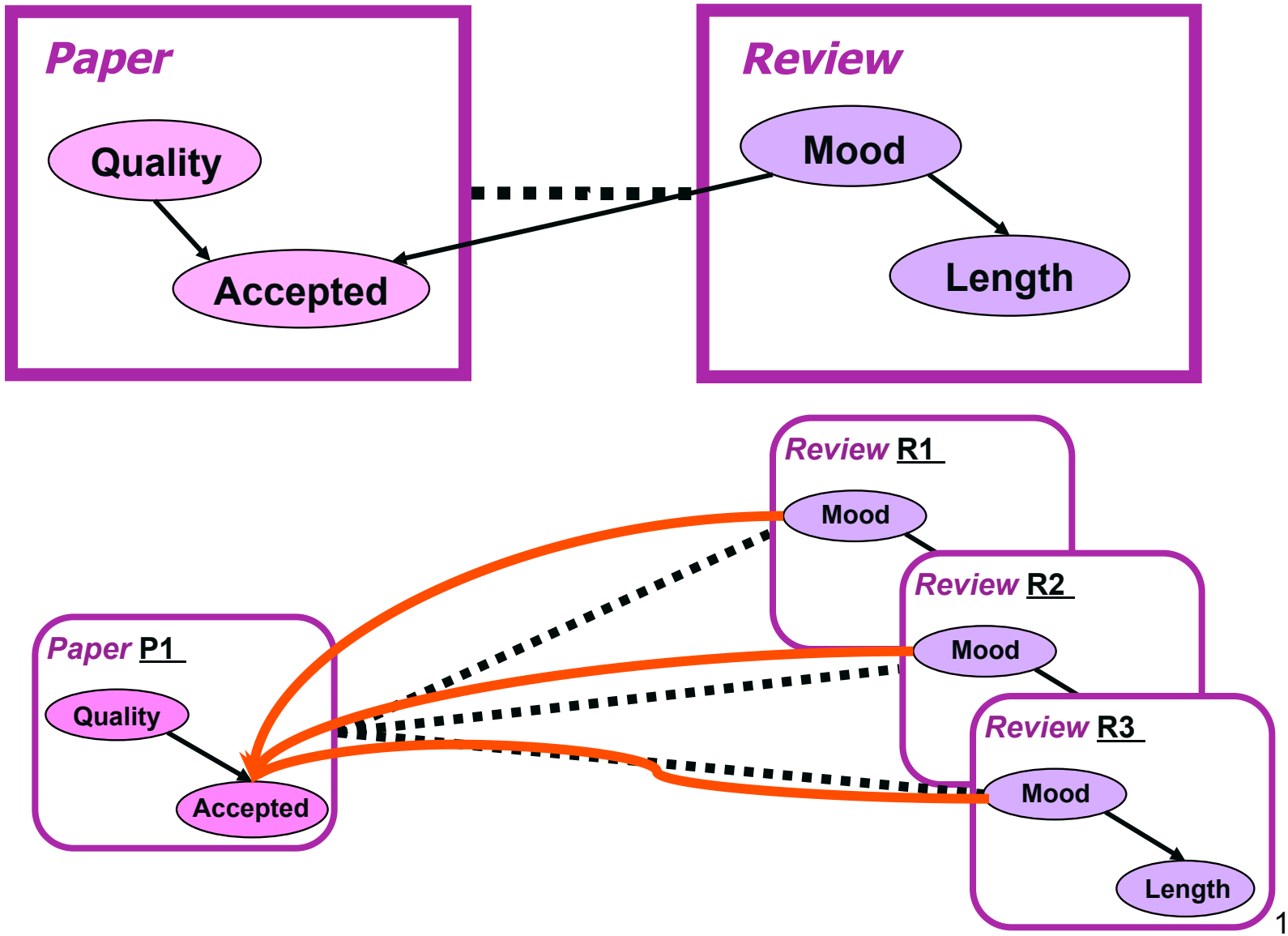
A Portion of the BN



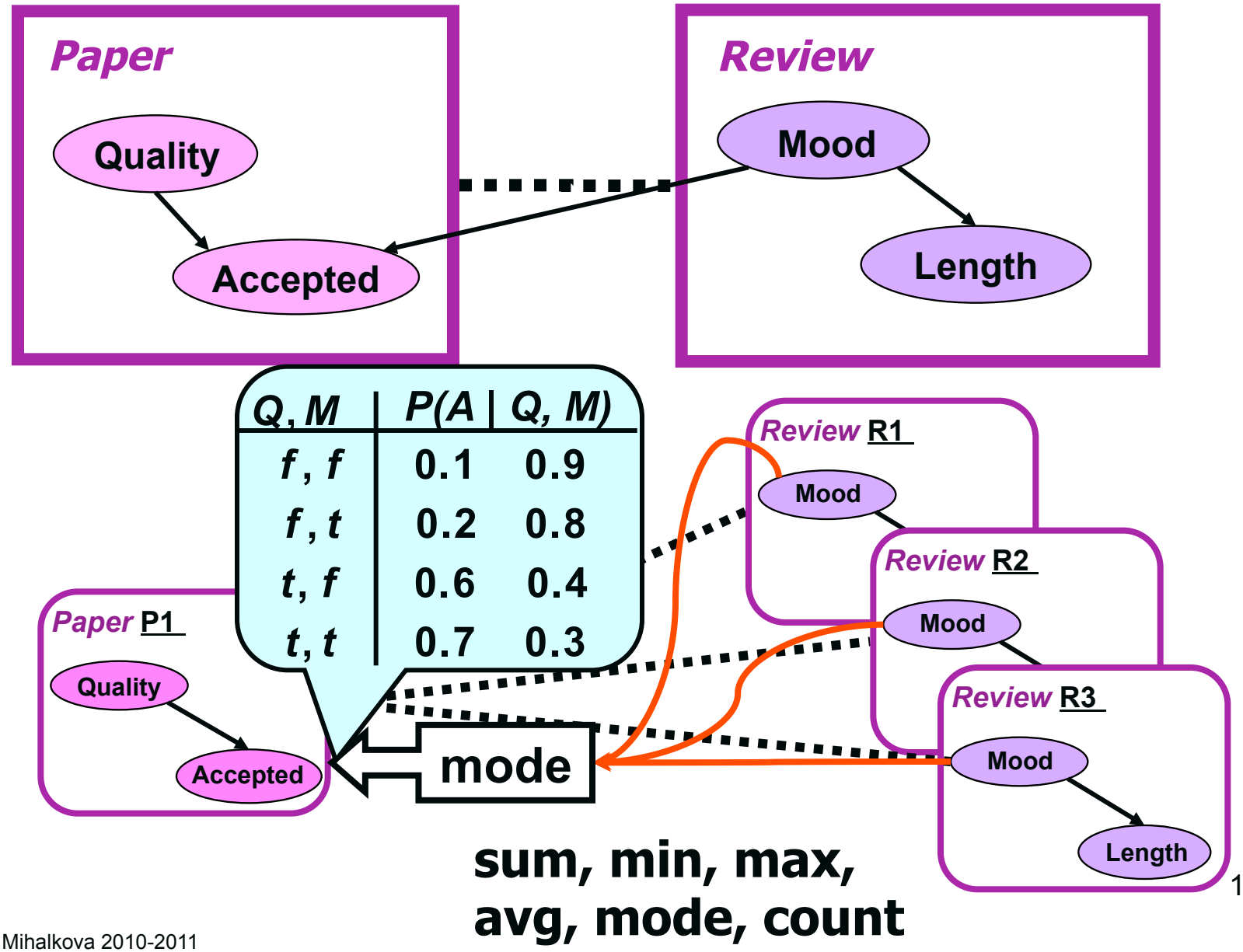
A Portion of the BN



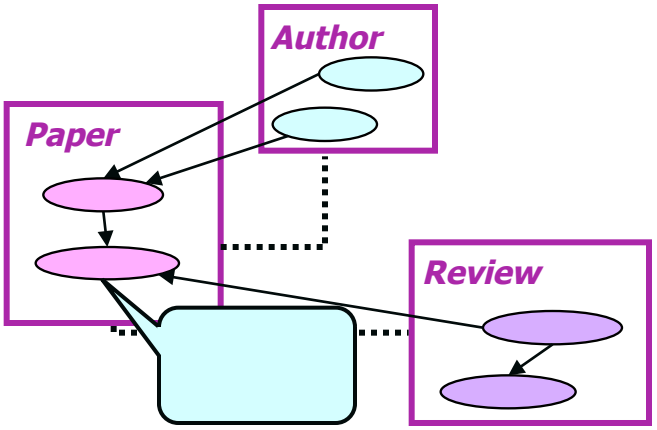
PRM: Aggregate Dependencies

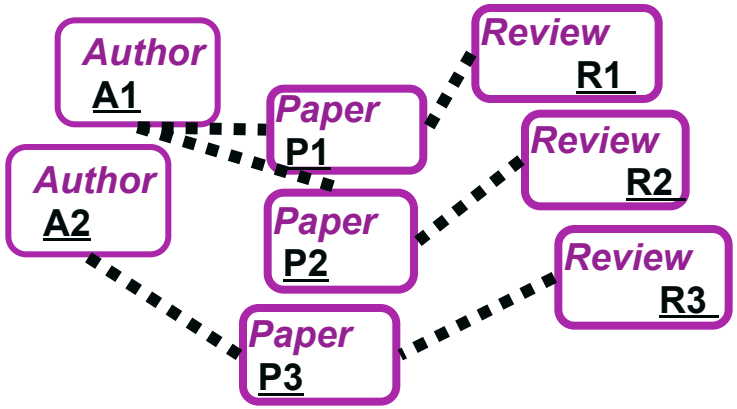


PRM: Aggregate Dependencies



PRM Semantics





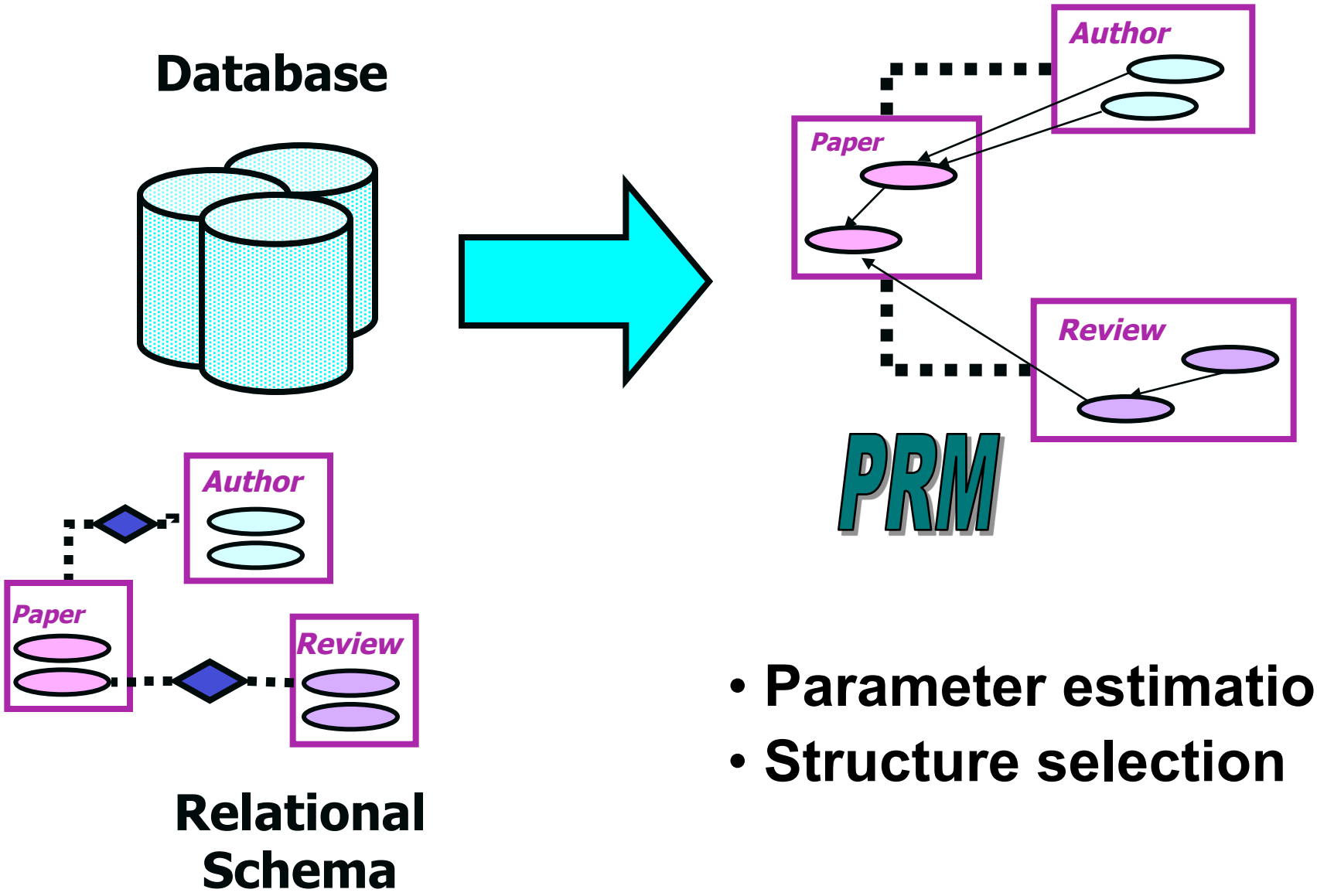
PRM + relational skeleton σ =

probability distribution over completions I:

$$P(\mathbf{I} \mid \sigma, S, \Theta) = \prod_{x \in \sigma} \prod_{x.A} P(x.A \mid parents_{S,\sigma}(x.A))$$

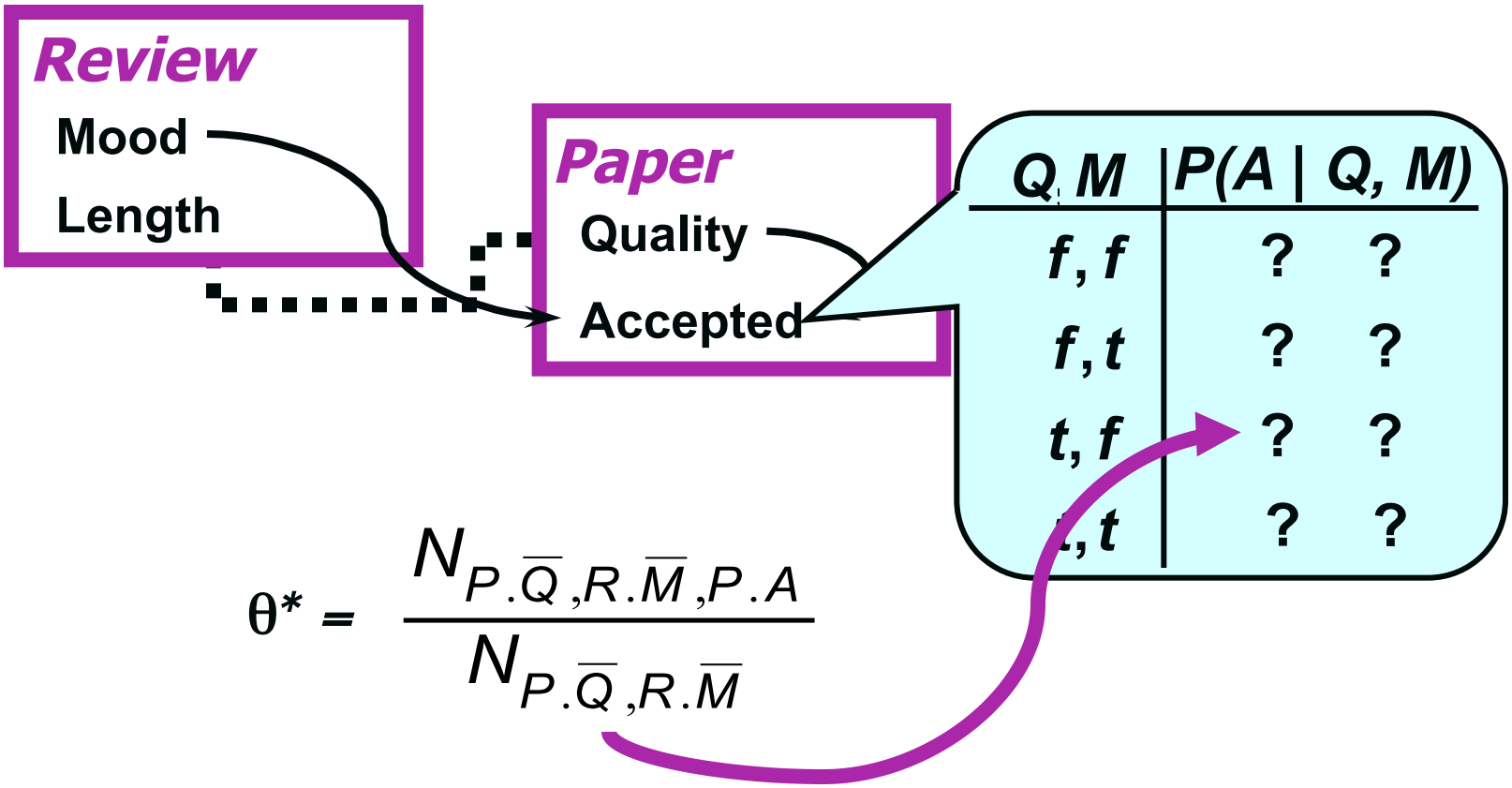
Objects **Attributes**

Learning PRMs



- **Parameter estimation**
- **Structure selection**

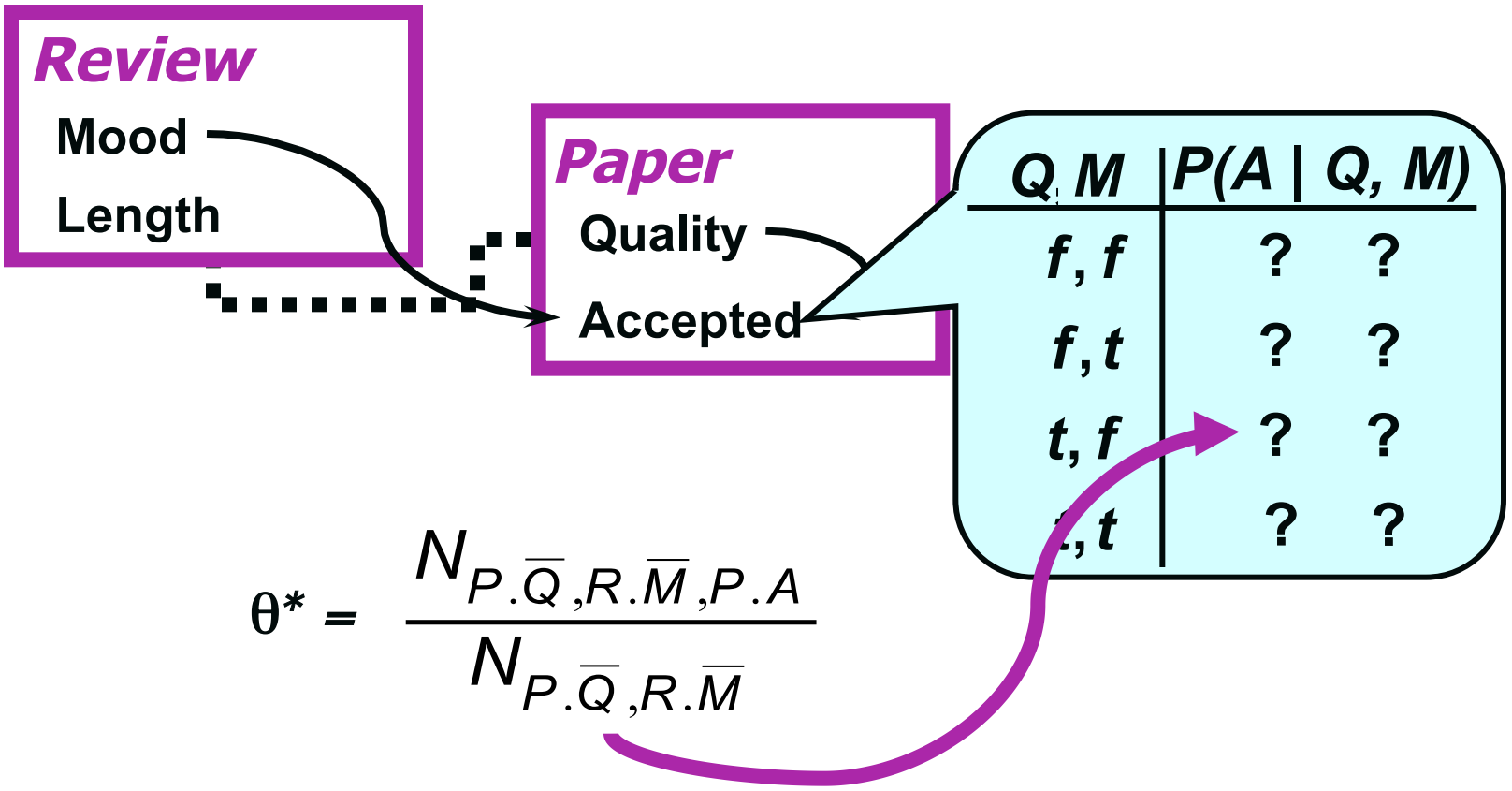
ML Parameter Estimation



$$\theta^* = \frac{N_{P.\overline{Q},R.\overline{M},P.A}}{N_{P.\overline{Q},R.\overline{M}}}$$

where $N_{P.\overline{Q},R.\overline{M},P.A}$ **is the number of accepted, low quality papers whose reviewer was in a poor mood**

ML Parameter Estimation

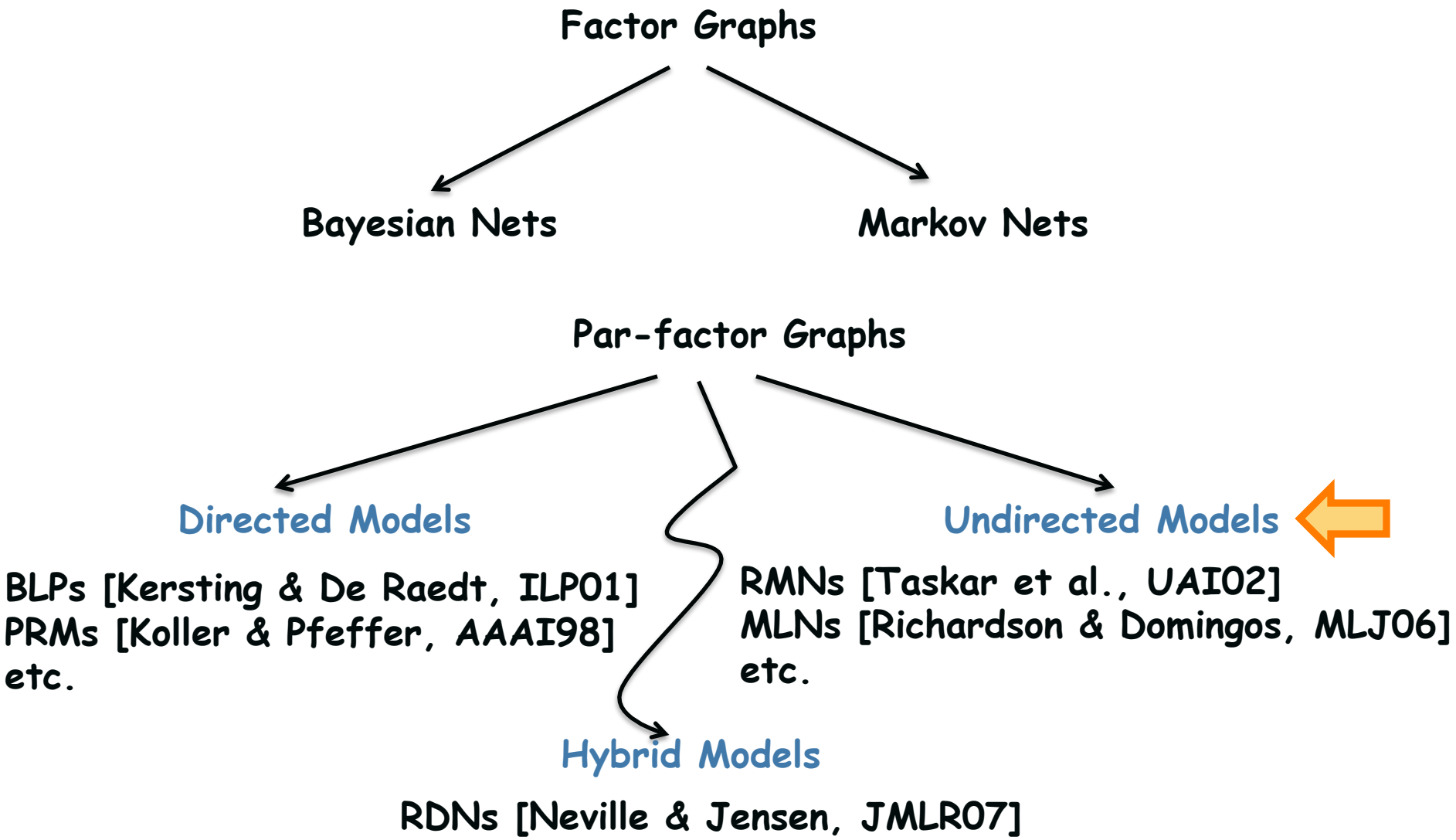


$$\theta^* = \frac{N_{P.\bar{Q},R.\bar{M},P.A}}{N_{P.\bar{Q},R.\bar{M}}}$$

Query for counts:

$$Count\left(\pi_{\substack{P.Quality \\ R.Mood \\ P.Accepted}}\left(\text{Review table} \bowtie \text{Paper table}\right)\right)$$

Road Map



Undirected Models

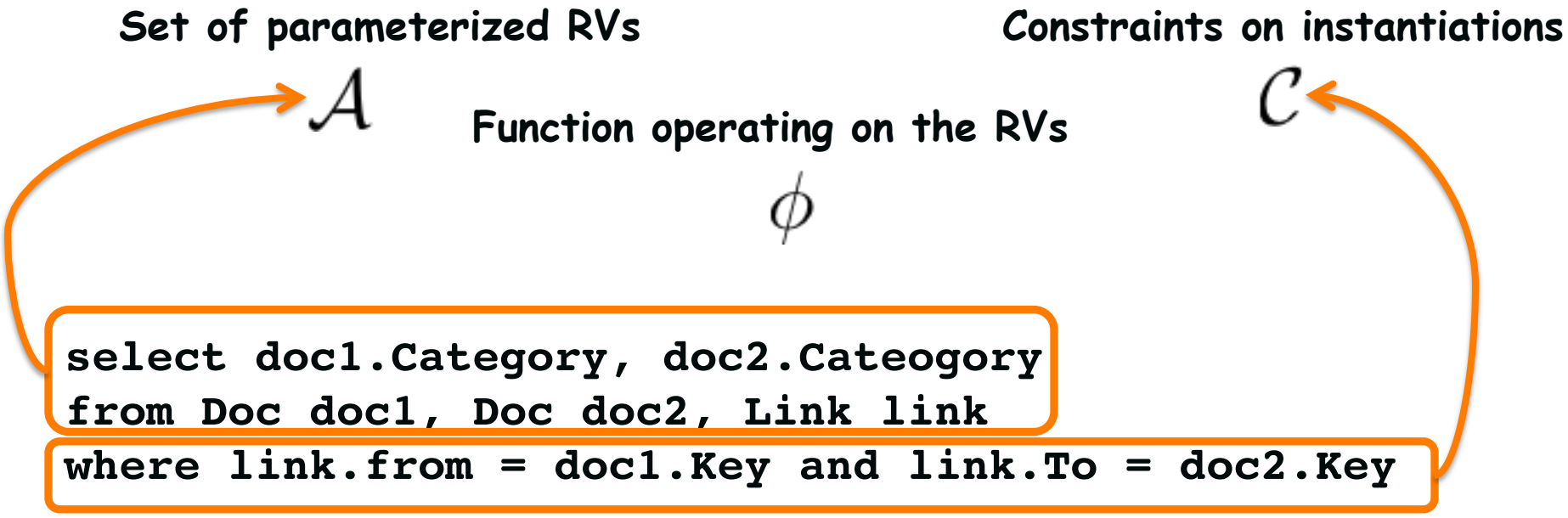
- Relational Markov networks
  - Using database query language (SQL)
  - [Taskar et al., UAI02]
- Markov logic networks
  - Use first-order logic
  - [Richardson & Domingos, MLJ06]
- Both define a Markov network over relational data

Relational Markov Networks

[Taskar et al., UAI02]

- Par-factors are defined using SQL statements
  - Essentially selecting the relational tuples that should be connected in a clique

A par-factor consists of:

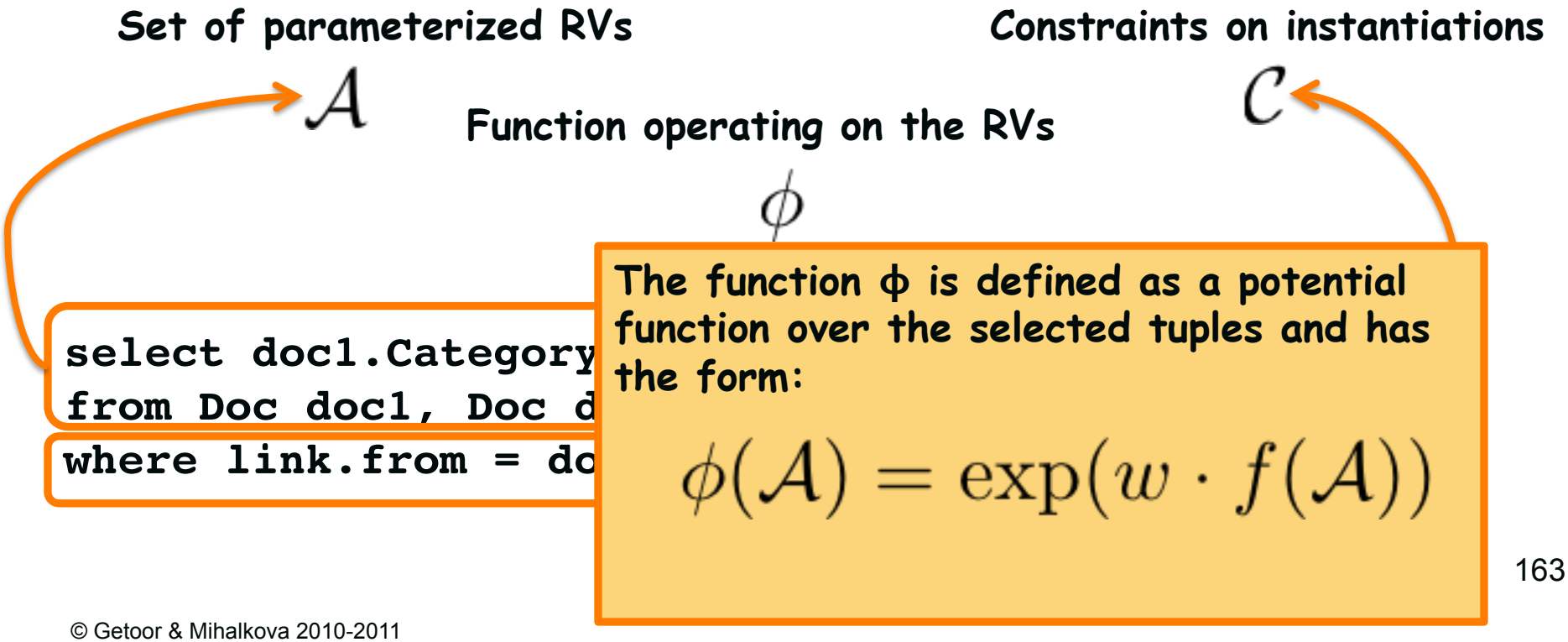


Relational Markov Networks

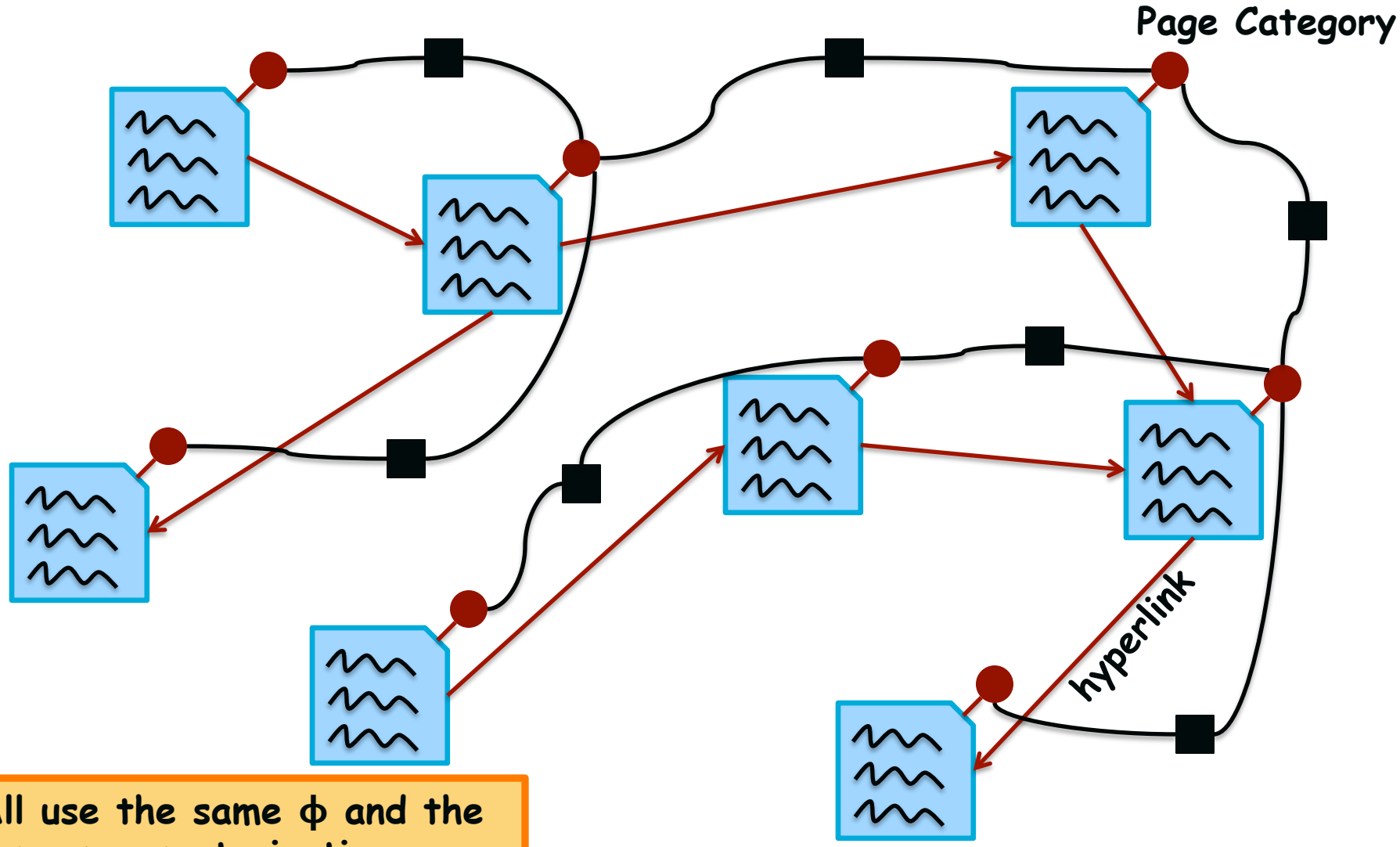
[Taskar et al., UAI02]

- Par-factors are defined using SQL statements
  - Essentially selecting the relational tuples that should be connected in a clique

A par-factor consists of:



Unrolling the RMN

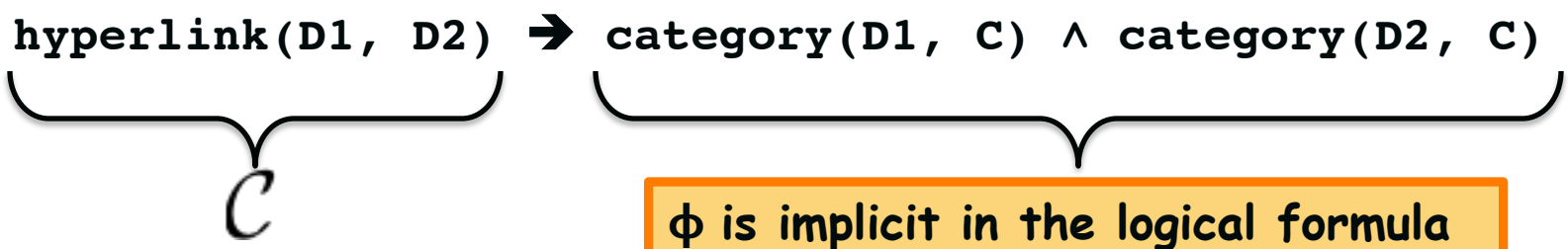


All use the same ϕ and the same parameterization

Markov Logic Networks

[Richardson & Domingos, MLJ06]

- Par-factors are defined using first-order logic statements



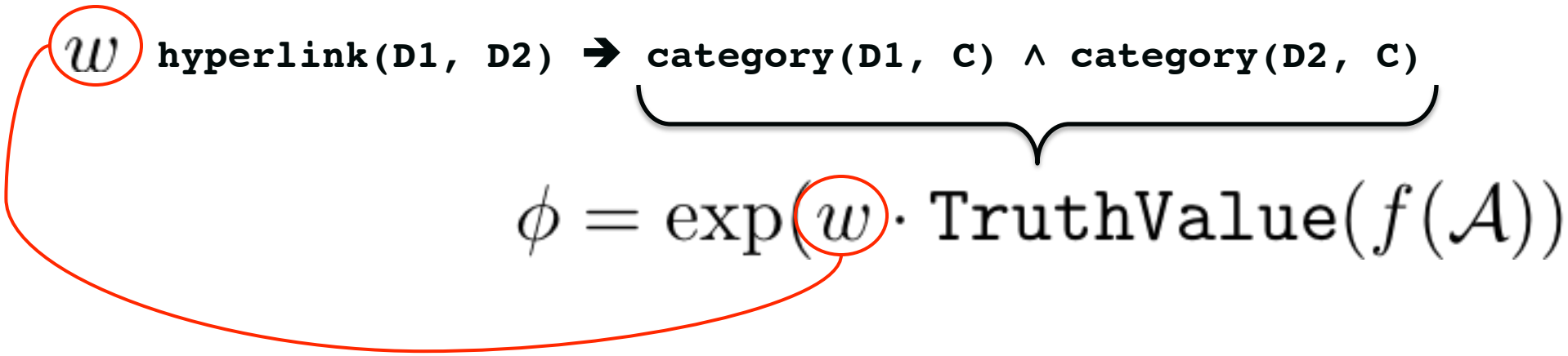
The predicates whose values are known during inference can be seen as constraining the cliques that are constructed over the unknown ones.

ϕ is implicit in the logical formula

$$\mathcal{A} = \{\texttt{category}(\texttt{D1}, \texttt{C}), \texttt{category}(\texttt{D2}, \texttt{C})\}$$

Markov Logic Networks

- Par-factors are defined using first-order logic statements



MLNs Unrolling & Joint Distribution

$$\mathbf{X} = \begin{pmatrix} \text{Category(D1, C1)} \\ \text{Category(D1, C2)} \\ \dots \\ \text{Category(D2, C1)} \\ \text{Category(D2, C2)} \\ \dots \\ \text{Category(D100, C1)} \\ \dots \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ \dots \\ 1 \\ 0 \\ \dots \\ 1 \\ \dots \end{pmatrix}$$

Possible world

$$P(\mathbf{X} = \mathbf{x}) = \frac{\exp\left(\sum_{f_i \in \mathcal{F}} w_i n_i(\mathbf{x})\right)}{\mathbf{Z}}$$

Number of satisfied instantiations

For each formula in the MLN

Case Study

- Next we consider an application of Markov logic to web query disambiguation

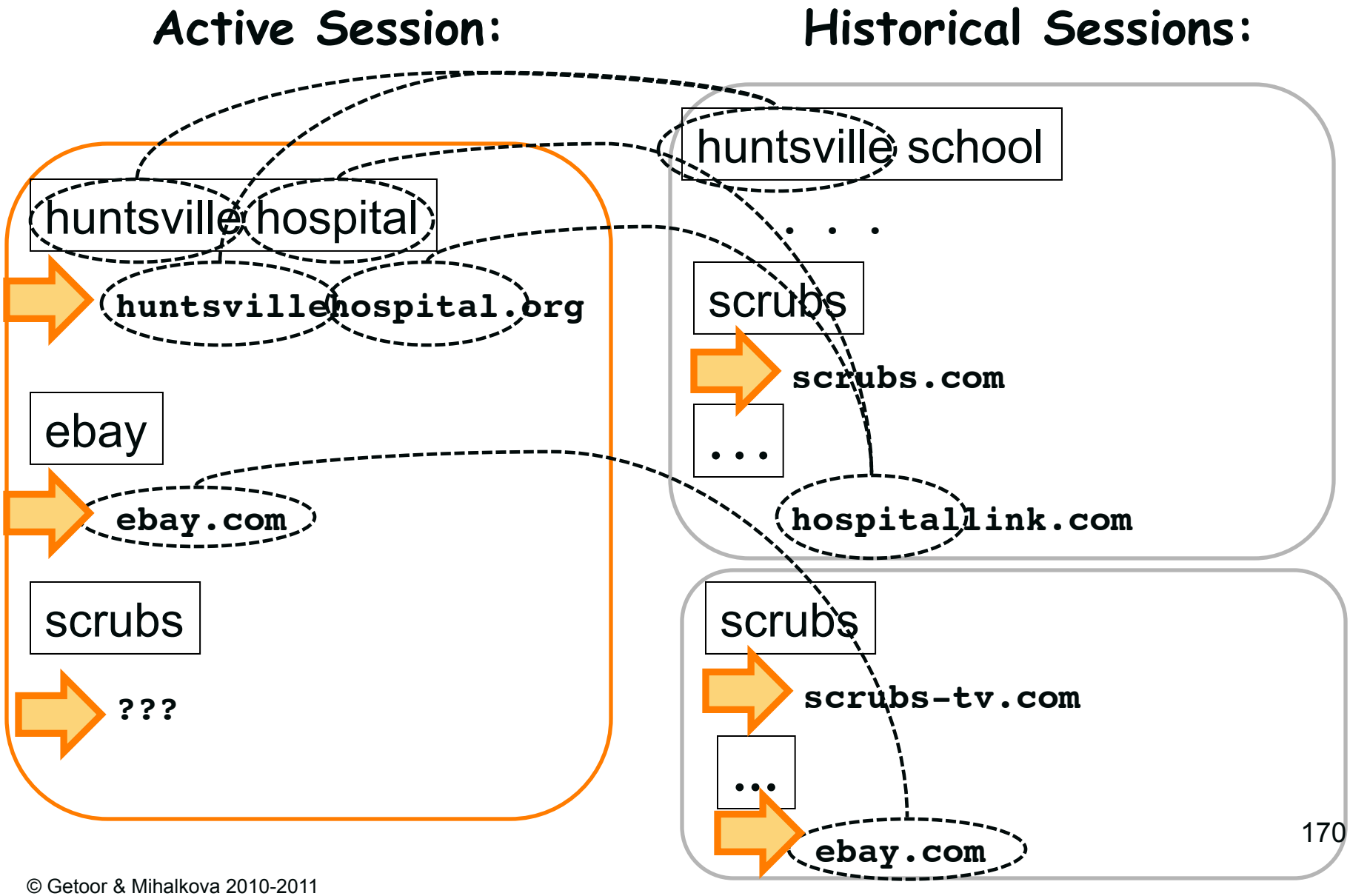
- Based on [Mihalkova & Mooney, ECML09]

Web Query Disambiguation

- Problem: Given an ambiguous query, determine which URLs more likely reflect user interest

- [Mihalkova & Mooney, ECML09] considered a constrained setting in which very little was known about previous user browsing history
  - About 3 previous searches on average

Relationships



Clauses

- Collaborative: User will click on result chosen in sessions related by:
  - Shared click
  - Shared keyword click-to-click, click-to-search, search-to-click, or search-to-search
  - e.g.,

$$\texttt{SharesClick}(\texttt{S},\texttt{D}) \wedge \texttt{ChoseResult}(\texttt{S},\texttt{R},\texttt{Q}) \Rightarrow \texttt{ClickOn}(\texttt{R},\texttt{Q})$$

- Popularity: User will choose result chosen by any previous session, regardless of whether it is related

Clauses Continued

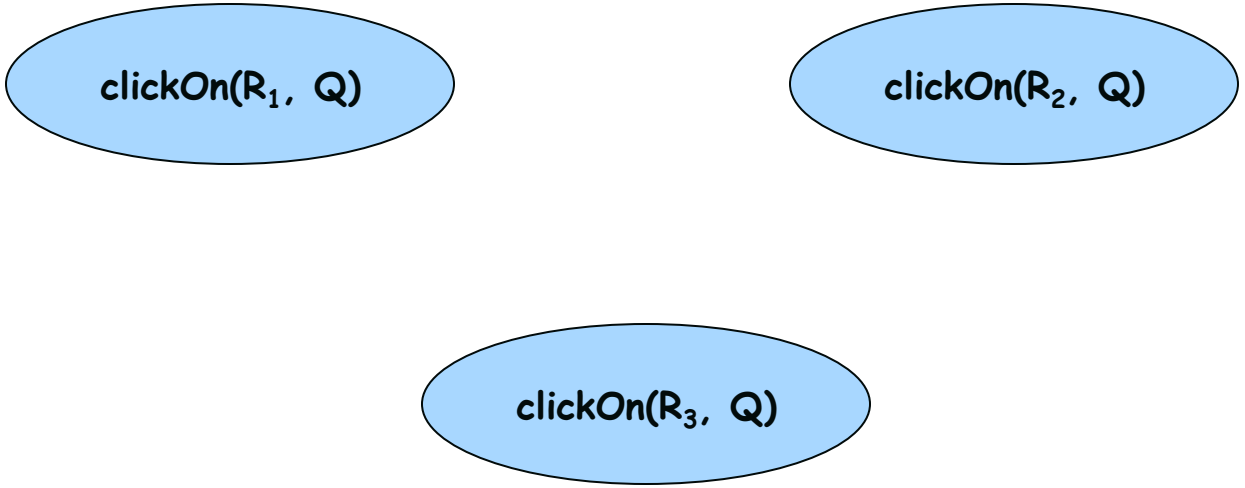
- Local: User will choose result that shares keyword with previous search or click in current session
  - Didn't find it to be effective because of brevity of sessions
- If the user chooses one of the results, she will not choose another
  - Sets up a competition among possible results
  - Allows the same set of weights to work well for different-size problems

$$\texttt{ClickOn}(\texttt{R1}, \texttt{Q}) \wedge \texttt{R1} \neq \texttt{R2} \Rightarrow \neg\texttt{ClickOn}(\texttt{R2}, \texttt{Q})$$

Instantiated Factor Graph

- Let's see how these rules define a factor graph
- Will do this for
  - a single query Q and
  - a set of possible results R_1, R_2, and R_3 for it

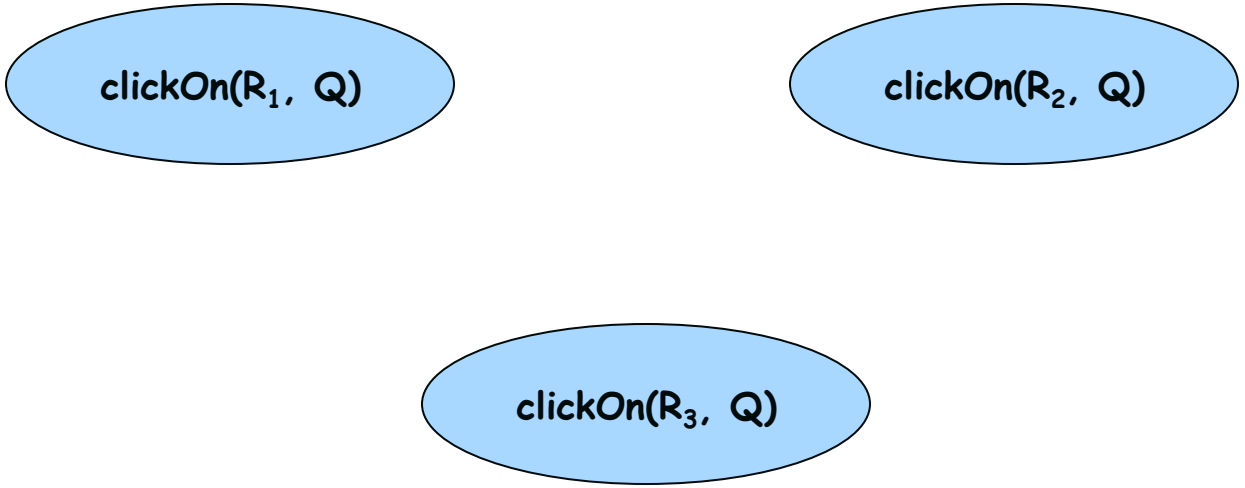
1. Set up decision nodes.
We have one for each grounding of the unknown predicate ClickOn



Instantiated Factor Graph

- Let's see how these rules define a factor graph
- Will do this for
  - a single query Q and
  - a set of possible results R_1, R_2, and R_3 for it

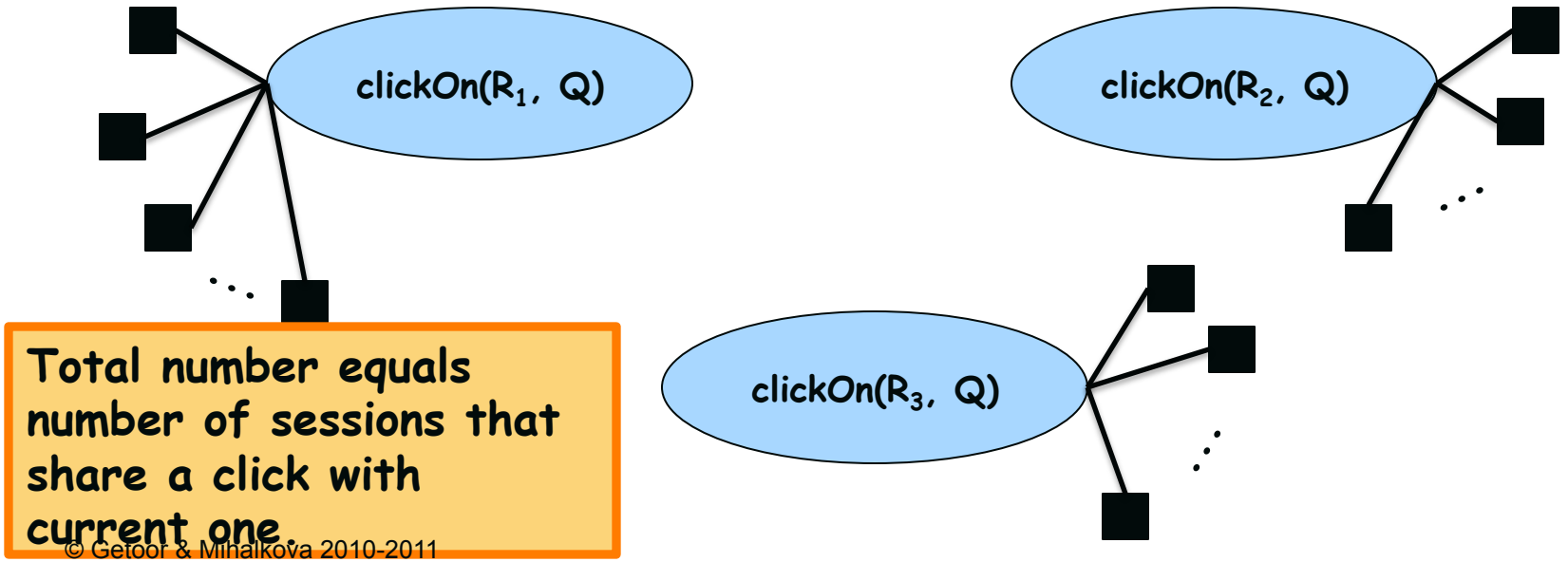
2. Ground out each clause and construct factors corresponding to groundings



Instantiated Factor Graph

- Let's see how these rules define a factor graph
- Will do this for
  - a single query Q and
  - a set of possible results R_1, R_2, and R_3 for it

$$\texttt{SharesClick}(S, D) \wedge \texttt{ChoseResult}(S, R, Q) \Rightarrow \texttt{ClickOn}(R, Q)$$

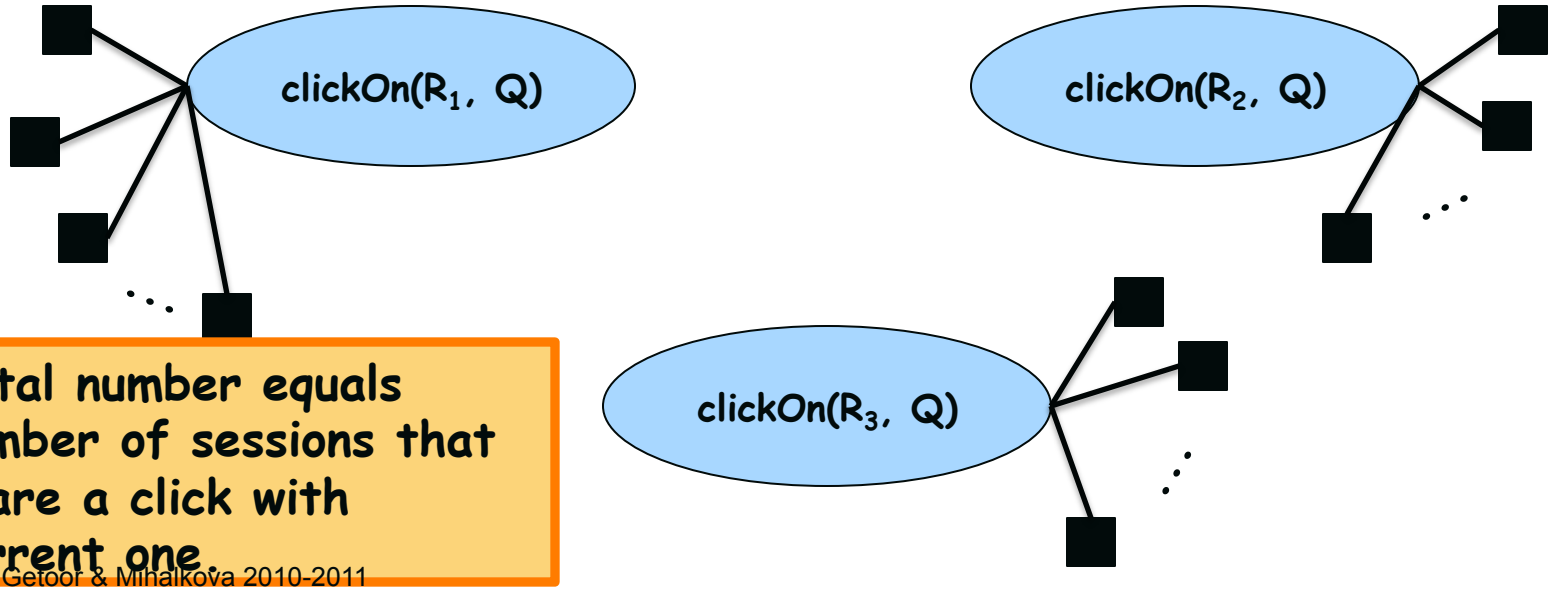


Instantiated Factor Graph

- Let
- Wil
  - a
  - a

Note: So far, what we have is a flat relational classifier: no connection between the decision nodes!

$$\texttt{SharesClick}(\texttt{S},\texttt{D}) \wedge \texttt{ChoseResult}(\texttt{S},\texttt{R},\texttt{Q}) \Rightarrow \texttt{ClickOn}(\texttt{R},\texttt{Q})$$

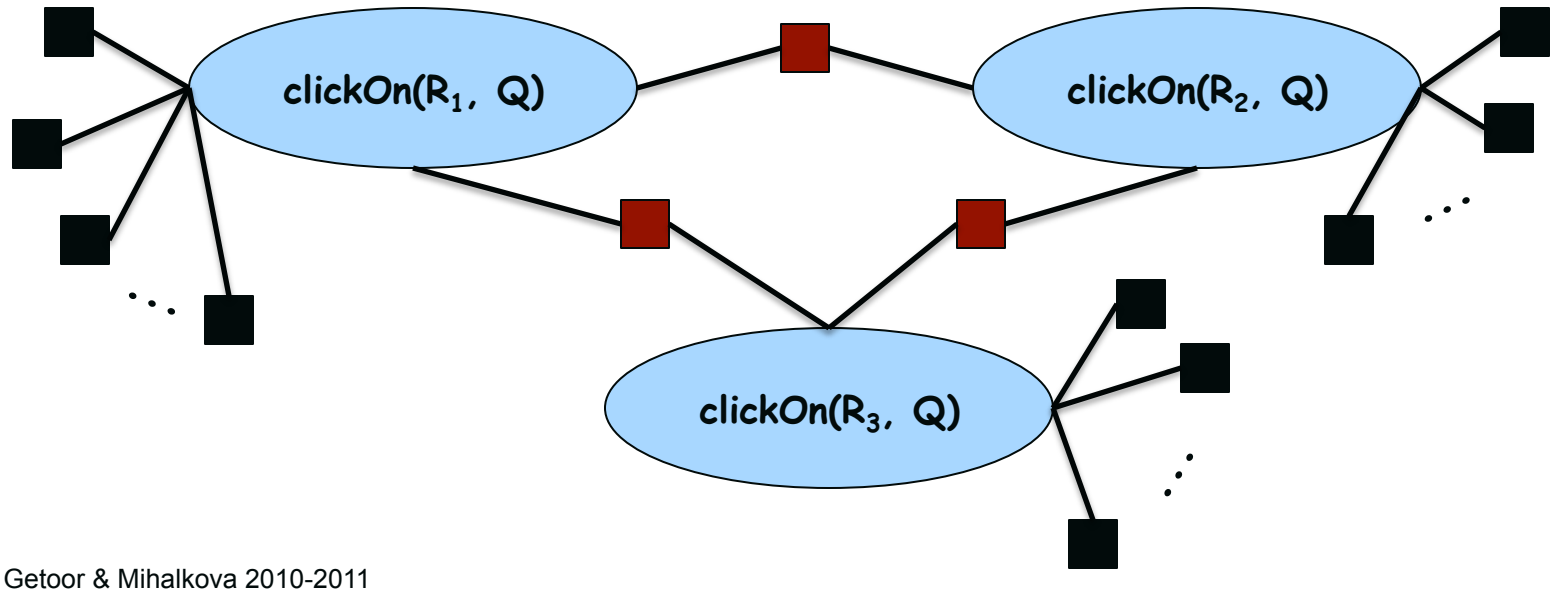


Total number equals number of sessions that share a click with current one.

Instantiated Factor Graph

- Let's see how these rules define a factor graph
- Will do this for:
  - a single query Q and
  - a set of possible results R_1, R_2, and R_3 for it

$$\texttt{ClickOn}(\texttt{R1}, \texttt{Q}) \wedge \texttt{R1} \neq \texttt{R2} \Rightarrow \neg\texttt{ClickOn}(\texttt{R2}, \texttt{Q})$$



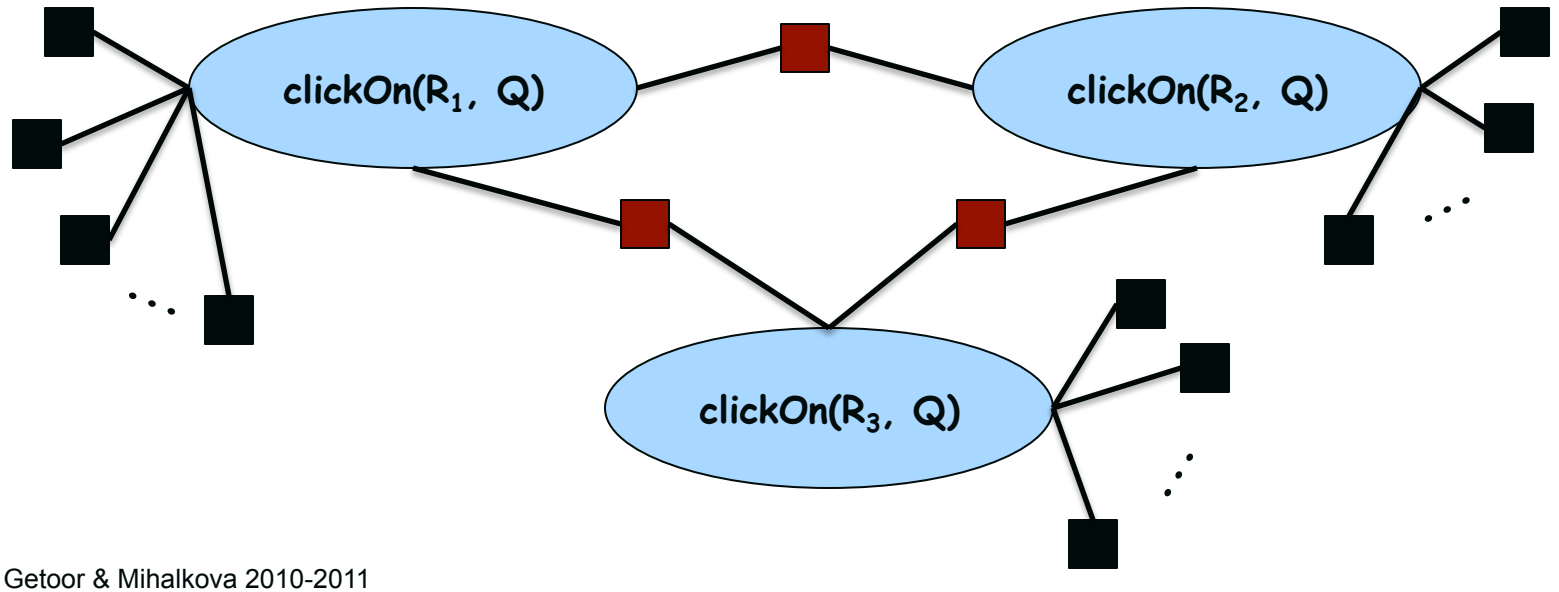
Instantiated Factor Graph

- Let's … graph
- Will …
  - a …
  - a … t

This demonstrates an advantage of using a richer statistical relational representation:

Making our model collective was as easy as adding a rule!

$$\texttt{ClickOn}(\texttt{R1}, \texttt{Q}) \wedge \texttt{R1} \neq \texttt{R2} \Rightarrow \neg\texttt{ClickOn}(\texttt{R2}, \texttt{Q})$$



Directed vs Undirected Models

	Directed	Undirected
Representation	• Capture causal relationships	• Capture symmetric relationships
Parameter Learning	• Amounts to counting	• Cannot compute in closed form • Requires running inference
Structure Learning	• Parameters updated only where structure changed • Need to maintain acyclicity	• Parameters updated globally
Inference		• Need to compute normalizing function

Hybrid Models

- Hybrid models aim at combining the advantages of directed and undirected ones, while avoiding the disadvantages

- Next we briefly introduce relational dependency networks (RDNs) [Neville & Jensen, JMLR07]

Relational Dependency Networks

[Neville & Jensen, JMLR07]

- An extension of dependency networks [Heckerman et al., JMLR00] to relational domains
- In a dependency network:
  - As in Markov nets, one's neighbors render it independent of all other variables
    - No need to worry about maintaining acyclicity
  - As in Bayesian nets, potential functions are represented as conditional probability tables (CPTs)
    - No normalization necessary
- RDN "lift" DNs to relational domains:
  - Dependencies are described for parameterized RVs
  - Upon instantiation, RDNs define a DN in which CPTs are shared

Summary So Far

- Started with relational classifiers
  - Focus: relational feature construction
- Moved to collective classification models
  - Focus: propagating label assignments
- Considered advanced SRL languages
  - Focus: representing shared structure while allowing for principled learning and inference

LOOKING AHEAD

Statistical Relational Learning and the Web

What are some challenges that we swept under the rug?

Some

Challenges Addressed by SR Learning and Inference

- Multi-relational data
  - Entities can be of different types
  - Entities can participate in a variety of relationships
- Probabilistic reasoning under noise and/or uncertainty

Challenges Arising in Web Applications

- Entities of different types
  - E.g., users, URLs, queries
- Entities participate in variety of relations
  - E.g., click-on, search-for, link-to, is-refinement-of
- Noisy, sparse observations

Improving Scalability

- Improving efficiency of inference through continuous random variables and sets
- Lifted Inference
- Learning from data streams

Improving Scalability

- Improving efficiency of inference through continuous random variables and sets
  - Probabilistic Soft Logic (PSL) 
    - [Bröcheler et al., UAI10]
- Lifted Inference
- Learning from data streams

Probabilistic Soft Logic

- First-order-logic-like language for expressing relational dependencies

$$\texttt{Category}(\texttt{A},\texttt{C}) \Leftarrow \texttt{Category}(\texttt{B},\texttt{C}) \wedge \texttt{Unknown}(\texttt{A}) \wedge \texttt{link}(\texttt{A},\texttt{B}) \wedge \texttt{A} \neq \texttt{B}$$

- Arbitrary similarity functions on entity attributes:

$$A \approx B \Leftarrow A.name \approx B.name$$

- Relation-defined sets:

$$A \approx B \Leftarrow \{A.friends\} \approx \{B.friends\}$$

Combining Soft Values in PSL

$$H_1 \oplus H_2 \oplus \cdots \oplus H_m \Leftarrow B_1 \otimes B_2 \otimes \cdots \otimes B_n$$

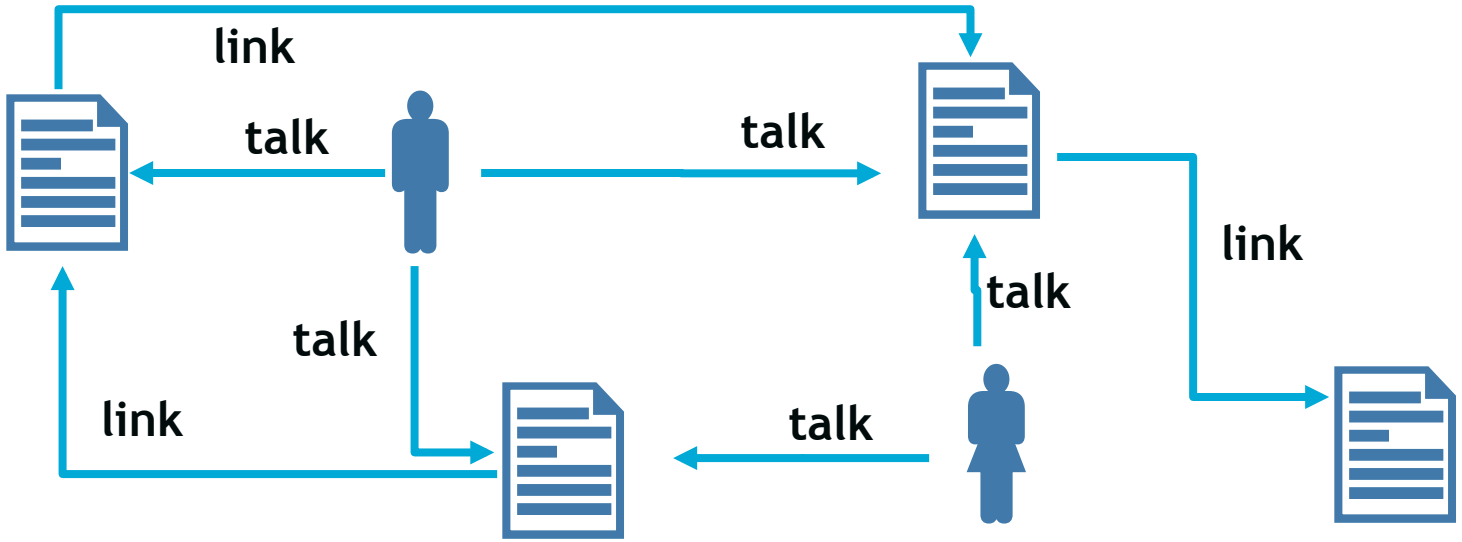
- Soft values in a rule are combined using T-norms:
  - Lukasiewicz T-norm (can be customized)
    - $\oplus(h_1, h_2) = \min(1, h_1+h_2)$
    - $\otimes(h_1, h_2) = \max(0, 1- h_1+h_2)$

Slide credit: Adapted from slides by Matthias Bröcheler

Efficient Inference in PSL

- Attribute and set similarity functions computed externally as “black boxes”
- PSL rules are instantiated “lazily,” on an as-needed basis
- Inference is cast as a constrained continuous numerical optimization problem, solved in polynomial time

PSL in Wikipedia



Graphic credit: Matthias Bröcheler

Wikipedia Rules

$$\text{hasCat}(A,C) \leftarrow \text{hasCat}(B,C) \wedge A!=B \wedge \text{unknown}(A) \wedge \text{document}(A,T) \wedge \text{document}(B,U) \wedge \text{similarText}(T,U)$$

$$\text{hasCat}(A,C) \leftarrow \text{hasCat}(B,C) \wedge \text{unknown}(A) \wedge \text{link}(A,B) \wedge A!=B$$

$$\text{hasCat}(D,C) \leftarrow \text{talk}(D,A) \wedge \text{talk}(E,A) \wedge \text{hasCat}(E,C) \wedge \text{unknown}(D) \wedge A!=B$$

Slide credit: Matthias Bröcheler

Improving Scalability

- Improving efficiency of inference through continuous random variables and sets
- Lifted inference
  - What is lifted inference?
- Learning from data streams

Lifted Inference Intuitions

- Instantiating an SRL model fully can:
  - Result in intractably large inference problem
  - Be wasteful because computations are repeated due to tying of factors
- Lifted inference approaches recognize redundancies due to symmetries and organize computations to avoid them
  - e.g., summing over entire sets of variables, recognizing identical messages being sent and consolidating them
- Active area of research and a promising direction for successfully scaling to large domains

Improving Scalability

- Improving efficiency of inference through continuous random variables and sets
- Lifted Inference
- Learning from data streams
  - Work on accurate parameter learning from data streams, e.g. [Huynh & Mooney; SDM11]

Some Other Things We Skipped

- Probabilistic databases
  - [Dalvi & Suciu, VLDB04; Das Sarma et al., ICDE06; Antova et al., VLDBJ09; Sen et al. VLDBJ09]
- Other lifted inference techniques, e.g.,
  - Lifted variable elimination: [Poole, IJCAI03; de Salvo Braz et al IJCAI05, AAAI06;Milch et al., AAAI08]
  - Lifted belief propagation[Jaimovich et al., UAI07; Singla & Domingos, AAAI08; Kersting et al., UAI09; de Salvo Braz et al, SRL-09]
- SRL Models based on probabilistic programming languages
  - E.g., IBAL [Pfeffer, IJCAI01], BLOG [Milch et al., IJCAI05], Church [Goodman et al., UAI08], Factorie [McCallum et al., NIPS09]

Conclusion

- Web & Social Media inherently noisy and relational
- Described a set of well-suited tools for dealing with noisy, relational data
- However, as of yet, not many success stories
- Enablers:
  - Scaling
  - Online Feature construction
  - Dealing with dynamic data
- Time is right: technology & data
  - New platforms, parallel processing
  - More data
  - Growing need for both personalization and privacy

Acknowledgements

- Thank you to the Search Group at Microsoft Research, Misha Bilenko, Matthias Bröcheler, Amol Deshpande, Nir Friedman, Ariel Fuxman, Aris Gionis, Anitha Kannan, Alex Ntoulas, Hossam Sharara, Marc Smith, Elena Zheleva and others for their slides, comments, and/or helpful discussions

- Lily is supported by a CI Fellowship under an NSF CRA grant

- The linqs group at UMD http://www.cs.umd.edu/linqs is supported by ARO, KDD, NSF, MIPS, Microsoft Research, Google, Yahoo!