Reasoning about Independence in Probabilistic Models of Relational Data

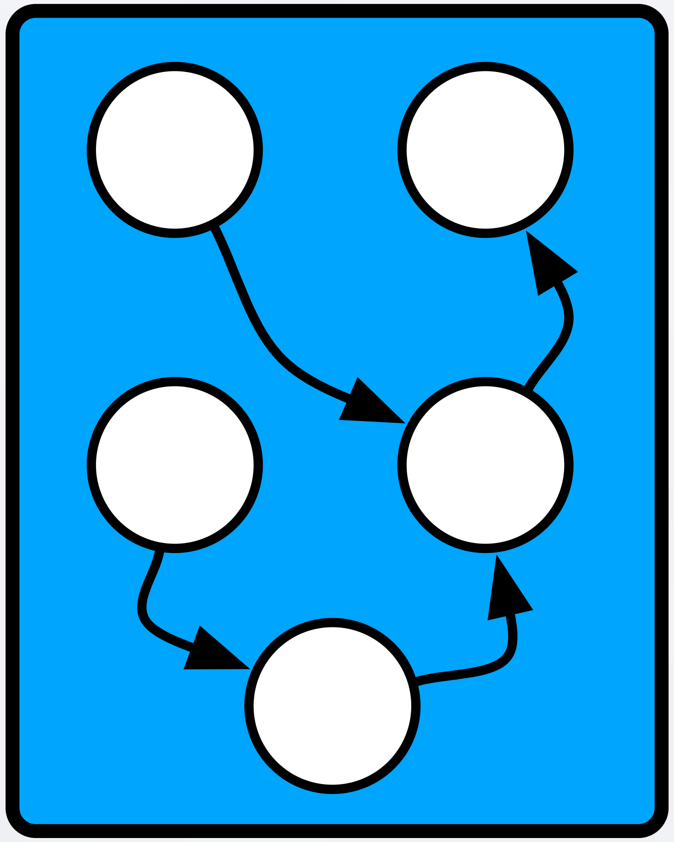
Marc Maier, Katerina Marazopoulou, David Jensen

A Sound and Complete Algorithm for Learning Causal Models from Relational Data

Marc Maier, Katerina Marazopoulou, David Arbour, David Jensen

Knowledge Discovery Laboratory • University of Massachusetts Amherst

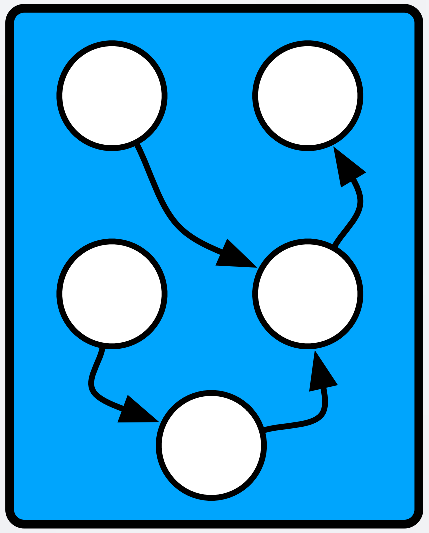
The "world"



Bayesian network

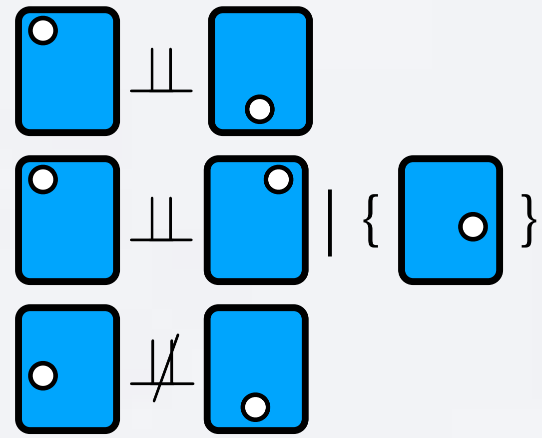
[Pearl 1988]

The “world”



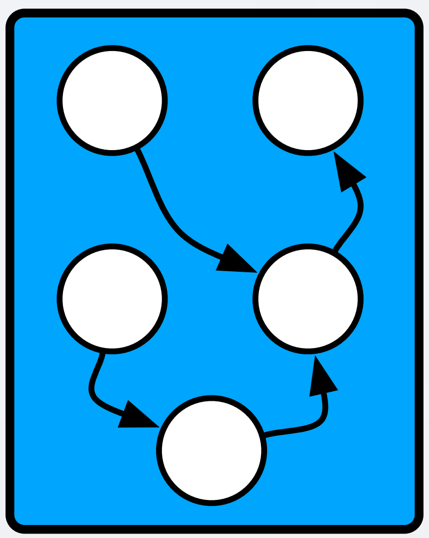
Bayesian network

[Verma & Pearl 1988; Geiger & Pearl 1988] conditional independencies



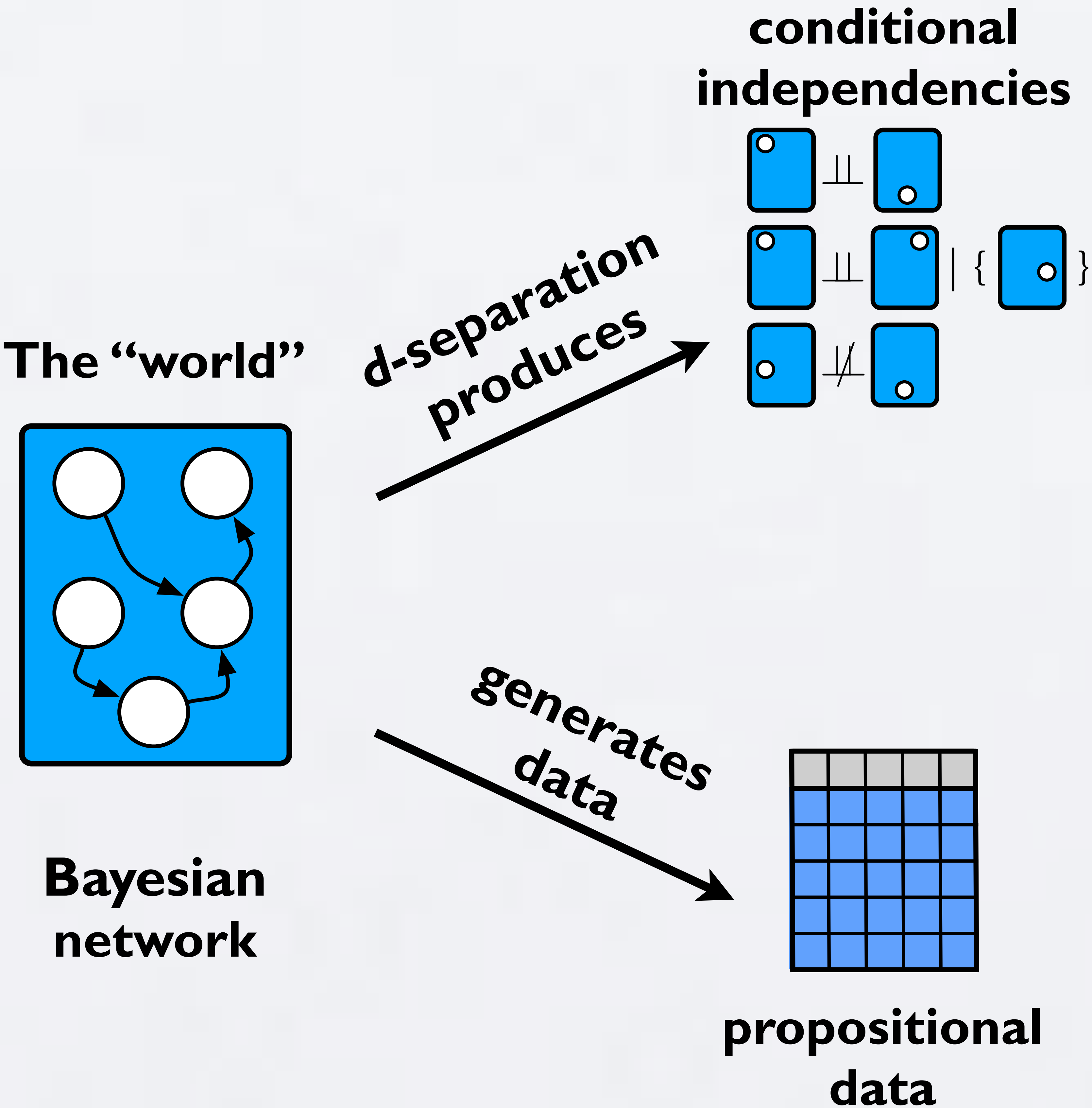
The "world"

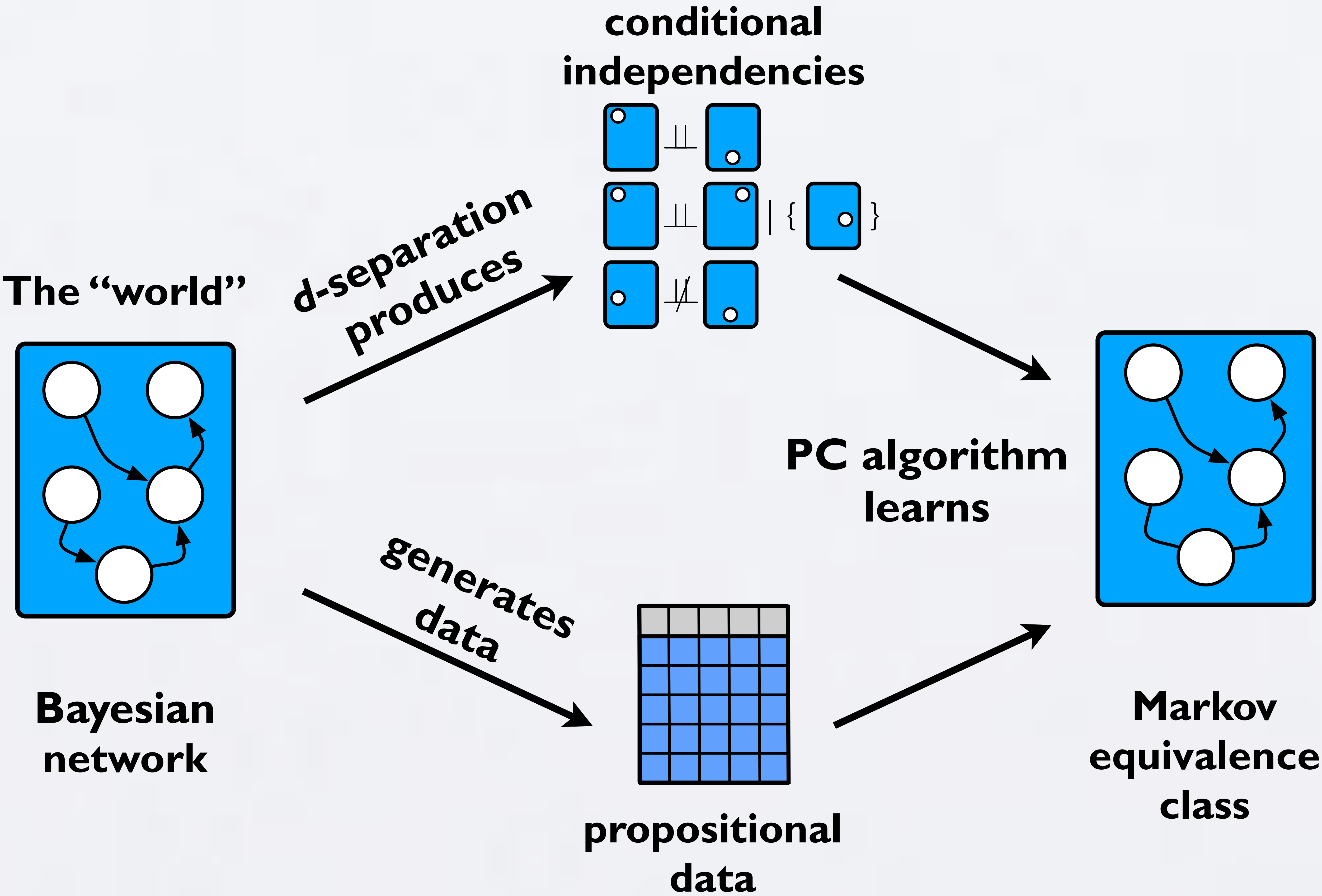
d-separation produces



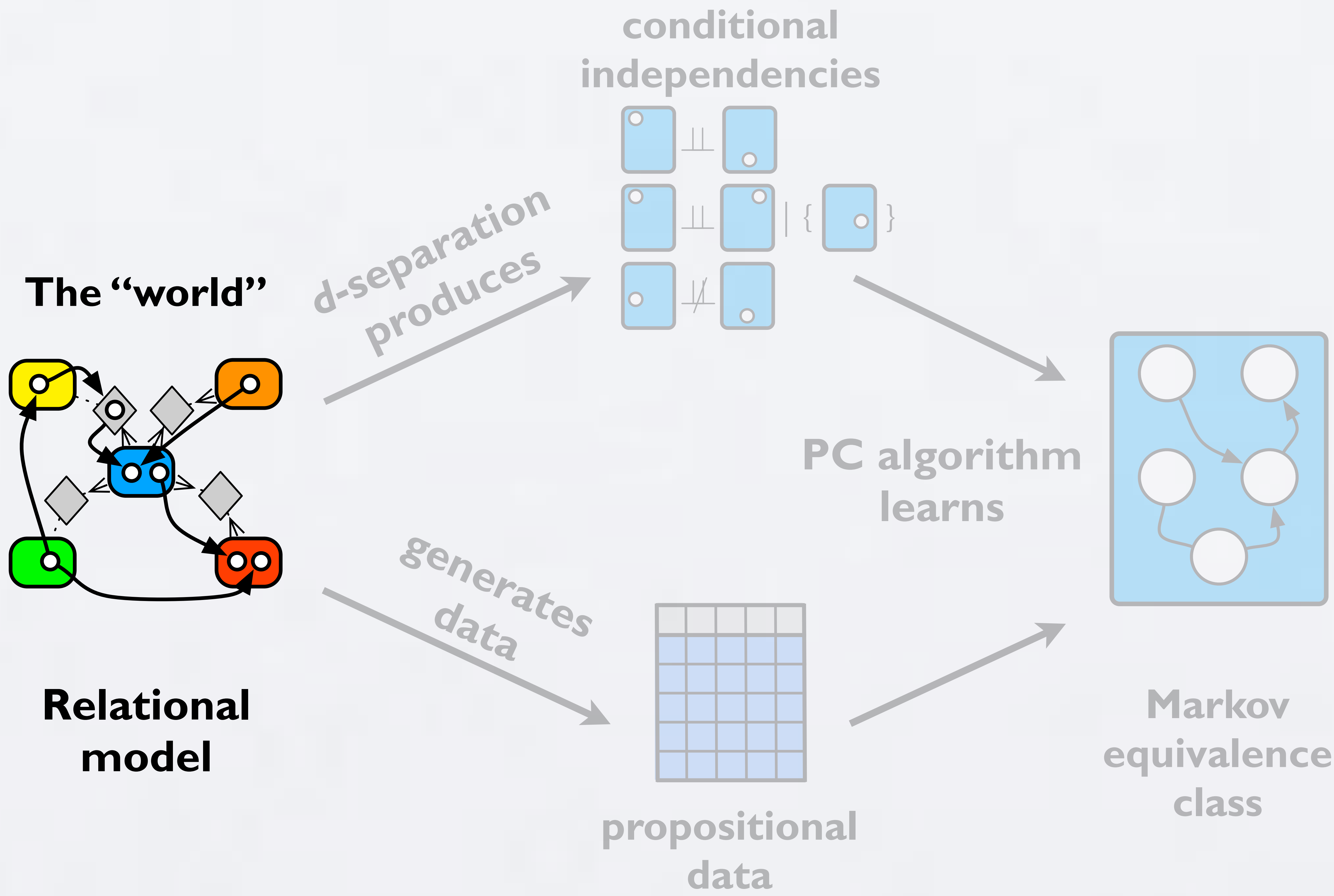
Bayesian network

[Verma & Pearl 1988; Geiger & Pearl 1988]

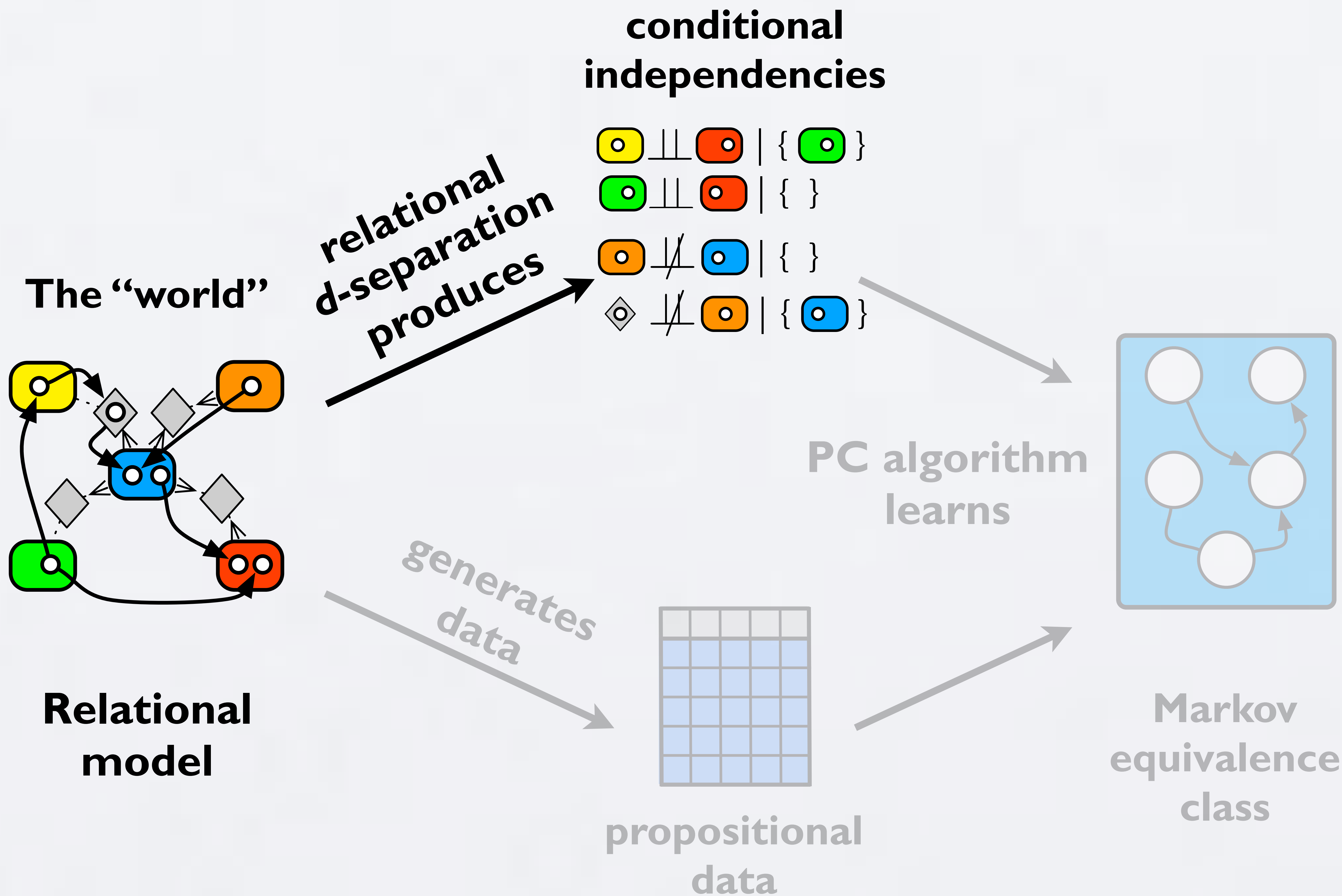

conditional independencies
The "world"
d-separation produces
generates data
Bayesian network
propositional data



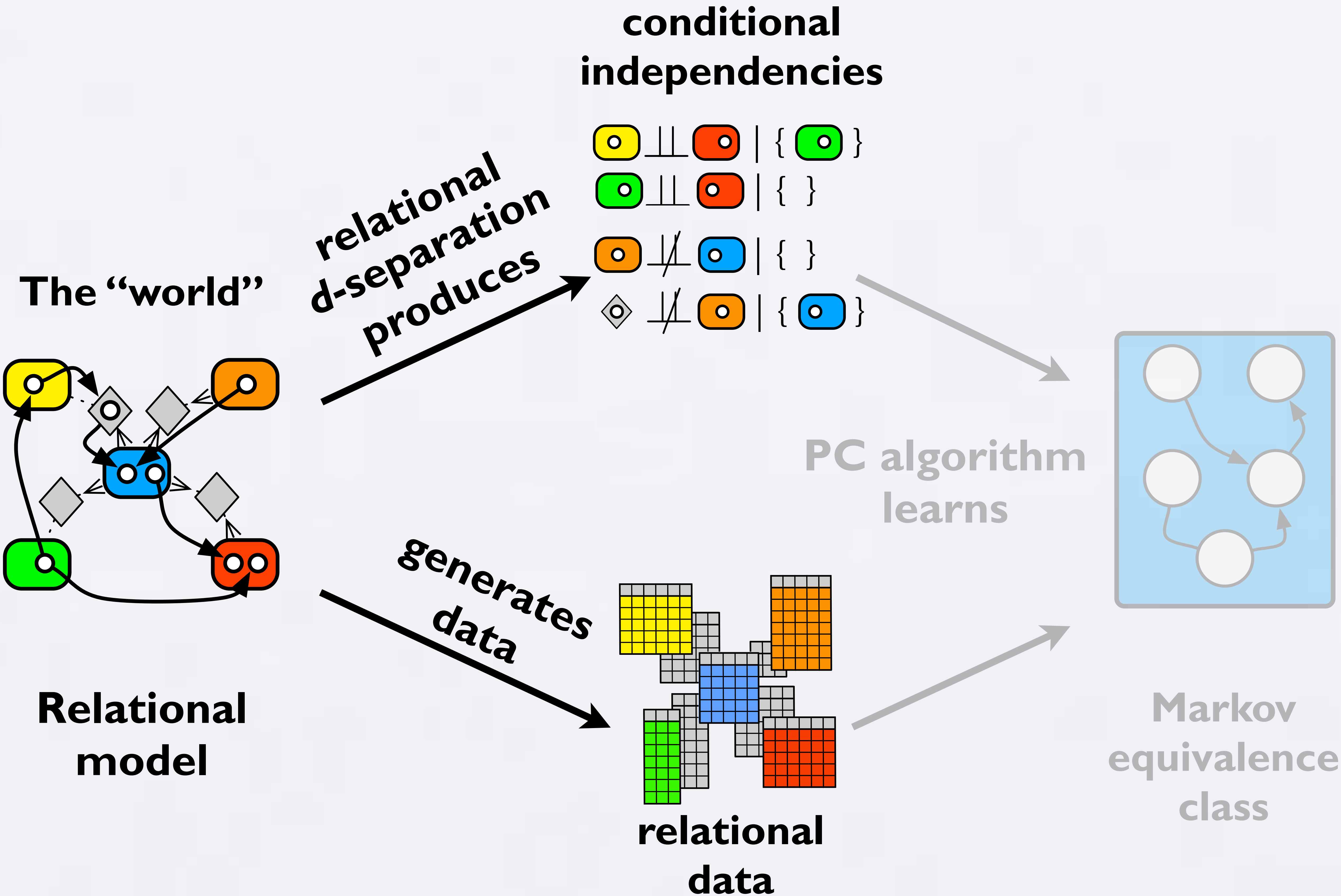
[Pearl 2000; Spirtes et al. 2000]



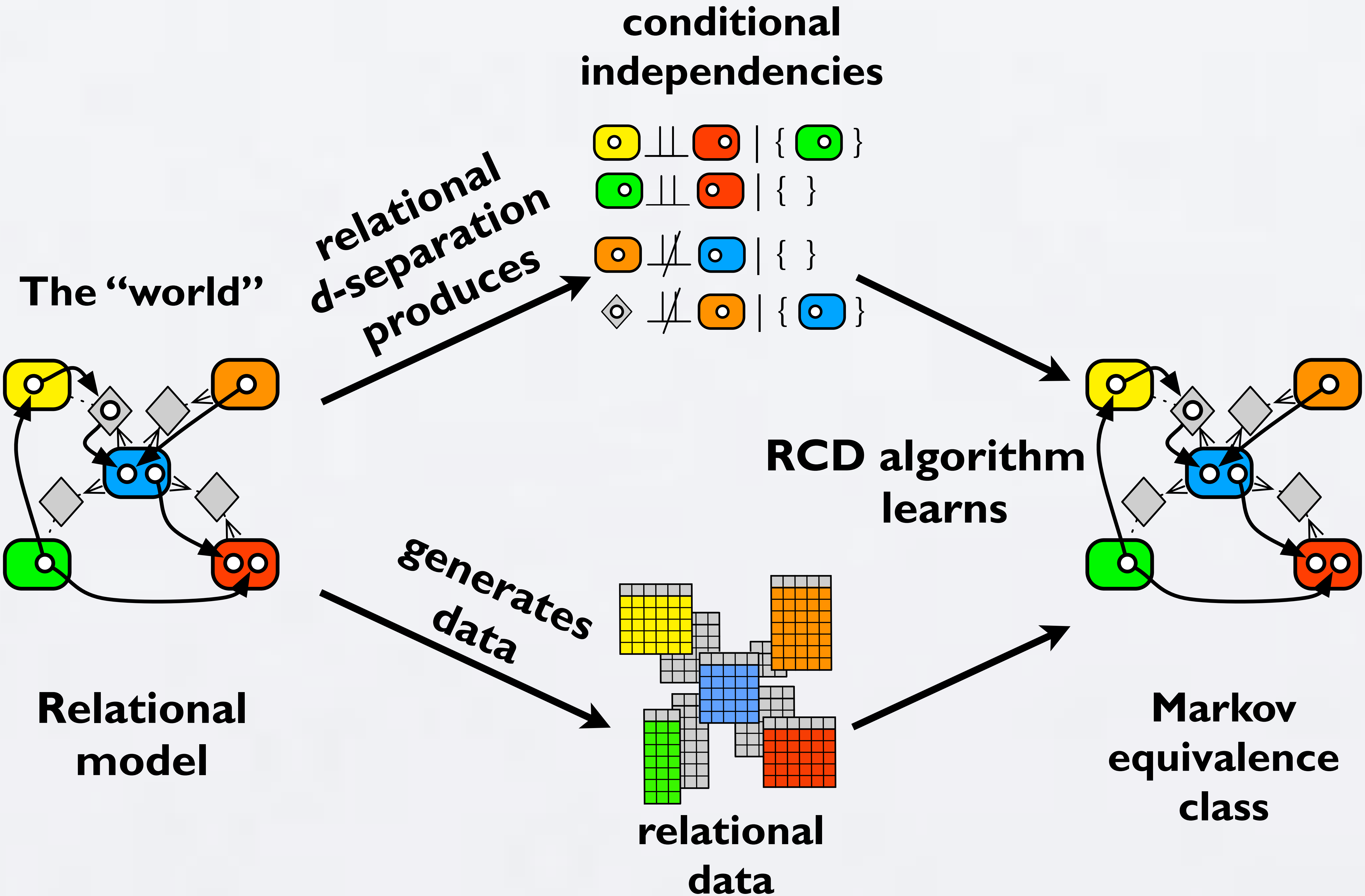
[Getoor et al. 2007; Heckerman et al. 2007]



[Maier et al. 2013]



conditional independencies
The "world"
relational d-separation produces
Relational model
generates data
relational data
PC algorithm learns
Markov equivalence class

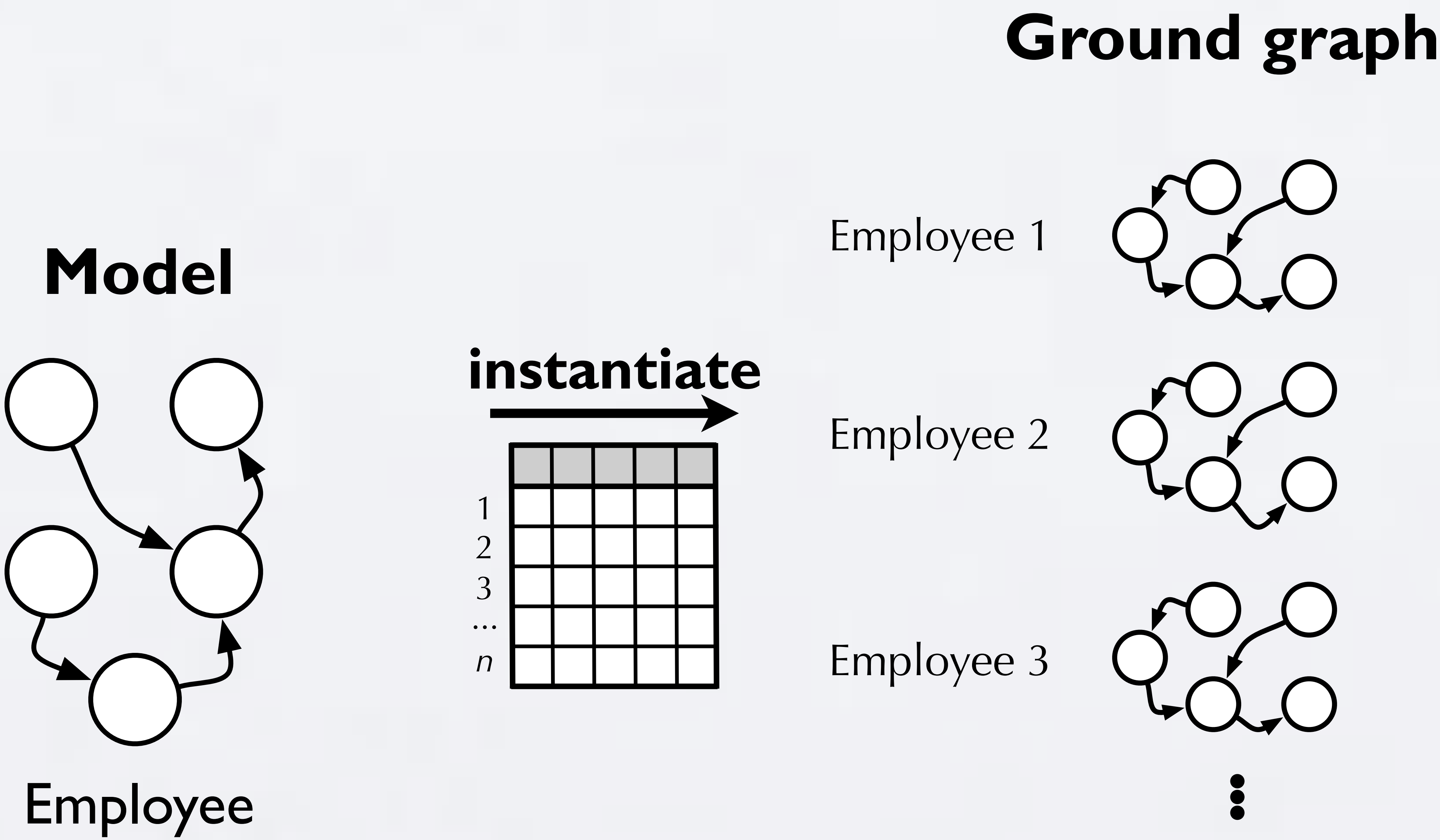


[Maier et al. 2013]

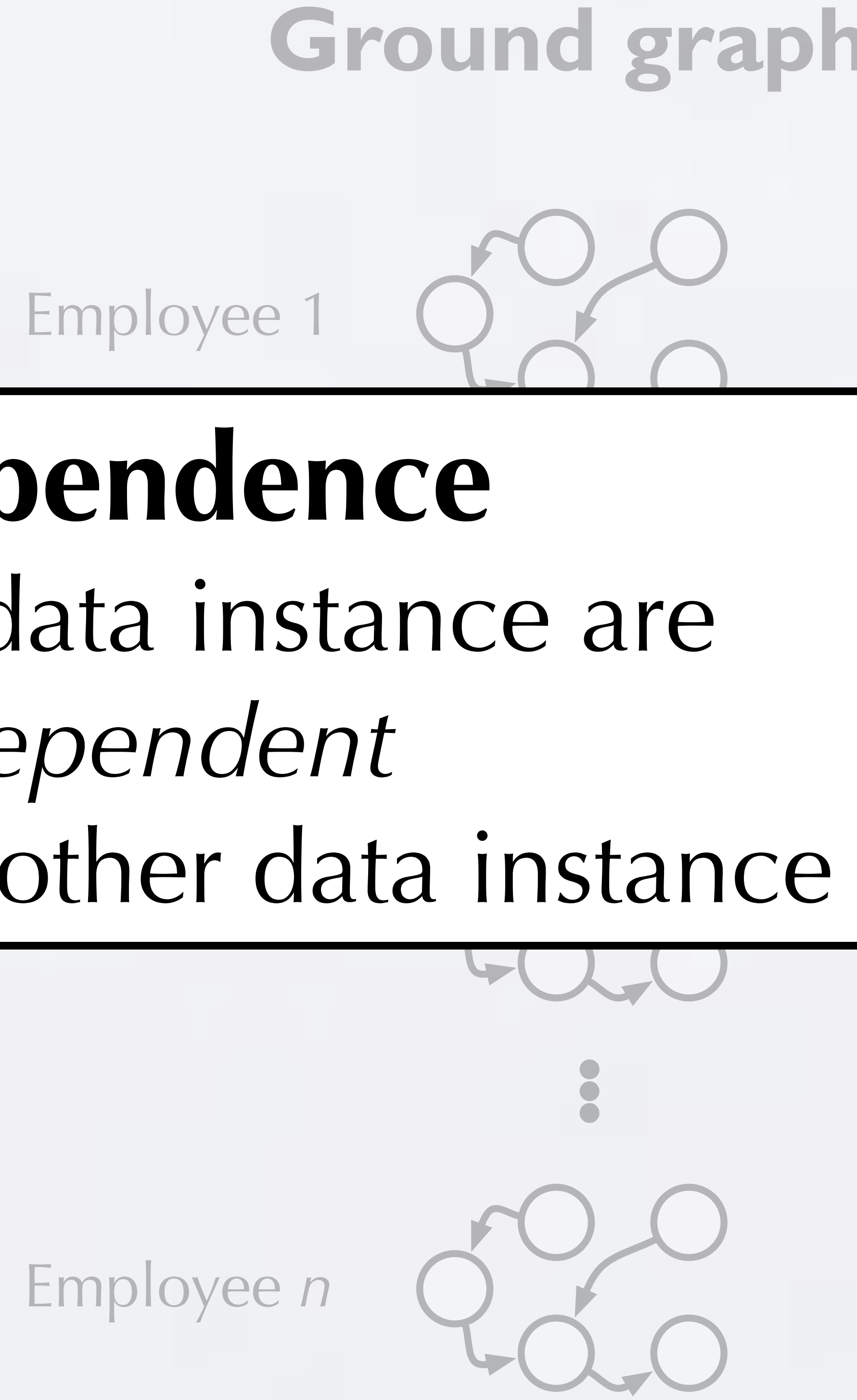
Topics

- **Background on relational data and models**
- Relational *d*-separation
- The RCD algorithm

Bayesian networks and i.i.d. data



Bayesian networks and i.i.d. data



Model

Employee

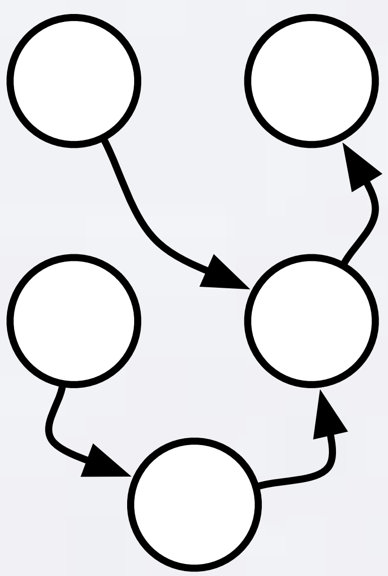
Instance independence

The variables on any data instance are *marginally independent* of all variables on every other data instance

Bayesian networks and i.i.d. data

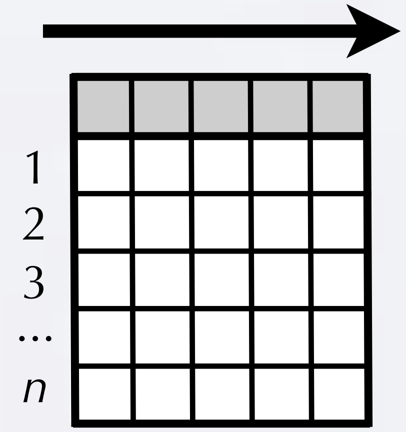
Ground graph

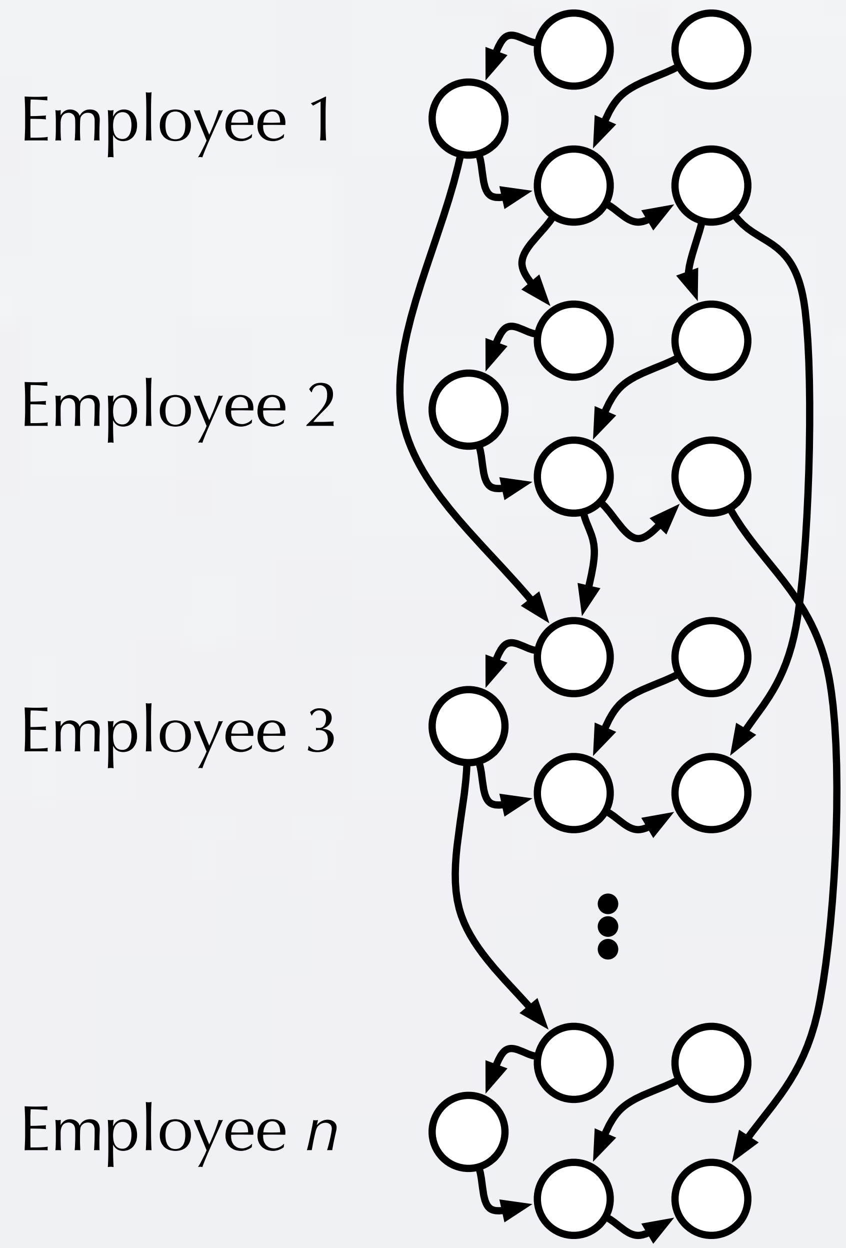
Model



Employee

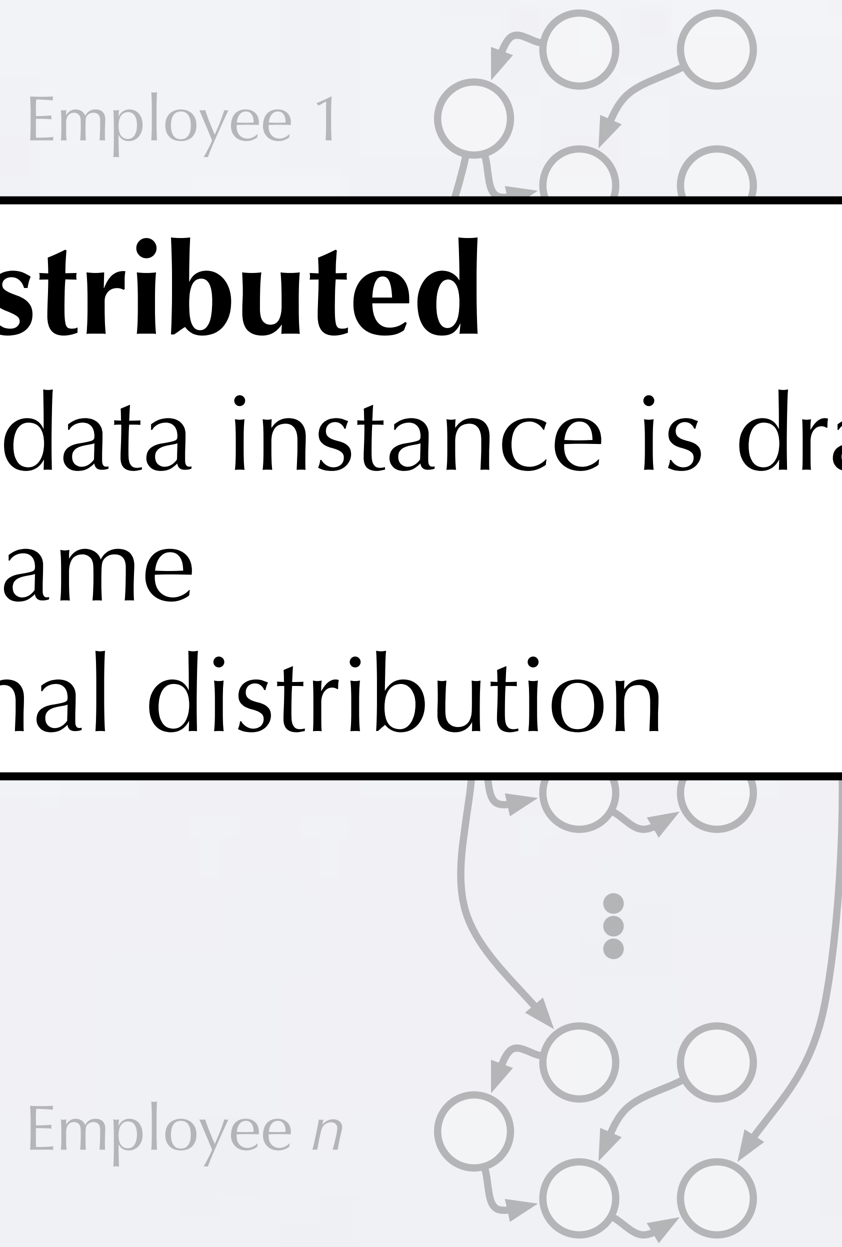
instantiate





Bayesian networks and i.i.d. data

Ground graph



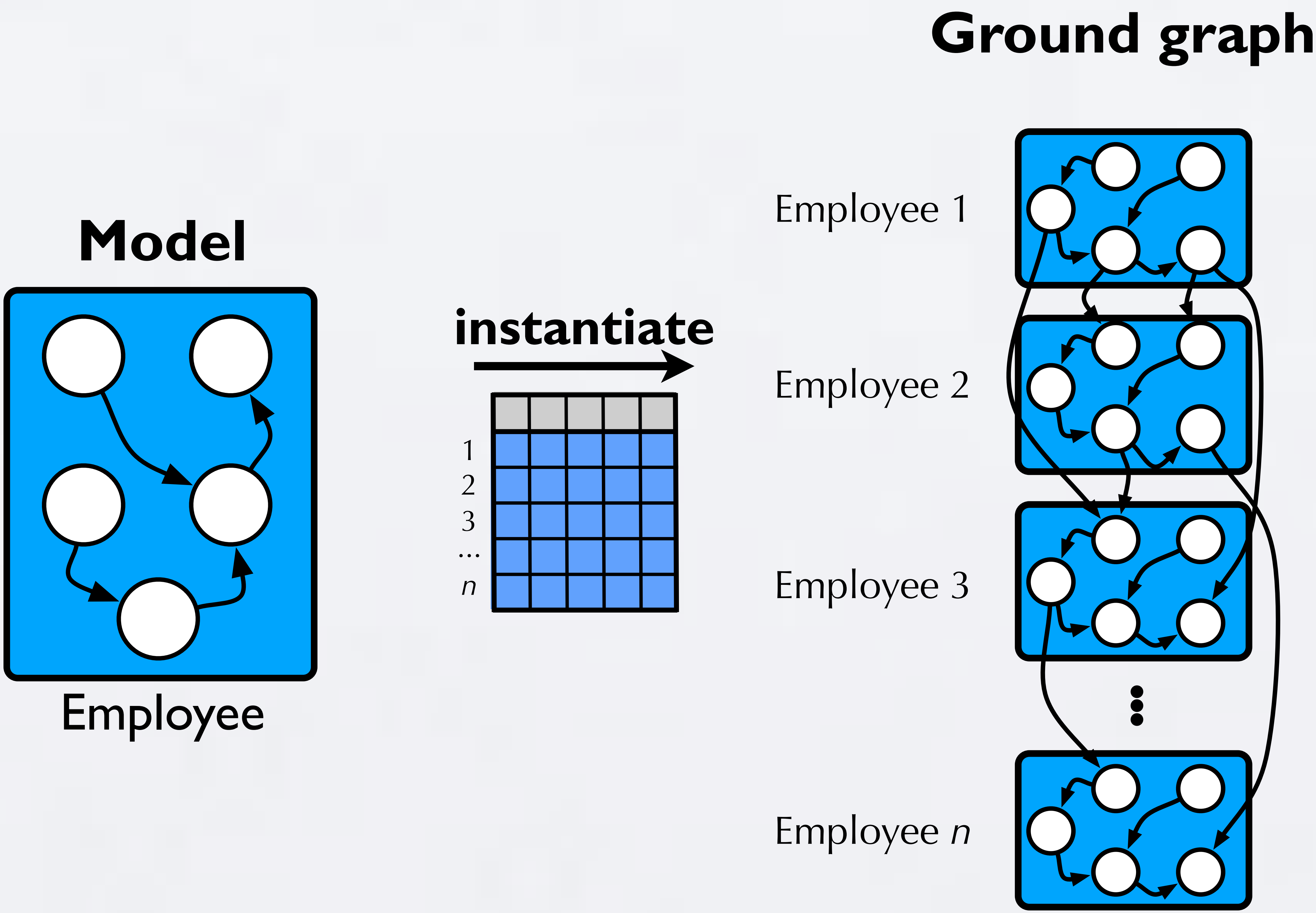
Model

Employee

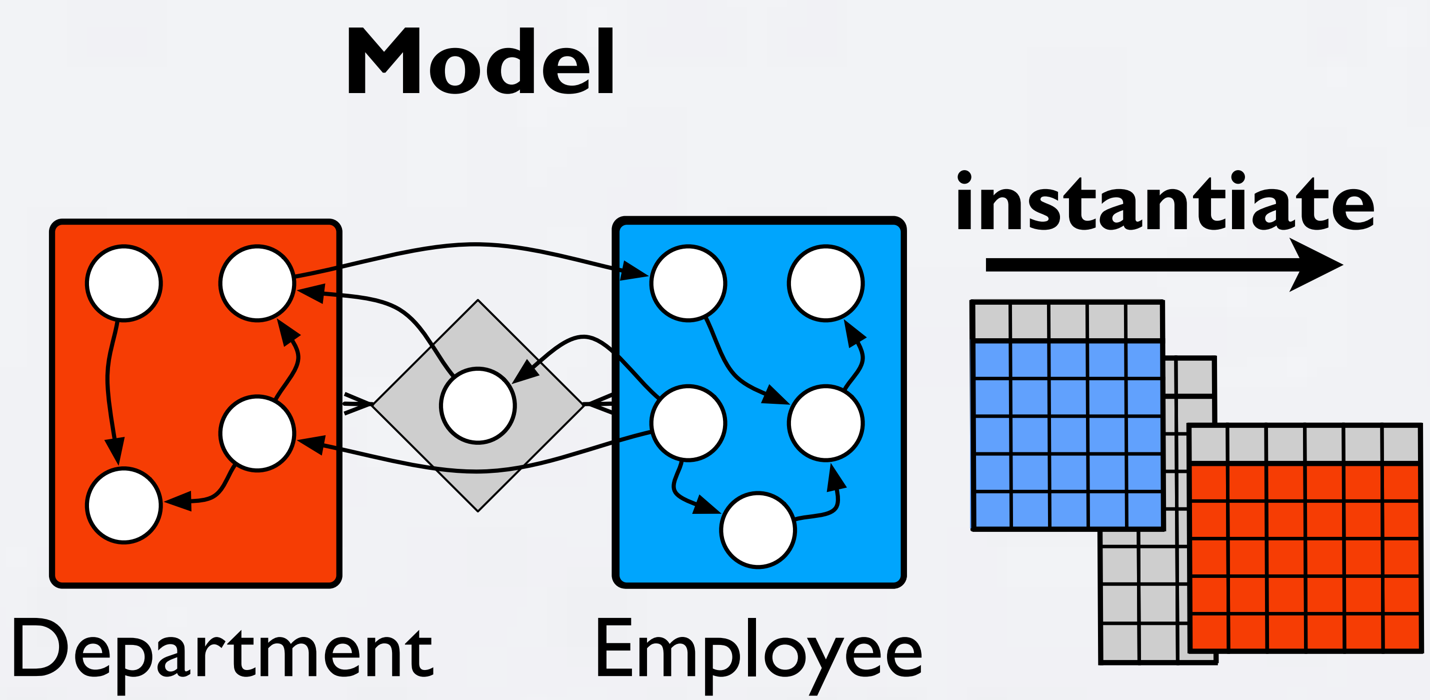
Identically distributed

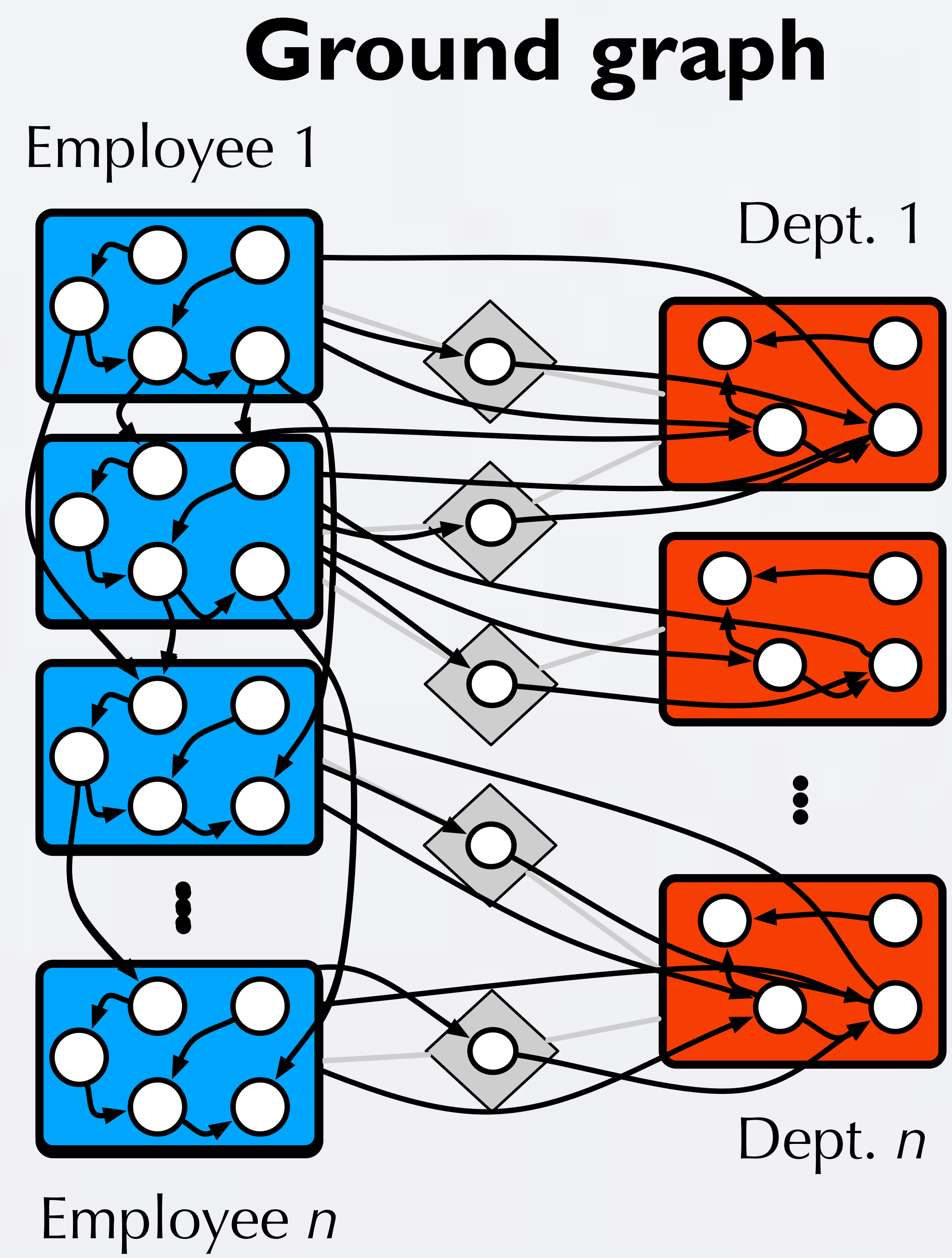
The same variable on every data instance is drawn from the same underlying conditional distribution

Bayesian networks and i.i.d. data

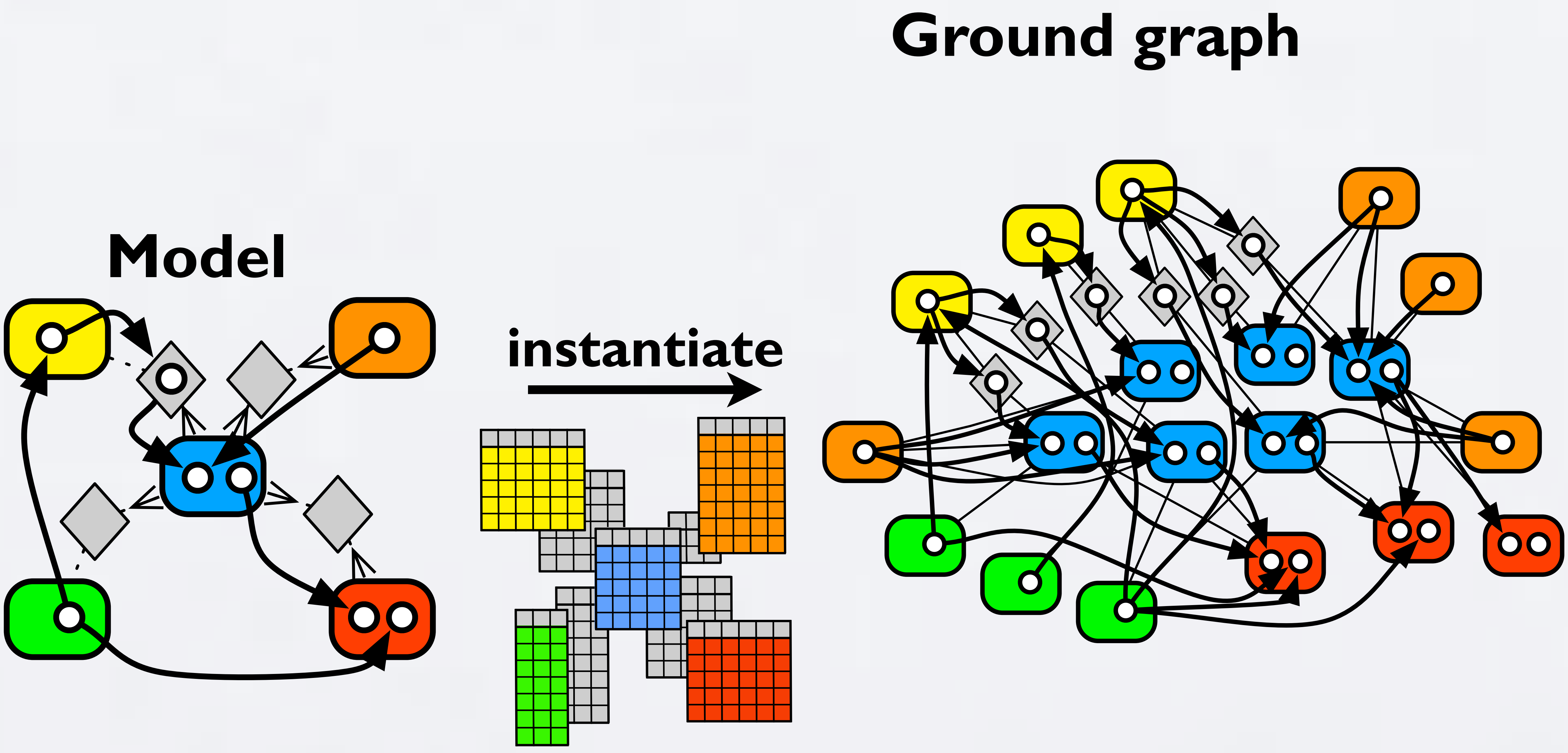


Bayesian networks and i.i.d. data





Relational models and non-i.i.d. data



Focus on directed graphical models of relational data to represent causal dependencies (e.g., PRMs, DAPER models, plate models).

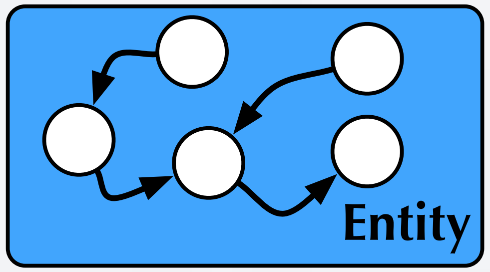
[Getoor, Friedman, Koller, Pfeffer & Taskar 2007; Heckerman, Meek & Koller 2007; Buntine 1994; Gilks, Thomas & Spiegelhalter et al. 1994]

Examples of relational data

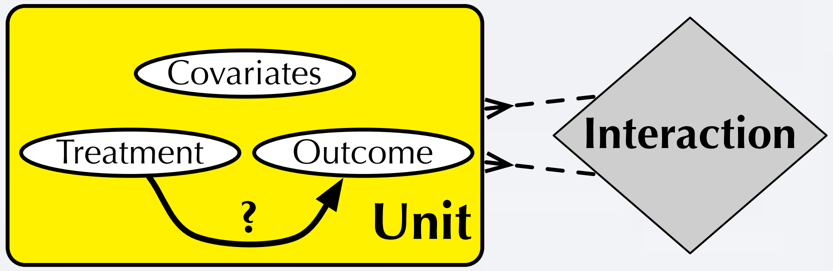
- Scholarly publishing
  - Researchers, articles, citations, venues
- Epidemiology
  - Individuals, contagions, treatments, interactions
- Sports
  - Athletes, teams, coaches, referees, competitive interactions
- Neuroscience
  - Molecular, cellular, system, cognitive levels
- Movie industry
  - Movies, actors, directors, studios, critic reviews
- Organizations
  - Employees, products, business units

Relational models generalize other classes of models

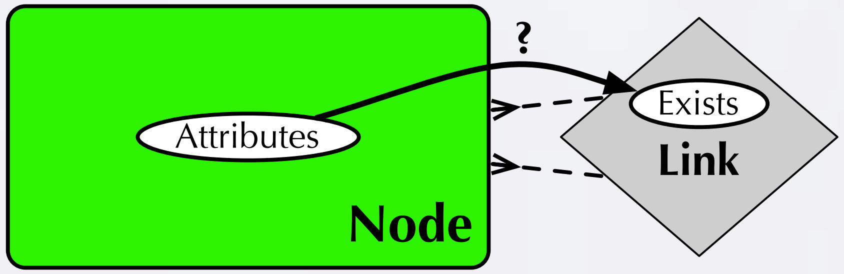
- Bayesian networks
  [Pearl 2000; Spirtes et al. 2000]



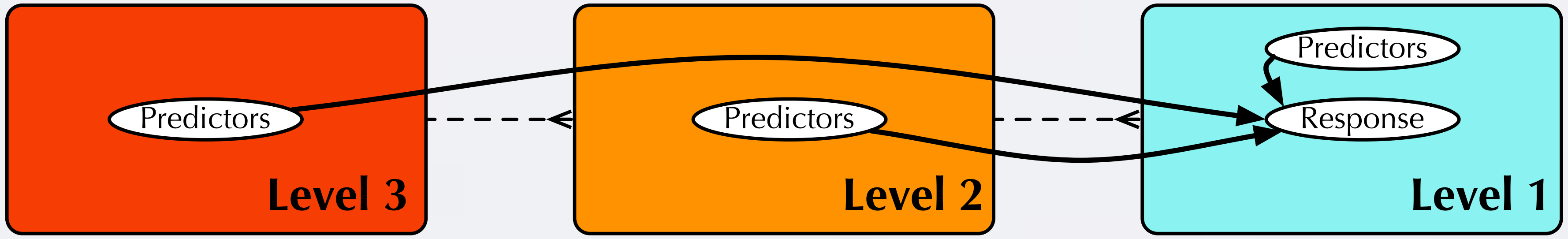
- Models of interference / spillover effects / violations of SUTVA
  [Rosenbaum 2007; Hudgens & Halloran 2008; Manski 2010; Tchetgen Tchetgen & VanderWeele 2012]



- Models of networks (e.g., p1, p*, ERGMs)
  [Holland & Leinhardt 1981; Snijders 2002; Robins et al. 2007]



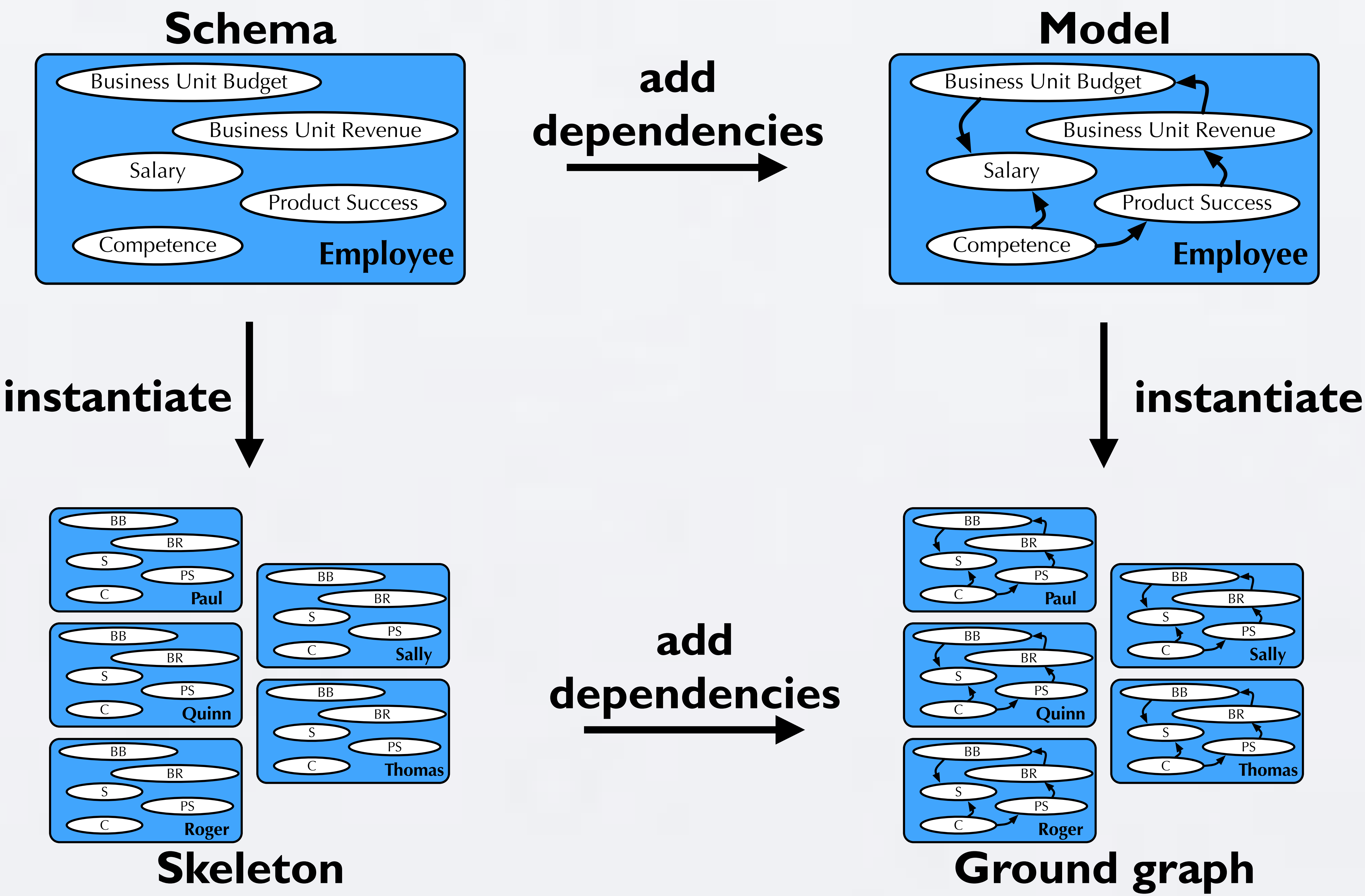
- Multilevel / hierarchical / random effects models
  [Gelman & Hill 2007]



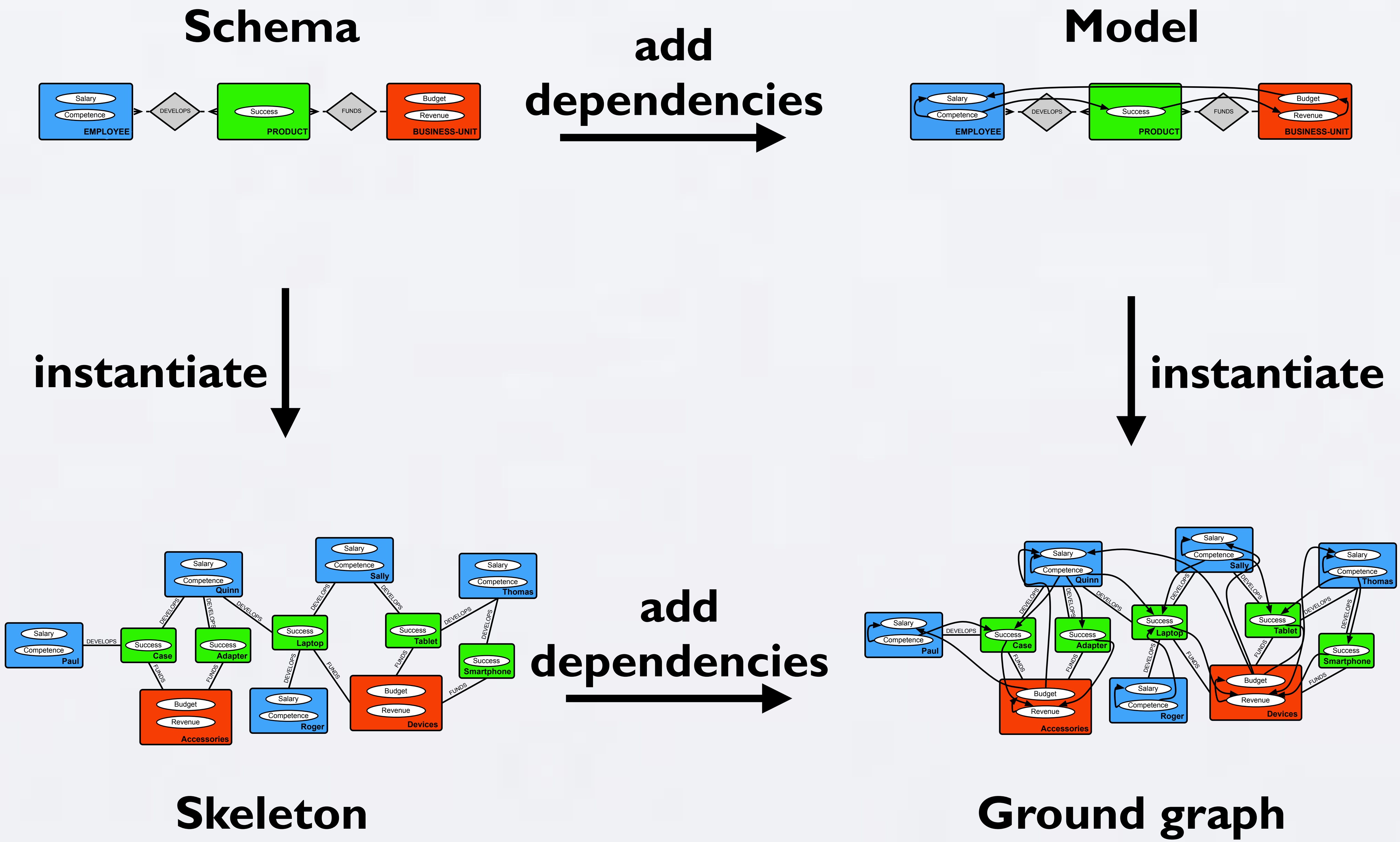
Overview of template models

$$
\begin{tikzcd}
\textbf{Schema} \arrow[r, "\textbf{add dependencies}"] \arrow[d, "\textbf{instantiate}"'] & \textbf{Model} \arrow[d, "\textbf{instantiate}"] \\
\textbf{Skeleton} \arrow[r, "\textbf{add dependencies}"] & \textbf{Ground graph}
\end{tikzcd}
$$

Bayesian networks as template models



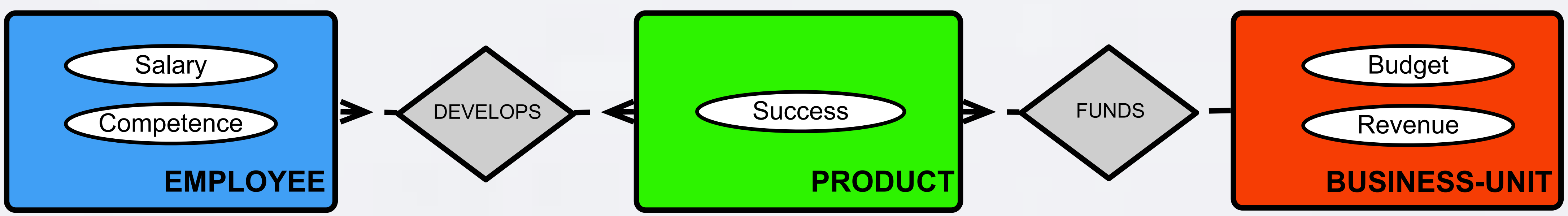
Relational models as template models



Relational schemas

A relational schema describes what relational data exist

- Expected types of items
- Expected attributes
- How often entities can participate in relationships

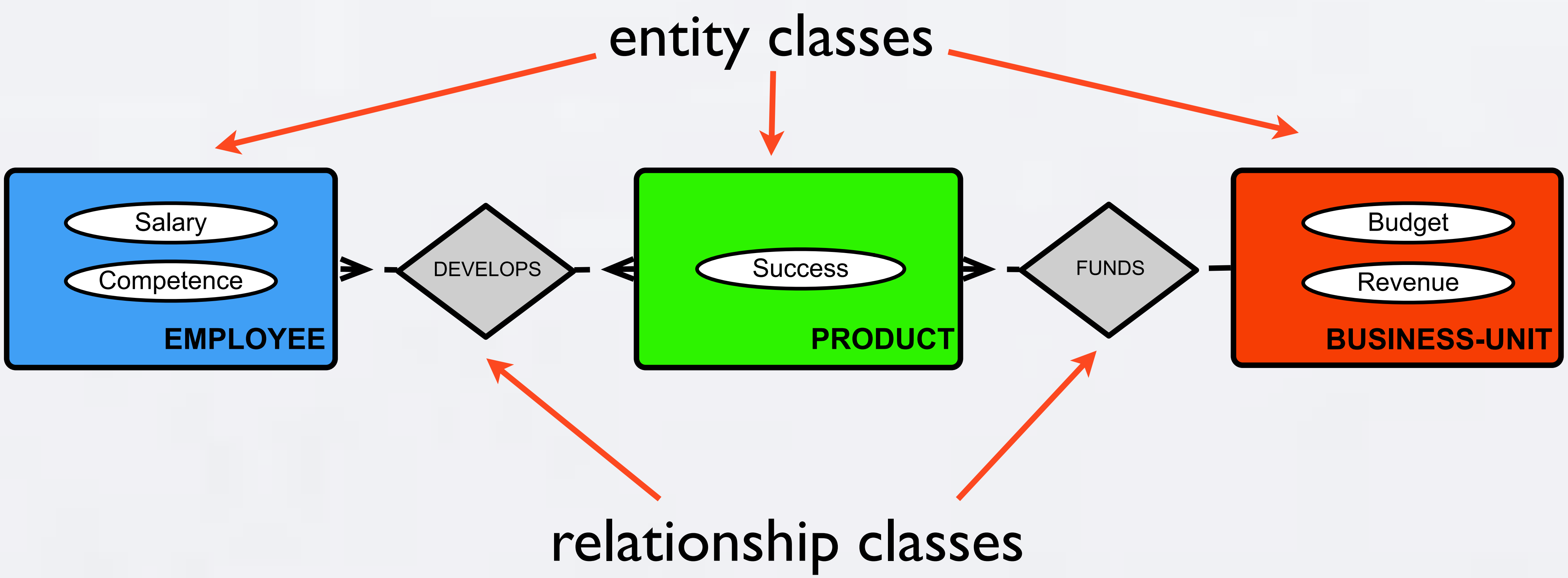


[Heckerman et al. 2007]

Relational schemas

A relational schema describes what relational data exist

- Expected types of items
- Expected attributes
- How often entities can participate in relationships

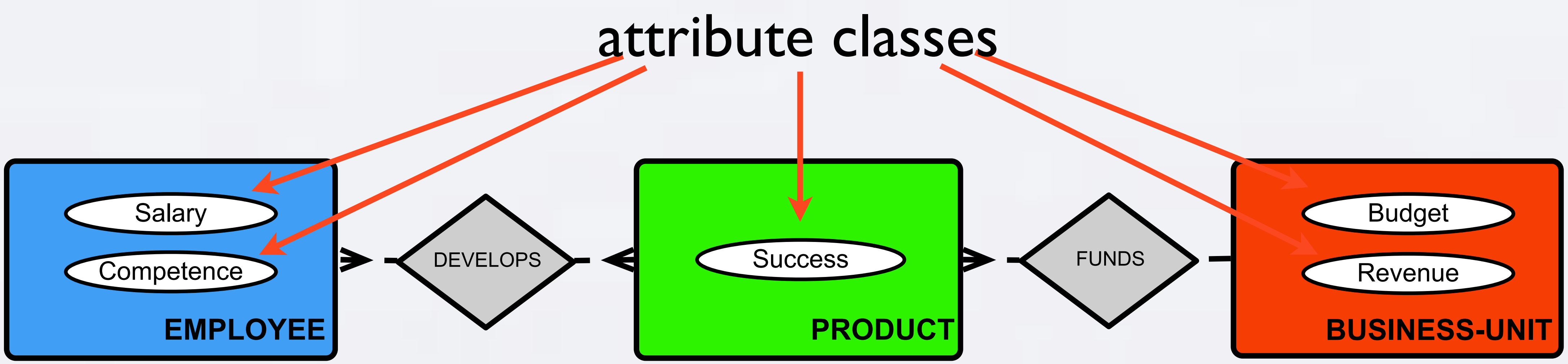


[Heckerman et al. 2007]

Relational schemas

A relational schema describes what relational data exist

- Expected types of items
- Expected attributes
- How often entities can participate in relationships

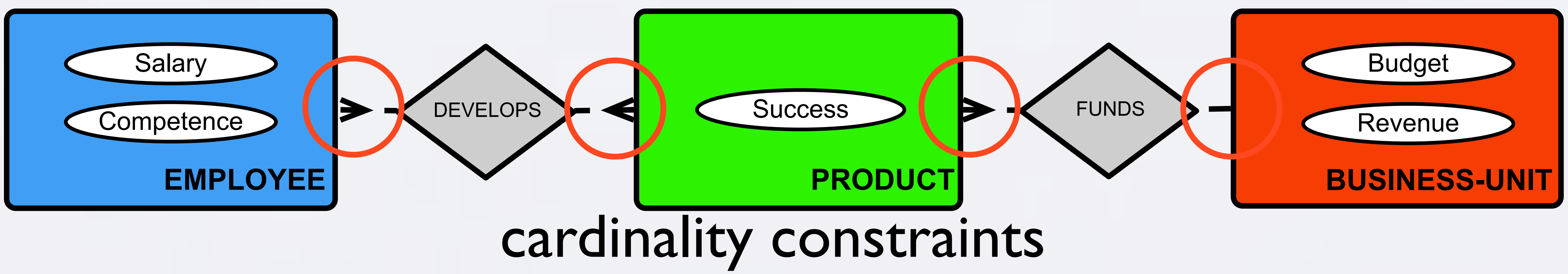


[Heckerman et al. 2007]

Relational schemas

A relational schema describes what relational data exist

- Expected types of items
- Expected attributes
- How often entities can participate in relationships

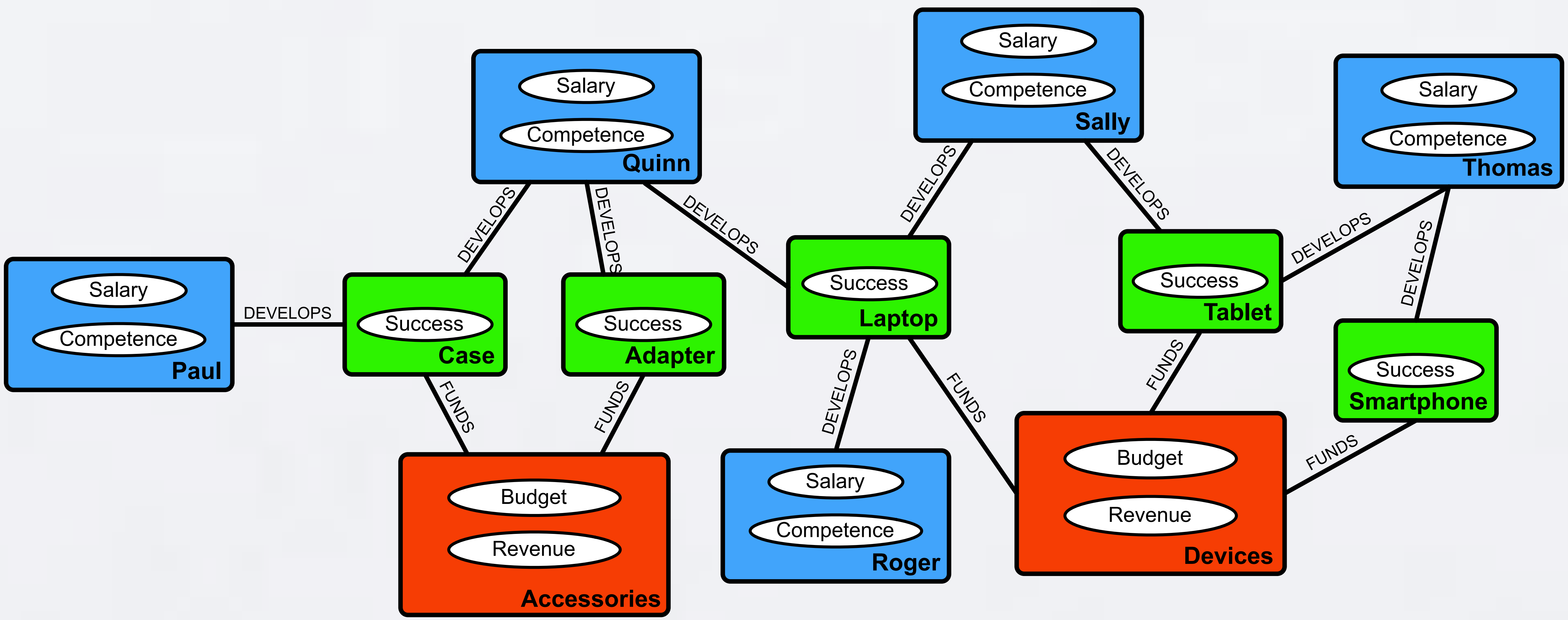


[Heckerman et al. 2007]

Relational skeletons

A relational skeleton is an instantiated relational schema

- Set of entity and relationship instances
- Adheres to cardinality constraints



[Heckerman et al. 2007]

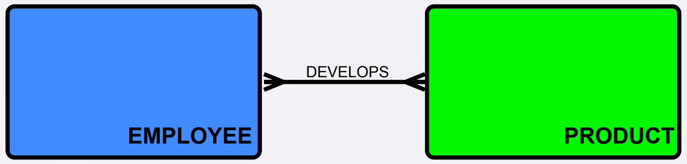
Relational paths

A relational path is an alternating sequence of entity and relationship classes

- Specifies how to get from one type of item to another
- Building blocks for relational variables
- Length limited by domain-specific, user-defined *hop threshold*
- Base item on path has the special designation of *perspective*

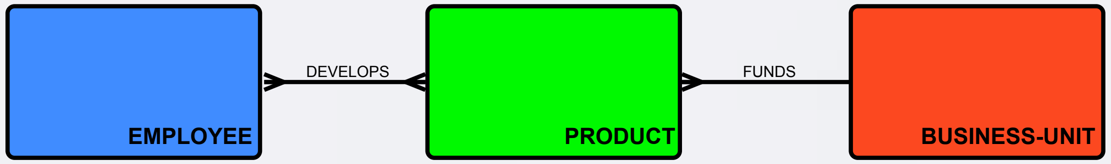
An employee's developed products
(2 hops)

[Employee, Develops, Product]



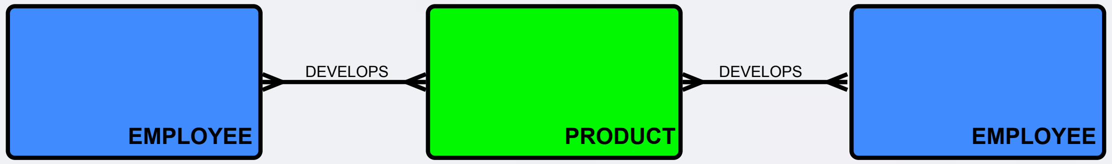
An employee's funding business units
(4 hops)

[Employee, Develops, Product, Funds, Business-Unit]



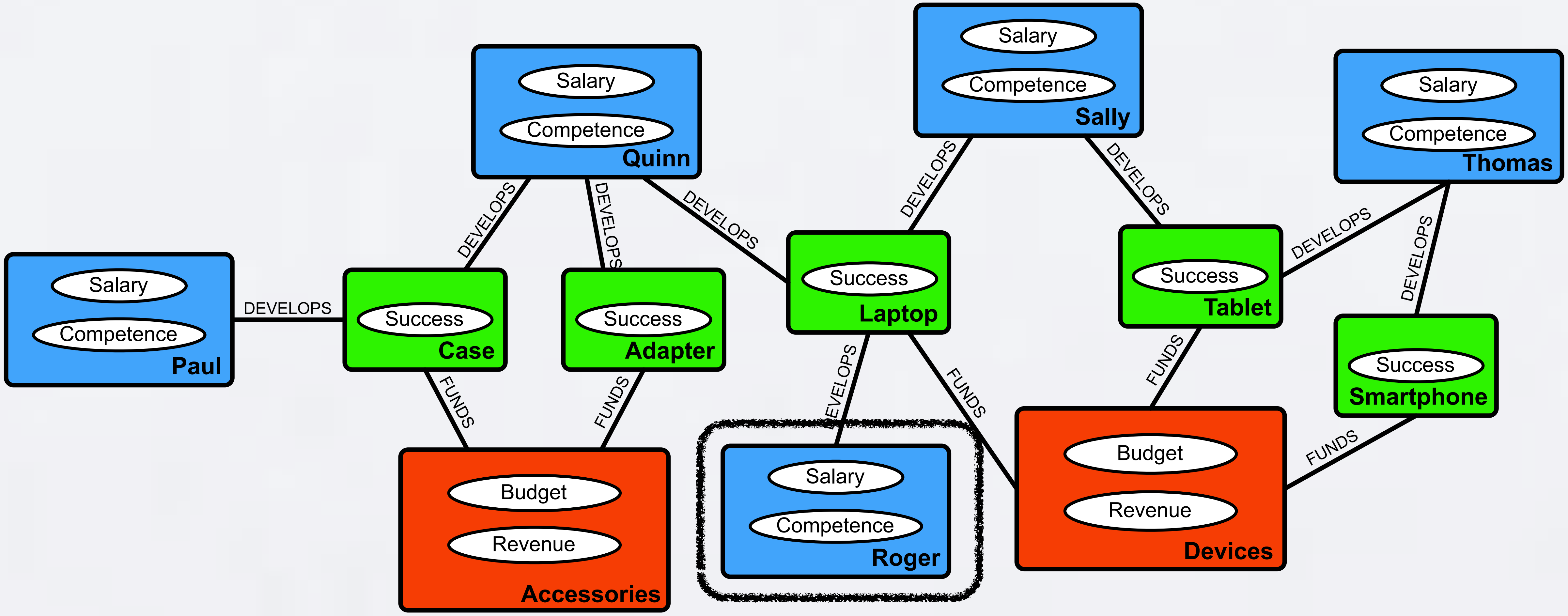
An employee's co-workers
(4 hops)

[Employee, Develops, Product, Develops, Employee]



Terminal sets of relational paths

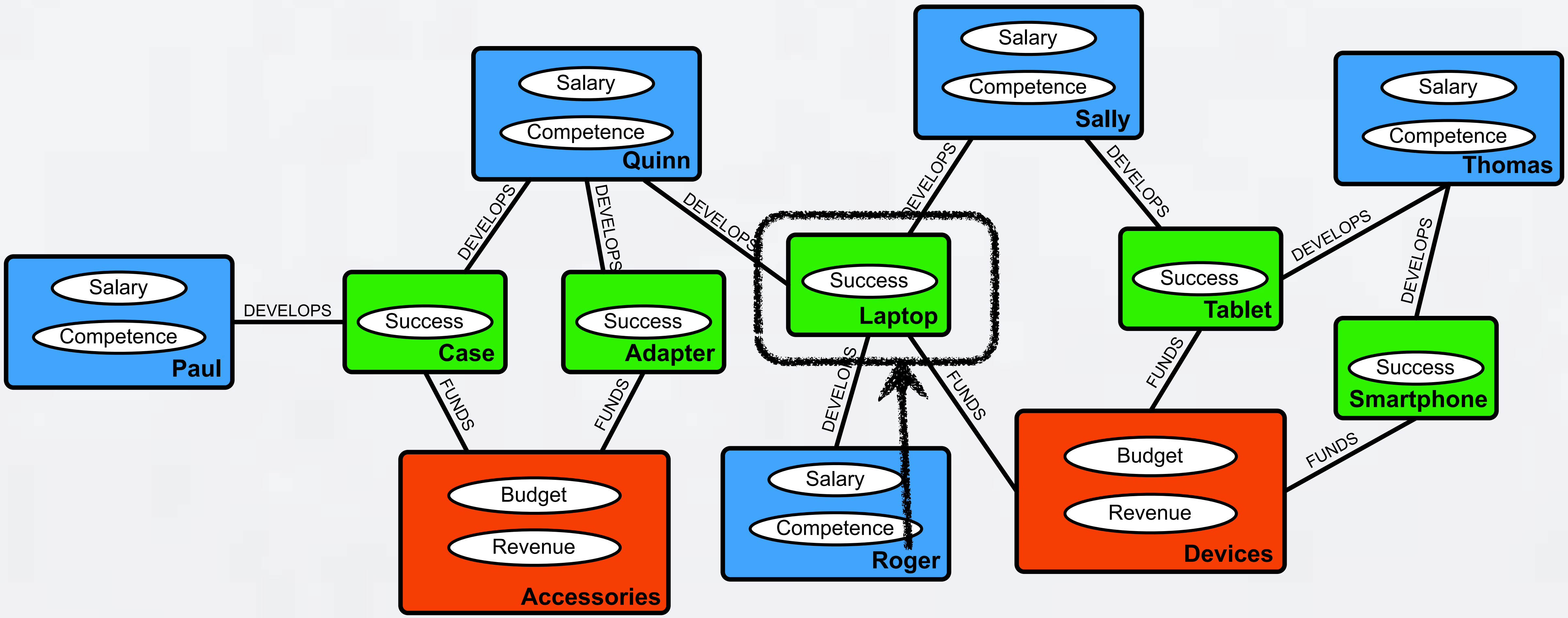
The set of terminal items reached by a particular base item instance via a relational path on a relational skeleton



[Employee]|$_{\text{Roger}}$ = {Roger}

Terminal sets of relational paths

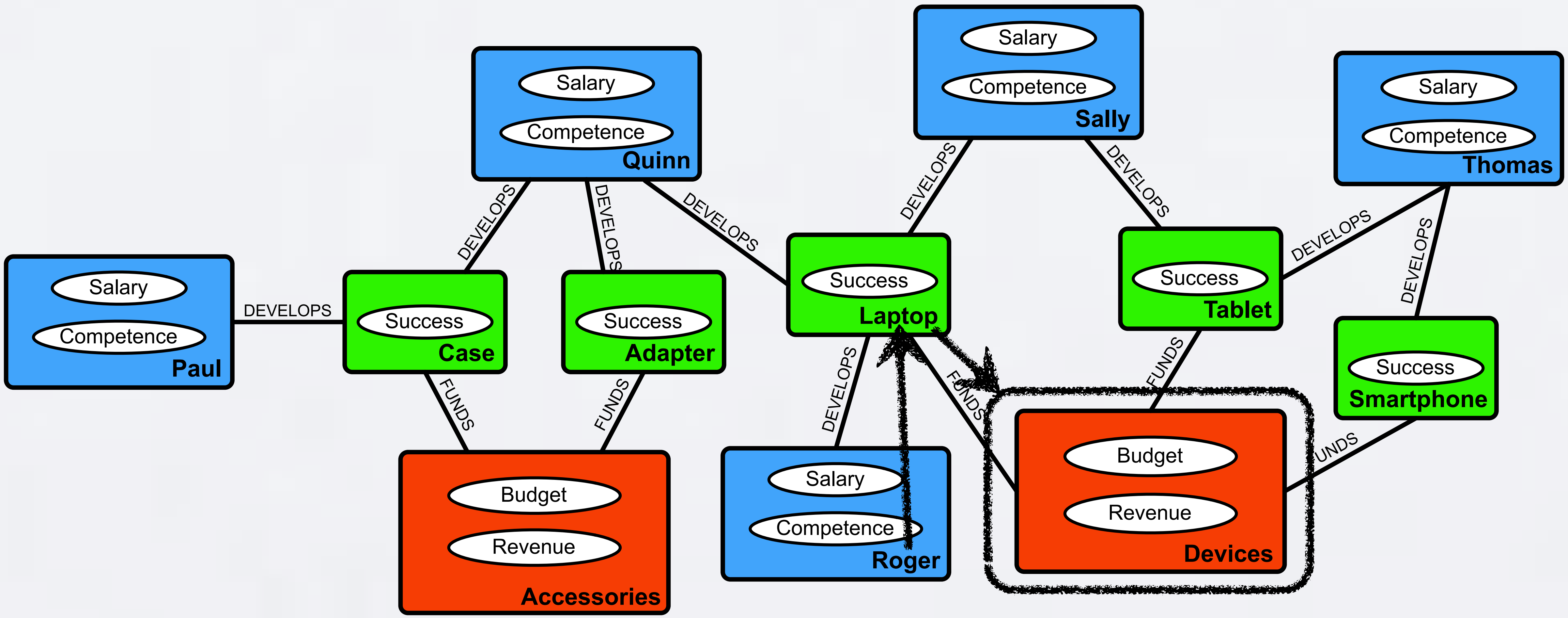
The set of terminal items reached by a particular base item instance via a relational path on a relational skeleton



[Employee, Develops, Product]|$_{\text{Roger}}$ = {Laptop}

Terminal sets of relational paths

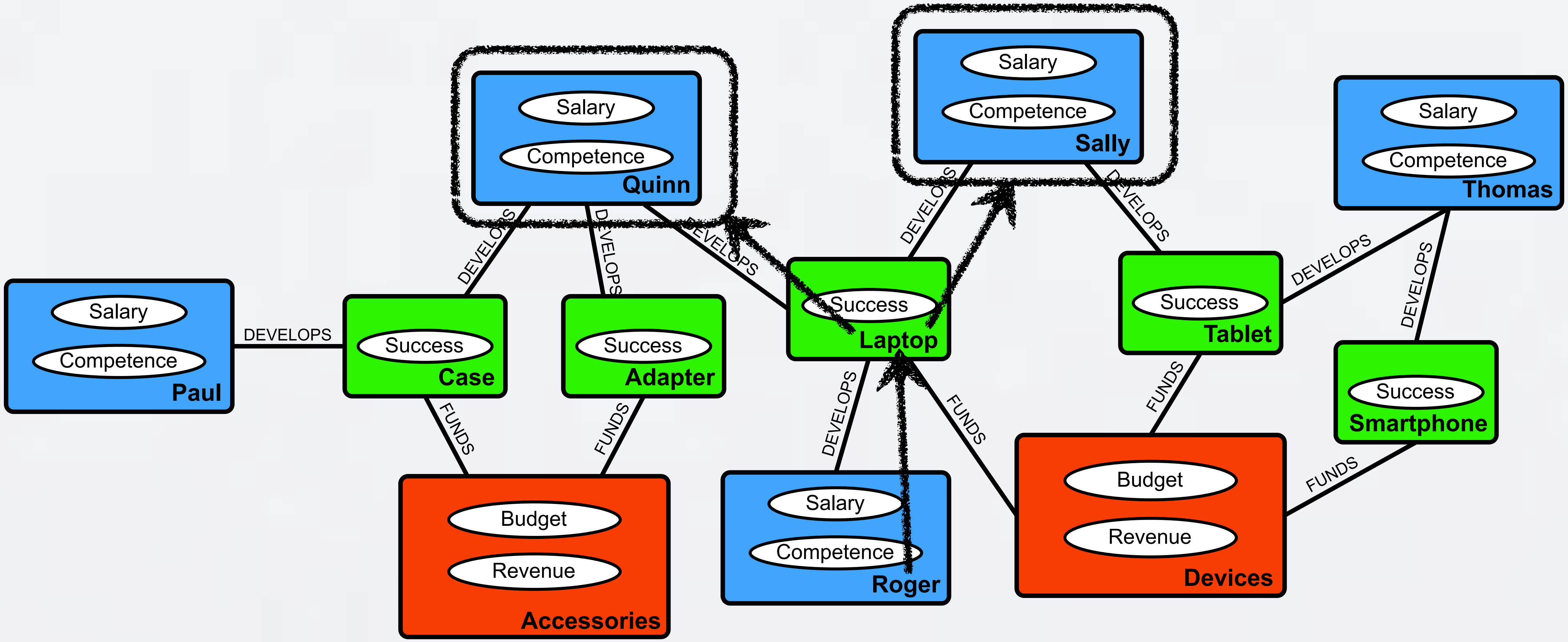
The set of terminal items reached by a particular base item instance via a relational path on a relational skeleton



[Employee, Develops, Product, Funds, Business-Unit]|$_{\text{Roger}}$ = {Devices}

Terminal sets of relational paths

The set of terminal items reached by a particular base item instance via a relational path on a relational skeleton



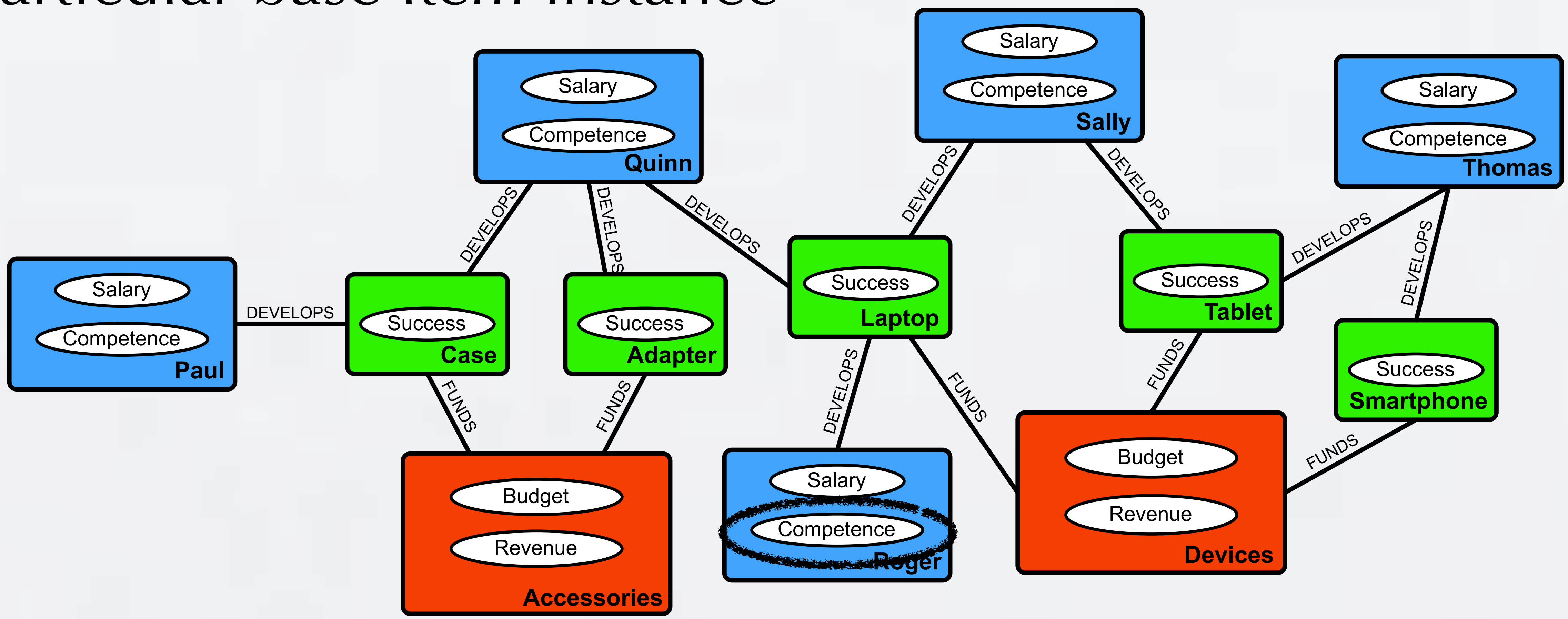
[Employee, Develops, Product, Develops, Employee]|$_{\text{Roger}}$ = {Quinn, Sally}

Relational variables and their terminal sets

Relational variables attach an attribute to a relational path

- Building blocks for relational dependencies

Instantiations are sets of random variable instances for a particular base item instance



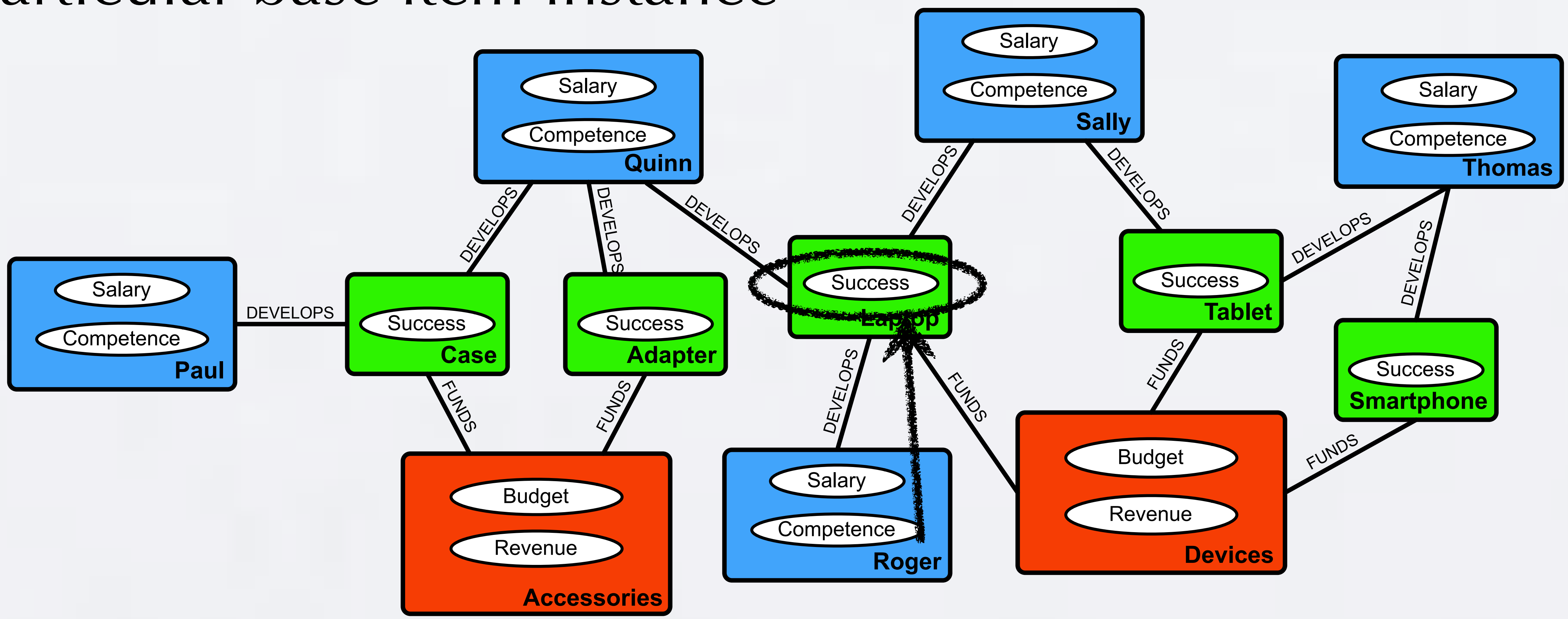
$[\text{Employee}].\text{Competence}|_{\text{Roger}} = \{\text{Roger.Competence}\}$

Relational variables and their terminal sets

Relational variables attach an attribute to a relational path

- Building blocks for relational dependencies

Instantiations are sets of random variable instances for a particular base item instance



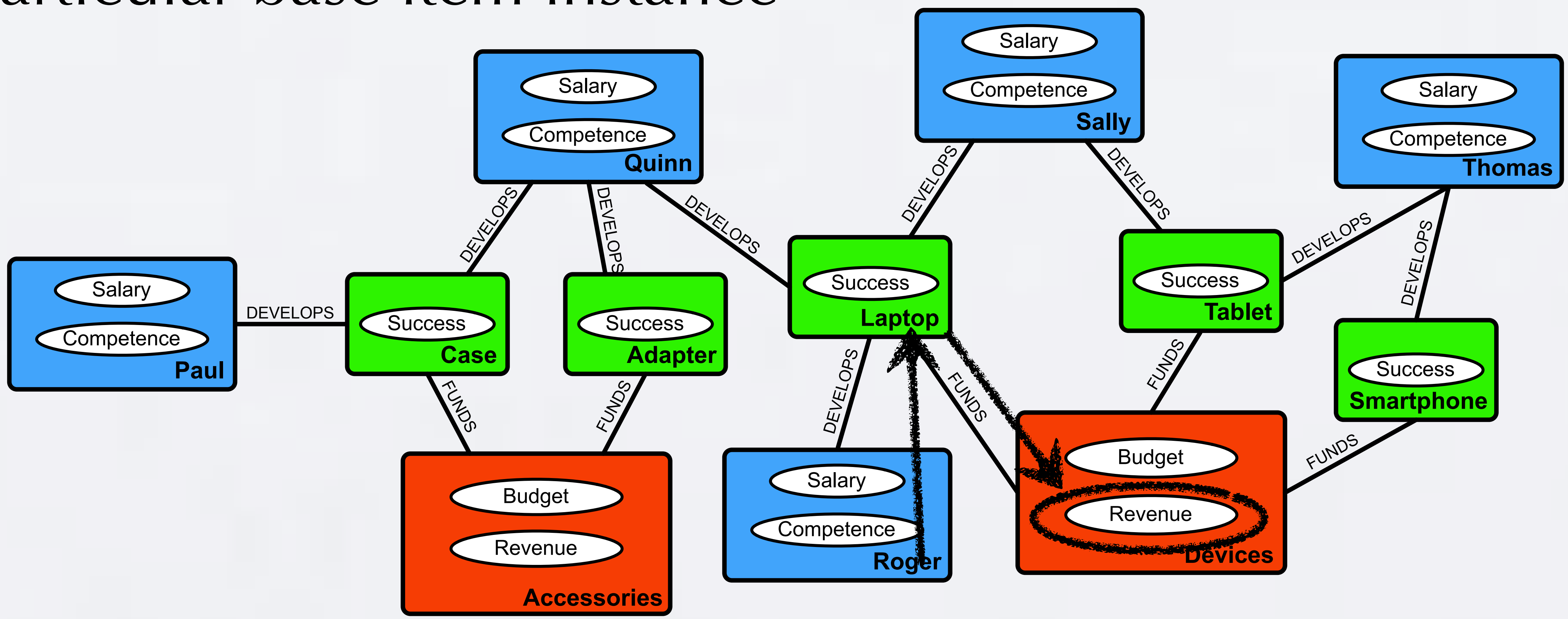
[Employee, Develops, Product].Success$|_{\text{Roger}}$ = {Laptop.Success}

Relational variables and their terminal sets

Relational variables attach an attribute to a relational path

- Building blocks for relational dependencies

Instantiations are sets of random variable instances for a particular base item instance



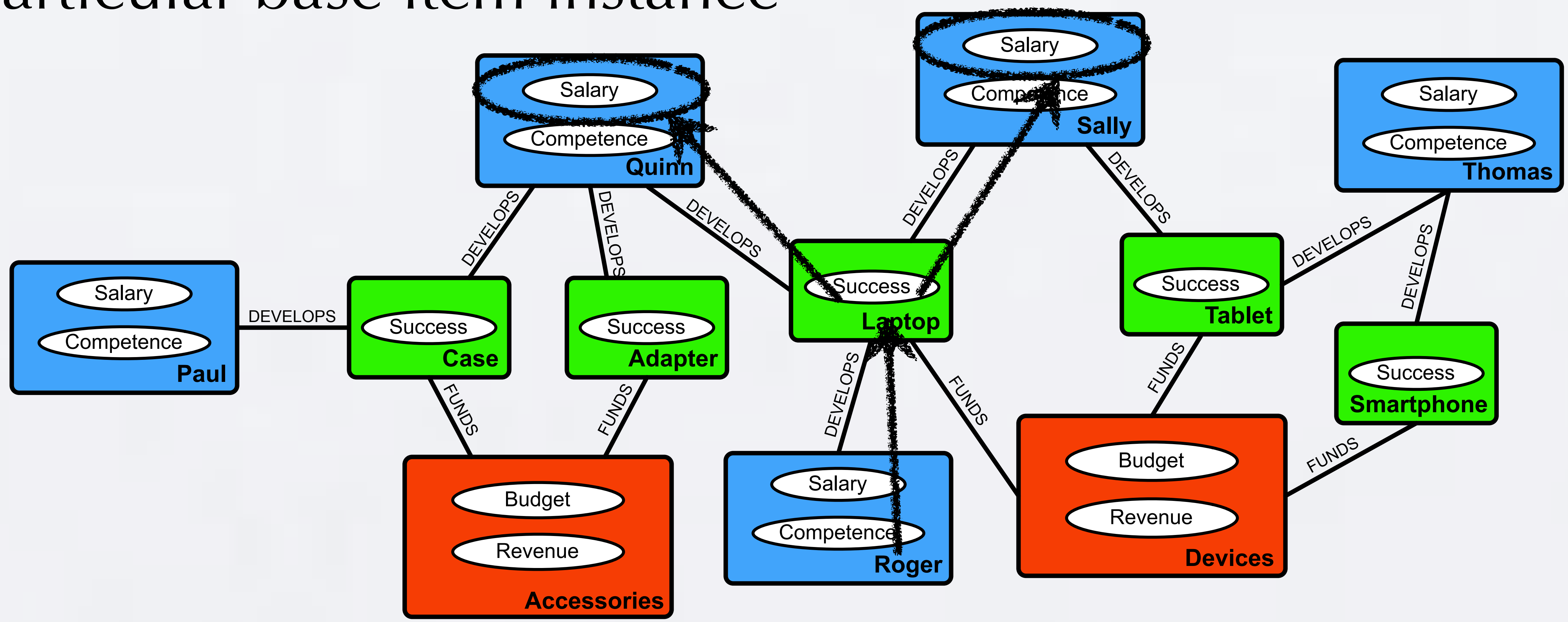
[Employee, Develops, Product, Funds, Business-Unit].Revenue|$_{\text{Roger}}$ = {Devices.Revenue}

Relational variables and their terminal sets

Relational variables attach an attribute to a relational path

- Building blocks for relational dependencies

Instantiations are sets of random variable instances for a particular base item instance



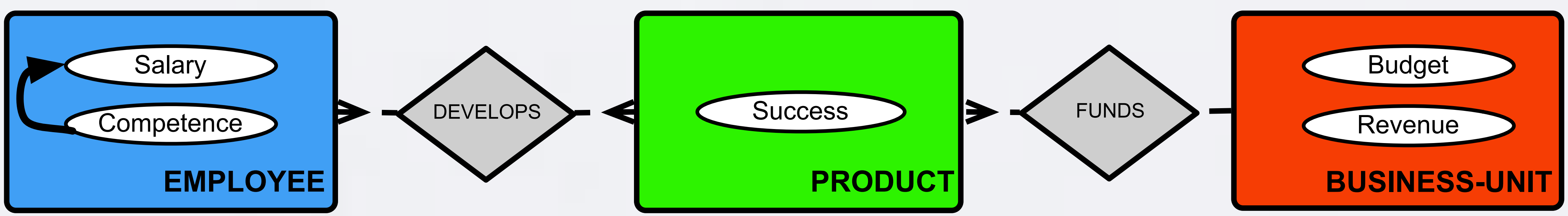
[Employee, Develops, Product, Develops, Employee].Salary$|_{\text{Roger}}$ = {Quinn.Salary, Sally.Salary}

Relational dependencies and models

A relational dependency combines a pair of relational variables with a common perspective

- Referred to as treatment/outcome, cause/effect, parent/child
- Canonical form has singleton outcome path
- Building blocks for relational models

A relational model is a collection of relational dependencies defined over a relational schema



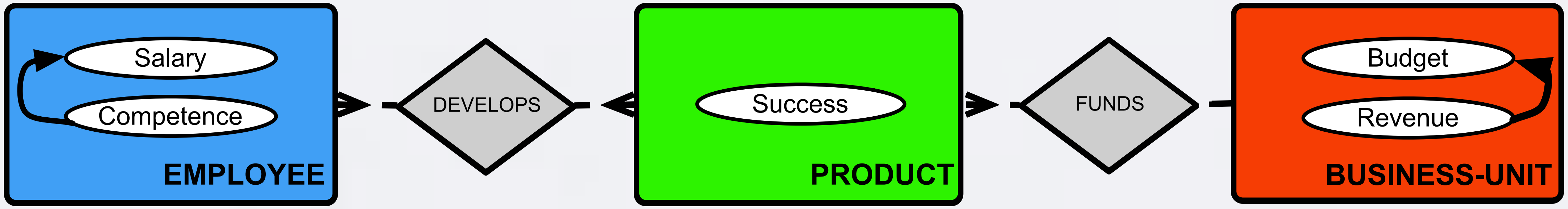
[EMPLOYEE].Competence $\longrightarrow$ [EMPLOYEE].Salary

Relational dependencies and models

A relational dependency combines a pair of relational variables with a common perspective

- Referred to as treatment/outcome, cause/effect, parent/child
- Canonical form has singleton outcome path
- Building blocks for relational models

A relational model is a collection of relational dependencies defined over a relational schema



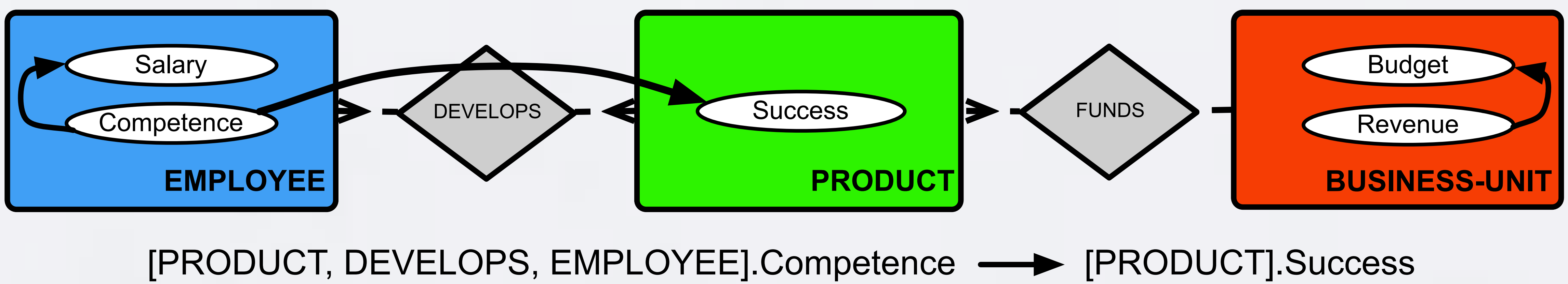
[BUSINESS-UNIT].Revenue $\longrightarrow$ [BUSINESS-UNIT].Budget

Relational dependencies and models

A relational dependency combines a pair of relational variables with a common perspective

- Referred to as treatment/outcome, cause/effect, parent/child
- Canonical form has singleton outcome path
- Building blocks for relational models

A relational model is a collection of relational dependencies defined over a relational schema

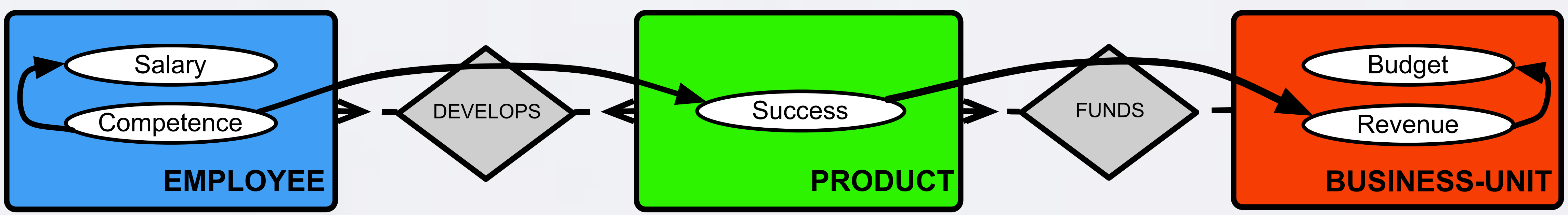


Relational dependencies and models

A relational dependency combines a pair of relational variables with a common perspective

- Referred to as treatment/outcome, cause/effect, parent/child
- Canonical form has singleton outcome path
- Building blocks for relational models

A relational model is a collection of relational dependencies defined over a relational schema



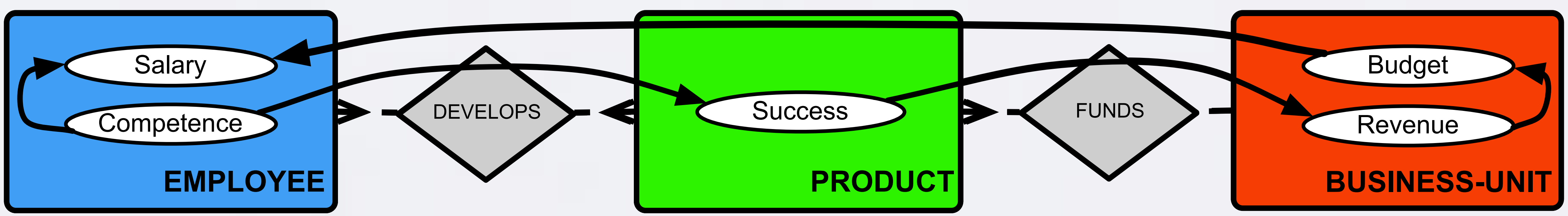
[BUSINESS-UNIT, FUNDS, PRODUCT].Success ⟶ [BUSINESS-UNIT].Revenue

Relational dependencies and models

A relational dependency combines a pair of relational variables with a common perspective

- Referred to as treatment/outcome, cause/effect, parent/child
- Canonical form has singleton outcome path
- Building blocks for relational models

A relational model is a collection of relational dependencies defined over a relational schema

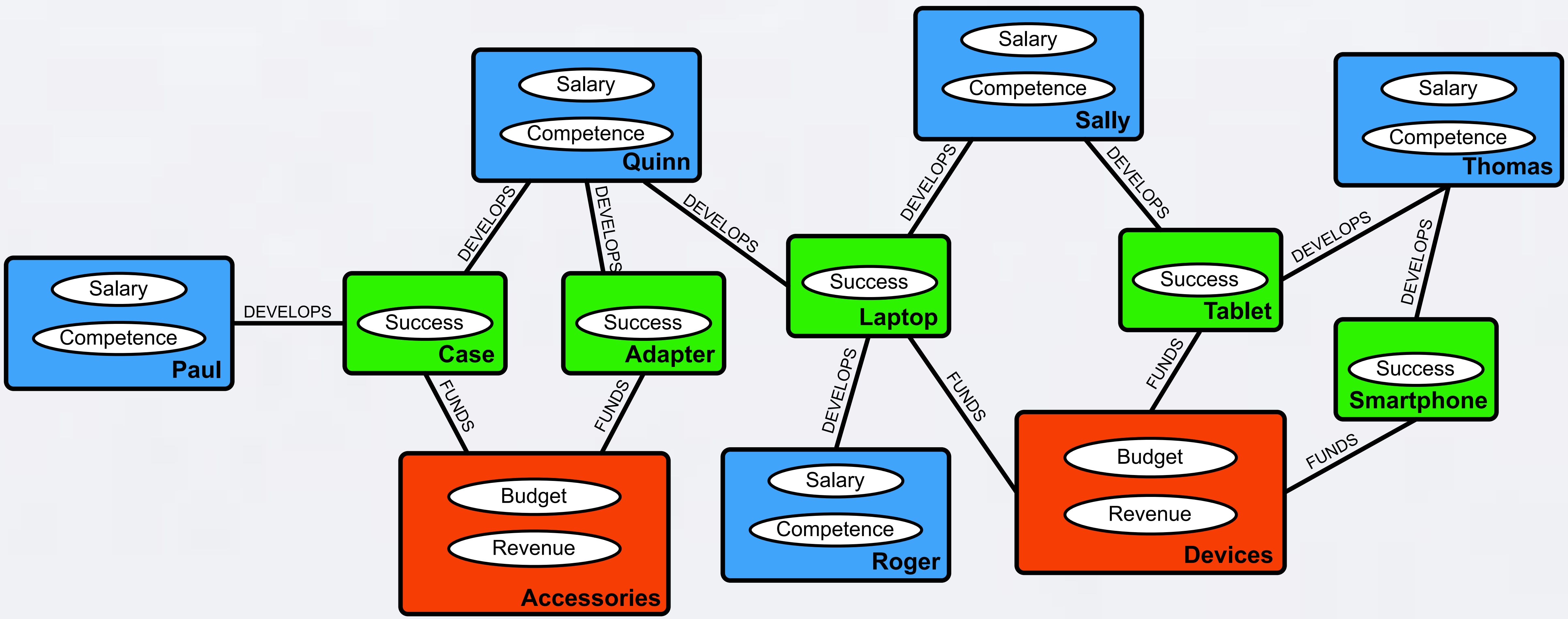


[EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT].Budget $\longrightarrow$ [EMPLOYEE].Salary

Ground graphs

A ground graph is an instantiated relational model for a given relational skeleton

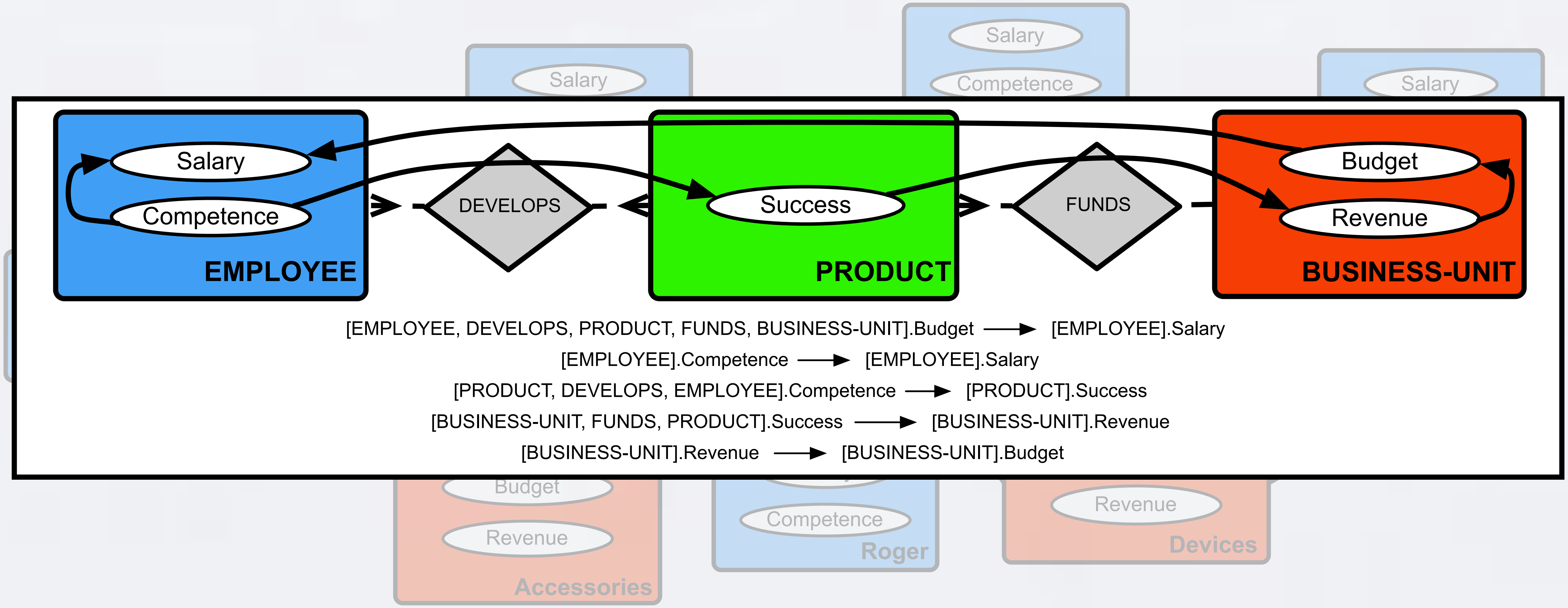
- Applies relational dependencies to the variable instances governed by a relational skeleton
- Connects the terminal sets of the parent relational variable to the terminal set of the child relational variable



Ground graphs

A ground graph is an instantiated relational model for a given relational skeleton

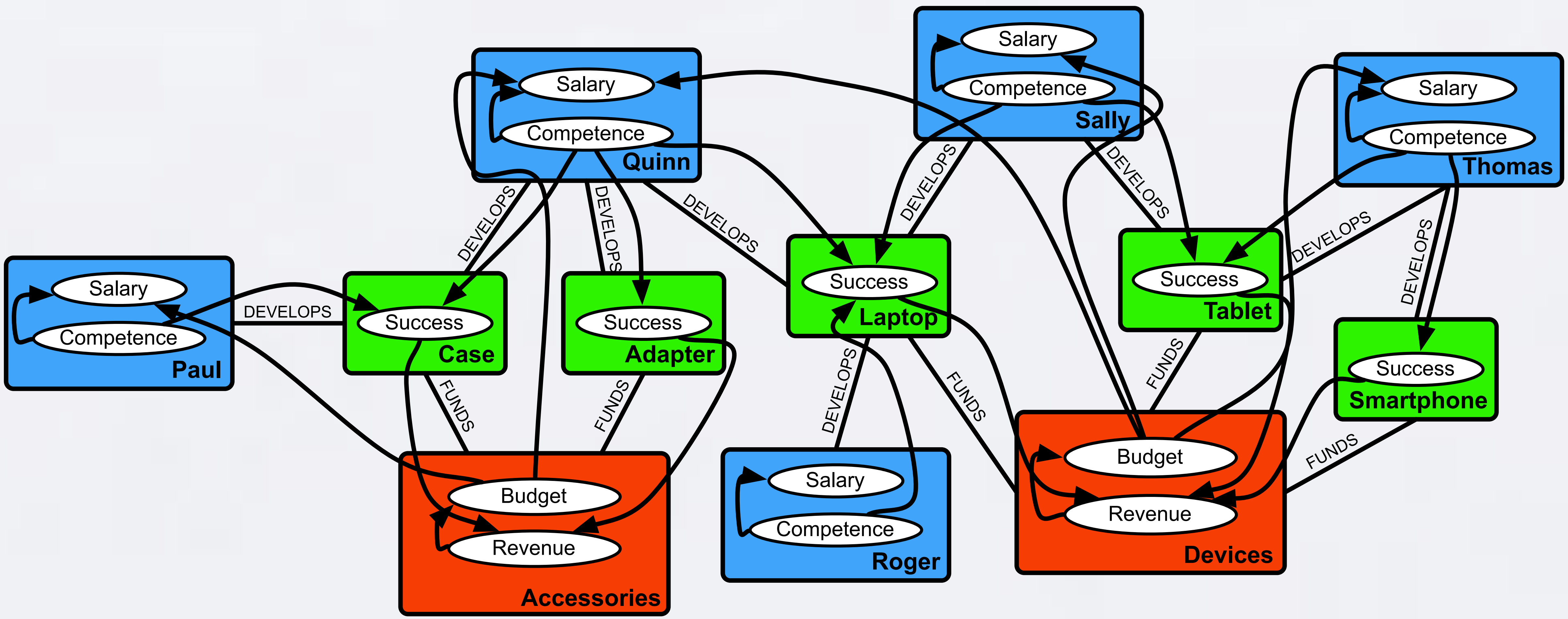
- Applies relational dependencies to the variable instances governed by a relational skeleton
- Connects the terminal sets of the parent relational variable to the terminal set of the child relational variable



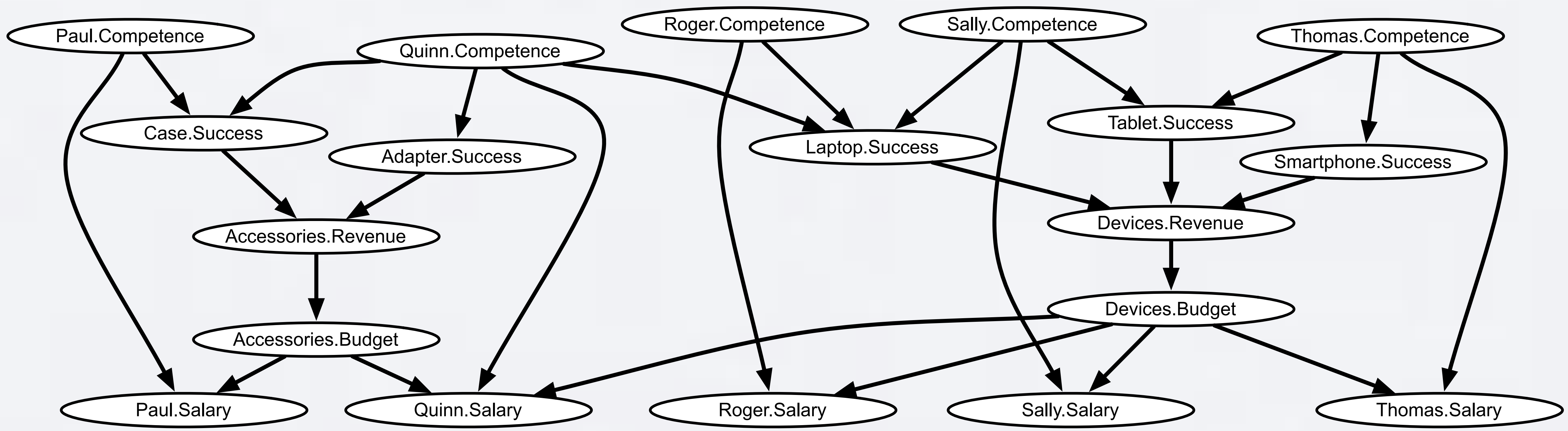
Ground graphs

A ground graph is an instantiated relational model for a given relational skeleton

- Applies relational dependencies to the variable instances governed by a relational skeleton
- Connects the terminal sets of the parent relational variable to the terminal set of the child relational variable

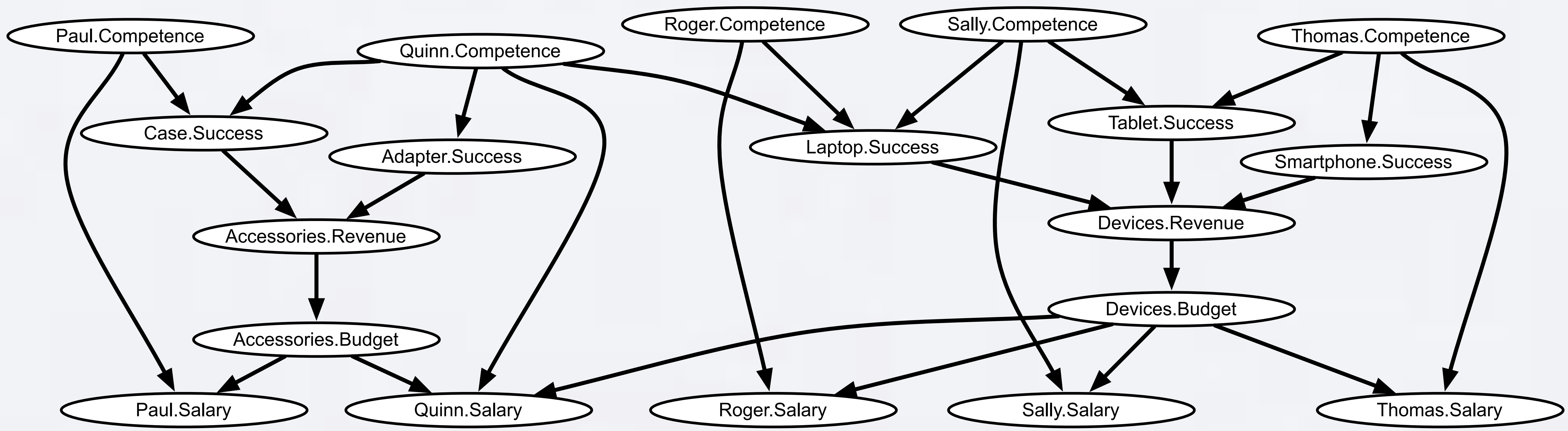


Probabilistic semantics of ground graphs



- If a ground graph is acyclic, then it has a coherent joint probability distribution
- If a relational model is acyclic, then any ground graph is acyclic [Getoor 2001]

Probabilistic semantics of ground graphs



$$P(\mathcal{V}) = \prod_{v \in \mathcal{V}} P(v \mid parents(v))$$ Independent instance

$$P(GG_{\mathcal{M}\sigma}) = \prod_{v \in \mathcal{V}} \prod_{i \in \sigma(I)} P(v_i \mid parents(v_i))$$ Set of independent instances (ground graph of a Bayesian network)

$$P(GG_{\mathcal{M}\sigma}) = \prod_{I \in \mathcal{E} \cup \mathcal{R}} \prod_{X \in \mathcal{A}(I)} \prod_{i \in \sigma(I)} P(i.X \mid parents(i.X))$$ Ground graph of a relational model

Summary of relational concepts

Relational paths
compose
relational variables
compose
relational dependencies
compose
relational models
(all constrained by a **relational schema**),
which applied to a **relational skeleton**
produces a **ground graph**.

Concepts underlie the theory of relational *d*-separation and support the algorithmic details of the relational causal discovery algorithm.

Questions?

Topics

✓ Background on relational data and models

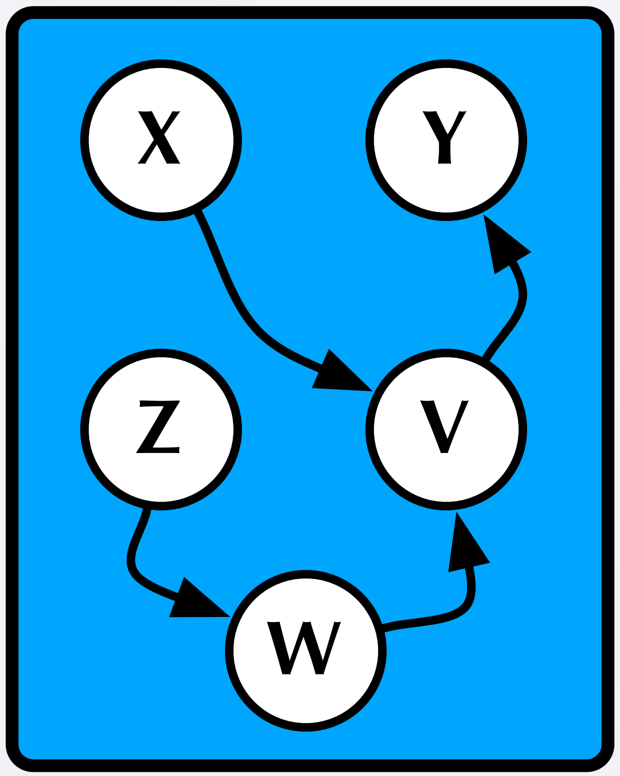
‣ **Relational *d*-separation**

‣ The RCD algorithm

Why is *d*-separation useful?

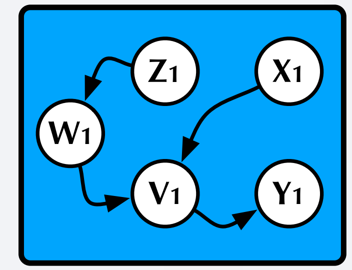
- *Grounded in theory*—Equivalent to global Markov condition
- *Algorithmic*—Simple set of graphical rules for derivation of conditional independence facts
- *Sound and complete*—Produces model implications that hold for all possible model instantiations
- *Enables constraint-based learning*—Algorithms can leverage the connection between causal structure and conditional independence

d-separation and ground graphs



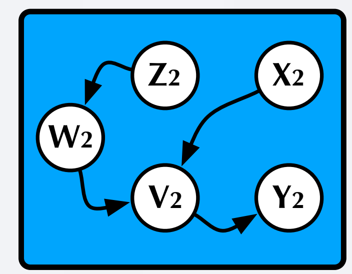
$X \perp\!\!\!\perp Y \mid \{ V \}$

$X \not\perp\!\!\!\perp W \mid \{ V \}$



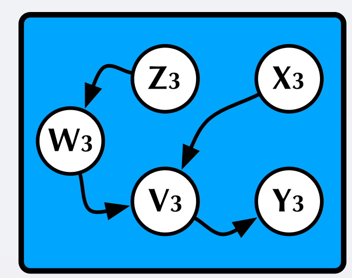
$X_1 \perp\!\!\!\perp Y_1 \mid \{V_1\}$

$X_1 \not\perp\!\!\!\perp W_1 \mid \{V_1\}$



$X_2 \perp\!\!\!\perp Y_2 \mid \{V_2\}$

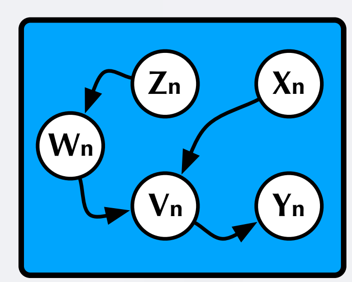
$X_2 \not\perp\!\!\!\perp W_2 \mid \{V_2\}$



$X_3 \perp\!\!\!\perp Y_3 \mid \{V_3\}$

$X_3 \not\perp\!\!\!\perp W_3 \mid \{V_3\}$

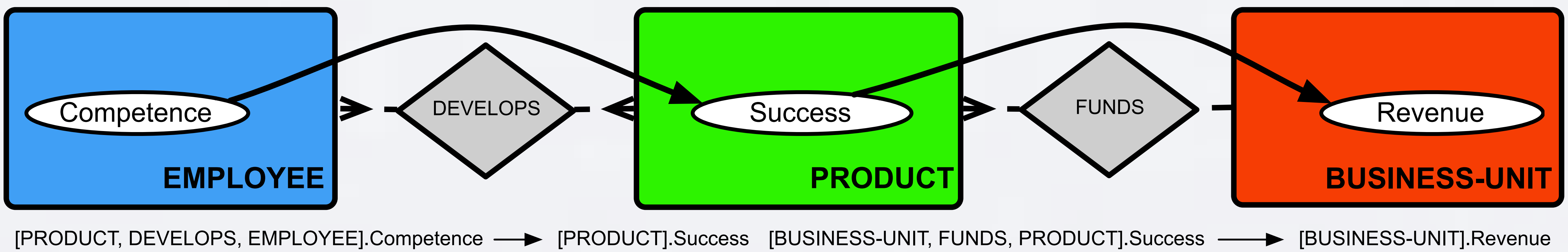
⋮



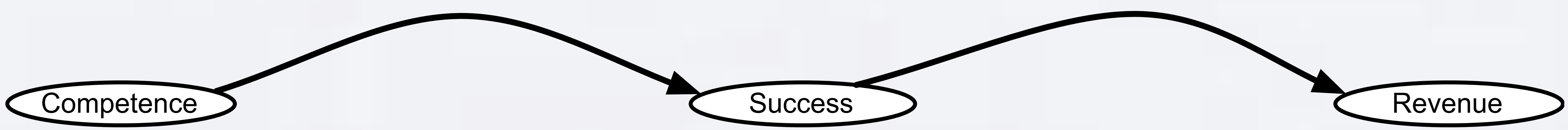
$X_n \perp\!\!\!\perp Y_n \mid \{V_n\}$

$X_n \not\perp\!\!\!\perp W_n \mid \{V_n\}$

d-separation applied to relational models

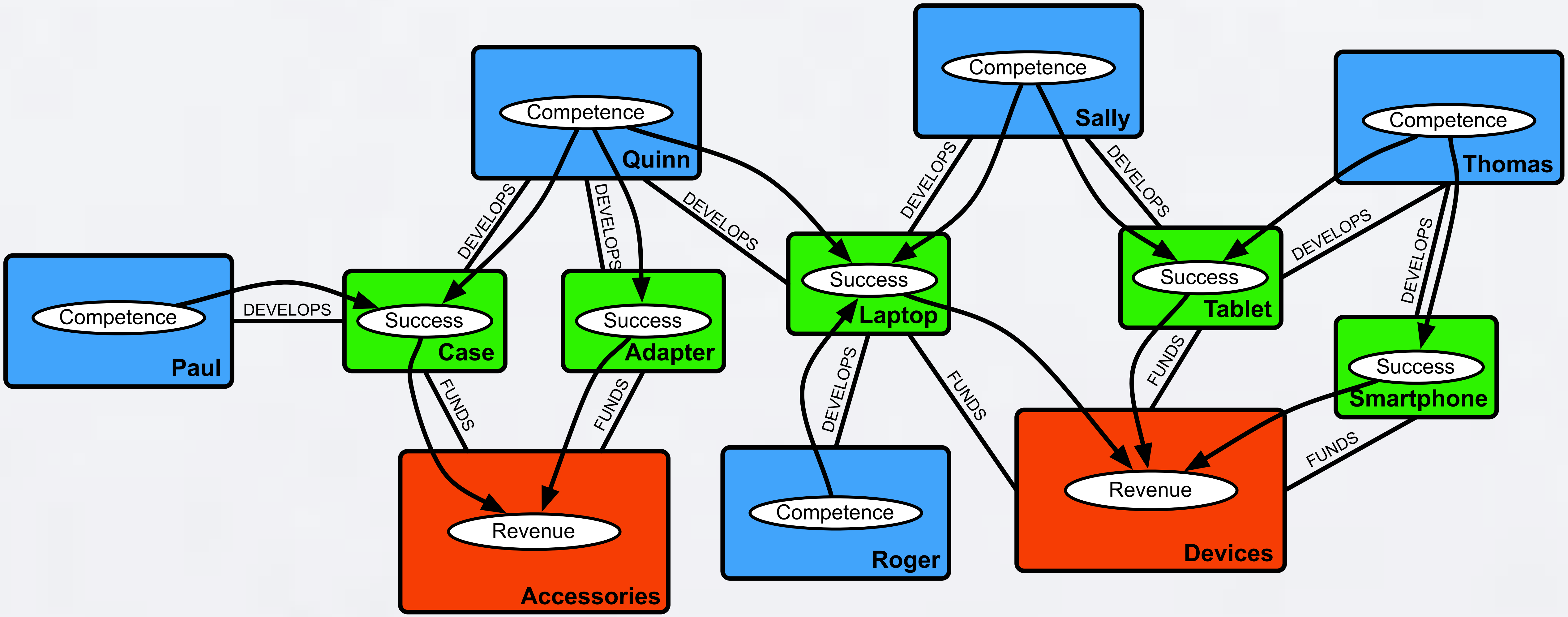


d-separation applied to relational models

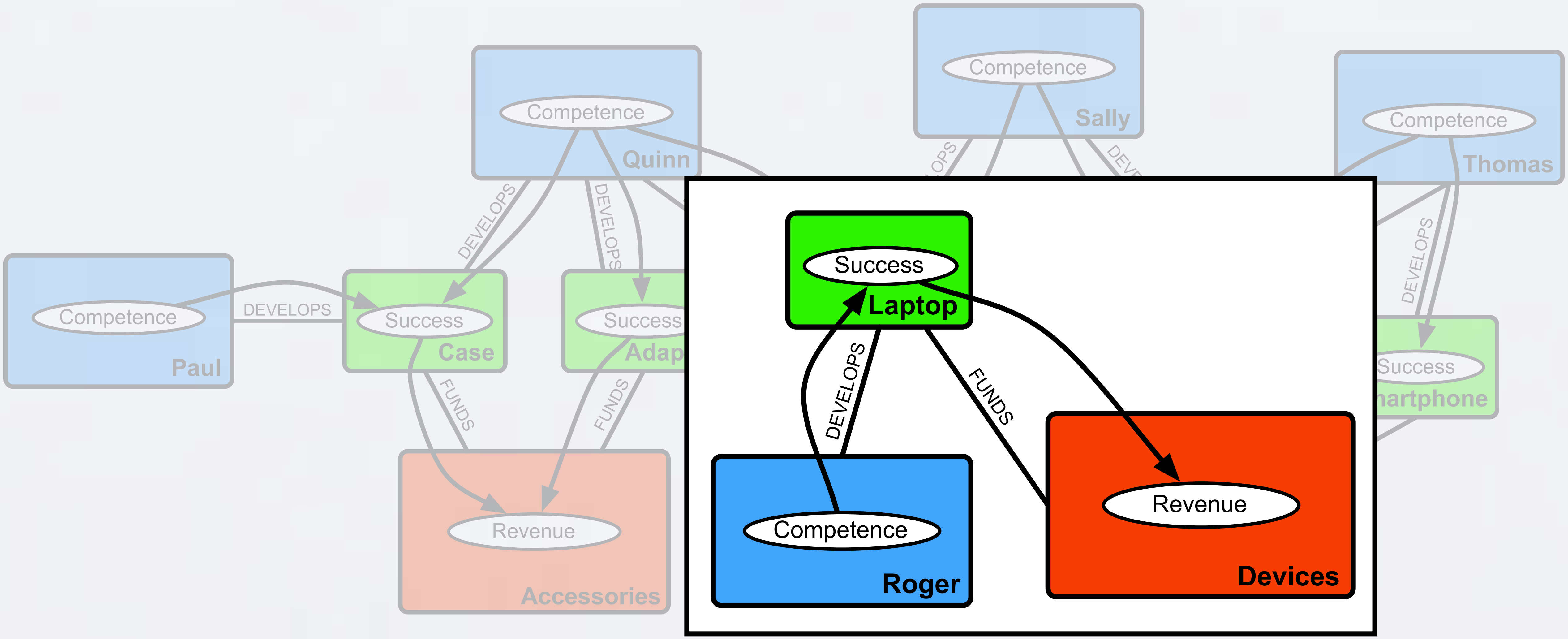


$$\text{Competence} \perp\!\!\!\perp \text{Revenue} \mid \{\text{Success}\}$$

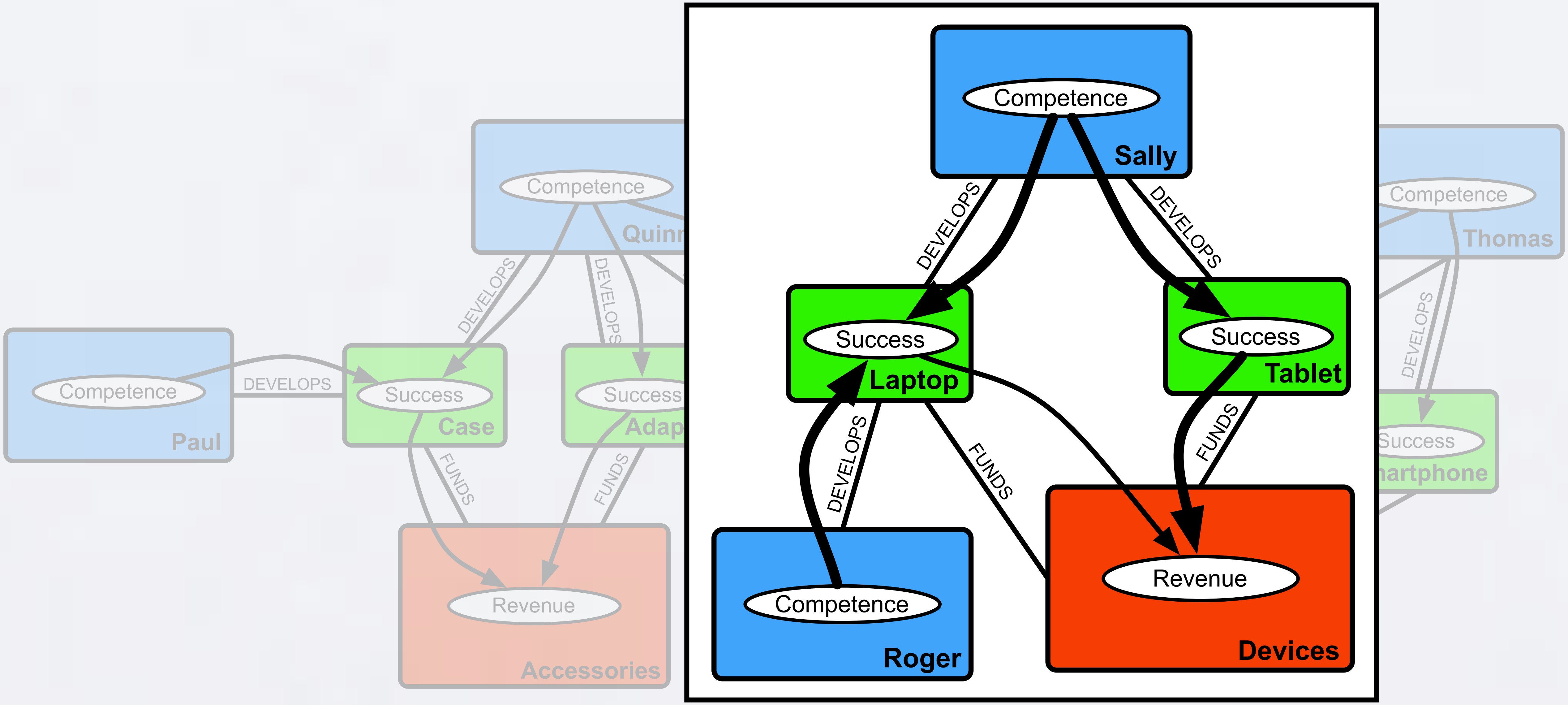
d-separation applied to relational models



d-separation applied to relational models



d-separation applied to relational models



[Employee].*Competence* $\perp\!\!\!\perp$ [Employee, Product, Business-Unit].*Revenue* |

{ [Employee, Product].*Success*, [Employee, Product, Employee].*Competence* }

Towards a theory of *d*-separation for relational models

- Why not test for *d*-separation at the *model level*?
  - **Relational *d*-connecting paths** that are only manifest in ground graphs.
- Why not test for *d*-separation on *ground graphs*?
  - Impractical to have tests on a representation that **scales with sample size** (ground graphs can be arbitrarily large).
  - A ground graph is a **single data sample** from all represented skeletons and distributions of a relational model.

Defining relational d-separation

Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ be distinct sets of relational variables for perspective $B \in \mathcal{E} \cup \mathcal{R}$ for relational schema $\mathcal{S}$.

For relational model $\mathcal{M}$, $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ if and only if, for any skeleton σ, $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ are d-separated by $\mathbf{Z}|_b$ in ground graph $GG_{\mathcal{M}\sigma}$ for all $b \in \sigma(B)$.

all possible ground graphs

all instances

...which suggests we need a representation that **abstracts** over all possible ground graphs.

Defining abstract ground graphs

An **abstract ground graph** $AGG_{\mathcal{M}Bh} = (V, E)$
for relational model $\mathcal{M} = (\mathcal{S}, \mathcal{D})$, perspective $B \in \mathcal{E} \cup \mathcal{R}$,
and hop threshold $h \in \mathbb{N}^0$ abstracts dependencies $\mathcal{D}$
for all possible ground graphs $GG_{\mathcal{M}\sigma}$ of $\mathcal{M}$ for all skeletons σ.

Abstract ground graphs capture all possible paths of dependence with two primary innovations:

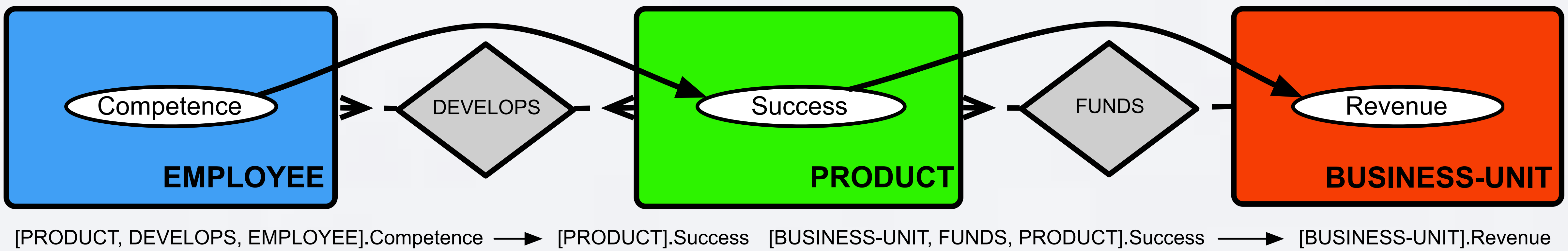
(1) Dependencies are translated across all perspectives

(2) Intersection variables are explicitly represented for pairs of relational variables that may intersect in some skeleton

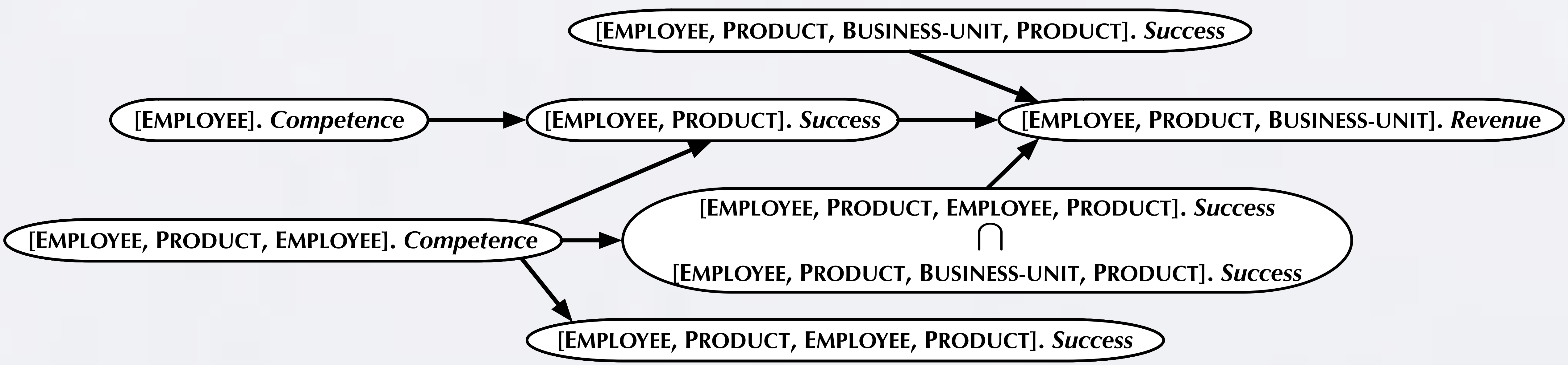
Abstract ground graphs abstract ground graphs

- *Lifted representation*: Lies between the model level and the ground graph level.
- *Data-free*: Constructed with knowledge of only the model structure (*M)*, a single perspective (*B*), and a hop threshold (*h*).
- *Sound and complete*: (1) Every dependency in the abstract ground graph exists in some ground graph and (2) any dependency in any ground graph exists in the abstract ground graph.
- *Generalizes Bayesian networks*: For schemas with a single entity class, the abstract ground graph is equivalent to the model.

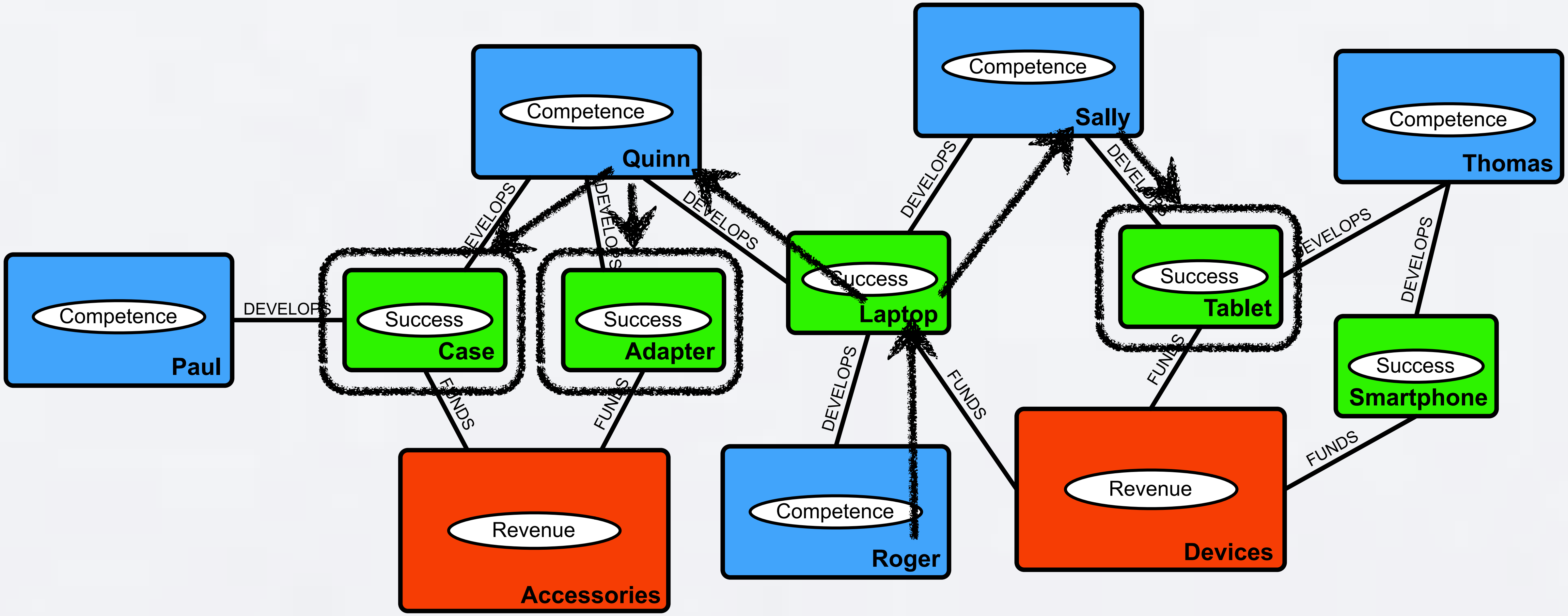
Constructing abstract ground graphs



EMPLOYEE perspective ↓ hop threshold h = 6

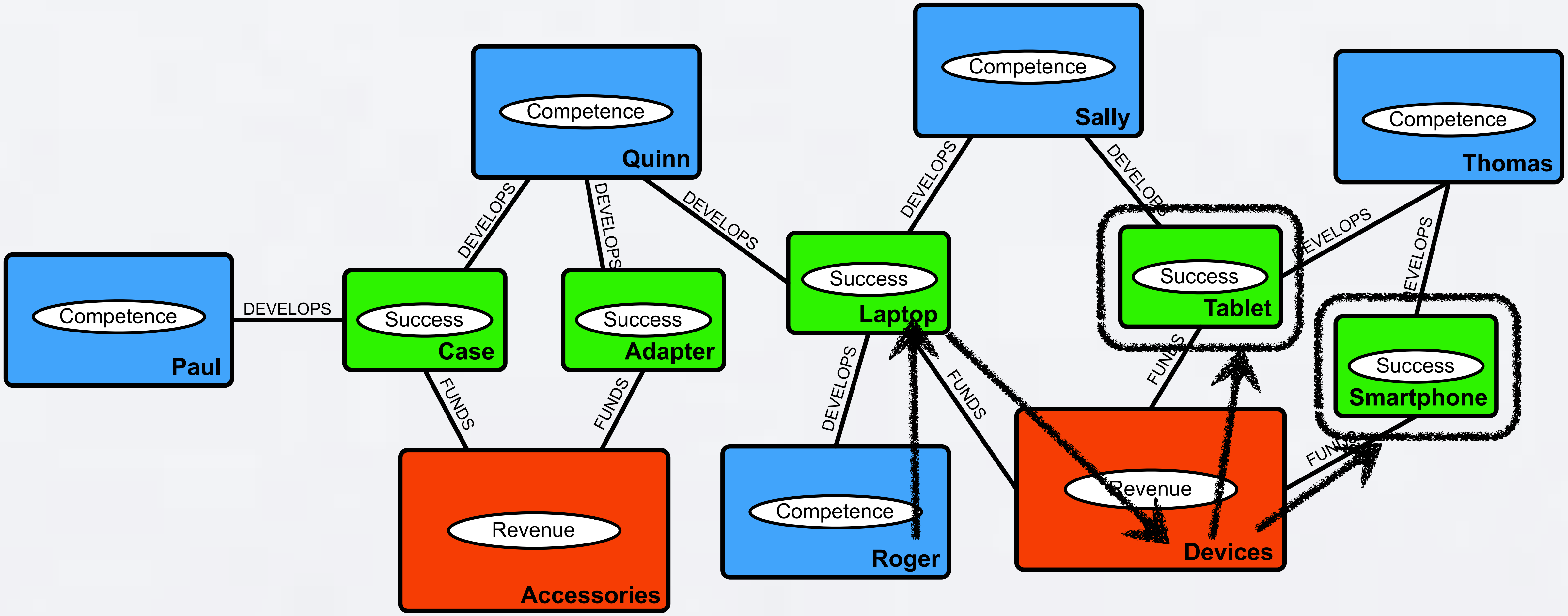


Intersecting terminal sets of relational paths



[Employee, Product, Employee, Product]|$_{\text{Roger}}$ = {Case, Adapter, Tablet}

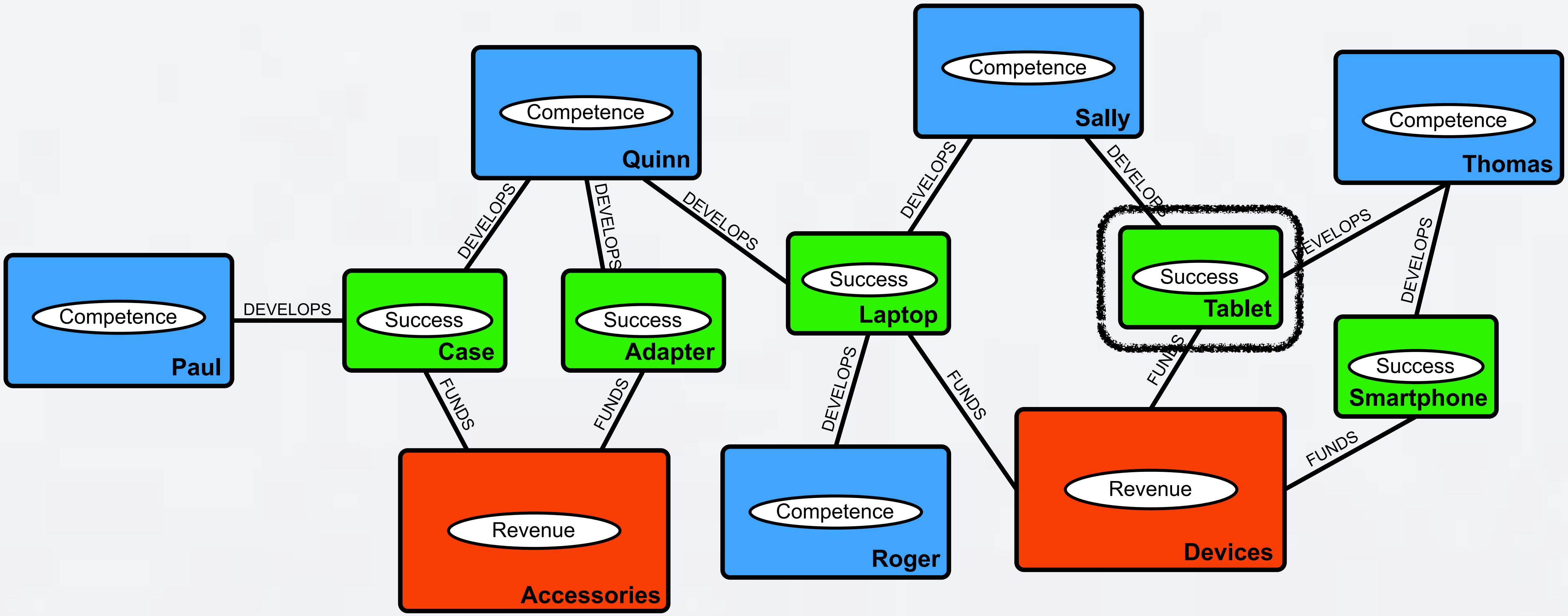
Intersecting terminal sets of relational paths



[Employee, Product, Employee, Product]|$_{\text{Roger}}$ = {Case, Adapter, Tablet}

[Employee, Product, Business-Unit, Product]|$_{\text{Roger}}$ = {Tablet, Smartphone}

Intersecting terminal sets of relational paths



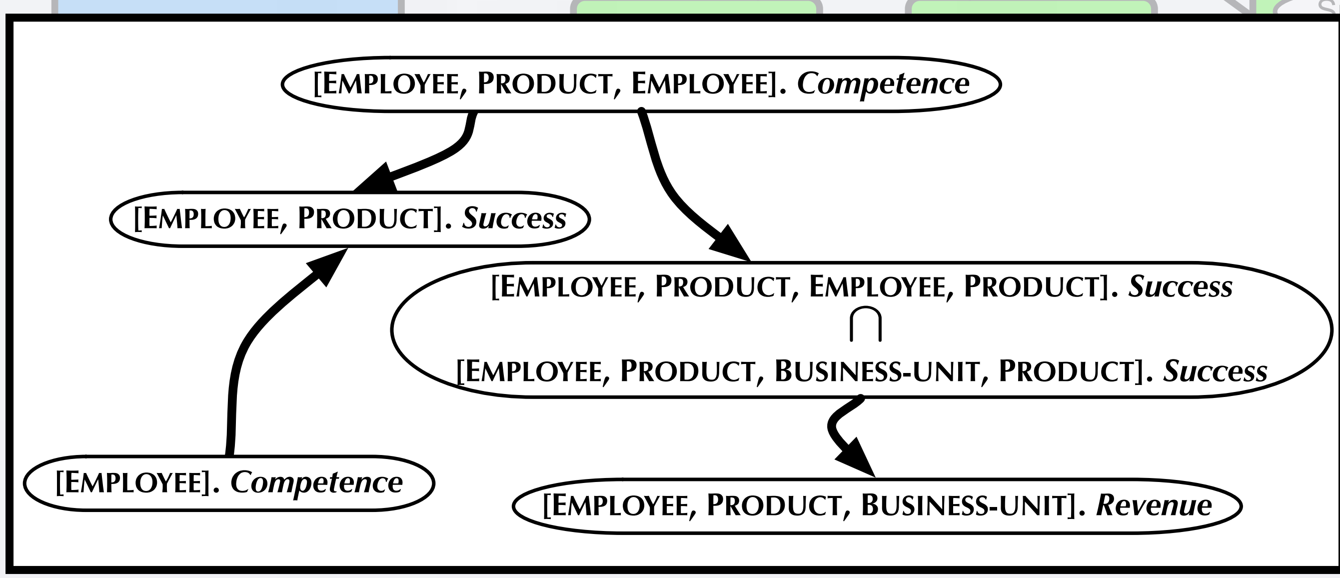
$$[\text{Employee, Product, Employee, Product}]|_{\text{Roger}} = \{\text{Case, Adapter, Tablet}\}$$

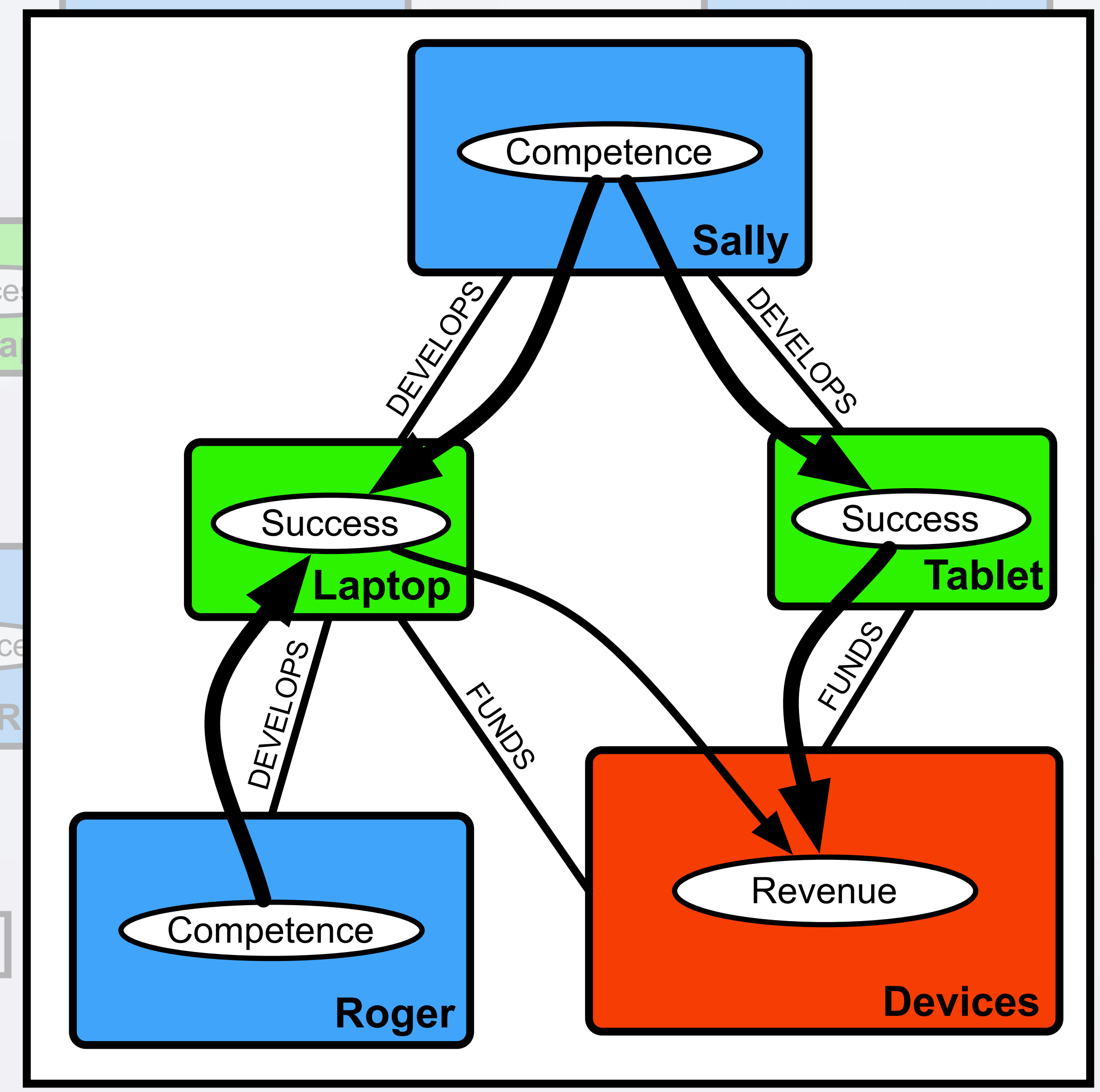
$$\cap$$

$$[\text{Employee, Product, Business-Unit, Product}]|_{\text{Roger}} = \{\text{Tablet, Smartphone}\}$$

$$= \{\text{Tablet}\}$$

Intersecting terminal sets of relational paths





[Employee, Product, Employee, Product]

∩

[Employee, Product, Business-Unit, Product]|$_{\text{Roger}}$ = {Tablet, Smartphone}

= {Tablet}

d-separation on abstract ground graphs

Given a query:

$$\text{Is } \mathbf{X}\ d\text{-separated from } \mathbf{Y} \text{ given } \mathbf{Z}?$$

Answer by checking:

$$\text{Is } \bar{\mathbf{X}}\ d\text{-separated from } \bar{\mathbf{Y}} \text{ given } \bar{\mathbf{Z}}?$$

on the abstract ground graph for the common **perspective**, where the **augmented** sets include subsumed intersection variables

- Because abstract ground graphs capture all paths of dependence, it suffices to check if all pairwise elements in $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are *d*-separated by $\bar{\mathbf{Z}}$.
- Reflects all dependency paths for any possible variable instance pair in any ground graph represented by the abstract ground graph

Relational *d*-separation is sound and complete

Proof sketch

(1) *d*-separation for DAGs is
sound [Verma & Pearl 1988] and complete [Geiger & Pearl 1988]

(2) Abstract ground graphs are directed and acyclic

(3) Abstract ground graphs are sound and complete

(4) Abstract ground graph completeness ⇒
Relational *d*-separation soundness

(5) Abstract ground graph soundness ⇒
Relational *d*-separation completeness

Valid up to a specified hop threshold *h*

Naïvely applying *d*-separation is frequently incorrect

Synthetic generation:

Schemas: $|\text{Entity classes}| \in [1, 4]$

Models: $|\text{Dependencies}| \in [1, 10]$

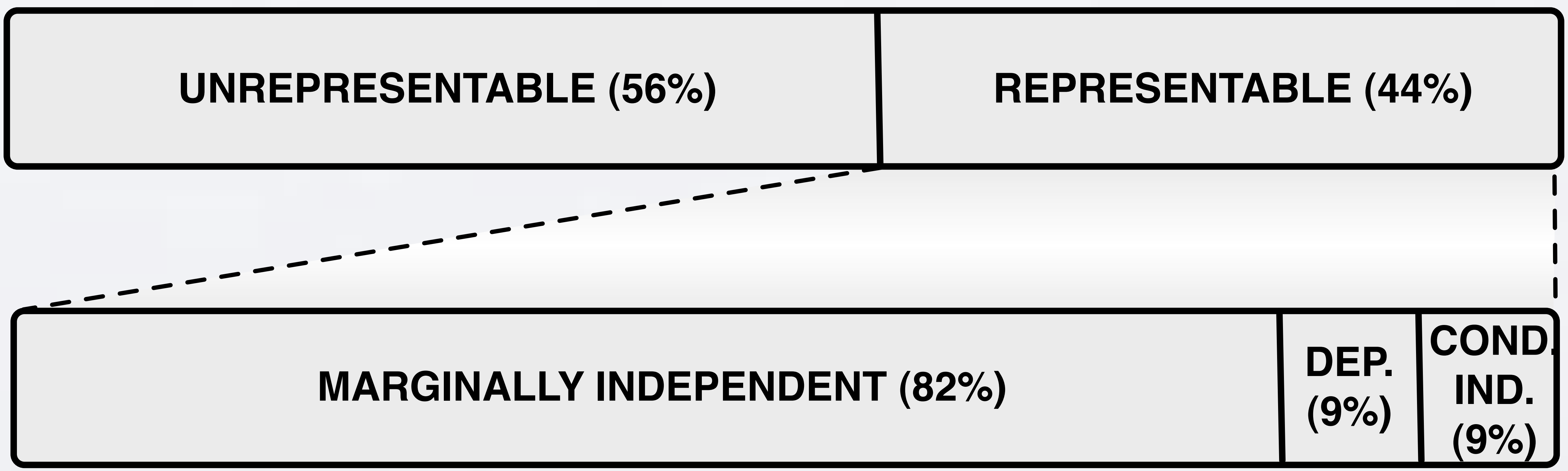
3.6 million pairs of relational variables

UNREPRESENTABLE (56%)	**REPRESENTABLE (44%)**

Unrepresentable: Either the treatment or outcome relational path includes an item class more than once.

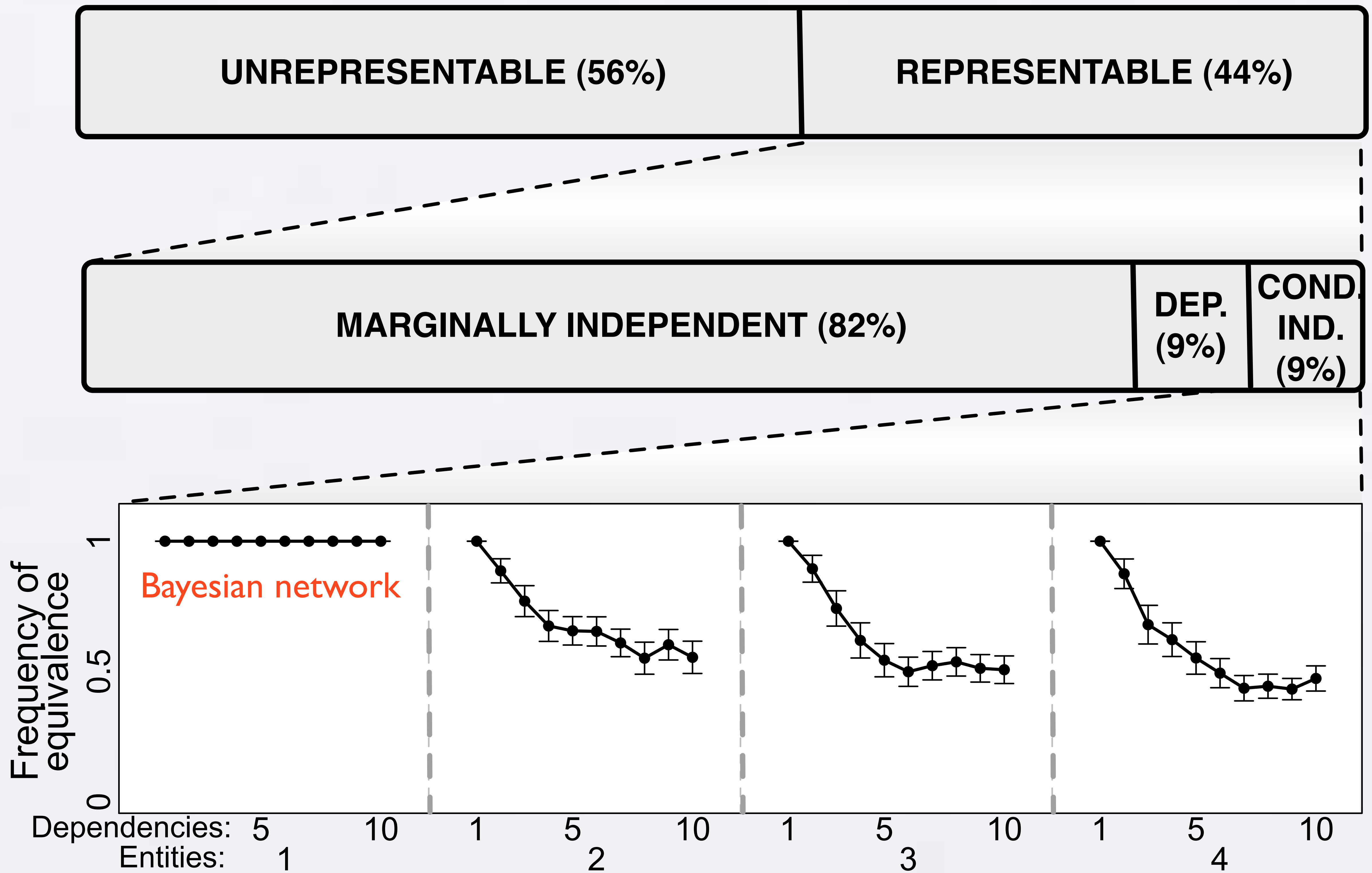
E.g, [Employee, Develops, Product, Develops, Employee]

Naïvely applying *d*-separation is frequently incorrect



Most representable queries are marginally independent because the total dependencies varies from 1 to 15.

Naïvely applying *d*-separation is frequently incorrect



Future work

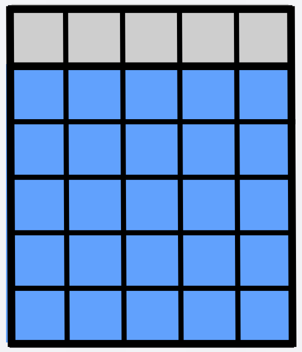
- Include deterministic/functional dependencies (D-separation)
- Reason about models of entity and relationship existence
- Develop the implications of relational *d*-separation and abstract ground graphs (next—the RCD algorithm!)

Questions?

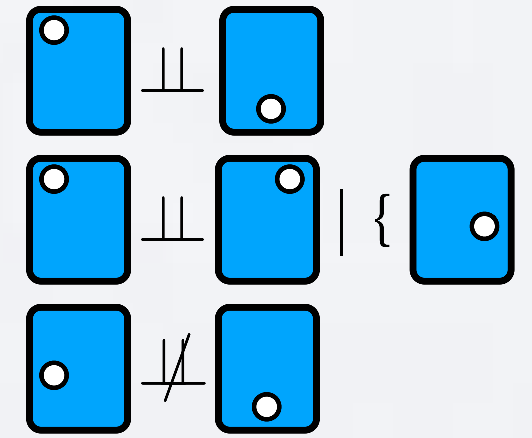
Topics

- ✓ Background on relational data and models
- ✓ Relational *d*-separation
- ‣ **The RCD algorithm**

The PC algorithm is sound and complete

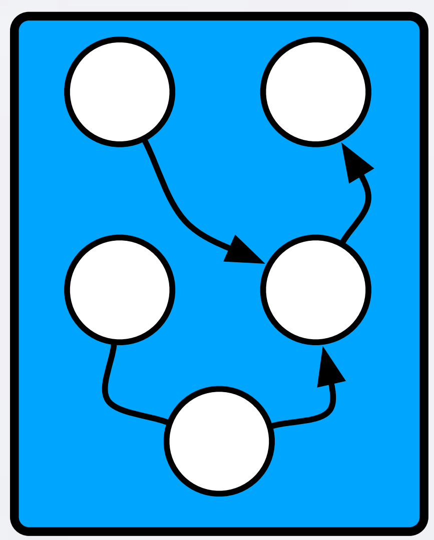


propositional data



conditional independencies

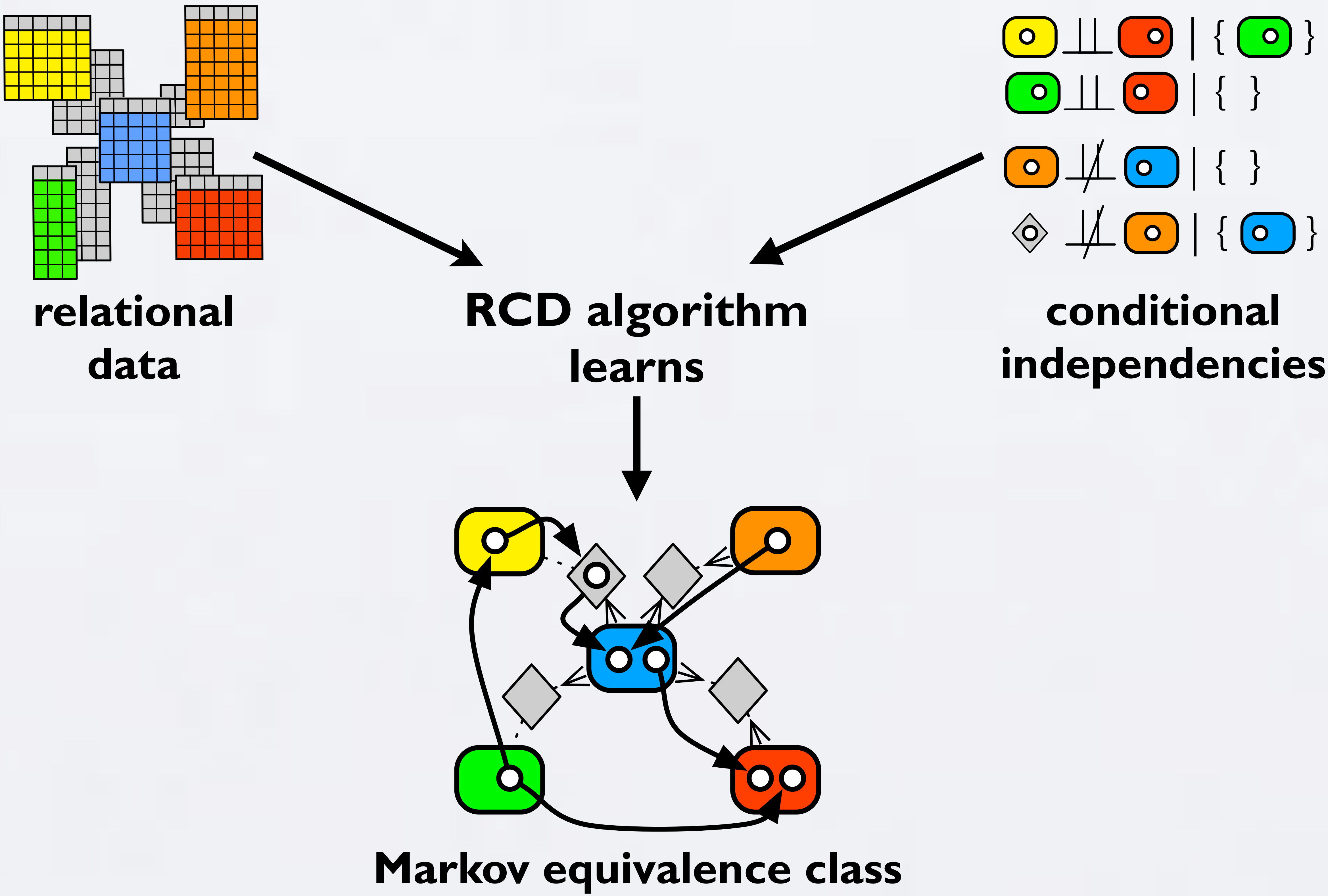
PC algorithm learns



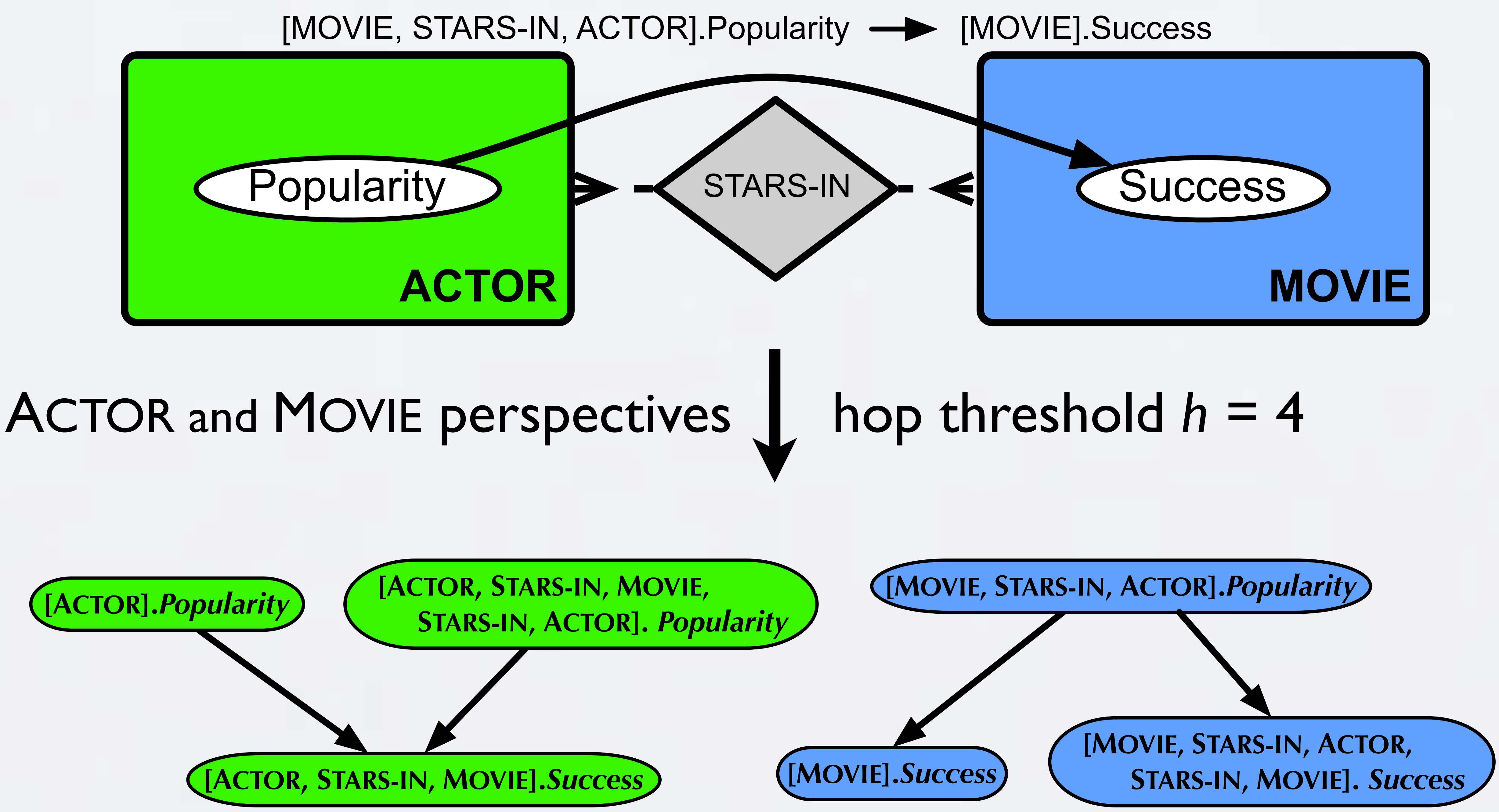
Markov equivalence class

[Meek 1995]

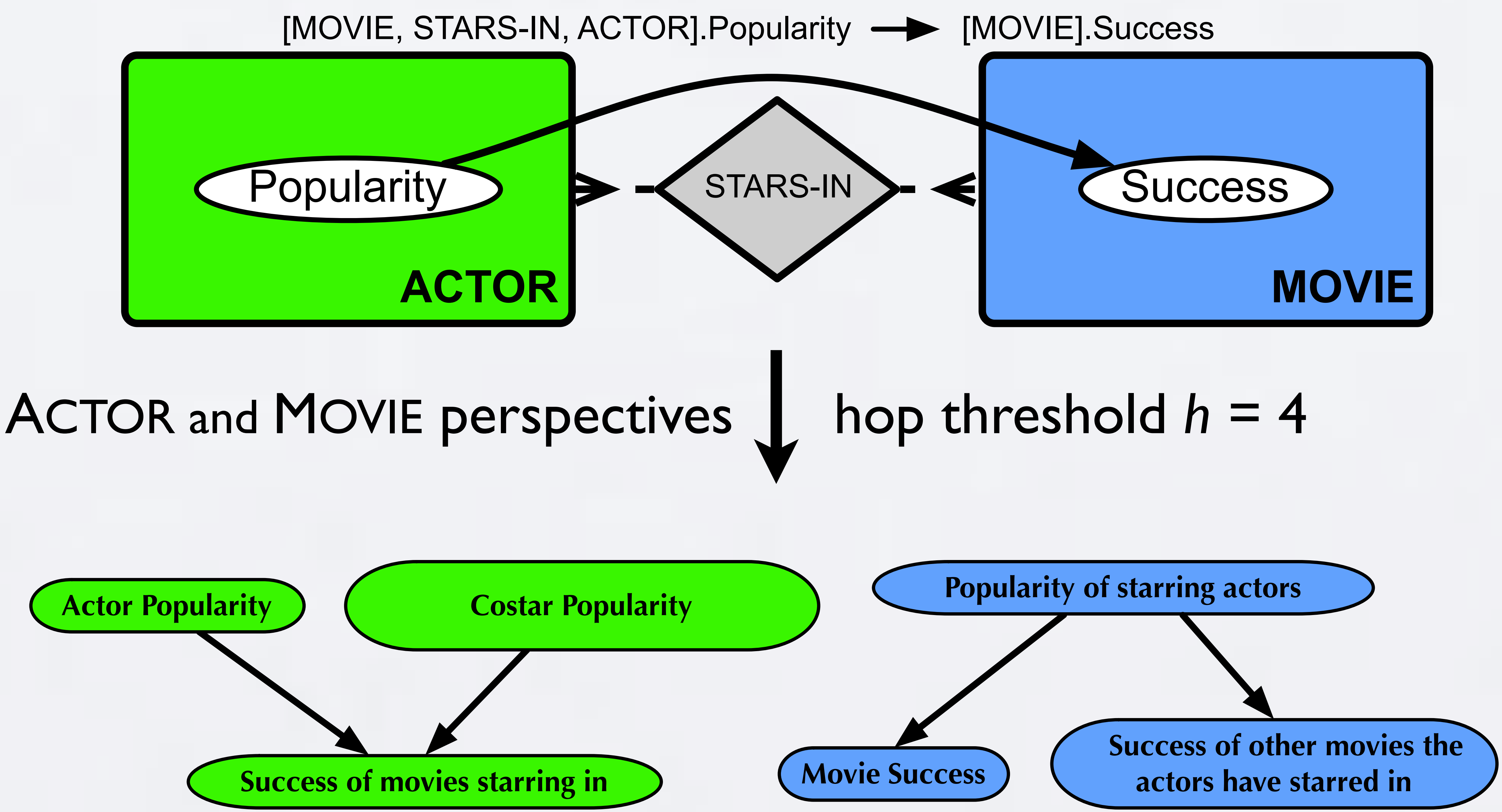
Relational analog



Abstract ground graphs enable new constraints

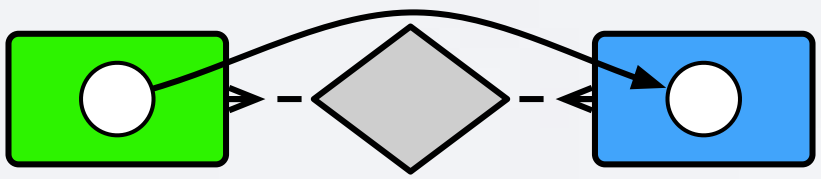


Abstract ground graphs enable new constraints



Relational bivariate orientation (RBO)

- RBO leverages relational dependencies that cross relationships with a MANY **cardinality**.



- **Assumes only model acyclicity** (no assumptions about functional form or conditional densities).
  - Other bivariate dependency orientation methods can be used where RBO cannot [Shimizu et al. 2006; Hoyer et al. 2009; Zhang & Hyvärinen 2009; Peters et al. 2010].

- RBO can be described as detecting **relational autocorrelation** [Jensen & Neville 2002] and testing if a distinct variable is a member of the separating set that eliminates the autocorrelation.

Relational bivariate orientation (RBO)

Abstract ground graph from I_X perspective

$$\begin{tikzcd}
{[I_X].X} \arrow[dr, dash] & & {[I_X \ldots I_Y \ldots I_X].X} \arrow[dl, dash] \\
& {[I_X \ldots I_Y].Y} &
\end{tikzcd}$$

Does $[I_X \ldots I_Y].Y$ help remove autocorrelation?

$$[I_X \ldots I_Y].Y \in sepset([I_X].X,\ [I_X \ldots I_Y \ldots I_X].X)?$$

YES

Orient as common cause

$$\begin{tikzcd}
{[I_X].X} & & {[I_X \ldots I_Y \ldots I_X].X} \\
& {[I_X \ldots I_Y].Y} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

NO

Orient as common effect

$$\begin{tikzcd}
{[I_X].X} \arrow[dr] & & {[I_X \ldots I_Y \ldots I_X].X} \arrow[dl] \\
& {[I_X \ldots I_Y].Y} &
\end{tikzcd}$$

Extending PC orientation rules relationally

Collider Detection (CD)

$$
\begin{tikzcd}[column sep=small]
{[B\ldots I_X].X} \arrow[dr, dash] & & {[B\ldots I_Z].Z} \arrow[dl, dash] \\
& {[B\ldots I_Y].Y} &
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}[column sep=small]
{[B\ldots I_X].X} \arrow[dr] & & {[B\ldots I_Z].Z} \arrow[dl] \\
& {[B\ldots I_Y].Y} &
\end{tikzcd}
$$

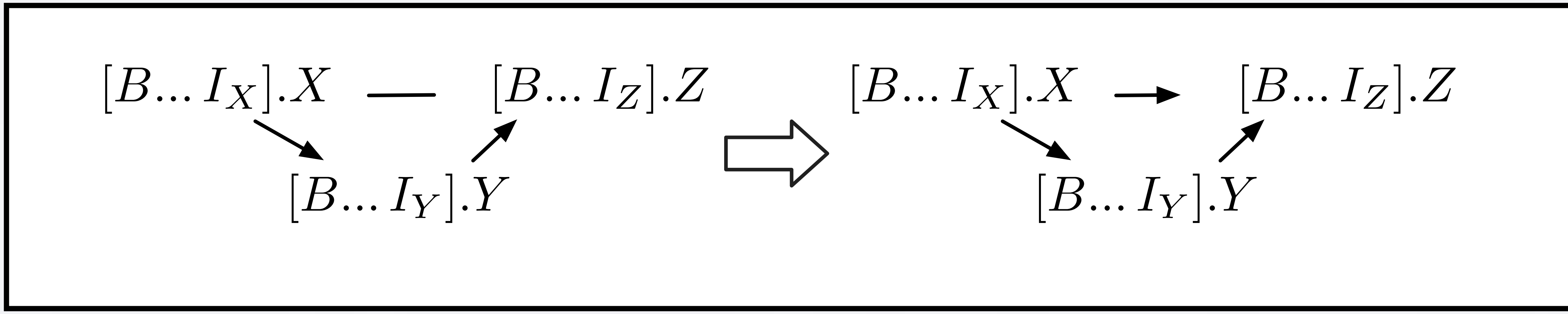
$$[B\ldots I_Y].Y \notin sepset([B\ldots I_X].X,\ [B\ldots I_Z].Z)$$

Known Non-Colliders (KNC)

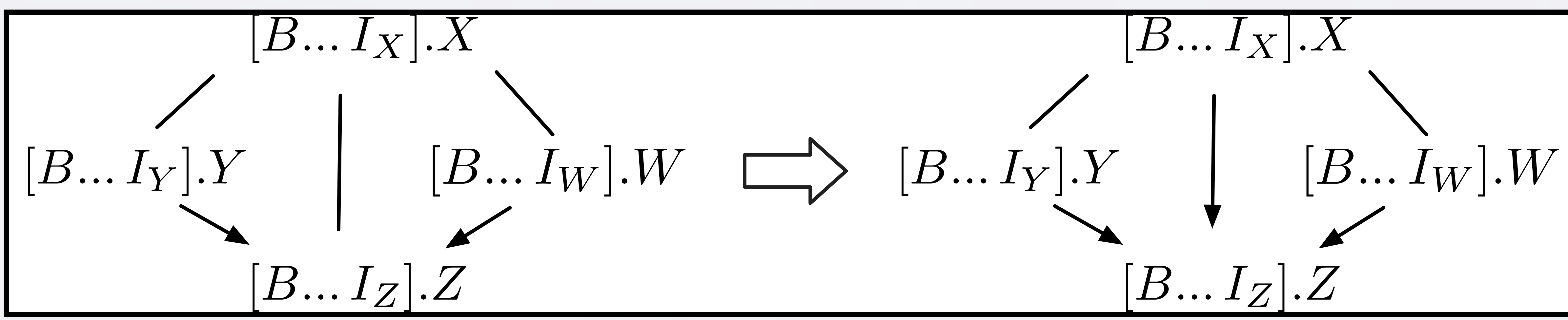
$$
\begin{tikzcd}[column sep=small]
{[B\ldots I_X].X} \arrow[dr] & & {[B\ldots I_Z].Z} \arrow[dl, dash] \\
& {[B\ldots I_Y].Y} &
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}[column sep=small]
{[B\ldots I_X].X} \arrow[dr] & & {[B\ldots I_Z].Z} \\
& {[B\ldots I_Y].Y} \arrow[ur] &
\end{tikzcd}
$$

Extending PC orientation rules relationally

Cycle Avoidance (CA)

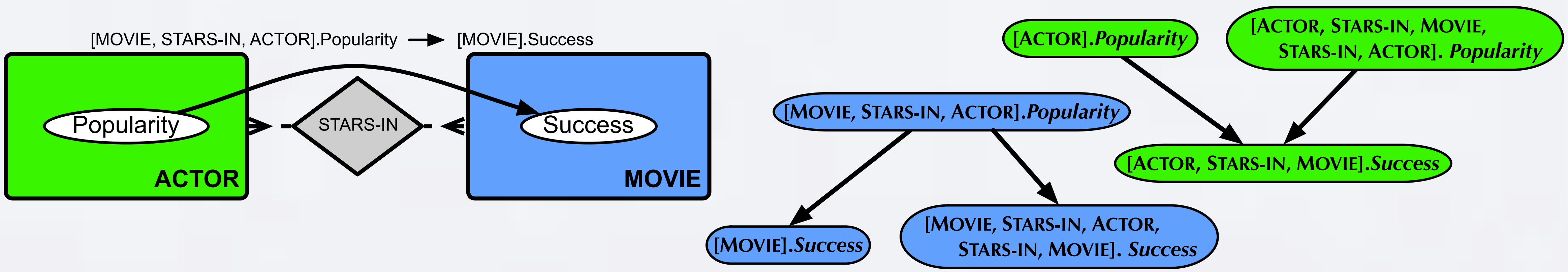


Meek Rule 3 (MR3)



Orientation propagation

- A single relational dependency supports many edges within and across the set of abstract ground graphs for a relational model.



- When a rule is activated for a *specific* abstract ground graph, the orientation of the underlying relational dependency must be *propagated* within and across *all* abstract ground graphs.

Orientation rule soundness and completeness

Soundness definition: An orientation rule is *sound* if any orientation not indicated by the rule introduces either

(1) An unshielded collider in some abstract ground graph
(2) A directed cycle in some abstract ground graph
(3) A model-level cycle

[Adapted from Meek 1995]

Completeness definition: A set of orientation rules is *complete* if any orientation of an unoriented edge is consistent with a member of the Markov equivalence class.

[Meek 1995]

Proof: Shown for individually for soundness and collectively for completeness (CD, KNC, CA, MR3, RBO, and propagation)

The relational causal discovery algorithm

Initialize set of potential dependencies

Phase I: Identify skeleton via separating sets

Phase II: Build abstract ground graphs and orient dependencies

ALGORITHM 1: RCD($schema$, $depth$, $hopThreshold$, P)

$PDs \leftarrow$ `getPotentialDeps`($schema$, $hopThreshold$)
$N \leftarrow$ `initializeNeighbors`($schema$, $hopThreshold$)
$S \leftarrow \{\}$
`// Phase I`
for $d \leftarrow 0$ **to** $depth$ **do**
    for $X \rightarrow Y \in PDs$ **do**
        foreach $condSet \in$ `powerset`($N[Y] \setminus \{X\}$) **do**
            if $|condSet| = d$ **then**
                if $X \perp\!\!\!\perp Y \mid condSet$ in P **then**
                    $PDs \leftarrow PDs \setminus \{X \rightarrow Y, Y \rightarrow X\}$
                    $S[X, Y] \leftarrow condSet$
                    break
`// Phase II`
$AGGs \leftarrow$ `buildAbstractGroundGraph`(PDs)
$AGGs, S \leftarrow$ `ColliderDetection`($AGGs, S$)
$AGGs, S \leftarrow$ `BivariateOrientation`($AGGs, S$)
while $changed$ **do**
    $AGGs \leftarrow$ `KnownNonColliders`($AGGs, S$)
    $AGGs \leftarrow$ `CycleAvoidance`($AGGs, S$)
    $AGGs \leftarrow$ `MeekRule3`($AGGs, S$)
return `getCanonicalDependencies`($AGGs$)

RCD correctness

RCD correctly learns a **maximally oriented model**

Assumptions

(1) Sufficient hop threshold *h*
(2) Sufficient *depth*
(3) Causal sufficiency
(4) Faithfulness
(5) Perfect conditional independence tests

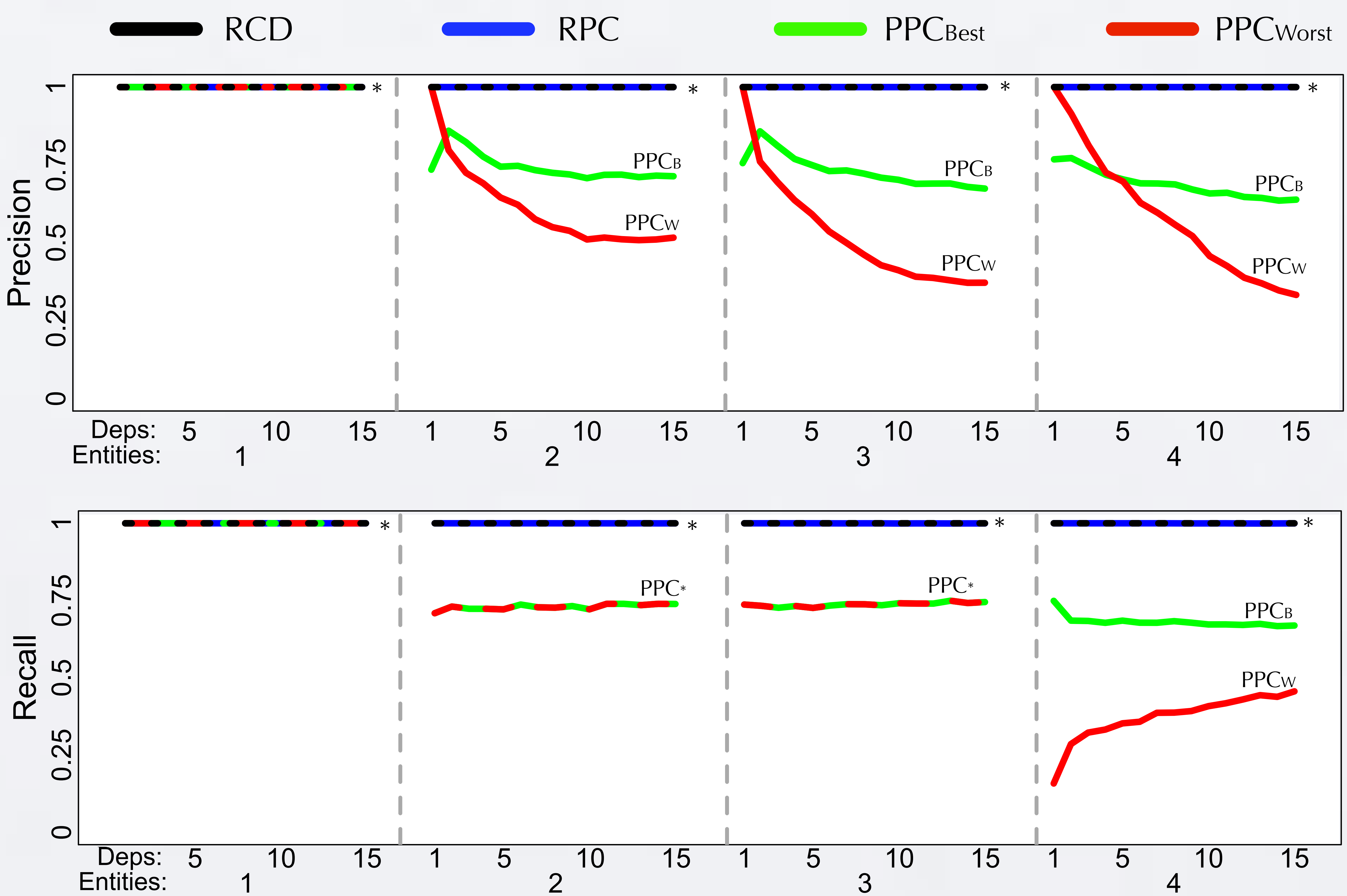
Proof

Follows similarly to PC Phase I correctness and edge orientation rule completeness

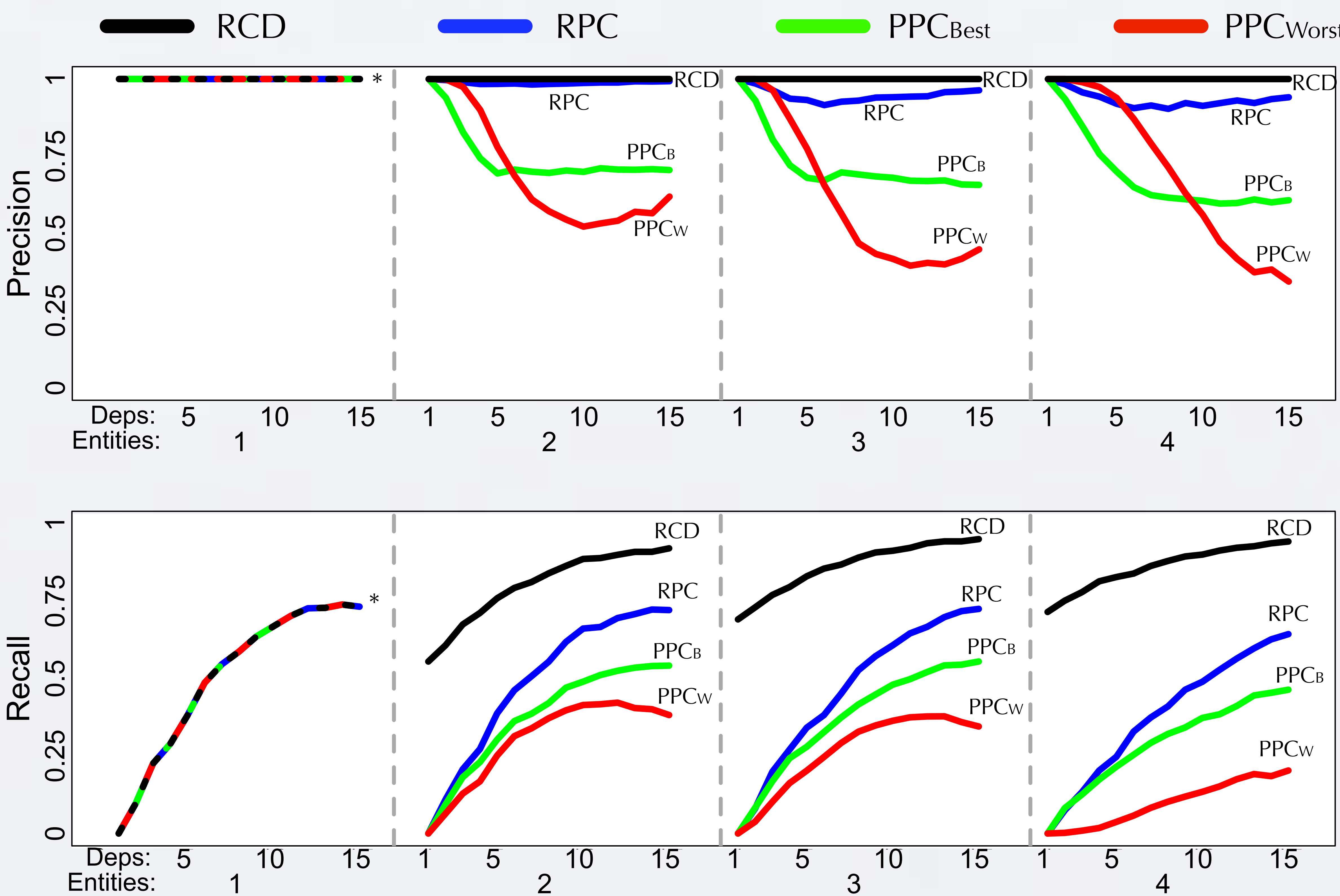
Empirical evaluation

- **Synthetic model structure generation**
  - $|\text{Entity classes}| \in [1, 4]$
  - $|\text{Relationship classes}| = |\text{Entity classes}| - 1$
  - $|\text{Attributes}| \sim \text{Pois}(\lambda=1) + 1$
  - $|\text{Dependencies}| \in [1, 15]$
- **Algorithms**
  - RCD
  - Relational PC (RPC) [Maier et al. 2010]
  - Propositionalized PC (PPC)—best and worst perspectives
  - (All using a relational *d*-separation oracle)
- **Evaluation measures**
  - Precision: $\dfrac{|\text{Correctly Learned}|}{|\text{Learned}|}$ Recall: $\dfrac{|\text{Correctly Learned}|}{|\text{True}|}$
  - For skeleton and oriented model

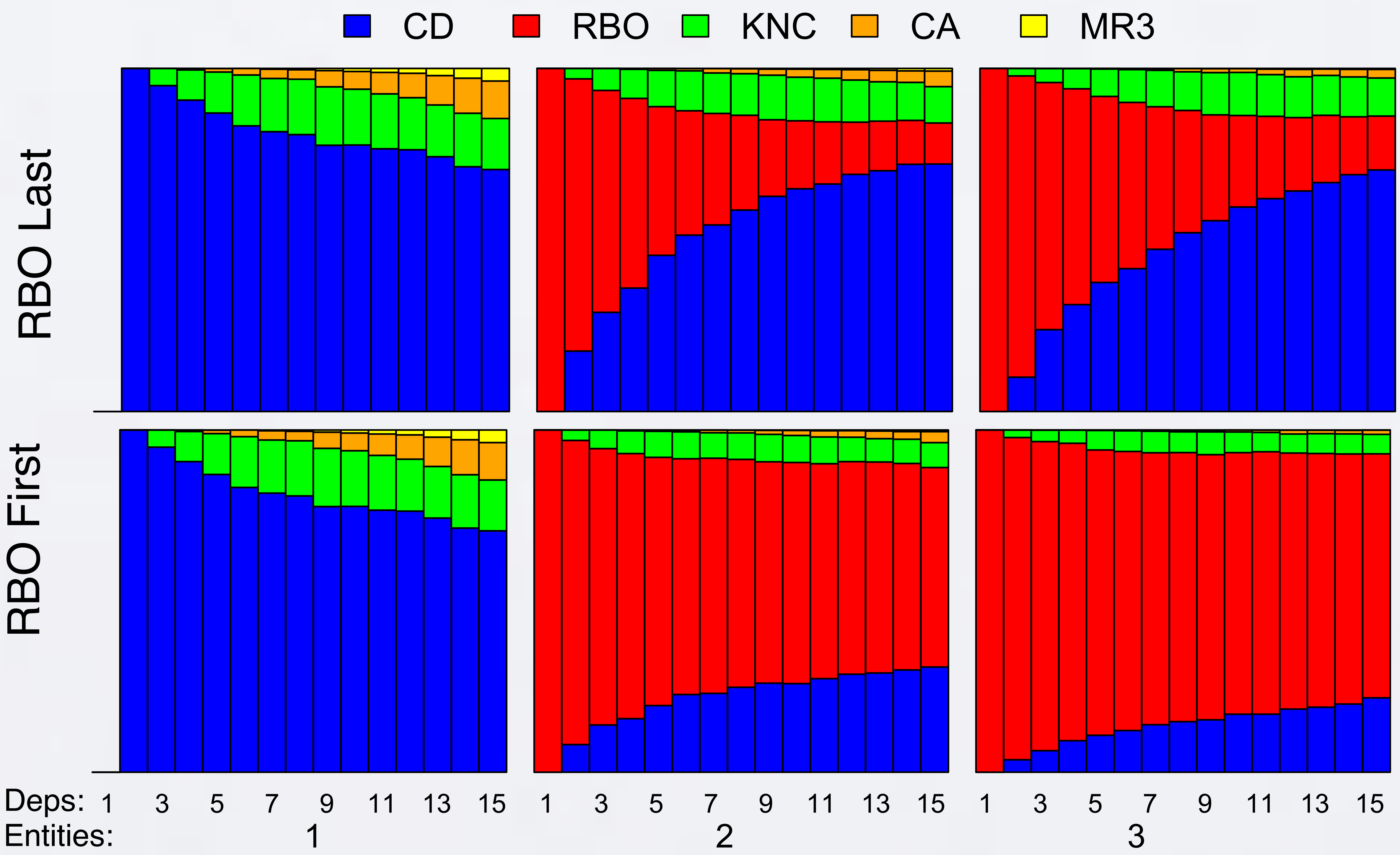
Identifying (causal) skeletons



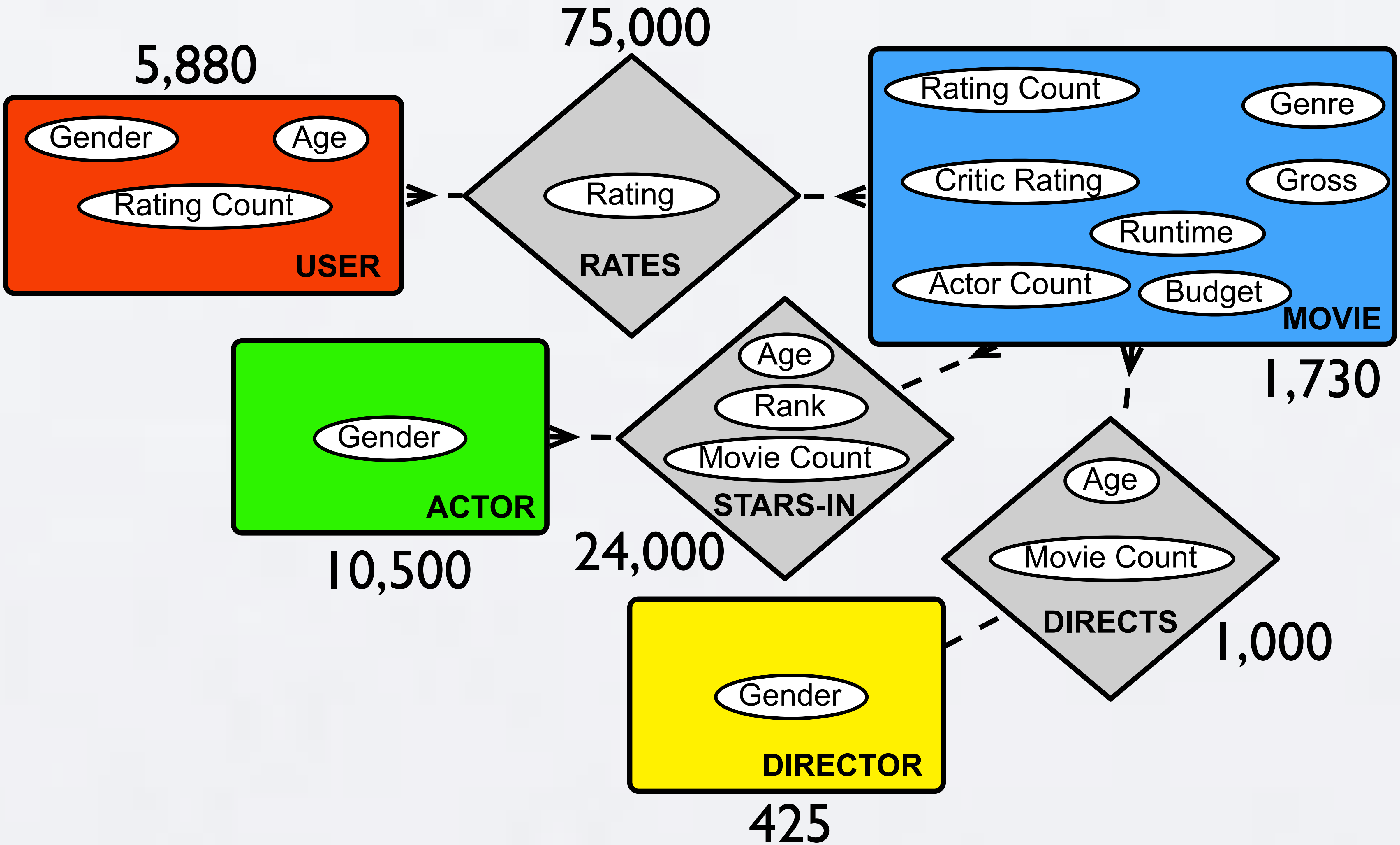
Orienting dependencies



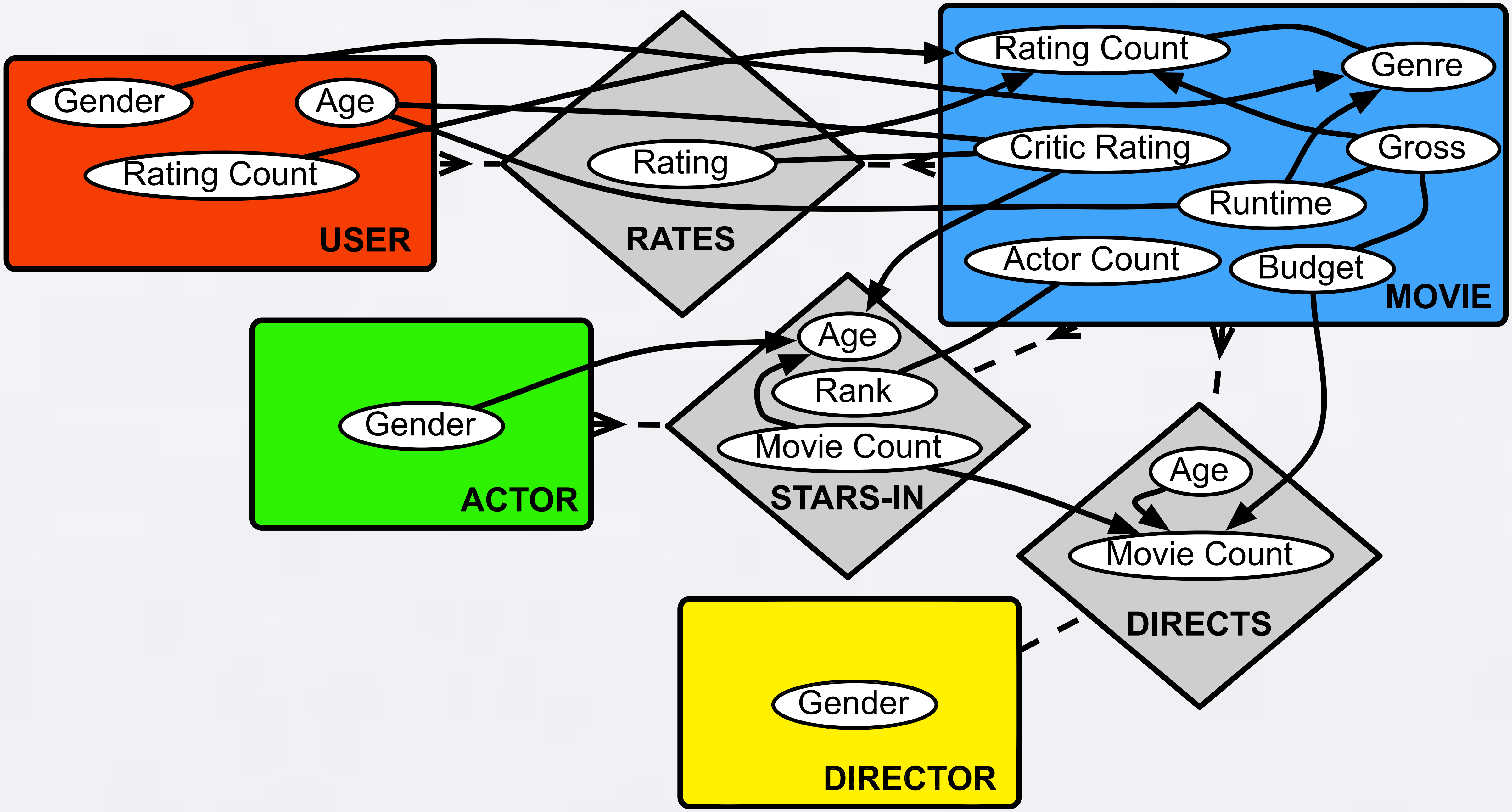
The unique contribution of RBO



Learning a causal model of the movie industry



Learning a causal model of the movie industry



Future work

- Develop more accurate tests of conditional independence for relational data
- Learn causal models of relationship existence
- Relax causal sufficiency by incorporating the relational blocking operator [Rattigan, Maier & Jensen 2011]
- Learn causal relational models with temporal dynamics

Summary

- **Bayesian networks**, ***d*-separation**, and the **PC algorithm** have provided a solid foundation for research on causal structure learning

- We now have an analogous basis for causal structure learning from relational data
  - New representation (**abstract ground graphs**), capabilities for reasoning about independence (**relational *d*-separation**), and a sound and complete algorithm (**RCD**)

Thank you!

Questions?

maier@cs.umass.edu

http://kdl.cs.umass.edu/rcd