Property Graph Model

Prof. Gianmaria Silvello

Department of Information Engineering
University of Padua

silvello@dei.unipd.it

http://www.dei.unipd.it/~silvello/

Fundamentals of Information Systems
Master Degree in Data Science

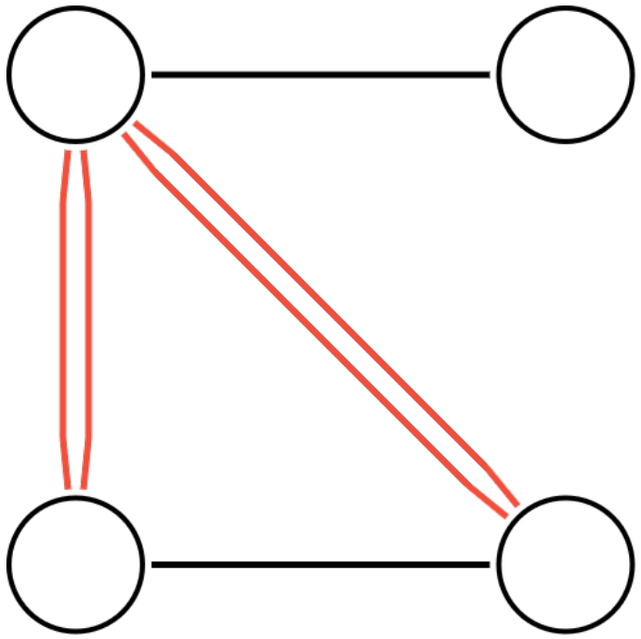
Data Model

Property Graph Model

- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph

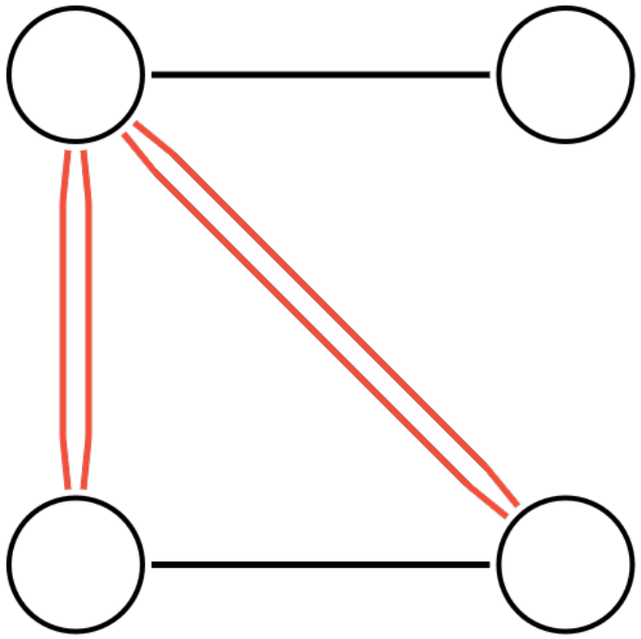
Property Graph Model

- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph



Property Graph Model

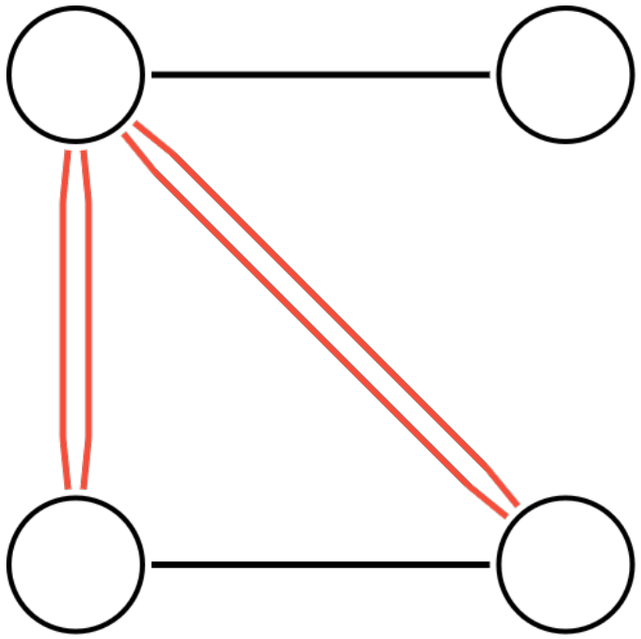
- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph



A multigraph is a graph in which multiple edges between the same two nodes are permitted

Property Graph Model

- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph

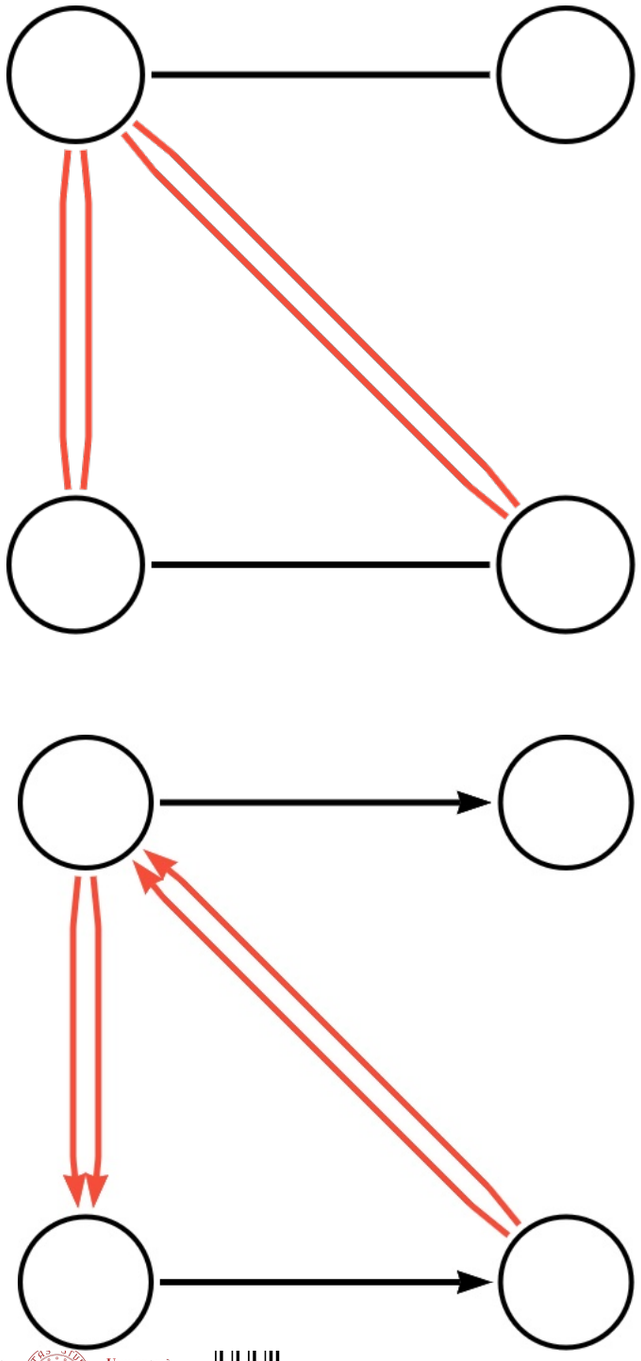


A multigraph is a graph in which multiple edges between the same two nodes are permitted

Note that this is an undirected multigraph

Property Graph Model

- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph

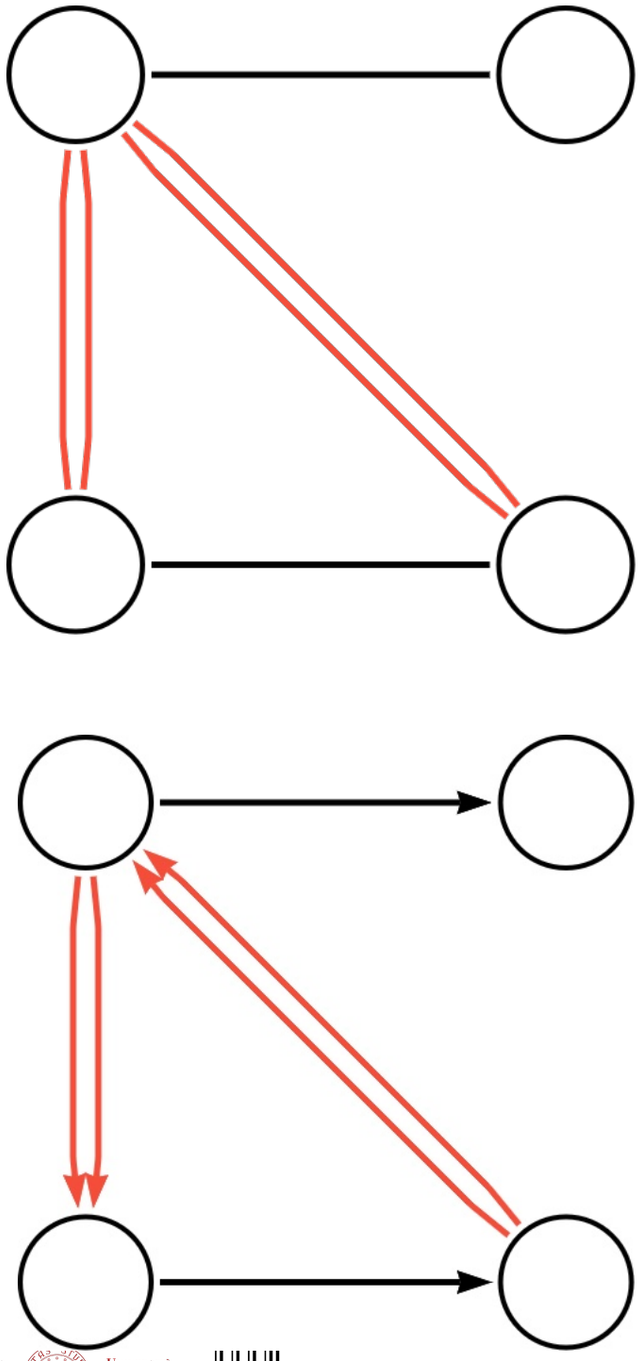


A multigraph is a graph in which multiple edges between the same two nodes are permitted

Note that this is an undirected multigraph

Property Graph Model

- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph



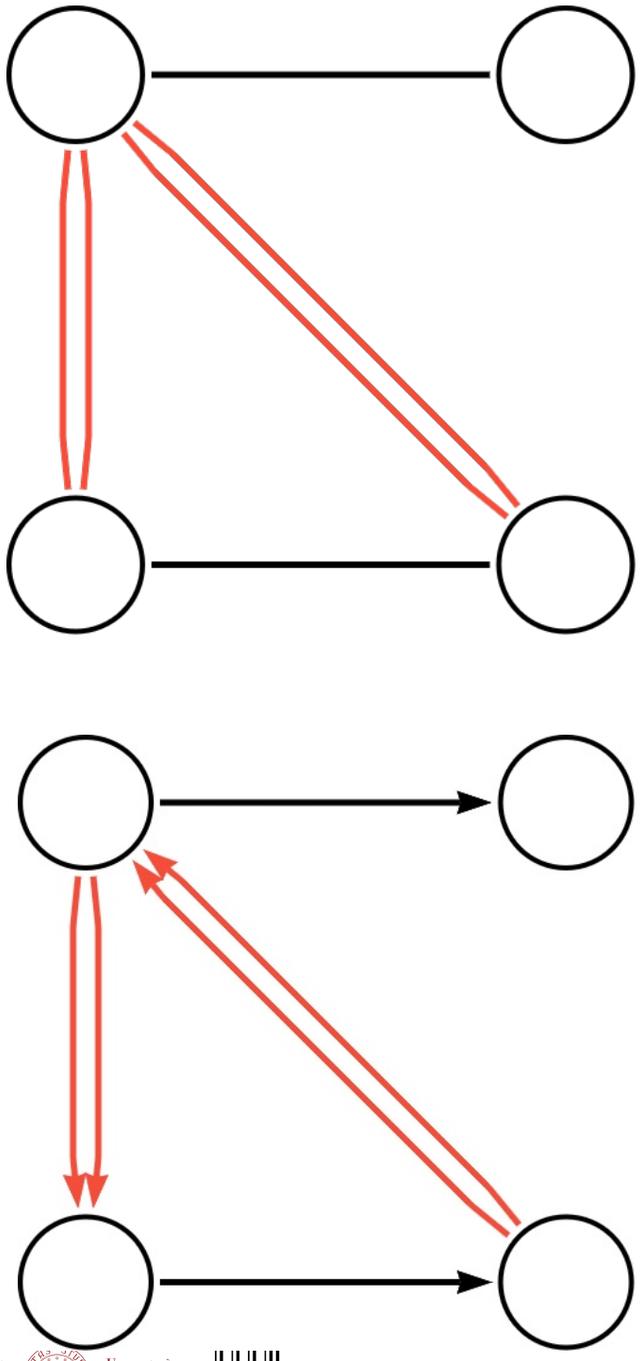
A multigraph is a graph in which multiple edges between the same two nodes are permitted

Note that this is an undirected multigraph

A directed multi-graph is a graph allowing multiple edges with the same starting and end nodes

Property Graph Model

- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph



A multigraph is a graph in which multiple edges between the same two nodes are permitted

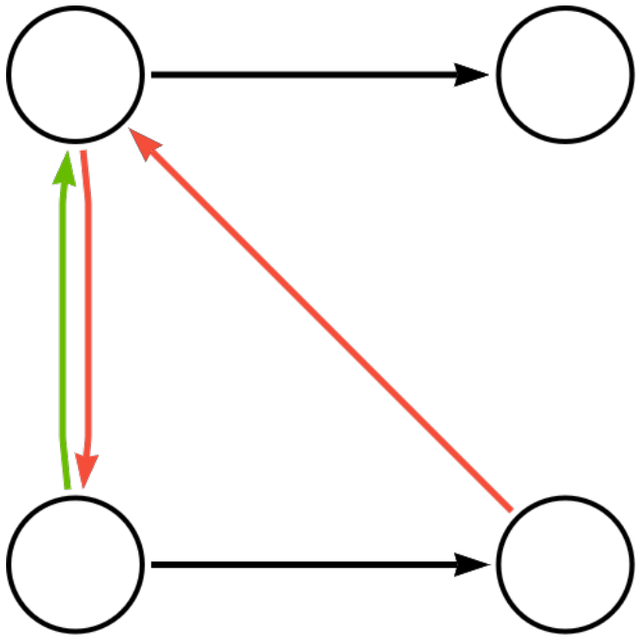
Note that this is an undirected multigraph

A directed multi-graph is a graph allowing multiple edges with the same starting and end nodes

A directed graph may have multiple directed edges from a vertex to a second (possibly the same) vertex are called as directed multigraphs

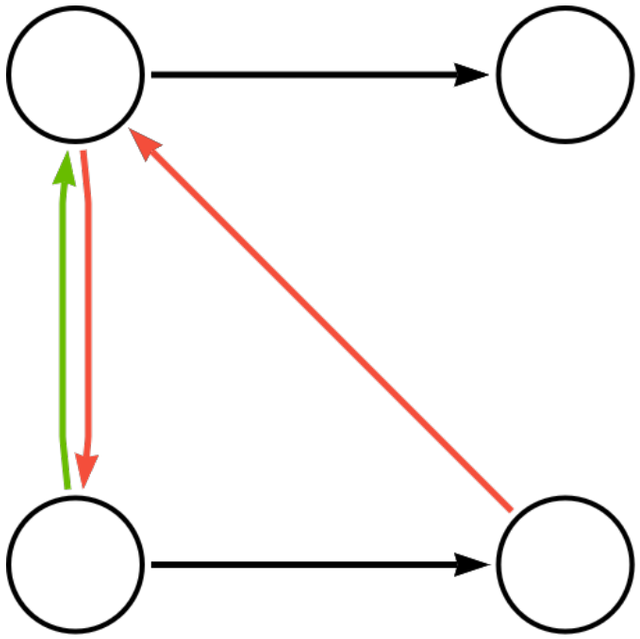
Property Graph Model

- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph



Property Graph Model

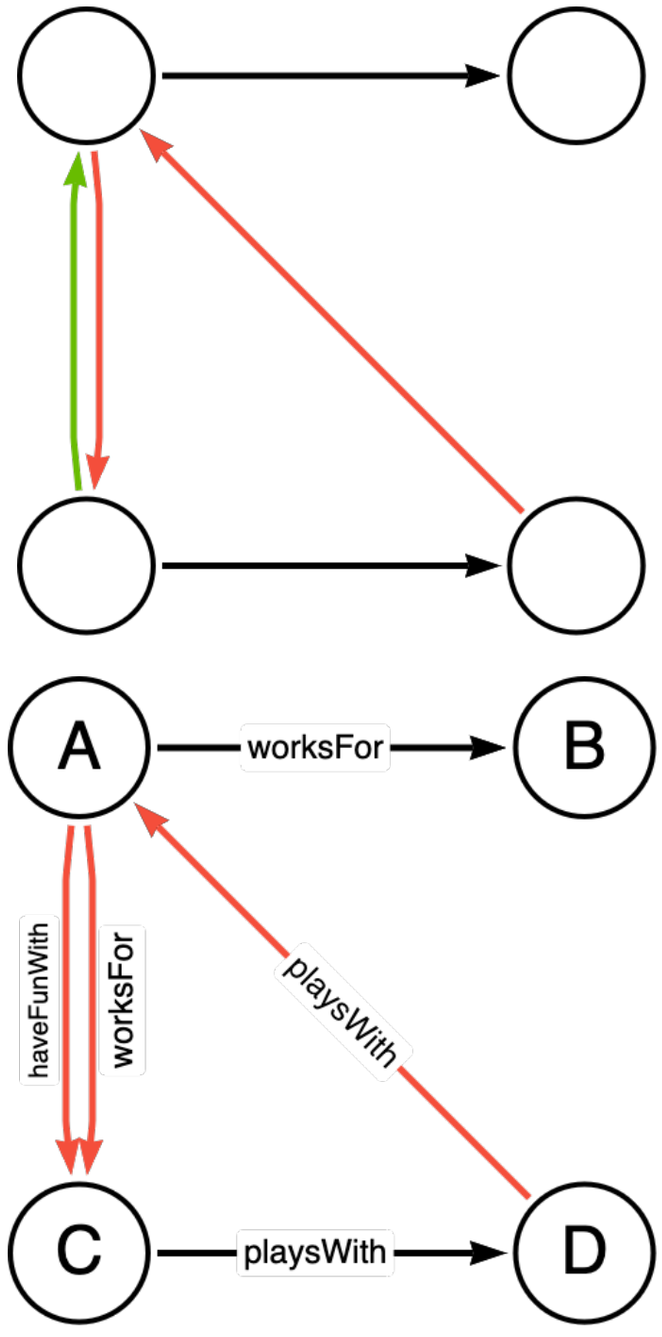
- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph



This is not a multi-graph

Property Graph Model

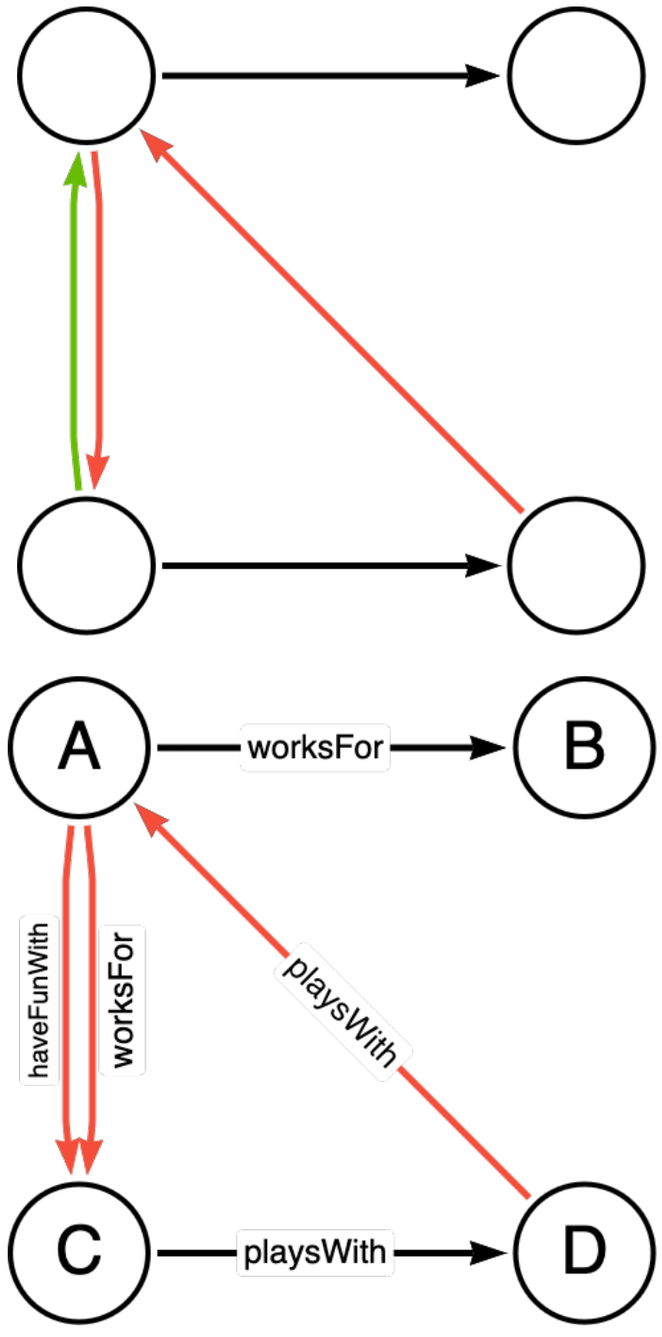
- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph



This is not a multi-graph

Property Graph Model

- The Property Graph Model (PGM) represents data as a directed, attributed multi-graph



This is not a multi-graph

A directed labeled multi-graph is a graph allowing multiple edges with the same starting and end nodes and where nodes and vertices are labelled

Property Graph Model

- In a PGM, vertices and edges are rich objects with a set of keys, a set of labels, and a set of values
- The pair (key, value) are called properties. E.g., name:'Jason'

Property Graph Model

- In a PGM, vertices and edges are rich objects with a set of keys, a set of labels, and a set of values
- The pair (key, value) are called properties. E.g., name:'Jason'

Property Graph Model

- In a PGM, vertices and edges are rich objects with a set of keys, a set of labels, and a set of values
- The pair (key, value) are called properties. E.g., name:'Jason'

property keys = {name, born, middleInitial, twitter}

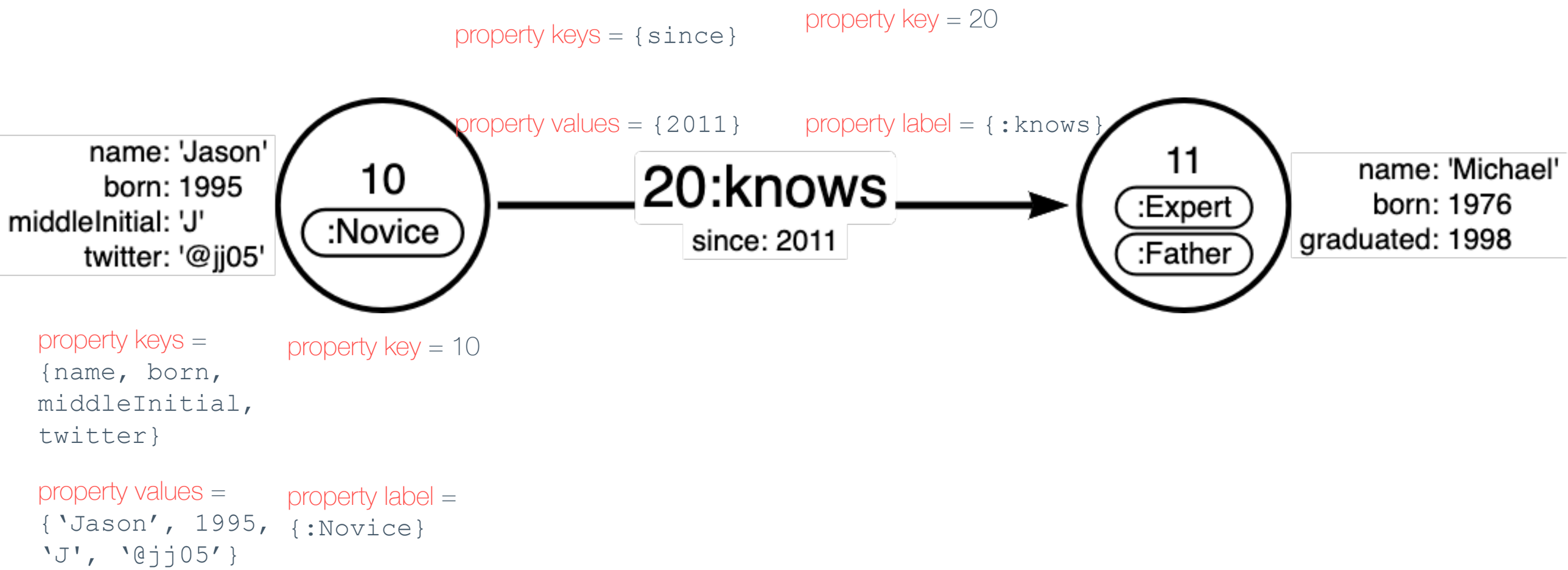
property key = 10

property values = {'Jason', 1995, 'J', '@jj05'}

property label = {:Novice}

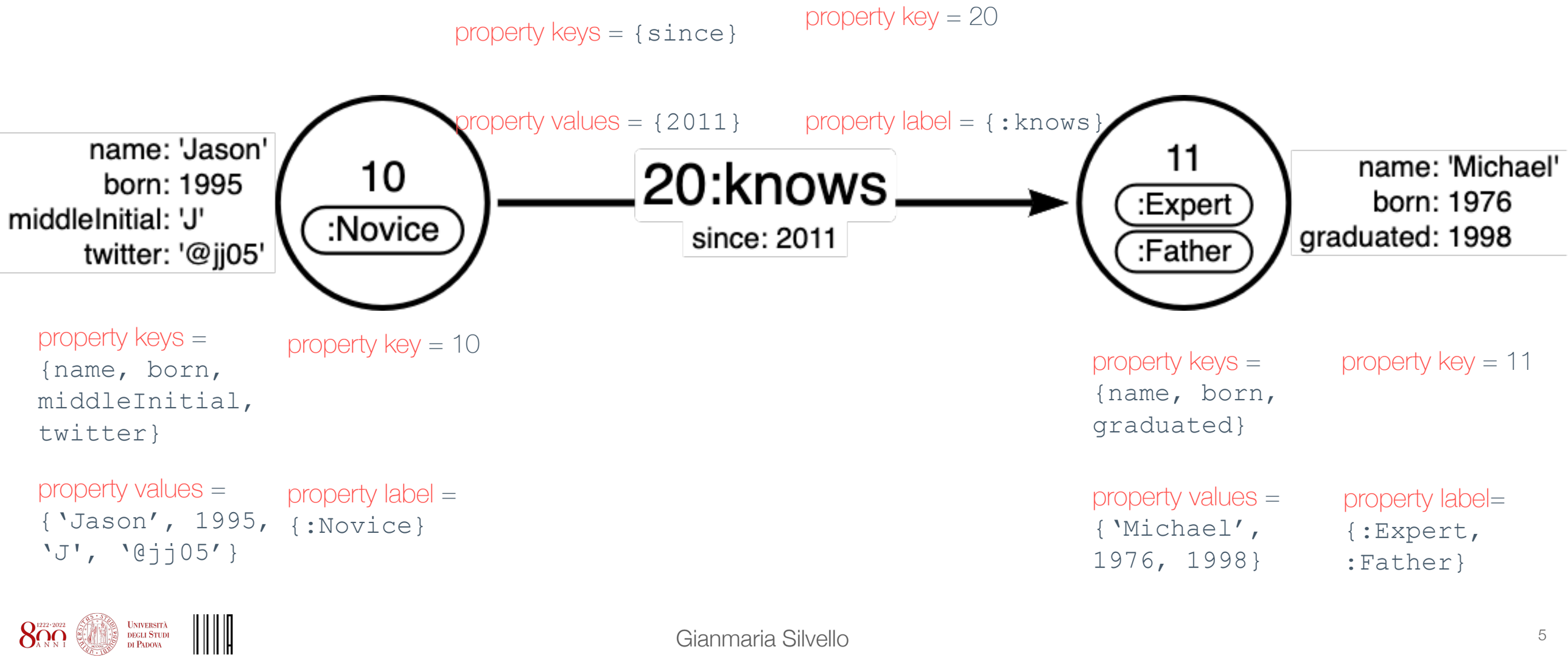
Property Graph Model

- In a PGM, vertices and edges are rich objects with a set of keys, a set of labels, and a set of values
- The pair (key, value) are called properties. E.g., name:'Jason'



Property Graph Model

- In a PGM, vertices and edges are rich objects with a set of keys, a set of labels, and a set of values
- The pair (key, value) are called properties. E.g., name:'Jason'



PGM

- We say that vertex 11 (or the vertex with key 11) has two labels: Expert and Father
- Labels are preceded by a colon and they are purely descriptive
- Properties contain actual data and are composed of a key and a value (may have a complex data type)
  - For instance, edge 20 has one label (knows) and one property (since) with a value (2011)

PGM definition

$\mathcal{O}$ is a set of objects

$\mathcal{L}$ is a finite set of labels

$\mathcal{K}$ is a set of property keys

$\mathcal{N}$ is a set of values

A property graph is a structure $(V, E, \eta, \lambda, \nu)$ where

(eta: η; ni: ν; lambda: λ)

- $V \subseteq \mathcal{O}$ is a finite set of objects, called vertices;
- $E \subseteq \mathcal{O}$ is a finite set of objects, called edges;
- $\eta : E \to V \times V$ is a function assigning to each edge an ordered pair of vertices;
- $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$ is a function assigning to each object a finite set of labels; and,

  $V \cap E = \emptyset$
- $\nu : V \cup E \times \mathcal{K} \to \mathcal{N}$ is a partial function assigning values for properties to objects.

PGM example


name: 'Jason'
born: 1995
middleInitial: 'J'
twitter: '@jj05'
10
:Novice
20:knows
since: 2011
11
:Expert
:Father
name: 'Michael'
born: 1976
graduated: 1998

PGM example

$$V = \{10, 11\}$$

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

$$\lambda(11) = \{\text{Expert}, \text{Father}\}$$

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

$$\lambda(11) = \{\text{Expert}, \text{Father}\}$$

$$\nu(10, \text{name}) = \{\text{Jason}\}$$

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

$$\lambda(11) = \{\mathsf{Expert}, \mathsf{Father}\}$$

$$\nu(10, \mathsf{name}) = \{\mathsf{Jason}\}$$

$$\eta(20) = (10, 11) \neq \{10, 11\}$$

PGM example

V (v1,v2) is an ordered set defining an order relation amongst the set members.

E

Let P be a set and ⊑ a partial order over P such that a,b ∈ P ⇒ a ⊑ b or b ⊑ a or a || b (incomparable).

λ

ν If all the elements of P are comparable then P is a totally ordered set.

$$\eta(20) = (10, 11) \neq \{10, 11\}$$

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

$$\lambda(11) = \{\text{Expert}, \text{Father}\}$$

$$\nu(10, \text{name}) = \{\text{Jason}\}$$

$$\eta(20) = (10, 11) \neq \{10, 11\}$$

Notation

π_i projects an n-tuple to its i-th element

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

$$\lambda(11) = \{\mathsf{Expert}, \mathsf{Father}\}$$

$$\nu(10, \mathsf{name}) = \{\mathsf{Jason}\}$$

$$\eta(20) = (10, 11) \neq \{10, 11\}$$

Notation

π_i projects an n-tuple to its i-th element

$$\pi_2(a, b, c, d, e) = b$$

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

$$\lambda(11) = \{\mathsf{Expert}, \mathsf{Father}\}$$

$$\nu(10, \mathsf{name}) = \{\mathsf{Jason}\}$$

$$\eta(20) = (10, 11) \neq \{10, 11\}$$

Notation

π_i projects an n-tuple to its i-th element

$$\pi_2(a, b, c, d, e) = b$$

$$\mathtt{in}(v) = \{e \in E \mid \pi_2(\eta(e)) = v\}$$

is the set of incoming edges of v

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

$$\lambda(11) = \{\mathsf{Expert}, \mathsf{Father}\}$$

$$\nu(10, \mathsf{name}) = \{\mathsf{Jason}\}$$

$$\eta(20) = (10, 11) \neq \{10, 11\}$$

Notation

π_i projects an n-tuple to its i-th element

$$\pi_2(a, b, c, d, e) = b$$

$$\mathtt{in}(v) = \{e \in E \mid \pi_2(\eta(e)) = v\}$$

is the set of incoming edges of v

$$\mathtt{out}(v) = \{e \in E \mid \pi_1(\eta(e)) = v\}$$

is the set of outgoing edges of v

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

$$\lambda(11) = \{\mathsf{Expert}, \mathsf{Father}\}$$

$$\nu(10, \mathsf{name}) = \{\mathsf{Jason}\}$$

$$\eta(20) = (10, 11) \neq \{10, 11\}$$

Notation

π_i projects an n-tuple to its i-th element

$$\pi_2(a, b, c, d, e) = b$$

$$\mathtt{in}(v) = \{e \in E \mid \pi_2(\eta(e)) = v\}$$

is the set of incoming edges of v

$$\mathtt{out}(v) = \{e \in E \mid \pi_1(\eta(e)) = v\}$$

is the set of outgoing edges of v

$$\mathtt{src}(e) = \pi_1(\eta(e))$$

is the source vertex of e

PGM example

$$V = \{10, 11\}$$

$$E = \{20\}$$

$$\lambda(11) = \{\mathsf{Expert}, \mathsf{Father}\}$$

$$\nu(10, \mathsf{name}) = \{\mathsf{Jason}\}$$

$$\eta(20) = (10, 11) \neq \{10, 11\}$$

Notation

π_i projects an n-tuple to its i-th element

$$\pi_2(a, b, c, d, e) = b$$

$$\mathtt{in}(v) = \{e \in E \mid \pi_2(\eta(e)) = v\}$$

is the set of incoming edges of v

$$\mathtt{out}(v) = \{e \in E \mid \pi_1(\eta(e)) = v\}$$

is the set of outgoing edges of v

$$\mathtt{src}(e) = \pi_1(\eta(e))$$

is the source vertex of e

$$\mathtt{trg}(e) = \pi_2(\eta(e))$$

is the target vertex of e

Query Languages

Regular path queries (RPQ)

Regular path queries express reachability queries

RPQ ask for all pair of vertices connected by at least one path where the sequence of edge labels along the path forms a word in the language of a given regular expression

Regular path queries (RPQ)

Regular path queries express reachability queries

RPQ ask for all pair of vertices connected by at least one path where the sequence of edge labels along the path forms a word in the language of a given regular expression

Regular path queries (RPQ)

Regular path queries express reachability queries

RPQ ask for all pair of vertices connected by at least one path where the sequence of edge labels along the path forms a word in the language of a given regular expression

Syntax

set of labels

if $a \in \mathcal{L} \Rightarrow a \in RPQ$

if $e \in RPQ \Rightarrow e^- \in RPQ$

if $e, f \in RPQ \Rightarrow e/f \in RPQ$

if $e, f \in RPQ \Rightarrow e+f \in RPQ$

if $e \in RPQ \Rightarrow e^+ \in RPQ$

Regular path queries (RPQ)

Regular path queries express reachability queries

RPQ ask for all pair of vertices connected by at least one path where the sequence of edge labels along the path forms a word in the language of a given regular expression

Syntax

set of labels

if $a \in \mathcal{L} \Rightarrow a \in RPQ$

if $e \in RPQ \Rightarrow e^- \in RPQ$ → 2RPQ: 2-way regular path query

if $e, f \in RPQ \Rightarrow e/f \in RPQ$

e^- is the inverse of e

↳ it goes from the object to the subject

if $e, f \in RPQ \Rightarrow e+f \in RPQ$

if $e \in RPQ \Rightarrow e^+ \in RPQ$

Regular path queries (RPQ)

Regular path queries express reachability queries

RPQ ask for all pair of vertices connected by at least one path where the sequence of edge labels along the path forms a word in the language of a given regular expression

Syntax — set of labels

if $a \in \mathcal{L} \Rightarrow a \in RPQ$

if $e \in RPQ \Rightarrow e^- \in RPQ$ → 2RPQ: 2-way regular path query

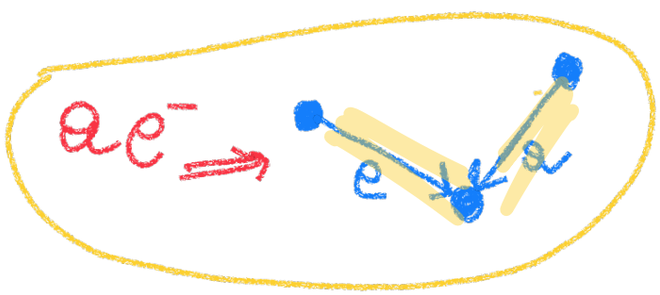
if $e, f \in RPQ \Rightarrow e/f \in RPQ$

e^- is the inverse of e

↳ it goes from the object to the subject

if $e, f \in RPQ \Rightarrow e+f \in RPQ$

if $e \in RPQ \Rightarrow e^+ \in RPQ$



Regular path queries (RPQ)

Regular path queries express reachability queries

RPQ ask for all pair of vertices connected by at least one path where the sequence of edge labels along the path forms a word in the language of a given regular expression

Syntax

set of labels

if $a \in \mathcal{L} \Rightarrow a \in RPQ$

if $e \in RPQ \Rightarrow e^- \in RPQ$

if $e, f \in RPQ \Rightarrow e/f \in RPQ$

if $e, f \in RPQ \Rightarrow e+f \in RPQ$

if $e \in RPQ \Rightarrow e^+ \in RPQ$

Regular path queries (RPQ)

Regular path queries express reachability queries

RPQ ask for all pair of vertices connected by at least one path where the sequence of edge labels along the path forms a word in the language of a given regular expression

Syntax

set of labels

if $a \in \mathcal{L} \Rightarrow a \in RPQ$

if $e \in RPQ \Rightarrow e^- \in RPQ$

if $e, f \in RPQ \Rightarrow e/f \in RPQ$

↳ e minus f

if $e, f \in RPQ \Rightarrow e+f \in RPQ$

if $e \in RPQ \Rightarrow e^+ \in RPQ$

Regular path queries (RPQ)

Regular path queries express reachability queries

RPQ ask for all pair of vertices connected by at least one path where the sequence of edge labels along the path forms a word in the language of a given regular expression

Syntax

set of labels

if $a \in \mathcal{L} \Rightarrow a \in RPQ$

if $e \in RPQ \Rightarrow e^- \in RPQ$

if $e, f \in RPQ \Rightarrow e/f \in RPQ$

↳ e minus f

if $e, f \in RPQ \Rightarrow e+f \in RPQ$

↳ union of e and f

if $e \in RPQ \Rightarrow e^+ \in RPQ$

Regular path queries (RPQ)

Regular path queries express reachability queries

RPQ ask for all pair of vertices connected by at least one path where the sequence of edge labels along the path forms a word in the language of a given regular expression

Syntax

set of labels

if $a \in \mathcal{L} \Rightarrow a \in RPQ$

if $e \in RPQ \Rightarrow e^- \in RPQ$

if $e, f \in RPQ \Rightarrow e/f \in RPQ$

↳ e minus f

if $e, f \in RPQ \Rightarrow e+f \in RPQ$

↳ union of e and f

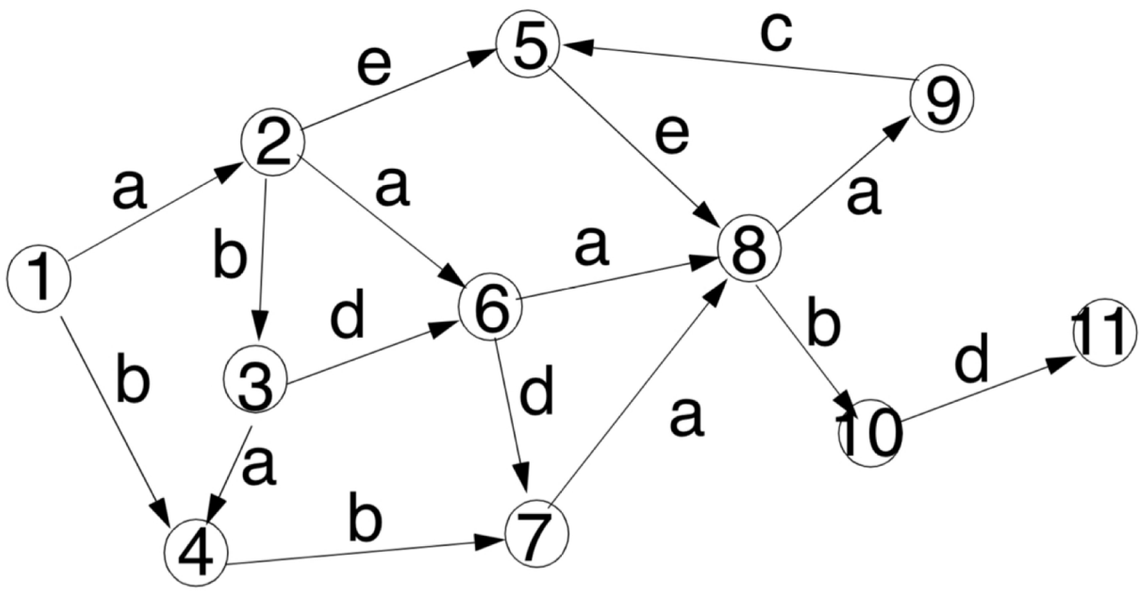
if $e \in RPQ \Rightarrow e^+ \in RPQ$

↳ one or more e

Example

$R = a^{+}(b|d)\,ab$

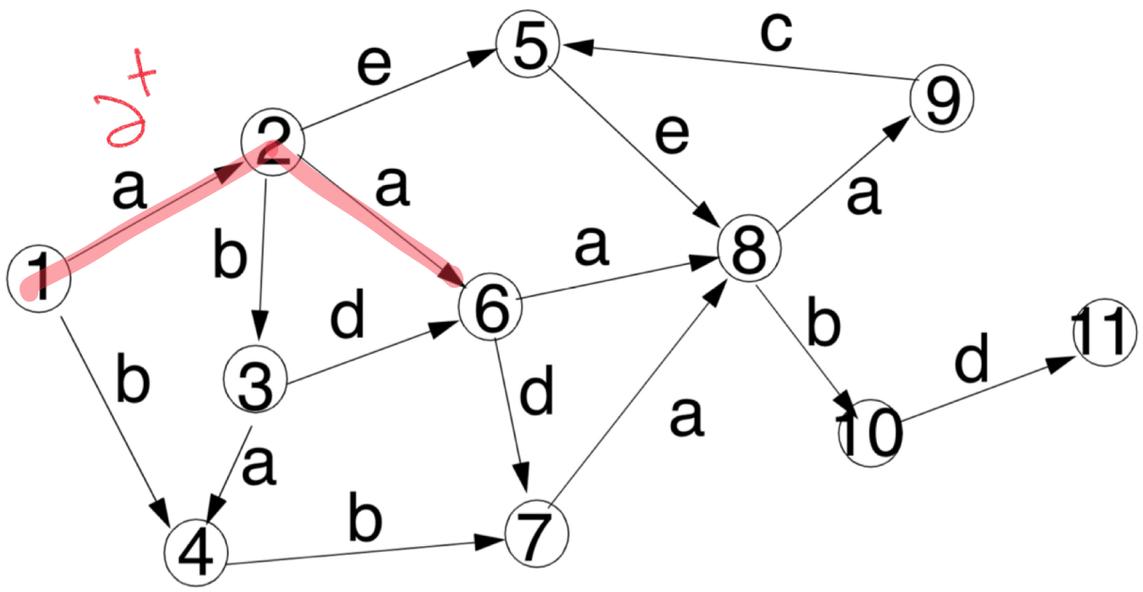
- a^{+}: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b



Example

$R = a^{+}(b|d)\,ab$

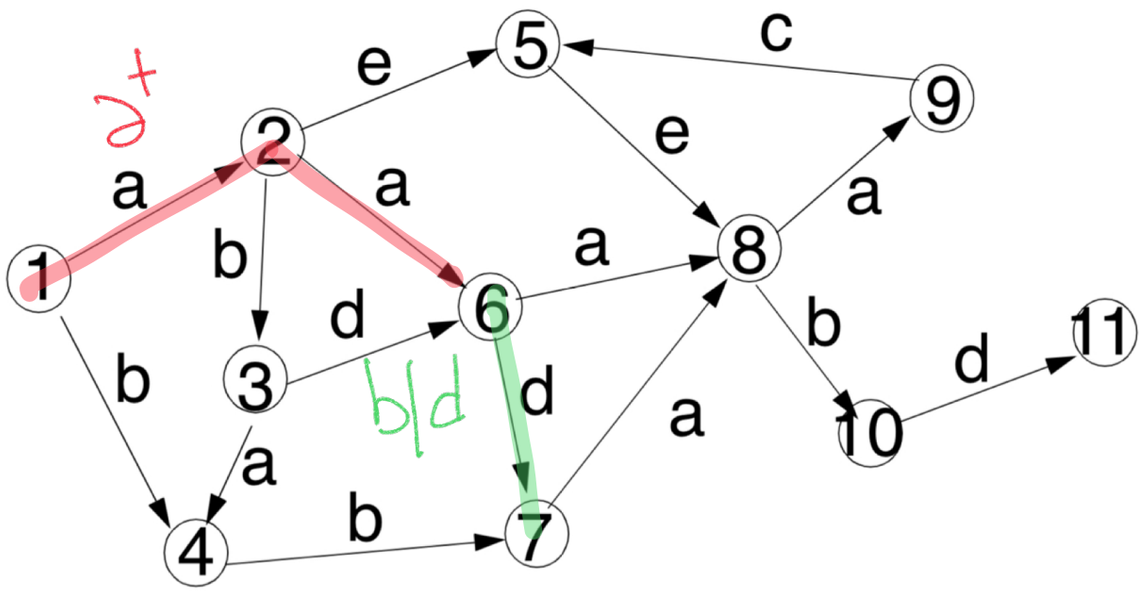
- a^{+}: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b



Example

$R = a^{+}(b|d)\,ab$

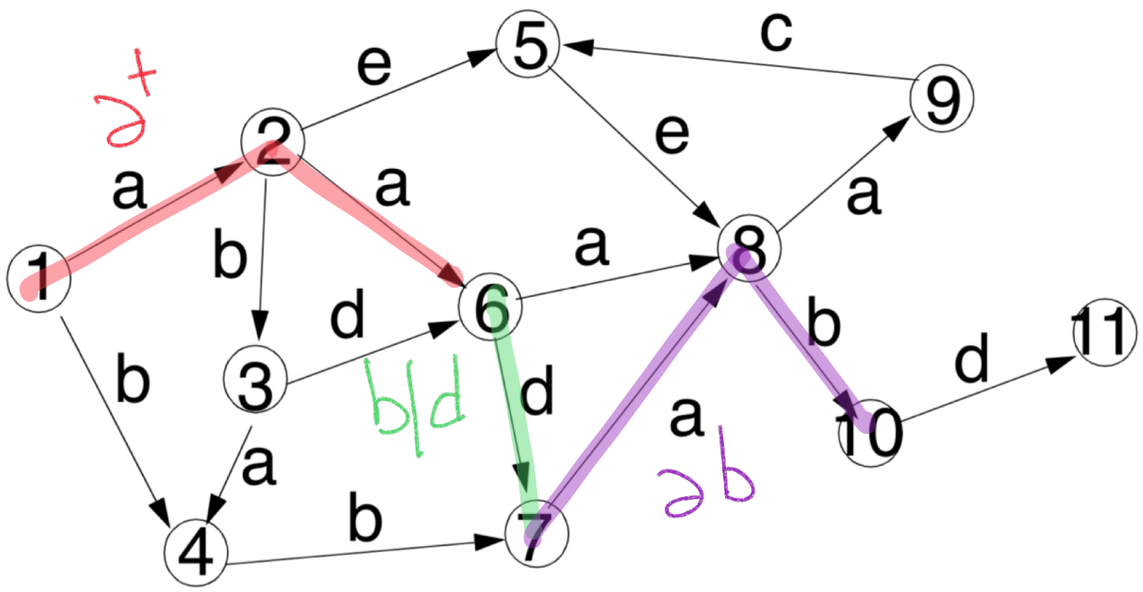
- a^{+}: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b



Example

$R = a^+ (b|d)\ ab$

- a^+: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b



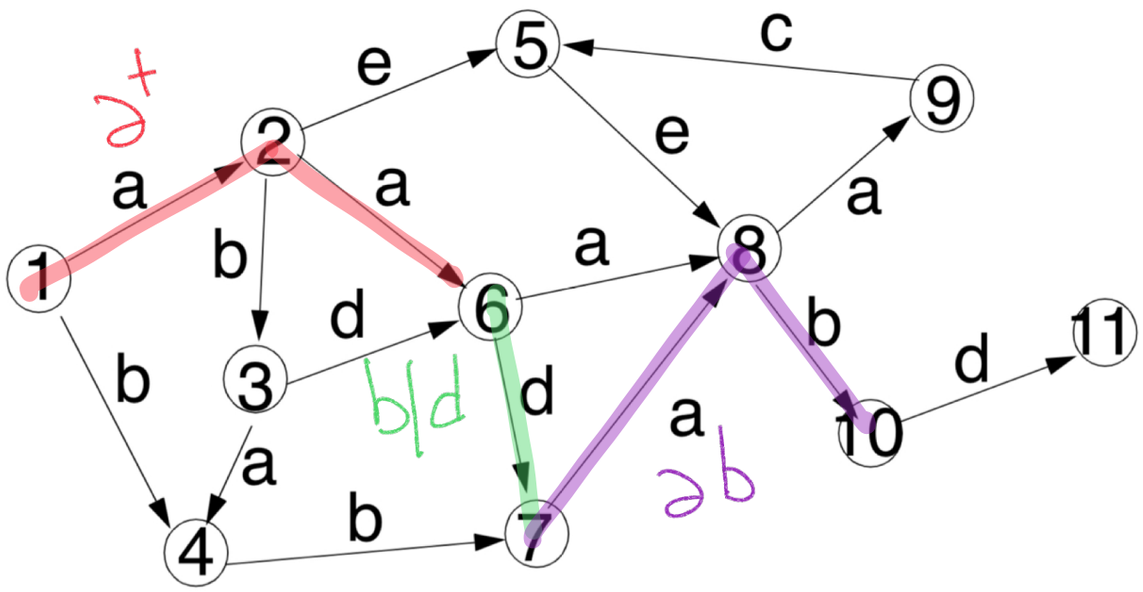
Example

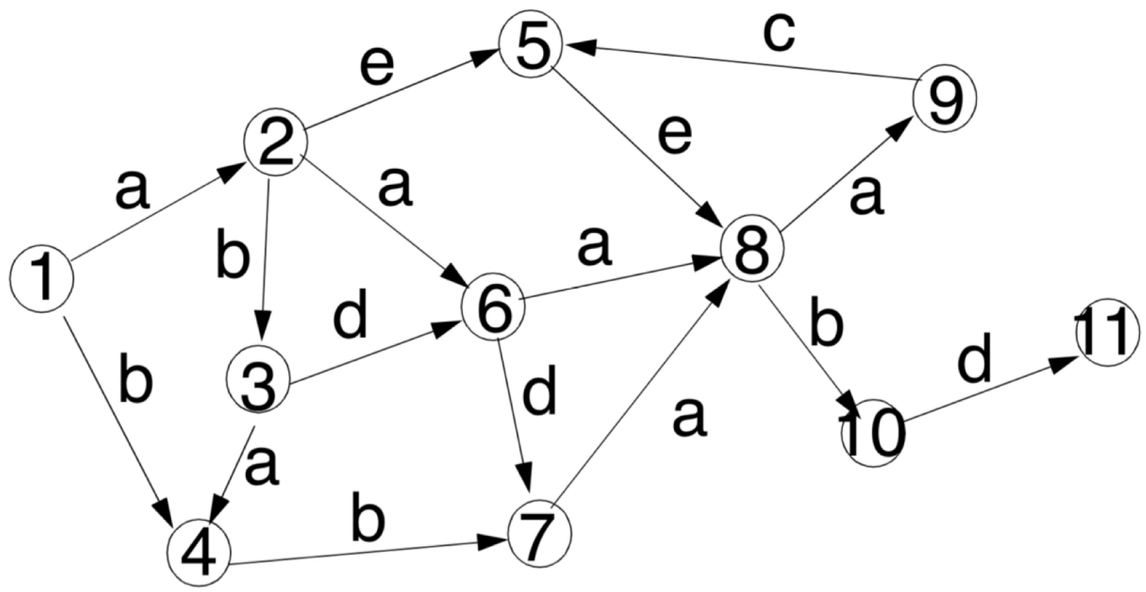
$R = a^+ (b|d)\ ab$

one or more repetitions of a

one occurrence of b OR d

one occurrence of a AND b

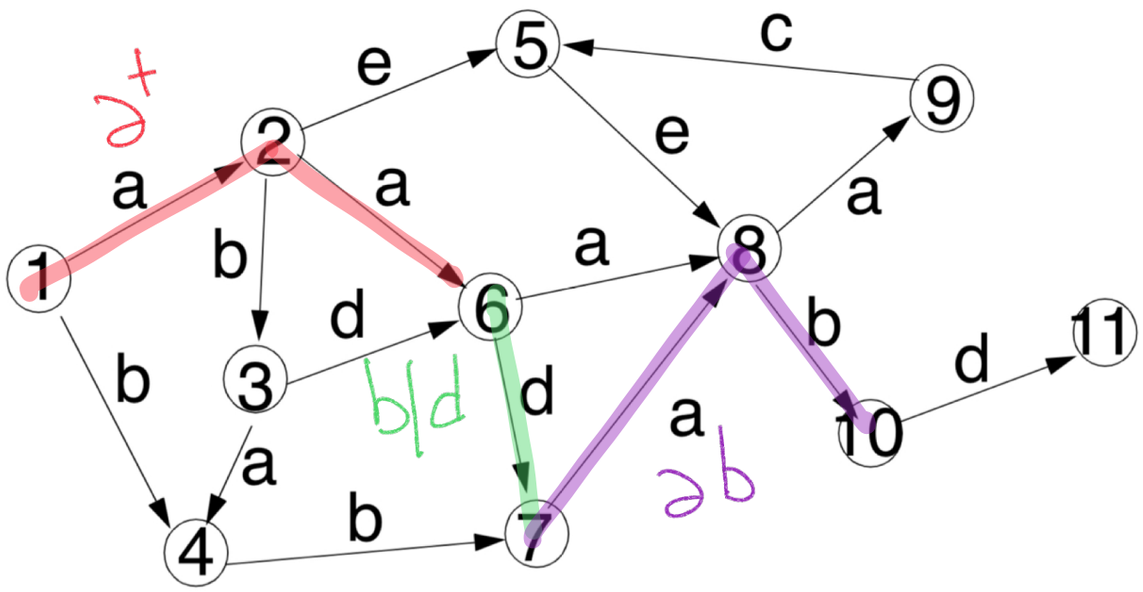


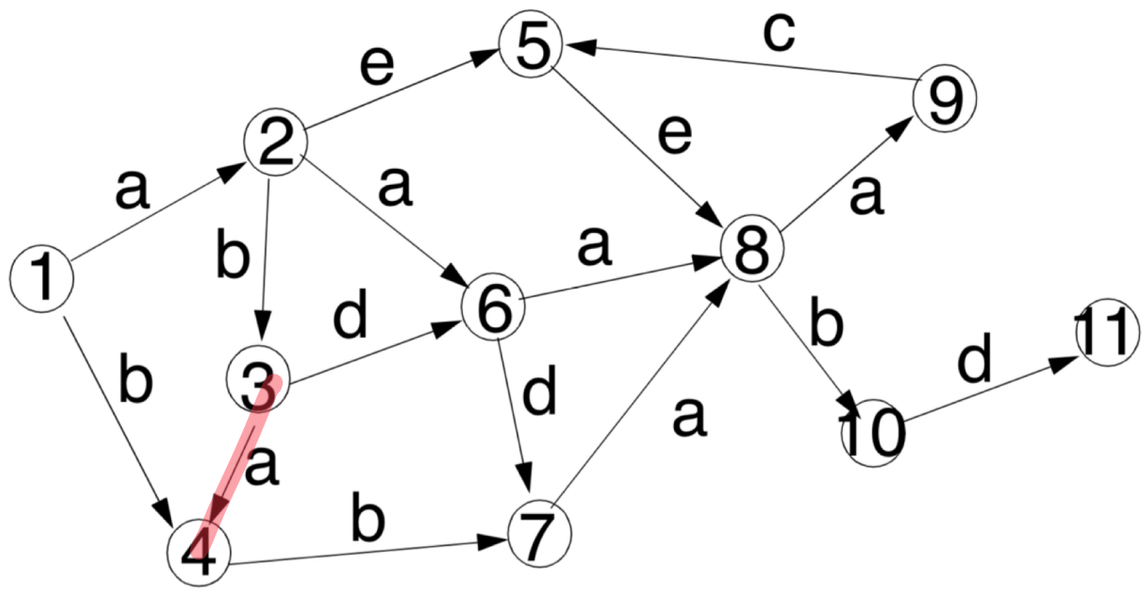


Example

$R = a^+ (b|d)\ ab$

- a^+: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b

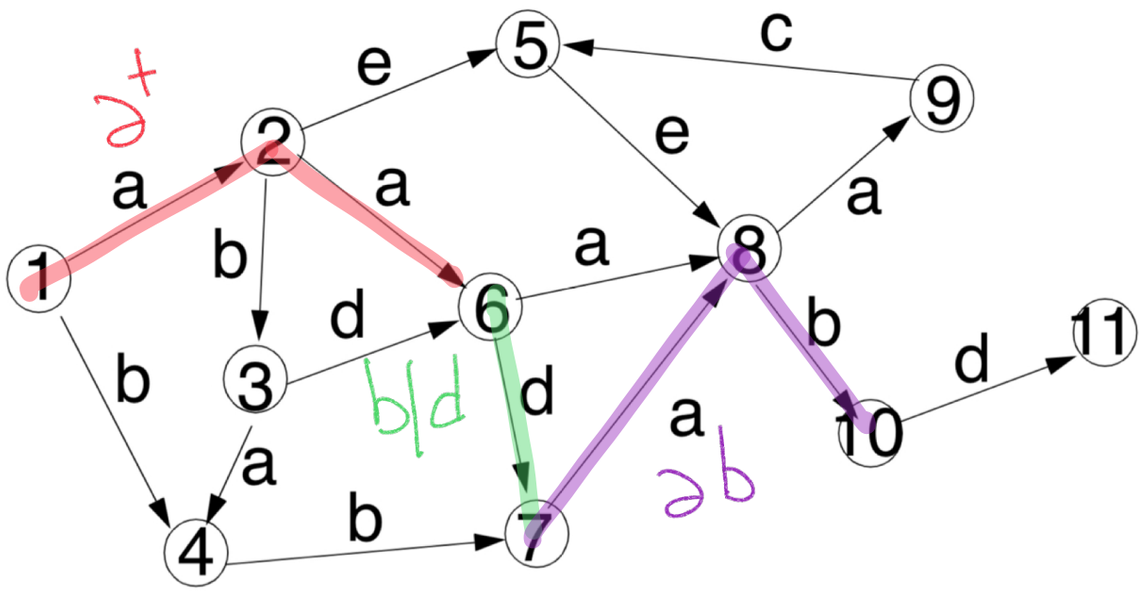


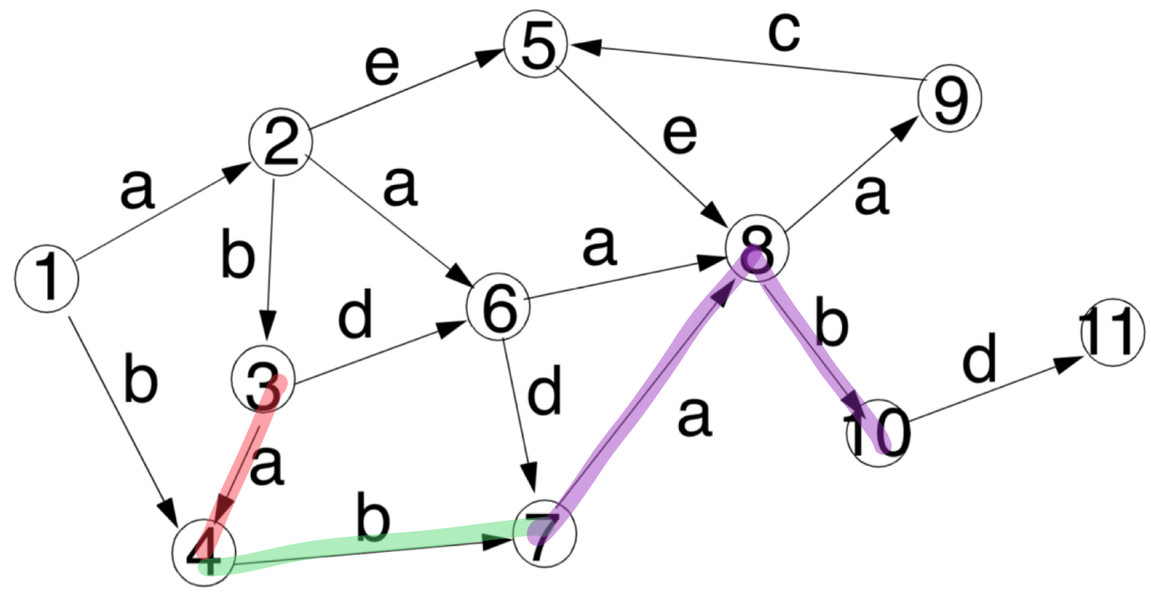


Example

$R = a^+ (b|d)\, ab$

- a^+: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b

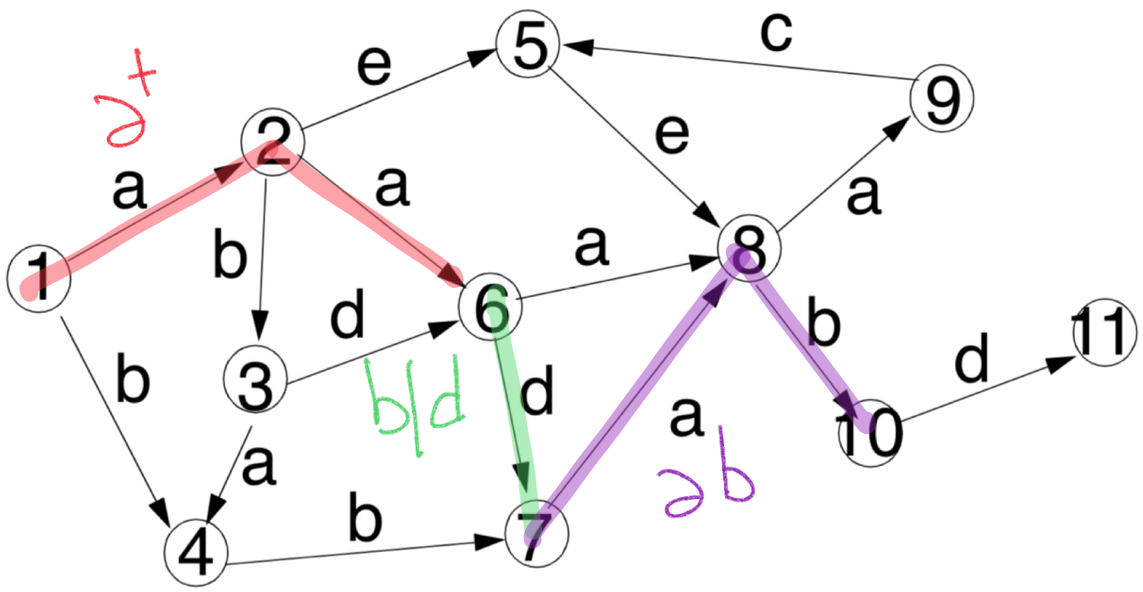


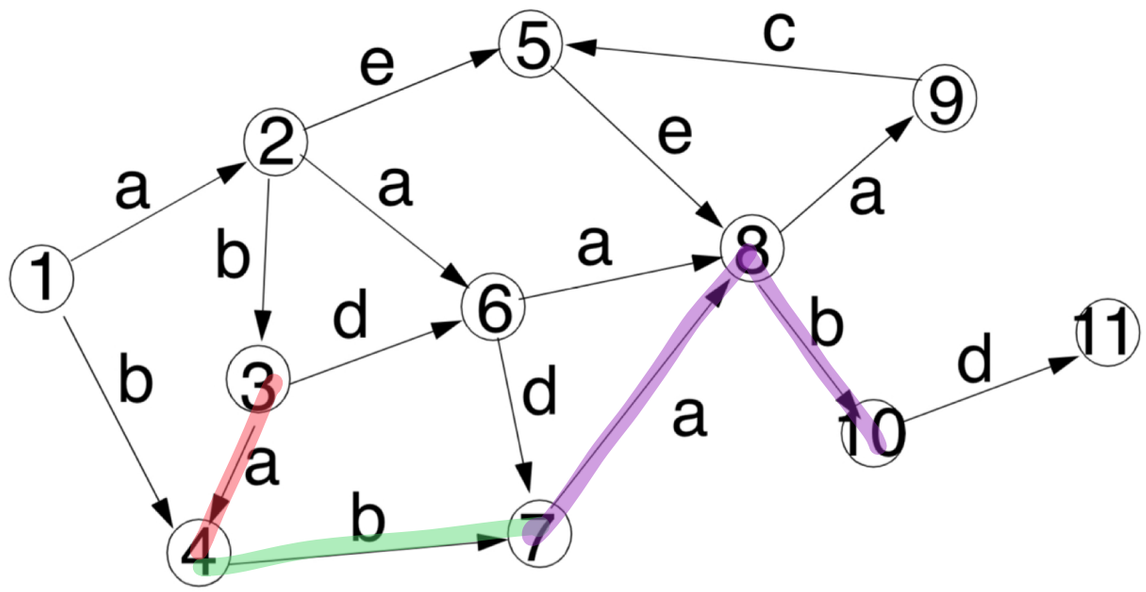


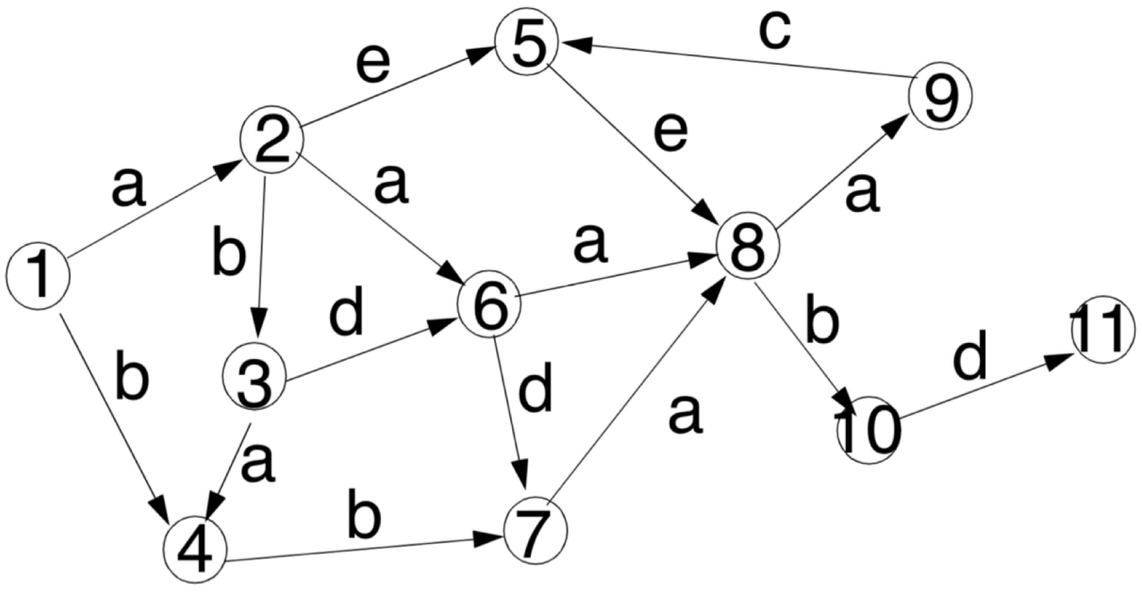
Example

$R = a^+ (b|d)\ ab$

- a^+: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b



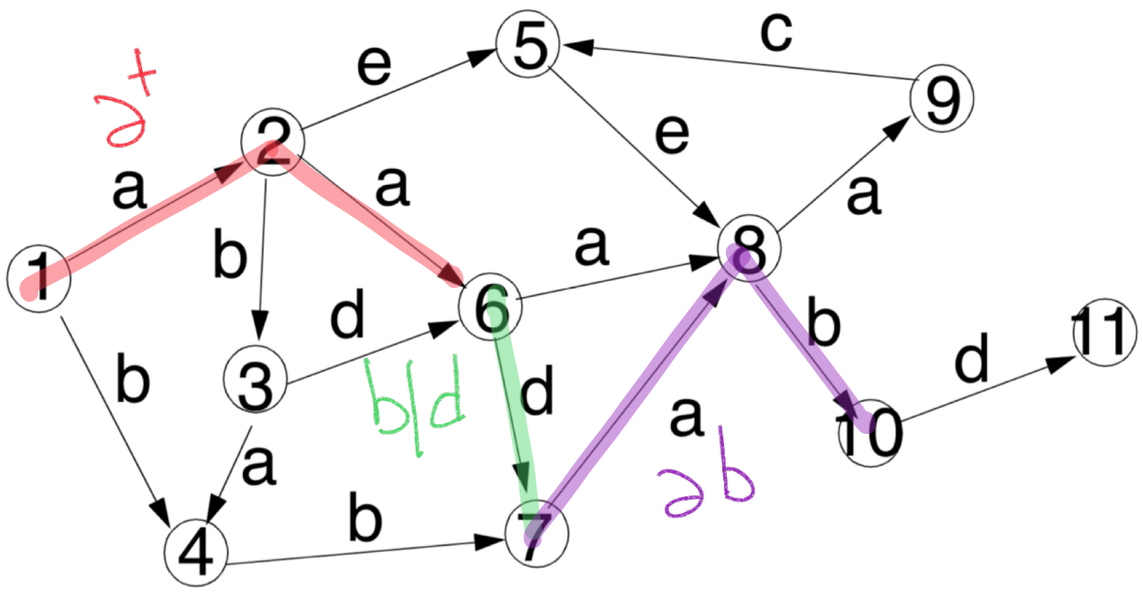


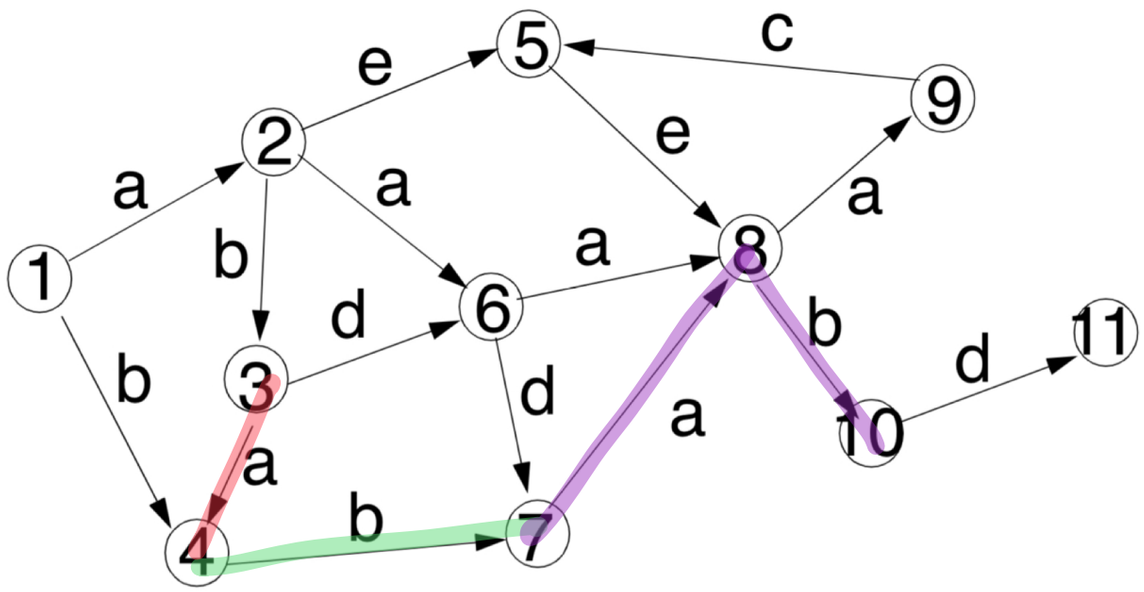


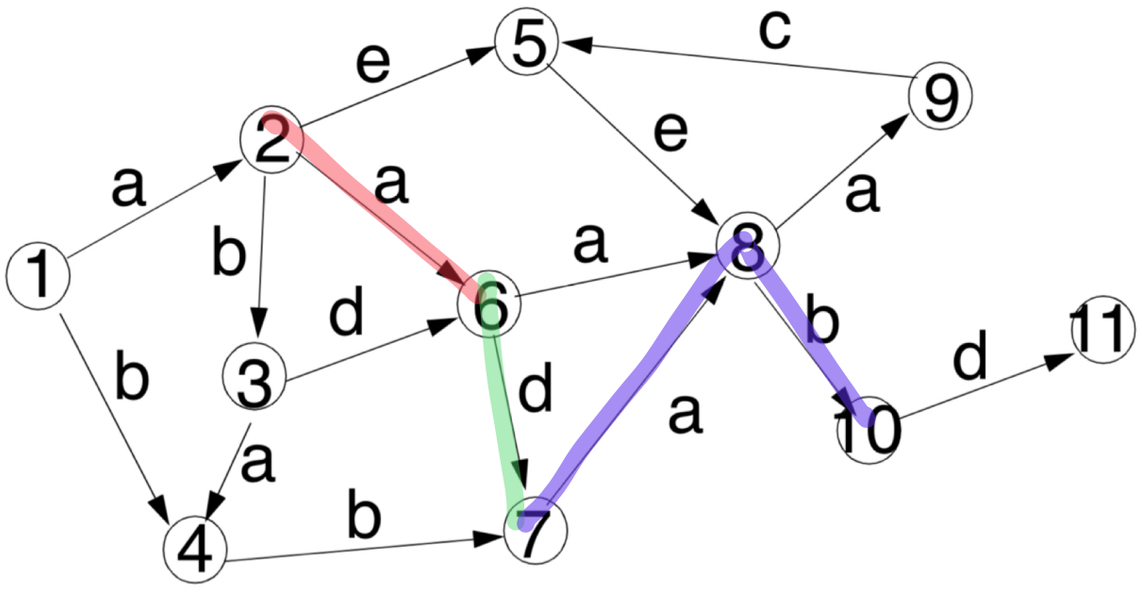
Example

$R = a^+ (b|d)\ ab$

- a^+: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b



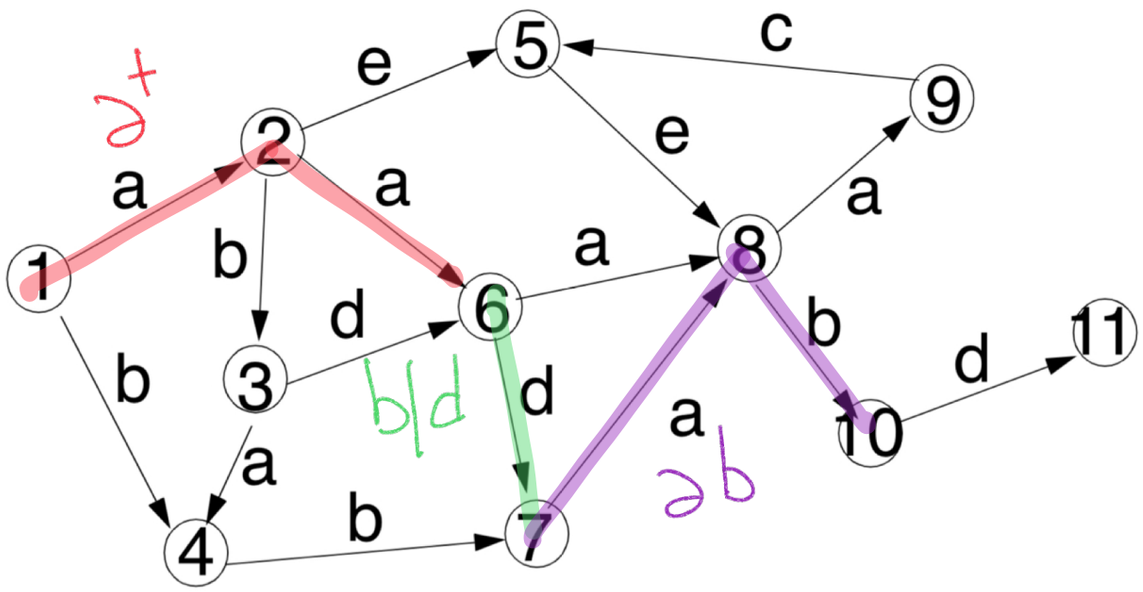


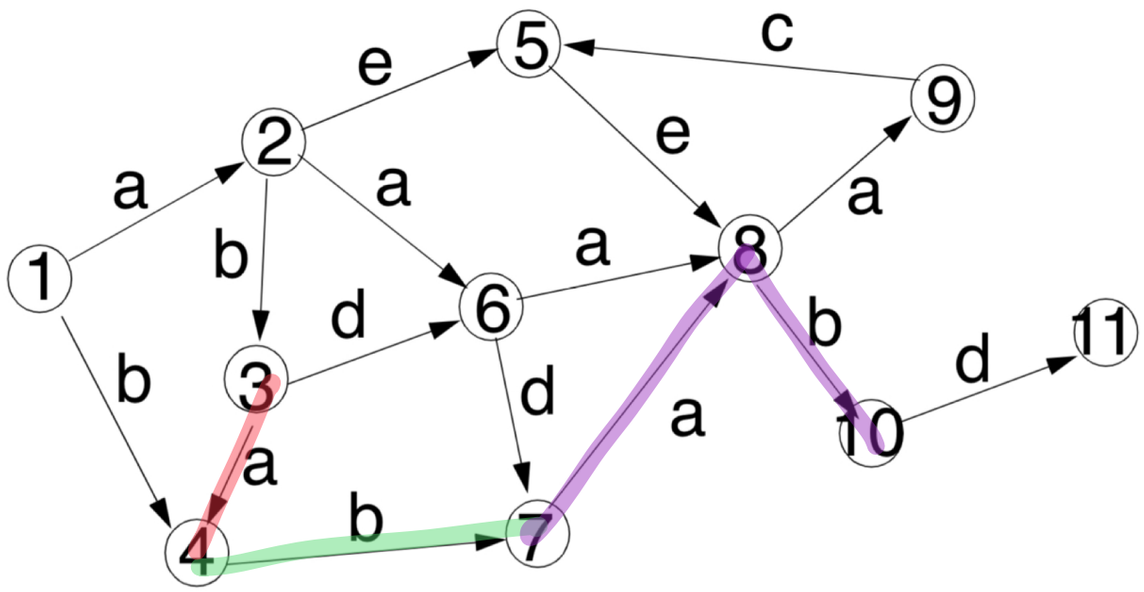


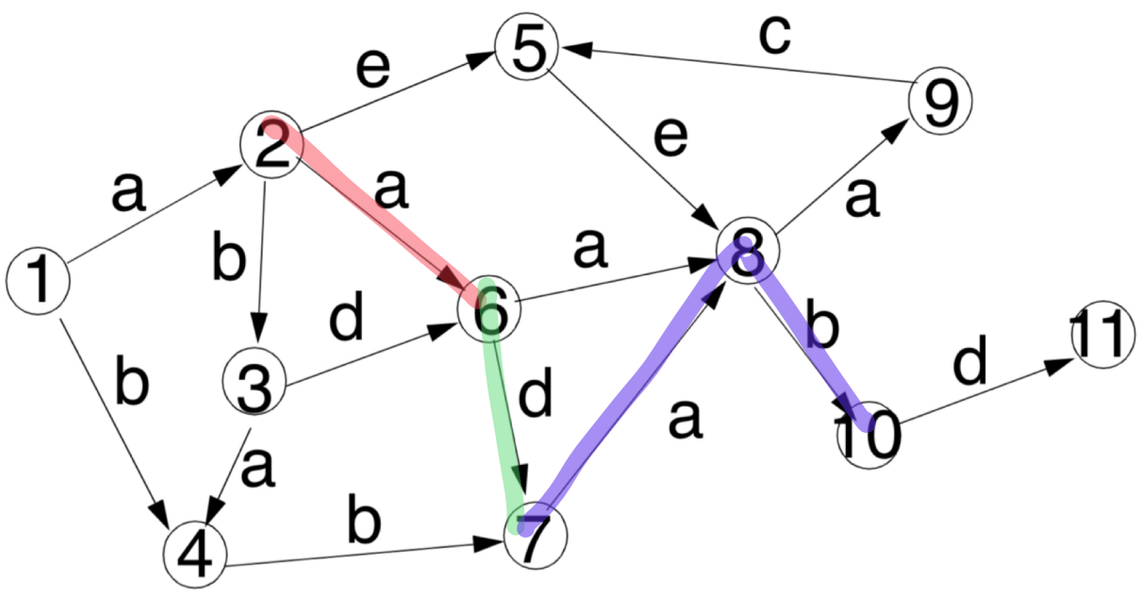
Example

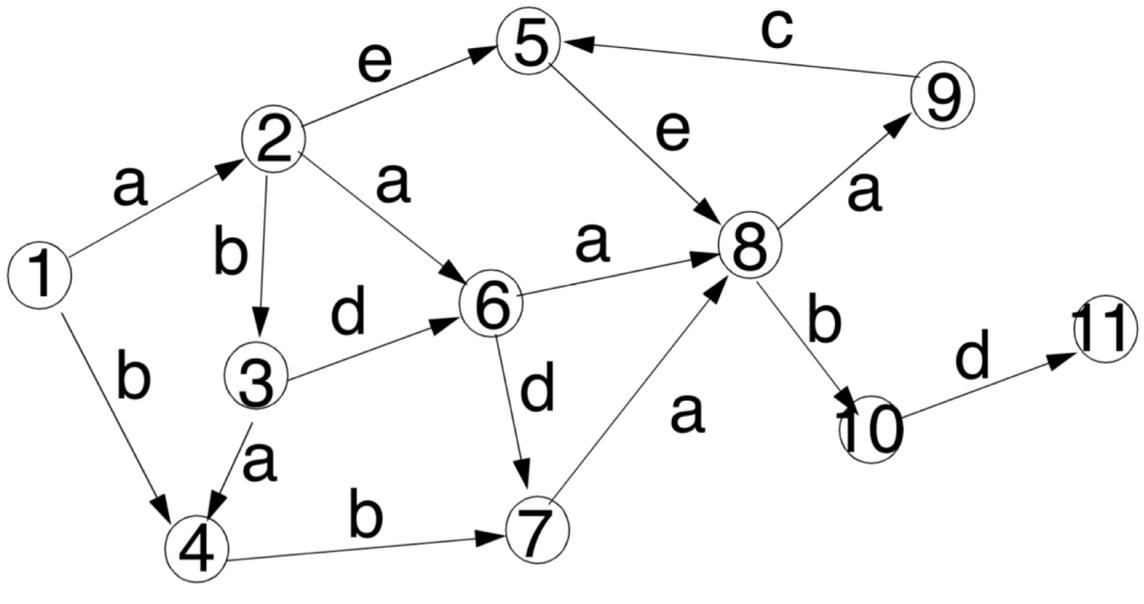
$R = a^+ (b|d)\ ab$

- a^+: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b





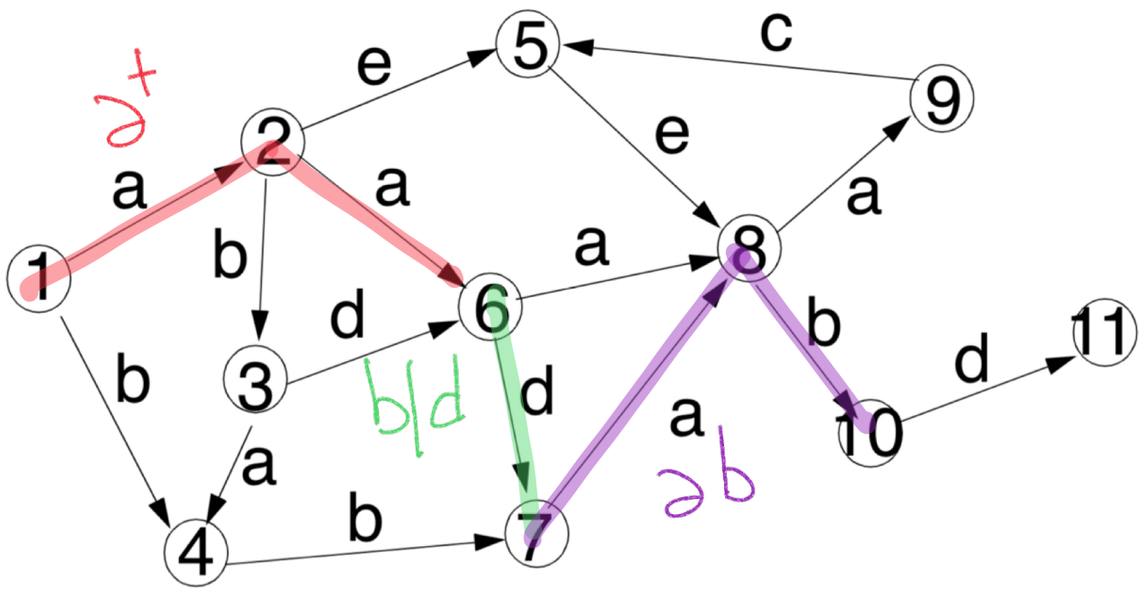


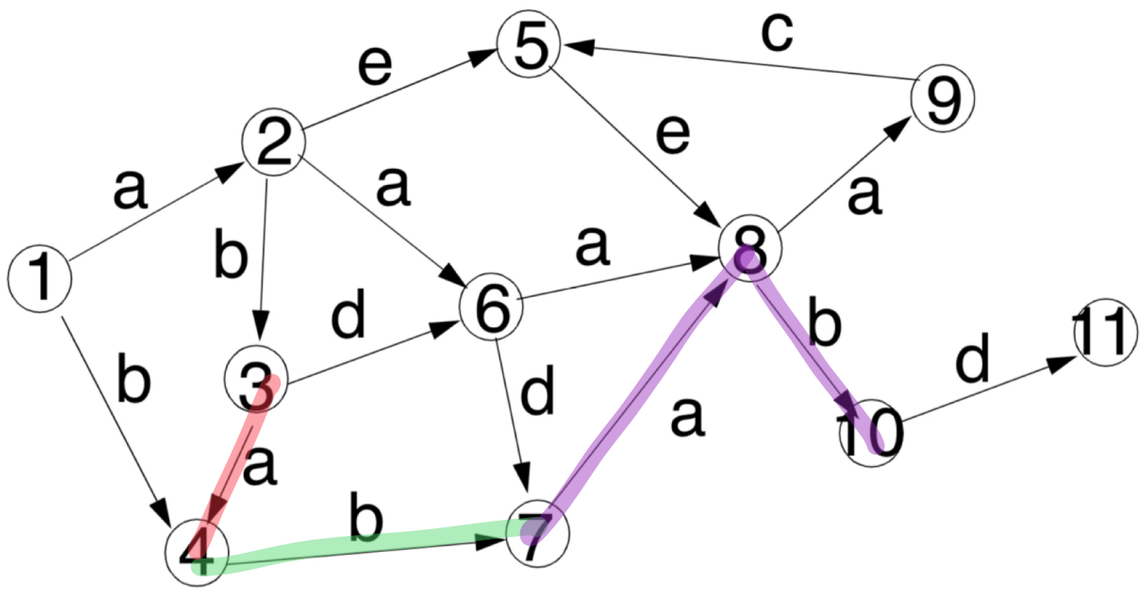


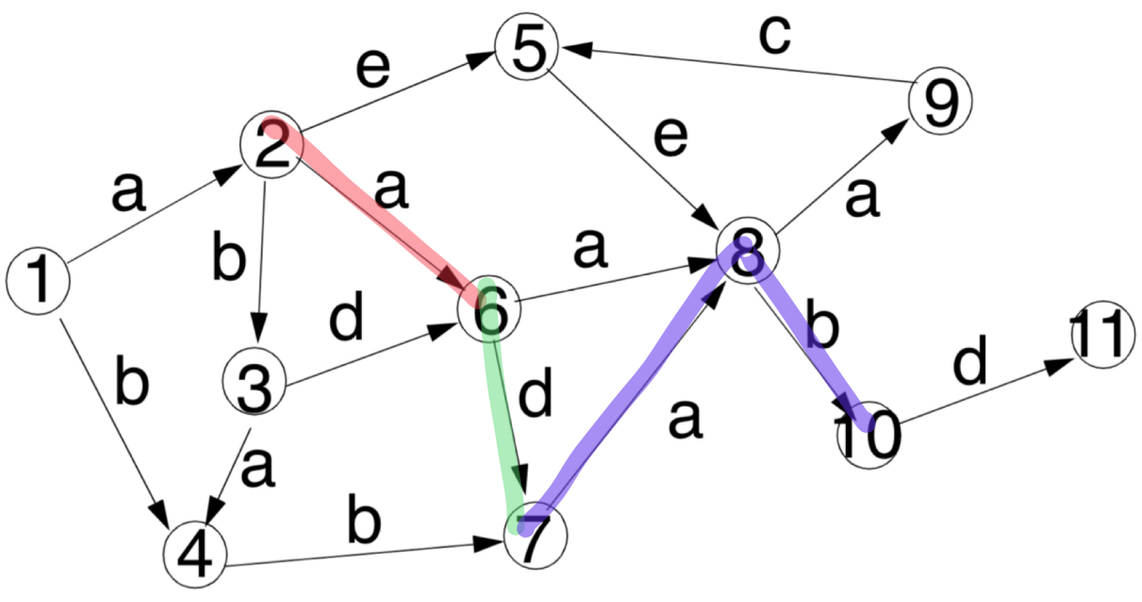
Example

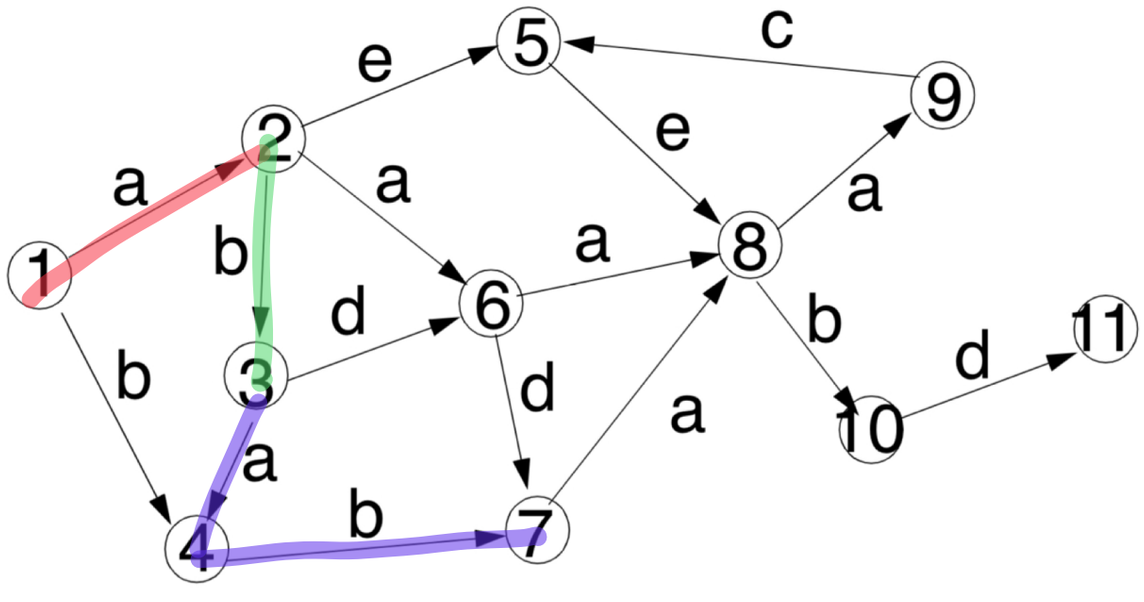
$R = a^+ (b|d)\ ab$

- a^+: one or more repetitions of a
- $(b|d)$: one occurrence of b OR d
- ab: one occurrence of a AND b









Example 2

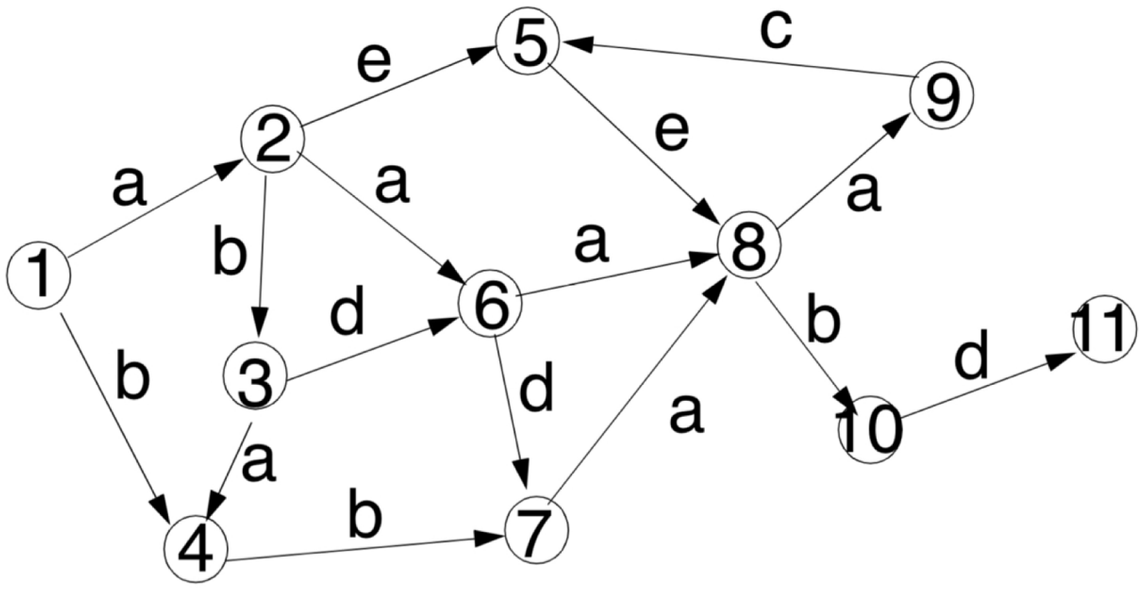
$R = a^{+} d^{-} a\ b^{-}$

a^{+} → one or more a

d^{-} → reverse d

a → one a

b^{-} → reverse b



Example 2

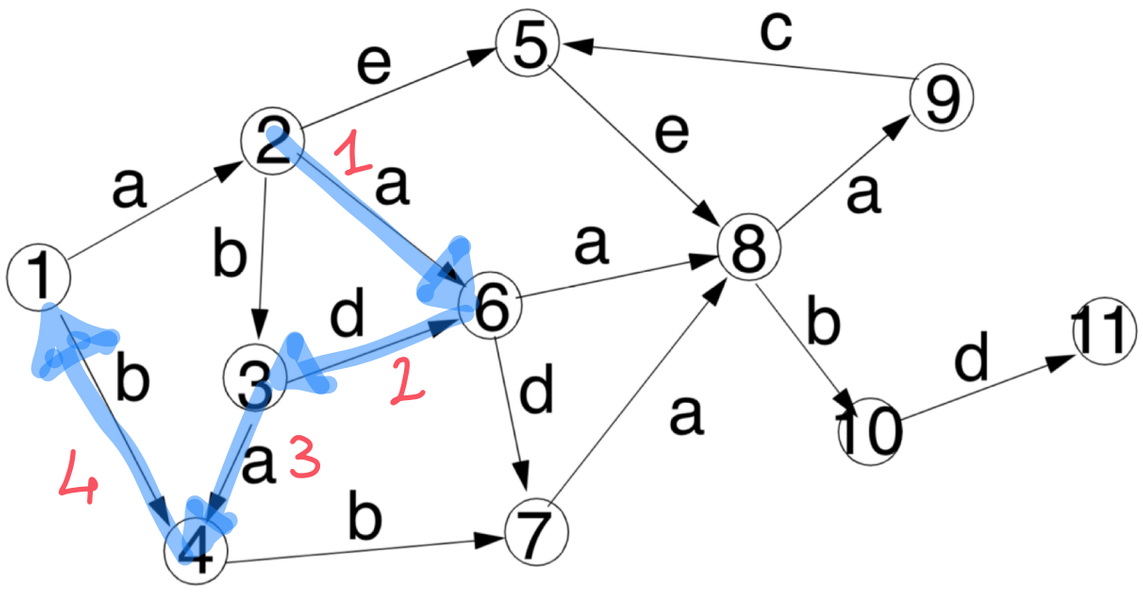
$R = a^{+} d^{-} a \; b^{-}$

a^{+} → one or more a

d^{-} → reverse d

a → one a

b^{-} → reverse b



Example 2

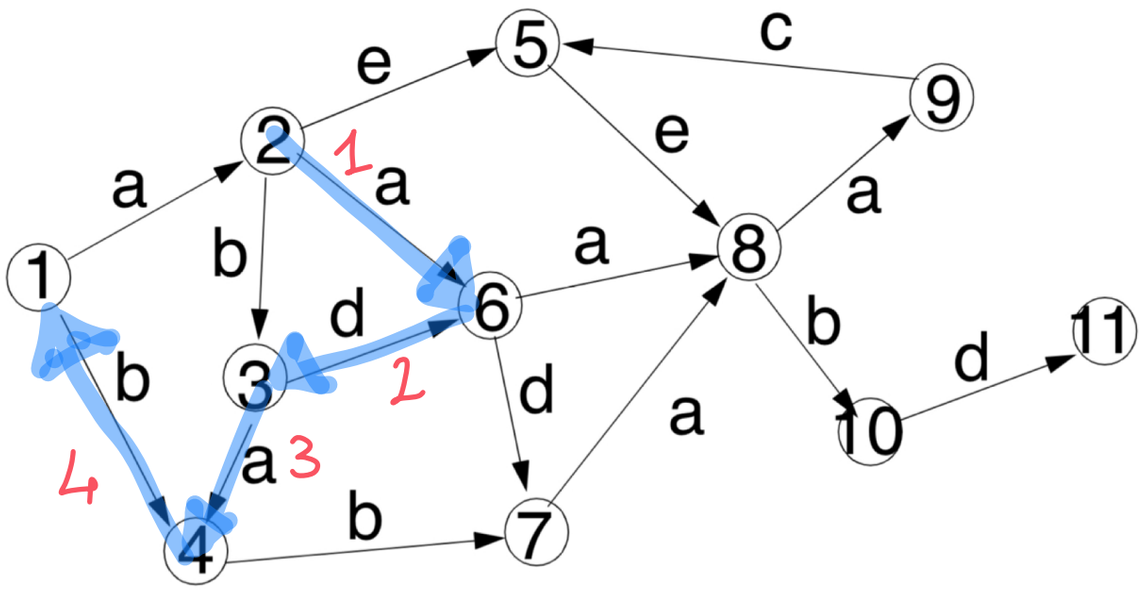
$R = a^{+} d^{-} a \; b^{-}$

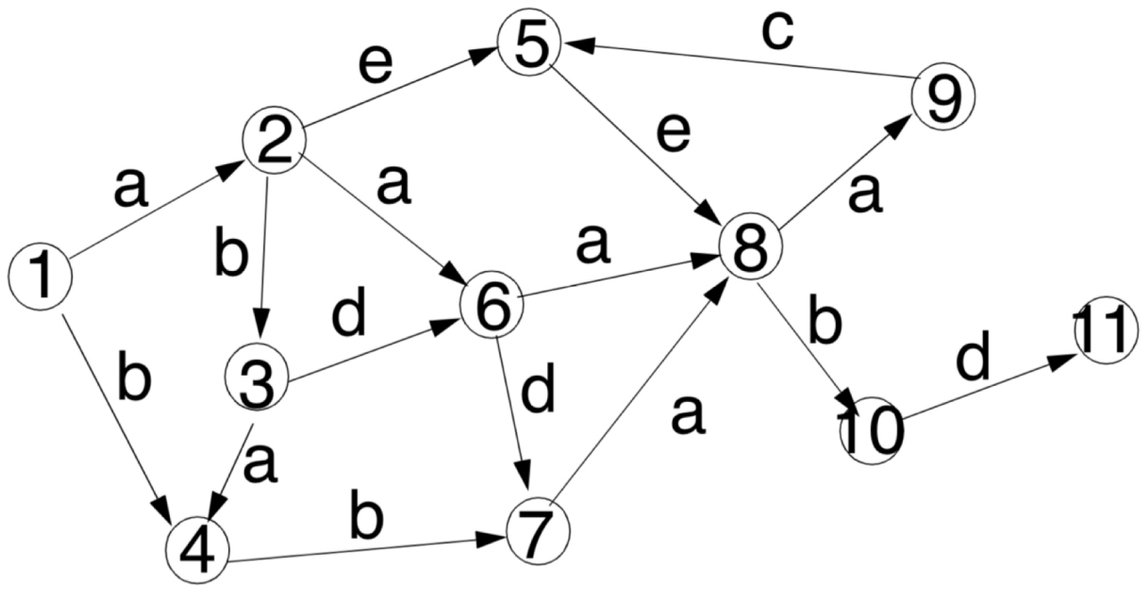
a^{+} → one or more a

d^{-} → reverse d

a → one a

b^{-} → reverse b





Example 2

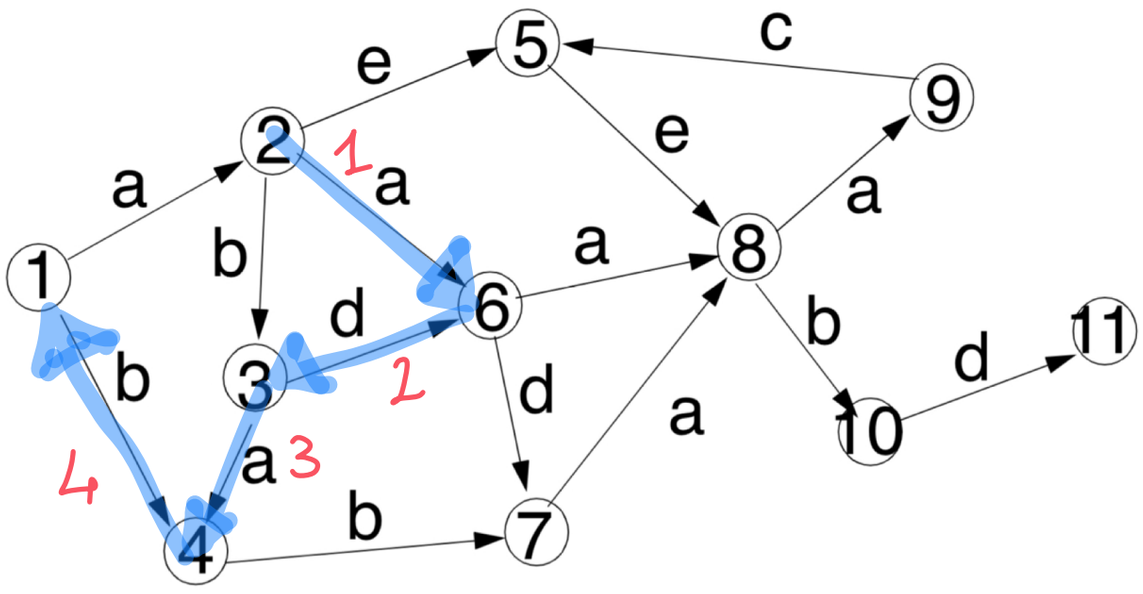
$R = a^{+} d^{-} a \; b^{-}$

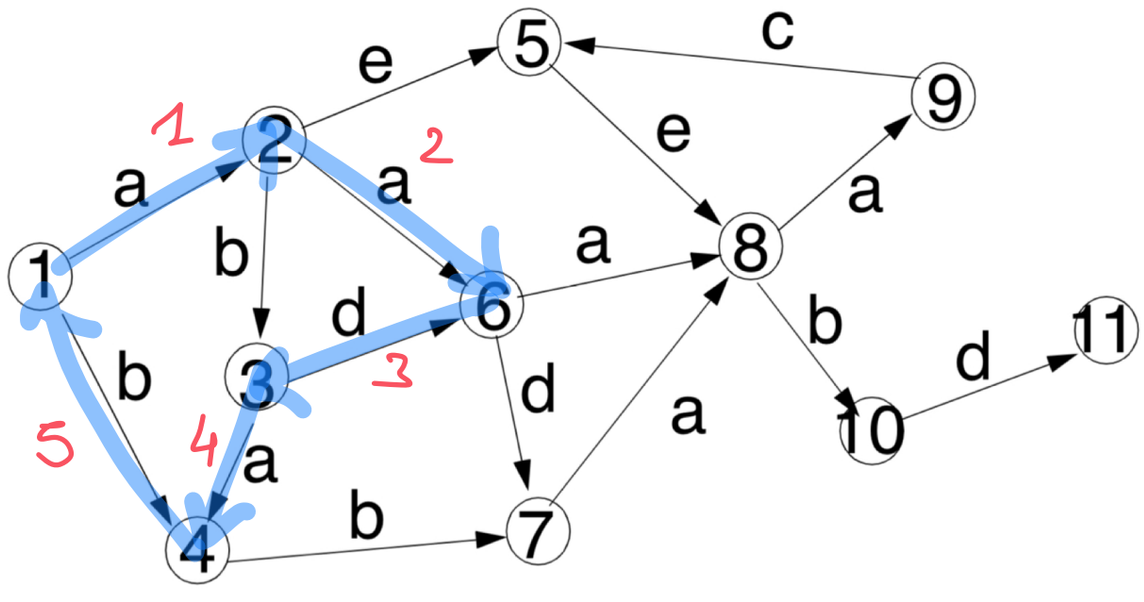
a^{+} → one or more a

d^{-} → reverse d

a → one a

b^{-} → reverse b





Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Let $G = (V, E, \eta, \lambda, \nu)$ be a PG, $q \in RPQ$ over G is the set of vertex pairs $[\![q]\!]_G \subseteq V \times V$

Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Let $G = (V, E, \eta, \lambda, \nu)$ be a PG, $g \in RPQ$ over G is the set of vertex pairs $[\![g]\!]_G \subseteq V \times V$

set of vertex pairs over G matching the RPQ g

Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Let $G = (V, E, \eta, \lambda, \nu)$ be a PG, $g \in RPQ$ over G is the set of vertex pairs $[\![g]\!]_G \subseteq V \times V$

set of vertex pairs over G matching the RPQ g

1) IF $g = a \in \Sigma$, then $[\![g]\!]_G = \{(s,t) \mid \exists e \in E \text{ s.t. } \eta(e) = (s,t) \wedge a \in \lambda(e)\}$

Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Let $G = (V, E, \eta, \lambda, \nu)$ be a PG, $g \in RPQ$ over G is the set of vertex pairs $[\![g]\!]_G \subseteq V \times V$

$[\![g]\!]_G$: set of vertex pairs over G matching the RPQ g

If $g = a \in \mathcal{L}$, then $[\![g]\!]_G = \{(s,t) \mid \exists e \in E \text{ s.t. } \eta(e) = (s,t) \wedge a \in \lambda(e)\}$

$\lambda(e)$ → return the label of e

Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Let $G = (V, E, \eta, \lambda, \nu)$ be a PG, $g \in RPQ$ over G is the set of vertex pairs $[\![g]\!]_G \subseteq V \times V$

$[\![g]\!]_G$: set of vertex pairs over G matching the RPQ g

If $g = a \in \mathcal{L}$, then $[\![g]\!]_G = \{(s,t) \mid \exists e \in E \text{ s.t. } \eta(e) = (s,t) \wedge a \in \lambda(e)\}$

$\lambda(e)$ → return the label of e

Given an RPQ a we get the vertex pair (s,t) connected by an edge "e" with label $\lambda(e) = a$

Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Let $G = (V, E, \eta, \lambda, \nu)$ be a PG, $g \in RPQ$ over G is the set of vertex pairs $[\![g]\!]_G \subseteq V \times V$

set of vertex pairs over G matching the RPQ g

1) If $g = a \in \mathcal{L}$, then $[\![g]\!]_G = \{(s,t) \mid \exists e \in E \text{ s.t. } \eta(e) = (s,t) \wedge a \in \lambda(e)\}$

$\lambda(e)$ → return the label of e

Given an RPQ a we get the vertex pair (s,t) connected by an edge "e" with label $\lambda(e) = a$

2) if $g = (e)^- \in RPQ$ then $[\![g]\!]_G = \{(t,s) \mid (s,t) \in [\![e]\!]_G\}$

Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Let $G = (V, E, \eta, \lambda, \nu)$ be a PG, $g \in RPQ$ over G is the set of vertex pairs $[\![g]\!]_G \subseteq V \times V$

set of vertex pairs over G matching the RPQ g

1) If $g = a \in \mathcal{L}$, then $[\![g]\!]_G = \{(s,t) \mid \exists e \in E \text{ s.t. } \eta(e) = (s,t) \wedge a \in \lambda(e)\}$

$\lambda(e)$ → return the label of e

Given an RPQ a we get the vertex pair (s,t) connected by an edge "e" with label $\lambda(e) = a$

2) if $g = (e)^- \in RPQ$ then $[\![g]\!]_G = \{(t,s) \mid (s,t) \in [\![e]\!]_G\}$

There is an edge $s \xrightarrow{e} t$ and we return (t,s) i.e. the inverse

Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Let $G = (V, E, \eta, \lambda, \nu)$ be a PG, $g \in RPQ$ over G is the set of vertex pairs $[\![g]\!]_G \subseteq V \times V$

set of vertex pairs over G matching the RPQ g

1) If $g = a \in \mathcal{L}$, then $[\![g]\!]_G = \{(s,t) \mid \exists e \in E \text{ s.t. } \eta(e) = (s,t) \wedge a \in \lambda(e)\}$

$\lambda(e) \rightarrow$ return the label of e

Given an RPQ a we get the vertex pair (s,t) connected by an edge "e" with label $\lambda(e) = a$

2) if $g = (e)^- \in RPQ$ then $[\![g]\!]_G = \{(t,s) \mid (s,t) \in [\![e]\!]_G\}$

There is an edge $s \xrightarrow{e} t$ and we return (t,s) i.e. the inverse

3) if $g = e/f \in RPQ$ then $[\![g]\!]_G = \{(s,t) \mid \exists u \in V \text{ s.t. } (s,u) \in [\![e]\!]_G \wedge (u,t) \in [\![f]\!]_G\}$

Semantics

A pair of vertices is a valid answer to an RPQ if and only if the respective vertices are connected in the data graph by a path conforming to the RPQ

Let $G = (V, E, \eta, \lambda, \nu)$ be a PG, $g \in RPQ$ over G is the set of vertex pairs $[\![g]\!]_G \subseteq V \times V$

set of vertex pairs over G matching the RPQ g

1) If $g = a \in \mathcal{L}$, then $[\![g]\!]_G = \{(s,t) \mid \exists e \in E \text{ s.t. } \eta(e) = (s,t) \wedge a \in \lambda(e)\}$

$\lambda(e) \rightarrow$ return the label of e

Given an RPQ a we get the vertex pair (s,t) connected by an edge "e" with label $\lambda(e) = a$

2) if $g = (e)^- \in RPQ$ then $[\![g]\!]_G = \{(t,s) \mid (s,t) \in [\![e]\!]_G\}$

There is an edge $s \xrightarrow{e} t$ and we return (t,s) i.e. the inverse

3) if $g = e/f \in RPQ$ then $[\![g]\!]_G = \{(s,t) \mid \exists u \in V \text{ s.t. } (s,u) \in [\![e]\!]_G \wedge (u,t) \in [\![f]\!]_G\}$

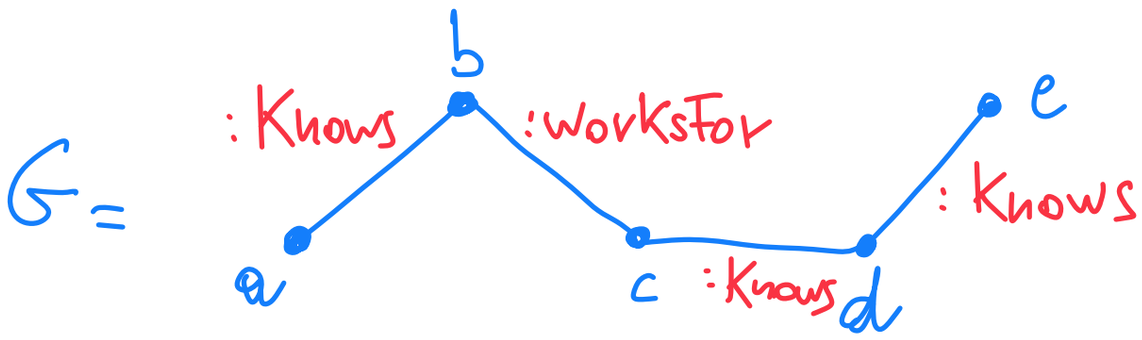
Semantics

4) if $g = e + f \in RPQ$, then $[\![e]\!]_G \cup [\![f]\!]_G$

5) if $g = (e)^+ \in RPQ$, then $[\![g]\!]_G = \{(s,t) \mid (s,t) \in TC([\![e]\!]_G)$ where TC is the transitive closure of binary relation R

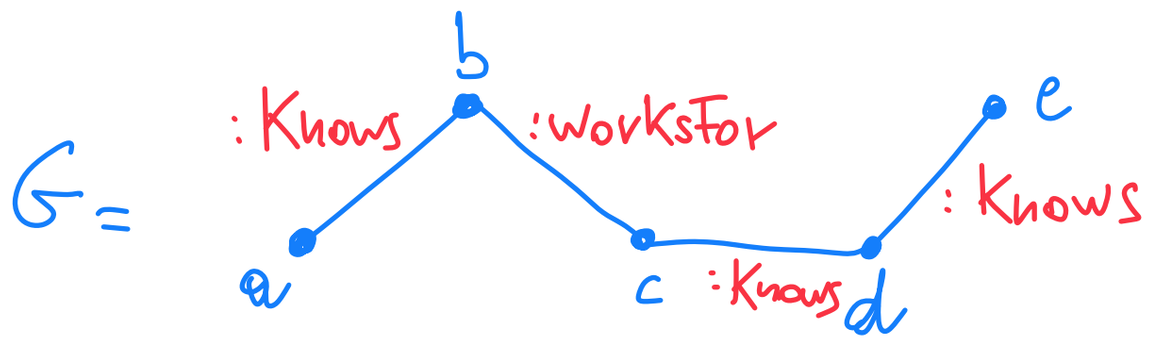
Example 1

$q = (:Knows)/(:worksFor)/(:Knows)^+$



Example 1

$q = (:Knows)/(:worksFor)/(:Knows)^+$



→ we have to find a path between two nodes (s and t) such that

$\exists e \mid \lambda(e) =$:Knows $\wedge$:worksFor $\wedge$:Knows

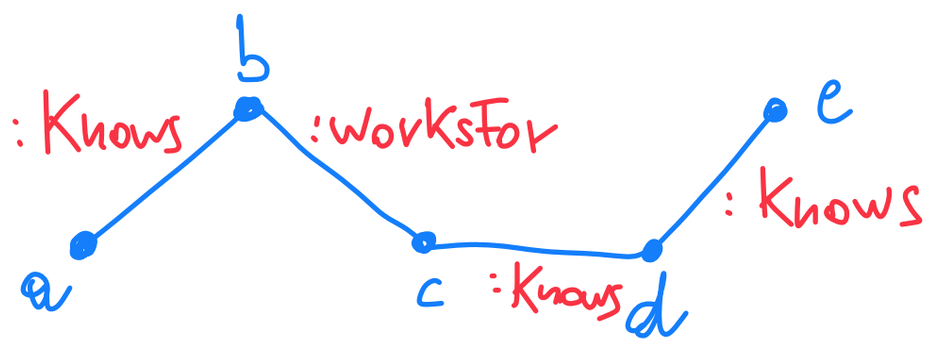
Example 1

$$q = (:Knows)/(:worksFor)/(:Knows)^+$$

→ we have to find a path between two nodes (s and t) such that

$\exists e \mid \lambda(e) =$:Knows $\wedge$:worksFor $\wedge$:Knows

$G =$

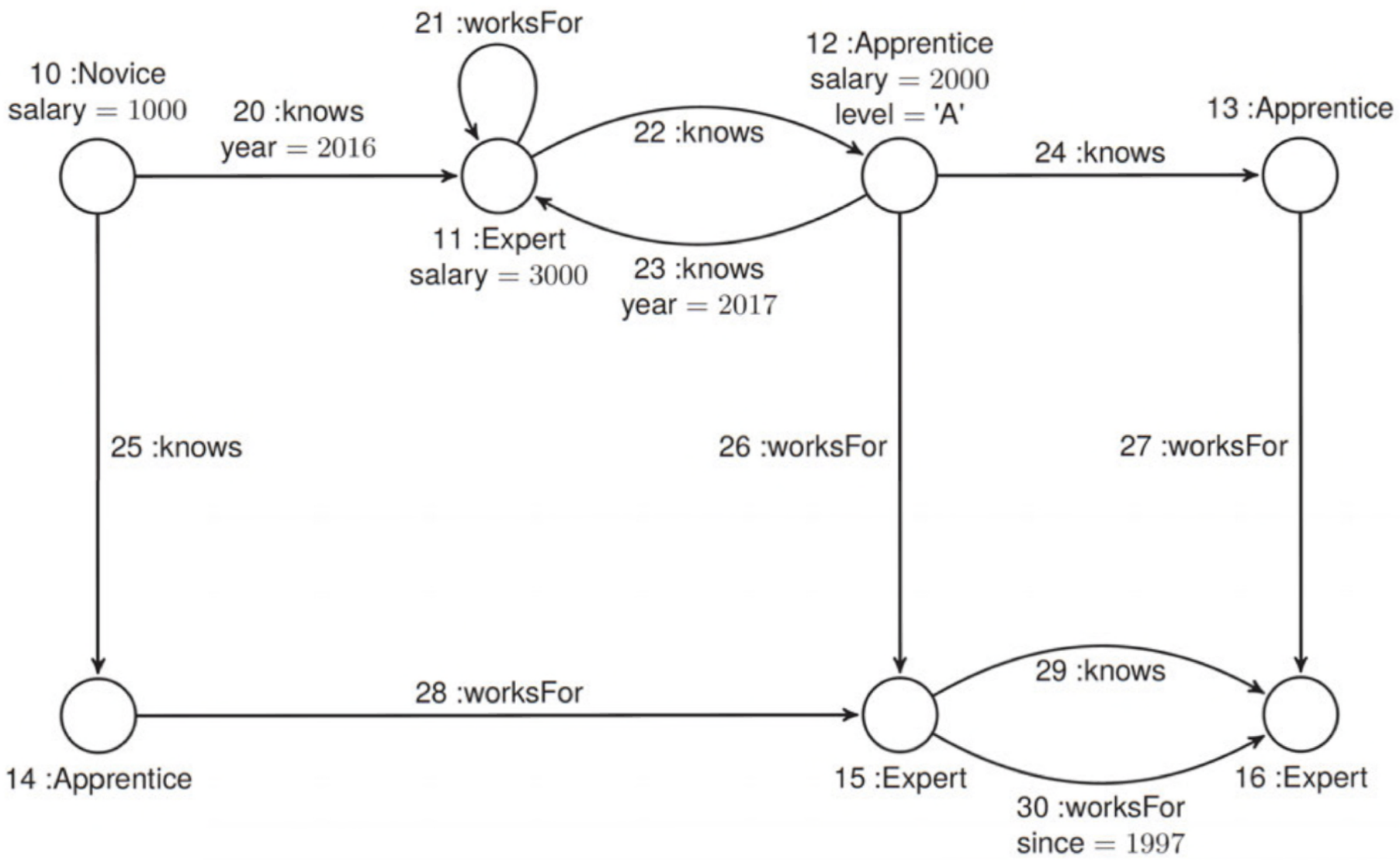


$$[\![q]\!]_G = \{(a,d), (a,e)\}$$

Example 2

Given the query $q = :knows / :worksFor / :knows^+$ evaluated on G_{ex} get $[\![q]\!]_{G_{ex}}$.

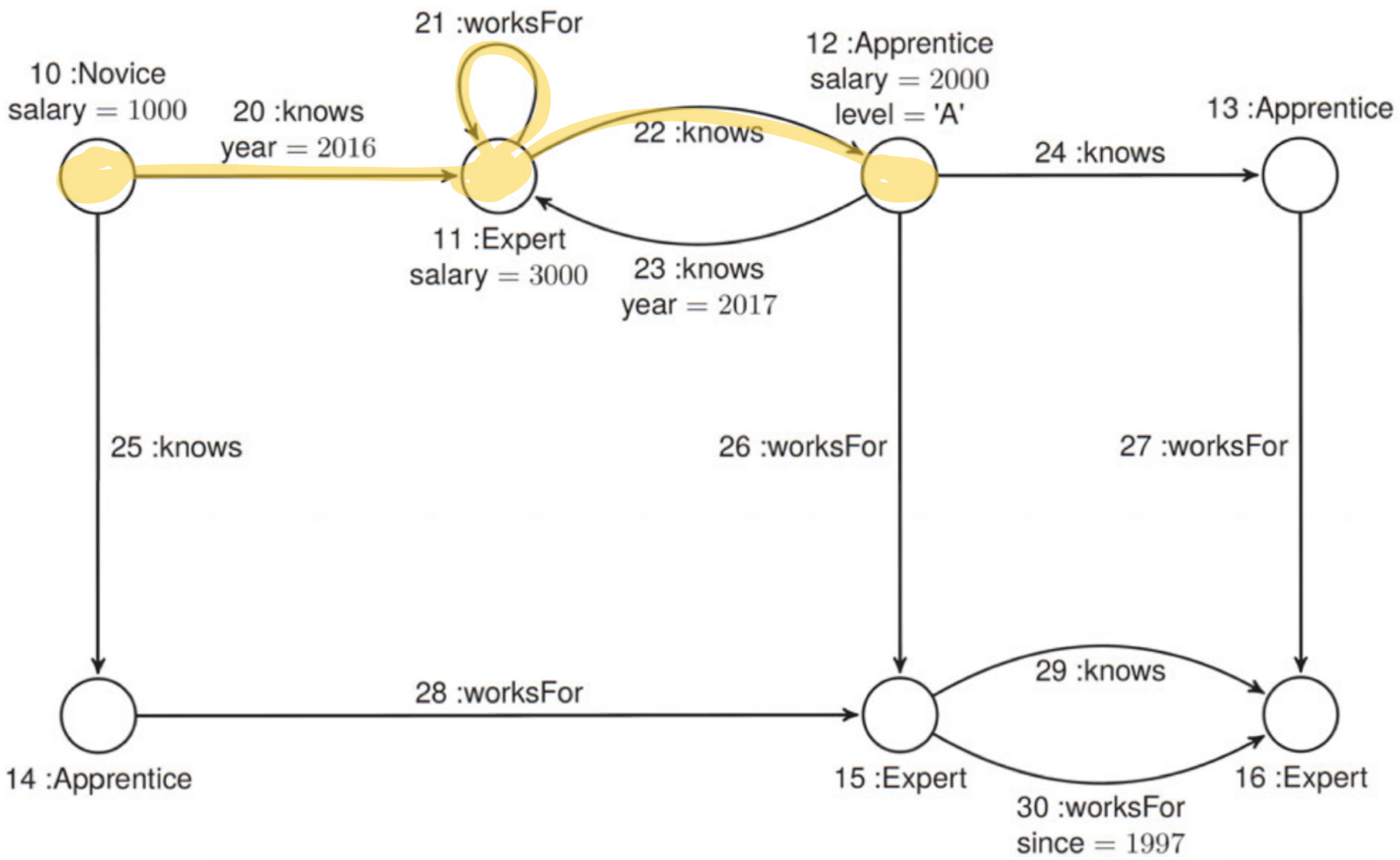
G_{ex}



Example 2

Given the query $q = :knows / :worksFor / :knows^+$ evaluated on G_{ex} get $[\![q]\!]_{G_{ex}}$.

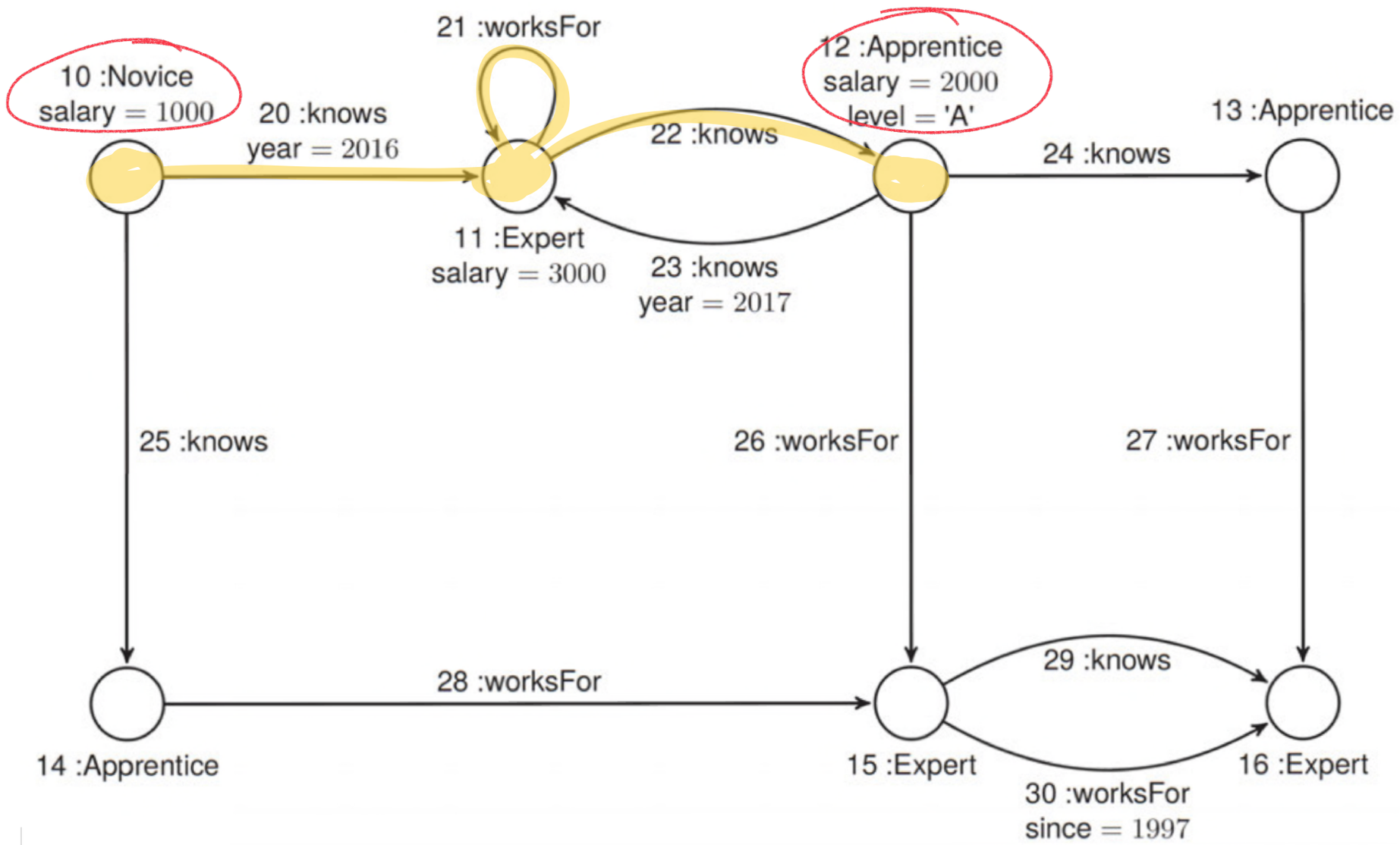
G_{ex}



Example 2

Given the query $q = :knows/:worksFor/:knows^+$ evaluated on G_{ex} get $[\![q]\!]_{G_{ex}}$.

G_{ex}

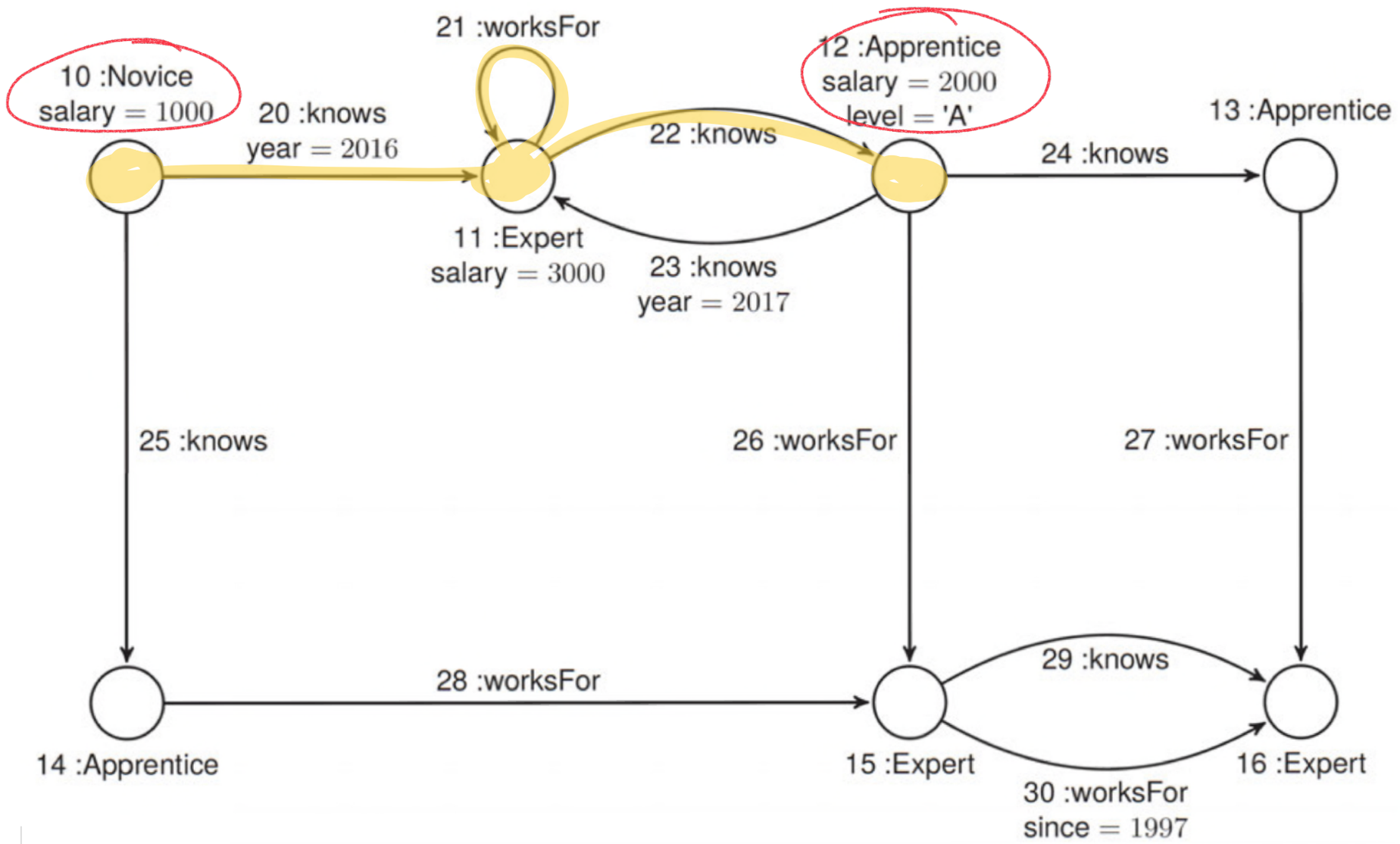


Example 2

Given the query $q = :knows / :worksFor / :knows^+$ evaluated on G_{ex} get $[\![q]\!]_{G_{ex}}$.

$[\![q]\!]_{G_{ex}} = \{(10, 12)\}$

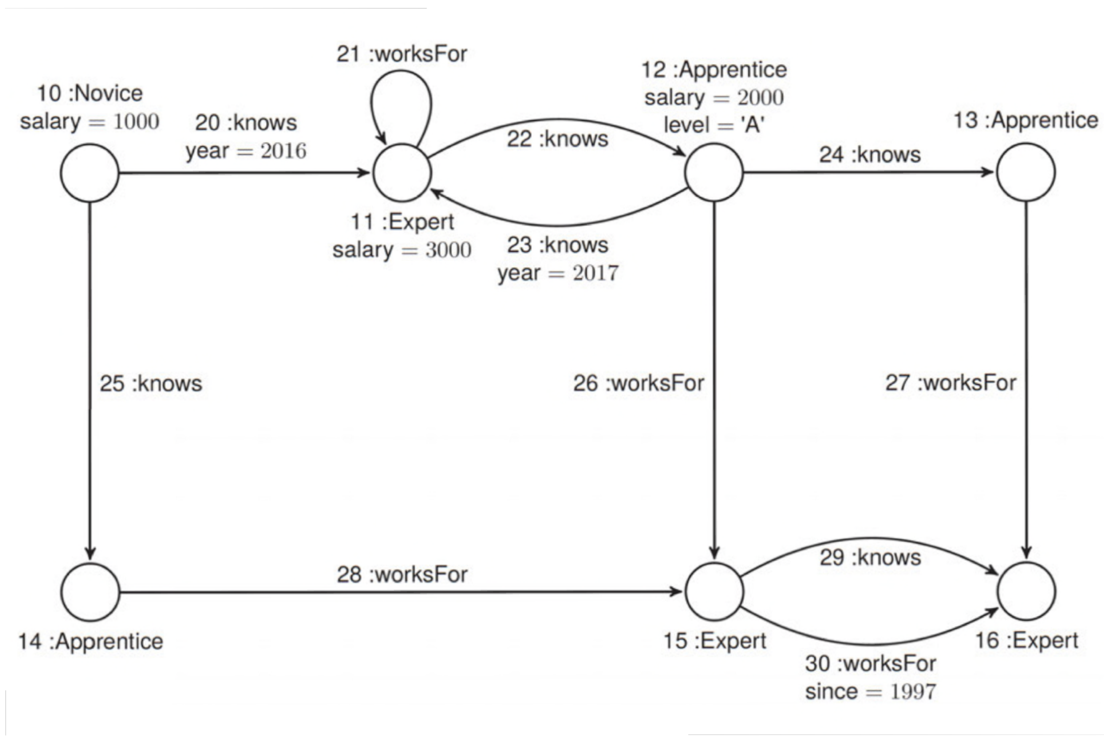
G_{ex}

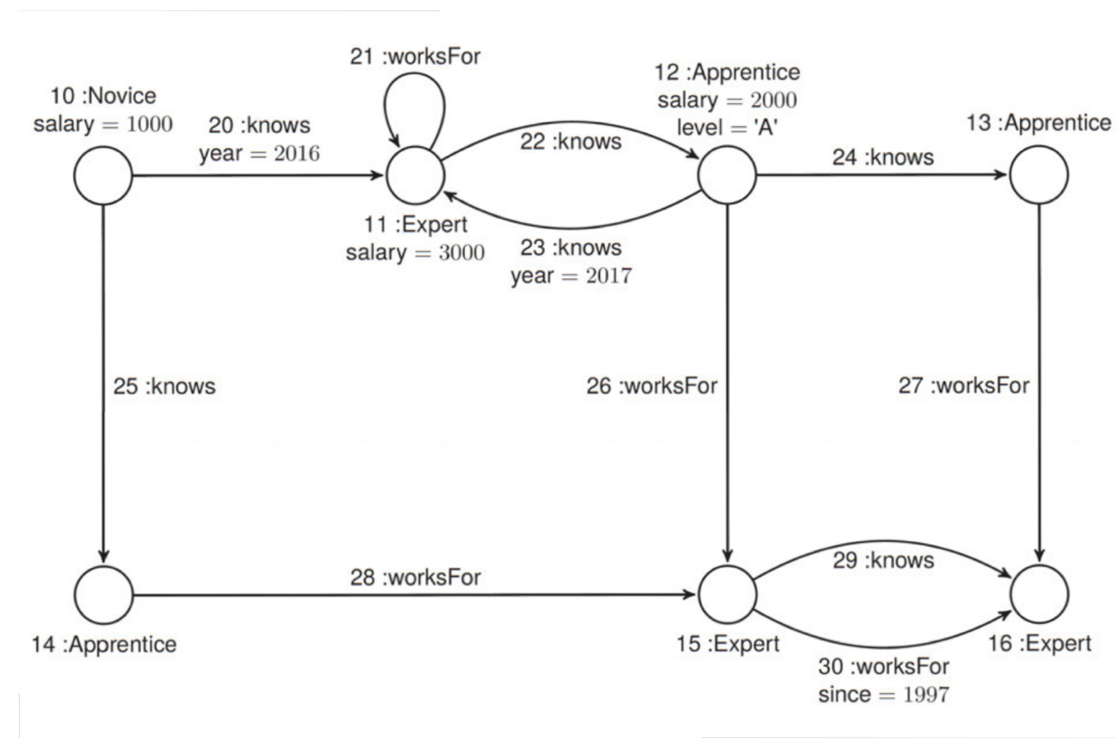


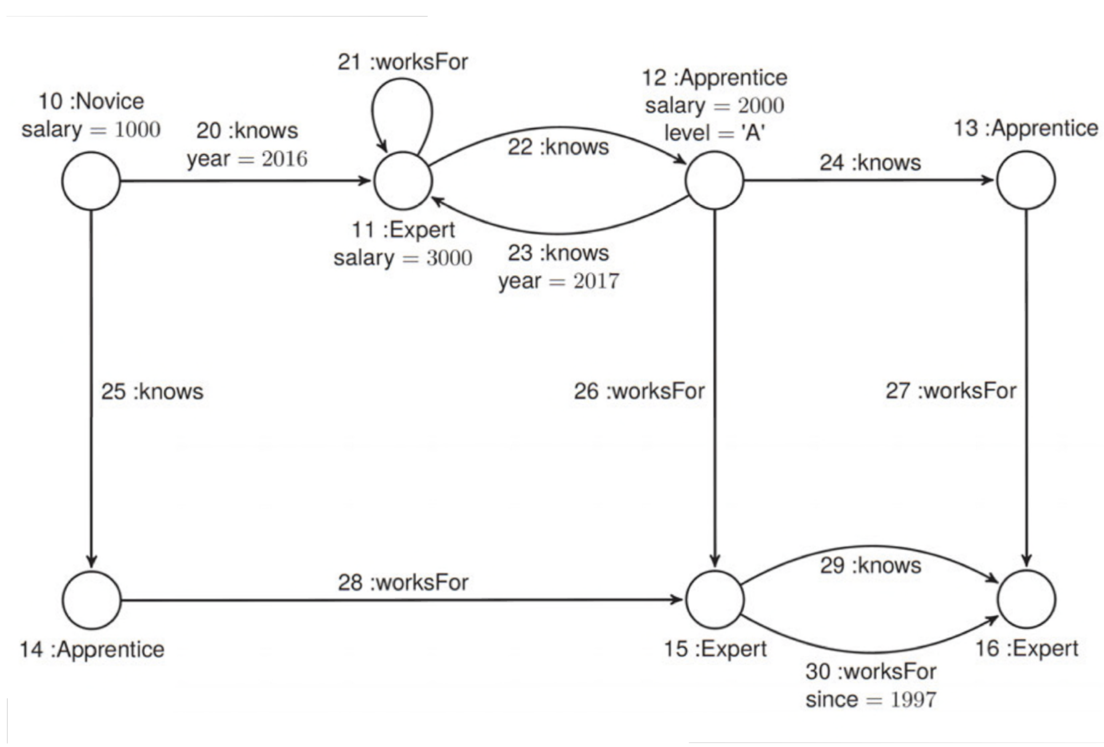
Example 2

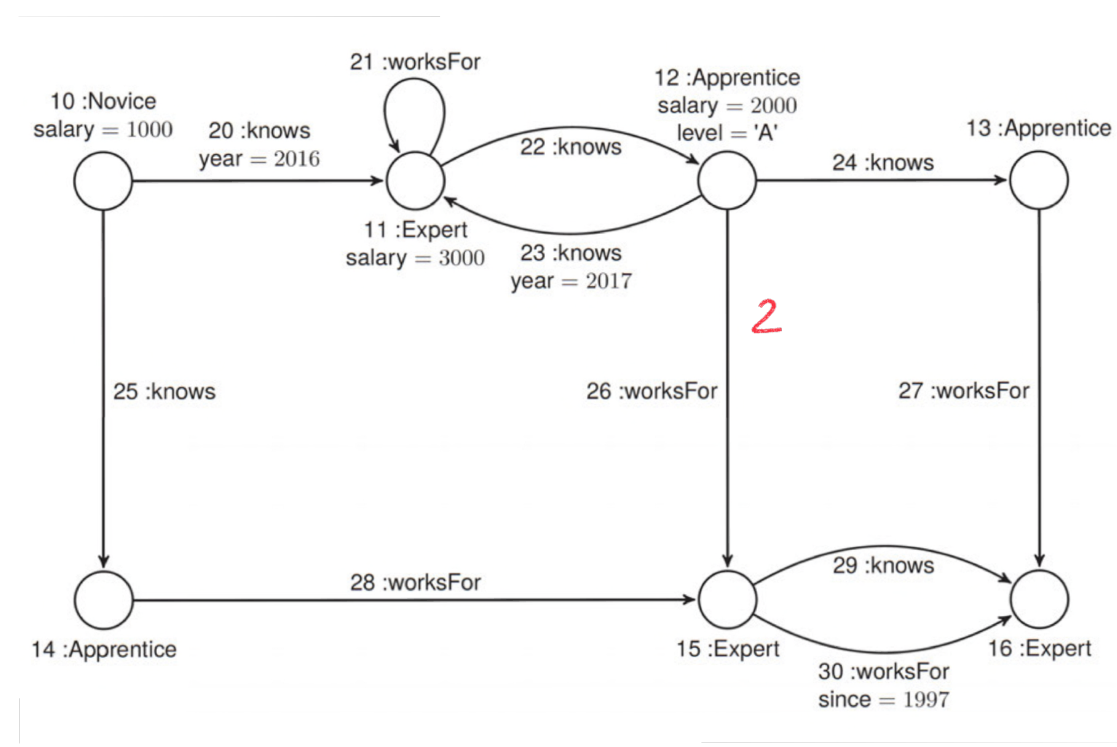
$q = :knows / :workFor / :knows^{+}$

$[\![q]\!]_{G_{ex}} = \{(10, 12)\}$





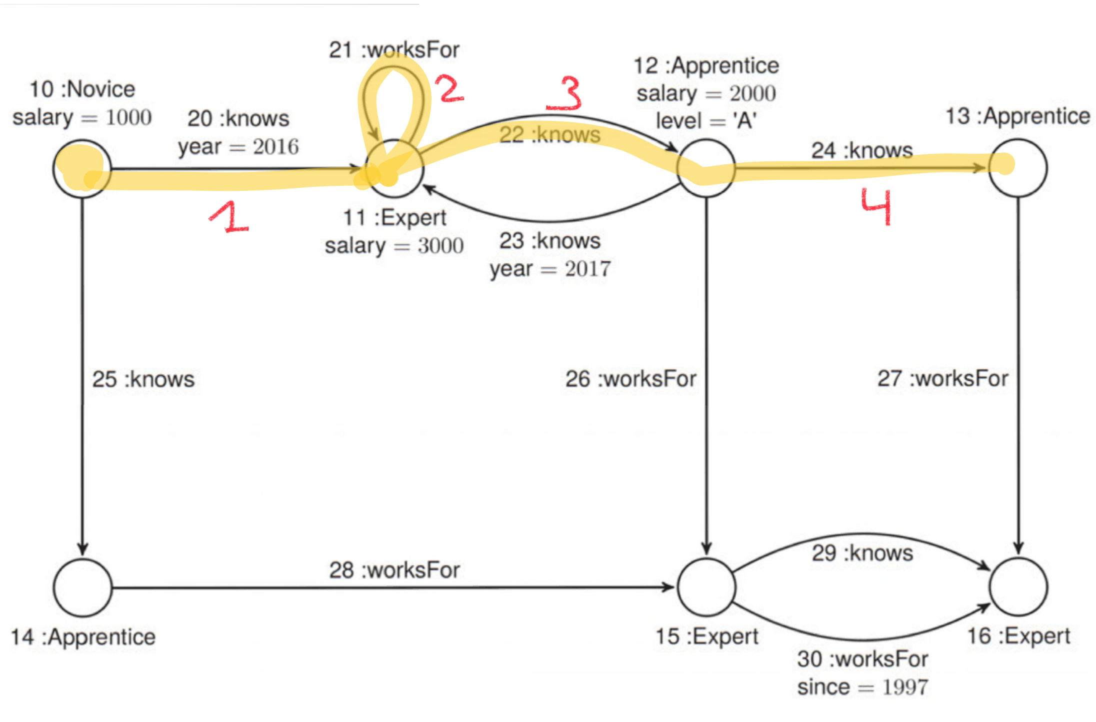


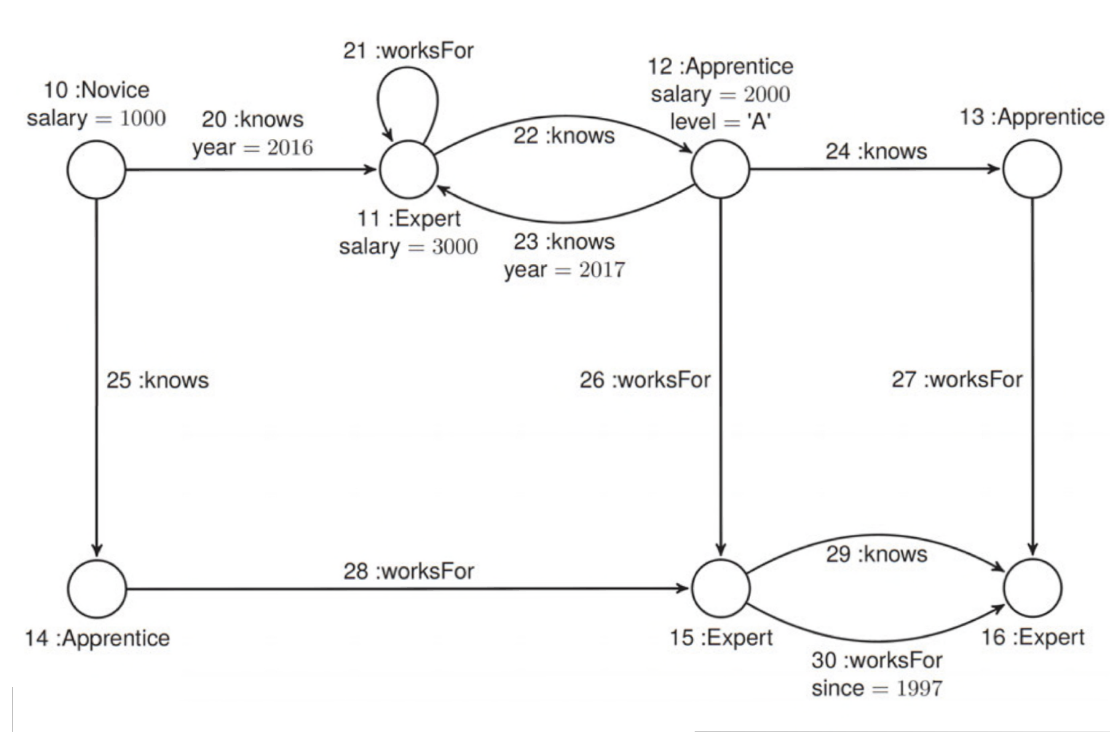


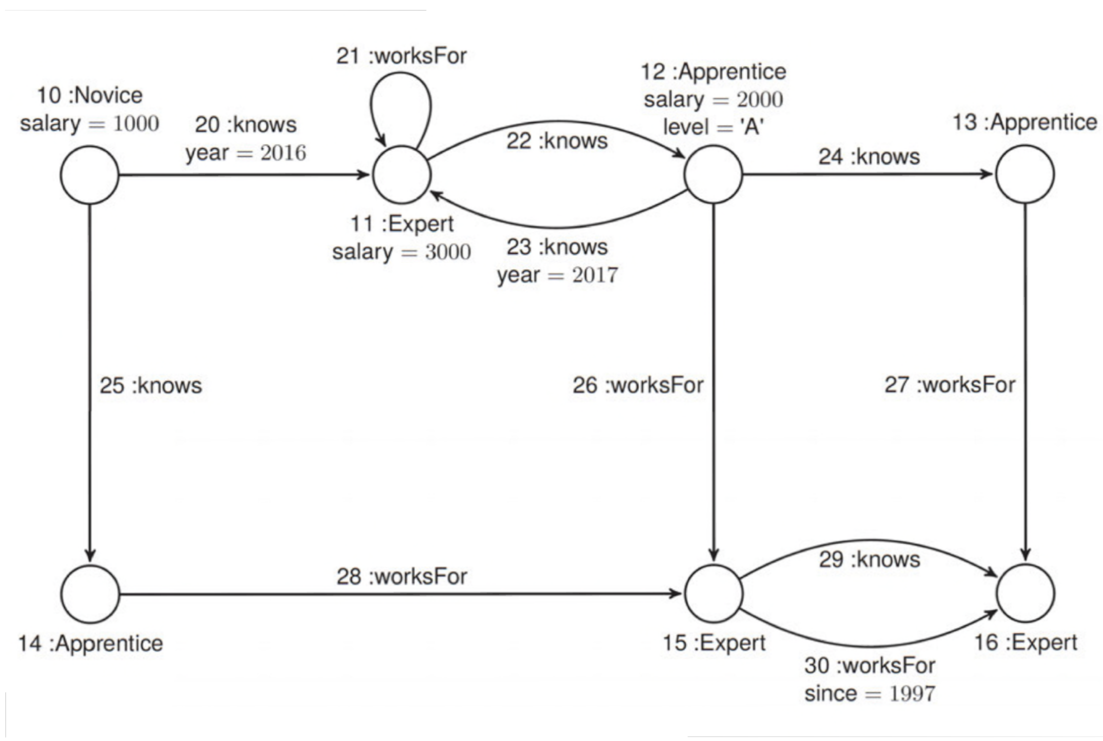
Example 2

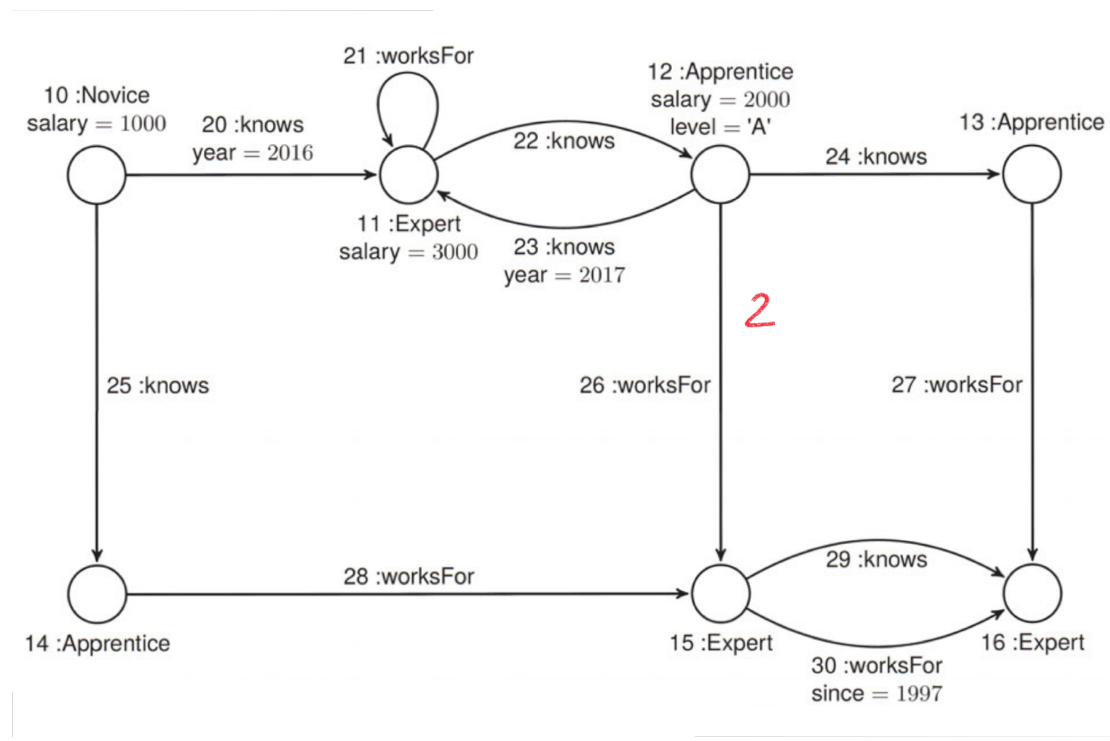
$q = :knows / :workFor / :knows^{+}$

$[\![q]\!]_{G_{ex}} = \{(10, 12)\}$





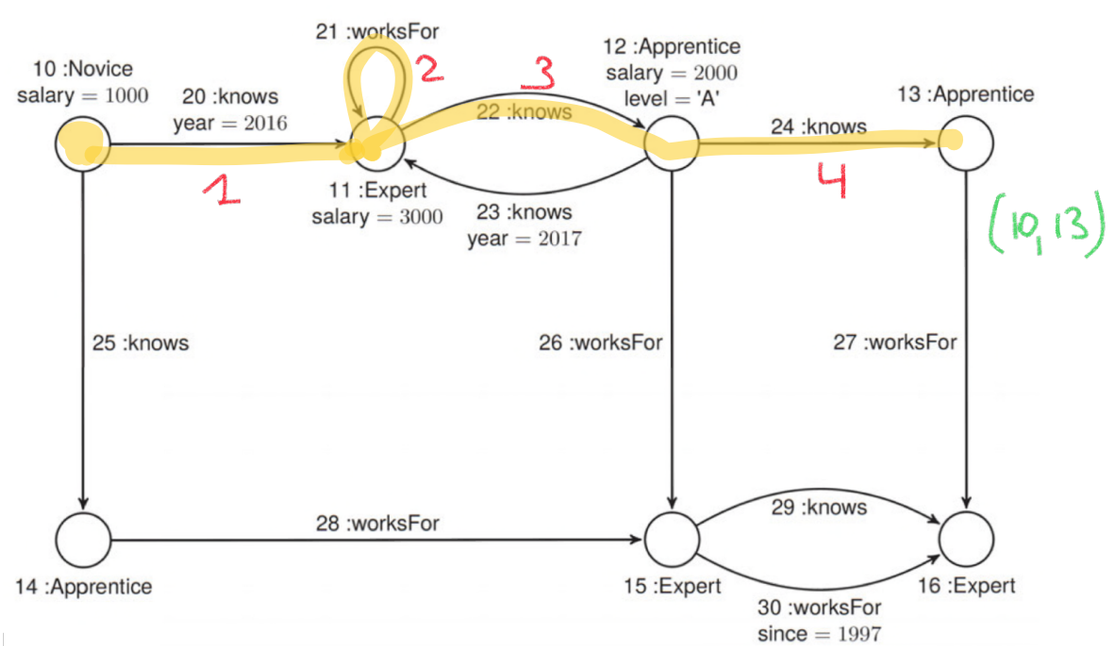


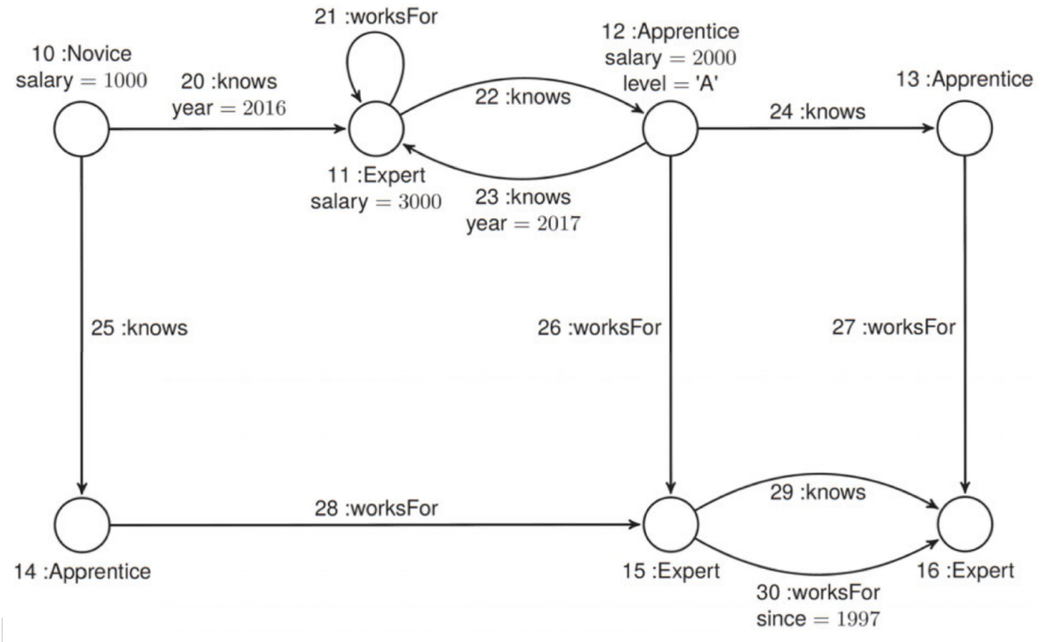


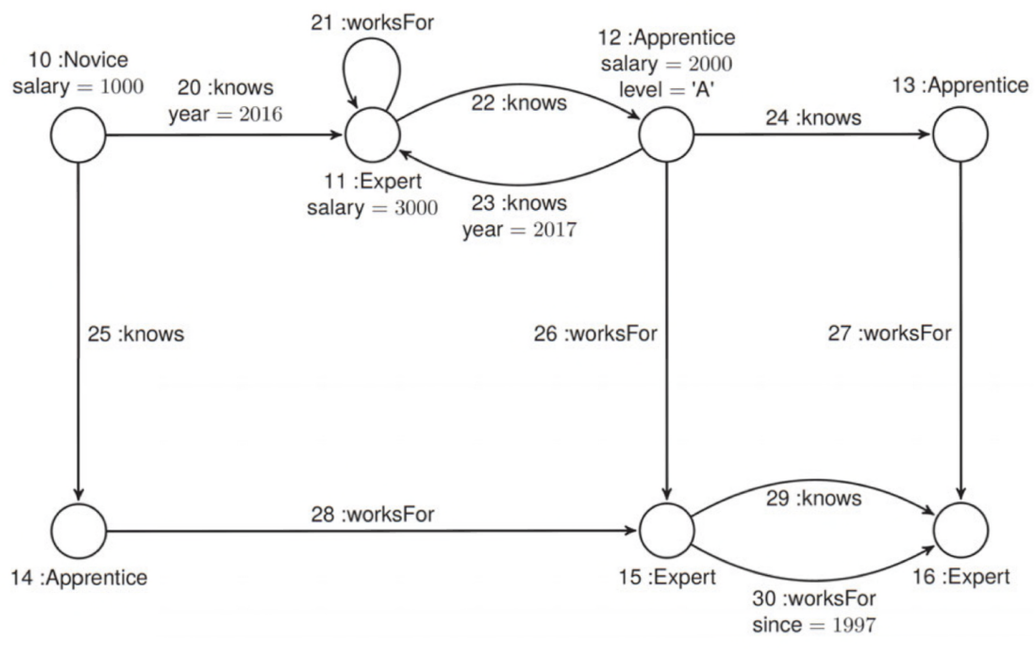
Example 2

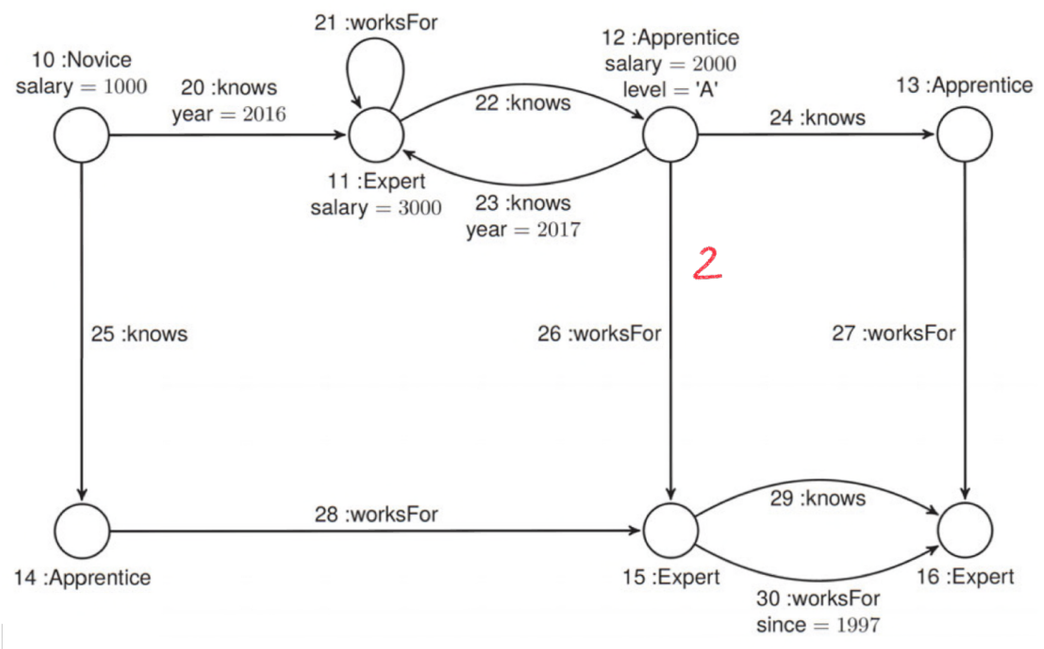
$q = :knows / :worksFor / :knows^+$

$[\![q]\!]_{G_{ex}} = \{(10,12)$





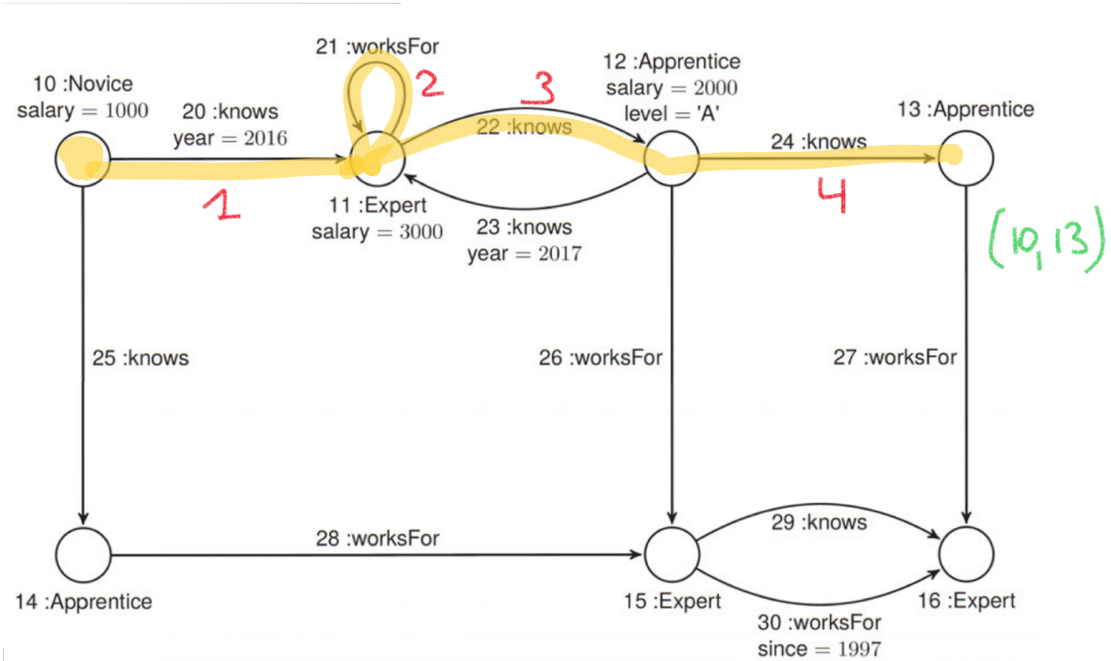


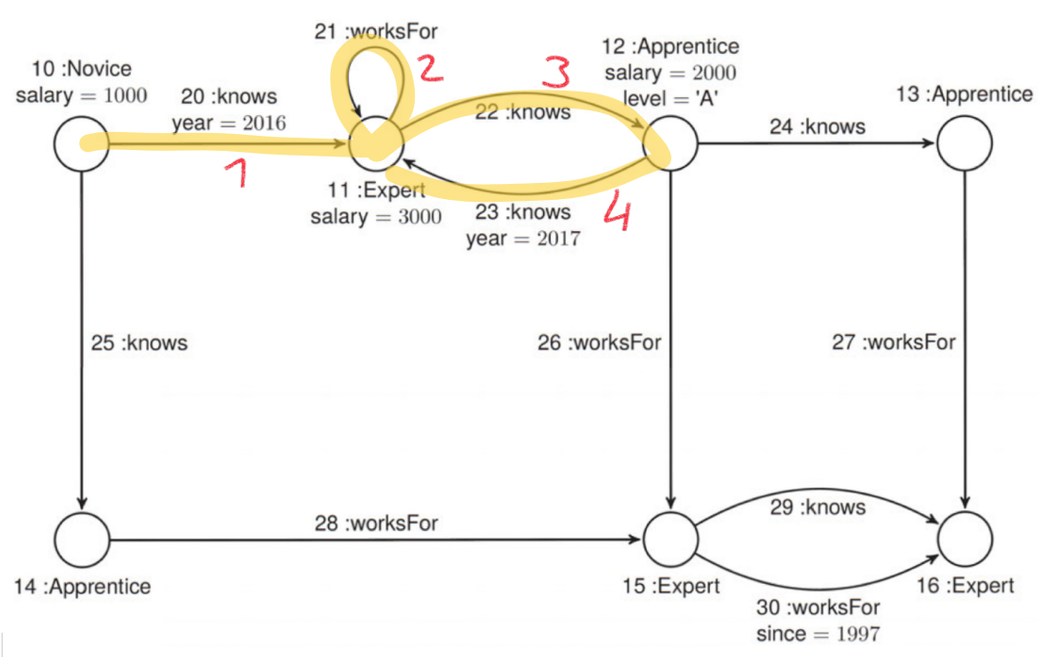


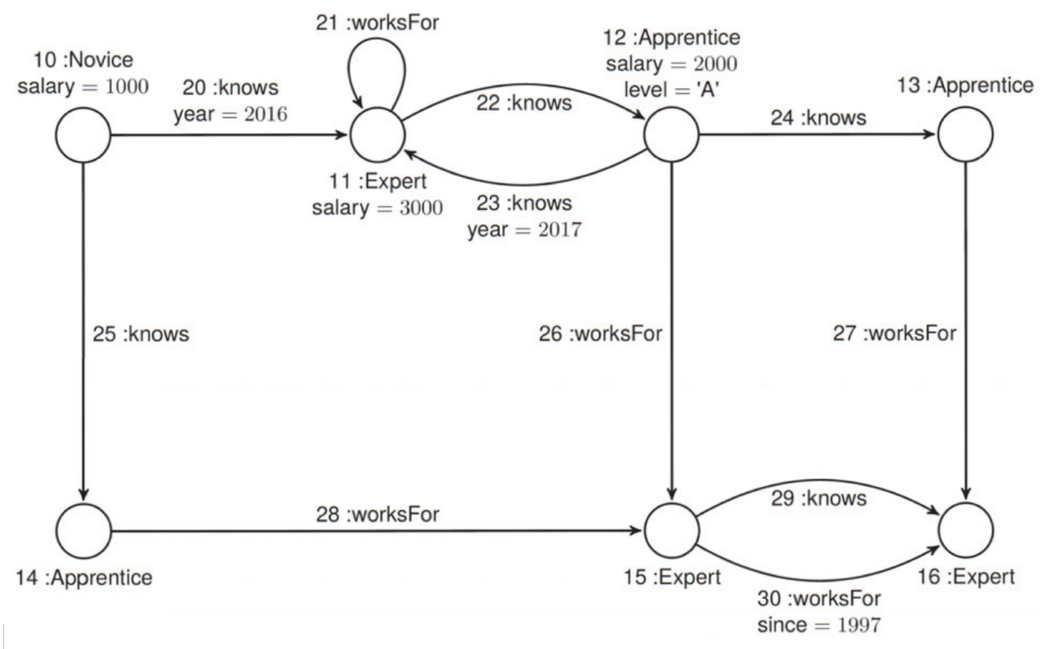
Example 2

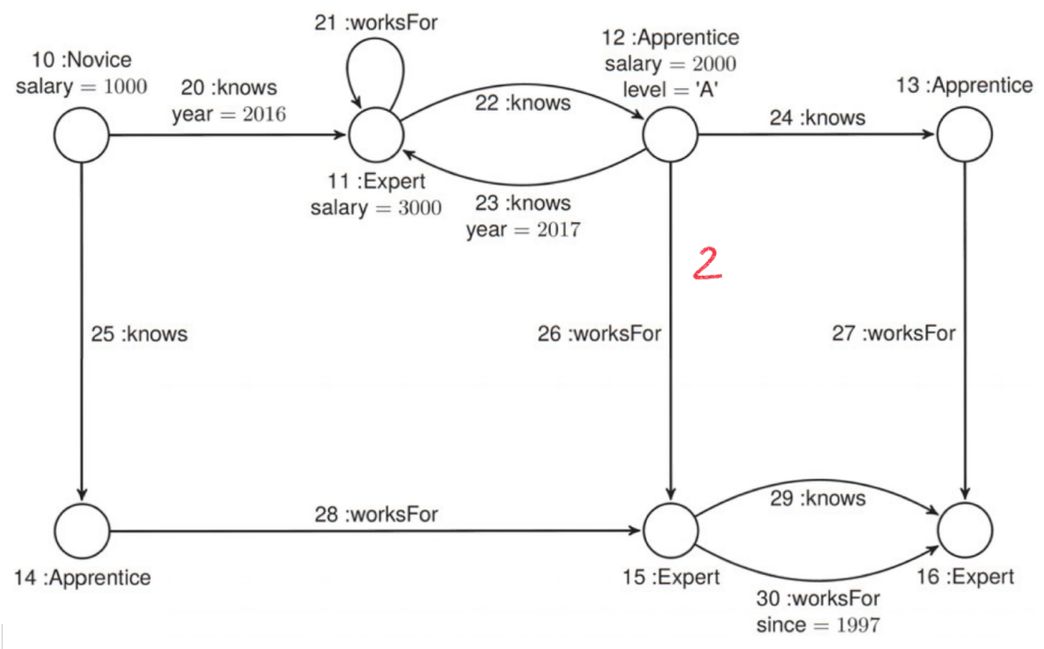
$q = :knows / :worksFor / :knows^+$

$[\![q]\!]_{G_{ex}} = \{(10,12)$





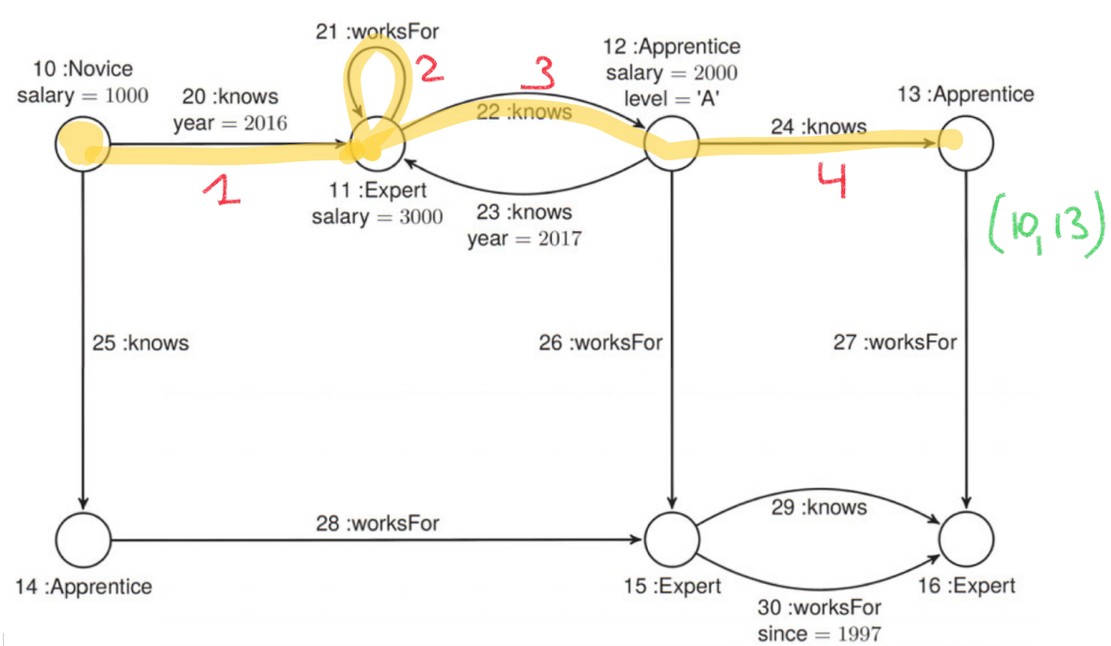


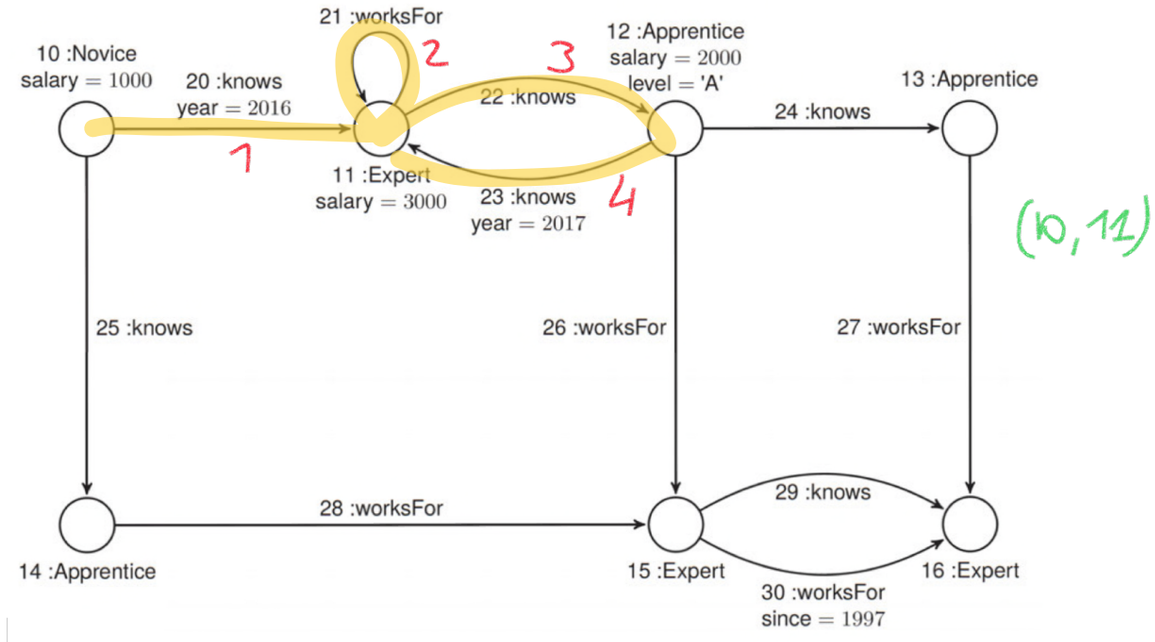


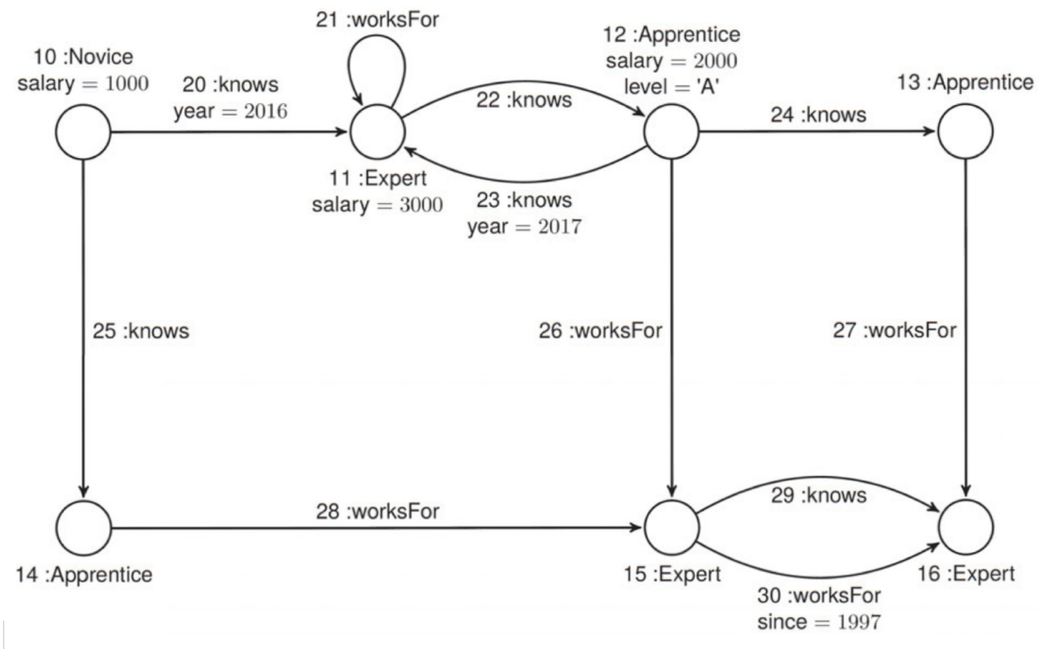
Example 2

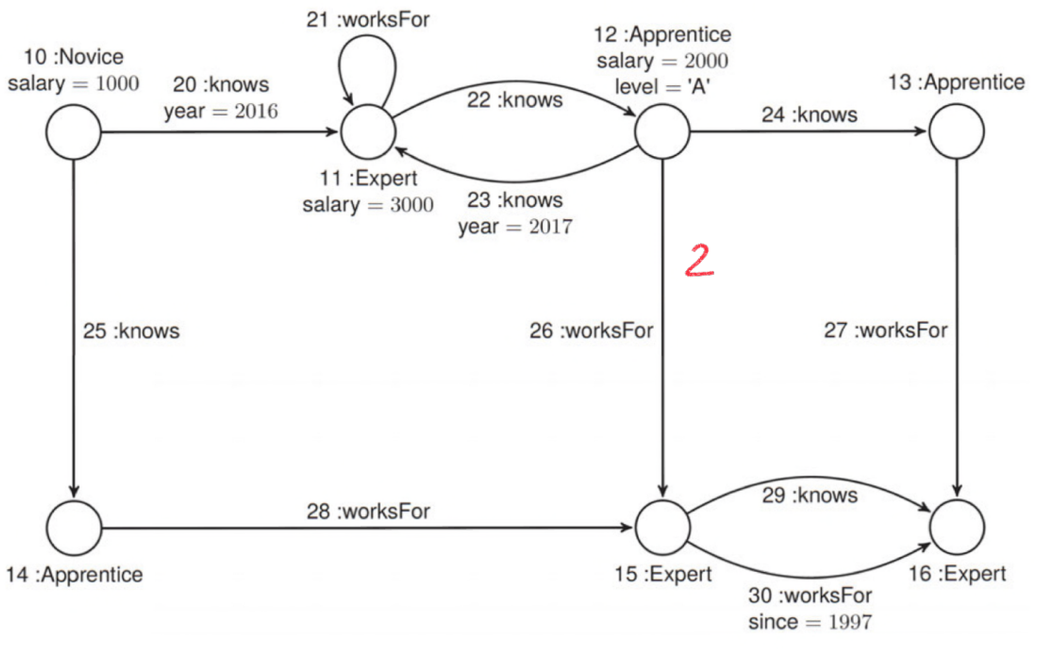
$q = :knows / :worksFor / :knows^+$

$[\![q]\!]_{G_{ex}} = \{(10,12)$





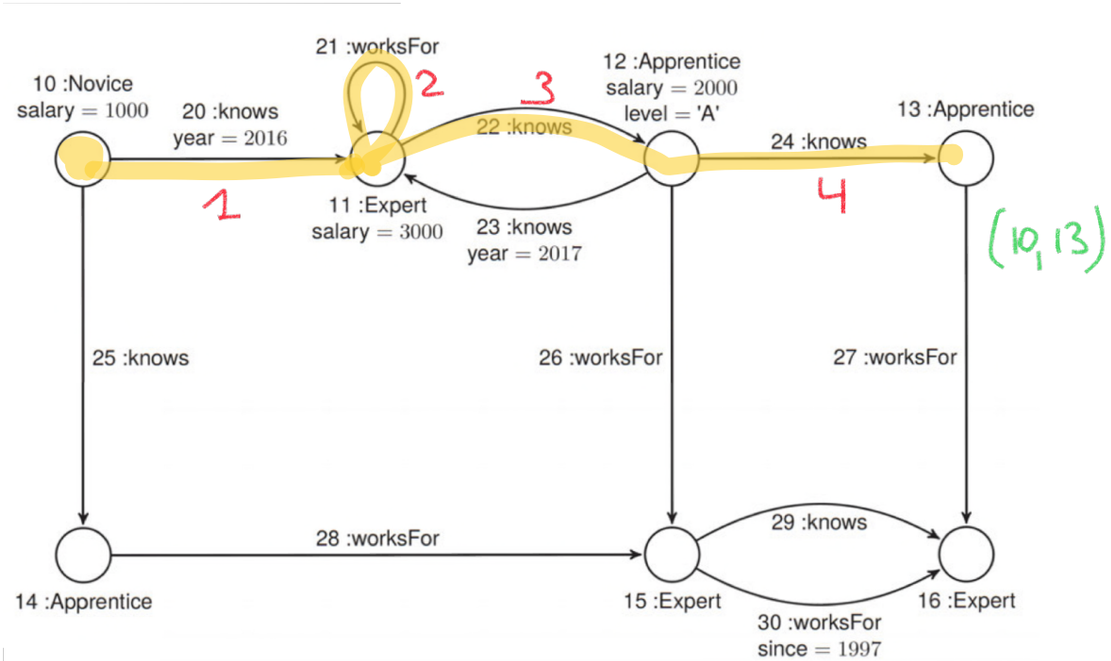


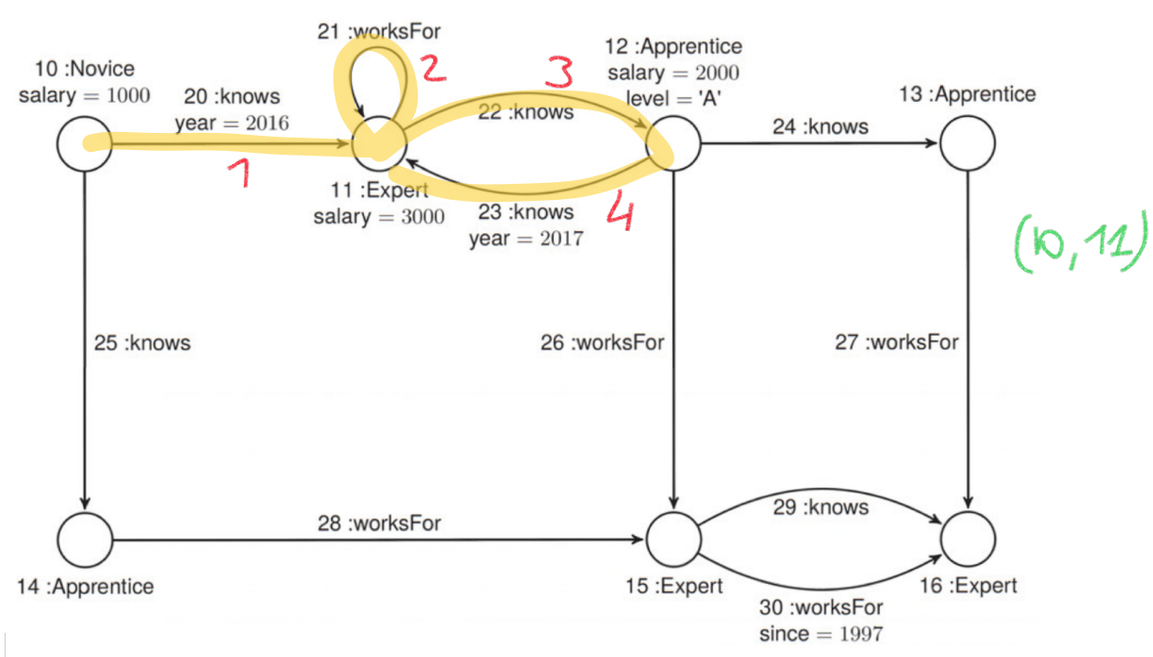


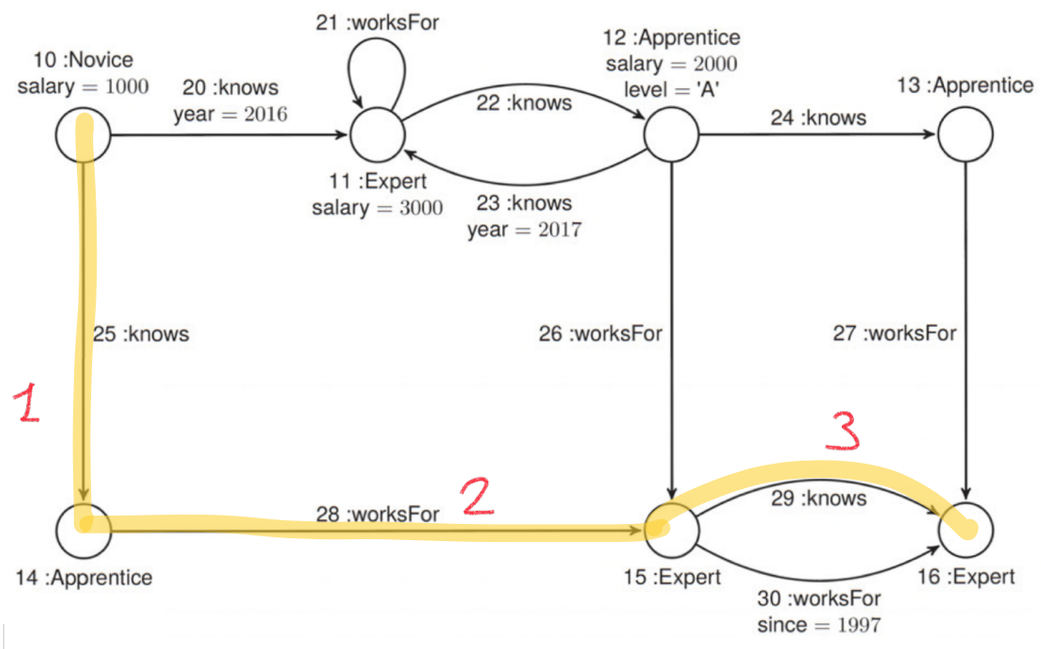
Example 2

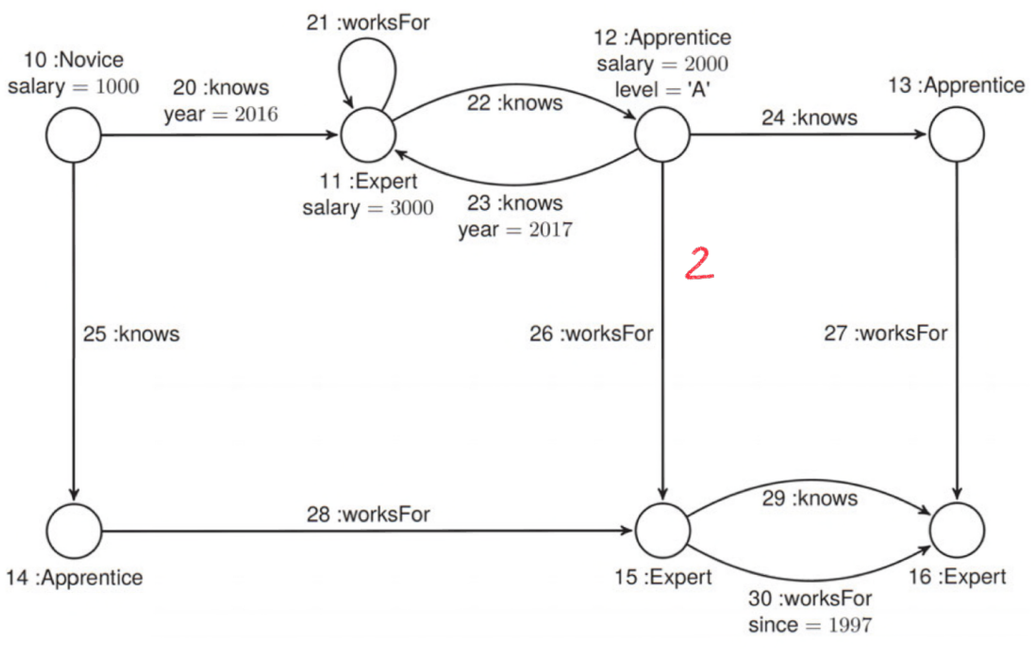
$q = :knows / :worksFor / :knows^+$

$[\![q]\!]_{G_{ex}} = \{(10,12)$





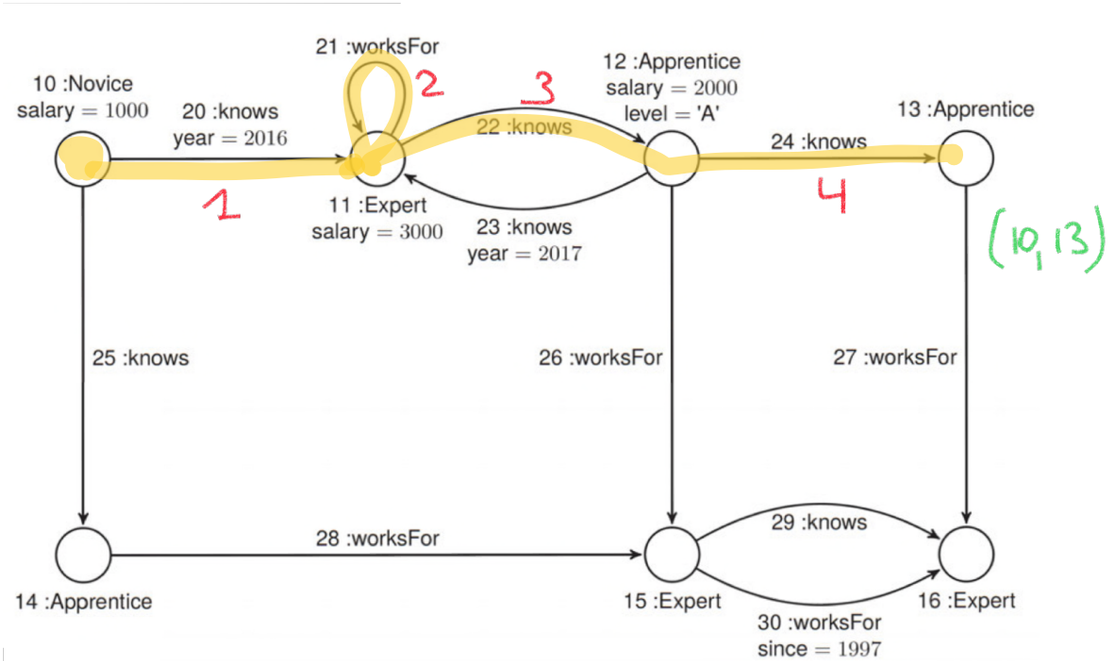


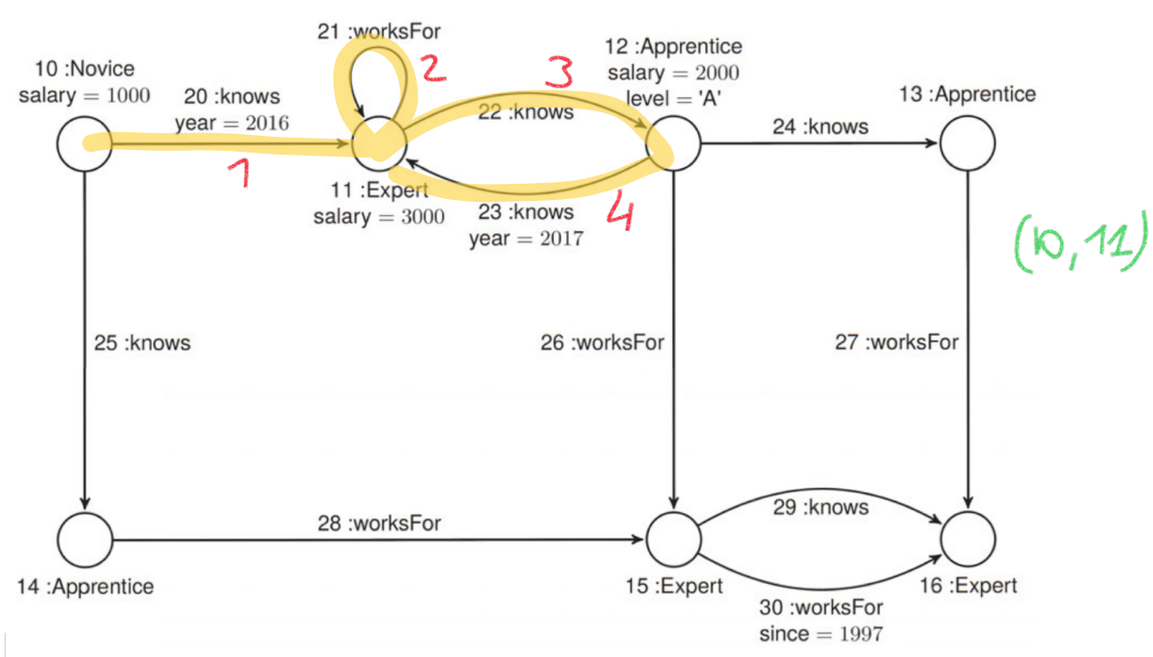


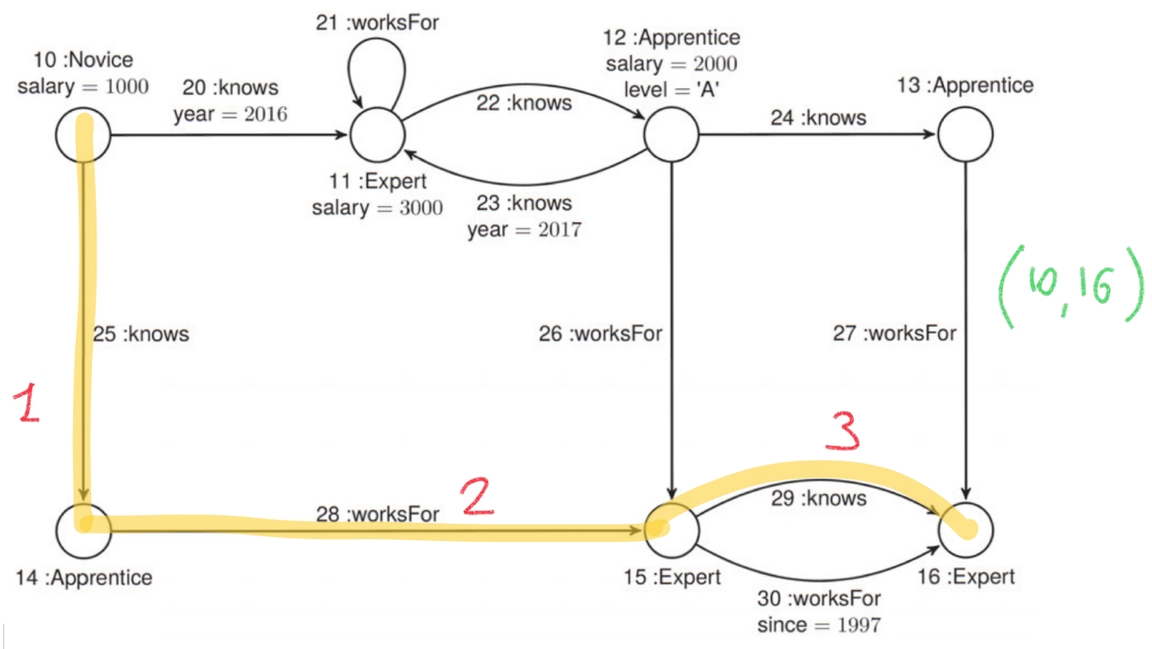
Example 2

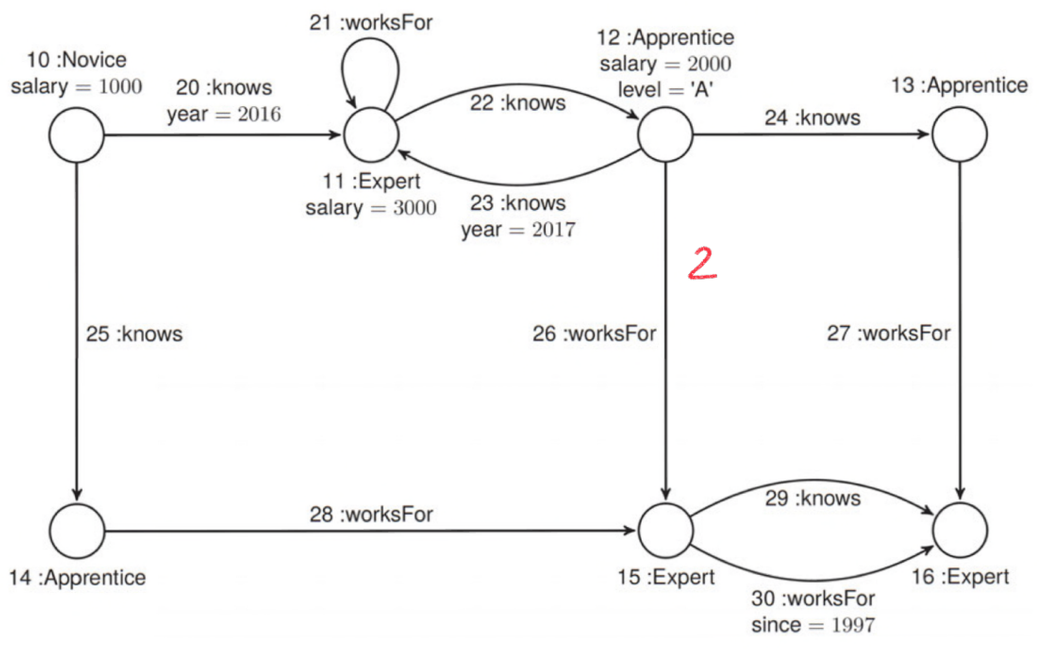
$q = :knows / :worksFor / :knows^+$

$[\![q]\!]_{G_{ex}} = \{(10,12)$





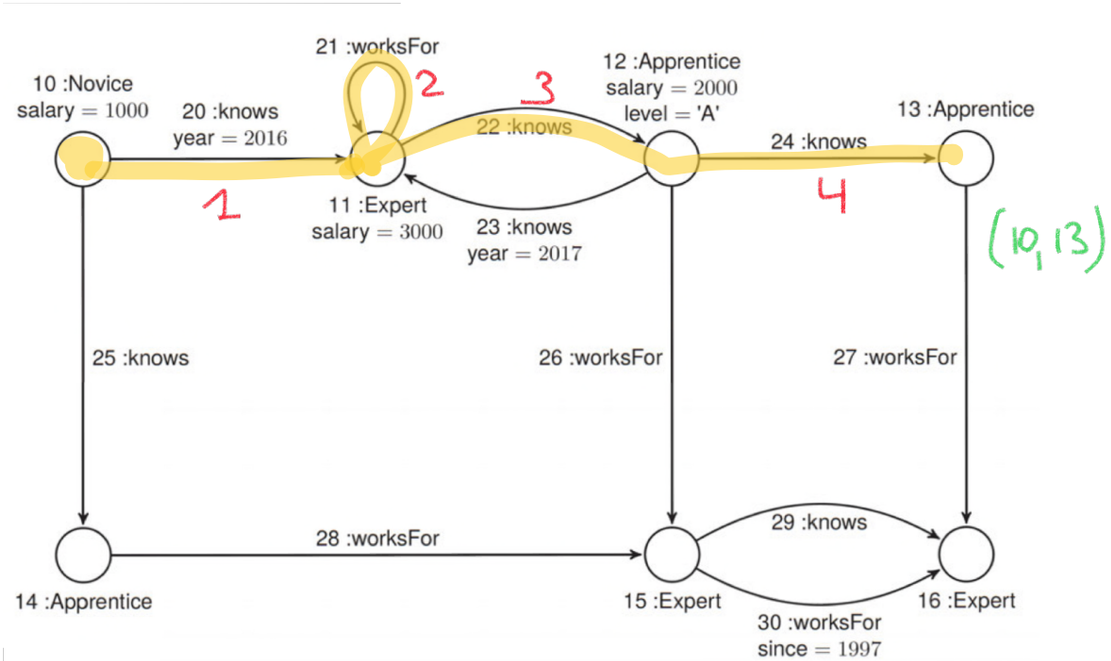


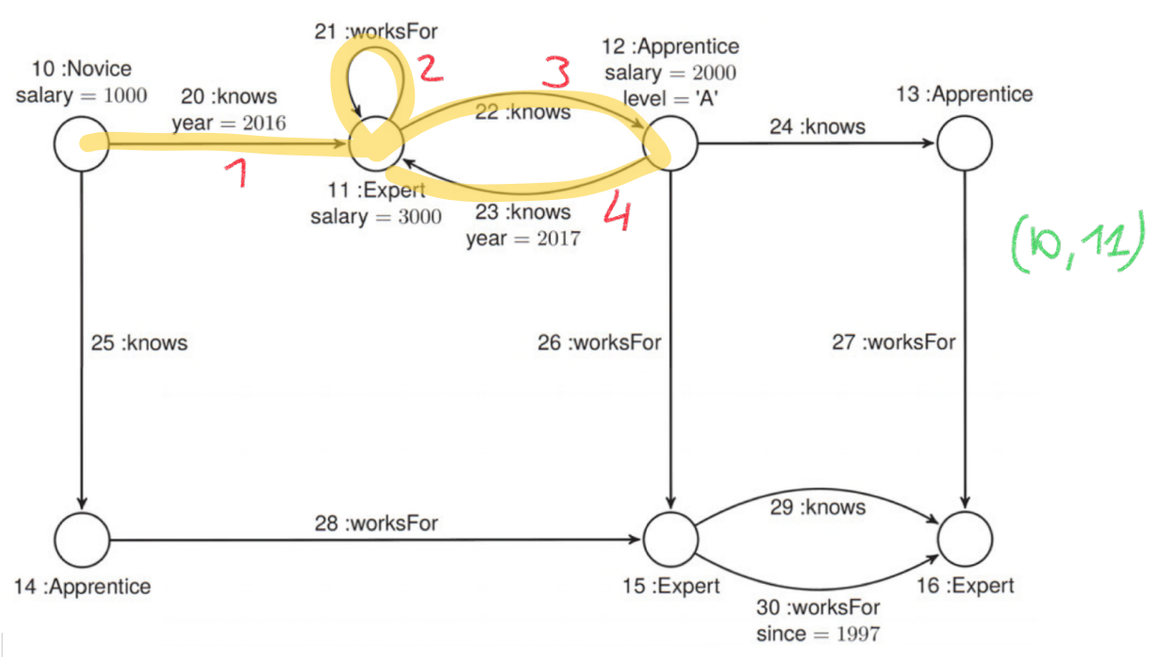


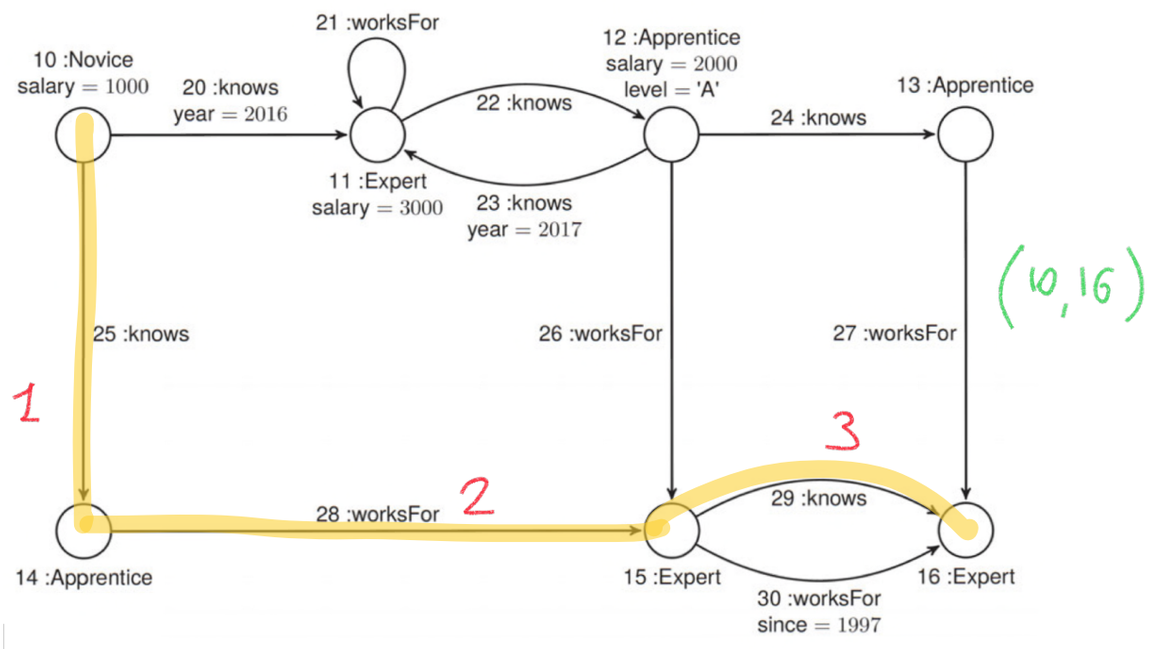
Example 2

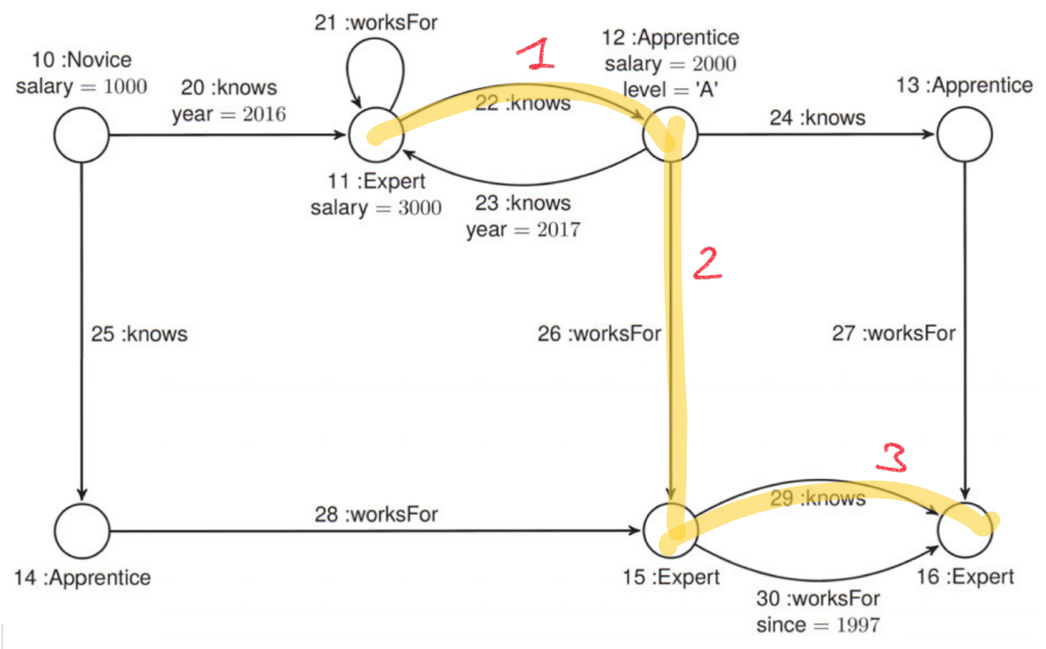
$q = :knows / :worksFor / :knows^+$

$[\![q]\!]_{G_{ex}} = \{(10,12)$





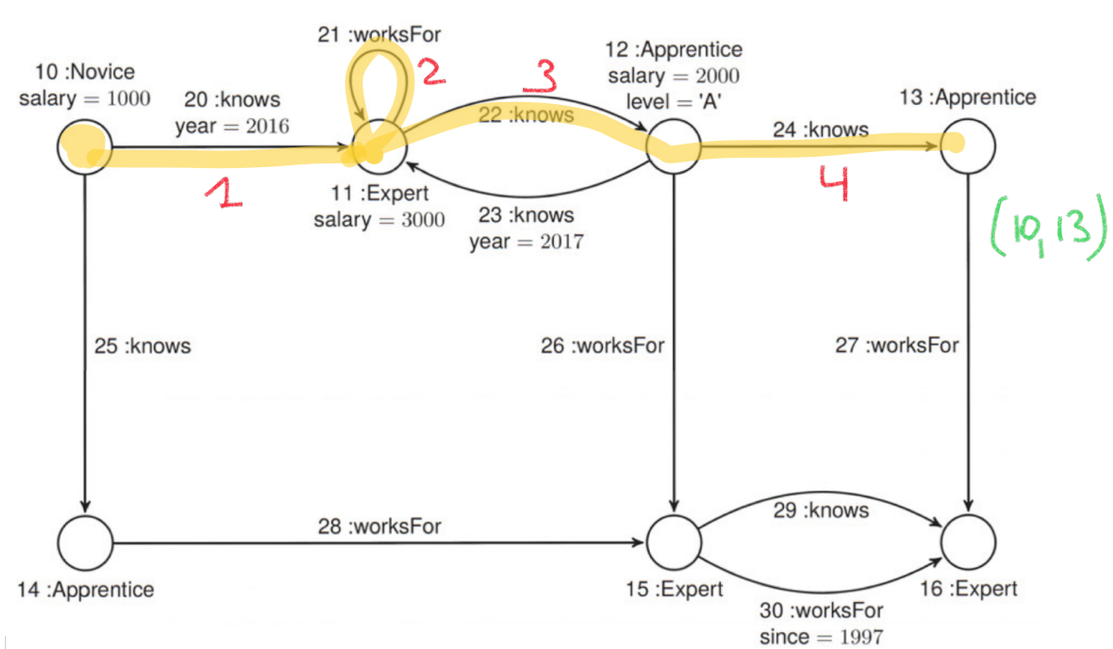


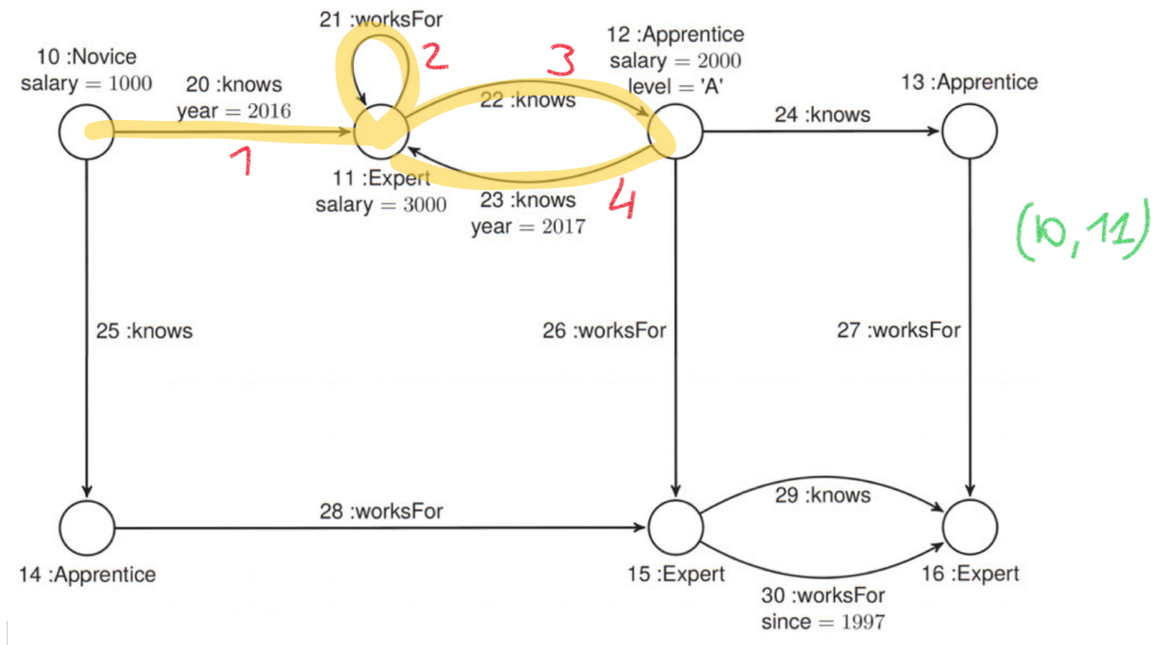


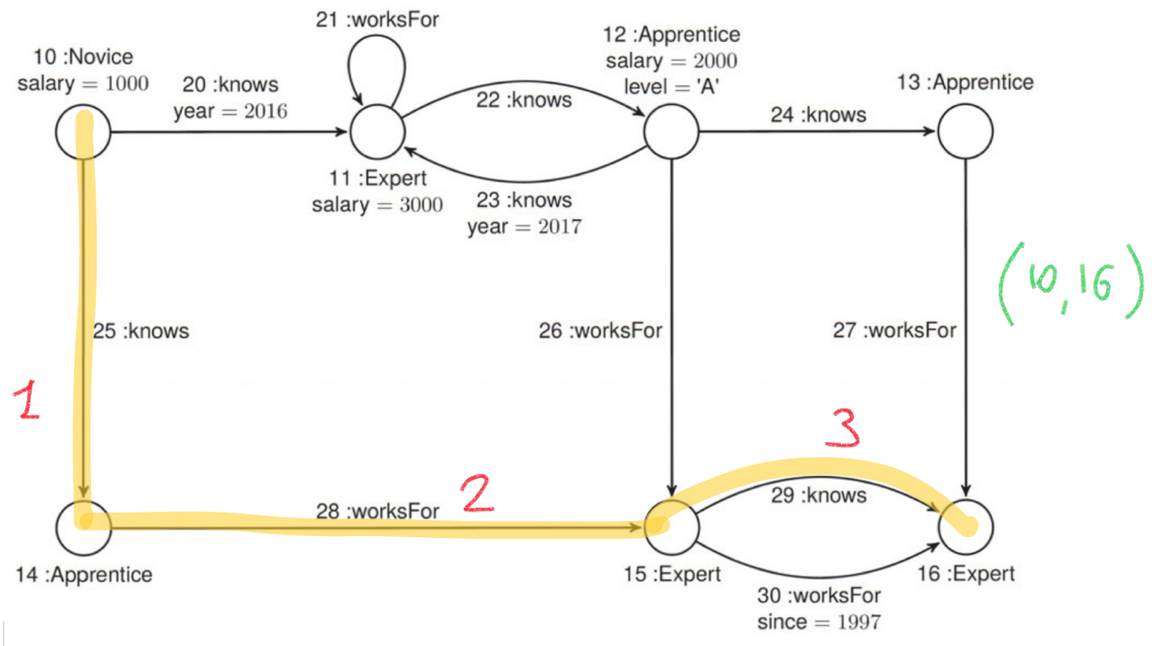
Example 2

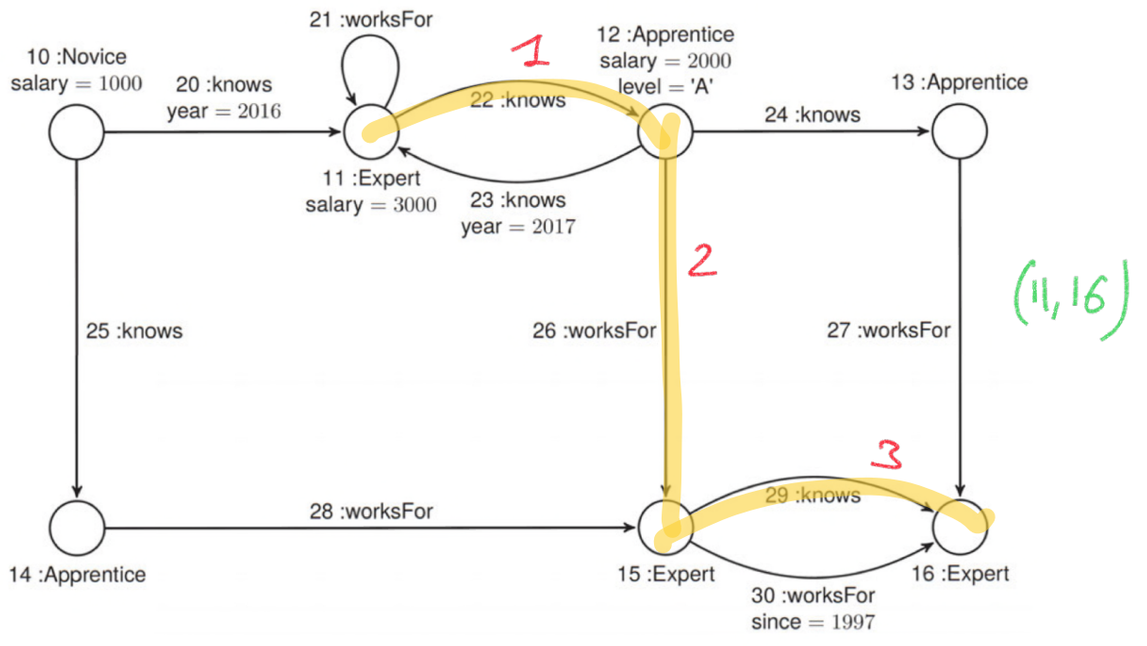
$q = :knows / :worksFor / :knows^+$

$[\![q]\!]_{G_{ex}} = \{(10,12)$





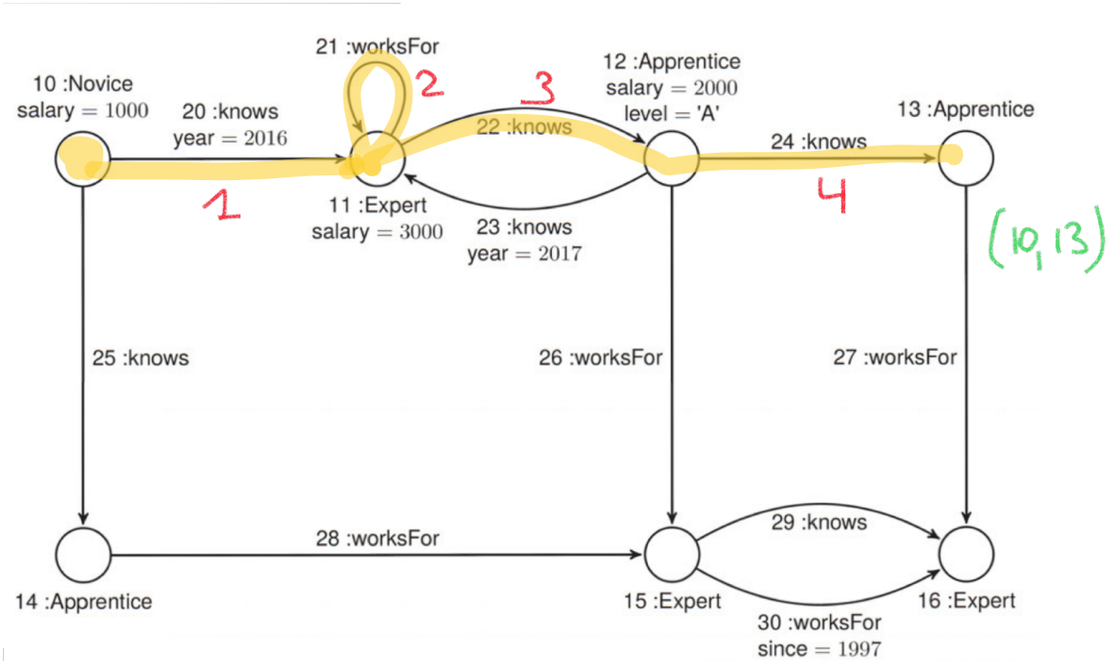


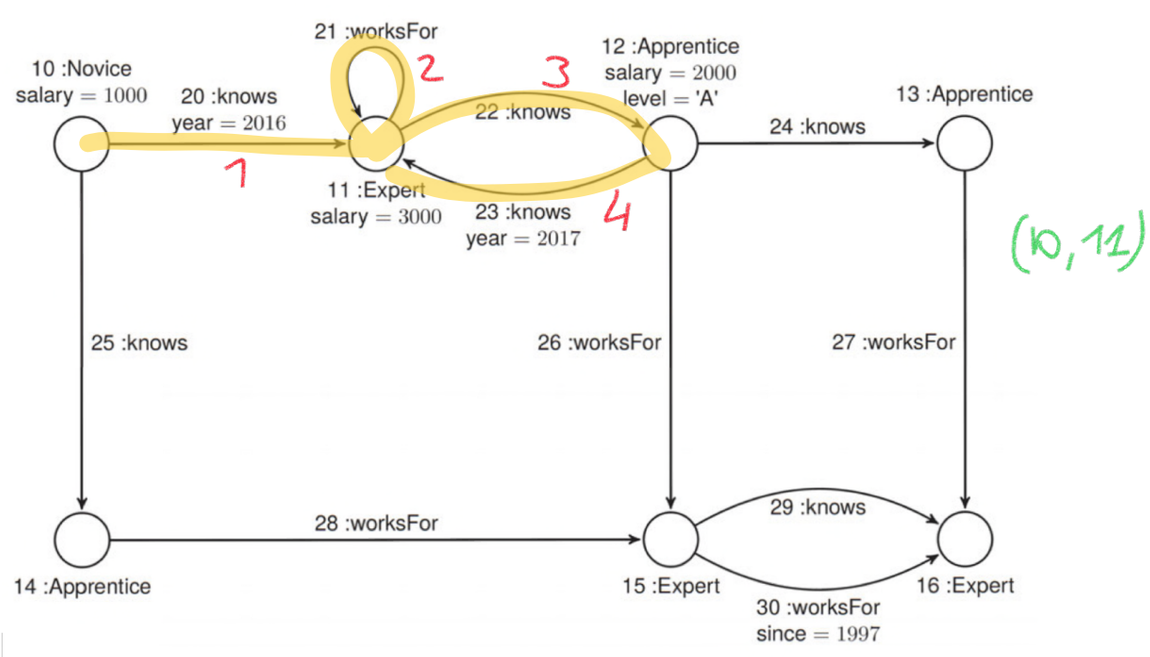


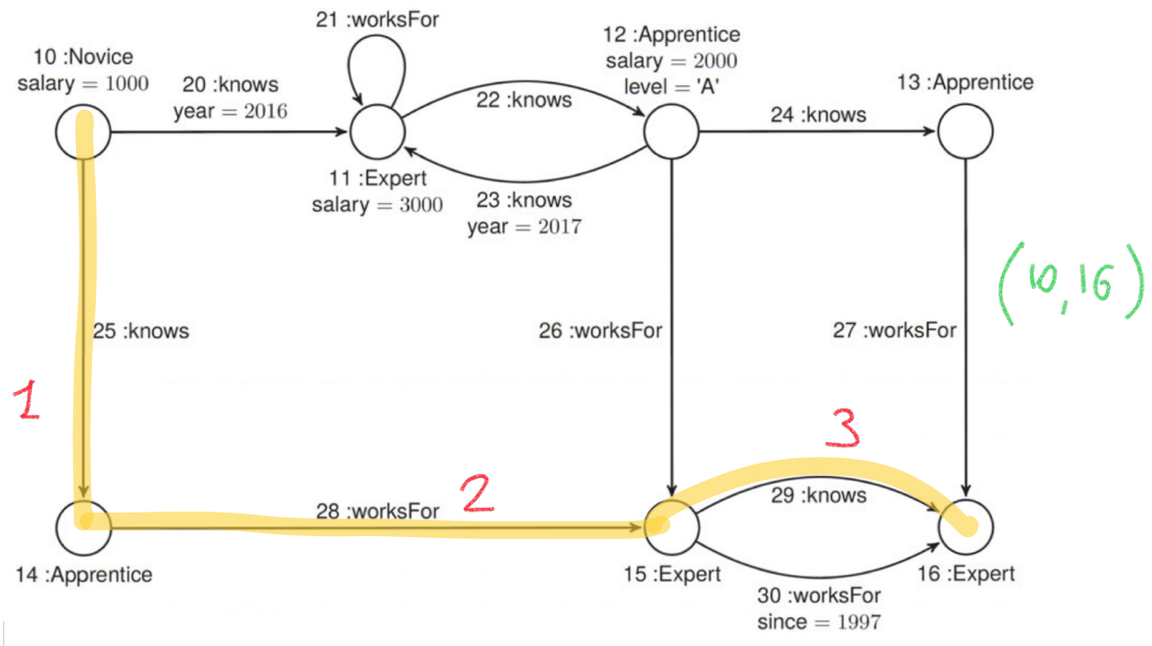
Example 2

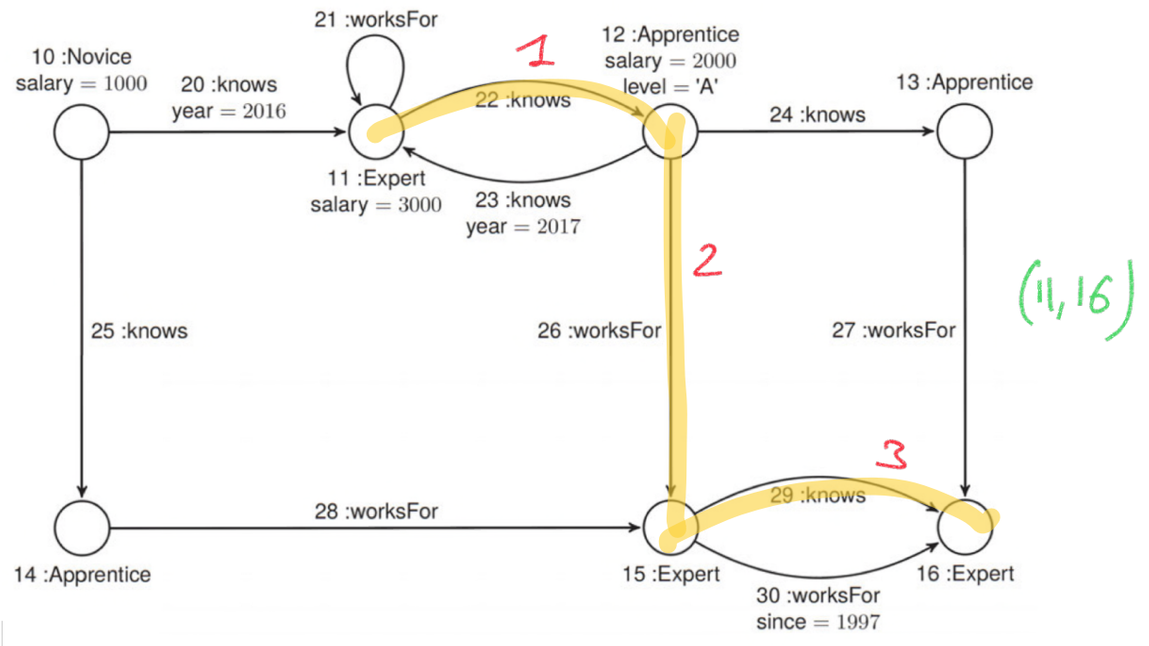
$q = :knows / :worksFor / :knows^+$

$[\![q]\!]_{G_{ex}} = \{(10,12), (10,13), (10,11), (10,16), (11,16)\}$



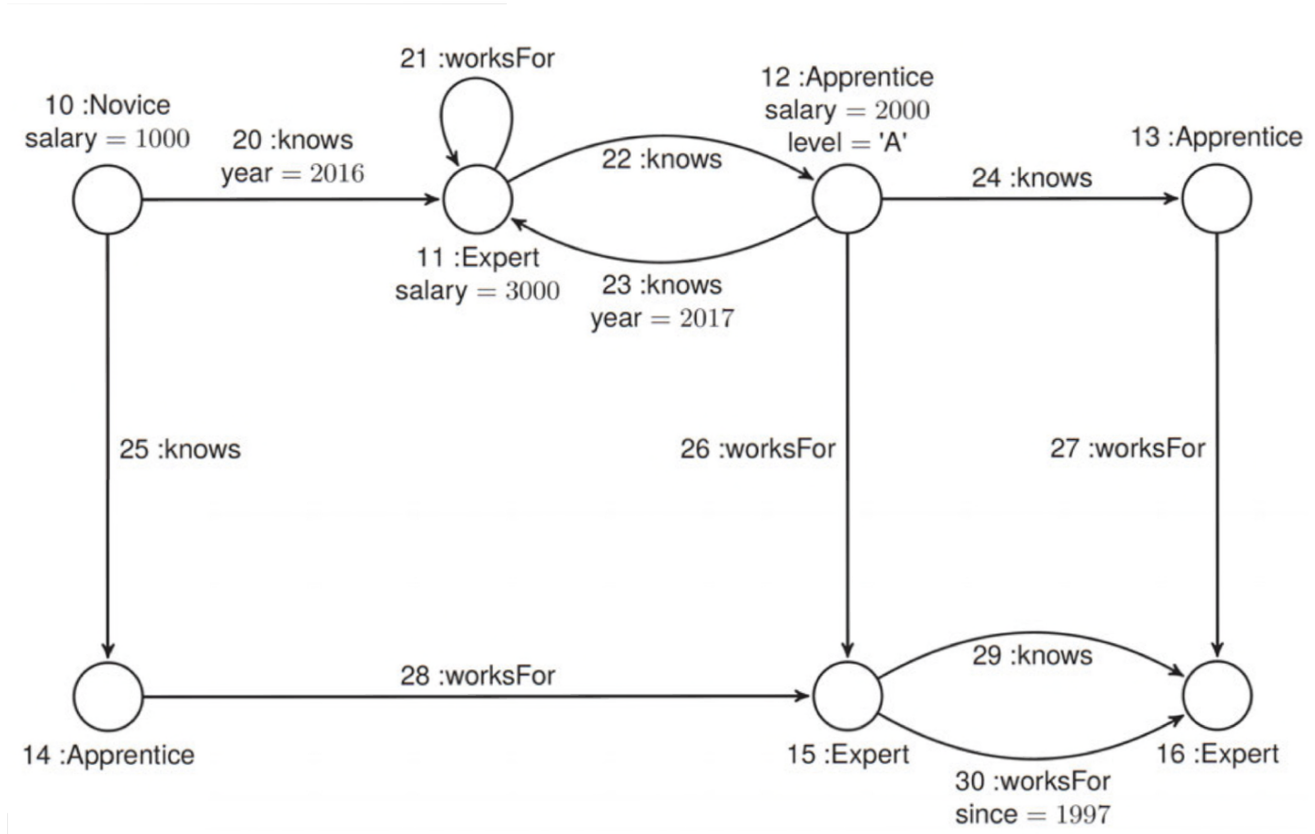






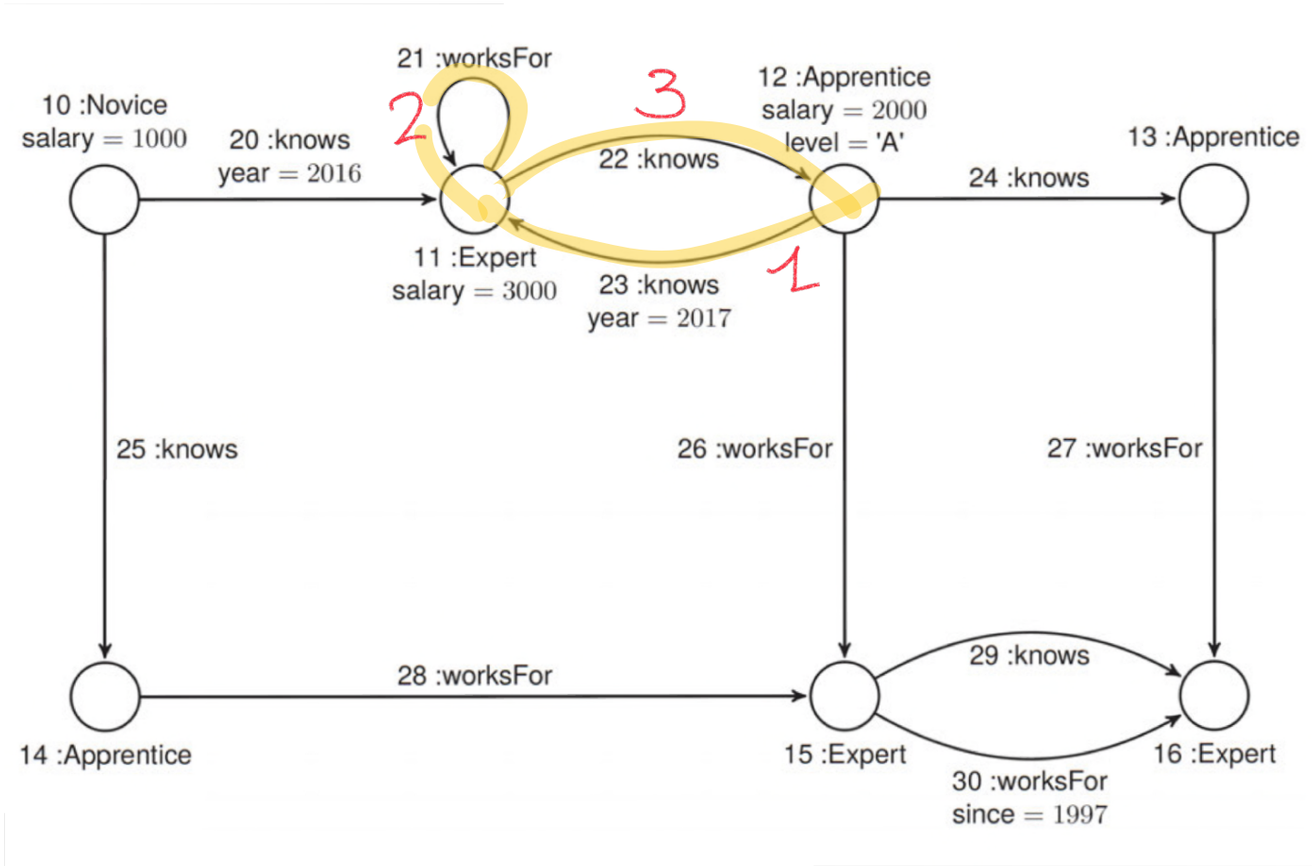
Example 2 $q = :\text{knows} / :\text{worksFor} / :\text{knows}^+$

$[\![q]\!]_{G_{ex}} = \{(10,12), (10,13), (10,11), (10,16), (11,16)\}$



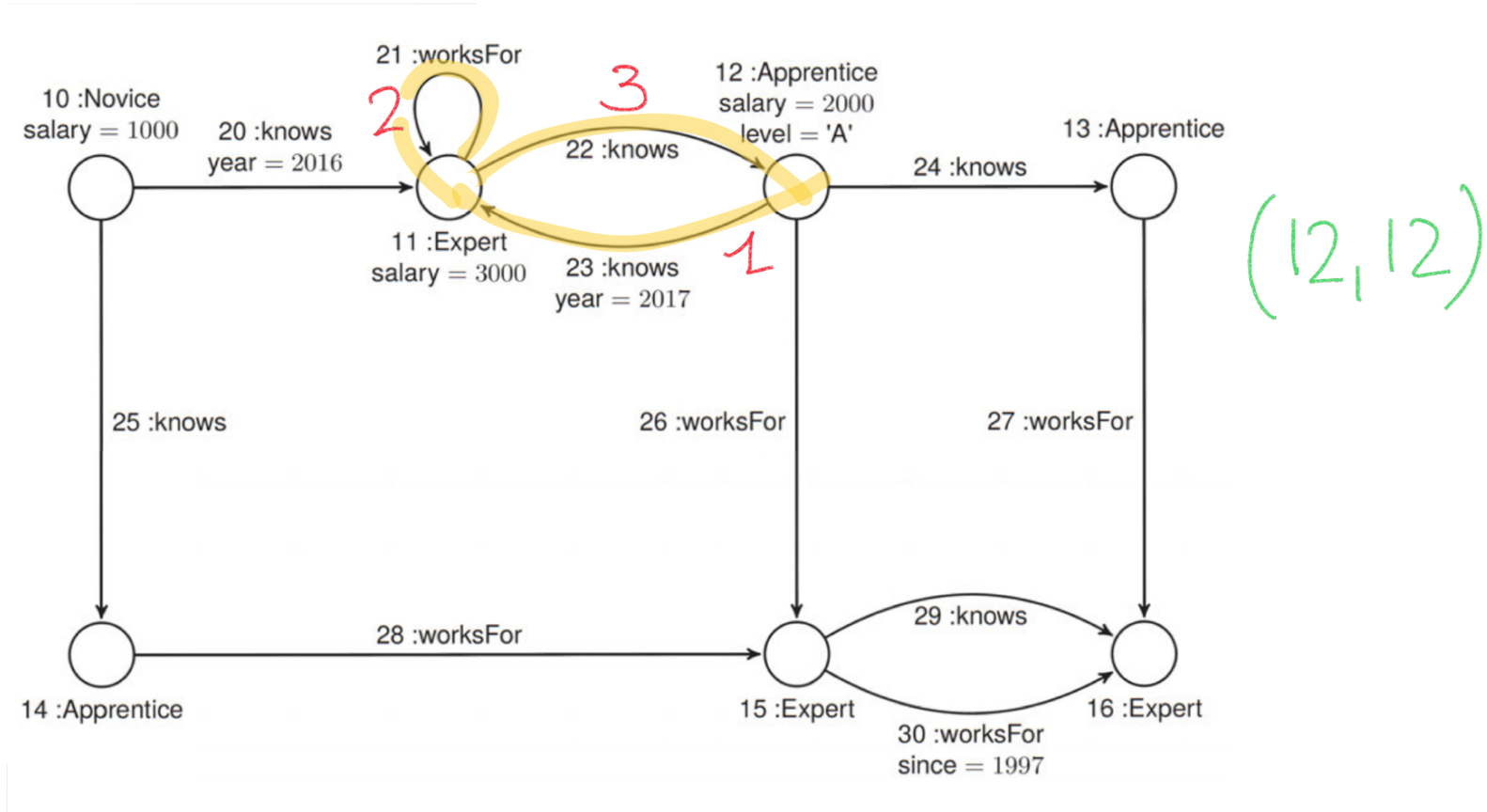
Example 2 $q = :\text{knows} / :\text{worksFor} / :\text{knows}^+$

$[\![q]\!]_{G_{ex}} = \{(10,12), (10,13), (10,11), (10,16), (11,16)$



Example 2 $q = :\text{knows} / :\text{worksFor} / :\text{knows}^+$

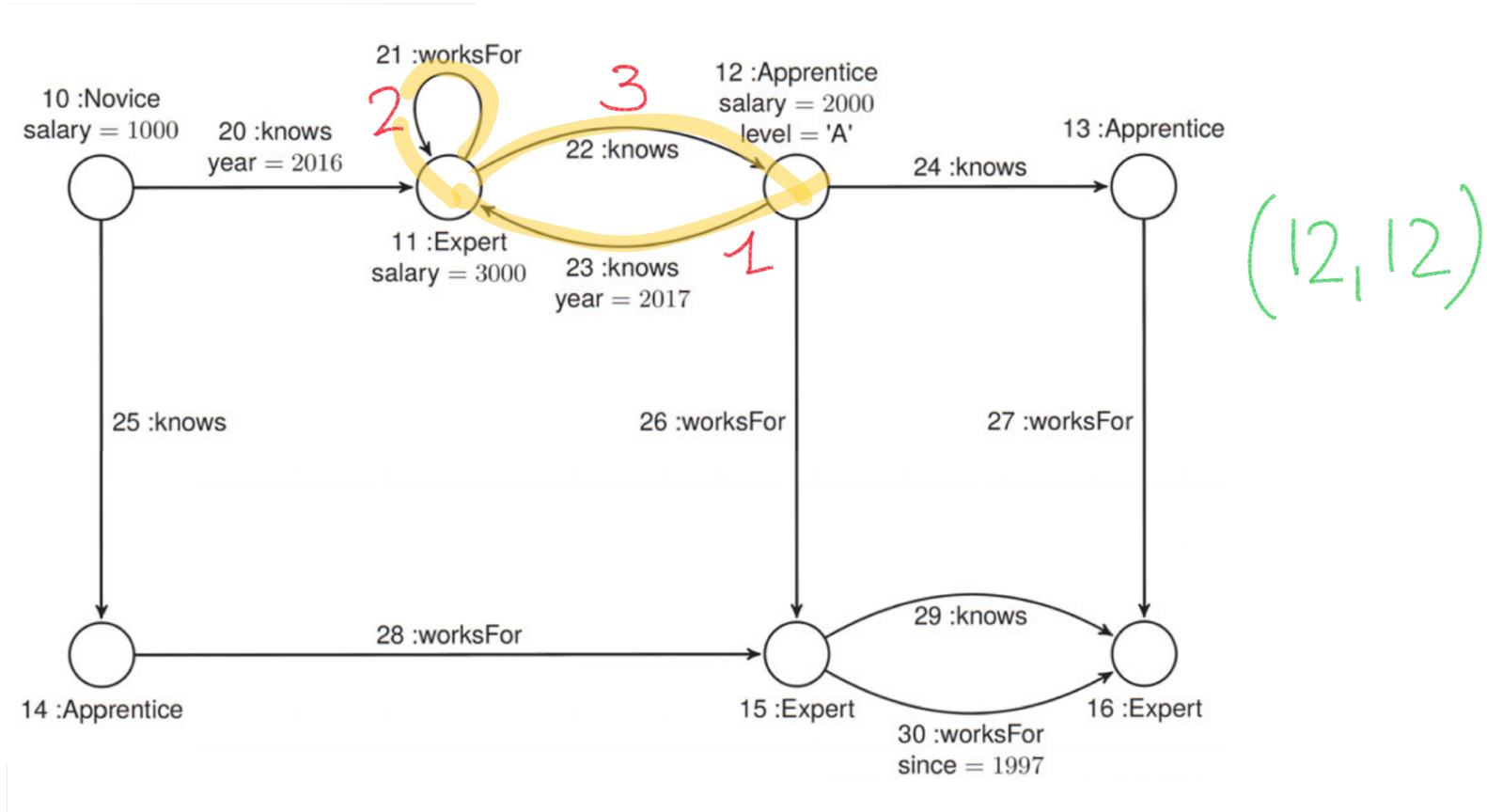
$[\![q]\!]_{G_{ex}} = \{(10,12), (10,13), (10,11), (10,16), (11,16)$



$(12,12)$

Example 2 $q = :\text{knows} / :\text{worksFor} / :\text{knows}^+$

$[\![q]\!]_{G_{ex}} = \{(10,12), (10,13), (10,11), (10,16), (11,16), (12,12)\}$



(12,12)

Conjunctive graph queries (CGQ)

CGQs identify sub structures in a graph

Let $\mathcal{V}$ be a set of vertex variables, a conjunctive query is defined as

$$\underbrace{(z_1, \dots, z_m)}_{\text{set of nodes}} \leftarrow \underbrace{a_1}_{\text{edge label}} \underbrace{(x_1, y_1)}_{\text{vertex variables}}, \dots, a_n\ (x_n, y_n)$$

where

- $m \geq 0, n \geq 0$
- $x_1, y_1, \dots, x_n, y_n \in \mathcal{V}$
- $a_1, \dots, a_n \in \mathcal{L}$
- for each $0 \leq i \leq m$, it holds that $z_i \in \{x_1, y_1, \dots, x_n, y_n\}$
  where m is the arity of the expression

Conjunctive graph queries

Given $r = (z_1, \dots, z_m) \leftarrow a_1(x_1, y_1), \dots, a_n(x_n, y_n)$

A mapping for r on G is a function $\mu : V \to V$ s.t.
for each $1 \leq i \leq n$, $\exists\ e_i \in E$ where

$$\eta(e_i) = (\mu(x_i), \mu(y_i)) \wedge a_i \in \lambda(e_i)$$

Conjunctive graph queries

Given $r = (z_1, \dots, z_m) \leftarrow a_1(x_1, y_1), \dots, a_n(x_n, y_n)$

A mapping for r on G is a function $\mu : V \to V$ s.t.
for each $1 \leq i \leq n$, $\exists\ e_i \in E$ where

$$\eta(e_i) = (\mu(x_i), \mu(y_i)) \wedge a_i \in \lambda(e_i)$$

this is a function
binding a variable
to a vertex

Conjunctive graph queries

Given $r = (z_1, \dots, z_m) \leftarrow a_1(x_1, y_1), \dots, a_n(x_n, y_n)$

A mapping for r on G is a function $\mu : V \to V$ s.t. for each $1 \le i \le n$, $\exists\, e_i \in E$ where

$$\eta(e_i) = (\mu(x_i), \mu(y_i)) \wedge a_i \in \lambda(e_i)$$

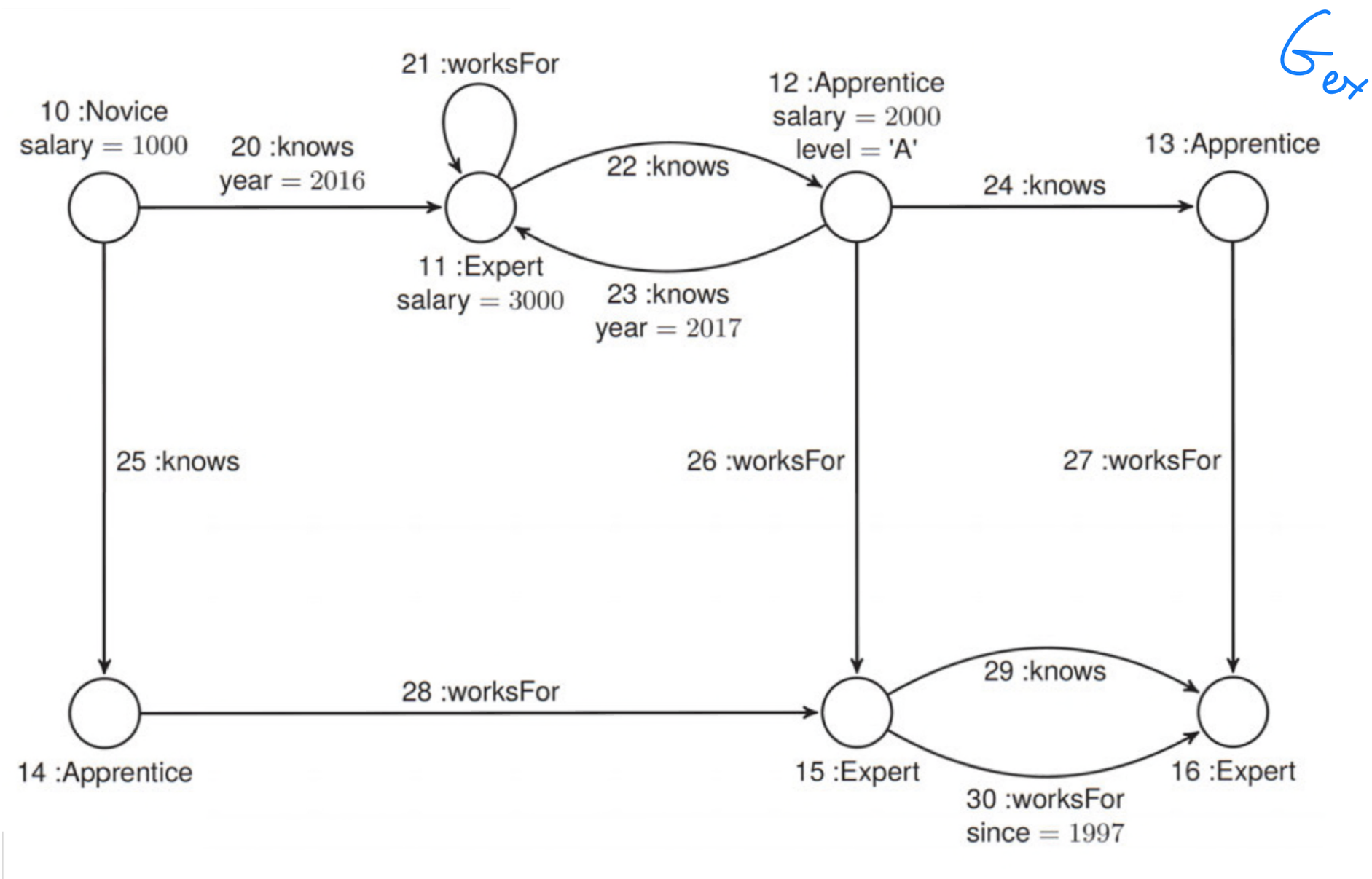
this is a function binding a variable to a vertex

The evaluation of r on G is

$$[\![r]\!]_G = \{ \mu(z_1), \dots, \mu(z_m) \mid \mu \text{ is a mapping for } r \text{ on } G \}$$

Example 1

$q_1 = (a, b) \leftarrow :knows(a, b)$

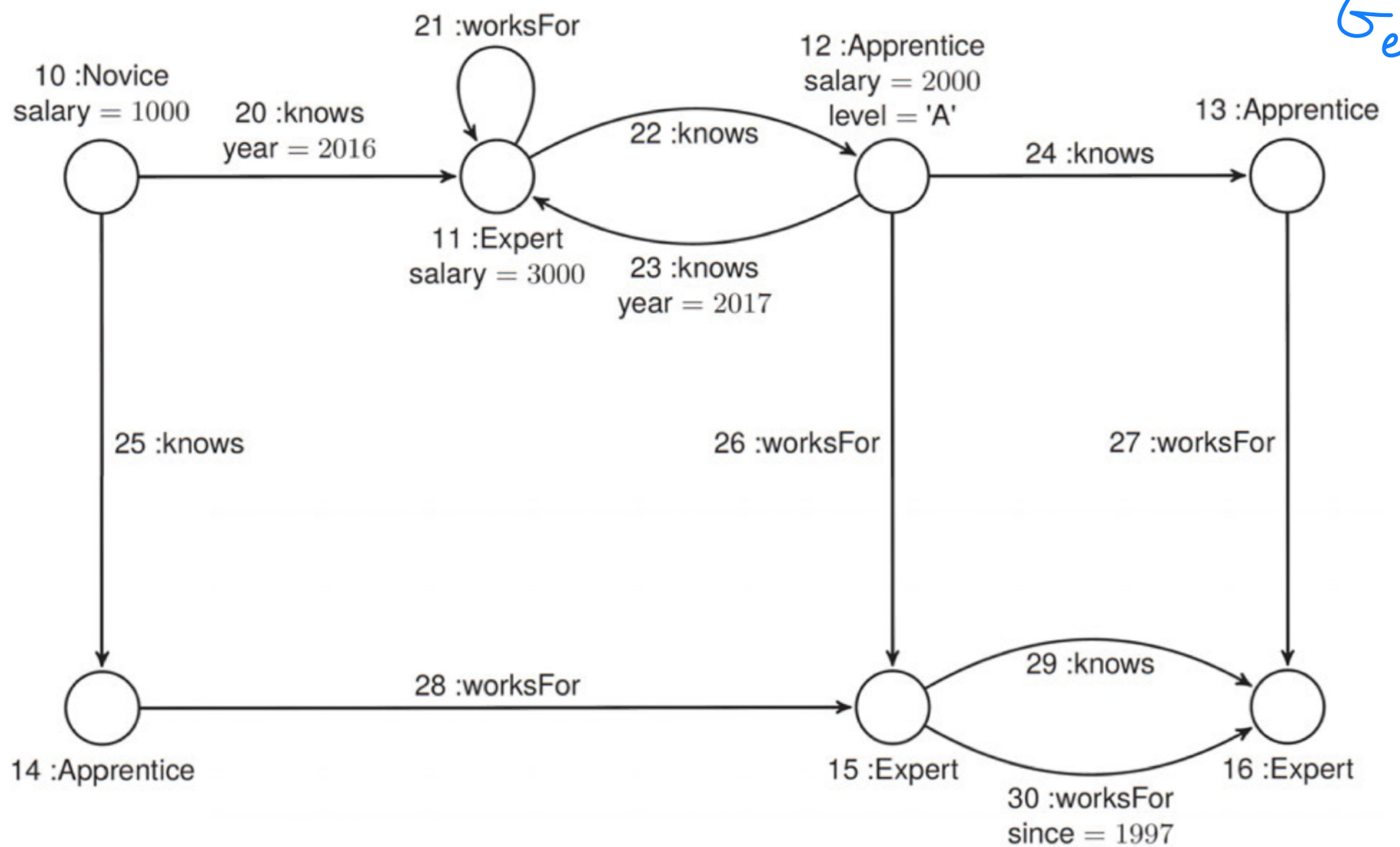


Example 1

$q_1 = (a, b) \longleftarrow :knows(a, b)$

We need to find a function μ mapping a and b to $V_{ex} = \{10, 11, 12, \ldots, 16\}$

G_{ex}



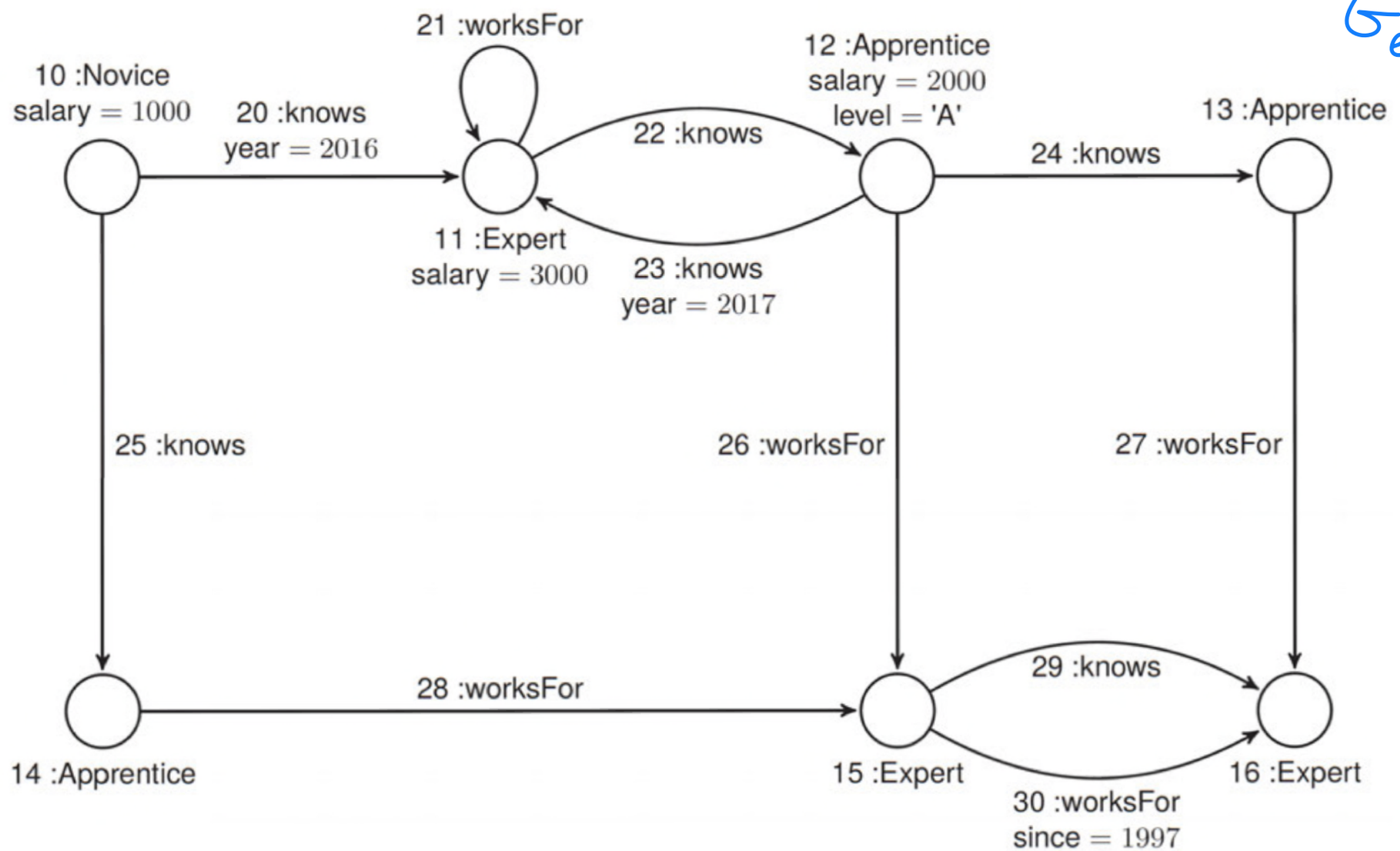
Example 1

$q_1 = (a, b) \longleftarrow :knows(a, b)$

We need to find a function μ mapping a and b to $V_{ex} = \{10, 11, 12, \ldots, 16\}$

The variable a can be bound to to all vertices that are sources of an edge labelled “knows”

G_{ex}



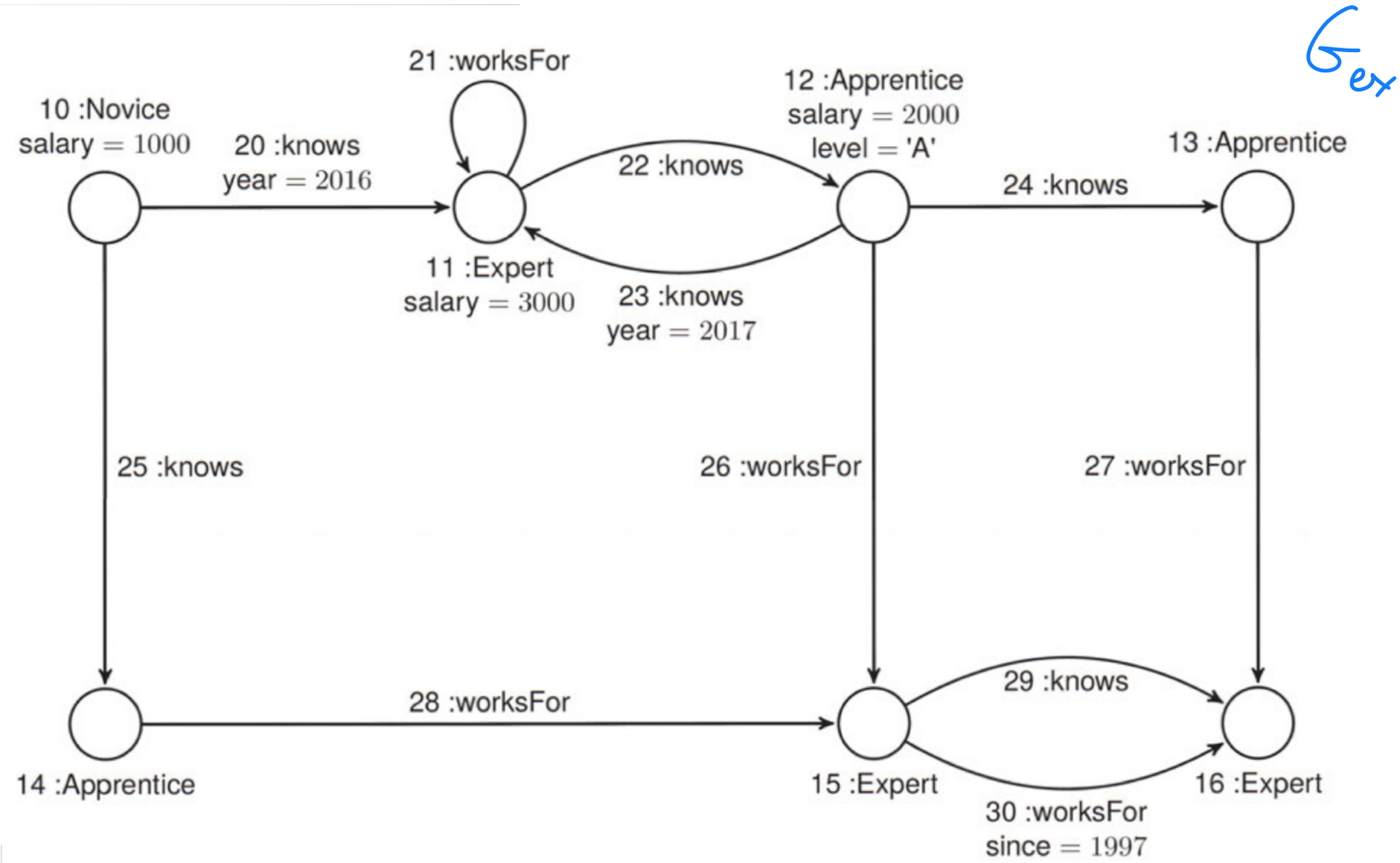
Example 1

$q_1 = (a, b) \leftarrow :knows(a, b)$

We need to find a function μ mapping a and b to $V_{ex} = \{10, 11, 12, \ldots, 16\}$

The variable a can be bound to to all vertices that are sources of an edge labelled "knows"

$a = \{10,11,12,15\}$



Example 1

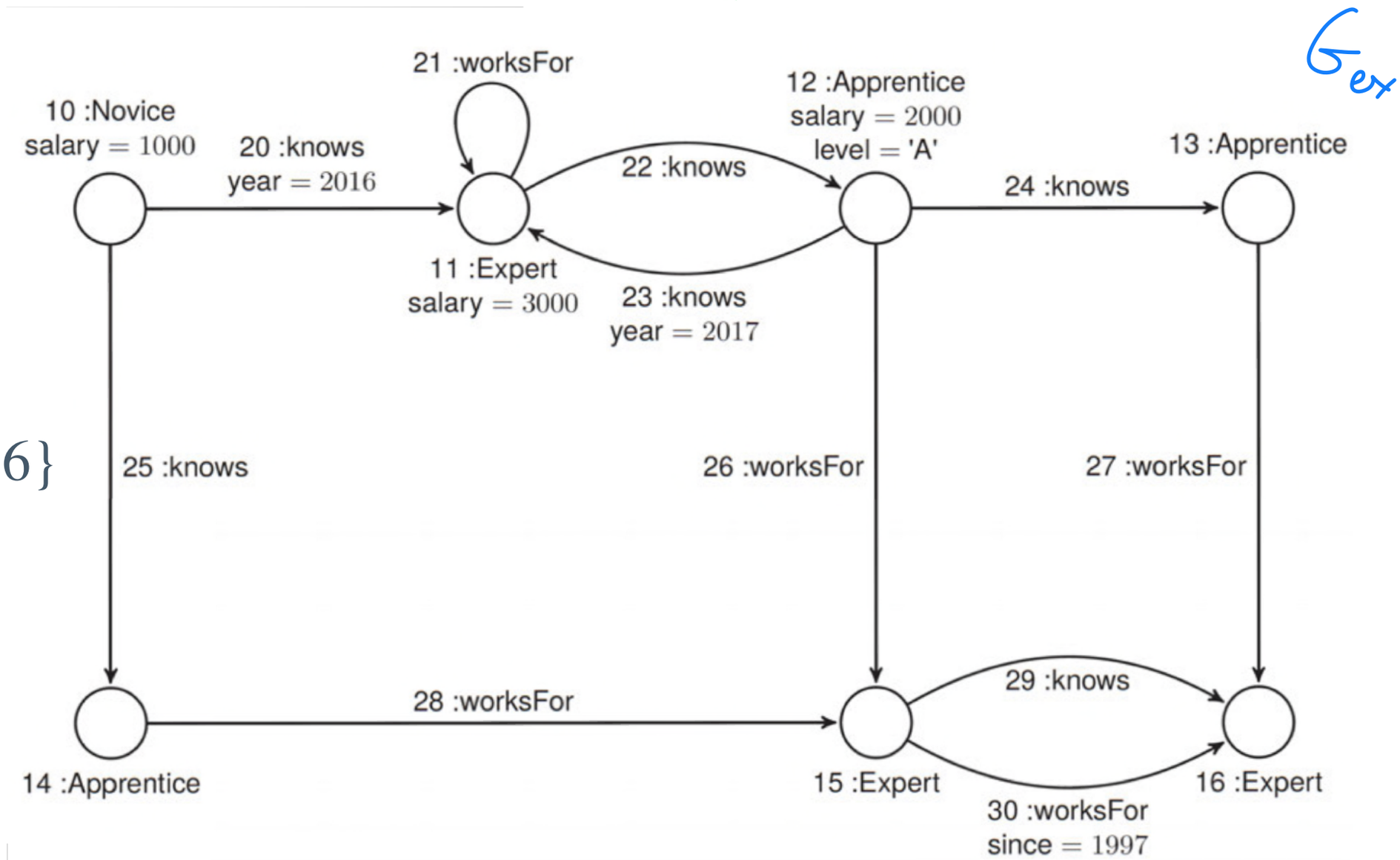
$q_1 = (a, b) \longleftarrow :knows(a, b)$

We need to find a function μ mapping a and b to $V_{ex} = \{10, 11, 12, \ldots, 16\}$

The variable a can be bound to to all vertices that are sources of an edge labelled “knows”

$a = \{10,11,12,15\}$

$b = \{11,12,13,14,16\}$



Example 1

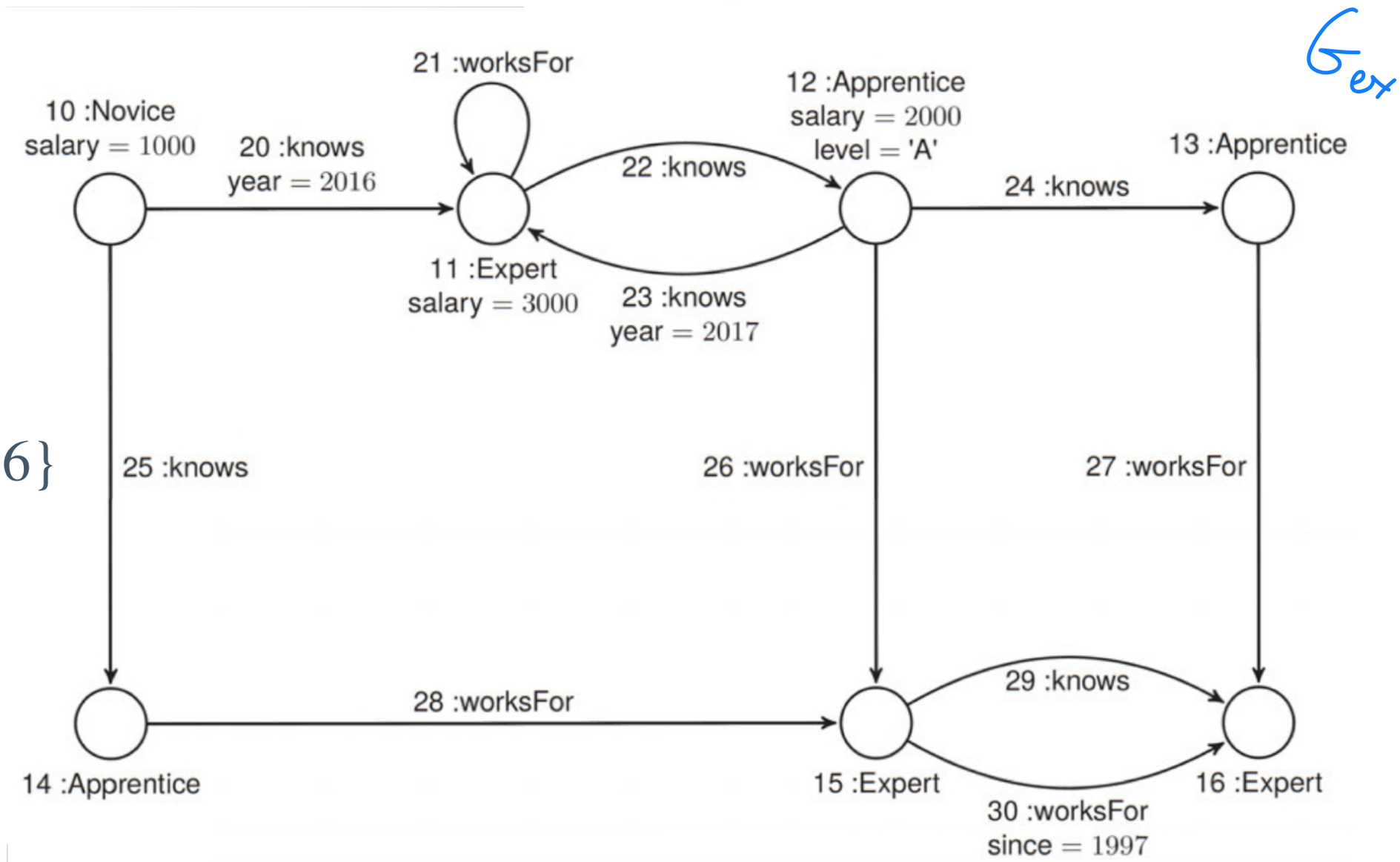
$q_1 = (a, b) \leftarrow :knows(a, b)$

We need to find a function μ mapping a and b to $V_{ex} = \{10, 11, 12, \ldots, 16\}$

The variable a can be bound to to all vertices that are sources of an edge labelled "knows"

$a = \{10,11,12,15\}$

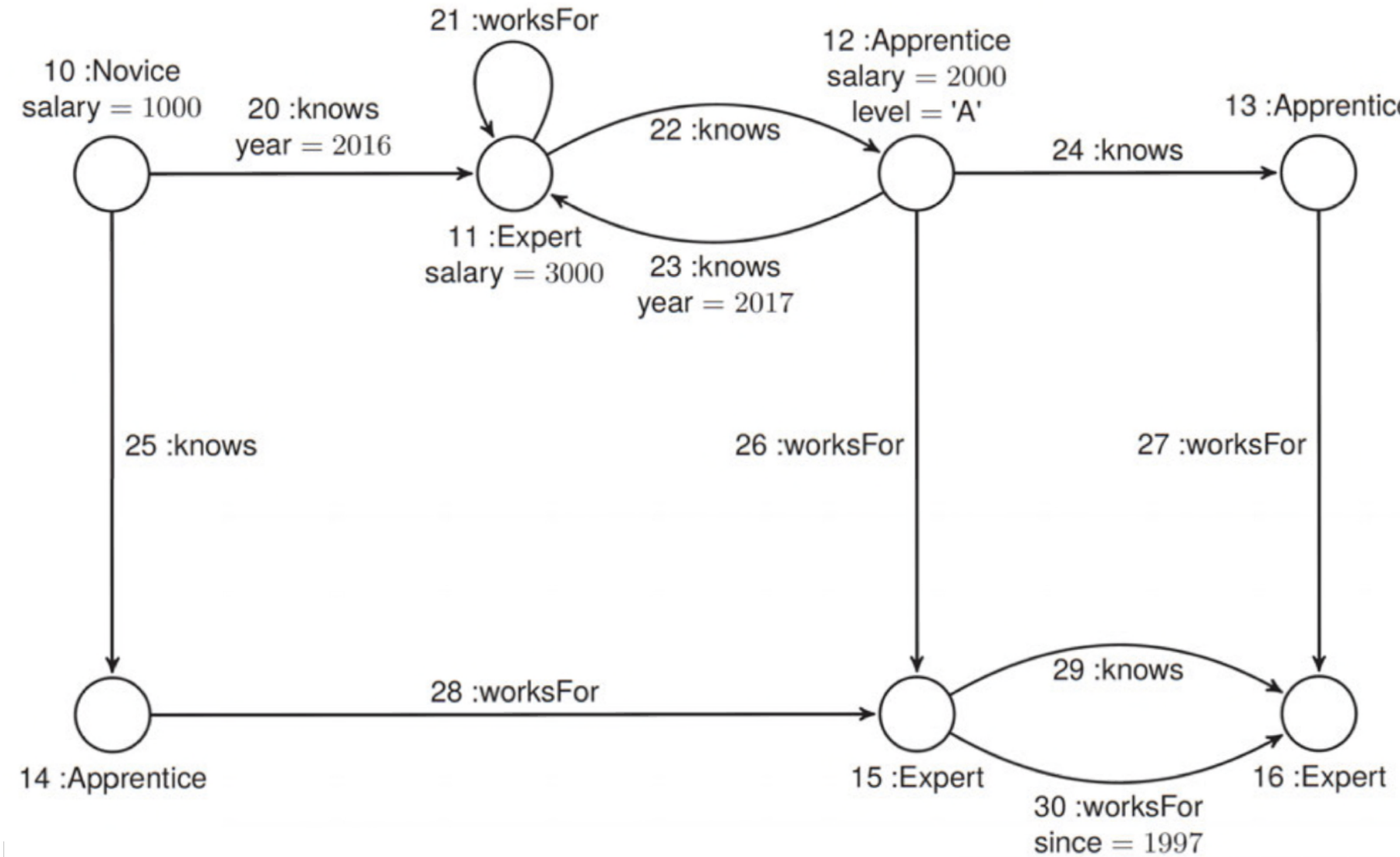
$b = \{11,12,13,14,16\}$



$[\![q_1]\!]_{G_{ex}} = (10,11), (11,12), (12,11), (12,13), (10,14), (15,16)$

Example 2

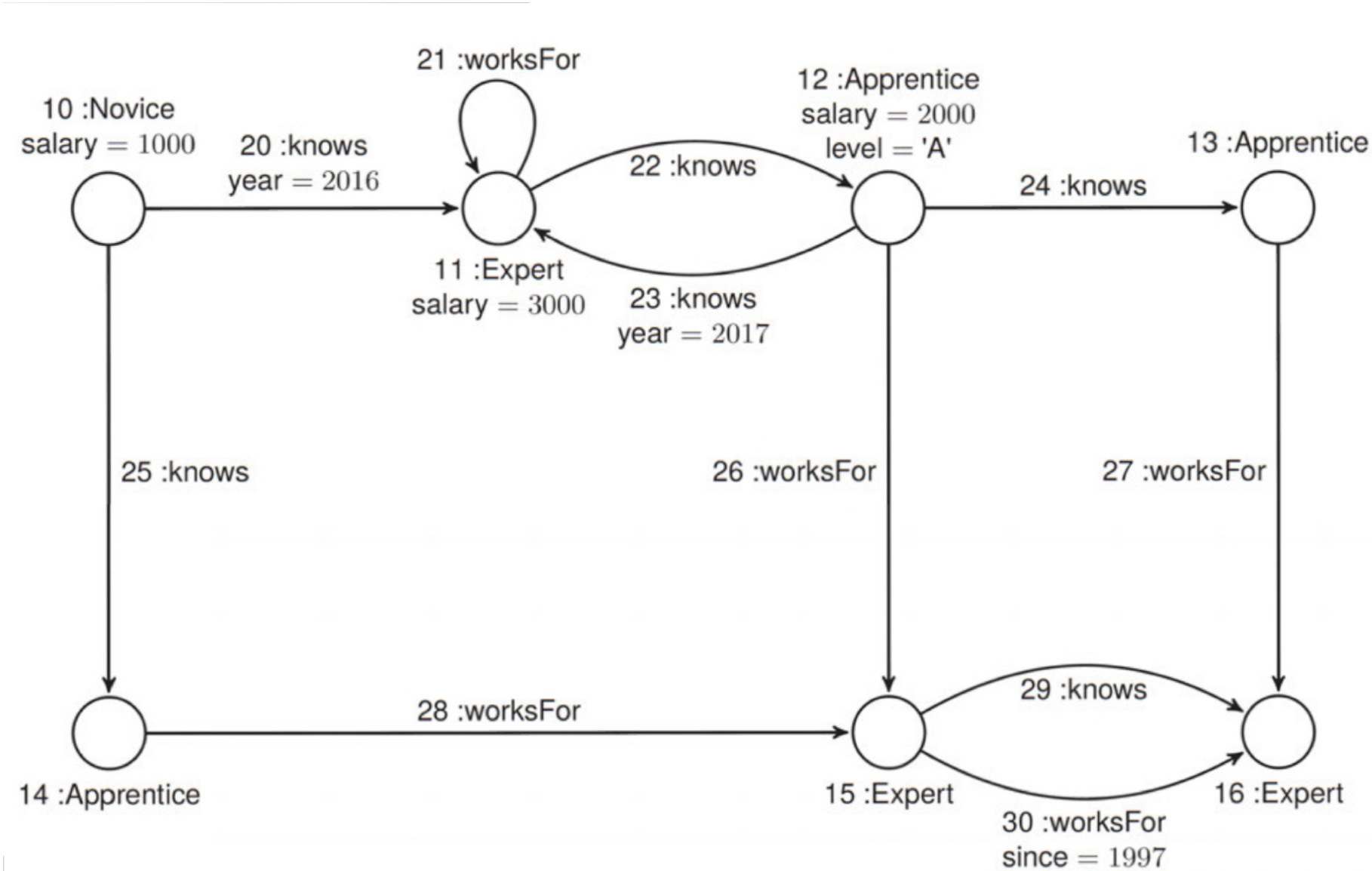
$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$



Example 2

$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

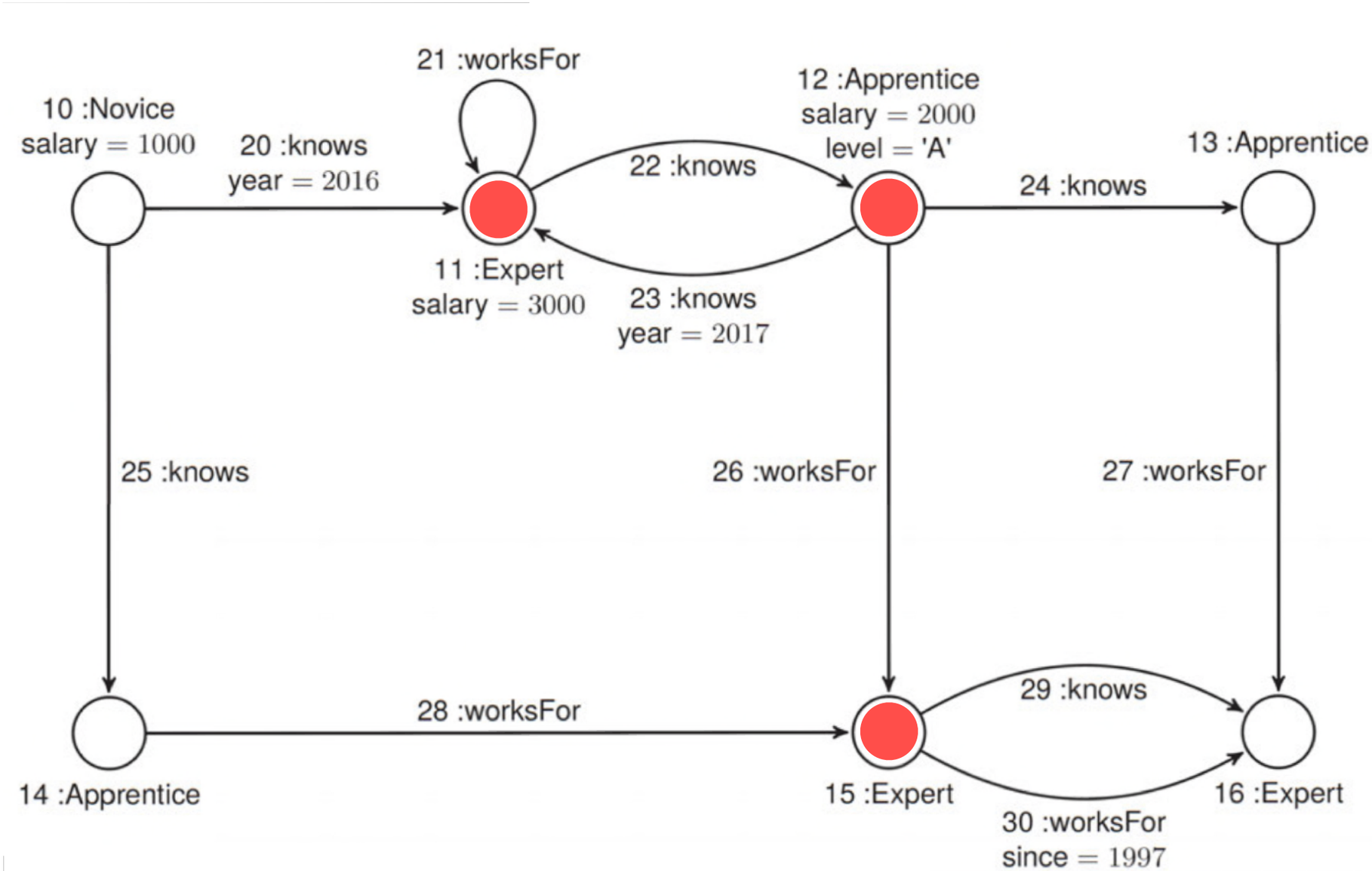
Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”



Example 2

$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

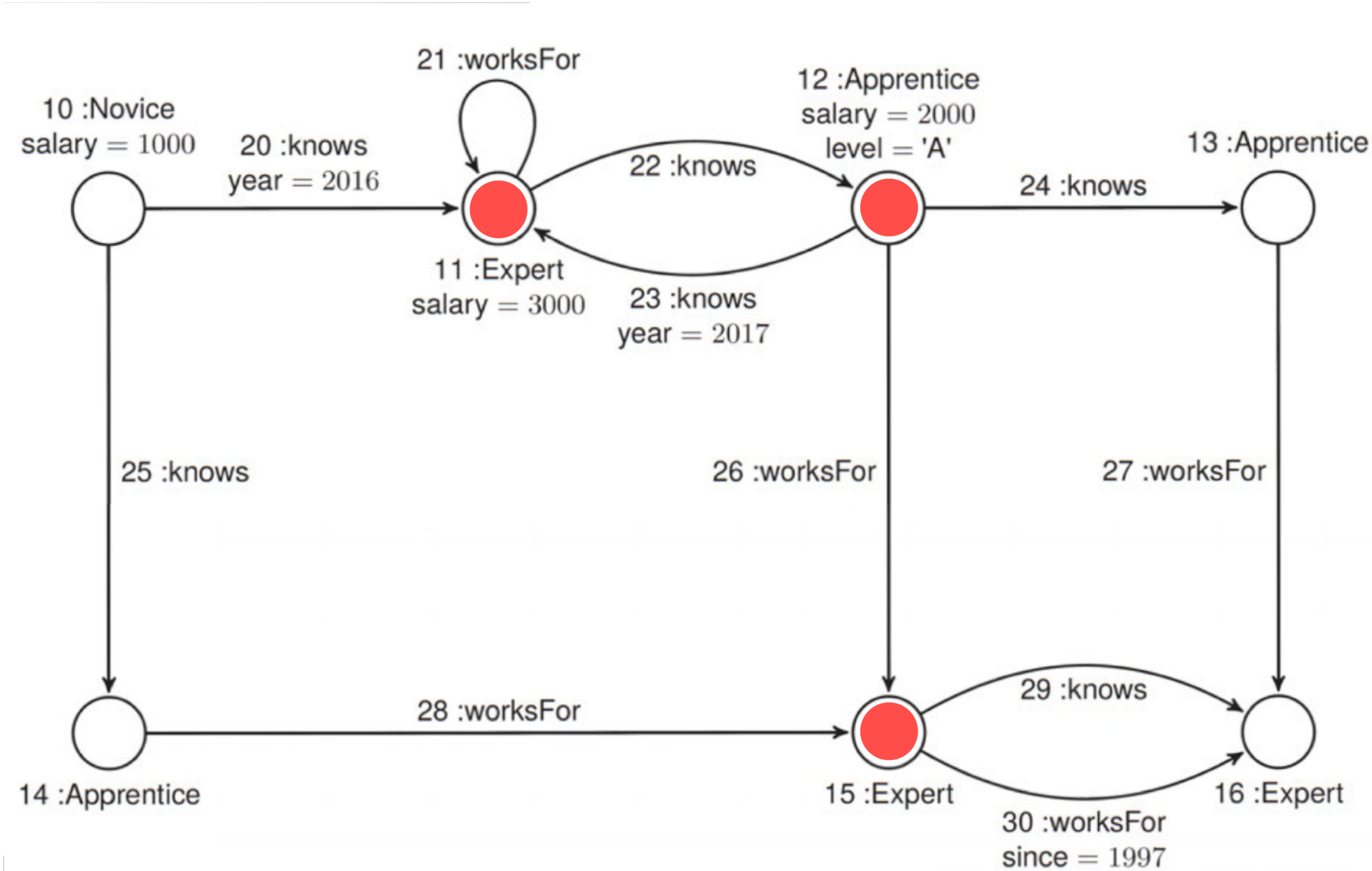


Example 2

$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”

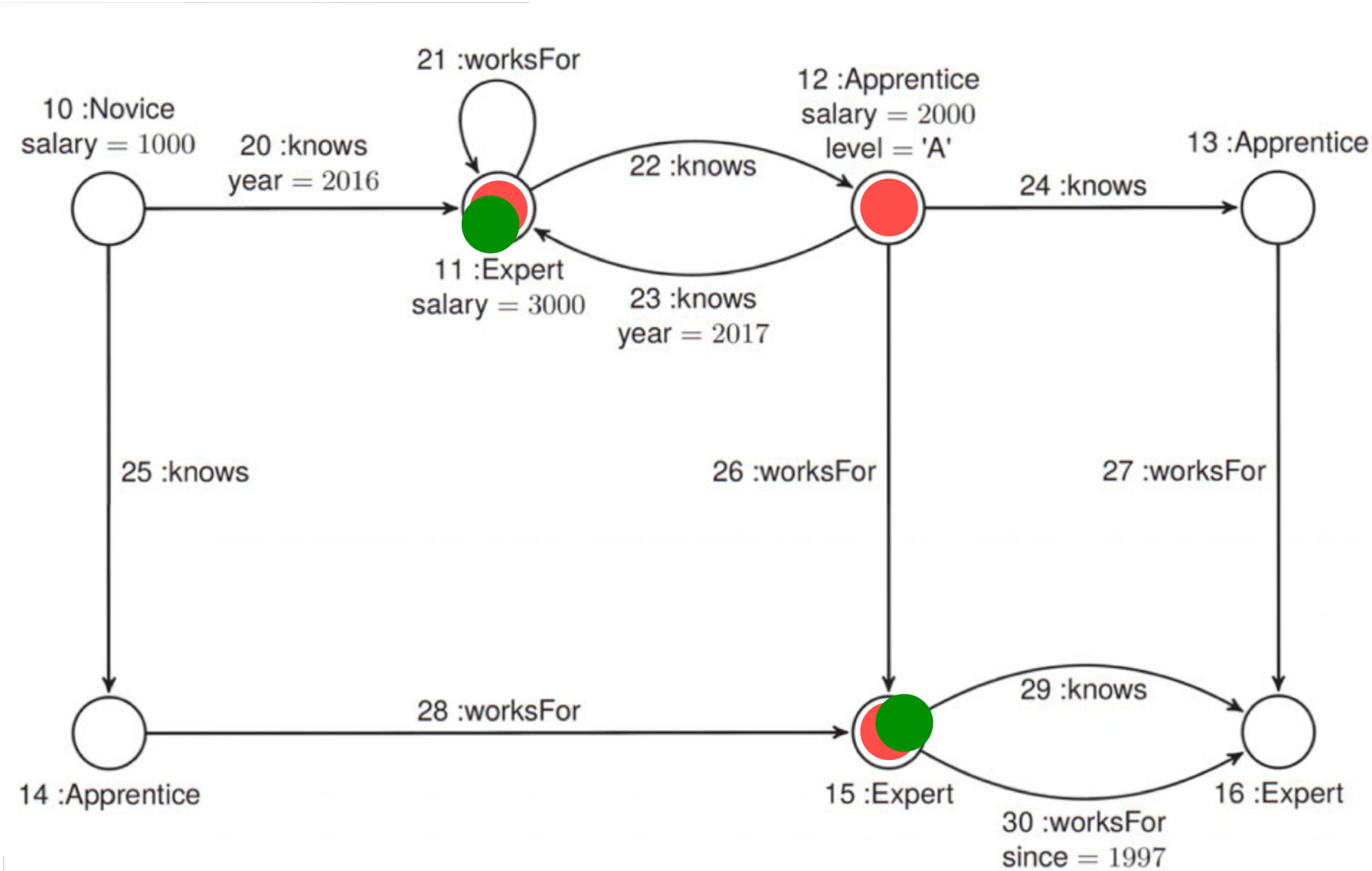


Example 2

$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”



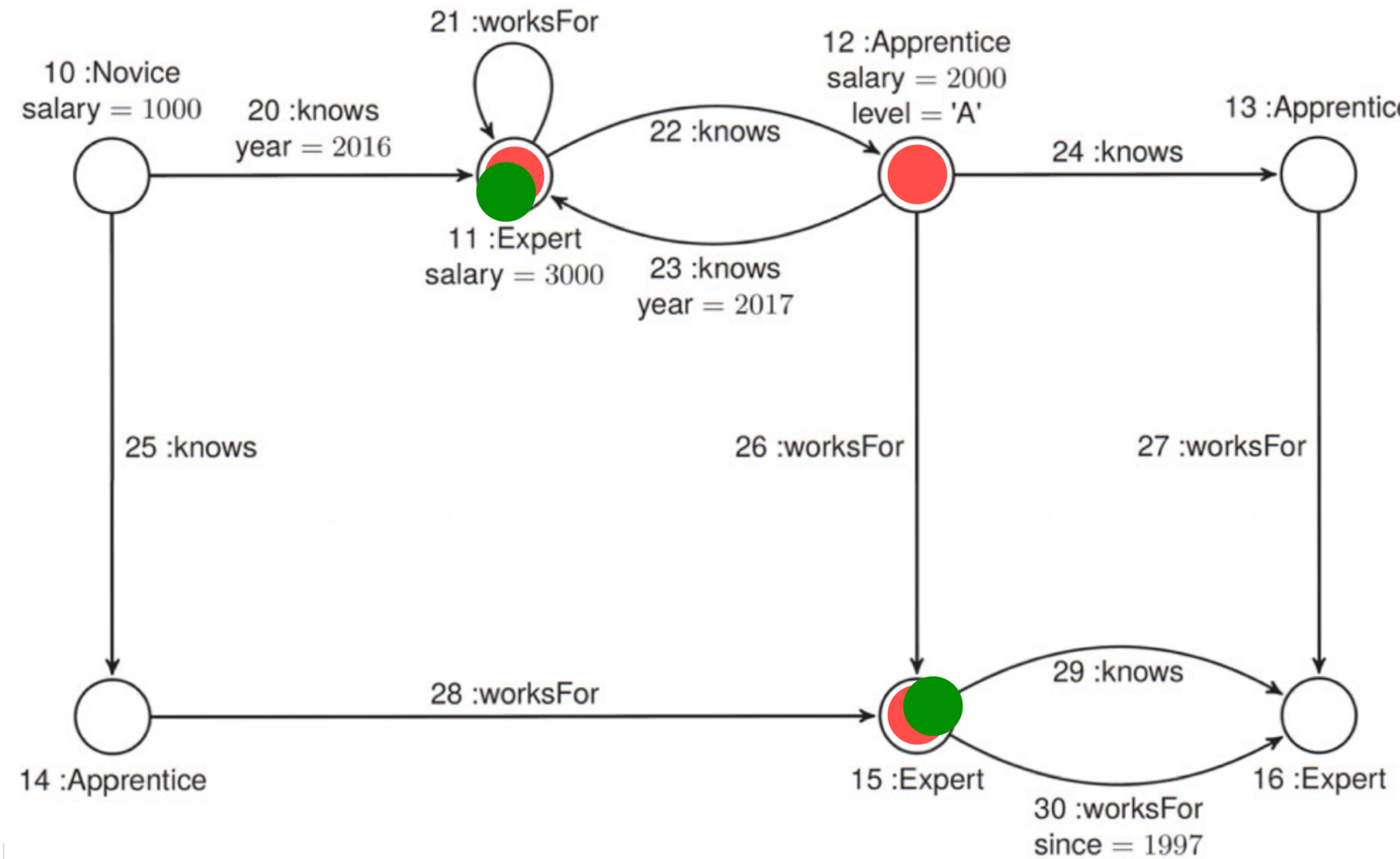
Example 2

$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”

Check all target vertices of an edge labeled “worksFor” and of an edge labeled “knows”



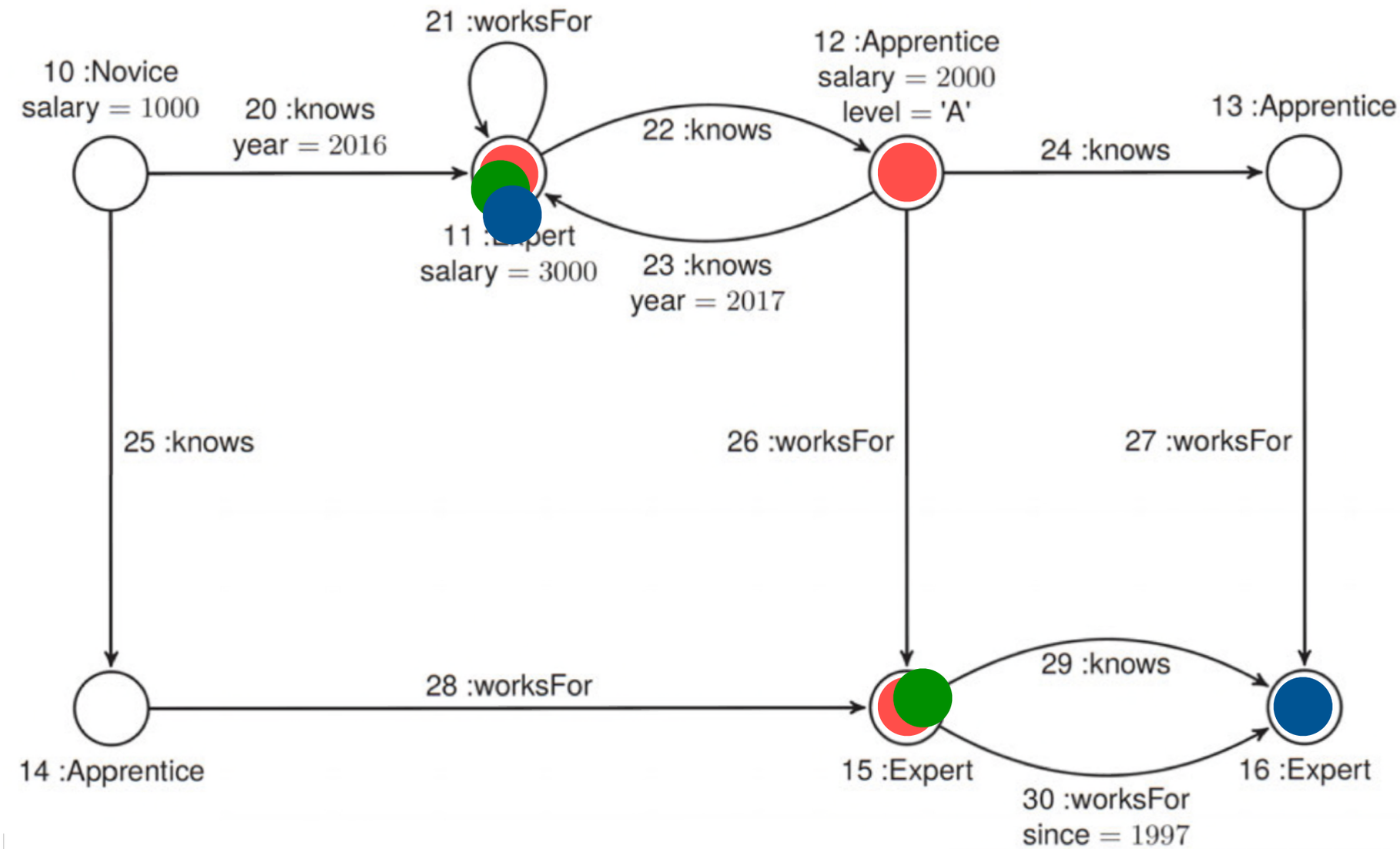
Example 2

$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”

Check all target vertices of an edge labeled “worksFor” and of an edge labeled “knows”



Example 2

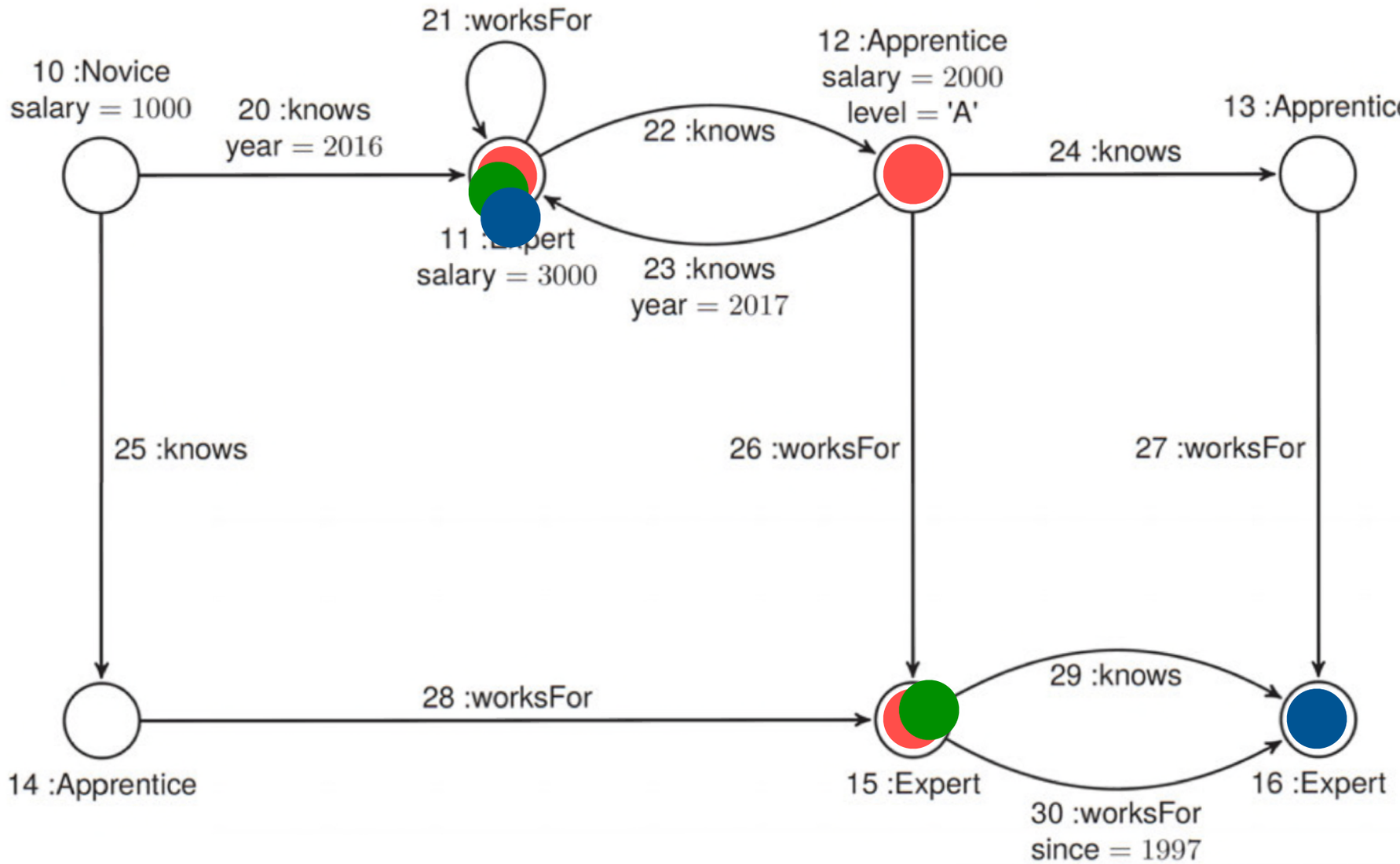
$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”

Check all target vertices of an edge labeled “worksFor” and of an edge labeled “knows”

Check all target vertices of an edge labeled “knows” that are sources for an edge labeled “worksFor”



Example 2

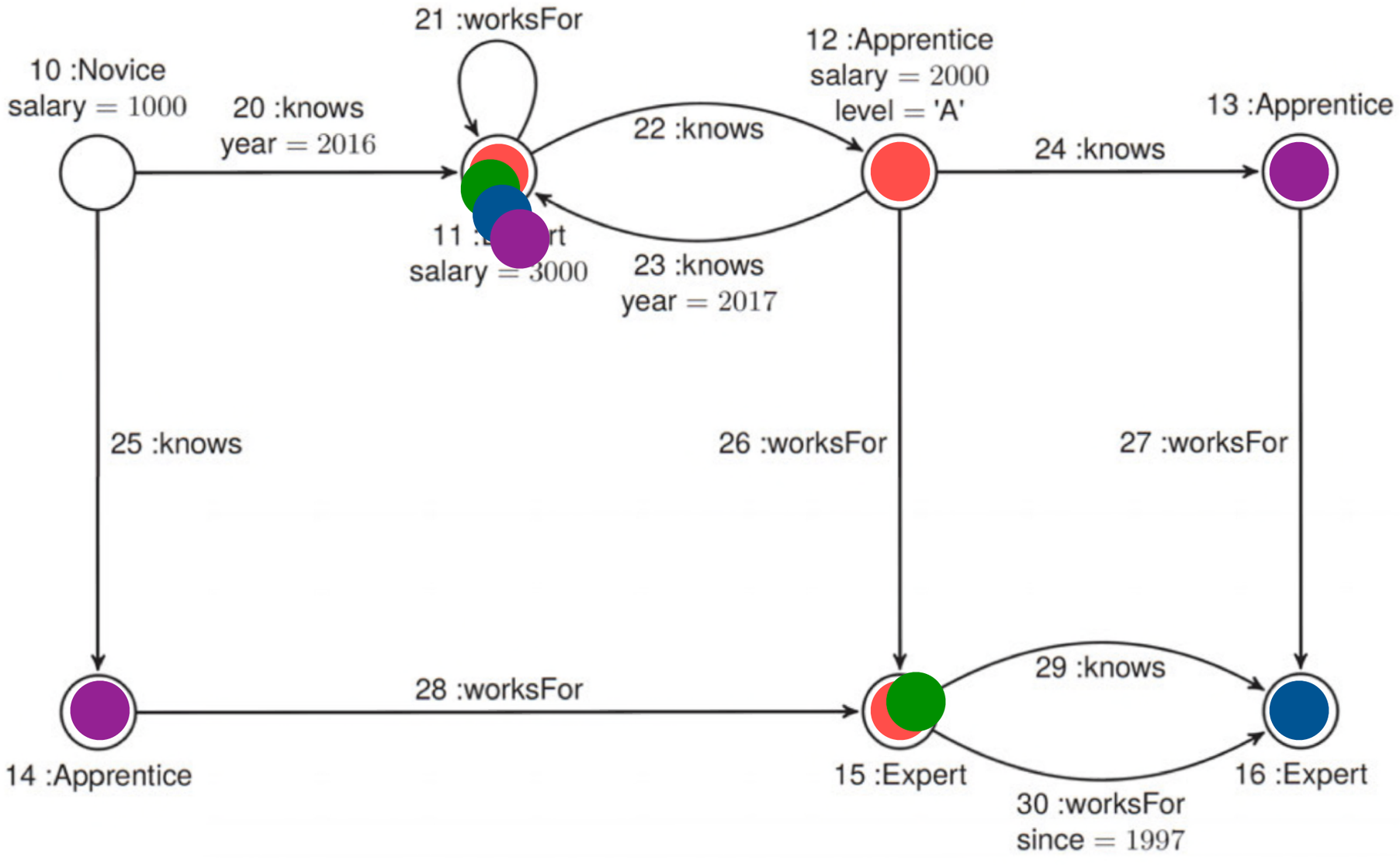
$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”

Check all target vertices of an edge labeled “worksFor” and of an edge labeled “knows”

Check all target vertices of an edge labeled “knows” that are sources for an edge labeled “worksFor”



Example 2

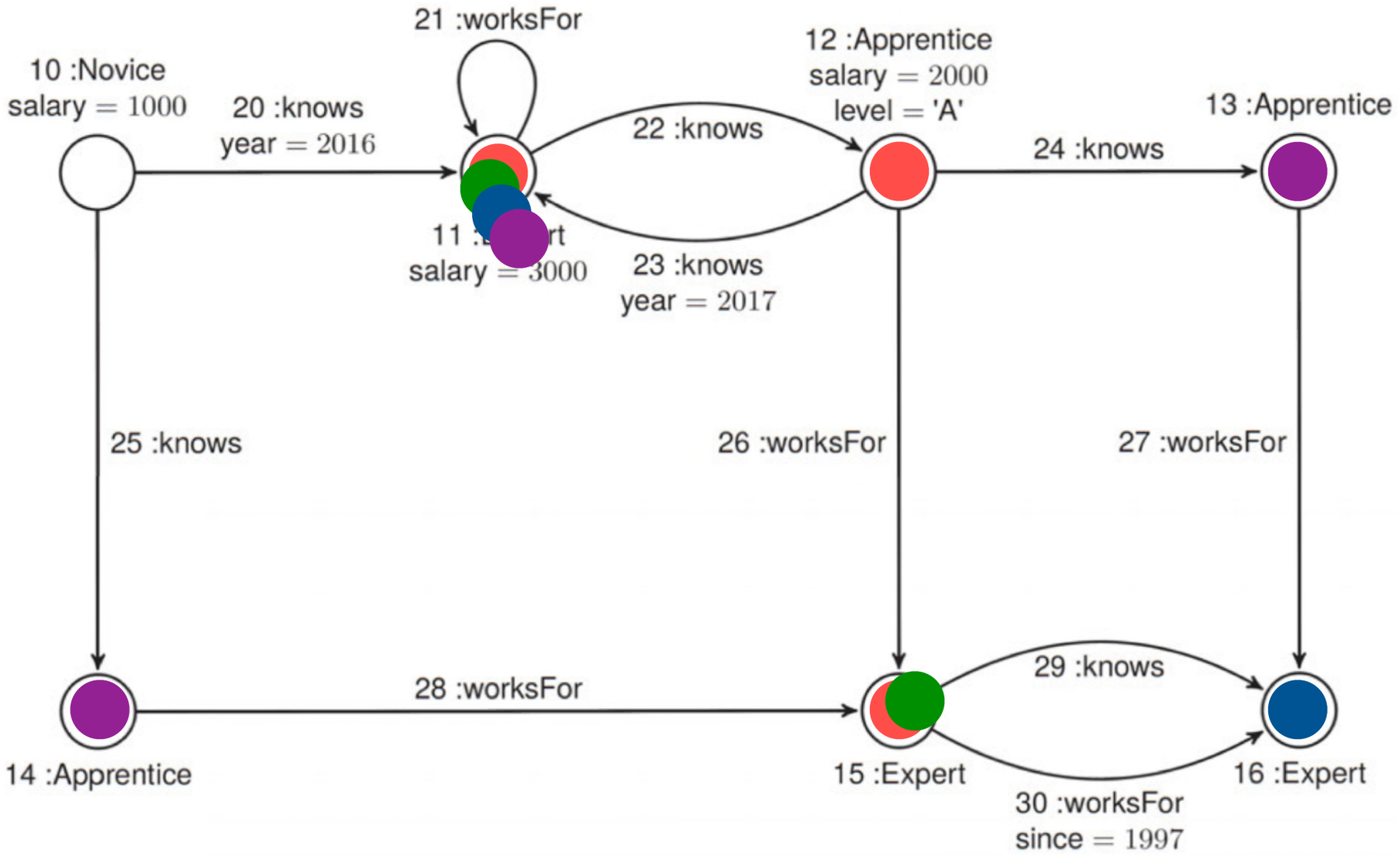
$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

a_1 Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

e_1 Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”

e_2 Check all target vertices of an edge labeled “worksFor” and of an edge labeled “knows”

a_2 Check all target vertices of an edge labeled “knows” that are sources for an edge labeled “worksFor”



Example 2

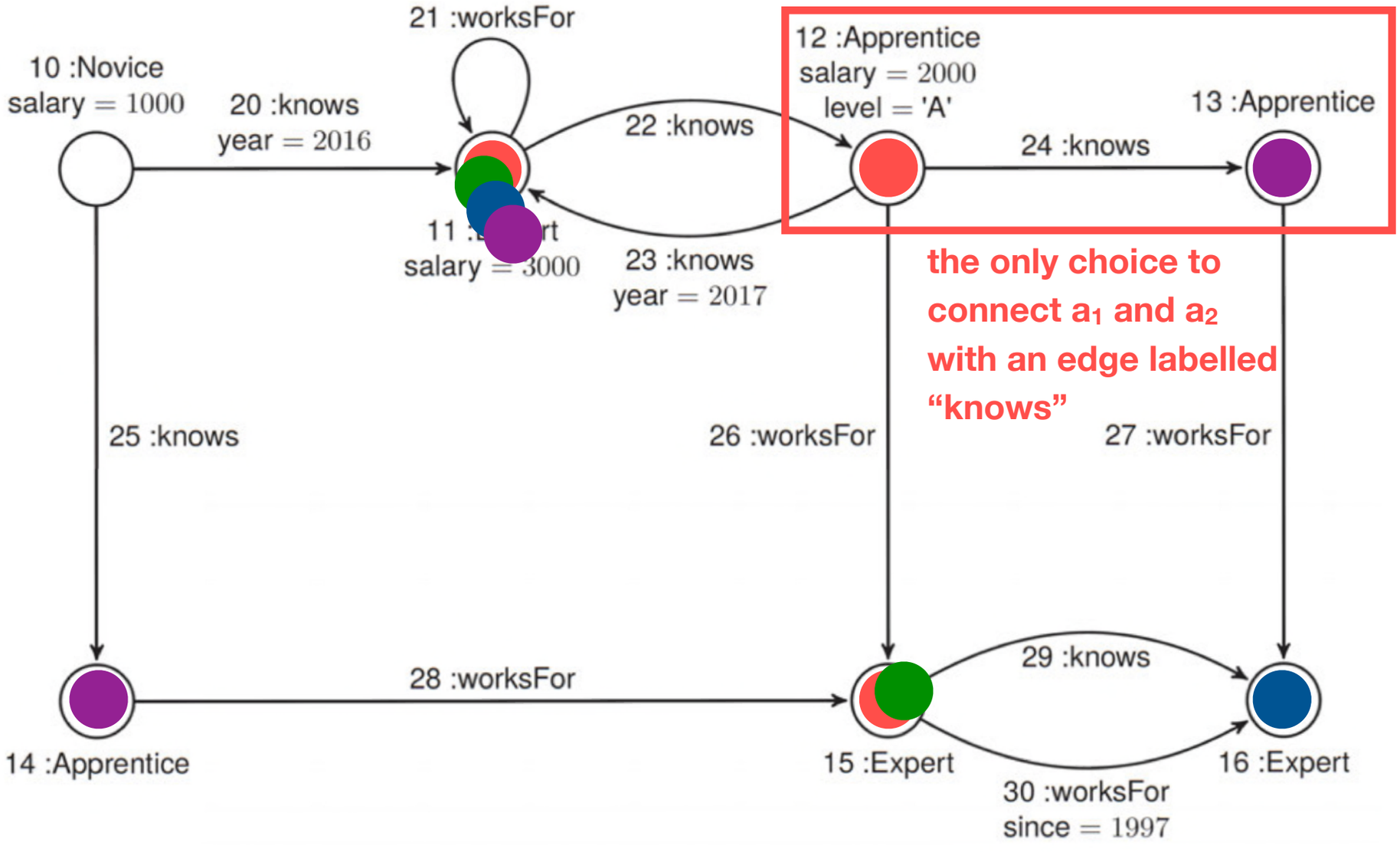
$$q_2 = (a_1, e_2) \leftarrow :\text{knows}(a_1, a_2), :\text{worksFor}(a_1, e_1), :\text{knows}(e_1, e_2), :\text{worksFor}(a_2, e_2)$$

a_1 Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

e_1 Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”

e_2 Check all target vertices of an edge labeled “worksFor” and of an edge labeled “knows”

a_2 Check all target vertices of an edge labeled “knows” that are sources for an edge labeled “worksFor”



Example 2

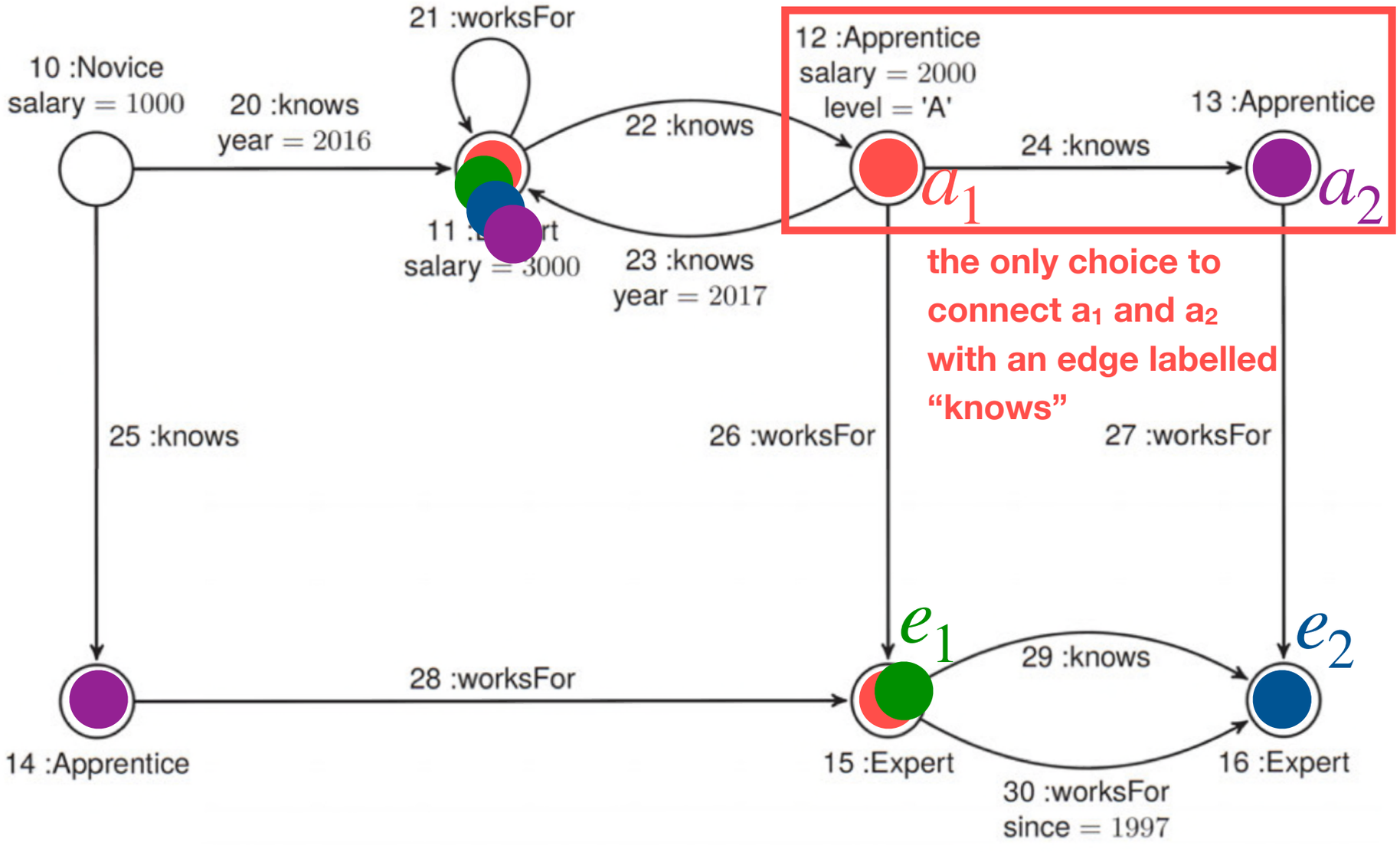
$$q_2 = (a_1, e_2) \leftarrow \text{:knows}(a_1, a_2), \text{:worksFor}(a_1, e_1), \text{:knows}(e_1, e_2), \text{:worksFor}(a_2, e_2)$$

a_1 Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

e_1 Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”

e_2 Check all target vertices of an edge labeled “worksFor” and of an edge labeled “knows”

a_2 Check all target vertices of an edge labeled “knows” that are sources for an edge labeled “worksFor”



Example 2

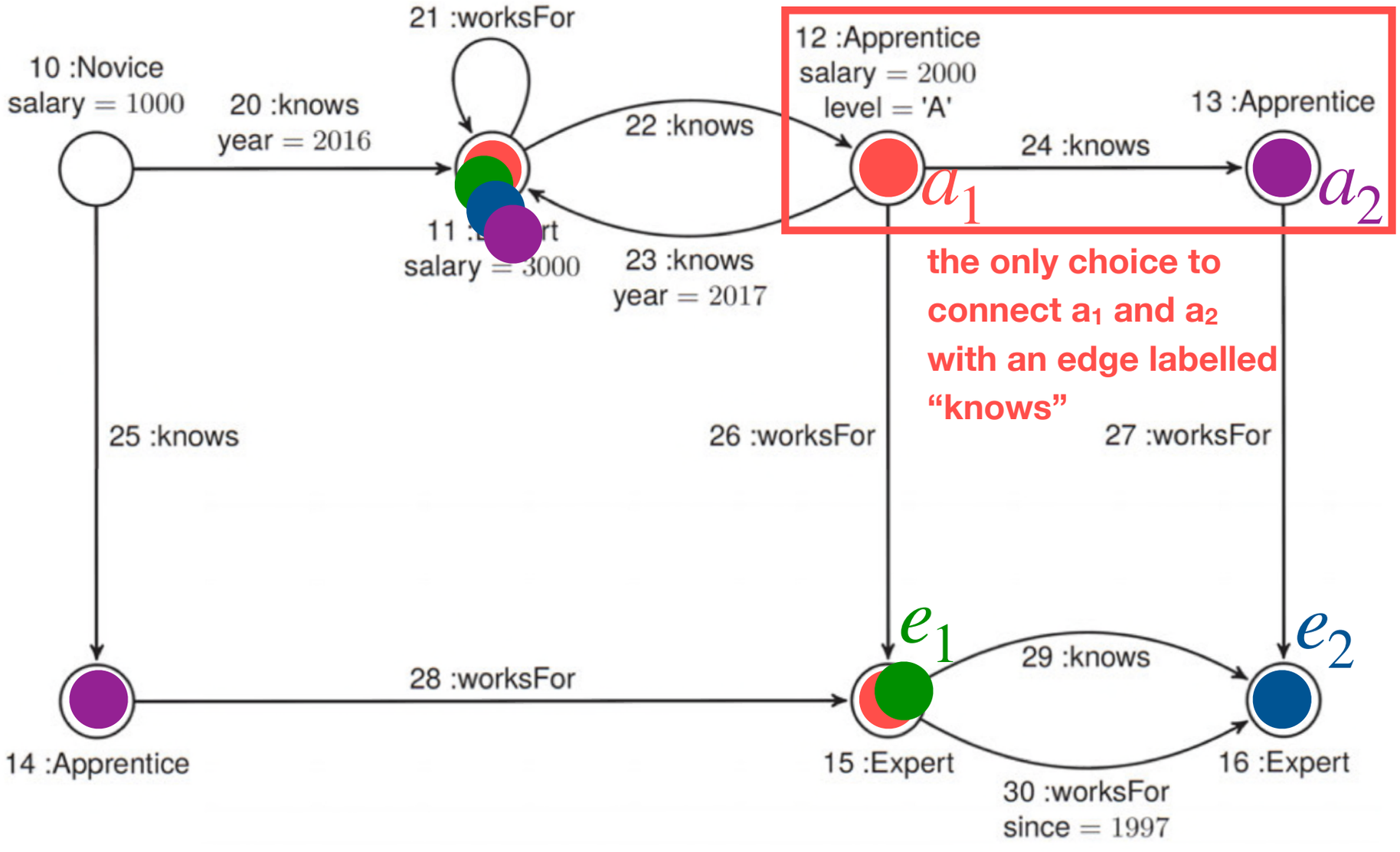
$$q_2 = (a_1, e_2) \leftarrow :\text{knows}(a_1, a_2), :\text{worksFor}(a_1, e_1), :\text{knows}(e_1, e_2), :\text{worksFor}(a_2, e_2)$$

a_1 Check all source vertices for one edge labeled “knows” and one edge labeled “worksFor”

e_1 Check all target vertices for an edge labeled “worksFor” that are also source vertices for an edge labeled “knows”

e_2 Check all target vertices of an edge labeled “worksFor” and of an edge labeled “knows”

a_2 Check all target vertices of an edge labeled “knows” that are sources for an edge labeled “worksFor”



Solution

$$[\![q_2]\!]_{G_{ex}} = (12,16)$$

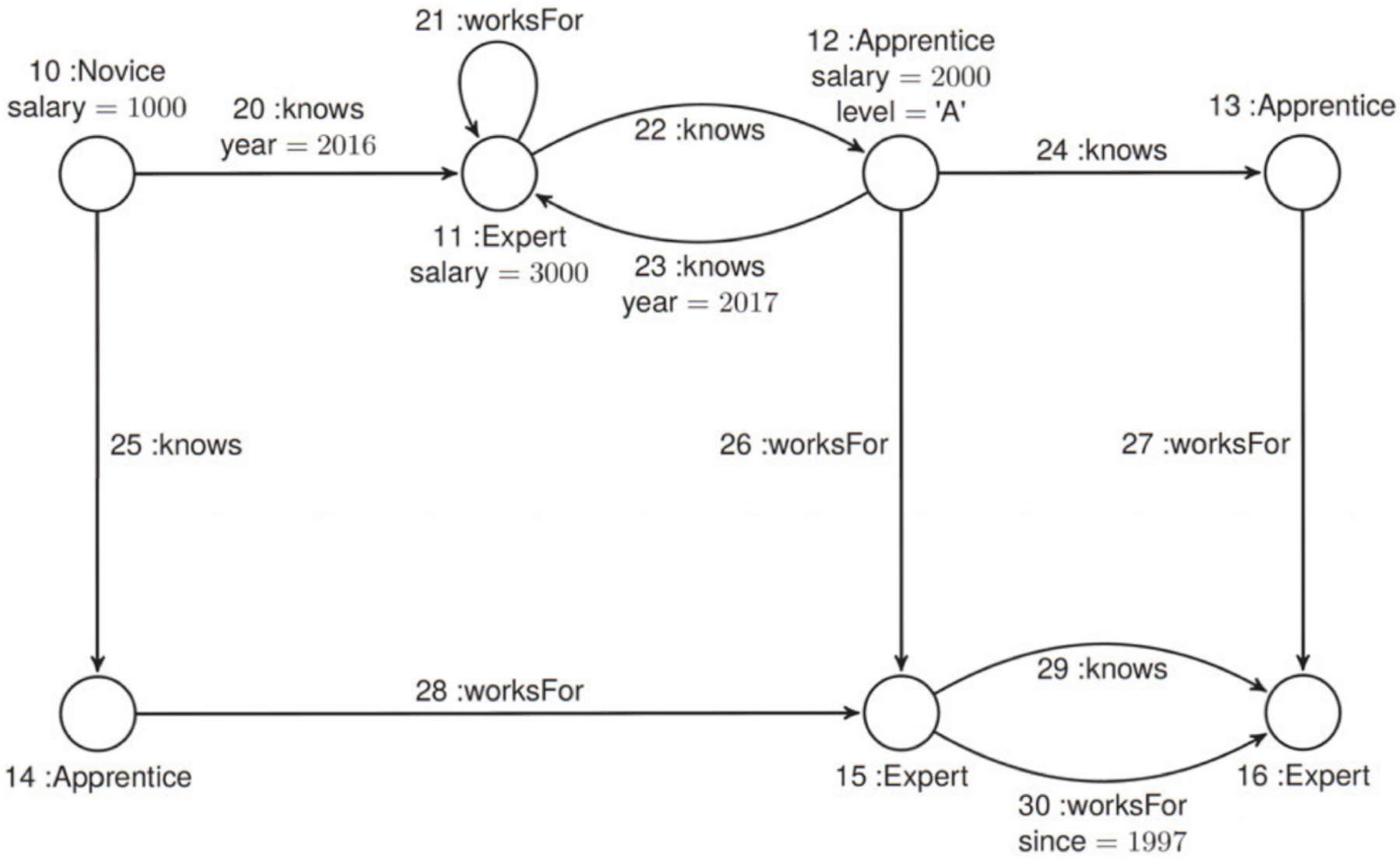
Conjunctive Regular Path Queries (CRPQ)

$(z_1, \ldots, z_m) \leftarrow \alpha_1(x_1, y_1), \ldots, \alpha_n(x_n, y_n)$, where $\alpha_1, \ldots, \alpha_n$ are RQ

Conjunctive Regular Path Queries (CRPQ)

$(z_1, \ldots, z_m) \leftarrow \alpha_1(x_1, y_1), \ldots, \alpha_n(x_n, y_n)$, where $\alpha_1, \ldots, \alpha_n$ are RQ

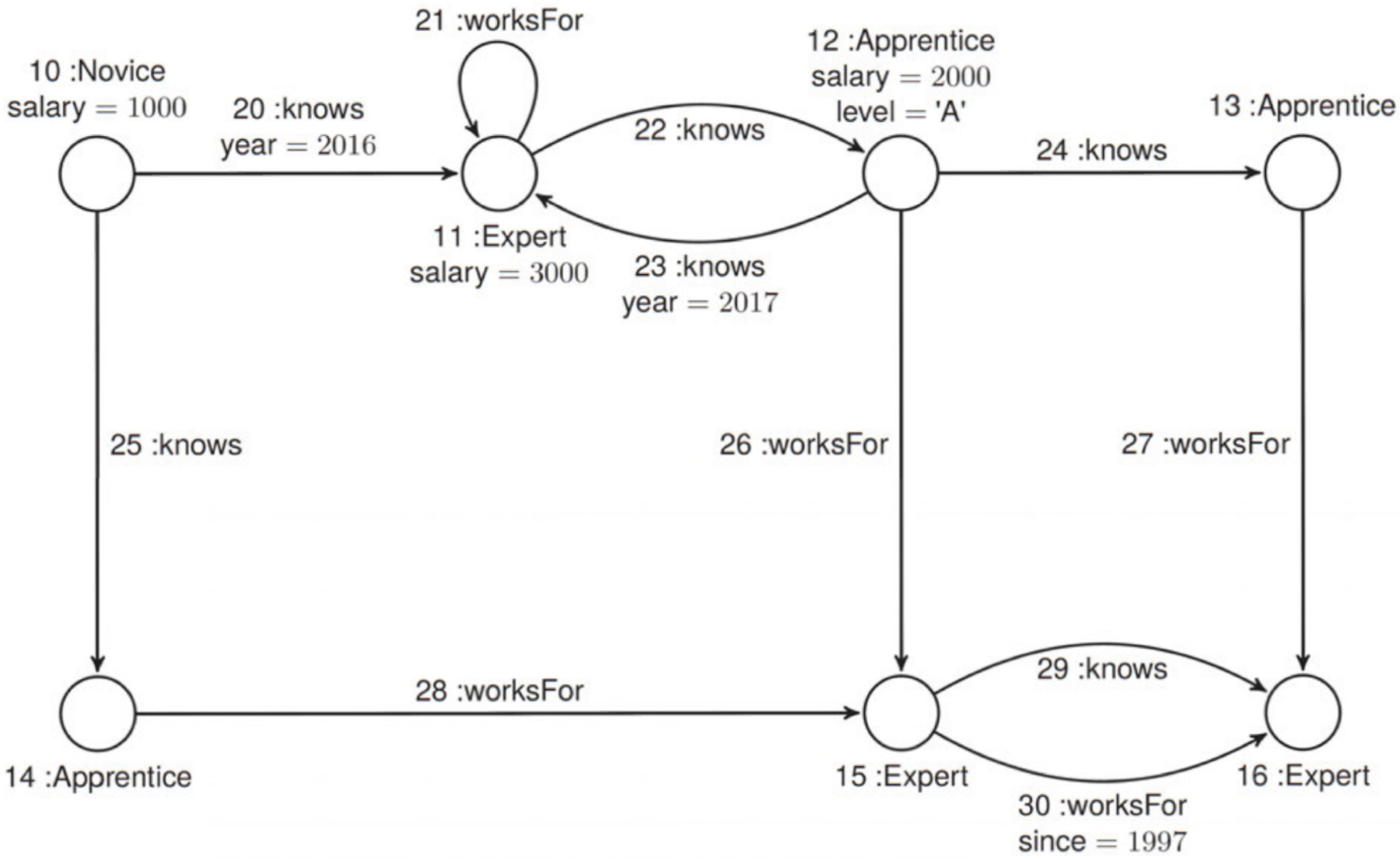
Example $q = (a, b) \leftarrow$:knows/:worksFor/:knows$^+(a, b)$



Conjunctive Regular Path Queries (CRPQ)

$(z_1, \ldots, z_m) \leftarrow \alpha_1(x_1, y_1), \ldots, \alpha_n(x_n, y_n)$, where $\alpha_1, \ldots, \alpha_n$ are RQ

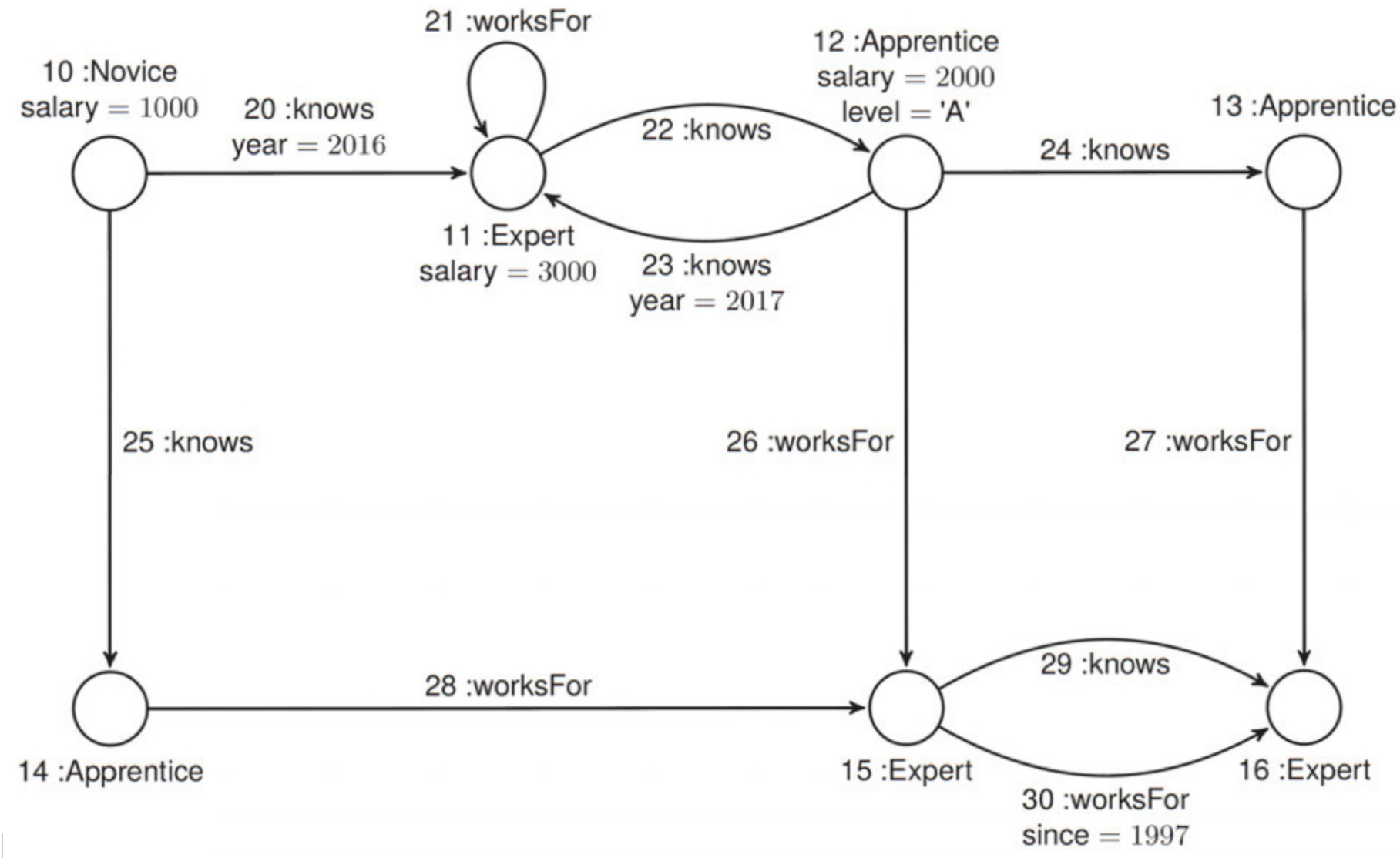
Example $q = (a, b) \leftarrow$:knows/:worksFor/:knows$^+(a, b)$



Solution: it is the same as Example 2 on Slide 9

Example 3

$$q = (n, e) \leftarrow :\text{knows}^{+}(n, a_1), :\text{worksFor}(a_1, e), :\text{knows}(n, a_2), :\text{worksFor}(a_2, e)$$



Example 3

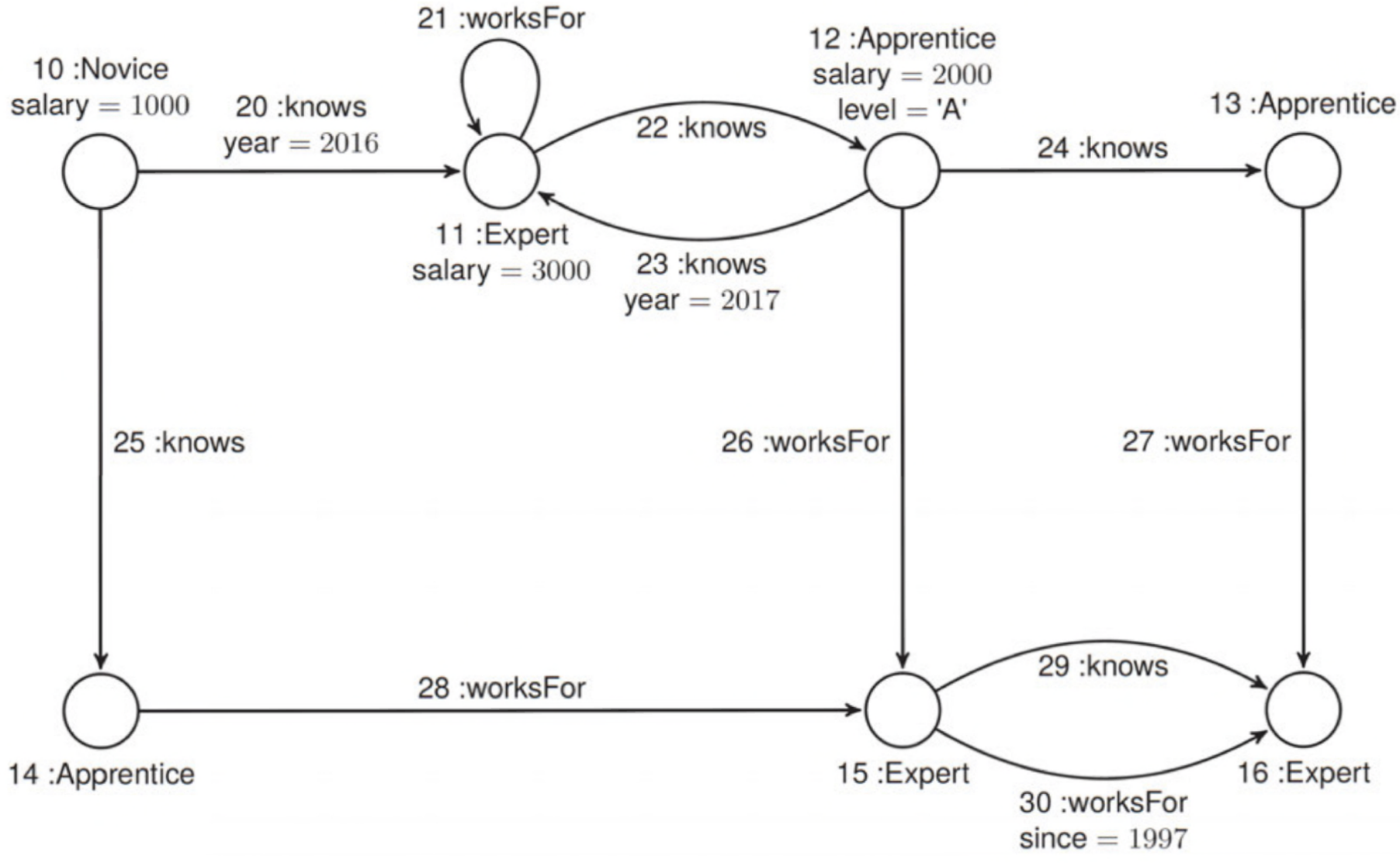
$$q = (n, e) \leftarrow :\text{knows}^{+}(n, a_1), :\text{worksFor}(a_1, e), :\text{knows}(n, a_2), :\text{worksFor}(a_2, e)$$

$n = \{10,11,12\}$

$a_1 = \{11,13,12,14\}$

$a_2 = \{14,11,12\}$

$e = \{11,16,15\}$



Example 3

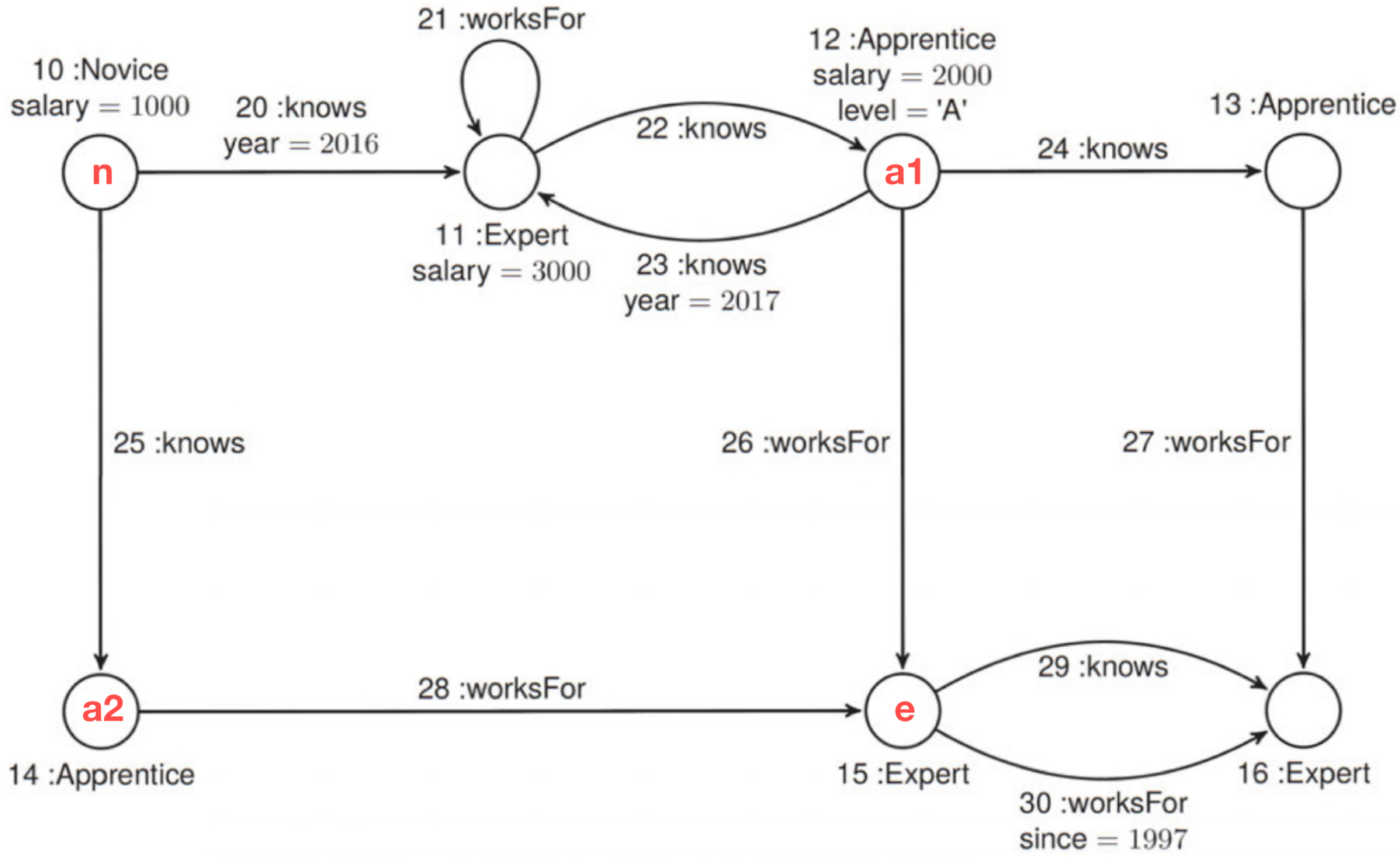
$$q = (n, e) \leftarrow :\text{knows}^{+}(n, a_1), :\text{worksFor}(a_1, e), :\text{knows}(n, a_2), :\text{worksFor}(a_2, e)$$

$n = \{10,11,12\}$

$a_1 = \{11,13,12,14\}$

$a_2 = \{14,11,12\}$

$e = \{11,16,15\}$



Example 3

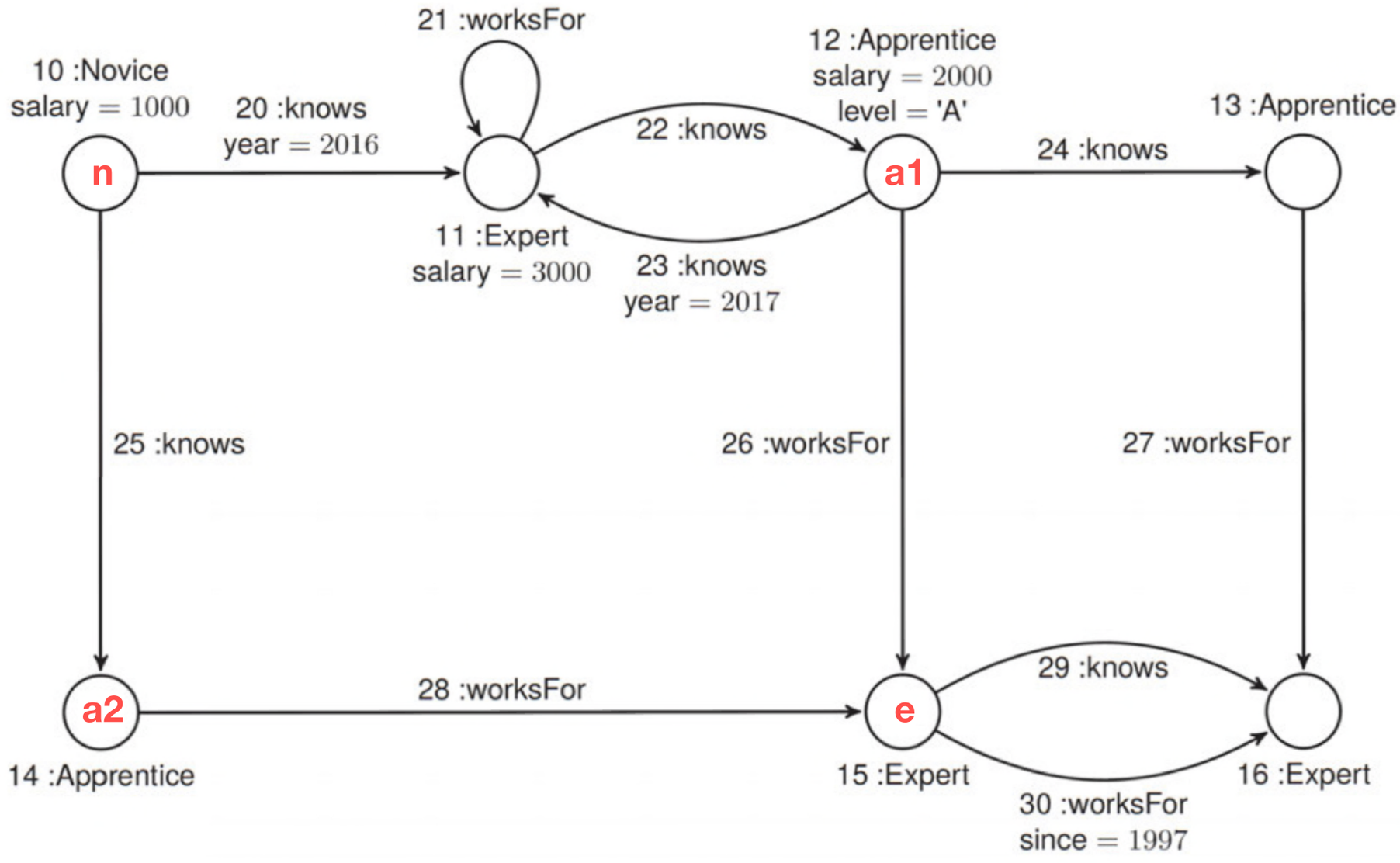
$$q = (n, e) \leftarrow :\text{knows}^{+}(n, a_1), :\text{worksFor}(a_1, e), :\text{knows}(n, a_2), :\text{worksFor}(a_2, e)$$

$n = \{10,11,12\}$

$a_1 = \{11,13,12,14\}$

$a_2 = \{14,11,12\}$

$e = \{11,16,15\}$



$[\![q]\!]_{G_{ex}} = \{(10,15)\}$

Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

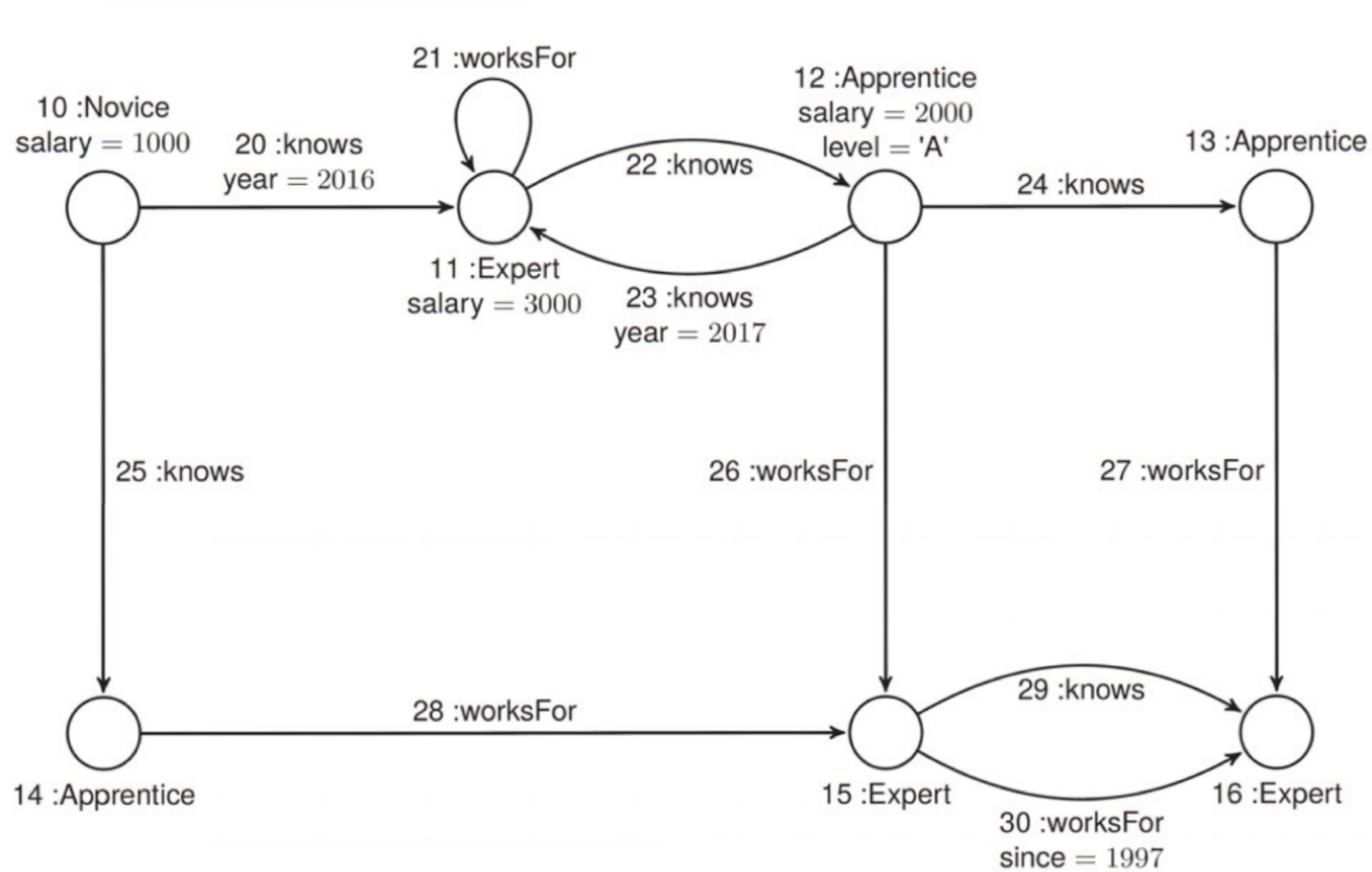
Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

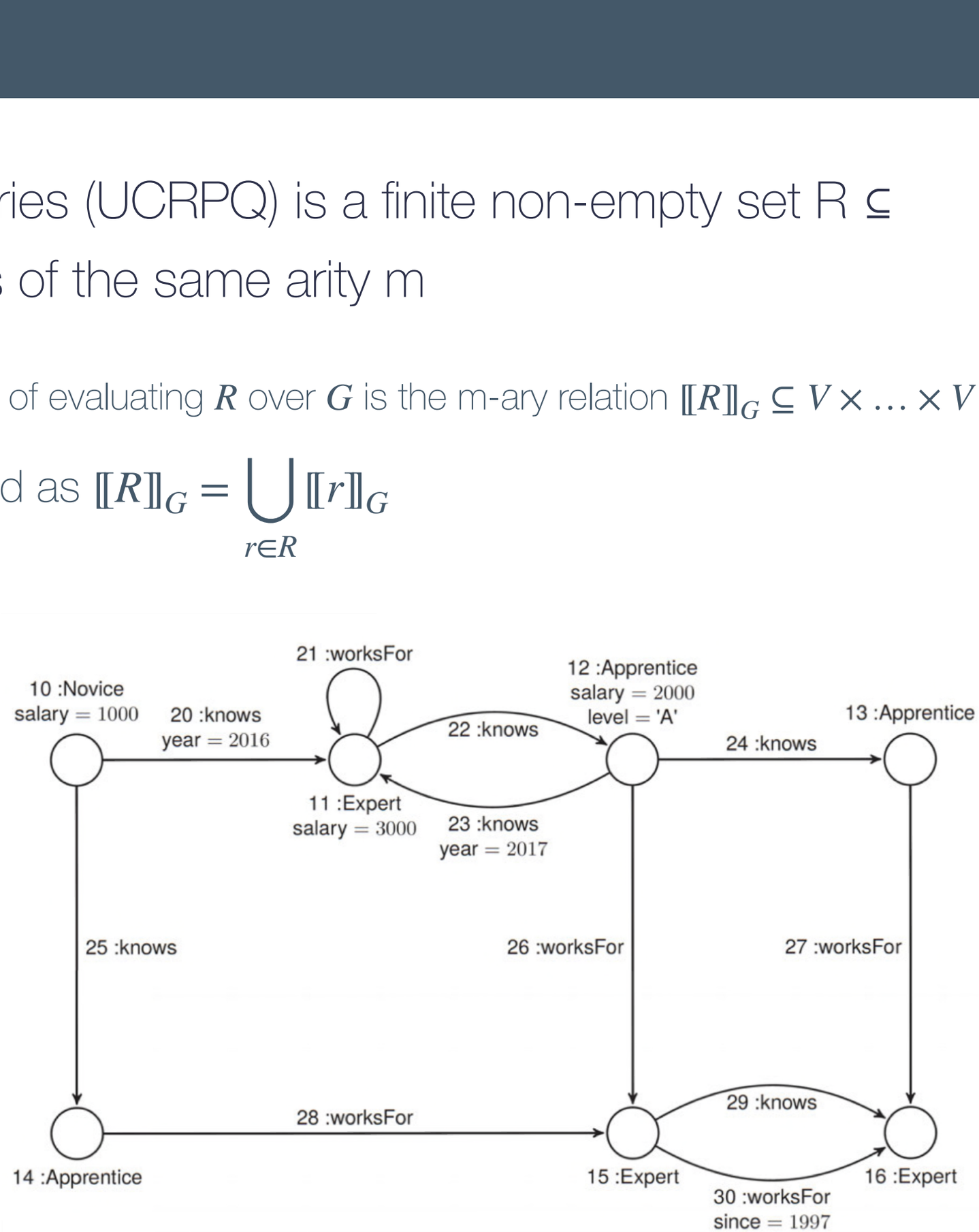
defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

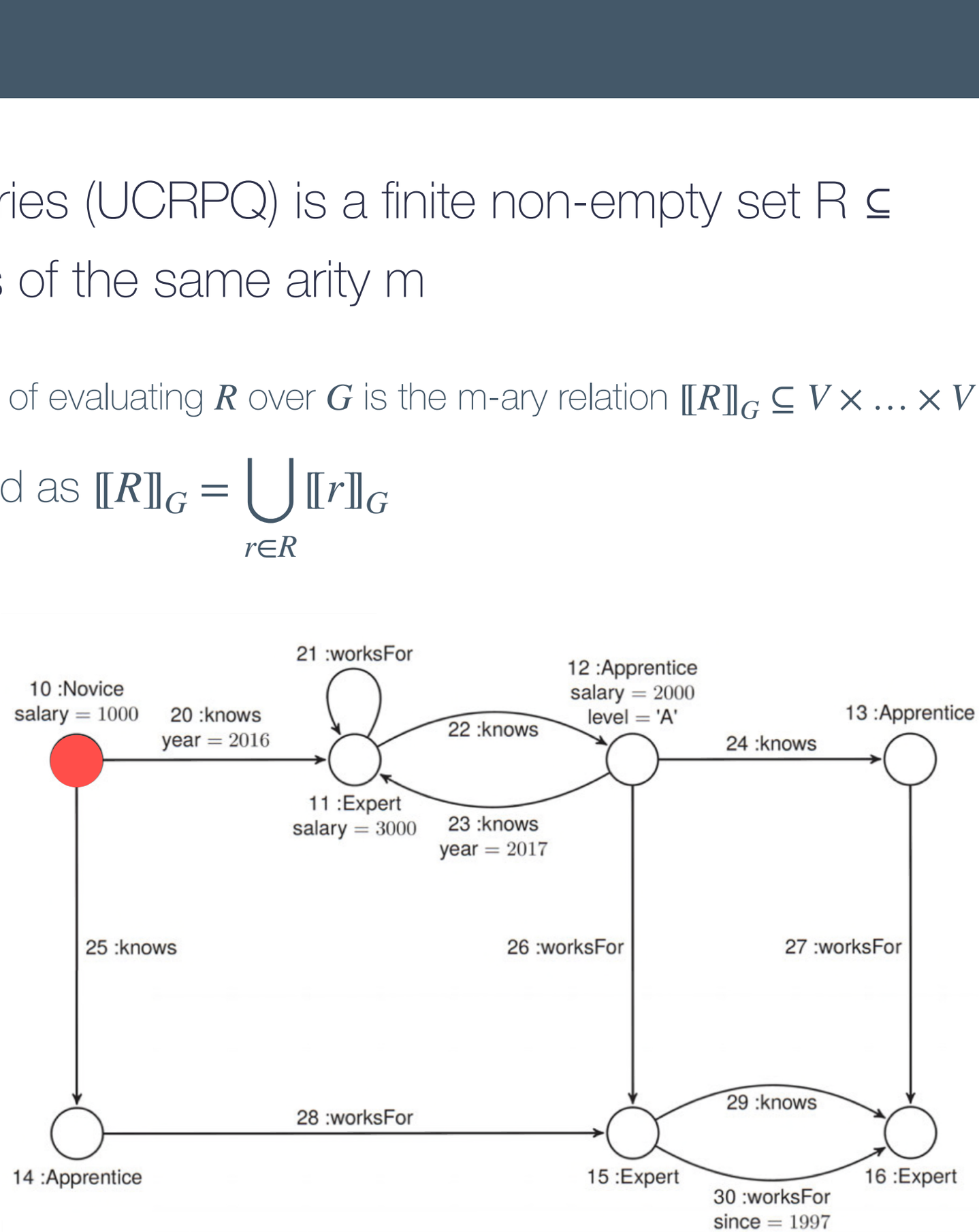
defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

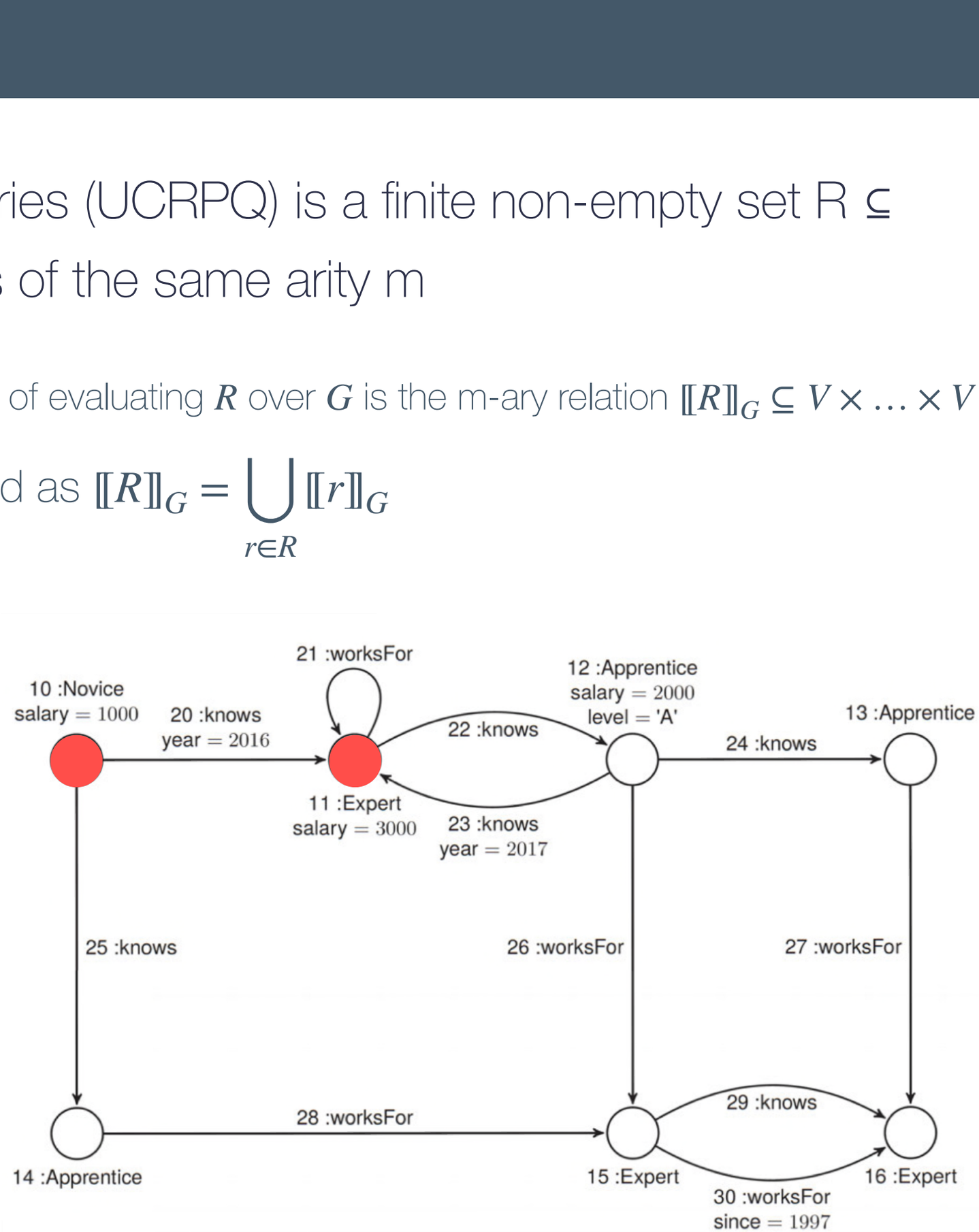
defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

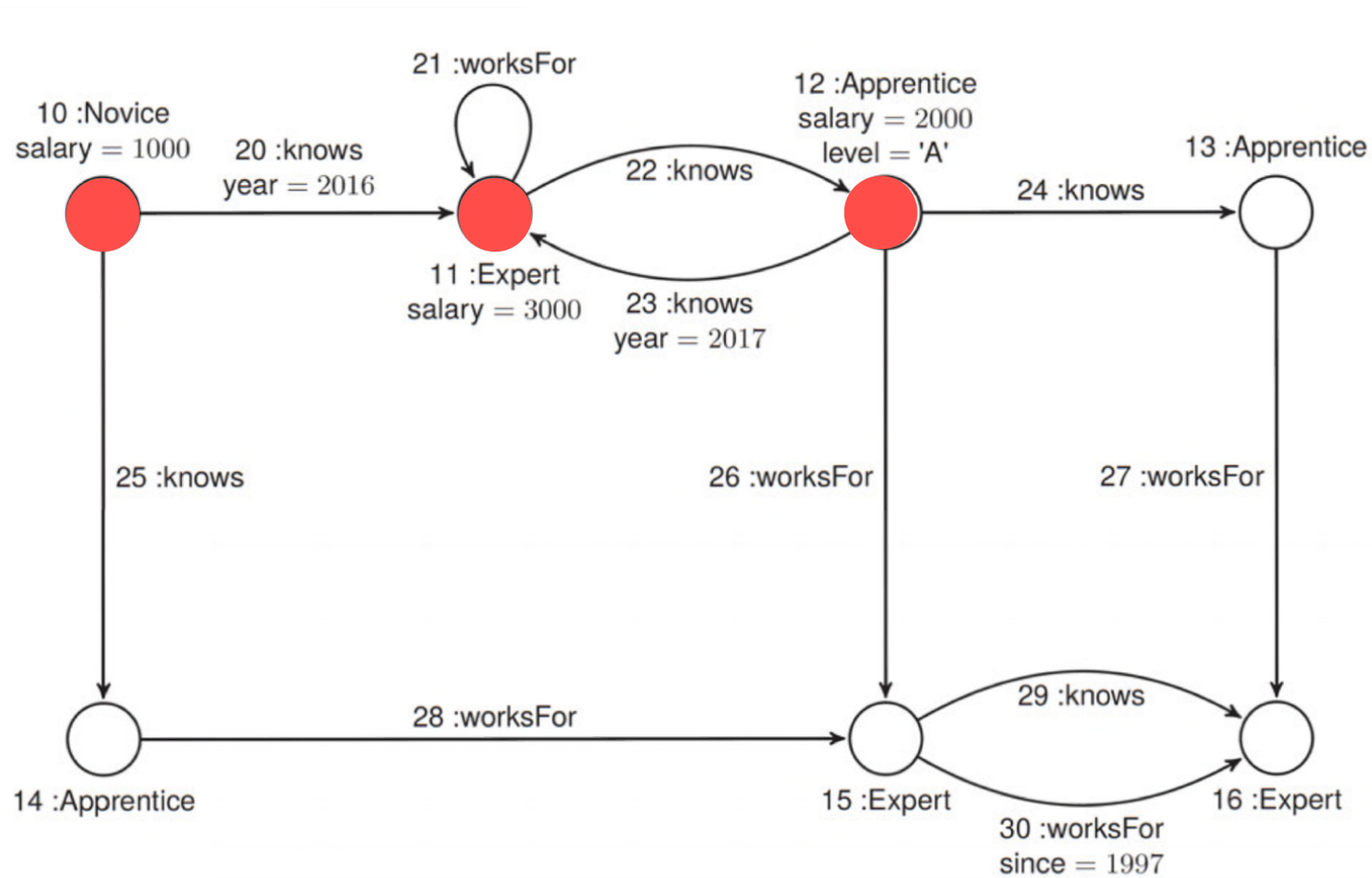
defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

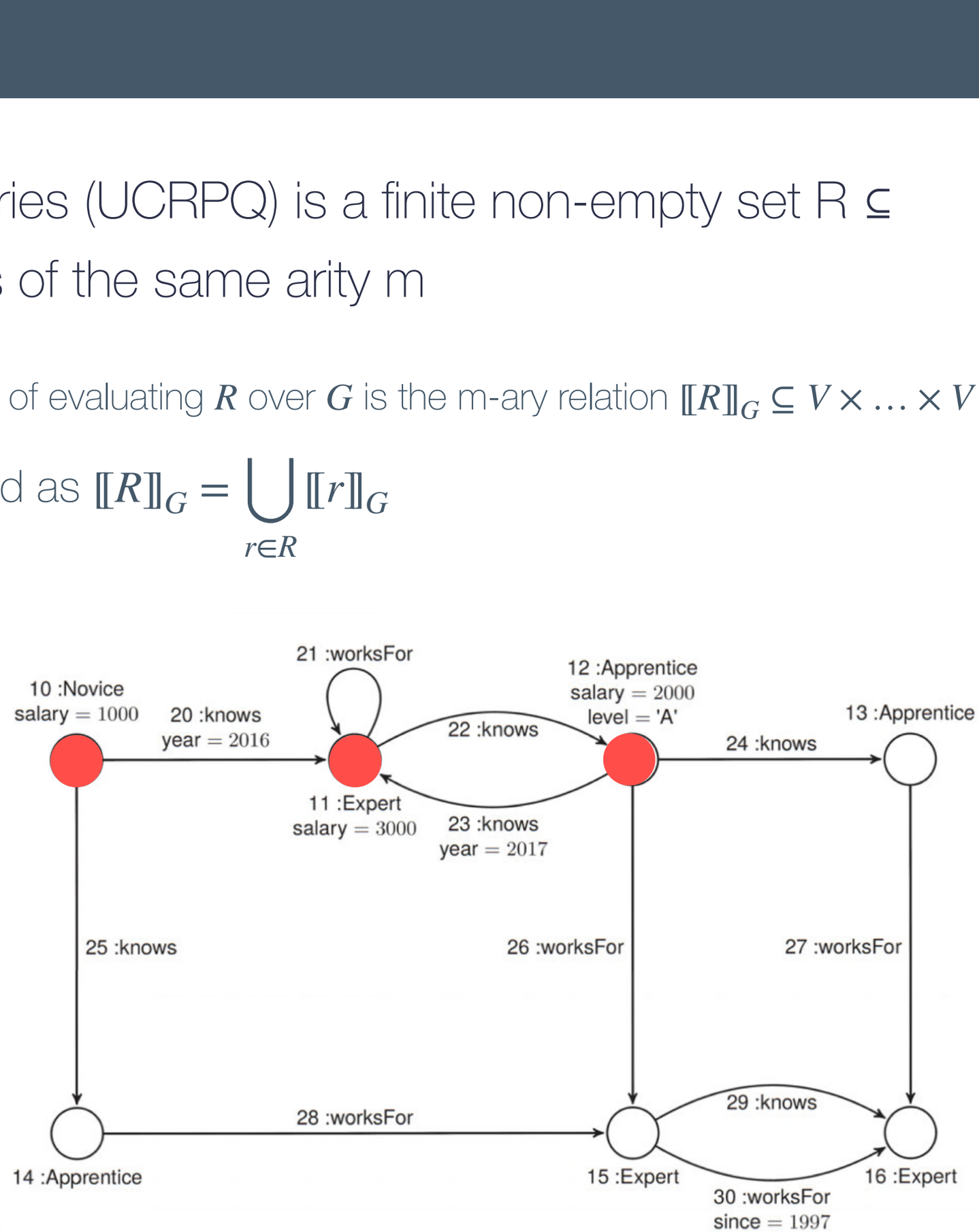
Example

$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)

$x = \{10,11,12\}$



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

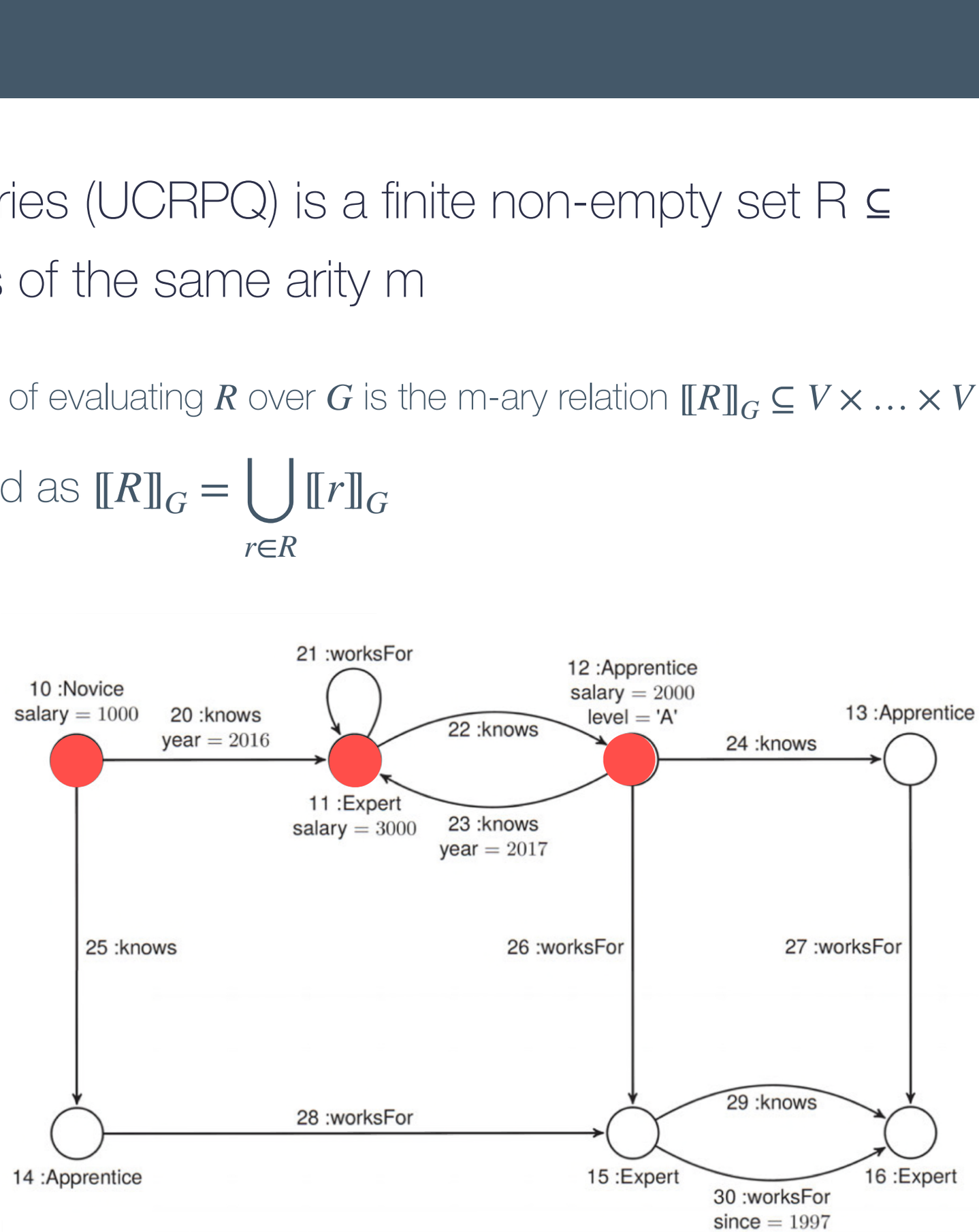
$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)

$x = \{10,11,12\}$

$(x) \leftarrow$:worksFor/:worksFor(x, y)



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

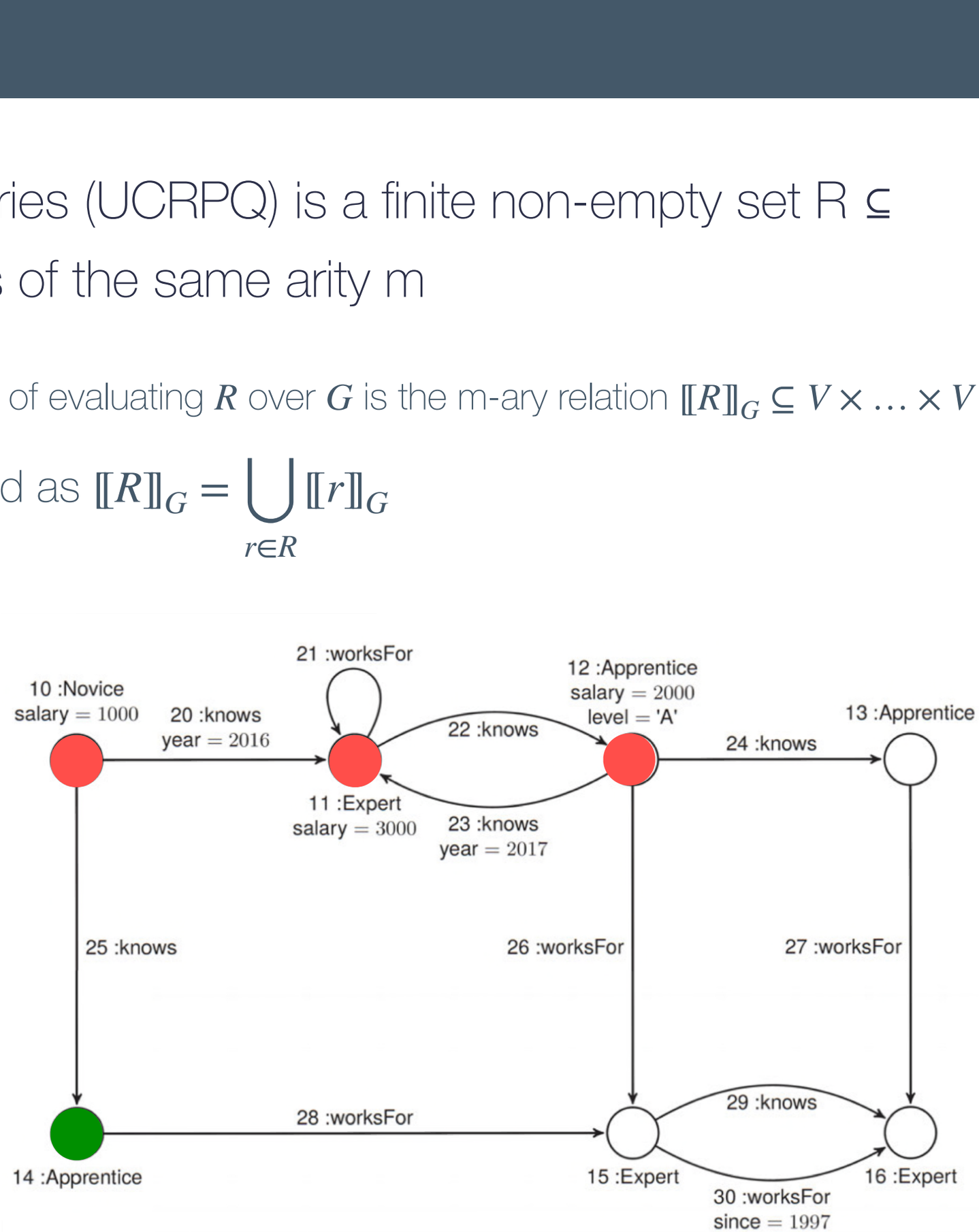
$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)

$x = \{10,11,12\}$

$(x) \leftarrow$:worksFor/:worksFor(x, y)



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

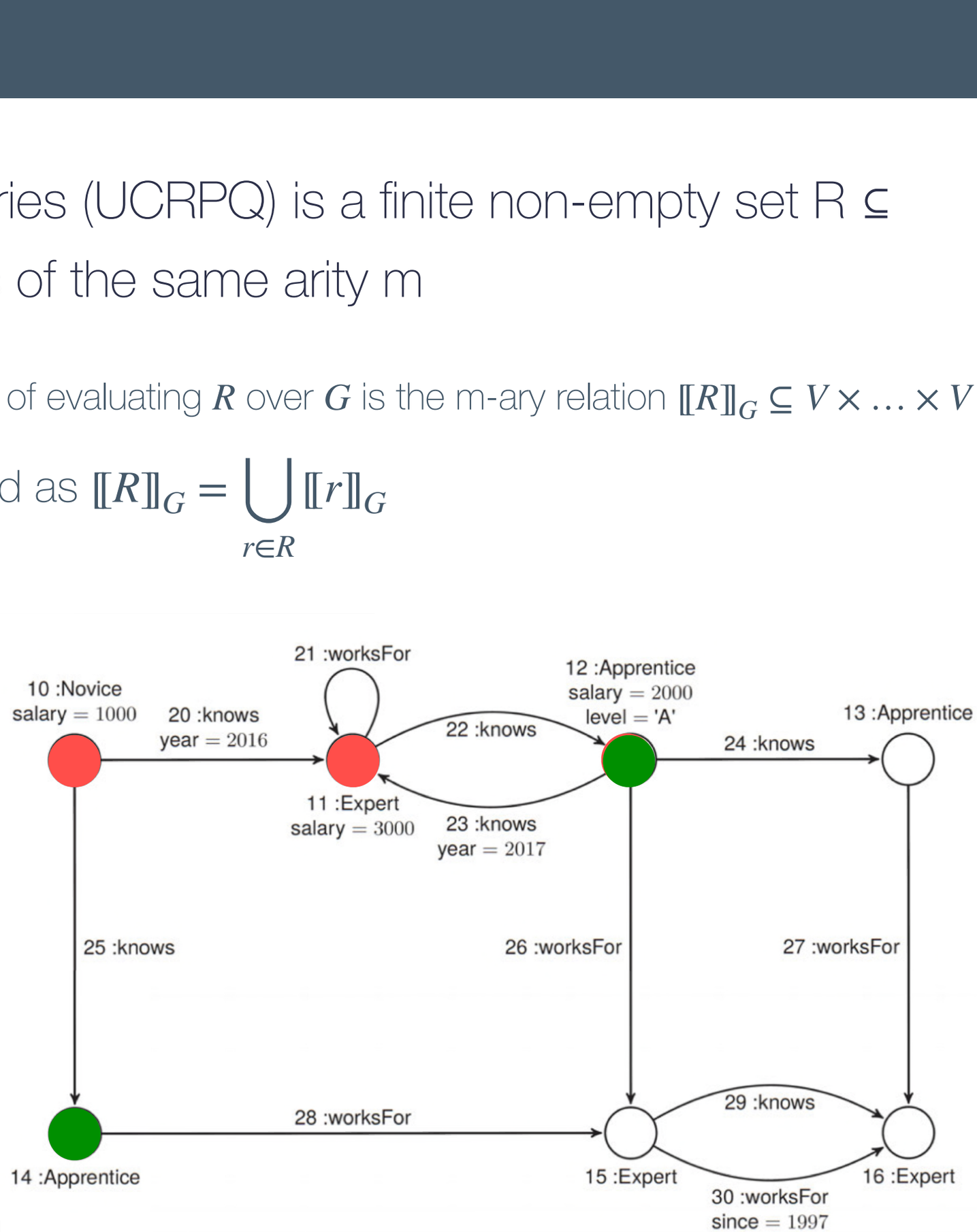
$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)

$x = \{10,11,12\}$

$(x) \leftarrow$:worksFor/:worksFor(x, y)



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

$(x) \leftarrow$:knows/:worksFor(x, y)

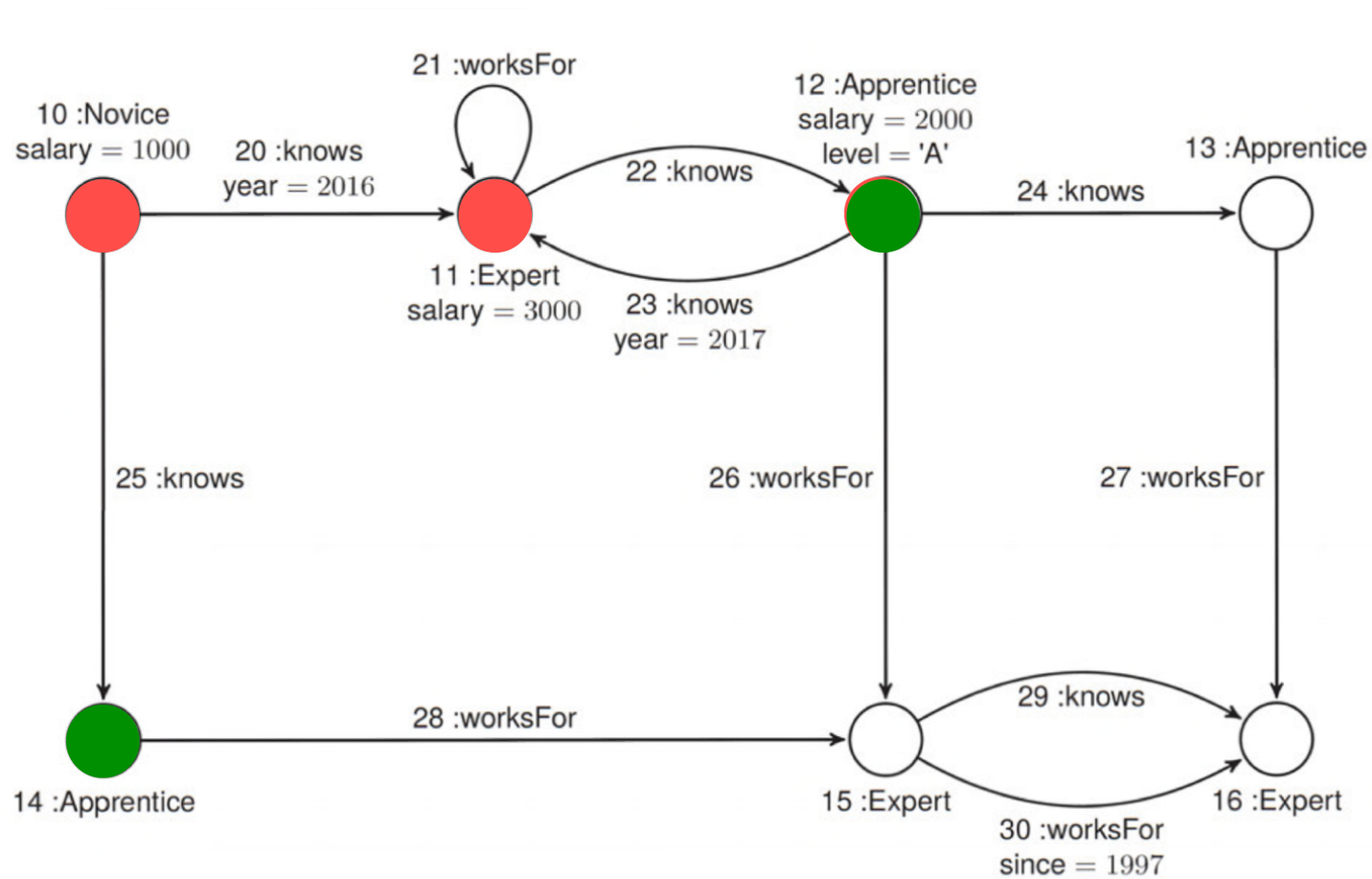
$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)

$x = \{10,11,12\}$

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$x = \{12,14\}$



Union of CRP queries

Union of conjunctive regular queries (UCRPQ) is a finite non-empty set R ⊆ CRPQ. each element of which is of the same arity m

Given $R \in$ UCRPQ of arity m, the semantics of evaluating R over G is the m-ary relation $[\![R]\!]_G \subseteq V \times \ldots \times V$

defined as $[\![R]\!]_G = \bigcup_{r \in R} [\![r]\!]_G$

Example

$(x) \leftarrow$:knows/:worksFor(x, y)

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$(x) \leftarrow$:knows/:worksFor(x, y)

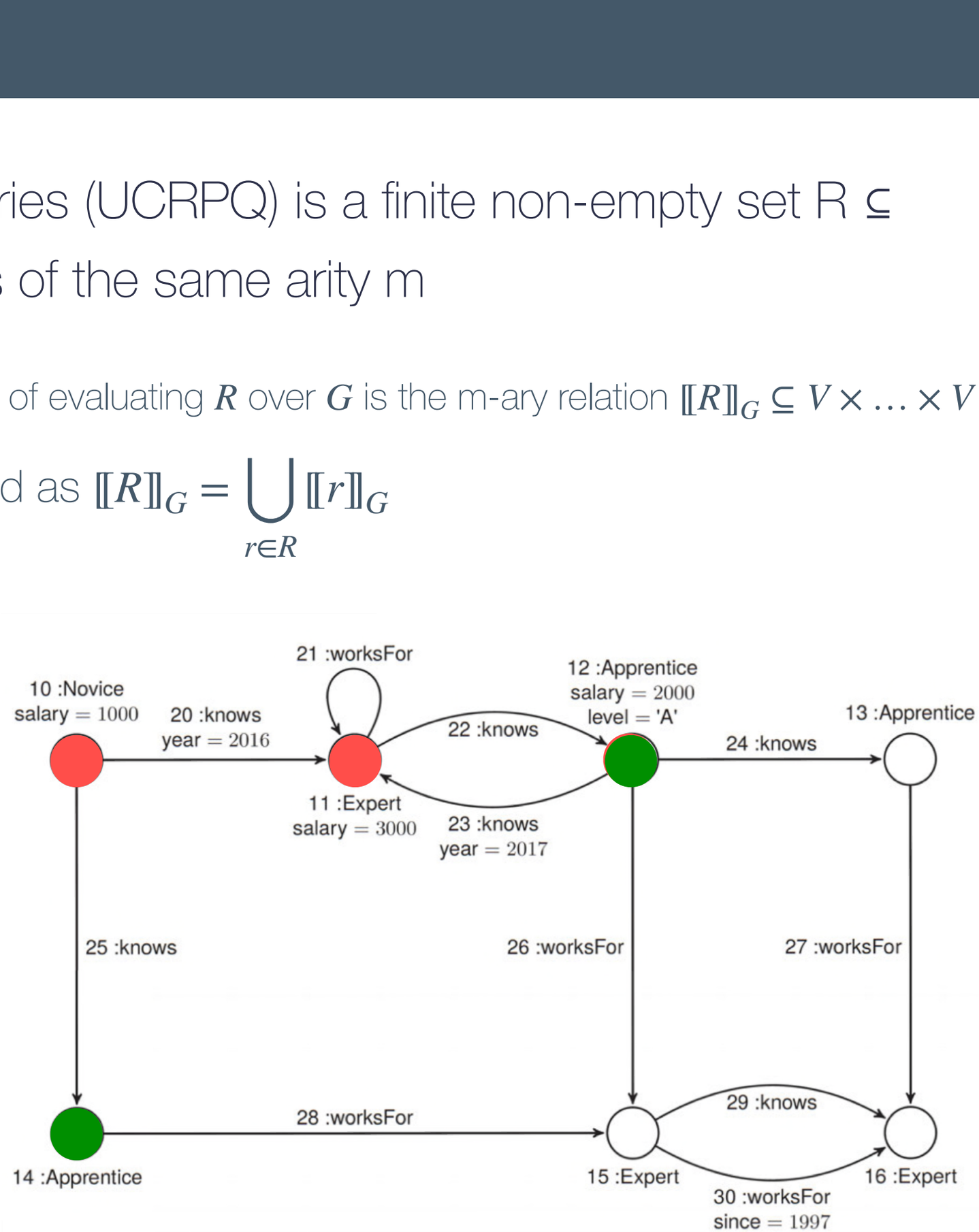
$x = \{10,11,12\}$

$(x) \leftarrow$:worksFor/:worksFor(x, y)

$x = \{12,14\}$

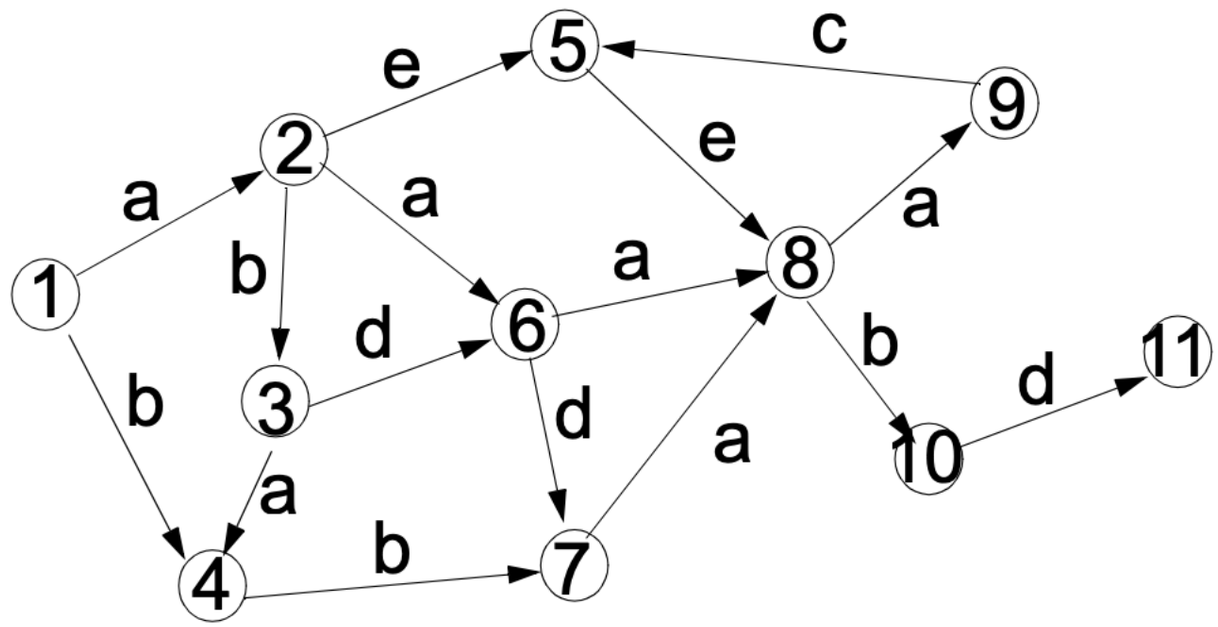
Solution

$x = \{10,11,12,14\}$



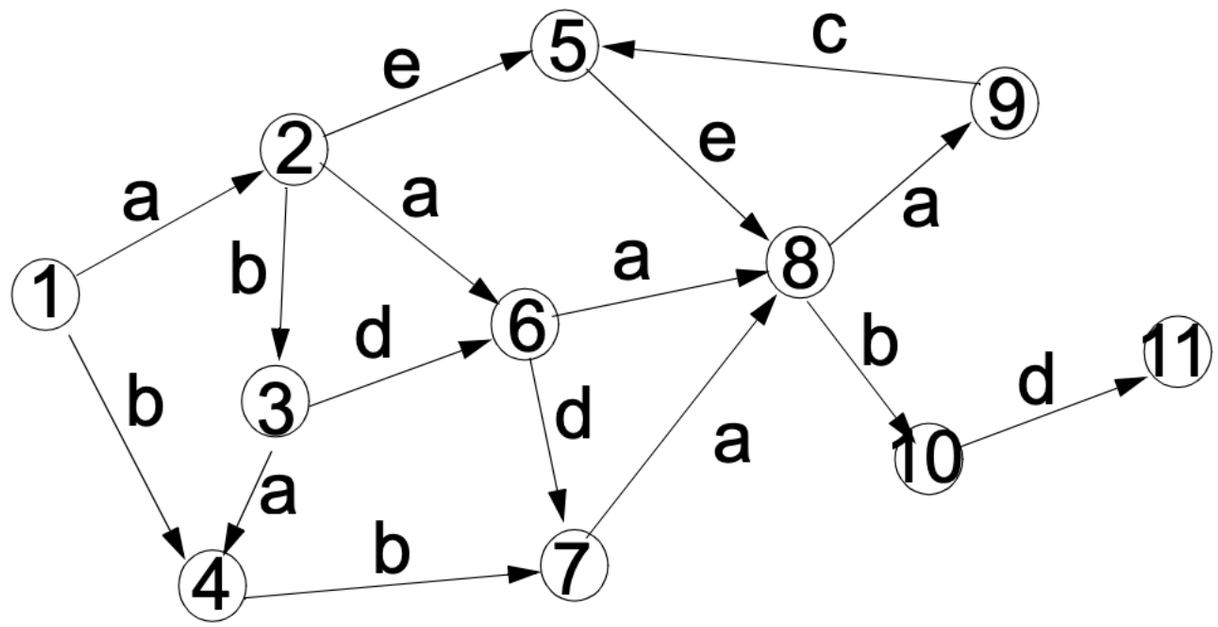
Another example of CRPQ

$$(x, y) \leftarrow a^+(x, y), e^+(x, y)$$



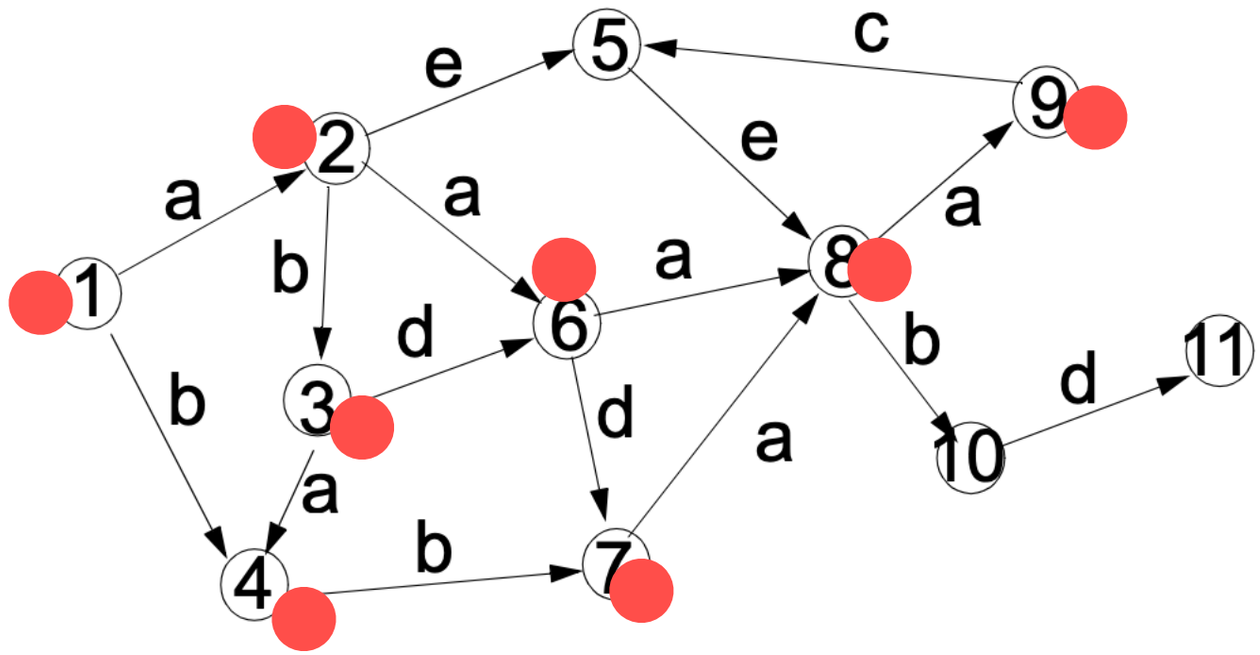
Another example of CRPQ

$$(x, y) \leftarrow a^+(x, y), e^+(x, y)$$



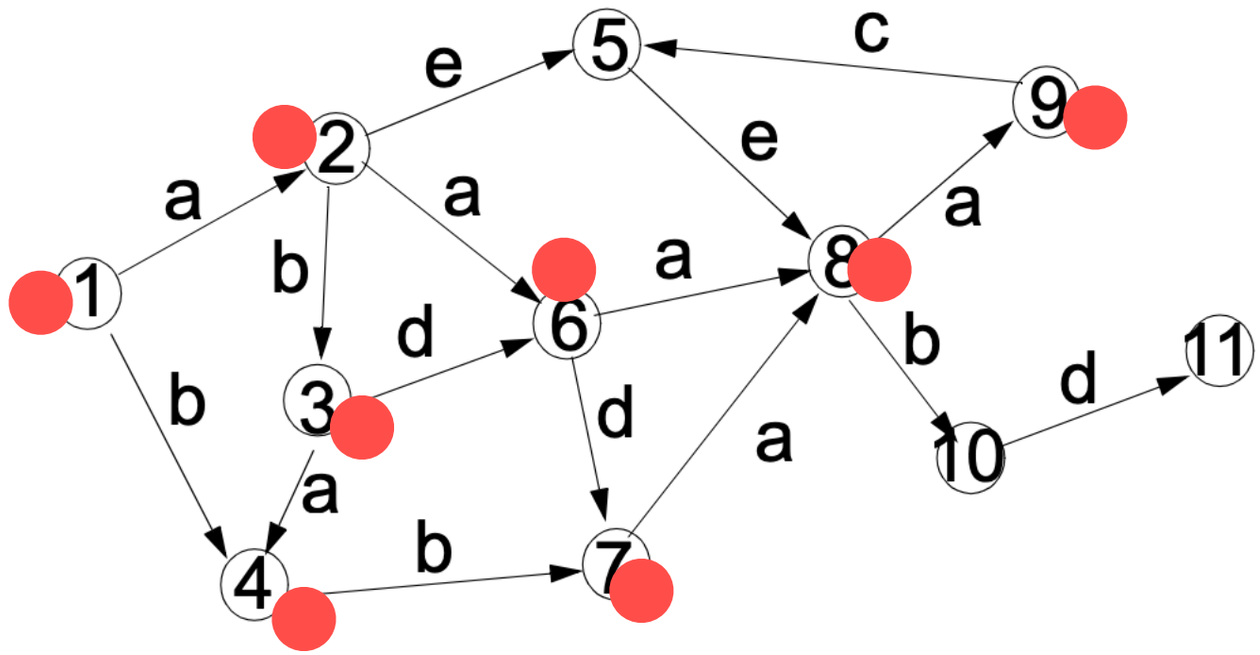
Another example of CRPQ

$$(x, y) \leftarrow a^+(x, y), e^+(x, y)$$



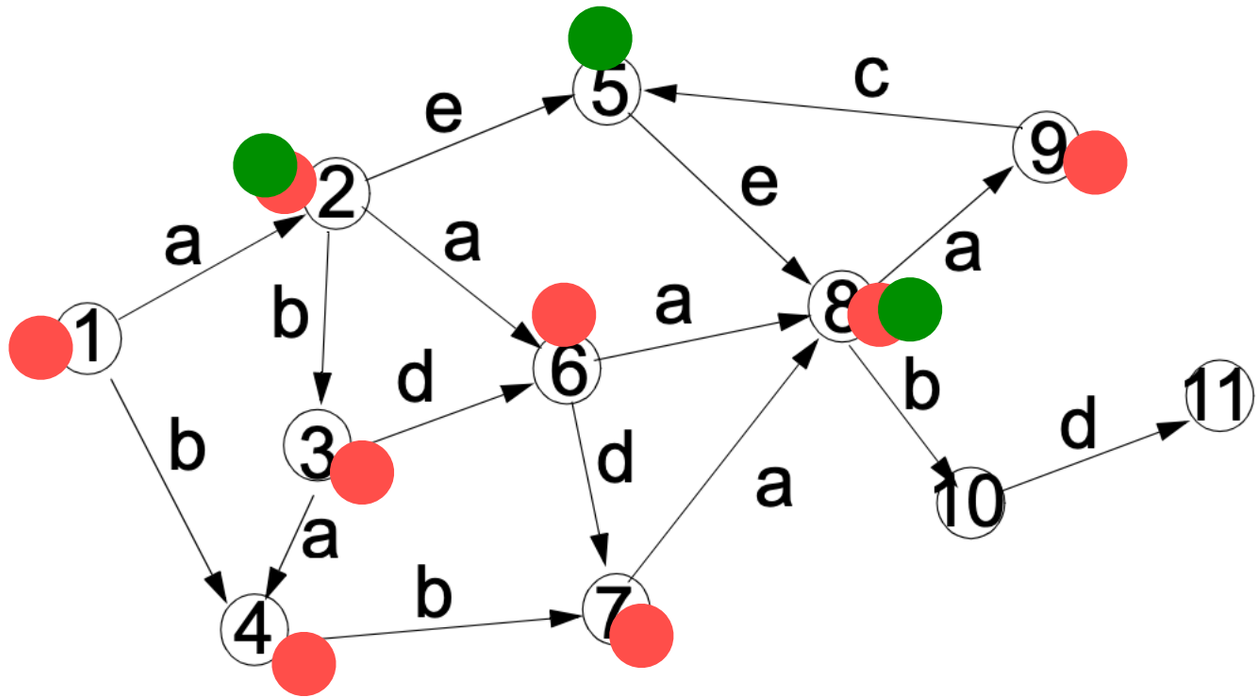
Another example of CRPQ

$$(x, y) \leftarrow a^+(x, y), e^+(x, y)$$



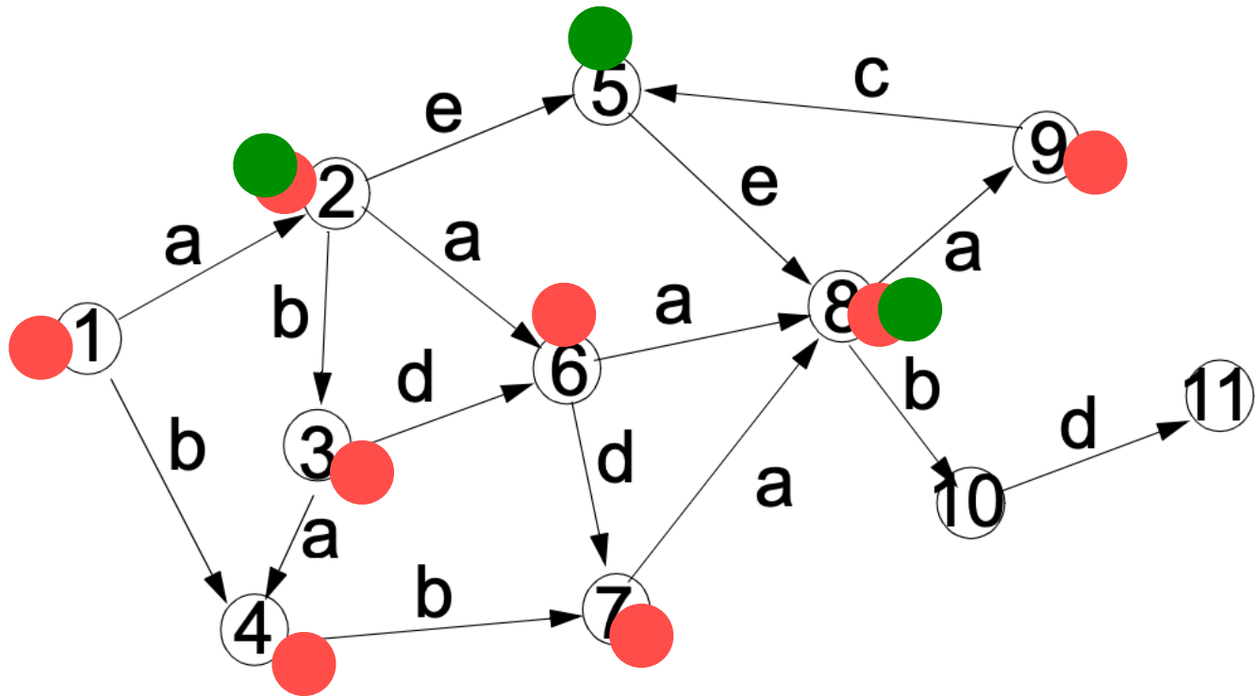
Another example of CRPQ

$$(x, y) \leftarrow a^+(x, y), e^+(x, y)$$



Another example of CRPQ

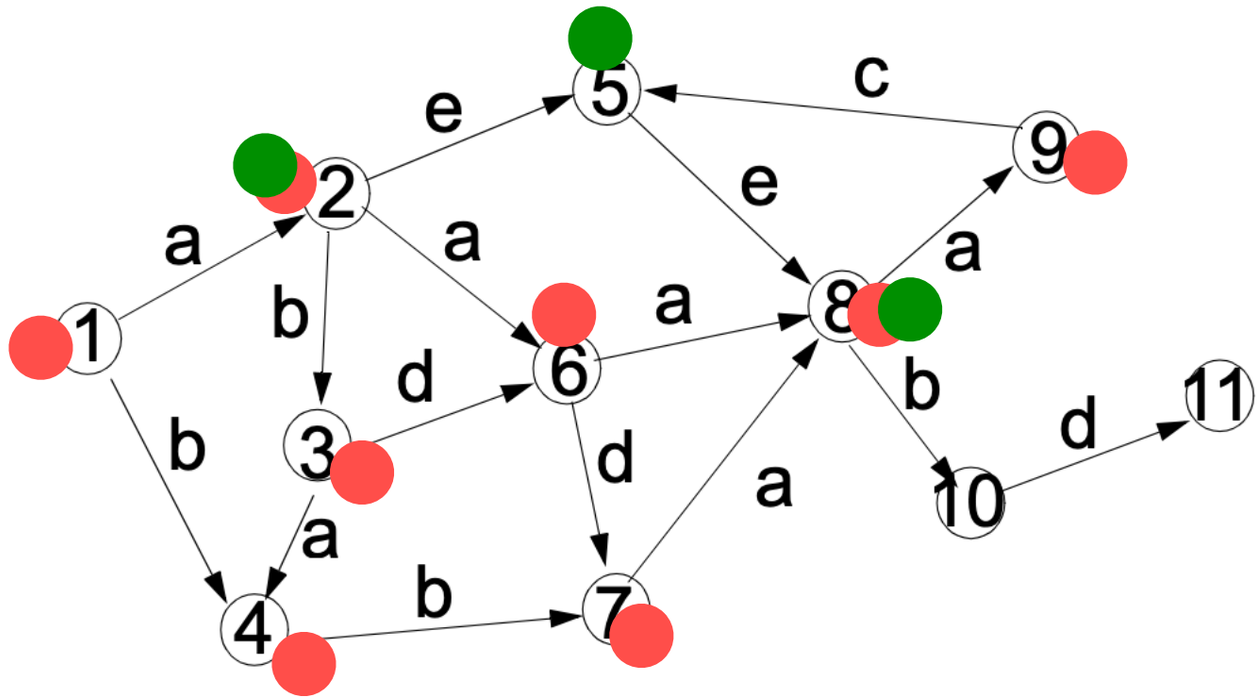
$$(x, y) \leftarrow a^+(x, y), e^+(x, y)$$



We need to take the intersection between the two sets

Another example of CRPQ

$$(x, y) \leftarrow a^+(x, y), e^+(x, y)$$

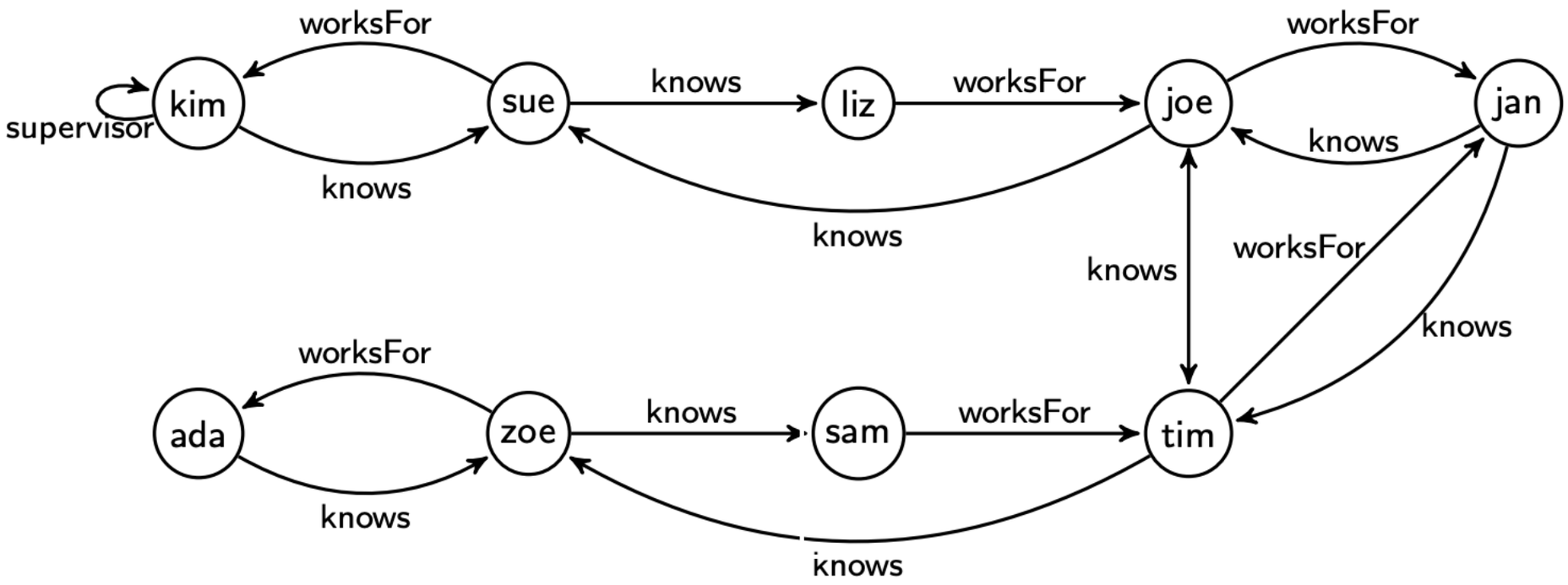


We need to take the intersection between the two sets

Solution: $(x, y) = (2,8)$

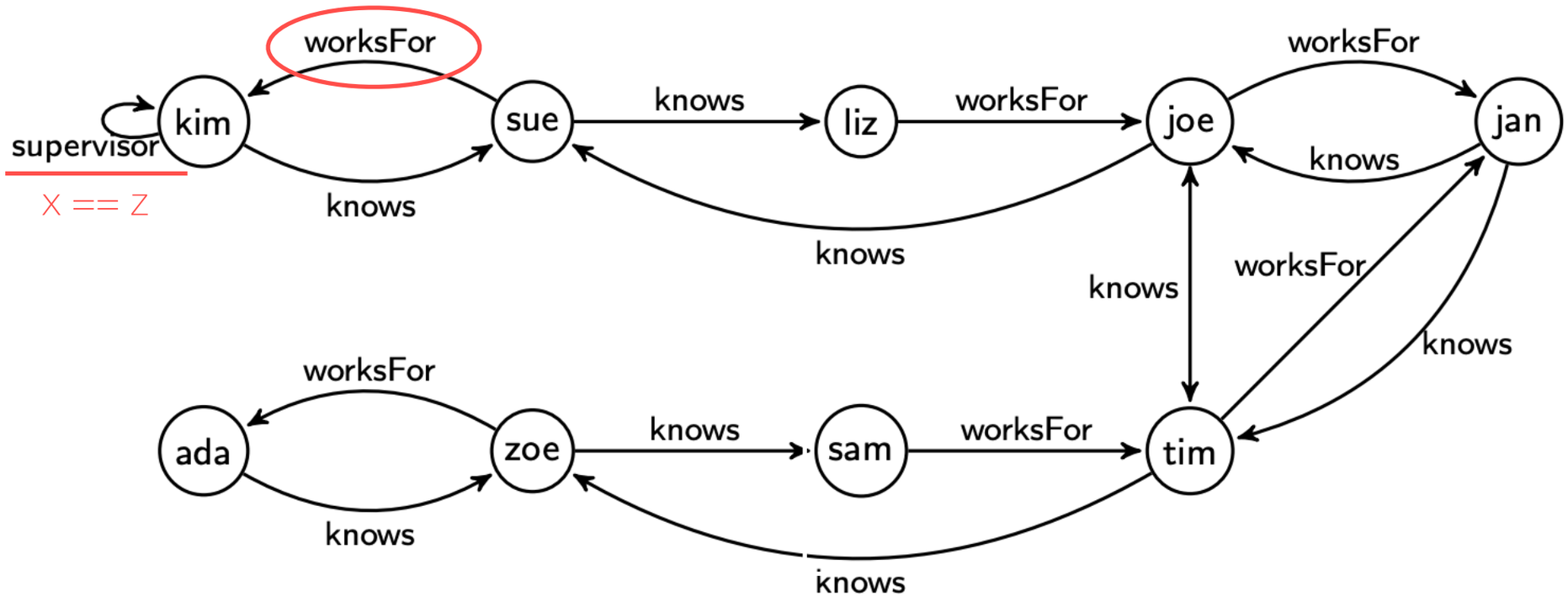
Exercise

$$(x, y) \leftarrow \text{supervisor}(x, z)/\text{worksFor}^{-}(x, y)$$



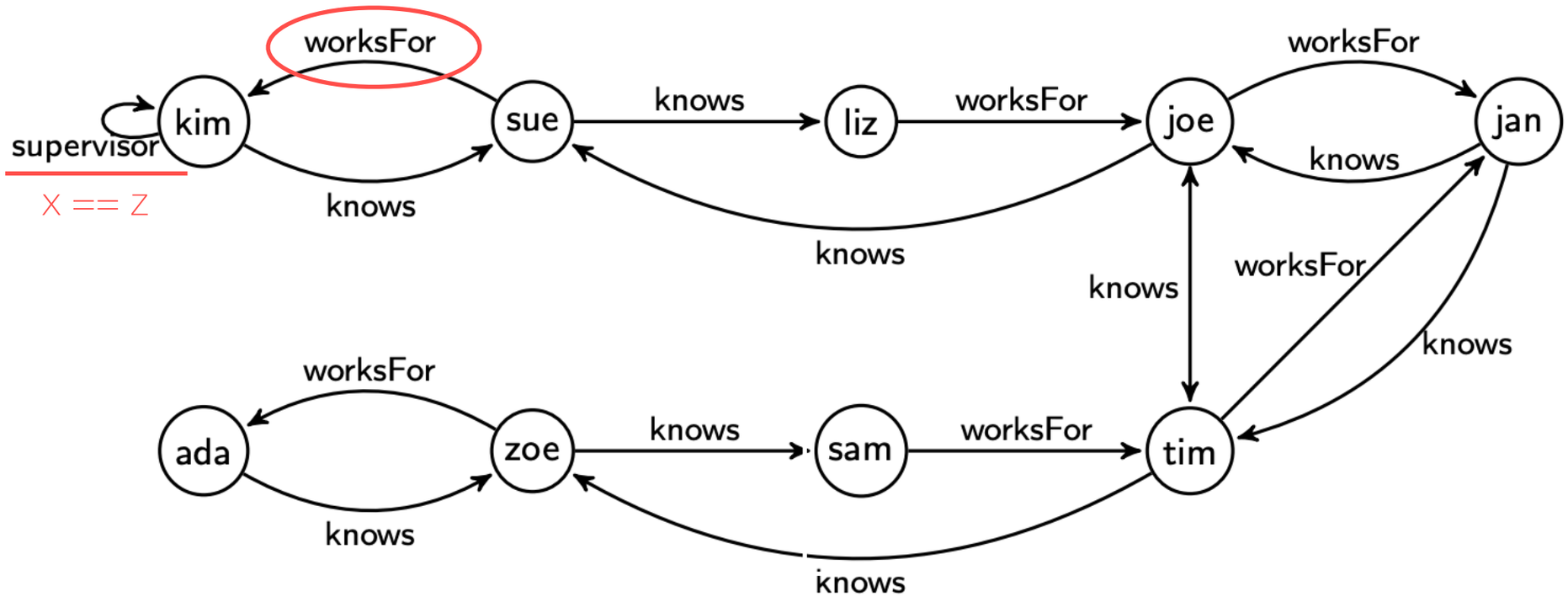
Exercise

$$(x, y) \leftarrow \text{supervisor}(x, z)/\text{worksFor}^{-}(x, y)$$



Exercise

$$(x, y) \leftarrow \text{supervisor}(x, z)/\text{worksFor}^{-}(x, y)$$



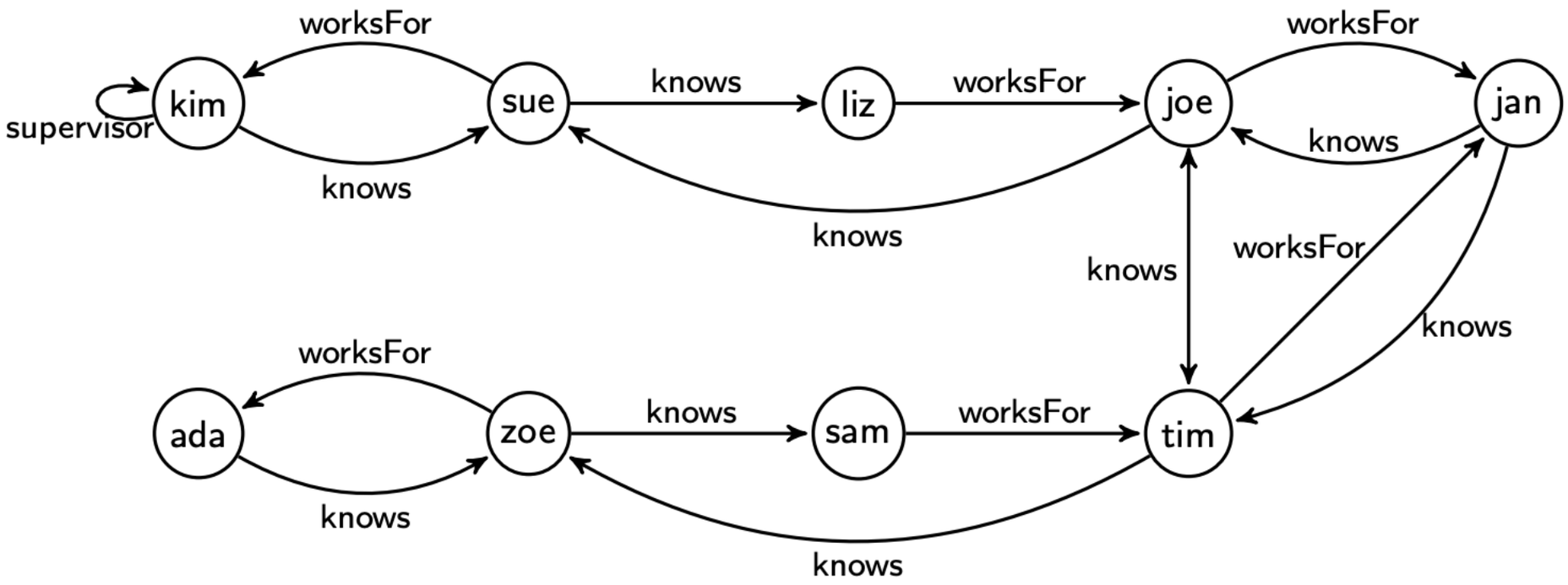
Solution

(x,y) = (kim,sue)

Exercise

Find two people that know the same person

$$(x, y) \leftarrow \text{knows}(x, z)/\text{knows}^{-}(z, y)$$



(x,y) = {(tim, ada), (kim,joe), (jan,tim), (joe,jan)} [note that all the reversed pairs are valid results too]