M. Besta, A. C. Catarino, L. Gianinazzi, N. Blach, P. Nyczyk, H. Niewiadomski, T. Hoefler

HOT: Higher-Order Dynamic Graph Representation Learning with Efficient Transformers

M. BESTA, A. C. CATARINO, L. GIANINAZZI, N. BLACH, P. NYCZYK, H. NIEWIADOMSKI, T. HOEFLER

HOT: Higher-Order Dynamic Graph Representation Learning with Efficient Transformers

@ LOG'23

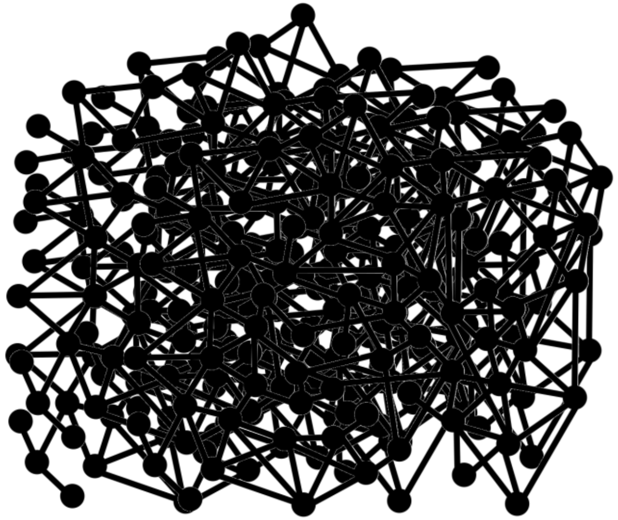
HOT: Higher-Order Dynamic Graph Representation Learning with Efficient Transformers

Maciej Besta[1]* Afonso Claudino Catarino[1]* Lukas Gianinazzi[1] Nils Blach[1]
Piotr Nyczyk[2] Hubert Niewiadomski[2] Torsten Hoefler[1]

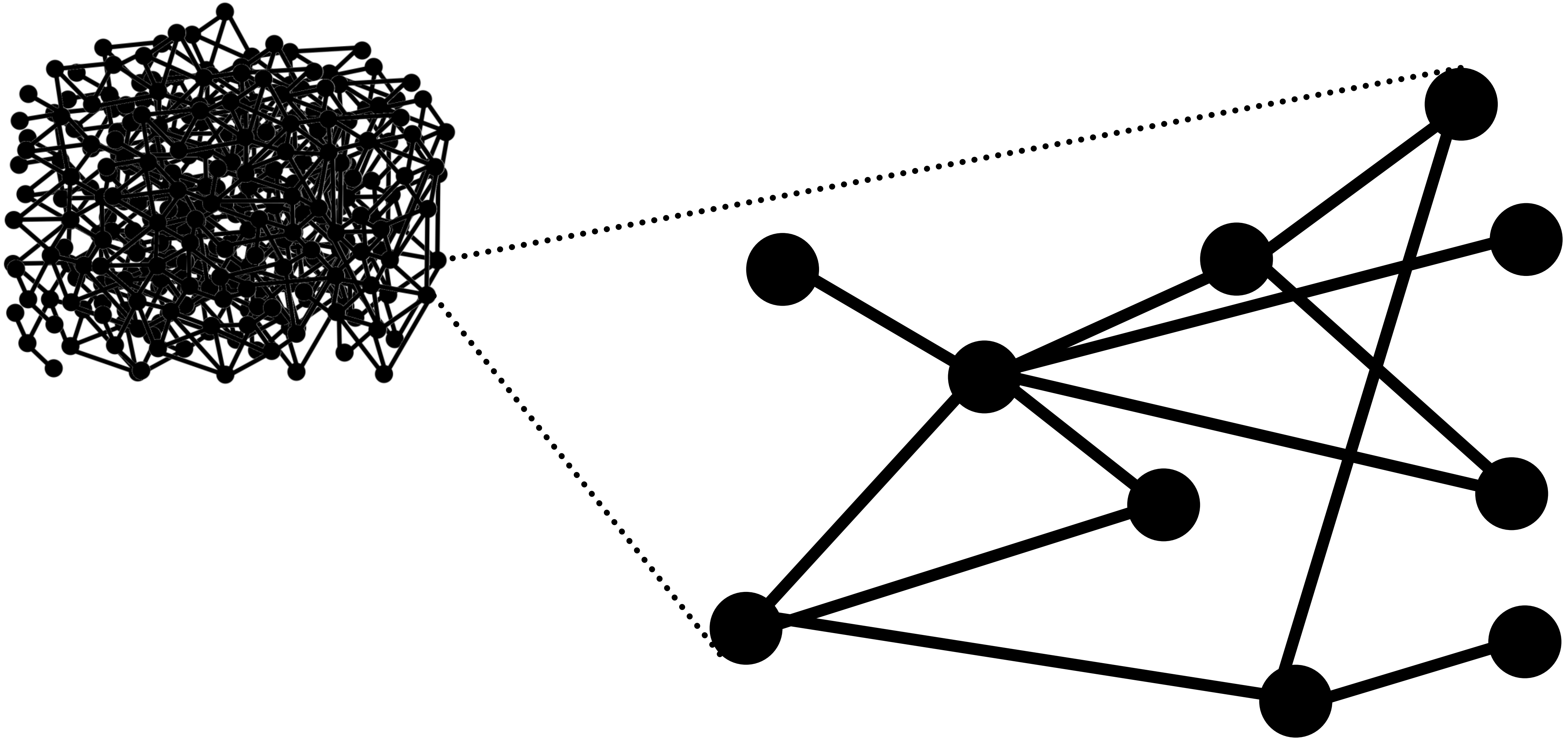
[1]Department of Computer Science, ETH Zurich; [2]Cledar

Link Prediction

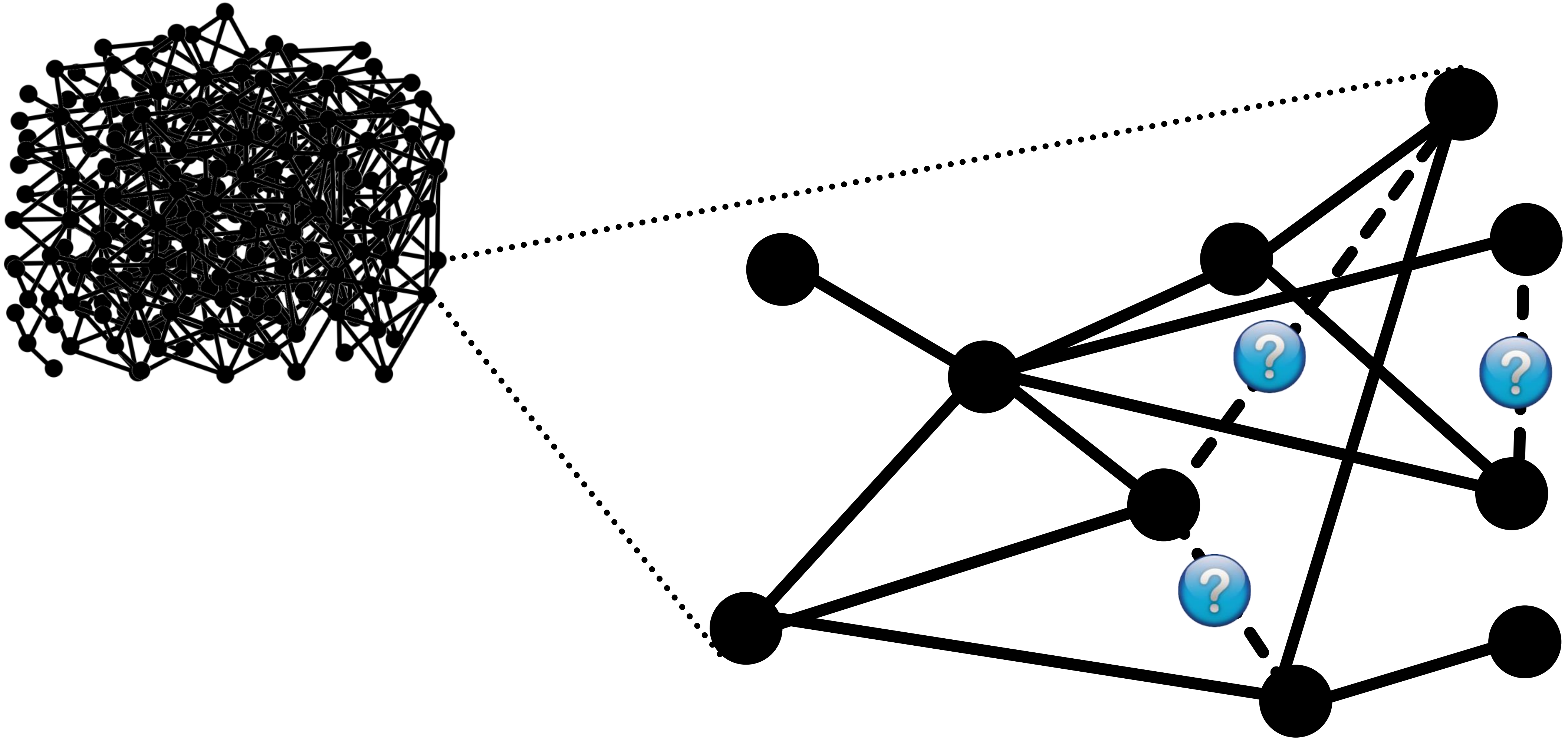
Link Prediction



Link Prediction

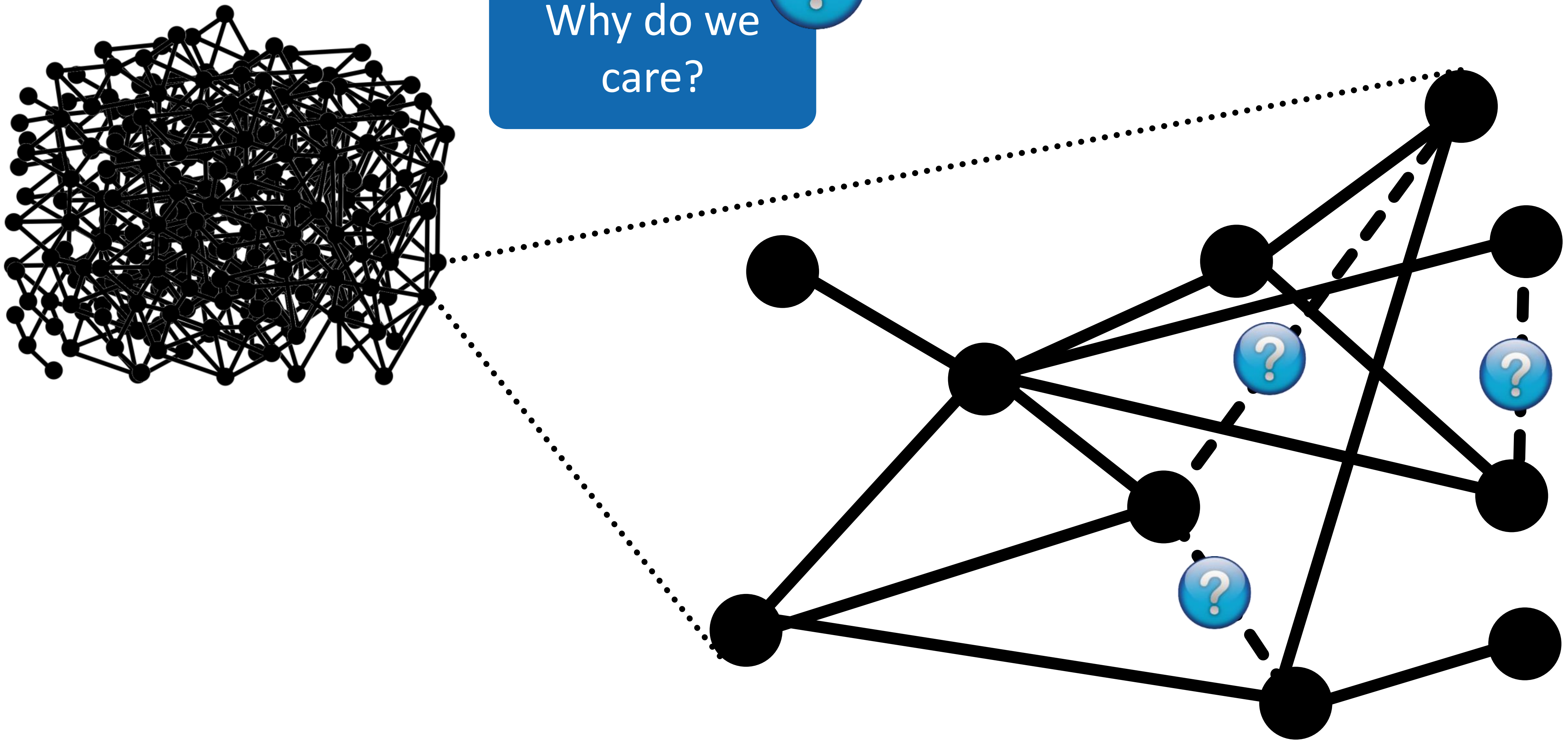


Link Prediction



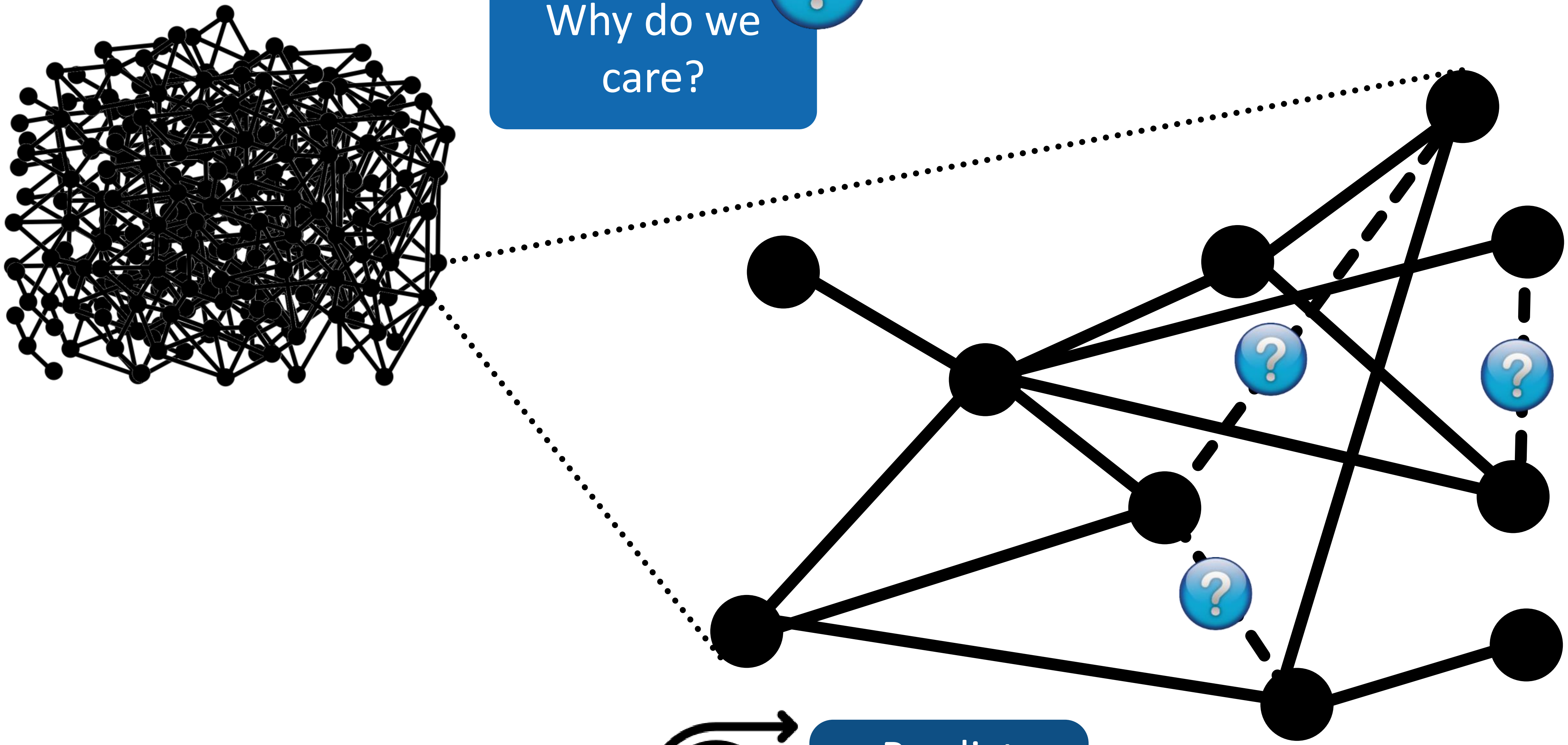
Link Prediction

Why do we care?



Link Prediction

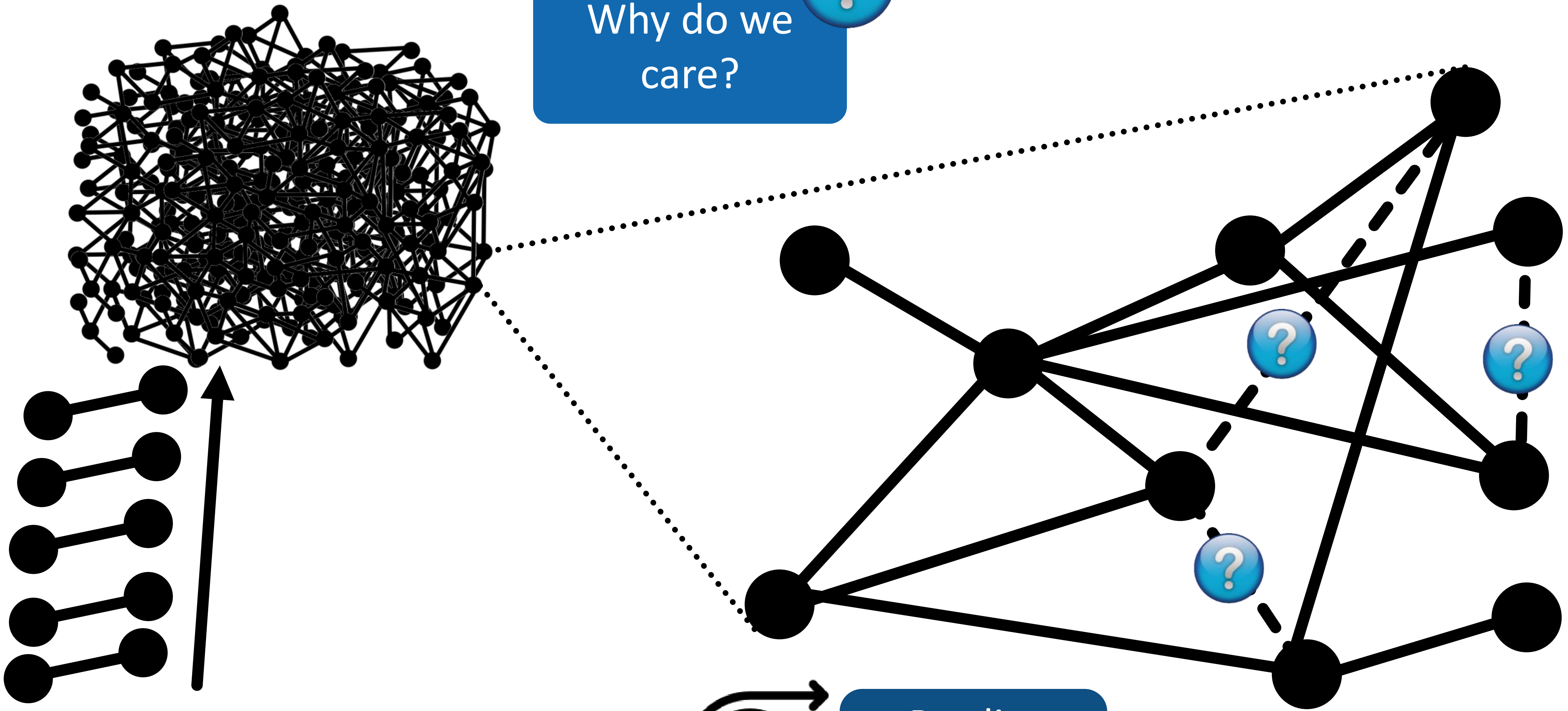
Why do we care?



Predict future data

Link Prediction

Why do we care?

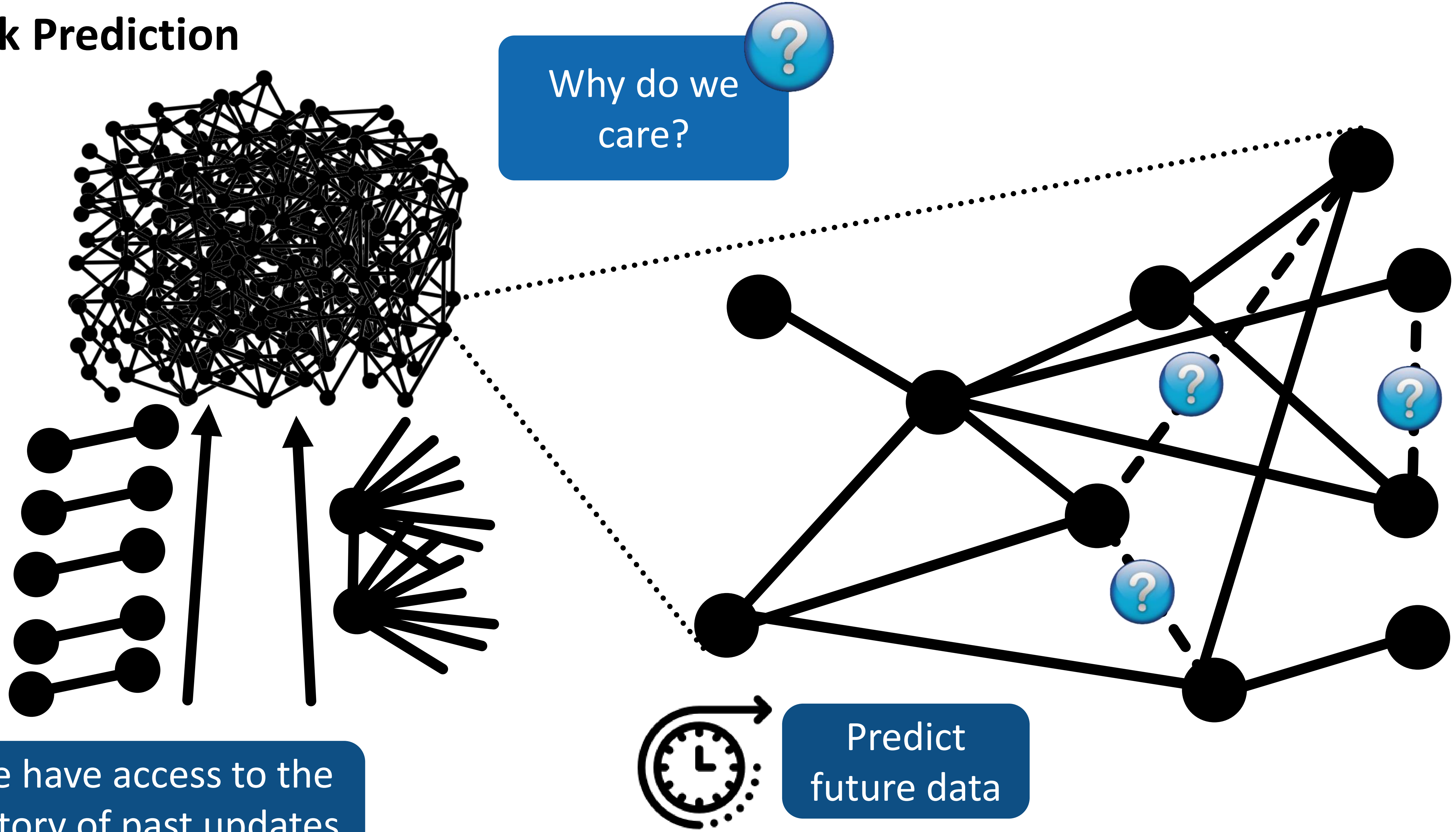


We have access to the history of past updates

Predict future data

Link Prediction

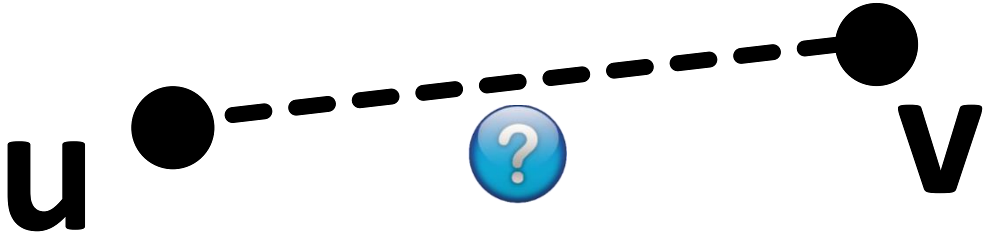
Why do we care?



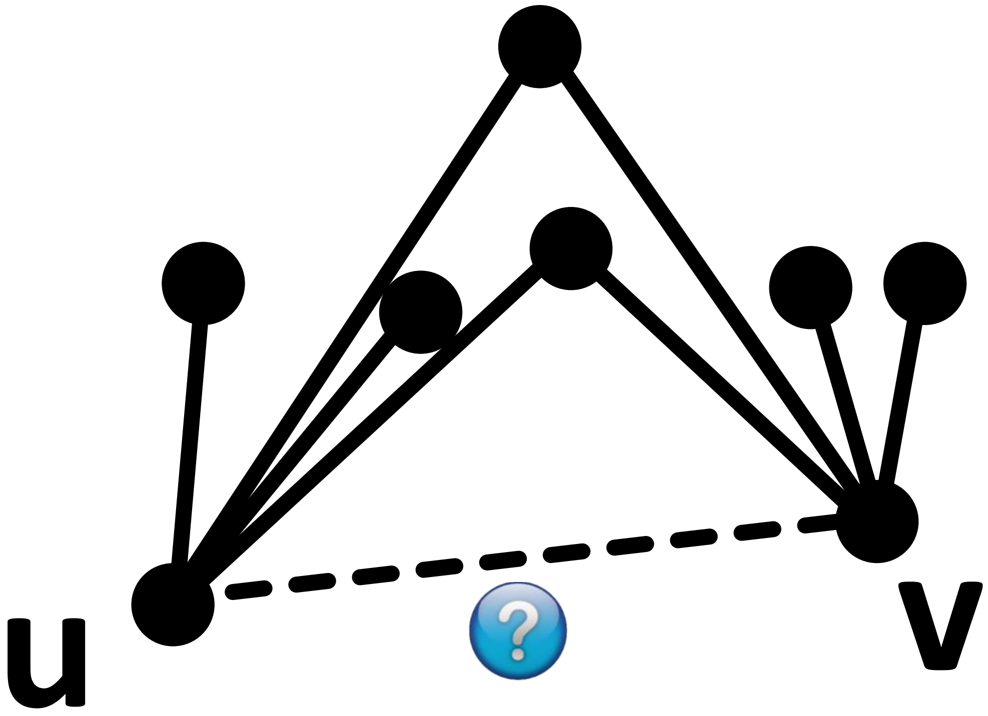
We have access to the history of past updates

Predict future data

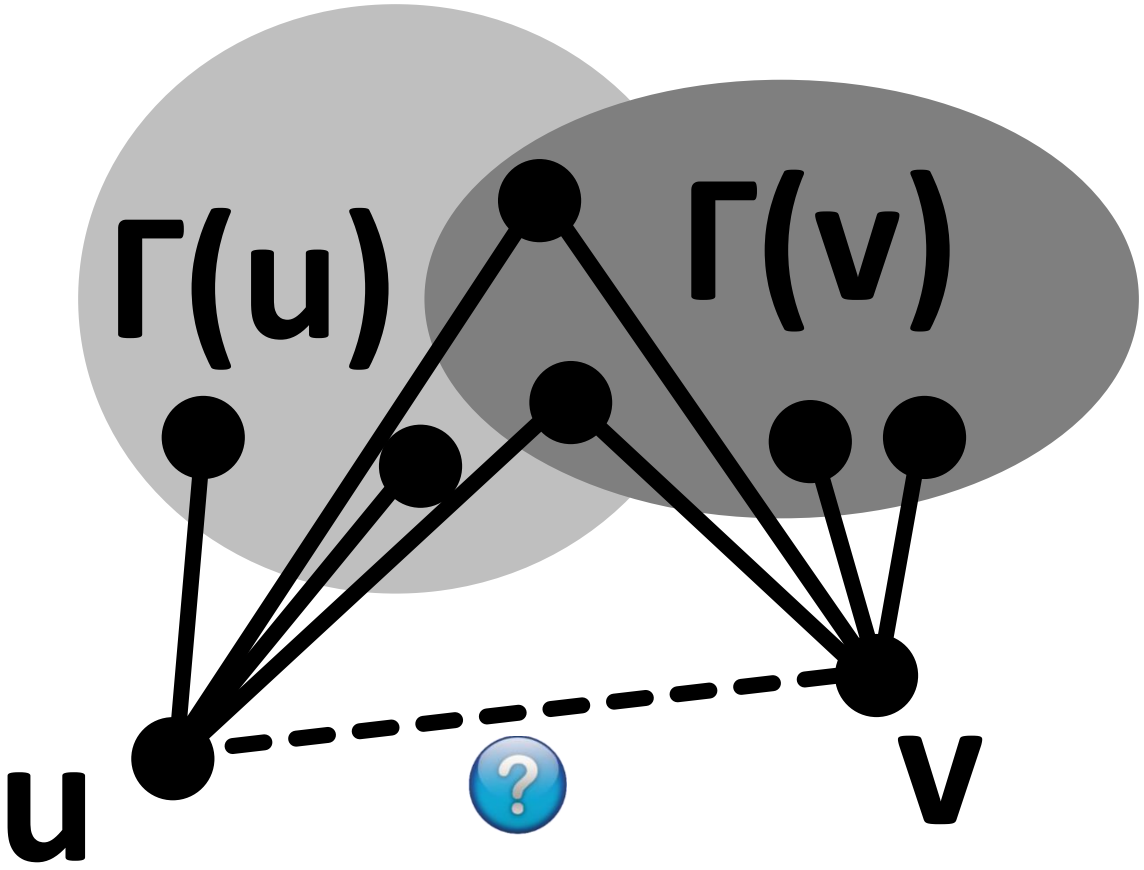
Link Prediction



Link Prediction



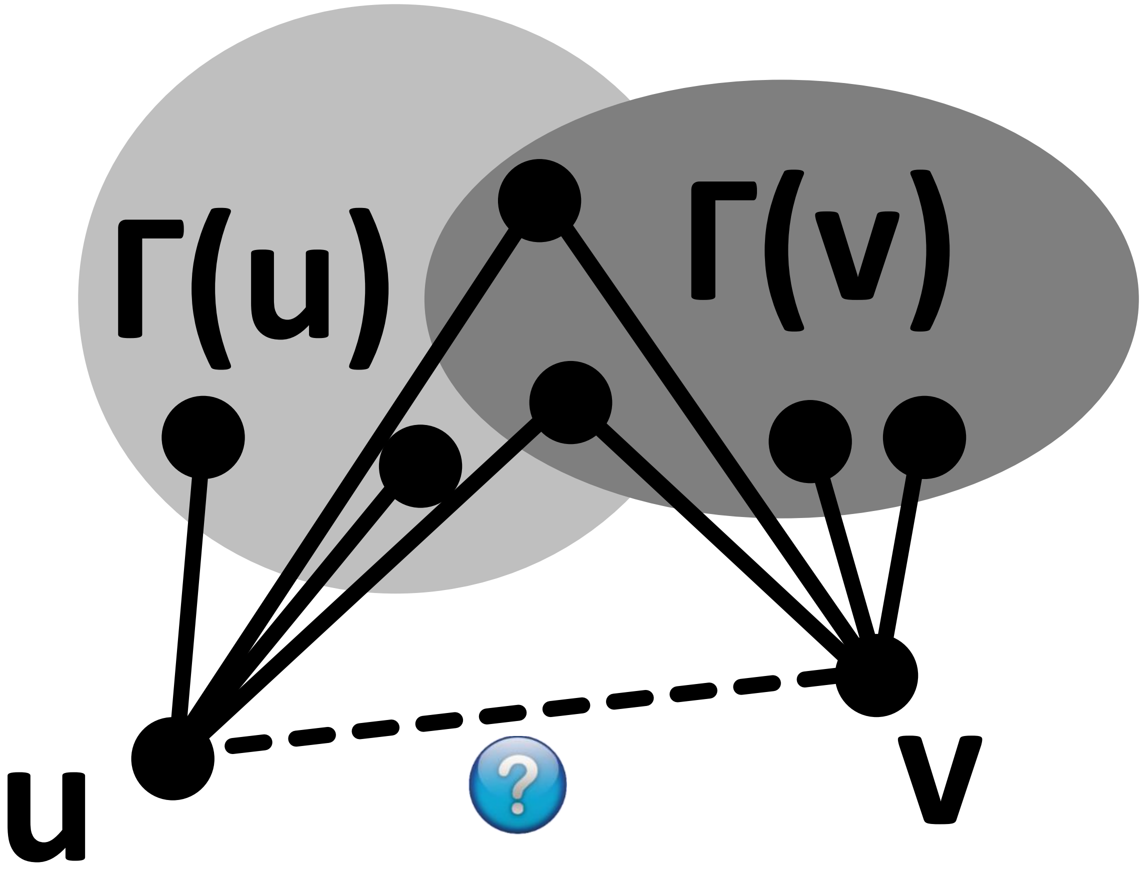
Link Prediction



Link Prediction

$$s_{u,v}^{Jaccard} = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

$$s_{u,v}^{CN} = |\Gamma(u) \cap \Gamma(v)|$$



Link Prediction

$$s_{u,v}^{Jaccard} = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

$$s_{u,v}^{RA} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{d_z}$$

$$s_{u,v}^{CN} = |\Gamma(u) \cap \Gamma(v)|$$

$$s_{u,v}^{HPI} = \frac{|\Gamma(u) \cap \Gamma(v)|}{min\{d_u, d_v\}}$$

$$s_{u,v}^{HDI} = \frac{|\Gamma(u) \cap \Gamma(v)|}{max\{d_u, d_v\}}$$

$$s_{u,v}^{Salton} = \frac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{d_u d_v}}$$

$$s_{u,v}^{Sorensen} = \frac{2|\Gamma(u) \cap \Gamma(v)|}{d_u + d_v}$$

$$s_{u,v}^{AA} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log d_z}$$

$$s_{u,v}^{PAI} = |\Gamma(u)||\Gamma(v)| = d_u d_v$$

Link Prediction

$$s_{u,v}^{Jaccard} = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

$$s_{u,v}^{RA} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{d_z}$$

$$s_{u,v}^{CN} = |\Gamma(u) \cap \Gamma(v)|$$

$$s_{u,v}^{HPI} = \frac{|\Gamma(u \ldots}{min \ldots}$$

$$\ldots \frac{\ldots \cap \Gamma(v)|}{\ldots d_u d_v}$$

Only „simple" (dyadic) edges are considered

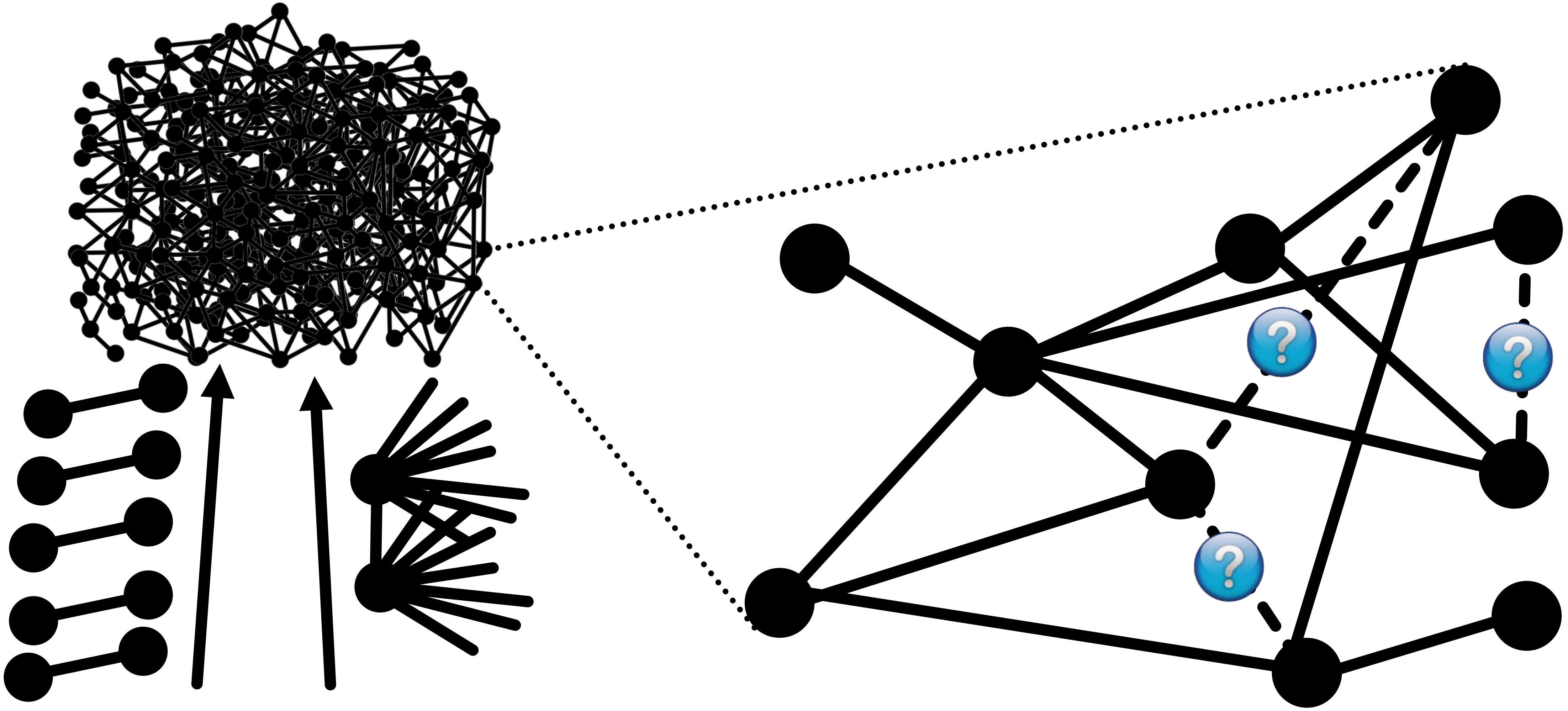
$$s_{u,v}^{HDI} = \frac{|\Gamma(u) \cap \Gamma(v)|}{max\{d_u, d_v\}}$$

$$s_{u,v}^{Sorensen} = \frac{2|\Gamma(u) \cap \Gamma(v)|}{d_u + d_v}$$

$$s_{u,v}^{AA} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log d_z}$$

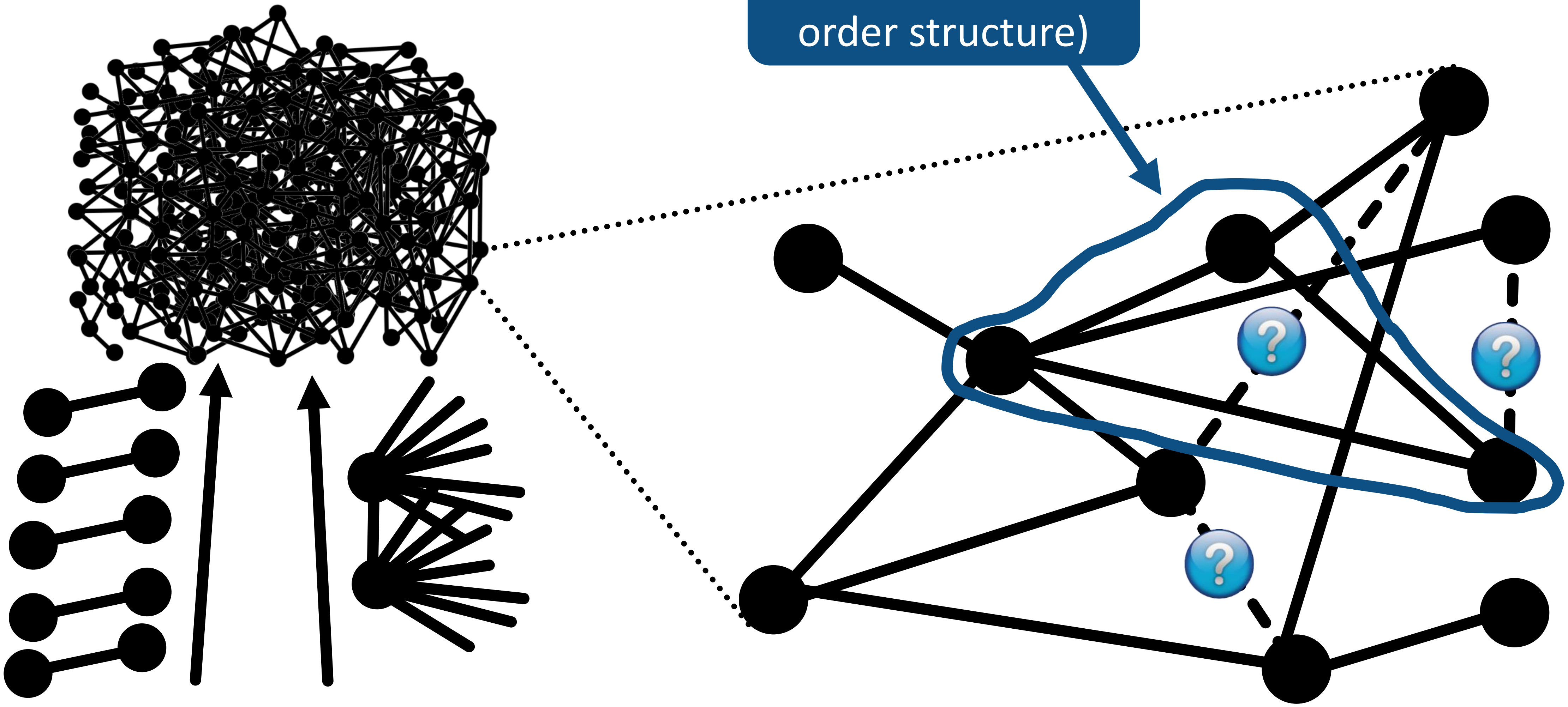
$$s_{u,v}^{PAI} = |\Gamma(u)||\Gamma(v)| = d_u d_v$$

Link Prediction



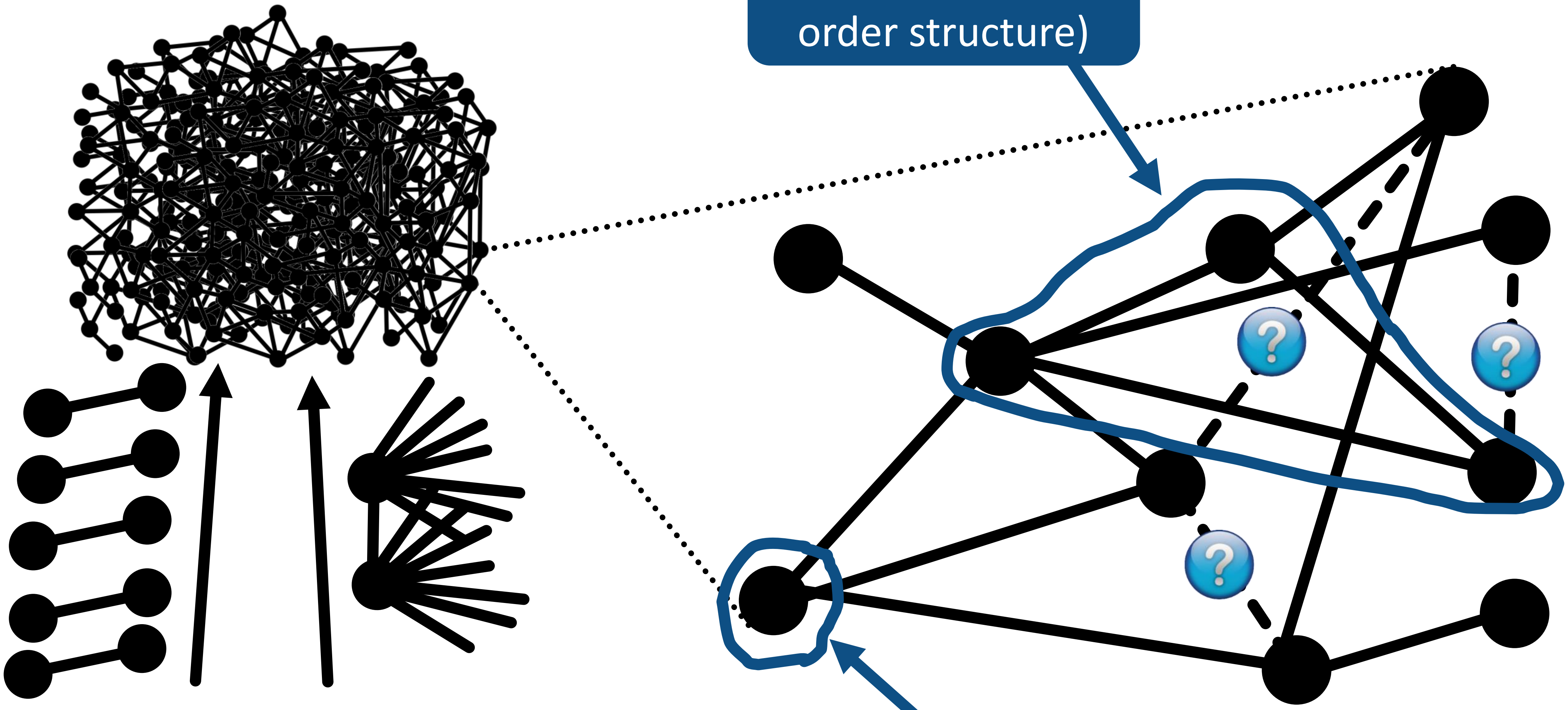
We have access to the history of past updates

Link Prediction



We have access to the history of past updates

Link Prediction



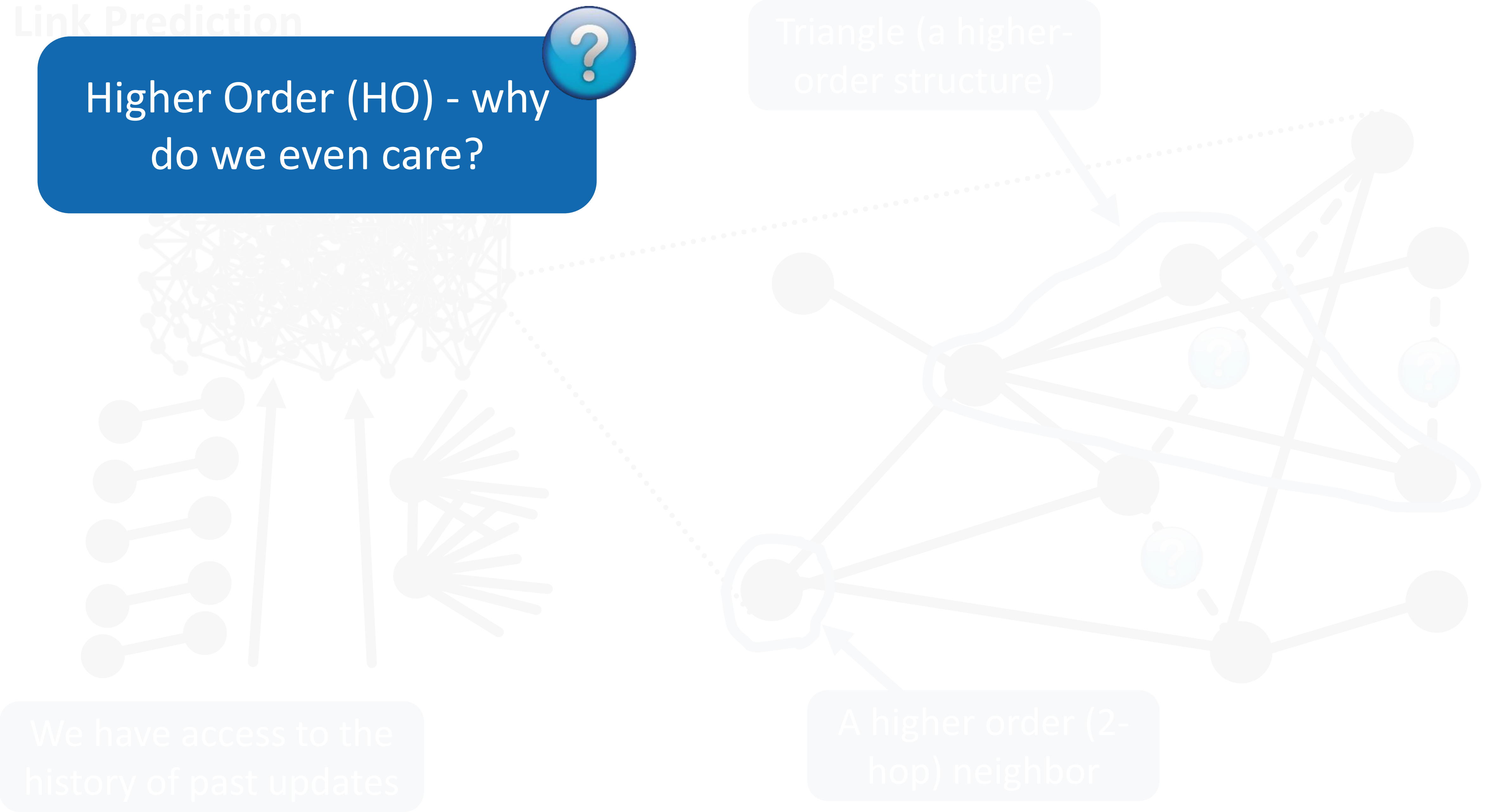
Triangle (a higher-order structure)

We have access to the history of past updates

A higher order (2-hop) neighbor

Link Prediction

Higher Order (HO) - why do we even care?



Higher Order (HO) - why do we even care?

Incorporating HO graph structures results in fundamentally more powerful predictions for many workloads [1, ...]

[1] C. Morris et al. Weisfeiler and leman go neural: Higher-order graph neural networks. AAAI 2019.

Higher Order (HO) - why do we even care?

Incorporating HO graph structures results in fundamentally more powerful predictions for many workloads [1, ...]

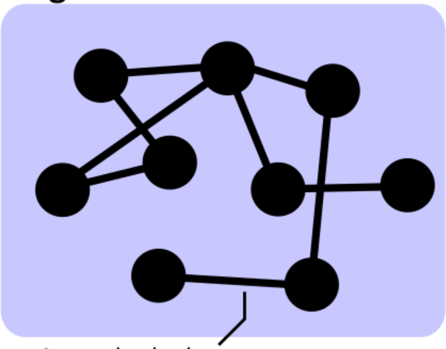
Before we go on... what Higher Order exactly is?

[1] C. Morris et al. Weisfeiler and leman go neural: Higher-order graph neural networks. AAAI 2019.

Higher-Order Graph Structures: Summary & Scope

HO: „Anything that goes beyond a simple dyadic edge”

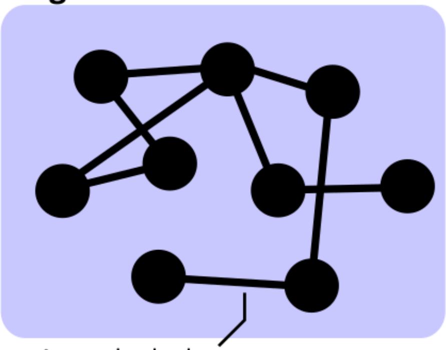
Edge



Higher-Order Graph Structures: Summary & Scope

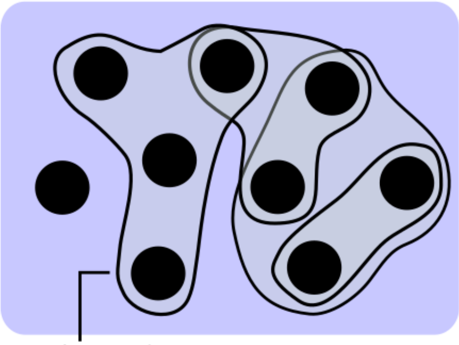
HO: „Anything that goes beyond a simple dyadic edge”

Edge



A standard edge
between two vertices

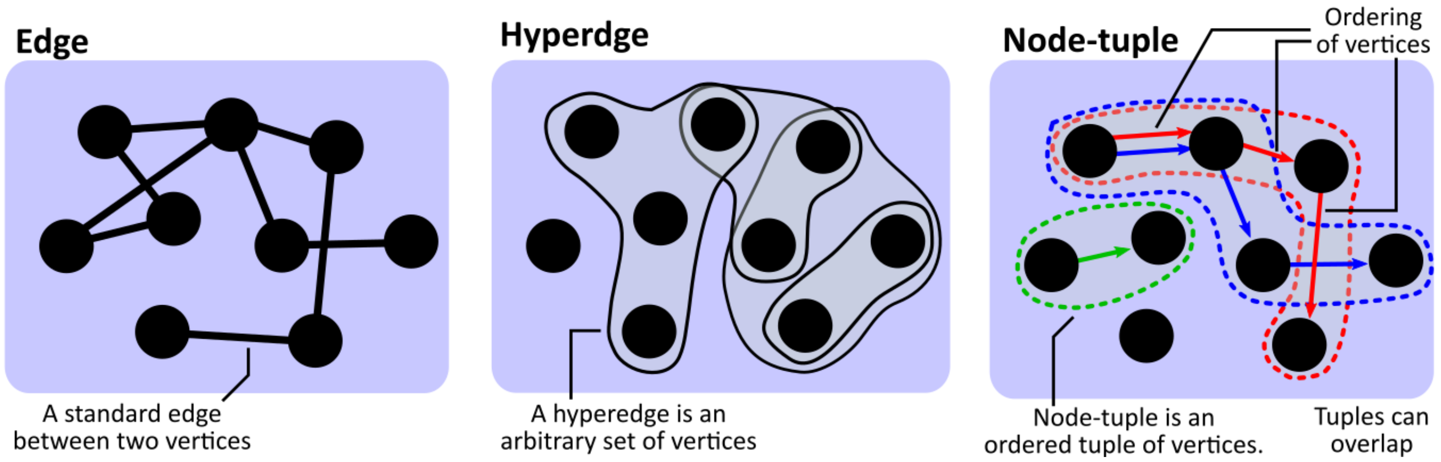
Hyperdge



A hyperedge is an
arbitrary set of vertices

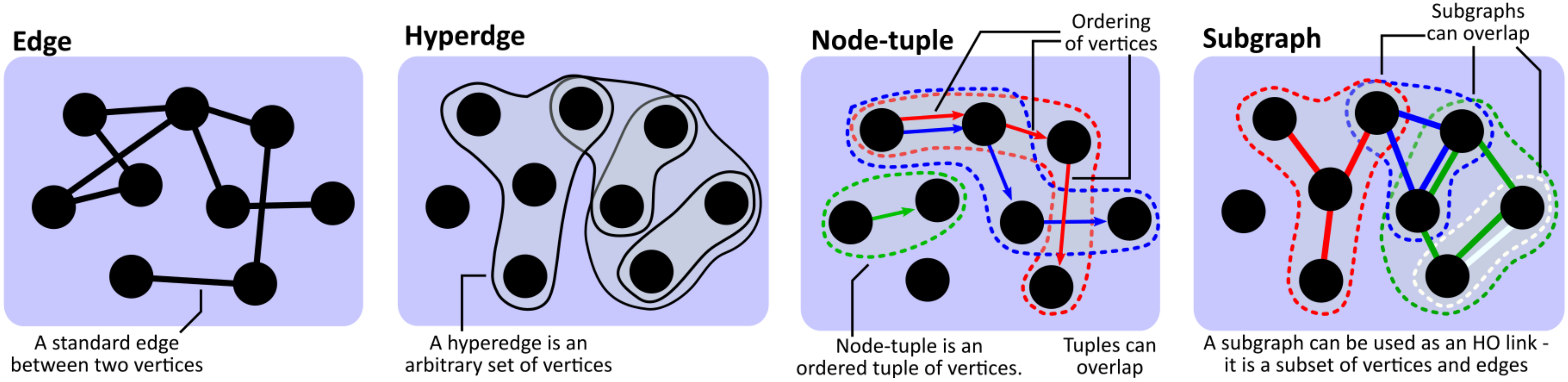
Higher-Order Graph Structures: Summary & Scope

HO: „Anything that goes beyond a simple dyadic edge"



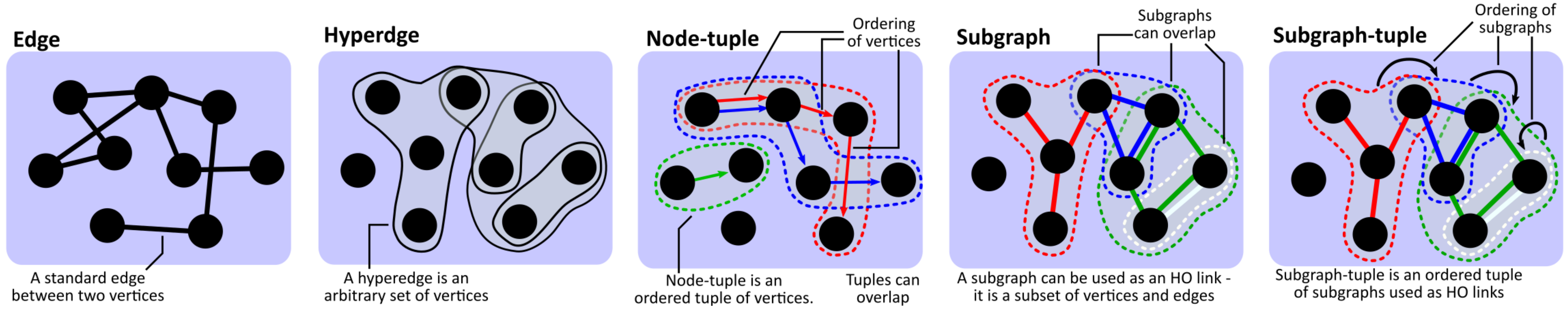
Higher-Order Graph Structures: Summary & Scope

HO: „Anything that goes beyond a simple dyadic edge"



Higher-Order Graph Structures: Summary & Scope

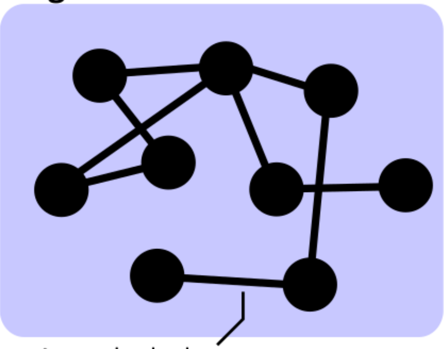
HO: „Anything that goes beyond a simple dyadic edge”



Higher-Order Graph Structures: Summary & Scope

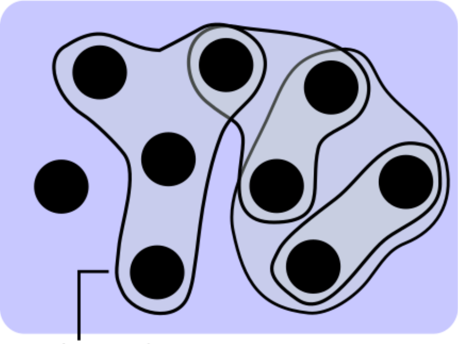
HO: „Anything that goes beyond a simple dyadic edge"

Edge



A standard edge between two vertices

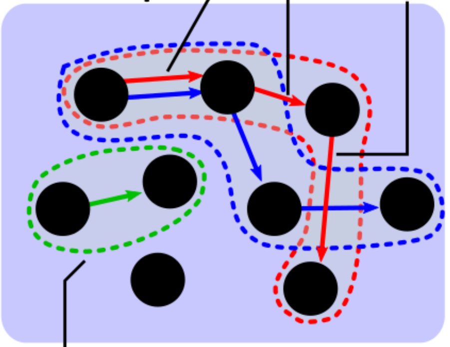
Hyperdge



A hyperedge is an arbitrary set of vertices

Node-tuple

Ordering of vertices

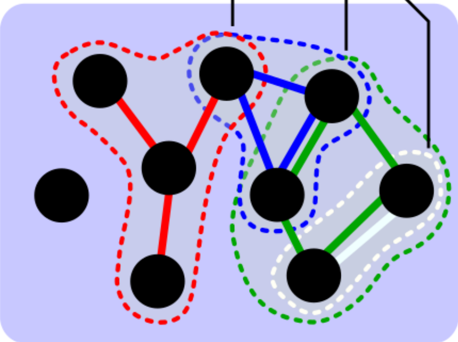


Node-tuple is an ordered tuple of vertices.

Tuples can overlap

Subgraph

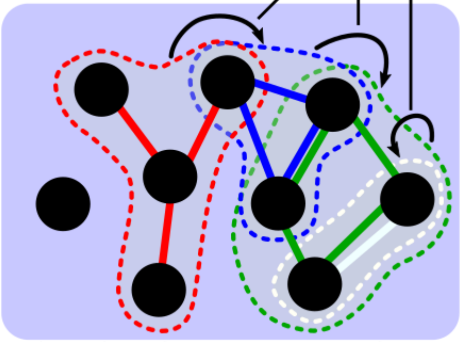
Subgraphs can overlap



A subgraph can be used as an HO link - it is a subset of vertices and edges

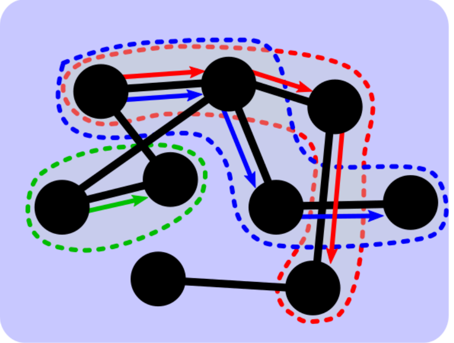
Subgraph-tuple

Ordering of subgraphs

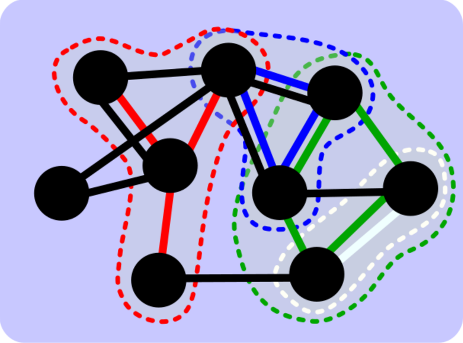


Subgraph-tuple is an ordered tuple of subgraphs used as HO links

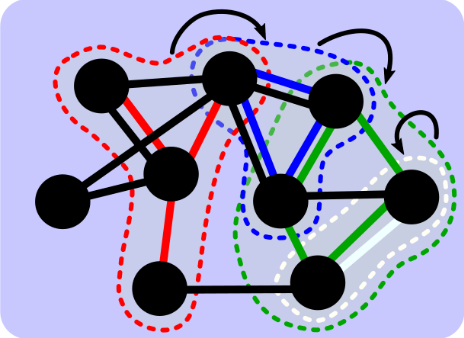
Node-tuples + edges



Subgraphs + edges



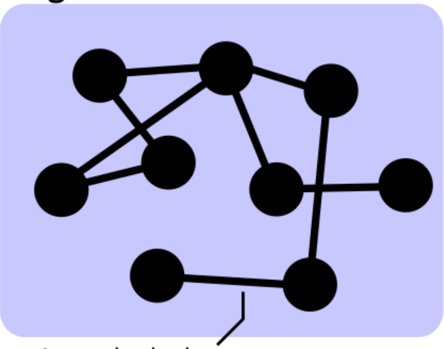
Subgraph-tuples + edges



Higher-Order Graph Structures: Summary & Scope

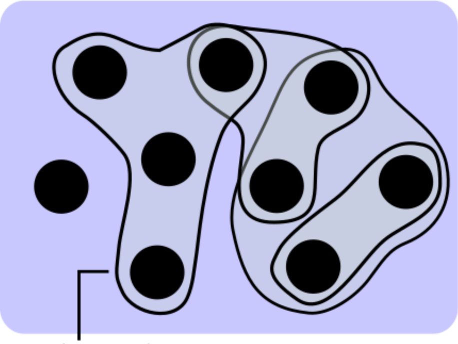
HO: „Anything that goes beyond a simple dyadic edge”

Edge



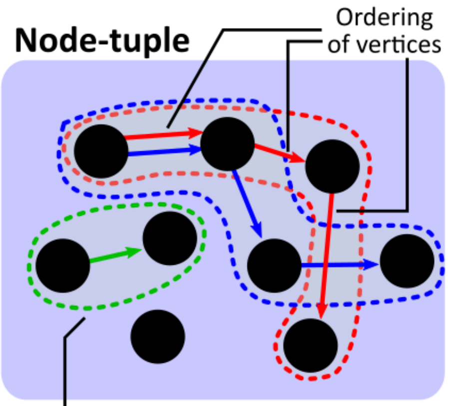
A standard edge between two vertices

Hyperdge



A hyperedge is an arbitrary set of vertices

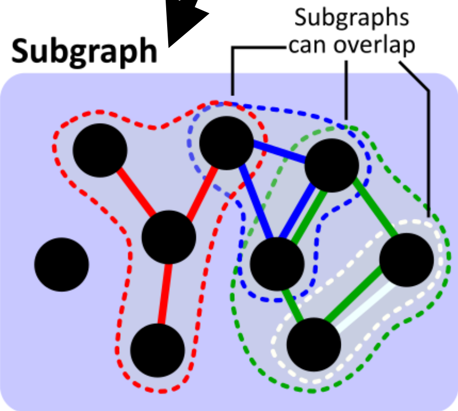
Node-tuple



Node-tuple is an ordered tuple of vertices.

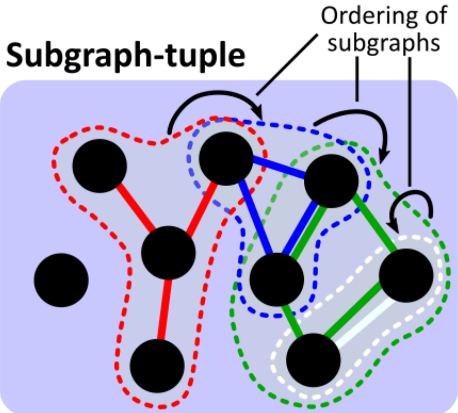
Tuples can overlap

Subgraph



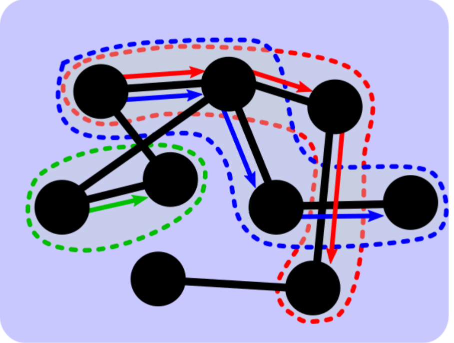
A subgraph can be used as an HO link - it is a subset of vertices and edges

Subgraph-tuple

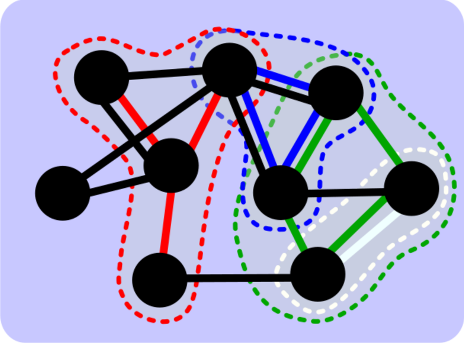


Subgraph-tuple is an ordered tuple of subgraphs used as HO links

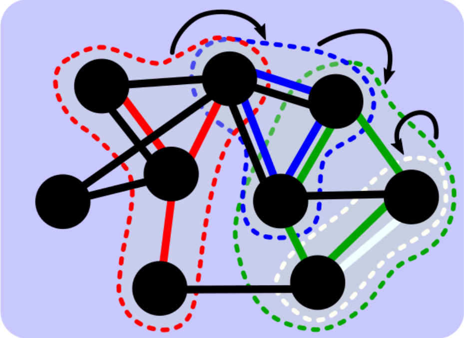
Node-tuples + edges



Subgraphs + edges

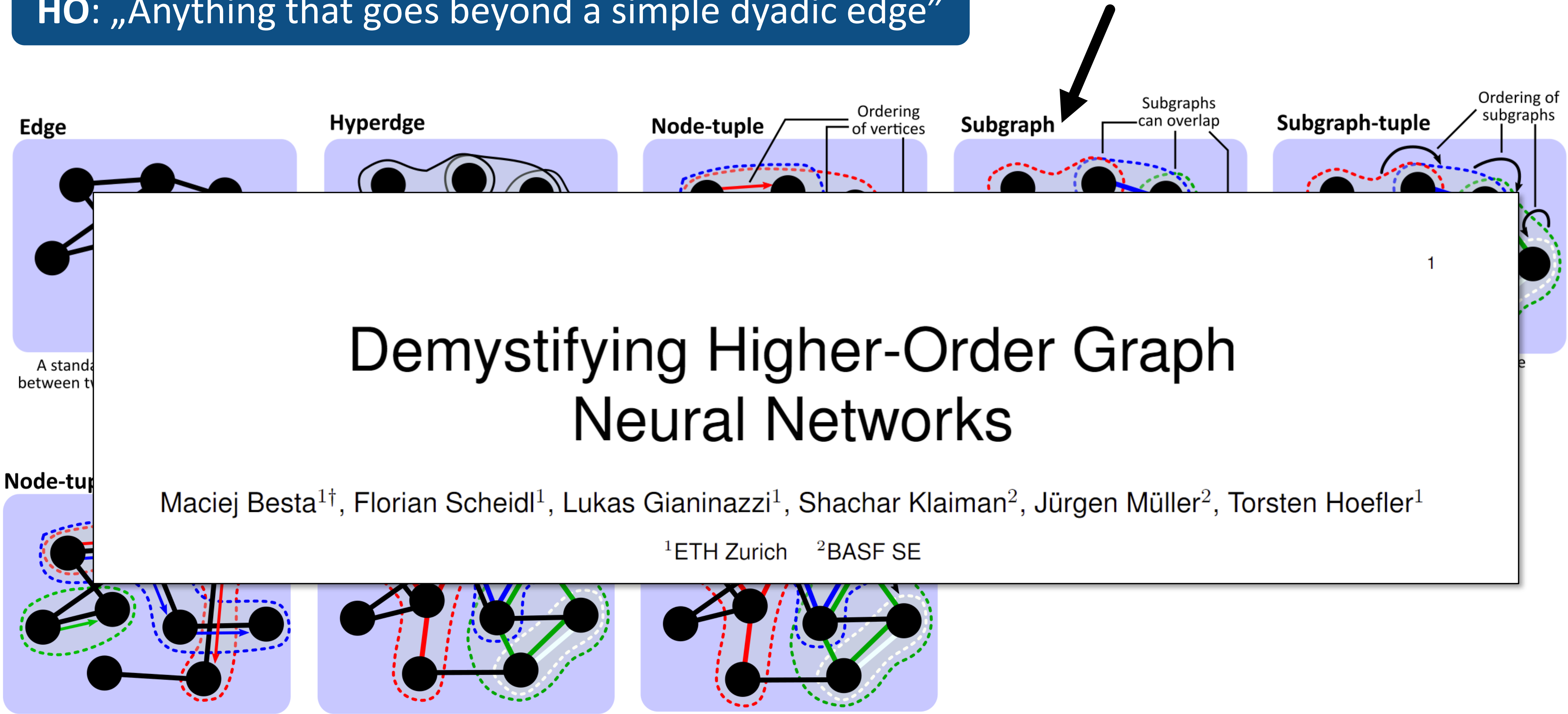


Subgraph-tuples + edges



Higher-Order Graph Structures: Summary & Scope

HO: „Anything that goes beyond a simple dyadic edge”



1

Demystifying Higher-Order Graph Neural Networks

Maciej Besta[1†], Florian Scheidl[1], Lukas Gianinazzi[1], Shachar Klaiman[2], Jürgen Müller[2], Torsten Hoefler[1]

[1]ETH Zurich [2]BASF SE

Higher-Order Graph Structures: Summary & Scope

HO: „Anything that goes beyond a simple dyadic edge"

Edge | Hyperdge | Node-tuple | Subgraph | Subgraph-tuple

Maciej Besta[1†], Florian Scheidl[1], Lukas Gianinazzi[1], Shachar Klaiman[2], Jürgen Müller[2], Torsten Hoefler[1]

[1]ETH Zurich [2]BASF SE

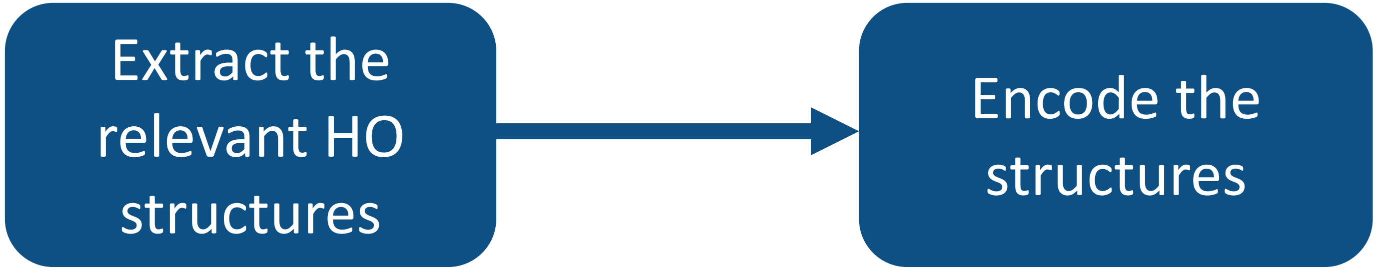
So we want to harness HO...
But what's the whole pipeline architecture?

HO-Enhanced Pipeline for Dynamic Link Prediction

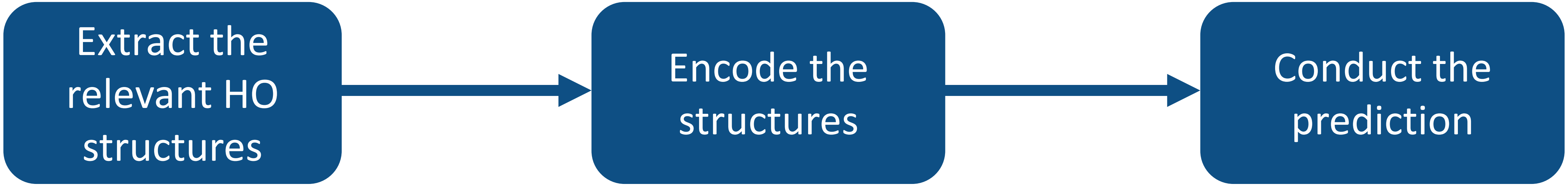
HO-Enhanced Pipeline for Dynamic Link Prediction

Extract the relevant HO structures

HO-Enhanced Pipeline for Dynamic Link Prediction



HO-Enhanced Pipeline for Dynamic Link Prediction

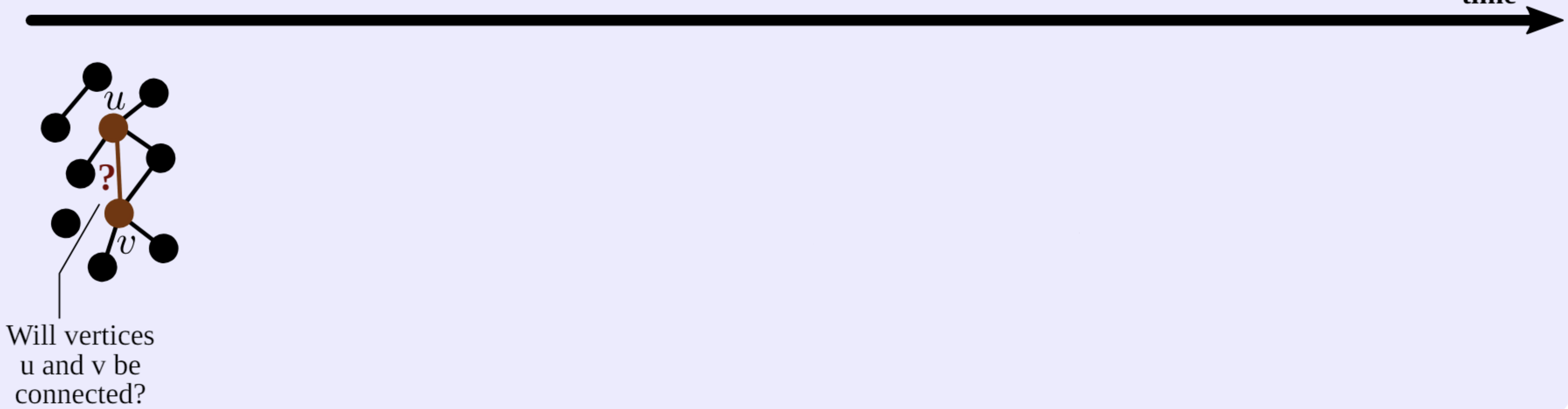


HO-Enhanced Pipeline for Dynamic Link Prediction

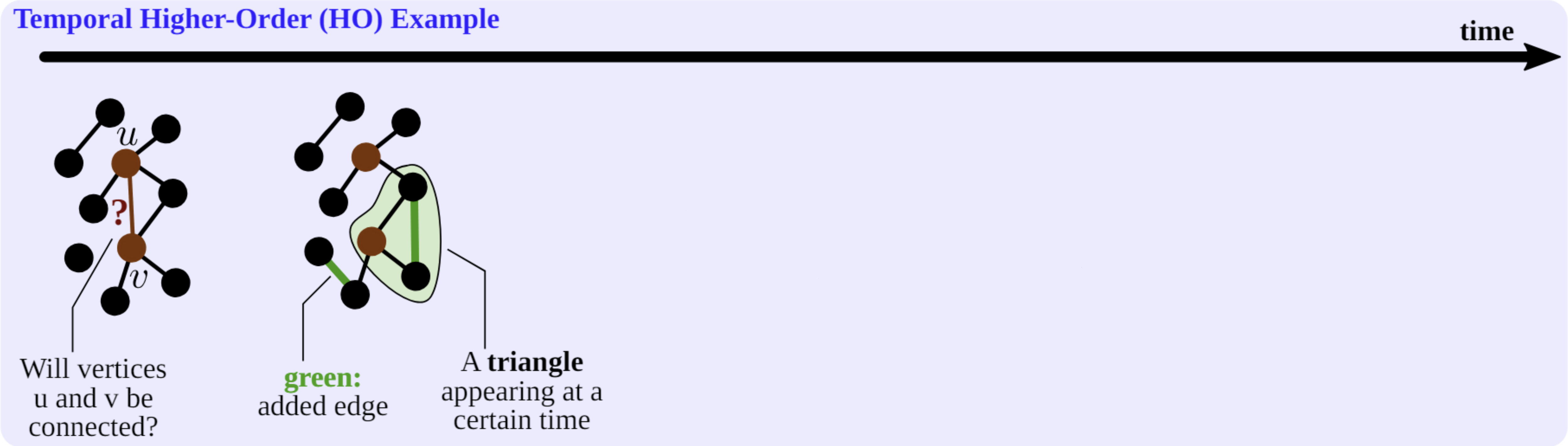
Extract the relevant HO structures → Encode the structures → Conduct the prediction

Temporal HO Structures

Temporal Higher-Order (HO) Example

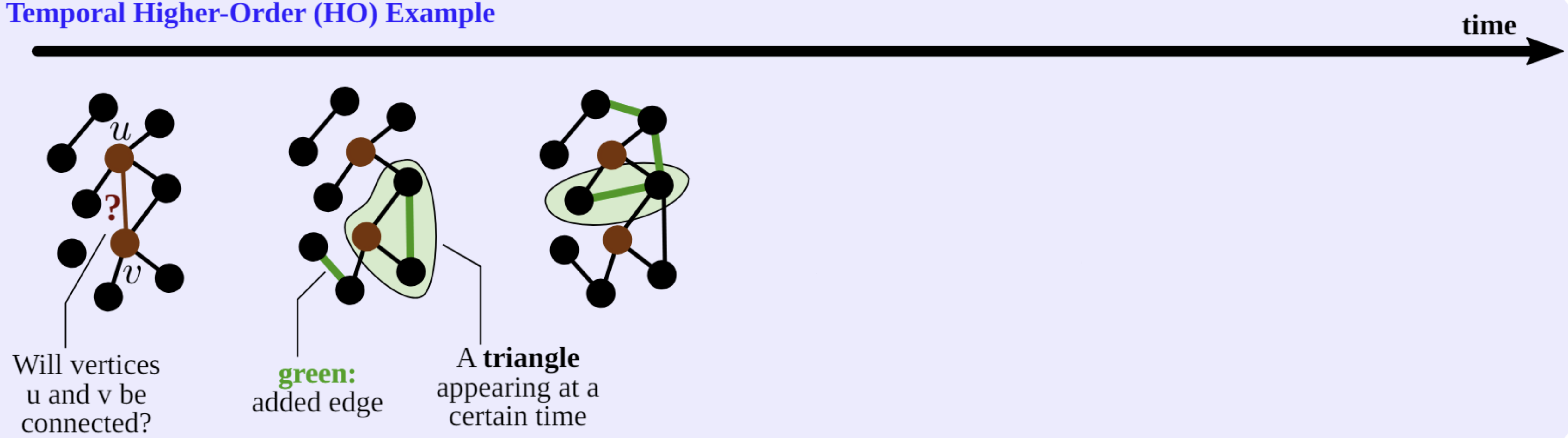


Temporal HO Structures

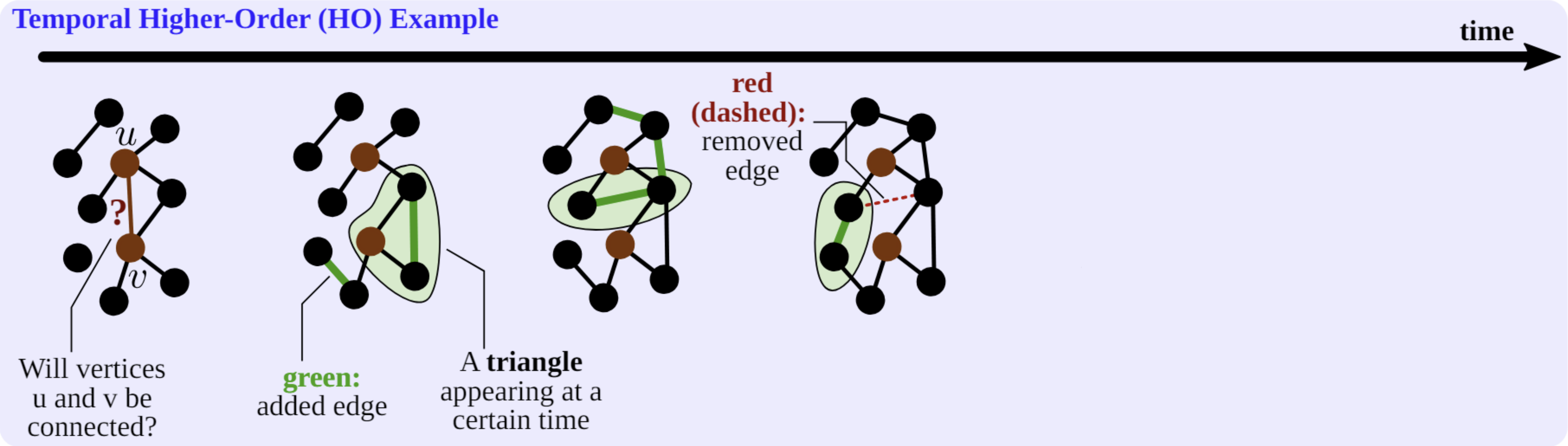


Temporal HO Structures

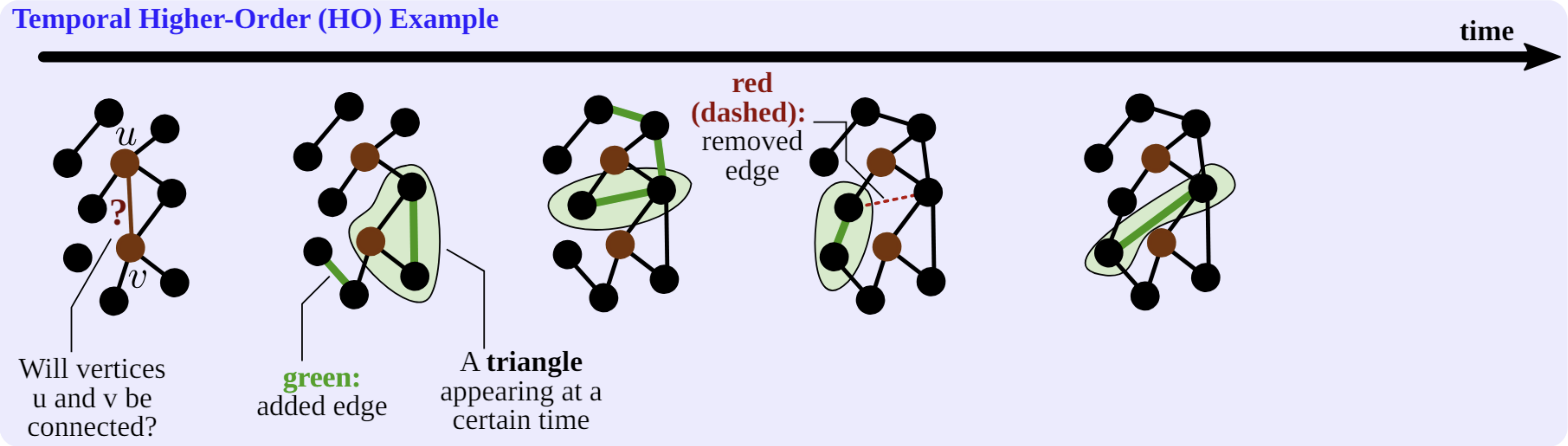
Temporal Higher-Order (HO) Example



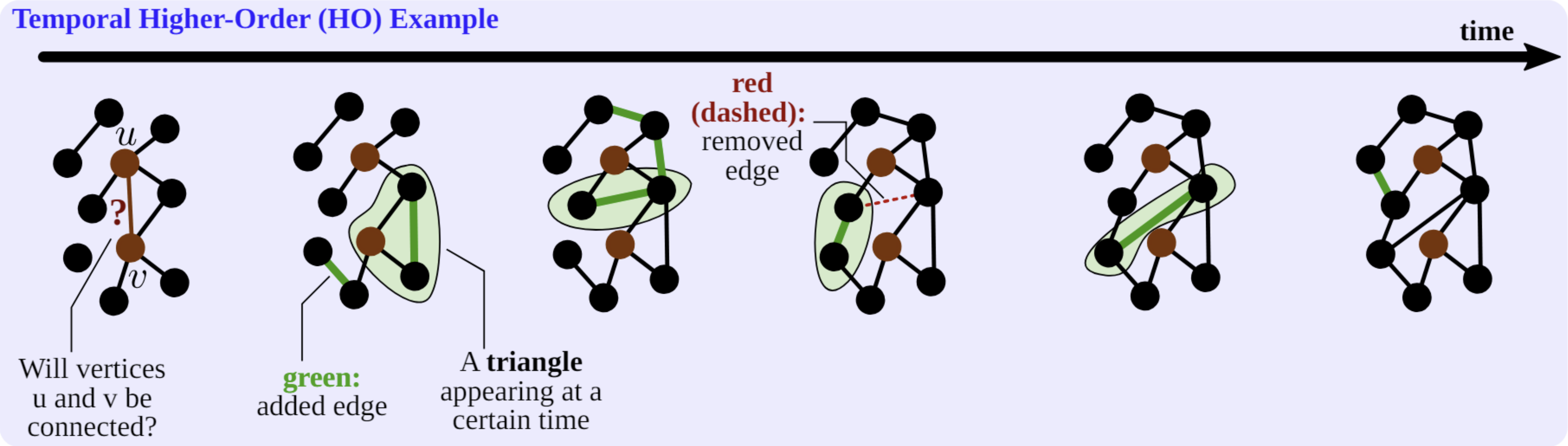
Temporal HO Structures



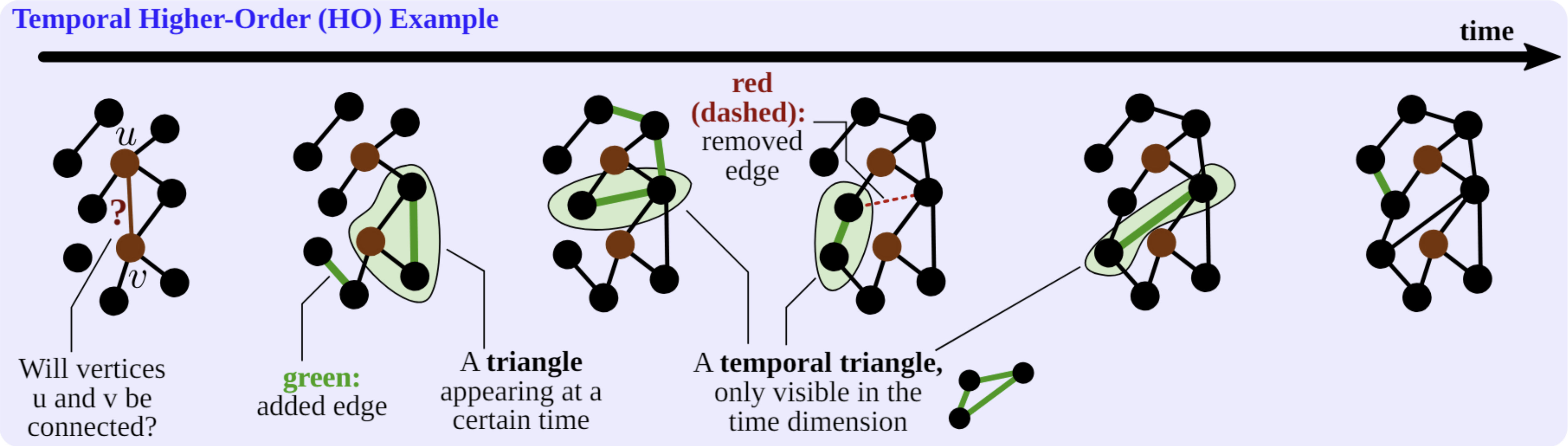
Temporal HO Structures



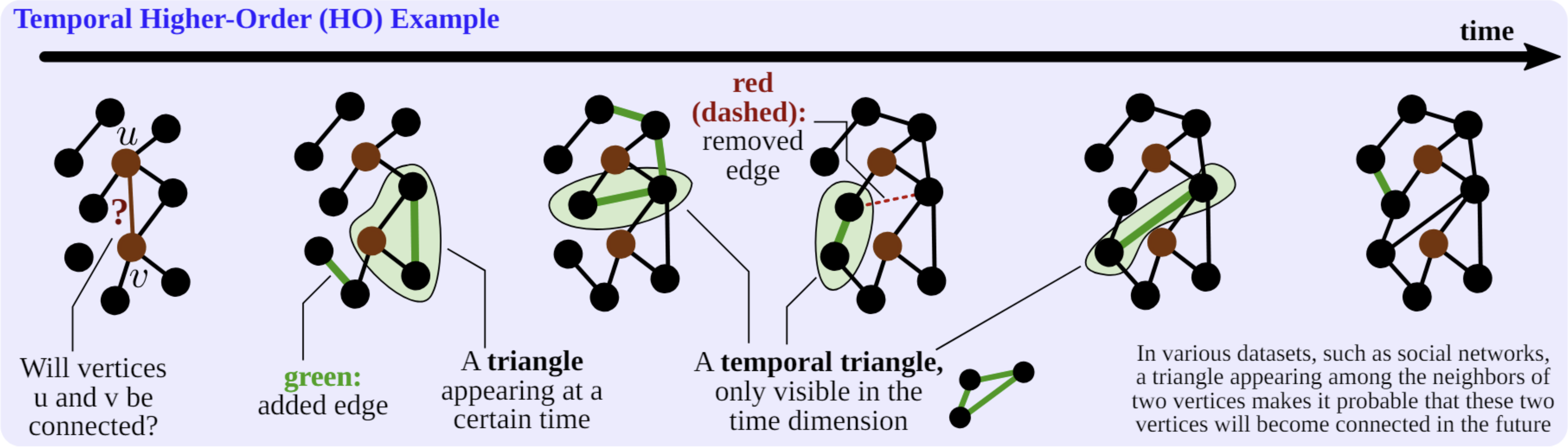
Temporal HO Structures



Temporal HO Structures



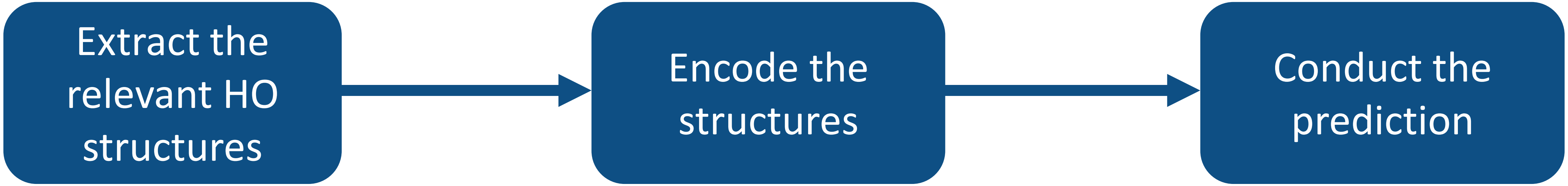
Temporal HO Structures



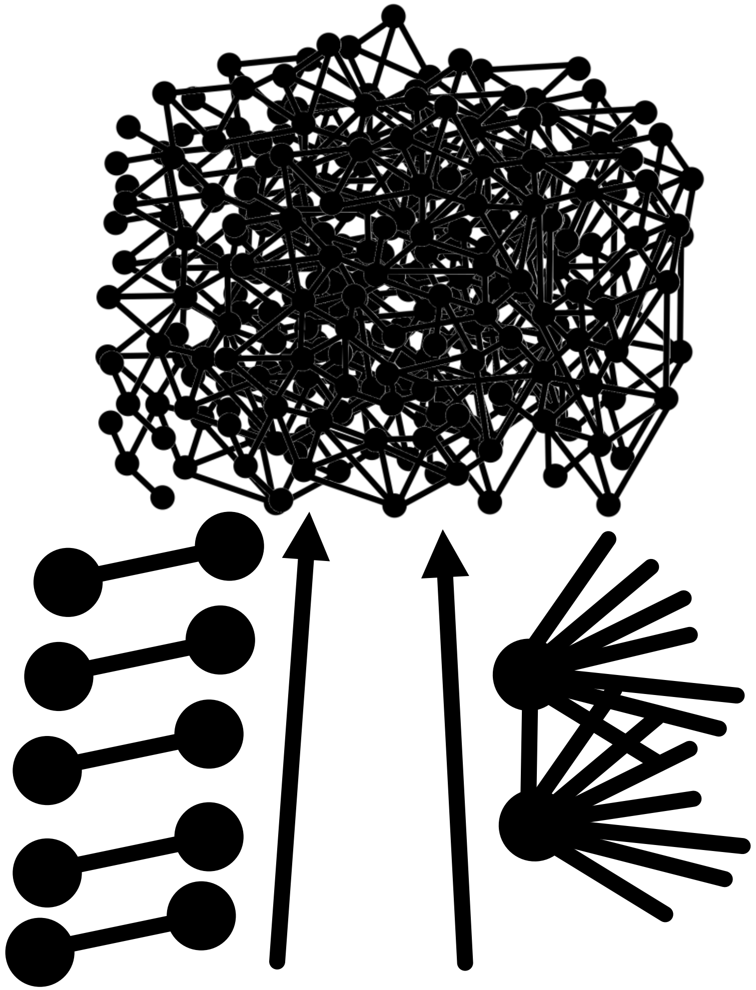
HO-Enhanced Pipeline for Dynamic Link Prediction

Extract the relevant HO structures → Encode the structures → Conduct the prediction

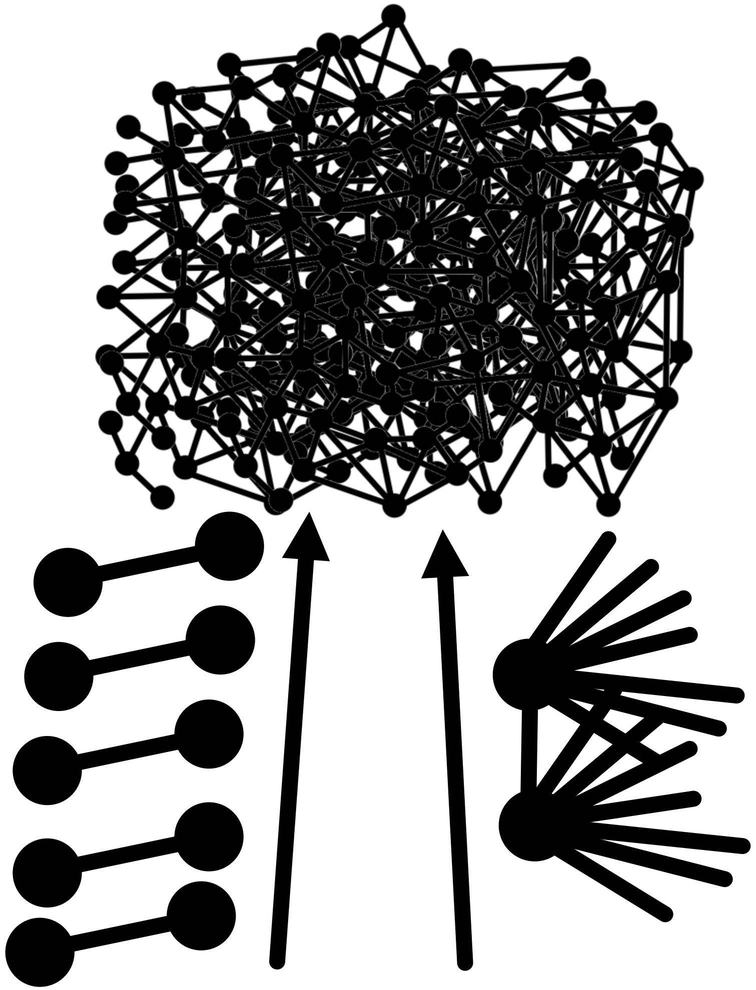
HO-Enhanced Pipeline for Dynamic Link Prediction



Formal Setting of Dynamic Link Prediction

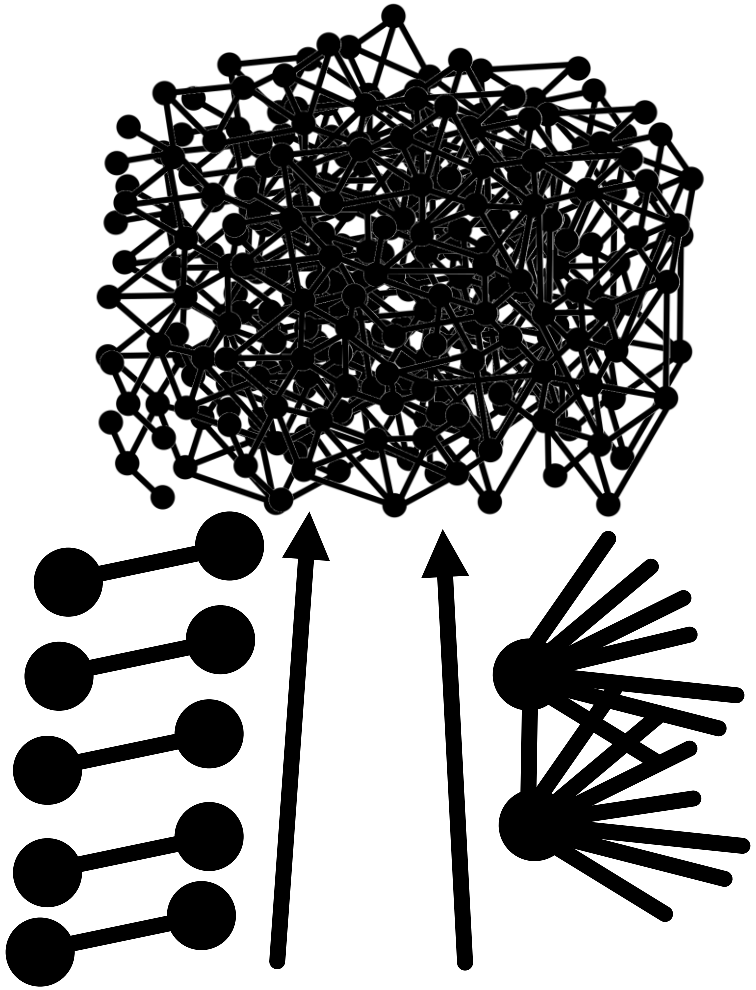


Formal Setting of Dynamic Link Prediction



Continuous-Time Dynamic Graph (**CTDG**) representation is a tuple (G(0) , T), where...

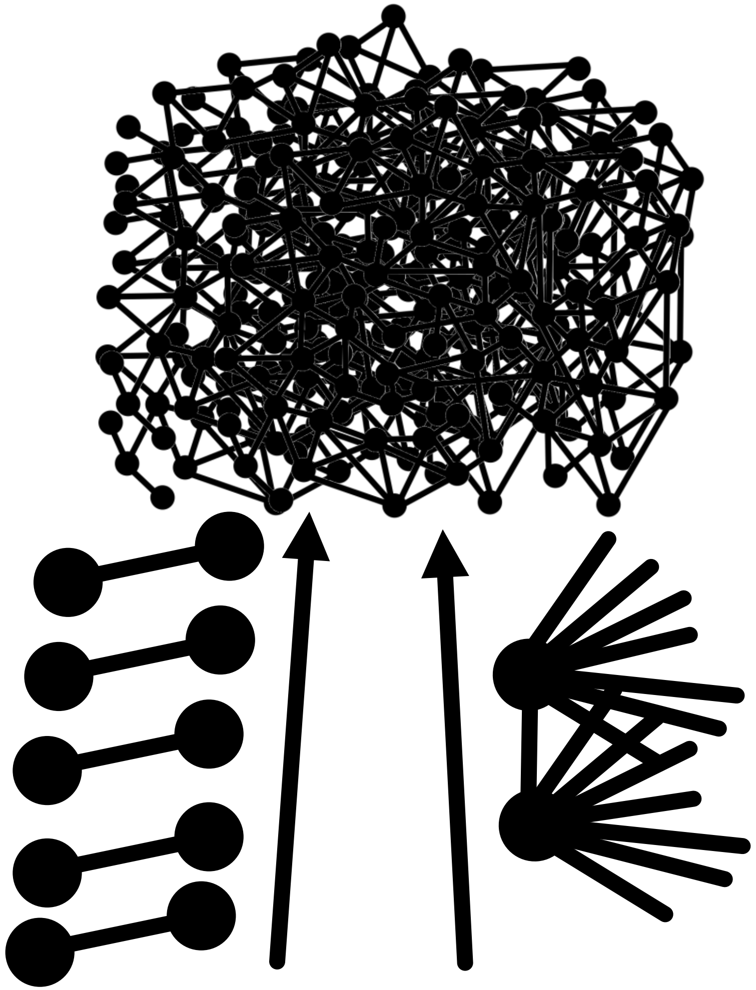
Formal Setting of Dynamic Link Prediction



Continuous-Time Dynamic Graph (**CTDG**) representation is a tuple (G(0) , T), where...

... G(0) = (V(0) , E(0) , **f**(0) , **w**(0)) represents the initial state of the graph

Formal Setting of Dynamic Link Prediction

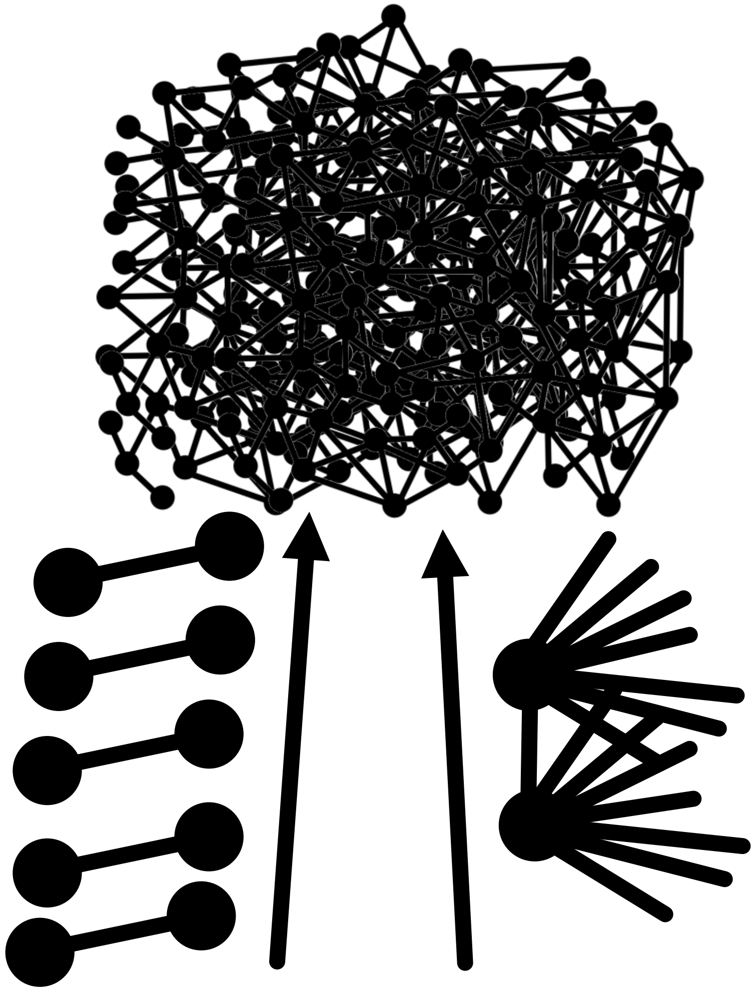


Continuous-Time Dynamic Graph (**CTDG**) representation is a tuple (G(0) , T), where...

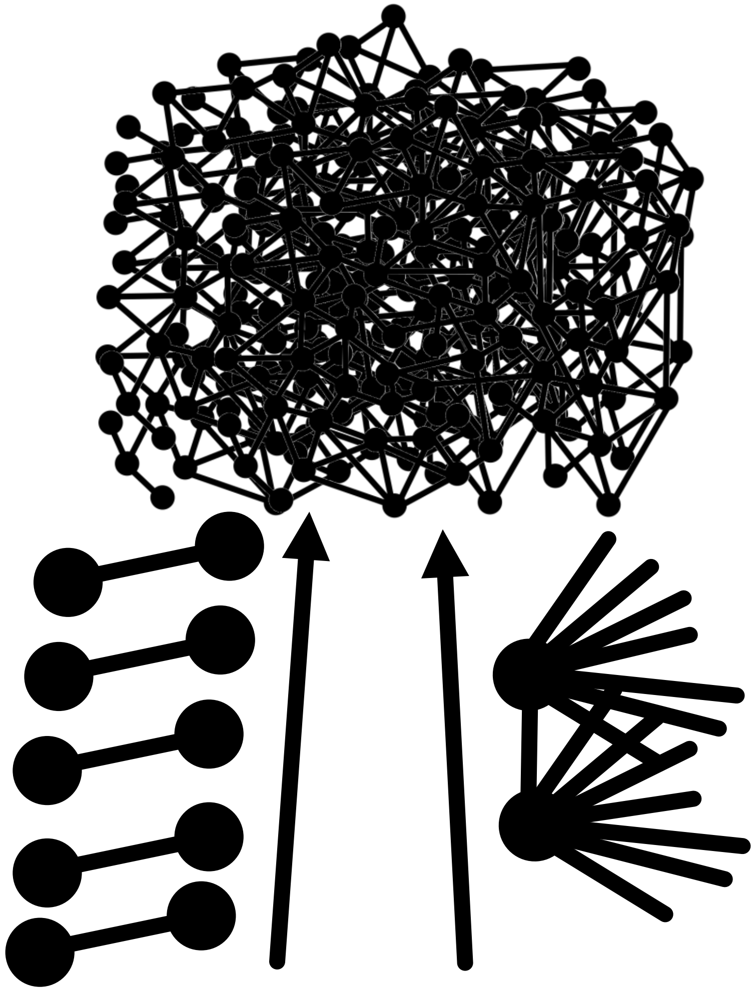
... G(0) = (V(0) , E(0) , **f**(0) , **w**(0)) represents the initial state of the graph

...T is a set of tuples of the form (timestamp, event) representing events to be applied to the graph at given timestamps.

Formal Setting of Dynamic Link Prediction

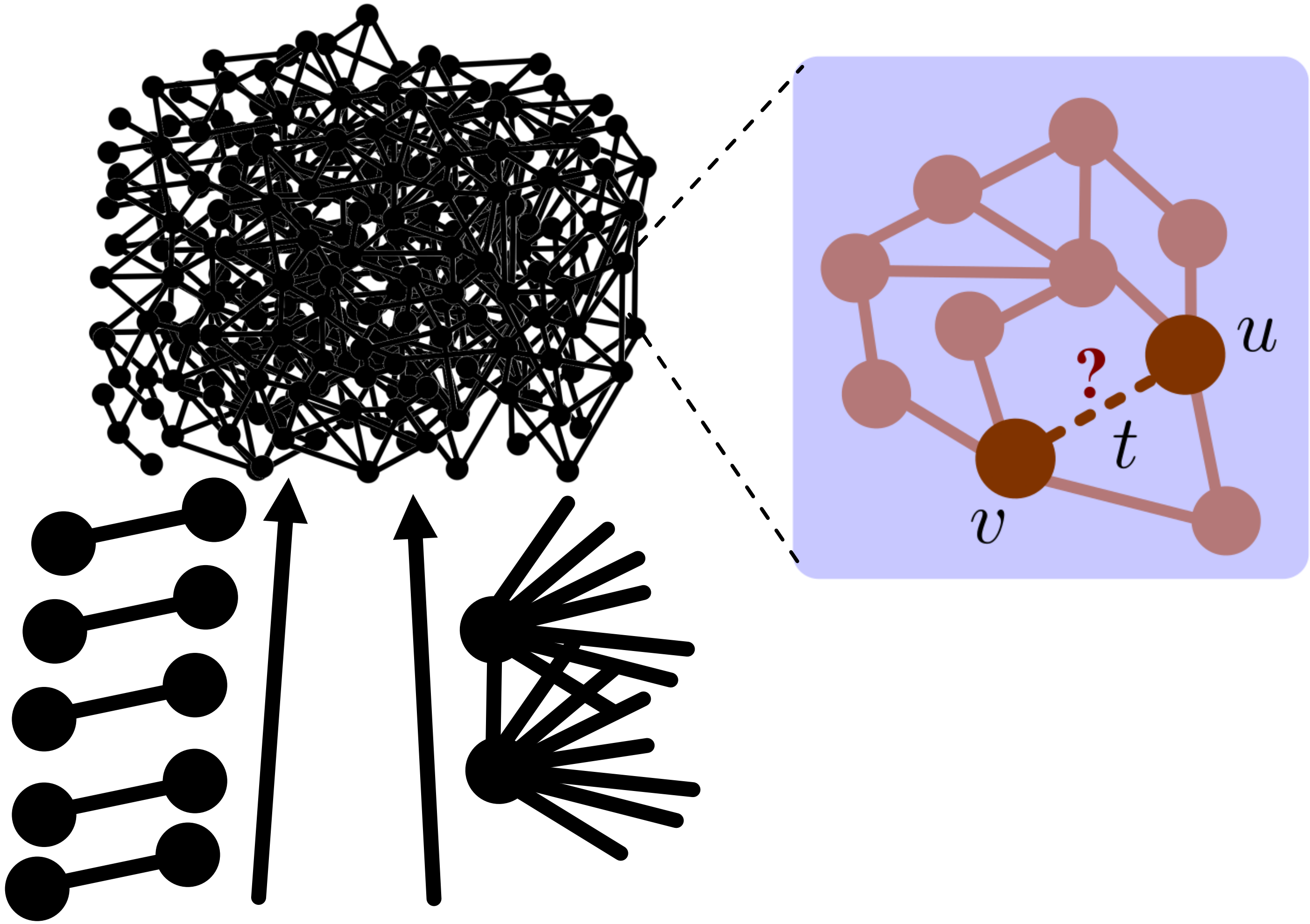


Formal Setting of Dynamic Link Prediction



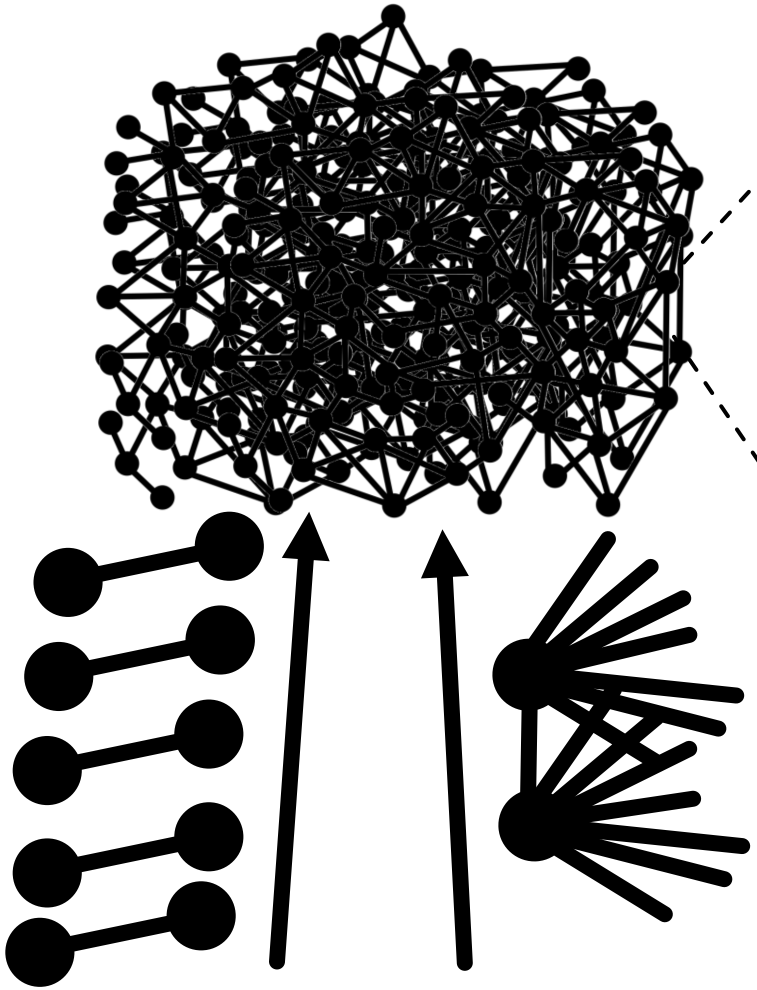
Assume:
- a CTDG (G(0) , T),
- a timestamp t ∈ N,
- an edge (u, v).

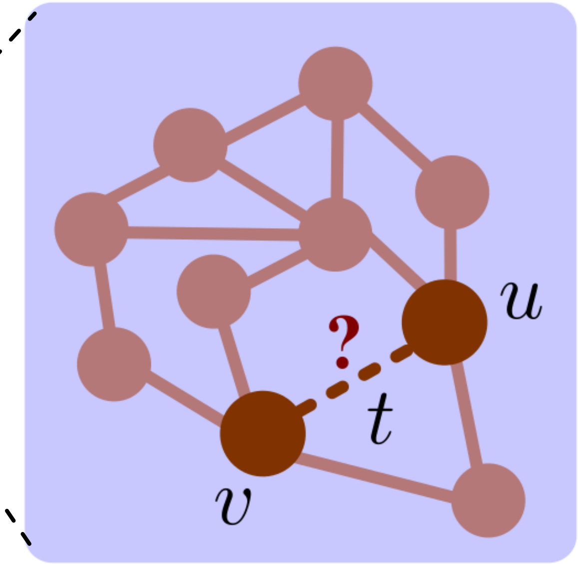
Formal Setting of Dynamic Link Prediction



Assume:
- a CTDG (G(0) , T),
- a timestamp t ∈ N,
- an edge (u, v).

Formal Setting of Dynamic Link Prediction

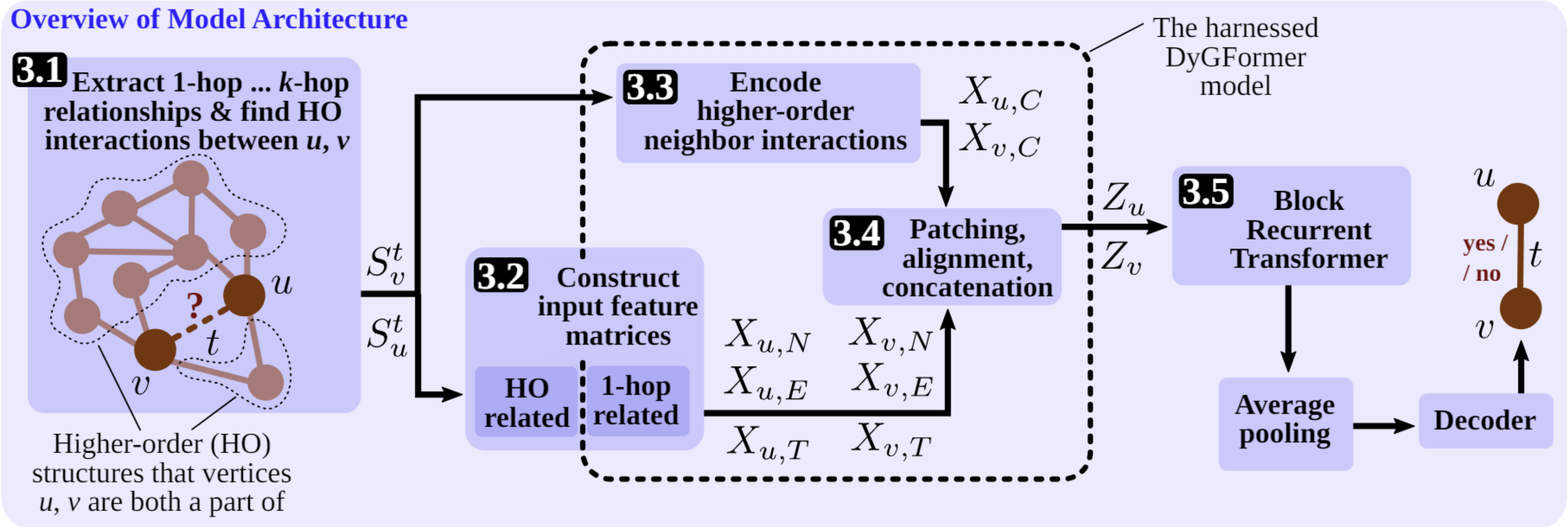




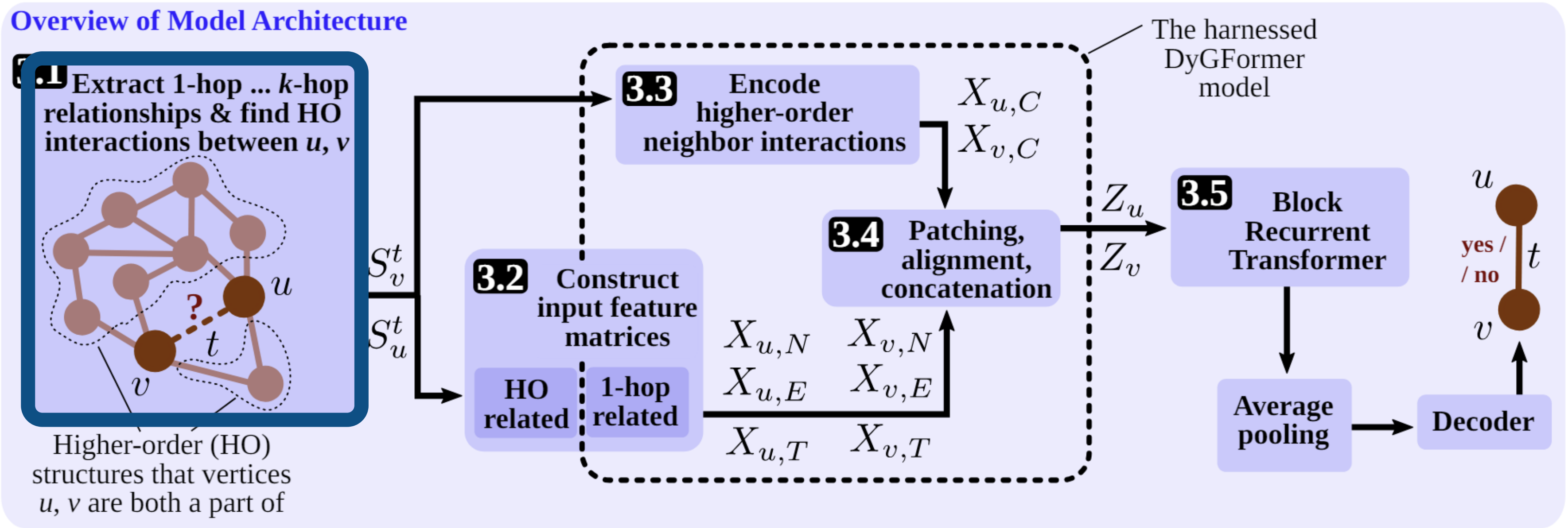
Assume:
- a CTDG (G(0) , T),
- a timestamp t ∈ N,
- an edge (u, v).

The task is to decide, whether there is some event e pertaining to (u, v), such that (t, e) ∈ { (t ′ , e) ∈ T | t ′ = t }, while only considering the CTDG (G(0), { (t ′ , e) ∈ T | t ′ < t }).

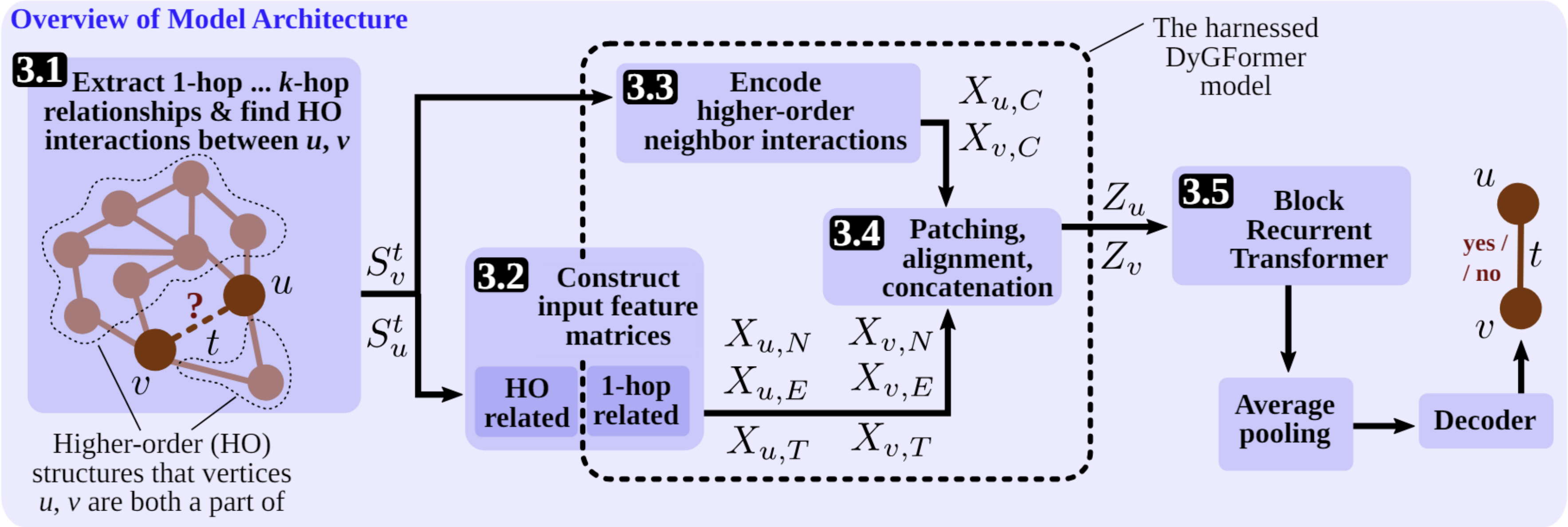
Encoding HO Structures



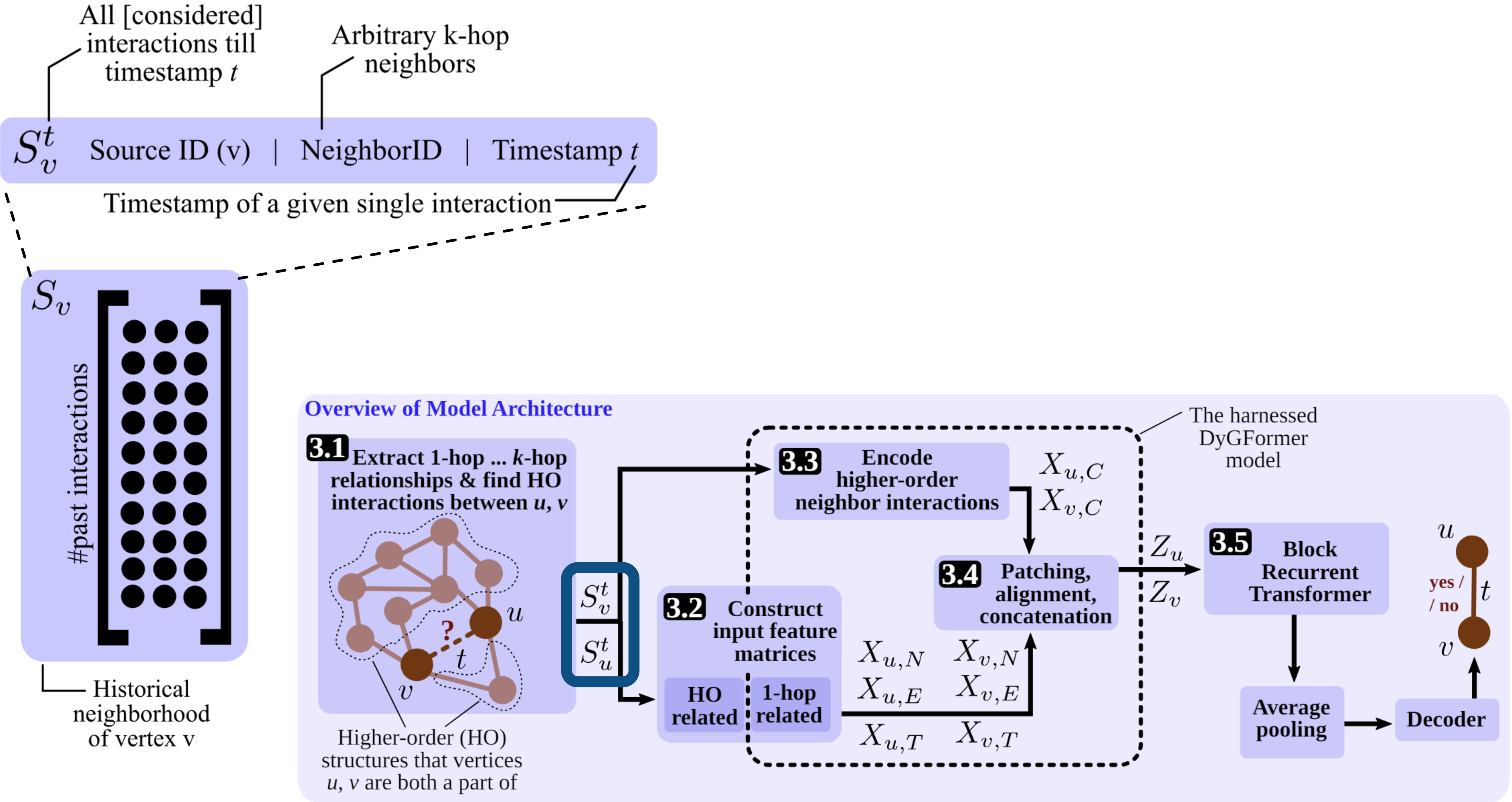
Encoding HO Structures



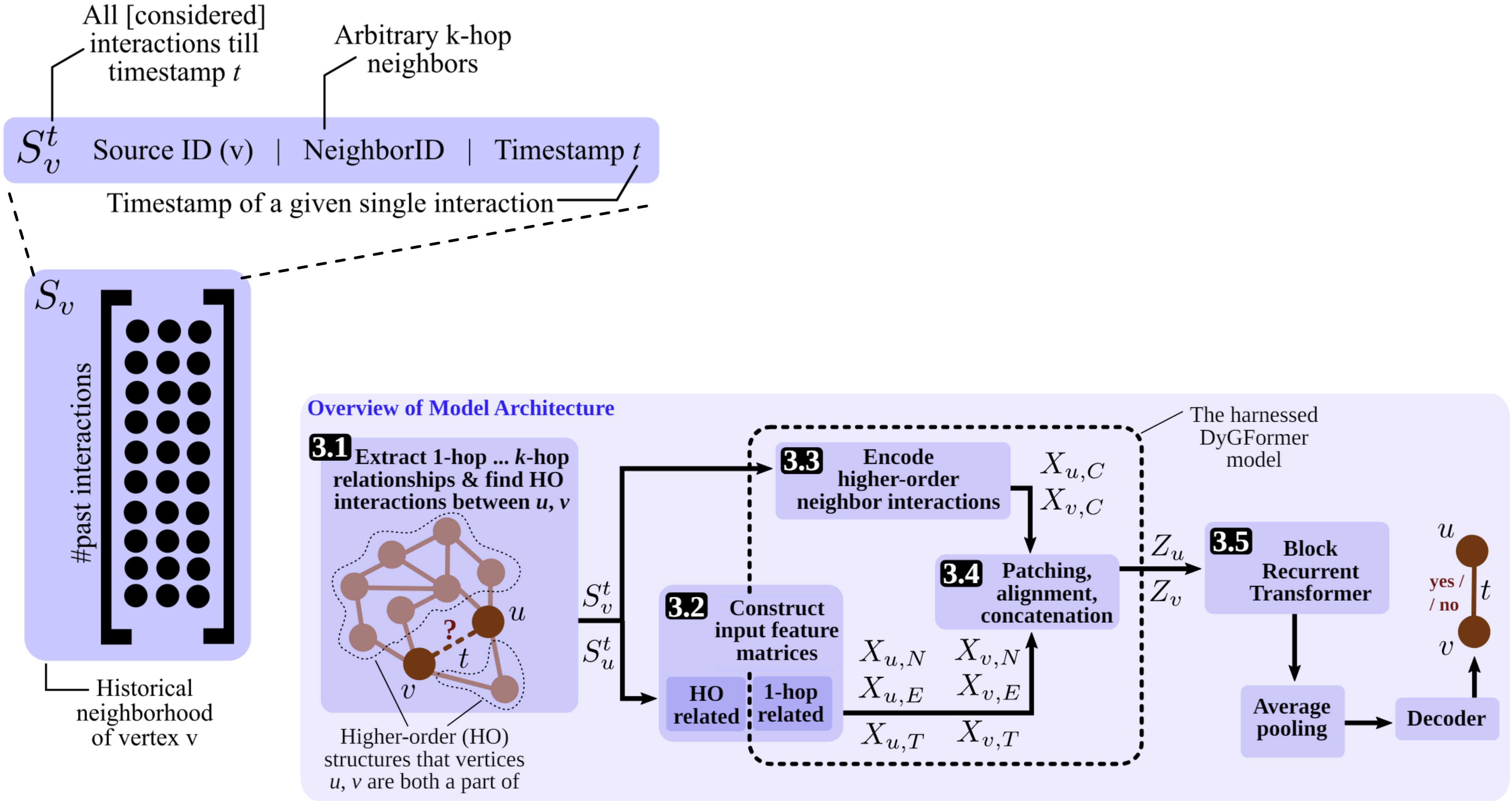
Encoding HO Structures



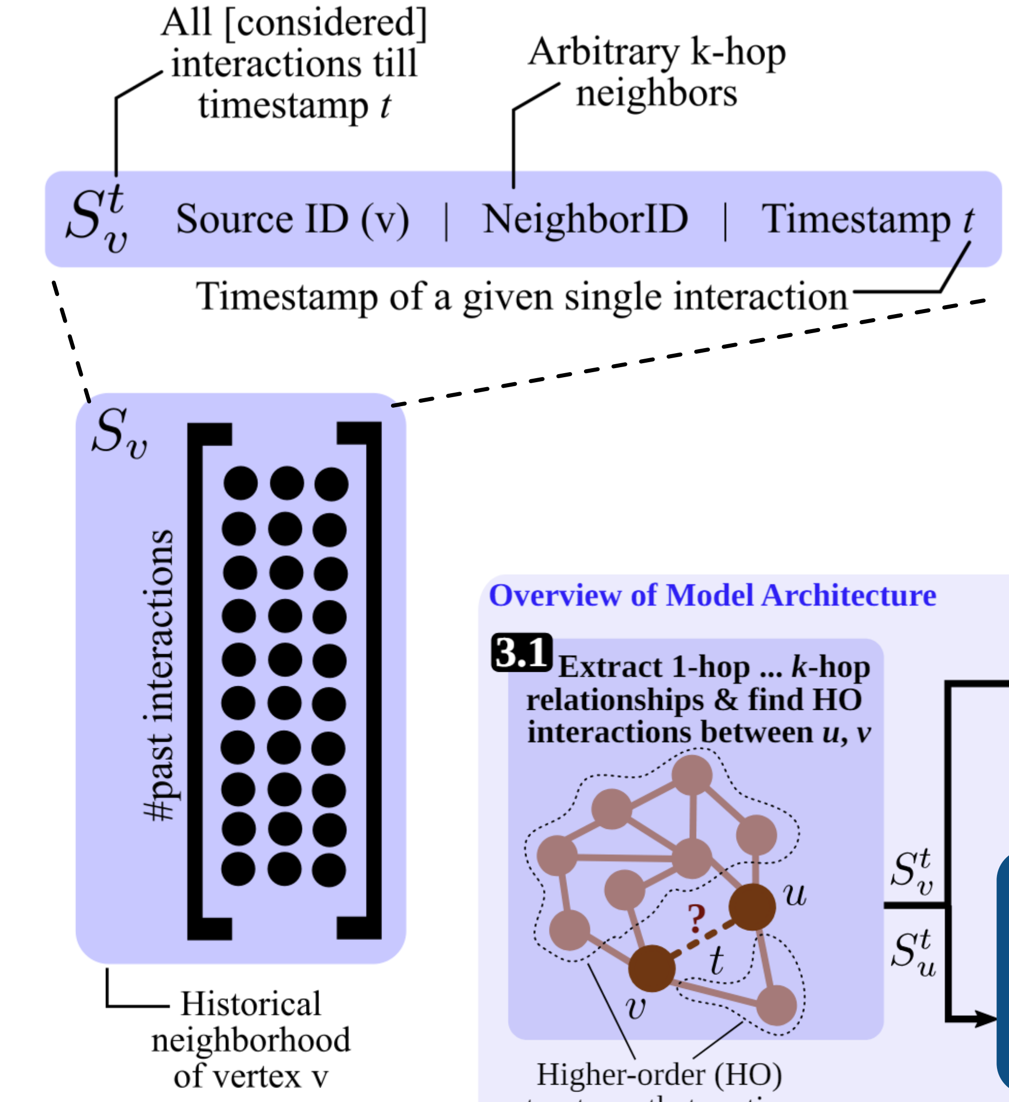
Encoding HO Structures

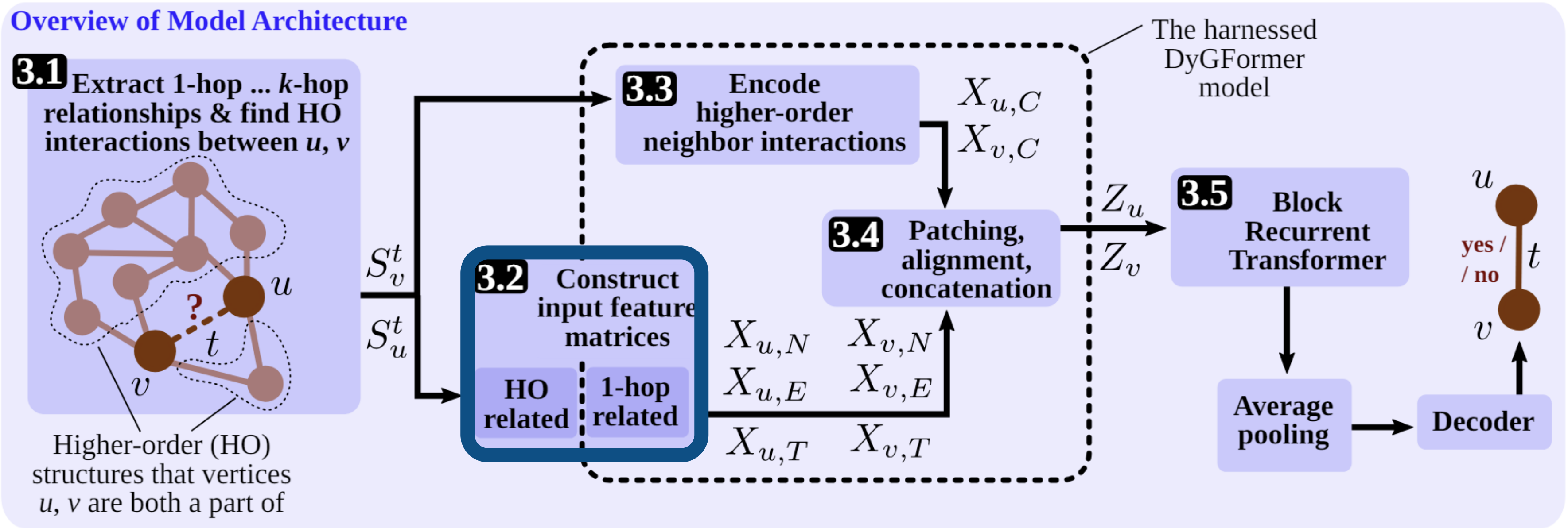


Encoding HO Structures

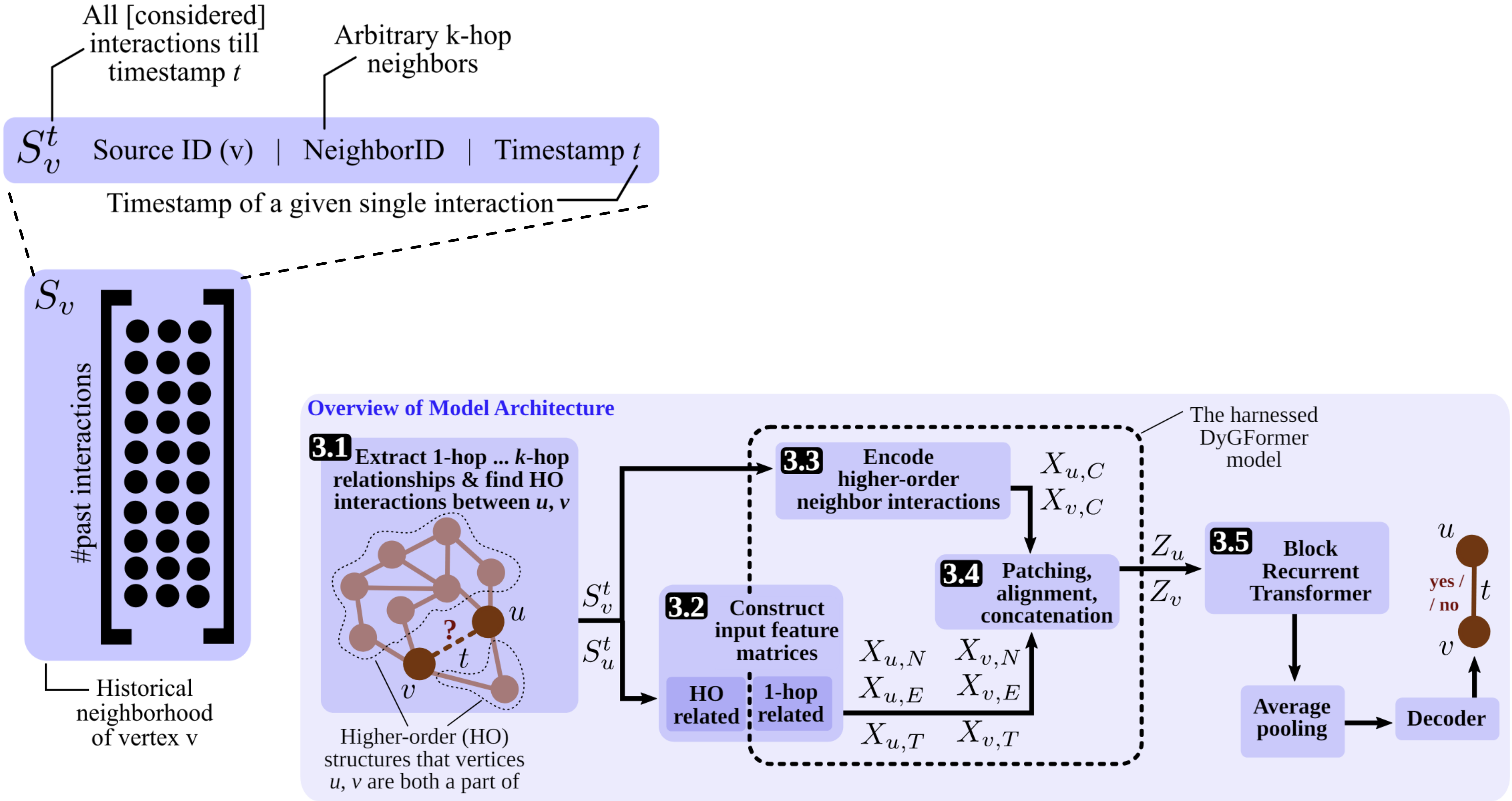


Encoding HO Structures

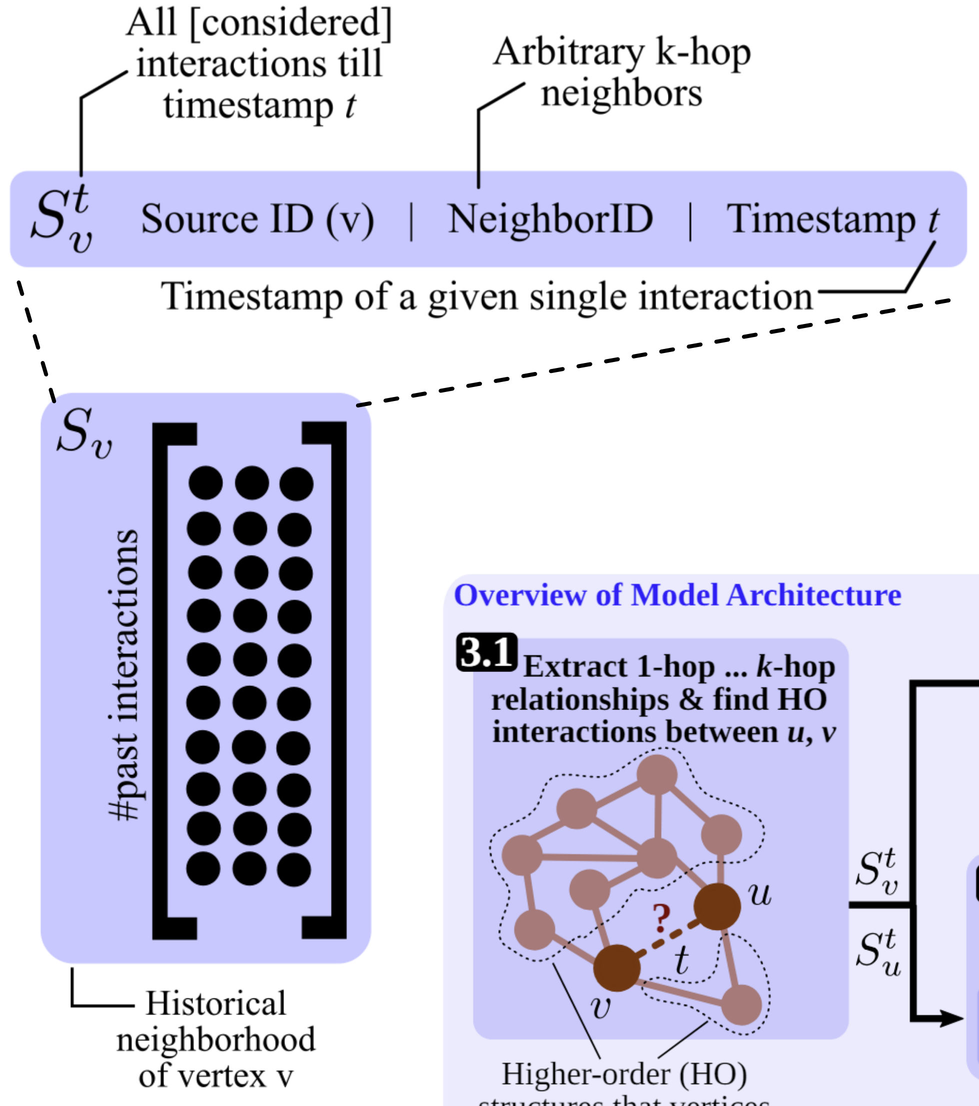


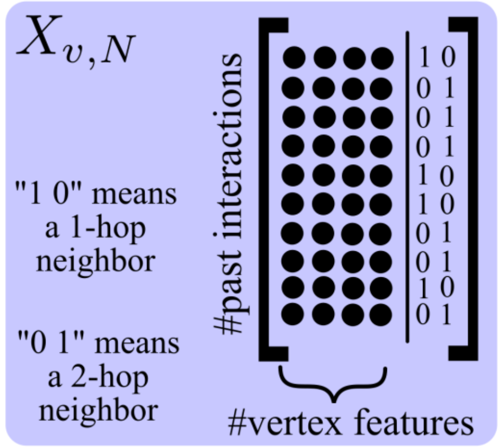


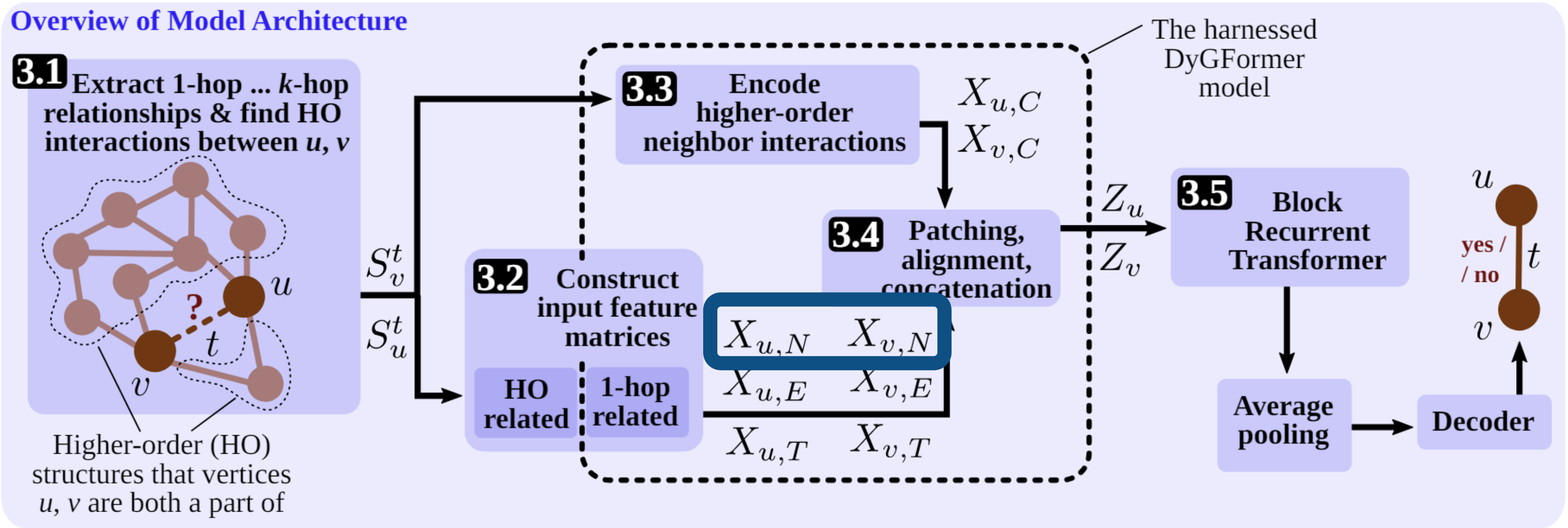
Encoding HO Structures



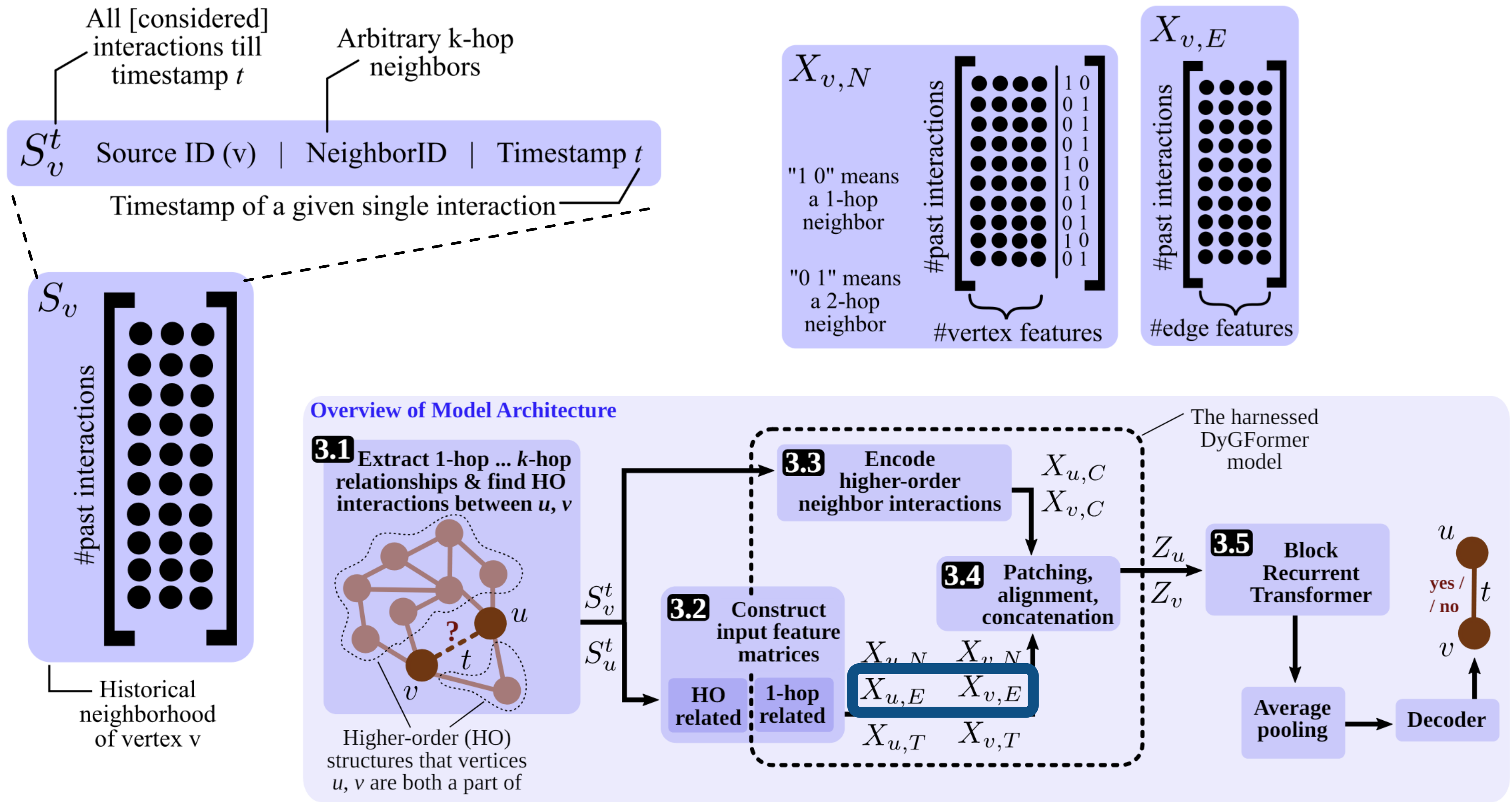
Encoding HO Structures



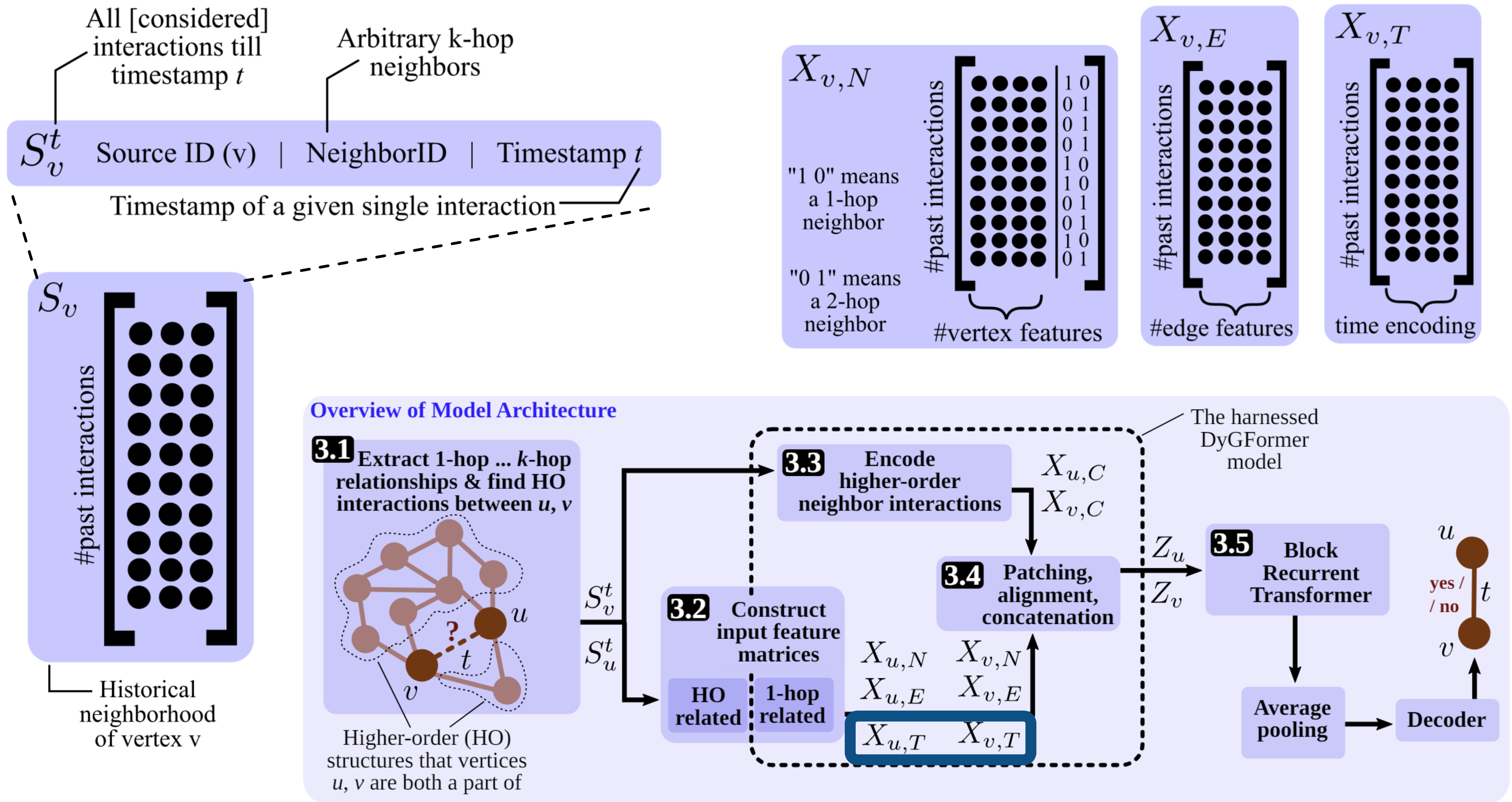




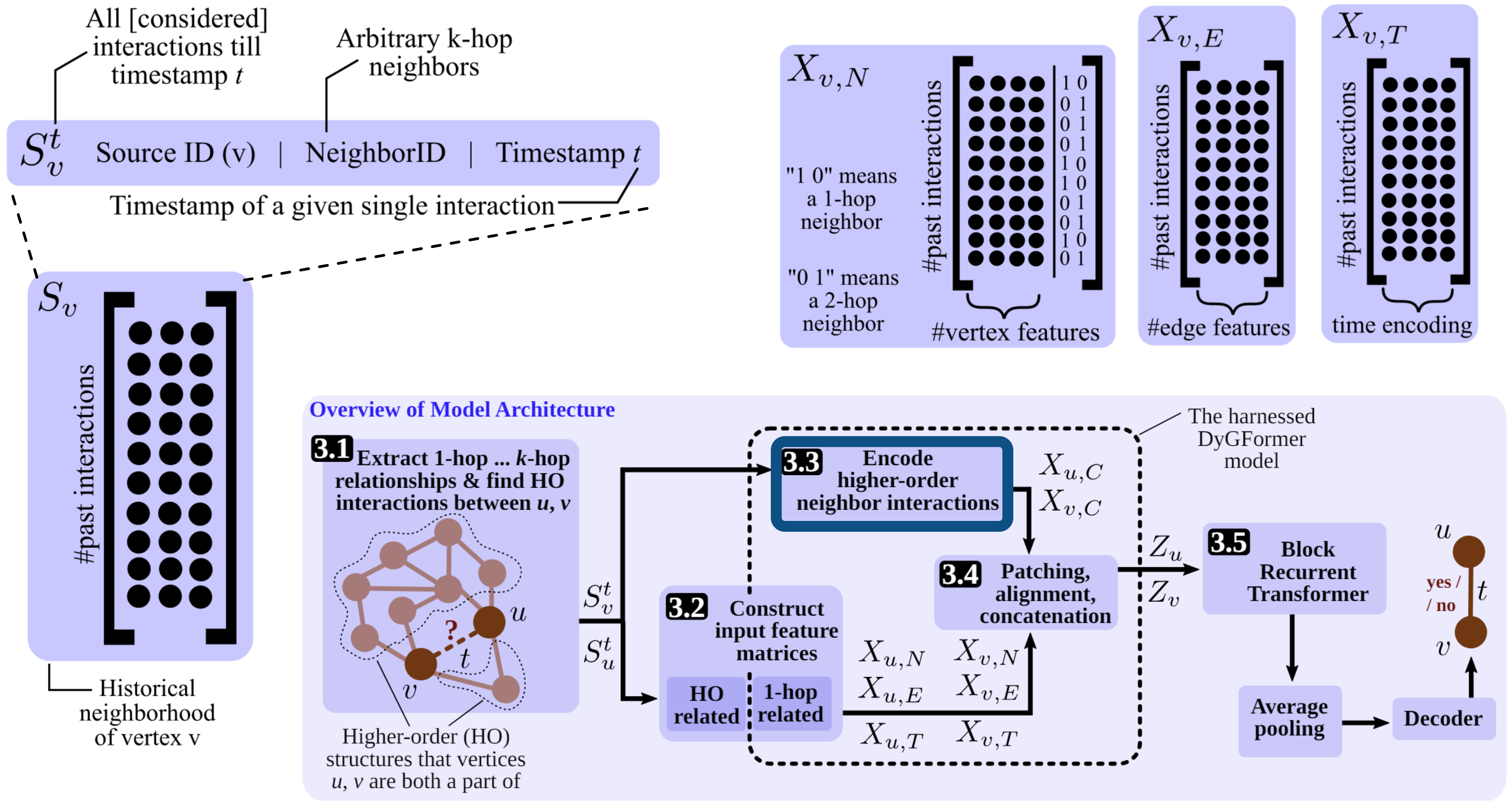
Encoding HO Structures



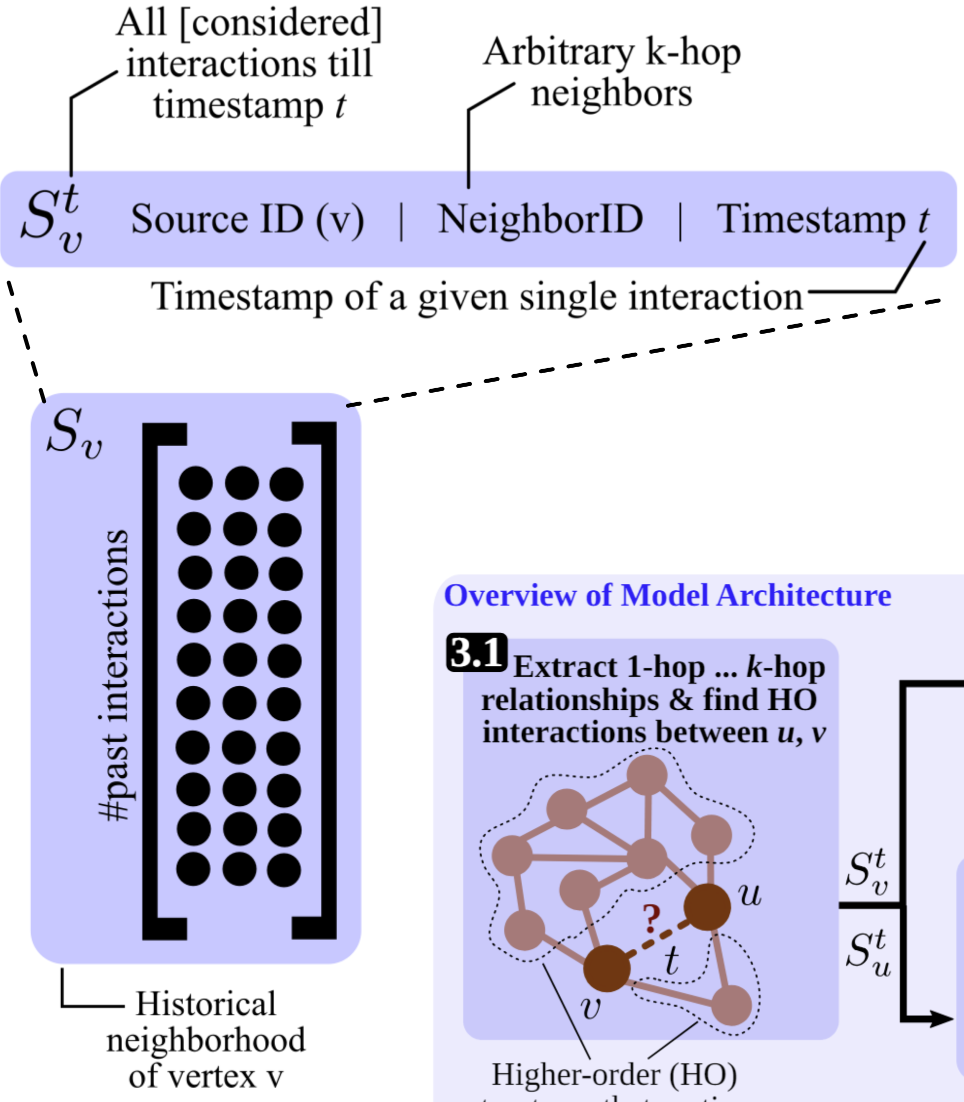
Encoding HO Structures

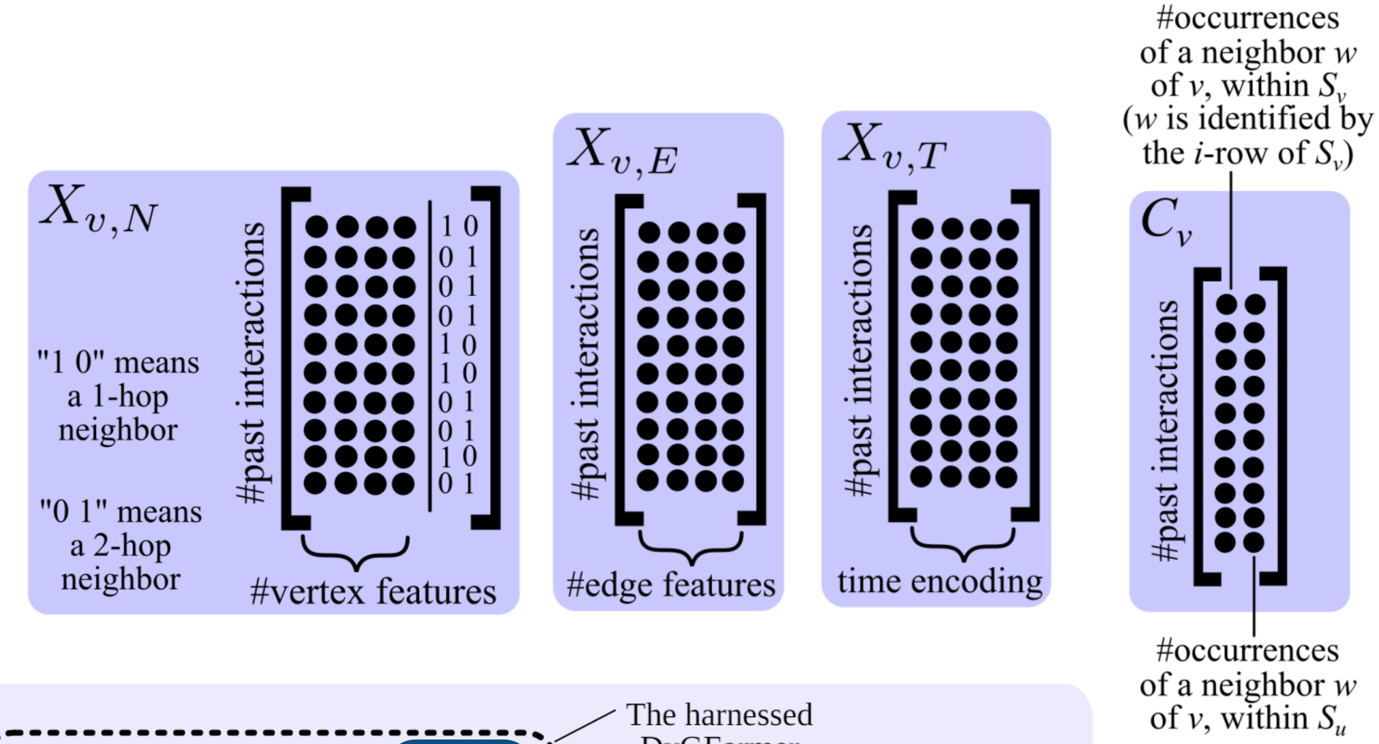


Encoding HO Structures

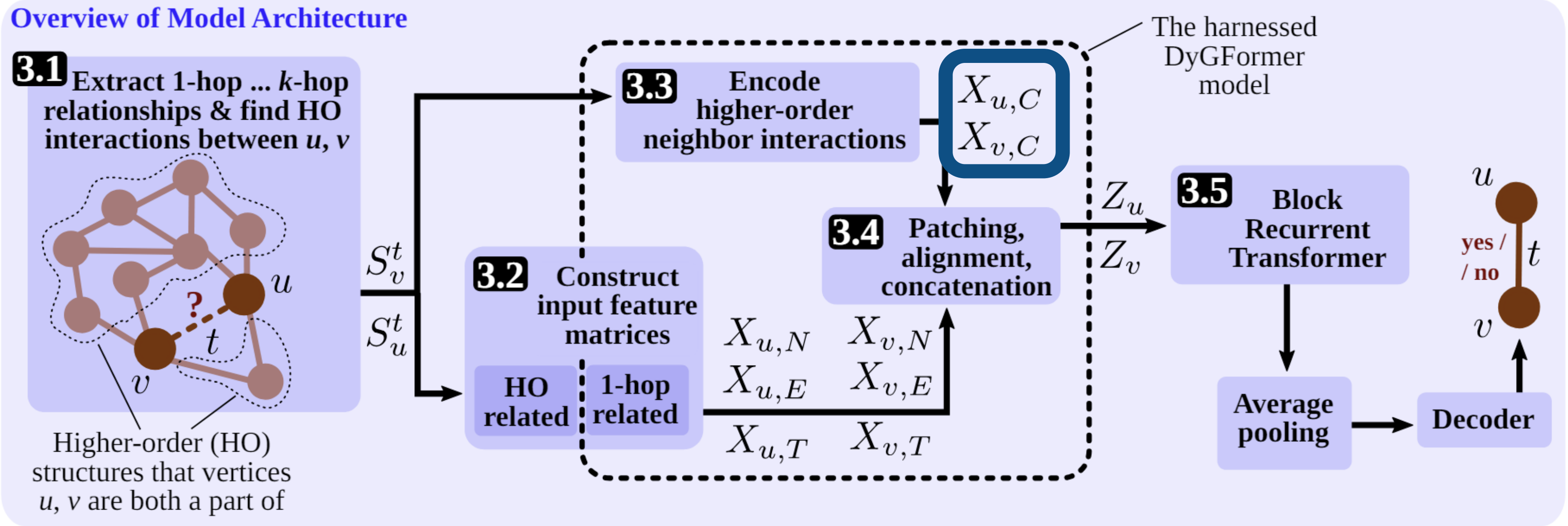


Encoding HO Structures

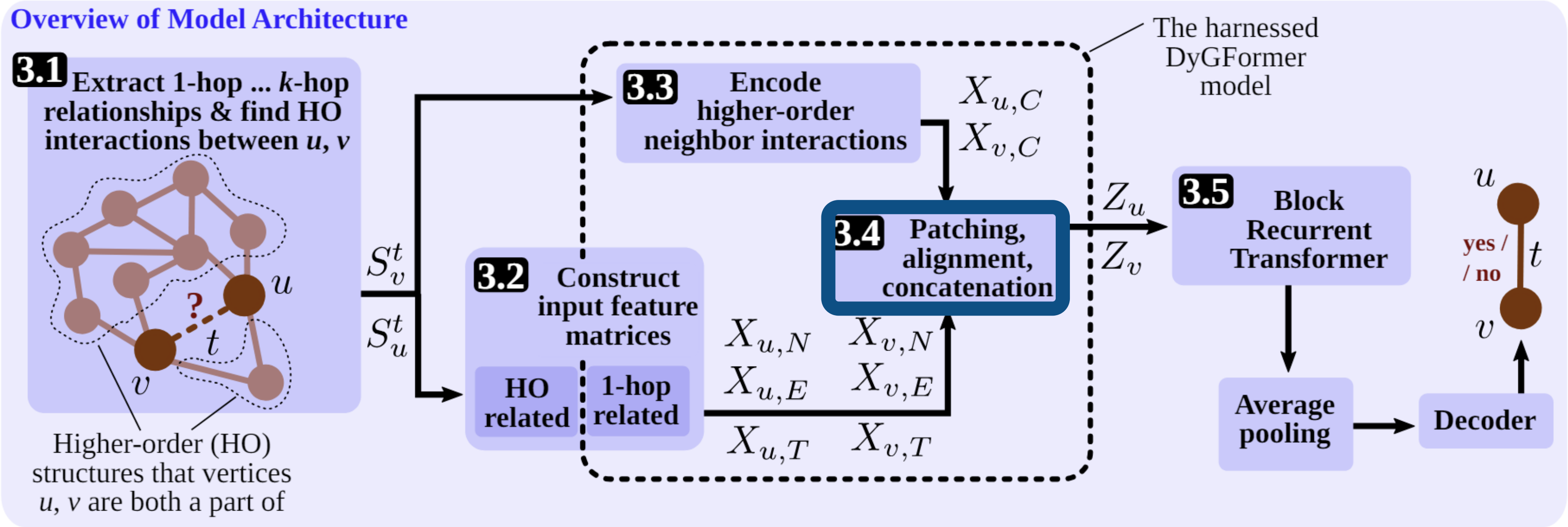




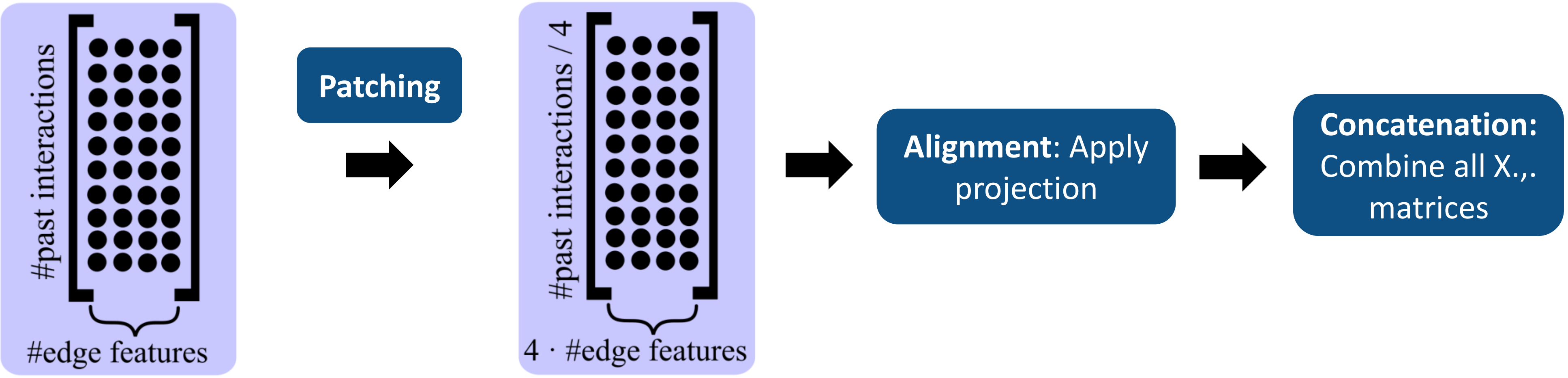
$X_{v,C}$ = $MLP(C_v)$

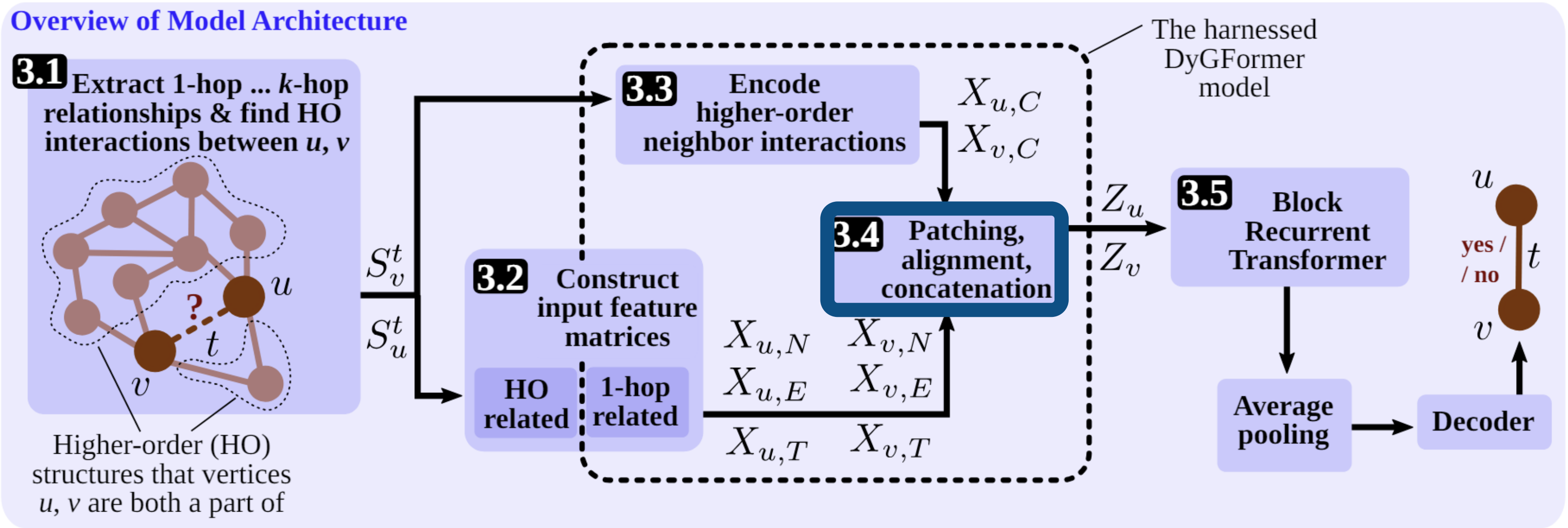


Encoding HO Structures

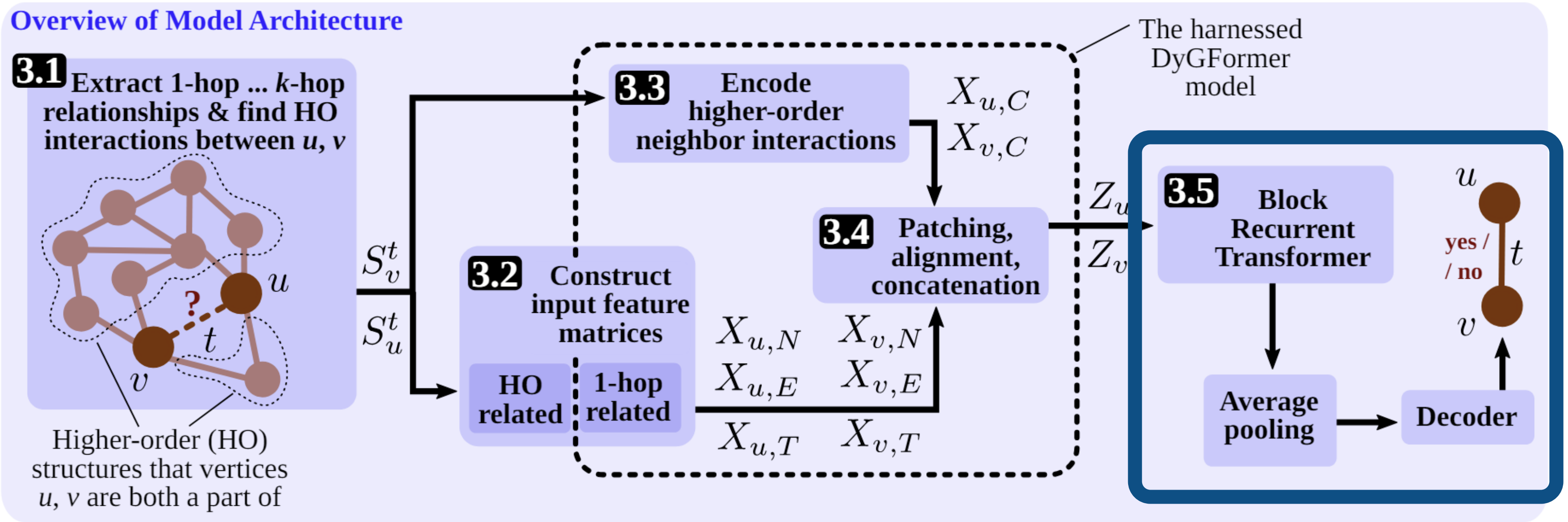


Encoding HO Structures



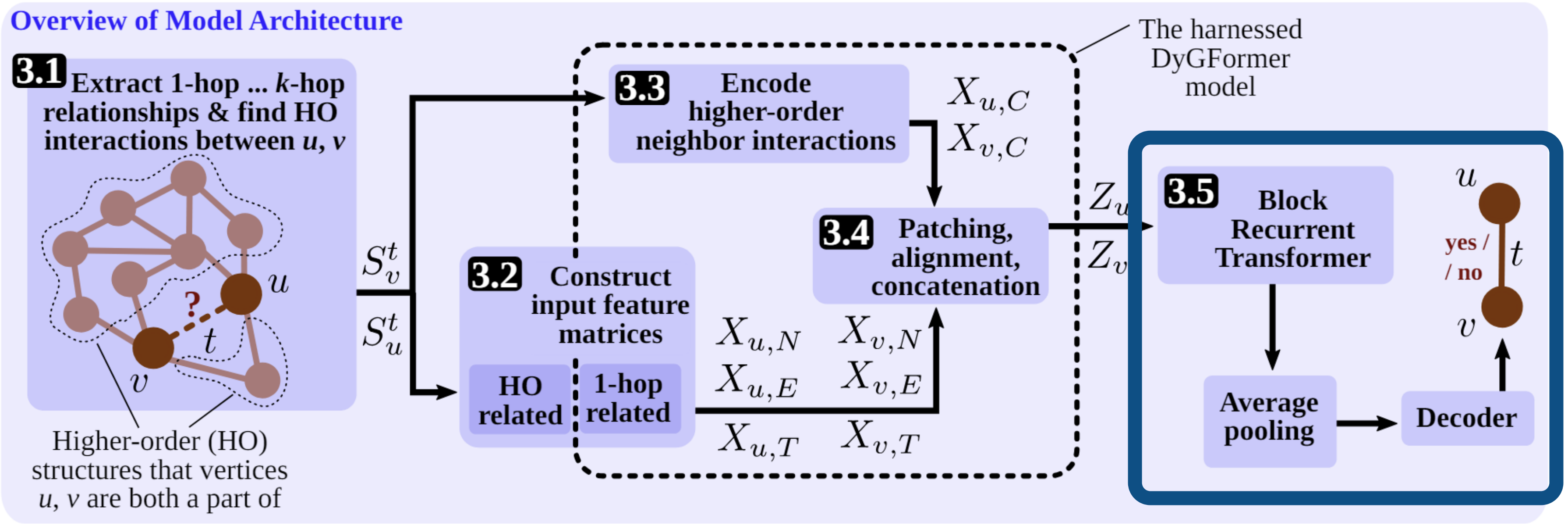


Making Predictions



Making Predictions

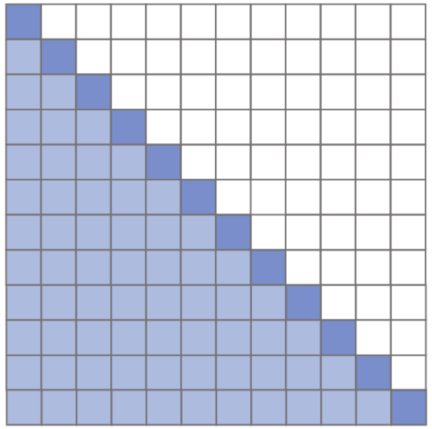
We use the module as a black box, any scheme in the domain of [efficient] Transformers could be applied



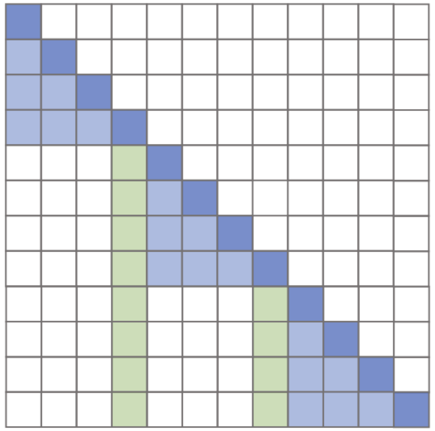
[1] Tay et al. Efficient Transformers: A Survey. 2020

Making Predictions

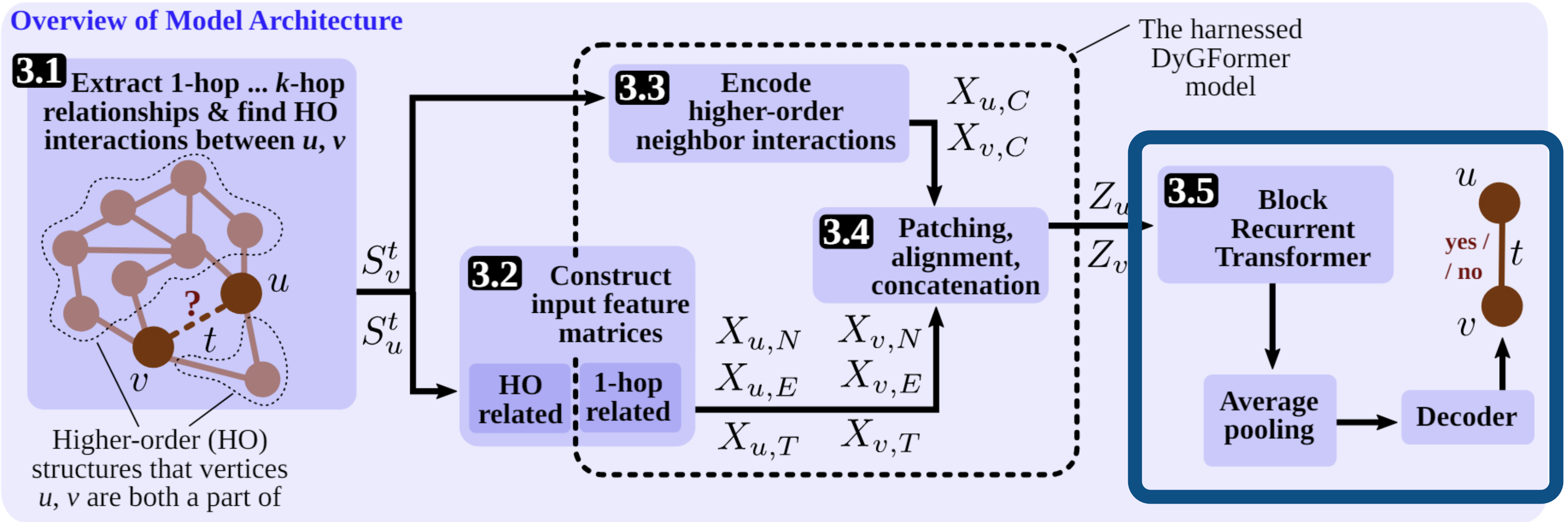
Attention matrix [1]



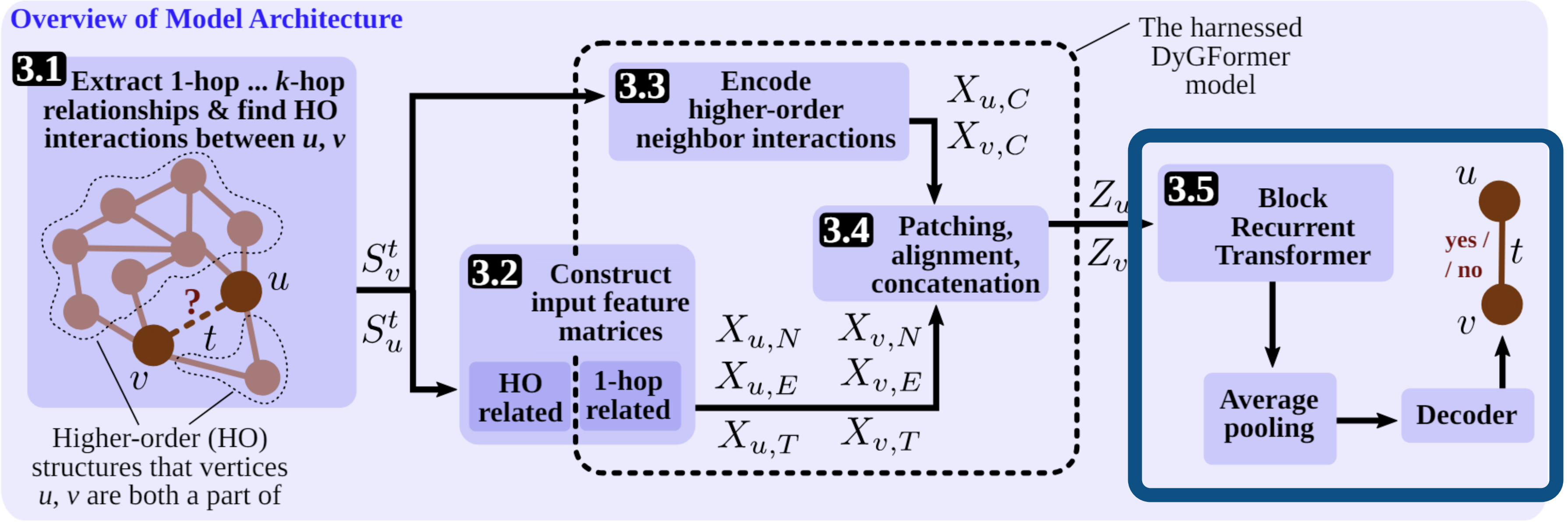
Sparse attention matrix [1]



We use the module as a black box, any scheme in the domain of [efficient] Transformers could be applied



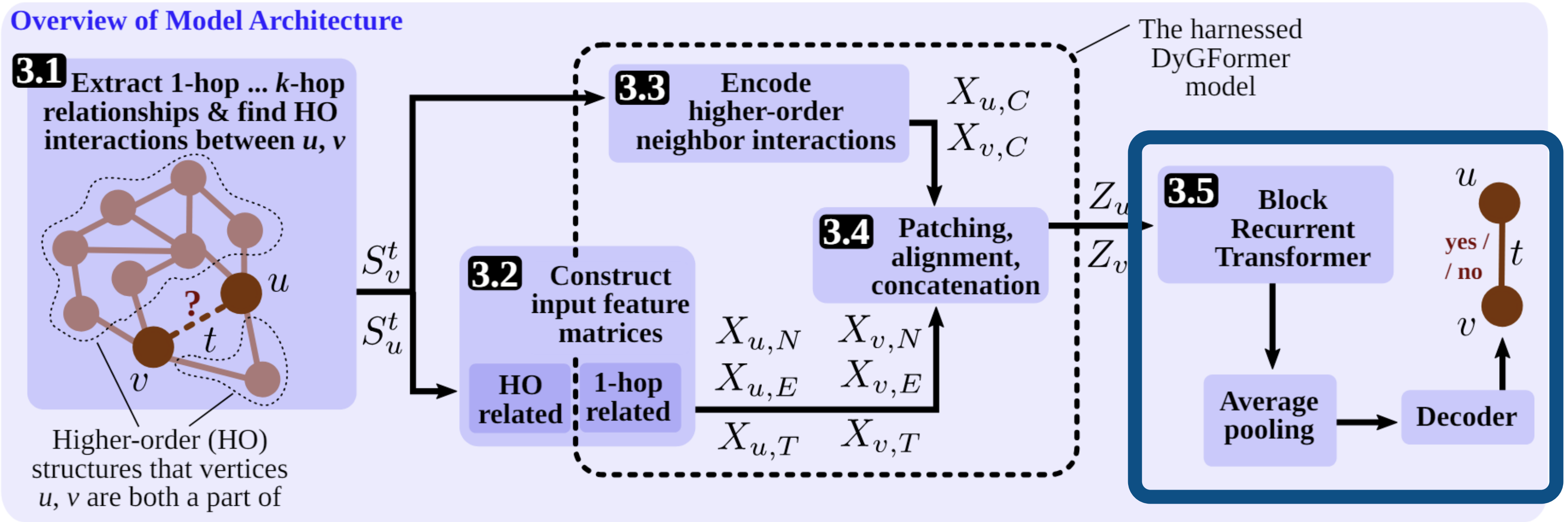
Making Predictions



[1] Hutchins et al. Block-Recurrent Transformers. NeurIPS 2022

Making Predictions

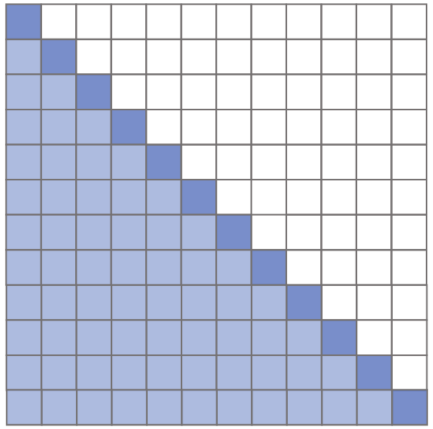
Scheme harnessed:
Block-Recurrent Transformer [1]



[1] Hutchins et al. Block-Recurrent Transformers. NeurIPS 2022

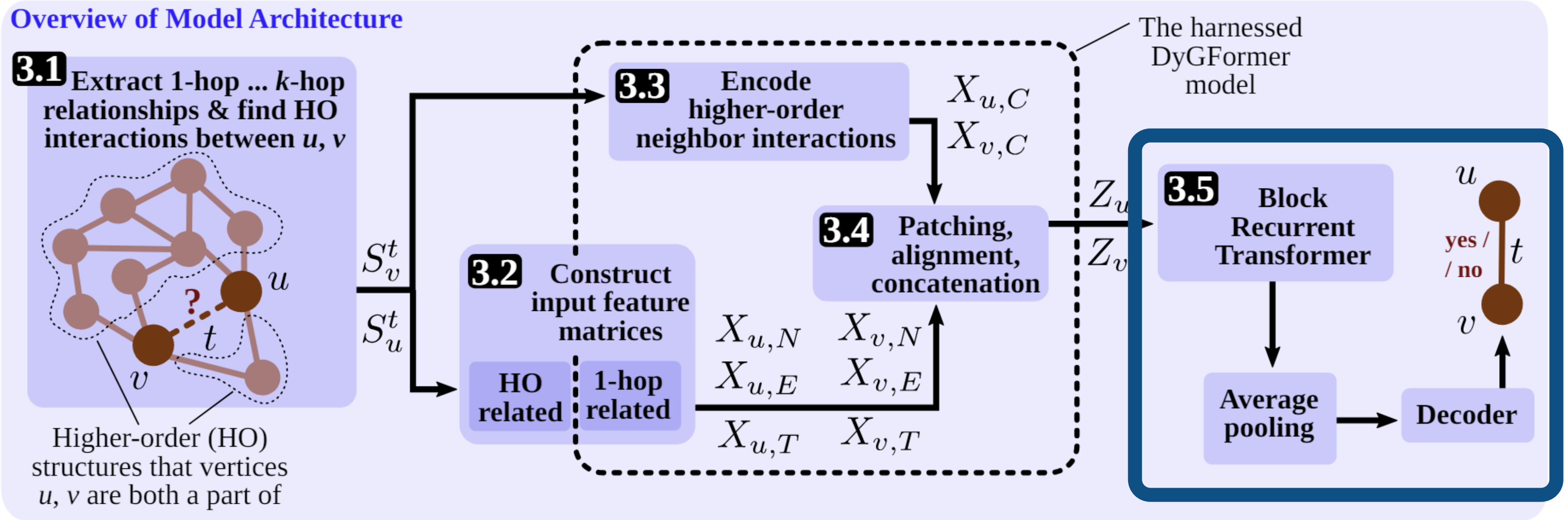
Making Predictions

Standard attention



Large storage requirements when #tokens is growing

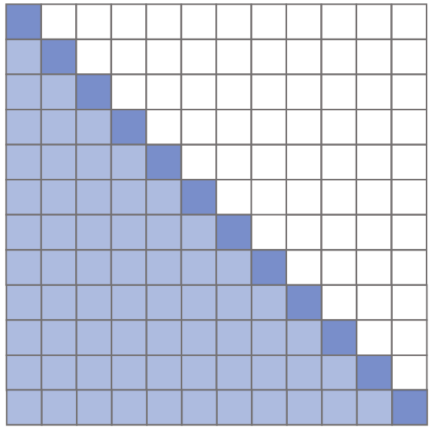
Scheme harnessed:
Block-Recurrent Transformer [1]



[1] Hutchins et al. Block-Recurrent Transformers. NeurIPS 2022

Making Predictions

Standard attention

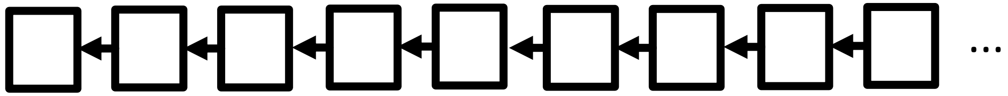


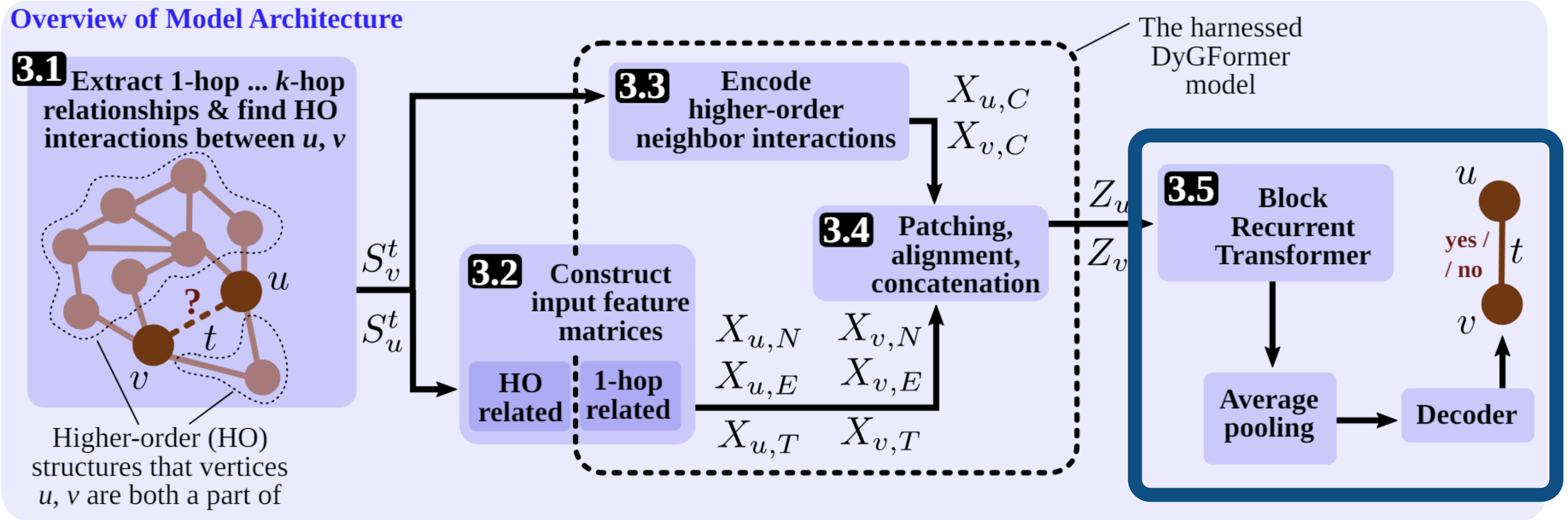
Large storage requirements when #tokens is growing

Scheme harnessed:
Block-Recurrent Transformer [1]

Low accuracy when #tokens is growing

Standard RNN

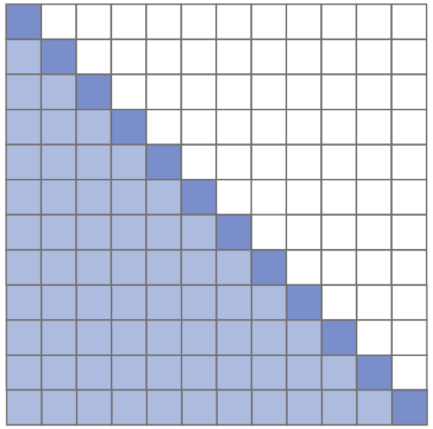




[1] Hutchins et al. Block-Recurrent Transformers. NeurIPS 2022

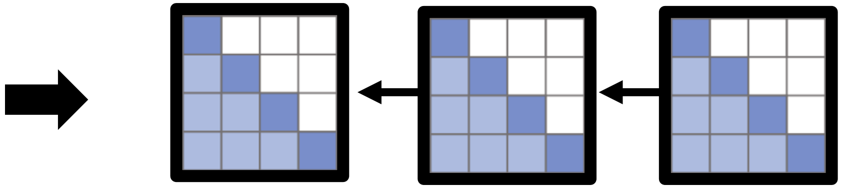
Making Predictions

Standard attention



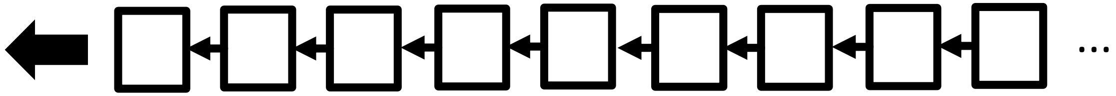
Large storage requirements when #tokens is growing

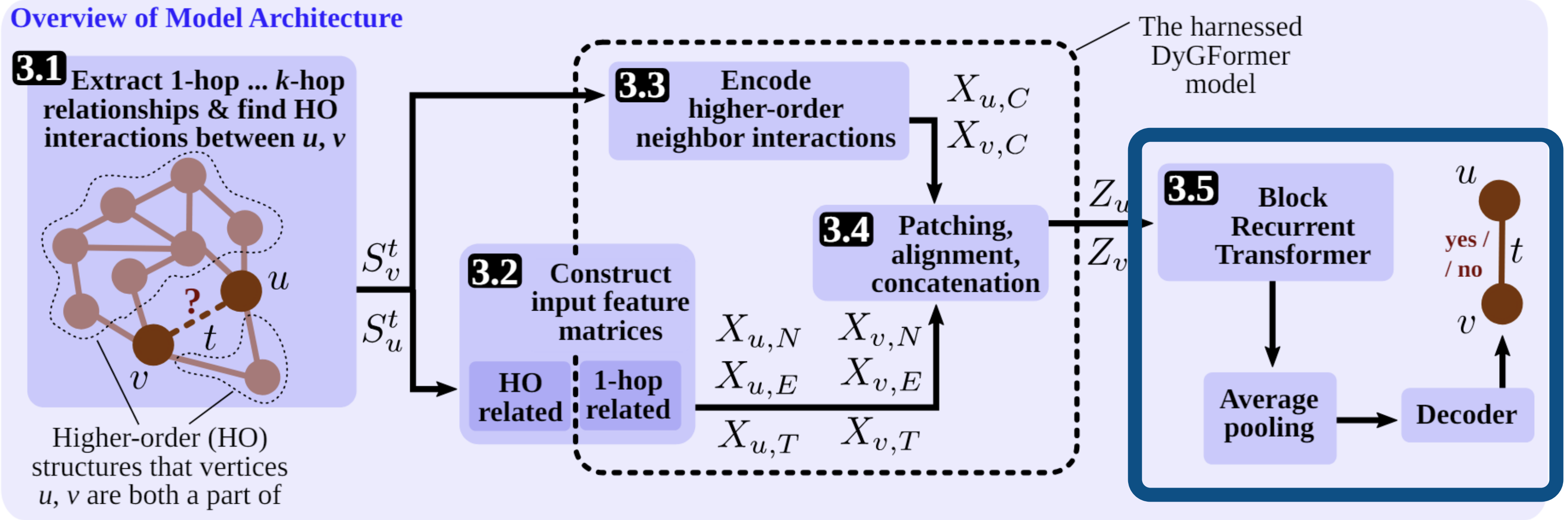
Scheme harnessed:
Block-Recurrent Transformer [1]



Standard RNN

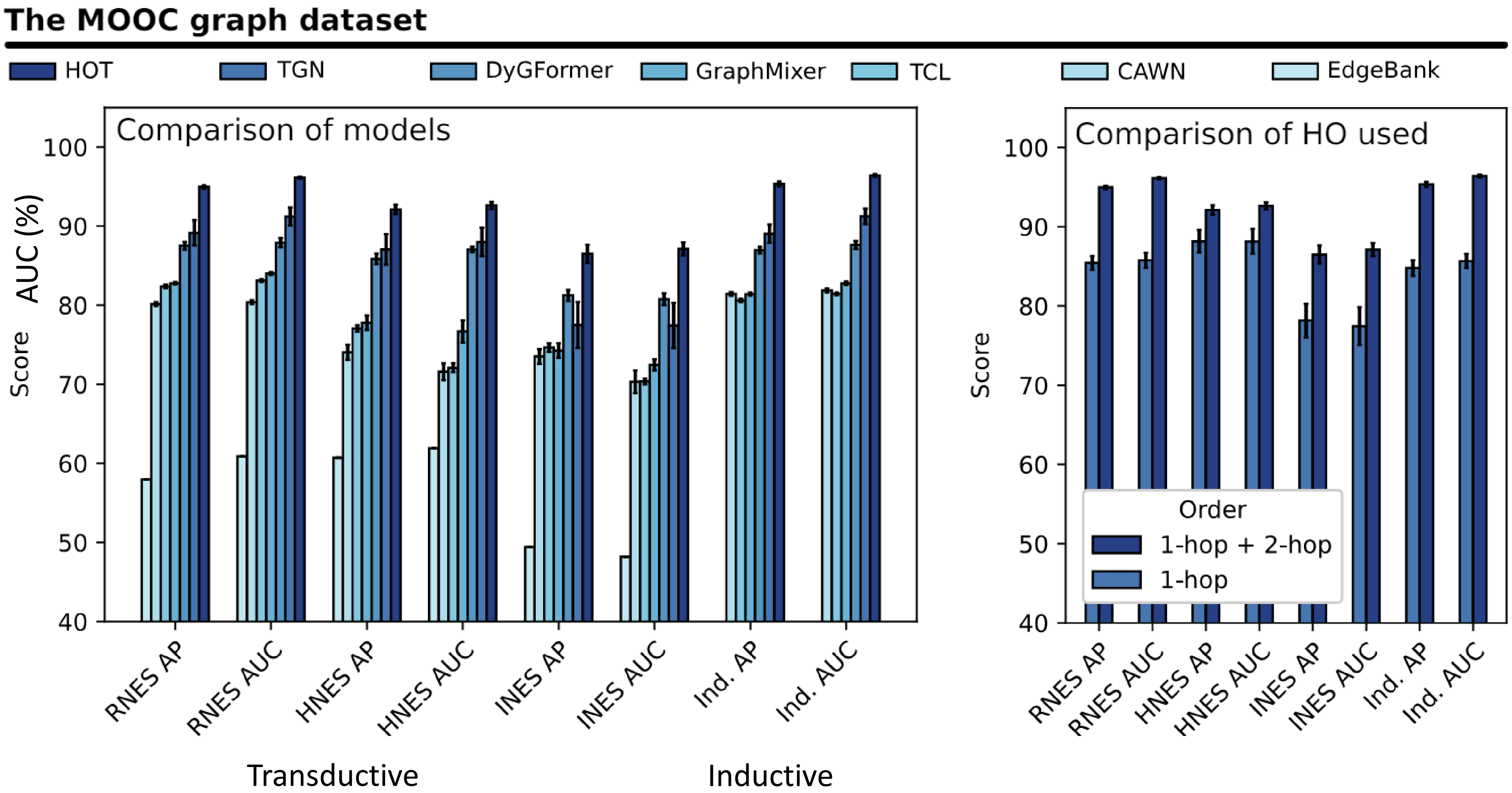
Low accuracy when #tokens is growing





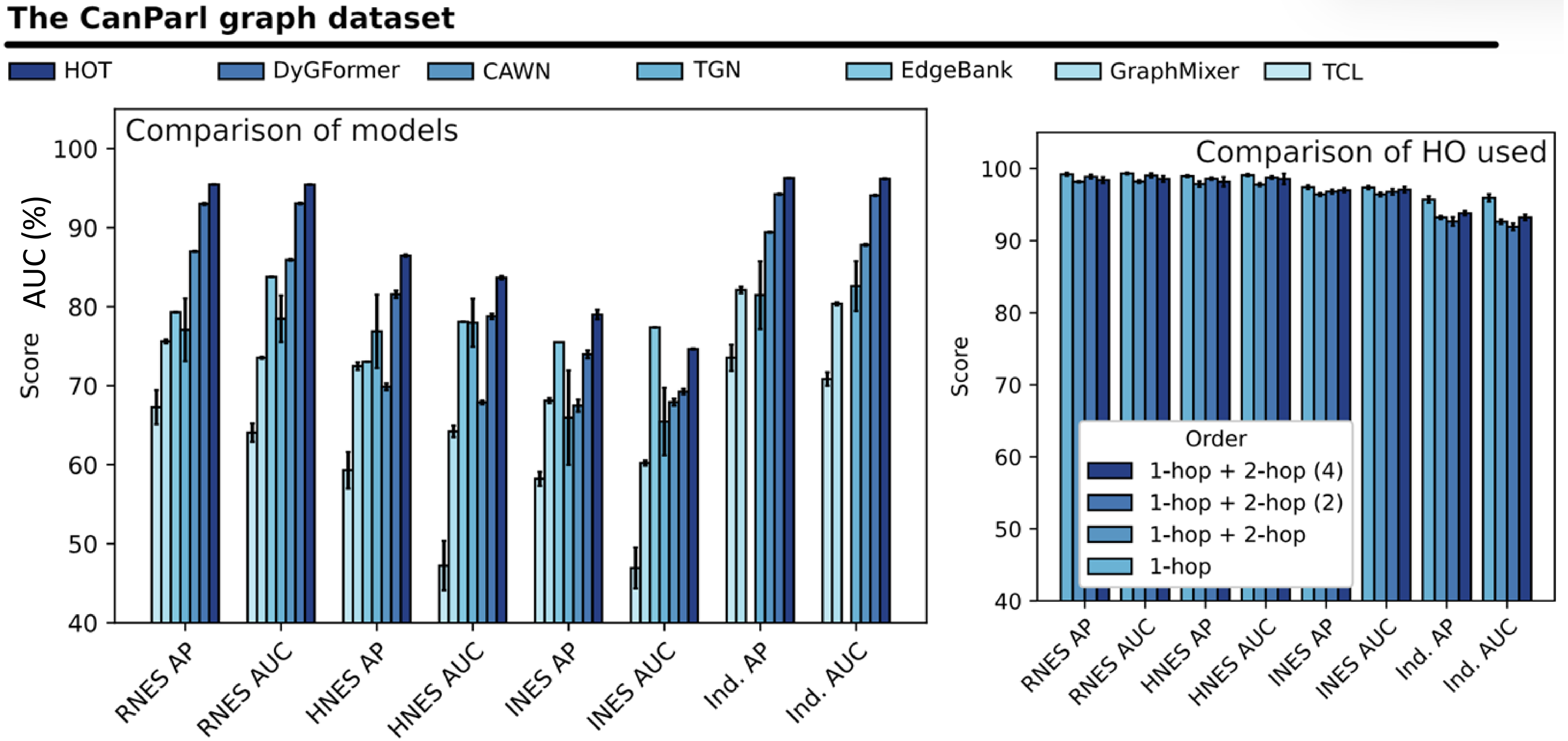
HOT Evaluation

HOT successfully leverages 2-hop interactions to make its predictions more accurate



HOT Evaluation

Here, the additional HO graph structural information does not seem to be adding significant amount of value to the model in this case.



Transductive

Inductive

HO-Enhanced Pipeline for Dynamic Link Prediction

Extract the relevant HO structures → Encode the structures → Conduct the prediction

HO-Enhanced Pipeline for Dynamic Link Prediction

@ LOG'23

HOT: Higher-Order Dynamic Graph Representation Learning with Efficient Transformers

Maciej Besta[1*] Afonso Claudino Catarino[1*] Lukas Gianinazzi[1] Nils Blach[1]
Piotr Nyczyk[2] Hubert Niewiadomski[2] Torsten Hoefler[1]

[1]Department of Computer Science, ETH Zurich; [2]Cledar

HO-Enhanced Pipeline for Dynamic Link Prediction

Extract the relevant HO structures → Encode the structures → Conduct the prediction