frontiers | Frontiers in Artificial Intelligence

TYPE Review
PUBLISHED 22 August 2023
DOI 10.3389/frai.2023.1124718

Check for updates

OPEN ACCESS

EDITED BY
Dursun Delen,
Oklahoma State University, United States

REVIEWED BY
Parisa Kordjamshidi,
Michigan State University, United States
Fabrizio Riguzzi,
University of Ferrara, Italy

*CORRESPONDENCE
Manfred Jaeger
✉ jaeger@cs.aau.dk

RECEIVED 15 December 2022
ACCEPTED 24 July 2023
PUBLISHED 22 August 2023

CITATION
Jaeger M (2023) Learning and reasoning with graph data. *Front. Artif. Intell.* 6:1124718.
doi: 10.3389/frai.2023.1124718

COPYRIGHT
© 2023 Jaeger. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Learning and reasoning with graph data

Manfred Jaeger*

Department of Computer Science, Aalborg University, Aalborg, Denmark

Reasoning about graphs, and learning from graph data is a field of artificial intelligence that has recently received much attention in the machine learning areas of graph representation learning and graph neural networks. Graphs are also the underlying structures of interest in a wide range of more traditional fields ranging from logic-oriented knowledge representation and reasoning to graph kernels and statistical relational learning. In this review we outline a broad map and inventory of the field of learning and reasoning with graphs that spans the spectrum from reasoning in the form of logical deduction to learning node embeddings. To obtain a unified perspective on such a diverse landscape we introduce a simple and general semantic concept of a model that covers logic knowledge bases, graph neural networks, kernel support vector machines, and many other types of frameworks. Still at a high semantic level, we survey common strategies for model specification using probabilistic factorization and standard feature construction techniques. Based on this semantic foundation we introduce a taxonomy of reasoning tasks that casts problems ranging from transductive link prediction to asymptotic analysis of random graph models as queries of different complexities for a given model. Similarly, we express learning in different frameworks and settings in terms of a common statistical maximum likelihood principle. Overall, this review aims to provide a coherent conceptual framework that provides a basis for further theoretical analyses of respective strengths and limitations of different approaches to handling graph data, and that facilitates combination and integration of different modeling paradigms.

KEYWORDS

graph data, representation learning, statistical relational learning, graph neural networks, neuro-symbolic integration, inductive logic programming

1. Introduction

Graphs are a very general mathematical abstraction for real world networks such as social-, sensor-, biological- or traffic-networks. These types of networks often generate large quantities of observational data, and using machine learning techniques to build predictive models for them is an area of substantial current interest. Graphs also arise as abstract models for knowledge, e.g., in the form of semantic models for a logic knowledge base, or directly as a knowledge graph. In most cases, an appropriate representation as a graph will require that one goes beyond the fundamental graph model, and allows that nodes are annotated with attributes, and that there are several distinct edge relation.

Evidently, in machine learning the main interest is in learning from graph data, whereas in knowledge representation and reasoning the primary focus is on deductive reasoning. However, both learning and reasoning play a role in all disciplines: making class label predictions from a learned model is a highly specialized (and limited) form of reasoning, and learning logic rules from examples has a long history in symbolic AI. It is the goal of this review to survey the large and diverse area of approaches for learning and reasoning with graphs in different areas of AI and adjacent fields of mathematics. Given the scope of the subject, our discussion will be mostly at a high, conceptual level. For more technical details, and more comprehensive literature reviews, we will point to relevant specialized surveys. The main objective of this review is to establish a coherent formal framework that facilitates a unified analysis of a wide variety of learning and reasoning frameworks. Even though this review aims to cover a broad range of methods and disciplines, there will be a certain focus on graph neural networks (GNNs) and statistical relational learning (SRL), whereas the fields of graph kernels and purely logic-based approaches receive a little less attention than they deserve.

The following examples illustrate the range of modeling, learning, and reasoning approaches that we aim to cover in this review. Each example describes a general task, a concrete instance of that task, and a particular approach for solving the task. The examples should not be construed in the way that the described solution approach is the only or even most suited one to deal with the given task. The intention is to illustrate the diversity of tasks and solution techniques.

Example 1.1. (Node classification with graph neural networks) One of the most common tasks in machine learning with graphs is *node classification*, i.e., predicting an unobserved node label. A standard instance of such a classification task is subject prediction in a bibliographic data graph: nodes are scientific papers that are connected by citation links, and the node labels consist of a subject area classification of the paper. In the *inductive* version of this task, one is given one or several training graphs containing labeled *training nodes*. The task is to learn a model that allows one to predict class labels of unlabeled nodes that are not already contained in the training graphs. For example, the training graph may consist of the current version of a bibliographic database, whereas the unlabeled nodes are new publications when they are added to the database. In the *transductive* version of the task, both the labeled training nodes and the unlabeled test nodes reside in the same graph, which is already fully known at the time of learning. This is the case when an incomplete subject area labeling is to be completed for a given bibliographic database. Graph neural networks are a state-of-the-art approach to solve such classification tasks (e.g., Niepert et al., 2016; Hamilton et al., 2017; Welling and Kipf, 2017; Veličković et al., 2018).

Example 1.2. (Link prediction via node embeddings) Here the task is to predict whether two nodes are connected by an edge. This prediction problem is usually considered in a transductive setting, where all the nodes and some of the edges are given, and edges between certain test pairs of nodes have to be predicted. Bibliographic data graphs are again a popular testbed for link prediction approaches (Kipf and Welling, 2016; Pan et al., 2021). *Recommender systems* also can be seen as handling a link prediction problem: the underlying graph here contains *user* and *product* nodes, and edges connect users with products that the user has bought (or provided some other type of positive feedback for). The link prediction problem then amounts to predicting positive user/product relationships that have not yet been observed. Numerous different link prediction approaches exist (Kumar et al., 2020 gives a comprehensive survey). A variety of different approaches is based on constructing for each node in the graph a d-dimensional real-valued *embedding vector*, and to score the likelihood of the existence of an edge between two nodes by considering the proximity (according to a suitable metric) of their embedding vectors. This general paradigm encompasses approaches such as matrix factorization (Koren and Bell, 2015) and random walk based approaches (Perozzi et al., 2014; Grover and Leskovec, 2016).

A good and concise monograph that covers the modern machine learning methods described in the preceding two examples is (Hamilton, 2020).

Example 1.3. (Graph Classification with inductive logic programming) One may also want to predict a class label associated with a whole graph. A classic example is predicting properties of molecules, where molecules are represented as graphs consisting of nodes representing the atoms, and links representing bonds between the atoms. The famous *Mutagenesis* dataset (Srinivasan et al., 1996), for example, consists of 188 molecules with a Boolean *mutagenic* class label. A predictor for this label may be given in the form of a logic program, such as the following:

$$\begin{array}{lll}
carbon\_path(A,B) & \leftarrow & carbon(A), carbon(B), bond(A,B) \\
carbon\_path(A,B) & \leftarrow & carbon(B), carbon\_path(A,C), bond(B,C) \\
carbon\_cycle & \leftarrow & carbon\_path(A,A) \\
mutagenic & \leftarrow & carbon\_cycle
\end{array}$$

(this program here is purely given for expository purposes, and does not resemble realistic classification programs for this task). A particular molecule specified by a list of *ground facts*, such as $carbon(at\_1), carbon(at\_2), nitrogen(at\_3), \ldots, bond(at\_1, at\_3)$, would then be classified as mutagenic, if *mutagenic* can be proven by the program from the ground facts. This will be the case if and only if the molecule contains a cycle consisting of carbon atoms. Parts of the program (e.g., the definitions of carbon path and cycle) may be provided by experts as background knowledge, whereas other parts (e.g., the dependence of mutagenic on the existence of a carbon cycle) would be learned from labeled examples.

Example 1.4. (Graph similarity and classification with graph kernels) The problem of graph classification (especially in bio-molecular application domains) has also extensively been approached with kernel techniques. Graph kernels (see Kriege et al., 2020 for an excellent survey) are functions k that map pairs of graphs G, H to a value $k(G,H) \in \mathbb{R}$ that is usually interpreted as a similarity measure for G and H, and which must be of the form $k(G,H) = \phi(G) \cdot \phi(H)$ for some finite or infinite dimensional real-valued feature vectors $\phi(G), \phi(H)$. The graph kernel can then be used for graph classification by using it as input for a support vector machine classifier. Based on its interpretation as a similarity measure, graph kernels can also support other types of similarity-based analyses, e.g., clustering. Most graph kernels are defined by an explicit definition of the mapping from graphs G to feature vectors $\phi(G)$. Important examples are the *Weisfeiler-Lehman kernel (WLK)* (Shervashidze et al., 2011), the *graphlet kernel* (Shervashidze et al., 2009), and the *random walk kernel* (Gärtner et al., 2003). The components of the feature vectors in the first two contain statistics on the occurrence of local neighborhood structures in G, whereas the features of the random walk kernel represent statistics on node label sequences that are generated by random walks on the graph.

The previous examples were concerned with specific prediction tasks. In the following examples we move toward more general forms of reasoning.

Example 1.5. (Probabilistic inference with SRL) The task is to learn a probabilistic graph model that supports a rich class of queries. An example is a probabilistic model for the genotypes of people in a *pedigree* graph. The model should support a spectrum of queries. A basic type are conditional probability queries of the form: given (partial) genotype information for some individuals, what are the probabilities for the genotypes of the other individuals? This is still very similar to the node classification task of Example 1.1. A query that goes beyond what has been described in previous examples is a *most probable explanation (MPE)* query: again, given partial genotype information, what is the most probable joint configuration of the genotypes for all individuals? *Statistical relational learning (SRL)* approaches such as *Relational Bayesian Networks (RBNs)* (Jaeger, 1997), *Markov logic networks (MLNs)* (Richardson and Domingos, 2006), or *ProbLog* (De Raedt et al., 2007) provide modeling and inference frameworks for solving such tasks.

Example 1.6. (Logical reasoning with first-order logic) Going beyond the flexible, but still rather structured type of queries considered in Example 3.6, we can consider more general logical reasoning tasks. There are no standard example instances of this, so we illustrate this task and its solution by deduction in first-order logic by the following example: given the following knowledge about a social network:

- Every user follows at least one other user. Expressed as a first-order logic formula, this reads as:

$$\forall x \exists y\, follows(x, y). \tag{1}$$

- There is a user who is not followed by anyone:

$$\exists y \neg \exists x\, follows(x, y) \tag{2}$$

Does this knowledge imply that there must be a user who has at least two followers, or there must be at least four different users:

$$(\exists y \exists^{\geq 2} x\, follows(x, y)) \vee \exists^{\geq 4} x? \tag{3}$$

Considering first all finite graphs, one finds that when (1) and (2) are true there must be a node with at least two incoming edges, i.e., the first part of the disjunction in (3) is true. When the graph is infinite, then this implication no longer holds, but then the second disjunct of (3) will be true (where the number 4 may be replaced with any natural number). Thus, we find that (1) and (2) logically entail (3). This inference can be performed by automated theorem provers. For example the SPASS prover (Weidenbach et al., 2009) can answer our query.

Example 1.7. (Limit behavior of random graphs) Many probabilistic models have been developed for the temporal evolution of growing networks. The simplest possible model is to assume that every new node is connected to the already existing nodes with a fixed probability p, and these connections are formed independently of each other. One may then ask how the probabilities of certain graph properties develop, as the size of the graph grows to infinity. Classic results of the Erdős-Rényi random graph theory establish, for example, that the limiting probability for the evolving graph to be connected is 1 (Erdős and Rényi, 1960) [the actual results of the Erdős-Rényi random graph theory are actually much more sophisticated, as they pertain to models where the edge probability is a function $p(n)$ of the number n of vertices]. Similarly, it is known that the probability of every property that can be expressed in first-order logic converges to either 0 or 1 (Fagin, 1976). Reasoning about such limiting probabilities goes beyond reasoning about all graphs as in Example 1.6, since we now consider probability distributions over infinite sequences of graphs. Some types of queries about the limit behavior of random graphs are formally decidable. This is the case, for example, for the limit probability of a first-order sentence (Grandjean, 1983). However, the computational complexity of these reasoning tasks puts them outside the range of practically feasible implementations.

Figure 1 arranges the landscape of reasoning scenarios we have considered in the preceding examples in two dimensions: one dimension characterizes the domain of graphs that we reason about: at the bottom of this dimension is the transductive setting of Examples 1.1 and 1.2 in which reasoning is limited to a single given graph. At the next level, labeled "one graph at a time," any single reasoning task is about a specific graph, but this graph under consideration can vary without the need to change or retrain the model. This corresponds to the inductive setting in Example 1.1, as well as the tasks described in Examples 1.4 and 3.6. At the third level, the reasoning concerns several or all graphs at once (Example 1.6). Finally, we may go beyond reasoning about properties of individual graphs, and consider global properties of the space of all graphs. This is exemplified by Example 1.7, where the reasoning pertains to the relationship between different probability distributions on graphs. These informal distinctions about the domain of reasoning will be partly formalized by technical definitions in Section 5.

The second dimension in Figure 1 is correlated with the first, but describes the type of reasoning that is performed. The most restricted type here is to perform a narrowly defined prediction task such as node classification or link prediction. Based on the distinction between "model checking" and "theorem proving" promoted by Halpern and Vardi (1991) we label the next level as "model checking": this refers to evaluating properties of a single given structure, where the class of properties that can be queried is defined by a rich and flexible query language. "Deduction" then refers to (logical) inference about a class of structures, and in "model theory" such a class of structure becomes the object of reasoning itself. Within this two-dimensional schema, Figure 1 indicates what areas of reasoning are covered by different types of frameworks. The reasoning landscape here delineated extends beyond the boundaries of what can currently be tackled with automated, algorithmic reasoning methods in AI, and extends to

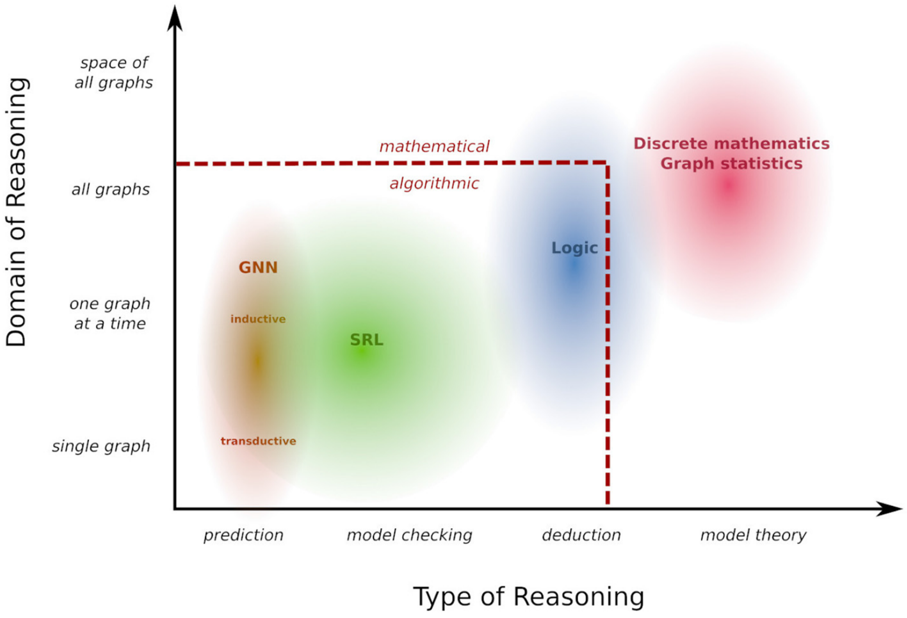

FIGURE 1
Reasoning landscape.

human mathematical inference. In the remainder of this review we will focus on algorithmic reasoning. However, it is an always open challenge to extend the scope of what can be accomplished by algorithmic means.

Figure 1 focuses on reasoning rather than learning. This is for two reasons: first, reasoning covers a somewhat wider ground that includes scenarios (e.g., logical deduction) not yet supported by learning methods. Second, a taxonomy of learning scenarios in terms of data requirements and learning objectives will naturally follow from the reasoning taxonomy (cf. Section 6). The link between reasoning and learning is a *model* that can be learned from data and that supports the reasoning tasks under consideration. In this review we use a general probabilistic concept of a model formally defined in in Section 3, that allows us to express a wide range of reasoning tasks as different forms of querying a model (Section 5). Similarly, a wide range of learning scenarios can be understood in a coherent manner as constructing models from data under a maximum likelihood objective (Section 6). This review is partly based on Jaeger (2022), which already contains an in-depth exploration of GNN and SRL methods for learning and reasoning with graphs. The current review takes a much broader high-level perspective. It contains fewer technical details on GNNs and SRL, and instead develops a general conceptual framework for describing a much wider range of learning and reasoning frameworks.

TABLE 1 Overview of notation.

$[n]$	The (node) set $\{1, \ldots, n\}$
$\mathcal{R}$	Signature of relation symbols
$\mathcal{R}_{in}, \mathcal{R}_{out}$	Sets of designated input and output relations
$e, r, \ldots$	(Lower case) specific symbols in $\mathcal{R}$
$i, j, \ldots$	Nodes
$\boldsymbol{i}, \boldsymbol{j}, \ldots$	Tuples of nodes
$E, R, \ldots$	(Upper case) interpretations of $e, r, \ldots$ in a specific graph
$\boldsymbol{R}$	Tuple of interpretations for all $r \in \mathcal{R}$
$\tilde{R}, \tilde{\boldsymbol{R}}$	Partial interpretation(s)
$\tilde{R} \preceq \tilde{R}'$	$\tilde{R}'$ Extends partial interpretation $\tilde{R}$
$Int_V(\mathcal{R})$	Set of interpretations of $\mathcal{R}$ over domain V
$\mathcal{G}(V, \mathcal{R})$	Set of graphs for signature $\mathcal{R}$ with domain V
$\mathcal{G}(< \infty, \mathcal{R})$	Set of all finite graphs for signature $\mathcal{R}$
$\mathcal{G}(\mathcal{R})$	Set of all graphs for signature $\mathcal{R}$
$\tilde{\mathcal{G}}(\ldots)$	Corresponding sets of partial graphs
$\Delta\mathcal{G}(V, \mathcal{R})$	Space of probability distributions on $\mathcal{G}(V, \mathcal{R})$
$Int_{(V, \tilde{\boldsymbol{R}})}(\mathcal{R})$	Set of completions of partial interpretation $\tilde{\boldsymbol{R}}$
$\{\!\{\ldots\}\!\}$	Delimiters for multisets

2. Graphs

In this section we establish the basic definitions and terminology we use for graphs. Table 1 collects the main notations. Our graphs will actually be multi-relational, attributed hypergraphs, but we just say shortly "graphs."

Different attributes and (hyper-) edge relations are collected in a *signature*: a set $\mathcal{R} = \{r_1, \ldots, r_m\}$ of *relation symbols*. Each relation symbol has an $arity(r_i) \in \{0, 1, \ldots\}$. Relations of arity 0 are global graph attributes, such as *toxic* for a graph representing a chemical molecule. Relations of arity 1 are node attributes. Relations of arities ≥ 3, can in principle be reduced to a set of binary relations by materializing tuples as nodes. For example, the 3-ary relation $shortest\_path(a, b, c)$ representing that node b lies on the shortest path from a to c can be encoded by creating a new shortest path node *sp*, and three binary relations *start*, *on*, *end*, so that $start(sp, a)$, $on(sp, b)$, $end(sp, c)$ are true. However, this leads to very un-natural encodings, and therefore we allow relations of arities ≥ 3.

A $\mathcal{R}$*-graph* is a structure $(V, \boldsymbol{R})$, where V is a finite or countably infinite set of nodes (also referred to as a *domain*), and $\boldsymbol{R} = (R_1, \ldots, R_m)$ are the *interpretations* of the relation symbols: $R_i : V^{arity(r_i)} \rightarrow \{0, 1\}$. In the case of $arity(r_i) = 0$ this is just a constant 0 or 1.

We write $Int_V(r)$ for the set of possible interpretations of the relation symbol $r \in \mathcal{R}$ over the domain V, and $Int_V(\mathcal{R}) = \times_{r \in \mathcal{R}} Int_V(r)$ for the set of all interpretations of the whole signature $\mathcal{R}$. In most cases it is sufficient to consider the domains $V = [n] := \{1, \ldots, n\}$ for $n \in \mathbb{N}$. We use $i, j, \ldots$ as generic symbols for nodes, and bold face $\boldsymbol{i}, \boldsymbol{j}, \ldots$ for tuples of nodes. We here use somewhat logic-inspired terminology and notation. Taking the logic conventions even further, we can equivalently define an interpretation of r as an assignment of true/false values to *ground atoms* $r(\boldsymbol{i})$ $[\boldsymbol{i} \in |V|^{arity(r)}]$. Going in the opposite direction toward the terminological conventions of neural networks, such an interpretation also can be seen as a $|V| \times \cdots \times |V|$-dimensional ($arity(r)$ many factors) tensor.

In the following, we usually take the signature $\mathcal{R}$ as given by the context, and do not refer to it explicitly, thus saying "graph" rather than $\mathcal{R}$*-graph*, and also abbreviating $Int_V(\mathcal{R})$ by Int_V Also, when the intended meaning is clear from the context, we use the simple term "relation" to either refer to a relation symbol r, or an interpretation R in a specific graph.

Note that according to our definitions all relations are *directed*. Undirected edges/relations are obtained as the special case where the interpretation R_i for a tuple $\boldsymbol{i}$ only depends on the elements of $\boldsymbol{i}$, not their order. Furthermore, only Boolean node attributes are permitted. Multi-valued attributes can be represented by multiple Boolean ones in a one-hot encoding.

For a given domain V and signature $\mathcal{R}$ we denote with $\mathcal{G}(V, \mathcal{R})$ the set of all $\mathcal{R}$-graphs with domain V, with $\mathcal{G}(< \infty, \mathcal{R})$ the set of all finite $\mathcal{R}$-graphs, and with $\mathcal{G}(\mathcal{R})$ the set of all graphs, also allowing (countably) infinite domains V. As a basis for probabilistic graph models, we denote with $\Delta\mathcal{G}(V, \mathcal{R})$ the set of probability distributions over $\mathcal{G}(V, \mathcal{R})$ (in the case of infinite V this must be based on suitable measure-theoretic definitions that we do not elaborate here).

3. Models

We introduce a general concept of a model, so that all reasoning tasks become different forms of querying a model. Ours will be a high-level, purely semantic notion of a model that imposes no restrictions on how models are represented, implemented or constructed. Our model definition is probabilistic in nature. As we shall see, this still allows us to capture purely qualitative, logic-based frameworks, although at the cost of casting them in a slightly contrived way into the probabilistic mold via extreme 0, 1-valued probabilities. Roughly speaking, a model in our sense will be a mapping from partly specified graphs to probability distributions over their possible completions.

A *partial graph* is a structure $(V, \tilde{\boldsymbol{R}})$ where V is a finite or countably infinite domain, and $\tilde{\boldsymbol{R}} = (\tilde{R}_1, \ldots, \tilde{R}_m)$ are *partial interpretations* of the relation symbols: $\tilde{R}_i : V^{arity(r_i)} \rightarrow \{0, 1, ?\}$. We write $\tilde{R}_i \preceq \tilde{R}'_i$ ($\tilde{R}'_i$ *extends* $\tilde{R}_i$) nif

$$\forall \boldsymbol{i} : \tilde{R}_i(\boldsymbol{i}) \neq ? \;\Rightarrow\; \tilde{R}'_i(\boldsymbol{i}) = \tilde{R}_i(\boldsymbol{i}).$$

If $\tilde{R}_i \preceq \tilde{R}'_i$ for $i = 1, \ldots, m$ we write $\tilde{\boldsymbol{R}} \preceq \tilde{\boldsymbol{R}}'$.

$\tilde{\mathcal{G}}(V, \mathcal{R})$ denotes the set of all partial $\mathcal{R}$-graphs with domain V, and $\tilde{\mathcal{G}}(< \infty, \mathcal{R})$ the set of all finite partial $\mathcal{R}$-graphs. Finally, $Int_{(V, \tilde{\boldsymbol{R}})}(\mathcal{R})$ denotes the set of complete interpretations $\boldsymbol{R} \in Int_V(\mathcal{R})$ with $\tilde{\boldsymbol{R}} \preceq \boldsymbol{R}$.

Definition 3.1. A $\mathcal{R}$-model $\mathcal{M} = (\mathcal{I}, \mu)$ consists of

- a subset $\mathcal{I} \subseteq \tilde{\mathcal{G}}(<\infty, \mathcal{R})$
- a mapping

$$\mu : (V, \tilde{\boldsymbol{R}}) \mapsto P_{(V, \tilde{\boldsymbol{R}})} \in \Delta\mathcal{G}(V, \mathcal{R})$$

defined for all $(V, \tilde{\boldsymbol{R}}) \in \mathcal{I}$, such that

$$P_{(V, \tilde{\boldsymbol{R}})}(Int_{(V, \tilde{\boldsymbol{R}})}(\mathcal{R})) = 1.$$

In the case where the input just consists of a domain V (i.e., $\tilde{\mathcal{R}}$ is completely unspecified), we simply write P_V for $P_{(V, \tilde{\boldsymbol{R}})}$.

Definition 3.1 has several important special cases: when a model is for classification of a specific label relation l, then $\mathcal{I} = \mathcal{G}(< \infty, \mathcal{R} \setminus \{l\})$. The model then defines for every input graph with complete specifications of relations $r \in \mathcal{R} \setminus \{l\}$ a distribution over interpretations L for l. We refer to such models as *discriminative*. A model that, in contrast, takes as input only a finite domain, i.e., $\mathcal{I} = \mathcal{G}(< \infty, \emptyset)$, and thereby maps a domain V to a probability distribution over $\mathcal{G}(V, \mathcal{R})$ is called *fully generative*. In between these two extremes are models where $\mathcal{R}$ is partitioned into a set of *input* relations $\mathcal{R}_{in}$ and output relations $\mathcal{R}_{out}$, and $\mathcal{I} = \mathcal{G}(< \infty, \mathcal{R}_{in})$. This is the typical case for SRL models. We refer to such models as *conditionally generative* (discriminative then is just a borderline case of conditionally generative). In all these special cases the input graph contains complete specifications of a selected set of relations. This covers many, but not all models used in practice: e.g., models for link prediction (cf. Example 3.3 below) operate on inputs with partial specifications of the edge relation. Transductive models are characterized as the special case $|\mathcal{I}| = 1$.

The abstract definition 3.1 accommodates a multitude of concrete approaches for learning and reasoning about graphs. We call a *modeling framework* any approach that provides computational tools for representing, learning, and reasoning with models. Figure 2 gives an overview of several classes of modeling frameworks, and representatives for each class. The selection of representatives does not attempt a complete coverage of even the most important examples. In the following we continue the examples of Section 1 to illustrate how these frameworks indeed fit into our general Definition 3.1.

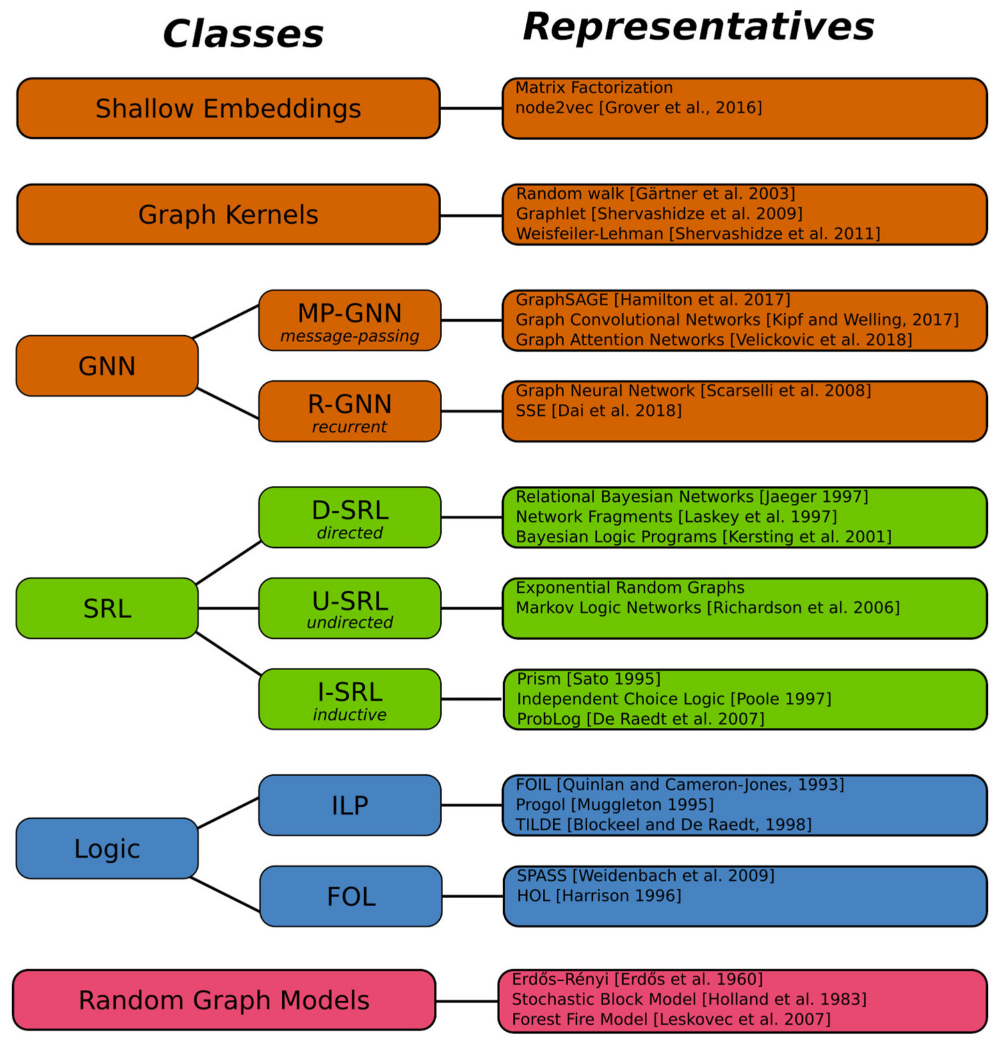
Classes
Representatives
Shallow Embeddings
Matrix Factorization
node2vec [Grover et al., 2016]
Graph Kernels
Random walk [Gärtner et al. 2003]
Graphlet [Shervashidze et al. 2009]
Weisfeiler-Lehman [Shervashidze et al. 2011]
GNN
MP-GNN
message-passing
GraphSAGE [Hamilton et al. 2017]
Graph Convolutional Networks [Kipf and Welling, 2017]
Graph Attention Networks [Velickovic et al. 2018]
R-GNN
recurrent
Graph Neural Network [Scarselli et al. 2008]
SSE [Dai et al. 2018]
SRL
D-SRL
directed
Relational Bayesian Networks [Jaeger 1997]
Network Fragments [Laskey et al. 1997]
Bayesian Logic Programs [Kersting et al. 2001]
U-SRL
undirected
Exponential Random Graphs
Markov Logic Networks [Richardson et al. 2006]
I-SRL
inductive
Prism [Sato 1995]
Independent Choice Logic [Poole 1997]
ProbLog [De Raedt et al. 2007]
Logic
ILP
FOIL [Quinlan and Cameron-Jones, 1993]
Progol [Muggleton 1995]
TILDE [Blockeel and De Raedt, 1998]
FOL
SPASS [Weidenbach et al. 2009]
HOL [Harrison 1996]
Random Graph Models
Erdős–Rényi [Erdős et al. 1960]
Stochastic Block Model [Holland et al. 1983]
Forest Fire Model [Leskovec et al. 2007]

FIGURE 2
Framework classes and representatives.

Example 3.2. (Node classification; GNNs) In the standard node classification scenario, the signature $\mathcal{R}$ consists of one binary edge relation e, observed node attributes $\boldsymbol{a} = a_1, \ldots, a_l$, and a node label l. A partial input graph has the form $(V, \tilde{\mathcal{R}})$, where in the partial interpretation $\tilde{\mathcal{R}} = (E, \boldsymbol{A}, \tilde{L})$ the edge relation and node attributes are fully observed, and the node label is unobserved for some (possibly all) nodes. Given a Graph neural networks then define label probabilities

$$P(l(i)|(V,(E,\boldsymbol{A}))) \qquad (i : \tilde{L}(i) = ?).$$

for the unlabeled nodes. Since these predictions are independent for all nodes, they define a distribution over completions L of $\tilde{L}$ via

$$P_{(V,\tilde{\mathcal{R}})}(L) = \Pi_{i:\tilde{L}(i)=?}P(l(i) = L(i)|(V,(E,\boldsymbol{A}))). \qquad (4)$$

In Figure 2, the class of GNN frameworks is divided into the sub-classes of *message-passing* and *recurrent* GNNs. MP-GNNs compute node feature vectors in a fixed number of iterations, whereas R-GNNs perform feature updates until a fixed point is reached. We return to this distinction in Section 4.2.

Example 3.3. (Link prediction; shallow embeddings) In its most basic form, a transductive link prediction problem is given by a single input graph $\mathcal{I} = \{(V, \tilde{E})\}$ with an partially observed edge relation. Usually, all edges that are not observed as existent are candidates for being predicted, so that $\tilde{E}(i,j) \in \{1, ?\}$ for all $i, j \in V$ (in practice, however, represented in the form $\tilde{E}(i,j) \in \{1, 0\}$, with the understanding that $\tilde{E}(i,j) = 0$ still allows the prediction $E(i,j) = 1$). *Shallow embeddings* construct for each node $i \in V$ an embedding vector $em(i)$, and define a scoring function $score[em(i), em(j)]$ for the likelihood of $E(i,j) = 1$. As the objective usually is to just rank candidates edges, this score need not necessarily be probabilistic [concrete examples are the dot product or cosine similarity of $em(i), em(j)$]. However, turning the scores into probabilities by applying a sigmoid function does not affect the ranking and hence we may assume that the link prediction model defines edge probabilities $P[e(i,j)|(V, \tilde{E})]$ $(i, j : \tilde{E}(i,j) \;\; = ?)$.

Via the same implicit independence assumptions as in the previous example, then a distribution in the sense of Definition 3.1 is defined as

$$P_{(V,\tilde{E})}(E) = \Pi_{i,j\,:\,\tilde{E}(i,j)=?} P(e(i,j) = E(i,j)|(V,\tilde{E})). \quad (5)$$

Example 3.4. (ILP) A logic program as shown in Example 1.3 can be interpreted as a model in our sense. For any input $(V, \tilde{\boldsymbol{R}})$, where $\tilde{\boldsymbol{R}}$ is specified by a list of true ground facts, the program defines the unique interpretation $\boldsymbol{R}^*$ in which a ground fact $r(\boldsymbol{i})$ is true, iff it is provable from the program and $\tilde{\boldsymbol{R}}$. This can be expressed in the form of a degenerate probability distribution with

$$P_{(V,\tilde{\boldsymbol{R}})}(\boldsymbol{R}^*) = 1.$$

The set $\mathcal{I}$ of possible inputs for this model consists of all $(V, \tilde{\boldsymbol{R}})$ where $\tilde{\boldsymbol{R}}$ is defined by positive ground facts only, i.e., $\tilde{R}(\boldsymbol{i}) \in \{1, ?\}$ for all $\tilde{R}, \boldsymbol{i}$.

Example 3.5. (Graph Classification; graph kernels) A graph kernel $k(G, H)$ alone is not a model in our sense. However, a support-vector machine classifier built on the kernel (denoted k-SVM) is such a model: a k-SVM maps graphs to label values in $\{0, 1\}$, which similar to Example 3.4 can be seen as a degenerate probabilistic model. Alternatively, since the k-SVM actually produces numeric scores for the labels (as the distance to the decision boundary), one can transform the score by a sigmoid function to non-degenerate probability values.

Example 3.6. (SRL) To solve the probabilistic inference tasks of Example an SRL model will be defined for input structures (V, E) with a fully observed edge relation e defining the pedigree structure. The model will then define a distribution $P_{(V,E)}(\boldsymbol{A})$ over interpretations $\boldsymbol{A}$ of node attributes $\boldsymbol{a}$. In a typical solution for this type of problem this will be done by defining a marginal distribution over genotypes for the individuals in V whose parents are not included in V, and a conditional distribution over child genotypes given parent genotypes. In order to solve the reasoning tasks described in Example , the framework must support queries about conditional probabilities of the form $P_{(V,E)}(a(i)|\tilde{\boldsymbol{A}})$ (genotype probabilities for individual i given partial information $\tilde{\boldsymbol{A}}$ about genotypes in the pedigree) and $arg\,max_{\boldsymbol{A}}\, P_{(V,E)}(\boldsymbol{A}|\tilde{\boldsymbol{A}})$ (MPE inference).

The model outlined here falls into the sub-class of *directed* SRL frameworks, where the distributions defined by a model can be represented in the form of directed probabilistic graphical models. Other important sub-classes of SRL distinguished in Figure 2 are frameworks based on *undirected* probabilistic graphical models, and probabilistic generalizations of inductive logic programming. Because of their close relationship with the most popular undirected SRL framework, Markov logic networks, Figure 2 also lists *exponential random graph models* under the U-SRL class, even though in terms of historical background and applications, exponential random graphs rather fall into the random graph model category.

Example 3.7. (First-order logic) A logical knowledge base KB as exemplified by (1) and (2) in Example 1.6 can be seen as a discriminative model in our sense: for any graph $G = (V, \boldsymbol{R}) \in \mathcal{G}(\mathcal{R})$ the semantics of the logic defines whether KB is true in G.[1] To formalize this as a discriminative model we augment the signature $\mathcal{R}$ with a binary graph label l_{KB}, set $\mathcal{I} = \mathcal{G}(\mathcal{R})$, and $P_G(l_{KB} = 1) = 1$ if KB is true in G, and $P_G(l_{KB} = 0) = 1$, otherwise. Logical inference as described in Example 1.6 then amounts to determining whether for all graphs G in which (3) is false one has $P_G(l_{KB} = 0) = 1$. This is a probabilistic rendition of the task of automated theorem proving.

Example 3.8. (Random graph models) Classical random graph models are generative models in our sense for the signature $\mathcal{R} = \{e\}$ containing a single edge relation.

The preceding examples show that the quite simple Definition 3.1 is sufficient as a unifying semantic foundation for a large variety of frameworks for reasoning about graphs. Of course, most of these frameworks also (or primarily) are designed for learning models from graph data, but our initial focus here is on modeling and reasoning, before we turn to learning in Section 6.

4. Modeling tools

Our Definition 3.1 of a model is completely abstract and does not entail any assumptions or prescriptions about the syntactic form and computational tools for model specification and reasoning. In the following, we consider several key modeling elements that are used across multiple concrete frameworks.

4.1. Factorization

We first consider conditionally generative models that for an input graph $(V, \boldsymbol{R}_{in})$ define a probability distribution $P_{(V,\boldsymbol{R}_{in})}$ (in the following abbreviated as P) over interpretations $\boldsymbol{R}_{out}$ of output relations. Enumerating $\mathcal{R}_{out}$ as $r_1, \ldots, r_n$, one can factorize P as

$$P(\boldsymbol{R}_{out}|\boldsymbol{R}_{in}) = P(R_1|\boldsymbol{R}_{in})P(R_2|R_1, \boldsymbol{R}_{in})P(R_3|R_1, R_2, \boldsymbol{R}_{in})\cdots P(R_n|R_1, \ldots, R_{n-1}, \boldsymbol{R}_{in}). \quad (6)$$

Thus, a generative model is decomposed into a product of discriminative models. This factorization at the relation level is often used in SRL models that are based on directed graphical models (Breese et al., 1994; Ngo and Haddawy, 1995; Jaeger, 1997; Laskey and Mahoney, 1997; Friedman et al., 1999; Kersting and De Raedt, 2001; Heckerman et al., 2007; Laskey, 2008).

Individual discriminative factors $P(R_k|R_1, \ldots, R_{k-1})$ can further be decomposed into a product of atom probabilities:

$$P(R_k|R_1, \ldots, R_{k-1}, \boldsymbol{R}_{in}) = \prod_{\boldsymbol{i}\in V^{arity(R_k)}} P(r_k(\boldsymbol{i}) = R_k(\boldsymbol{i})|R_1, \ldots, R_{k-1}, \boldsymbol{R}_{in}) \quad (7)$$

[1] In logic terminology, these graphs would be called the models of *KB*. This is in direct conflict with our terminology, where rather the knowledge base is considered a model; therefore we avoid to use the term "model" in the sense of logic.

We have seen that this factorization is implicitly present in GNNs (4) and shallow embeddings (5). Unlike (6), which is a generally valid application of the chain rule, the factorization (7) represents a quite restrictive assumption that the R_k-atoms are conditionally independent given relations $R_1, \ldots, R_{k-1}$. This leads to some challenges, e.g., for modeling *homophily* properties of R_k. A useful strategy to address such limitations of (7) is to include among the $R_1, \ldots, R_{k-1}$ *latent relations* that only serve to induce dependencies among the R_k-atoms.

Frameworks that are based on undirected probabilistic models, notably *exponential random graph models* and the closely related *Markov logic networks (MLNs)* (Richardson and Domingos, 2006), decompose the joint distribution $P(\boldsymbol{R}_{out})$ into factors that are defined by *features* $F(\boldsymbol{i})$ of node tuples $\boldsymbol{i}$. Examples of such features are the degree of a node $F(i) = |\{j|edge(i,j)\}|$, or, in MLNs, 0,1-valued featured expressed by Boolean formulas over ground atoms, e.g., $F(i,j) = edge(i,j) \wedge r(i) \wedge \neg r(j)$. Every such feature has an arity, and the distribution defined by features $F_1, \ldots, F_K$ then is

$$P(\boldsymbol{R}_{out}) = \frac{1}{Z}\exp\left(\sum_{k=1}^{K}\sum_{\boldsymbol{i}\in V^{arity(F)}} F_k(\boldsymbol{i})(\boldsymbol{R}_{out}, \boldsymbol{R}_{in})\right), \quad (8)$$

where Z is a normalizing constant, and on the right-hand side we make explicit that the feature is a function of the interpretations $\boldsymbol{R}_{out}$ and $\boldsymbol{R}_{in}$.

4.2. Feature construction

Both the basic factors $P(R_k(\boldsymbol{i})|R_1, \ldots, R_{k-1}, \boldsymbol{R}_{in})$ in (7) and $F(\boldsymbol{i})(\boldsymbol{R}_{out}, \boldsymbol{R}_{in})$ in (8) are functions that take as input a graph $(V, \boldsymbol{R}')$ containing interpretations $\boldsymbol{R}'$ for some subset $\mathcal{R}' \subseteq \mathcal{R} = \mathcal{R}_{in} \cup \mathcal{R}_{out}$, and return a mapping of entity tuples $\boldsymbol{i}$ into $\mathbb{R}$. Such entity feature functions also lie at the core of many other modeling frameworks than those that use these features inside a probabilistic factorization approach: graph neural networks define a sequence of node feature vectors (a.k.a. embedding or representation vectors). Each component in such a vector is a feature function in our sense. A FOL formula with k free variables defines a 0,1-valued feature of arity k.

Graph kernels, by definition, are based on feature functions defined on whole graphs, i.e., features of arity zero in our sense. These features, however, often are aggregates of features defined at the single node level (e.g., in the Weisfeiler-Lehman kernel), or k-tuple level (e.g., in the graphlet kernel, whose features are closely related to the MLN features).

In many cases, feature functions are nested constructs where complex features are built from simpler ones. An important consideration for feature construction is whether they are used in models for transductive or inductive reasoning tasks. In the latter case the required generalization capabilities of the model imply that all features should be *invariant under isomorphisms*, i.e., $F(\tilde{\boldsymbol{i}})(\tilde{V}, \tilde{\boldsymbol{R}}') = F(\boldsymbol{i})(V, \boldsymbol{R}')$ whenever there is a graph isomorphism from $(V, \boldsymbol{R}')$ to $(\tilde{V}, \tilde{\boldsymbol{R}}')$ that maps $\boldsymbol{i}$ to $\tilde{\boldsymbol{i}}$.

Table 2 summarizes some characteristics of the feature functions used in different frameworks. The column "Arity" indicates whether the framework uses features of graphs (0), nodes (1), or tuples of any arities $\geq$ 0. The remaining columns are addressed in the following sub-sections.

4.2.1. Initial features

All feature constructions start with some initial base features (we note that when here and in the following we talk about "construction," then this is not meant to imply manual construction; it may very well be automated construction by a learner). Initial features often are node features. Here already important distinctions arise that essentially determine whether the model will have a transductive or inductive use. If one uses *unique node identifiers* as initial features (denoted "id" in the "Initial" column of Table 2), then models constructed from these features are not invariant under isomorphisms, and will be limited to transductive reasoning. As an example consider the feature $F(i)$: "i is at most three edges away from node '26'." This feature is illustrated by the red coloring of nodes in Figure 3A. While this feature can be useful for one specific graph (e.g., for predicting a node label), it does not generalize in a meaningful way to other graphs, even if they also contain a node with identifier "26." Node identifiers are mostly used for transductive reasoning with GNNs. They are usually not used in SRL or kernel frameworks.

The most commonly used initial features are *node attributes* ("at" in Table 2). For example, if nodes have a color attribute with values "blue" and "yellow," then with this initial features one can define a feature $F(i)$: "i is at most two edges away from a blue node." This feature, illustrated by a red coloring in Figure 3B, also applies to other graphs sharing the same signature $\mathcal{R}$ containing the blue/yellow attribute (we note that the regular grid structures of the graphs (a) and (b) are for illustrative clarity only; these features are in no way linked to such regular structures).

In some cases, there are no informative initial features used, or available. Formally, this can be expressed by a single *vacuous* node attribute that has a constant value for all nodes. Such a vacuous initial feature can still be the basis for the construction of useful complex features using the construction methods described below. Figure 3C illustrates this for a constructed feature $F(i)$: "i is at most two edges away from a node with degree $\geq$ 5."

Similar to node attributes, also *non unary relations* of $\mathcal{R}'$ can serve as initial features ("rel" in Table 2). Most logic-based or SRL frameworks will allow binary initial features, enabling, e.g., the construction of features like $F(i,j,k) = edge(i,j) \wedge edge(i,k) \wedge edge(j,k)$ expressing that i,j,k form a triangle. Such features are outside the reach of most GNN frameworks, though higher-order GNNs (Morris et al., 2019) overcome this limitation.

A special approach that has been proposed in connection with GNNs is the use of *random node attributes* (Abboud et al., 2021; Sato et al., 2021) as initial features. Such random attributes can serve as a substitute for unique node identifiers. Due to their random nature, however, it does not make sense to construct features based on their absolute value, like the one illustrated in Figure 3A. However, they enable the construction of features based on equalities $rid(i) = rid(j)$ (rid being the random attribute), which with high probability just encodes identity $i = j$, and which is robust with regard to the random actual values. One can then construct features like $F(i)$: "i lies on a cycle of length 5" as "j is

TABLE 2 Properties of feature functions underlying different frameworks.

Framework	Arity	Initial	Aggregation	Final
Shallow embedding	0,1	n/a	n/a	Shallow
Kernel	≥ 0	at	*div*	Deep, shallow
MP-GNN	0,1	id,at	Sum, mean, max,...	Deep
R-GNN	0,1	id,at	Sum, mean, max,...	Sat
D-SRL	≥ 0	at,rel	Noisy-or, mean,...	Deep
U-SRL	≥ 0	at,rel	n/a	Shallow
I-SRL	≥ 0	at,rel	$\exists$	Sat
ILP	≥ 0	at,rel	$\exists$	Sat
FOL	≥ 0	at,rel	$\exists, \forall$	Deep
Random graph models	*div*	*div*	*div*	*div*

div: the framework class is too loosely defined to allow a meaningful entry.

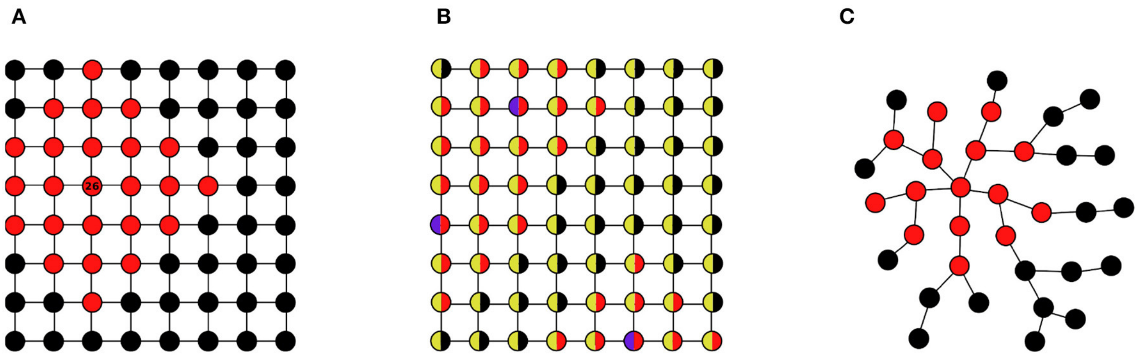

FIGURE 3
Initial node features and their use: **(A)** node identifiers; **(B)** node attributes; **(C)** vacuous.

reachable from i in 5 steps, and $rid(i) = rid(j)$." A similar capability for constructing equality-based features is presented by Vignac et al. (2020). Here the initial features are in fact unique identifiers, but the subsequent constructions are limited in such a way that again absolute values are not used in an informative manner, and invariance under isomorphisms is ensured.

4.2.2. Construction by aggregation

Complex features are constructed out of simpler ones using simple operations such as Boolean or arithmetic operators, and linear or non-linear transformations. The essential tool for feature construction in graphs, however, is the aggregation of feature values from related entities. The general structure of such a construction can be described as

$$F_{new}(\boldsymbol{i}) := agg\{\!\!\{F_{old}(\boldsymbol{i},\boldsymbol{j}) | \boldsymbol{j} : \phi(\boldsymbol{i},\boldsymbol{j})\}\!\!\}, \qquad (9)$$

where $F_{old}(\boldsymbol{i},\boldsymbol{j})$ is an already constructed feature, the delimiters $\}\!\!\}, \{\!\!\{$ are used to denote a multiset, $\phi(\boldsymbol{i},\boldsymbol{j})$ expresses a relationship between $\boldsymbol{i}$ and $\boldsymbol{j}$, and *agg* is an aggregation function that maps the multiset of values $F_{old}(\boldsymbol{i},\boldsymbol{j})$ to a single number. For better readability we here omit the common dependence of F_{old} and F_{new} on the input graph $(V, \boldsymbol{R}')$. As usual, vector-valued features are also covered by considering each vector component as a scalar feature. In the case of GNNs, Equation (9) takes the special form $F_{new}(i) := agg\{\!\!\{F_{old}(j) | j : edge(i,j)\}\!\!\}$, and is often referred to as a *message passing* operation. Another special form of aggregation used in GNNs is the *readout* operation that aggregates node features into a single graph level feature: $F_{readout}() := agg\{\!\!\{F_{node}(j) | j : true(j)\}\!\!\}$, where $true(j)$ stands for a tautological condition that holds for all nodes j. D-SRL and I-SRL frameworks are particularly flexible with regard to aggregate feature construction by supporting rich classes of relationships ϕ. U-SRL frameworks, on the other hand, do not support the nested construction of features by (9) in general, and are limited to a single sum-aggregation of basic features F_k via (8).

Applicable aggregation functions *agg* depend on the type of the feature values. When in a logic-based framework all features are Boolean, then *agg* usually is existential or universal quantification. When in a probabilistic framework all feature values are (conditional) probabilities, then *noisy-or* as the probabilistic counterpart of existential quantification is a common aggregator. When features have unconstrained numeric values, then standard aggregators are *sum*, *mean*, *min*, or *max*. Table 2 lists under "Aggregation" characteristic forms of aggregation functions in different frameworks (n/a for frameworks that do not support nested aggregations).

In (9), we have written the aggregation function as operating on a multiset. In this form one immediately obtains that if F_{old} is invariant under isomorphisms, then so is F_{new}. However, in practice the multiset $\{\!|F_{old}(\boldsymbol{i},\boldsymbol{j})|\ldots)|\!\}$ will be stored as a vector. When *agg* then is defined for vector inputs, one has to require that *agg* is *permutation invariant* (which just means that only the multiset content of the vector affects the computed value) in order to ensure invariance under isomorphisms.

4.2.3. Final features

The inductive feature construction described in the preceding subsections leads to features that have a certain *depth* of nestings of aggregation functions. Often this depth corresponds to a maximal radius of the graph neighborhood of $\boldsymbol{i}$ that can affect the value $F(\boldsymbol{i})$. This example is the case for features constructed by message passing aggregation in GNNs (but not for *readout* features), and the features of the WLK. Many modeling frameworks only are based on features of a limited depth that are obtained by a fixed sequence of feature constructions. This case is denoted by "deep" in the "Final" column of Table 2, whereas "shallow" stands for feature constructions that do not support nested aggregation.

In contrast, I-SRL and R-GNN models are based on an a-priori unbounded sequence of feature constructions that for each input graph $(V, \boldsymbol{R}')$ proceeds until a saturation point is reached ("sat" in Table 2). This enables models which are based on final features that are outside the reach of any fixed depth construction. An example is the "contains cycle" graph feature of Example 1.3.

4.2.4. Numeric and symbolic representations

In the previous sections we have considered features mostly at an abstract semantic level. For any given semantic feature there can be very different forms of formal representation. We here illustrate different paradigms on a concrete example. Assume that we have a signature containing a single binary *edge* relation, l different Boolean node attributes $a_1, \ldots, a_l$, and a node class attribute *class*. A node classification model may depend on the node feature $F(i)$ defined as the number of distinct paths of length 2 that lead from i to a node j for which $a_1(j)$ is true. The concrete classification model then can be defined as a logistic regression model based on F:

$$P_{(V,\boldsymbol{R})}(class(i) = 1) = \sigma(aF(i) + b), \tag{10}$$

where $a, b \in \mathbb{R}$, and σ denotes the sigmoid function.

In a MP-GNN framework, the construction of F and the classification model can be implemented in a two layer network with a generic structure such as

$$\begin{aligned}
\boldsymbol{h}^{(1)}(i) &= relu(sum\{\!|U^{(1)}\boldsymbol{a}(h)|h : edge(i,h)|\!\} \oplus V^{(1)}\boldsymbol{a}(i)) \\
\boldsymbol{h}^{(2)}(i) &= relu(sum\{\!|U^{(2)}\boldsymbol{h}^{(1)}(h)|h : edge(i,h)|\!\} \oplus V^{(2)}\boldsymbol{h}^{(1)}(i)) \\
\boldsymbol{out}(i) &= softmax(W\boldsymbol{h}^{(2)}(i) + \boldsymbol{b})
\end{aligned} \tag{11}$$

where $\boldsymbol{h}^{(k)}$ are the embedding vectors computed at hidden layer k, $\boldsymbol{a}(i)$ is the attribute vector of i, the $U^{(k)}$, $V^{(k)}$ and W are weight matrices, $\boldsymbol{b}$ is a bias vector, $\oplus$ denotes vector concatenation, and *relu* is component-wise application of the *relu* activation function.

Finally, $\boldsymbol{out}$ is a two-dimensional output vector whose components represent the probabilities for $class(i) = 1$ and $class(i) = 0$. The embedding vector $\boldsymbol{h}^{(2)}(i)$ defines a whole set of (scalar) features. The feature $F(i)$ can be obtained as the first component of $\boldsymbol{h}^{(2)}(i)$ when in both matrices $U^{(1)}$ and $U^{(2)}$ the first row is set to $(1, 0, \ldots, 0)$. With a suitable setting of W and $\boldsymbol{b}$, $\boldsymbol{out}(i)$ can then represent the model (10). Clearly, the representational capacity of (11) is not nearly exhausted when used in this manner to implement (10). The architecture (11) would usually encode a model where the output probabilities are a complex function of a multitude of different features encoded in the hidden embedding vectors.

A symbolic representation in a D-SRL framework would take a form like

$$\begin{aligned}
\#a_1\_neighbors(i) &\leftarrow sum\{\!|a_1(h)|h : edge(i,h)|\!\} \\
F(i) &\leftarrow sum\{\!|\#a_1\_neighbors(h)|h : edge(i,h)|\!\} \\
class(i) &\leftarrow \sigma(aF(i) + b)
\end{aligned} \tag{12}$$

The first two lines define the feature F, while the last implements the classification rule. In contrast to (11), where F is defined numerically through the entries in the parameter matrices, it is here defined symbolically in a formal language that combines elements of logic and functional programming languages. The only numeric parameters in the specification are the coefficients a, b of the logistic regression model. The representation (12) is more interpretable than (11). However, a new classification model depending on other features than F would here require a whole new specification, whereas in (11) this is accomplished simply by a different parameter setting. Also, the symbolic representation will grow in size (and loose in interpretability), when more complex models depending on a large number of features are needed.

Though on different sides of the symbolic/numeric divide, model specifications (11) and (12) are very similar in nature in that they use the same three-step strategy to first define the number of direct a_1-neighbors as an auxiliary node feature, then define F, and finally the logistic regression model based on F. This iterative construction is not supported in U-SRL frameworks. However, Equation (10) can still be implemented in an U-SRL framework using a specification of the form

$$\begin{array}{ll}
class(i) \wedge edge(i,j) \wedge edge(j,h) \wedge a_1(h) & w_1 \\
\neg class(i) \wedge edge(i,j) \wedge edge(j,h) \wedge a_1(h) & w_2 \\
class(i) & w_3
\end{array} \tag{13}$$

This is an MLN type specification that defines a model of the form (8) with $K = 3$ features defined by Boolean properties and associated weights. The feature $F_k(\boldsymbol{i})$ evaluates to w_k if the Boolean property is satisfied by $\boldsymbol{i}$, and otherwise evaluates to zero [$\boldsymbol{i} = (i,j,h)$ for the first two features, and $\boldsymbol{i} = (i)$ for the third]. This defines a fully generative model without a separation into feature construction and classification model. However, with suitable setting of the weights w_k, the conditional distributions of the $class(i)$ atoms given full instantiations of the *edge* relation and the node attributes, the model (10) can be obtained.

4.2.5. Expressivity

The general question of expressivity of feature construction frameworks can be viewed from an *absolute* or *comparative*

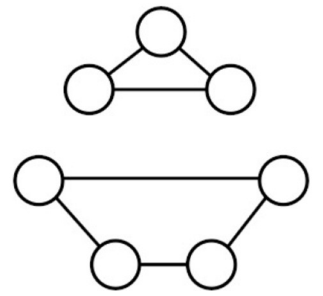

FIGURE 4
Indistinguishable nodes.

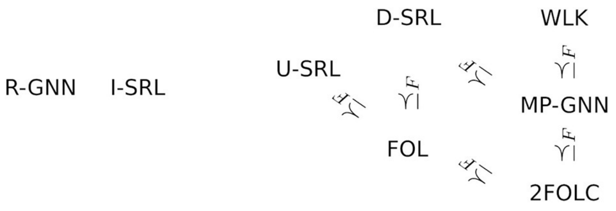

FIGURE 5
Some expressivity relations.

perspective. From the absolute point of view, one asks whether a feature construction framework is generally able to distinguish different entities, i.e., whether for $\boldsymbol{i} \in (V, \boldsymbol{R}')$ and $\tilde{\boldsymbol{i}} \in (\tilde{V}, \tilde{\boldsymbol{R}}')$ a feature F can be constructed with $F(\tilde{\boldsymbol{i}})(\tilde{V}, \tilde{\boldsymbol{R}}') \neq F(\boldsymbol{i})(V, \boldsymbol{R}')$ (usually only required or desired when $\boldsymbol{i}$ and $\tilde{\boldsymbol{i}}$ are not isomorphic). In the context of graph kernels this question was first investigated by Gärtner et al. (2003), who showed that graph kernels that are maximally expressive in this sense will be computationally intractable due to their implicit ability to solve the subgraph isomorphism problem.

Figure 4 (adapted from Abboud et al., 2021) gives an example of a graph whose nodes do not have any attributes. The nodes on the three-node cycle are not isomorphic to the nodes on the 4-node cycle. However, features that are constructed starting with the vacuous initial feature using the aggregation mechanisms of WLK or GNNs will not be able to distinguish these nodes. As already mentioned in Section 4.2.1, GNNs with random node attributes as initial features, on the other hand, will be able to distinguish the nodes on the three-cycle from the nodes on the four-cycle. Without the need for random node attributes this distinction also is enabled by most SRL frameworks due to their support for binary feature constructions, which here can be used to first construct Boolean features $k\text{-}path(i,j)$ representing whether there exists a path of length k from i to j, and then distinguish the nodes on the three-cycle by the feature $3\text{-}cycle(i) := 3\text{-}path(i,i)$.

The discriminative capabilities of feature functions largely depend on the discriminative capabilities of the aggregators (9). It has been suggested that in combination with suitable feature transformation functions applied to F_{old} before aggregation, and F_{new} after aggregation, *sum* is a universally expressive aggregation function (Zaheer et al., 2017). As pointed out by Wagstaff et al. (2019), however, the requirements on the transformation functions are not realistic for actual implementations. See also Jaeger (2022) for a detailed discussion.

In the comparative view, one asks whether all features that can be constructed in a framework ABC can also be constructed in a framework XYZ. If that is the case, we write ABC$\preceq_F$ XYZ. We use the subscript F here in order to emphasize that this is an expressivity relationship about the feature construction capabilities of the frameworks. Different frameworks use features in a different ways, and therefore ABC$\preceq_F$XYZ does not directly imply that every model of ABC also can be represented as a model in XYZ. However, feature expressivity is the most fundamental ingredient for modeling capacity.

Figure 5 gives a small and simplistic overview of some expressivity relationships between different frameworks. In this overview we gloss over many technical details regarding the exact representatives of larger classes such as MP-GNN for which the relations have been proven, and the fact that in some cases the comparison so far only has been conducted for graphs with a single edge relation (though generalizations to multi-relational graphs seem mostly straightforward). The relationship FOL$\preceq_F$D-SRL has been shown in Jaeger (1997). That MP-GNNs are at least as expressive as the 2-variable fragment of first-order logic with counting quantifiers (2FOLC) is shown by Barceló et al. (2020) on the basis of MP-GNNs that allow feature constructions using both message-passing and readout aggregations. The relationship between MP-GNNs and the Weisfeiler-Lehman graph-isomorphism test was demonstrated by Morris et al. (2019) and Xu et al. (2019). A detailed account of expressivity of different GNN frameworks and their relationship to Weisfeiler-Lehman tests is given by Sato (2020). The MP-GNN$\preceq_F$D-SRL relation is demonstrated in Jaeger (2022), and FOL$\preceq_F$U-SRL is shown by Richardson and Domingos (2006). It must be emphasized, however, that these results only pertain to the feature expressivity of the frameworks. The reasoning capabilities of logical deduction in FOL are not provided by the SRL frameworks.

The figure also shows that R-GNN and I-SRL are (presumably) incomparable to the other frameworks here shown: their saturation based feature construction, on the one hand, provides expressivity not available to the fixed depth features of the other frameworks. On the other hand, I-SRL has limitations regarding expressing logical features involving negation or universal quantification, and R-GNN cannot aggregate features over a fixed number of steps using different aggregators at each step.

5. Reasoning: tasks and techniques

Given a model in the sense of Definition 3.1, we consider several classes of reasoning tasks. Narrowing down the range of possible reasoning tasks considered in Section 1, we are now only concerned with algorithmic reasoning.

5.1. Sampling

The sampling task consists of generating a random graph $(V, \boldsymbol{R}) \in \mathcal{G}(V, \mathcal{R})$ according to the distribution $P_{(V,\tilde{\boldsymbol{R}})}$. Here we consider this sampling problem as a task in itself, which is to be distinguished from sampling as a method for approximately solving other tasks (see below). Sampling as a main reasoning task is mostly considered in the context of graph evolution models, where the comparison of observed statistics in the random sample with real-world graphs is used to validate the evolution model (e.g., Leskovec et al., 2007). A concrete application of sampling from generative models is to create realistic benchmark datasets for other computational tasks on graphs. Bonifati et al. (2020) give a comprehensive overview of graph generator models in this application context.

Sampling is typically a task for fully generative models. However, not all models that are fully generative in the semantic sense of Section 3 necessarily support efficient sampling procedures. This is the case, in particular, for U-SRL models whose specification (8) does not translate into an operational sampling procedure. D-SRL models following the factorization strategy of (6) and (7), on the other hand, allow for an efficient sampling procedure by successively sampling truth values for ground atoms $R_k(\boldsymbol{i})$. The same is true for dedicated graph evolution models, and, with slight modifications, for I-SRL models.

While mostly designed for prediction tasks, MP-GNNs have also been adapted as models for random graph generation (e.g., Li et al., 2018; Simonovsky and Komodakis, 2018; You et al., 2018; Dai et al., 2020). Differently from more traditional graph evolution models, which typically only have a small number of calibration parameters, generative GNNs are highly parameterized and can be fitted to training data sets of graphs with different structural properties.

5.2. Domain-specific queries

A large class of reasoning tasks falls into the following category of performing inference based on a given model for a single graph under consideration. This covers what in the introduction informally was referred to as reasoning about a single graph, and reasoning about one graph at a time.

Definition 5.1. Let $\mathcal{M} = (\mathcal{I}, \mu)$ be a model. A *domain specific query* is given by

(a.) an input graph $(V, \tilde{\boldsymbol{R}}) \in \mathcal{I}$,
(b.) a *query set* $\rho \subseteq Int_{(V,\tilde{\boldsymbol{R}})}(\mathcal{R})$,
(c.) a *query objective* that consists of calculating a property of the set $\{P_{(V,\tilde{\boldsymbol{R}})}(\boldsymbol{R}) | \boldsymbol{R} \in \rho\}$.

Query sets are typically specified by one or several query atoms $r_{k_1}(\boldsymbol{i}_1), \ldots, r_{k_q}(\boldsymbol{i}_q)$, where ρ then consists of all interpretations $\boldsymbol{R}$ in which the query atoms are true. A slightly more general class of queries is obtained when the query atoms can also be negated, with the corresponding condition that the negated atoms are false in $\boldsymbol{R}$. Part (c.) of Definition 5.1 is formulated quite loosely. The most common query objective is to calculate the probability of ρ:

$$P_{(V,\tilde{\boldsymbol{R}})}(\rho) = \sum_{\boldsymbol{R} \in \rho} P_{(V,\tilde{\boldsymbol{R}})}(\boldsymbol{R}). \tag{14}$$

When ρ is defined by a list of query atoms we can write for $P_{(V,\tilde{\boldsymbol{R}})}(\rho)$ also more intuitively

$$P_{(V,\tilde{\boldsymbol{R}})}(r_{k_1}(\boldsymbol{i}_1), \ldots, r_{k_q}(\boldsymbol{i}_q)). \tag{15}$$

The prediction tasks of Examples 1.1–1.4 all fall into the category of computing (15) for a single query atom. More general *probabilistic inference* is concerned with computing (15) for general lists of (possibly negated) query atoms. Furthermore, we often are interested in conditional queries of the form

$$P_{(V,\tilde{\boldsymbol{R}})}(r_{k_1}(\boldsymbol{i}_1), \ldots, r_{k_q}(\boldsymbol{i}_q) | r_{k_{q+1}}(\boldsymbol{i}_{q+1}), \ldots, r_{k_p}(\boldsymbol{i}_p)) \tag{16}$$

where $r_{k_{q+1}}(\boldsymbol{i}_{q+1}), \ldots, r_{k_p}(\boldsymbol{i}_p)$ represents *observed evidence*, and $r_{k_1}(\boldsymbol{i}_1), \ldots, r_{k_q}(\boldsymbol{i}_q)$ represents the uncertain *target* of our inference. If the inference framework permits arbitrary unconditional queries (15), then (16) can be computed from the definition of conditional probabilities as the ratio of two unconditional queries. Thus, Equation (15) already covers most cases of probabilistic inference.

In the most probable genotype problem of Example 3.6 the query set is given by a list of atoms (possibly negated) representing observed genotype data, and the query objective is to compute

$$\mathop{arg\,max}_{\boldsymbol{R} \in \rho} P_{(V,\tilde{\boldsymbol{R}})}(\boldsymbol{R}). \tag{17}$$

The computational tools and computational complexity for computing a query differ widely according to the underlying modeling framework, and the class of queries it supports. The simplest case is given by specialized prediction models that only support single (class label) atoms $class(\boldsymbol{i})$ as queries. When, moreover, the model (as in GNNs) is directly defined by functional expressions for the conditional probabilities $P(class(\boldsymbol{i}) | \boldsymbol{R}_{in})$, then the computation consists of a single *forward computation* that is typically linear in the size of the input graph $(V, \tilde{\boldsymbol{R}}) = (V, \boldsymbol{R}_{in})$ and the model specification. Similarly, performing prediction by a kernel-SVM is usually computationally tractable, though the necessary evaluations of the kernel function here add another source of complexity.

Dedicated classification models provide for efficient predictive inference, because for their limited class of supported queries (14) can be evaluated without explicitly performing the summation over $\boldsymbol{R} \in \rho$. The situation is very different for SRL models that are designed to support a much richer class of queries defined by arbitrary lists of atoms in the $\mathcal{R}_{out}$ relations, and where the summation in (14) cannot be bypassed. A naive execution of the summation over all $\boldsymbol{R}$ is usually infeasible, since the number of $\boldsymbol{R}$ one needs to sum over is exponential in the number of unobserved atoms, i.e., atoms that are not instantiated in the input $\tilde{\boldsymbol{R}}$, or included among the query atoms. The number of such unobserved atoms, in turn, is typically polynomial in $|V|$. For SRL models, the basic strategy to evaluate (14) is to compile the distribution $P_{(V,\tilde{\boldsymbol{R}})}$ and the query ρ into a standard inference problem in a probabilistic graphical model (e.g., Ngo and Haddawy, 1995; Jaeger, 1997; Richardson and Domingos, 2006), or into a *weighted model counting* problem (e.g., Fierens et al., 2011). These compilation targets are *ground* models in the sense that they are expressed in terms of ground atoms $r(\boldsymbol{i})$ as basic random variables. While often effective, they still may exhibit a computational complexity that is exponential in $|V|$. The idea of *lifted inference* is to exploit uniformities and symmetries in the model $P_{(V,\tilde{\boldsymbol{R}})}$, which may allow us to sum over whole classes of essentially indistinguishable interpretations $\boldsymbol{R}$ at once (Poole, 2003; Van den Broeck, 2011). While more scalable than ground inference in some cases, the potential of such lifted techniques is still limited by general intractability results: Jaeger (2000) shows that inference for single atom queries is exponential in $|V|$ in the worst case,[2] if the modeling framework is expressive enough to support FOL features. Van den Broeck and Davis (2012) obtain a related result under weaker assumptions on the expressiveness of the modeling framework, but for more complex queries.

When exact computation of (14) becomes infeasible, one typically resorts to approximate inference via sampling random $\boldsymbol{R}$ according to $P_{(V,\tilde{\boldsymbol{R}})}$, and taking the empirical frequency of samples that belong to ρ as an estimate of $P_{(V,\tilde{\boldsymbol{R}})}(\rho)$. When directly sampling from $P_{(V,\tilde{\boldsymbol{R}})}$ is not supported (cf. Section 5.1), then this is performed by a *Markov Chain Monte Carlo* approach where samples follow the distribution $P_{(V,\tilde{\boldsymbol{R}})}$ only in the limit of the sampling process.

5.3. Cross-domain queries

The reasoning tasks considered in the previous section arise when a single known domain of entities is the subject of inference. Cross-domain queries in the sense of the following definition capture reasoning tasks that arise when the exact domain is unknown, or one wants to reason about a range of possible domains at once.

Definition 5.2. Let $\mathcal{M} = (\mathcal{I}, \mu)$ be a model. A *cross-domain query* is given by

- a set $\mathcal{J} \subseteq \mathcal{I}$ of input graphs
- a *query set* $\rho = \{\rho_{(V,\tilde{\boldsymbol{R}})} | (V, \tilde{\boldsymbol{R}}) \in \mathcal{J}\}$, where each $\rho_{(V,\tilde{\boldsymbol{R}})} \subseteq Int_{(V,\tilde{\boldsymbol{R}})}$
- a *query objective* that consists of calculating a property of the set of sets

$$\{\{P_{(V,\tilde{\boldsymbol{R}})}(\boldsymbol{R}) | \boldsymbol{R} \in \rho\} | (V, \tilde{\boldsymbol{R}}) \in \mathcal{J}\}. \tag{18}$$

Definition 5.1 now is just the special case $|\mathcal{J}| = 1$ of Definition 5.2.

Example 5.3. (Deduction, cont.) Consider a logic knowledge base KB as a discriminative model as described in Example 3.7. The question whether KB implies a statement ϕ then is a cross-domain query where $\mathcal{J}$ contains the set of graphs $G \in \mathcal{G}(<\infty, \mathcal{R})$ in which ϕ does not hold, ρ_G is the extension of G in which $l_{KB} = 1$, and the query objective is to decide whether $P_G(\rho_G) = 0$ for all $G \in \mathcal{J}$.

Example 5.4. (Model explanation) Consider a GNN model for the graph classification task described in Example 1.3. This will be a discriminative model that takes fully specified graphs representing molecules as input, and returns a probability distribution over the *mutagenic* graph label as output. An approach to *explain* such a model is to identify for a given target value of the graph label, e.g., *mutagenic* $=$ *true*, the input molecule that leads to the highest probability for that label (which can be interpreted as an ideal prototype for the label—according to the model we are trying to explain; Yuan et al., 2020). Finding such an explanation is a cross-domain query in our sense: the set $\mathcal{J}$ is the set of all possible input molecules (or the set of all molecules with a given size). For all $(V, \boldsymbol{R}) \in \mathcal{J}$ the query set is the single labeled molecule $\rho_{(V,\boldsymbol{R})} = \{(V, \boldsymbol{R}, mutagenic = true)\}$, and the objective is to find the argmax of (18). This is very similar to the most probable genotype queries considered above, but subtly different: since the underlying model here only is discriminative for the class label, the maximization of (18) does not depend on any prior probabilities for the input graphs $(V, \boldsymbol{R})$. Also, what is a cross-domain query for a discriminative model can just be a single domain query for a fully generative model: if here we are looking for an explanation of a fixed size n, then we only need to consider the single input domain $V = [n]$, the query set $\rho_{(V)} = \{([n], \boldsymbol{R}, mutagenic = true) | \boldsymbol{R} \in Int_{([n])}(\mathcal{R})\}$, and the query objective (17). While on the one hand merely a technical semantic distinction, this can make a significant difference when in the first case the existing framework does not directly support the argmax computation for (18), whereas in the second case the computation of (17) may be supported by native algorithmic tools in the framework.

Example 5.5. (Limit behavior, cont.) For a fully generative model consider $\mathcal{J} = \mathcal{I} = \mathcal{G}(<\infty, \emptyset)$. For each input graph $G_n := ([n], \emptyset)$ let $\rho_n := \rho_{G_n}$ contain the set of graphs with some property of interest, such as being connected, or satisfying a given logic formula. An important cross-domain reasoning task then is characterized by the query objective to determine the existence and value of the limit $\lim_{n\to\infty}[P_n(\rho_n)]$.

Automated theorem provers provide algorithmic solutions for the reasoning tasks described in Example 5.3. For the analysis of limit probabilities as in Example 5.5 there exist a few theoretical results that puts them into the reach of general automated inference methods (Grandjean, 1983; Jaeger, 1998; Cozman and Mauá, 2019; Koponen and Weitkämper, 2023). However, these results have not yet been carried over to operational implementations.

6. Learning: settings and techniques

So far our discussion has largely focused on modeling and reasoning. Our formal definitions in Sections 3 and 5 draw a close link between types of models and the reasoning tasks they support (discriminative, generative, transductive, inductive, ...). Turning now to the question of how a model is learned from data, a similar close linkage arises between model types and learning scenarios. Based on the unifying probabilistic view of models according to Definition 3.1, different learning scenarios are essentially just distinguished by the structure of the training data, but unified by a common maximum likelihood learning principle.

[2] Subject to the complexity-theoretic assumption that NETIME≠ETIME.

The class of input structures $\mathcal{I}$ of a model $\mathcal{M} = (\mathcal{I}, \mu)$ characterizes its basic functionality and will be assumed to be fixed a-priori by the learning/reasoning task at hand. What is to be learned is the mapping μ. For this one needs training data of the following form.

Definition 6.1. Let $\mathcal{I} \subseteq \tilde{\mathcal{G}}(< \infty, \mathcal{R})$ be given. A dataset for learning the mapping μ consists of a set of examples

$$(V_n, \tilde{\boldsymbol{R}}_n), \tilde{\boldsymbol{R}}_n^+ \ (n = 1, \ldots, N), \tag{19}$$

where $(V_n, \tilde{\boldsymbol{R}}_n) \in \mathcal{I}$, and $\tilde{\boldsymbol{R}}_n^+ \in \tilde{\mathcal{G}}(V_n, \mathcal{R})$ complements $\tilde{\boldsymbol{R}}_n$ in the sense that $\tilde{R}_{n,i}^+(\boldsymbol{i}) \neq ? \Rightarrow \tilde{R}_{n,i}(\boldsymbol{i}) = ?$ $(i = 1, \ldots, m)$.

The unifying learning principle is to find the mapping μ that maximizes the log-likelihood

$$\sum_{n=1}^{N} \log \mu(V_n, \tilde{\boldsymbol{R}}_n)(\tilde{\boldsymbol{R}}_n^+). \tag{20}$$

In practice, the pure likelihood (20) will often be modified by regularization terms or prior probabilities, which, to simplify matters, we do not consider here. Furthermore, some learning objectives, such as the max-margin objective in learning a kernel-SVM, are not based on the log-likelihood as the central element at all. However, as the following examples show, Equation (20) still covers a fairly wide range of learning approaches.

Example 6.2. (Node classification with GNN) Consider $\mathcal{R}$ consisting of a single binary *edge* relation, node attributes $a_1, \ldots, a_k$, and a node label l. For training a discriminative model with $\mathcal{R}_{in} = \{edge, a_1, \ldots, a_k\}$, and $\mathcal{R}_{out} = \{l\}$, the training examples consist of complete input graphs $(V_n, \boldsymbol{R}_n) \in \mathcal{G}(< \infty, \mathcal{R}_{in})$, and $\tilde{\boldsymbol{R}}_n^+$ is a partial interpretation $\tilde{L}_n$ of l. In the transductive setting, $N = 1$, and $(V_1, \boldsymbol{R}_1)$ is equal to the single input graph for which the model is defined. Under the factorization (7) the log likelihood (20) then becomes

$$\sum_{n=1}^{N} \sum_{i\,:\,\tilde{L}_n(i) \neq ?} \log \mu(V_n, \boldsymbol{R}_n)(\tilde{L}_n(i)).$$

The usual training objective for GNNs of minimizing the *log-loss* is equivalent to maximizing this log-likelihood.

Example 6.3. (Generative models from incomplete data) The discriminative learning scenario of Example 6.2 requires training data in which input relations $\mathcal{R}_{in}$ are completely observed. When also some of the attributes a_i, and maybe the *edge* relation are only incompletely observed in the training data, then no valid input structure for a discriminative model is given. However, no matter which relations are fully or partially observed, a partial interpretation $\tilde{\boldsymbol{R}}_n$ can always be written as a valid training example

$$(V_n, \emptyset), \tilde{\boldsymbol{R}}_n \tag{21}$$

for a fully generative model. Without any assumptions on the factorization of the distributions P_n, the log-likelihood now takes the general form

$$\sum_{n=1}^{N} \log \mu(V_n, \emptyset)(\tilde{\boldsymbol{R}}_n) = \sum_{n=1}^{N} \sum_{\boldsymbol{R} \in Int_{(V_n, \tilde{\boldsymbol{R}}_n)}(\mathcal{R})} \log \mu(V_n, \emptyset)(\boldsymbol{R}).$$

While always well-defined, this likelihood may be intractable for optimization. An explicit summation over all $\boldsymbol{R} \in Int_{(V_n, \boldsymbol{R}_n)}(\mathcal{R})$ is almost always infeasible. When optimization of the *complete data likelihood* $\mu(V_n, \emptyset)(\boldsymbol{R})$ for $\boldsymbol{R} \in Int_V(\mathcal{R})$ is tractable, then the *expectation-maximization* strategy can be applied, where one iteratively imputes expected completions $\boldsymbol{R}_n$ for the incomplete observations $\boldsymbol{R}_n$ under a current model μ, and then optimizes μ under the likelihood induced by this complete data. For U-SRL models, even the complete data likelihood usually is intractable due to its dependence on the *partition function* $Z = Z(\mu)$ in (8). In this case the true likelihood may be approximated by a *pseudo-likelihood* (Besag, 1975; Richardson and Domingos, 2006).

Example 6.4. (Learning logic theories) Consider now the case of a logical framework where a model is a knowledge base *KB* that we consider as a discriminative model in the sense of Example 3.7. Learning logical theories or concepts is usually framed in terms of learning from positive and negative examples, where the example data can consist of full interpretations, logical statements, or even proofs (De Raedt, 1997). The learning from interpretations setting fits most closely our general data and learning setup: examples then are fully observed graphs $G_n = (V_n, \boldsymbol{R}_n)$ together with a label $\tilde{\boldsymbol{R}}_n^+ = L_n \in \{0, 1\}$, and optimizing (20) amounts to finding a knowledge base *KB* such that *KB* is true in all G_n with $L_n = 1$, and false for G_n with $L_n = 0$, i.e., the standard objective in logic concept learning.

6.1. Numeric and symbolic optimization

Examples 6.2–6.4 present a uniform perspective on learning in very different frameworks. However, the required techniques for solving the likelihood optimization problem are quite diverse. Corresponding to the combination of symbolic and numeric representation elements of a modeling framework, the learning problem decomposes into a *structure learning* part for the symbolic representation, and a *parameter learning* part for the numeric elements. Since the numeric parameterization usually depends on the chosen structure, this can lead to a nested optimization in which structure learning is performed in an outer loop that contains parameter learning as an inner loop. Structure learning amounts to search in a potentially infinite combinatorial space. Parameter learning, on the other hand, typically is reduced to the optimization of a differentiable objective function, for which powerful gradient-based methods are available.

The (empirical) fact that numeric optimization of parameters is somewhat easier than combinatorial optimization of symbolic structure favors frameworks that are primarily numeric, notably GNNs. As illustrated in Section 4.2.4, feature constructions that in other frameworks require symbolic specifications (12), (13) here are encoded in numeric parameter matrices (11). As a result, learning that in symbolic representations requires a search over symbolic representations, here is accomplished by numeric optimization. However, even GNNs are not completely devoid of "symbolic" representation elements: the neural network architecture is a component of the model specification in a discrete design space, and finding the best architecture via *neural architecture search* (Elsken et al., 2019) leads to optimization problems in discrete, combinatorial search spaces that have much in common with structure learning in more manifestly symbolic frameworks.

While presenting a harder optimization task for machine learning, symbolic, structural parts of a model may also be supplied manually by domain experts, thus reducing the machine learning task to the optimization of the numeric parameters. Many SRL frameworks, so far, rely to a greater or lesser extent on such a humans-in-the-loop scenario.

7. Integration

In view of the complementary benefits of symbolic and numeric models, it is natural to aim for combinations of both elements that optimally exploit the strengths of each. Emphasizing neural network frameworks as the prime representatives for numeric approaches, these efforts are currently mostly pursued under the name of *neuro-symbolic integration* (Sarker et al., 2021). An important example in the context of this review is the integration of the I-SRL ProbLog framework with deep neural networks (not specifically GNNs). The underlying philosophy for the proposed *DeepProbLog* framework (Manhaeve et al., 2018) is that neural frameworks excel at solving low-level "perceptual" reasoning tasks, whereas symbolic frameworks support higher-level reasoning. The integration therefore consists of two layers, where the lower (neural) layer provides inputs to the higher (symbolic) layer.

The unifying perspective on model semantics and model structure we have developed in Sections 3 and 4 gives rise to more homogeneous integration perspectives: a conditionally generative model that is constructed via the factorization (6), (7) consists of individual discriminative models (7) for each relation R_k as building blocks. In principle, different such building blocks can be constructed in different frameworks, and combined into a single model via (6). Moreover, this approach will be consistent in the sense that if all constructions of the component discriminative models use (20) as the objective, then the combination of these objectives is equivalent to the maximizing the overall log-likelihood $\mu(V_n, \boldsymbol{R}_{in})(\boldsymbol{R}_{out})$ directly for the resulting conditional generative model. Here we deliberately speak of "constructing" rather than "learning" component models in order to emphasize the possibility that some model components may be built by manual design, whereas others can be learned from data.

Piecing together a combined model from heterogeneous model components only is useful when the resulting model then can be used to perform inference tasks. This will limit the reasoning capabilities to tasks that can be broken down into a combination of tasks supported by the component models. A possible alternative is to compile all individual components into a representation in a common framework with high expressivity and flexible reasoning capabilities. As shown in Jaeger (2022), quite general GNN architectures for representing discriminative models can be compiled into an RBN representation, and integrated as components into a bigger generative model. While theoretically sound, it is still an open question whether this approach is practically feasible for GNN models of the size needed to obtain high accuracy on their specialized discriminative tasks.

8. Conclusion

We have given a broad overview over modeling, reasoning, and learning with graphs. The main objective of this review was to view the area from a broader perspective than more specialized existing surveys, while at the same time developing a coherent conceptual framework that emphasizes the commonalities between very diverse approaches to dealing with graph data. Our central definitions of models, reasoning types and learning tasks cover a wide range of different frameworks and approaches. While the uniformity we thereby obtain sometimes is a bit contrived (notably by casting logical concepts in probabilistic terms), it still may be useful to elucidate the common ground among quite disparate traditions and approaches, to provide a basis for further theoretical (comparative) analyses, and to facilitate the combination and integration of different frameworks.

Author contributions

The author confirms being the sole contributor of this work and has approved it for publication.

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Abboud, R., Ceylan, I. I., Grohe, M., and Lukasiewicz, T. (2021). "The surprising power of graph neural networks with random node initialization," in *Proceedings of IJCAI 2021*. doi: 10.24963/ijcai.2021/291

Barceló, P., Kostylev, E., Monet, M., Pérez, J., Reutter, J., and Silva, J.-P. (2020). "The logical expressiveness of graph neural networks," in *8th International Conference on Learning Representations (ICLR 2020)*.

Besag, J. (1975). Statistical analysis of non-lattice data. *J. R. Stat. Soc. Ser. D* 24, 179–195. doi: 10.2307/2987782

Blockeel, H., and De Raedt, L. (1998). Top-down induction of first-order logical decision trees. *Artif. Intell.* 101, 285–297. doi: 10.1016/S0004-3702(98)00034-4

Bonifati, A., Holubová, I., Prat-Pérez, A., and Sakr, S. (2020). Graph generators: state of the art and open challenges. *ACM Comput. Surveys* 53, 1–30. doi: 10.1145/3379445

Breese, J. S., Goldman, R. P., and Wellman, M. P. (1994). Introduction to the special section on knowledge-based construction of probabilistic decision models. *IEEE Trans. Syst. Man Cybern.* 24, 1580–1592. doi: 10.1109/21.328909

Cozman, F. G., and Mauá, D. D. (2019). The finite model theory of Bayesian network specifications: Descriptive complexity and zero/one laws. *Int. J. Approx. Reason.* 110, 107–126. doi: 10.1016/j.ijar.2019.04.003

Dai, H., Kozareva, Z., Dai, B., Smola, A., and Song, L. (2018). "Learning steady-states of iterative algorithms over graphs," in *International Conference on Machine Learning (PMLR)*, 1106–1114. Available online at: https://proceedings.mlr.press/v80/dai18a.html

Dai, H., Nazi, A., Li, Y., Dai, B., and Schuurmans, D. (2020). "Scalable deep generative modeling for sparse graphs," in *International Conference on Machine Learning*, 2302–2312.

De Raedt, L. (1997). Logical settings for concept-learning. *Artif. Intell.* 95, 187–201. doi: 10.1016/S0004-3702(97)00041-6

De Raedt, L., Kimmig, A., and Toivonen, H. (2007). "Problog: a probabilistic prolog and its application in link discovery," in *IJCAI, Vol. 7*, 2462–2467.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: a survey. *J. Mach. Learn. Res.* 20, 1997–2017. doi: 10.1007/978-3-030-05318-5_3

Erdős, P., and Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.* 5, 17–60.

Fagin, R. (1976). Probabilities on finite models. *J. Symb. Logic* 41, 50–58. doi: 10.2307/2272945

Fierens, D., den Broeck, G. V., Thon, I., Gutmann, B., and De Raedt, L. (2011). "Inference in probabilistic logic programs using weighted CNF's," in *Proceedings of UAI 2011.*

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. (1999). "Learning probabilistic relational models," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99).*

Gärtner, T., Flach, P., and Wrobel, S. (2003). "On graph kernels: hardness results and efficient alternatives," in *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003* (Springer), 129–143. doi: 10.1007/978-3-540-45167-9_11

Grandjean, E. (1983). Complexity of the first-order theory of almost all finite strucutures. *Inform. Control* 57, 180–204. doi: 10.1016/S0019-9958(83)80043-6

Grover, A., and Leskovec, J. (2016). "node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–864. doi: 10.1145/2939672.2939754

Halpern, J. Y., and Vardi, M. Y. (1991). "Model checking vs. theorem proving: a manifesto," in *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, 151–176. doi: 10.1016/B978-0-12-450010-5.50015-3

Hamilton, W. L. (2020). *Graph Representation Learning, Vol. 46 of Synthesis Lectures on Artifical Intelligence and Machine Learning.* Morgan & Claypool Publishers. doi: 10.1007/978-3-031-01588-5

Hamilton, W. L., Ying, Z., and Leskovec, J. (2017). "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017* (Long Beach, CA), 1024–1034.

Harrison, J. (1996). "HOL light: A tutorial introduction," in *International Conference on Formal Methods in Computer-Aided Design* (Springer), 265–269. doi: 10.1007/BFb0031814

Heckerman, D., Meek, C., and Koller, D. (2007). "Probabilistic entity-relationship models, PRMs, and plate models," in *Introduction to Statistical Relational Learning*, eds L. Getoor and B. Taskar (MIT Press). doi: 10.7551/mitpress/7432.003.0009

Holland, P. W., Laskey, K. B., and Leinhardt, S. (1983). Stochastic blockmodels: first steps. *Soc. Netw.* 5, 109–137. doi: 10.1016/0378-8733(83)90021-7

Jaeger, M. (1997). "Relational Bayesian networks," in *Proceedings of the 13th Conference of Uncertainty in Artificial Intelligence (UAI-13)*, eds D. Geiger and P. P. Shenoy (Providence, RI: Morgan Kaufmann), 266–273.

Jaeger, M. (1998). "Convergence results for relational Bayesian networks," in *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS-98)*, ed V. Pratt (Indianapolis, IN: IEEE Technical Committee on Mathematical Foundations of Computing; IEEE Computer Society Press), 44–55. doi: 10.1109/LICS.1998.705642

Jaeger, M. (2000). On the complexity of inference about probabilistic relational models. *Artif. Intell.* 117, 297–308. doi: 10.1016/S0004-3702(99)00109-5

Jaeger, M. (2022). "Learning and reasoning with graph data: neural and statistical-relational approaches," in *International Research School in Artificial Intelligence in Bergen (AIB 2022), Vol. 99 of Open Access Series in Informatics (OASIcs)*, eds C. Bourgaux, A. Ozaki, and R. Pe naloza (Schloss Dagstuhl-Leibniz-Zentrum für Informatik), 5:1–5:42.

Kersting, K., and De Raedt, L. (2001). "Towards combining inductive logic programming and Bayesian networks," in *Proceedings of the Eleventh International Conference on Inductive Logic Programming (ILP-2001).* doi: 10.1007/3-540-44797-0_10

Kipf, T. N., and Welling, M. (2016). Variational graph auto-encoders. *arXiv [Preprint].* arXiv: 1611.07308. doi: 10.48550/arXiv.1611.07308

Kipf, T. N., and Welling, M. (2017). "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations.* Available online at: https://openreview.net/forum?id=SJU4ayYgl

Koponen, V., and Weitkämper, F. (2023). Asymptotic elimination of partially continuous aggregation functions in directed graphical models. *Infm. Comput.* 293, 105061. doi: 10.1016/j.ic.2023.105061

Koren, Y., and Bell, R. (2015). "Advances in collaborative filtering," in *Recommender Systems Handbook,* eds F. Ricci, L. Roakach, and B. Shapira (Boston, MA: Springer), 77–118. doi: 10.1007/978-1-4899-7637-6_3

Kriege, N. M., Johansson, F. D., and Morris, C. (2020). A survey on graph kernels. *Appl. Netw. Sci.* 5, 1–42. doi: 10.1007/s41109-019-0195-3

Kumar, A., Singh, S. S., Singh, K., and Biswas, B. (2020). Link prediction techniques, applications, and performance: a survey. *Phys. A Stat. Mech. Appl.* 553:124289. doi: 10.1016/j.physa.2020.124289

Laskey, K. B. (2008). MEBN: a language for first-order Bayesian knowledge bases. *Artif. Intell.* 172, 140–178. doi: 10.1016/j.artint.2007.09.006

Laskey, K. B., and Mahoney, S. M. (1997). "Network fragments: representing knowledge for constructing probabilistic models," in *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)* (San Francisco, CA: Morgan Kaufmann Publishers), 334–341.

Leskovec, J., Kleinberg, J., and Faloutsos, C. (2007). Graph evolution: densification and shrinking diameters. *ACM Trans. Knowl. Discov Data* 1:2-es. doi: 10.1145/1217299.1217301

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. (2018). Learning deep generative models of graphs. *arXiv [Preprint].* arXiv: 1803.03324. doi: 10.48550/arXiv.1803.03324

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. (2018). "DeepProbLog: Neural probabilistic logic programming," in *Advances in Neural Information Processing Systems, Vol. 31.* p. 3749–3759. Available online at: https://proceedings.neurips.cc/paper_files/paper/2018/file/dc5d637ed5e62c36ecb73b654b05ba2a-Paper.pdf

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., et al. (2019). "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 4602–4609. doi: 10.1609/aaai.v33i01.33014602

Muggleton, S. (1995). Inverse entailment and progol. *New Generat. Comput.* 13, 245–286.

Ngo, L., and Haddawy, P. (1995). "Probabilistic logic programming and Bayesian networks," in *Algorithms, Concurrency and Knowledge (Proceedings ACSC95), Springer Lecture Notes in Computer Science 1023*, 286–300. doi: 10.1007/3-540-60688-2_51

Niepert, M., Ahmed, M., and Kutzkov, K. (2016). "Learning convolutional neural networks for graphs," in *International Conference on Machine Learning*, 2014–2023.

Pan, L., Shi, C., and Dokmanić, I. (2021). "Neural link prediction with walk pooling," in *International Conference on Learning Representations.*

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710. doi: 10.1145/2623330.2623732

Poole, D. (1997). The independent choice logic for modelling multiple agents under uncertainty. *Artif. Intell.* 94, 7–56. doi: 10.1016/S0004-3702(97)00027-1

Poole, D. (2003). "First-order probabilistic inference," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03).*

Quinlan, J. R., and Cameron-Jones, R. M. (1993). "FOIL: A midterm report," in *Machine Learning: ECML-93: European Conference on Machine Learning Vienna, Austria, April 5–7, 1993 Proceedings 6* (Springer), 1–20. doi: 10.1007/3-540-56602-3_124

Richardson, M., and Domingos, P. (2006). Markov logic networks. *Mach. Learn.* 62, 107–136. doi: 10.1007/s10994-006-5833-1

Sarker, M. K., Zhou, L., Eberhart, A., and Hitzler, P. (2021). Neuro-symbolic artificial intelligence: current trends. *arXiv preprint arXiv:2105.05330.* doi: 10.3233/AIC-210084

Sato, R. (2020). A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*.

Sato, R., Yamada, M., and Kashima, H. (2021). "Random features strengthen graph neural networks," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)* (SIAM), 333–341. doi: 10.1137/1.9781611976700.38

Sato, T. (1995). "A statistical learning method for logic programs with distribution semantics," in *Proceedings of the 12th International Conference on Logic Programming (ICLP'95)*. p. 715–729.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE Trans. Neur. Netw.* 20, 61–80. doi: 10.1109/TNN.2008.2005605

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. (2009). "Efficient graphlet kernels for large graph comparison," in *Artificial Intelligence and Statistics*, 488–495.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.* 12, 2539–2561. doi: 10.5555/1953048.2078187

Simonovsky, M., and Komodakis, N. (2018). "Graphvae: towards generation of small graphs using variational autoencoders," in *International Conference on Artificial Neural Networks* (Springer), 412–422. doi: 10.1007/978-3-030-01418-6_41

Srinivasan, A., Muggleton, S. H., Sternberg, M. J., and King, R. D. (1996). Theories for mutagenicity: a study in first-order and feature-based induction. *Artif. Intell.* 85, 277–299. doi: 10.1016/0004-3702(95)00122-0

Van den Broeck, G., and Davis, J. (2012). "Conditioning in first-order knowledge compilation and lifted probabilistic inference," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Van den Broeck, G. (2011). "On the completeness of first-order knowledge compilation for lifted probabilistic inference," in *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). "Graph attention networks," in *International Conference on Learning Representations*.

Vignac, C., Loukas, A., and Frossard, P. (2020). "Building powerful and equivariant graph neural networks with structural message-passing," in *NeurIPS*.

Wagstaff, E., Fuchs, F., Engelcke, M., Posner, I., and Osborne, M. A. (2019). "On the limitations of representing functions on sets," in *International Conference on Machine Learning*, 6487–6494.

Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., and Wischnewski, P. (2009). "Spass version 3.5," in *International Conference on Automated Deduction* (Springer), 140–145. doi: 10.1007/978-3-642-02959-2_10

Welling, M., and Kipf, T. N. (2017). Semi-supervised classification with graph convolutional networks.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). "How powerful are graph neural networks?," in *International Conference on Learning Representations*.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. (2018). "GraphRNN: generating realistic graphs with deep auto-regressive models," in *International Conference on Machine Learning*, 5708–5717.

Yuan, H., Tang, J., Hu, X., and Ji, S. (2020). "XGNN: towards model-level explanations of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 430–438. doi: 10.1145/3394486.3403085

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). "Deep sets," in *Advances in Neural Information Processing Systems, Vol. 30*, eds I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc.).