Journal of Machine Learning Research 24 (2023) 1-34 Submitted 6/22; Revised 12/22; Published 3/23

Knowledge Hypergraph Embedding Meets Relational Algebra

Bahare Fatemi BFATEMI@CS.UBC.CA
University of British Columbia
Vancouver, BC V6T 1Z4, Canada

Perouz Taslakian PEROUZ.TASLAKIAN@SERVICENOW.COM
ServiceNow Research
Montreal, QC H2S 3G9, Canada

David Vazquez DAVID.VAZQUEZ@SERVICENOW.COM
ServiceNow Research
Montreal, QC H2S 3G9, Canada

David Poole POOLE@CS.UBC.CA
University of British Columbia
Vancouver, BC V6T 1Z4, Canada

Editor: Gal Elidan

Abstract

Relational databases are a successful model for data storage, and rely on query languages for information retrieval. Most of these query languages are based on relational algebra, a mathematical formalization at the core of relational models. Knowledge graphs are flexible data storage structures that allow for knowledge completion using machine learning techniques. *Knowledge hypergraphs* generalize knowledge graphs by allowing multi-argument relations. This work studies knowledge hypergraph completion through the lens of relational algebra and its core operations. We explore the space between relational algebra foundations and machine learning techniques for knowledge completion. We investigate whether such methods can capture high-level abstractions in terms of relational algebra operations. We propose a simple embedding-based model called *Relational Algebra Embedding* (ReAlE) that performs link prediction in knowledge hypergraphs. We show theoretically that ReAlE is fully expressive and can represent the relational algebra operations of renaming, projection, set union, selection, and set difference. We verify experimentally that ReAlE outperforms state-of-the-art models in knowledge hypergraph completion, and in representing each of these primitive relational algebra operations. For the latter experiment, we generate a synthetic knowledge hypergraph, for which we design an algorithm based on the Erdős-Rényi model for generating random graphs.

Keywords: Knowledge Hypergraphs, Relational Algebra, Knowledge Hypergraph Completion.

1. Introduction

Knowledge hypergraphs are knowledge bases that store information about the world in the form of tuples describing relations among entities. Knowledge graphs are a specific form of knowledge hypergraphs that represent relations between *exactly* two entities. Knowledge hypergraphs thus generalize knowledge graphs by allowing multi-argument relations. Knowledge hypergraphs are inherently incomplete; for example, Wikidata (Vrandečić and Krötzsch, 2014) contains a very small proportion of the true statements about the notable people included. The goal of *link prediction* in

knowledge hypergraphs (or *knowledge hypergraph completion*) is to predict unknown relationships among entities based on existing ones.

The most dominant recent paradigm on knowledge completion has been to learn and reason on knowledge graphs, but the original structure of many existing graph datasets is in terms of more than just binary relations. Wen et al. (2016) observe that in the original FREEBASE (Bollacker et al., 2008) more than $\frac{1}{3}$ of the entities participate in non-binary relations. Fatemi et al. (2020) observe, in addition, that 61% of the relations in the original FREEBASE are non-binary. While it is possible to convert a knowledge hypergraph into a knowledge graph and apply existing methods on it, Fatemi et al. (2020) show that embedding-based methods for knowledge graph completion do not work well out of the box for knowledge graphs obtained through such conversion techniques. An alternative way is to modify the learning model to explicitly handle non-binary relations. However, a model designed to reason over binary relations does not necessarily generalize well to non-binary relations. With most existing models being extensions of those used for link prediction in knowledge graphs, knowledge hypergraph completion remains a relatively underexplored research area.

Recent research (*e.g.*, Battaglia et al., 2018; Teru et al., 2020) has highlighted the importance of relational inductive biases in building learning agents that learn entity-independent relational semantics and reason in a compositional manner. In this work, we explore the foundations of knowledge hypergraph completion; we aim to design a model for reasoning in knowledge hypergraphs that is simple, expressive, and can represent high-level abstractions in terms of the operations of relational databases. We hypothesize that models that can reason about relations in terms of relational algebra operations have better generalization power. Many relational methods involve explicit knowledge construction (*e.g.*, rules, constraints, or ontologies) that can be used to extend a partial knowledge base or detect inconsistencies. Our focus in this work is to design a model that exploits relational inductive biases by incorporating relationships among relations, without explicitly constructing this knowledge. In the proposed model, all the relationships are implicit in the similarities in the learned parameters for relations.

Constraint vs. query languages. In relational databases, constraints and queries play complementary roles. Constraints specify restrictions to impose on the database, while queries extract information that may not be explicitly encoded. In other words, constraints imply that some combination of tuples must be false, while queries help us determine what else must be true. Both constraints and queries can be written as (a subset of) first-order logic rules. Consider the following example. Suppose the data tells us that *sam* lives with *sally* (which we write as $livesWith(sam, sally)$) and *sally* lives in *paris* (written as $livesIn(sally, paris)$). A query language would let us state that someone lives in the same city as a person they live with, which would let us infer $livesIn(sam, paris)$. The constraint that someone only lives in one city makes us reject the statement $livesIn(sally, berlin)$.

Unlike in constraint languages, defining one relation in terms of others in a query language does not have *side effects* – i.e. the representation of a relation r does not affect the truth of relations not defined in terms of r, which is crucial when we want to capture multiple inference patterns jointly. We can think of a query language as being similar to a directed graphical model and a constraint language as an undirected graphical model. In a directed graphical model, we add a new random variable by using existing variables as its parents. Adding the variable does not change the distribution of the parents. Whereas, in an undirected graphical model, adding a new variable can change the distribution of other variables; indeed for relational undirected models (in particular Markov logic networks), it has been proved that it is impossible to add a new variable without changing the distribution over the existing variables except in trivial cases (Buchman and Poole, 2015). A learning model that is

based on a query language (e.g. one with relational algebra foundations) can easily capture multiple inference patterns jointly, a challenge underlined by Abboud et al. (2020) who state that "capturing multiple inference patterns jointly is significantly more challenging [than capturing them singly]." This is true of languages that include constraints, but not of query languages.

Relational algebra. *Relational algebra* is a formalization of queries and defines relations (views) at the core of relational databases. It consists of several *primitive operations* that can be combined to synthesize all other operations used. The primitive operations are renaming, projection, selection, set union, set difference, and Cartesian product. Each such operation takes relations as input and returns a relation as output. Renaming changes the order of the entities in a relation. Projection takes a relation and some positions as input and returns a new relation with the entities in specific positions removed from each tuple. Selection returns a subset of tuples for a relation that satisfies a given condition. Set union takes as input two relations of the same arity and returns a new relation containing the tuples that appear in at least one of the relations. Set difference also gets two relations of the same arity as input and returns a new relation containing tuples from the first relation that do not appear in the second relation. Cartesian product takes two relations and returns a relation in which the tuples are the concatenation of the tuples of the input relations. One non-primitive operation is join, which is Cartesian product followed by selection and projection; selecting the elements in the Cartesian product with matching values on corresponding attributes, and projecting onto the different attributes. Another non-primitive operation is set intersection, which can be defined in terms of set union and set difference.

Relational Algebra in Knowledge Hypergraphs. In a knowledge hypergraph, the relational algebra operations can describe how relations depend on each other. To illustrate the connection between relational algebra operations and relations in knowledge hypergraphs, consider the example in Figure 1 that shows tuples from a train and test splits of a knowledge hypergraph. The train set contains tuples $sold(drew, alex, book)$, $buyer(alex, book)$, $sold(mike, sam, tv)$, and $bought(sam, mike, tv)$. The relations in the example feature the two primitive relational algebra operations renaming and projection. Relation *bought* is a renaming of *sold*. Relation *buyer* is a projection of relation *sold*. If a model is able to represent these two operations, it can potentially learn at train time that a tuple $bought(X, Y, I)$ (person X bought from person Y item I) is implied by tuple $sold(Y, X, I)$ (person Y sold to person X item I); or that a tuple $buyer(X, I)$ (person X is the buyer of item I) is implied by the tuple $sold(Y, X, I)$. An embedding model that cannot represent the operations renaming and projection would not be able to learn that relation *bought* in Figure 1 is a renaming of relation *sold*. It would thus be difficult for such a model to reason about the relationship between these two relations. In contrast, a model that can represent renaming and projection operations is potentially able to determine that $bought(alex, drew, book)$ is true because the train set contains $sold(drew, alex, book)$ and *bought* is a renaming of *sold*, $buyer(sam, tv)$ is true because the train set contains $sold(mike, sam, tv)$ and *buyer* is a projection of *sold*, and $sold(mike, sam, tv)$ is true because the train set contains $bought(sam, mike, tv)$ and *sold* is a renaming of *bought*.

Designing reasoning methods that can capture the relational semantics in terms of relational algebra is especially important in the context of knowledge hypergraphs, where relations can be defined on an arbitrary number of entities. Domains with beyond-binary relations provide multiple methods of expressing the same underlying notion, as seen in the above example where relations *sold* and *bought* encode the same information. Since all relations in a knowledge graph are binary (have arity 2), many of the relational algebra operations (such as projection, which changes the arity of the relation; for instance, *buyer* with arity 2 is a projection of *sold* with arity 3), are not applicable to this

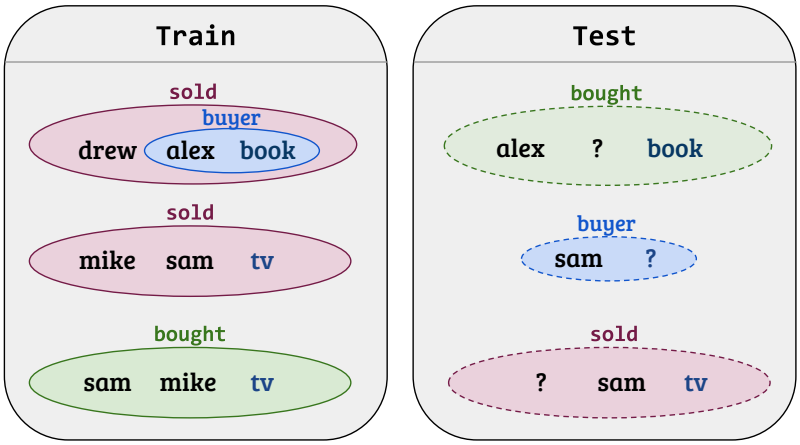

Figure 1: An example of a knowledge hypergraph. The train set contains tuples $sold(drew, alex, book)$, $buyer(alex, book)$, $sold(mike, sam, tv)$, and $bought(sam, mike, tv)$. Relation *bought* can be obtained by applying a renaming operation to relation *sold*. Similarly, relation *buyer* is a projection of relation *sold*. Learning these relational algebra operations can help the model generalize to the tuples in the test set. The test responses, from top to bottom, are *drew*, *tv*, and *mike*.

setting. This also highlights the importance of knowledge hypergraphs as a data model that encodes rich relational structures ripe for further exploration.

The main contributions of this work are summarized as follows.

- We introduce ReAlE, an embedding-based method for knowledge hypergraph completion that can provably represent the relational algebra operations renaming, projection, set union, selection, and set difference,
- A *framework* for generating synthetic knowledge hypergraphs, which is based on the Erdős-Rényi random graph generation model and can generate relations by repeated application of primitive relational algebra operations.
- *Experimental results* that show that ReAlE outperforms or is comparable to the state-of-the-art on well-known public datasets, and the synthesized dataset.

2. Related work

Existing work for knowledge hypergraph completion can be grouped into the following categories.

Statistical relational learning. Models under the umbrella of statistical relational learning (Raedt et al., 2016) can handle variable arity relations and explicitly model the inter-dependencies of relations. Our work is complementary to these approaches in that ReAlE is an embedding-based model that represents relational algebra operations implicitly, rather than representing them explicitly.

Translational models. Translational models for knowledge hypergraphs (Wen et al., 2016; Zhang et al., 2018; Guan et al., 2019) are extensions of approaches for binary relations and have

restrictions on the types of relations they can model (see Section 5). One of the earliest models in this category is m-TransH (Wen et al., 2016), which extends TransH (Wang et al., 2014) to knowledge hypergraph embedding. RAE (Zhang et al., 2018) extends m-TransH by adding the *relatedness* of values – the likelihood that two values co-participate in a common instance – to the loss function. NaLP (Guan et al., 2019) uses a similar strategy to RAE, but models the relatedness of values based on the roles they play in different tuples. Models in this category have restrictions on the types of relations they can model (see Sec. 5). Liu et al. (2020) discuss these limitations, and we address them more formally in Section 5. More recently, Abboud et al. (2020) proposed a fully-expressive translational model based on box embedding (Li et al., 2018). We address this work in the *Rule capturing models* paragraph.

Tensor factorization models. Models in this category extend tensor factorization models for knowledge graph completion to knowledge hypergraph completion. GETD (Liu et al., 2020) extends Tucker (Balažević et al., 2019) to n-ary relations. The memory complexity of GETD grows exponentially with the arity of relations. HypE (Fatemi et al., 2020), which is motivated by SimplE (Kazemi and Poole, 2018; Fatemi et al., 2019), disentangles the embeddings of relations from the positions of its arguments and thus the memory complexity grows linearly with the arity of the relations. HypE is fully expressive but cannot represent relational algebra operations (see Section 5).

Key-value pair based models. Models in this category (Galkin et al., 2020; Guan et al., 2020; Rosso et al., 2020) assume that a tuple is composed of a triple (binary relation), plus a list of attributes in the form of key-value pairs. The problem these works study is slightly different as we consider all information as part of a tuple. They evaluate their models on datasets for which they obtain the tuple attributes from external data sources using heuristics and thus their results are not comparable to ours.

Graph neural network models. These approaches extend graph neural networks to hypergraph neural networks (Feng et al., 2019; Yadati et al., 2018). G-MPNN (Yadati, 2020) further extends these models to knowledge hypergraphs (directed and labeled hyperedges). G-MPNN utilizes message passing in knowledge hypergraphs but the scoring function assumes relations are symmetric, and thus has restrictions in modeling the non-symmetric relations and is not fully expressive.

Rule capturing models. The closest work to ours is BoxE (Abboud et al., 2020), which is able to represent a subset of first-order logic rules and is also a translational model. ReAlE differs from BoxE in two ways. First, theoretical analysis of BoxE only concerns binary relations, while we capture inference patterns for relations defined on any number of entities; some of the relational algebra relations (e.g, projection) do not make sense with only binary relations. Second, we analyze the theoretical aspects of the proposed model in terms of relational algebra, a query language in which a relation is defined in terms of others without affecting other relations. In contrast, first-order logic rules in BoxE are a mixture of constraints (mutual exclusion and anti-symmetry) and rules that imply what else must be true (*e.g.*, symmetry and intersection). Capturing multiple patterns from first-order logic does not necessarily provide evidence for the rest of the first-order logic. Capturing the primitive operations of relational algebra is important as all other operations are composed of multiple primitive operations. We compare our model empirically with BoxE in Section 8.

Present work. Reasoning in hypergraphs is a relatively underexplored area that has recently gained more attention. Knowledge hypergraphs are isomorphic to relational databases and relational algebra is the calculus of queries in relational models. In this work, we design a model based on relational algebra operations. Besides the theoretical contributions, we show empirically how basing our model on relational algebra operations gives us improvements compared to existing work.

3. Definition and notation

Assume a finite set of entities $\mathcal{E}$ and a finite set of relations $\mathcal{R}$. Each relation has a fixed *arity*. A *tuple* is in the form of $r(x_1, \ldots, x_n)$ where $r \in \mathcal{R}$, $n = |r|$ is the arity of r, and each $x_i \in \mathcal{E}$. Let τ be a set of tuples – the ground truth – specifying all of the tuples that are true. If a tuple is not in τ, it is false. A knowledge hypergraph consists of a subset of the tuples $\tau' \subseteq \tau$. Knowledge hypergraph completion is the problem of predicting the missing tuples in τ', that is, finding the tuples $\tau \setminus \tau'$. A knowledge graph is a special case of a knowledge hypergraph where all relations have an arity 2.

An *embedding* is a function from an entity or a relation to one or more vectors, matrices, or higher-order tensors of numbers over a field (typically the real numbers). We use bold lower-case for embeddings, so that, $\mathbf{x}$ is the embedding of entity x, and $\mathbf{r}$ is the embedding of relation r. For the task of knowledge hypergraph completion, an embedding-based model having parameters θ defines a scoring function ϕ_θ that takes a tuple as input and generates a prediction, *e.g.*, a score (or a probability) of the tuple being true. A model is *fully expressive* if given any assignment of truth values to all tuples, there exists an assignment of values to θ that accurately separates the true tuples from the false ones.

Following Python notation, for a vector $\mathbf{x}$, $\mathbf{x}[k]$ represents the k-th index, for a matrix $\mathbf{r}$, $\mathbf{r}[i]$ represents the i-th row of $\mathbf{r}$ and $\mathbf{r}[i][k]$ represents the k-th index of vector $\mathbf{r}[i]$. We use $\times$ for the multiplication of two scalars. A permutation function $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ is a bijective function. For example, if $\pi = \{(1,2),(2,1),(3,3)\}$ then, $(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}) = (x_2, x_1, x_3)$.

4. ReAlE: A Basic Embedding Algorithm

ReAlE (Relational Algebra Embedding) is a knowledge hypergraph completion model that has parameters θ and a scoring function ϕ_θ. We motivate our model bottom-up by first describing an intuitive model for this task (Equation 1); we then discuss why this formulation does not work well and adjust it in ReAlE (Equation 2). Given a tuple $r(x_1, \ldots, x_n)$, determining whether it is true or false depends on the relation and the entities involved; it also depends on the position of each entity in the tuple, as the role of an entity changes with its position and relation. For example, the role of *alex* is different in the tuples $sold(drew, alex, book)$ and $sold(alex, drew, book)$ as its position is different. The role of *alex* is also different in the tuples $sold(drew, alex, book)$ and $bought(drew, alex, book)$ as the relation is different.

An intuitive model to decide whether a tuple is true or not is one that embeds each entity $x_i \in \mathcal{E}$ into a vector $\mathbf{x_i} \in [0,1]^d$ of length d, and the relation r into a matrix $\mathbf{r} \in \mathbb{R}^{|r| \times d}$, where the i^{th} row in $\mathbf{r}$ operates over the entity at position i. Each relation r has a learnable bias term b_r as a part of the relation embedding as a relation dependent constant that does not depend on any entities and allows the model. Such a model defines the following scoring function, where σ is a nonlinear function that is differentiable almost everywhere.

$$\phi^{\circ}_{\theta}(r(x_1, \ldots, x_n)) = \sigma(b_r + \sum_{i=1}^{|r|} \sum_{k=0}^{d-1} \mathbf{x_i}[k] \times \mathbf{r}[i][k]) \tag{1}$$

Instead of performing a single version of the above scoring function, we found it beneficial to consider multiple versions of the scoring function and use the output of all of the functions to obtain the final predictive performance. Scores from multiple versions of Equation 1 are summed to produce the final score for the input tuple. The idea of using an ensemble of models (Dietterich, 2000) allows

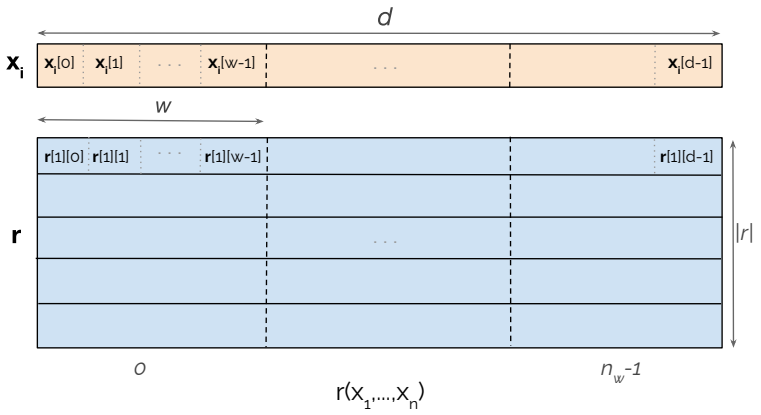

Figure 2: A schematic of the entity and relation embeddings in ReAlE: the embedding dimension d is divided into n_w windows of size w.

each of the scoring functions to focus on a different aspect of the problem. This is similar to ideas for having multiple heads in self-attention layers (Vaswani et al., 2017).

Here, we introduce the concept of *windows*, a range of indices whereby elements within the same window are used in one scoring function. The number of embedding elements for each entity in a window, the *window size*, is a hyperparameter (see Figure 2). Let w denote the window size, $n_w = \lfloor \frac{d}{w} \rfloor$ the number of windows, and b_r^j the learnable bias term of relation r for the j^{th} window, for all $j = 0, \ldots, n_w - 1$. Equation 2 defines ReAlE's score of a tuple $r(x_1, x_2, \ldots, x_n)$, where σ is a monotonically-increasing nonlinear function that is differentiable almost everywhere.

$$\phi_\theta(r(x_1, \ldots, x_n)) = \sum_{j=0}^{n_w-1} \sigma(b_r^j + \sum_{i=1}^{|r|} \sum_{k=0}^{w-1} \mathbf{x_i}[jw + k] \times \mathbf{r}[i][jw + k]) \tag{2}$$

The model presented in Equation 2 is an ensemble of n_w models of Equation 1. For instance, if embedding dimension d is 100 and window size w is 5, the model defined in Equation 2 is equivalent to creating an ensemble model of 20 variants of models in Equation 1. The aggregation of the scores of Equation 1 could be done with a summation or average, which differ by a constant. We use the sum as it is simpler.

Learning ReAlE model. To learn a ReAlE model, we use stochastic gradient descent with mini-batches. In each iteration, we take in a batch of positive tuples from the knowledge hypergraph. As a knowledge hypergraph, as defined, only has positive instances and we need to also train our model on negative instances, we follow the literature (Abboud et al., 2020; Fatemi et al., 2020) and generate negative examples by following the contrastive approach of Bordes et al. (2013). Given a knowledge hypergraph defined on τ', we let τ'_{train}, τ'_{test}, and τ'_{valid} denote the (pairwise disjoint) train, test, and validation sets, respectively, so that $\tau' = \tau'_{\text{train}} \sqcup \tau'_{\text{test}} \sqcup \tau'_{\text{valid}}$ where $\sqcup$ is disjoint set union. To build a model that completes τ', we train it using τ'_{train}, tune the hyperparameters of the model using τ'_{valid}, and evaluate its efficacy on τ'_{test}. For any tuple t in τ', we let $T_{\text{neg}}(t)$ be a function that generates a set of related negative samples. The number of negative samples per tuple is called *negative ratio* and is a hyperparameter. For a ReAlE model with parameters θ (including relation and

entity embeddings) and ϕ_θ as the function in Equation 2, we minimize the cross entropy loss:

$$\mathcal{L}(\theta, \tau'_{\text{train}}) = \sum_{t \in \tau'_{\text{train}}} -\log \frac{e^{\phi_\theta(t)}}{\sum_{t' \in \{t\} \cup T_{\text{neg}}(t)} e^{\phi_\theta(t')}}.$$

Appendix A shows a high-level description of the algorithm for learning a ReAlE model. Code and data is available at `https://github.com/baharefatemi/ReAlE`.

5. Theoretical analysis

To better understand the expressive power of ReAlE and the types of reasoning it can perform, we analyze the extent of its expressivity and its capacity to represent relational algebra operations without the operations being known or given to the learner. Relational algebra allows for quantification, in particular, statements that are true for all entities or statements that are true for at least one. The semantics of the relational algebra is defined using first-order logical statements, using the conventions of Datalog, with *variables* written in upper case, *e.g.*, $X_1, \ldots, X_n$ and particular entities in lower case, *e.g.*, $x_1, \ldots, x_n$. To make a meaningful statement, each variable is quantified using $\forall$ (the statement is true for all assignments of entities to the variable) and $\exists$ (the statement is true if there exists an assignment of an entity to the variable). Here, $\bar{x}$ is a sequence of particular entities and $\bar{X}$ is a sequence of variables. We use $\neg$ as negation, $\wedge$ as conjunction (and), and $\vee$ as disjunction (or). In relational algebra, each relation has a unique definition and there are no cyclic or recursive definitions.

We use first-order logic rules to express relational algebra operations. Both constraints and queries can be written as (a subset of) first-order logic rules. Clark's completion (Clark, 1978) provides a semantics for Prolog (and Datalog[1]) with negation-as-failure by mapping logic programs to corresponding first-order logic statements. It provides if-and-only-if statements from a set of clauses defining a relation under the assumption that the clauses cover all of the cases where the relation is true.

For any $\bar{x}$ and any relation r, we define the *relation complement* function f as $f(\phi_\theta(r(\bar{x}))) = \phi_\theta(\neg r(\bar{x}))$. This function depends on the choice of the nonlinearity σ of the scoring function in Equation 2. For example, if σ is the Sigmoid function, then $f(\phi_\theta(r(\bar{x}))) = 1 - \phi_\theta(r(\bar{x}))$; if it is the hyperbolic tangent (tanh), then $f(\phi_\theta(r(\bar{x}))) = -\phi_\theta(r(\bar{x}))$.

In this section, we state the theorems and provide proof sketches for most of the theorems and defer all the proofs to Appendix B.

5.1 Full expressivity

The two results in this section state that ReAlE is fully expressive and m-TransH, RAE, and NaLP are not fully expressive (G-MPNN scoring function and its non-expressiveness were discussed in 2).

Theorem 1 (Full Expressivity) *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$ containing λ true tuples with $\alpha = \max_{r \in \mathcal{R}}(|r|)$ as the maximum arity over all relations in $\mathcal{R}$, there is a ReAlE model with $n_w = \lambda$, $w = \alpha$, $d = \max(\alpha\lambda, \alpha)$, and $\sigma(x) = \frac{1}{1+\exp(-x)}$ that accurately separates the true tuples from the false ones.*

1. One way to write a query language is to use Datalog (without recursion), which can be seen as Prolog without function symbols. The semantics of Prolog, and so of Datalog, for the acyclic case is in terms of Clark's completion (Clark, 1978), where relations are defined using if-and-only-if ($\leftrightarrow$) formulae. We write out definitions using this formulation (as opposed to writing the clauses and assuming the completion, which gives the if-and-only-if).

Proof Sketch. To prove full expressivity of ReAlE, we show that there exists an assignment of embedding values that enables the scoring function ϕ of ReAlE to correctly separate the true tuples from the rest, for any set of entities and relations, and for any number of true tuples. Our construction sets the window size to the maximum arity and the number of windows to the number of true tuples. Each window encodes one true tuple. □

Theorem 2 *m-TransH, RAE, and NaLP are not fully expressive.*

The proof of the above theorem follows from Lemma 2.1 and Lemma 2.2 below.

Lemma 2.1 *m-TransH and RAE are not fully expressive and have restrictions on what relations these approaches can represent.*

Proof Kazemi and Poole (2018) prove that TransH Wang et al. (2014) is not fully expressive and explore its restrictions. As m-TransH reduces to TransH for binary relations, it inherits all its restrictions and is not fully expressive. RAE also follows the same strategy as m-TransH in modeling the relations. Therefore, both m-TransH and RAE are not fully expressive. ■

Lemma 2.2 *NaLP is not fully expressive.*

Proof NaLP first concatenates the embeddings of entities and the embedding of their corresponding roles in the tuples, then applies to them the following functions: 1D convolution, projection layer, minimum, and another projection layer. Looking carefully at the output of the model, the NaLP scoring function for a tuple $r(x_1,\ldots,x_n)$ is in the form of $|\mathbf{P_1 x_1} + \cdots + \mathbf{P_n x_n} + \mathbf{r}|_1$ with $\mathbf{P_i}$ as learnable diagonal matrices with some shared parameters. In the binary setup ($n = 2$), the score function of NaLP is $|\mathbf{P_1 x_1} + \mathbf{P_2 x_2} + \mathbf{r}|_1$. Kazemi and Poole (2018), however, proved that translational methods having a score function of $|\mathbf{P_1 x_1} - \alpha\mathbf{P_2 x_2} + \mathbf{r}|_i$ are not fully expressive and have severe restrictions on what relations these approaches can represent. NaLP has the same score function with $\alpha = -1$ and $i = 1$ and therefore is not fully expressive. ■

5.2 Representing relational algebra with ReAlE

Here, we describe some primitive operations and prove how closely ReAlE can represent each.

5.2.1 RENAMING

Renaming changes the order of one or more entities in a relation. A renaming operation can be written as the following logical rule, where t is defined in terms of s and π defines a permutation function.

$$\forall X_1 \ldots \forall X_n \qquad t(X_1,\ldots,X_n) \leftrightarrow s(X_{\pi(1)},\ldots,X_{\pi(n)}) \tag{3}$$

For example, $\forall\, X\, \forall\, Y\, \forall\, I$ $\mathit{bought}(X,Y,I) \leftrightarrow$ $\mathit{sold}(Y,X,I)$ represents renaming relation (person X *bought* I from person Y) into relation (person Y *sold* I to person X).

Theorem 3 (Renaming) *Given permutation function π, and relation s, there exists a parametrization for relation t in ReAlE such that for entities $x_1, \ldots, x_n$, with arbitrary embeddings,*

$$\phi_\theta(t(x_1, \ldots, x_n)) = \phi_\theta(s(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}))$$

Proof Sketch. The proof considers the formulation of the scoring function and shows (by expanding and rearranging terms) that the equality condition holds when we set the relation embedding of t to be a permutation of that of s while setting the biases to the same value. □

5.2.2 Projection

Projection takes a relation as input and removes some entities corresponding to some specific positions in the relation. A projection operation that defines t as a projection of s can be written as the following (for $m < n$).

$$\forall X_1 \ldots \forall X_m \quad t(X_1, \ldots, X_m) \leftrightarrow \exists X_{m+1} \ldots \exists X_n \quad s(X_1, \ldots, X_m, \ldots, X_n) \tag{4}$$

Note that projection can be paired with renaming to allow for arbitrary subsets and ordering of arguments. For example, $\forall X\ \forall I\ \mathit{seller}(X, I) \leftrightarrow \exists P\ \mathit{bought}(P, X, I)$.

Theorem 4 (Projection) *For any relation s on n arguments there exists a parametrization for relation t on $m < n$ arguments in ReAlE such that for any arbitrary sequence $x_1, \ldots, x_n$,*

$$\phi_\theta(t(x_1, \ldots, x_m)) \geq \phi_\theta(s(x_1, \ldots, x_n))$$

Inequality is the best we can hope for, because multiple tuples with relation s might project to the same tuple with relation t. The score of the tuple with relation t should thus be greater than or equal to the maximum score for s.

Proof Sketch. The embedding for t is the same as for the first m positions of s, and the bias for t is set to be the worst case, which is the bias of s plus the sum over the windows k, remaining positions i of the maximum of 0 and $s[i][k]$. □

5.2.3 Selection

Selection returns the subset of tuples of a relation that satisfies a given condition. Here, we consider equality conditions whereby a selection operation reduces the number of arguments, and has two forms defining t as a selection of s.

$$\forall X_1 \ldots \forall X_n \ \ t(X_1, \ldots, X_{p-1}, X_{p+1}, \ldots, X_q, \ldots, X_n) \leftrightarrow \exists X_p \ \ s(X_1, \ldots, X_n) \wedge (X_p = X_q) \tag{5}$$

$$\forall X_1 \ldots \forall X_n \ \ t(X_1, \ldots, X_{p-1}, X_{p+1}, \ldots, X_n) \leftrightarrow \exists X_p \ \ s(X_1, \ldots, X_n) \wedge (X_p = c) \quad \text{for fixed } c \tag{6}$$

For example, $\forall X\ \forall Y\ \mathit{sold\_coffee}(X, Y) \leftrightarrow \exists I\ \mathit{sold}(X, Y, I) \wedge (I = \mathit{coffee})$. Observe that selecting tuples with the condition $X_p = X_q$ for arbitrary p and q is equivalent to first renaming the tuple so that X_p is in position n and X_q is in position $n-1$; and then performing a selection with the condition $X_{n-1} = X_n$ or $X_n = c$. Thus, we show the selection operation for the cases when $X_{n-1} = X_n$ or $X_n = c$.

Theorem 5 (Selection 1) *For arbitrary relation s, there exists a parametrization for relation t in ReAlE such that for arbitrary entities $x_1, \ldots, x_n$,*

$$\phi_\theta(t(x_1, \ldots, x_{n-1})) = \phi_\theta(s(x_1, \ldots, x_{n-1}, x_{n-1}))$$

Proof Sketch. Similar to the previous proofs, we show a setting for the biases and relation embeddings for which the equality holds. In this case, the biases of the two relations are set to be equal, and the relation embeddings for t and s are set to be equal except for the last entry (at position $n-1$), which we set as $\mathbf{t}[n-1][k] = \mathbf{s}[n-1][k] + \mathbf{s}[n][k]$. □

Theorem 6 (Selection 2) *For arbitrary relation s and for a fixed constant c, there exists a parametrization for relation t in ReAlE such that for arbitrary entities $x_1, \ldots, x_n$*

$$\phi_\theta(t(x_1, \ldots, x_{n-1})) = \phi_\theta(s(x_1, \ldots, x_{n-1}, c))$$

Proof Sketch. The proof is similar to that of the previous Selection operation, but with a slightly different setting: here, relation embeddings are set to be the same, while the biases in the two relations differ by an additive factor, which is a function of the relation embeddings. □

5.2.4 SET UNION

Set union operates on relations of the same arity, and returns a new relation containing the tuples that appear in at least one of the relations. A set union operation can be written as the following logical rule, with relation t as the union of s and r.

$$\forall \bar{X} \;\; t(\bar{X}) \leftrightarrow s(\bar{X}) \vee r(\bar{X}) \tag{7}$$

For example, $\forall X_1 \; \forall X_2 \; \forall I \quad traded(X_1, X_2, I) \leftrightarrow \; sold(X_1, X_2, I) \vee bought(X_1, X_2, I)$.

For a ReAlE model to be able to represent the set union operation, first observe that any score for a tuple t that represents the union of relations r and s depends on how dependent the two relations r and s are. For example, if s is a subset of r, then the score of t is equal to that of r. But since we do not know about such dependence relations in the data, then a reasonable bound for the score of t is *at least* as high as the maximum score of either r or s, as the following lemma states.

Theorem 7 (Set Union) *For arbitrary relations s and r with the same arity, there exists a parametrization for relation t in ReAlE such that for arbitrary entity set $\bar{x}$*

$$\phi_\theta(t(\bar{x})) \geq \max(\phi_\theta(s(\bar{x})), \phi_\theta(r(\bar{x})))$$

Proof Sketch. Expanding the scoring functions on each side of the inequality, the inequality holds by setting the embedding of t to the maximum of that of s and r for each embedding position, and the bias of t to the maximum of that of s and r. □

5.2.5 Set difference

Set difference operates on relations of the same arity, and returns a new relation containing the tuples from the left relation that do not appear in the right one. The set difference operation can be written as the following logical rule, where relation t is set difference of s and r.

$$\forall \bar{X} \;\; t(\bar{X}) \leftarrow s(\bar{X}) \wedge \neg r(\bar{X}) \tag{8}$$

For example, $\forall X \, \forall Y \;\; needs\_filter(X, Y) \leftarrow bought\_coffee(X, Y) \wedge \neg bought\_filter(X, Y)$.

Similar to set union, the score of a set difference operator depends on how dependent the relations r and s are. For the same reasons, the best we can hope for in this case is to show that the score of t is smaller than that of both s and $\neg r$ (since $t(\bar{X})$ is true only when both $s(\bar{X})$ and $\neg r(\bar{X})$ are true, then the scores of the latter two must be higher). In the lemma that follows, f is the relation complement function described in the introduction of Section 5. Here, we assume that f exists for the selected σ.

Theorem 8 (Set Difference) *For arbitrary relations r and s with the same arity, if f is a linear relation complement function and $f(\sigma(x)) = \sigma(c*x)$ with c as a constant, there exists a parametrization for relation t in ReAlE such that for arbitrary entities $x_1, \ldots, x_n$*

$$\phi_\theta(t(\bar{x})) \leq \min(\phi_\theta(s(\bar{x})), f(\phi_\theta(r(\bar{x}))))$$

Proof Sketch. We define t so that for each tuple, it is true whenever s holds and r does not hold. The minimum ensures that both hold. The relation complement function was designed so that the score of the negation of r can be computed from the score of r. To show the theorem, we expand the scoring functions of r and s, then distribute the linear relation complement function f, first inside the summation, then inside the σ function (as $f(\sigma(x)) = \sigma(c \times x)$). Setting the bias and relation embedding terms for t to the minimum across that of s and $r \times c$, and minimum across the bias of s and that of $r \times c$, respectively, makes the inequality hold for all input entities. □

5.2.6 Joint representation of operations

The following theorems establish the ability of ReAlE to jointly capture the relational algebra operations discussed above. This is of interest, particularly because capturing multiple inference patterns jointly has been deemed challenging in some existing methods (*e.g.*, Abboud et al., 2020). In our case, the parametrizations do not interfere with each other, as each rule (operation) defines the relation in the head without side effects on the relations in the body.

Theorem 9 (Composition) *For an arbitrary set of relations S_r and arbitrary non-empty composition of operations S_{op} from the set renaming, projection, selection, set union, and set difference, there exists a parametrization for relation t in ReAlE with t as the resulting relation of applying S_{op} to S_r.*

Proof Sketch. The proof is by induction on the number of primitive operations in S_{op}. The induction step relies on the fact that a parametrization for the last operation is independent of the parameters of all operations that come before it in the sequence (see Theorems 3-8). □

Theorem 10 (Joint representation) *ReAlE is able to jointly represent a set of relations each being either the result of a relational algebra operation renaming, projection, selection, set difference, set union, or a composition of these operations.*

			number of tuples			number of tuples with respective arity				
Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	#train	#valid	#test	#arity=2	#arity=3	#arity=4	#arity=5	#arity=6
JF17K	29,177	327	61,911	15,822	24,915	56,322	34,550	9,509	2,230	37
FB-AUTO	3,388	8	6,778	2,255	2,180	3,786	0	215	7,212	0
M-FB15K	10,314	71	415,375	39,348	38,797	82,247	400,027	26	11,220	0

Table 1: Dataset Statistics.

Proof All the parametrizations proposed in the proofs of Theorems 3, 4, 5, 6, 7, 8, and 9 are solely based on defining the parametrization of the output relation t based on the input relation(s) without changing that of other relations (including t). Since each relation t uses its own parametrization without affecting any parameters of other relations, then all such relations can be represented concurrently. This proves the theorem. ■

6. Datasets

6.1 Real-world datasets

We use three real-world datasets for our experiments: JF17K (Wen et al., 2016), and FB-AUTO and M-FB15K (Fatemi et al., 2020). Table 1 summarizes the statistics for the datasets JF17K, FB-AUTO, and M-FB15K.

6.2 Synthetic dataset

To study and evaluate the generalization power of models in a controlled environment, we generate a synthetic dataset. This practice has become common in recent years, with the creation of several procedurally generated benchmarks to study the generalization power of models in different tasks. Examples of such datasets include CLEVR (Johnson et al., 2017) for images and TextWorld (Côté et al., 2018) for text data, and GraphLog (Sinha et al., 2020) for graph data. To create a benchmark for analyzing the relational algebraic generalization power of ReAlE for hypergraph completion, we consider the following criteria.

1. *Completeness*: The benchmarks must contain the desired relational algebra operations; in our case: renaming, projection, selection, set union, and set difference.

2. *Diversity*: The benchmark must contain a variety of relations that are the result of *repeated application* of relational algebra operations having varying depths.

3. *Compositional generalization*: The benchmark must contain relations that are the result of repeated application of operations of different types.

To synthesize a dataset that satisfies the above conditions, we extend the Erdős-Rényi model (Erdős and Rényi, 1959) for generating random graphs to directed edge-labeled hypergraphs. The Erdős-Rényi model is a random graph generation method that is widely used in simulation, whose properties are well-studied and, for our purposes, is simple to extend to directed ordered hypergraphs. We use this hypergraph generation model to first generate a given number of true tuples; we then apply the five

relational algebra operations to these tuples (repeatedly and recursively) to obtain new tuples with varying depths. In what follows, we discuss the details of our dataset generation algorithm.

6.2.1 Knowledge Hypergraphs

A *knowledge hypergraph* is a directed hypergraph $H = (V, E, R)$ with nodes (entities) V, edges (tuples) E, and edge labels (relations) R such that:

- every edge in the hypergraph consists of an *ordered* sequence of nodes,
- every edge has a label $r_i \in R$, and
- edges with the same label are defined on the same number of nodes.

Observe that in knowledge hypergraphs, edges having the same label form a uniform directed hypergraph (all edges defined on the same number of nodes). We can thus think of H as the combination of $|R|$ directed uniform hypergraphs.

6.2.2 Extending Erdős-Rényi to Knowledge Hypergraphs

In the Erdős-Rényi model, all graphs with a fixed number of nodes and edges are equally likely. Equivalently, in such a random graph, each edge is present in the graph with a fixed probability p, independent of other edges. In this section, we describe a method of generating a random *knowledge hypergraph* inspired by the Erdős-Rényi process.

Let n be the (predefined) number of nodes in the hypergraph and n_r be the number of relations. We let R be a list of relations defined in terms of arity and a probability that influences the number of tuples generated for that given relation. More formally,

$$R = \{(k_i, p_i) | k_i = arity, 0 \leq p_i \leq 1, \forall i = 0, \ldots, n_r\}$$

The expected number of edges generated for a given relation r_i is $m_i = k_i! \binom{n}{k_i} p_i$. As the process of including edges in the graph is a Binomial, we can compute this expected value by sampling from the following probability density.

$$P(m_i) = \binom{N}{m_i} p_i (1 - p_i)^{N - m_i}$$

where $N = k_i! \binom{n}{k_i}$ is the number of possible k_i-uniform (directed) edges in the hypergraph.

The running time of Algorithm 1 depends on the number of nodes n and the arity k_i of each relation. Let $k = max_{k_i \in R} k_i$. Thus the running time of Algorithm 1 is $O(|R|n^k)$.

Algorithm 1: generate_knowledge_hypergraph(V, R)

```
edge_list = []
n = len(V)
for r, (k, p) in enumerate(R) do
    N = k! (n choose k)
    m = random.binomial(N, p) {result of flipping a coin N times with probability of success p}
    edge_count = 0
    while edge_count <= m do
        edge = random.sample(V, k) {select k vertices from V at random}
        if edge not in edge_list then
            edge_list.append([r] + edge)
            edge_count = edge_count + 1
        end if
    end while
end for
return edge_list
```

6.2.3 Dataset Generation

To evaluate a model on how well it represents relational algebra operations, we generate a set of ground-truth true tuples, each of which is the result of repeated application of a primary relational algebra operation to an existing tuple (hyperedge). The operations we are interested in are renaming, projection, selection, set union, and set difference.

Algorithm 2: generate_ground_truth(V, R, *n_derived_tuples*)

```
E = generate_knowledge_hypergraph(V, R)
for i in range(n_derived_tuples) do
    op = randomly select one primary operation
    tuple = randomly select one hyperedge from E
    apply op to tuple
    add tuple to the set of edges E
end for
```

Finally, the complete algorithm to generate the train, valid, and test sets of the synthetic dataset is described in Algorithm 3 below.

Algorithm 3: synthesize_dataset(V, R, *n_derived_tuples*)

```
ground_truth = generate_ground_truth(V, R, n_derived_tuples)
relational_data = sub-sample from ground_truth
train, valid, test = randomly split relational_data into train, valid and test
```

To evaluate our model in a controlled environment, we create a synthetic dataset whose statistics are proportional to that of JF17K, as follows:

- We create a set of primitive relations by sampling (with replacement) from the relations in JF17K, and we use the arity of the sampled relation for the primitive relation.
- For each primitive relation r, we select a relation in JF17K with the same arity and populate r with the same number of tuples as the selected relation.
- We create the derived relations using Algorithm 2.
- Finally, we generate the train, valid, and test sets of REL-ER using Algorithm 3.

This procedure allows us to create a synthetic dataset close to JF17K. For instance, we observe that in JF17K, the number of tuples per relation follows a long-tailed right-skewed distribution with a minimum value of 1, maximum of 8955, mean of 313, mode of 3, and median of 31. This algorithm gives us a distribution with a minimum of 1, maximum of 3559, mean of 203, mode of 2, and median of 43. We call this synthetic dataset *REL-ER* and use it in our experiments.

7. Experimental setup

In this section, we explain the evaluation metrics used in our experiments. We defer the implementation details of our model and baselines to Appendix D. Following the literature (Abboud et al., 2020; Fatemi et al., 2020), we evaluate the link prediction performance with Mean Reciprocal Rank (MRR) and Hit@k, $k \in \{1, 3, 10\}$. Both MRR and Hit@k rely on the *ranking* of a tuple $x \in \tau'_{\text{test}}$ within a set of *corrupted* tuples. For each tuple $r(x_1, \ldots, x_n)$ in τ'_{test} and each entity position i in the tuple, we generate $|\mathcal{E}|-1$ corrupted tuples by replacing the entity x_i with each of the entities in $\mathcal{E} \setminus \{x_i\}$. For example, by corrupting entity x_i, we obtain a new tuple $r(x_1, \ldots, x_i^c, \ldots, x_n)$ where $x_i^c \in \mathcal{E} \setminus \{x_i\}$. Let the set of corrupted tuples, plus $r(x_1, \ldots, x_n)$, be denoted by $\zeta_i(r(x_1, \ldots, x_n))$ (Note that $T_{\text{neg}}(r(x_1, \ldots, x_n))$ introduced in Section 4 is a random sample of the corrupted tuples in $\zeta_1 \cup \ldots \cup \zeta_n$ for $r(x_1, \ldots, x_n)$).

Let $\text{rank}_i(r(x_1, \ldots, x_n))$ be the ranking of $r(x_1, \ldots, x_n)$ within $\zeta_i(r(x_1, \ldots, x_n))$ based on the score $\phi_\theta(x)$ for each $x \in \zeta_i(r(x_1, \ldots, x_n))$. In an ideal knowledge hypergraph completion method, $\text{rank}_i(r(x_1, \ldots, x_n))$ is 1 among all corrupted tuples $\zeta_i(r(x_i, \ldots, x_n))$. The mean reciprocal rank is $\frac{1}{N} \sum_{r(x_1,\ldots,x_n) \in \tau'_{\text{test}}} \sum_{i=1}^{n} \frac{1}{\text{rank}_i(r(x_1,\ldots,x_n))}$ where $N = \sum_{r(x_1,\ldots x_n) \in \tau'_{\text{test}}} |r|$ is the number of prediction tasks. Hit@k measures the proportion of tuples in τ'_{test} that rank among the top k in their corresponding corrupted sets. Following the literature, we remove all corrupted tuples that are in τ' from our computation of MRR and Hit@k.

8. Experiments

We organize our experiments into three groups with different objectives. The goal of the first set of experiments is to evaluate the proposed method on real datasets and compare its performance to that of existing work. Our second goal is to test the ability of ReAlE to represent the relational algebra operations. The final set of experiments is an ablation study that examines the effect of window size. More empirical comparisons and ablation studies are presented in Appendix C.

	JF17K				FB-AUTO				M-FB15K			
Model	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
m-DistMult (Fatemi et al., 2020)	.463	.372	.510	.634	.784	.745	.815	.845	.705	.633	.740	.844
m-CP (Fatemi et al., 2020)	.392	.303	.441	.560	.752	.704	.785	.837	.680	.605	.715	.828
m-TransH (Wen et al., 2016)	.444	.370	.475	.581	.728	.727	.728	.728	.623	.531	.669	.809
RAE (Zhang et al., 2018)	.310	.219	.334	.504	-	-	-	-	-	-	-	-
NaLP (Guan et al., 2019)	.366	.290	.391	.516	-	-	-	-	-	-	-	-
GETD (Liu et al., 2020)	.151	.104	.151	.258	.367	.254	.422	.601	-	-	-	-
HSimplE (Fatemi et al., 2020)	.472	.378	.520	.645	.798	.766	.821	.855	.730	.664	.763	.859
HypE (Fatemi et al., 2020)	.494	.408	.538	.656	.804	.774	.823	.856	.777	.725	.800	.881
G-MPNN (Yadati, 2020)	.501	.425	.537	.660	-	-	-	-	.779	.732	.805	.894
BoxE (Abboud et al., 2020)	.553	.467	**.596**	**.711**	.844	.814	.863	.898	.761	.702	.791	.877
ReAlE (Ours)	**.559**	**.482**	.594	.705	**.873**	**.852**	**.886**	**.909**	**.801**	**.755**	**.823**	**.901**

Table 2: Knowledge hypergraph completion results on JF17K, FB-AUTO and M-FB15K for baselines and the proposed method. Our method ReAlE outperforms (or is competitive to) the baselines on all datasets. Higher values are better for all columns. To measure the stability of ReAlE, we ran it on JF17K 10 times with the best hyperparameters and the standard deviation of MRR is 0.0004. So, the answer is correct to three significant digits which is why the error bars are not shown. The baselines did not report standard deviation.

8.1 Results on real datasets

We evaluate ReAlE on the three public datasets JF17K, FB-AUTO, and M-FB15K and compare its performance to existing models. For all variants of ReAlE, we use the *sigmoid* function as σ in Equation 2. This is because the *sigmoid* was the best among all non-linear functions tested in Section C.2.

ReAlE is comparable to BoxE on JF17K. On FB-AUTO and M-FB15K, ReAlE outperforms all existing models. It is interesting to note that M-FB15K is the largest of all three datasets, and that most of the tuples in both FB-AUTO and M-FB15K are beyond-binary relations, in contrast to JF17K, in which more than half the tuples (54%) are binary (See Table 1 for more details). Results are summarized in Table 2. Overall, ReAlE is competitive on all benchmarks and is state-of-the-art on FB-AUTO and M-FB15K. Among the metrics reported, Hit@1 is a notion of precision and Hit@10 is a notion of recall; ReAlE outperforms all the baselines in precision (Hit@1).

8.2 Results on synthetic datasets

We also evaluate our model on REL-ER dataset created in Section 6.2.3. We break down the performance of our model based on the type of relational algebra operation and its depth. As a first step of the synthetic dataset generation, we create a list of tuples at random. The relations involved in these tuples are called *elementary* relations, as they are not generated based on any relational algebra operation. We then create a set of tuples that are based on *composite* relations: relations that are built from previously defined elementary and composite relations. We let the *type* of a composite relation be the last operation applied. We decompose our results by type in Table 3.

Note that the choice of the *last* operation for defining the type does not change the overall performance of the model; it merely helps us break down the performance to gain insight. As there is no obvious way of defining the type of a composite relation, we experimented with letting it be

		Operation type					Depth			
Model	All	elementary	renaming	projection	set union	set difference	1	2	3	4
m-TransH (Wen et al., 2016)	.387	.166	.447	.652	.459	.464	.531	.499	.352	.03
HypE (Fatemi et al., 2020)	.689	.335	.877	.856	.888	.894	.881	.872	.897	**.976**
BoxE (Abboud et al., 2020)	.695	.327	**.916**	.852	.900	.918	.885	.904	.908	**.976**
ReAlE (Ours)	**.709**	**.336**	.882	**.877**	**.932**	**.923**	**.887**	**.950**	**.945**	.938
#test tuples	5378	1833	458	762	1676	649	2075	1204	245	21

Table 3: Breakdown performance of MRR across composite relations (based on a sequence of primitive operations) and of varying depth on the REL-ER dataset along with their statistics. Higher values are better.

determined by the first, last, or all operations; our experiments showed little variation between the results.

We define the *depth* of a relation recursively. An elementary relation has a depth of 0. A relation that is the result of a unary operation (*e.g.*, projection or renaming) has a depth of one plus the depth of the input relation. For a relation that is the result of a binary operation (*e.g.*, set union or set difference), the depth is one plus the maximum depth of the input relations. Note that there are only 21 test tuples for depth 4. The results of our experiments on REL-ER are summarized in Table 3 and show that our proposed model outperforms the state-of-the-art in almost all cases. In the case of renaming, BoxE is able to better represent the operation, most probably due to its ability to model inversion. As the decomposed performance shows, the improvement to the general result is due mostly to improvements in the performance of the operations as well as improvements for elementary relations. These results confirm that our theoretical findings are in line with the practical results. We do not have tuples for the selection operation in the test set because, by nature, selection generates very few tuples; and as the split of train/test/validation in REL-ER is done at random, the chance of having selection tuples in the test set is very low and did not occur when we synthesized the dataset (selection tuples are still present in the train data).

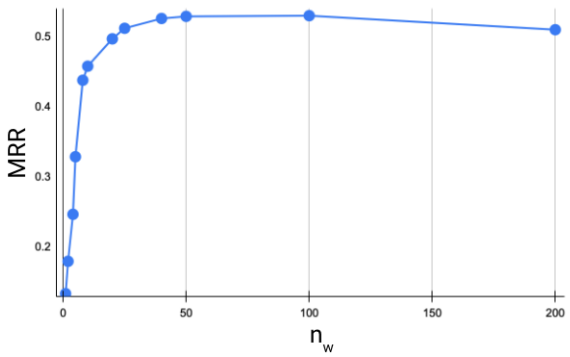

Figure 3: MRR of ReAlE for different number of windows n_w on JF17K with fixed values for negative ratio, batch size, and embedding dimension of 10, 128, and 200 respectively. The total number of parameters for each value of n_w is fixed.

8.3 Ablation study on varying number of windows

Here, we compare the performance of ReAlE with different numbers of windows n_w. The negative ratio, batch size, and embedding dimension are fixed to 10, 128, and 200 respectively for this experiment. Here, the total number of parameters for each value of n_w is fixed. The outcome of the

study is summarized in Figure 3. Number of windows n_w of 1 is the same as Equation 1. Increasing n_w means having more models in the ensemble learner. As we increase the number of learners, the performance (MRR) on the test set increases and it plateaus when having a large number of windows. This result confirms the importance of having multiple learners as an ensemble model. Therefore, window size or number of windows is a sensitive hyperparameter that needs to be tuned for the best performance.

9. Conclusion and future work

In this work, we introduce ReAlE for reasoning in knowledge hypergraphs. To design a powerful method, we build on the primitives of relational algebra, which is the calculus of relational models. We prove that ReAlE can represent the primitive relational algebra operations renaming, projection, selection, set union, and set difference, and is fully expressive. The results of our experiments on real and synthetic datasets are consistent with the theoretical findings. Like some other proposals (*e.g.*, BoxE), ReAlE does not represent Cartesian product and modeling this operation remains an open problem. As the join operation is a Cartesian product followed by a projection, the model is not able to represent join. This paper is not the end of the story, but provides a solution to one part of the puzzle while providing a building block for more sophisticated models that can also deal with richer reasoning patterns.

Appendix A. Learning ReAlE Model

For completeness, we restate the scoring function and the cross-entropy loss used to train the model. In what follows, we use lower case $x_1, \ldots, x_n$ to denote particular entities, $\bar{x}$ a sequence of particular entities, and $\mathbf{x}$ the embedding of x. Recall that our model embeds each entity x_i into a vector $\mathbf{x_i} \in [0,1]^d$ of length d, and each relation r into a matrix $\mathbf{r} \in \mathbb{R}^{|r| \times d}$. Recall that w denotes the window size, $n_w = \lfloor \frac{d}{w} \rfloor$ the number of windows, and b_r^j the bias term of relation r for the j^{th} window, for all $j = 0, \ldots, n_w - 1$. Equation 1 defines ReAlE's score of a tuple $r(x_1, x_2, \ldots, x_n)$, where σ is a monotonically-increasing nonlinear function that is differentiable almost everywhere and θ is the set of all entity and relation embeddings.

$$\phi_\theta(r(x_1, \ldots, x_n)) = \sum_{j=0}^{n_w - 1} \sigma \left(b_r^j + \sum_{i=1}^{|r|} \sum_{k=0}^{w-1} \mathbf{x_i}[jw + k] \times \mathbf{r}[i][jw + k] \right) \tag{1}$$

We minimize the following cross-entropy loss:

$$\mathcal{L}(\theta, \tau'_{\text{train}}) = \sum_{t \in \tau'_{\text{train}}} -\log \left(\frac{e^{\phi_\theta(t)}}{\sum_{t' \in \{t\} \cup T_{\text{neg}}(t)} e^{\phi_\theta(t')}} \right) \tag{2}$$

Here, θ represents parameters of the model including relation and entity embeddings, and ϕ_θ is the function given by Equation 1 that maps a tuple to a score using parameters θ. Algorithm 4 shows a high-level description of how we train a ReAlE model.

Algorithm 4: Learning ReAlE

Input: Tuples τ'_{train}, loss function $\mathcal{L}$, scoring function ϕ_θ
Output: Embeddings $\mathbf{x}$ and $\mathbf{r}$ for all entities and relations in τ'_{train}.
Initialize $\mathbf{x}$ and $\mathbf{r}$ (at random)
for *every batch* τ'_{batch} *of tuples in* τ'_{train} **do**
  for *tuple* t *in* τ'_{batch} **do**
    Generate negative tuples $T_{neg}(t)$
    for $t' \in \{t\} \cup T_{neg}(t)$ **do**
      Compute $\phi_\theta(t')$ (Equation 1)
    end for
  end for
  Compute the loss $\mathcal{L}(\theta, \tau'_{\text{batch}})$ (Equation 2)
  Compute the gradient of loss with respect to $\mathbf{x}$ and $\mathbf{r}$
  Update embeddings $\mathbf{x}$ and $\mathbf{r}$ through back-propagation
end for

Appendix B. Theoretical Analysis

The current section groups the theoretical analysis of our work into four parts. In particular, Section B.1 proves that ReAlE is fully expressive. Section B.2 shows how closely ReAlE can represent relational algebra operations. Section B.3 further proves that HypE cannot represent all relational algebra operations (in particular, we show that HypE cannot represent selection). For completeness, we restate the theorems.

B.1 Full Expressivity of ReAlE

The following result proves that there exists a setting of the parameters for which ReAlE can separate true and false tuples for arbitrary input. In particular, we show it for the case where σ is the sigmoid function.

Theorem 1 (Full Expressivity) *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$ containing λ true tuples with $\alpha = \max_{r\in\mathcal{R}}(|r|)$ as the maximum arity over all relations in $\mathcal{R}$, there is a ReAlE model with $n_w = \lambda$, $w = \alpha$, $d = \max(\alpha\lambda, \alpha)$, and $\sigma(x) = \frac{1}{1+\exp(-x)}$ that accurately separates the true tuples from the false ones.*

Proof Let $T = \{\tau_0, \tau_1, \ldots, \tau_{\lambda-1}\}$ be all the true tuples defined over $\mathcal{E}$ and $\mathcal{R}$. To prove the theorem, we show an assignment of embedding values for each of the entities and relations in T such that the scoring function of ReAlE is as follows.

$$\phi(\tau)\begin{cases}\geq 1-\epsilon & \text{if } \tau \in T\\ < \lambda\epsilon & \text{otherwise}\end{cases}$$

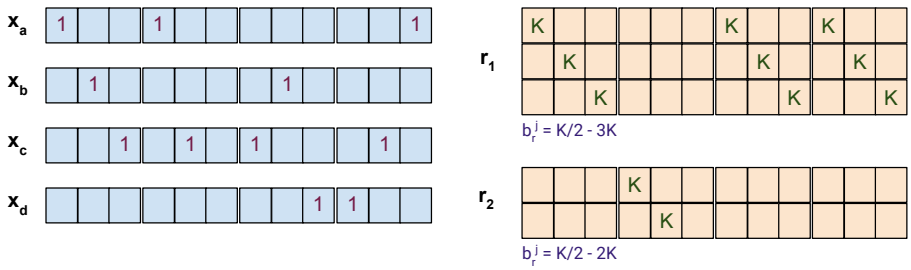

Figure 4: An example of a ReAlE embedding assignments for true tuples $\tau_0 = r_1(x_a, x_b, x_c)$, $\tau_1 = r_2(x_a, x_c)$, $\tau_2 = r_1(x_c, x_b, x_d)$ and $\tau_3 = r_1(x_d, x_c, x_a)$. Here, the number of true tuples $\lambda = n_w = 4$, maximum arity $\omega = 3$. Cells that are set to zero are left empty for better readability. As an example, given that x_b is in position 1 of τ_2, we set cell $2 \times \omega + 1 = 7$ of x_b to 1. We further set the value of r_1 at positions [1][7] to K.

Here, ϵ is an arbitrary small value such that $\epsilon < \frac{1}{2+\lambda^2}$. Observe that λ (the number of true tuples) is positive. So, for any value of λ, $\lambda\epsilon$ and $1 - \epsilon$ never meet (As $\epsilon < \frac{1}{2+\lambda^2}$ and also $\epsilon < \frac{1}{1+\lambda}$, therefore $\lambda\epsilon < 1 - \epsilon$).

We first consider the case where $\lambda > 0$. Let $K = 2 \times \sigma^{-1}(1 - \epsilon)$. Then, $\sigma(\frac{K}{2}) = 1 - \epsilon$ and $\sigma(\frac{-K}{2}) = \epsilon$.

We begin the proof by first describing an assignment of the embeddings of each of the entities and relations in ReAlE; we then proceed to show that with such an embedding, ReAlE accurately separates the true tuples from the false ones.

We consider the embeddings of entities to be $n_w = \lambda$ blocks of size w each, such that each block i is conceptually associated with true tuple τ_i for all $0 \leq i < \lambda$. Then, for a given entity x_m at position m of tuple τ_i, we set the value $\mathbf{x_m}[\text{iw} + \text{m}]$ to 1, for all $0 \leq i < \lambda$. All other values in $\mathbf{x_m}$ are set to zero. For the relation embeddings, first, recall that these embeddings are matrices of dimension $|r| \times w\lambda$, and we consider each of the $|r|$ rows of this matrix to be λ blocks of size w, where $|r| > 1$ is the arity of the given relation. If a given relation r appears in true tuple τ_i, we set the $|r|$ values at position $[iw + k][iw + k]$ to K, for all $0 \leq k < |r|$ and $0 \leq i < \lambda$; all other values in the relation embedding are set to zero. Finally, we set all the bias terms in our model to $b_r^j = \frac{K}{2} - |r| \times K$, for all $0 \leq j < \lambda$. As an example, consider Figure 4, in which the first, third and fourth tuples τ_0, τ_1 and τ_3 are defined on relation r_1; thus the embedding of r_1 has K in the diagonal of the first, third and fourth 3×3 blocks. The tuple τ_0 has entity x_b at its second position; hence the embedding of x_b has a 1 in its second position. We claim that with such an assignment, the score of tuples that are true is $\geq 1 - \epsilon$ and $< \lambda\epsilon$ otherwise.

To see why this assignment works, firs, observe that our scoring function ϕ_θ is a summation of λ sigmoids; each sigmoid is defined on an embedding block where we sum the bias term with the sum of pairwise product between the entity and relation embeddings of the given block.

Let $\tau_p = r(x_1, \ldots, x_m)$ be a true tuple and observe the embeddings of its entities and relations. The blocks at position p of the embeddings of each of $x_1, \ldots, x_m$ contain exactly one value 1 each (with the rest being zero); and the block at position p of the relation embedding for r contains Ks in the diagonal and zeros elsewhere. With such an assignment, the block at position p will contribute

the following sigmoid to the scoring function.

$$\sigma\left(b_r^p + |r|K\right) = \sigma\left(\frac{K}{2} - (|r|K) + (|r|K)\right) = \sigma\left(\frac{K}{2}\right) = 1 - \epsilon \tag{3}$$

All other blocks $q \neq p$ in τ_p will contribute to the scoring function in one of two ways, depending on whether or not r is a relation in τ_q.

If the relation r of τ_p is also the relation in τ_q (e.g. r_1 is a relation in τ_0, τ_2 and τ_3 in Figure 4), then when we multiply the $|r|$ w-sized blocks at position q of the relation embedding to their corresponding entity blocks at position q (the inner double-summation in the scoring function 2), there will be at most $c < |r|$ instances where K (in the relation embedding) is multiplied by 1 (in the entity embedding). This is because otherwise the fact at position q is the same as the fact at position p in T, indicating there is a duplicate in our set of facts. More precisely, for all blocks $q \neq p$ in τ_p such that r is the relation of τ_q, the block at position p will contribute the following sigmoid to the scoring function.

$$\begin{aligned}\sigma\left(b_r^q + cK\right) &= \sigma\left(\frac{K}{2} - |r|K + cK\right) \\ &= \sigma\left(\frac{K}{2} - (r-c)K\right) < \epsilon\end{aligned} \tag{4}$$

If r is not a relation in τ_q, then the block at position q of the relation embedding is all zeros. In this case,

$$\sigma\left(b_r^q + 0\right) = \sigma\left(\frac{K}{2} - (|r|K)\right) < \epsilon \tag{5}$$

In the end, the score of any true tuple $\phi_\theta(\tau_p)$ will be the sum of λ sigmoids such that exactly one of these sigmoids (block at position p) has a value $1 - \epsilon$, while the other $\lambda - 1$ sigmoids are $< \epsilon$. The score of a true tuple τ_p can thus be bounded as follows.

$$\phi_\theta(\tau_p) >= 1 - \epsilon \tag{6}$$

In the case of a false tuple, all the sigmoids will yield values $< \epsilon$ (as they will be of the forms in either of Equations 4, or 5). The score of a false tuple τ_f can thus be bounded as follows.

$$\phi_\theta(\tau_f) < \lambda\epsilon \tag{7}$$

To complete the proof, we consider the case when $\lambda = 0$. In this case, we let the entity and relation embeddings be blocks of arbitrary size and set all values to zero. We set the bias terms as before. Then, the score of any tuple will be $\phi_\theta(\tau_f) < \epsilon$ (Equation 7), which is what we want. This completes the proof. ■

B.2 Representing Relational Algebra with ReAlE

Theorem 3 (Renaming) *Given permutation function π, and relation s, there exists a parametrization for relation t in ReAlE such that for entities $x_1, \ldots, x_n$, with arbitrary embeddings*

$$\phi_\theta(t(x_1, \ldots, x_n)) = \phi_\theta(s(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}))$$

Proof To prove the above statement, we show that given entity embeddings $\mathbf{x_1}, \ldots, \mathbf{x_n}$ and an embedding for s, there exists a parametrization for t that satisfies the above equality.

We claim that the following settings for t are enough to show the theorem.

1. $\mathbf{t}[\pi(i)][k] = \mathbf{s}[i][k] \quad \forall\ 1 \leq i \leq n,\ \forall\ 0 \leq k < d$, and
2. $b_t^j = b_s^j \quad \forall\ 0 \leq j < n_w$

To see why, we simply expand the score function of s and replace the values for s by that of t as described, to obtain the score for t.

$$\begin{aligned}
\phi_\theta(s(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})) &= \sum_{j=0}^{n_w-1} \sigma(b_s^j + \sum_{i=1}^{|s|} \sum_{k=0}^{w-1} \mathbf{x}_{\pi(i)}[j \times w + k] \times \mathbf{s}[i][j \times w + k]) \\
&= \sum_{j=0}^{n_w-1} \sigma(b_t^j + \sum_{i=1}^{|s|} \sum_{k=0}^{w-1} \mathbf{x}_{\pi(i)}[j \times w + k] \times \mathbf{t}[\pi(i)][j \times w + k]) \\
&= \sum_{j=0}^{n_w-1} \sigma(b_t^j + \sum_{i=1}^{|t|} \sum_{k=0}^{w-1} \mathbf{x}_i[j \times w + k] \times \mathbf{t}[i][j \times w + k]) \\
&= \phi_\theta(t(x_1, x_2, \ldots, x_n))
\end{aligned}$$

■

Theorem 4 (Projection) *For any relation s on n arguments there exists a parametrization for relation t on $m < n$ arguments in ReAlE such that for any arbitrary sequence $x_1, \ldots, x_n$*

$$\phi_\theta(t(x_1, \ldots, x_m)) \geq \phi_\theta(s(x_1, \ldots, x_n))$$

Proof To prove the above statement, we first expand the score function of each side of the inequality.

$$\phi_\theta(t(x_1, \ldots, x_m)) = \sum_{j=0}^{n_w-1} \sigma(b_t^j + \sum_{i=1}^{m} \sum_{k=0}^{w-1} \mathbf{x}_i[j \times w + k] \times \mathbf{t}[i][j \times w + k]) \tag{8}$$

$$\begin{aligned}
\phi_\theta(s(x_1, \ldots, x_n)) = \sum_{j=0}^{n_w-1} \sigma(b_s^j &+ \sum_{i=1}^{m} \sum_{k=0}^{w-1} \mathbf{x}_i[j \times w + k] \times \mathbf{s}[i][j \times w + k] \\
&+ \sum_{i=m+1}^{n} \sum_{k=0}^{w-1} \mathbf{x}_i[j \times w + k] \times \mathbf{s}[i][j \times w + k])
\end{aligned} \tag{9}$$

Because each $\mathbf{x}_i[\cdot] \leq 1$, the theorem holds with the following assignments:

1. $\mathbf{t}[i][k] = \mathbf{s}[i][k] \quad \forall\ 1 \leq i \leq m,\ \forall\ 0 \leq k < d$, and
2. $b_t^j = b_s^j + \sum_{i=m+1}^{n} \sum_{k=0}^{w-1} \max(0, \mathbf{s}[i][k]) \quad \forall\ 0 \leq j < n_w$

■

Theorem 5 (Selection 1) *For arbitrary relation s, there exists a parametrization for relation t in ReAlE such that for arbitrary entities* $x_1, \ldots, x_{n-1}$

$$\phi_\theta(t(x_1, \ldots, x_{n-1})) = \phi_\theta(s(x_1, \ldots, x_{n-1}, x_{n-1}))$$

Proof To prove the above statement, we first expand the score function of ϕ_θ for the right side of the equality.

$$\begin{aligned}
&\phi_\theta(s(x_1, \ldots, x_{n-1}, x_{n-1})) \\
&= \sum_{j=0}^{n_w-1} \sigma(b_s^j + \sum_{i=1}^{n-2} \sum_{k=0}^{w-1} \mathbf{x}_i[j \times w + k] \times \mathbf{s}[i][j \times w + k] \\
&\quad + \sum_{k=0}^{w-1} \mathbf{x}_{n-1}[j \times w + k] \times \mathbf{s}[n-1][j \times w + k] \\
&\quad + \sum_{k=0}^{w-1} \mathbf{x}_{n-1}[j \times w + k] \times \mathbf{s}[n][j \times w + k]) \qquad (10) \\
&= \text{(grouping terms)} \\
&\quad \sum_{j=0}^{n_w-1} \sigma(b_s^j + \sum_{i=1}^{n-2} \sum_{k=0}^{w-1} \mathbf{x}_i[j \times w + k] \times \mathbf{s}[i][j \times w + k] \\
&\quad + \sum_{k=0}^{w-1} \mathbf{x}_{n-1}[j \times w + k] \times (\mathbf{s}[n-1][j \times w + k] + \mathbf{s}[n][j \times w + k]))
\end{aligned}$$

For the lemma to hold, the above score has to be equal to that of t, which is described as follows.

$$\begin{aligned}
\phi_\theta(t(x_1, \ldots, x_{n-1})) = &\sum_{j=0}^{n_w-1} \sigma(b_t^j + \sum_{i=1}^{n-2} \sum_{k=0}^{w-1} \mathbf{x}_i[j \times w + k] \times \mathbf{t}[i][j \times w + k] \\
&+ \sum_{k=0}^{w-1} \mathbf{x}_{n-1}[j \times w + k] \times \mathbf{t}[n-1][j \times w + k])
\end{aligned} \qquad (11)$$

The scores in Equations 10 and 11 are equal when the embedding and bias of t are set as follows.

1. $\mathbf{t}[i][k] = \mathbf{s}[i][k] \quad \forall\ 1 \leq i \leq n-2,\ \forall\ 0 \leq k < d$, and
2. $\mathbf{t}[n-1][k] = \mathbf{s}[n-1][k] + \mathbf{s}[n][k] \quad \forall\ 0 \leq k < d$, and
3. $b_t^j = b_s^j \quad \forall\ 0 \leq j < n_w$

■

Theorem 6 (Selection 2) *For arbitrary relation s and for a fixed constant c, there exists a parametrization for relation t in ReAlE such that for arbitrary entities* $x_1, \ldots, x_{n-1}$

$$\phi_\theta(t(x_1, \ldots, x_{n-1})) = \phi_\theta(s(x_1, \ldots, x_{n-1}, c))$$

Proof Using similar score expansions as in the proof of Lemma 5, we can rewrite the scores of the two relations as follows.

$$\begin{aligned}
&\phi_\theta(s(x_1, \ldots, x_{n-1}, c)) \\
&= \sum_{j=0}^{n_w-1} \sigma(b_s^j + \sum_{i=1}^{n-2} \sum_{k=0}^{w-1} \mathbf{x}_i[j \times w + k] \times \mathbf{s}[i][j \times w + k] \\
&+ \sum_{k=0}^{w-1} \mathbf{x}_{n-1}[j \times w + k] \times \mathbf{s}[n-1][j \times w + k] \\
&+ \sum_{k=0}^{w-1} \mathbf{c}[j \times w + k] \times \mathbf{s}[n][j \times w + k])
\end{aligned} \tag{12}$$

$$\begin{aligned}
\phi_\theta(t(x_1, \ldots, x_{n-1})) = \sum_{j=0}^{n_w-1} \sigma(b_t^j + \sum_{i=1}^{n-2} \sum_{k=0}^{w-1} \mathbf{x}_i[j \times w + k] \times \mathbf{t}[i][j \times w + k] \\
+ \sum_{k=0}^{w-1} \mathbf{x}_{n-1}[j \times w + k] \times \mathbf{t}[n-1][j \times w + k])
\end{aligned} \tag{13}$$

The scores in Equations 12 and 13 are equal when the embedding and bias of t are as set follows.

1. $\mathbf{t}[i][k] = \mathbf{s}[i][k] \quad \forall\ 1 \leq i \leq n-1,\ \forall\ 0 \leq k < d$, and
2. $b_t^j = b_s^j + \sum_{k=0}^{w-1} \mathbf{s}[n][j \times w + k] \times \mathbf{c}[j \times w + k] \quad \forall\ 0 \leq j < n_w$

■

Theorem 7 (Set Union) *For arbitrary relations s and r with the same arity, there exists a parametrization for relation t in ReAlE such that for arbitrary entity set* $\bar{x}$

$$\phi_\theta(t(\bar{x})) \geq \max(\phi_\theta(s(\bar{x})), \phi_\theta(r(\bar{x})))$$

Proof Given that ReAlE embeds entities in non-negative vectors and examining the scoring functions of each of the relations t, r, s, it can be observed that the above inequality holds by setting the following values.

1. $\mathbf{t}[i][k] = \max(\mathbf{s}[i][k], \mathbf{r}[i][k]) \quad \forall\ 1 \leq i < |\bar{x}|$ and $0 \leq k < d$.
2. $b_t^j = \max(b_s^j, b_r^j) \quad \forall\ 0 \leq j < n_w$.

■

In the lemma that follows, recall that the *relation complement* function (if it exists) is some linear function $f(\phi_\theta(r(\bar{x}))) = \phi_\theta(\neg r(\bar{x}))$ for arbitrary relation r and entities $\bar{x}$ that also has the form $f(\sigma(x)) = \sigma(c \times x)$ with c as a constant. For instance, when σ is sigmoid, $f(x) = 1 - x$ and $c = -1$ ($f(\sigma(x)) = 1 - \sigma(x) = \sigma(-x)$) and when σ is tanh, $f(x) = -x$ and $c = -1$ ($f(\sigma(x)) = -\sigma(x) = \sigma(-x)$).

Theorem 8 (Set Difference) *For arbitrary relations r and s with the same arity, if f is a linear relation complement function and $f(\sigma(x)) = \sigma(c \times x)$ with c as a constant, there exists a parametrization for relation t in ReAlE such that for arbitrary entities $x_1, \ldots, x_n$*

$$\phi_\theta(t(\bar{x})) \leq \min(\phi_\theta(s(\bar{x})), f(\phi_\theta(r(\bar{x}))))$$

Proof

As f is the relation complement, we have:

$$\phi_\theta(\neg r(x_1, \ldots, x_n)) = f(\phi_\theta(r(x_1, \ldots, x_n)))$$

As f is linear, we can distribute it inside the summation as follows:

$$f(\phi_\theta(r(x_1, \ldots, x_n))) = \sum_{j=0}^{n_w - 1} f(\sigma(b_r^j + \sum_{i=1}^{|r|} \sum_{k=0}^{w-1} \mathbf{x_i}[jw + k] \times \mathbf{r}[i][jw + k]))$$

Now, as $f(\sigma(x)) = \sigma(c \times x)$, we can distribute it inside the σ as follows:

$$f(\phi_\theta(r(x_1, \ldots, x_n))) = \sum_{j=0}^{n_w - 1} \sigma(b_r^j \times c + \sum_{i=1}^{|r|} \sum_{k=0}^{w-1} \mathbf{x_i}[jw + k] \times \mathbf{r}[i][jw + k] \times c)$$

Therefore, for the above inequality to hold, the bias terms and embedding values of t must be at most that of each of s and the complement of the score of r. Examining the scoring functions of each of the relations t, r, s, it can be observed that the lemma holds when the following is set.

1. $\mathbf{t}[i][k] = \min(\mathbf{s}[i][k], \mathbf{r}[i][k] \times c) \quad \forall \ 1 \leq i < |\bar{x}|$ and $0 \leq k < d$.
2. $b_t^j = \min(b_s^j, b_r^j \times c) \quad \forall \ 0 \leq j < n_w$.

■

The following theorem establishes the ability of ReAlE to jointly capture the relational algebra operations discussed above. This is of interest, particularly as Abboud et al. (2020) claimed "Capturing multiple inference patterns jointly is significantly more challenging [than capturing them singly]." Here the parametrizations do not interfere with each other; each relation has its own parametrization.

Theorem 9 (Composition) *For arbitrary set of relations S_r and arbitrary non-empty composition of operations S_{op} from the set renaming, projection, selection, set union, and set difference, there exists a parametrization for relation t in ReAlE with t as the resulting relation of applying S_{op} to S_r.*

Proof Given any operation in S_{op} composed of k primitive operations, we prove the theorem by induction on k for arbitrary k.

Base case: For one operation ($k = 1$), according to Theorems 3, 4, 5, 6, 7, and 8, ReAlE is able to represent single operations for arbitrary relation(s) as input to the operation. So the base case holds.

Induction step (operation k to k + 1): Assume a parametrization exists for all compositions of length k. Now consider an operation composed of $k + 1$ operations from S_{op}. If we ignore the last operation in the sequence, we know by the induction hypothesis that there exists a parametrization for a relation that represents the composition of the first k operations. Adding the $(k + 1)^{\text{st}}$ relation back to the sequence, the preceding theorems (Theorems 3-8) show that we can define a parametrization for this last operation based on that of the previous that is independent of the parameters of the first k operations. This completes the proof. ■

B.3 Representing Relational Algebra with HypE

So far, we showed that ReAlE is fully expressive and can represent the relational algebra operations renaming, projection, selection, set union, and set difference. We also showed that most other models for knowledge hypergraph completion are not fully expressive. In this section, we show that even as HypE (Fatemi et al. (2020)) is fully expressive in general, it cannot represent some relational algebra operations (namely, selection) while at the same time retaining its full expressivity. In special cases, such as when all tuples are false, it obviously can, however, we show that in the general case it cannot.

We proceed by first showing in Theorem 11 that HypE cannot represent selection for *arbitrary* entity and relation embeddings while retaining full expressivity. Now, one might think that even if representing selection is not possible for all embeddings, there might be some embedding space for which HypE would be able to represent selection. In Theorem 12 we show that when the embedding size is less than the number of entities (as it usually is the case), then there is no setting for which HypE would be able to represent selection.

HypE embeds an entity x_i and a relation r in vectors $\mathbf{x_i} \in \mathbb{R}^d$ and $\mathbf{r} \in \mathbb{R}^d$ respectively, where d is the embedding size. In what follows, we let $f(x, p)$ be a function that computes the convolution of the embedding of entity x with the corresponding convolution filters associated to position p and outputs a vector (see Fatemi et al. (2020) for more details). We let ϕ_θ^H to be the HypE scoring function.

Theorem 11 (HypE Selection 1 (a)) *For arbitrary relation s and arbitrary embeddings for entities $x_1, \ldots, x_{n-1}$, there exists **no** parametrization for relation t in HypE such that*

$$\phi_\theta^H(t(x_1, \ldots, x_{n-1})) = \phi_\theta^H(s(x_1, \ldots, x_{n-1}, x_{n-1})) \tag{14}$$

Proof

To see why there is no parametrization for t that satisfies Equation 14, we first expand the left and right hand side of the equality as follows.

$$\phi_\theta^H(t(x_1, \ldots, x_{n-1})) = \phi_\theta^H(s(x_1, \ldots, x_{n-1}, x_{n-1}))$$

$$\sum_{i=1}^{d} \mathbf{t}[i] \times f(x_1, 1)[i] \times \ldots \times f(x_{n-1}, n-1)[i] =$$
$$\sum_{i=1}^{d} \mathbf{s}[i] \times f(x_1, 1)[i] \times \ldots \times f(x_{n-1}, n-1)[i] \times f(x_{n-1}, n)[i]$$

For this equation to hold, the following should hold for arbitrary entities $x_1, \ldots, x_{n-1}$, and $\forall\ 1 \leq i \leq d$.

$$or \begin{cases} f(x_1, 1)[i] \times \ldots \times f(x_{n-1}, n-1)[i] = 0 \\ \mathbf{t}[i] = \mathbf{s}[i] \times f(x_{n-1}, n)[i] \end{cases} \tag{15}$$

For an entity x at position p in a tuple, the output of function $f(x,p)$ depends on the embedding of entity x and the convolution filters associated with position p. Note that these convolution filters are shared among all relations in the knowledge hypergraph and are not specific to relation s or t.

As we want Equation 15 to hold for arbitrary entity embeddings, it is easy to see that there exists at least one setup for convolution filters and entity embeddings for which none of the factors in the product $f(x_i, 1)[i] \times \ldots \times f(x_{n-1}, n-1)[i]$ is zero.

Now consider such an embedding setting. The only way Equation 15 is satisfied is when we set $\mathbf{t}[i] = \mathbf{s}[i] \times f(x_{n-1}, n)[i]$ for all $1 \leq i \leq d$. Observe that by setting the embedding of relation t to the embedding of s times $f(x_{n-1}, n)$ for a particular entity x_{n-1}, we have effectively set it to a fixed value. Now consider an entity x_k such that $f(x_k, n) \neq f(x_{n-1}, n)$, and apply the selection t to x_k by replacing x_{n-1} with x_k in Equation 14. The equality condition in the equation will not hold, because none of the conditions in Equation 15 hold. Therefore, in this setup, t fails to represent s for arbitrary entity embeddings.

Given that relation t should represent selection of s for all the entities in the hypergraph, setting it to a fixed value will make it incapable of representing selection for arbitrary entities. We can thus conclude that there is no parametrization for HypE such that it can represent selection for arbitrary embeddings of relations and entities. ■

We now show that when the embedding size is less than the number of entities $|\mathcal{E}|$ in the hypergraph, there is no setting for which HypE would be able to represent selection without losing full expressivity.

Theorem 12 (HypE Selection 1 (b)) *For arbitrary relation s, there exists **no** parametrization for relation t in HypE having scoring function ϕ^H_θ such that for arbitrary entities $x_1, \ldots, x_n$ and embedding dimension $d < |\mathcal{E}|$, where $|\mathcal{E}|$ is the number of entities in the knowledge hypergraph, HypE remains fully expressive and*

$$\phi^H_\theta(t(x_1, \ldots, x_{n-1})) = \phi^H_\theta(s(x_1, \ldots, x_{n-1}, x_{n-1}))$$

Proof Sketch. We expand the scoring functions on both sides of the equality, and show that the only way for the equality to hold is to set the embedding of t to a value that is a function of entity x_{n-1}. This effectively sets the embedding of t to a fixed value, making t incapable of representing selection for arbitrary entities. □

Proof

Consider $|s|= 2$ and $|t|= 1$ and tuples $t(x_j)$ and $s(x_j, x_j)$ such that relation t is a selection of s. We will show that when the embedding dimension d of the entities is smaller than the number of entities

in the knowledge hypergraph, there is no parametrization of HypE that gets t to represent s while retaining full expressivity.

For the sake of contradiction, assume that the lemma statement is false. Then there exists a parametrization for t such that HypE is fully expressive and that

$$\phi_\theta^H(t(x_j)) = \phi_\theta^H(s(x_j, x_j))$$

Expanding the score function for s and t we get

$$\sum_{i=1}^{d} \mathbf{t}[i] \times f(x_j, 1)[i] = \sum_{i=1}^{d} \mathbf{s}[i] \times f(x_j, 1)[i] \times f(x_j, 2)[i]$$

$$\Rightarrow \sum_{i=1}^{d} f(x_j, 1)[i] \times \big(\mathbf{t}[i] - \mathbf{s}[i] \times f(x_j, 2)[i]\big) = 0$$

For this equation to hold for arbitrary entities $x_1, x_2, \ldots, x_n$, it should hold for all $1 \leq i \leq d$ and all $1 \leq j \leq |\mathcal{E}|$ for which $\mathbf{s}(x_j, x_j)$ is true. Assuming that $d < |\mathcal{E}|$, we need to have the following hold for all $1 \leq j \leq |\mathcal{E}|$ and $1 \leq i \leq d$.

$$or \begin{cases} f(x_j, 1)[i] = 0 \\ \mathbf{t}[i] = \mathbf{s}[i] \times f(x_j, 2)[i] \end{cases} \tag{16}$$

We claim that to satisfy the above equation simultaneously for all possible i and j, there must be at least one entity x_k for which the convolution function returns a zero vector; that is, $f(x_k, 1)[i] = 0$ for all $0 \leq i \leq d$. To see why this is true, assume the contrary; that is, for each convolution filter $f(x_j, 1)$ has at least one bit that is different than zero for arbitrary entity x_j. Without loss of generality, let the jth bit $f(x_j, 1)[j] = K_j$, for all $1 \leq j \leq |\mathcal{E}|$ and $K_j \in \mathcal{R}^*$. Then, to satisfy Equation 16 at index j, we must set $\mathbf{t}[j] = \mathbf{s}[j] \times f(x_j, 2)[j]$. As Equation 16 must be satisfied for all entities, then all other entities x_k with $k \neq j$ must have their j-th bit set to zero. Thus $f(x_k, 1)[j] = 0$ for all $0 \leq j \leq d$ and $j \neq k$. See Figure 5. Since we have $d < |\mathcal{E}|$, and by the pigeon-hole principle, there must be at least one entity x_{d+1} such that $f(x_{d+1}, 1) = 0$ for all $1 \leq i \leq d$. This contradicts the original assumption, and thus there exists at least one k for which $f(x_k, 1)$ returns a zero vector.

This would further imply that any tuple having x_k in the first position will have a score ϕ_θ^H of zero. This would be regardless of the relation in the tuple, or whether or not it is true. This violates the full-expressivity of HypE, thus contradicting the original assumption that the lemma statement is False. Therefore, when $d < |\mathcal{E}|$, there is no parametrization for t such that HypE represents selection while retaining full expressivity. ■

Appendix C. More Experiments

C.1 ReAlE with a fixed negative ratio and batch size

The results reported in Table 1 of the main paper were obtained by running experiments using the BoxE setup: the negative ratio and batch size were tuned for each model.

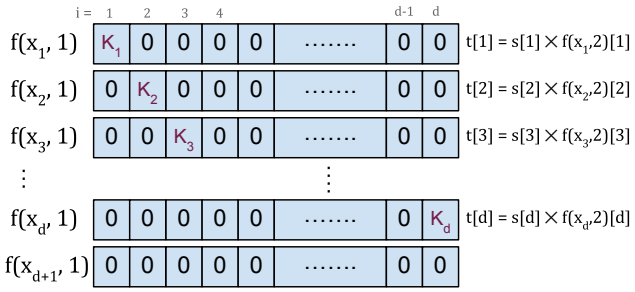

Figure 5: Proof of Theorem 12: If at least one value of each vector $f(x_i, 1)$ is different than zero ($K_i > 0$) and the value of at most one position i can be non-zero (the d columns in the figure), then by the pigeon-hole principle, we will run out of indices as we have more entities in the knowledge hypergraph. In the above image, the entity x_{d+1}.

Here, we follow the setup used by all other baselines to report their results and run experiments by fixing the negative ratio to 10 and batch size to 128. We summarize the results for the baselines and ReAlE in Table 4. *ReAlE (Small)* refers to a ReAlE model that has a negative ratio of 10 and a batch size of 128 (called *Small* because it has a smaller negative ratio and bath size compared to the best set of hyperparameters).

	JF17K				FB-AUTO				M-FB15K			
Model	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
m-DistMult (Fatemi et al., 2020)	.463	.372	.510	.634	.784	.745	.815	.845	.705	.633	.740	.844
m-CP (Fatemi et al., 2020)	.392	.303	.441	.560	.752	.704	.785	.837	.680	.605	.715	.828
m-TransH (Wen et al., 2016)	.444	.370	.475	.581	.728	.727	.728	.728	.623	.531	.669	.809
GETD (Liu et al., 2020)	.151	.104	.151	.258	.367	.254	.422	.601	-	-	-	-
HSimplE (Fatemi et al., 2020)	.472	.378	.520	.645	.798	.766	.821	.855	.730	.664	.763	.859
HypE (Fatemi et al., 2020)	.494	.408	.538	.656	.804	.774	.823	.856	.777	.725	.800	.881
G-MPNN (Yadati, 2020)	.501	.425	.537	.660	-	-	-	-	.779	.732	.805	.894
ReAlE (Small)	**.530**	**.454**	**.563**	**.677**	**.861**	**.836**	**.877**	**.908**	**.801**	**.755**	**.823**	**.901**

Table 4: Knowledge hypergraph completion results on JF17K, FB-AUTO and M-FB15K for baselines and the proposed method for a fixed embedding dimension of 200, negative ratio of 10, and batch size of 128. Our method ReAlE (Small) outperforms the baselines on all datasets with these settings.

C.2 Results of ReAlE with different non-linear functions

Following the last experiment with a fixed negative ratio and batch size, we tried different non-linear functions (σ). The results of the different nonlinear functions are in Table 5. As *sigmoid* is the winner for JF17K, we only experiment with *sigmoid* for the remaining datasets.

	JF17K			
Model	MRR	Hit@1	Hit@3	Hit@10
ReAlE (σ as exponent) (Ours)	0.394	0.311	0.428	0.548
ReAlE (σ as tanh) (Ours)	0.512	0.430	0.548	0.667
ReAlE (σ as sigmoid) (Ours)	**0.530**	**0.454**	**0.563**	**0.677**

Table 5: Knowledge hypergraph completion results for ReAlE on JF17K for different σ (nonlinear function).

C.3 Do the learned embeddings follow the theoretical findings?

Our theoretical analysis shows that ReAlE is able to represent a large subset of relational algebra operations. Here, we further investigate whether, in practice, the embeddings learned by ReAlE follow the theoretical results.

We thus first look at the relations in REL-ER that are the result of the renaming operation and observe the embeddings that ReAlE generates for each. If relation t is a renaming of relation s, one possible parametrization for the embedding of t is the one proposed in the proof of Theorem 3. Analysing multiple examples of relations t representing renaming of s in REL-ER, we observe that in all cases, the embedding of relations t is the closest to the parametrization proposed in the proof compared to all other relation embeddings. We did the same analysis with the rest of the operations (and their respective theorems), and the same holds for all. The results of Table 2 in the main paper show the breakdown performance for each operation: further evidence for learnability of the operations in ReAlE.

This experiment was only done for renaming because according to the proofs for the other operations, there are multiple ways for ReAlE to represent them.

Appendix D. Implementation Details

We implement ReAlE in PyTorch (Paszke et al., 2019) and use Adagrad (Duchi et al., 2011) as the optimizer and dropout (Srivastava et al., 2014) to regularize the model. We perform early stopping and hyperparameter tuning based on the MRR on the validation set. We fix the maximum number of epochs to 1000 and embedding size to 200. We tune *lr* (learning rate) and *w* (window size) using the sets $\{0.05, 0.08, 0.1, 0.2\}$, and $\{1, 2, 4, 5, 8\}$ (first five divisors of 200). We tune σ (nonlinear function) using the set $\{tanh, sigmoid, exponent\}$ for the JF17K dataset. As *sigmoid* outperforms *tanh* and *exponent* on JF17K, we only tried *sigmoid* for other datasets. For the experiment on REL-ER and also in Section C.1, we fixed the negative ratio and batch size of all baselines and our model to 10 and 128 respectively.

Reported results for the baselines on JF17K, FB-AUTO, and M-FB15K are taken from the original paper except for that of GETD Liu et al. (2020) and BoxE Abboud et al. (2020) on M-FB15K. The original paper of GETD only reports results for arity 3 and 4 as trained and tested separately in the corresponding arity. However, in our experimental setup, we train and test in a dataset containing relations of different arities. For that, we train and test GETD. As GETD learns a tensor of dimension $|r|$ for each relation r, it needs $d^{|r|}$ (with d as embedding size) number of parameters. The original paper proposes smart strategies to reduce the number of parameters to be learned by the model.

However, we still need to store the relation embedding and thus need to store $d^{|r|}$ floating-point numbers for each relation r. Because of our memory limitation (12GB GPU), we could only train the GETD model for an embedding size of less than 10.

The sign "-" in Table 1 indicates that the corresponding paper has not provided the results.

For the experiment on the synthetic dataset, we compare our model with m-TransH [2] (Wen et al., 2016), HypE [3] (Fatemi et al., 2020), and BoxE [4] (Abboud et al., 2020), which are the only competitive baselines that have provided the code.

For all the experiments we use a single 12GB GPU (NVIDIA Tesla P100 PCIe 12 GB).

Appendix E. Runtime and Space Complexity of ReAlE

Runtime complexity. For a tuple $r(x_1, \ldots, x_n)$, ReAlE performs $O(nd)$ multiplications to compute the score in Equation 2. This implies that ReAlE scales linearly with the arity of the relations in a knowledge hypergraph.

Space complexity. ReAlE embeds a relation r with arity n into a matrix of size $n \times d$ and each entity into a vector of size d. Therefore, for a knowledge hypergraph with $|\mathcal{E}|$ entities and $|\mathcal{R}|$ relations with arity n, ReAlE requires $(|\mathcal{E}|+n|\mathcal{R}|)d$ parameters.

References

Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: A box embedding model for knowledge base completion. In *Proceedings of the Thirty-Fourth Annual Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM ICMD*, 2008.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

David Buchman and David Poole. Representing aggregators in relational probabilistic models. In *Proc. Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, 1978.

2. https://github.com/wenjf/multi-relational_learning
3. https://github.com/ElementAI/HypE
4. https://github.com/ralphabb/BoxE

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer, 2018.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(Jul):2121–2159, 2011.

Paul Erdős and Alfred Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

Bahare Fatemi, Siamak Ravanbakhsh, and David Poole. Improved knowledge graph embedding using background taxonomic information. In *AAAI*, 2019.

Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. Knowledge hypergraphs: Prediction beyond binary relations. In *IJCAI*, 2020.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *AAAI*, 2019.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. Message passing for hyper-relational knowledge graphs. *arXiv preprint arXiv:2009.10847*, 2020.

Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. Link prediction on n-ary relational data. In *The World Wide Web Conference*, pages 583–593, 2019.

Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. Neuinfer: Knowledge inference on n-ary facts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6141–6151, 2020.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *NIPS*, 2018.

Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. Smoothing the geometry of probabilistic box embeddings. In *International Conference on Learning Representations*, 2018.

Yu Liu, Quanming Yao, and Yong Li. Generalizing tensor decomposition for n-ary relational knowledge bases. In *Proceedings of The Web Conference 2020*, pages 1104–1114, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(2):1–189, 2016.

Paolo Rosso, Dingqi Yang, and Philippe Cudré Mauroux. Beyond triplets: hyper-relational knowledge graph embedding for link prediction. In *Proceedings of The Web Conference 2020*, pages 1885–1896, 2020.

Koustuv Sinha, Shagun Sodhani, Joelle Pineau, and William L Hamilton. Evaluating logical generalization in graph neural networks. *arXiv preprint arXiv:2003.06560*, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. On the representation and embedding of knowledge bases beyond binary relations. In *IJCAI*, 2016.

Naganand Yadati. Neural message passing for multi-relational ordered and recursive hypergraphs. *Advances in Neural Information Processing Systems*, 33, 2020.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. Hypergcn: Hypergraph convolutional networks for semi-supervised classification. *arXiv preprint arXiv:1809.02589*, 2018.

Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *Proceedings of the 2018 World Wide Web Conference*, pages 1185–1194, 2018.