TransINT: Embedding Implication Rules in Knowledge Graphs with Isomorphic Intersections of Linear Subspaces

So Yeon Min  
MIT CSAIL  
Cambridge, MA 02142  
symin95@gmail.com

Preethi Raghavan  
IBM Research  
Cambridge, MA 02142  
praghav@us.ibm.com

Peter Szolovits  
MIT CSAIL  
Cambridge, MA 02142  
psz@mit.edu

Abstract

Knowledge Graphs (KG), composed of entities and relations, provide a structured representation of knowledge. For easy access to statistical approaches on relational data, multiple methods to embed a KG into \( f(KG) \in \mathbb{R}^d \) have been introduced. We propose TransINT, a new KG embedding method that isomorphically preserves the implication ordering among relations in the embedding space. TransINT maps set of entities (tied by a relation) to continuous sets of vectors that are inclusion-ordered isomorphically to relation implications. We further suggest possibilities of learning semantic relatedness among sets of entities with magnitude of angle between the embedded continuous sets.

1 Introduction

Recently, learning distributed vector representations of multi-relational knowledge has become an active area of research [1, 8, 5, 4, 2]. These methods map components of a KG (entities and relations) to elements of \( \mathbb{R}^d \) and capture statistical patterns, regarding vectors close in distance as representing similar concepts. However, they lack common sense knowledge which are essential for reasoning [13, 4, 7]. For example, “parent” and “father” would be deemed similar by KG embeddings, but by common sense, “father ⇒ parent” yet not the other way around.

Thus, one focus of current research is to bring common sense rules to KG embeddings [4, 13, 15]. Some methods impose hard geometric constraints and embed asymmetric orderings of knowledge [7, 11, 12]. However, they only embed hierarchy (unary \( \text{Is}_a \) relations), and cannot embed n-ary relations in KG’s. Moreover, their hierarchy learning is largely incompatible with conventional relational learning, because they put hard constraints on distance to represent partial ordering, which is a common metric of similarity/relatedness in relational learning.

We propose TransINT, a new KG embedding method that isomorphically preserves the implication ordering among relations in the embedding space. With rank-based parameter sharing, TransINT restrict entities tied by a relation to be embedded to vectors in a particular region of \( \mathbb{R}^d \). For example, we map any entities tied by \( \text{is_father_of} \) to vectors in a region that is part of the region for \( \text{is_parent_of} \); so that we can automatically know that if John is a father of Tom, he is also his parent even if such a fact is missing in the KG. Such set-to-region mapping allows using angles between regions as semantic relatedness among sets - an extension of the line of thought as in word/image embedding methods such as [6, 3] to relational embedding.

Figure 1: Two equivalent ways of expressing relations. (a): relations defined in a hypothetical KG. (b): relations defined in a set-theoretic perspective (Definition 1). Because \( \text{is_father_of} ⇒ \text{is_parent_of} \), the set for \( \text{is_father_of} \) is a subset of that for \( \text{is_parent_of} \) (Definition 2). (c): Hierarchical depiction of familial relations.

The main contributions of our work are: (1) A novel KG embedding such that implication rules in the original KG are guaranteed to unconditionally, not approximately, hold. (2) We introduce a novel parameter sharing regularization and negative example construction methods. (3) Our model suggests possibilities of learning semantic relatedness between groups of objects.

2 TransINT

In this section, we describe the intuition and justification of our method. We first define relation as sets, and revisit TransH as mapping relations to sets in $\mathbb{R}^d$. Finally, we propose TransINT, which connects the ordering of the two aforementioned sets. We put * next to definitions and theorems we propose/introduce. Otherwise, we use existing definitions and cite them.

2.1 Sets as Relations

Relations can be defined as sets. For example, in Figure 1a, 1b, $\text{Is\_Father\_Of}(\text{Tom}, \text{Harry})$ is equivalent to $(\text{Tom}, \text{Harry}) \in R_{\text{Is\_Father\_Of}}$. $R_i$ is called the relation set of $r_i$. \[9\]

**Definition (Relation Set):** Let $r_i$ be a binary relation $x, y$ entities. Then, $r_i(x, y)$ iff there exists some set $R_i$ such that the pair $(x, y) \in R_i$. $R_i$ is called the relation set of $r_i$. \[9\]

**Definition (Logical Implication):** For two relations, $r_1$ implies $r_2$ (or $r_1 \Rightarrow r_2$) iff $\forall x, y$, 

\[
(x, y) \in R_1 \Rightarrow (x, y) \in R_2
\]

or equivalently,

\[R_2 \subset R_1\] \[9\]

For example, $\text{Is\_Father\_Of} \Rightarrow \text{Is\_Parent\_Of}$ by common sense. We can see in Figure 1b that, $R_{\text{Is\_Father\_Of}} \subset R_{\text{Is\_Parent\_Of}}$.

2.2 Background: TransE and TransH

Given a fact triple $(h, r, t)$ in a given KG (i.e. $(\text{Harry}, \text{is\_father\_of}, \text{Tom})$), TransE wants $\vec{h} + \vec{r} \approx \vec{t}$ where $\vec{h}, \vec{r}, \vec{t}$ are embeddings of $h, r, t$. In other words, the distance between two entity vectors is equal to a fixed relation vector. TransE applies well to 1-to-1 relations but has issues for N-to-1, 1-to-N and N-to-N relations, because the distance between two vectors are unique and thus two entities can only be tied with one relation.

To address this, TransH constraints the distance of entities in a multi-relational way, by decomposing distance with projection (Figure 2a). TransH first projects an entity vector into a hyperplane unique to each relation, and then requires their difference is some constant value. Like TransE, it embeds an entity to a vector. However, for each relation $r_j$, it assigns two components: a relation-specific hyperplane $H_j$ and a fixed vector $\vec{r}_j$ on $H_j$. For each fact triple $(h, r_j, t)$, TransH wants (Figure 2) 

\[
\vec{h}_\bot + \vec{r}_j \approx \vec{t}_\bot \cdots \cdot \text{(Eq. 1)}
\]

where $\vec{h}_\bot, \vec{t}_\bot$ are projections on $\vec{h}, \vec{t}$ onto $H_j$ (Figure 2a).

**Revisiting TransH** We interpret TransH in a novel perspective. An equivalent way to put Eq.1 is to change the order of subtraction and projection:

Projection of $(t - h)$ onto $H_j \approx \vec{r}_j$.
For example, if

\[ \text{Main Theorem 1} \]

where

\[ H \]

This means that all entity vectors \((\vec{h}, \vec{t})\) such that their distance \(\bar{t} - \bar{h}\) belongs to the red line are considered to be tied by relation \(r_j\) (Figure 2b) i.e. \(R_j \approx \text{red line}\). For example,

\[
(Tom, Sue) \in R_j \cong (Sue - Tom) \in \text{red line}
\]

The red line is the set of all vectors whose projection onto \(H_j\) is the fixed vector \(\vec{r_j}\). Thus, TransH actually embeds a relation set in KG (figure 1b) to a continuous set in \(\mathbb{R}^d\). We call such sets relation space; in other words, a relation space includes the other two’s.

Thus, in TransH:

\[
\begin{align*}
& r_i(x,y) \equiv (x,y) \in R_i \quad \text{(relation in KG)} \\
& \cong (x-y) \in \text{relation space of } r_i \quad \text{(relation in } \mathbb{R}^d) \\
\end{align*}
\]

### 2.3 TransINT

Like TransH, TransINT embeds a relation \(r_j\) to a (subspace, vector) pair \((H_j, \vec{r_j})\). However, TransINT modifies the relation embeddings \((H_j, \vec{r_j})\) so that the relation spaces (i.e. red line of Figure 2b) are ordered by implication; we do so by intersecting the \(H_j\)’s and projecting the \(\vec{r_j}\)’s (Figure 3a).

More generally, TransINT requires two hard geometric constraints on \((H_j, \vec{r_j})\)’s that

For distinct relations \(r_i, r_j\), require the following if and only if \(r_i \Rightarrow r_j\):

**Intersection Constraint:** 
\[
H_j = H_i \cap H_j.
\]

**Projection Constraint:** Projection of \(\vec{r}_i\) to \(H_j\) is \(\vec{r}_j\).

where \(\vec{H}_i, \vec{H}_j, \vec{r}_i, \vec{r}_j\) are distinct. For example, we assign

\[
H_{is\_parent\_of} = H_{is\_father\_of} \cap H_{is\_mother\_of},
\]

and require \(\vec{r}_{is\_father\_of}\) and \(\vec{r}_{is\_mother\_of}\) to have the same projection onto \(H_{is\_parent\_of}\) (orange dotted line of Figure 3a), which is \(\vec{r}_{is\_parent\_of}\).

Our main result is that the two above constraints order relation spaces by inclusion isomorphically to implication ordering. Figure 3b graphically illustrates that \(is\_parent\_of\)’s relation space (green hyperplane) includes those of \(is\_father\_of\) (blue line) and \(is\_mother\_of\) (red line)\(^1\).

In other words, the two constraints guarantee that an ordering isomorphic to implication holds in the embedding space: \((r_i \Rightarrow r_j) \iff (r_j \text{’s rel. space } \subset r_j \text{’s rel. space})\). For example, if \(is\_parent\_of\) subsumes \(is\_father\_of\), the first relation space includes the latter’s; the converse also holds (Figure 3). At first sight, it may look paradoxical that the \(H_j\)’s and the relation spaces are inversely ordered; however, it is a natural consequence of the rank-based geometry in \(\mathbb{R}^d\).

**Main Theorem 1** (Isomorphism): Let \(\{(R_i, \vec{r}_i)\}_n\) be the (subspace, vector) embeddings assigned to relations \(\{R_i\}_n\) by the *Intersection Constraint* and the *Projection Constraint*; \(P_i\) the projection matrix of \(H_i\). Then, \(\{(Sol(P_i, \vec{r}_i))_n, C\} \text{ is isomorphic to } (\{R_i\}_n, C)\).

\(^1\)Proofs and details are present in the appendix.
In actual implementation and training, TransINT requires something less strict than \( P_i(t-h) = r_i \):

\[
P_i(t-h) - r_i \approx 0 \iff ||P_i(t-h - r_i)||_2 < \epsilon,
\]

for some non-negative and small \( \epsilon \). This bounds \( t-h - r_i \) to regions with thickness \( 2\epsilon \), centered around \( \text{Sol}(P_i, r_i) \) (Figure 6). We prove that isomorphism still holds with this weaker requirement.

**Definition** \( ^* \) (\( \text{Sol}(P, k) \)) : Given a projection matrix \( P \), the solution space of \( ||P \vec{x} - k||_2 < \epsilon \) is denoted as \( \text{Sol}(P, k) \).

**Main Theorem 2** \( ^* \) (Margin-aware Isomorphism) : For all non-negative scalar \( \epsilon \), \( \{\text{Sol}(P_i, r_i)\}_{i=1}^{n} \subset \) is isomorphic to \( \{\text{R}_i\}_{i=1}^{n}, \subset \).

To intuitively explain our formulation, the lesser constraint the space to be projected onto, the more freedom a vector is given; which is analogous to that, for example, \( \text{is}_\text{family}_\text{of} \) puts more freedom on who can be tied by it than \( \text{is}_\text{father}_\text{of} \). (Figure 1b).

### 2.4 Training TransINT

We construct negative examples (wrong fact triplets) and train with a margin-based loss, as in TransE and TransH. We introduce a novel parameter sharing scheme to tie the basis of the \( H_r \)'s. This scheme automatically makes traversing through a particular triple also execute training with all triple that it implies (even if that triple itself is missing in the KG), eliminating the need to manually create missing triples that are true by implication rule. We discuss details in the appendix.

![Figure 4](image-url)

**Figure 4**: Figure 3(b)’s relation spaces when \( P_i(t-h) - r_i \approx 0 \iff ||P_i(t-h - r_i)||_2 < \epsilon \) is required. (a): Each relation space now becomes regions with thickness \( \epsilon \), centered around figure 3(b)’s relation space. (b): Relationship of the angle and area of overlap between two relation spaces. With respect to the green region, the nearly perpendicular cylinder overlaps much less with it than the other cylinder with much closer angle.

### 3 Future Work: Semantically Interpreting Overlap Between Regions

Traditional embedding methods that map an object (i.e. words, images) to a singleton vector learn soft tendencies between embedded vectors, such as semantic similarity [6], [3]. A common metric for such tendency is cosine similarity, or angle between two embeddings. TransINT extends such line of thought to semantic relatedness between groups of objects - as angle between regions of embeddings. In Figure 6b, one can observe that the closer the angle between two regions, the larger the overlap in area. For entities \( h \) and \( t \) to be tied by both relations \( r_1, r_2 \), \( t-h \) has to belong to the intersection of their relation spaces. We hypothesize that if the angle between two relation spaces is near 0, then two relations are semantically very similar. For example, relations such as \( \text{was}_\text{born}_\text{in} \) and \( \text{has}_\text{citizenship}_\text{of} \) do not imply one another but are closely related. In such a case, both a majority of \( V_1 \) and \( V_2 \) belong to the intersection. Conversely, if the angle between two relations are near 0, the area of the intersection takes up the majority of both relation spaces. We will investigate this as an ongoing work.

### 4 Conclusion

We presented a new KG embedding method that embed sets of entities (tied by relations) to continuous sets in \( \mathbb{R}^d \). We show that the inclusion ordering of the embedded sets are isomorphic to the implication ordering relating relations. We further suggest possibilities of applying our model for mining semantic similarity among sets of entities.
References


Appendix

A Proof For TransINT’s Isomorphic Guarantee

Here, we provide the proofs for Main Theorems 1 and 2. We also explain some concepts necessary in explaining the proofs. We put * next to definitions and theorems we propose/introduce. Otherwise, we use existing definitions and cite them.
A.1 Linear Subspace and Projection

We explain in detail elements of $\mathbb{R}^d$ that were intuitively discussed. In this and later sections, we mark all lemmas and definitions that we newly introduce with *; those not marked with * are accompanied by reference for proof. We denote all $d \times d$ matrices with capital letters (ex) $A$) and vectors with arrows on top (ex) $\mathbf{x}$).

A.1.1 Linear Subspace and Rank

The linear subspace given by $A(x - \mathbf{b}) = 0$ ($A$ is $d \times d$ matrix and $\mathbf{b} \in \mathbb{R}^d$) is the set of $x \in \mathbb{R}^d$ that are solutions to the equation; its rank is the number of constraints $A(x - \mathbf{b}) = 0$ imposes. For example, in $\mathbb{R}^3$, a hyperplane is a set of $\mathbf{x} = [x_1, x_2, x_3] \in \mathbb{R}^3$ such that $ax_1 + bx_2 + cx_3 - d = 0$ for some scalars $a, b, c, d$; because vectors are bound by one equation (or its "A" only really contains one effective equation), a hyperplane’s rank is 1 (equivalently $\text{rank}(A) = 1$). On the other hand, a line in $\mathbb{R}^3$ imposes to 2 constraints, and its rank is 2 (equivalently $\text{rank}(A) = 2$).

Consider two linear subspaces $H_1, H_2$, each given by $A_1(\mathbf{x} - \mathbf{b}_1) = 0, A_2(\mathbf{x} - \mathbf{b}_2) = 0$. Then, $(H_1 \subset H_2) \iff (A_1(\mathbf{x} - \mathbf{b}_1) = 0 \Rightarrow A_2(\mathbf{x} - \mathbf{b}_2) = 0)$ by definition. In the rest of the paper, denote $H_i$ as the linear subspace given by some $A_i(\mathbf{x} - \mathbf{b}_i) = 0$.

A.1.2 Properties of Projection

**Invariance** For all $\mathbf{x}$ on $H$, projecting $\mathbf{x}$ onto $H$ is still $\mathbf{x}$; the converse is also true.

**Lemma 1** $P \mathbf{x} = \mathbf{x} \iff \mathbf{x} \in H$ \[10\].

**Orthogonality** Projection decomposes any vector $\mathbf{x}$ to two orthogonal components - $P \mathbf{x}$ and $(I - P) \mathbf{x}$ (Figure 4). Thus, for any projection matrix $P$, $I - P$ is also a projection matrix that is orthogonal to $P$ (i.e. $P(I - P) = 0$) \[10\].

**Lemma 2** Let $P$ be a projection matrix. Then $I - P$ is also a projection matrix such that $P(I - P) = 0$ \[10\].

The following lemma also follows.

**Lemma 3** $\|P \mathbf{x}\| \leq \|P \mathbf{x} + (I - P) \mathbf{x}\| = \|\mathbf{x}\|$ \(10\).

**Projection onto an included space** If one subspace $H_1$ includes $H_2$, the order of projecting a point onto them does not matter. For example, in Figure 3, a random point $\mathbf{a}$ in $\mathbb{R}^3$ can be first projected onto $H_1$ at $\mathbf{b}$, and then onto $H_3$ at $\mathbf{d}$. On the other hand, it can be first projected onto $H_3$ at $\mathbf{d}$, and then onto $H_1$ at still $\mathbf{d}$. Thus, the order of applying projections onto spaces that includes one another does not matter.

If we generalize, we obtain the following two lemmas (Figure 6):

**Lemma 4** Every two subspaces $H_1 \subset H_2$ if and only if $P_1P_2 = P_2P_1 = P_1$.

\[6\]
proof) By Lemma 1, if \( H_1 \subset H_2 \), then \( P_2 \vec{x} = \vec{x} \) \( \forall \vec{x} \in H_1 \). On the other hand, if \( H_1 \not\subset H_2 \), then there is some \( \vec{x} \in H_1, \vec{x} \not\in H_2 \) such that \( P_2 \vec{x} \neq \vec{x} \). Thus,
\[
H_1 \subset H_2 \Leftrightarrow \forall \vec{x} \in H_1, \quad P_2 \vec{x} = \vec{x}
\]
\[
\Leftrightarrow \forall \vec{y}, \quad P_2(P_1 \vec{y}) = P_1 \vec{y} \Leftrightarrow P_2P_1 = P_1.
\]
Because projection matrices are symmetric (10),
\[
P_2P_1 = P_1 = P_1^T = P_1P_2^T = P_1P_2. \quad \blacksquare
\]

Lemma 5* For two subspaces \( H_1, H_2 \) and vector \( \vec{k} \subset H_2 \),
\[
H_1 \subset H_2 \Leftrightarrow \text{Sol}(P_2, \vec{k}) \subset \text{Sol}(P_1, \vec{k}).
\]

proof) \( \text{Sol}(P_2, \vec{k}) \subset \text{Sol}(P_1, \vec{k}) \) is equivalent to \( \forall \vec{x} \in \mathbb{R}^d, P_2 \vec{x} = \vec{k} \Rightarrow P_1 \vec{x} = P_1 \vec{k} \).

By Lemma 4, if \( H_1 \subset H_2 \Leftrightarrow P_1P_2 = P_1 \). Since \( \vec{k} \subset P_2, P_2 \vec{x} = \vec{k} \Leftrightarrow P_2(x - \vec{k}) = 0 \Leftrightarrow P_1(P_2 \vec{x} - \vec{k}) = 0 \Leftrightarrow P_1P_2 \vec{x} = P_1 \vec{k} \Leftrightarrow P_1 \vec{x} = P_1 \vec{k} \). \( \quad \blacksquare \)

Partial ordering If two subspaces strictly include one another, projection is uniquely defined from lower rank subspace to higher rank subspace, but not the other way around. For example, in Figure 3, a point \( \vec{b} \) in \( R^3 \) (rank 1) always projected onto \( H_1 \) (rank 1) at point \( \vec{b} \). Similarly, point \( \vec{b} \) on \( H_1 \) (rank 1) is always projected onto similarly, onto \( H_3 \) (order 2) at point \( d \). However, "inverse projection" from \( H_3 \) to \( H_1 \) is not defined, because not only \( \vec{b} \) but other points on \( H_1 \) (such as \( \vec{b} \)) project to \( H_3 \) at point \( \vec{d} \); these points belong to \( \text{Sol}(P_3, \vec{d}) \). In other words, \( \text{Sol}(P_1, \vec{b}) \subset \text{Sol}(P_3, \vec{d}) \). This is the key intuition for isomorphism, which we prove in the next chapter.

A.2 Proof for Isomorphism

Now, we prove that TransINT’s two constraints (section 2.3) guarantee isomorphic ordering in the embedding space.

Two posets are isomorphic if their sizes are the same and there exists an order-preserving mapping between them.] Thus, any two posets \( (\{A_i\}_{n}, \subset), (\{B_i\}_{n}, \subset) \) are isomorphic if \( |\{A_i\}_n| = |\{B_i\}_n| \) and
\[
\forall i, j \quad A_i \subset A_j \Leftrightarrow B_i \subset B_j
\]

Main Theorem 1 (Isomorphism): Let \( \{\{H_i, \vec{r}_i\}\}_{n} \) be the (subspace, vector) embeddings assigned to relations \( \{R_i\}_n \) by the Intersection Constraint and the Projection Constraint; \( P_i \) the projection matrix of \( H_i \). Then, \( \{\{\text{Sol}(P_i, \vec{r}_i)\}_{n}, \subset\} \) is isomorphic to \( \{\{R_i\}_n, \subset\} \).

proof) Since each \( \text{Sol}(P_i, \vec{r}_i) \) is distinct and each \( R_i \) is assigned exactly one \( \text{Sol}(P_i, \vec{r}_i) \),
\[
|\{\text{Sol}(P_i, \vec{r}_i)\}_n| = |\{R_i\}_n|. \quad (1)
\]

Now, let’s show
\[
\forall i, j \quad R_i \subset R_j \Leftrightarrow \text{Sol}(P_i, \vec{r}_i) \subset \text{Sol}(P_j, \vec{r}_j).
\]

Because the \( \forall i, j \), intersection and projection constraints are true iff \( R_i \subset R_j \), enough to show that the two constraints hold iff \( \text{Sol}(P_i, \vec{r}_i) \subset \text{Sol}(P_j, \vec{r}_j) \).

First, let’s show \( R_i \subset R_j \Rightarrow \text{Sol}(P_i, \vec{r}_i) \subset \text{Sol}(P_j, \vec{r}_j) \). From the Intersection Constraint, \( R_i \subset R_j \Rightarrow H_j \subset H_i \). By Lemma 5, \( \text{Sol}(P_i, \vec{r}_i) \subset \text{Sol}(P_j, \vec{r}_j) \). From the Projection Constraint, \( \vec{r}_j = P_j \vec{r}_i \). Thus, \( \text{Sol}(P_i, \vec{r}_i) \subset \text{Sol}(P_j, P_j \vec{r}_i) = \text{Sol}(P_j, \vec{r}_j) \cdots \cdots (2) \)

Now, let’s show the converse; enough to show that if \( \text{Sol}(P_i, \vec{r}_i) \subset \text{Sol}(P_j, \vec{r}_j) \), then the intersection and projection constraints hold true.
\[
\text{Sol}(P_i, \vec{r}_i) \subset \text{Sol}(P_j, \vec{r}_j)
\]
\[
\Leftrightarrow \forall \vec{x}, \quad P_i \vec{x} = \vec{r}_i \Rightarrow P_j \vec{x} = \vec{r}_j
\]
\[
\forall \vec{x}, \quad P_j P_i \vec{x} = P_j \vec{r}_i
\]
\[
\forall \vec{x}, \quad P_j \vec{x} = \vec{r}_j
\]
both have to be true. For any \( \mathcal{X} \in H_i \), or equivalently, if \( \mathcal{X} = P_j \mathcal{Y} \) for some \( \mathcal{Y} \), then the second equation becomes \( \forall \mathcal{Y}, P_j P_j \mathcal{Y} = \mathcal{Y}_j \), which can be only compatible with the first equation if \( \mathcal{Y}_j = P_j \mathcal{Y}_i \), since any vector’s projection onto a subspace is unique. (Projection Constraint)

Now that we know \( \mathcal{Y}_j = P_j \mathcal{Y}_i \), by Lemma 5, \( H_i \subset H_j \) (intersection constraint). \( \cdots \) From (1), (2), (3), the two posets are isomorphic.

In actual implementation and training, TransINT requires something less strict than \( P_j(t - h) = \mathcal{Y}_i \):

\[
P_j(t - h) - \mathcal{Y}_i \approx 0 \iff \| P_j(t - h) - \mathcal{Y}_i \|_2 < \epsilon,
\]

for some non-negative and small \( \epsilon \). This bounds \( t - h - \mathcal{Y}_i \) to regions with thickness \( 2\epsilon \), centered around \( \text{Sol}(P_i, \mathcal{Y}_i) \) (Figure 5). We prove that isomorphism still holds with this weaker requirement.

**Definition** \( \text{Sol}_e(P, k) \): Given a projection matrix \( P \), the solution space of \( \| P \mathcal{X} - \mathcal{K} \|_2 < \epsilon \) is denoted as \( \text{Sol}_e(P, \mathcal{K}) \).

**Main Theorem 2 (Margin-aware Isomorphism):** For all non-negative scalar \( \epsilon \), \( \{ \text{Sol}_e(P_i, \mathcal{Y}_i) \}_{n, \subset} \) is isomorphic to \( \{ \mathcal{R}_1 \}_{n, \subset} \).

**proof**) Enough to show that \( \{ \text{Sol}_e(P_i, \mathcal{Y}_i) \}_{n, \subset} \) and \( \{ \text{Sol}_e(P_i, \mathcal{Y}_i) \}_{n, \subset} \) are isomorphic for all \( \epsilon \). First, let’s show

\[
\text{Sol}(P_i, \mathcal{Y}_i) \subset \text{Sol}(P_j, \mathcal{Y}_j) \Rightarrow \text{Sol}_e(P_i, \mathcal{Y}_i) \subset \text{Sol}_e(P_j, \mathcal{Y}_j).
\]

By Main Theorem 1 and Lemma 4,

\[
\text{Sol}(P_i, \mathcal{Y}_i) \subset \text{Sol}(P_j, \mathcal{Y}_j) \iff \mathcal{Y}_j = P_j \mathcal{Y}_i, P_j = P_j P_i.
\]

Thus, for all vector \( \mathcal{B} \),

\[
P_j(x - \mathcal{Y}_i) = \mathcal{B} \\
\iff P_j P_j(x - \mathcal{Y}_i) = P_j \mathcal{B} \\
\iff P_j(x - \mathcal{Y}_i) = P_j \mathcal{B} (\cdot \text{ Lemma 4}) \\
\iff P_j(x - \mathcal{Y}_i) = P_j \mathcal{B} (\cdot \text{ P}_j \mathcal{Y}_j = \mathcal{Y}_j = P_j \mathcal{Y}_i)
\]

Thus, if \( \| P_j(x - \mathcal{Y}_i) \| < \epsilon \), then \( \| P_j(x - \mathcal{Y}_i) \| = \| P_j(P_j(x - \mathcal{Y}_i)) \| < \| P_j(x - \mathcal{Y}_i) \| + (I - P_j)(P_j(x - \mathcal{Y}_i)) \| < \| P_j(x - \mathcal{Y}_i) \| < \epsilon \). \( \cdots \) (1)

Now, let’s show the converse. Assume \( \| P_j(x - \mathcal{Y}_i) \| < \epsilon \) for some \( i \). Then,

\[
\| P_j(x - \mathcal{Y}_i) \| < \epsilon \text{ to bound } \| P_j(x - \mathcal{Y}_j) \| \text{ at all for all } x,
\]

\[
P_j(I - P_j) = 0, P_j(\mathcal{Y}_i - \mathcal{Y}_j) = 0
\]

need to hold. By Lemma 4 and 5,

\[
P_j = P_j P_i \iff H_j \subset H_i \\
\iff \text{Sol}(P_i, \mathcal{Y}_i) \subset \text{Sol}(P_j, P_j \mathcal{Y}_i) = \text{Sol}(P_j, \mathcal{Y}_i) \cdot \text{(2)}
\]

\[|\{ \text{Sol}_e(P_i, \mathcal{Y}_i) \}_{n}| = |\{ \text{Sol}(P_i, \mathcal{Y}_i) \}_{n}| \text{ holds obviously; each } \text{Sol}(P_i, \mathcal{Y}_i) \text{ has a distinct } \text{Sol}_e(P_i, \mathcal{Y}_i)
\]

and each \( \text{Sol}_e(P_i, \mathcal{Y}_i) \) also has a distinct "center" \( \{ \text{Sol}(P_i, \mathcal{Y}_i) \}_{\cdot} \cdot \text{ (3)}
\]

From (1), (2), (3), the two sets are isomorphic.

**B** Initialization and Training

The intersection and projection constraints can be imposed with parameter sharing. We describe how shared parameters are initialized and trained.

8
B.1 Parameter Sharing Initialization

From initialization, we bind parameters so that they satisfy the two constraints. For each entity $e_j$, we assign a $d$-dimensional vector $\vec{e}_j$. To each $R_i$, we assign $(H_i, r_i^*)$ (or $(A_i, r_i^*)$) with parameter sharing. We first construct the $H$’s.

**Intersection constraint** We define the $H$’s top-down, first defining the intersections and then the subspaces that go through it. To the head $R_h$, assign $a_h$ linearly independent rows for the basis of $H_h$. Then, to each $R_i$ that is not a head, additionally assign $a_i$ rows linearly independent to the bases of all of its parents, and construct $H_i$ with its bases and the bases of all of its parents. Projection matrices can be uniquely constructed given the bases (10).

Now, we initialize the $r_i^*$’s.

**Projection Constraint** To the head $R_h$, pick any random $x_h \in \mathbb{R}^d$ and assign $r_h^* = P_h x$. To each non-head $R_i$ whose parent is $R_p$, assign $r_i^* = r_p^* + (I - P_p)(P_i)x_i$ for some random $x_i$. This results in

$$P_p r_i^* = P_p r_p^* + P_p (I - P_p)(P_i)x_i = r_p^* + 0 = r_p^*$$

for any parent, child pair.

### B.1.1 Training

We construct negative examples (wrong fact triplets) and train with a margin-based loss, following the same protocols as in TransE and TransH.

**Training Objective** We adopt the same loss function as in TransH. For each fact triplet $(h, r_i, t)$, we define the score function

$$f(h, r_i, t) = ||P_i(t - h) - r_i^*||_2$$

and train a margin-based loss $L$ which is aggregates f’s and discriminates between correct and negative examples.

$$L = \sum_{(h, r_i, t) \in G} \max(0, f(h, r_i, t)^2 + \gamma - f(h', r_i', t')^2)$$

where $G$ is the set of all triples in the KG and $(h', r_i', t')$ is a negative triple made from corrupting $(h, r_i, t)$. We minimize this objective with stochastic gradient descent.

**Automatic Grounding of Positive Triples** The parameter sharing scheme guarantees two advantages during all steps of training. First, the intersection and projection constraint are met not only at initialization but always.

Second, traversing through a particular $(h, r_i, t)$ also automatically executes training with $(h, r_p, t)$ for any $r_i \Rightarrow r_p$. For example, by traversing (Tom, is_father_of, Harry) in the KG, the model automatically also traverses (Tom, is_parent_of, Harry), (Tom, is_family_of, Harry), even if the two triples are missing in the KG. This is because $P_p P_i = P_p$ with the given initialization (section 4.1.1) and thus,

$$f(h, r_p, t) = ||P_p(t - h) - r_p^*||_2^2 = ||P_p((t - h) - r_i^*)||_2^2 \leq ||(P_p + (I - P_p))((t - h) - r_i^*)||_2^2 = f(h, r_i, t)$$

In other words, training $f(h, r_i, t)$ towards less than $\epsilon$ automatically guarantees, or has the effect of training $f(h, r_p, t)$ towards less than $\epsilon$. This enables the model to be automatically trained with what exists in the KG, eliminating the need to manually create missing triples that are true by implication rule.