Recursive Relational Abstraction (RRA)

Mark Tracy

1 Overview

Recursive Relational Abstraction (RRA) is a framework for designing systems to simulate reasoning that requires identifying or manipulating analogies. Building on the formal theory of analogy developed in Tracy (2025), it defines a general process through which raw inputs are broken into components that are then transformed into abstract representations, relational structures, and higher-order comparisons between relational structures. This framework motivates an inductive bias for deep learning systems, particularly on analogical reasoning benchmarks such as Raven's Progressive Matrices (RPM).

2 The Relational Bottleneck and Deep Learning Approaches

Previous work by Webb et al. has shown that implementing a relational bottleneck is advantageous for relational reasoning tasks. [1] In particular, a relational bottleneck is "any mechanism that restricts the space of all possible compressed representations to be a subset of the set M of relational signals," where a relational signal $R \in M$ is given by:[1]

$$\begin{aligned} R &= r(x_i, x_j)_{i \neq j} \\ &= \{r(x_1, x_2), \quad r(x_1, x_3), \quad \ldots, \quad r(x_{N-1}, x_N)\} \end{aligned}$$

Webb et al. explain:[1]

> $r(x_i, x_j)$ is a learned relation function that satisfies certain key relational properties (e.g., transitivity). One type of operation that satisfies the relevant properties is inner products of the form $\langle \phi(x_i), \phi(x_j) \rangle$... In a relational task, there exists $R \in M$ which is sufficient for predicting Y (i.e., $X \to R \to Y$).

It is unclear whether the Raven's Progressive Matrix problem and other higher-order reasoning problems are relational tasks by this definition. Moreover, the approach considers only binary relations and not higher-order, e.g. ternary, relations.

Altabaa et al. introduce the abstractor, a variant on the transformer that implements the relational bottleneck principle.[2] The key idea is that instead of deriving keys, queries, and values from the input data X, the abstractor derives keys and queries from X, and derives values from a learned set of “symbols,” S. Effectively, then, the abstractor is reasoning over a relational signal derived from the original data X (inner products of linear transformations of X), rather than over the features of X itself. [2]

Finally, Webb et al. introduce the notion of temporal context normalization (TCN). [3] TCN is like a batch normalization, except that the dimension over which normalization occurs is a sequence dimension. This normalization has been shown to improve extrapolation in relational reasoning tasks on sequence data.[3]

3 The Recursive Relational Abstraction Framework

3.1 Motivation

The RRA framework is based on the previously expounded view that analogies correspond one-to-one with abstractions that generalize relational structures present in the source and target domains of the analogy. By calling RRA a “framework,” I mean that it is a set of inductive priors that I hypothesize to be useful for learning to simulate reasoning that requires recognizing or manipulating analogies.

The general strategy of RRA is to break a problem into primitive components, represent those components in an abstract manner, and then twice recursively (1) group representations; and (2) represent common structure between them. This allows for a certain type of bottleneck that restricts downstream processing to operate over (second-order) relations between (first-order) relations between abstract characterizations of primitive components.

First, we introduce the RRA framework semi-formally to provide intuition; then we bolster intuition with an applied example, and finally we provide a precise mathematical description of RRA.

The RRA framework is a composition of the following (informally described) operations:

- G_0: break a problem into primitives.
- R_0: map primitives onto abstractions (objects).
- G_1: group objects into domains.
- R_1: characterize relationships-within-domains.
- G_2: groups domains together.
- R_2: characterize relationships between relationships-within-domains.

We illustrate now with an informal example. When faced with a task like Raven’s Progressive Matrices, a human reasoner may repeatedly:

- Break the problem into primitive components (e.g. panels)
- Perceive the components and represent them in abstract terms (e.g. by mentally describing each panel)
- Group components into domains and represent relationships between them in abstract terms (e.g. "each panel in row 1 contains a square" and "each panel in row 2 contains a triangle" and "the first two panels in row 3 contain circles")
- Consider groups of domains and recognize shared relational structure between them (e.g. "each row–or partial row–features a repeating shape").

3.2 Formal RRA Structure

In this framework, a **relational abstraction function** is defined as a function with the following properties:

- **Non-injectivity**: Multiple distinct inputs may map to the same output, mirroring the way abstraction identifies common structure across distinct instances.
- **Order Sensitivity**: Relational abstraction functions are not permutation-invariant, except trivially when they are unary. This allows the function to preserve relational asymmetries such as sequence information, spatial relationships, or transformations.
- **Nonlinearity**: Linearity is a special property of some functions that relational abstraction functions need not possess.

A model is an instance of Recursive Relational Abstraction (RRA) if it is a composition of the following components:

- X: An input space of numerical tensors.

  Raven's Progressive Matrix (RPM) example: a 3 x 3 matrix of images representing a candidate completed matrix to an RPM problem.

- $G_0 : X \to S_0^{D_0}$: An invertible function that partitions a tensor $X_0 \in X$ into D_0 components, or **raw elements**, each in a shared vector space $S_0 \subset \mathbb{R}^{d_0}$ for some $d_0 \in \mathbb{N}$.

  Raven's Progressive Matrix (RPM) example: a partition of the matrix into individual panels.

- $R_0 : S_0 \to S_1^{M_1}$ for some $S_1 \subset \mathbb{R}^{d_1}$: A learned relational abstraction function that operates in parallel over raw elements and produces M_1 **first-order abstract representations** of each.

  Raven's Progressive Matrix (RPM) example: a module that maps raw panel images into some lower-dimensional embedding space ($M_1 = 1$ in this example).

- $G_1 : S_1^{D_0} \to S_1^{D_1 \times K_1}$: A grouping function that takes as input a set of D_0 first-order abstract representations and outputs K_1 groups of D_1**-tuples** of first-order abstract representations.

  - Acts in parallel across the first-order representation dimension (i.e. M_1 times).
  - May be learned or may reflect some task-relevant structure.

  Raven's Progressive Matrix (RPM) example: a method for grouping such panel encodings as mentioned above. For example, one may group panel encodings by rows and columns, producing $K_1 = 6$ groups of $3-$tuples of panel encodings.

- $R_1 : S_1^{D_1} \to S_2^{M_2}$ for some $S_2 \subset \mathbb{R}^{d_2}$: A learned relational abstraction function that operates over D_1-tuples of first-order abstract representations and produces M_2 **second-order abstract representations** of each.
  - Applied in parallel across the first-order representation and group dimensions (i.e. $M_1 \times K_1$ times).
  - Each second-order abstraction represents a group of first-order abstractions.

  Raven's Progressive Matrix (RPM) example: second-order encoding of relational structure along rows or columns.

- $G_2 : S_2^{K_1} \to S_2^{D_2 \times K_2}$: A grouping function that takes as input K_1 second-order abstractions and outputs K_2 groups of **D_2-tuples** of second-order abstract representations.
  - Acts in parallel across the first-order representation and second-order representation dimensions (i.e. $M_1 \times M_2$ times).
  - May be learned or may reflect task-relevant structure.

  Raven's Progressive Matrix (RPM) example: grouping the aforementioned second-order encodings of relational structure along rows and those of relational structure along columns separately. This produces $K_2 = 2$ groups of $3-$tuples of second-order abstractions.

- $R_2 : S_2^{D_2} \to S_3^{M_3}$ for some $S_3 \subset \mathbb{R}^{d_3}$: A learned relational abstraction function that takes as input a D_2-tuple of second-order abstract representations and outputs M_3 third-order abstractions for each.
  - Applied in parallel across the first-order representation, second-order representation, and second-order group dimensions (i.e. $M_1 \times M_2 \times K_2$ times).
  - Each third-order abstraction represents a group of second-order abstractions.

  Raven's Progressive Matrix (RPM) example: third-order encoding of relational structure preserved across columns or across rows.

- $f : S_3^{K_2 \times M_3 \times M_2 \times M_1} \to Y$: a learned function that operates on the set of third-order abstractions and utilizes them for a downstream task whose output is in some target set Y.
  - In particular, $y \in Y$ might represent a score for the input tensor X_0.

– Grounds the model in a particular task, allowing task-relevant abstractions to be learned.

Raven's Progressive Matrix (RPM) example: a head that takes the tensor produced by the most recent step above and outputs a score for input matrix.

4 ARoN, SAViR-T, and WReN: Three RRA Instances

The following table illustrates how the core components of ARoN instantiate the RRA framework:

RRA Component	**Mathematical Role**	**ARoN Instantiation**
X_0	A numerical tensor	A complete Raven's Progressive Matrices input (3x3 visual grid), where the bottom-right panel is one of the candidate answers to the reasoning problem
G_0	Decomposition into components	Split candidate completed matrix into 9 panels (each panel is an image)
R_0	First-order abstraction	Abstracting from pixel-level encoding to panel-level embeddings via auto-encoding backbone perception module ($M_1 = 1$), with parameters shared across panels
G_1	Grouping of first-order abstractions	Form row-wise and column-wise panel triplets
R_1	Second-order abstraction	Apply $\Phi_{MLP}(x_1, x_2, x_3)$ to triplets, abstracting from panel-level embeddings to ternary tokens representing rows and columns ($M_2 = 1$), with parameters shared across triplets
G_2	Grouping of second-order abstractions	Form all pairwise combinations of ternary tokens
R_2	Third-order abstraction	Compare pairs of ternary tokens using bilinear attention function ($M_3 = 1$)
f	Downstream inference	Use attention scores from R_2 to manipulate tensors of learned parameters (called "symbols") to score candidate completed matrix

This mapping demonstrates that the configuration of ARoN instantiates Recursive Relational Abstraction in a clean and interpretable way, highlighting the role of recursive grouping and abstraction in analogical reasoning tasks.

The SAViR-T model, which achieves the current state-of-the-art performance on the Raven's Progressive Matrix task, also implements the RRA framework:[4]

RRA Component	**Mathematical Role**	**SAViR-T Instantiation**
X_0	A numerical tensor	A complete Raven's Progressive Matrices input with candidate answers (16-dimensional array of image tensors)
G_0	Decomposition into components	Split problem into 16 panels (each panel is an image–8 context panels and 8 candidate answer panels)
R_0	First-order abstraction	Abstracting from pixel-level encoding to K^2 panel-level embeddings (called "tokens") of dimension D via backbone network and visual transformer ($M_1 = K^2$), with parameters shared between all panels
G_1	Grouping of first-order abstractions	For each of K^2 sets of 16 tokens representing distinct panels, form row-wise and column-wise token triplets, where the last 8 triplets correspond to the final row/column completed by one of the candidate answer panels
R_1	Second-order abstraction	Apply $\Phi_{MLP}(x_1, x_2, x_3)$ to triplets, abstracting from panel-level tokens to ternary tokens representing "rules" in rows/columns ($M_2 = 1$), with parameters shared between all triplets
G_2	Grouping of second-order abstractions	In parallel across the token dimension, form all pairwise combinations of row/column rule tokens
R_2	Third-order abstraction	Apply $\Psi_{MLP}([t_1; t_2])$ to all pairs of row/column rule tokens to extract "shared rules" ($M_3 = 1$), with parameters shared between all row/column rule tokens
f	Downstream inference	Use "shared rules" from R_2 (mean-pooling across the token dimension) to calculate similarity metric with "principal shared rule"; select candidate answer with the highest similarity score

The following table illustrates how the core components of the WReN model, one of the earliest deep learning attempts at solving the RPM task, instantiate the RRA framework: [5]

RRA Component	**Mathematical Role**	**WReN Instantiation**
X_0	A numerical tensor	A complete Raven's Progressive Matrices input (3x3 visual grid), where the bottom-right panel is one of the candidate answers to the reasoning problem
G_0	Decomposition into components	Split candidate completed matrix into 9 panels (each panel is an image)
R_0	First-order abstraction	Abstracting from pixel-level encoding to panel-level embeddings via CNN ($M_1 = 1$), with parameters shared across panels
G_1	Grouping of first-order abstractions	Form all pairs of panel embeddings
R_1	Second-order abstraction	Apply g_θ to pairs, abstracting from panel-level embeddings to tokens representing pairs ($M_2 = 1$), with parameters shared across all pairs
G_2	Grouping of second-order abstractions	Select all second-order abstractions (K_2=1, $D_2 = \binom{9}{2}$)
R_2	Third-order abstraction	Sum and pass to f_ϕ to extract score ($M_3 = 1$)
f	Downstream inference	Vacuously satisfied by the identity function

References

[1] Taylor W. Webb, Steven M. Frankland, Awni Altabaa, Simon Segert, Kamesh Krishnamurthy, Declan Campbell, Jacob Russin, Tyler Giallanza, Randall O'Reilly, John Lafferty, and Jonathan D. Cohen. The relational bottleneck as an inductive bias for efficient abstraction. *Trends in Cognitive Sciences*, 28(9):829–843, 2024. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2024.04.001. URL `https://www.sciencedirect.com/science/article/pii/S1364661324000809`.

[2] Awni Altabaa, Taylor Webb, Jonathan D. Cohen, and John Lafferty. Abstractors and relational cross-attention: An inductive bias for explicit relational reasoning in transformers. 2024. Publisher Copyright: © 2024 12th International Conference on Learning Representations, ICLR 2024. All rights reserved.; 12th International Conference on Learning Representations, ICLR 2024 ; Conference date: 07-05-2024 Through 11-05-2024.

[3] Taylor W. Webb, Zachary Dulberg, Steven M. Frankland, Alexander A. Petrov, Randall C. O'Reilly, and Jonathan D. Cohen. Learning representations that support extrapolation. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[4] Pritish Sahu, Kalliopi Basioti, and Vladimir Pavlovic. Savir-t: Spatially attentive visual reasoning with transformers. In Massih-Reza Amini, Stéphane Canu, Asja Fischer, Tias Guns, Petra Kralj Novak, and Grigorios Tsoumakas, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 460–476, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-26409-2.

[5] Adam Santoro, Felix Hill, David G. T. Barrett, Ari S. Morcos, and Timothy P. Lillicrap. Measuring abstract reasoning in neural networks. *ArXiv*, abs/1807.04225, 2018. URL `https://api.semanticscholar.org/CorpusID:49665167`.