This thesis has been submitted in fulfilment of the requirements for a postgraduate degree (e. g. PhD, MPhil, DClinPsychol) at the University of Edinburgh. Please note the following terms and conditions of use:

- This work is protected by copyright and other intellectual property rights, which are retained by the thesis author, unless otherwise stated.
- A copy can be downloaded for personal non-commercial research or study, without prior permission or charge.
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author.
- The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.
- When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.
Relation Learning and Reasoning in Computational Models of High Level Cognition

Guillermo Antonio Puebla Ramírez

Doctor of Philosophy

THE UNIVERSITY OF EDINBURGH

2022
Abstract

Relational reasoning is central to many cognitive processes, ranging from “lower” processes like object recognition to “higher” processes such as analogy-making and sequential decision-making. The first chapter of this thesis gives an overview of relational reasoning and the computational demands that it imposes on a system that performs relational reasoning. These demands are characterized in terms of the binding problem in neural networks. There has been a longstanding debate in the literature regarding whether neural network models of cognition are, in principle, capable of relation-base processing. In the second chapter I investigated the relational reasoning capabilities of the Story Gestalt model (St. John, 1992), a classic connectionist model of text comprehension, and a Seq-to-Seq model, a deep neural network of text processing (Bahdanau, Cho, & Bengio, 2015). In both cases I found that the purportedly relational behavior of the models was explainable by the statistics of their training datasets. We propose that both models fail at relational processing because of the binding problem in neural networks. In the third chapter of this thesis, I present an updated version of the DORA architecture (Doumas, Hummel, & Sandhofer, 2008), a symbolic-connectionist model of relation learning and inference that uses temporal synchrony to solve the binding problem. We use this model to perform relational policy transfer between two Atari games. Finally, in the fourth chapter I present a model of relational reinforcement that is able to select relevant relations, from a potentially large pool of applicable relations, to characterize a problem and learn simple rules from the reward signal, helping to bridge the gap between reinforcement learning and relational reasoning.
Nowadays, artificial intelligence is everywhere. For example, when one goes to the Internet and asks Google a question, there is an artificial intelligence system that generates the answer that we see on the screen. If we want to interact with artificial intelligence systems that we can understand and therefore trust, we need to build systems that think like humans do. Relational reasoning is the ability to think in terms of the relations between objects instead of in terms of just those objects. For example, if I see a person sitting on a chair, I can recognize that the person is on top of the chair or that the person is taller than the chair. This ability, that seems quite natural to people, is very hard to achieve in the kind of artificial intelligence systems which we interact with every day. In this thesis, I study to what extent current artificial intelligence systems can reason in terms of relations and how we can build models, based on psychology, that are naturally suited for relational reasoning. Furthermore, I show how a model that excels at this kind of reasoning can use it in order to interact effectively with its environment.
Acknowledgements

Thanks to Drew and Ella for being the best.
## Contents

Abstract ii

Lay Summary iii

Acknowledgements iv

Figures and Tables viii

1 Introduction 1
1.1 Relational Reasoning 1
1.2 The Binding Problem and Neural Synchrony 2
1.3 Relational Representations and Reinforcement Learning 4
1.4 Publications and Contributions 5

2 The Relational Processing Limits of Classic and Contemporary Neural Network Models of Language processing 6
2.1 Introduction 6
2.2 Models 10
2.2.1 Story Gestalt model 10
2.2.2 Sequence-to-Sequence with Attention model 12
2.3 Simulation 1 14
2.3.1 Training 16
2.3.2 Results 17
2.4 Simulation 2 19
2.4.1 Training 20
2.4.2 Results 21
2.5 General discussion 22

3 A Theory of Relation Learning and Cross-domain Generalization 26
3.1 Introduction 26
3.1.1 Relational Representations and Generalization 28
3.1.2 Learning Relational Representations 29
3.1.3 A Theory of Cross-domain Generalization 33
3.2 The Model 36
3.2.1 Representing Relational Knowledge: LISAese 37
3.2.2 Computational Macrostructure 38
3.2.3 Operation 39
## CONTENTS

5.2 Analogy Making, Relation Learning and Transfer ........................................ 107  
5.3 Relational Reinforcement Learning and Analogy Making .......................... 108  
5.4 Conclusion ................................................................................................... 109  

### Appendices

**A** Supplementary Materials Chapter 2 .................................................................. 110  
**B** DORA Computational Details ........................................................................ 114  
- B.1 Parts of DORA ................................................................................................. 114  
- B.2 Functional Overview of Processing in DORA .................................................. 115  
  - B.2.1 Retrieval ........................................................................................................ 115  
  - B.2.2 Mapping ......................................................................................................... 115  
  - B.2.3 Relation learning ........................................................................................... 116  
  - B.2.4 Refinement .................................................................................................... 116  
  - B.2.5 Relational Generalization ............................................................................. 117  
- B.3 Processing in DORA .......................................................................................... 117  
**C** Simulation Details Chapter 3 .......................................................................... 128  
- C.1 DORA: Learning representations from screens ............................................ 128  
- C.2 DORA: Q-learning for Game Play ................................................................. 129  
- C.3 Deep Q-Network .............................................................................................. 129  
- C.4 Supervised Deep Neural Network .................................................................... 130  
- C.5 Graph Neural Network ..................................................................................... 130  
**D** Supplemental Results Chapter 3 .................................................................... 132  
- D.1 Supplemental simulation 1: Pong to Breakout generalization results ............ 132  
- D.2 Supplemental data 1 ......................................................................................... 132  
- D.3 Supplemental simulation 2: Inverse Breakout ................................................ 134  
**E** State Representations Chapter 4 ...................................................................... 136  

Bibliography ............................................................................................................. 140
Figures and Tables

Figures

2.1 Story Gestalt model. (A) An example of a proposition as represented in the input layer. (B) Model architecture. (C) Model's operation unfolded over time. See text for details. ................................................................. 11

2.2 Seq2Seq model with attention. (A) Input representation. (B) Model's architecture unfolded over time. (C) Attention mechanism. See text for details. ................................. 12

2.3 Training Process for the Seq2seq+Attention Model. See text for details. ............. 14

2.4 Accuracy per test and model. Both models perform well in the baseline condition. Furthermore, their performance was affected differentially in our critical conditions. The Story Gestalt model was more susceptible to the concept violation and correlation violation manipulations while the Seq2Seq+Attention model was more susceptible to the correlation violation and shuffled propositions manipulations. As none of these manipulations changed the relational structure of the task, these results suggest that neither model was able to capture it during training. Error bars are 95% confidence intervals. ................................................................. 17

2.5 Accuracy per condition, role and model. Both models perform reasonably well in the baseline condition for all roles. For both models there is a significant drop in accuracy when the agent-1 and agent-2 roles are filled with concepts different than those used in training (relational condition). For the correlation-violating filler the drop in accuracy is more pronounced for the Seq2Seq+Attention model although is still appreciable in the SG model. As in Simulation 2 we probed for relational roles directly, these results strengthen the conclusion that neither model has grasped the relational structure of the task. Error bars are 95% confidence intervals. ................................................................. 21
3.1 Knowledge representation and time-based binding in DORA. (A) Representation of a single proposition (*above*(ball, paddle)) in DORA. Feature units represent properties of objects and relational roles in a distributed manner. Token units in T1 represent objects and roles in a localist fashion; token units in T2 conjunctively bind roles to their arguments (e.g., objects); token units in T3 conjunctively link role-argument pairs into multi-place relations. Colors of units in (A) correspond to the time of a unit’s firing, as indicated in (B). (B) A time-series illustration of the activation of the units illustrated in (A). Each graph corresponds to one unit in (A) (i.e., the unit with the same name and color as the graph). The abscissa of the graph represents time, and the ordinate represents the corresponding unit’s activation. (C) Time-based binding illustrated as a sequence of discrete frames (i...iv). (i) Units encoding *higher-than-something* fire. (ii) Units encoding ball fire. (iii) Units encoding *lower-than-something* fire. (iv) Units encoding paddle fire. Labels in units indicate what the unit encodes (see key). The labels on the units are provided for clarity and are meaningless to DORA. . . . . . . . . . . . . . . 37

3.2 Macrostructure of DORA. (A) DORA’s long-term-memory (LTM), consisting of layers of token units (black rectangles), and the feature units connected to the bottom layer of LTM. During processing, some units in LTM enter active memory (AM). (B) Expanded view of AM. AM is composed of two sets, the driver (the current focus of attention) and the recipient (the content of working memory available for immediate processing). Black lines indicate bidirectional excitatory connections. . 39
3.3 Representation learning in DORA. (A) Learning a single-place predicate representation by comparing two objects. (i) A representation of a ball in the driver is mapped (red double-headed arrow) to a different representation of a ball (e.g., from a different game screen) in the recipient. (ii) The representation of the ball in the driver activates the mapped unit in the recipient (through shared features and mapping connection); as units pass activation to their features, shared features become more active (red units) than unshared features (pink units). (iii) Units in T1 and T2 are recruited (activation clamped to 1; blue units) in the recipient, and weighted connections are learned via Hebbian learning (i.e., stronger connections between more active units). (iv) The result is an explicit representation of the featural overlap of the two balls—in this case the property of being higher-than-something (see main text)—that can be bound to an argument (as in Figure 1). (B) Learning a multi-place relational representation by linking a co-occurring set of role-argument pairs. (i) A representation of a ball that is higher-than-something and a paddle that is lower-than-something is mapped to a different representation of a paddle that is higher-than-something, and a ball that is lower-than-something (e.g., from a different game screen). (ii) When the representation of higher-than-something(ball) becomes active in the driver it activates mapped units in the recipient; a T3 unit is recruited (activation clamped to 1; blue unit) in the recipient and learns weighted connections to units in T2 via Hebbian learning. (iii) When the representation of lower-than-something(paddle) becomes active in the driver, it activates corresponding mapped units in the recipient; the active T3 unit learns weighted connections to T2 units. (iv) The result of learning is a LISAese representation of the relational proposition above(ball, paddle) (see also Figure 3.1).

3.4 (A) The relational invariance circuit. (B) (i) Activation flows from clamped feature units encoding a dimension or property to T1 units. (ii) T1 units compete via lateral inhibition to respond to the active feature units. (iii) T1 units activate proxy units, which feed activation to E units. E units pass activation to a subset of feature units and connections between active feature units and T1 units are updated via Hebbian learning. (iv) The active T1 unit is inhibited to inactivity by its inhibitor (blue square). (v-vi) The process repeats for the second active T1 unit.
3.5 Graphical depiction of analogical inference in DORA. The representation of the right-of(ball, paddle1), and right-of(paddle2, paddle1) in the driver maps to the representation of above(paddle2, paddle1) in the recipient (red double-arrowed lines indicate mappings). As the representation of right-of(ball, paddle1) becomes active in the driver, some active units have nothing to map to in recipient (i.e., the units representing ball, more-x+ball, less-x+paddle1, and right-of(ball, paddle1)). DORA recruits and activates units to match the unmapped driver units (blue units indicate recruited units; blue double-headed arrow indicates matched units). DORA learns connections between co-active token units in the recipient (heavier blue lines). The end result is a representation of the situation: above(ball, paddle1) & above(paddle2, paddle1) in the recipient.

3.6 Graphical depiction of generalizing known relations to new situations using the analogical inference algorithm in DORA. The representation of the above(object-X, object-Y) in the driver is maps to the representation of object-1, object-2 in the recipient (red double-arrowed lines indicate mappings). As the representations of higher(object-X) and lower(object-Y) become active in the driver, some active units have nothing to map to in recipient (e.g., the driver units representing higher and higher+object-X, and the units representing lower and lower+object-Y). DORA recruits and activates units to match the unmapped driver units (blue units indicate recruited units; blue double-headed arrow indicates matched units). DORA learns connections between co-active token units in the recipient (heavier blue lines). The end result is a representation of the situation: above(object-1, object-2).

3.7 Examples of the shape stimuli used for Simulation 1.

3.8 Mean relational selectivity of T1 units (as defined in text) as a function of number of training examples, simulation 1a.

3.9 The number of representations of single-place predicates, meaningful relations, and non-sense relational structures in DORA’s LTM after each 100 learning trails, simulation 1a.

3.10 Mean relational selectivity of T1 units (as defined in text) as a function of number of training examples, simulation 1b.

3.11 The number of representations of single-place predicates, meaningful relations, and non-sense relational structures in DORA’s LTM after each 100 learning trails, simulation 1b.
3.12 Results of game play simulations with DORA, the DQNs and the DNNs. Error bars represent 2 stderrors. (A) Performance humans and networks on Breakout as an average of 100 test games. (B) Results of networks playing Pong after training on Breakout as score on the first game played and mean score over the first 100 games played. (C) Results of networks when returning to play Breakout after playing or learning to play Pong as an average of the first 100 games played. 67

3.13 Results of game play performance for DORA from simulation 1 and after CLVR learning (simulation 2; blue and light blue bars), and DORA after ablation (simulation 4; green and light green bars). Error bars represent 2 stderrors. (A) Performance of DORA on Breakout as an average of 100 test games. (B) Results of DORA playing Pong after training on Breakout as the score of the first game played and an average score of the first 100 games played. (C) Results of DORA when returning to play Breakout after playing or learning to play Pong, as an average score for the first 100 games played. 70

3.14 Results of simulation of (Nelson & Benedict, 1974). 76

3.15 A recreated example of the stimuli used in (Rattermann & Gentner, 1998). 77

3.16 Results of simulation of Rattermann and Gentner (1998). (a) Performance of children and DORA on sparse trials. (b) Performance of children and DORA on rich trials. 79

4.1 Two different ways of representing state-action values. (a) A state of the Breakout environment where the player is to the left of the ball. (b) A logical relational regression tree representing the fact that the value of the action RIGHT is higher when the player is to the left of the ball, lower when the player and the ball are at the same x-coordinate, and lowest when the the player is to the right of the ball. (c) A comparative relational regression tree representation of the same fact. See text for details. 91

4.2 Cumulative return by model version and run on Breakout. 95

4.3 Mean returns by model version and run on 100 test episodes of Breakout. Error bars are 95% confidence intervals. The horizontal shadowed line represents the mean return and 95% confidence intervals for a random agent on 100 test episodes. 95

4.4 Best state-action trees on Breakout for the logical (a) and comparative (b) versions of RRTL. 96

4.5 If-then rule corresponding to the best state-action tree on Breakout. 96

4.6 A typical state of the Pong environment. 97

4.7 Cumulative return by model version and run on Pong. 98

4.8 Mean returns by model version and run on 100 test episodes of Pong. Error bars are 95% confidence intervals. The horizontal shadowed line represents the mean return and 95% confidence intervals for a random agent on 100 test episodes. 99
4.9 State-action trees on Pong for (a) the best run, and (b) the worst run. Both are runs from the comparative version of RRTL. The state-action tree of the worst run in (b) is based on the last training iteration. 99
4.10 If-then rule corresponding to the best state-action tree on Pong. 99
4.11 Two typical states of the Demon Attack environment. On state (A) there are big enemies that can shut missiles. On state (B) besides big enemies there are small enemies. The player looses a life if it is touched by a small enemy. 101
4.12 Cumulative return by model version and run on Demon Attack. 102
4.13 Mean returns by model version and run on 100 test episodes of Demon Attack. Error bars are 95% confidence intervals. The horizontal shadowed red line represents the mean return and 95% confidence intervals for a random agent on 100 test episodes. The horizontal shadowed pink line shows the same information for a random agent that only has access to the “fire” actions. 103
4.14 State-action trees on Demon Attack for (a) the best run, and (b) the worst run. Both are runs from the comparative version of RRTL. The state-action tree of the worst run in (b) is based on the last training iteration. 103
4.15 If-then rule corresponding to the best state-action tree on Demon Attack. 103

B.1 Pseudocode of processing in DORA. 118

D.1 Results of simulations with DORA trained on Pong and generalizing to Breakout, with DORA learning representations from Pong. Error bars represent 2 stderrors. (A) Performance of DORA on Breakout as an average of 100 test games. (B) Results of DORA playing Breakout after training on Pong as the score of the first game played and an average score of the first 100 games played. (C) Results of DORA when returning to play Pong, as an average score for the first 100 games played. 132

D.2 Figure S2. Results of game play simulations with Humans, DORA, the DQNs and the DNNs. Error bars represent 2 stderrors. (A) Performance humans and networks on Breakout as an average of 100 test games. (B) Results of humans and networks playing Pong after training on Breakout as score on the first game played and mean score over the first 100 games played. (C) Results of humans and networks when returning to play Breakout after playing or learning to play Pong as an average of the first 100 games played. 133
Tables

2.1 Restaurant Script. ................................................................. 9
2.2 Example of a Baseline Story (Restaurant). ............................... 10
2.3 Example of a Concept Violation Story (Restaurant). Lois and Albert were restric-
ted from instances of the Restaurant script during training. .................... 15
2.4 Example of a Correlation Violation Story (Restaurant). ................... 15
2.5 Example of a Shuffled Propositions Story (Restaurant). .................... 16
2.6 Example of a Relational Probe Story (Restaurant). ....................... 20
3.1 Core theoretical claims and their instantiation in DORA ................. 35
4.1 Breakout State Representation ............................................. 93
A.1 Concepts used in all the scripts. ............................................ 110
A.2 Park Script. ................................................................. 111
A.3 Airport Script. .............................................................. 111
A.4 Bar Script. .................................................................. 112
A.5 Beach Script. ............................................................... 113
E.1 Pong State Representation. ................................................... 137
E.2 Demon Attack State Representation ....................................... 138
E.2 Demon Attack State Representation (continued). ....................... 139
1.1 Relational Reasoning

Relational reasoning is defined by the ability to make inferences based on the relations between objects instead of making inferences based on the features of those objects (Holyoak, 2012). The capacity for relational reasoning seems to be a unique human ability dependent on the operation of the prefrontal cortex (Holyoak & Lu, 2021; Krawczyk, 2012; Penn, Holyoak, & Povinelli, 2008; Waltz et al., 1999). Indeed, even the most basic relational skill, judging whether two given objects are the same or different, is far more developed in humans and chimpanzees in comparison to other species (Gentner, Shao, Simms, & Hespos, 2021), develops during infancy in humans (Ferry, Hespos, & Gentner, 2015), and is beyond the capabilities of contemporary deep neural networks trained on image classification tasks (Puebla & Bowers, 2021).

Relational reasoning pervades human cognition from “low” to “high”-level processes (Hafri & Firestone, 2021). For example, to recognize objects humans rely heavily on the relationships between their constituent parts (Biederman, 1987). This results in robust object recognition insensitive to low-level (i.e., non-relational) distortion of the stimuli images, which stands in stark contrast to modern deep neural networks for object recognition (e.g., Geirhos et al., 2019; Jo & Bengio, 2017). Relation-based reasoning also plays a major role in scene perception (e.g., Green & Hummel, 2006), language comprehension (e.g., Martin, 2020; Martin & Doumas, 2017), generalization of concepts (e.g., Goldwater, Don, Krusche, & Livesey, 2018) and the development of conceptual information (e.g., Gentner, 2005), and many other processes.

Analogy-making is a high-level form of relational reasoning, where the reasoner considers two domains in terms of their defining relations and establishes correspondences between the elements across domains based on the similarity of the functional (i.e., objects participating on specific relations) structures (Holyoak, 2012). This allows, for example, to “fill in” missing information in one domain based on its relational similarity with another domain (a.k.a., analogical inference, for a detailed exposition see Chapter 3) or to form schemas that abstract away the non-relational content of the domain (a.k.a., relational categories). A good example of these processes are the classic studies of Gick and Holyoak (1980,
1.1. Relational Reasoning

In Gick and Holyoak (1980) participants read a story about the strategy used by a general to capture a fortress. Because there was not a single road to the fortress that would allow his army to pass all at once, he opted for dividing his forces into smaller groups and sending them simultaneously through several roads. Then, they joined forces and captured the fortress. When participants read this story before trying to solve a medical problem, where they needed to come up with a strategy to use rays to destroy a tumour without injuring the patient's healthy tissue, they generated a “convergence” solution analogous to the general's strategy. Furthermore, Gick and Holyoak (1983) showed that, when asked to write a comparison between the general story to an analogous story before trying to solve the medical problem, participants' comparisons reflected a “convergence” schema, with the quality of this schema correlating with the generation of a convergence solution.

Analogy-making requires that relations are represented explicitly and independently of the objects participating in the relation, so that they can be compared across domains. Note also that, for this process to be computationally efficient, the relational similarity between relations across domains should exploit the semantic similarity among relations. Otherwise, this matching process would have to rely on exhaustively enumerate all possible relation correspondences and identify the best match afterwards (as in the Structure Mapping Engine (SME), Falkenhainer, Forbus, & Gentner, 1989).

A major question addressed in this thesis is how a neural system is able to represent—and learn to represent—relations in such a way that allows to generalize based on relational content and make analogical inferences. To understand the complexity of this issue, the next section reviews the binding problem, which revolves around the computational requisites to achieve relational processing in neural networks.

1.2 The Binding Problem and Neural Synchrony

At the most general level of description, the binding problem concerns the capacity of a neural system—natural or artificial—to dynamically integrate the information distributed throughout the system into meaningful entities and relations between them that can be flexibly used for reasoning (for reviews, see Greff, van Steenkiste, & Schmidhuber, 2020; Treisman, 1999; von der Malsburg, 2021). To understand the computational problems posed by the binding problem I use the functional division proposed by Greff et al. (2020, cf. Treisman, 1999). This analysis distinguishes three sub-problems that a neural system needs to solve in order to bind
1.2. The Binding Problem and Neural Synchrony

information in the sense aforementioned: (a) segregation, isolating meaningful entities from raw inputs; (b) representation, to maintain this separation at the representational level; and (c) composition, to use these representations for inference, prediction and behaviour. This thesis concentrates on the problems of representation and composition\(^1\).

Traditional feed-forward neural networks, including modern convolutional neural networks (CNNs) and recurrent neural networks, have a fixed architecture that does not even allow to maintain the representation of individual objects separated during processing. For example, when training a network to identify whether two objects are the same or not, one can use a Siamese Network (Bromley, Guyon, LeCun, Säckinger, & Shah, 1993), an architecture that assumes the solution to the segregation problem by using separated channels to process each object. This network, however, will not be able to maintain the segregation of the objects at the representational level, because the internal representations will necessarily be melted into a single vector when the channels meet to produce the final output. This results in a pattern of generalization that is feature based: instead of generalizing based on the trained relation (same/different) the network will generalize based on the feature-wise similarity of the training examples with the test examples (Kim, Ricci, & Serre, 2018; Puebla & Bowers, 2021). This problem is even worse if a multi-object image is processed through a single channel, as in modern CNNs used for object recognition. In this case, the network suffers from a problem know as the superposition catastrophe (Malsburg, 1986), in which the representation of, for example, a red apple and a green pear becomes indistinguishable from the representation of a green apple and a red pear (Greff et al., 2020). This kind of situation leads networks to learn conjunctive codes (e.g., a unit that activates only for “red apple and green pear”, and another unit that activates only for “green apple and red pear”; Bowers, Vankov, Damian, & Davis, 2014). This same conjunctive coding strategy can be applied to relations and objects, which results in representations that do not respect the independence of relations and objects (e.g., a unit that codes for “cup above table”, and another that codes for “table above cup”; Hummel, 2011). Chapter 2 explores this representational limitation of traditional neural networks in the context of neural networks models of language processing.

While Chapter 2 concentrates on the limitations of traditional neural networks regarding relational processing, Chapter 3 presents an updated version of the DORA model of relation learning and analogical reasoning (Doumas et al., 2008). DORA is a symbolic-connectionist model that uses a solution to the representation and composition problems known as neural synchrony (Milner, 1974; von der Malsburg, 1981). Neural synchrony is a mechanism for dynamic binding that becomes available in oscillatory networks that fire at a regular frequency. In neural synchrony representations of objects and relations that are bound together fire together (Hummel & Holyoak, 1997). Importantly, these bindings can easily be undone by desynchron-
izing the corresponding representations. Because relations are represented independently from the objects that instantiate them, and relations are represented in a distributed fashion, they can be compared and matched based on their semantic similarity. This allows DORA to compare domains in terms of their functional structures. Among other results, Chapter 3 shows that through analogical inference DORA can transfer a relational policy learned in the game Breakout to the game Pong zero-shot.

1.3 Relational Representations and Reinforcement Learning

In Chapter 3, to achieve relational policy transfer, DORA used tabular q-learning (Watkins, 1989) to learn the source policy. However, the problem of selecting which of all possible relations available to the agent to characterize the environment was assumed as solved. Q-learning is a simple form of reinforcement learning (RL) where the agent interacts with the environment receiving rewards and taking actions. In general terms, q-learning learns an approximation of the action-value function (i.e., the expected cumulative reward for each action in each state) by predicting a value for each state-action combination and updating this prediction to match the received rewards during the interaction with the environment. Although tabular q-learning is guaranteed to converge to the optimal action-value function under some conditions (see Sutton & Barto, 2018), its convergence time becomes prohibitive as the size of the state space grows. This is critical for an agent that perceives the environment in terms of relations between objects, because for any pair of objects there is a myriad of relations that hold between them. Therefore, Chapter 4 concentrates on the problem of learning a relational policy when many relations are available to describe the state of the environment.

In order to deal with the size of the state space, RL algorithms use function approximators to make the search space tractable. Unfortunately, the most common kind of function approximator in RL, deep neural networks, are not well suited to deal with relational representations as defined earlier, and therefore could not form the basis for analogical policy transfer. Furthermore, a relational representation of the state induces discrete partitions the state space, which is hard to approximate for continuous functions such as neural networks (but see Jiang & Luo, 2019; Zimmer et al., 2021, for research efforts in this direction). To tackle this problem, chapter 4 uses a method developed in the field of relational reinforcement learning (Džeroski, De Raedt, & Driessens, 2001). This function approximator is a relational regression tree (Driessens, Ramon, & Blockeel, 2001), that incrementally splits the state-action space according to the relations potentially relevant to the policy. In chapter 4, I show that, with some modifications, this method is able to learn incrementally relational policies in three Atari games: Breakout, Pong and Demon Attack. Furthermore, I discuss the limitations of the approach and future directions of research.
1.4 Publications and Contributions

The main chapters of this thesis have been published or submitted for publication. This section lists the relevant publication or preprint corresponding to each chapter as well as the authors' contributions according to the Contributor Roles Taxonomy: https://onlinelibrary.wiley.com/doi/full/10.1002/leap.1210

Chapter 2 was published as a journal article in:


**Guillermo Puebla:** Conceptualization, Investigation, Formal analysis, Writing- Original draft preparation, Writing- Reviewing and Editing. **Andrea E. Martin:** Supervision, Writing- Reviewing and Editing. **Leonidas A. A. Doumas:** Supervision, Conceptualization, Writing- Reviewing and Editing.

Chapter 3 was published as a journal article in:


**Leonidas A. A. Doumas:** Supervision; Conceptualization; Investigation; Formal analysis; Writing- Original draft preparation; Writing- Reviewing and Editing. **Guillermo Puebla:** Conceptualization; Investigation; Formal analysis; Writing- Original draft preparation; Writing- Reviewing and Editing. **Andrea E. Martin:** Supervision; Writing- Original draft preparation; Writing- Reviewing and Editing; Resources. **John E. Hummel:** Writing- Original draft preparation, Writing- Reviewing and Editing.

Chapter 4 has been submitted to a journal. A preprint is available:


**Guillermo Puebla:** Conceptualization, Investigation, Formal analysis, Writing- Original draft preparation, Writing- Reviewing and Editing. **Leonidas A. A. Doumas:** Supervision, Writing- Reviewing and Editing.
Chapter 2

The Relational Processing Limits of Classic and Contemporary Neural Network Models of Language processing

2.1 Introduction

The ability to represent and reason in terms of the relations between objects plays a crucial role across many aspects of human cognition, from visual perception (Biederman, 1987), to higher cognitive processes such as analogy (Holyoak, 2012), categorization (Medin, Goldstone, & Gentner, 1993), concept learning (Doumas & Hummel, 2013), and language (Gentner, 2016). Furthermore, comparative evidence suggests that relational thinking may be the key cognitive process distinguishing the abilities of humans from those of other species (Christie & Gentner, 2010; Penn et al., 2008). Given the relevance of the capacity to represent and reason about relations across cognitive domains, several computational models in cognitive science have sought to capture its main characteristics and development (e.g., D. Chen, Lu, & Holyoak, 2017; Doumas et al., 2008; Falkenhainer et al., 1989; Halford, Wilson, & Phillips, 1998; Hummel & Holyoak, 1997, 2003; Kollias & McClelland, 2013; Leech, Mareschal, & Cooper, 2008; Lu, Chen, & Holyoak, 2012; Lu, Wu, & Holyoak, 2019; Van der Velde & De Kamps, 2006).

Computational models of relational thinking differ in their representational assumptions. In the canonical view, relational thinking entails using predicate representations. A predicate is an abstract structure that can be dynamically bound to an argument, specifying a set of properties about that argument (Doumas & Hummel, 2005). For example, $\text{predator}(x)$ specifies a series of properties about the variable $x$ (e.g., carnivore, hunts, etc.). Predicate representations have two main attributes. In the first place, predicates maintain role-filler independence in that at least some aspect of the semantic content of the predicate is invariant with respect to its arguments. For example, $\text{predator}(\text{fox})$ and $\text{predator}(\text{lynx})$ will specify the
same set of properties (e.g., carnivore, hunts, etc.) about the objects fox and lynx. In the second place, predicates can be dynamically bound to arguments, namely, fillers can be assigned and reassigned to different roles as needed during processing. That predicates can be dynamically bound to arguments allows a given concept to play different roles at different times or in different situations. For example, in a scene where a fox is preying on a hen, but then a lynx comes and eats the fox, the initial binding of fox to predator (i.e., predator(fox)) is easily broken and new binding of fox to prey (i.e., prey(fox)) is easily formed. Models based on predicates or formally equivalent systems (i.e., systems that perform dynamic binding of independent representations of roles and fillers, or symbolic systems) successfully account for a wide variety of phenomena in the relational thinking literature (for a review see Forbus, Liang, & Rabkina, 2017).

By contrast, traditional Parallel Distributed Processing (PDP) models explicitly eschew the need for structured representations (see, e.g., Rogers & McClelland, 2014). Representations in a PDP model correspond to patterns of activation across a fixed-size layer of units (i.e., an activation vector). These representations are unstructured in the sense that relational roles and objects are not independently represented, but instead are represented simultaneously as a single entity. PDP approaches to relational thinking seek to obtain relational behavior without invoking symbolic machinery (Kollias & McClelland, 2013; Leech et al., 2008; St. John, 1992; St. John & McClelland, 1990; Yuan, 2017). The reasoning behind these models is that if a traditional PDP model successfully performs some relational reasoning task, then predicates are not strictly necessary for that task, and, by extension, might not actually be accurate approximations of human mental representations. Recently, some researchers have argued that PDP models are capable of handling relational knowledge. For example, Rogers and McClelland (2008, 2014) have proposed that the gestalt models of text comprehension (Rabovsky, Hansen, & McClelland, 2018; Rabovsky & McClelland, 2020; Rohde, 2002; St. John, 1992; St. John & McClelland, 1990) exhibit successful effective role-to filler binding. The evidence presented by these models consists invariably of demonstrations of generalizations to “unseen” sentences. However, as is going to be clear in the simulations of the present work, these “unseen” sentences consist typically of known combinations of roles and concept fillers, which allows these models to succeed in the generalization tests by memorizing combinations of roles and fillers in the training dataset. As to which specific mechanism would allow these models to learn to form role-filler bindings, these researchers usually appeal to the concept of emergence, arguing that domain general learning algorithms such as back-propagation, in conjunction with the distributed nature of the internal representations of PDP models, allows for learning open-ended relations (Rogers & McClelland, 2014).
2.1. Introduction

Some of the optimism in the connectionist literature is based, at least partially, on the achievements of deep learning architectures in natural language processing. For example, Rabovsky et al. (2018) argue that the success of Google’s neural machine translation system (Wu et al., 2016) implies that structured representations are, in fact, an obstacle to accurately capturing the subtle regularities of human language (also see Rabovsky & McClelland, 2020). In the present study, we tested the Story Gestalt (SG) model (St. John, 1992) and a Sequence-to-Sequence with Attention (Seq2Seq+Attention) model (Bahdanau et al., 2015)—the architecture behind Google’s neural machine translation system—in a series of tasks requiring binding a number of concepts to several roles in a story. All stories had relational structure in the sense that (1) the thematic roles were organized in specific ways and (2) filling the roles with different concepts yielded different instantiations of the story. In our simulations we trained both models on a large number of these stories to answer questions about the stories and then tested the models with new (untrained) stories. Importantly, we maintained the relational structure of the test stories relative to the training stories while varying their statistical structure (by changing the stories’ typical role fillers) in several ways. Next, we describe the generalities of our task and each model's operation in detail.

Our task, based on the original materials of St. John (1992), consists on answering questions about stories generated by a series of (5) scripts. All the scripts describe events as a sequence of propositions where several concepts play different thematic roles: agent-1, agent-2, topic, patient-theme, recipient-destination, location, manner and attribute. As an illustrative example, consider the Restaurant script (Table 2.1). This script describes an event where two people go to a restaurant. Each sentence of the Restaurant script defines fillers for some roles. To generate a specific instance of a Restaurant script (i.e., a Restaurant story) the roles are given values corresponding to specific concepts. Table 2 (column 1) presents an example of an instantiated Restaurant story in a pseudo-natural language format. The first sentence of this story corresponds to the proposition: agent-1 = Anne, agent-2 = Gary, topic = decided-to-go, patient-theme = None, recipient-destination = restaurant, location = None, manner = None, attribute = None. Appendix A presents all possible concepts values for each role. Note that our scripts produce stories with no repeated topic concepts across propositions.

Each script implements a tree structure where each node represents a proposition and each branch of the tree represents a story. The scripts also implement rules that specify the probability of transitioning from one node to another conditioned on the value of a character or location role. For example, a rule in the Restaurant script (see Table 2.1) specifies that if the restaurant is expensive, it will be located far away.

We trained the models in two different conditions. In the concept restricted condition, some character or object names were never used in specific scripts. For example, in the Restaurant stories the characters Lois and Albert were never used to fill the roles agent-1 or agent-2 (see Table 2.1; Appendix A presents detailed descriptions of the remaining scripts, their concept
Table 2.1: Restaurant Script.

<table>
<thead>
<tr>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. &lt;agent-1&gt; and &lt;agent-2&gt; decided restaurant</td>
</tr>
<tr>
<td>2. Restaurant quality &lt;expensive/cheap&gt;</td>
</tr>
<tr>
<td>3. Distance to restaurant &lt;far/near&gt;</td>
</tr>
<tr>
<td>4. &lt;agent-1/agent-2&gt; ordered &lt;cheap-wine/expensive-wine&gt;</td>
</tr>
<tr>
<td>5. &lt;agent-1/agent-2&gt; paid bill</td>
</tr>
<tr>
<td>6. &lt;agent-1/agent-2&gt; tipped waiter &lt;big/small/not&gt;</td>
</tr>
<tr>
<td>7. Waiter gave change to &lt;agent-1/agent-2&gt;</td>
</tr>
</tbody>
</table>

Concept restrictions

The roles agent-1 and agent-2 are never ‘Lois’ or ‘Albert’

Deterministic rule

The quality of the restaurant determines the distance completely: expensive $\rightarrow$ far, cheap $\rightarrow$ near

restrictions and rules). In the concept unrestricted condition all concepts were used in all stories. Stories in both conditions were generated according to the following procedure: (1) a script is chosen at random, (2) a sequence of propositions is generated by traversing the probabilistic tree structure of a script and (3) character and vehicles names are given specific values (respecting the script’s deterministic rule and the script's concept restrictions in the concept restricted condition).

To get a criterion for each model's performance we designed a baseline test. In this test we presented the models trained in the unrestricted condition with concept unrestricted stories and asked questions about the stories. The questions corresponded to the concepts filling the topic role. The models generated an answer in the form of a full proposition. The correct answer was the full proposition in which the topic concept was involved. For example, if a proposition in a restaurant story stated that the “waiter gave change to Anne” and the model was asked about the “gave” proposition the correct answer was “waiter gave change to Anne”. Because in our stories there was no repeated topics the correct answer was unequivocal. Table 2.2 presents an example of a Restaurant baseline story, its questions and their corresponding correct answers.
2.2 Models

### 2.2.1 Story Gestalt model

The SG model (St. John, 1992, see Figure 2.1) integrates a sequence of propositions into a distributed representation of a story, which is then used to answer questions about the story. The model represents all propositions in its input layer through 137 localist units coding for each possible filler of each role. For example, there is a unit coding for Albert-agent and another unit coding for Albert-recipient (table A.1 presents these 137 “concepts”). To represent a complete proposition, the units coding for the concept filling each role are activated. For example, a representation of the sentence “Anne and Gary decided to go to the restaurant” would consist of a vector of 137 units were the three units coding for Anne-agent, Gary-agent, decided-topic and restaurant-location are set to 1 and all other units are set to 0 (Figure 2.1A).

Figure 2.1B illustrates the SG model’s architecture. The model is composed of two subsystems. The first “comprehension” subsystem (input proposition, combination and gestalt layers), receives each proposition of a story one at the time as input. The activation in the proposition layer feeds forward to the combination and gestalt layers (100 units each). The gestalt layer has recurrent connections to the combination layer, which allows the model to form a representation of the story presented so far (see Figure 2.1C). The second “query” subsystem (gestalt, question, extraction and output proposition layers), receives as input the activation of the gestalt layer and the question layer. The question layer (34 units) consists of a vector of units representing all topic concepts in a localist fashion. The extraction layer (100 units) combines the activation of the gestalt and question layers and feeds forward to the output layer, which has the same dimensionality as the input layer.

To train a single story the model is presented with increasing longer sequences of the story propositions and, after each successive sequence, is asked about the last proposition. For example, imagine a story composed by the last three propositions of the Restaurant story in Table 2.2 (i.e., “Anne paid bill”, “Anne tipped waiter big”, “waiter gave change to Anne”). This story would be trained by presenting the model with the sequences: ["Anne paid bill"],

### Table 2.2: Example of a Baseline Story (Restaurant).

<table>
<thead>
<tr>
<th>Story</th>
<th>Questions</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. &lt;Anne&gt; and &lt;Gary&gt; decided restaurant</td>
<td>decided</td>
<td>&lt;Anne&gt; and &lt;Gary&gt; decided restaurant</td>
</tr>
<tr>
<td>2. Restaurant quality &lt;expensive&gt;</td>
<td>quality</td>
<td>Restaurant quality &lt;expensive&gt;</td>
</tr>
<tr>
<td>3. Distance to restaurant &lt;far&gt;</td>
<td>distance</td>
<td>Distance to restaurant &lt;far&gt;</td>
</tr>
<tr>
<td>4. &lt;Anne&gt; ordered &lt;cheap-wine&gt;</td>
<td>ordered</td>
<td>&lt;Anne&gt; ordered &lt;cheap-wine&gt;</td>
</tr>
<tr>
<td>5. &lt;Anne&gt; paid bill</td>
<td>paid</td>
<td>&lt;Anne&gt; paid bill</td>
</tr>
<tr>
<td>6. &lt;Anne&gt; tipped waiter &lt;big&gt;</td>
<td>tipped</td>
<td>&lt;Anne&gt; tipped waiter &lt;big&gt;</td>
</tr>
<tr>
<td>7. Waiter gave change to &lt;Anne&gt;</td>
<td>gave</td>
<td>Waiter gave change to &lt;Anne&gt;</td>
</tr>
</tbody>
</table>
Figure 2.1: Story Gestalt model. (A) An example of a proposition as represented in the input layer. (B) Model architecture. (C) Model’s operation unfolded over time. See text for details.

[“Anne paid bill”, “Anne tipped waiter big”] and [“Anne paid bill”, “Anne tipped waiter big”, “waiter gave change to Anne”]. The question for each sequence would be the topic concept of the last proposition of the sequence (i.e., “paid”, “tipped” and “gave”) and the target (i.e., what the model was trained to output) would be the last proposition of each sequence (i.e., “Anne paid bill”, “Anne tipped waiter big”, “waiter gave change to Anne”). The difference between the actual output and the target is used to train the model through a standard gradient descent algorithm. Once trained, the model can recover the full proposition associated with each topic of a story. For example, if a trained SG model is presented with the complete sequence of sentences on Table 2.2 and then asked about the topic “decided” (by activating the corresponding localist unit in the question layer) the model would output an activation vector corresponding to the proposition “Anne and Gary decided to go to the restaurant”.

St. John (1992) showed that the SG model can recover missing sentences from a story, review its predictions as it encounters new propositions and resolve pronouns. For example, if the model is presented with the complete sequence of propositions on Table 2.2 except for the third (“distance to restaurant far”) and is asked about the topic “distance”, the model would output an activation vector corresponding to the proposition “distance to restaurant far” because in its training data expensive restaurants are always far away (see Table 2.1).
2.2. Models

2.2.2 Sequence-to-Sequence with Attention model

In order to test the performance of a contemporary deep learning system on our task, we implemented a version of the Seq2Seq+Attention model (Bahdanau et al., 2015, see Figure 2.2)—a deep neural network architecture designed originally to solve machine translation problems. In translation problems, a source sentence in a given language (e.g., English) has to be translated into a different language (e.g., French). Typically, the source and target sentences have different lengths. In general, a Seq2Seq model consist of an encoder network and a decoder network. Both are recurrent neural networks with their own independent time steps (\( t \) for the encoder and \( t' \) for the decoder in Figure 2.2B). The encoder transforms the input sequence into a sequence of fixed-size vectors and the decoder processes these transformed vectors to get the output sequence. Two important features of this model are the use of Word2Vec representations for the input words (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) and an attention mechanism that allows the model to selectively “attend” to different parts of the encoder’s output (Bahdanau et al., 2015).

Word2Vec embeddings (Mikolov et al., 2013) are dense distributed representations obtained by extracting the activation vector of the hidden layer of shallow neural network trained to predict the surrounding words given an input word in large corpus of text. Word2Vec representations maintain the distributional patterns of similarity between words, such that words used in similar contexts have similar representations (however, see Nematzadeh, Meylan, & Griffiths, 2017, for evidence of discrepancies between the patterns of similarity between...
Word2Vec representations and the patterns of similarity in human word association data). Our version of the Seq2Seq+Attention model represents a single word at each time step $t$ through a layer with localist units for each unique word in the data set (105 units). To represent a word the corresponding unit is given an activation of 1 while all other units are given an activation of 0 (i.e., a one-hot vector). This one-hot vector is transformed into a Word2Vec embedding (size 300) by a single feed-forward layer with a fixed set of weights (see Figure 2.2A). We did not allow the training process to change these weights.

The encoder (bottom part of Figure 2.2B) corresponds to a bidirectional long short-term memory neural network (Bidirectional LSTM, Graves & Schmidhuber, 2005). The Bidirectional LSTM is composed of two LSTM neural networks (250 units each in our model). The first LSTM reads the input from the beginning until the end of the sequence while the second reads the sequence in a backwards fashion. At each time step $t$ both LSTMs produce their own output. The full output of the encoder at is the concatenation of the outputs of the forward and backward LSTMs. The encoder’s output at each time step $t$ can be understood as a summary of all precedent and following words to the current word with an emphasis on the words surrounding it (Bahdanau et al., 2015).

The attention mechanism (center part of Figure 2.2B and Figure 2.2C) corresponds to a feed-forward neural network that, at each decoder’s time step $t'$, takes as input the decoder previous state $s_{t'-1}$, and all encoder outputs $a_1$ to $a_T$ (see Figure ??C). This feed-forward network produces a single number $e_t$ for each encoder’s output. This number is intended to capture the degree of alignment between the current word in the decoder with each word in the input sequence. This alignment score is normalized using a softmax function, yielding a single attention weight $w_t$ for each encoder’s output. The output of the attention mechanism is a context vector $c'_t$, which corresponds to the summation of all encoder’s outputs weighted by their corresponding attention weight. In short, the vector $c'_t$ represents a summary of the input words with an emphasis on the words that “correspond” better with the current output word.

The decoder (top part of Figure 2.2B) corresponds to a standard LSTM network (200 units) followed by feed-forward layer with softmax activation. This layer has a unit for each unique word in the data set (105 units) so that the decoder’s output at each time step corresponds to a probability distribution over the dataset vocabulary. The model’s answer at each time step is taken to be the word with maximum predicted probability. As this model is designed to receive words as inputs, during training we feed the propositions of our task to the model one word at the time. For each unfilled role we presented the special $<$NONE$>$ word. After presenting the complete story, we input a special word $<$Q$>$ to demarcate the beginning of the question, then we input the topic question, and finally we input a special word $<$GO$>$ to tell the model to start the decoding process. The target output was the sequence of words corresponding to the full proposition involving the topic concept. The difference between the actual output and the target was used to train the model in the same way as in the SG model. Figure 2.3
presents an example of this process. Here, the Seq2Seqs+Attention model (represented by the box) is presented with the complete sequence of words corresponding to the Restaurant story in Table 2.2. The model is asked about the “decided” topic and it responds by outputting the sequence of words corresponding to the proposition “Anne and Gary decided to go to restaurant”.

2.3 Simulation 1

In contrast to previous research with Gestalt models (Rabovsky et al., 2018; Rohde, 2002; St. John, 1992; St. John & McClelland, 1990), our manipulations aimed to disentangle the task’s relational structure from its statistical structure. Specifically, our tests were designed to keep the relational structure of the test stories constant relative to the training data while varying their statistical properties. In short, these tests relied on capturing bindings between roles and fillers in specific instances while ignoring the statistical regularities from the training data. We termed our first test concept violation. In this test, we trained the models in the concept restricted condition and then tested them with stories where the agent-1, agent-2 or the patient-theme roles were filled by the restricted concepts. The questions consisted on all the topic concepts of the propositions in which the restricted concepts were used. A role-based answer to the question required using the restricted concept to fill the corresponding role. Table 2.3 presents an example of a Restaurant concept violation story. In this example, the concepts Albert and Lois had never appeared as agents in any Restaurant story during the model’s training. The model was then tested using a story in which Albert or Lois appeared as agents in a Restaurant story by asking, for example, about the “tipped” proposition. The correct (role-based) answer was “Lois tipped waiter big”. Note that, while the model was trained in stories where Lois appeared as an agent in other locations, and had been trained to output that someone tipped big with other agents, it had never been trained to output the exact proposition “Lois tipped waiter big”. Table 2.3 also presents all the story questions and their corresponding role-based answers.

In our second test, termed correlation violation, we presented the models trained in the concept unrestricted condition with stories where we inverted a perfect statistical regularity of the story script. For example, in the Restaurant script the value of the attribute role in the second proposition determines the value of the attribute role in the third proposition in that if
Table 2.3: Example of a Concept Violation Story (Restaurant). Lois and Albert were restricted from instances of the Restaurant script during training.

<table>
<thead>
<tr>
<th>Story</th>
<th>Questions</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. &lt;Lois&gt; and &lt;Albert&gt; decided restaurant</td>
<td>decided</td>
<td>&lt;Lois&gt; and &lt;Albert&gt; decided restaurant</td>
</tr>
<tr>
<td>2. Restaurant quality &lt;expensive&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Distance to restaurant &lt;far&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. &lt;Lois&gt; ordered &lt;cheap-wine&gt;</td>
<td>ordered</td>
<td>&lt;Lois&gt; ordered &lt;cheap-wine&gt;</td>
</tr>
<tr>
<td>5. &lt;Lois&gt; paid bill</td>
<td>paid</td>
<td>&lt;Lois&gt; paid bill</td>
</tr>
<tr>
<td>6. &lt;Lois&gt; tipped waiter &lt;big&gt;</td>
<td>tipped</td>
<td>&lt;Lois&gt; tipped waiter &lt;big&gt;</td>
</tr>
<tr>
<td>7. Waiter gave change to &lt;Lois&gt;</td>
<td>gave</td>
<td>Waiter gave change to &lt;Lois&gt;</td>
</tr>
</tbody>
</table>

the restaurant was cheap it was nearby and if it was expensive it was far away (see Table 2.1). To create a Restaurant correlation violation story, we switched the value of the attribute role in the third proposition (i.e., a cheap restaurant was now far away, and an expensive restaurant was now nearby). A role-based answer to the questions of this test would use the input concept in the third proposition to fill the attribute role, even though it corresponds to a violation of a correlation seen during training. Table 2.4 presents an example of a Restaurant correlation violation story, its question and corresponding role-based answer. In this example the model had been trained in Restaurant stories where expensive restaurants are always far away and cheap restaurants are always nearby and the model is tested in a Restaurant story where an expensive restaurant is close by. The model is asked about the “distance” proposition and the correct (role-based) answer is that the restaurant is close by (i.e., “Distance to restaurant near”).

Table 2.4: Example of a Correlation Violation Story (Restaurant).

<table>
<thead>
<tr>
<th>Story</th>
<th>Questions</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. &lt;Anne&gt; and &lt;Gary&gt; decided restaurant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Restaurant quality &lt;expensive&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Distance to restaurant &lt;near&gt;</td>
<td>distance</td>
<td>Distance to restaurant &lt;near&gt;</td>
</tr>
<tr>
<td>4. &lt;Anne&gt; ordered &lt;cheap-wine&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. &lt;Anne&gt; paid bill</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. &lt;Anne&gt; tipped waiter &lt;big&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Waiter gave change to &lt;Anne&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In our third test, termed shuffled propositions, we presented the models trained in the concept unrestricted condition with stories where we randomized the order of the propositions. Recall that in our stories there are no repeated topic concepts. As a direct consequence, a role-based answer to a question should use the concepts of the proposition corresponding to each question to fill its roles, ignoring the ordering of the propositions. Table 2.5 presents an
example of a Restaurant shuffled propositions story, its questions and their corresponding role-based answers. In this example the model had been trained in stories that followed the same order of propositions as the Restaurant script (see Table 2.1). The model was presented with sequences of propositions that corresponded to a standard unrestricted Restaurant story, with the only difference being that the order of the propositions was randomized (e.g., the propositions in Table 2.5 are exactly the same as the ones on Table 2.2), so although the model had received all the individual propositions of the story during training, the model was never trained in the specific sequence being tested. After receiving the propositions, the model was asked about any of the topics of the story. For example, when asked about the “quality” topic, the correct (role-based) answer was the proposition “Restaurant quality expensive”. It is worth to note that in all our tests the correct (role-based) answers required simply filling the roles of the answer proposition with the concepts that the model had received as input.

### Table 2.5: Example of a Shuffled Propositions Story (Restaurant).

<table>
<thead>
<tr>
<th>Story</th>
<th>Questions</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. &lt;Anne&gt; ordered &lt;cheap-wine&gt;</td>
<td>decided</td>
<td>&lt;Anne&gt; and &lt;Gary&gt; decided restaurant</td>
</tr>
<tr>
<td>5. &lt;Anne&gt; paid bill</td>
<td>quality</td>
<td>Restaurant quality &lt;expensive&gt;</td>
</tr>
<tr>
<td>1. &lt;Anne&gt; and &lt;Gary&gt; decided restaurant</td>
<td>distance</td>
<td>Distance to restaurant &lt;far&gt;</td>
</tr>
<tr>
<td>3. Distance to restaurant &lt;far&gt;</td>
<td>ordered</td>
<td>&lt;Anne&gt; ordered &lt;cheap-wine&gt;</td>
</tr>
<tr>
<td>7. Waiter gave change to &lt;Anne&gt;</td>
<td>paid</td>
<td>&lt;Anne&gt; paid bill</td>
</tr>
<tr>
<td>6. &lt;Anne&gt; tipped waiter &lt;big&gt;</td>
<td>tipped</td>
<td>&lt;Anne&gt; tipped waiter &lt;big&gt;</td>
</tr>
<tr>
<td>2. Restaurant quality &lt;expensive&gt;</td>
<td>gave</td>
<td>Waiter gave change to &lt;Anne&gt;</td>
</tr>
</tbody>
</table>

#### 2.3.1 Training

We trained two versions of the SG model, one in 1,000,000 randomly generated concept restricted stories and another in 1,000,000 randomly generated concept unrestricted stories. We also trained two versions of the Seq2seq+Attention model, one in 500,000 randomly generated concept restricted stories and another in 500,000 randomly generated concept unrestricted stories (the difference in number of training examples was due to the Seq2seq+Attention model achieving ceiling performance at training time). We used the Nadam optimization algorithm (Dozat, 2016) with default learning parameters. All our models were implemented in Keras (Chollet et al., 2015) with TensorFlow backend (Abadi et al., 2016). Full code for all simulations is available from [https://github.com/GuillermoPuebla/RelationReasonNN](https://github.com/GuillermoPuebla/RelationReasonNN).
2.3. Simulation 1

2.3.2 Results

For each of our tests, we created a dataset of 728 randomly generated stories. This number corresponds to the number of all possible concept violation stories, which is the script with the lower number of possible stories. For all tests we compared the proposition generated by the model with the role-based answer. We coded the answer as correct (with a value of 1) if all the concept fillers in the answer corresponded to the concept fillers in the role-based answer and as a non-match (with a value of 0) otherwise. Figure 3.4 shows the proportion of correct answers per test and model. Recall that in our baseline test we presented the models trained in the concept unrestricted condition with concept unrestricted stories and asked questions about all the propositions in the stories (see Table 2.2 for an example of a Restaurant baseline story, its questions and corresponding correct answers). Because the test stories came from exactly the same distribution as the training stories this test is akin to a recall test of the training dataset. As can be appreciated in Figure 2.4, both models performed well in our baseline test. It is noteworthy that the Seq2Seq+Attention model showed a better baseline performance than the SG model even though it was trained in half the number of stories (accuracy of 0.96 vs. 0.92).

![Figure 2.4: Accuracy per test and model. Both models perform well in the baseline condition. Furthermore, their performance was affected differentially in our critical conditions. The Story Gestalt model was more susceptible to the concept violation and correlation violation manipulations while the Seq2Seq+Attention model was more susceptible to the correlation violation and shuffled propositions manipulations. As none of these manipulations changed the relational structure of the task, these results suggest that neither model was able to capture it during training. Error bars are 95% confidence intervals.](image-url)
Recall that in our concept violation test we trained the models in the concept restricted condition and then tested them with stories where the agent-1, agent-2 or the patient-theme roles were filled by the restricted concepts\(^1\). The questions consisted of all the topics of the propositions in which the restricted concepts were used and a correct (role-based) answer required using the restricted concepts to fill the corresponding roles (see Table 2.3 for an example of a Restaurant concept violation story, its questions and corresponding correct answers). In this test the SG model was unable to use the concepts restricted during training to answer the questions (accuracy of 0.08). Instead, the SG model almost invariably filled the roles of the restricted concepts with the most common concepts playing that role during training, which is a direct replication of the results of (St. John, 1992). For example, if the SG model was presented with a story like the one on Table 2.3 were the roles agent-1 and agent-2 corresponded to the restricted concepts “Lois” and “Albert”, the model tended to output answers where the agent-1 and agent-2 were any of the other unrestricted agents (e.g., “Barbara” or “Clement”). The Seq2Seq+Attention model performed significantly better at this test, achieving a slightly better level of accuracy than in the baseline test (accuracy of 0.99). The attention mechanism seems to allow this model to apply its word representations to sequences where the words appeared in previously unseen stories.

Recall that in our correlation violation test we presented the models trained in the concept unrestricted condition with stories where we inverted a perfect statistical regularity of the story script and asked about the proposition that violated the perfect statistical regularity. The correct (role-based) answer required using the input concept even though it violated a statistical correlation from the training dataset. For example, because in the Restaurant script expensive restaurants are always far away, a Restaurant correlation violation story stated that an expensive restaurant is close by and the correct answer to the “distance” question was that the restaurant is indeed close by (see Table 2.4). Importantly, both models performed poorly in the correlation violation test, in other words, neither model was consistently able to correctly process texts that violated a perfect correlation seen in the training dataset (accuracy of 0.165 and 0.23 for the SG and Seq2Seq+Attention models, respectively). Such behavior would seem quite unnatural for a human reader as it would be analogous to say that my friend John, who I just saw eating salad at the restaurant, ate chicken just because I’ve only seen him eating chicken in the restaurant in the past. Of course, it is possible to achieve perfect performance in this test by training the models in a corpus where all possible role-filler combinations appear in several contexts (e.g., several “establishments” other than the restaurant that are cheap and far away, cheap and close by, expensive and far away and expensive and close by, see e.g., St. John, 1992)\(^2\). However, the point of the simulation is that it shows that the inferences

---

1. Although these concepts were never used the in the context of each specific script, they were seen in the training dataset as a whole. By definition, the output of any traditional neural network to a completely new (unseen) concept depends on its initial weights. Given that these weights are initialized randomly, the behavior of a neural network regarding an unseen input will be essentially random (Marcus, 1998).

2. We actually run that simulation and, unsurprisingly, obtained perfect “generalization”.
these models can make are in strictly limited by the statistical structure of its training corpus. It is noteworthy that the SG model achieved a higher accuracy than Seq2Seq+Attention model in this test (although both models performed quite poorly). We suspect that the more powerful Seq2Seq+Attention model is more likely to overfit to a perfect correlation in the dataset.

Recall that in our shuffled propositions test we presented the models trained in the concept unrestricted condition with concept unrestricted stories where the order of the propositions was randomized. A correct (role-based) answer required to use the concepts of the proposition corresponding to each question to fill its roles, ignoring their ordering (see Table 2.5 for an example). While the randomization of the order of the propositions affected both models, the SG model performed significantly better than the Seq2Seq+Attention model in this test (accuracy of 0.73 vs. 0.39). We hypothesize that the attention mechanism is the main reason for this difference in performance. Unfortunately, because of the length of our stories, taking out the attention mechanism yields the Seq2Seq+Attention model unable to pass our baseline test (baseline performance around 0.5), so for now we were not able to test our hypothesis directly.

2.4 Simulation 2

Simulation 1 showed how a series of manipulations that should not affect a model that learns a relational representation of a story affects a classic and contemporary neural network model of language processing. This entails that neither model is learning a relation-based representation of the story, but instead they are relying on the statistical regularities of the training dataset to answer the questions. A potential issue with Simulation 1 is that the training objective of the task is rather indirect: it demands to learn to find the sentence the probe corresponds to from the test story. Arguably, this does not necessarily require to learn relationships between the objects and roles in the story to succeed at training time (although humans seem to naturally do so in equivalent situations Lake, Linzen, & Baroni, 2019).

To address this potential issue we adapted the original task of St. John (1992) to probe for relational roles directly. To accomplish this we added five new words to the models’ vocabulary: agent-1, agent-2, attribute, manner and patient. In the Story Gestalt model these words corresponded to new localist units in the question layer and in the Seq2Seq+Attention model these words were added to the Word2Vec embeddings (we used the embeddings of the words agent and actor for agent-1 and agent-2, respectively). We trained both models by presenting stories and asking about a specific role in the story. The models had to answer with the concept word that played that role in the story (see Table 2.6). In the SG model this meant to activate only the corresponding concept unit in the answer layer (as opposed to activate a group of units representing a sentence), while in the Seq2Seq+Attention model this meant to return a single concept word. Because in our stories the roles are specified at sentence level
only a few roles remain constant in each story. In particular, the roles agent-1 and agent-2 are always filled by a single character throughout a story. This means that it is possible to test for relational generalization in the models by training these roles in a set of characters and test in a disjoint set. Importantly, these characters are seen during training across all scripts, just not filling the agent-1 and agent-2 roles. For this manipulation we created 4 new characters. The characters Will and Tina never filled the agent-1 role but were free to fill the agent-2 role and the characters Alex and Kate followed the opposite pattern.

Table 2.6: Example of a Relational Probe Story (Restaurant).

<table>
<thead>
<tr>
<th>Story</th>
<th>Questions</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <code>&lt;Anne&gt;</code> and <code>&lt;Gary&gt;</code> decided restaurant</td>
<td>agent-1</td>
<td><code>&lt;Anne&gt;</code></td>
</tr>
<tr>
<td>2. Restaurant quality <code>&lt;expensive&gt;</code></td>
<td>agent-2</td>
<td><code>&lt;Gary&gt;</code></td>
</tr>
<tr>
<td>3. Distance to restaurant <code>&lt;near&gt;</code></td>
<td>attribute</td>
<td><code>&lt;near&gt;</code></td>
</tr>
<tr>
<td>4. <code>&lt;Anne&gt;</code> ordered <code>&lt;cheap-wine&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. <code>&lt;Anne&gt;</code> paid bill</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. <code>&lt;Anne&gt;</code> tipped waiter <code>&lt;big&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Waiter gave change to <code>&lt;Anne&gt;</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Additionally, we sought to measure the models’ answers to direct relational questions when there was a strong distribution shift at test time with known concepts like in the correlation violation condition of Simulation 1. For this we trained the models to answer direct relational questions to the role involved in the correlation violation manipulation while maintaining the same statistical regularities of Simulation 1. For example, in the restaurant stories we trained the models to answer a question about the attribute role in the third proposition. The models had to answer with the concept that filled that role (i.e., whether the restaurant was “near” or “far” which was perfectly predictable from the quality of the restaurant, see Table 1). At test time the models were tested in a story where the filler of the role breaks the perfect correlation in the training distribution. For instance, the models were asked about the attribute role in the third proposition in a story where the restaurant was close by but it was expensive instead of cheap (see Table 2.6). Because the correlation violation manipulation is necessarily script-type specific, so it is the specific role asked about for each story type (see Appendix A for all the deterministic rules used in the correlation manipulation for each script type).

2.4.1 Training

We trained the models on randomly generated batches. A single batch contained three story-question pairs, where the story across pairs was the same. The first question asked about the agent-1 role, the second about the agent-2 role and the third about the script-type-specific role. We trained the SG model in 200,000 batches and the Seq2Seq+Attention model
2.4. Simulation 2

in 30,000. Both models achieved ceiling performance during training (as in Simulation 1 this
difference was due to the Seq2Seq+Attention model achieving ceiling performance earlier in
the training process). We used the same optimization algorithm and training parameters as in
Simulation 1.

2.4.2 Results

We tested both models on 536 batches of stories where the fillers of the roles corresponded to
the usual fillers seen during training (baseline condition) and on 536 batches where the fillers
of the roles corresponded to the role-filler combinations withheld during training (relational
condition). As can be appreciated in Figure 2.5, both models achieved good performance
when the fillers of the roles corresponded to the usual fillers seen during training. For example,
both models would answer correctly to a question about the role agent-1 when Kate played
that role (accuracy of 1.0 in the baseline condition for the agent-1 role). It is worth noting,
however, that the Seq2Seq+Attention model performed slightly worse than the SG model in
the baseline condition for the agent-2 role (accuracy of 0.74 vs. 1.0). It is also clear that both
models performed worse when the agent-1 and agent-2 roles were filled by concepts that
did not play those roles during training. In this case the Seq2Seq+Attention model performs
slightly better than the SG model (accuracy of 0.2 vs. 0.0 for the agent-1 role and 0.52 vs.
0.18 for the agent-2 role).

![Figure 2.5: Accuracy per condition, role and model. Both models perform reasonably well in the baseline condition for all roles. For both models there is a significant drop in accuracy when the agent-1 and agent-2 roles are filled with concepts different than those used in training (relational condition). For the correlation-violating filler the drop in accuracy is more pronounced for the Seq2Seq+Attention model although is still appreciable in the SG model. As in Simulation 2 we probed for relational roles directly, these results strengthen the conclusion that neither model has grasped the relational structure of the task. Error bars are 95% confidence intervals.](image_url)

Regarding the script-type-specific role, our results show that both models perform well in the
baseline condition (accuracy of 0.96 for both models). In contrast, the Seq2Seq+Attention
model performs significantly worse than the SG model in the relational condition (accuracy of
0.02 vs. 0.73). Note that unlike the agent-1 and agent-2 roles there is not a sharp division in the
2.4. Simulation 2

set of fillers of the baseline and relational conditions of the script-type-specific role. Instead, the main challenge of this test is to answer with the correct filler even though there is a strong distribution shift in the test stories. The poor performance of the Seq2Seq+Attention model in this task suggest that its comparatively better performance to the SG model in the agent-1 and agent-2 roles does not come from a more relational representation of the stories per se. The fact that a classic connectionist model performed better than a modern deep neural network in this task highlights how different experimental manipulations have different (and sometimes surprising) effects on different architectures, which necessitates to perform several tests when trying to characterize the relational reasoning capabilities of different models.

Overall, the results of this simulation mimic those of Simulation 1. It is not the case that the structure of the task and the training objective on Simulation 1 was the main factor that lead to the non-relational solution found by the models, as in this simulation we showed that directly probing for relational roles does not seem to improve the relational generalization capabilities of either model.

2.5 General discussion

We tested the relational processing capabilities of the SG model and the Seq2Seq+Attention model, a classic connectionist model of text comprehension and a contemporary language processing deep learning architecture, respectively. In Simulation 1 we varied the statistical properties of the test stories while keeping their relational structure intact. Our results show that both models are able to use the statistical regularities of the training data to learn to answer questions correctly for stories that came from the same distribution as the training corpus. More importantly, however, our simulations demonstrate that the performance of both models is severely affected when the statistical properties of the test stories differ from those in the training corpus. Because we kept the relational structure of the test stories intact, our results show clearly that these models are not using the relational information of the stories to answer the questions, but instead they are relying on the statistical regularities of the training dataset. In Simulation 2 we showed that this is true even when the models are asked directly about relational roles. In addition, although the technical advances of Seq2Seq+Attention model made it able to pass our concept violation test in Simulation 1, this performance did not transfer to the direct relational questions of Simulation 2. Overall, neither model showed a better capacity to deal with relational reasoning tasks, as both models performed worse than the other in some condition of our simulations.

It is worth noting that the Seq2Seq+Attention model has been highly influential in the machine reading comprehension (MRC) literature. Attention mechanisms are a key component of virtually all mayor deep learning MRC architectures (for a review see X. Zhang, Yang, Li, & Wang, 2019). This literature has also produced a set of databases to test the reading
comprehension capabilities of these systems. For example, the popular bAbI dataset (Weston, Bordes, Chopra, & Mikolov, 2016) consist of 20 tasks aimed to test basic forms of logical understating, such as deduction, induction, compound co-reference and many more. The texts and questions are generated automatically from a simulation of characters moving around and manipulating objects in a simple environment. Another popular dataset, the Stanford Question Answering Dataset (SQuAD) (Rajpurkar, Zhang, Lopyrev, & Liang, 2016), consist of a large set of questions generated by humans on a collection of passages extracted from Wikipedia. The answer to each question is a section of text from the corresponding article. Interestingly, different researchers have shown that these kinds of datasets are less rigorous tests of reading comprehension than previously thought. For example, (Kaushik & Lipton, 2018) showed that deep learning architectures can perform surprisingly well in many MRC datasets (including bAbI) even without seeing either the input text or the question. Another example is Jia and Liang (2017), who showed that deep learning models trained on SQuAD are susceptible to adversarial attacks that add untrained sentences, that share words with the correct answer, to the test texts (Jia & Liang, 2017). Notably, ungrammatical distractor sentences have a stronger adversarial effect than grammatical ones, which suggest that these models are relying in a superficial strategy to solve the reading comprehension task. We believe that highly controlled experiments such as the ones performed in the present research are necessary to evaluate neural network models (deep or otherwise) of language processing. Fortunately, some MRC researchers seem to taking this direction (Dunietz et al., 2020).

Our results are highly consistent with the findings of Lake and Baroni (2018) and Loula, Baroni, and Lake (2018), who found that sequence-to-sequence models (with and without attention mechanism) failed at a command-to-action translation task that required composing the meaning of new commands formed by using known primitive concepts combined in ways unseen during training. Even in the minority of cases were their models showed behavior that seemed compositional, they did it in a very non-human way (e.g., in one test their best performing model could correctly produce the action sequences corresponding to the instructions “turn left”, and “jump right and turn left twice”, but not the one corresponding to “jump right and turn left”). Hupkes, Dankers, Mul, and Bruni (2019) showed comparable results in a artificial grammar learning task with a Seq2Seq+Attention model, a Convolutional Seq2Seq model and a Transformer model.

Truly compositional behavior requires independent representations of objects and roles that can be bound together dynamically (i.e., compositional representations require a solution to the binding problem). In particular, compositionality results when a system can recursively apply predicate representations over other predicate representation (e.g., loves(John, loves(Mary, Richard)), for discussions see Fodor, 1975; Marcus, 2001; Martin & Doumas, 2019b; Tenenbaum, Kemp, Griffiths, & Goodman, 2011). We have shown that traditional PDP
models (including current deep learning models) do not, as instantiated, perform dynamic binding. As a consequence, these models systematically fail when a task requires violating well learned statistical associations. As such, while there are certainly instances wherein the representations that these models learn will produce the same results as compositional representations, the resulting representations are not truly compositional.

One of the most important evolutionary advantages of relational reasoning is the ability to base inferences on relational roles disregarding the content of their arguments. This capacity allows us to make relational generalizations to completely new inputs (Penn et al., 2008). As traditional neural networks can’t, by definition, make use of untrained units to perform successfully in given a task (Marcus, 1998), these models rely on spanning the input space to achieve good generalization (see Doumas & Hummel, 2012). Word embeddings like Word2Vec (Mikolov et al., 2013, cf. Miikkulainen and Dyer (1991)) can be seen as a technique to deal with this phenomenon. Even though in our Seq2Seq+Attention model some concepts were not trained in some contexts, the vector representation of all concepts of a certain type (e.g., agents like “Anne” and “Lois”) had similar representations because they appear in similar contexts in the Word2Vec training dataset. Another strategy to deal with new concepts (or new combinations of concepts) involves directly spanning the input space so that there are no truly new inputs to the model. For example, it is standard practice in neural networks research to make random splits of the data to obtain the training and test datasets. When the data are instantiations of relational structures (as in our tasks) this makes very likely that most objects appear as the fillers of most relational roles in the training dataset, which transforms the relational generalization problem in an interpolation problem, where the correct answer corresponds to an intermediate answer between two known cases (see Lake & Baroni, 2018, for a demonstration of the effects of random versus systematic splits on the training/test datasets). It is for this reason that traditional PDP models (e.g., O’reilly & Busby, 2002) and contemporary deep learning models (e.g., Hill, Santoro, Barrett, Morcos, & Lillicrap, 2019) targeted to solve relational reasoning tasks rely on spanning the input space in order to achieve high levels of generalization. Importantly, none of these techniques are solutions to the deeper problem of generalizing to new concepts or new combinations of concepts based on abstract relations.

However, all of the above is not to say that neural network models cannot, in principle, integrate operations that allow them to implement a truly symbolic dynamic binding system. For example, the symbolic-connectionist models SHRUTI (Shastri & Ajjanagadde, 1993), LISA (Hummel & Holyoak, 1997, 2003), and DORA (Doumas et al., 2008; Doumas & Martin, 2018), use time as a binding signal that allows for role-filler independence and dynamic binding.
Interestingly, there has been a resurgence of interest on the binding problem in the neural networks (Besold et al., 2017; Franklin, Norman, Ranganath, Zacks, & Gershman, 2019) and computational neuroscience literature (Fitz et al., 2019; Pina, Bodner, & Ermentrout, 2018). Moreover, relational learning and reasoning have become a core topic on deep learning research (Bahdanau et al., 2018; Battaglia et al., 2018; Greff, Srivastava, & Schmidhuber, 2015; Hill et al., 2019; Santoro et al., 2017) with some deep learning architectures starting to implement operations traditionally associated with symbolic processing such as a content-addressable memory (Graves et al., 2016; Santoro, Bartunov, Botvinick, Wierstra, & Lillicrap, 2016; Weston, Chopra, & Bordes, 2014). Whether these non-traditional neural network architectures are capable of relational reasoning remains an open question that we plan to address in future research. Our results suggest, however, that for a model to successfully account for all aspects of relational processing, it will need to implement a solution to the binding problem.

Finally, while we herein illustrate the limitations of traditional neural networks when facing relational reasoning tasks, we hope that the results will motivate cognitive scientists and machine learning researchers to tackle the problem of relational learning and reasoning by first tackling the problem of dynamic binding. In the domain of neural network models, doing so will most likely will require us to go beyond the architectural constrains of traditional neural networks.
Chapter 3

A Theory of Relation Learning and Cross-domain Generalization

3.1 Introduction

Many children learn the apocryphal story of Newton discovering his laws of mechanics when an apple fell from a tree and hit him on the head. We were told that this incident gave him the insight that ultimately led to a theory of physics that, hundreds of years later, would make it possible for a person to set foot on the moon. The point of this story is not about the role of fruit in humankind’s quest to understand the universe; the point is that a single incident—a child’s proxy for a small number of observations—led Newton to discover laws that generalize to an unbounded number of new cases involving not only apples, but also cannonballs and celestial bodies. Without being told, we all understand that generalization—the realization that a single set of principles governs falling apples, cannonballs, and planets alike—is the real point of the story.

Although the story of Newton and the apple is held up as a particularly striking example of insight leading to broad generalization, it resonates with us because it illustrates a fundamental property of human thinking: People are remarkably good—irresponsibly good, from a purely statistical perspective—at applying principles discovered in one domain to new domains that share little or no superficial similarity with the original. A person who learns how to count using stones can readily apply that knowledge to apples. A graduate student who learns to analyze data in the context of psychological experiments can readily generalize that knowledge to data on consumer preferences or online search behavior. And a person who learns to play a simple videogame like Breakout (where the player moves a paddle at the bottom of the screen horizontally to hit a ball towards bricks at the top of the screen) can readily apply that knowledge to similar games, such as Pong (where the player moves a paddle on the side of the screen vertically and tries to hit a ball past the opponent paddle on the other side of the screen).
This kind of “cross-domain” generalization is so commonplace that it is tempting to take it for granted, to assume it is a trivial consequence of the same kind of statistical associative learning taught introductory psychology. But the truth is more complicated. First, there is no clear evidence that any species other than humans is capable of the kind of flexible cross-domain generalization we find so natural (see Penn et al., 2008). And second, while “cross-domain” generalization has also been the subject of substantial research in the field of machine learning (see, e.g., Gamrian & Goldberg, 2019; Kansky et al., 2017; C. Zhang, Vinyals, Munos, & Bengio, 2018), robust, human-like cross-domain generalization continues to frustrate even the most powerful deep neural networks (DNNs; see, e.g., Bowers, 2017; Geirhos et al., 2020).

In the following we present a theory of human cross-domain generalization. Our primary claim is that human cross-domain generalization is a product of analogical inference performed over structured relational representations of multiple domains. We instantiate our theory in a unification and extension of two existing computational models of relational reasoning, LISA (Hummel & Holyoak, 1997, 2003) and DORA (Doumas et al., 2008). The resulting model accounts for how we acquire structured relational representations from simple visual inputs, integrates with current methods for reinforcement learning to learn how to apply these representations in some domain, and accounts for how we leverage these representations to generalize knowledge across different domains.

In what follows, we first review evidence for the role of structured relational representations in generalization, broadly defined, but especially in cross-domain generalization. Along the way, we discuss what it means for a representation to be structured and relational, contrasting the strengths and limitations of implicit and explicit representations of relations. Next, we outline our theory of human cross-domain generalization and describe the computational model that instantiates it. We then present a series of simulations demonstrating that the model learns structured representations of relations and uses these representations to perform zero-shot (i.e., first trial) cross-domain generalization between different video games and between completely different tasks (video games and analogy-making). We show that the model’s learning trajectory closely mirrors the developmental trajectory of human children. Finally, we discuss the implications for our account, contrasting it with purely statistical learning accounts such as DNNs, and we consider future extensions of the theory.

In addition to providing an account of cross-domain transfer, the model also represents theoretical advance in at least two other domains, (1) the discovery of the kind of invariant semantic properties that constitute the meaning of a relation, and (2) the integration of explicitly relational representations with reinforcement learning to produce representations with the expressive power of an explicitly relational representation with the productive capacity of implicit representations of relations (e.g., as weight matrices).
3.1. Introduction

3.1.1 Relational Representations and Generalization

Psychologists have long observed that people’s concepts, even of “ordinary” domains like cooking and visual occlusion, are like theories or models of the domains in question (Carey, 2000, 2009; Murphy & Medin, 1985), in that they specify the relations among the critical variables in the domain. For example, our understanding of visual occlusion specifies that a larger surface can occlude a smaller surface more than vice-versa (one can hide behind a hundred-year-old oak better than a sapling; e.g., Hespos & Baillargeon, 2001); that an object continues to exist, even when it’s hidden behind an occluder (e.g., Piaget, 1954); and that the ability of an occluder to hide an object depends on the relative distances between the occluder, the observer, and the hidden object. Our model of biology tells us that the offspring are the same species as the parents (e.g., Gelman et al., 2003). Our understanding of cooking specifies that the amount of salt one adds to a dish should be proportional to the size of the dish. And our model of a game like tennis, baseball, or Pong tells us that the ability of the racket, bat, or paddle to hit the ball depends on the locations and trajectories of these objects relative to one another.

The critical property of all these models is that they specify—and thus depend on the capacity to learn and represent—an open-ended vocabulary of relations among variables: Whether \( x \) can occlude \( y \) from viewer \( v \) depends on the relative sizes of \( x \) and \( y \), and the relative distances and angles between \( x \), \( y \), and \( v \). And whether a given amount of salt is too much, too little, or just right depends on the relation between the amount of salt, the size of the dish, and the taste of the diner. Accordingly, learning a model of a domain entails learning a representation of the relations characterizing it. We take the claim that an “understanding” of a domain consists of a representation of the relations characterizing that domain to be uncontroversial. However, our claim is stronger than that. We claim that an understanding of a domain consists of structured relational representations of the domain. As elaborated shortly, by structured relational representation, we mean a representational format that explicitly captures both the semantic content (i.e., meaning) and the compositional structure (i.e., bindings of arguments to relational roles) of a relationship.

One of the most important manifestations of a capacity to reason about relations is analogical inference: inferences based on the relations in which objects are engaged, rather than just the literal features of the objects themselves. Analogical inference is evident in almost every circumstance in which a person demonstrates knowledge on which she was never explicitly trained. The physics student learning Newton’s law, \( f = ma \), does not need to learn multiplication de novo in the context of this new equation. So instead of teaching her an enormous lookup table with all possible \( m \) and \( a \) as input and the corresponding \( f \) as output, the physics teacher simply gives her the equation, knowing her knowledge of multiplication will generalize to the domain of physics. The student who knows he needs at least a grade of 70% to pass a course also knows, without additional training, that 69%, 68%, 67%, etc. are
all failing grades, an inference based on the relation between 70 and all the numbers smaller than it. And if you have a meeting at 2:00 pm, and the current time is 1:00, then you know you are not yet late. Importantly, we know all of this without explicit training on each domain individually. An understanding of a relation such as less-than or multiplied-by simultaneously confers understanding to all domains in which it applies (e.g., grades, appointments, recipes, automobile manufacture, etc.). As a result, cross-domain generalization based on structured relational representations is not the exception in human thinking, it is the default.

3.1.2 Learning Relational Representations

These facts have not escaped the notice of cognitive modelers (Doumas et al., 2008; Falkenhainer et al., 1989; Halford, Wilson, & Phillips, 1998; Hummel & Holyoak, 1997, 2003; Pacanaro & Hinton, 2001), and recent years have seen increased interest in getting neural networks trained by back propagation to learn useful representations of relations in domains such as relation extraction from pictures (Haldekar, Ganesan, & Oates, 2017), visual question answering (e.g., Cadene, Ben-Younes, Cord, & Thome, 2019; Ma et al., 2018; Santoro et al., 2017; H. Xu & Saenko, 2016), same-different discrimination (Funke et al., 2021; Messina, Amato, Carrara, Gennaro, & Falchi, 2021), and even visual (Hill et al., 2019; Hoshen & Werman, 2017) and verbal (Mikolov et al., 2013) analogy-making. The core assumptions underlying these efforts are that (1) the relevant relational properties can be discovered as a natural consequence of the statistical properties of the input-output mappings, and that (2) the relevant relations will be represented in the learned weight matrices and will permit relational generalization.

This statistical approach to relation learning has met with some substantial successes. One strength of this approach is that because the learned relations are represented implicitly in the networks’ weight matrices, they are functional in the sense that they directly impact the model’s behavior: Given one term of a relation, for instance, along with a weight matrix representing a relation, a network of this kind can produce the other term of the relation as an output (see e.g., Leech et al., 2008; Lu et al., 2012, but cf. Lu, Ichien, and Holyoak, 2021). By contrast, models based on more explicit representations of relations (e.g., Anderson, 2007; Doumas et al., 2008; Falkenhainer et al., 1989; Hummel & Holyoak, 1997, 2003), including the model presented here, must explicitly decide how to apply the relations it knows to the task at hand (e.g., by adding an inferred proposition to a database of known facts; see Anderson, 2007). As elaborated in the Simulations section, an advance presented in this paper is a technique for using reinforcement learning to choose which relations to use in what circumstances in the context of video game play.
3.1. Introduction

Although statistical learning of implicit relations achieves impressive results when tested on examples that lie within the training distribution, their performance deteriorates sharply on out-of-distribution samples. For example, the relational network of Santoro et al. (2017) was trained to answer same-different questions about pairs of objects in an image. When the model was tested on shape-color combinations withheld from the training data (e.g., a test image with two identical cyan squares where the model had seen squares and the color cyan individually but not on combination), its performance dropped to chance (Kim et al., 2018). The limited applicability of the relations learned by these models holds across application domains (for recent reviews see, Peterson, Chen, & Griffiths, 2020; Ricci, Cadène, & Serre, 2021; Sengupta & Friston, 2018; Stabinger, David, Piater, & Rodríguez-Sánchez, 2020).

Why are useful relational representations so hard to learn using traditional statistical learning? The short answer is that although these approaches might capture the content of (some of) the relevant relations in their respective domains, they do not represent those relations in a form that supports broad relational generalization. We argue that flexible, cross-domain generalization relies, not on implicit representations (like weight matrices), but instead on explicitly relational representations, that simultaneously (a) represent the semantic content of a relation, e.g., the meaning of the relation \textit{left-of()}, and (b) represent that content in a format that makes it possible to dynamically bind the relation to its arguments without altering the representation of either. The following expands on the distinction between form and content with the goal of clarifying our claims about the nature and utility of structured relational representations for the purposes of generalization. We return to this issue in much more detail in the Discussion, where we relate our simulation results to the differences between structured relational representations, as endorsed in the current theory, and implicit representations of relations, as represented in the weight matrices of some neural networks.

\textbf{Relational Content} What does it mean to represent a relation such as \textit{left-of()}? To a first approximation, it means having a unit (or pattern of activation over multiple units) that become active if and only if some part of the network’s input is to the left of some other part. For example, imagine a neural network that learns to activate a node in its output layer or in some hidden layer if and only if the network’s input contains at least two objects, \textit{i} and \textit{j}, whose locations in the horizontal dimension of the display are unequal. Note that in order to represent \textit{left-of()}, \textit{per se}, as opposed to instances of \textit{left-of()} at a specific location in the visual field, the unit or pattern must become active whenever any object \textit{i} is left of any other object \textit{j}, regardless of the specific objects in question, and regardless of their specific locations in the visual field. That is, a representation of a relation, such as \textit{left-of}(\textit{i}, \textit{j}), is useful precisely to the extent that it is invariant with the specific conditions (e.g., the particular retinal coordinates,
3.1. Introduction

$x_i$ and $x_j$ giving rise to it and with the objects (arguments, $i$ and $j$) bound to it (Hummel & Biederman, 1992): Such a unit or pattern would, by virtue of its 1:1 correspondence with the presence of things that are left-of other things in the input, represent the semantic content of the relation left-of().

The invariance of a relational representation is partly responsible for the flexible generalization afforded by such representations. A system that can represent left-of(paddle, ball) in a form that is invariant with the specific locations of the paddle and ball is well-prepared to learn a rule such as “if the paddle is left of the ball, then move the paddle to the right” and then apply that rule regardless of where the paddle and ball are located in the game display. Such a representation would even permit generalization to a screen wider than the one used during training (i.e., with previously unseen values of $x$). Conversely, representing left-of(paddle, ball) with different units depending on where the paddle and ball are in the display will not permit generalization across locations: Having learned what to do when the paddle is at $x = 10$ and the ball is at $x = 11$ (represented by a unit we’ll call left-of([10,11])), such a network would not know what to do when the paddle is at $x = 12$ and the ball at $x = 15$ (represented by a different unit, left-of([12,15])).

The Form of Structured Relational Representations  Being able to represent relational invariants such as left-of() and above() is extremely useful, if not necessary, for broad, cross-domain generalization, but it is not sufficient. Simply activating a unit or pattern representing left-of() does not specify what is to the left if what: Is the paddle left of the ball, or the ball left of the paddle? Or is one of them, or some other object, to the left of some third or fourth object? Knowing only that something is left of something else provides no basis for deciding whether to move the paddle to the left or right.

Representing a relation such as left-of($i, j$) in a way that can guide reasoning or behavior entails representing both the relational content of the relation (e.g., that something is left of something else, as opposed to, say, larger than something else), and specifying that content in a format that makes the bindings of arguments to relational roles explicit. Following the literature on analogy and relational reasoning, we will use the term predicate to refer to a representation in this format. For our current purposes, a predicate is a representation (a symbol) that can be bound dynamically to its argument(s) in a way that preserves the invariance of both (see, e.g. Halford, Wilson, & Phillips, 1998; Hummel & Biederman, 1992; Hummel & Holyoak, 1997, 2003). By “dynamic binding” we mean a binding that can be created and destroyed as needed: To represent left-of(paddle, ball), the units representing the paddle must be bound to the units representing the first role of left-of() while the units representing the ball are bound to the units representing the second role; and to represent left-of(ball, paddle), the very same units must be rebound so that ball is bound to the first role and paddle to the second. In propositional notation, these bindings are represented by the order
of the arguments inside the parentheses. Neural networks need a different way to signal these bindings, and the work reported here follows Doumas et al. (2008, see also Hummel & Biederman, 1992; Hummel & Holyoak, 1997, 2003) and others in using systematic synchrony and asynchrony of firing for this purpose.

What matters about dynamic binding is not that it is based on synchrony of firing; one might imagine other ways to signal bindings dynamically in a neural network. What matters is only that the dynamic binding tag, whatever it is, is independent of the units it binds together. That is, the binding tag must be a second degree of freedom, independent of the units’ activations, with which to represent how those units (representing roles and objects) are bound together. Synchrony of firing happens to be convenient for this purpose, as well as neurally plausible (e.g., Hummel & Biederman, 1992; Hummel & Holyoak, 1997; Rao & Cecchi, 2010, 2011; Reichert & Serre, 2013; Shastri & Ajanagadde, 1993). For example, to represent left-of(paddle, ball), neurons representing left-of would fire in synchrony with neurons representing the paddle, while neurons representing right-of fire in synchrony with neurons representing the ball (and out of synchrony with the paddle and left-of neurons).

The very same neurons would also represent left-of(ball, paddle), but the synchrony relations would be reversed. In this way, the representation captures the form of the representation (distinguishing left-of(ball, paddle) from left-of(paddle, ball)), without sacrificing the content of left-of(), ball, or paddle. This ability to bind ball and paddle dynamically to the roles of left-of() without changing the representation of either derives from the fact that when a unit fires, is independent of how strongly it fires: The representation is explicitly relational because timing (which carries binding) and activation (which carries content) are independent.

We posit that representing a relation as a structure that is invariant with its arguments and can be dynamically bound to arguments permits immediate (i.e., zero-shot) generalization to completely novel arguments, including arguments never seen during training. For example, a Breakout player who represents left-of() in a way that remains unaffected by the arguments of the relation could adapt rapidly if the paddle and ball were suddenly replaced by, say, a triangle and a square, or a net and a bunny. Armed with the capacity to map a given relation such as left-of() in one domain onto a different relation such as above() in another—that is, armed with a capacity for analogical mapping—a player could also rapidly adapt “keep the paddle aligned with the ball in the horizontal direction”, as in Breakout, to “keep the paddle aligned with the ball in the vertical direction”, as in Pong. Such a player would exhibit very rapid cross domain generalization from Breakout to Pong.

In summary, representing relations explicitly, with a pattern or unit that remains invariant over different instantiations of the relation (e.g., left-of() in one location vs. another) and different role bindings (e.g., left-of(paddle, ball) vs. left-of(triangle, square)) affords enormous flexibility in generalization. We argue that it is precisely this kind of relational generalization that gives rise to cross-domain transfer.
Learning Relations  An account of how people learn explicitly relational representations must explain how we learn both their content and their form (Doumas et al., 2008). To account for the discovery of relational content, it must specify how we come to detect the basic relational invariants that remain constant across instances of the relation. For example, how can we discover an invariant that hold true across all instances of \textit{left-of()} in independent of location in the visual field, given that we only ever observe specific instances of \textit{left-of()} at specific locations? In order to account for the learning of the form of an structured relational representation— that is, the capacity to bind relational roles to their arguments dynamically without destroying the invariant representations of either—the system must be able to solve two problems. First, it must be able to isolate the relevant invariants from the other properties of the objects engaged in the relation to be learned. Part of what makes relation learning difficult is that although a goal is to discover an invariant representation of the relation, during acquisition relations never occur in isolation, but always in the context of specific objects engaged in the relation (e.g., it is impossible to observe a disembodied example of \textit{left-of()} as an abstract invariant). Finally, having discovered and isolated the relevant invariants, the system must learn a structured \textit{predicate} representation of the relation that can be stored in long-term memory, and when retrieved (or perceived) can be dynamically bound to arbitrary arguments while remaining independent of those arguments.

3.1.3 A Theory of Cross-domain Generalization

We propose that human cross-domain generalization a special case of analogical inference over explicitly relational representations. Accordingly, we propose that cross-domain generalization is subject to the constraints on relation learning summarized previously, plus the familiar constraints on analogical reasoning (see Holyoak, Holyoak, & Thagard, 1995; Hummel & Holyoak, 1997, 2003). Specifically, we propose that cross-domain generalization is a consequence of four fundamental operations: (1) detecting (or learning to detect) relational invariants; (2) learning structured (i.e., predicate) representations of those invariants; (3) using such explicitly relational representations to construct relational models of various domains (including arithmetic and physics, or video games) procedurally via processes like reinforcement learning; and (4) using those representations to understand new domains by analogy to familiar ones.

We do not propose that all four of these operations take place \textit{de novo} every time a person generalizes from one domain to another. In particular, if a given domain is familiar to a person, then they will already have performed steps (1)...(3) with respect to that domain. Moreover, even most novel domains are almost never completely novel. By the time a person learns Newton’s laws of mechanics, for instance, they have already mastered arithmetic, so although the equations themselves are new to the student, the arithmetic operations they represent are not. Although children likely engage actively in all four steps, by adulthood, the
majority of cross-domain transfer—whether from arithmetic to physics, or from one game to another—likely relies to an extent on step (3) learning what known relations might be relevant in a given situation, and most heavily on step (4), using existing relational concepts and domain models to make inferences about novel domains that are themselves represented in terms of relations and objects already familiar to the learner.

The following presents our model of cross-domain transfer, which performs all four of the steps outlined above: invariant discovery, relation isolation and predication, model construction, and relational inference based on those models. The model is an integration and augmentation of the LISA (Hummel & Holyoak, 1997, 2003; Knowlton, Morrison, Hummel, & Holyoak, 2012) model of analogical reasoning and the DORA (Doumas et al., 2008; Doumas & Martin, 2018) model of relational learning and cognitive development. LISA and DORA account for over 100 major findings in human perception and cognition, spanning at least seven domains: (a) shape perception and object recognition (Doumas & Hummel, 2010; Hummel, 2001; Hummel & Biederman, 1992); (b) relational thinking (Choplin & Hummel, 2002; Hummel & Holyoak, 1997, 2003; Krawczyk, Holyoak, & Hummel, 2004, 2005; Kroger, Holyoak, & Hummel, 2004; Kubose, Holyoak, & Hummel, 2002; Taylor & Hummel, 2009), (c) relation learning (Doumas & Hummel, 2012; Doumas et al., 2008; Jung & Hummel, 2015a, 2015b; Livins & Doumas, 2015; Livins, Doumas, & Spivey, 2016), (d) cognitive development (Doumas et al., 2008; Licato, Bringsjord, & Hummel, 2012; Lim, Doumas, & Sinnett, 2013; Sandhofer & Doumas, 2008), (e) language processing (Doumas & Martin, 2018; Martin & Doumas, 2017, 2019a; Rabagliati, Doumas, & Bemis, 2017), (f) cognitive aging (Viskontas, Morrison, Holyoak, Hummel, & Knowlton, 2004), and (g) decline due to dementia, stress, and brain damage (Morrison, Doumas, & Richland, 2011; Morrison et al., 2004). Accordingly, we view these systems as a promising starting point for an account of human-level cross-domain generalization. Importantly, LISA provides a solution to problem (4) above (the problem of inference), and DORA provides a solution to problem (2) (learning structured representations from non-structured inputs).

The current model integrates LISA and DORA into a single framework, and then extends the resulting model to address problem (1) (the discovery of abstract relational invariants) and problem (3) (model construction via reinforcement learning).

Our core theoretical claims and their instantiation in the proposed model are summarized in Table 3.1. These claims along with the core claims of the previous LISA/DORA papers (Doumas et al., 2008; Hummel & Holyoak, 1997, 2003) compose the primary assumptions of the approach. We claim that structured relational representations underlie cross-domain generalization as a natural consequence of (a) their general applicability across domains, and (b) their ability to underlie analogies between domains. Cross-domain analogical inference occurs because we build models of domains consisting of an open-ended vocabulary of relations among of the elements the domains (see also, Carey, 2000; Murphy & Medin, 1985). These representations are structured and relational in that they express the invariant content
of the relations they specify in a structured (i.e., symbolic) format that dynamically binds arguments to relational roles. We learn both the content and the format of structured relational representations from experience by explicitly comparing examples. We learn which relations are important for characterizing and acting in a domain through a process of reinforcement learning.

**Table 3.1:** Core theoretical claims and their instantiation in DORA

<table>
<thead>
<tr>
<th>Core theoretical claim</th>
<th>Instantiation in DORA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Cross-domain generalization is a natural consequence of structured relational</td>
<td>DORA represents relations between elements in a format that makes explicit the relational invariants, the bindings of relational roles to arguments, and the integration of multiple bindings into propositions, and the capacity for cross-domain generalization follows from operations over these representations.</td>
</tr>
<tr>
<td>representations, which express the invariant content of relations and compose dynamic</td>
<td>(a) Structured relational representations support cross-domain generalization because they are generally applicable across domains. DORA’s representations can be applied promiscuously to characterize genuinely new situations. DORA uses the representations that it has learned in the past to represent novel situations by dynamically binding previously learned predicate representations to objects in those situations.</td>
</tr>
<tr>
<td>role-argument bindings into propositions.</td>
<td>(b) Cross-domain generalization is a case of analogical inference over domain models. DORA performs analogical mapping, discovering correspondences between situations based on shared relational structure.</td>
</tr>
<tr>
<td>2. Structured relational representations are acquired by a comparison process that</td>
<td>DORA discovers invariant relational properties by exploiting properties inherent to rate-coded representations of magnitude. DORA then learns structured relational representations of these properties through a process of comparison-based intersection discovery and Hebbian learning.</td>
</tr>
<tr>
<td>reveals and isolates relational invariants and composes them into predicates that can</td>
<td>DORA and its representations integrate smoothly with existing methods for reinforcement learning.</td>
</tr>
<tr>
<td>bind dynamically to their arguments.</td>
<td></td>
</tr>
<tr>
<td>3. Relations relevant for characterizing and acting in a domain are learned procedurally</td>
<td></td>
</tr>
<tr>
<td>through reinforcement learning.</td>
<td></td>
</tr>
</tbody>
</table>

The remainder of the paper proceeds as follows. First, we summarize our integration of the LISA/DORA frameworks and describe the current extensions for invariant discovery and reinforcement learning. Next, we report simulations demonstrating that (a) the model learns structured relational representations from simple visual inputs without assuming a vocabulary
3.1. Introduction

of structured representations a priori. (b) These representations support the development of
more complex domain models. (c) The model uses these representations to generalize to
a new domain in a single exposure, exhibiting zero-shot transfer. (d) Generalization in the
model fails without the structured format of the representations it learns. (e) The representa-
tions that the model learns from simple domains transfer readily to more complex tasks like
adult analogy problems, and the representations that the model learns meet the hallmarks
of human relational cognition. (f) The trajectory of the model as it learns closely mirrors
the developmental trajectory of human children and that the representations learned in one
domain transfer readily to laboratory tasks, allowing the model to capture several phenomena
from the developmental literature. Finally, we discuss some implications and possible future
extensions of the model and contrast our approach with purely statistical accounts of human
learning.

3.2 The Model

As noted previously, our model of cross-domain generalization is based on an integration
and augmentation of the LISA and DORA models of relational reasoning (henceforth, simply
DORA). We begin by reviewing how DORA represents relational knowledge, how it uses those
representations for reasoning, and how it learns structured representations of relations from
unstructured vector-based inputs.

We next present a novel algorithm for discovering invariant relational properties—that is,
the semantic content of relational representations—from simple, nonrelational visual inputs.
The resulting model provides the first complete account of how structured representations
of visual relations can be learned de novo from simple nonrelational inputs without feedback
and without assuming a vocabulary of relations a priori. The resulting model also provides an
account of human cross-domain generalization as a natural consequence.

The following description of the model presents published details of DORA’s operation only in
broad strokes, going into detail only when those details are relevant to understanding the novel
extensions of the work (e.g., relation discovery). Complete details of the model’s operation can
be found in Appendix C (which includes a functional description of the model, pseudocode,
and full computational details). The model’s source code is available online.

1. Source code available at https://github.com/alexdoumas/BrPong_1
3.2. The Model

3.2.1 Representing Relational Knowledge: LISAese

We begin by describing the final (i.e., post-learning) state of DORA’s knowledge representations. These representations do not serve as the input to the model but are the result of its learning (described below). DORA represents propositions using a format, LISAese, which is a hierarchy of distributed and progressively more localist units whose activation oscillates over a hierarchy of progressively slower time scales (Figure 3.1A). At the bottom of the hierarchy, feature units represent the basic features of objects and relational roles in a fully distributed manner. Tokens at the lowest level of the hierarchy (T1) take inputs directly from feature units and learn, without supervision, to respond to objects and relational roles in a localist fashion. Tokens in the next layer (T2) take their inputs from PO tokens and learn, in an unsupervised fashion to respond to pairs of PO units—that is, to roles and the objects (arguments) to which they are bound. Tokens in the highest layer (T3) learn, in an unsupervised fashion, to respond to collections of RB units firing in close temporal proximity to one another.

![Figure 3.1: Knowledge representation and time-based binding in DORA.](image)

When a unit in T3 becomes active, it excites the units in T2 to which it is connected. T2 units inhibit one another, which, in combination with each unit’s yoked inhibitory unit, causes the excited T2 units to oscillate out of synchrony with one another. These same temporal dynamics are instantiated at a faster time scale in the T1 units. When an T2 unit becomes active, it excites the T1 units to which it is connected, and inhibitory connections between
those T1 units cause them to oscillate out of synchrony with one another. The result is that bound roles and objects fire in direct sequence (Figure 3.1B). For example, to represent \textit{above}(ball, paddle) (i.e., the binding of \textit{higher-than-something} to ball and \textit{lower-than-something} to paddle), the units corresponding to \textit{higher-than-something} will fire directly followed by the units corresponding to ball (Figure 3.1Ci-ii), followed by the units for coding \textit{lower-than-something} followed by the units for paddle (Figure 3.1Ciii-iv). Because of these dynamics, the network moves between stable states, with binding information carried by the sequence of such states. Thus, the network represents relational roles and fillers independently and simultaneously represents the binding of roles to fillers.

\textit{Interpreting LISAese} One advantage of the representations DORA learns is that they are easily interpretable. Units in T1 will learn to represent objects and relational roles, and by inspecting the features to which any given T1 unit is connected, it is possible to determine which object or role it represents. Units in T2 will learn to represent specific predicate-argument bindings, which are interpretable by inspecting the T1 units to which they are connected. And units in T3 will learn to represent complete propositions, which are interpretable by inspecting the connected T2 units. Accordingly, in the following, we will refer to the units DORA learns in terms of these interpretations. We do so solely for clarity of exposition. The labels we use have no meaning to DORA and no impact on its operation.

\subsection{3.2.2 Computational Macrostructure}

Figure 1 depicts the representation of an individual proposition in DORA's working memory (as synchronized and desynchronized patterns of activation) and long-term-memory (LTM; the units T1…T3). Figure 2 provides an overview of DORA's macrostructure.

A complete analog—situation, story, schema, etc.—consists of the collection of token units that collectively encode its propositional content. Within an analog a single token represents a given object, role, role-binding, or proposition, no matter how many propositions refer to that entity. For example, the same T1 unit for \textit{left-of} represents that role in all propositions containing \textit{left-of} as a role. However, separate analogs do not share tokens. For example, one unit would represent \textit{left-of} in DORA's representation of the game it is currently playing (represented in one analog) and a completely separate token would represent \textit{left-of} in DORA's representation of a game it had played in the past (represented in a separate analog). Collections of token units (i.e., T1 … T3) representing the situations and schemas (i.e., “analogs”) DORA knows collectively form its LTM (Figure 3.2).

For the purposes of learning and reasoning—for example when making an analogy between one situation and another—the propositions representing those analogs enter active memory (green box in Fig. 3.2A), a state in which they are readily accessible for processing, but not fully active (see, e.g., Cowan, 2001; Hummel & Holyoak, 1997, 2003). As depicted in Figure
3.2B, the analogs in active memory play different roles: The *driver* corresponds to the current focus of attention (e.g., the state of the current game, as delivered by perceptual processing), and maps onto one or more *recipients* (e.g., an analog describing the model’s emerging understanding of the game)\(^2\). The driver/recipient distinction in DORA is different from the more familiar source/target distinction discussed in the analogical reasoning literature. The *target* of analogical reasoning is the novel problem to be solved (or situation to be reasoned about), and the *source* is the analog from which inferences are drawn about the target. As summarized shortly, in DORA, the target analog tends to serve as the driver (i.e., the focus of attention) during memory retrieval and in the initial stages of analogical mapping, whereas the source serves as the driver during analogical inference.

![Figure 3.2: Macrostructure of DORA. (A) DORA’s long-term-memory (LTM), consisting of layers of token units (black rectangles), and the feature units connected to the bottom layer of LTM. During processing, some units in LTM enter active memory (AM). (B) Expanded view of AM. AM is composed of two sets, the driver (the current focus of attention) and the recipient (the content of working memory available for immediate processing). Black lines indicate bidirectional excitatory connections.](image)

### 3.2.3 Operation

Processing in DORA is directional, with activation starting in the driver, passing through the feature units (and any mapping connections, as detailed shortly), into other analogs in LTM (for memory retrieval), including any analogs in the recipient (for mapping, learning, and inference). Token units in the driver compete to become active, generating patterns of activation on the feature units (as described previously; Figure 3.1). Units in the recipient (or LTM) compete via lateral inhibition to respond to the resulting patterns on the feature units. This inhibitory competition is hierarchical in time, reflecting the temporal dynamics of the driver and features: T1 units (relational roles and objects) in LTM/recipient compete to respond to patterns generated by individual roles and objects in the driver; T2 units (role bindings) in LTM/recipient compete to respond to specific role/filler bindings; and T3 units

---

\(^2\) The idea that mutually exclusive sets are fundamental for analogical reasoning goes back to (Gentner, 1983), and has been implemented in a variety of models (SME, LISA, etc.). As detailed in Knowlton et al. (2012), we assume these sets are implemented by neurons in posterior frontal cortex with rapidly modifiable synapses that act as “proxies” for larger structures represented elsewhere in cortex.
3.2. The Model

3.2. The Model compete to respond to complete propositions. The result is a winner-take-all inhibitory competition operating at multiple temporal scales and serves as the foundation of all the functions DORA performs, including memory retrieval, analogical mapping (Hummel & Holyoak, 1997), analogical inference (Hummel & Holyoak, 2003), and relation discovery (Doumas et al., 2008).

Memory Retrieval Patterns of activation imposed on the feature units by active tokens in the driver will tend to activate token units in LTM that have learned to respond to similar patterns (Appendix C3 for details). For example, the features activated by a paddle in the driver will tend to activate T1 units responsive to paddle features, and the features activated by leftmost in the driver will tend to activate T1 units connected to leftmost features. Together, these T1 units will tend to excite T2 units for leftmost+paddle. Features consistent with ball and rightmost would likewise activate T1 units for ball and rightmost, which would excite a T2 unit for rightmost+ball. Together, the T2 units for leftmost+paddle and rightmost+ball will tend to activate any T3 unit(s) encoding the proposition left-of(paddle, ball): The model will have recognized the desynchronized patterns of features as representing the fact that the paddle is left of the ball, which can be retrieved into the recipient.

Mapping One of the most important operations DORA performs is analogical mapping. During mapping, DORA discovers structural correspondences between tokens in the driver and recipient. When tokens in the driver become active, similar tokens are activated in the recipient via the shared feature units. Using a kind of Hebbian learning, the model learns mapping connections between coactive units in the same layer across driver and recipient (Hummel & Holyoak, 1997, 2003, see Appendix C2.2 and Appendix C4 for details). The resulting connections serve both to represent the mappings DORA has already discovered, and to constrain its discovery of additional mappings. The algorithm provides an excellent account of human analogical mapping performance (Doumas et al., 2008; Hummel & Holyoak, 1997, 2003).

Analogical Inference Augmented with a simple algorithm for self-supervised learning (Hummel & Holyoak, 2003), DORA’s mapping algorithm also provides a psychologically and neurally-realistic account of analogical inference (making relational inferences about one situation based on knowledge of an analogous one; Appendix B.2.5 for details). The algorithm implements a version of Holyoak, Novick, and Melz (1994) copy-with-substitution-and-generalization (CWSG) framework. In CWSG, when two situations are analogically mapped, information about one situation can be inferred about the other. For example, if one knows about situation-1, where chase(Fido, Rosie), and scared(Rosie) are true, and maps that onto situation-2, where chase(Spot, Bowser) is true, one can copy the representation of the scared predic-
3.2. The Model

ate from situation-1 to situation-2, and then use the mapping of Bowser to Rosie, to copy Bowser as the argument of scared to infer that scared(Bowser) is true. As elaborated below, this process serves as the basis for our proposed solution to the problem of cross-domain generalization.

3.2.4 Learning Relational Format: LISAese from Non-Structured Inputs

DORA generalizes the operations described above to address the problem of learning structured representations of relations from unstructured “flat” vector/feature-based representations (Doumas et al., 2008). DORA represents relations as collections of linked roles, rather than as monolithic structures: For example, the relation above composes the roles higher and lower rather than consisting strictly of the single atom above, as it would in propositional notation or a labeled graph. This role-based approach to representing relations offers several advantages over alternative approaches (see Doumas & Hummel, 2005, for a review), one of which is that it makes it possible to learn relations by (a) first learning their roles and then (b) linking those roles together into multi-argument relational structures (as described in Doumas et al., 2008).

DORA’s unsupervised relation learning algorithm (Doumas et al., 2008) begins with representations of objects encoded as nonrelational vectors of features. DORA learns single-place predicates—that is, individual relational roles—as follows (see Appendix B for details): (1) By the process of analogical mapping (summarized above) the model maps objects in one situation (the driver; e.g., a previous state of the game of Breakout) onto objects in a similar known situation (the recipient; e.g., the current state of a game). For example, DORA might map a T1 unit representing the paddle in its previous location onto a T1 unit representing the paddle in its current location (Fig. 3.3Ai). Early in learning, these tokens will be holistic feature-based representations specifying all the paddle’s attributes (e.g., its location, color, etc.) in a single vector. (2) As a result of this mapping, the T1 unit representing the paddle in driver and will be coactive with the T1 unit representing the paddle in the recipient. T1 units in both the driver and recipient pass activation to the feature units to which they are connected, so any features connected to the T1 units in both the driver and recipient will receive about twice as much input—and therefore become about twice as active—as any features unique to one or the other (Fig. 3.3Aii). As a result, the intersection of the two instances becomes highlighted as the collection of most active features. (3) DORA recruits (activates) a T1 unit in its representation of the current game (i.e., the recipient), and updates their connections to the feature units via simple Hebbian learning (Fig. 3.3Aiii): The new T1 unit encodes the shared features (the intersection) of the mapped objects. If the compared objects are both, say, higher than something—and so have the features of higher in their vectors—then DORA will learn an explicit representation of the features corresponding to being higher. (In the following section we describe how features implicitly encoding this sort of relative information can be
learned and extracted from absolute information rate coded representations as delivered by the perceptual system.) (4) The resulting token (Fig. 3.3Aiv) can now function as single-place predicate (i.e., relational role), which can bind to new arguments (i.e., other units in T1) by asynchrony of firing (see Fig. 3.1). Applied iteratively, this kind of learning produces progressively more refined single-place predicates (Doumas et al., 2008).

The same Hebbian learning algorithm links token units into complete propositions in the recipient by allowing tokens units in successive layers to integrate their inputs over progressively longer temporal intervals (Doumas et al., 2008, see Appendix C2.3.1 for details). This algorithm exploits the fact that objects playing complementary roles of a single relation will tend to co-occur in the environment. For example, the representation of a ball to the higher than something (i.e., higher-than-something(ball)) will systematically co-occur with another object (e.g., the paddle), which is lower than something (e.g., lower-than-something(paddle); Fig 3.3Bi). When two sets of co-occurring role-argument pairs are mapped (e.g., an instance where ball is higher-than-something and a paddle is lower-than-something is mapped to an instance where a paddle is higher-than-something and a ball is lower-than-something; Fig. 3.3Bi), a diagnostic pattern of firing emerges (see Appendix B for details): (1) specifically, the T2 units coding each predicate-argument binding will oscillate systematically across both driver and recipient (Fig. 3.3Bii-iii). (2) In response, DORA recruits a unit in T3 that learns (via Hebbian learning) connections to the T2 units as they become active (Fig. 3.3Bii-iii). (3) The resulting representation encodes a multi-place relational structure (equivalent to above (paddle, ball); Fig. 3.3Biv). Applied iteratively over many examples, this algorithm learns abstracted structured representations describing a domain in terms of the properties and relations that characterize that domain (Doumas et al., 2008).

Doumas et al. (2008) demonstrated that this algorithm for relation discovery provides a powerful account of numerous phenomena in cognitive development. However, although the Doumas et al. algorithm can learn the form of a relational representation from repeated exposures to nonrelational (i.e., nonstructured) inputs, Doumas et al. provided little basis for discovering the invariant content of those relations—that is, the relational features themselves. Instead, their simulations were based mostly on hand-coded invariant features. The next section describes a novel algorithm for discovering invariant relational features from non-relational inputs.

---

3. In the above example we describe learning a 2-place relation composed of 2 role-filler pairs for the purposes of brevity. DORA learns relations of arity n by linking n role-filler pairs (e.g., a 3-place relation is composed of three role-filler pairs; see Doumas & Hummel, 2012).
3.2. The Model

Figure 3.3: Representation learning in DORA. (A) Learning a single-place predicate representation by comparing two objects. (i) A representation of a ball in the driver is mapped (red double-headed arrow) to a different representation of a ball (e.g., from a different game screen) in the recipient. (ii) The representation of the ball in the driver activates the mapped unit in the recipient (through shared features and mapping connection); as units pass activation to their features, shared features become more active (red units) than unshared features (pink units). (iii) Units in T1 and T2 are recruited (activation clamped to 1; blue units) in the recipient, and weighted connections are learned via Hebbian learning (i.e., stronger connections between more active units). (iv) The result is an explicit representation of the featural overlap of the two balls—in this case the property of being higher-than-something (see main text)—that can be bound to an argument (as in Figure 1). (B) Learning a multi-place relational representation by linking a co-occurring set of role-argument pairs. (i) A representation of a ball that is higher-than-something and a paddle that is lower-than-something is mapped to a different representation of a paddle that is higher-than-something, and a ball that is lower-than-something (e.g., from a different game screen). (ii) When the representation of higher-than-something(ball) becomes active in the driver it activates mapped units in the recipient; a T3 unit is recruited (activation clamped to 1; blue unit) in the recipient and learns weighted connections to units in T2 via Hebbian learning (iii) When the representation of lower-than-something(paddle) becomes active in the driver, it activates corresponding mapped units in the recipient; the active T3 unit learns weighted connections to T2 units. (iv) The result of learning is a LiSAese representation of the relational proposition above(ball, paddle) (see also Figure 3.1). Labels in units indicate what the unit encodes (see key). The labels on the units are provided for clarity and are meaningless to DORA.
3.2.5 Learning relational content: Relational Invariant Discovery

To learn an abstract representation of a relation that remains invariant with the relation’s arguments, there must be at least one invariant that characterizes that relation. For example, to learn a representation of right-of that captures every instance of right-of-ness, there must be detectable property(ies) that remain constant over all instances of above-ness (see, e.g., Biederman, 2013; Harnad, 1990; Hummel & Biederman, 1992; Kellman, Burke, & Hummel, 2020). Most previous work on relational perception and thinking has tacitly assumed the existence of such invariants (e.g., Anderson, 2007; Doumas et al., 2008; Falkenhainer et al., 1989; Hummel, 2001; Hummel & Biederman, 1992; Hummel & Holyoak, 1997, 2003). But unless all these invariants are assumed to be innate, there must be some basis for discovering them from representations of values on the underlying dimensions over which the relations are defined (e.g., somehow discovering the notion of right-of by observing examples of objects arrayed in the horizontal dimension of space).

Part of what makes invariant discovery difficult is that it poses a kind of chicken and egg problem: An invariant only seems to be discoverable in a non-invariant input if one knows to look for that invariant in the first place. Consider an invariant like “square”. Of all the possible arrangements of pixels on a computer screen, some of them form squares and others do not. Whether a set of pixels forms a square does not depend on the color of the pixels, the color of the background, the locations of the pixels on the screen, or their distances from one another: Provided they are arranged relative to one another in a way that forms a square, then they satisfy the invariant “square”. “Square” is a higher-order relational property that is independent of—that is, invariant with—the properties of any of the pixels composing it. Making matters more complicated, “square” is only one of an infinity of such higher-order relational invariants one could find in visual images. Others include rhombi, various triangles, and countless random-looking clouds of points. All these configurations are defined by the spatial relations among sets of points, so any one of them could, in principle, become a perceptual invariant like “square”. But not all of them do. Why do we recognize “square” as an invariant, but not any of the nearly infinite random looking (but nonetheless invariant) clouds of points?

Following this intuition, we developed a simple relational invariant discovery circuit (henceforth, simply relational invariance circuit) to discover the invariants greater-than, equal-to, and less-than on any metric dimension, m. This circuit exploits computational properties that naturally emerge whenever magnitudes are rate coded, either in terms of the number of units that fire in response to a given magnitude, or in terms of the rates at which individual neurons fire. The basic idea is that for any magnitude represented as a rate code, computing relations such as greater-than, less-than and equal-to is a straightforward matter of responding to the
difference between two rates. The sign of this difference (+, 0, or −) becomes an invariant signature of the categorical relations greater-than, less-than, and equal-to, respectively; and by summing over greater-than and less-than, this same operation yields the invariants same and different with respect to the dimension in question.

Let \( \mathbf{m} \) be an \( n \)-dimensional vector space, for example a collection of neurons that codes a simple magnitude, \( m \), such as size. The vector \( \mathbf{a} \) (in \( \mathbf{m} \)) then represents an object with size \( a \), and \( \mathbf{b} \) represents size \( b \). Armed with these rate codes, the difference between sizes \( a \) and \( b \), \( E_{a,b} \), is the directional difference (e.g., Gallistel & Gelman, 2000; Zorzi, Stoianov, & Umiltà, 2005):

\[
E_{a,b} = \sum_i (a_i - b_i)
\]  

(3.1)

When \( a \) is larger than \( b \), \( E_{a,b} \) will be positive; when \( a \) is smaller than \( b \), \( E_{a,b} \) will be negative; and when they are equal, \( E_{a,b} \) will be zero.

Using unsupervised learning, it is straightforward to exploit this regularity to train units to respond explicitly to whether any metric values \( a \) and \( b \) are equal, unequal with \( a > b \), or unequal with \( a < b \). The resulting neurons will be invariant representations of the relations equal, greater-than, and less-than, on metric dimension \( m \), for any rate-coded \( m \). In the language of LISAese, the resulting units could serve as feature units of dimension-specific relations such as right-of, above, larger, etc.

**Circuit Architecture** The circuit begins with a rate-coded representation of a metric dimension, \( m \) (e.g., size, or location in the horizontal dimension of the visual field; “Feature units \( m \)” in Figure 3.4). These units share bidirectional excitatory connections with a collection of T1 units (like role units in DORA), which mutually inhibit one another, and each of which excites a single “proxy” unit. As noted previously, each T1 unit codes for a single property (e.g., size) of a single object in a visual display. The proxy units in turn excite a collection of four E units (for their relation to Eq. 3.1), which excite feature units outside the set of features representing \( m \) (“Feature units non-\( m \)” in Figure 3.4A). All other connections depicted in Figure 3.4 start with weights of 1.0. The connections to and from feature units change during learning. Initially, the connection weights, \( w_{ij} \), from each \( E \) unit, \( j \), to a collection of 10 feature units, \( i \), are initialized randomly with values between 0 and 1. As detailed below, after learning each \( E \) unit is most strongly connected to a different subset of the non-\( m \) feature units, and these subsets become representations greater-than, less-than, and equal-to (as in Figure 3.4A).

4. We have left the arguments \( a \) and \( b \) out of the relational expressions here because these units represent invariant relational content, but by themselves do not specify how the roles of those relations bind to arguments; that is, they do not specify the relational format.
3.2. The Model

Figure 3.4: (A) The relational invariance circuit. (B) (i) Activation flows from clamped feature units encoding a dimension or property to T1 units. (ii) T1 units compete via lateral inhibition to respond to the active feature units. (iii) T1 units activate proxy units, which feed activation to E units. E units pass activation to a subset of feature units and connections between active feature units and T1 units are updated via Hebbian learning. (iv) The active T1 unit is inhibited to inactivity by its inhibitor (blue square). (v-vi) The process repeats for the second active T1 unit.

Circuit Operation  Processing in the relational invariance circuit is assumed to begin after perceptual processing has segmented the image into objects and encoded each object in terms of its various attributes (e.g., size, location in the horizontal and vertical dimensions, etc.). As elaborated under Simulations, DORA is equipped with a simple perceptual preprocessor that accomplishes this segmentation and encoding. The T1 units depicted in Figure 4 correspond to a subset of this encoding (i.e., representing each object’s size). For the purposes of illustration, we shall assume that each T1 unit in Figure 4 represents the size of one object (in a display containing two objects, \(a\) and \(b\)). For simplicity, we also assume that the pattern of activation on the feature units representing dimension \(m\) is the superposition (i.e., sum) of the vectors, \(m_1\), representing the sizes of the two objects in the display. The learning algorithm does not require the feature inputs, \(m_1\), to be segmented into separate objects, \(i\); instead, it is sufficient to encode this information in the connections from \(m_1\) to T1.

Once the objects and their attributes are encoded by the preprocessor, the superimposed vector \(m = m_a + m_b\) is clamped on the subset of feature units, \(m\), representing \(m\) (Figure 3.4B(i)). This vector serves as input to the T1 units, which compete via lateral inhibition to become active. The input to the T1 units is given by:

\[
n_i = \sum_j a_{wi} - \sum_{k \neq i} a_k - LI
\]

where \(j\) are feature units connected to T1 unit \(i\), \(k\) are other active T1 units \((k \neq i)\), and \(LI\) is the activation of the local inhibitor (a refresh signal given when no active T1 units are active in the driver (as in, Doumas et al., 2008; Horn, Sagi, & Usher, 1991; Horn & Usher, 1990; Hummel & Holyoak, 1997, 2003; Usher & Niebur, 1996; von der Malsburg & Buhmann, 1992, see Appendix C1.4 for details). Activation of T1 units is calculated as:
3.2. The Model

\[ \Delta a_i = \gamma n_i (1.1 - a_i) - \delta a_i \]  

(3.3)

where \( \Delta a_i \) is the change in activation of unit \( i \), \( \gamma = 0.3 \) is a growth parameter, \( n_i \) is the net input to unit \( i \), and \( \delta = 0.1 \) is a decay parameter.

Since the vectors \( \mathbf{m}_a \) and \( \mathbf{m}_b \) are superimposed on the feature units, the T1 unit with more connections to the feature units (in this case, the unit coding for the larger size), say T1\(_a\), will initially win the inhibitory competition, inhibiting T1\(_b\) to inactivity. For this reason, the T1 unit with the larger input vector will always win the initial inhibitory competition (Figure 3.4Bii). (The T1 units in this circuit are otherwise identical to other T1 units in DORA.)

Each proxy unit, \( i \), has a connection weight of 1 from T1\(_i\), and a weight of 0 from all other T1\(_j \neq i\). A proxy unit is simple binary threshold unit whose activation is given by:

\[ p_i = \begin{cases} 1, & n_i \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \]  

(3.4)

where, \( p_i \) is the activation of proxy unit \( i \).

Input, \( n_i \), to proxy unit \( i \) is calculated as:

\[ n_i = \sum_j a_j w_{ij} - \rho_i \]  

(3.5)

where, \( j \) is an active T1 unit and \( \rho_i \) is the refraction of unit \( i \). The refraction, \( \rho_i \) is given:

\[ \rho_i = \frac{1}{1 + 10^{x}} \]  

(3.6)

where, \( x \) is the number of iterations since unit \( i \) last fired, and \( t = 1 \times 10^{-7} \) is a scaler. Proxy unit \( i \) will be active if and only if T1\(_i\) is active (i.e., \( a_i > 0.5 \)) and proxy unit \( i \) has not recently been active.

\( E \) units take their inputs from the proxy units. The connections from proxy units to \( E \) units have temporal delays built into them, so that each \( E \) unit has a Gaussian receptive field in the three-dimensional space formed by the two T1 cells’ activations, plus time. Input to \( E \) units is given by Eq. 3.5 such that \( j \) are proxy units, and change in activation is calculated using:

\[ \Delta a_i = \gamma e^{-\left(\frac{(n_i - \theta_E)^2}{2t^2}\right)} - 0.1a_i - 0.5 \sum_j a_j - LI \]  

(3.7)
3.2. The Model

where, $\gamma$ is a growth parameter, $\theta_E$ is the threshold on unit $E$, $k = .2$, $j$ are all other units in $E$ $j \neq i$. The circuit contains four $E$ units each with a $\gamma$ of .1 or .3, and a $\theta_E$ of 1 or 2, such that all four combinations of $\gamma$ and $\theta_E$ values are present in an individual $E$ unit. As a result, some $E$ units respond preferentially to proxy units firing early in processing, others respond preferentially to proxy units firing later, and still others respond preferentially to the two proxy units firing at the same time. Like the T1 units, $E$ units laterally inhibit one another to respond in a winner-take-all fashion so that only one $E$ unit tends to become active in response to any (temporally-extended) pattern of activation over the proxy units.

$E$ units are randomly connected to a collection of feature units that are not part of the vector space $m$ (henceforth, non-$m$ features; Figure 3.4). Active $E$ units, $i$, both excite the non-$m$ feature units, $j$, and during learning, update their connections to them. Feature units connected to $E$ units update their input by:

$$n_i = \sum_j a_i w_{ij}$$  \hspace{1cm} (3.8)

where $i$ and $j$ are feature units and $E$ units respectively. Feature unit activation is updated as:

$$a_i = \frac{n_i}{\max(n_j)}$$  \hspace{1cm} (3.9)

where $a_i$ is the activation of feature unit $i$, $n_i$ is the net input to feature unit $i$, and $\max(n_j)$ is the maximum input to any feature unit. There is physiological evidence for divisive normalization in the feline visual system (e.g., Bonds, 1989; Foley, 1994; Heeger, 1992) and psychophysical evidence for divisive normalization in human vision (e.g., Thomas & Olzak, 1997). While the energy circuit is being learned, connections between units in $E$ and feature units are updated by the equation:

$$\Delta w_{ij} = a_i(a_j - w_{ij})\gamma$$  \hspace{1cm} (3.10)

where, $i$ and $j$ refer to units in $E$ and feature units respectively, and $\gamma = .1$ is the growth parameter.

When the circuit is running, connections between the feature units and active T1 units are updated by Eq. 3.10 (Figure 3.4Biii). As a result, $T_{1_a}$ (the T1 unit we assumed won the initial inhibitory competition) will learn positive connections to whatever feature unit(s) the active $E$ unit has learned to activate (recall that the connections from $E$ units to non-$m$ features are initially random). Importantly, the active non-$m$ feature units, $j$, to which $T_{1_a}$ has learned
connections effectively represent greater-than: These features will become active whenever the E unit that responds to early firing proxy units. In short, T1a has gone from representing a particular value on m to representing the conjunction of that value (it is still connected to the features in m) along with the relational invariant greater-than.

Recall that the T1 units are oscillators (see also Appendix B). As a result, after some iterations have transpired with T1a active, that unit’s inhibitor will become active, inhibiting T1a and allowing T1b to become active (Figure 3.4Biv). The same operations described with respect to T1a will take place with respect to T1b, which will learn connection(s) to feature unit(s) representing less-than (Figure 3.4Bv-vi).

After these operations have taken place on a single pair of objects, DORA will have constructed a representation of that pair of objects, each with a specific value on metric dimension m (represented by T1a and T1a in Figure 3.4) explicitly tagged with an invariant specifying that its value is either greater-than (in the case of T1a) or less-than (T1b) some other value on m. Importantly, these representations do not yet constitute an explicitly representation of the relation greater-than(a, b), because they are not linked into a propositional structure (e.g., by T2 and T3 units) specifying that the individual roles, greater-than and less-than, are linked into a single relation. Moreover, the emerging representation of these roles, as instantiated in the feature units connected to T1a and T1b, still retain a full representation of the specific metric values of T1a and T1b on m. In other word, T1a and T1b do not represent greater-than and less-than in the abstract, but instead represent something closer to greater-than-and-value-a and less-than-and-value-b. However, this kind of almost-relational representation, in which relational invariants are present, but (a) are still associated with other, nonrelational features (e.g., specific values on m), and (b) are not yet composed into an explicitly relational structure, are precisely the kind of representations DORA uses as the starting point for learning explicitly structured relations (see above).

To illustrate, consider what will happen when a different pair of objects, c and d, are engaged in the process described above. For the purposes of illustration, assume that c has a larger value on m than d does, but both have different values than a and b. The processes described previously will attach T1c to the same invariant greater-than feature(s) as T1a and T1d to the same less-than feature(s) as T1b. It is in this sense that those feature units represent greater-than and less-than, respectively: The relational invariance circuit will, by virtue of the E unit-to-feature connections learned in the context of objects a and b, be biased to activate the feature unit(s) recruited for greater-than in the context of T1a in response to the “greater-than-ness” of T1c, and the feature unit(s) recruited for less-than in response to the “less-than-ness” of T1d. If, subsequently, DORA compares the pair [a, b] to the pair [c, d], it will learn predicates
(T1 units) strongly connected to greater-than and less-than, and only weakly connected to the specific values of \( a \) and \( c \), and \( b \) and \( d \), respectively (see Doumas et al., 2008, see also Appendix B). After exposure to as few as two pairs of objects, DORA has started to explicitly predicate the invariant relation \( \text{greater-than}(x, y) \) with respect to dimension \( m \).

In fact, DORA has learned something much more general than that, because the invariant features greater-than and less-than, learned in the context of metric dimension \( m \), will generalize to any other rate-coded metric dimension, \( n \neq m \). The reason is that the rate code that serves as the input to the relational invariance circuit operates on the magnitudes of the T1 units’ inputs, regardless of their origin: So long as the T1 units in question are (a) coupled oscillators that compete with one another to become active, and (b) receive rate-coded inputs from whatever metric dimension they represent, so that (c) the T1 unit with the larger value fires earlier than the T1 unit connected to the smaller value, the T1 unit connected to the larger value of the dimension will become connected to greater-than and the T1 unit connected to the smaller value on the dimension will become connected to less-than.

This same property of the relational invariance circuit renders it vulnerable to incorrectly assigning the invariants more-than and less-than to any pair of object properties (represented as T1 units) that get passed into the circuit, even if those properties do not lie on a metric dimension. For this reason, there need to be constraints on when the relational invariance circuit is invoked. One obvious constraint is that it should only be invoked when the property coded by a T1 unit is a value on a metric dimension. As discussed previously, additional constraints, for example, regarding which metric dimensions are most likely to invoke the circuit under what circumstances, are likely also important, but consideration of what those constraints are is beyond the scope of the current work (but see Spelke & Kinzler, 2007).

3.2.6 Learning the Content and the Format of Relational Representations

By integrating the relational invariance circuit with the DORA algorithm for learning structured representations of relations (Doumas et al., 2008), we have developed a single system that explains learning structured relational representations of similarity and relative magnitude from very simple (non-relational) beginnings without assuming any structured representations, or even relational invariants, \textit{a priori}. The model starts with flat feature vector representations of object properties. These vectors contain no relational features, just absolute information about properties and magnitudes along dimensions. As described above, when objects with these feature encodings are compared, invariant patterns emerge, which mark similarities and differences in featural encoding and absolute magnitudes. The relational invariance circuit exploits these patterns to identify relational instances and return invariant features of identifying those relations.
3.2. The Model

The DORA learning algorithm identifies invariant features of compared objects and learns structured representations of those features in a format akin to a single-place predicate. The model then links systematically occurring predicate-argument bindings to develop functional multi-place predicate representations. That is, over a series of progressive comparisons, the model isolates collections of object features, represents these as functional single-place predicates, links systematically co-occurring single-place predicates to form multi-place predicates, and produces increasingly more refined versions of these representations. When the representational content of these objects is relational, DORA will learn structured representations of this relational content. In the next section we present a series of simulations to assess relation learning in the model, and then to evaluate the capacity of these representations to support relational thinking and, critically, cross-domain generalization.

Finally, as demonstrated in the simulations, the representations DORA learns integrate naturally with reinforcement learning to allow DORA to learn which relations to attend to and apply in what contexts.

3.2.7 A Mechanism for Generalization

We propose that operations on relational representations underlie human generalization and that generalization based on relations occurs in (at least) two ways. First, relational representations learned in one context are readily applicable to characterize new contexts. Relational representations are useful for characterizing multiple domains because the same relations apply across domains regardless of the objects involved. Second, theories and schemas learned from one domain allow us to make inferences about other domains using analogical inference.

Analogical inference—in this case, using a model of one domain to reason about another domain—follows directly from DORA’s mapping process. For example, suppose that DORA has learned about spatial relations (e.g., above, right-of, larger) and then learned that when playing the game Breakout—where the goal of the game is to hit a ball with a paddle moving horizontally—relations between the ball and paddle predict actions to take. Specifically, DORA has learned that the state right-of(ball, paddle1) supports moving right (i.e., right-of(paddle2, paddle1); where paddle1 is the state of the paddle before the move, and paddle2 after the move), the state left-of(paddle1, ball) supports moving left, and the state same-x(ball, paddle1) supports making no move. When DORA encounters a game like Pong—where the goal of the game is to hit a ball with a paddle moving vertically—the moves available in Pong (up and down) might remind DORA of the moves available in Breakout (left and right). With the representation of the available Pong actions in the driver (e.g., above(paddle2, paddle1), and representations of the Breakout strategy retrieved into the recipient (e.g., right-of(ball, paddle) → right-of(paddle2, paddle)), DORA performs analogical
3.2. The Model

mapping. Because of the shared relational similarity, corresponding moves between Pong and Breakout map—e.g., \textit{above}(paddle2, paddle1) in the driver will map to \textit{right-of}(paddle2, paddle1) in the recipient (Figure 3.5; Appendix C2.2 and Appendix C4 for details of how such mappings are discovered). Generalization is performed on the basis of these mappings.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Figure3.5.png}
\caption{Graphical depiction of analogical inference in DORA. The representation of the \textit{right-of}(ball, paddle1), and \textit{right-of}(paddle2, paddle1) in the driver maps to the representation of \textit{above}(paddle2, paddle1) in the recipient (red double-arrowed lines indicate mappings). As the representation of \textit{right-of}(ball, paddle1) becomes active in the driver, some active units have nothing to map to in recipient (i.e., the units representing ball, more-x+ball, less-x+paddle1, and \textit{right-of}(ball, paddle1)). DORA recruits and activates units to match the unmapped driver units (blue units indicate recruited units; blue double-headed arrow indicates matched units). DORA learns connections between co-active token units in the recipient (heavier blue lines). The end result is a representation of the situation: \textit{above}(ball, paddle1) & \textit{above}(paddle2, paddle1) in the recipient.}
\end{figure}

During analogical generalization, propositions with unmapped elements enter the driver, and any propositions to which they map enter the recipient (information is generalized from the driver to recipient; see Hummel & Holyoak, 2003). So, if DORA has mapped \textit{above}(paddle2, paddle1) to \textit{right-of}(paddle2, paddle1), the representation of the rule from Breakout enters the driver, and the mapped representation of the move from Pong enters the recipient. Figure 3.5 depicts a case where the rule \textit{right-of}(ball, paddle) \rightarrow \textit{right-of}(paddle2, paddle) is in the driver, and the mapped \textit{above}(paddle, paddle2) is in the recipient (mappings depicted as red double-arrowed lines). When a unit, $i$, in the driver learns an excitatory mapping condition to a given unit, $j$, in the recipient, it also learns a global inhibitory mapping connection to all other units, $k \neq j$, in the recipient. Similarly, $j$ learns a global inhibitory connection to all units $i \neq l$ in the driver. These global inhibitory connections play a vital role in the inference process. Continuing the example, all units encoding the representation of Pong in the recipient have a positive mapping connection to a corresponding unit in the driver (i.e., \textit{right-of} maps to \textit{above},
3.2. The Model

paddle1 maps to paddle1 and paddle2 maps to paddle2; Figure 3.5). By consequence, all units in the recipient also have a global inhibitory mapping connection to all other units in the driver. Therefore, when more-x(ball) becomes active in the driver (along with the T2 unit encoding more-x+ball and the T3 unit encoding right-of(ball, paddle1)), the T1 unit encoding ball inhibits all T1 units in the recipient (except for more-y, which is excited by more-x), the T2 unit encoding more-x+ball inhibits all T2 units in the recipient, and the T3 unit representing right-of(ball, paddle2) inhibits all T3 units in the recipient.

This form of generalized inhibition occurs when all units in the recipient map to some unit in the driver, and no units in the recipient map to the currently active driver units. That is, the signal indicates that there are elements in the driver that map to nothing in the recipient. This occurrence signals DORA to initiate analogical inference. During analogical inference, DORA recruits and activates units in the recipient that corresponds to the unmapped (i.e., inhibitory) unit in the driver (e.g., DORA recruits a T1 unit in the recipient corresponding to an unmapped T1 unit in the driver). Newly recruited units are assigned positive mapping connections with the driver units that initiated their recruitment (blue double-headed arrows in Figure 3.5), and they are learn connections to other recipient units by simple Hebbian learning (e.g., active T1 units learn connections to active T2 units and active feature units). When more-x(ball) is active in the driver, the driver T1 unit representing more-x activates the recipient T1 unit more-y; as they map, however, the active ball T1 unit, more-x+ball T2 unit, and above(ball, paddle1) T3 unit will activate nothing and inhibit all recipient units (as they map to nothing in the recipient). In response to this generalized inhibition in the recipient, DORA recruits a T1 unit corresponding to the unmapped active driver T1 unit (representing ball), a T2 unit corresponding to the active unmapped driver T2 unit (representing more-x+ball), and a T3 unit is recruited corresponding to the active unmapped driver T3 unit (representing right-of(ball, paddle1); blue units in Figure 3.5). The recruited T1 unit learns connections to the active features of ball and to the recruited T2 unit (as they are all co-active; blue connections in Figure 3.5). The recruited T2 unit learns connections to the recruited T3 unit (as they are co-active), and then to the T1 unit representing more-y when it is active (blue connections in Figure 3.5).

As such, the recruited T1 unit becomes a representation of ball, and the recruited T2 links the representation of ball and more-y(more-y+ball). Similarly, when less-x(paddle1) becomes active in the driver (activating the less-y and paddle1 T1 units in the recipient), a T2 unit will be recruited to match the unmapped active driver T2 unit. That T2 unit will learn connections to the less-y and paddle1 T1 units, and to the active recruited T3 unit (which remains active as its corresponding driver T3 unit remains active). The result is a representation of above(ball, paddle1) in the recipient (Figure 3.5).
3.2. The Model

The same process also accounts for how we might predicate (or explicitly represent) known relations about new situations. As a simple example, suppose DORA encounters two objects involved in a relation such as when object-1 is above object-2. Those objects will have properties of that relation (e.g., object-1 might have features such as “more” and “y”; delivered by the relation invariance circuit). If DORA has learned explicit structured representations of the relation above (e.g., two linked predicates strongly connected to the features “more” and “y” and “less” and “y” respectively), then it might retrieve a representation of that relation from LTM, say \( \text{above}(P, Q) \). The retrieved relational representation can then be projected on to object-1 and object-2 (Fig. 3.6). Specifically, P, the higher item, will correspond to object-1, and Q, the lower item, will correspond to object-2. Based on these correspondences, DORA will infer the predicates bound to P and Q about object-1 and object-2 via the analogical inference algorithm. Shifting focus to the \( \text{above}(P, Q) \) proposition (\( \text{above}(P, Q) \) is in the driver; Figure 3.6), when the higher unit becomes active it inhibits all units in the recipient, signaling DORA to recruit a T1 unit in the recipient to match the active higher unit and a T2 in the recipient to match the active higher \(+P\) unit. The recruited T1 unit learns connections to the active feature units, becoming an explicit representation of higher, and the recruited T2 unit learns connections to the recruited T1 unit (representing higher) and the object-1 unit when it becomes active (Fig. 3.6). Similarly, a representation of lower is represented about object-2. The result is the known relation above predicated about object-1 and object-2 (Fig. 3.6).

Figure 3.6: Graphical depiction of generalizing known relations to new situations using the analogical inference algorithm in DORA. The representation of the above(object-X, object-Y) in the driver is mapped to the representation of object-1, object-2 in the recipient (red double-arrowed lines indicate mappings). As the representations of higher(object-X) and lower(object-Y) become active in the driver, some active units have nothing to map to in the recipient (e.g., the driver units representing higher and higher+object-X, and the units representing lower and lower+object-Y). DORA recruits and activates units to match the unmapped driver units (blue units indicate recruited units; blue double-headed arrow indicates matched units). DORA learns connections between co-active token units in the recipient (heavier blue lines). The end result is a representation of the situation: \( \text{above}(\text{object-1, object-2}) \).
In the following simulations we demonstrate the efficacy of the computational account of relation learning and relation-based generalization that we have proposed. We show how DORA learns structured relational representations from simple visual non-structured non-relational inputs, and how it then uses these representations to support human-level cross-domain generalization—by characterizing novel domains in terms of known relations, and then driving inferences about the new domain based on the systems of relations learned in previous domains. Additionally, we demonstrate that that DORA model captures several key properties of human representation learning and the development of generalization.

3.3 Simulations

As described previously, explaining cross-domain generalization as analogical inference entails explaining how a system: (1) detects (or learns to detect) relational content; (2) learns structured representations of that relational content; (3) uses these representations to characterize and behave in the domains it experiences (e.g., to build models of the domain to guide behavior); (4) uses the representations learned from previously experienced domains to make analogies and subsequently inferences about new domains. In addition, there is a distinction between representing a domain and acting on those representations. Extrapolation from one domain to another relies on both using representations learned in one domain to characterise another (i.e., representational transfer), and adopting strategies from one domain for use in another (i.e., policy transfer).

Below we report a series of simulations evaluating various capacities of the model. In simulation 1, we show that the model learns structured relational representations—both their form and content—from non-structured and non-relational visual inputs without assuming a vocabulary of structured representations a priori. In simulations 2-4, we show that the model can be integrated with methods for reinforcement learning to use the representations that it learns to build more complex models (or policies) for behaving in the domain, and then use its representations to perform zero-shot (i.e., first trial) cross-domain generalization. Specifically, we show that after the model learns to play one video game (Breakout), it can generalize its knowledge to play a structurally similar but featurally very different game (Pong). Moreover, we show that generalization in the model relies exquisitely on the structured format of the representations that it learns. In simulation 5, we evaluate whether the representations that the model learned in previous simulations generalize to more complex tasks like adult analogy problems, support generalization to completely novel stimuli (i.e., approximating universal generalization), and meet the hallmarks of human relational cognition. Finally, we are proposing an account of human generalization that includes relational representation learning. As such, it should be the case that our model mirrors the capacities of children as they learn relational representations. In simulations 6 and 7 we use the learning trajectory the model
underwent during simulations 2-4 to simulate studies from the developmental literature on children’s magnitude reasoning (simulation 6) and relational problem solving (simulation 7). Additionally, simulations 6 and 7 provide further tests of the model’s capacity for cross-domain generalization: After learning representations in a domain like Breakout, will the model extend these representations to reason in the domain of a psychology experiment. The model not only generalizes the representations that it learns from one domain to reason about a new task (as children do when they enter the laboratory) but also that it goes through the same behavioral trajectory as children. Details of all simulations appear in Appendix D.

3.3.1 Visual Front End

Generalizing DORA to work with perceptual inputs, such as pixel images necessitated supplying the model with a basic perceptual front end capable of segmenting simple objects (e.g., paddles and balls) from visual displays. We endeavoured to keep this extension as simple as possible, importing existing solutions.

We used a visual pre-processor that delivers object outlines using edge detection (via local contrast) with a built-in bias such that enclosed edges are treated as a single object. In brief, the pre-processor identifies “objects” (enclosed edges) and represents them in terms of their location on the “retina”, size, and color. This information roughly corresponds to the total retinal area of the object and the enervation of the superior, inferior, lateral, and medial rectus muscles in reaching the (rough) centre of the object from a reference point (see Demer, 2002). This information is encoded as the raw pixels and direction (specific muscle) between the rough object centre and the reference point, and the RGB encoding of the pixels composing the object. One consequence of this encoding is that the model shows the same bias to classify along the cardinal directions observed in humans (Girshick, Landy, & Simoncelli, 2011).

This pre-processor is clearly a vast oversimplification of human perception. However, we chose it because it is adequate for our current purposes, it is computationally inexpensive, and the representations it generates are at least broadly consistent with what is known about human vision. For example, the visual system detects edges by local contrast (e.g., Marr & Hildreth, 1980), represents objects and their spatial dimensions (e.g., Wandell, Dumoulin, & Brewer, 2007), and that these representations and the visual image are quasi-homomorphic (e.g., Demer, 2002; Engel et al., 1994; Furmanski & Engel, 2000; Moore & Engel, 2001). We certainly do not claim the pre-processor is an accurate model of human vision; only that it is not grossly inconsistent with what is known about biological vision, and that it is adequate to our current goal, which is to model learning and generalization in the domain of simple visual images, with generalization to novel domains. In addition, it is possible to use the pre-processor as a front-end to both DORA and to the comparison DNNs, allowing us to equate the inputs used by DORA with those used by the DNNs.
3.4 Simulation 1: Unsupervised Discovery of Relations

The goal of simulation 1 was to evaluate the capacity of the model to learn, without supervision, structured representations of relational concepts from representations of non-structured representations of objects that include only absolute (non-relational) information. We ran two simulations, 1a and 1b, testing the model’s capacity to learn from both simpler and more featurally complex stimuli.

3.4.1 Simulation 1a

Simulation 1a served as a basic proof of concept. In this simulation, we tested whether DORA would learn structured relational concepts when presented with simple visual displays.

Visual displays We started with 200 two-dimensional images, each differing in shape, contrast, size, width, and height (see Figure 3.7 for examples of the images). Each of the 150 shapes was then randomly grouped with between 1 and 4 other shapes to create a total of 150 multi-object displays. The displays were then processed by the visual preprocessor. As described above, the visual preprocessor identified an object as any item with a continuous and connected edge and represented that object as a T1 unit connected to a collection of features corresponding to the pixels composing its absolute width and height (x- and y-extent), size, and vertical and horizontal deviation from the edge of the screen (x- and y-deviation). As the images were grey-scale, we left out the RGB information in this simulation. To add extraneous noise to the objects, each T1 unit encoding an object was also randomly connected to 10 “noise” features from a set of 1000. The result was 150 “scenes” containing between 2 and 5 objects, with each object represented as a set of absolute rate-coded spatial dimensions and noise features.

![Figure 3.7: Examples of the shape stimuli used for Simulation 1.](image-url)
Learning structured relational representations  We designed this simulation to mimic a child noticing a visual display (e.g., a scene) and attempting to use their memory of previous experiences to understand and learn about that display. DORA started with no representations (i.e., all weights set to 0). The representations of the 150 multi-object scenes were placed in DORA’s LTM. DORA attempted to learn from these stimuli, but it did not otherwise have any “task” to perform, and it received no feedback on its performance during the simulation. Rather, DORA performed 3000 “learning trials”. On each learning trial, DORA randomly selected one collection of objects from LTM and placed that collection in the driver, thus simulating the perception of a visual display. DORA ran driver representations through the local energy circuit, and then performed memory retrieval, analogical mapping, and representation learning (as described previously and in Appendix B). For the current simulations, we constrained DORA’s retrieval algorithm to favor more recently experienced displays (such recency effects are common in the memory literature; e.g., Logie, Camos, & Cowan, 2020). With probability \( \frac{2}{3} \), DORA attempted to retrieve from the last 100 analogs that it had learned, otherwise it attempted to retrieve from LTM generally.

In evaluating DORA’s learning, what we want to know is whether the model learned structured representations of relation content. That is, we want to know first, if the model has learned T1 units connected strongly to features defining a specific relational role concept (and only weakly to other features), and second, whether the model linked representations of complementary relational roles into multi-place relational structures. For example, if DORA learned a representation of a T1 unit connected strongly to the features encoding for “more” and “x-extent” and weakly to all other features, then it had learned a relative (relational) representation of more-x-extent. Similarly, if the model learned a representation of a T1 unit connected strongly to the features encoding for “less” and “x-extent” and weakly to all other features, then it had learned a relative (relational) representation of less-x-extent. Finally, if the model learned a full LISaeese structure wherein these T1 units (one representing more-x-extent and another representing less-x-extent) were bound to objects via T2 units and linked via a T3 unit (as in Fig. 4.1), then it had learned a structured multi-place relational representation.

To this end, we first defined a set of meaningful relational roles that the model could learn given the input images. This list comprised the set of relative encodings of the absolute dimensional information returned by the visual preprocessor: That is, the features encoding “more”, “less”, and “same” paired with the encoding of x-extent ([“more”, “x-extent”], [“less”, “x-extent”], [“same”, “x-extent”]), y-extent ([“more”, “y-extent”], [“less”, “y-extent”], and [“same”, “y-extent”]), size ([“more”, “size”], [“less”, “size”], and [“same”, “size”]), x-deviation (hereafter x; ([“more”, “x”], [“less”, “x”], and [“same”, “x”]), and y-deviation (hereafter y; ([“more”, “y”], [“less”, “y”], and [“same”, “y”]).

Next, in order to evaluate DORA’s learning of the relations in the displays, we define the relational selectivity metric, \( Q_i \), for a T1 unit \( i \) as:
3.4. Simulation 1: Unsupervised Discovery of Relations

\[ Q_i = \frac{\frac{1}{n} \sum_{j \in r'} w_{i,j}}{1 + \frac{1}{m} \sum_{k \notin r'} w_{i,k}} \]  

(3.11)

where \( r' = \arg\max_r \frac{1}{n} \sum_{j \in r} w_{i,j} \) is the relational role that maximizes the mean weight of unit \( i \) to the features, \( j = 1 \ldots n \), that make up the role’s content, and \( k = 1 \ldots m \) are all other features. \( Q_i \) scales with the degree to which unit \( i \) codes selectively for a relational role, where \( Q_i = 1.0 \) indicates that the unit responds exclusively to the features of a single relational role, \( r' \). We measured the relational specificity of the T1 units in DORA’s LTM over the course of 2500 learning trials. As Figure 3.8 illustrates, DORA learns representations (T1 units) encoding meaningful relational roles. That is, DORA learns T1 units encoding roles like more-x-extent (strongly connected only to features for “more” and “x-extent”), less-y (strongly connected only to “less” and “y”), or same-size (strongly connected only to “same” and “size”). The results indicate that DORA’s learning algorithm produces representations that encode invariant relational content.

![Figure 3.8: Mean relational selectivity of T1 units (as defined in text) as a function of number of training examples, simulation 1a.](image)

Next, we checked whether the representations DORA learned were composed into meaningful relational structures (i.e., whether representations of complementary roles were linked into multi-place structures). For example, if DORA links more-y(obj1) and less-y(obj2) to form the relation above(obj1, obj2), or links more-x-extent(obj2) and less-x-extent(obj1) to form the relation wider(obj2, obj1), then it has learned representations of meaningful relations. To this end, we checked the number of representations in LTM representing single-place predicates (a learned T1 unit linked to another T1 unit representing an object via a T2 unit but not connected to a T3 unit), meaningful multi-place relations (T1 units representing complementary relational roles, each linked to an object T1 unit via T2 units that were also linked via a single T3 unit), and meaningless multi-place relations (T1 units not representing complementary
relational roles, each linked to an object T1 unit via T2 units that were linked via a single T3 unit). As presented in Figure 3.9, DORA learns representations of meaningful relations with experience. By the 1000th learning trial, DORA had learned representations of all possible meaningful relations (i.e., above, below, same-vertical, right-of, left-of, same-horizontal, wider, thinner, same-width, taller, shorter, same-height, larger, smaller, same-size), and it learned progressively more refined representations of these relations with additional learning trials.

As described above, DORA learns multi-place relations by comparing sets of single-place predicates. During learning, this process runs in parallel with the discovery of the single-place predicates that will form the roles of these relations. However, because the linking operation depends on having a vocabulary of single-place predicates to combine, DORA necessarily follows a developmental trajectory in which it acquires single-place predicates before it acquires multi-place relations. As shown in Figure 3.9 roughly the first 300 learning trials are dominated by the discovery of single-place predicates like more-x, less-x-extent, and their complements. After that initial period, learning is dominated by the discovery of multi-place relations like above.

In contrast to error-correction learning (such as back propagation), DORA’s learning algorithm does not replace old knowledge (e.g., predicates discovered early in learning) with new knowledge (predicates learned later), but rather adds new knowledge to its existing knowledge. For example, the multi-place relations it learns do not replace the single-place predicates from which they were composed, and refined predicates and relations do not replace their less refined predecessors. However, as a consequence of DORA’s retrieval algorithm, less refined predicates become less likely to be retrieved (and thus used as the basis of new comparisons) than their (increasingly common) more-refined counterparts (the retrieval algorithm is biased
in favor of retrieving the simplest pattern that fits the retrieval cue Hummel & Holyoak, 1997). Effectively, the less refined predicates simply fall out of service as they become obsolete. As a result, there is a large difference between the predicates DORA knows (i.e., has stored in memory) and those it frequently uses.

Moreover, DORA not only learns specific relations such as above and below, but it also discovers more abstract relations such as greater-than and same-as as a natural consequence. For example, if DORA compares two different instances of wider(x, y), then it will learn a more refined representation of wider(x, y) (as described previously). But if it compares an instance of wider(x, y) to an instance of taller(z, w), then it will learn a representation that retains what wider has in common with taller, or a generic greater(a, b) relation. This result mirrors the development of abstract magnitude representations in children (e.g., Sophian, 2008).

In total, these results indicate that DORA learns structured representations of relative magnitude and similarity relations from unstructured (i.e., flat feature vector) representations of objects that include only absolute values on dimensions and extraneous noise features. However, a potential criticism of this simulation is that the starting representations are quite simple. Perhaps DORA only learns useful relational representations because each object only has five dimensions, whereas real objects have many more. Simulation 1b address this potential limitation.

### 3.4.2 Simulation 1b: Scaling Up

In this simulation we tested DORA’s capacity to learn from messier examples containing more competing and extraneous information. Just as in Simulation 1a, we created 200 scenes each containing between 2 and 5 objects. We then altered the objects in two important ways. First, we added 1000 distractor features from a pool of 100,000. Second, we added absolute encodings from 45 additional dimensions. That is, while in simulation 1a, each object was connected to rate-coded features encoding an absolute value on 5 dimensions (as delivered by the visual pre-processor), in this simulation each object was connected to rate-coded features encoding an absolute value on 50 dimensions (the five from simulation 1a, and 45 additional “dimensions”). As a consequence, each object was now much messier, containing not only more noise features, but also encoding more dimensions (that DORA could potentially learn explicit representations of). If DORA’s learning algorithm is indeed robust, then we would expect it to (a) learn relational representations of all dimensions (there should be nothing special about the 5 used in simulation 1a), and (b) learn these representations in a number of learning trials that scales proportionally with the number of items to be learned (i.e., the model should take roughly 10 times as long to learn comparably refined structured relational representations of 50 dimensions as it took to learn five).
Simulation 1b proceeded like Simulation 1a. A no-representations version of DORA was created. The representations of the 150 multi-object scenes were placed in DORA’s LTM. As in simulation 1a, DORA attempted to learn from these stimuli, but it did not otherwise have any “task” to perform, and it received no feedback on its performance during the simulation. In this simulation DORA performed 10,000 learning trials.

Figure 3.10 shows the progression of the relational selectivity of DORA’s T1 units over the course of training. Just as in simulation 1a, DORA learned progressively more refined representations of relational content. Vitally, DORA learned structured relational (relative; more/less/same) representations of all 50 dimensions, as well as of general greater, lesser, and same. In addition, as seen in Figure 3.10, the number of learning trials required to learn refined representations of structured representations scales linearly. While DORA learned representations of all meaningful relations from five dimensions in roughly 1000 learning trials (see simulation 1a), DORA learned meaningful relational representations of 50 dimensions in 10,000 trials. In addition, with more learning trials, the representations that DORA learned became progressively more refined. Just as in simulation 1a, after 3000 learning trials the relational selectivity in the model was just below 0.7, though with additional learning trials in simulation 1b, relational selectivity continued to increase. Finally, Figure 3.11 shows that just as in simulation 1a, DORA learns meaningful structured representations of relational representations.

The results of Simulation 1b show that DORA’s learning algorithm scales well with the complexity of the learning environment. Finally, it is worth noting that although DORA’s learning in this simulation was unsurprisingly slightly slow in simulations 1a and b (as DORA received no feedback or guidance), Sandhofer and Doumas (2008) showed that DORA’s learning
accelerates (to the same rate as human learners) when the model receives the kind of general guidance children routinely receive from adults during the normal course of cognitive development (e.g., being guided to make specific comparisons in specific sequences; Sandhofer & Smith, 1999; Sandhofer, Smith, & Luo, 2000).

### 3.5 Simulation 2: Cross-domain Transfer in Simple Video Games

Our second simulation was designed as a test of cross-domain generalization. We wanted to evaluate whether the model could learn representations from a domain, use those representations to perform intelligently in that domain, and then transfer that knowledge to a new domain in a single shot (i.e., without any additional training). We used transfer between different video games as a case study. In this simulation, after DORA learned to play Breakout, we tested its capacity to generalize, without additional training, to Pong\(^5\) (which is a structurally analogous to Breakout but featurally quite different—among other differences, the player moves the paddle up and down in Pong but left and right in Breakout), and then tested its capacity to return to playing Breakout.

For the purposes of comparison, we also trained four statistical learning systems, including (1) a Deep Q-learning Network (DQN; Mnih et al., 2015) with the standard convolutional neural network front end; (2) a DQN with the same visual front end as DORA; (3) a supervised DNN with the same visual front end as DORA; and (4) a graph network (e.g., Battaglia et al., 2018)

---

5. In this simulation we discuss DORA’s performance transferring from Breakout to Pong for clarity. As shown in supplemental simulation 1, the results were the same when DORA learned to play Pong and attempted to transfer to Breakout.
3.5. Simulation 2: Cross-domain Transfer in Simple Video Games

with the same front end as DORA. These controls allowed us to compare DORA to systems that do not have structured relational representations, and to control for the visual front-end and its assumptions: Networks 2-4 also had objects individuated and contained rate-coded dimensional information as inputs.

In addition, we ran a small transfer study with humans. Unlike the current networks, humans come into the game situation with a wide range of knowledge beyond simple video games generally let alone only Breakout, but an account of human generalization should be able to match the qualitative property that humans do transfer between things like games. Human players either played 50 minutes of Breakout followed by 10 minutes of Pong, or the reverse. The results indicate cross-game transfer between Breakout and Pong and Pong and Breakout. Details appear in Supplemental Results.

3.5.1 Learning to Play a Game in DORA

In this simulation, as in simulation 1, DORA started with no knowledge. To begin, DORA learned representations from Breakout game screens. Learning in this simulation proceeded as described in simulation 1, with the difference that we used game screens from Breakout rather than images of collections of 2D shapes. We allowed the system to play 250 games of Breakout making completely random responses, which produced game screens. These game screens ran through the visual pre-processor described above generating representations of scenes composed of simple objects (e.g., the paddle, the ball, the rows of bricks), which were stored in LTM. DORA attempted to learn from these stimuli performing 2500 learning trials. It did not have any “task” to perform, and it received no feedback during this part of the simulation. DORA successfully learned structured representations of above, below, same-vertical, right-of, left-of, same-horizontal, wider, thinner, same-width, taller, shorter, same-height, larger, smaller, same-size, same-color, and different-color (screen images were colored).

The next step was for DORA to use the representations that it learned from the game environment to engage intelligently with it. Several accounts of how relational representations, once available, may be used to characterize particular domains have been proposed (e.g., Lake, Salakhutdinov, & Tenenbaum, 2015; Nye, Solar-Lezama, Tenenbaum, & Lake, 2020). However, these probabilistic program induction approaches do not directly address the problem of building a relational model of the environment from a reward signal (these models are supervised). As learning to play a video game entails learning to associate actions with states of the game based on a numerical reward signal (points), reinforcement learning methods (Sutton & Barto, 2018) are a natural starting point to solve this problem. Reinforcement learning has been widely applied to account for several aspects of human learning and exploratory behaviour (e.g., Gershman, 2018; Otto, Markman, Gureckis, & Love, 2010; Rich & Gureckis, 2018). In tabular RL, which is the version of RL that we use in this simulation,
the state-action space is represented as a table where the rows are defined by the individual states and the columns are defined by the actions. A known problem with tabular RL is that as the size of the table increases, learning becomes intractable. As relational representations can be combined, the size of the table grows exponentially on the number of relations considered when describing the state. Therefore, in our simulations we make the simplifying assumption that the agent knows what the relevant relations to build a model of the domain are (from the relations that the model had learned from game screens previously). By definition a full account of how to build a relational model of the domain from the reward signal would need to solve the problem of selecting the relevant relations from a potentially very large set of relations. We return to this point in the general discussion.

As mentioned above, in our simulations states were represented as the relevant relations to learn to play a game. In Breakout the relations considered were right-of, left-of, and same-horizontal applied over the paddle and the ball and the same relations applied over the ball at times $t-1$ and $t$. For example, one state in Breakout could be $[\text{right-of}(\text{ball1}, \text{paddle1})]$. On the other hand, actions were represented as a relation between the object that the action was performed over at time $t$, and the same object at time $t+1$. For example, in Breakout the action move-right was represented as $\text{right-of}(\text{paddle2}, \text{paddle1})$, where paddle1 is the paddle before acting, and paddle2 is the paddle after. To associate actions with states of the game we augmented DORA with the capacity for reinforcement learning (e.g., Sutton & Barto, 2018). Specifically, we used tabular Q-learning (Watkins, 1989). Reinforcement learning algorithms seek to maximize the expected discounted cumulative reward, or return, by interacting with the environment. In each iteration of this process the environment produces a state $S_t$ and a reward $R_t$ and the agent takes an action $A_t$ in response. The goal of reinforcement learning is to find the optimal policy $\pi^*$ that maximizes the return. To do this Q-learning utilizes action-values as the basis for this search. The action-value of a state-action pair $Q(s,a)$ is the return when the agent is in state $s$ at time $t$, $S_t = s$ and takes action $a$, $A_t = a$. Q-learning follows an epsilon-greedy policy, where most of the time the action is selected greedily regarding the current action values and with a small probability the action is selected randomly, while updating the action-values according to the equation:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$ (3.12)

where $\gamma$ is a discount factor (in all simulation we set this value to .99).
3.5. Simulation 2: Cross-domain Transfer in Simple Video Games

Applied iteratively, this algorithm approximates the true action-values and, therefore, the output policy (greedy regarding these values) will approximate the optimal policy. We trained DORA for 1000 games using tabular Q-learning. The model learning to associate relational states with relational representations of the available actions. Importantly for our purposes, because the states are relational the resulting policy corresponds to a set of relational rules that can be used as a basis for analogical inference (see below).

3.5.2 Generalizing to a New Game in DORA

As described above, analogical inference occurs when a system uses analogical correspondences between two situations to flesh out one situation based on knowledge of the other. This method is precisely how DORA infers how to play a game like Pong based on its experience with a game like Breakout. While learning to play Breakout, DORA had learned that relations between the ball and paddle predicted actions. Specifically, DORA learned that the state right-of(ball, paddle1) supported moving right (i.e., right-of(paddle2, paddle1)), that the state left-of(paddle1, ball) supported moving left, and that the state same-x(ball, paddle1) supported making no move. With the representation of the available Pong actions in the driver, these Breakout representations are retrieved into the recipient. DORA then performs analogical mapping. Because of the shared relational similarity, corresponding moves between Pong and Breakout map—e.g., above(paddle2, paddle1) in the driver will map to right-of(paddle2, paddle1) in the recipient. DORA then performs analogical generalization on the basis of these mappings.

As described above and illustrated in Figure 5, after mapping the moves in Breakout and Pong, DORA infers the relational configurations that might reward specific moves in Pong based on the relational configurations that reward specific moves in Breakout. For example, given that right-of(ball, paddle) tends to reward a right response (right-of(paddle2, paddle1)) in Breakout, and the mapping between the right response and the up response (above(paddle2, paddle1)) in Pong, DORA infers that above(ball, paddle) tends to reward a up response in Pong. The same process allowed DORA to generalise other learned rules (e.g., below(paddle2, paddle1) then move down).

Importantly, like DORA’s representation learning algorithm, mapping and analogical inference are completely unsupervised processes: The model discovers the correspondences between the games on its own, and based on those correspondences, makes inferences about what kinds of moves are likely to succeed in the new situation.
3.5.3 Results

The DQN, the DQN with the same visual front end as DORA, and the graph network were all trained for 31,003, 20,739, and 10,000 games respectively. The DNN was trained via back-propagation for 4002 games. Models that use structured representations often require far fewer training examples than networks trained with traditional feature-based statistical learning algorithms (see e.g., Bowers, 2017; Hummel, 2011). It is therefore unsurprising that DORA learned to play Breakout much faster than the other networks. Figure 3.12A shows the mean score over the last 100 games of Breakout for all five networks. As expected, all the networks performed well.

![Figure 3.12: Results of game play simulations with DORA, the DQNs and the DNNs. Error bars represent 2 standard errors. (A) Performance humans and networks on Breakout as an average of 100 test games. (B) Results of networks playing Pong after training on Breakout as score on the first game played and mean score over the first 100 games played. (C) Results of networks when returning to play Breakout after playing or learning to play Pong as an average of the first 100 games played.](image)

We then had the human players and networks attempt to play a new game, Pong, for 100 games (see Appendix D for a description of the study with humans). Figure 3.12B show the human players’ and the models’ zero-shot (i.e., immediate) transfer from Breakout to Pong. Both DORA and the human players performed above chance on their very first game of Pong (left columns) and over their first 100 games. That is, both DORA and the human learners demonstrated zero-shot transfer between the games: Having learned to play Breakout, these systems already knew how to play Pong. In contrast, the statistical learning algorithms did not transfer from Breakout to Pong at all: Having learned to play Breakout, the statistical algorithms knew nothing at all about Pong. (The bars for these other networks are not missing from Figure 3.12C, they are simply at zero.) This result is largely unsurprising: One does not expect a lookup table for addition to do subtraction, and one does not expect a lookup table for Breakout to play Pong.
3.5 Simulation 2: Cross-domain Transfer in Simple Video Games

The reason for DORA’s zero-shot transfer from Breakout to Pong is straightforward. As described above, during its first game of Pong, DORA represented the game state using the relations it had learned playing Breakout. Armed with these relations, the model used analogical mapping to discover the correspondences between the two games, and based on those correspondences, made inferences about what kinds of moves are likely to succeed in the new situation. The model’s prior experience with Breakout thus allows it to play its first game of Pong like a good rookie rather than a rank novice.

As a final test, we trained the DNNs to play Pong until they could play with competence, and then retested them, the humans, and DORA for their ability to play Breakout. Of interest in this simulation was whether the various systems, upon learning to play Pong, would still know how to play Breakout. Figure 3.12C shows the performance of the networks on the first 100 games of Breakout after learning Pong. DORA returned to Breakout with little difficulty (again DORA engaged in no learning during these test games). By contrast, the deep DNNs showed extremely poor performance, indicating that learning to play Pong had completely overwritten their ability to play Breakout (i.e., the networks suffered interference from Pong to Breakout; see French, 1999).

It is important to stress that the supervised DNN, one of the DQNs, and the graph net used the same visual pre-processor as DORA, so the differences in their generalization performance cannot be attributed to differences in their inputs. Rather, the DNNs’ generalization failure reflects the purely statistical nature of their representations. For a DNN screens from Breakout and screens from Pong are simply from different distributions, and therefore, it has no reason to generalize between them. By contrast, relation-based learning—like a variablized algorithm—naturally generalizes to novel values (arguments) bound to the variables (relational roles) composing in the algorithm (model of the task).

3.6 Simulation 3: Cross-domain transfer from Shape Images to Video Games

Most people don’t start to play video games until they have had several years’ experience acquiring basic concepts, like “above”. By contrast, most DNNs begin each new task with virtually no knowledge (inductive biases aside). This difference between how people and models typically learn is important as it speaks directly to the importance of cross-domain

---

6. Catastrophic forgetting can be avoided by interleaved training (i.e., training on to-be-learned tasks simultaneously, with “batch” updating of connection weights; e.g., Kirkpatrick et al., 2017). Sequential training of the type people routinely encounter continues to produce catastrophic forgetting in DNNs.
3.6 Simulation 3: Cross-domain transfer from Shape Images to Video Games

transfer: Whereas purely associative systems such as DNNs often suffer from retraining on a new task, people rely on it. A person does not learn concepts like “above” when playing Breakout or Pong, but rather uses those concepts learned in other domains to play the video game.

This simulation explored more realistic learning in DORA. Instead of learning the relations relevant to playing the game from the game itself, DORA first learned representations from a different domain, the first 300 images from the CLEVR dataset (pictures consisting of multiple objects on a screen; Johnson et al., 2017). These images ran through the pre-processor and were encoded into DORA’s LTM. We then ran DORA for 2500 unsupervised learning trials (as in simulation 1). DORA successfully learned structured representations of above, below, same-vertical, right-of, left-of, same-horizontal, wider, thinner, same-width, taller, shorter, same-height, larger, smaller, same-size, same-color, and different-color.

Following representation learning from the CLEVR images, DORA learned to play Breakout via Q-learning for 800 games. Again, the key difference from previous simulation was that DORA used the representations that it had learned from the CLEVR images to encode game screens. No additional representation learning occurred from experience with Breakout. Only associations between the previously learned representations and successful moves were updated via Q-learning. As in Simulation 2, after training with Breakout, we tested the model’s ability to generalize to playing Pong, and then return to playing Breakout. Using representations learned from CLEVR, DORA learned to play Breakout and transferred learning form Breakout to Pong and back to Breakout in a manner very similar to the results of Simulation 2 (Figure 3.13A-C, blue and light blue bars). However, DORA learned to play Breakout in fewer games when it started with the representations learned from the CLEVR images than it did starting with a blank slate in Simulation 1 (800 vs. 1250 games, respectively; as it did not need to learn representations, only a policy for associating representational states with actions). This simulation demonstrates that DORA—like a human learner—exploits cross-domain transfer rather than suffering from it. DORA’s capacity to do so is a direct reflection of its ability to represent the domain-relevant relations explicitly, bind them to their arguments, and map them onto corresponding elements between the familiar and novel games.

3.7 Simulation 4: The Centrality of Binding in DORA

DORA relies on neural oscillations to dynamically bind distributed representations of objects and relational roles into relational structures. According to our account, these oscillations play a central role in learning and generalization because without them, DORA’s representations would be non-relational feature lists—akin to the representations used by DNNs and other associative learning algorithms—and its generalization ability would be correspondingly limited. To explore the role of neural oscillations—that is, explicitly relational representations—in
3.7. Simulation 4: The Centrality of Binding in DORA

DORA’s performance, we reran Simulation 2 (allowing the model to learn to play Breakout, and then attempting to generalize to Pong), but with two different ablated versions of the model. In both ablated versions we disrupted the lateral inhibition between token units (specifically, we reduced the weight of the inhibitory lateral connections between tokens from -1 to -0.1), disrupting the model’s ability to maintain systematic oscillatory behavior. In the first ablated model (A1), we ablated the inhibitory connections from the onset of the simulation. As a result, neural oscillations were disrupted both during predicate learning and thereafter. In the second ablated model (A2), we ablated the inhibitory connections after the model had learned to play Breakout: Although the model was intact when it learned to play Breakout, the neural oscillations, and thus role-argument bindings, were disrupted during generalization to Pong.

The current simulations were otherwise identical to the first two parts of Simulations 3 and 4. As expected, model A1 failed to learn any useful predicate representations. Disrupting the model’s neural oscillations completely eliminated its capacity to learn predicates, and thus greatly reduced its capacity to learn Breakout. The model resorted to learning based on the absolute features of the stimuli, and thus learned much like a less sophisticated DQN. Based on these representations, the model struggled with Breakout even after 20,000 training games and failed to generalize to Pong (Figure 3.13, dark green bars). Model A2, which was intact during predicate learning and Breakout training, learned predicate representations and achieving good performance on Breakout within 1000 games (Fig. 3.13A, light green bar). However, when the oscillations were disabled after training, the model failed to generalize to Pong (Figure 3.13B, light green bar). This result demonstrates the centrality of systematic oscillations in the model’s capacity to learn relational representations (model A1) and to generalize that learning (models A1 and A2).
3.8 Simulation 5: Transfer from Games to More Complex Tasks

This simulation was designed to further challenge the capacity of the representations that DORA learns. In this simulation we investigated whether the representations DORA learned playing video games and from CLEVR (i.e., simulations 2 and 3) would allow the model to generalize to the very different domain of analogical reasoning. To this end, we used the same model and representations from simulations 2 and 3 and set it a number of tasks representing characteristics of human-level analogical reasoning (Bassok & Olseth, 1995; Gick & Holyoak, 1983; Holyoak, 2012; Holyoak et al., 1995). Specifically, after learning representations from Breakout and CLEVR, we tested whether it could immediately (i.e., with no additional experience) use those representations to: (i) solve analogical cross mappings; (ii) analogically map similar, but non-identical predicates; (iii) analogically map objects with no featural overlap—including completely novel objects—that play similar roles; and (iv) map the arguments of an \( n \)-place relation onto those of an \( m \)-place relation even when \( n \) and \( m \) are unequal (i.e., called violation the \( n \)-ary restriction; Hummel & Holyoak, 1997). As such, the simulation had two purposes: (a) to evaluate the capacity of the representations the model learns to support human level analogical reasoning; (b) to provide a further test of the model's capacity for cross-domain generalization: Just like humans do, the model had to learn representations in one domain, and use these representations to reason in a novel laboratory task.

During a cross-mapping, an object (object1) is mapped to a featurally less similar object (object2) rather than a feurally more similar object (object3) because it (object1) plays the same role as the less similar object. For example, if cat1 chases mouse1 and mouse2 chases cat2, then the structural cross-mapping places cat1 into correspondence with mouse2 because both are bound to the chaser role. The ability to find such mappings is a key property of human relational (i.e., as opposed to feature-based) reasoning (e.g., Bassok & Olseth, 1995; Gick & Holyoak, 1983; Holyoak, 2012; Richland, Morrison, & Holyoak, 2006). Cross-mappings serve as a stringent test of a computational system's structure sensitivity as they require the system to discover mappings based on relational similarity in the face of competing featural or statistical similarity.

We first tested the representations DORA learned in Simulations 1 and 2 for their ability to support cross-mapping. DORA randomly selected two of the predicates (T1 units) it had learned during Simulations 1 and 2, such that both predicates coded for the same relation (e.g., both coded for \( \text{above} \), or both coded for \( \text{same-width} \)). DORA bound the relations to new objects, to form two new propositions, P1 P2 (e.g., \( \text{above} \)(object1, object2)) and P2 (e.g., \( \text{above} \)(object3, object4). We manipulated the objects such that the agent of P1 (object1) was featurally identical to the patient of P2 (object 4) and the patient of P1 (object 2) was featurally identical to the agent of P2 (object 3). Based on the objects' featural similarity, DORA would therefore map object1 to object4 and object2 to object3, but based on their
roles in the relational structures, it would map object1 to object3 and object2 to object4. We
repeated this procedure 100 times (each time with different randomly chosen T1 units) using
representations from Simulations 1 and 2. In every case, DORA successfully mapped object1
to object3 and object2 to object4 (the structurally consistent mappings) rather than object1 to
object4 and object2 to object3 (the feature-based mappings). This result demonstrates that
the relations DORA learned in Simulations 1 and 2 immediately transfer to analogy tasks and
support relational cross-mapping.

We then tested whether DORA’s relational representations can support mapping similar but
non-identical relations (such as mapping above to greater-than) and mapping objects with no
featural overlap at all based only on their bindings to similar roles. DORA randomly selected
two of the relations, R1 and R2 (e.g., above(x,y) or wider(x,y)), that it had learned during
Simulations 1 and 2 such that each role of R1 shared roughly half of its features with the
corresponding role of R2 (e.g., the role more-y has half of its features in common with the role
wider). The objects serving as arguments of the relations were designed to have no featural
overlap at all. (To ensure that the objects had no feature overlap, we used object features
unique to these simulations—units DORA had not “experienced” previously.) We repeated
this process 100 times and each time, DORA mapped the agent role of R1 to the agent role of
R2 and the patient role of R1 to the patient role of R2. Even though the objects had no features
in common, and even though the relations to which they were bound were not identical, DORA
found the structurally correct object and role mappings. The model demonstrated the ability
to map completely novel objects—that is, objects composed of features it had never before
experienced—in a single trial with no additional experience. This result demonstrates that
the representations that DORA learned in Simulations 1 and 2 not only transfer to a new
task, but also supported extrapolation to completely novel objects (features not previously
experienced).

Next, we tested whether the representations DORA learned can violate the n-ary restriction,
mapping the arguments of an n-place predicate onto those of an m-place predicate when
n \neq m. In each of these simulations, DORA randomly selected a relation, R1, that it had
learned in Simulations 1 and 2, and then we created a single-place predicate (r2) that shared
50% of its features with the agent role of R1 and none of its features with the patient role.
DORA then bound two objects to the roles of R1 to form the proposition R1(object1, object2),
and bound a third object to r2, to form the proposition r2(object3). Object3 shared half its
features with object1 and the other half with object2 (i.e., it was equally similar to both object1
and object2). DORA then attempted to map r2(object3) onto R1(object1, object2). If the model
can violate the n-ary restriction, then it will consistently map object3 to object1 based on the
similarity of r2 to the first (agent) role of R1 (recall that R1 is represented as a linked set of
roles). This process was repeated 100 times using a different randomly chosen R1 each time.
Each time DORA successfully mapped object3 to object1, along with corresponding relational
3.8. Simulation 5: Transfer from Games to More Complex Tasks

roles (i.e., DORA maps the predicate representing one of the roles of R1 and the predicate representing the single-place predicate r2). We then ran 100 simulations in which r2 shared half its features with the second (patient) role of R1 rather than the first (agent) role. In 100 additional simulations, DORA successfully mapped the patient role of R1 to r2 (along with their arguments).

Finally, we tested whether the representations that DORA learns support generalization to completely novel (i.e., never before experienced) stimuli. The ability to make generalization about completely novel items is the hallmark of the capacity for universal generalization (see, e.g., Marcus, 2001). In this simulation, DORA randomly selected a relation, R1, that it had learned in Simulations 1 and 2, and then bound two objects to the roles of R1 to form the proposition \( R1(\text{object1}, \text{object2}) \). This representation was placed in the driver. We then created two objects, object3 and object4 that were composed of object features unique to these simulations—units DORA had not “experienced” previously. These objects were placed in the recipient. DORA then selected a role r1 at random from LTM that matched one of the roles in R1 (e.g., if \( R1 \) was a representation of above, then \( r1 \) might be more-y or less-y). \( r1 \) was bound to object3 to form \( r1(\text{object3}) \) in the recipient. DORA then attempted to map the representation in the driver to the representations in the recipient. If DORA found a mapping, it then attempted to perform analogical inference. We ran the simulation 100 times. In each simulation, DORA mapped the representation of \( r1(\text{object3}) \) to the corresponding role and object from \( R1 \), and then generalized the unmapped role from \( R1 \) to object4. For example, if \( r1 \) was more-y and \( R1 \) was above(\text{object1, object2}) then DORA mapped \( r1 \) to the more-y role of \( R1 \), and object3 to the object1. DORA then generalized the less-y role to object4, to form the proposition above(\text{object3, object4}) in the recipient. That is, armed with the knowledge that some object it had never before experienced played a particular relational role, DORA generalized that a proximal object (that it had also never experienced before) might play the complementary role. The inability to reason about completely novel features (i.e., features not part of the training space) is a well-known limitation of traditional neural networks (e.g., Bowers, 2017). However, this limitation does not apply to DORA. Not only can DORA represent that novel objects can play certain roles (because it dynamically binds roles to fillers), but it can also use its representations to make inferences about other completely novel objects.

In total, simulation 5 demonstrates that the relational representations DORA learned during Simulations 2 and 3 immediately support a performance of an unrelated task (analogical reasoning) even with completely novel objects. After learning representations in one domain (game play and images of shapes), DORA, with no additional experience (zero-shot), used these representations to solve a set of analogical reasoning tasks representing four hallmarks of human analogical thinking, and then used these representations to generalize to completely
novel objects. The results provide further evidence that DORA’s representations support cross-domain transfer and highlight the generality of the DORA framework: a single model learned to play video games and then used the representations it learned during that task to solve analogical mapping and generalization problems with human-level flexibility.

3.9 Simulation 6: Development of Representations of Relative Magnitude

We have previously shown that DORA’s format learning algorithm provides a good account of several developmental phenomena in representational development (e.g., Doumas et al., 2008). The purpose of simulations 6 and 7 was to examine whether the representations that DORA learned, and the trajectory of the representation learning mirror human development when DORA is learning both relational content and relational format. Additionally, The simulations provided another opportunity to evaluate generalization of representations across domains: Learning representations in one domain and deploying those representations to reason about a new domain (as human often do when they engage in laboratory experiments).

Children develop the ability to reason about similarity and relative magnitude on a variety of dimensions (e.g., Smith, 1984). The development of children’s capacity to reason about basic magnitudes is well demonstrated in a classic study by Nelson and Benedict (1974). In their experiment, children aged three to six years-old were given a simple identification task. An experimenter presented the child with two pictures of similar objects that differed on some dimensions. The experimenter then asked the child to identify the object with a greater or lesser value on some dimension. For example, the child might be shown pictures of two fences that differed in their height, their size, and their color, and then asked which of the two fences was taller or shorter. The developmental trajectory was clear: Children between 3-years-10-months and 4-years-4 months (mean age ∼48 months) made errors on 34% of trials, children between 4-years-7-months and 5-years-5-months (mean age ∼60 months) made errors on 18% of trials, and children aged 5-years-6-months and 6-years-6-months (mean age ∼73 months) made errors on only 5% of trials. In short, as children got older, they developed a mastery of simple magnitude comparisons on a range of dimensions.

If DORA is a good model of human representational development, then it should be the case that DORA’s representations follow a similar developmental trajectory. To test this claim, we used the representations that DORA had learned during Simulation 2. If DORA develops like a human child, then early in the learning process, DORA’s performance on the Nelson and Benedict task should mirror 3-4 year-old children, later in the learning process DORA’s performance should mirror 4-5 year-old children, and later in the learning process DORA’s performance should mirror 5-6 year old children.
To simulate children of different ages we stopped DORA at different points during learning and used the representations that it had learned to that point (i.e., the state of DORA’s LTM) to perform the magnitude reasoning task. To simulate each trial, we created two objects instantiated as T1 units attached to features. These features included 100 random features selected from the pool of 10,000, and features encoding height, width, and size (dimensions used in Nelson & Benedict, 1974) in pixel format (e.g., for an object 109 pixels wide, one feature unit describing “width0” and 109 features that together encoded “109 pixels wide”). A dimension was selected at random as the question dimension for that trial (i.e., the dimension on which the question would be based). DORA sampled at random a representation from its LTM that was strongly connected to that dimension (with a weight of .95 or higher). If the sampled item was a relation or a single-place predicate, DORA applied it to the objects, and placed that representation in the driver. For example, if the key dimension was size, the two objects (obj1 and obj2) were then run through the local energy circuit on the dimension of size, marking one (assume obj1) as relatively larger and the other (assume obj2) as relatively smaller. If DORA had sampled a representation of the relation larger(x,y), then the more-size T1 unit was linked (via its T2 unit) to the obj1 T1 unit and the less-size T1 unit was linked (via its T2 unit) to the obj2 T1 unit. To simulate a dimensional question, DORA randomly sampled a representation of the question dimension from LTM and placed that in the recipient. For example, if the question was, “which is bigger”, DORA sampled a representation of a T1 unit encoding more-size from LTM. DORA then attempted to map the driver and recipient representation. If DORA mapped a driver a representation in the driver to a representation in the recipient, the mapped driver item was taken as DORA’s response on the task. If DORA failed to find a mapping, then an item was chosen from the driver at random and taken as DORA’s response for that trial (implying that DORA was guessing on that trial). The probability of guessing the correct item by chance was .5. To simulate 4 year-olds, we used the representations in DORA’s LTM after 1000 total training trials, to simulate 5 year-olds we used the representations in DORA’s LTM learned after 1000 additional training trials (2000 total trials), and to simulate 6 year-olds we used the representations in DORA’s LTM learned after 1000 additional training trials (3000 total trials). We ran 50 simulations each with 20 trials at each age level (each simulation corresponding to a single child). The results of the simulation and the original results of Nelson and Benedict are presented in Figure 3.14.

The qualitative fit between DORA’s performance and the performance of the children in Nelson and Benedict’s study is close. Just like the children in the original study, DORA is better than chance, but still quite error prone early during learning, but gradually comes to learn representations that support very successful classification of dimensional magnitudes. These simulation results provide evidence that the trajectory of the development of DORA’s representations of relative dimensional magnitude mirrors that of humans, and also that the representations that DORA learns support the same kinds of processing that humans come to excel in.
3.10 Simulation 7: The Relational Shift

One of the key findings from work on the development of analogical reasoning in children, is that children go through a relational shift (e.g., Gentner, Rattermann, Markman, & Kotovsky, 1995). The relational shift describes a qualitative change in children’s reasoning wherein they progress from making analogies based on the literal features of things, to making analogies based on the relations that objects are involved in (e.g., Richland et al., 2006). With development, children learn progressively more powerful representations of similarity and relative magnitude relations that support more proficient relational generalization (Smith, 1984). In addition, children develop the capacity to integrate multiple relations in the service of reasoning (e.g., Halford & Wilson, 1980), and their relational representations grow more robust with learning, and allow them to overcome ever more excessive featural distraction (Halford & Wilson, 1980; Rattermann & Gentner, 1998).

One of the classic examples of the relational shift and the associated phenomena is given in Rattermann and Gentner (1998). In their experiment, Rattermann and Gentner had 3-, 4-, and 5-year-old children participate in a relational matching task. Children were presented with two arrays, one for the child and one for the experimenter. Each array consisted of three items that varied on some relative dimension. For example, the three items in each array might increase in size from left to right or decrease in width from left to right. The dimensional relation in both presented arrays was the same (e.g., if the items in one array increased in size from left to right, the items in the other array also increased in size from left to right). The items in each array were either sparse (simple shapes of the same color) or rich (different objects of different colors). The child watched the experimenter hide a sticker under one of the items in the experimenter’s array. The child was then tasked to look for a sticker under the item from the child’s array that matched the item selected by the experimenter. The correct item was always the relational match—e.g., if the experimenter hid a sticker under the largest item,
the sticker was under the largest item in the child's array. Critically, at least one item from the child's array matched one of the items in the experimenter's array exactly except for its relation to the other items in its array. To illustrate, if the experimenter might have an array with three squares increasing in size from left to right (Fig. 3.15A). The child might have an array of three squares also increasing in size from left to right, but with the smallest item in the child's array identical in all featural properties to the middle item in the experimenter's array (Fig. 3.15B). Thus, each trial created a cross-mapping situation, where the relational choice (same relative size in the triad) was at odds with the featural choice (exact object match). The child was rewarded with the sticker if she chose correctly.

Figure 3.15: A recreated example of the stimuli used in (Rattermann & Gentner, 1998).

Rattermann and Gentner found a clear indication of a relational shift. Children between 3 and 4-years-old were very drawn by featural matches, and had trouble systematically making relational matches (making relational matches 32% of the time in the rich condition and 54% of the time in the sparse condition). Children between 4 and 5-years-old were quite good at making relational matches with sparse objects—making relational matches 62% of the time—but still had trouble with rich objects when featural matches were more salient—making relational matches 38% of the time. Children between 5 and 6-years-old were quite good at making relational matches in both the rich and the sparse conditions, with the rich condition providing more trouble than the sparse condition—making relational matches 68% for rich and 95% of the time for sparse stimuli.

We simulated the results of Rattermann and Gentner (1998) as in the simulation above. To simulate children of different ages we stopped DORA at different points during learning and used the representations that it had learned to that point. To simulate each trial, we created two arrays of three objects, each object instantiated as a T1 unit connected to features. For the
3.10. Simulation 7: The Relational Shift

sparse trials, each object was connected to feature units such that some features encoding absolute size, height, width, x-position, y-position, color, 10 features describing shape, and four features chosen at random from a pool of 1000. The identical objects from both arrays matched on all features. For the rich trials, each object was attached to a number of features: some features encoding absolute size, height, width, x-position, y-position, color, 10 features describing shape, four features describing object kind (e.g., “shoe”, “train”, “bucket”), and 40 features chosen at random from a pool of 1000. The identical objects from both arrays matched on all features.

We ordered the objects in both arrays according to some relation (e.g., increasing size, decreasing width). DORA sampled four representations from its LTM that were strongly connected to that dimension (with a weight of .95 or higher) and applied two of the sampled representations to each of the two arrays. If the sampled representation was a relation or a single-place predicate, it applied to the objects. For example, if the key dimension was size, and DORA sampled a representation of the relation $\text{larger}(x,y)$, it applied that representation to the objects, binding the larger object to the more-size role and the smaller object to the smaller role (as described in Simulation 6). If the sampled representation was a single-place predicate like $\text{more-size}(x)$, then it was bound to the larger object. As each array consisted of two instances of the key relation (e.g., $\text{larger}(\text{object1}, \text{object2})$, and $\text{larger}(\text{object2}, \text{object3})$), DORA applied one of the two sampled items to one of the relations in the array, chosen at random, and the other sampled item to the other relation in the array.

The representation of the child’s array entered the driver, and the experimenter’s array the recipient. An item from the recipient was chosen at random as the “sticker” item (i.e., the item under which the sticker was hidden). The capacity to ignore features is a function of the salience of those features, and so richer objects with more features are harder to ignore (see, e.g., Goldstone & Son, 2012). To simulate the effect of the rich vs. the sparse stimuli, on each trial, DORA made a simple similarity comparison before relational processing started. It randomly selected one of the items in the driver and computed the similarity between that item and the “sticker” item in the recipient using the equation:

$$sim_{ij} = \frac{1}{1 + \sum_i(1 - s_i)}$$

(3.13)

where, $sim_{ij}$ is the 0 to 1 normalized similarity of PO unit $i$ and PO unit $j$, and $s_i$ is the activation of feature unit $i$. If the computed similarity was above .8, then DORA learned a mapping connection between the two items. Finally, DORA attempted to map the items in the driver to the items in the recipient. If any driver representation was mapped to the “sticker”
item in the recipient, the mapped item was taken as DORA’s response on the task. If DORA failed to find a mapping, then it selected an item from the recipient at random as a response for that trial (implying that DORA was guessing on that trial). The probability of guessing the correct item by chance was .33.

To simulate 3 year-olds we used the representations in DORA’s LTM after 850 training trials, to simulate 4.5 year-olds we used the representations in DORA’s LTM after 1500 training trials, and to simulate 5.5 year-olds we used the representations in DORA’s LTM after 2500 training trials. We ran 50 simulations each consisting of 20 trials at each age level. The results of the simulation as well as those from the original Rattermann and Gentner experiment with both sparse and rich trials are presented in Figure 3.16A and 3.16B respectively.

As Fig. 3.16 shows, there is a close qualitative fit between DORA’s performance and the performance of the children in Rattermann and Gentner (1998). Initially, DORA, just as the 3-year-old children in the original study, had some trouble correctly mapping the items in the driver and the recipient, and struggled to solve the cross-mapping. As DORA learned more refined representations (after more training), like the 4-year-old children in the original study, DORA began to solve the sparse problems more successfully, while still struggling with the rich problems. Finally, like the 5-year-old children in the original study, after even more learning, DORA was quite successful at both rich and sparse trials, reaching ceiling level performance on the sparse problems. These simulation results indicate that, like humans, the trajectory of the development of DORA’s representations of relative dimensional magnitude undergoes a relational shift with learning. Additionally, the representations that DORA learns during its development support the same kind of performance on relational matching tasks that is evidenced by human children during their development.
3.11 General Discussion

3.11.1 Summary and Overview

We have presented a theory of human cross-domain generalization instantiated in the DORA computational framework. Our proposal is that people represent knowledge domains as models consisting of structured representations of the relations among the elements of those domains. These representations specify the invariant content of the relations and their arguments in a way that binds relational roles to their fillers while maintaining their independence. DORA learns both the content and structure of these relations from non-relational inputs, such as visual displays, without supervision. Instead of error correction, DORA uses comparison to bootstrap the learning of both the content and structure of relations. By integrating these representations with a capacity for reinforcement learning, DORA learns which relations to use in what contexts in the service of problem solving (e.g., game play). The resulting representations can be applied to any domain in which they are relevant, including completely novel ones, by a process of analogical inference. That is, they generalize across domains as a natural consequence of their ability to represent relations in a manner that is invariant with both the arguments of the relations and the specific circumstances in which those relations arise.

A series of simulations demonstrated that this approach to learning and knowledge representation greatly facilitates cross-domain generalization. Simulation 1 showed that the model is capable of learning structured representations of relations from unstructured, non-relational visual inputs. Simulation 2 showed that, as a result of learning to play one video game, DORA learns representations that support immediate (zero-shot) transfer to a different game (Pong). By contrast, four different associative networks (two DQNs, a DNN, and a GNN) both (a) failed to transfer knowledge from one game to another and (b) lost their ability to play the first game after training on the second. Simulation 3 demonstrated that the representations DORA learns in one domain (images of 3D shapes) support learning relations for video game play that immediately generalize to a new game. Simulation 4 demonstrated the essential role of structured representations in the model’s learning and generalization. Simulation 5 showed that the representations DORA learns from domains like video games and pictures also support successful zero-shot transfer to unrelated reasoning tasks (cross-mapping, mapping non-identical predicates, mapping novel objects, and violating the \( n \)-ary restriction), and, importantly, support generalization to completely novel (i.e., never previously experienced) stimuli. Finally, simulations 6 and 7 showed that DORA follows the same developmental trajectory as children as it learns representations. That is, DORA accounts for results from the literature on children’s reasoning as it learns to play video games.
3.11.2 LISAeese, Relational Databases, and Generalization

We have argued that the reason why people can learn relations and apply them to new domains is because we learn representations of those relations that specify relational invariants in a form that permits the binding of relational roles to their arguments without changing the representation of either (see also Doumas et al., 2008; Halford, Bain, Maybery, & Andrews, 1998; Halford, Wilson, & Phillips, 1998; Hummel & Biederman, 1992; Hummel & Holyoak, 1997, 2003; Phillips, 2018, 2021; Phillips, Halford, & Wilson, 1995). It turns out that our proposal has an analog in computer science in the form of relational databases.

In mathematical logic, a relation is defined as subset of the Cartesian product of two or more potentially infinite sets. For example, the relation $\text{larger-than}(\cdot)$, defined over the integers, is a matrix (a Cartesian product), with integers in the rows and columns, and 1s and 0s in the cells, such that a 1 appears in every cell whose row is larger than its column. More generally, a binary relation between sets $A$ and $B$ is a subset of the Cartesian product $A \times B$ for each pair $(a, b)$ over which the relation holds. A relation is thus represented by a characteristic function, $\chi_R(a, b)$, which maps to 1 if the relation is true for $(a, b)$ and 0 otherwise.

The characteristic function captures the same information as the relational invariants we described in the Introduction. In other words, the characteristic function specifies the content of the relation, so learning the invariant that defines a relation is a matter of executing the characteristic function. The circuit DORA uses to discover relational invariants is nothing more than an implementation of the characteristic function of $\text{larger-than}(\cdot)$ over a rate-coded neural representation of magnitude.

However, as we have argued extensively here, simply expressing an invariant (the output of the characteristic function) is not sufficient to form a structured representation of a relation. It is also necessary to somehow represent the bindings of arguments to roles of the relation in a way that preserves the identity of both the relational roles and their arguments. The representational format we have employed for this purpose, LISAeese, is isomorphic to a relational schema (or relational database) developed in computer science.

A relational schema (see, e.g., Phillips, 2018) is a table describing a single relation. It consists of a set of rows, representing instances of the relation, and columns, corresponding to the roles of the relation. For example, the relational schema for the relation $\text{larger-than}(x, y)$ includes two columns, one specifying the larger item and the other specifying the smaller, with each row of the table an instance of the $\text{larger-than}$ relation. This representational format has the property that relations (tables) are represented explicitly, as are their roles (columns), and arguments (cells), while simultaneously expressing the bindings of arguments to roles without altering the meaning of either.
Halford, Phillips, and colleagues (Halford, Bain, et al., 1998; Halford, Wilson, & Phillips, 1998; Phillips, 2018, 2021; Phillips et al., 1995) have argued that relational schemas are a good model of human mental representations. Specifically, (a) they identify a relation symbolically, (b) the roles (or argument slots) of the relation are represented independently of the fillers of those roles, (c) binding of roles to fillers is explicit, (d) the format supports representing higher-order relations (i.e., relations between relations), and (e) the resulting representations have the property of systematicity, meaning that they permit simultaneous expression of the meaning of (i) the relation, (ii) its roles, and (iii) their composition into a larger expression (see Halford, Bain, et al., 1998; Halford, Wilson, & Phillips, 1998; Phillips et al., 1995). Phillips (2018, 2021) has observed that LISAese is a representational format akin to a relational schema. And indeed, the representations that DORA learns (i.e., LISAese) satisfy all the properties of a relational database. In DORA relational representations are specified as sets of linked single-place predicates (columns) composing a header, and values bound to those predicates instantiating (rows) of the specific relation.

In a set of experiments, Halford and Wilson (1980) demonstrated that people's inferences in a complex learning task are better captured by representations based on relational schemas than simple associations. In their Experiment 2, participants learned to associate a shape and trigram pair with another trigram. The stimuli were composed from three shapes (say, square, circle, triangle) and three trigrams (say, BEJ, FAH, PUV). The shape acted like an operator, and the mapping from shape-trigram pair to the output (i.e., the other trigram) was given by a simple rotation rule. For example, one shape when linked with a trigram mapped to the identical trigram. The second shape when linked to a trigram, mapped to the next trigram from the list (trigram-1 mapped to trigram-2, trigram-2 mapped to trigram-3, and trigram-3 mapped to trigram-1). The third shape when linked to a trigram mapped to the trigram two jumps away (trigram-1 mapped to trigram-3, trigram-2 mapped to trigram-1, and trigram-3 mapped to trigram-2). As expected, participants learned to perform the task after several exposures. The experimenters reasoned that if participants had learned the mappings by association, then when given new trigrams and new shapes that followed the same rotation rule, the participants would need roughly the same number of exposures to learn them. However, if participants had learned the relation (i.e., rotation) between shape-trigram pair and trigram output, then they should be able to apply the relation two new shape and trigram sets within a few exposures. Participants did apply the rule to new shapes and trigram within a few trials, indicating that they had learned the relational table.

More recently, Phillips (2018, 2021) Phillips (2018, 2021), using tools from Category Theory, showed that performance on tasks like the relational schema inference task (above) requires structured relational representations (like a relational database) and cannot be accounted for by association alone. Our simulation results resonate with this claim, and suggest the
capacity extends to cross-domain transfer. As demonstrated in simulations 2-4, after learning structured relational representations, DORA can learn to play a video game and then immediately generalize to a relational similar (but featurally different) game. However, if these representations are removed, the model fails utterly at any kind of generalization.

DORA provides an account of human generalisation because it can learn explicit representations of relational concepts—both their content and their format—and then leverage those representations to solve problems. In addition, the model provides an account of how such knowledge structures can be implemented in a distributed neural system, and how they can be learned from non-relational inputs.

### 3.11.3 Other Classes of Relations

In the work reported here, we have focused on transitive relations, those that can be defined by differences on a single dimension, such as `larger-than()` and `left-of()`. It is natural to ask whether these same principles apply to other, nontransitive relations such as `chases()`, `mother-of()`, and `loves()`. The short answer is yes. Starting with whatever regularities it is given or can calculate from the environment, DORA’s learning algorithm will isolate those invariants, learn structured representations (i.e., functional predicates) that take arguments, and, where appropriate, compose them into relational structures. In short, given a set of invariants (or a means to calculate them), DORA’s learning mechanisms will produce explicit predicate and relational representations of those invariants. DORA will learn structured representations of concepts based on their invariant properties, whether the invariants the system detects are instances of stimulus magnitude or romantic love (see also, Doumas et al., 2008). The hard part is finding the invariants. And for this problem, the human visual system may give us a leg up.

Nontransitive relations like `chase`, `support`, or `love` fall on a gradient in terms of how spatial they are. A relation like chase is comparatively easy to reduce to spatial properties: There are two objects, \( a \) and \( b \), such that the vector characterizing the movement of \( a \) is in the direction of the vector characterizing the location of \( b \), and this configuration is maintained through time. The representations necessary to induce this relation are delivered by the visual system. Michotte (1963) showed that participants would overwhelmingly interpret chasing occurring in a situation where two dots moved across a screen (or, in Michotte’s original version, two lights moved on a grid of lights) such that one stayed in front of the other as they moved. Similarly, relations such as `support` (one object above and in contact with another object) or `lift` (one object supports and raises another object), are definable in spatial terms. Even a relation like loves might reduce, at least in part, to spatial relations, though. In a study by Richardson, Spivey, Barsalou, and McRae (2003), participants asked to use configurations of objects to represent a relation produced overwhelmingly similar spatial arrangements for relations like `love`, `admire`, and `hate`. 
3.11. General Discussion

We do not claim that all relations are spatial in origin, or that there are no invariants (e.g., characterizing social relations such as love, hate, friend, adversary, etc.) that have non-spatial origins. On the contrary, we are completely agnostic about the number and nature of the psychological dimensions over which relational invariants might be computed. What we do claim is that any psychologically privileged dimension, whether it be spatial, auditory, social, or what have you, is subject to the kind of invariant isolation and structure-inducing processes embodied in DORA: If there is an invariant, wherever it originates, intersection-discovery can find it and DORA can predicate it and use it for inference and cross-domain generalization.

Doumas (2005) proposed a compression mechanism, complementary to DORA’s refinement algorithm, which is a form of chunking. During compression, multiple roles attached to the same object fire together, and a unit learns to respond to that new conjunction as a unitary predicate. Compression allows DORA to combine multiple representations of the same object. For example, if DORA encounters situations in which one element is both larger and occludes some second object, DORA can compress the roles larger and occluder and the roles smaller and occluded to form a representation like cover(a, b). Such a procedure might serve as a basis for combining primitive transitive relations into more complex relations.

A second question is whether the role-filler representational system DORA uses is sufficient to represent all the relations people learn. Again, the short answer is, at least in principle, yes (as pointed out originally by Leibniz). Formally, any multi-place predicate is representable as a linked set of single-place predicates (Mints, 2001). Therefore, a role-filler system can, at least in principle, be used to represent higher-arity predicates (or relations; recall the distinction, noted above, between a relation qua a relational schema and a function). Models based on role-filler representations account for a large number of phenomena in analogy making, relational learning, cognitive development, perception, and learning (e.g., Doumas & Hummel, 2010; Doumas et al., 2008; Hummel, 2001; Hummel & Biederman, 1992; Hummel & Holyoak, 1997; Lim et al., 2013; Livins, Spivey, & Doumas, 2015; Martin & Doumas, 2017; Morrison et al., 2011, 2004; Sandhofer & Doumas, 2008; Son, Doumas, & Goldstone, 2010). Moreover, access to and use of role-based semantic information is quite automatic in human cognition, including during memory retrieval (e.g., Gentner et al., 1995; Ross, 1989), and analogical mapping and inference (Bassok & Olseth, 1995; Holyoak & Hummel, 2008; Krawczyk et al., 2004; Kubose et al., 2002; Ross, 1989). Indeed, the meanings of relational roles influence relational thinking even when they are irrelevant or misleading (e.g., Bassok & Olseth, 1995; Ross, 1989). Role information appears to be an integral part of the mental representation
of relations, and role-filler representations provide a direct account for why. Moreover, role-filler systems appear uniquely capable of accounting for peoples’ abilities to violate the \( n \)-ary restriction, i.e., mapping \( n \)-place predicates to \( m \)-place predicates, where \( n \neq m \) (e.g., mapping “\( x \) murdered \( y \)” onto “\( a \) caused \( b \) to die”; Hummel & Holyoak, 2003)\(^7\).

Livins et al. (2016) showed that we can affect the direction of a relation by manipulating which item one looks at first, for both obvious similarity and magnitude relations, and the other kinds of relations. Livins et al. showed participants images depicting a relation that could be interpreted in different forms (e.g., \textit{chase}/\textit{pursued-by}, \textit{lift}/\textit{hang}). Before the image appeared on screen, a dot appeared on the screen drawing the participants attention to a location that one of the objects involved in the relation would appear. For example, the image might show a monkey hanging from a man’s arm, and the participant might be cued to the location where the monkey would appear. The relation that the participant used to describe the image was strongly influenced by the object that they attended to first. That is, if the participant saw the image of the monkey hanging from the man’s arm, and she was cued to the monkey, they would describe the scene using a \textit{hanging} relation. However, if the participant was cued to the man, she would describe the scene using a \textit{lifting} relation. This result follows directly from a system based on role-filler representations wherein complementary relations are represented by a similar set of roles, but the predicate, or role, that fires first defines the subject of the relation.

3.11.4 Limitations and Future Directions

People routinely learn structured representations from experience, an ability we argue is fundamental to our understanding of the world and our ability to use the knowledge we have gained in one context to inform our understanding of another. We offer an account of this process that is based on minimal assumptions, assumptions that, with the exception of the capacity for dynamic role-filler binding, are standard in neural networks such as DNNs. Our account is, of course, limited in several ways. In the following, we outline some of the limitations of our model and suggest ways to address these limitations.

\(^7\) One might wonder how the role-based learning approach works for symmetrical relations like \textit{equals} or \textit{antonym}. In short, these kinds of relations are not a problem for DORA (for example, as demonstrated above the model has no problem learning relations like \textit{same-as}). If the relation is interpreted as referential then the relation is not symmetrical, and the roles are distinct. For example, in \textit{antonym}(\( x, y \)), \( y \) is the referent term, playing the \textit{referent-of-something} role, and \( x \) is the antonym of that term, playing the \textit{opposite-of-something} role. Alternately, if the relation is symmetrical and both arguments play the same role, then in a LISAsese representation of that relation, there will only be a single token for the role (recall T1 tokens are not repeated within a proposition). For example, if \textit{antonym}(\( x, y \)) is symmetrical, and both arguments play the same role—like \textit{opposite-of}—then a single T1 token unit will represent that role in the proposition and both \( x \) and \( y \) will be bound to that role by distinct T2 role-binding units in LTM and by asynchrony of firing in WM. That is, the relation does not need two distinct roles, but rather is representable (and learnable) as a single role involved in two (or more) role-bindings, with those role-bindings linked to form a higher-arity proposition.
First, the constraints on learning in DORA are underdetermined. DORA learns when it can and stores all the results of its learning. We have implemented a crude form of recency bias in our simulations (by biasing retrieval of the most recently learned representations during learning), but future work should focus on development of more principled mechanisms for constraining learning and storage. Such mechanisms might focus on either constraining when learning takes place, or on when the results of learning are stored for future processing. Most likely, though, it will be necessary to account for both.

Constraining when DORA learns amounts to constraining when it performs comparison. We have previously proposed several possible constraints on comparison such as language (e.g., shared labels) and object salience, and have shown how direction to compare (i.e., instruction) serves as a very powerful constraint on learning (see Doumas & Hummel, 2013; Doumas et al., 2008). These constraints may also serve to limit when the results of learning are stored in memory. DORA might be extended or integrated with existing accounts of language or perceptual (feature) processing to implement such constraints (see Martin & Doumas, 2017).

Both these limitations might be addressed by refining DORA’s control structure. The quality of comparisons DORA makes and the representations it learns may serve as important constraints on the control process it uses. Reinforcement learning provides a useful tool for implementing these constraints.

Second, as pointed out in Simulation 2, we don’t have yet a complete solution for the problem of how to select the right representations to build a relational model of a domain from the reward signal when the domain of potential relations is large. In artificial intelligence the problem of learning relational (a.k.a., first-order) policies has been studied under the name of relational reinforcement learning (Driessens & Džeroski, 2004; Driessens & Ramon, 2003; Džeroski et al., 2001), but these early models do not scale well to large problems involving multiple relations. However, recent models based on differentiable versions of inductive logic programming (Evans & Grefenstette, 2018; Jiang & Luo, 2019) seem a promising approach to this problem. These systems have shown that it is possible to use gradient descent methods to prune a prebuilt large set of rules to obtain a program (i.e., a refined sets of rules) that allows the agent to interact effectively with the environment. We are currently working to toward integrating this kind of error-correction learning with DORA.

The discovery of invariance has relevance beyond the few problems presented here. For example, detecting invariants in speech and language is a defining and unsolved problem in language acquisition and adult speech processing, including in automatic speech recognition by machines. Similarly, whether the generalization of grammatical rules can be fully accounted for in systems that rely on statistical learning alone remains contentious. The account of learning invariance from experience offered here, combined with principles like the compression of
role information (Doumas, 2005), may present new computational vistas on these classic problems in the language sciences (see Martin, 2016; Martin & Doumas, 2017). Systems with the properties of DORA may offer an inroad to representational sufficiency across multiple domains, built from the same mechanisms and computational primitives.

3.11.5 Conclusion

A cognitive architecture that is prepared to learn structured representations of relations is prepared to generalize broadly on the basis of those relations. This kind of generalization includes cross-domain transfer as a special case. In fact, it is a mundane consequence of the way people conceptualize the world.

Purely statistical learning systems will most likely continue to outperform people at any single task on which we choose to train them. But people, and cognitive architectures capable learning relations in an open-ended fashion, will continue to outperform any finite set of purely statistical systems as generalists. And general intelligence, we argue, is not the capacity to be optimal at one task, but is instead the capacity to excel, albeit imperfectly, at many.
Chapter 4

Learning Relational Rules from Rewards

4.1 Introduction

Being able to represent the world in terms of relations between objects unlocks forms of generalisation for humans that are beyond the capabilities of any other species (Penn et al., 2008), and current artificial intelligence (AI) systems based on statistical learning alone (e.g., Geirhos et al., 2020; Kansky et al., 2017; Lake, Salakhutdinov, & Tenenbaum, 2019, for a review see Mitchell 2021). Ultimately, relational representations allow people to generalize what they have learned about one task to seemingly unrelated ones, based on the process of analogical inference (Doumas, Puebla, Martin, & Hummel, 2022). Throughout the lifespan humans acquire a vast vocabulary of relations, and then apply them freely to many disparate situations (Gentner & Hoyos, 2017). This capacity requires the solution to a key problem: How does the cognitive system choose which relations to use to represent the task at hand? This is similar to the representational learning problem studied in sequential decision making research, where subjects are presented with multiple choice tasks in which objects vary in several dimensions and they receive rewards for selecting an object with a specific combination of values on those dimensions (e.g., Canas & Jones, 2010; Leong, Radulescu, Daniel, DeWoskin, & Niv, 2017; Niv et al., 2015; Wang & Rehder, 2017). In this previous research, however, the tasks studied, and the posited underlying representations, are feature-based (or propositional) instead of relational. As previous theories of human representation learning (Jones & Canas, 2010; Niv, 2019; Radulescu, Niv, & Ballard, 2019; Radulescu, Shin, & Niv, 2021), we propose that this problem can be tackled via reinforcement learning (RL). However, because relational representations pose distinct problems for standard RL algorithms, we adapt methods developed in the sub-field of relational RL (RRL; for a review see Otterlo, 2012) to build a simple model that learns to select adequate relational representations when learning a policy for a particular task. We tested our model in relational versions of three Atari...
4.1. Introduction

In the following, we give a brief overview of RL, RRL and then describe our model. We present our simulations, and then discuss our model in the context of current models of theory-based RL and analogical reasoning.

4.1.1 Reinforcement Learning

In RL an agent interacts with the environment, taking actions in order to maximise rewards and avoid punishment. The environment is conceptualised as a set of states with transitions between them probabilistically determined by the agent's actions (e.g., in a discrete maze environment the action NORTH might lead the agent to the room upfront with a probability of 0.75, and to a random room with a probability of 0.25). RL algorithms are designed to learn an optimal policy (i.e., a mapping between states and actions) through this interaction (Sutton & Barto, 2018).

In general terms, RL algorithms can be classified into model-based and model-free methods. In model-based methods the agent learns a transition function from states and actions to new states and a reward function from states to rewards. These two functions correspond to the model of the environment. The agent can use the model to plan the best course of action at a given state through simulation. In contrast, in model-free methods the agent uses prediction errors to learn the value (i.e., the expected cumulative reward) of taking each action in each state directly, without learning a model of the environment. These values can then be used to build the policy by greedy selection of the best action in each state. In the present paper we are mostly concerned with model-free learning. The most influential model-free algorithm is called q-learning (Watkins & Dayan, 1992). This method approximates the action value function according to the following update rule:

\[ Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left( R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right) \]  \hspace{1cm} (4.1)

where \( Q(S_t, A_t) \) (aka q-value) is the current value of taking action \( A \) on state \( S \), \( \alpha \) is the learning rate, \( \gamma \) is a discount factor that has the effect of weighting more rewards closer in time to rewards farther away in the future, and the subtraction term is the prediction error. Note that q-learning updates its value estimates based on the best next action, \( \max_a Q(S_{t+1}, a) \), even though the agent's current policy might choose a different next action. This is important because it allows q-learning to approximate the optimal policy while following a different sub-optimal policy (and thus more fully exploring the environment). As is usually done in q-learning, in the present paper we use a suboptimal policy known as \( \epsilon \)-greedy, where all
4.1. Introduction

the non-optimal actions are given a selection probability of $\frac{\varepsilon}{|A|}$, where $|A|$ is the size of the action space, and the action with the highest value is given a selection probability of $1 - \varepsilon + \frac{\varepsilon}{|A|}$. During training, the value of $\varepsilon$ is slowly decreased, so as to allow the agent to explore progressively less as it improves its value estimate.

While q-learning is guaranteed to converge to the optimal policy as long as all state-action pairs are updated during learning (Watkins & Dayan, 1992), it becomes prohibitively expensive for large state spaces. This is especially problematic in relational settings, where the size of the state space grows combinatorially with the number of relations and objects. As explained below, RRL algorithms use specialized function approximators to handle this problem.

4.1.2 Relational Reinforcement Learning

The goal of RRL is to learn an optimal policy in an environment described as a set of objects and relations between them (Džeroski et al., 2001). Notably, the policy is represented as a set of variabilized rules in first first order logic (or a subset of it). For example, in a version of the classic problem known as blocks world (Slaney & Thiébaux, 2001), where the agent is rewarded for unstacking a group of blocks, one such rule could be: move($X$, floor) ← on($X$, $Y$) $\land$ top($X$) (i.e., “if block $X$ is on any other block $Y$, and block $X$ is on top of a pile, then move block $X$ to the floor”). In contrast, the present paper concentrates in the learning of ground rules, i.e., rules that apply to specific objects in a specific task, such as: LEFT ← more-x(player, ball) (i.e., “if the player is to the right of the ball, then move left”). This is because we think that is more likely that people learn relational rules that apply to specific situations and, later on, those rules are generalized through the process of schema induction (e.g., Z. Chen & Mo, 2004; Gick & Holyoak, 1983).

From a cognitive point of view, an interesting attribute of classic RRL algorithms is that, in general, they build policies incrementally, gradually adding rules to the policy that improve its the overall quality (Driessens et al., 2001, see next section). This stands in stark contrast to theory-based Bayesian approaches to reinforcement learning (e.g., Tsividis et al., 2021), where the complete space of programs (which includes relational policies) is defined a priori and learning equates to infer the best-fitting program from all possible programs through probabilistic inference (we elaborate on the relation between top-down and bottom-up approaches in the Discussion). Another important attribute of RRL algorithms is that they have to deal with the discrete nature of relational representations. This is because describing the state of the environment in terms of relations imposes sharp partitions of the state space. For instance, on the aforementioned LEFT rule, the relation more-x(player, ball) partitions the state space into states where the player is to the right of the ball, and states where it is not. This fact requires the use of an specialized function approximator that can make use of relational representations to abstract away irrelevant aspects of the state space. As described below, our model makes use of a specialized function approximator developed for RRL.
4.1.3 Relational Regression Tree Learner

Our model, which we term relational regression tree learner (RRTL), is based on the function approximator for RRL proposed by Driessens et al. (2001, see also Driessens 2004). This algorithm uses regression trees to represent the state-action value function. In brief, for each action there is a tree where each node represents a conjunction of ground relations and each leaf is a predicted q-value. In the original model of Driessens et al. (2001), these trees were based on logical splits of the state-action space. To illustrate, Figure 4.1a shows a typical state of the game Breakout, where the player gains points by making the ball bounce against the wall of bricks at the top of the screen. In general, the action RIGHT has the highest value when the player is to the left of the ball, has a lower value when the ball and the player are at the same position on the x-axis, and has the lowest value when the player is to the right of the ball. To represent this ranking of values a logical regression tree needs to make two splits (true and false for more-x and true and false for same-x), as shown in Figure 4.1b. An alternative way of representing the same ranking is to make splits based on the comparative values more, same and less of the x relation between the player and the ball. In this case the tree needs to make only a single split, as depicted in Figure 4.1c.

To make a prediction the agent traverses the tree corresponding to each action according to the relations present in the state until it reaches a leaf. The predicted q-value can then be used to select an action and then updated according to 4.1. At the beginning of the learning process, all the trees have a single leaf. At this stage, the q-value represents the overall value of the action in the environment. All state-action trees consider the same initial set of candidate relations to grow new leaves. As the agent interacts with the environment, each state-action tree keeps track of the current number of visits to the candidate relation, n, the mean, \( \mu_n \), and the
scaled variance, $J_n = \sigma^2_n \cdot n$, of the q-values produced at each time step, as well as the same statistics for all potential partitions induced by the candidate (i.e., true and false for logical partitions and more, same and less for comparative ones). These statistics are calculated incrementally according to Equations 4.2-4.3 (for derivations see Finch 2009)¹:

$$\mu_n = \mu_{n-1} + \frac{x_n - \mu_{n-1}}{n}$$  (4.2)
$$J_n = J_{n-1} + (x_n - \mu_{n-1})(x_n - \mu_n)$$  (4.3)

were $\sigma^2_n = J_n / n$.

After a minimal sample size (a free parameter of the model) has been reached, these statistics can be used to compute, for each candidate, the $F$-ratio between the variance of the q-values if the leaf was split according to the candidate and the variance of the q-values of the unsplit leaf. Equations 4.4-4.5 show the $F$-ratio for logical and comparative splits, respectively:

$$F = \frac{\frac{n_T}{n_0} \sigma^2_T + \frac{n_F}{n_0} \sigma^2_F}{\sigma^2_O} = \frac{J_T/n_0 + J_F/n_0}{J_O/n_0}$$  (4.4)

$$F = \frac{\frac{n_M}{n_0} \sigma^2_M + \frac{n_S}{n_0} \sigma^2_S + \frac{n_L}{n_0} \sigma^2_L}{\sigma^2_O} = \frac{J_M/n_0 + J_S/n_0 + J_L/n_0}{J_O/n_0}$$  (4.5)

were $\sigma^2$ is the variance, $n$ is the total number of visits to the partition, and the subscript now indicates the partition ($T =$ true, $F =$ false, $M =$ more, $S =$ same, $L =$ less and $O =$ overall).

With this ratio, the tree calculates the $p$-values of a standard one-tailed $F$-tests for all candidates. If the smallest $p$-value is smaller than the significance level the leaf is split according to the candidate and the process continues until the tree cannot find new splits or reaches a maximum tree depth. In all our simulations we set the maximum tree depth to 10, the minimal sample size to 100,000, and the significance level to 0.001.

### 4.2 Simulation 1: Breakout

In Simulation 1, we tested our model on the game Breakout (see Figure 4.1a). This is the simplest environment that we used. As mentioned above, in Breakout the player controls a paddle and receives points when the ball bounces off the wall on the top of the screen. The player looses points (or lives) if the ball passes the paddle and disappears off the bottom of the screen (the paddle’s $y$-position is fixed at bottom of the screen). The actions available to

¹. The implementation described by Driessens 2004 uses the sum of squared q-values to calculate the variance, however, this can be numerically unstable.
the agent are: NOOP, FIRE, RIGHT, LEFT. Note that the actions NOOP and FIRE have the same effect during the game (i.e., the paddle does not move), however, at the beginning of the game the agent has to execute the FIRE action to start. To succeed on this game the agent needs to learn to follow the ball, which requires paying more attention to the relations across the x-dimension than to the relations across the y-dimension

To represent the state of the environment, we used the x and y relations between the player and the ball (henceforth, object relations) and the x and y relations between the ball at the current time step and the ball at the previous time step (henceforth, trajectory relations). In the object relations the first object was always the player and the second object was always the ball. In the trajectory relations the first object was always the object at the current time step and the second was the object at the previous time step. We created two versions of the state, a logical version and a comparative version. Table 4.1 presents all relations considered in this simulation. Note that even in this simple environment a tabular representation of the same state-action space would need $3^4 = 81$ rows $\times 4$ columns. During the construction of the state, we filtered out all states where the ball was not present (i.e., those states were treated as empty). This was done because of the frequentist statistics approach used to determine the state-action tree splits: to compete in equal grounds, all candidates require the same number of visits.

2. There are certainly more complex and effective policies like digging a tunnel through the wall to allow the ball to bounce above the blocks (e.g., Mnih et al., 2015). However, as we are not representing the wall, following the ball is the optimal policy in the environment represented at this level of abstraction.

3. We did not represent the trajectory relations for the player because, as noted above, the y-trajectory of the player is a constant and, for any consistent policy in Breakout, there is a correlation between the x-trajectory of the player $x(\text{player}_t, \text{player}_{t-1})$ and the action taken $x(\text{player}_t, \text{player}_{t+1})$.

---

**Table 4.1: Breakout State Representation**

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Object-1</th>
<th>Object-2</th>
<th>Logical</th>
<th>Comparative</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>player,</td>
<td>ball,</td>
<td>more-x(player, ball)</td>
<td>x(player, ball)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, ball)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, ball)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player,</td>
<td>ball,</td>
<td>more-y(player, ball)</td>
<td>y(player, ball)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, ball)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, ball)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>ball,</td>
<td>ball_{t-1}</td>
<td>more-x(ball, ball_{t-1})</td>
<td>x(ball, ball_{t-1})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(ball, ball_{t-1})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(ball, ball_{t-1})</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>ball,</td>
<td>ball_{t-1}</td>
<td>more-y(ball, ball_{t-1})</td>
<td>y(ball, ball_{t-1})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(ball, ball_{t-1})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(ball, ball_{t-1})</td>
<td></td>
</tr>
</tbody>
</table>
4.2. Simulation 1: Breakout

We trained and tested our model in the environment “BreakoutDeterministic-v4” of the OpenAI gym toolkit (Brockman et al., 2016). To obtain a relational state from the gym environment we created a visual pre-processor, which used the color and shape of the objects to calculate the x and y positions of the centers of the ball and the paddle. To obtain the relations shown on Table 4.1, we set the roles “Object-1” and “Object-2” according to a hierarchy of objects where “Object-1” was always the player and “Object-2” was always the ball. To calculate the comparative value associated with each relation we subtracted the x and y positions according to the same hierarchy and categorized the difference with a tolerance level of six pixels (e.g., if the x-center of the player was above 6 pixels to the right of the ball the comparative value was more, if the difference was below six pixels the comparative value was same, and else the comparative value was less). We calculate the trajectory relations in the same way, except that we used a tolerance level of zero. All RL agents in the present paper included an action buffer that store the last 10 actions and checked whether the agent had taken the same action 10 times in a row, in which case the current action was uniformly sampled from the actions space. For each of the two versions of the model, we trained 10 runs on 2,000,000 iterations with RL parameters $\alpha = 0.1$, $\gamma = 0.99$ and $\varepsilon$ was decayed from 1.0 to 0.1 according to the formula: $\varepsilon_t \leftarrow d\varepsilon_{t-1}$ with $d = 0.9999995$. The reward signal was transformed by the sign function. The code for all our simulations is available at https://github.com/GuillermoPuebla/rrl.

4.2.1 Results and Discussion

Figure 4.2 shows the cumulative return for the logical and comparative versions of the model for each one of the 10 runs. Overall, the runs of the comparative version received more rewards during training. Because in Breakout episodes with higher returns are necessarily longer, the runs in this version reached the 2,000,000 iterations in fewer episodes. Furthermore, the runs on the comparative version were also more consistent, namely, there was less variability on total return and number of episodes across runs than in the logical version.

During training, we saved the agent’s set of state-action trees every 200,000 iterations. After the training session finished, we tested each set of trees on 10 games and picked the best one, which became the agent test trees. For testing we set $\varepsilon$ to 0.05. Figure 4.3 shows the average return on 100 test episodes. As can be seen, all runs in both models achieved higher than chance returns (defined as the average returns of an agent taking random actions on 100 episodes). Furthermore, the maximum average return across runs was similar in both versions. However, the results were much more consistent in the comparative version, with eight out of 10 runs achieving average returns above 50 in comparison to only three in the logical condition.
4.2. Simulation 1: Breakout

Figure 4.2: Cumulative return by model version and run on Breakout.

Figure 4.3: Mean returns by model version and run on 100 test episodes of Breakout. Error bars are 95% confidence intervals. The horizontal shadowed line represents the mean return and 95% confidence intervals for a random agent on 100 test episodes.
Figure 4.4: Best state-action trees on Breakout for the logical (a) and comparative (b) versions of RRTL.

```python
    if more-x(player, ball) in state
        LEFT
    else if same-x(player, ball) in state
        FIRE
    else if less-x(player, ball) in state
        RIGHT
    else
        sample_action_space()
```

Figure 4.5: If-then rule corresponding to the best state-action tree on Breakout.

Figure 4.4 shows the state-action trees of the best run of the logical (a) and comparative (b) versions of RRTL. As can be seen, the best run of the logical version of RRTL achieved to discriminate between the different comparative values of the object x relation only regarding the LEFT action. However, this LEFT tree also introduced a test involving the trajectory of the ball in the y dimension, which increases the overall complexity of the policy. All other state-action trees made splits based on the on the object x relation but did not discriminate between all its comparative values. In contrast, the best run of the comparative version of RRTL discriminated between all the comparative values of the object x relation for all actions.

To make the relational policy learned by the agent in this run easier to read, the state-action trees were translated into the if-then relational rule shown in Figure 4.5. Clearly, the agent learned to follow the ball, going left if the player was to the right of the ball, going right if the player to the left of the ball and not moving if the player and the ball were at the same x-position. Notably, the agent preferred the FIRE action to the NOOP action for not moving, since the game would not start unless the FIRE action was executed and therefore the agent would not receive any rewards.
4.2 Simulation 1: Breakout

The results of Simulation 1 showed that the RRTL model is capable of learning a relational policy for a simple game—that nonetheless implies a large state-action space unmanageable for a straightforward tabular approach. Furthermore, we found a clear advantage for using comparative splits vs. the original logical splits. In Simulations 2 and 3 we tested our model in two environments that included increasing numbers of potentially relevant relations.

4.3 Simulation 2: Pong

In the Pong environment (see Figure 4.6) two paddles, one corresponding to the player and the other to the enemy, hit a ball with the objective of getting the ball passed the opponent. The player receives positive rewards (and the player scores a point) when the ball passes the enemy paddle, and negative rewards (and the opponent scores a point) when the ball passes their own paddle. The episode ends then either the player or the enemy score 21 points. Both the player and the enemy can move only on the y-axis, while the ball can move on the x- and y-axis. The actions available to the player in this environment are: NOOP, FIRE, RIGHT, LEFT, RIGHTFIRE, LEFTFIRE. However, as the last two actions had the same effect as RIGHT and LEFT, we omitted them for the sake of simplicity. Because in Pong there are three objects instead of two, the number of potential object and trajectory relations increases accordingly. Furthermore, besides the object and trajectory relations used in the previous simulation, we added two contact relations between the player and the ball, and the ball and the enemy (these relations are necessarily logical). Table E.1 on Appendix E presents all the relations used to represent the state of the Pong environment. Note that to represent the state of Pong in a tabular fashion using the relations of Table E.1 one would need to build a $3^9 \cdot 2^2 = 78,732$ rows $\times$ 4 columns table.

![Figure 4.6: A typical state of the Pong environment.](image)

We trained and tested our model in the environment “PongDeterministic-v4” of the OpenAI gym. As in Simulation 1 we created a visual pre-processor that calculated the x- and y-positions of the center of the objects. We used a tolerance level of 4 pixels to calculate the comparative value associated with each object relation. For the trajectory relations we used a tolerance level of zero. The following hierarchy of objects was used to represent the state:
player > ball > enemy (see Table E.1). For each of the two versions of the model, we trained 10 runs on 2,000,000 iterations with the same parameters as Simulation 1, except for two modifications. First, the \( \epsilon \) decay parameter, \( d \), was set to 0.9999977. This allowed for a longer period of initial exploration. The second modification is related to the fact that the enemy in Pong follows the ball by default \( r_{\text{enemy,ball}} = 0.84 \) in 10 random games. This has the effect of making the reward signal very sparse for an agent that follows a random policy (as is any RL agent following a \( \epsilon \)-greedy policy at the beginning of learning). To address this issue, we added 0.1 to the reward at each time step, encouraging the agent to play for as long as possible.

4.3.1 Results and Discussion

Figure 4.7 shows the cumulative return for the logical and comparative versions of RRTL for each one of the 10 runs. The cumulative returns on Pong are smoother in comparison to the ones on Breakout because of the extra reward term. Overall, the runs of the comparative version received slightly more rewards during training. However, it is clear from the training data that the three runs of the comparative version received only the rewards corresponding to the extra term. This also happened for two runs of the logical version.

We tested the model runs in the same way as Simulation 1. We re-scaled the returns to the \([0, 21]\) interval by adding 21. This can be interpreted as the number of times the player scored a point against the enemy. Figure 4.8 shows the average return on 100 test episodes. As can be seen, eight out 10 runs achieved performance above chance in the logical version and seven out of 10 in the comparative version. However, three runs of the comparative version score around 14 points against the enemy on average, clearly surpassing the best logical run, which scored around nine points.
4.3. Simulation 2: Pong

Figure 4.8: Mean returns by model version and run on 100 test episodes of Pong. Error bars are 95% confidence intervals. The horizontal shadowed line represents the mean return and 95% confidence intervals for a random agent on 100 test episodes.

Figure 4.9: State-action trees on Pong for (a) the best run, and (b) the worst run. Both are runs from the comparative version of RRTL. The state-action tree of the worst run in (b) is based on the last training iteration.

```python
if more-y(player_t, ball_t) in state
    RIGHT
else if same-y(player_t, ball_t) in state
    FIRE
else if less-y(player_t, ball_t) in state
    LEFT
else
    sample_action_space()
```

Figure 4.10: If-then rule corresponding to the best state-action tree on Pong.
4.3 Simulation 2: Pong

Figure 4.9 shows the state-action trees of the best (a) and worst (b) run on on Pong. Both are runs of the comparative version of RRTL. The best run selected the \( y \) relation between the player and the ball and discriminated between all its comparative values for all actions. Figure 4.10 shows the corresponding if-then relational rule. As in Breakout, in this environment RRTL also learned to follow the ball, but this time in the \( y \) dimension, going down if the player was above the ball, going up if the player was below the ball and not moving if the player and the ball were at the same \( y \)-position. The agent preferred the FIRE action to the NOOP action for not moving, although a close inspection of Figure 4.9 (a) shows that the predicted q-values are similar for both actions. Figure 4.9 (b) shows the state-action trees at the last training iteration. As can be seen, in this run the FIRE and RIGHT actions based their first split on the \( y \) trajectory relation of the ball, and the resulting set of trees was not able to compensate for these early splits. This highlights one of the limitations of the current version of RRTL model: it cannot unsplit a node that led to bad performance. We elaborate on this and other limitations of the RRT model in the Discussion.

Simulation 2 showed that the RRTL model is capable of learning a relational policy for a game larger than Breakout. Additionally, we found further evidence in favor of using comparative splits vs. logical splits in the regression trees. In simulation 3 we tested the RRTL model on an even larger environment that included many more objects than Pong, providing an even more stringent test of the capabilities of the RRTL model.

4.4 Simulation 3: Demon Attack

In Demon Attack (see Figure 4.11) the player controls a spaceship at the bottom of the screen that can only move on the \( x \)-dimension. In the initial levels, big enemies (or demons) appear in waves of three in the upper part of the screen. The bottom-most enemy shoots projectiles that on contact with the player make the player lose a life and receive a negative reward. The player can shoot missiles that destroy the enemies on contact, and give the player a positive reward. When an enemy is destroyed a new one appears to take its place until the current level is completed. Once all the enemies on a particular level are destroyed, the player moves on to the next, more difficult wave. On advanced levels the big enemies will split into two bird-like small enemies the first time they are shot. The small enemies will eventually attempt descent onto the spaceship, which will also cause the player to lose a life and receive a negative reward on contact. The actions available to the player in this environment are: NOOP, FIRE, RIGHT, LEFT, RIGHTFIRE, LEFTFIRE. We used all the available actions in this simulation. Because in Demon Attack there can be up to three big enemies and up to six small enemies at any given time\(^4\), the number of potential object relations is quite large. In this simulation we only used objects relations between the player and the other objects in the screen, with the

\(^4\) With the constraint that for each pair of small enemies there is one less possible big enemy.
exception of the player missile, which we treated as part of the player’s action. Furthermore, we treated the enemy’s projectiles as a single object, which we termed e-missile. We did not use any trajectory relations in this simulation\(^5\). When building the relational state we only considered states where there was an enemy missile. Table E.2 on Appendix E presents all the relations used to represent the state of this environment. Note that to represent the state of Demon Attack in a tabular fashion using the relations of Table E.2 one would need to build a \(4^{19} \cdot 3 = 824,633,720,832\) rows \(\times\) 6 columns table.

We trained and tested our model in the environment “DemonAttackDeterministic-v4” of OpenAI gym. As in previous simulations we created a visual pre-processor that calculated the x- and y-positions of the center of the objects. We used a tolerance level of 3 pixels to calculate the comparative value associated with each object relation. For each of the two versions of the model, we trained 10 runs on 3,000,000 iterations with the same parameters as Simulation \(^6\). In addition to the action buffer described earlier, RL agents in this simulation included a reward buffer that stored the last 300 rewards and checked whether the last 300 iterations yielded zero rewards, in which case the current episode was terminated and a new one was started.

### 4.4.1 Results and Discussion

Figure 4.12 shows the cumulative return for the logical and comparative versions of RRTL for each one of the 10 runs. Overall, the runs of the comparative version received more rewards during training. Interestingly, the episodes tended to be shorter in the comparative version, which resulted in more training episodes in comparison to the logical version.

\(^5\) This was due to the fact the demons follow a non-linear trajectory in the x- and y-dimensions even when stationary, circling around a fixed point.

\(^6\) Note that all runs on the comparative version of RRTL achieved their best performance before two million iterations, so the last million of training iterations was, in fact, unnecessary.
4.4. Simulation 3: Demon Attack

We tested the model runs in the same way as Simulation 1. Figure 4.13 shows the average return on 100 test episodes. We considered two baselines, a first random agent that sampled actions uniformly from the full set of actions, and a second random agent that sampled actions only from the “fire” actions (i.e., FIRE, RIGHTFIRE, and LEFTFIRE), and therefore was always firing missiles. As can be seen, all runs in all conditions achieved performance above our two baselines. However, seven runs of the comparative version score above 2500 points in average, clearly surpassing the best logical run, which scored around 1700 points in average.

Figure 4.14 shows the state-action trees of the best (a) and worst (b) run on Demon Attack. Both are runs of the comparative version of RRTL. The best run selected the $x$ relation between the player and the enemy missile, although it only made splits for the FIRE and RIGHTFIRE actions. Figure 4.15 shows the corresponding if-then relational rule. As can be seen, this model run learned a simple policy that essentially amounts to stay away from the enemy missile while shooting as much as possible. In particular, the agent will stay put and fire if the player was to the right of the enemy missile, will go right and fire if the player and the enemy missile are at the same $x$-position, and will go left and fire if the player is to the left of the enemy missile. As in Simulation 2, to evaluate the learning process of the worst run, Figure 4.14 (b) shows the state-action trees of the worst run at the last training iteration. As can be seen, in this run the FIRE action based its first split on the $y$ relation between the player and the enemy missile, which led to a set of trees that was not able to compensate for this split. This further highlights of RRTL’s incapability of undoing a bad split during learning.
4.4. Simulation 3: Demon Attack

Figure 4.13: Mean returns by model version and run on 100 test episodes of Demon Attack. Error bars are 95% confidence intervals. The horizontal shadowed red line represents the mean return and 95% confidence intervals for a random agent on 100 test episodes. The horizontal shadowed pink line shows the same information for a random agent that only has access to the “fire” actions.

Figure 4.14: State-action trees on Demon Attack for (a) the best run, and (b) the worst run. Both are runs from the comparative version of RRTL. The state-action tree of the worst run in (b) is based on the last training iteration.

if more-x(player, e-missile)
    FIRE
else if same-x(player, e-missile)
    RIGHTFIRE
else if less-x(player, e-missile)
    LEFTFIRE
else
    sample_action_space()

Figure 4.15: If-then rule corresponding to the best state-action tree on Demon Attack.
4.5 General Discussion

In this paper, we studied the problem of how to select appropriate relational representations to build a policy when there is a large vocabulary of relations available to describe the state of the environment. Using a function approximator developed in RRL, we developed a model, RRTL, that learns ground relational policies by making ternary splits based on the comparative values more, same and less that characterize comparative relations like “above” or “bigger-than”. We tested our model on three Atari games, Breakout, Pong and Demon Attack, that involved increasing numbers of potential relations. In each case, RRTL built simple relational policies based on a set of few relevant relations. RRTL can be considered a proof-of-principle of the idea of using RL to learn which relations between the objects in the environment are relevant to characterize the task at hand.

As previous models developed in RRL (e.g., Driessens et al., 2001), RRTL builds relational policies incrementally, adding relational representations gradually during learning. In contrast, the Bayesian model of theory-based RL of Tsividis et al. (2021), represents the full set of possible programs and selects, through Bayesian inference, the best possible program that explains the observed data and yields high rewards. While this model is an impressive demonstration of the capabilities of a fully top-down approach, it comes with at the cost of intractable computation for a human learner in naturalistic settings, a problem that has been pointed out before regarding Bayesian models of cognition (Jones & Love, 2011). We believe that, ultimately, top-down and bottom-up approaches are complementary. On the one hand, RRL can be seen as a largely associative process running “in the background”, that pre-selects relevant relational representations, limiting the potential search space and therefore making Bayesian inference over potential programs more tractable. On the other hand, theory-based hypothesis generation could, occasionally, inform the generation of splits in RRTL. As incremental learning of programs is a hard problem, this will be necessary to learn relational policies in more complex environments than the ones studied in this paper. In either case, we think that RRTL provides a starting point to model these interactions.

In its current form, RRTL has several limitations. The first one has to do with the frequentist approach used to select the relation candidates for splitting the state-action trees. Because frequentist tests are highly sensitive to the sample size, all candidates should have similar numbers of visits to be comparable. This is a problem for any realistic environment, since there can be relations that apply only rarely that, nonetheless, have a large impact of the optimal policy. For example, imagine a version of Demon Attack where the small enemies descend onto the spaceship too fast to try to shoot them from below, in which case the best option would be to avoid the small enemies and, consequently, the x relation between the player and the small enemy would be very important, but only so when there is a small enemy on the screen. We think that in this case taking a Bayesian approach to candidate selection would be useful to account for this kind of phenomenon. A related problem is the
fact that RRTL currently does not have a way to undo a bad split made during learning, which can completely ruin the learning trajectory of the state-action trees (as seen in Simulations 2 and 3). Fortunately, Ramon, Driessens, and Croonenborghs (2007) developed a set of tree restructuring operations that they used for partial policy transfer on RRL. We think that combining these operations with a Bayesian approach to splitting candidate selection is a promising future direction for RRTL.

In contrast to most RRL systems developed in AI, RRTL learns policies composed of a set of ground rules. We proposed that, instead of learning variabilized rules directly through interaction with the environment, humans “lift” initially ground policies to a representation more akin to a first-order relational policy through the process of schema induction. This process involves the extraction of the shared relational structure through analogical comparison and mapping of two different situations (Doumas et al., 2008; Hummel & Holyoak, 2003). This same process could be used to compare different instances of the ground policy to isolate its relational content and abstract away the specific objects attached to it. Foster and Jones (2017) developed a computational model that uses RL to guide the induction of useful schemas to solve a task. Analogy-making also plays a major role in transfer learning, allowing humans to apply what they learned about one task to another that only shares abstract relational structure. Doumas et al. (2022) showed that DORA (Doumas et al., 2008), a model of relation learning and relational reasoning, was able to perform relational policy transfer between Breakout and Pong through analogical inference. We think that RRTL could be naturally integrated with both approaches.

RRTL uses the relational regression tree developed by Driessens et al. (2001) to approximate the state-action value function. We chose this function approximator because it integrates naturally with existing models of analogical reasoning based on relational representations (e.g., Doumas et al., 2008; Falkenhainer et al., 1989). However, it is worth noting that recently several deep neural network models of RRL have been proposed (e.g., Dong et al., 2019; Jiang & Luo, 2019; Zimmer et al., 2021). In general, these models define a soft logic (i.e., one with truth values falling into the $[0, 1]$ interval) and a series of differentiable gating operations that allow to approximate the behaviour of a logic program. While these models demonstrate the possibility of combining symbolic and neural-based computation in AI, they tend to suffer from scalability (Dong et al., 2019), or lack of interpretability (Dong et al., 2019). Both of these issues are the subject of current research in neuro-symbolic AI (e.g., Zimmer et al., 2021).

To conclude, RRL provides a computational framework to understand how the cognitive system learns which relational representations are relevant for a given task, as well as how to build relational policies incrementally. We hope that the present paper helps to stimulate further theoretical and empirical research in this area.
Chapter 5

General Discussion

This thesis has explored several aspects of relational reasoning, including its computational prerequisites, the specific mechanisms needed to achieve effective relational responding, and how these mechanisms are implemented in DORA, a neurally plausible model of analogical reasoning. Furthermore, it was shown that, through analogical inference, DORA is capable of transferring what it has learned about one domain to another superficially dissimilar domain that is nevertheless structurally similar. When applied in the context of an reinforcement learning agent learning to play a simple Atari video game, this results in far transfer of the learned policy to a new game. The last part of this thesis used tools from the AI field of relational reinforcement learning to develop a model that can select adequate relational representations to characterize its environment and learn effective relational policies from rewards. The following sections expand and reflect on the main topics covered throughout this thesis.

5.1 Relational Reasoning and the Need for a Binding Mechanism

Chapter 2 showed that both a traditional connectionist model and a modern deep neural network of language comprehension were able to mimic relational responding under certain conditions. However, closer examination showed that both models were doing so by exploiting the statistical regularities of their training data. When presented with new tests that shared the same relational structure as the training data but varied its surface statistical properties with respect to the training data, both models failed to generalize. I argued that the main reason for this failure at abstracting the relational structure of the task and using it as a basis for generalization, is the lack of a capacity to perform dynamic binding of the representations of objects and relations. This is consistent with the pattern of results shown in simulation 4 of Chapter 3, where disrupting DORA’s binding mechanism, neural synchrony, yielded the model incapable of learning relational representations or performing analogical inference. Although there is still controversy in the cognitive neuroscience literature regarding the role of neural synchrony in feature binding and relational reasoning (for reviews see, Holyoak & Monti, 2021; Uhlhaas et al., 2009), there has been a resurgence in the interest on temporal synchronization in the artificial neural networks community as a potential mechanism for binding (e.g., Reichert
5.1. Relational Reasoning and the Need for a Binding Mechanism

& Serre, 2013; Ricci, Jung, et al., 2021, for a review see Greff et al., 2020). As shown in Chapter 3, neural synchrony provides an effective mechanism for binding representations of relational roles and objects, which in turn forms the basis upon which a neural system can perform analogical inference. To the best of my knowledge, an alternative mechanism that can enable the emergence of high-level forms of relational reasoning in a neural system has not been identified yet. Ultimately, I think that to advance towards artificial intelligence systems that can learn and reason in a human-like way, artificial neural network researchers will need to face the binding problem head on and provide mechanisms for it, as well as compatible training algorithms.

5.2 Analogy Making, Relation Learning and Transfer

The kind of relational processing studied in Chapter 2 constitutes a very basic form of relational reasoning where the reasoner needs to identify the object that is playing a particular role when asked directly about it (e.g.: “Was the restaurant near of far away?” in Simulation 2). In contrast, analogy-making is an advanced form of relational reasoning where the reasoner needs to recognise the functional structure of a target situation, retrieve a relevant source memory, and map the elements of the target to the elements of the source while respecting the functional dependencies between the elements of both the source and the target. To do all of this, an effective model of relational reasoning needs to be able to keep objects and relational roles separated at the representational level, as well as representing their bindings in a dynamic way. The LISaese representational format allows to keep objects and relational roles separated and to exploit the semantic content of relational roles. This, in turn, enables retrieving source representations from memory based solely on their relational similarity to the target, or to “fill in” missing information about the target representation based on its structural similarity with the source representation (i.e., a familiar situation). As shown in Chapter 3, when these same processes are applied to relational representations of a policy, it results in analogical policy transfer. In general, many forms of “far transfer” can be seen as instances of analogical inference where a response to a new situation is generated through analogy to a previously formed representation.

Interestingly, the same cognitive machinery that enables analogy making plays a central role in the development of the relational representations that are used in this advanced form of relational reasoning. As shown in Chapter 3, analogical comparison forms the basis of the discovery of relational invariant content. More generally, analogy making is a mechanism for hypothesis generation that plays a key role in the development of high order abstractions such as relational categories (e.g., carnivore) that constrain generalization in different situations (Gentner & Hoyos, 2017). In conjunction with other “constructive thinking” mechanisms, such as thought experiments and explanation, it provides a means for genuine conceptual change and building intuitive theories of various knowledge domains (F. Xu, 2019).
5.3 Relational Reinforcement Learning and Analogy Making

Historically, theories of analogy making have been disconnected from notions of utility such as rewards or prediction errors (but see Foster & Jones, 2017, for a model of reward-driven schema induction). One reason for this disconnect is the emphasis of these theories on unsupervised discovery of relational representations and higher order abstractions, such as schemas or relational categories. However, this state of affairs is odd. After all, the capacity to form and manipulate relational representations provides humans with a clear evolutionary advantage over any other species (Penn et al., 2008) and therefore allows humans to act more effectively in their environment. To start to bridge this theoretical gap, this thesis looked into ways of integrating relational representations with the reinforcement learning framework. This is a hard problem because of the discrete nature of relational representations and the combinatorial explosion that results from applying them to describe the state of the environment.

The AI literature provides a good first approximation to this problem in the form of relational reinforcement learning. Chapter 4 showed that, with a few modifications, the function approximator developed by (Driessens et al., 2001) selects relevant relational representations from a large vocabulary and use them to build a relational policy. Specifically, Chapter 4 showed that by organizing the (state-action) value space in terms of the ternary partitions implied by the values “more”, “same” and “less” of comparative relations, the proposed model, RRTL, achieved higher and more stable rewards in three Atari games. Importantly, and in contrast with a recent Bayesian approach to relational reinforcement learning (Tsividis et al., 2021), this model learns relational policies incrementally. This is important from a cognitive and developmental point of view because a satisfactory model of relational RL has to operate within the cognitive resource constrains of human information processing.

This does not mean that Bayesian hypothesis selection and belief revision do not play an important role in relational RL. As argued in Chapter 4, I believe that, ultimately, bottom-up relational RL and top-down hypothesis testing should interact. In contrast to the proposal of Tsividis et al. (2021), I think that top-down processes should be invoked only when necessary. In the context of the RRTL model, the place for this interaction should be in the sampling of splitting relations inside the state-action trees during learning. As RRTL is strictly incremental, without sampling of candidate partitions the model will not be able to learn policies that cannot be built step by step. As I mention in Chapter 4, a main challenge before pursuing this kind of integration is to figure out a way to compare competing splitting candidates that may be seen with very different frequencies in the environment and implement unsplitting tree operations.

More generally, I believe that integrating analogical reasoning with utility-related notions from sequential decision making is a promising avenue of future research. In this sense, it is worth noting that RRTL is compatible with the analogical transfer learning model presented in Chapter 3, and with the reward-driven schema induction model of Foster and Jones (2017).
5.4 Conclusion

This thesis explored several aspects of relational processing, including representation, reasoning, and generalization. One of the main conclusions of this thesis is that, to achieve human-like relational and analogical generalisation, neural networks models need to integrate specific mechanism for dynamic binding of independent representations of objects and relational roles. A second main conclusion is that there is much to gain by integrating relational reasoning with reinforcement learning. After all, relational representations evolved because they allowed us to act more effectively in the environment. I hope that the work presented here helps to bring these two separate disciplines together.
Table A.1: Concepts used in all the scripts.

<table>
<thead>
<tr>
<th>Roles</th>
<th>Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>agents</td>
<td>Albert, Clement, Gary, Adam, Andrew, Lois, Jolene, Anne, Roxanne, Barbara, he, she, jeep, station-wagon, Mercedes, Camaro, policeman, waiter, judge, AND</td>
</tr>
<tr>
<td>topics</td>
<td>decided, distance, entered, drove, proceeded, gave, parked, swam, surfed, spun, played, weather, returned, mood, found, met, quality, ate, paid, brought, counted, ordered, served, enjoyed, tipped, took, tripped, made, rubbed, ran, tired, won, threw, sky</td>
</tr>
<tr>
<td>patients or themes</td>
<td>Albert, Clement, Gary, Adam, Andrew, Lois, Jolene, Anne, Roxanne, Barbara, he, she, jeep, station-wagon, Mercedes, Camaro, ticket, volleyball, restaurant, food, bill, change, chardonnay, prosecco, credit-card, drink, pass, slap, cheek, kiss, lipstick, race, trophy, frisbee</td>
</tr>
<tr>
<td>recipients or destinations</td>
<td>Albert, Clement, Gary, Adam, Andrew, Lois, Jolene, Anne, Roxanne, Barbara, he, she, jeep, station-wagon, Mercedes, Camaro, beach, home, airport, gate, restaurant, waiter, park</td>
</tr>
<tr>
<td>locations</td>
<td>beach, airport, restaurant, bar, race, park</td>
</tr>
<tr>
<td>manners</td>
<td>long, short, fast, free, pay, big, small, not, politely, obnoxiously</td>
</tr>
<tr>
<td>attribute</td>
<td>far, near, sunny, happy, raining, sad, cheap, expensive, clear, cloudy</td>
</tr>
</tbody>
</table>
Table A.2: Park Script.

<table>
<thead>
<tr>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;agent-1&gt; and &lt;agent-2&gt; decided to go to the park</td>
</tr>
<tr>
<td>The distance to the park was &lt;near/far&gt;</td>
</tr>
<tr>
<td>&lt;agent-1&gt; got in &lt;vehicle&gt;</td>
</tr>
<tr>
<td>&lt;agent-1&gt; drove &lt;vehicle&gt; to the park for a &lt;short/long&gt; time</td>
</tr>
<tr>
<td>&lt;agent-1&gt; proceed to the park fast</td>
</tr>
<tr>
<td>&lt;agent-1&gt; parked at the park for &lt;free/pay&gt;</td>
</tr>
<tr>
<td>The weather was sunny</td>
</tr>
<tr>
<td>&lt;agent-1&gt; ran through the park</td>
</tr>
<tr>
<td>&lt;He/She&gt; threw a Frisbee to &lt;agent-1/agent-2&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>The roles agent-1 and agent-2 never correspond to ‘Clement’ or ‘Roxanne’</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deterministic rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>The distance to the park determines driving time completely: near → short, far → long</td>
</tr>
</tbody>
</table>

Table A.3: Airport Script.

<table>
<thead>
<tr>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;agent-1&gt; decided to go to airport</td>
</tr>
<tr>
<td>Distance to airport &lt;near/far&gt;</td>
</tr>
<tr>
<td>&lt;agent-1&gt; found change</td>
</tr>
<tr>
<td>&lt;agent-1&gt; drove &lt;vehicle&gt; to airport &lt;short/long&gt;</td>
</tr>
<tr>
<td>&lt;agent-1&gt; ran to gate</td>
</tr>
<tr>
<td>&lt;agent-1&gt; met &lt;agent-2&gt; at airport</td>
</tr>
<tr>
<td>&lt;agent-1&gt; &lt;agent-2&gt; returned home</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>The roles agent-1 and agent-2 never correspond to ‘Gary’ or ‘Jolene’</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deterministic rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>The distance to the airport determines driving time completely: near → short, far → long</td>
</tr>
</tbody>
</table>
Table A.4: Bar Script.

Script

<agent-1> met <agent-2> at the bar
AND if agent1 = rich (1.0):
    <agent-1> enjoyed expensive-wine at the bar
AND if agent1 = cheap (1.0):
    <agent-1> did not enjoy expensive-wine at the bar
<agent-2> ordered a drink to the waiter at the bar
AND if agent2 = rich (1.0):
    The drink was expensive
AND if agent2 = cheap (1.0):
    The drink was cheap
OR (2):
    (0.5):
    <agent-2> made a polite pass at <agent-1>
OR (2):
    (0.3):
    <agent-1> gave a slap to <agent-2>
    <agent-2> rubbed cheek
    (0.7):
    <agent-1> gave a kiss to <agent-2>
    <agent-2> rubbed lipstick
(0.5):
    <agent-2> made an obnoxious pass at <agent-1>
OR (2):
    (0.7):
    <agent-1> gave a slap to <agent-2>
    <agent-2> rubbed cheek
    (0.3):
    <agent-1> gave a kiss to <agent-2>
    <agent-2> rubbed lipstick

Concept restrictions

The roles agent-1 and agent-2 never correspond to 'Andrew' or 'Barbara'

Deterministic rule

The action of agent-1 determines what agent-2 rubes completely: slap → cheek, kiss → lipstick
**Table A.5:** Beach Script.

<table>
<thead>
<tr>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;\text{agent}&gt;) decided to go to the beach</td>
</tr>
<tr>
<td>The beach was far away</td>
</tr>
<tr>
<td>OR (2):</td>
</tr>
<tr>
<td>(0.5):</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) entered (&lt;\text{vehicle}&gt;)</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) drove (&lt;\text{vehicle}&gt;) to the beach for a long time</td>
</tr>
<tr>
<td>AND if agent1 = male (1.0):</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) proceeded (&lt;\text{vehicle}&gt;) to the beach fast</td>
</tr>
<tr>
<td>AND (0.5):</td>
</tr>
<tr>
<td>The policeman gave a ticket to (&lt;\text{agent}&gt;)</td>
</tr>
<tr>
<td>(0.5):</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) drove (&lt;\text{vehicle}&gt;) to the beach for a long time</td>
</tr>
<tr>
<td>AND (0.8):</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) swam in the beach</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) won the race in the beach</td>
</tr>
<tr>
<td>AND if agent1 = male (0.87):</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) surfed on the beach</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) spun</td>
</tr>
<tr>
<td>AND if agent1 = female (0.33)</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) surfed on the beach</td>
</tr>
<tr>
<td>AND (0.33):</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) played volleyball in the beach</td>
</tr>
<tr>
<td>OR (2)</td>
</tr>
<tr>
<td>(0.8)</td>
</tr>
<tr>
<td>The weather was (&lt;\text{sunny}&gt;)</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) returned home for a long time</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) was in a (&lt;\text{happy}&gt;) mood</td>
</tr>
<tr>
<td>(0.2):</td>
</tr>
<tr>
<td>The weather was (&lt;\text{cloudy}&gt;)</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) returned home for a long time</td>
</tr>
<tr>
<td>(&lt;\text{agent}&gt;) was in a (&lt;\text{sad}&gt;) mood</td>
</tr>
</tbody>
</table>

**Concept restriction**

The roles recipient and patient never correspond to ‘Camaro’

**Deterministic rule**

The weather determines the agent’s mood completely:

\[ \text{sunny} \rightarrow \text{happy}, \text{cloudy} \rightarrow \text{sad} \]
Appendix B

DORA Computational Details

B.1 Parts of DORA

As described in the main text, DORA consists of a long-term-memory (LTM) composed of three bidirectionally connected layers of units. Units in LTM are referred to as token units (or tokens). Units in the lowest layer of LTM are connected to a common pool of feature units. Token units are yoked to integrative inhibitors that integrate input from their yoked unit and token units in higher layers.

DORA learns representations of a form we call LISAese via unsupervised learning. Propositions in LISAese are coded by layers of units in a connectionist computing framework (see main text). At the bottom of the hierarchy, feature (or semantic) nodes code for the featural properties of represented instances in a distributed manner. At the next layer, localist predicate and object units (T1) conjunctively code collections of feature units into representations of objects and roles. At the next layer localist role-binding units (T2) conjunctively bind object and role T1 units into linked role-filler pairs. Finally, proposition units (T3) link T2 units to form whole relational structures.

Sets, groups of potentiated units, correspond to attention or working memory (WM) within a cognitive framework. The driver corresponds to DORA’s current focus of attention. The recipient corresponds to active memory. Token units are laterally inhibitive (units in the same layer inhibit one another) within, but not across, sets.

Each layer of token units is negatively connected to a local inhibitor, and all token unit are connected to a global inhibitor (I). Active token units in a layer inhibit the local inhibitor to inactivity. When no token units in a given layer are active, the local inhibitor becomes active, and sends a refresh signal to all tokens in that layer and below across LTM (see below). When no token units in the driver are active, the global inhibitor becomes active, and sends a refresh signal to all tokens across LTM (see below). Each layer of token units is connected to a clamping unit (C), that is excited by unclamped units in the layer below and inhibited by unclamped units in the layer above (see below). C units play a role in recruiting and activating token units during learning.
We use the term analog to refer to a complete story, event, or situation (e.g., from a single object in isolation, to a full proposition in LISAese). Analogs are represented by a collection of token units (T1-T3). Token units are not duplicated within an analog (e.g., within an analog, each proposition that refers to Don connects to the same “Don” unit). Separate analogs do have non-identical token units (e.g., Don will be represented by one T1 unit in one analog and by a different T1 in another analog). The feature units thus represent general type information and token units represent instantiations (or tokens) of those types in specific analogs.

B.2 Functional Overview of Processing in DORA

In this section we describe DORA’s operation in strictly functional terms. For a detailed description of how these operations are instantiated in the neural network using traditional connectionist computing principles see Doumas et al. (2008).

B.2.1 Retrieval

When there are items in the driver (i.e., DORA is attending to something), but nothing in the rest of AM, then DORA performs retrieval. In short, some representation \( b \) is retrieved into the recipient to the extent that it is similar to \( a \) in the driver and prevalent in memory, and other representations \( c \) in memory are not similar to \( a \) and are not prevalent in memory. Functionally, retrieval works as follows:

\[
\text{Ret}(B) \leftarrow f_{\text{Ret}}(\text{sim}(B,A^D),\text{sim}(C \neq B,A^D), p(B), p(C)) \quad (B.1)
\]

where, \( \text{Ret}(B) \) is a retrieved representation \( B \), \( A^D \) is a driver representation (i.e., a collection of connected token units instantiating a LISAese representation), \( p(B) \) is the prevalence of representation \( B \) in LTM, and \( f_{\text{Ret}} \) is the retrieval function.

B.2.2 Mapping

When there are items in the driver (i.e., DORA is attending to something), and in the recipient, then DORA performs mapping. In short, representation \( a \) in the driver will map to representation \( b \) in the recipient to the extent that there are correspondences between \( a \) and \( b \), and there are not correspondences between \( a \) and any other items \( c \) in the recipient. Functionally, mapping works as follows:

\[
M(A^D,B^R) \leftarrow f_{\text{M}}(\text{sim}(A^D,B^R),\text{sim}(A^D,C \neq B^R)) \quad (B.2)
\]
B.2. Functional Overview of Processing in DORA

where, $M(A, B)$ is a mapping between $A$ and $B$ (instantiated as a learned bidirectional weighted connection), $B^R$ is a recipient representation, $\text{sim}(A, B)$ is a similarity function, and $f_M$ is the mapping function.

B.2.3 Relation learning

If DORA has learned mapping connections between representations in driver and recipient, then DORA can learn from the mapping. During learning there are two possibilities. In the first case, if two objects (e.g., $a$ and $b$) that are not already bound to predicates (i.e., no $T_2$ units are active) are mapped, then DORA learns a single-place predicate composed of the featural intersection of $a$ and $b$. In the second case, when sets of role-filler pairs are mapped—e.g., $P_i(a)$ and $P_j(b)$ are mapped to $P_k(c)$ and $P_l(d)$—and are not already linked into multi-place relational structures (i.e., no $T_3$ units are active), then DORA links one of the mapped pairs (via a $T_3$) unit, forming a functional multi-place relation. Functionally, learning can be defined as follows:

$$E_i = \begin{cases} P^{R}_{a,b} \leftarrow f_L(M(a^D, b^R)), & \#T_2 \\ R^{R}_{i,n} \leftarrow f_L(M(P^D_1(a), P^R_1(b)) \ldots M(P^D_n(c), P^R_n(d))), & \#T_3 \end{cases} \quad (B.3)$$

where $E_i$ is a learned representation, $P^I_i(a)$ is a single-place predicate $i$, in set $J = \{D = \text{driver}, R = \text{recipient}, M = \text{LTM}\}$ that takes the argument $a$. Lowercase letters $a$, $b$, and $c$ indicate objects, $a \cap b$ indicates the intersection of the features of $a$ and $b$, $R^I_i$ is a relational structure $i$ (consisting of linked predicate argument pairs; see Eq. B.4, directly below) of arity $n$ in set $J$, $T_k$ is an active token unit, and $f_L$ is the learning function.

The relational structure $R^{R}_{i,n}$ is instantiated in DORA functionally as:

$$R^{R}_{i,n} = [P^R_1(a) \oplus \ldots P^R_n(c)] \quad (B.4)$$

where $\oplus$ is a linking operator, and all $\oplus$ in a single $R_i$ are instantiated as a $T_3$ unit linking together (i.e., conjoining) predicate-argument pairs $P^R_1(a) \ldots P^R_n(c)$.

B.2.4 Refinement

If DORA has learned mapping connections between representations in driver and recipient, then DORA can also learn a refined (or schematized) representation consisting of the featural intersection of the mapped representations. Refinement is defined as follows:

$$R^{R,M}_{i,n} \leftarrow f_R(M(R^D_n, R^R_n)) \quad (B.5)$$
where \( R'_{m,n} \) (defined directly below) is a refined relational structure of arity \( n \) in set \( M \), and \( f_R \) is the refinement function. The refined structure \( R'_{m,n} \) is then:

\[
R'_{m,n} = [P^M_{D1 \cap R1}(a) \oplus \ldots P^M_{Dn \cap Rn}(b)]
\]

where \( P^D_i \cap P^R_j \) is a single-place predicate composed of the featural intersection of mapped predicates \( P^D_i \) and \( P^R_j \).

### B.2.5 Relational Generalization

If DORA has learned mapping connections between representations in driver and recipient, then DORA can also perform relational generalization, inferring structure from the driver about items in the recipient. Generalization in DORA follows the standard copy-with-substitution-and-generalization format common in models of relational reasoning can be defined as follows:

\[
G^R_i \leftarrow f_G(M(R^D_j, R^R_j) \land \sim M(R^D_k))
\]

where \( G_i \) is a generalized structure (see Eq. B.8, directly below), \( \sim M(A) \) is an unmapped structure \( A \), and \( f_G \) is the generalization function. The generalized structure \( G^R_i \) is then:

\[
G^R_i = [R^R_j \land R^R_k]
\]

where \( R^R_j \) is the mapped relational structure from Eq. B.7, and \( R^R_k \) is generalized relational information in the recipient that matches the unmapped \( R^D_k \) from Eq. A7.

### B.3 Processing in DORA

DORA’s operation is outlined in pseudocode in Figure B.1. The details of each step, along with the relevant equations and parameter values, are provided in the subsections that follow. DORA is very robust to the values of the parameters (see Doumas et al., 2008). For equations in this section, we use the variable \( a \) to denote a unit’s activation, \( n \) its (net) input, and \( w_{ij} \) to denote the connection from unit \( i \) to unit \( j \).

An analog, \( F \) (selected at random, or based on the current game screen), enters the driver. Network activations are initialized to 0. Either (a) the firing order of propositions in \( F \) is random (however, see Hummel and Holyoak 2003, for a detailed description of how a system like DORA can set its own firing order according to the constraints of pragmatic centrality and text coherence), or (b) a roughly random firing order is instantiated by passing a top
### Processing in DORA

<table>
<thead>
<tr>
<th>Processing step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items $F$ in $D$</td>
<td>$w(\tau, \tau, \sigma_i) \leftarrow L(E(\tau, \tau))$</td>
</tr>
<tr>
<td>For operation in {retrieving, mapping, predicate learning, refining, generalizing}:</td>
<td></td>
</tr>
<tr>
<td>For $i \in F$</td>
<td></td>
</tr>
<tr>
<td>Until $Y_i &gt; \theta_i$</td>
<td>${a_j, n_j, Y_i, I} \leftarrow f(a_j, a_k, w_{jk}, Y_j, I)$</td>
</tr>
<tr>
<td>When mapping :</td>
<td>$\Delta h_{jk} \leftarrow a_j a_k$</td>
</tr>
<tr>
<td>If $\exists (m_{D,R})$ :</td>
<td></td>
</tr>
<tr>
<td>When predicate learning :</td>
<td>$(a_{\tau,j,R,L} = 1) \leftarrow (\Sigma a_{\tau,y,R,L-1} - \Sigma a_{\tau,y,j,R,L}) &gt; \theta_p$</td>
</tr>
<tr>
<td>When refining :</td>
<td>$(a_{\tau,x,LTM,L} = 1) \leftarrow \exists (a_{\tau,y,D,L} &gt; \theta_p \cap m_{\tau,y,R})$</td>
</tr>
<tr>
<td>When generalizing :</td>
<td>$(a_{\tau,x,R,L} = 1) \leftarrow a_{\tau,y,D,L} \cap \exists m_{\tau,y,R}$</td>
</tr>
<tr>
<td>$\Delta w_{ij} \leftarrow a_i(a_j - w_{ij}) \gamma$</td>
<td></td>
</tr>
<tr>
<td>End If</td>
<td></td>
</tr>
<tr>
<td>End Until</td>
<td></td>
</tr>
<tr>
<td>End For</td>
<td></td>
</tr>
<tr>
<td>When retrieving :</td>
<td>$g_R \leftarrow p\left(\frac{A(g)}{\sum_i A(h_i)}\right), h \neq g$</td>
</tr>
<tr>
<td>When mapping :</td>
<td>$\Delta m_{jk} \leftarrow \eta(1.1 - h_{jk}) h_{jk}$</td>
</tr>
<tr>
<td>End For</td>
<td></td>
</tr>
</tbody>
</table>

$F$ is the set of units at the highest token layer for any set of connected units.

When unit $i$’s yoked inhibitor has not fired.

(2.1) Update inputs and activation of units in the network.

Update weights via Hebbian learning.

--

**Figure B.1**: Pseudocode of processing in DORA.
down input signal to all units \( i \) in the highest layer of \( D \) sampled from a uniform distribution with values between 0 and 0.4. DORA performs similarity and relative magnitude calculation through the relational invariance circuit, then runs retrieval from LTM, analogical mapping, and comparison-based unsupervised learning (predicate learning, refinement, and (relational) generalization). Currently, the order of operations of these routines is set to the order: retrieval, mapping, learning (predicate learning, refinement, and generalization).

Relational invariance circuit

The operation of the relational invariance circuit is described in the main text. The relational invariance circuit runs when two or more items (T1 units) are present in the driver and those items are connected to similar magnitude representations (e.g., pixels, etc.). It is inhibited to inactivity when two or more circuit output features are active above threshold (\( = .9 \)) when both T1 units are active. As noted in the main text, we presently make no strong commitment to an account of dimensional salience (as discussed in, e.g., Spelke & Kinzler, 2007). As such, two T1 units in the driver, both encoding an object or both encoding a predicate, are selected at random, and when those items are connected to multiple dimensions, a dimensional encoding becomes active at random (activation of features encoding that magnitude for the two T1 units are clamped to 1). The choice of two T1 units follows work on the WM capacity of children (e.g., Halford, Wilson, & Phillips, 1998). As described in the main text, proxy units connected to the active driver T1 units update their activation and input by Eqs. 3.4 and 3.5 respectively. \( E \) unit activation is updated by Eq. 3.7. Feature units connected to \( E \) units update their input by Eq. 3.8, and their activation by Eq. 3.9. Weights between feature units and active T1 units are updated by Eq. 3.10.

Main DORA operations

Repeat the following until each token unit \( i \) in the highest layer of \( F \) has fired three times if mapping, or once, otherwise (each token unit at the highest layer firing is referred to as the phase set). If a firing order has been set, select the current unit \( i \) in the firing order and set \( a_i \) to 1.0. Otherwise, pass a top-down input (\( n = \text{unif}(0, 0.4) \)) to token units in the highest layer of \( F \).

Update inputs and activations of network units
B.3. Processing in DORA

**Update mode of all T3 units in driver and recipient**  
T3 units in all propositions operate in one of three modes: Parent, child, and neutral (Hummel & Holyoak, 1997, 2003). T3 mode is important for representing higher-order relations \( R1(x, R2(y, z)) \); Hummel and Holyoak (1997). As detailed below, higher-order relations are represented in DORA such that if one proposition takes another as an argument, the T3 unit of the lower-order proposition serves as the object of an T2 unit for the higher-order proposition (i.e., the lower-order T3 unit is downwardly connected to the T2 unit, as a T1 unit would be), and the T3 unit represented the lower-order proposition operates in child mode. By contrast, when a T3 unit is not acting as the argument of another proposition, it operates in parent mode. The mode \( m_i \) of T3 unit \( i \) is updated by the equation:

\[
m_i = \begin{cases} 
    Parent(1), & T_{2\above} < T_{2\below} \\
    Child(-1), & T_{2\above} > T_{2\below} \\
    Neutral(0), & \text{otherwise}
\end{cases}
\]  

(B.9)

where, \( T_{2\above} \) is the summed input from all T2 units to which \( i \) is upwardly connected (i.e., relative to which \( i \) serves as an argument), and \( T_{2\below} \) is the summed input from all T2 units to which \( i \) is downwardly connected. In the current simulations, T3 mode did not have to change their mode. We include this step here solely for the purposes of completeness (Hummel & Holyoak, 1997, 2003).

**Update input to all units in the network**  
Update input to all token units in driver: Token units in the driver update their input by the equation:

\[
n_i = \sum_j a_{ji} G - \sum_k a_k - s \sum_m 3a_m - 10I_i
\]  

(B.10)

where \( j \) are all units above unit \( i \) (i.e., T3 units for T2 units, T2 units for T1 units), \( G \) is a gain parameter attached to the weight between the T2 and its T1 units (T1 units learned via DORA's comparison-based predication algorithm (see below) have \( G = 2 \) and all other T1 units have \( G = 1 \)), \( k \) is all units in the driver in the same layer as \( i \) (for T1 units, \( k \) is only those T1 units and T3 units currently in child mode not connected to the same T2 as unit \( i \); see step 2.1.1), \( m \) are T1 units that are connected to the same T2 (or T2 units) as \( i \), and \( I_i \) is the activation of the T1 inhibitor yoked to \( i \). When DORA is operating in binding-by-asynchrony mode, \( s = 1 \); when it is operating in binding-by-synchrony mode (i.e., like LISA), \( s = 0 \).

**Update input to feature units:** Feature units update their input as:

\[
n_i = \sum_{j \in S \in (D,R)} a_{ji}
\]  

(B.11)
where \( j \) is all T1 units in \( S \), which is the set of propositions in driver, \( D \), and recipient \( R \), and \( w_{ij} \) is the weight between T1 unit \( j \) and feature unit \( i \). In LISA [see][hummel1997distributed], when multiple propositions are in the driver simultaneously it ignores the features of the arguments. This convention follows from the assumption that people default to thinking of single propositions at a time, and the only reason to consider multiple propositions simultaneously is to consider structural constraints (e.g., Hummel & Holyoak, 2003; Medin et al., 1993), and has been adopted in DORA. When there are multiple positions in the driver, input to semantics is taken only from T1 unit \( j \) acting as roles (i.e., with \( mode = 1 \); see below).

**Update input to token units in recipient and LTM:** Input to all token units in recipient and LTM are not updated for the first 5 iterations after the global or local inhibitor fires. Token units in recipient and token units in LTM during retrieval update their input by the equation:

\[
n_i = \sum_j a_j w_{ij} + SEM_i + M_i - \sum_k a_k - s \sum_m 3a_m - \sum_n a_n - \Gamma_G - \Gamma_L
\]  

(B.12)

where \( j \) is any units above token unit \( i \) (i.e., T3 units for T2 units, T2 units for T1 units; input from \( j \) is only included on phase sets beyond the first), \( SEM_i \) is the feature input to unit \( i \) if unit \( i \) is a PO, and 0 otherwise, \( M_i \) is the mapping input to unit \( i \), \( k \) is all units in either recipient (if unit \( i \) is in recipient) or LTM (if unit \( i \) is in LTM) in the same layer as \( i \) (for T1 units, \( k \) is only those T1 units and T3 units currently in child mode not connected to the same T2 as unit \( i \)), \( m \) is T1 units connected to the same T2 as \( i \) (or 0 for non-T1 units), \( n \) is units above unit \( i \) to which unit \( i \) is not connected, \( \Gamma_G \) is the activation of the global inhibitor (see below), and \( \Gamma_L \) is the activation of the local inhibitor in the same layer as or any layer above \( i \). When DORA is operating in binding-by-asynchrony mode, \( s = 1 \); when it is operating in binding-by-synchrony mode (i.e., like LISA), \( s = 0 \). \( SEM_i \), the feature input to \( i \), is calculated as:

\[
SEM_i = \frac{\sum_j a_j w_{ij}}{1 + num(j)}
\]  

(B.13)

where \( j \) are feature units, \( w_{ij} \) is the weight between feature unit \( j \) and T1 unit \( i \), and \( num(j) \) is the total number of feature units \( i \) is connected to with a weight above \( \theta (=0.1) \). \( M_i \) is the mapping input to \( i \):

\[
M_i = \sum_j a_j (3w_{ij} - \max(Map(i)) - \max(Map(j)))
\]  

(B.14)
where \( j \) are token units of the same type as \( i \) in driver (e.g., if \( i \) is a T2 unit, \( j \) is all T2 units in driver), \( \text{Max}(\text{Map}(i)) \) is the highest of all unit \( i \)'s mapping connections, and \( \text{Max}(\text{Map}(j)) \) is the highest of all unit \( j \)'s mapping connections. As a result of Eq. B.14, active token units in driver will excite any recipient units of the same type to which they map and inhibit all recipient units of the same type to which they do not map.

**Update input to the yoked inhibitors** Every token unit is yoked to an inhibitor unit \( i \). T2 and T3 inhibitors are yoked only to their corresponding T2. T1 inhibitors are yoked both to their corresponding T1 and all T2 units in the same analog. Inhibitors integrate input over time as:

\[
\begin{align*}
n_i^{(t+1)} &= n_i^{(t)} + a_j + \sum_k a_k \\
\end{align*}
\]

where \( t \) refers to the current iteration, \( j \) is the token unit yoked to inhibitor unit \( i \), and \( k \) is any T2 units if \( j \) is a T1, and 0 otherwise. Inhibitor units become active \( (a_i = 1) \) when \( n_i \) is greater than the activation threshold \( (= 220 \text{ for } \text{T2 and } \text{T1 units}; 220 + n \text{ for } \text{T3 units}—\text{where } n \text{ is the number of T2 units the T3 units is connected to}) \). All T1 and T2 inhibitors become refreshed \( a_i = 0 \) and \( n_i = 0 \) when the global inhibitor \( (\Gamma^G; \text{described below}) \) fires.

**Update the local and global inhibitors** The local inhibitors, \( \Gamma_L \), are inhibitory units connected to all units in a single layer of LTM (i.e., there is a local inhibitor for T1 units, another for T2 units). The local inhibitor is potentiated \( (P(\Gamma_L) = 1) \) when a driver unit in \( \Gamma_L \)'s layer is active, is inhibited to inactivity \( (\Gamma_L = 0) \) by any driver unit in its layer with activation above \( \Theta_L = 0.5 \), and becomes active \( (\Gamma_L = 10) \) when no token unit in its layer has an activity above \( \Theta_L \). A firing local inhibitor sets the activation and potentiation of all other local inhibitors below and including itself to 0. The global inhibitor, \( \Gamma_G \), is potentiated \( (P(\Gamma_G) = 1) \) when any driver units are active, and is inhibited to inactivity \( (\Gamma_G = 0) \) by any driver unit in its layer with activation above \( \Theta_G = 0.5 \), and becomes active \( (\Gamma_G = 10) \) when no T1 in its layer has an activity above \( \Theta_G \). The global inhibitor sets activation and potentiation of all other local inhibitors to 0.

**Update activations of all units in the network** All token units in DORA update their activation by the leaky integrator function:

\[
\Delta a_i = \gamma n_i (1.1 - a_i) - \delta a_i |^0_1
\]

where \( \Delta a_i \) is the change in activation of unit \( i \), \( \gamma = 0.3 \) is a growth parameter, \( n_i \) is the net input to unit \( i \), and \( \delta = 0.1 \) is a decay parameter. Activation of all token units \( i \), is hard limited to between 0 and 1 inclusive.
Feature units update their activation by the equation:

\[ a_i = \frac{n_i}{\max(n_j)} \]  

(B.17)

where \( a_i \) is the activation of feature unit \( i \), \( n_i \) is the net input to feature unit \( i \), and \( \max(n_j) \) is the maximum input to any feature unit. There is physiological evidence for divisive normalization in the feline visual system (e.g., Bonds, 1989; Heeger, 1992) and psychophysical evidence for divisive normalization in human vision (e.g. Foley, 1994; Thomas & Olzak, 1997).

Token unit inhibitors, \( i \), update their activations according to a threshold function:

\[ a_i = \begin{cases} 
1, & n_i > \Theta_{IN} \\
0, & \text{otherwise} 
\end{cases} \]  

(B.18)

where \( \Theta_{IN} = 220 \) for T1 and T2 units and \( 220 * n \) for T3 units (where \( n \) is the number of T2 units to which that T3 unit is connected).

Update mapping hypotheses  If mapping is licensed, DORA learns mapping hypotheses between all token units in driver and token units of the same type in recipient (i.e., between T3 units, between T2 units and between T1 units in the same mode [described below]). Mapping hypotheses initialize to zero at the beginning of a phase set. The mapping hypothesis between a driver unit and a recipient unit of the same type is updated by the equation:

\[ \Delta h_{ij} = a'_t a_j \]  

(B.19)

where \( a'_t \) is the activation of driver unit \( i \) at time \( t \).
Comparison-based unsupervised learning  If licensed, DORA will perform comparison-based-learning (CBL). CBL is unsupervised. In the current version of the model, learning is licensed whenever 70% of driver token units map to recipient items (this 70% criterion is arbitrary, and in practice either 0% or 100% of the units nearly always map).

Predicate and relation learning  During predicate and relation learning, DORA recruits (and clamps the activation of) token units in the recipient to respond to patterns in firing in adjacent layers. The recruitment procedure is a simplified version of ART (Carpenter & Grossberg, 1990). Each layer of token units \( i \) is connected to a clamping unit \( C_i \), which 10 iterations after any inhibitor unit has fired, is activated by the equation:

\[
C_i = \begin{cases} 
1, & \left( \sum_j \max(a_j) - \sum_i \max(a_i) \right) \geq \theta_c \\
0, & \text{otherwise} 
\end{cases} \tag{B.20}
\]

where \( a_j \) is the activation of unclamped token units in the layer below \( i \) for T2 and T3 units, \( a_i \) is the activation of unclamped token units in layer \( i \), \( \max(a_j) \) is the maximum activation of a unit in layer \( j \), and \( \theta_c \) is a threshold (=0.6). \( C_i \) for the T1 layer is equal to \( C_i \) for the T2 layer.

An active \( C_i \) in the recipient sends an input, \( (p_j = 1.0) \), to a randomly selected token unit, \( j \) (where \( j \) is not connected to units in other layers), in layer \( i \) \( (p + k = 0 \text{ for all units } k \neq j) \). Token units are clamped by the equation:

\[
c_j = \begin{cases} 
1, & p_i - 3\sum_a a_k > 0 \\
0, & \text{otherwise} 
\end{cases} \tag{B.21}
\]

where \( c_j \) is the clamped activation of unit \( j \) in layer \( i \), and \( a_k \) is the activation of all clamped token units in the same layer as \( j \), where \( k \neq j \), if \( j \) is in T1, and all token units in the same layer as \( j \), where \( k \neq j \), otherwise. Unit \( j \) remains clamped until \( \Gamma_L \) fires and \( j \) is inhibited to inactivity. If the recruited token is in T1 its mode is set to 1 (marking it as a learned representation; although the idea of units firing in modes sounds nonneural, Hummel and Holyoak 1997 described how it can be accomplished with two or more auxiliary nodes with multiplicative synapses) and connections between the recruited token unit and all active features update by the equation:

\[
\Delta w_{ij} = a_i (a_j - w_{ij}) \gamma \tag{B.22}
\]
where $\Delta w_{ij}$ is the change in weight between the new T1 unit $i$, and feature unit $j$, $a_i$ and $a_j$ are the activations of $i$ and $j$, respectively, and $\gamma$ is a growth rate parameter. Additionally, connections between corresponding token units (i.e., between T3 and T2, or T2 and T1 units) are also updated by Eq. B.22, where $i$ are recipient token units in layers adjacent to recruited unit $j$. When the phase set ends, connection weights between a T2 or T3 unit $i$ and any token unit in the adjacent lower layer $j$ (i.e., $j$ is a T2 unit when $i$ is a T3 unit, and $j$ is a T1 unit when $i$ is a T2 unit), are updated by the equation:

$$w_{ij} = \begin{cases} w_{ij}, & \sum_k w_{ik} \geq 2 \\ 0, & otherwise \end{cases}$$  \hspace{1cm} (B.23)$$

where $k$ is all other units, including $j$, in the same layer as $j$. This operation removes weights to redundant tokens that do not conjunct two or more units at a lower layer.

**Refinement Learning** During refinement, DORA infers token units in the LTM that match active tokens in the driver. Specifically, DORA infers a token unit in the LTM in response to any mapped token unit in the driver. If unit $j$ in the driver maps to nothing in the LTM, then when $j$ fires, it will send a global inhibitory signal to all units in the LTM (Eq. B.14). This uniform inhibition, unaccompanied by any excitation in recipient, is a signal that DORA exploits to infer a unit of the same type (i.e., T1, T2, T3) in LTM. Inferred T1 units in the LTM have the same mode as the active T1 in driver. The activation of each inferred unit in the LTM is set to 1. DORA learns connections between corresponding active tokens in the LTM (i.e., between T3 and T2 units, and between T2 and T1 units) by Eq. B.22 (where unit $j$ is the newly inferred token unit, and unit $i$ is any other active token unit). To keep DORA’s representations manageable (and decrease the runtime of the simulations), at the end of the phase set, we discard any connections between feature units and T1 units whose weights are less than 0.1. When the phase set ends, connection weights between any T2 or T3 unit $i$ and token units at a lower adjacent layer $j$ to which $i$ has connections are updated by Eq. B.23.

**Relational generalization** The relational generalization algorithm is adopted from Hummel and Holyoak (2003). As detailed in Eq. B.14, when a token unit $j$ in driver is active, it will produce a global inhibitory signal to all recipient units to which it does not map. A uniform inhibition in recipient signals DORA to activate a unit of the same type (i.e., T1, T2, T3) in recipient as the active token unit in driver. DORA learns connections between corresponding active tokens in the LTM (i.e., between T3 and T2 units, and between T2 and T1 units) by the simple Hebbian learning rule in Eq. B.22 (where unit $j$ is the newly active token unit, and unit
$i$ is the other active token unit). Connections between T1 units and feature units are updated by Eq. B.22. When the phase set ends, connection weights between any T2 or T3 unit $i$ and any token units in an adjacent lower layer $j$ to which $i$ has connections are updated by Eq. B.23.

**Retrieval**

DORA uses a variant of the retrieval routine described in Hummel and Holyoak (1997). During retrieval units in the driver fire as described above for one phase set. Units in the LTM become active. After one phase set representations are retrieved from LTM into the recipient probabilistically using the Luce choice axiom:

$$L_i = \frac{R_i}{\sum_j R_j}$$  

(B.24)

where $L_i$ is the probability that T3 unit $i$ will be retrieved into working memory, $R_i$ is the maximum activation T3 unit $i$ reached during the retrieval phase set and $j$ are all other T3 units in LTM. If a T3 unit is retrieved from LTM, the entire structure of tokens (i.e., connected T1 ... T3 units) are retrieved into recipient.

**Update mapping connections**

If DORA is mapping, mapping connections are updated at the end of each phase set. First, all mapping hypotheses are normalized by the equation:

$$h_{ij} = \left( \frac{h_{ij}}{\text{MAX}(h_{i}, h_{j})} \right) - \text{MAX}(h_{kl})$$  

(B.25)

where, $h_{ij}$ is the mapping hypothesis between units $i$ and $j$, $\text{MAX}(h_{i}, h_{j})$ is the largest hypothesis involving either unit $i$ or unit $j$, and $\text{MAX}(h_{kl})$ is the largest mapping hypothesis where either $k = i$ and $l \neq j$, or $l = j$ and $k \neq i$. That is, each mapping hypothesis is normalized divisively: Each mapping hypothesis, $h_{ij}$ between units $i$ and $j$, is divided by the largest hypothesis involving either unit $i$ or $j$. Next each mapping hypothesis is normalized subtractively: The value of the largest hypothesis involving either $i$ or $j$ (not including $h_{ij}$ itself) is subtracted from $h_{ij}$. The divisive normalization keeps the mapping hypotheses bounded between zero and one, and the subtractive normalization implements the one-to-one mapping constraint by forcing mapping hypotheses involving the same $i$ or $j$ to compete with one another. Finally, the mapping weights between each unit in driver and the token units in recipient of the same type are updated by the equation:

$$\Delta w_{ij} = \eta (1.1 - w_{ij}) h_{ij}^0$$  

(B.26)
where $\Delta w_{ij}$ is the change in the mapping connection weight between driver unit $i$ and recipient unit $j$, $h_{ij}$ is the mapping hypothesis between unit $i$ and unit $j$, $\eta$ is a growth parameter, and $\Delta w_{ij}$ is truncated for values below 0 and above 1. After each phase set, mapping hypotheses are reset to 0. The mapping process continues for three phase sets.

**Learning the relational invariance circuit**

As described in the main text, the relational invariance circuit consists of three layers of nodes. At the top layer, proxy units are connected to individual T1 units in the driver. The next layer, $E$, consists of four nodes and takes input from any active proxy units. At the bottom are feature units, initially randomly connected to nodes in $E$. Weights between units in $E$ and feature units are initialized to random numbers between 0 and 0.9, and lateral connections for $E$ are set to -1. Connections between units in $E$ and feature units are updated by Eq. 3.10 in the main text.

**Higher-order relations**

Although they are not necessary for the current simulations, for the purposes of completeness it is important to note that DORA easily represents higher-order relations (i.e., relations between relations; see Doumas et al., 2008; Hummel & Holyoak, 1997). In short, when a proposition takes another proposition as an argument, the T3 unit of the lower-order proposition serves as the object of an T2 unit for the higher-order proposition. For example, in the higher-order relation greater (distance (a, b), distance (c, d)), the T3 unit of the proposition distance (a, b) serves as the argument of the more role of the higher-order greater relation, and the T3 unit of the proposition distance (c, d) serves as the argument of the less role of the higher-order greater relation. When a T3 unit serves as the object of an T2 unit, it operates in child mode (see above). The modes of T3 units change as a function of whether they are receiving top-down input. A T3 unit receiving top-down input from an T2 unit (i.e., when the T3 unit is serving as the argument of that T2 unit) will operate in child mode, while a T3 unit not receiving input from any T2 units will operate in parent mode.
Appendix C

Simulation Details Chapter 3

C.1 DORA: Learning representations from screens

We used DORA to simulate learning structured representations from screen shots from the game Breakout. This simulation aims to mirror what happens when a child (or adult) learns from experience in an unsupervised manner (without a teacher or guide). While we describe the results in terms of DORA learning to play Breakout and generalizing to Pong, but results were the same when run in the other direction (i.e., train on Pong and generalize to Breakout; Supplemental results, Figure S1).

For Simulation 2, screens were generated from Breakout during 250 games with random move selection. Each screen from each game was processed with the visual pre-processor that identified objects and returned the raw pixel values as features of those objects. When learning in the world, objects have several extraneous properties. To mirror this point, after visual pre-processing, each object was also attached to a set of 100 additional features selected randomly from a set of 10000 features. These additional features were included to act as noise, and to make learning more realistic. (Without these noise features, DORA learned exactly as described here, only more quickly.)

DORA learned from object representations in an unsupervised manner. On each learning trial, DORA selected one pair of objects from a screen at random. DORA attempted to characterize any relations that existed between the objects using any relations it has previously learned (initially, it had learned nothing, and so nothing was returned) by selecting a dimension at random and running the two objects through the relational invariance circuit (described above) over that dimension. If the features returned matched anything in LTM (e.g., “more” and “less” “x”), then DORA used that representation from LTM to characterize the current objects. DORA then ran (or attempted to run) retrieval from LTM, the relational invariance circuit, mapping, and representation learning (see above). Learned representations were stored in LTM. We placed the constraint on DORA’s retrieval algorithm such that more recently learned items were favoured for retrieval. Specifically, with probability .6, DORA attempted to retrieve from the last 100 representations that it had learned. This constraint followed our assumption that items learned more recently are more salient and more likely to be available for retrieval.
C.1. DORA: Learning representations from screens

The process was identical for Simulation 3, except that instead of screens from Breakout, we used the first 300 images from the CLVR dataset for representation learning. In simulation 4, we had two ablated versions of the model: In the first ablated model (A1), we ablated the inhibitory connections from the onset of the simulation; in the second ablated model (A2), we ablated the inhibitory connections after the model had learned to play Breakout. Representation learning for both models was as in simulation 1.

C.2 DORA: Q-learning for Game Play

For Simulations 2, 3, and 4, for a given screen, DORA used the representations it had previously learned to represent the relations between objects on that screen and the previous screen. That is, for any pair of objects, if DORA had learned a representation that characterized the relation between the two objects (in LTM and as measured by the relational invariance circuit), DORA used that representation to characterize the objects.

The relations were then used to form a table of encountered relational states, and Q-learning (Watkins, 1989) was used to learn the approximate action-value function for Breakout. We used a rule length constraint of two relations per state, reflecting the simplicity of the game and the WM capacity exhibited by humans Logie et al. (2020). We trained DORA decreasing the learning rate linearly from 0.1 to 0.05 and the exploration rate linearly from 0.1 to 0.01 throughout the training session. We saved the version of the table that yielded the maximum score during the session.

C.3 Deep Q-Network

A Deep Q-Network (DQN; Mnih et al., 2015) was trained to play Breakout and Pong. The raw $210 \times 160$ frames were pre-processed by first converting their RGB representation to grayscale and down-sampling it to a $105 \times 80$ image. We stacked the last 4 consecutive frames to form the input each state.

The input to the neural network was the $105 \times 80 \times 4$ pre-processed state. The first hidden convolutional layer applied 16 filters of size $8 \times 8$ with stride 4 with a relu activation function. The second hidden convolutional layer applied 32 filters of size $4 \times 4$ with stride 2 with a relu activation function. The third hidden layer was fully connected of size 256 with a relu activation function. The output layer was fully connected with size 6 and a linear activation function.

We implemented all the procedures of the DQN to improve training stability, in particular: (a) We used memory replay of size 1,000,000. (b) We used a target network which was updated every 10,000 learning iterations. (c) We fixed all positive rewards to be 1 and all negative rewards to be -1, leaving 0 rewards unchanged. (d) We clipped the error term for the update through the Huber loss.
C.3. Deep Q-Network

We also ran the same network using the input from the visual preprocessor described above.

C.4 Supervised Deep Neural Network

We trained a deep neural network (DNN) in a supervised manner to play Breakout and Pong and tested generalization between games. One network was trained using random frame skipping and the other with fixed frame skipping.

The inputs to the network were the output of the visual preprocessor described above. Specifically, the network took as input the x and y positions of the ball and player-controlled paddle, as well as the left paddle for Pong (left as zeros when playing Breakout). The input to the neural network was a vector of size 24 corresponding to the pre-processed last seen 4 frames. This was fed to three fully connected layers of size 100 each with a relu activation function. The output layer was fully connected with size 6 and a softmax activation function.

The criteria for training was the correct action to take in order to keep the agent-controlled paddle aligned with the ball. In Breakout if the ball was to the left of the paddle the correct action was “LEFT”, if the ball was to the right of the paddle the correct action was “RIGHT” and if the ball and the paddle were at the same level on the x-axis the correct action was “NOOP”. In Pong if the ball was higher than paddle the correct action was “RIGHT”, if the ball was lower than paddle the correct action was “LEFT” and if the ball and the paddle were at the same level on the y-axis the correct action was “NOOP”. This action was encoded as a one-hot vector (i.e., activation of 1 for the correct action and zero for all other actions).

C.5 Graph Neural Network

Graph networks (see Battaglia et al., 2018, for a review) are neural network models designed to approximate functions on graphs. A graph is a set of nodes, edges, and a global feature. The representation of the nodes, edges, and the global attribute encode feature information. A graph network takes as input a graph and returns a graph of the same size and shape, but with updated attributes.

Our graph net agent used a encode-process-decode architecture (Battaglia et al., 2018) where three different graph networks are arranged in series. The first graph net encodes the nodes, edges and global attributes independently, the second graph net performs three recurrent steps of “message passing” and the third graph net decodes the nodes, edges and global attributes independently.
C.5. Graph Neural Network

The graph agent takes in a graph-structured representation of the screen where each object corresponds to a node in the graph. In our simulations, the node representation corresponds to the position, area, color and velocity of the objects in the screen. In order to use the graph network as a reinforcement learning agent we set the number of edge attributes to the number of possible actions. In this way, our agent produces a vector of Q-values for each edge, corresponding to the valid actions in each game. To choose actions, the agent takes an argmax across all edges’ Q-values.

To train our agent we used a replay memory of size 50000. Before training we feed the replay memory with 1600 memories (i.e., tuples containing a state graph, action, edge, reward, next state graph, and a “done” variable). At each time step, we saved the current memory and sample a batch of 32 memories from the replay memory to train the agent. We used the Adam learning algorithm with a learning rate of 0.01 and default learning parameters.
Appendix D

Supplemental Results Chapter 3

D.1 Supplemental simulation 1: Pong to Breakout generalization results

As described in the main text, DORA learned representations from Pong games. The model learned to play Pong first, and then generalized to Breakout. Results in Fig. D.1.

Figure D.1: Results of simulations with DORA trained on Pong and generalizing to Breakout, with DORA learning representations from Pong. Error bars represent ±2 standard errors. (A) Performance of DORA on Breakout as an average of 100 test games. (B) Results of DORA playing Breakout after training on Pong as the score of the first game played and an average score of the first 100 games played. (C) Results of DORA when returning to play Pong, as an average score for the first 100 games played.

D.2 Supplemental data 1

Two human novices were trained on Breakout for 300 games, then transferred to playing Pong for 100 games, followed by moving back to Breakout for 100 games (these games were played in 2 hours session spread across 6 days; the last 50 games of Breakout and first 20 games of Pong were completed in the same session). Human players, of course, come into playing these games with a life of experience with the world, spatial relations, and other video
games, and bring this experience to bear on playing both games. As humans regularly engage in cross-domain generalization, we expect the participants to generalize between games. A comparison of these highly trained humans and DORA and the various DNNs tested in the main text appears in Figure D.2.

In addition to the two participants who played several hundred games of Breakout and Pong, we ran 8 additional participants in a simple transfer task. Participants either played Breakout for 50 minutes followed by playing Pong for 10 minutes (4 participants) or played Pong for 50 minutes followed by playing Breakout for 10 minutes (4 participants). We had players play to a time limit rather than a number of games, as a game of Pong takes roughly 4 times as long as a game of Breakout. The average score on the first game of Pong when played first was 6.25 vs. 14.0 when played after Breakout. The average score on the first game of Breakout when played first was 9.5 vs. when played after Pong. We analyzed the effects of order (whether a game was played first or after another game) on performance using a simple linear mixed effects model with Score predicted by order with participants as a random variable. Because the scores in Breakout and Pong are on different scales (Pong goes to 21, Breakout is (theoretically) unbounded) we normalized all scores by subtracting each score from the grand mean of scores on that game (e.g., each Pong score had the mean of score of all Pong games subtracted from it). We then compared the model using order to predict score (with participants as a random variable) to the null model. The full model explained significantly more variance than the null model (chi-square(1) = 4.07, p < 0.05), with scores in the transfer condition significantly higher than scores in the initial game condition.

Participants were run using the online javatari system (https://javatari.org/).
One noteworthy limitation of DORA’s gameplay is that it was slower to learn Breakout than the human players we tested. We suspect the reason for this limitation is that in most simulations, DORA, like the DNNs we ran for comparison, began as a tabula rasa with no understanding of anything at the beginning of learning. As a result, DORA spent much of its early experience in these simulations simply acquiring basic relational concepts such as left-of(). By contrast, most people start playing video games long after they have acquired such basic concepts. In other words, our ability to play a game such as Breakout, even for the first time, is already facilitated by an enormous amount of cross-domain transfer: People know what to look for (e.g., “where is the paddle relative to the ball?”) and how to represent the answer (“to the left of the ball”) even before starting to learn how to play the game. DORA, by contrast, had to learn these basic concepts while learning to play the game.

We argue that the reason DORA learned so much faster than the DNNs is that DORA was biased from the beginning to look for the right thing. Whereas DNNs search for representations that minimize the error in the input-output mapping of the task at hand, DORA looks for systematic relations that allow it to build a model of the task it is learning. Once it has learned this model, DORA is off and running, prepared to transfer its learning to new tasks, such as Pong. By contrast, the DNN is trying to be the best it can at exactly this one task; it is trying to memorize exactly what to do in response to every possible situation. In the end, the DNN will be a better Breakout player than DORA. But DORA, unlike the DNN, will be able to transfer its learning to other tasks, including but not limited to Pong.

We argue that people are more like DORA than a DNN. You and I will never beat a well-trained DNN at chess, or Go, or probably any other task on which a DNN has been adequately trained. But at the end of the day, we will be able to drive home, make dinner, put our children to bed, and have a glass of wine. All the DNN will know how to do is beat the next competitor. And more importantly, the DNN will likely be unable to learn how to perform these other tasks without forgetting how to play chess. A human is a general-purpose learning machine that exceeds at using what it already knows to bootstrap its learning of things it doesn’t already know. A deep net is more a one-trick pony (though possibly the best in the world at its one over-trained task).

D.3 Supplemental simulation 2: Inverse Breakout

We ran a simple simulation of this capacity using a modified version of Breakout. In this version, the rules were adjusted such that missing the ball was rewarded and hitting the ball was punished (i.e., points were scored when the ball went past the paddle, and a life was lost when the ball struck the paddle; essentially the reverse of the regular Breakout rules). We ran tested a version of the DORA model that had previously learned to play Breakout successfully (see simulation 2, main text). Unsurprisingly, initially the model followed the
D.3. Supplemental simulation 2: Inverse Breakout

previously successful strategy of following the ball to contact it and send it towards the point-scoring bricks. However, upon contact with the paddle, the model was punished with a lost life. As noted in the main text, the model had previously learned that following the ball predicted reward (points), and that moving away from the ball predicted punishment (lost life). After three lost lives, DORA attempted to compare the representation of the current game to the representation it had previously learned from Breakout.

The current representation was that moving the paddle toward the ball resulted in punishment, or left-of (ball, paddle1) then left-of (paddle2, paddle1) $\rightarrow$ punishment signal. The previous representation of the game was that moving toward the ball resulted in reward and away from the ball resulted in punishment, or left-of (ball, paddle1) then left-of (paddle2, paddle1) $\rightarrow$ reward signal, and left-of (ball, paddle1) then right-of (paddle2, paddle1) $\rightarrow$ punishment signal. As described in the main text, DORA performed mapping and relational inference with these two representations. With P1: left-of(paddle2, paddle1) $\rightarrow$ punishment signal in the driver, and P2: left-of (paddle2, paddle1) $\rightarrow$ reward signal and P3: right-of (paddle2, paddle1) $\rightarrow$ punishment signal in the recipient, DORA mapped left-of (paddle2, paddle1) in P1 to left-of paddle2, paddle1) in P2, and punishment signal in P1 to punishment signal in P3. DORA then flipped the driver and recipient (P2 and P3 now in the driver and P1 in the recipient) and performed relational inference. During relational inference, it copied the unmapped reward-signal from P2 and right-of (paddle2, paddle1) from P3 into the recipient, thus inferring that right-of (paddle2, paddle1) predicts a reward signal. When adopting this strategy, moving away from the ball, DORA started scoring points (because the new task was so easy, we had to decide a point total to stop the game).
Appendix E

State Representations Chapter 4

The following tables present all the relational representations used in the games Pong and Demon Attack.
### Table E.1: Pong State Representation.

<table>
<thead>
<tr>
<th>Dim</th>
<th>Obj-1</th>
<th>Obj-2</th>
<th>Logical</th>
<th>Comparative</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>player,</td>
<td>ball,</td>
<td>more-x(player, ball)</td>
<td>x(player, ball)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, ball)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, ball)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player,</td>
<td>enemy,</td>
<td>more-x(player, enemy)</td>
<td>x(player, enemy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, enemy)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, enemy)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>ball,</td>
<td>enemy,</td>
<td>more-x(ball, enemy)</td>
<td>x(ball, enemy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(ball, enemy)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(ball, enemy)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player,</td>
<td>ball,</td>
<td>more-y(player, ball)</td>
<td>y(player, ball)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, ball)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, ball)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player,</td>
<td>enemy,</td>
<td>more-y(player, enemy)</td>
<td>y(player, enemy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, enemy)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, enemy)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>ball,</td>
<td>enemy,</td>
<td>more-y(ball, enemy)</td>
<td>x(ball, enemy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(ball, enemy)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(ball, enemy)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>ball,</td>
<td>ball_{t-1}</td>
<td>more-x(ball, ball_{t-1})</td>
<td>x(ball, ball_{t-1})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(ball, ball_{t-1})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(ball, ball_{t-1})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>ball,</td>
<td>ball_{t-1}</td>
<td>more-y(ball, ball_{t-1})</td>
<td>y(ball, ball_{t-1})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(ball, ball_{t-1})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(ball, ball_{t-1})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>enemy,</td>
<td>enemy_{t-1}</td>
<td>more-y(enemy, enemy_{t-1})</td>
<td>y(enemy, enemy_{t-1})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(enemy, enemy_{t-1})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(enemy, enemy_{t-1})</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>player,</td>
<td>ball,</td>
<td></td>
<td>in-contact(player, ball)</td>
<td></td>
</tr>
<tr>
<td>ball,</td>
<td>enemy,</td>
<td></td>
<td>in-contact(ball, enemy)</td>
<td></td>
</tr>
<tr>
<td>Dim</td>
<td>Obj-1</td>
<td>Obj-2</td>
<td>Logical</td>
<td>Comparative</td>
</tr>
<tr>
<td>-----</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>missile</td>
<td>more-x(player, e-missile)</td>
<td>x(player, e-missile)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-missile)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-missile)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>big-1</td>
<td>more-x(player, e-big-1)</td>
<td>x(player, e-big-1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-big-1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-big-1)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>big-2</td>
<td>more-x(player, e-big-2)</td>
<td>x(player, e-big-2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-big-2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-big-2)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>big-3</td>
<td>more-x(player, e-big-3)</td>
<td>x(player, e-big-3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-big-3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-big-3)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>small-1</td>
<td>more-x(player, e-small-1)</td>
<td>x(player, e-small-1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-small-1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-small-1)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>small-2</td>
<td>more-x(player, e-small-2)</td>
<td>x(player, e-small-2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-small-2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-small-2)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>small-3</td>
<td>more-x(player, e-small-3)</td>
<td>x(player, e-small-3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-small-3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-small-3)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>small-4</td>
<td>more-x(player, e-small-4)</td>
<td>x(player, e-small-4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-small-4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-small-4)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>small-5</td>
<td>more-x(player, e-small-5)</td>
<td>x(player, e-small-5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-small-5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-small-5)</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>player e-</td>
<td>small-6</td>
<td>more-x(player, e-small-6)</td>
<td>x(player, e-small-6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-x(player, e-small-6)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-x(player, e-small-6)</td>
<td></td>
</tr>
<tr>
<td>Dim</td>
<td>Obj-1</td>
<td>Obj-2</td>
<td>Logical</td>
<td>Comparative</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>------------</td>
<td>--------------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-missile</td>
<td>more-y(player, e-missile)</td>
<td>y(player, e-missile)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-missile)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-missile)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-big-1</td>
<td>more-y(player, e-big-1)</td>
<td>y(player, e-big-1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-big-1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-big-1)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-big-2</td>
<td>more-y(player, e-big-2)</td>
<td>y(player, e-big-2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-big-2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-big-2)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-big-3</td>
<td>more-y(player, e-big-3)</td>
<td>y(player, e-big-3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-big-3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-big-3)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-small-1</td>
<td>more-y(player, e-small-1)</td>
<td>y(player, e-small-1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-small-1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-small-1)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-small-2</td>
<td>more-y(player, e-small-2)</td>
<td>y(player, e-small-2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-small-2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-small-2)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-small-3</td>
<td>more-y(player, e-small-3)</td>
<td>y(player, e-small-3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-small-3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-small-3)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-small-4</td>
<td>more-y(player, e-small-4)</td>
<td>y(player, e-small-4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-small-4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-small-3)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-small-5</td>
<td>more-y(player, e-small-5)</td>
<td>y(player, e-small-5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-small-5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-small-5)</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>player</td>
<td>e-small-6</td>
<td>more-y(player, e-small-6)</td>
<td>y(player, e-small-6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>same-y(player, e-small-6)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less-y(player, e-small-6)</td>
<td></td>
</tr>
</tbody>
</table>


BIBLIOGRAPHY


Nematzadeh, A., Meylan, S. C., & Griffiths, T. L. (2017). Evaluating vector-space models of word representation, or, the unreasonable effectiveness of counting words near other words. In *Cogsci*.


doi: 10.18653/v1/d16-1264


