

## Analogies and cognitive modeling

## Overview

- Memory modeling revisited
- Examples of analogies
- Issues in modeling analogy
- Generalising memory models to analogy
- Copycat

## Memory modeling revisited

- Memory is of two main types
  - Recognition (given a cue and target, produces a Y/N answer)
    - Is her name "Mary"?
  - Recall (uses a cue to produce an item or vector as target)
    - What is her name?
  - Context can affect what target is recalled to what cue

## Memory cont.

	No context provided	Context provided
Cue and target provided	<b>Familiarity</b> Have you heard of a diploblast?	<b>Recognition</b> Did you use Pajek in the lab this week?
Cue only provided, target must be retrieved	<b>Free recall</b> Name a type of neural network	<b>Cued recall</b> Which analogy program was listed in the overview slide?

## Neural networks for modeling memory

- Recognition can be modeled with a vector memory
- Recall can be modeled with a matrix memory (a single layer, linear NN)
- Context cannot be modeled as a simple addition of extra inputs to a NN. It needs to be added as an orthogonal dimension (also called a sigma-pi network).

## What characterises a NN?

## What characterises a NN?

- No central control
- Distributed simple units
- Learning from patterns
- Generalising from patterns

## How powerful are NNs at generalising?

- What can NNs represent?  
I.e. what mappings can they map?
- What can they learn?  
I.e. what set of functions can they learn from what kinds of pattern sets?

## Generalisation problems

Kitten : cat :: joey : ?  
Person : house :: dog : ?  
Electron : atom :: planet : ?  
abc : abd :: efg : ?  
abc : abd :: 123 : ?

## Examples of analogies

- Letter styles
- Atom as solar system
- Electricity as water flow

## Issues in modeling analogy

- Analogy can be viewed as using the structural properties from one domain (the source) to understand or create structure in a second domain (the target).
- It requires equating two non-identical percepts at a higher level of abstraction

## When are two items "the same"?

- Is a chair facing left the same chair when seen from the other side?
- Are b and d the same?
- Are p and q the same?
- Are a a a a α A ∞ the same?

## Recognising similarity

- items can be recognised as similar to
  - Items: a and a
  - Groups: aaa
  - Properties: a and A
  - Sequences: abc and ijk

## Generalising memory models to analogy

- Neural network models of memory can provide a mechanism for mapping from one domain to another, but cannot create flexible representations for different levels of abstraction.
- “slippage” between non-identical percepts is required.

## Example of a microdomain

- Suppose the letter-string abc were changed to abd, how would you change the letter-string ijk in the same way?
- Or in shorthand  
abc : abd :: ijk : ?

## abc : abd :: ijk : ?

Rating	abc:abd :: ijk: ?	substitution	Rule	Slippage required
	ijl	k → l	third letter is replaced by its successor	c is replaced by third letter
	ijd	k → d	third letter is replaced by d	c is replaced by third letter
	ijk	none	c is replaced by d	

## abc : abd :: ijkk : ?

Rating	abc:abd :: ijkk: ?	substitution	Rule	Slippage required
			Replace the rightmost letter by its successor	
			Replace the rightmost group by its successor	

## General concepts from this microdomain

- Successor (or predecessor) is an idealised version of any non-identity relationship in a real-world domain s.a. parent-of, neighbour-of, friend-of, close-to, etc
- Successor group (eg abc) plays the role of any conceptual chunk based on such a relationship sa “family” “neighbourhood” “community” “region” etc
- Sameness and opposite are universal concepts
- Structures are linear strings of letters requiring a way of addressing the type/token distinction.

	Models of Analogical Reasoning: Copycat
	Scott Bolland

**Copycat**

- Copycat is a computational model of analogy making that was devised to capture and explore the mechanisms that underlie the flexible nature of high-level perception and analogy making
- Copycat explores the issues of context-dependent conceptual slippage, and the co-dependence of representation formation and mapping.

**Copycat**

- Copycat is a computational model that is capable of generating solutions to letter-string analogy problems.
- E.g., given *abc : abd*, the system is told to be a "copycat" and modify a target string such as *ijk* in the same way.
- Although the domain in which Copycat works seems rather restricted, it is surprisingly subtle. E.g. in the simple problem just stated, there are at least three solutions that both Copycat and human subjects produce : *ijl, ijd, ijk*

**Copycat**

- Example problem:


*abc : abd, kkjjii : ?*

How are the objects that are mapped in each case "playing the same role"?

**Copycat**

- Based on the premise that high-level perception (i.e. the representation of a situation) "emerges" as a statistical outcome of a plethora of independent processes that work in parallel, competing with and supporting each other.
- That is, there is no central executive telling the system what to do!

**Copycat**



- Similar to the emergent structure of a termites' mound – no individual termite has a blueprint of what the structure should look like – it simply emerges through the independent activities of thousands of agents

## Copycat

Copycat is comprised of 4 main constituents:

- WORKSPACE
- SLIPNET
- CODERACK
- TEMPERATURE

## Copycat – THE WORKSPACE

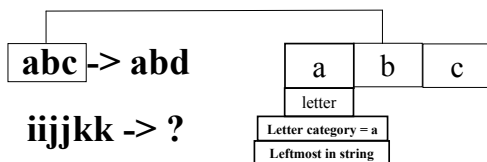
The Workspace is Copycat's "working memory" where the information that is perceived as being relevant to the current analogy is constructed.

This information includes the properties of the individual letters, relationships between letters, grouping of letters that share relationships and mappings between objects across the situations.

## Copycat – THE WORKSPACE

### DESCRIPTORS

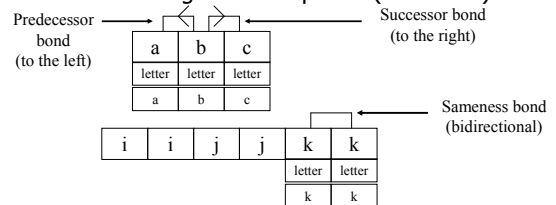
Descriptions denote perceived attributes of the objects in the workspace.



## Copycat – THE WORKSPACE

### BONDS

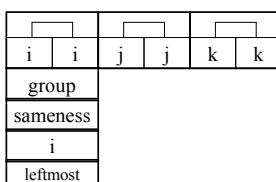
Bonds denote perceived relationships between adjacent objects in a string, based on their assigned descriptions (attributes).



## Copycat – THE WORKSPACE

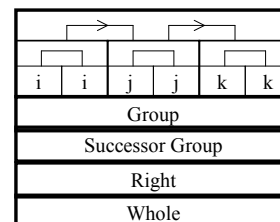
### GROUPS

Adjacent objects can be "chunked" together based on shared bonds. These new groups can be assigned descriptions themselves.



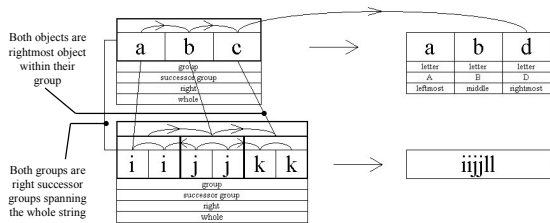
## Copycat – THE WORKSPACE

**Hierarchical Structures:** Based on their descriptors, bonds between adjacent groups can form, allowing for higher-level groups to form.



## Copycat – THE WORKSPACE

**Correspondences:** Based on similar descriptions, correspondences between situations can be made



## Copycat – THE WORKSPACE

**Rules:** To generate a letter string solution, Copycat firstly generates a rule that explains the transformation in the initial string.

This rule is in the form REPLACE the \_\_\_ of \_\_\_ by \_\_\_

e.g.  $abc \rightarrow abd =$  replace the *letter category* of the *rightmost letter*, by its *successor*

## Copycat – THE WORKSPACE

**Modified Rule:** To form a solution, the initial rule is modified, taking into account the conceptual slippages that occur between the domains. These slippages are present within the correspondences that have been built.

e.g.  $abc : abd, iijkk : ?$  - if a correspondence has been made between *c* (rightmost letter) and *kk* (rightmost group) the slippage *letter*->*group* will be made

This modifies the rule to "replace the *letter category* of the *rightmost group* by its *successor*" which yields the answer ***ijjll***

## Copycat – example solutions

$abc:abd, kji: ?$

■ **kjh**

**initial rule** = replace letter category of rightmost letter by successor

**modified rule** = replace letter category of rightmost letter by predecessor

This is because *abc* is perceived as a successor group to the right and *kji* is perceived as a predecessor group to the right, forming the slippage *successor*->*predecessor*.

## Copycat – example solutions

$abc:abd, kji: ?$

■ **lji**

**initial rule** = replace letter category of rightmost letter

by successor

**modified rule** = replace letter category of leftmost letter by successor

This is because *abc* is perceived as a successor group to the right and *kji* is perceived as a successor group to the left, forming the slippage *left*->*right*, which implies the slippage *rightmost* -> *leftmost*.

## Copycat – Parallel Terraced Scan

- Copycat's search strategy is called the "parallel terraced scan"
- This means that many alternatives are explored in parallel but are given a level of consideration based on the perceived level of promise
- This is similar to good solutions to the *k-armed bandit problem*.

## Copycat – Parallel Terraced Scan

- *k-armed bandit problem:*  
In the multi-armed bandit problem, a gambler in a rigged casino must decide which arm of K non-identical slot machines to play in a sequence of trials so as to maximize his reward.
- How do you approach this problem?

## Copycat – Parallel Terraced Scan

- *k-armed bandit problem:*  
This classical problem has received much attention because of the simple model it provides of the trade-off between exploration (trying out each arm to find the best one) and exploitation (playing the arm believed to give the best payoff).

## Copycat – Parallel Terraced Scan

- Can be thought of as similar to an ant colony's search for food:
- Most ant's roughly travel along the same path (I.e. there is one predominant solution being explored)
- However, each ant may stray from this path, with more ants following if the path pays off. The greater the payoff, the more ants will be recruited to explore the new path.

## The Workspace and the Parallel Terraced Scan

- The Workspace can be thought of as a construction site, with perceptual structures (bonds, groups etc), are built by the activities of a plethora of workers.
- Each worker is assigned a specific job (such as building a successor bond where it is applicable) and works independently of all other workers
- How many workers are assigned to a task is dependent on how relevant that task is deemed to be in the formation of a solution

## The Workspace and the Parallel Terraced Scan (cont.)

- As initially the program knows nothing about the problem, all workers are assigned jobs in a BOTTOM-UP fashion
- That is, they are asked to unbiasedly explore what concepts are present in the Workspace
- E.g. find any bond in the workspace between objects (remember *abc* can be thought to have right successor bonds or left predecessor bonds)

## The Workspace and the Parallel Terraced Scan (cont.)

- When concepts are discovered in the Workspace, TOP-DOWN pressure is exerted on the system.
- E.g. once a right successor bond has been discovered in the string "abc", more workers will be assigned to investigate where else they may lie. Furthermore, if they are discovered in many places, workers may be assigned the task of building successor groups etc.



## The Temperature

- Furthermore, the Temperature is used to implement a form of Simulated Annealing.
- Initially, the Temperature is quite high, reflecting the fact that little coherent structure has been built in the Workspace. This suggests that the activation of the Slipnet may not be reliable in the formation of a solution. Thus, at this point a high degree of randomness is injected into the decision making process.
- When, the Temperature falls (reflecting structural coherency), the decision making becomes more deterministic, using the activation of the Slipnet etc., as a guide for further exploration.

## Summary

- Copycat directly addresses two key aspects of analogical reasoning:
  1. Forming representations: selecting appropriate constituent features and relationships between them
  2. Mapping: equating the constituents in the source and target domains based on the roles they play.
- Copycat is composed of 4 parts:
  1. Workspace: trying out different representations of concepts based on context
  2. Slipnet: allowing "conceptual slippage" to equate non-identical concepts.
  3. Coderack: instructions for building and breaking apart concepts
  4. Temperature: balancing the exploration / exploitation constraints

## Summary (cont)

- The take home message of Copycat is that in a complex world (even Copycat's microworld) one cannot know what concepts will turn out to be relevant in a given situation ahead of time.
- The dilemma is to avoid totally open ended search strategies that entertain all possibilities equally seriously and also to avoid dogmatically closed-minded strategies that definitely rule out certain possibilities apriori
- Copycat provides an idea for a middle way, taking calculated risks all the time, but carefully controlling the degree of risk taking.