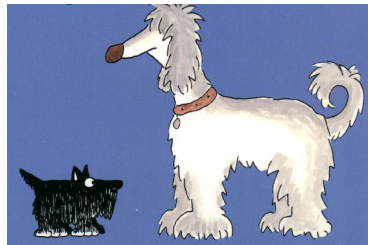
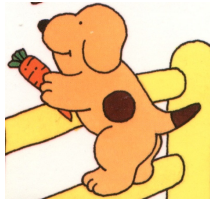
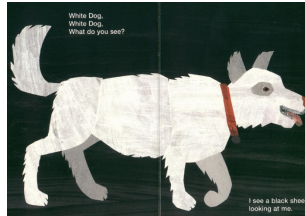


Analogy-Making

Consider the following cognitive activities

- Recognition:

- A child learns to recognize cats and dogs in books as well as in real life.



- People can recognize letters of the alphabet, e.g., 'A', in many different typefaces and handwriting styles.



– People can recognize styles of music:

- “That sounds like Mozart”
- “That’s a muzak version of ‘Hey Jude’”

– People can recognize abstract situations:

- A “Cinderella story”
- “Another Vietnam”
- “Monica-gate”
- “Shop-aholic”

• People make scientific analogies:

- “Biological competition is like economic competition” (Darwin)
- “The nuclear force is like the electromagnetic force” (Yukawa)
- “The computer is like the brain” (von Neumann)
- “The brain is like the computer” (Simon, Newell, etc.)

- People make unconscious analogies

Man: "I'm going shopping for a valentine for my wife."

Female colleague: "I did that yesterday."

- People make unconscious analogies

Newly married woman: "I often forget my new last name"

Man: "I have that trouble every January"

- People make unconscious analogies

Computer scientist: “I’m in artificial intelligence because it’s a mixture of psychology, philosophy, linguistics, and computer science”

Architect: “That’s the reason I’m in architecture”

What is common to all these examples?

Four Analogy-Making Systems

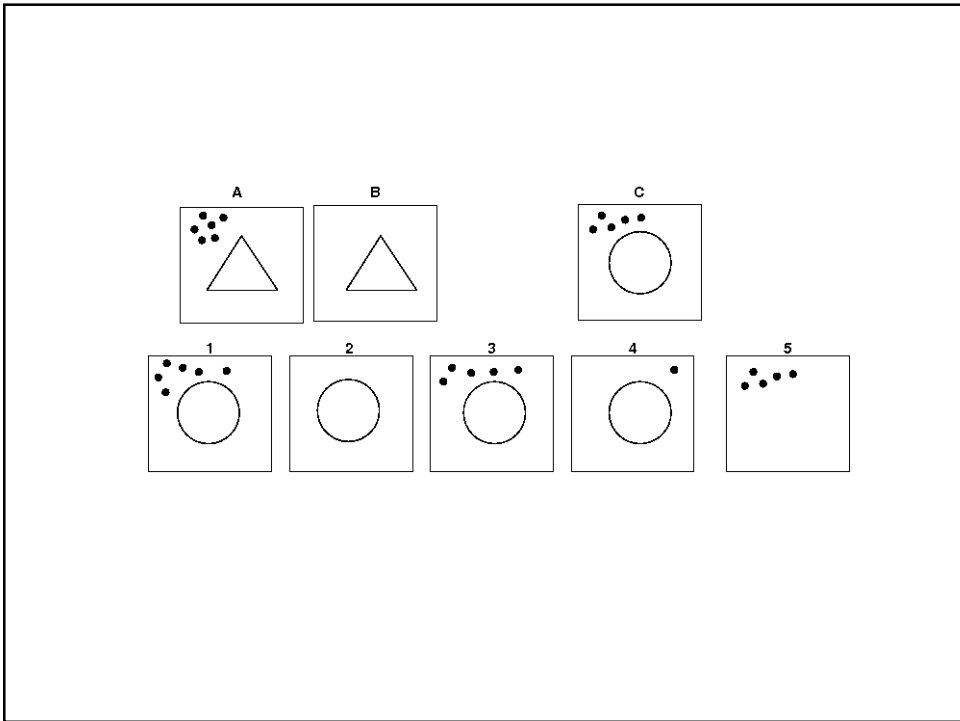
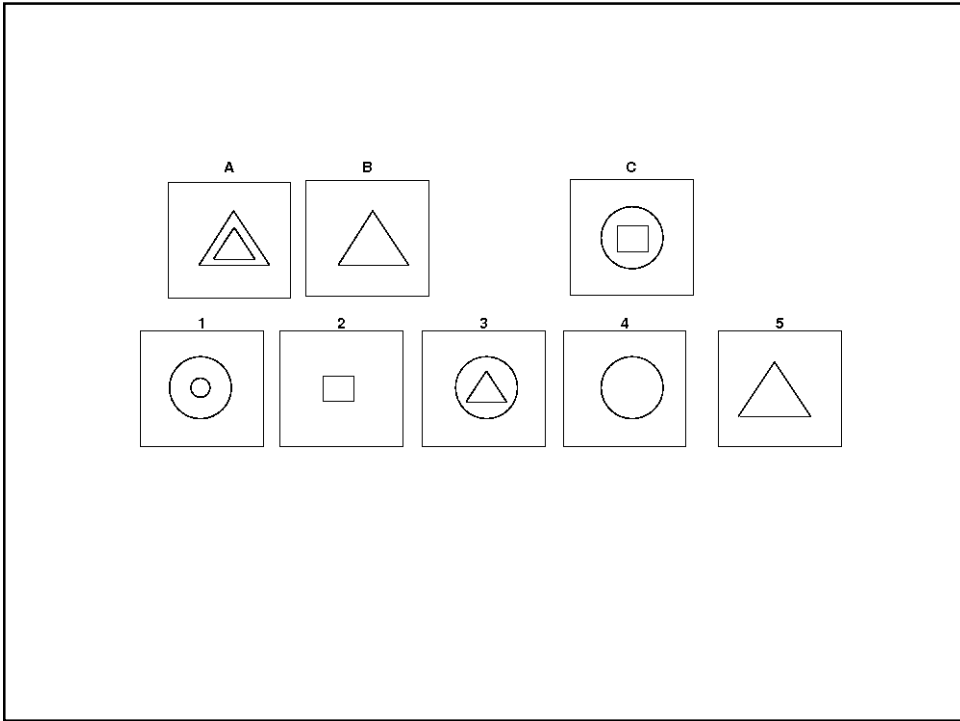
- ANALOGY (Evans)
- Structure Mapping Engine (Gentner, Forbus, Falkenhainer)
- Analogical Constraint Satisfaction Engine (Holyoak, Thagard)
- Copycat (Hofstadter, Mitchell)

ANALOGY

A Program for the Solution of a Class of Geometric-Analogy
Intelligence-Test Questions”

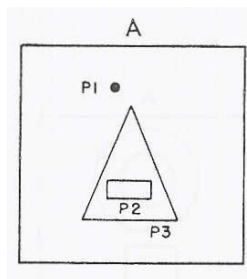
Thomas G. Evans

1968



- Program is given information on how many objects in each box, coordinates of vertices, curvature of lines.
- Program computes properties of figures, using predetermined set of possible properties and relations (e.g., **circular**, **elongated**, **inside-of**, **above**, **left-of**, etc.)
- Program uses given set of possible transformations to make all possible mappings from figures in box A to those in box B (e.g., removal of objects, horizontal reflection, vertical reflection, etc.)

Image encoding

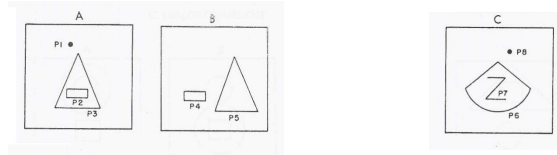


```

1. (
2. (DOT(.04 . 0.8))
3. (SCC((0.3 . 0.2) 0.0 (0.7 . 0.2) 0.0 (0.5 . 0.7) 0.0 (0.3 . 0.2) 0.0)))
4. (SCC((0.4 . 0.3) 0.0 (0.6 . 0.3) 0.0 (0.6 . 0.4) 0.0 (0.4 . 0.4) 0.0 (0.4 . 0.3)))
5. )

```

Line 2. defines the dot P1
 Line 3. defines the triangle P3
 Line 4. defines the rectangle P2

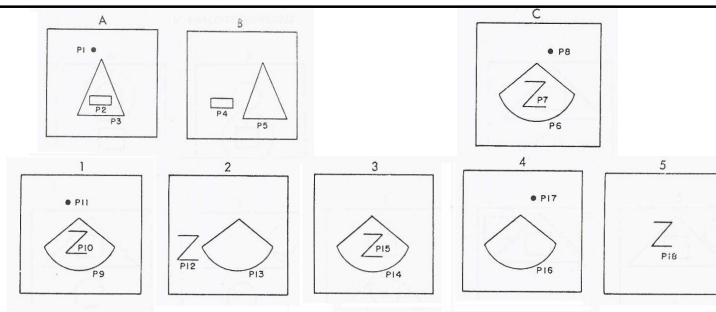


Point = A1, OB3
 Rectangle = A2, OB2
 Triangle = A3, OB1

```

1. (REMOVE A1 ((ABOVE A1 A3) (ABOVE A1 A2)
  (SIM OB3 A1 (((1.0 . 0.0). (N.N))))))
2. (MATCH A2 (((INSIDE A2 A3) (ABOVE A1 A2)
  (SIM OB2 A2 (((1.0 . 0.0). (N.N)))))) .
  ((LEFT A2 A3)
  (SIM OB2 A2 (((1.0 . 0.0). (N.N)) ((1.0 . 3.14) .
(N.N))))
  (SIMTRAN (((1.0 . 0.0). (N.N)) ((1.0 . 3.14) . (N.N))))))
3. (MATCH A3 (((INSIDE A2 A3) (ABOVE A1 A3)
  (SIM OB1 A3 (((1.0 . 0.0). (N.N)))))) .
  ((LEFT A2 A3)
  (SIM OB1 A3 (((1.0 . 0.0). (N.N))))
  (SIMTRAN (((1.0 . 0.0). (N.N))))))

```



- Program then tries to match box C with each of the numbered answer boxes, discarding an answer box if the matching does not agree with the A-to-B rules in terms of number of objects added, removed, or matched. (E.g., discards 1 and 5.)
- Program does exhaustive search through all possible ways of mapping C to each of the remaining answers, given the possible A-to-B rules (some of which can be ignored).
- Each of these mappings is scored on basis of length of the rule (simpler is better), etc. Answer with highest score is chosen.

Results

Accuracy:

ANALOGY accuracy: 15 / 20 problems

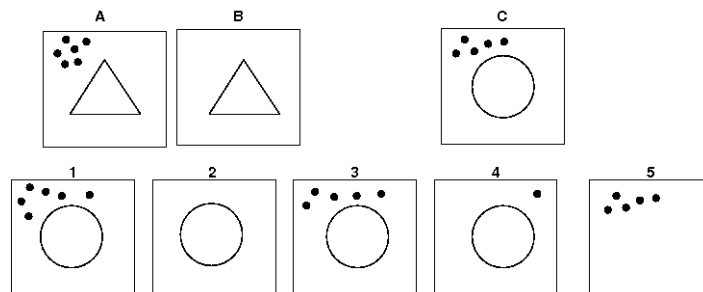
Human Accuracy:

Grade 9 – 17 / 20 problems

Grade 10 – 18 / 20 problems

Grade 11 – 19 / 20 problems

Grade 12 – 20 / 20 problems

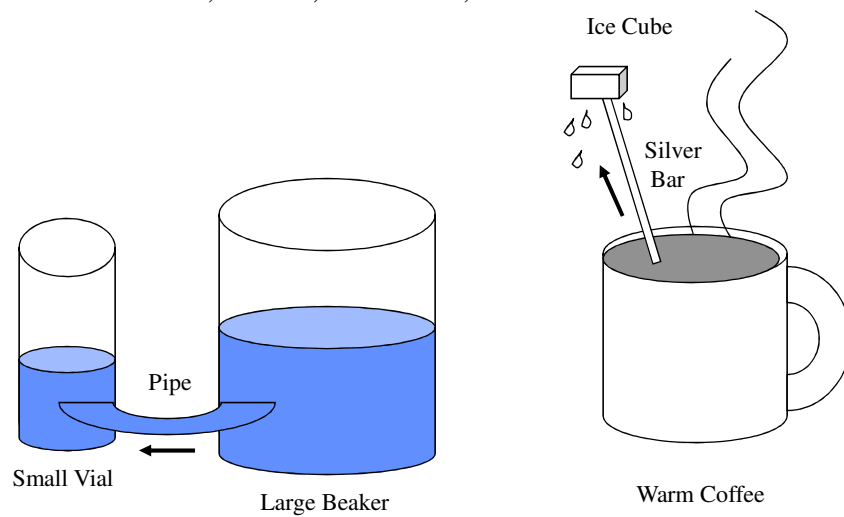


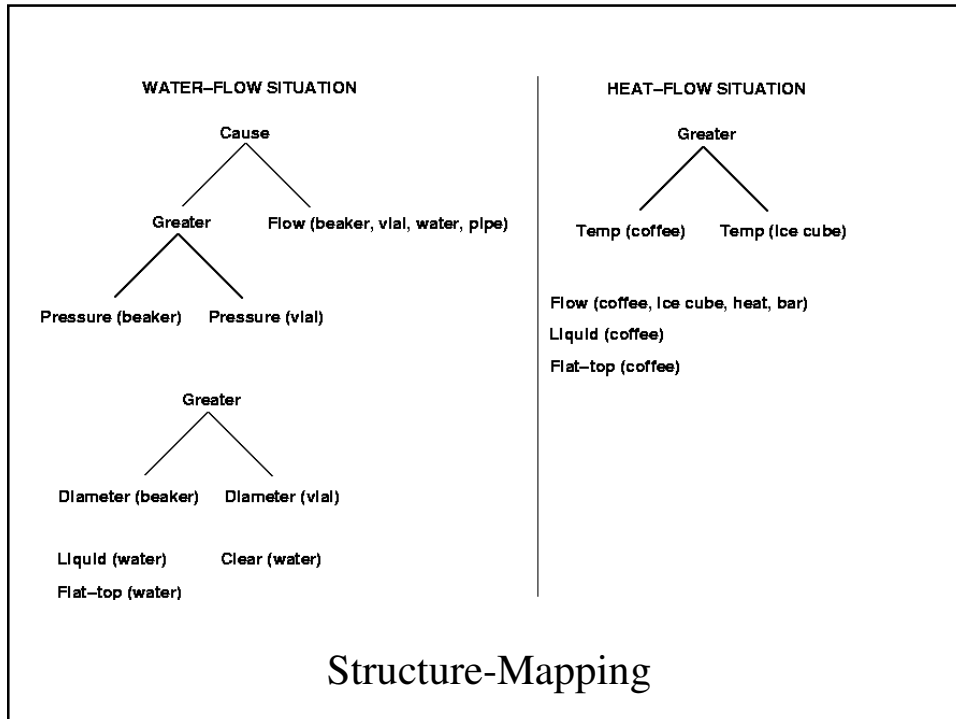
ANALOGY couldn't solve this one: no concept of "grouping".

The Structure-Mapping Engine

(Gentner, Forbus, and Falkenhainer, 1989)

From Falkenhainer, Forbus, & Gentner, 1989





Structure-Mapping Principles

- Richness (how many things in the source are mapped to the target)
- Abstractness (how abstract the things mapped are)
- Systematicity (degree to which the things mapped belong to a coherent interconnected system)

Analogical Constraint Mapping Engine (Holyoak and Thagard, 1989)

Understanding metaphors:

Socrates: "I am a midwife of ideas"

Midwife (source)

(midwife (obj_midwife) m1)
(mother (obj_m2)
(father (objfather) m3)
(child (obj.child) m4)
(matches (obj_midwife obj_mother objfather
(conceives (obj_mother obj.child) m6)
(cause (m5 m6) m7)
(in_labor_with (obj_mother obj.child) m8)
(helps (obj_midwife obj_mother) m10)
(give_birth_to (obj_mother obj.child) m11)
(cause (m10 m11) m12)

Socrates (target)

(philosopher (Socrates) si)
(student (obj_student) s2)
(intellectual_partner (obj_partner) s3)
(idea (obj.idea) s4)
(introduce (Socrates obj_student obj_partner) *5)
(formulates (obj_student obj.idea) so)
(cause (s5 s6) s7)
(thinks_about (obj_student obj.idea) s8)
(tests_truth (obj_student obj.idea) s9)
(helps (Socrates obj_student) \$10)
(knows_truth_or_falsity (obj_student obj.idea) s11)
(cause (\$10 s11) si2)

Best Mappings, With Asymptotic Activation Levels, for Objects and Predicates
in Three Versions of the Socrates/Midwife Metaphor

Cycles to Settle	Versions					
	Isomorphic, Nonpragmatic 34		Nonisomorphic, Nonpragmatic 105		Nonisomorphic, Pragmatic 83	
Objects:						
Socrates	obj_midwife	.87	objfather	.80	obj_midwife	.86
obj_student	obj_mother	.69	obj_mother	.69	obj_mother	.69
obj.partner	objfather	.81	none		obj.father	.80
objidea	obj_child	.90	obj_child	.69	obj.child	.70
*obj_soc-midwife	—		obj_midwife	.84	none	
*obj_soc-wife	—		obj_mother	.69	obj.mother	.69
•obj_soc-child	—		obj_child	.69	obj.child	.65
•obj.hemlock	—		none		none	
Predicates:						
philosopher	midwife	.58	none		midwife	.81
student	mother	.59	none		none	
intellectualpartner	father	.57	none		father	.57
idea	child	.59	none		child	.58
introduces	matches	.77	none		matches	.67
formulates	conceives	.72	conceives	.27	conceives	.31
thinks.about	in_labor_with	.36	none		none	
tests.truth	in_labor_with	.36	none		none	
knows_truth_or_falsity	gives_birth_to	.72	gives_birth_to	.29	gives_birth_to	.31
helps	helps	.77	helps	.79	helps	.80
cause	cause	.84	cause	.84	cause	.84

Limitations

- Hand-designed representations of situations
- Difficulty of encoding situations in predicate logic
- Exhaustive matching and scoring of matches
- (For SME and ACME) Using natural language terms makes program seem “smarter” than it really is.

Copycat

(Hofstadter, Mitchell, Marshall)

Idealizing analogy-making

abc	--->	abd
ijk	--->	?

Idealizing analogy-making

abc ---> abd
iijkk ---> ?

Idealizing analogy-making

abc ---> abd
kji ---> ?

Idealizing analogy-making

abc ---> abd
mrrjj ---> ?

Idealizing analogy-making

abc ---> abd
xyz ---> ?

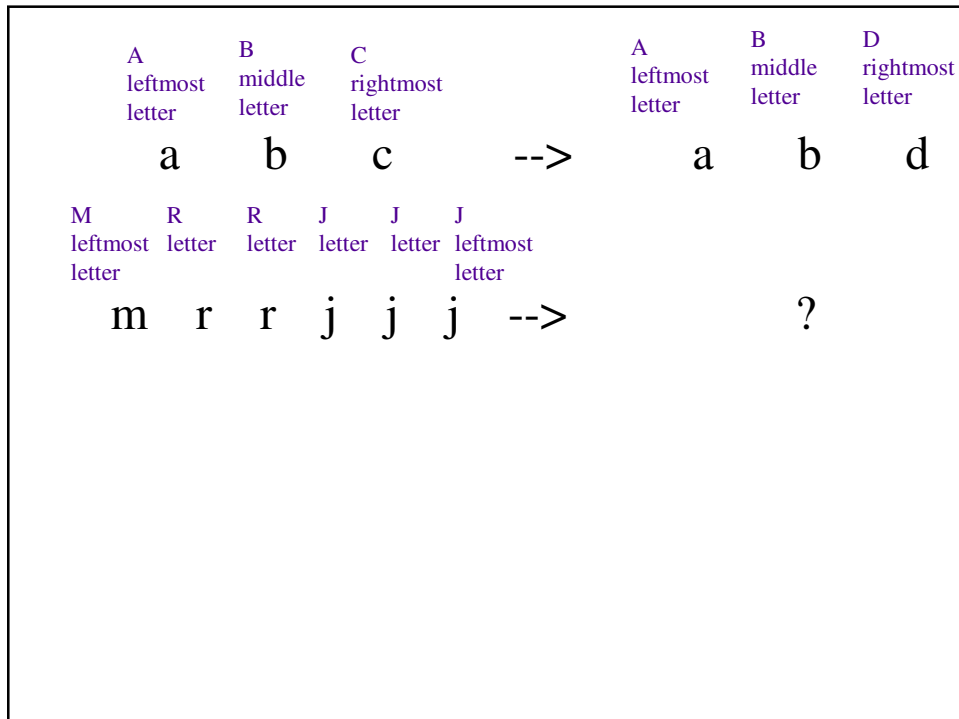
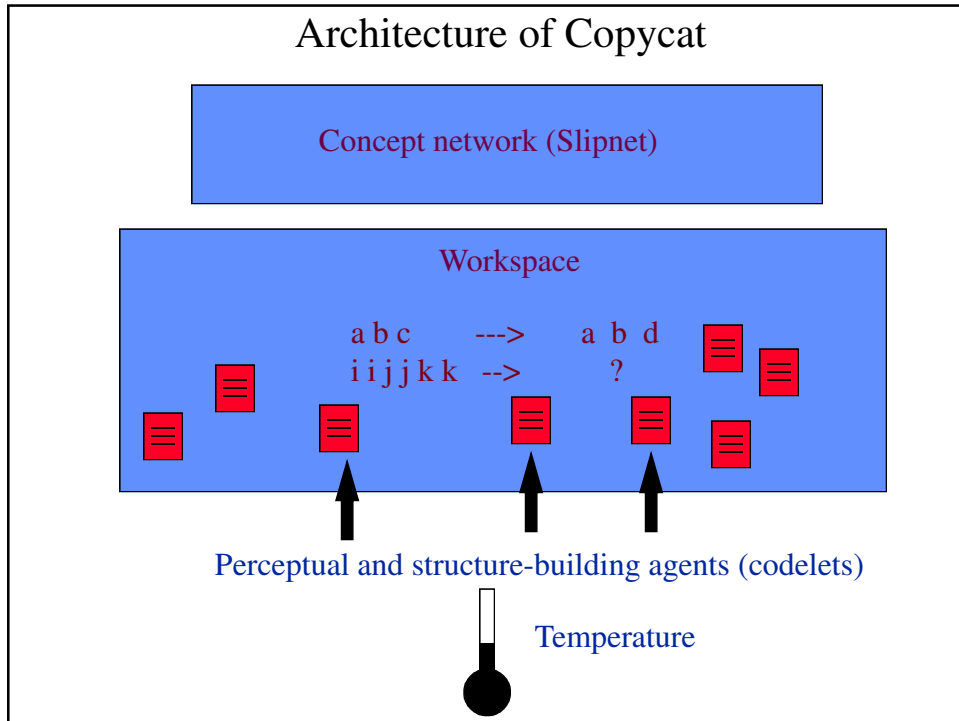
Abilities needed in the letter-string microworld

- Mentally constructing a coherently structured whole out of initially unattached parts
- Describing objects, relations, and events at the appropriate level of abstraction
- Chunking certain elements of a situation while viewing others individually
- Focusing on relevant aspects and ignoring irrelevant or superficial aspects of situations
- Taking certain descriptions literally and letting others slip
- Exploring many avenues of possible interpretations while avoiding a search through a combinatorial explosion of possibilities

The Copycat program (Hofstadter and Mitchell)

- Inspired by collective behavior in complex systems (e.g., ant colonies)
- Understanding and perception of similarity is built up collectively by many independent simple “agents” working in parallel
- Each agent has very limited perceptual and communication abilities
- Teams of agents explore different possibilities for structures, building on what previous teams have constructed.
- The resources (agent time) allocated to a possible structure depends on its promise, as assessed dynamically as exploration proceeds.
- The agents working together produce an “emergent” understanding of the analogy.

Architecture of Copycat

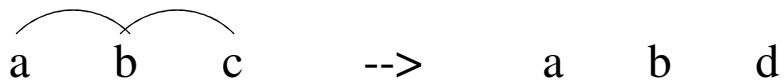


Workspace

- The Workspace starts out with letters in the analogy problem and their initial descriptions.
- Codelets gradually build up additional descriptions and structures.
- Codelets can be either “bottom-up” (noticers) or “top-down” (seekers).

successorship

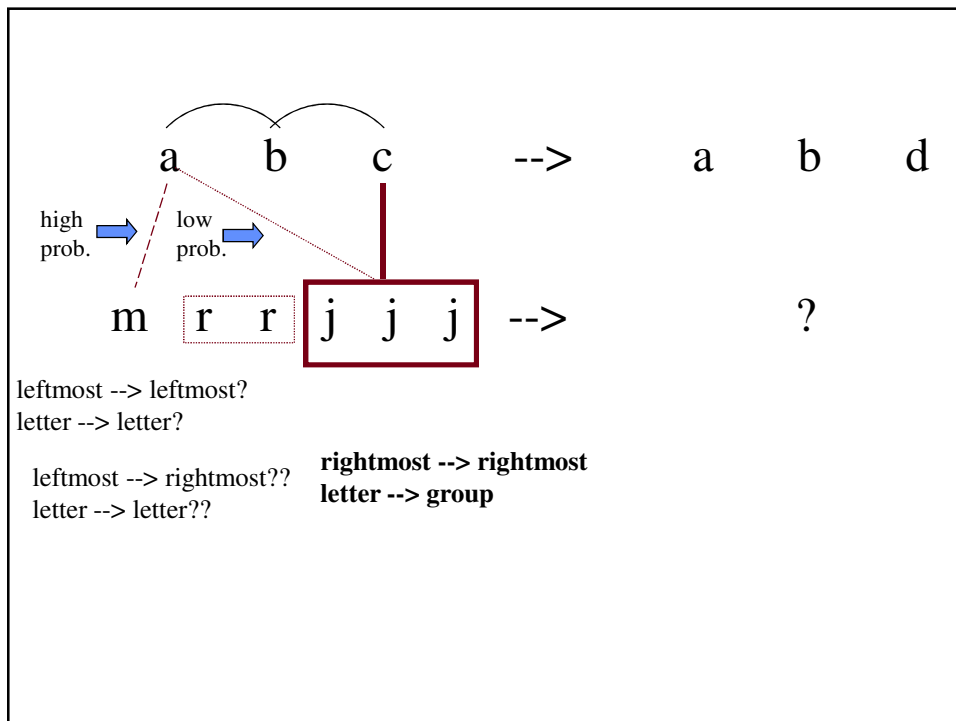
a b c --> a b d



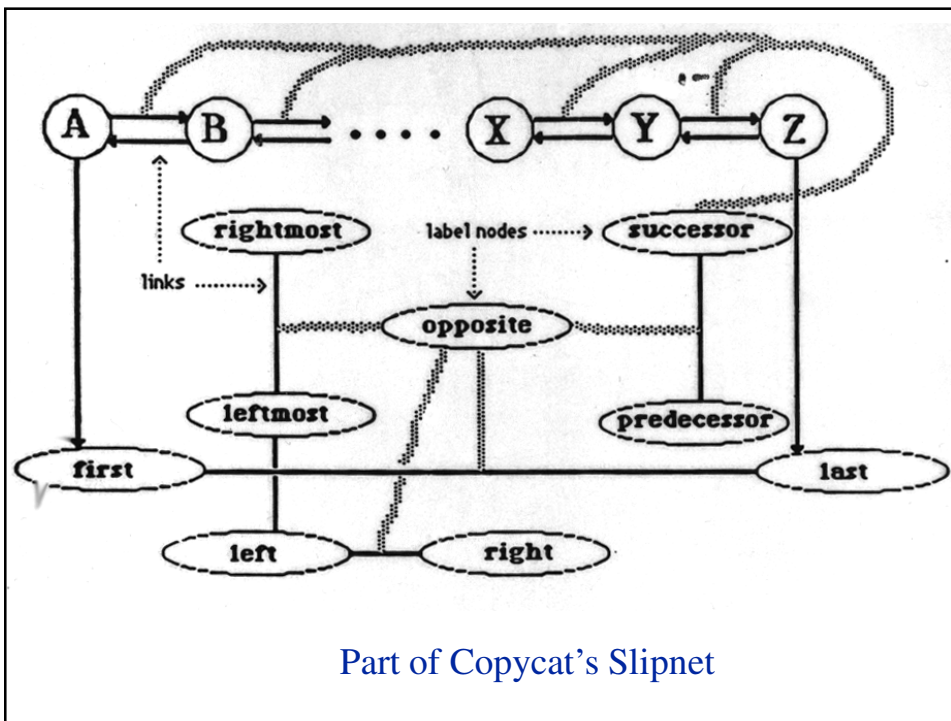
m r r j j j --> ?

Workspace

- Codelets make probabilistic decisions:
 - What to look at next
 - Whether to build a structure there
 - How fast to build it
 - Whether to destroy an existing structure there

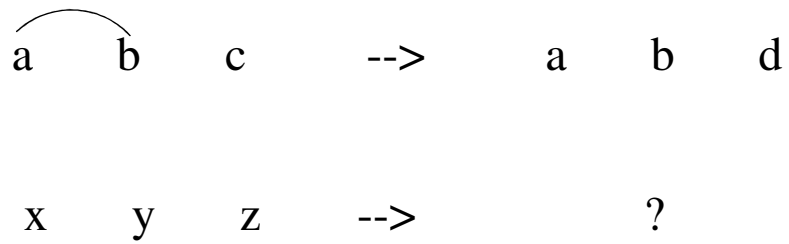


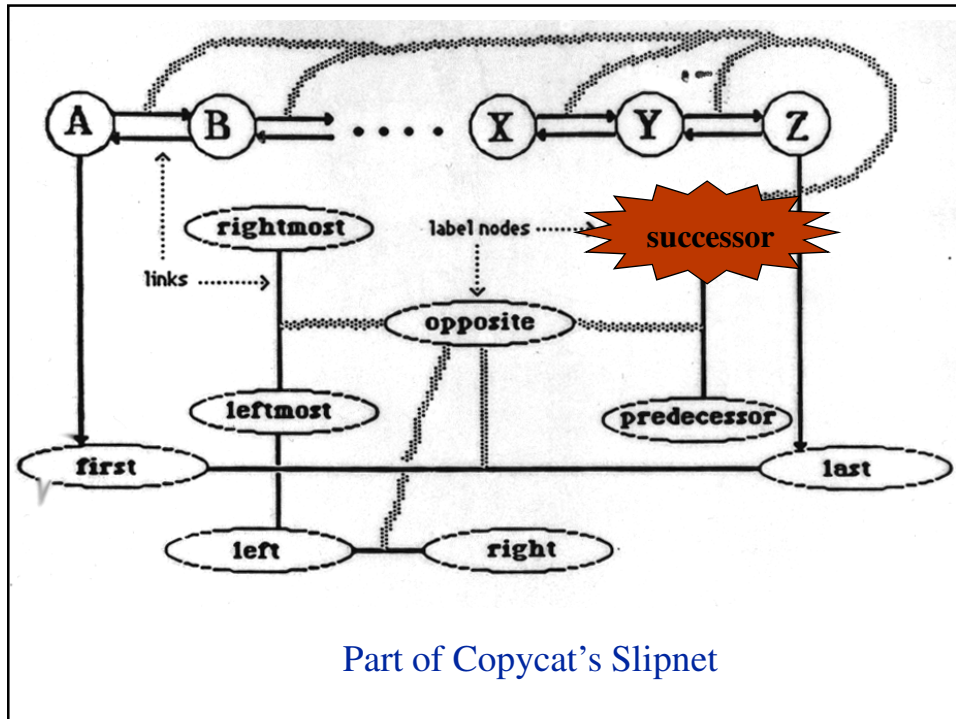
- Probabilities are used to insure that no possibilities are ruled out in principle, but that not all possibilities have to be considered.
- These decisions rely on information being obtained as the run takes place, e.g., pressure from current activation of concepts and neighboring structures.
- Therefore, the probabilities have to be updated continually.



Slipnet

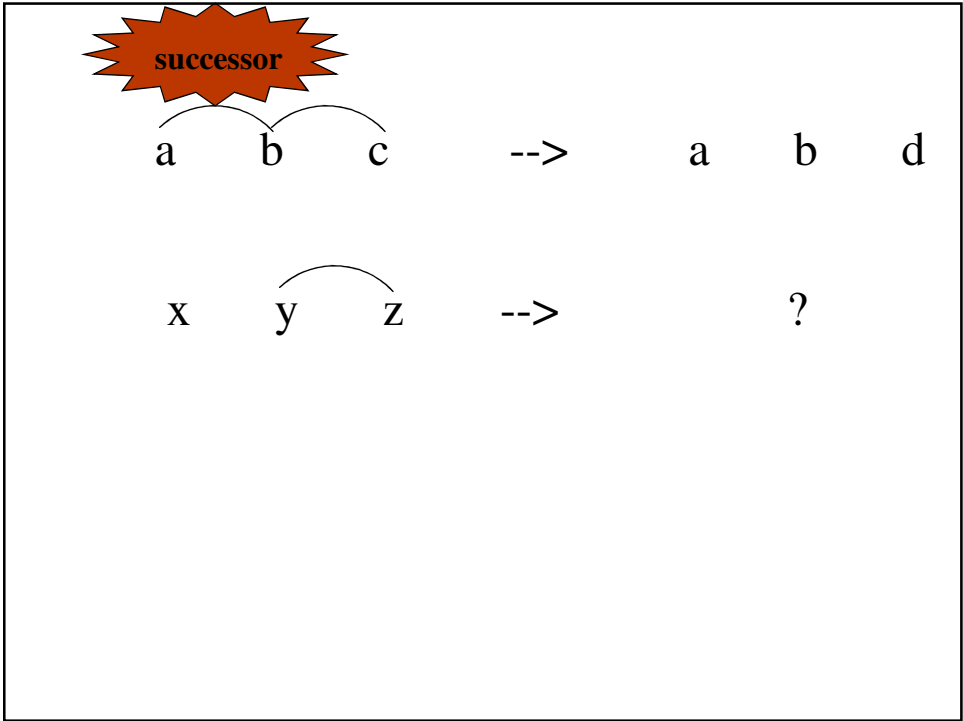
- Concepts are activated as instances are noticed in workspace.





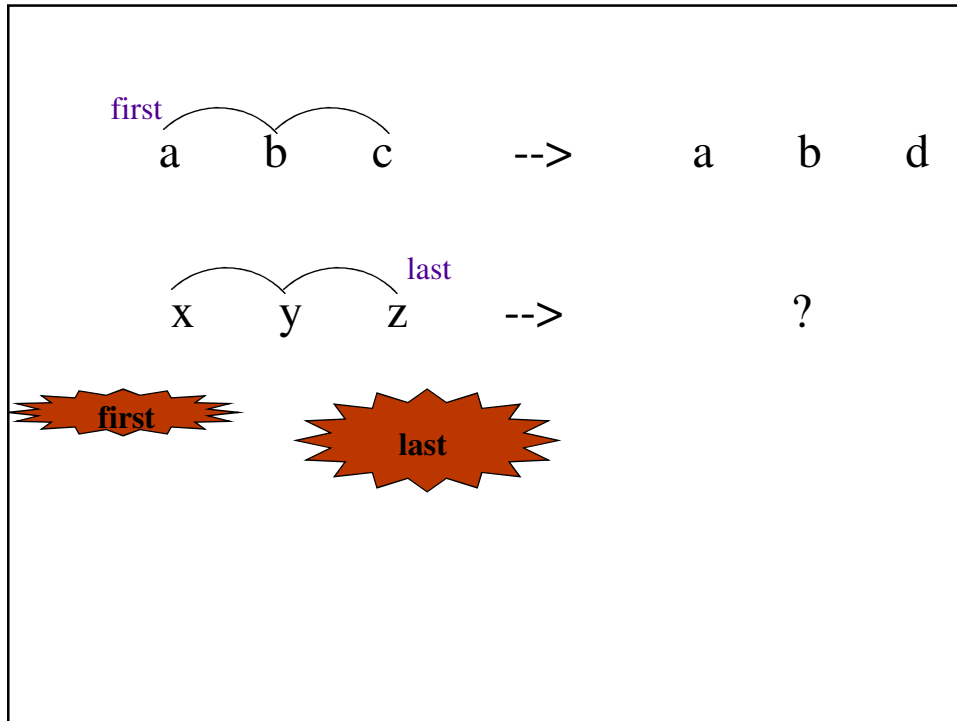
Slipnet

- Activation of concepts feeds back into “top-down” pressure to notice instances of those concepts in the workspace.



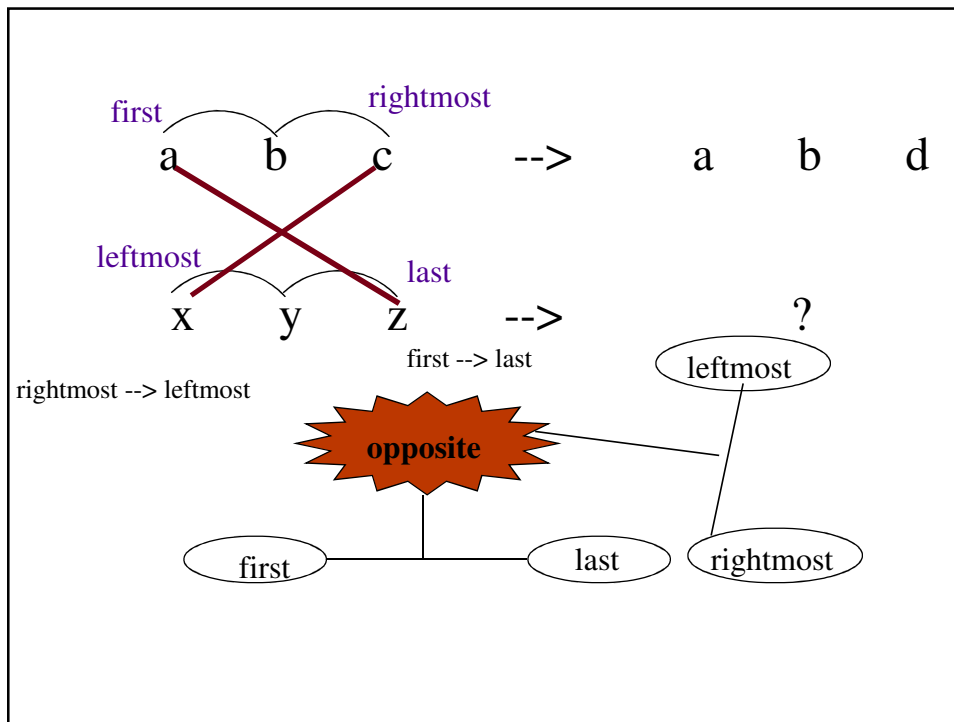
Slipnet

- Activated concepts spread activation to neighboring concepts.



Slipnet

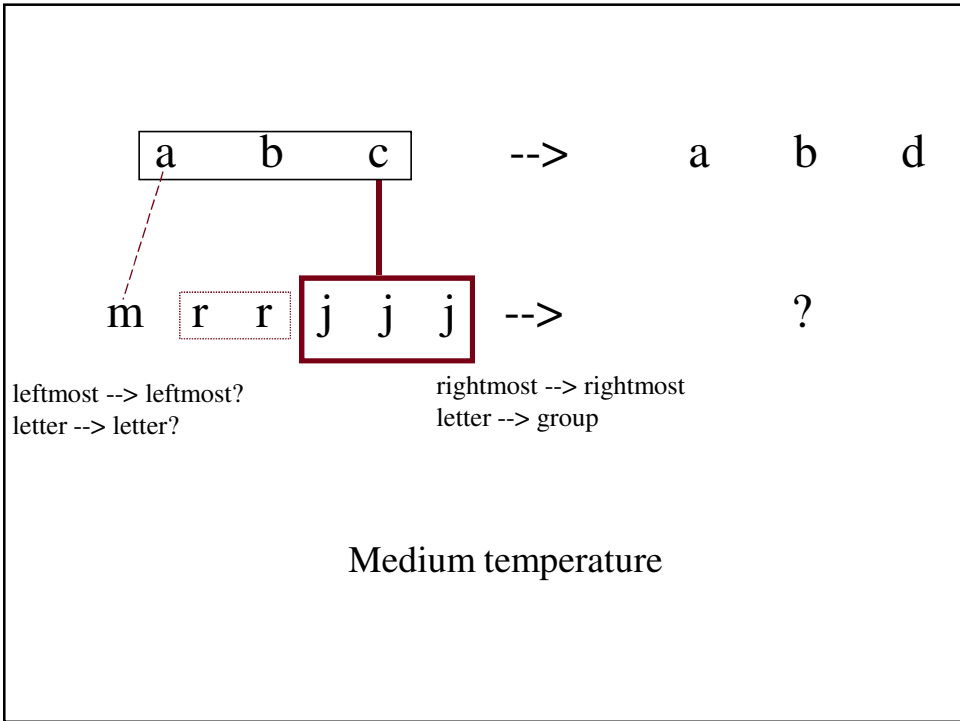
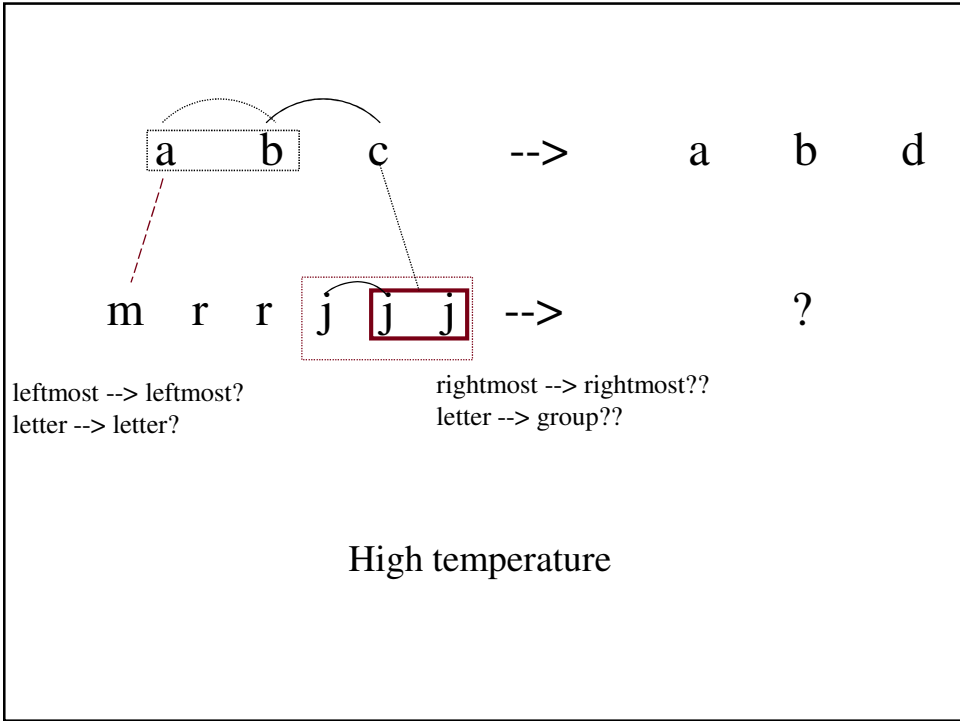
- Activation of *link* concepts determines current ease of slippages of that type (e.g., “opposite”).

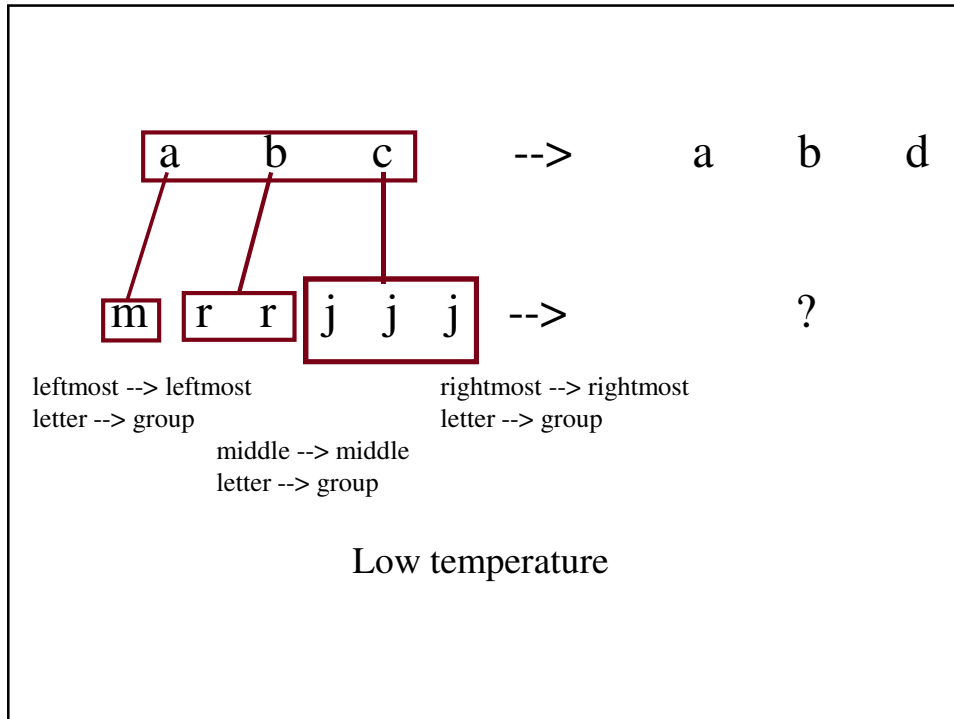


Temperature


- Measures how well organized the program's "understanding" is as processing proceeds (a reflection of how good the current worldview is)
 - Little organization → high temperature
 - Lots of organization → low temperature







Temperature



- Measures how well organized the program's "understanding" is as processing proceeds (a reflection of how good the current worldview is)
 - Little organization —> high temperature
 - Lots of organization —> low temperature
- Temperature feeds back to codelets:
 - High temperature —> low confidence in decisions —> decisions are made more randomly
 - Low temperature —> high confidence in decisions —> decisions are made more deterministically
- **Result:** System gradually goes from random, parallel, bottom-up processing to deterministic, serial, top-down processing

What's needed to apply these ideas in "real world" problems?

- Much expanded repertoire of concepts
- Ability to generate temporary concepts on the fly
(e.g., abc ---> abd, ace ---> ?)
- Ability to learn new permanent concepts
(e.g., bbb ---> ddd, ppp ---> ?)
- Ability for "self-watching"
(e.g., abc ---> abd, xyz ---> ?)

What's needed to apply these ideas in "real world" problems?

- Much expanded repertoire of concepts
- Ability to generate temporary concepts on the fly
(e.g., abc ---> abd, ace ---> ?)
- Ability to learn new permanent concepts
(e.g., bbb ---> ddd, ppp ---> ?)
- Ability for "self-watching"
(e.g., abc ---> abd, xyz ---> ?)

(cf. Marshall, *Metacat*, Ph.D. Dissertation, Indiana University, 1998)

Applications of Ideas from Copycat

- [Letter recognition](#) (McGraw, 1995, Ph.D Dissertation, Indiana University)
- [Natural language processing](#) (Gan, Palmer, and Lua, *Computational Linguistics* 22(4), 1996, pp. 531-553)
- [Robot control](#) (Lewis and Lugar, *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, Erlbaum, 2000.)

Our current work: visual pattern recognition

Bongard problems as a microworld
for pattern recognition, concept-learning

