

# Computer Immunology

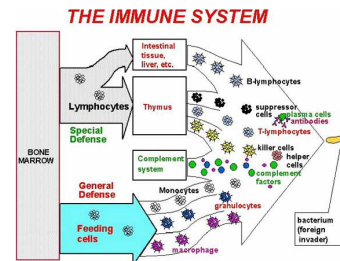
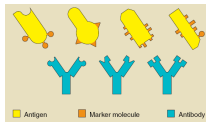


Figure 20. Immune system block diagram.

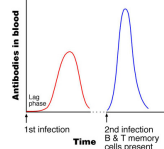
<http://www.cheriere.org/images/rife/rife20.jpg>

- **Primary response:** Antigen has not been “seen” before by the immune system.



[http://www.web-books.com/library/medicine/Physiology/Immune/Antigen\\_Antibody.gif](http://www.web-books.com/library/medicine/Physiology/Immune/Antigen_Antibody.gif)

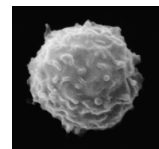
- **Secondary response:** Antigen has been seen; immune system has antibodies well-matched to this specific antigen.



[http://www.unis.org/UNIScienceNet/IBHbio2\\_knowledge.html](http://www.unis.org/UNIScienceNet/IBHbio2_knowledge.html)

## Detection of Pathogens via Affinity Maturation

B-lymphocyte

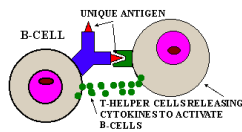


<http://www.miltenyibiotec.com/mac/products/human/h9111-51.htm>

- Trillions of lymphocytes continually circulate in blood and tissues, with continual turnover of lymphocytes.
- A lymphocyte’s surface is covered with identical receptors that bind to a range of molecular shapes with a range of affinities.
- Each individual lymphocyte is born with a set of unique receptors, due to random shuffling of variable gene libraries.

Continual random variation in receptors and individual receptor’s range of affinities:

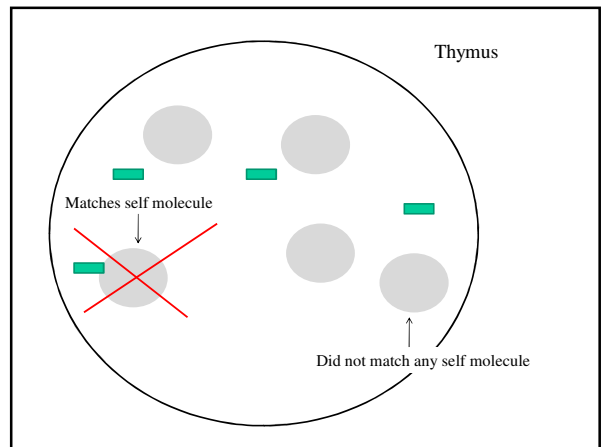
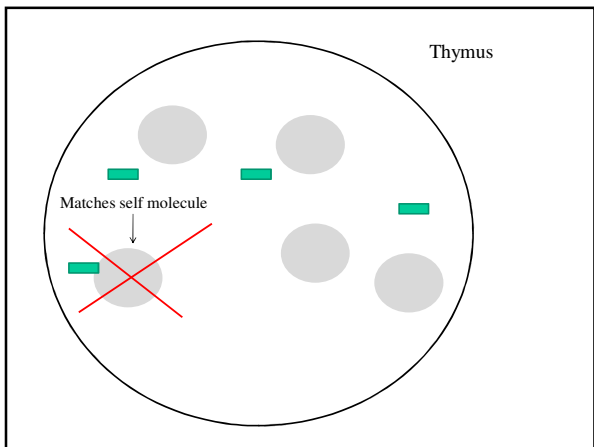
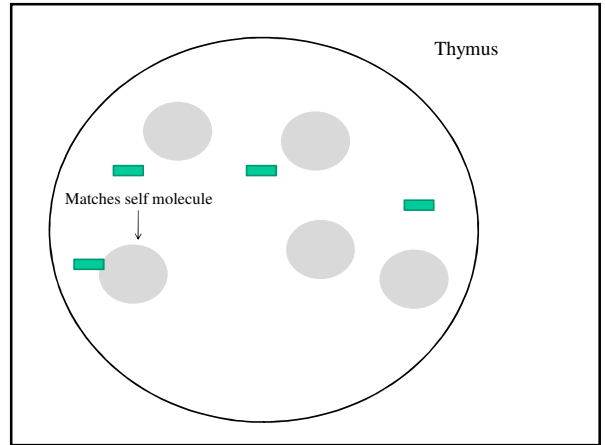
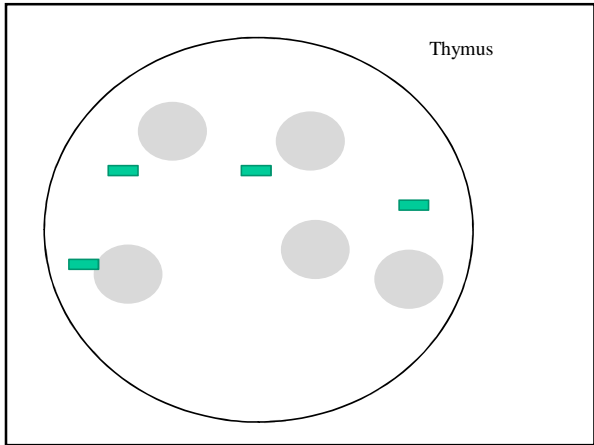
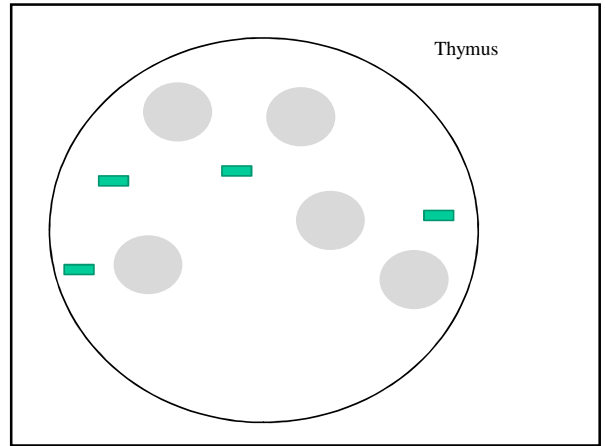
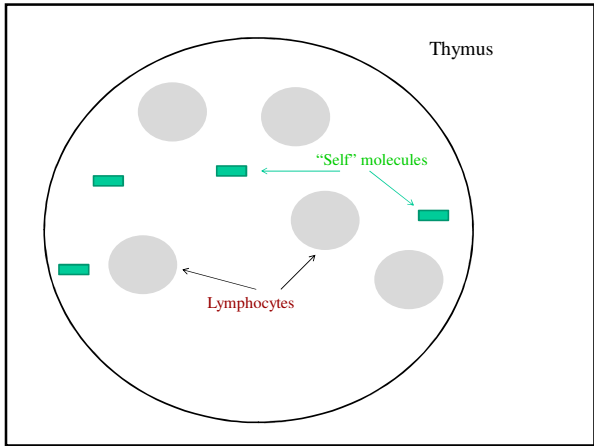
→ Good coverage of huge space of possible pathogen shapes.

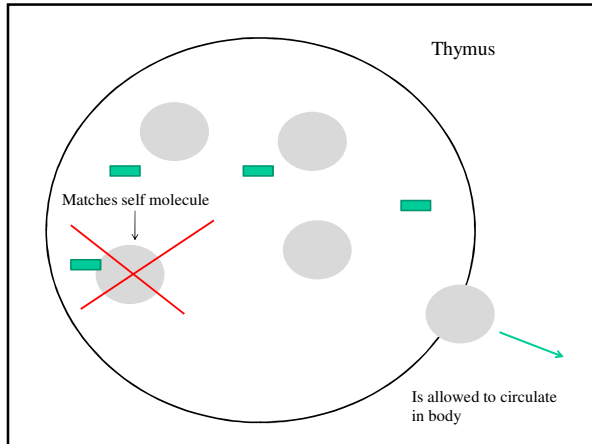


<http://www.slic2.wsu.edu:820/turbert/micro101/images/101ThwithBcell116.gif>

- Lymphocyte binding with antigen; range of possible affinities.
- If number of strongly bound receptors exceeds a threshold, and lymphocyte gets “go-ahead” signal from helper T-cells with similarly bound receptors, then lymphocyte is “activated”.

- How to avoid recognizing and destroying self?
  - Lymphocytes mature in isolated settings (thymus or bone marrow).
  - The ones that match self are killed off.
    - “Negative selection”
  - This is a statistical sampling technique.
  - Lymphocytes that survive are allowed to enter the blood stream.





### Immune-system inspired change detection algorithm

(Forrest, Perleson, et al.)

- Generate set of detectors (e.g., bit strings) at random
  - enough to have “coverage”

Detectors:

```
11110001010101
00010110010010
10001110010101
00100100100011
```

- Let them encounter “self”
  - Fragments of operating system, other code.
  - Kill off those detectors that match self sufficiently strongly.

Detectors:	“Self”:
11110001010101	00101010101110
00010110010010	00010010110101
10001110010101	10011100101111
00100100100011	01000101000010
	11110011010101
	.
	.
	.

- Let them encounter “self”
  - Fragments of operating system, other code.
  - Kill off those detectors that match self sufficiently strongly.

Detectors:	“Self”:
11110001010101	00101010101110
00010110010010	00010010110101
10001110010101	10011100101111
00100100100011	01000101000010
	11110011010101
	.
	.
	.

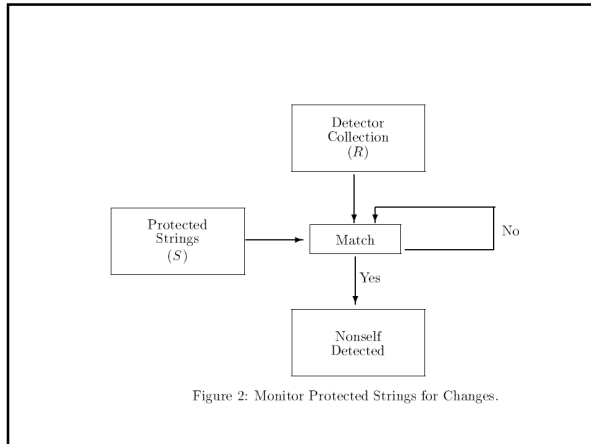
match →

- Let them encounter “self”
  - Fragments of operating system, other code.
  - Kill off those detectors that match self sufficiently strongly.

Detectors:	“Self”:
11110001010101	00101010101110
00010110010010	00010010110101
	10011100101111
00100100100011	01000101000010
	11110011010101
	.
	.
	.

- Use surviving detectors to monitor protected code.
  - If match is found at later time, assume something has changed.

Detectors:
11110001010101
00010110010010
00100100100011



### Intrusion Detection

(Forrest, Hofmeyr, et al.)

Example of “normal system calls” for a given program:  
 open, read, mmap, mmap, open, read, mmap

Create database of trigrams (“self”):

open, read, mmap  
 read, mmap, mmap  
 mmap, mmap, open  
 mmap, open, read

Monitor trigrams of ongoing system calls:  
 if enough mismatches with “self”, suspect intrusion

### Example

Self:

open, read, mmap  
 read, mmap, mmap  
 mmap, mmap, open  
 mmap, open, read

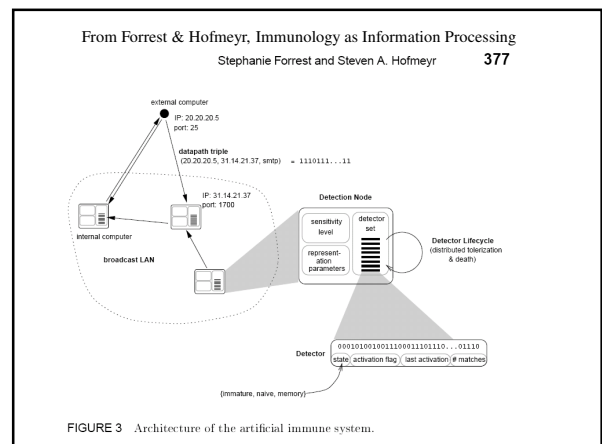
Observed system calls:  
 open, read, mmap, mmap, open, mmap, mmap

Observed trigrams:  
 open, read, mmap                      mmap, open, mmap  
 read, mmap, mmap  
 mmap, open, mmap

Program	Number Mismatches	% Mismatches	$\hat{S}_k$
ls	42	75	0.6
ls -l	134	91	1.0
ls -a	44	76	0.6
ps	539	97	0.6
ps -ux	1123	99	0.6
finger	67	83	0.6
ping	41	57	0.6
ftp	271	90	0.7
pine	450	77	1.0

Table 3. Distinguishing *sendmail* from other programs. Each column reports results for a single anomalous measures: Mismatches (column 2), percentage of mismatches over a trace (column 3), and  $\hat{S}_k$  (column 4). The results shown are for a sequence length of  $k = 10$ . There are no mismatches against *sendmail* itself because the database includes all variations.

- ### Network intrusion detection and other “anomalies” (e.g., distributed “denial of service” attacks):
- Local-area broadcast network:
    - Every computer on the network sees every packet passing through the network
    - Communication:
      - Pair-wise connection between computers in the LAN or between outside computer and computer in the LAN
    - Pair-wise connection represented by triple: (source-I, destination-IP, communication port)
    - Encoded as 49-bit string that unambiguously defines the connection



- Artificial immune system:

- "Self":

- Normal pair-wise connections between computers

- Non-self":

- All other all other possible pair wise connections

- Detectors:

- 49-bit strings with small amount of local state

- States: immature cell, mature but naive cell, memory cell

## Analogy-Making

### Idealizing analogy-making

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

### Idealizing analogy-making

abc ---> abd  
ijk ---> **ijl** (replace rightmost letter by successor)

### Idealizing analogy-making

abc ---> abd  
ijk ---> **ijl** (replace rightmost letter by successor)  
**ijd** (replace rightmost letter by 'd')

Idealizing analogy-making

abc ---> abd  
ijk ---> **ijl** (replace rightmost letter by successor)  
**ijd** (replace rightmost letter by 'd')  
**ijk** (replace all 'c's by 'd's')

Idealizing analogy-making

abc ---> abd  
ijk ---> **ijl** (replace rightmost letter by successor)  
**ijd** (replace rightmost letter by 'd')  
**ijk** (replace all 'c's by 'd's')  
**abd** (replace any string by 'abd')

Idealizing analogy-making

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

Idealizing analogy-making

abc ---> abd  
iijkk ---> **iijkl**

Replace rightmost letter by successor

Idealizing analogy-making

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

Idealizing analogy-making

abc ---> abd  
iijkk ---> **iijll**

Replace rightmost "letter" by successor

Idealizing analogy-making



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

Idealizing analogy-making



abc ---> abd  
kji ---> **kjj**

Replace rightmost letter by successor

Idealizing analogy-making

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


Idealizing analogy-making


 abc ---> abd  
kji ---> **lji**  


Replace "rightmost" letter by successor

Idealizing analogy-making

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

Idealizing analogy-making

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


Idealizing analogy-making

$\begin{matrix} \rightarrow \\ abc & \dashrightarrow & abd \\ \rightarrow \\ kji & \dashrightarrow & kjh \end{matrix}$

Replace rightmost letter by "successor"

Idealizing analogy-making

$\begin{matrix} abc & \dashrightarrow & abd \\ mrrjj & \dashrightarrow & ? \end{matrix}$

Idealizing analogy-making

$\begin{matrix} abc & \dashrightarrow & abd \\ mrrjj & \dashrightarrow & mrrjk \end{matrix}$

Replace rightmost letter by successor

Idealizing analogy-making

$\begin{matrix} abc & \dashrightarrow & abd \\ mrrjj & \dashrightarrow & ? \end{matrix}$

Idealizing analogy-making

$\begin{matrix} abc & \dashrightarrow & abd \\ mrrjj & \dashrightarrow & ? \\ 1\ 2\ 3 & & \end{matrix}$

Idealizing analogy-making

$\begin{matrix} abc & \dashrightarrow & abd \\ mrrjj & \dashrightarrow & ? \\ 1\ 2\ 3 & & 1\ 2\ 4 \end{matrix}$

Idealizing analogy-making

abc      --->    abd  
mrrjjj   --->    mrrjjj  
  1 2 3            1 2 4  
Replace rightmost "letter" by successor

Idealizing analogy-making

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

Idealizing analogy-making

abc      --->    abd  
xyz      --->    xya  
Replace rightmost letter by successor

Idealizing analogy-making

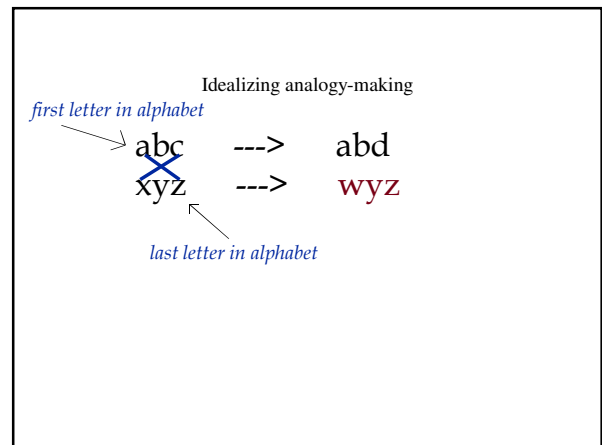
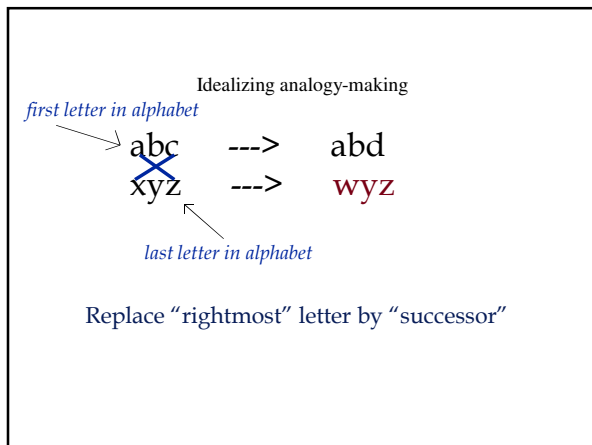
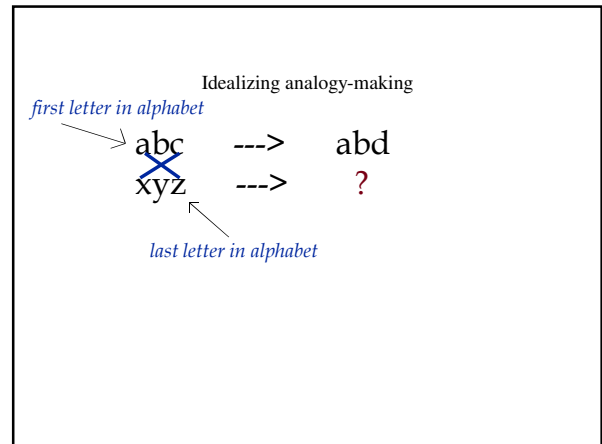
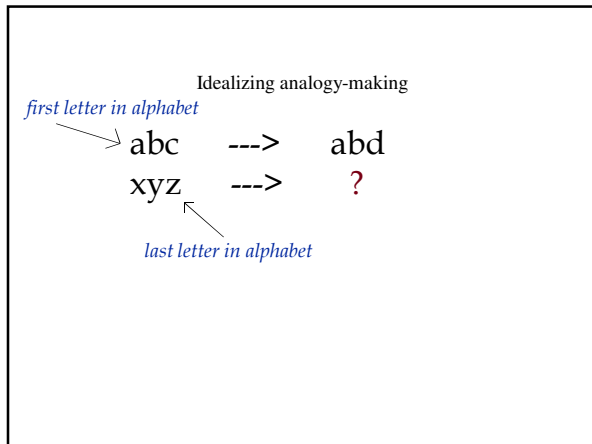
abc      --->    abd  
xyz      --->    ~~xya~~ (not allowed)  
Replace rightmost letter by successor

Idealizing analogy-making

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

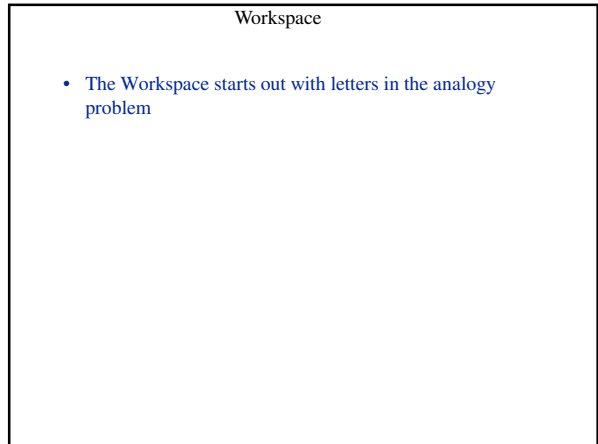
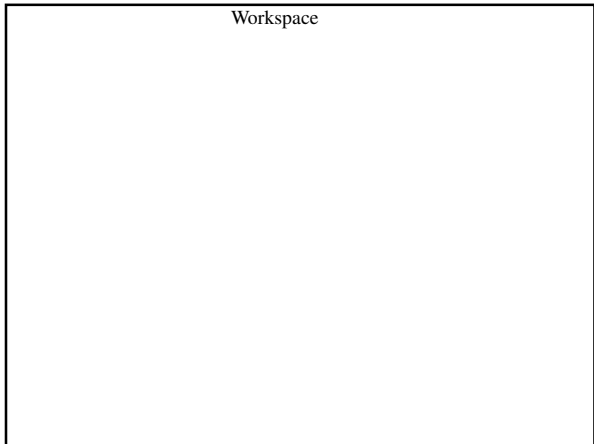
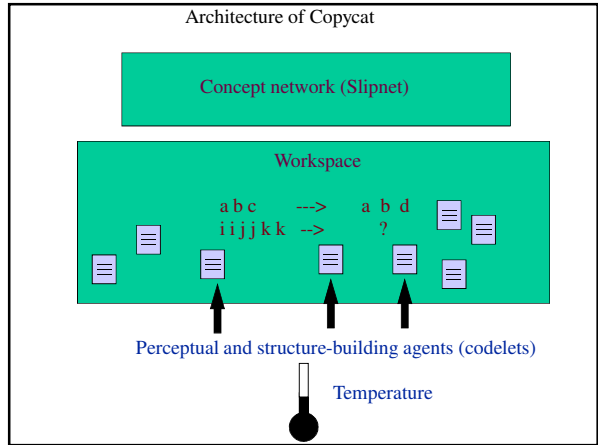
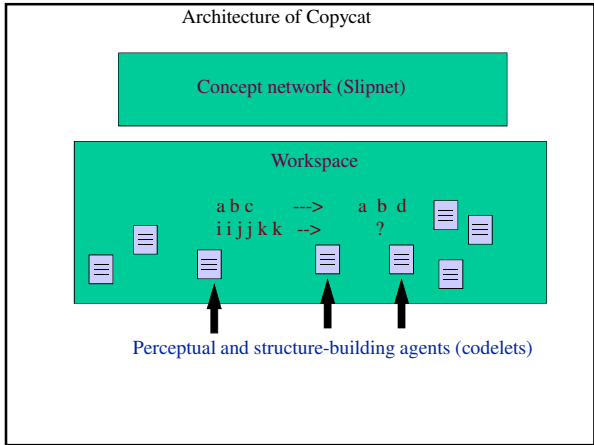
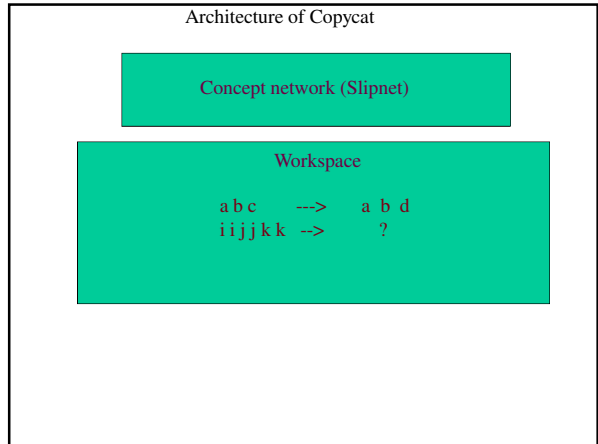
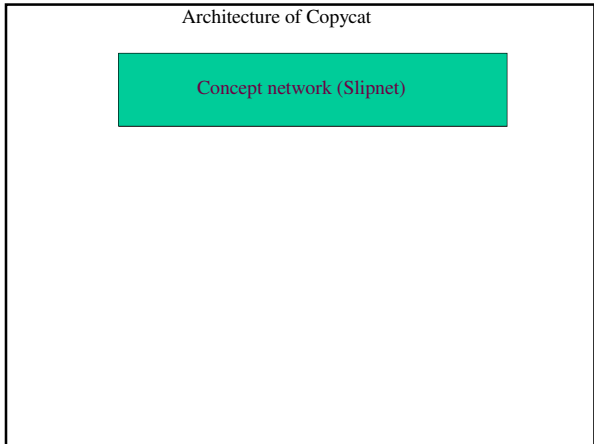
Idealizing analogy-making

abc      --->    abd  
xyz      --->    ?  
          ↖  
          *last letter in alphabet*



The Copycat program  
(Hofstadter and Mitchell)

Architecture of Copycat



a b c --> a b d  
m r r j j j --> ?

Workspace

Codelets gradually build up descriptions, relationships, and correspondences.

a b c --> a b d  
m r r j j j --> ?

a b c --> a b d  
m r r j j j --> ?

successorship

a b c --> a b d  
m r r j j j --> ?

successorship

a b c --> a b d  
m r r j j j --> ?

Workspace

- Codelets make probabilistic decisions:

Workspace

- Codelets make probabilistic decisions:
  - What to look at next

Workspace

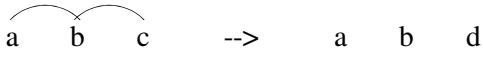
- Codelets make probabilistic decisions:
  - What to look at next
  - Whether to build a structure there

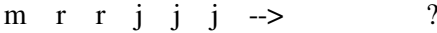
Workspace

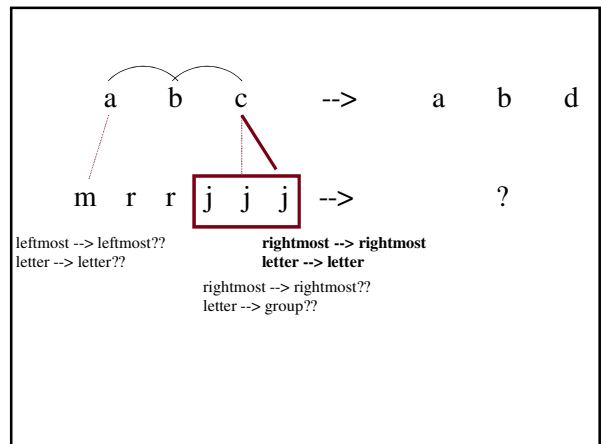
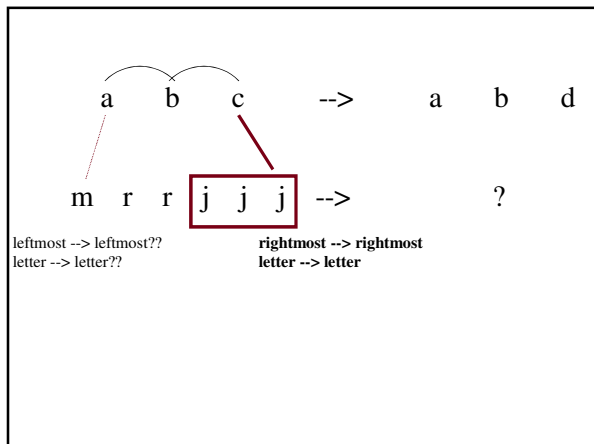
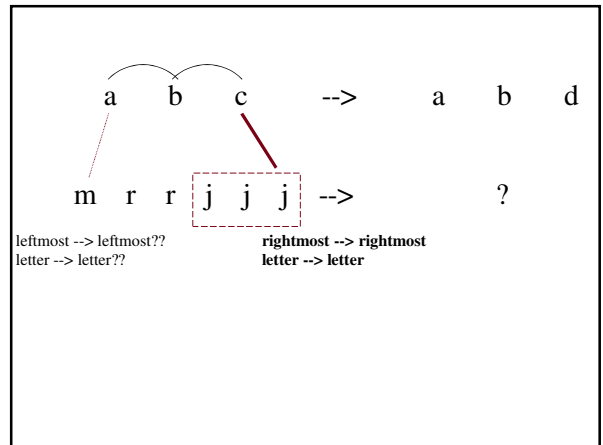
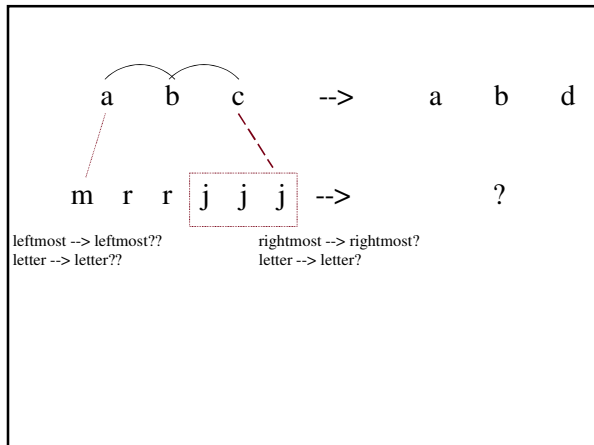
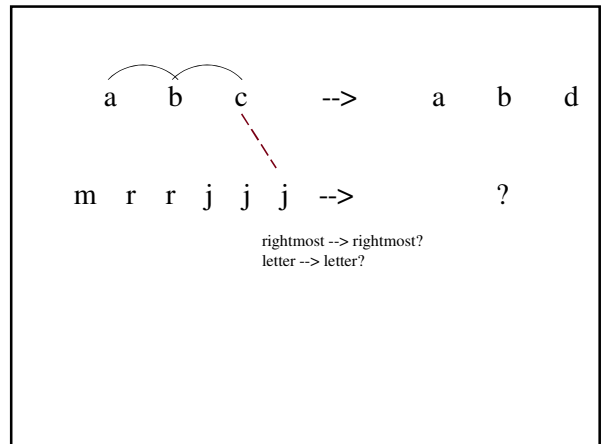
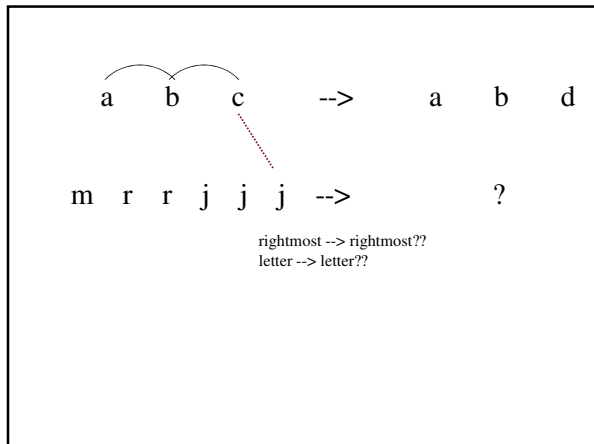
- Codelets make probabilistic decisions:
  - What to look at next
  - Whether to build a structure there
  - How fast to build it

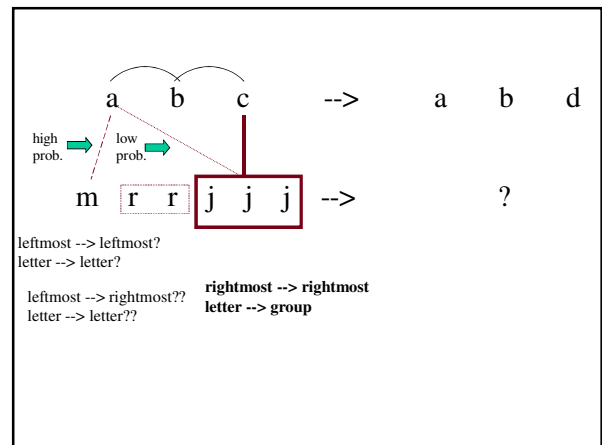
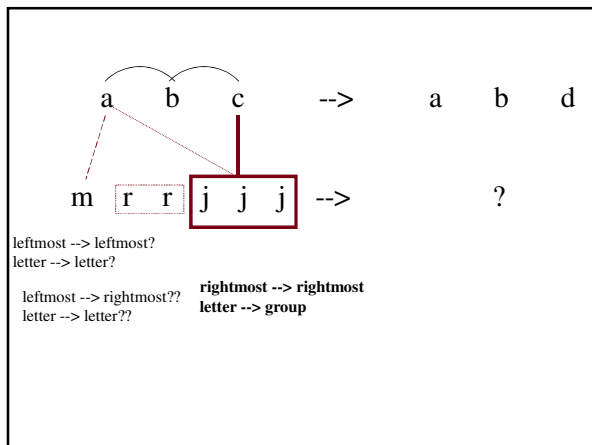
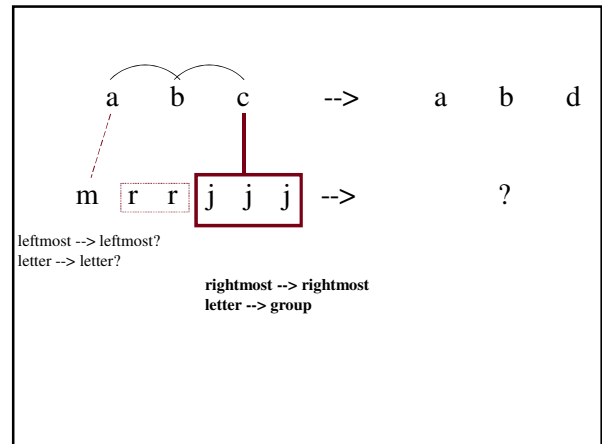
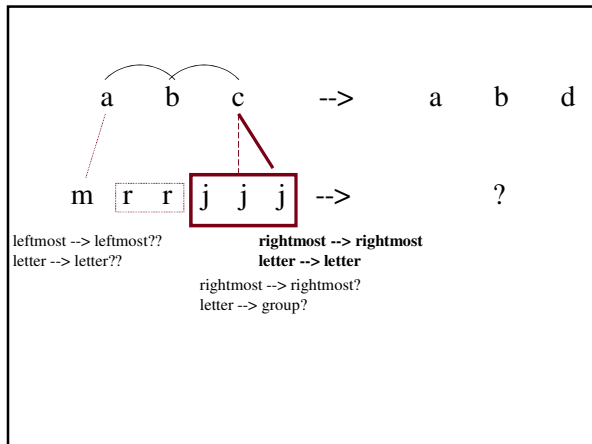
Workspace

- 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

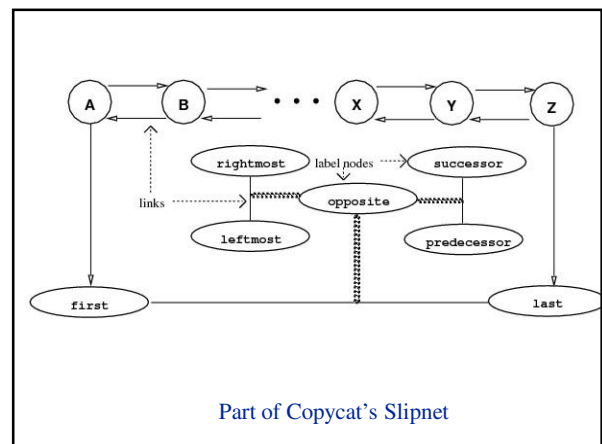








Slipnet

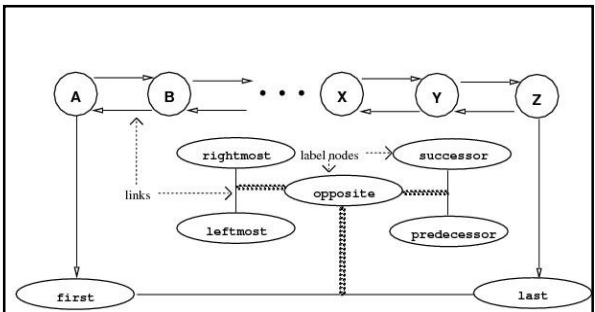


### Slipnet

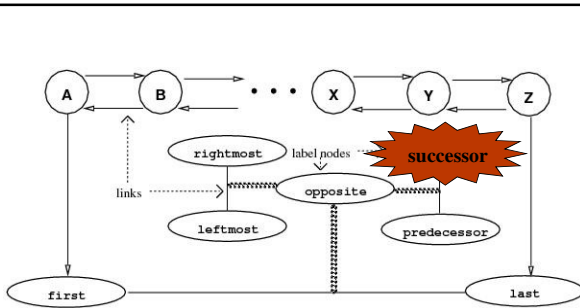
- Concepts are activated as instances are noticed in workspace.

a b c --> a b d  
 x y z --> ?

a b c --> a b d  
 x y z --> ?



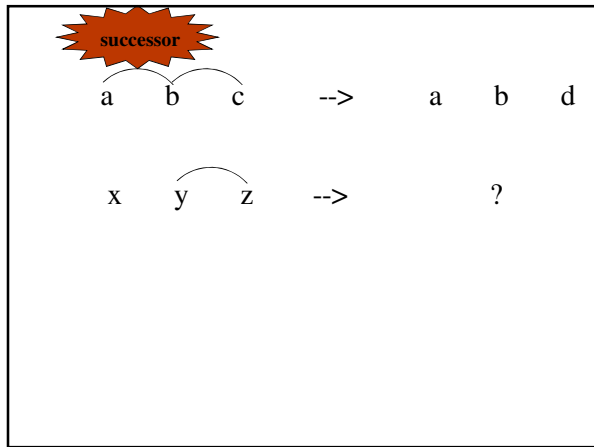
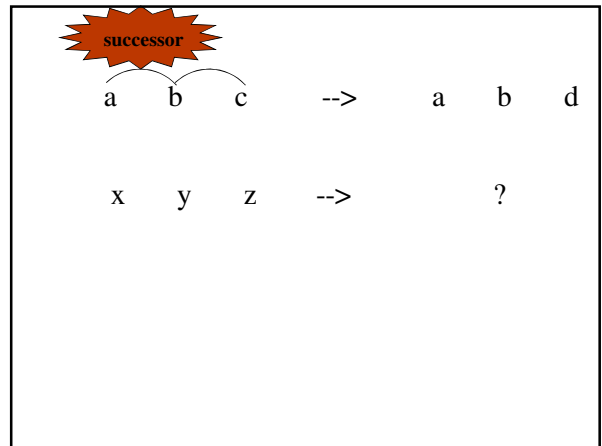
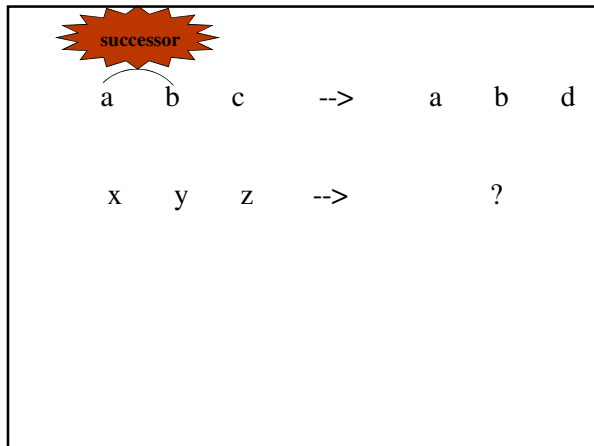
Part of Copycat's Slipnet



Part of Copycat's Slipnet

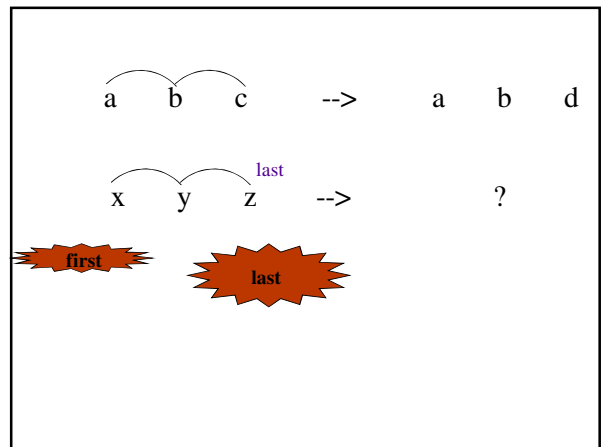
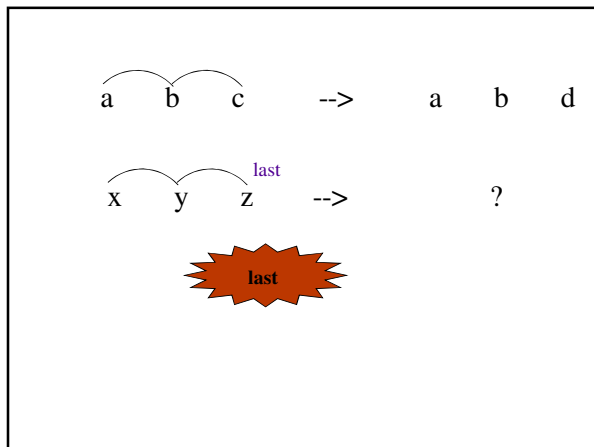
### Slipnet

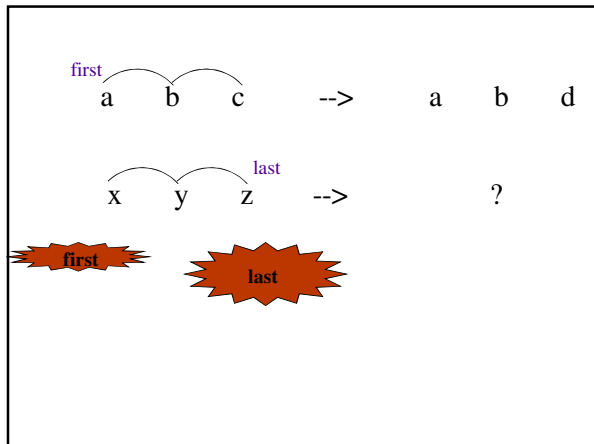
- Activation of concepts feeds back into "top-down" pressure to notice instances of those concepts in the workspace.



Slipnet

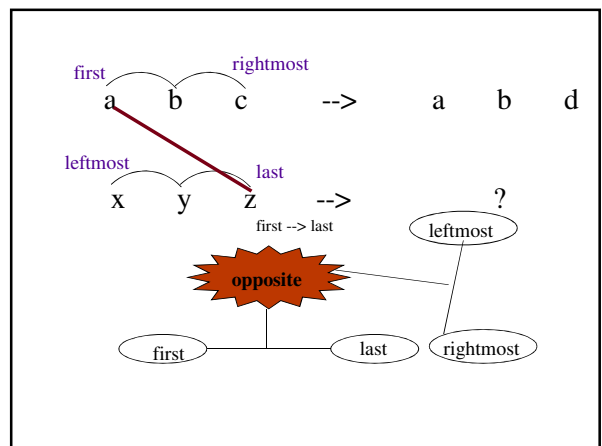
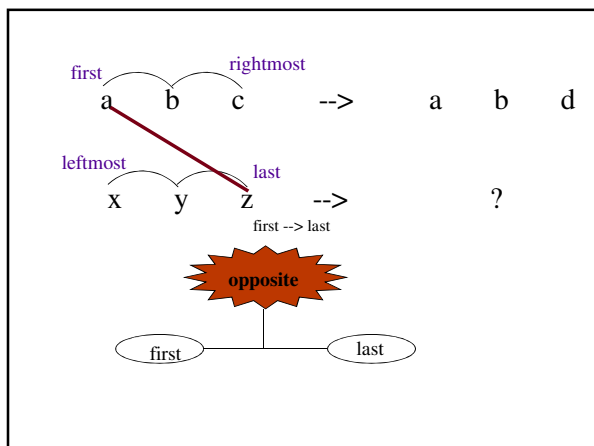
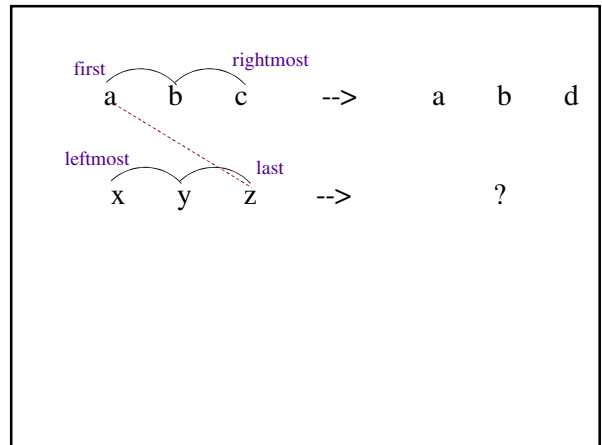
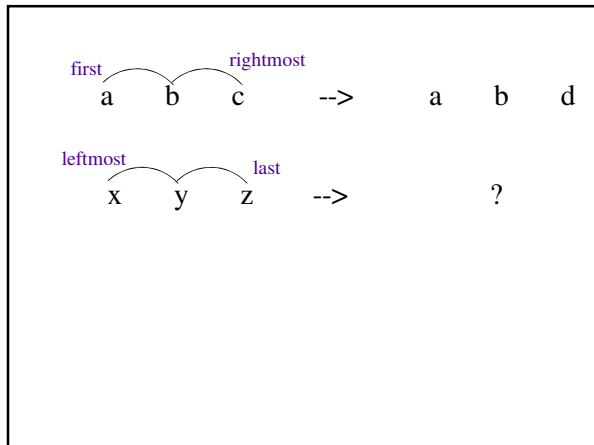
- Activated concepts spread activation to neighboring concepts.

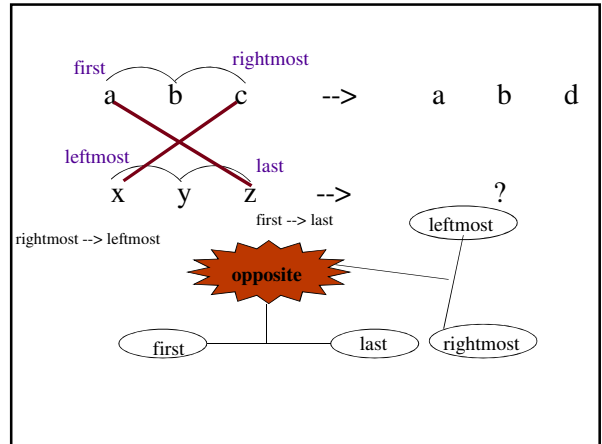
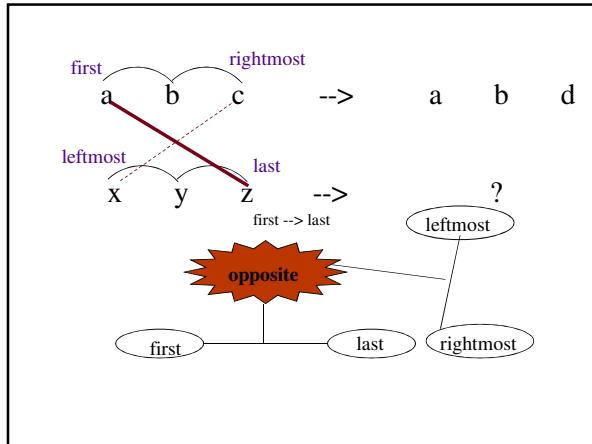




Slipnet

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





Temperature

Temperature

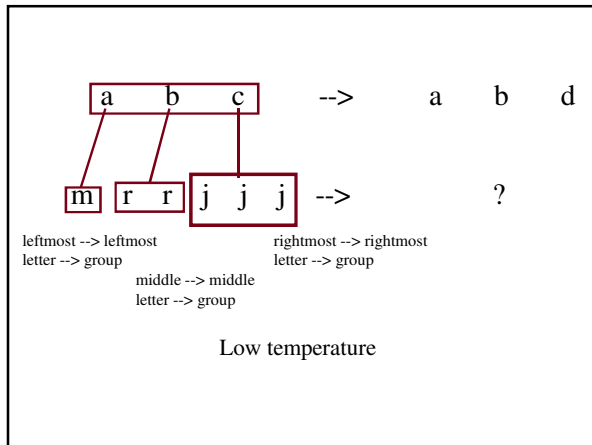
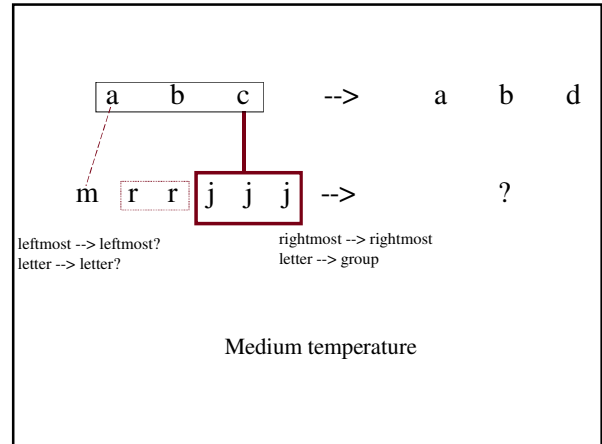
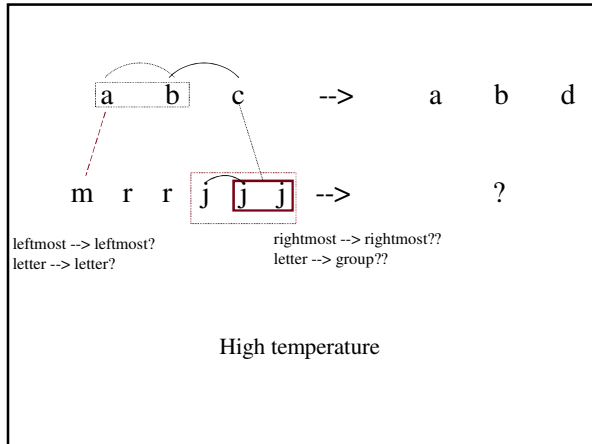
- Measures how well organized the program's "understanding" is as processing proceeds (a reflection of how good the current worldview is)

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

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

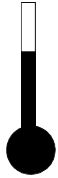
### Temperature

- 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:

### Temperature

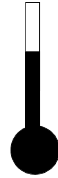
- Measures how well organized the program's "understanding" is as processing proceeds (a reflection of how good the current worldview is)
  - Little organization → high temperature
  - Lots of organization → low temperature
- Temperature feeds back to codelets:
  - High temperature → low confidence in decisions → decisions are made more randomly

## 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

## 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

Metacat Demo

(Jim Marshall)