Analogical Mapping with Vector Symbolic Architectures

Ross Gayler

r.gayler@gmail.com

Research question

How to structure a feasible physical system so that it can do the sort of information processing that humans can?

(What are the “design” principles?)

Then build it! (at least, in principle)

Dual objectives:

- Functional performance
- Practical feasibility

Methodological approach

Approach the research question as an experimental engineering problem

- Hypothesize design principles
- Build a test system
- Diagnose the problems
- Repeat

(Is this cognitive science? Why take this approach?)

A gap in cognitive science

Levels of explanation

- Psychology
  Looks at information content, not mechanisms
- Neurophysiology
  Too low level

  (Explain programming from semiconductor physics)
- Neuroanatomy
  Too coarse

  (Explain programming from cabinet wiring)
- Connectionist models
  Approximately the right level
  BUT – most are functionally inadequate

Mind the gap - Jackendoff's Challenges

Foundations of Language (2002)

- Language/cognition is neurally implemented
- Cognitive neuroscience view of linguistic phenomena seems naive

Four challenges for cognitive neuroscience

- Core linguistic/cognitive functionality (not obviously inherent in connectionist models)
  - Massiveness of binding
  - Problem of 2
  - Problem of variables
  - Equivalence of binding in WM and LTM

Challenge 1: Massiveness of binding

- Representations must be composed

  A commonly used example of binding:

  “red square” versus “blue circle”

- Jackendoff’s example sentence:

  “The little star’s beside a big star”

  Encoded with approximately 130 tokens and 160 relations between tokens

- Binding must be rapid and cope with novelty
- This a problem for localist connectionism

  Units represent by virtue of their identity (VW cells)

  Novel meanings demand novel units (and connections)

“The little star’s beside a big star”

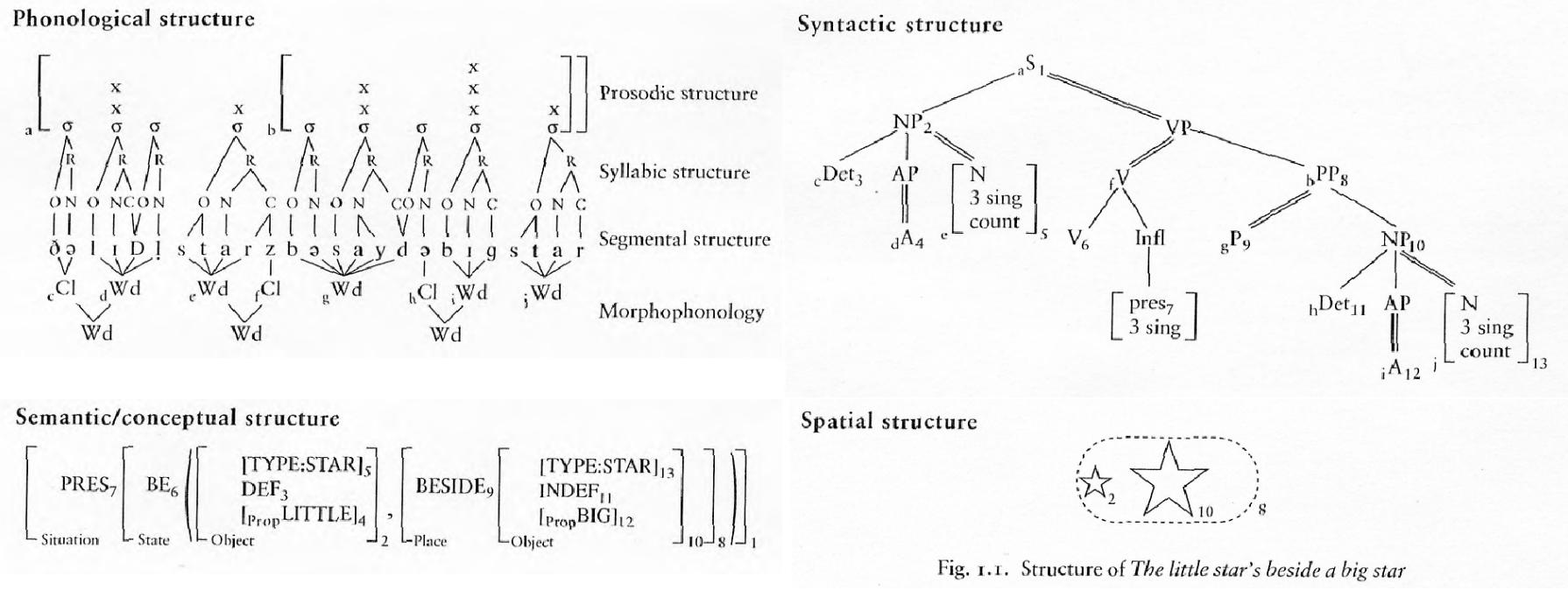


Fig. 1.1. Structure of *The little star’s beside a big star*

Challenge 2: The problem of 2

- How are multiple instances represented?

  “little star” & “big star”

- An inevitable consequence of binding

  (Multiple bindings of the same element)

- This is a problem for localist connectionism

  (If one unit represents “star” how does that unit represent multiple stars simultaneously?)

Challenge 3: The problem of variables

- Construe grammars as templates with variables
- Productivity arises from variables
- How are variables implemented?
- This is a problem for localist connectionism

  Units represent by virtue of their identity (VW cells)

  How can a unit representing a variable represent different content on different occasions?

Challenge 4: Equivalence of WM & LTM binding

- Linguistic/cognitive tasks require structured representations to move between Working Memory and Long Term Memory
- Transfers must occur on a very short timescale

  E.g. experimental instructions: “If a word has an *f* in it, stick out your tongue.”
- This is a problem for connectionist systems that rely on iterative synaptic weight modification

  E.g. backpropagation is implausibly slow

Vector Symbolic Architectures

- Family of connectionist architectures well suited to “symbolic” processing
- High-dimensional vectors (~10,000) of low resolution values (rates or weights)
- Fully distributed – patterns represent (not units)
- Small set of fixed vector operators (MAP)
- Able to represent complex data structures
- Everything (simple or complex) is represented in a fixed-dimensional space
- Viewable as an abstract mathematical model, but can be implemented with spiking neurons

How are VSA used?

VSA vectors and operators as the bricks and mortar of cognitive computation

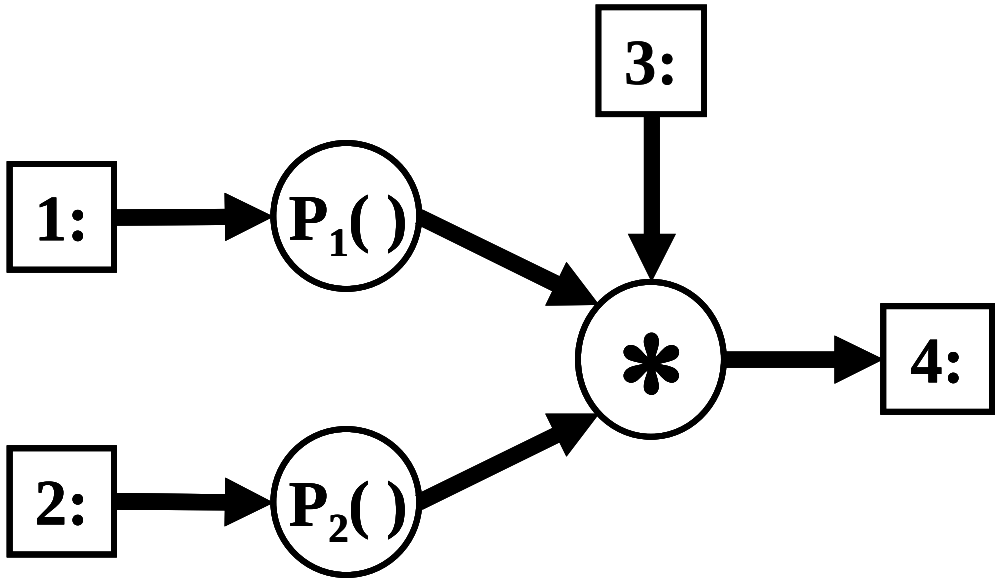
- Multiply ***** bind, query, apply mapping
- Add **+** superpose, add to set
- Permute $\mathbf{P_i( )}$ quote

Design a fixed circuit that calculates the desired result by virtue of it's structure

(Circuits work by virtue of correct design, not because of vast numbers of trainable weights)

Example VSA circuit

Multiset intersection circuit



1: $k_1A + k_2B + k_3C$

2: $k_4A + k_5B + k_6D$

4: $k_1k_4A + k_2k_5B +$ *noise*

Why are VSA useful?

- Support “symbolic” behaviour
- Implementable as massively parallel connectionist systems
- Robust to noise (30% corruption tolerable)
- Graceful degradation
- Operators not learned
- Operators blind to interpretation of vectors

Why are VSA incredibly useful?

Substitution is effectively a primitive operator

The product of two vectors can be applied as a substitution operator:

(A*B) (substitutes **A** for **B** and vice versa)

(A*B) * (A*X) = A*B*A*X = (A*A)*B*X = (B*X)

VSA are really good for problems where the solution depends on substitution (arises from binding & variables)

Shifter problem (Rumelhart & McClelland, 1986)

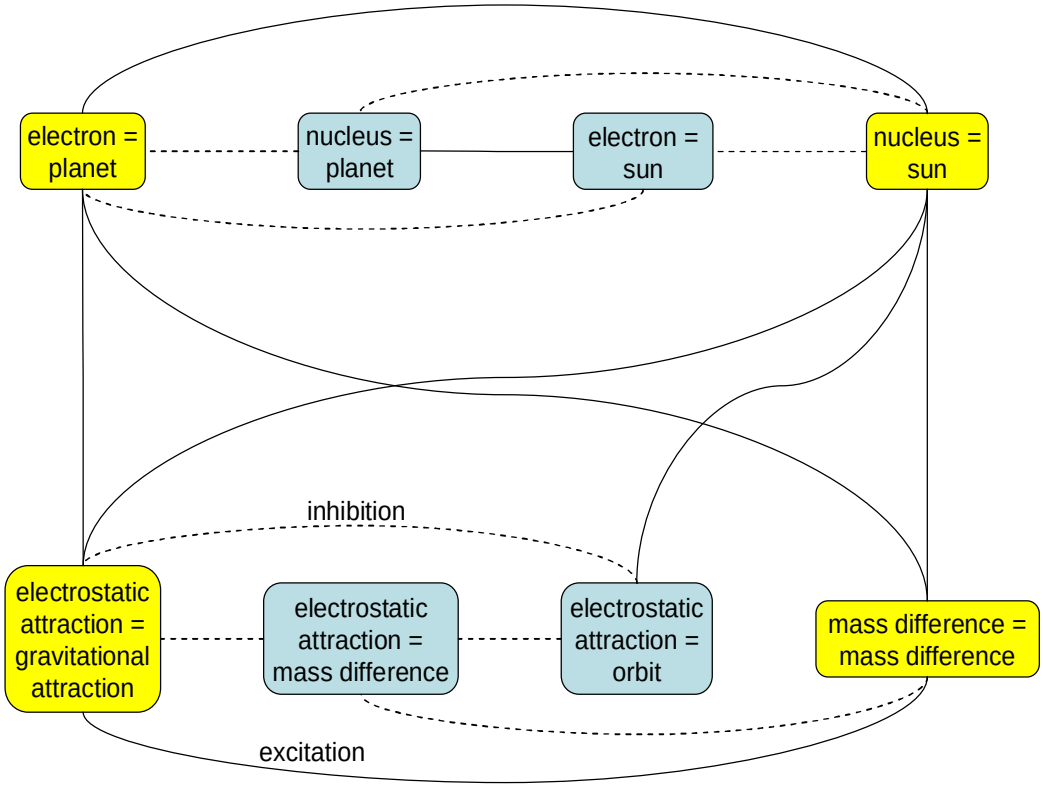
- Boltzmann net: 360,000 trials to train
- VSA circuit: 1 trial to train, generalises to novel inputs

Connectionist models of analogical mapping

- Analogy (structural transfer) is arguably central to cognition
- Analogical mapping is the process of finding structural correspondences
- ACME (Holyoak & Thagard, 1989) is a good example of a localist connectionist model of analogical mapping
- Shows the approach and illustrates the problem with localist connectionist models

ACME mapping network

- Source and target structures are graphs
- Units represent possible vertex mappings
- Unit outputs represent support for mappings
- Connections represent compatibility (based on graphs) between vertex mappings
- Settled state indicates a plausible mapping



Problems with ACME mapping network

- Represents one specific mapping problem
- Localist implementation implies creation/recruitment of new units and connections on the fly (sub-second)
- Process of creating the mapping network appears to be symbolic and serial
  - How to implement mapping network creation process as a connectionist system?
  - Computational cost of mapping network creation process

Replicator equations = formalised ACME

- Analogical mapping = subgraph isomorphism
- Pelillo (1999) used replicator equations to find subgraph isomorphisms
- Replicator equations arise in evolutionary game theory (extensively studied mathematically)
- Represent graphs by numerical matrices
- He mapped graph isomorphism to maximization of a continuous function of those matrices
- Embedded a discrete problem in continuous one
- Implementable as a localist connectionist circuit

Subgraph isomorphism via replicator equations

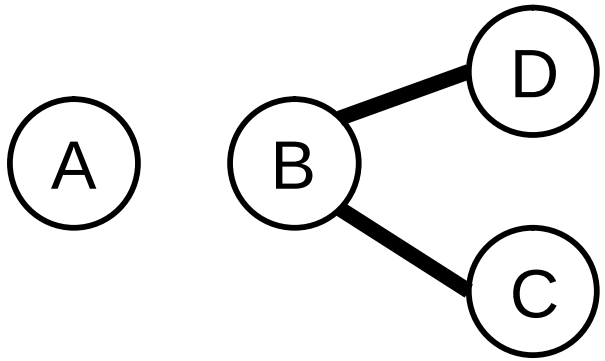
Given

- State vector representing current degree of support for all possible vertex mappings
- Matrix representing compatibility of vertex mappings (constructed from edge information in the graphs to be matched)

Perform

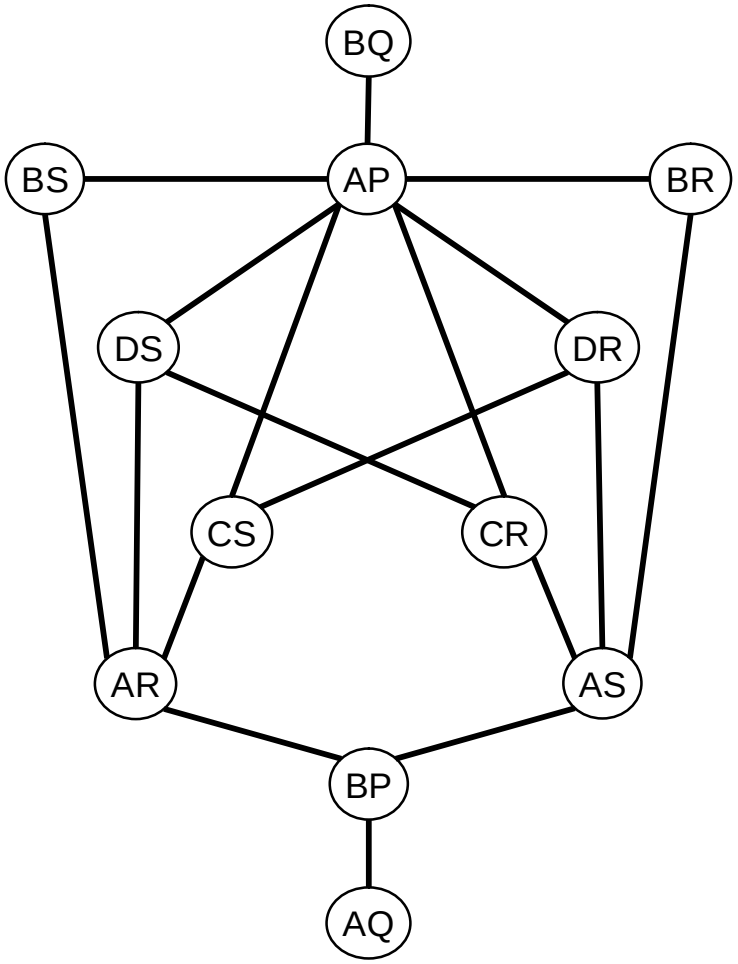
- Vector-matrix multiplication (propagation of support between vertex mappings via edge compatibility information)
- Update the state vector
- Iterate to convergence

Encode the source and target graphs



	A	**B**	**C**	**D**
A	0	0	0	0
B	0	0	1	1
C	0	1	0	0
D	0	1	0	0

Encode the association graph



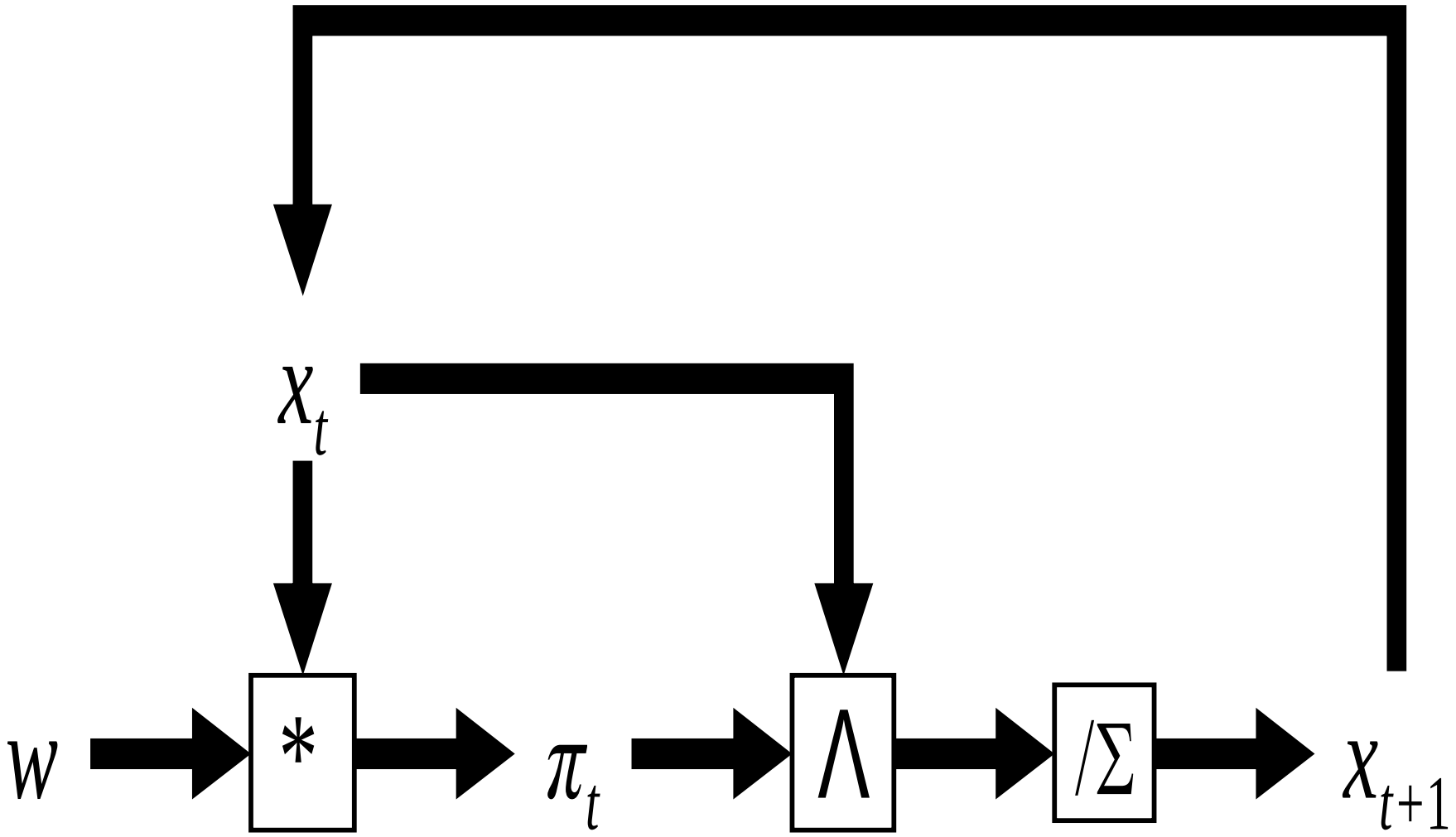
	AP	AQ	AR	AS	BP	BQ	BR	BS	CP	CQ	CR	CS	DP	DQ	DR	DS
AP	0	0	0	0	0	1	1	1	0	1	1	1	0	1	1	1
AQ	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
AR	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1
AS	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0
BP	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
BQ	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
BR	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0
BS	1	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
CP	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1
CQ	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0
CR	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1
CS	1	0	1	0	0	1	0	0	0	0	0	0	1	0	1	0
DP	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0
DQ	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
DR	1	0	0	1	0	1	0	0	1	0	0	1	0	0	0	0
DS	1	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0

The replicator equations

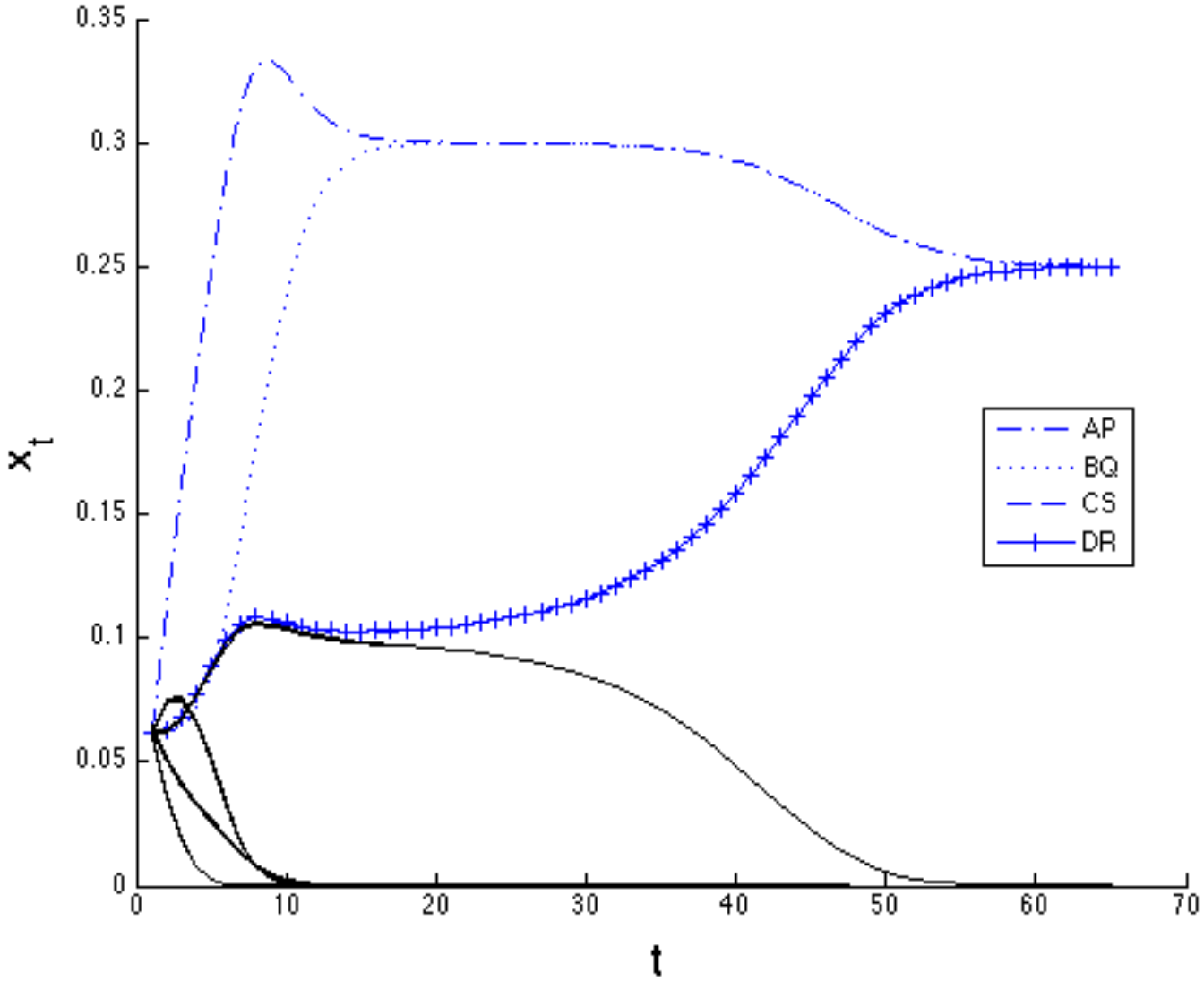
$$\pi_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t)$$

$$x_i(t+1) = \frac{x_i(t)\pi_i(t)}{\sum_{j=1}^{N} x_j(t)\pi_j(t)}$$

Localist architecture



Settling of localist system



Performance of replicator equations

- Competitive with other state of the art heuristic searches in quality of solution
- Generalised to weighted, attributed graphs
- Tested on graphs with > 65,000 vertices
- Typically settle in 100 or so iterations (fast parallel implementation)
- Settling time roughly independent of size
- Settling time depends strongly on how constraining the graphs are

Problems of replicator equations

- Equivalent to a localist connectionist architecture (and ACME mapping network)
- Construction of matrix (ACME connections) is done via matrix arithmetic
- Has the same localist problems as ACME
  - Circuit is specific to one mapping problem (because the labeling of the units is specific to one problem)
  - Construction of the circuit falls outside the capabilities of the circuit
  - Resources required increase rapidly with the size of the structures to be mapped

Subgraph isomorphism via VSA

- Translate the replicator equation algorithm to a VSA implementation
- How to represent the data structures?
- How to operate on the data structures?
- Preserve the replicator equation dynamics!
- Embody the algorithm as a fixed circuit

Represent graphs as VSA vectors

- vertices: randomly chosen vectors *A*, *B*, …
- vertex sets : (*A* + *B* + *C* + *D*).
- edges: *B***C*, B*D, …
- edge sets: (*B***C* + B*D)

- vertex mappings: A*P, B*Q, ...
- state vector: k_1A*P + k_2A*Q …
- compatibility: A*P*B*Q + A*P*B*R + …

Construct the initial values

- initial state vector (product of vertex sets)

$$x=(A+B+C+D)*(P+Q+R+S)$$
$$=A*P+A*Q+\ldots+B*P+B*Q+\ldots+D*R+D*S$$

- compatibility vector (product of edge sets)

$$w=(B*C+B*D)*(Q*R+Q*S)+$$
$$(A*B+A*C+A*D+C*D)*(P*Q+P*R+P*S+R*S)$$
$$=B*C*Q*R+B*C*Q*S+B*D*Q*R+B*D*Q*S$$
$$+A*B*P*Q+A*B*P*R+\ldots+A*B*R*S$$
$$+A*C*P*Q+A*C*P*R+\ldots+A*C*R*S+\ldots+C*D*R*S$$

Each needs only a single constant time operation

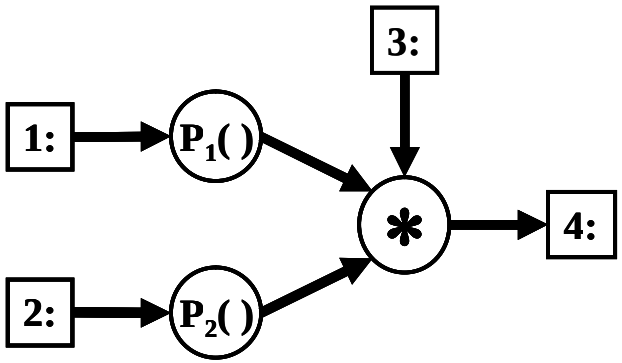
Evidence propagation and update

- evidence propagation

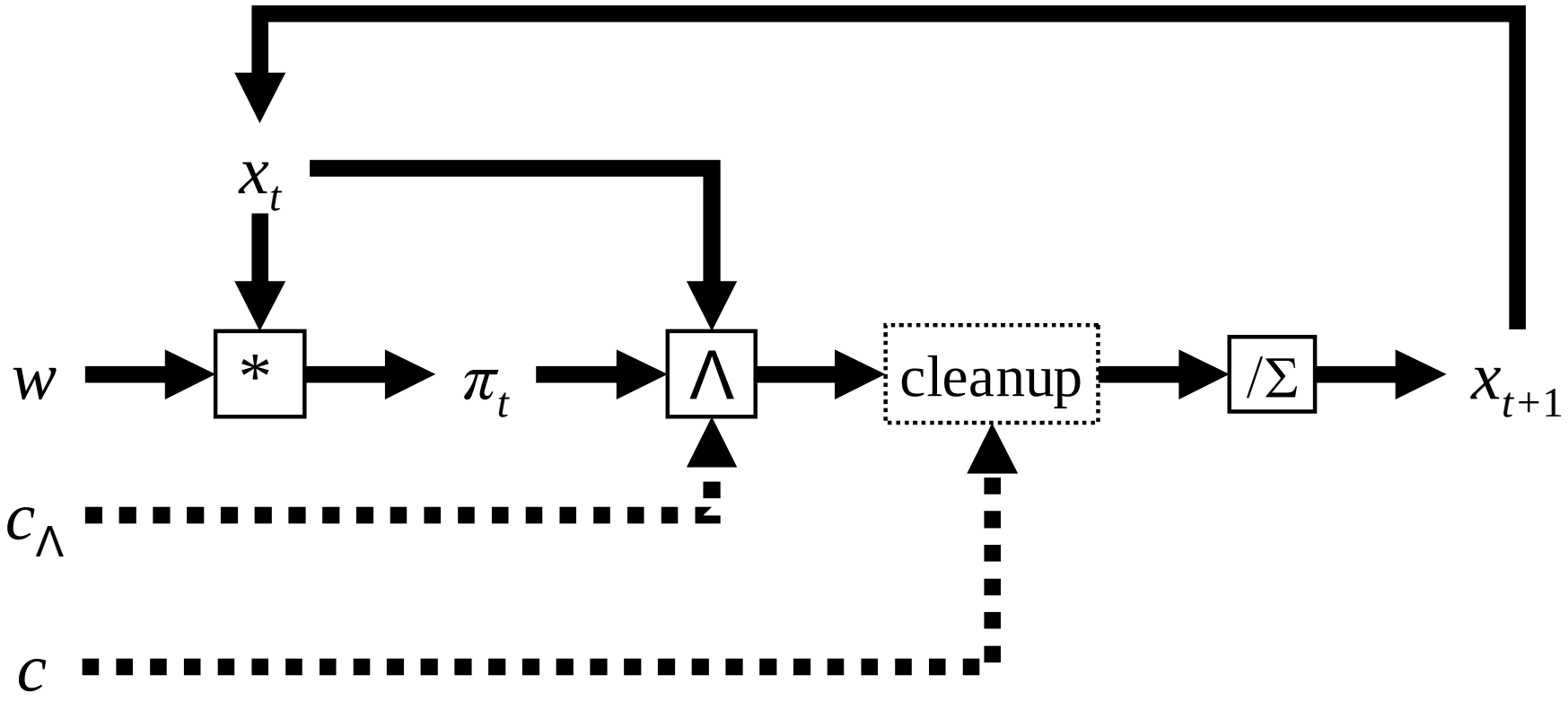
  Product of state and compatibility vectors (one constant-time operation)

- update operation

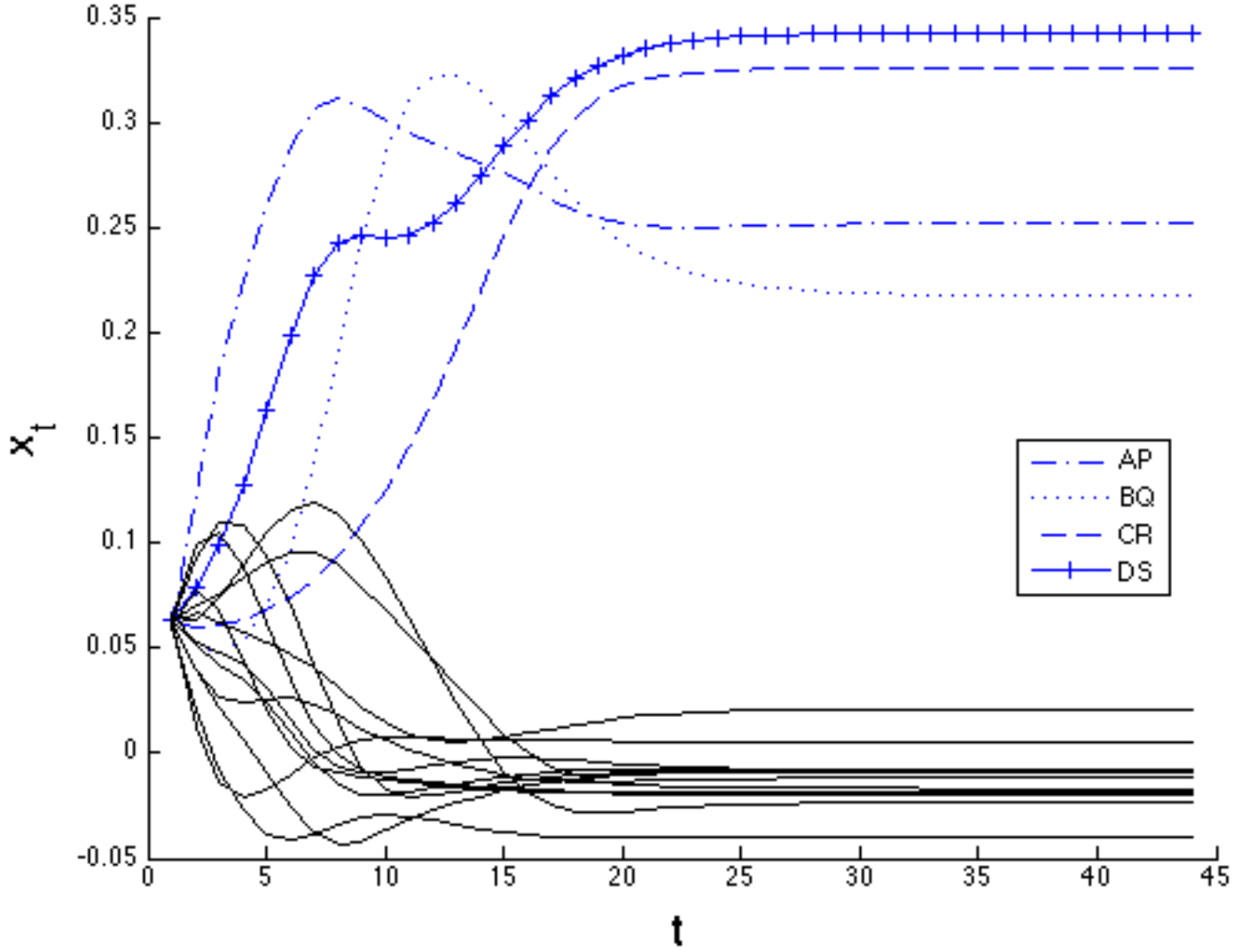
  Apply previously introduced intersection circuit (one constant-time operation)



Fully distributed architecture



Settling of distributed system



Salient points

- It works! (Only tested on a few graphs so far)
- Hardware is fixed. (No need to modify it for each graph matching problem.)
- Graphs to be matched are loaded into the circuit as patterns of activation
- Those patterns are calculated by constant time holistic computation from the vectors representing the graphs.

- Proof of concept for the practical feasibility of implementing complex symbolic computation in a connectionist system by using VSA