Analogical mapping with Vector Symbolic Architectures

Ross W. Gayler
La Trobe University
r.gayler@gmail.com

Simon D. Levy
Washington and Lee University
levys@wlu.edu

Mathematical psychology?

Approached as an engineering exercise (AI)

- Functional performance
- Practical feasibility

Design a compositional memory system

Recognise a novel compound stimulus

- As a novel structuring
- Of familiar components
- Via familiar relations

VSA basis has Math Psych antecedents

Outline

- Analogical mapping as subgraph isomorphism
- ACME localist connectionist model
- Replicator equations as localist model
- Introduction to Vector Symbolic Architectures
- Replicator equations via VSA

Analogical mapping

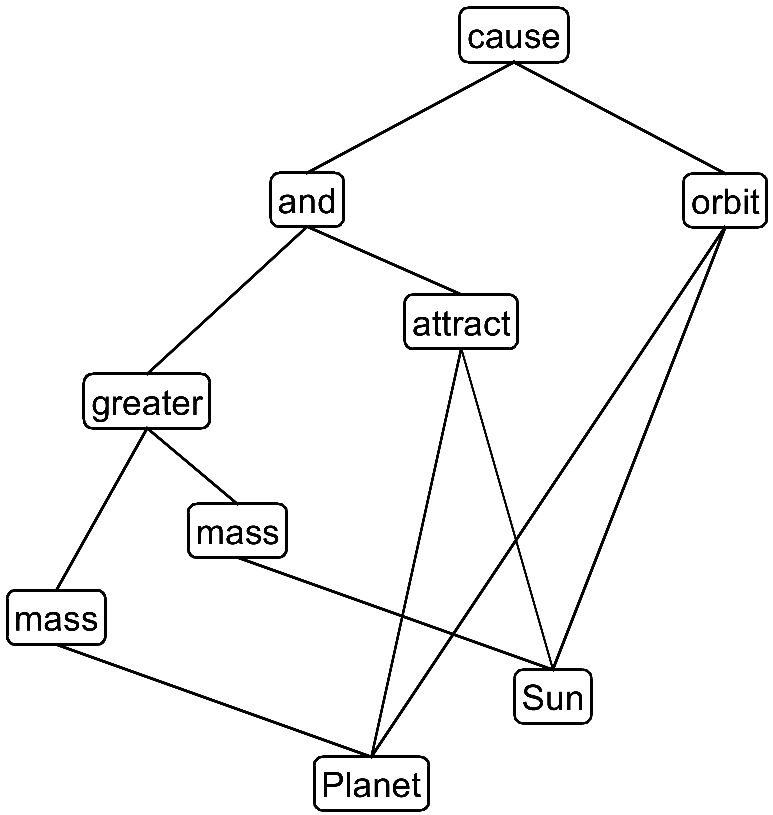
- Structural alignment of conceptual structures (source & target)
- Unmatched structure can be transferred from source to target as heuristic inference
- Rutherford analogy: solar system ↔ atom
- Source & target encoded as sets of facts
- Sets of facts represented as graphs
- Mapping is subgraph isomorphism

Encoded as facts

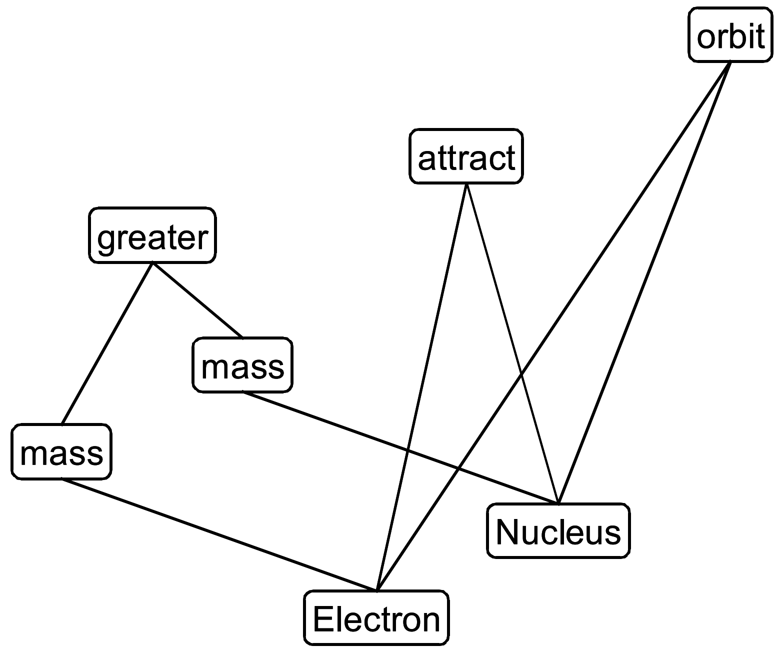
Solar system (Source)	Atom (Target)
S1: mass(Sun)	T1: mass(Nucleus)
S2: mass(Planet)	T2: mass(Electron)
S3: greater(S1, S2)	T3: greater(T1, T2)
S4: attract(Sun, Planet)	T4: attract(Nucleus, Electron)
S5: and(S3, S4)	
S6: orbit(Planet, Sun)	T6: orbit(Electron, Nucleus)
S7: cause(S5, S6)	

Encoded as graphs

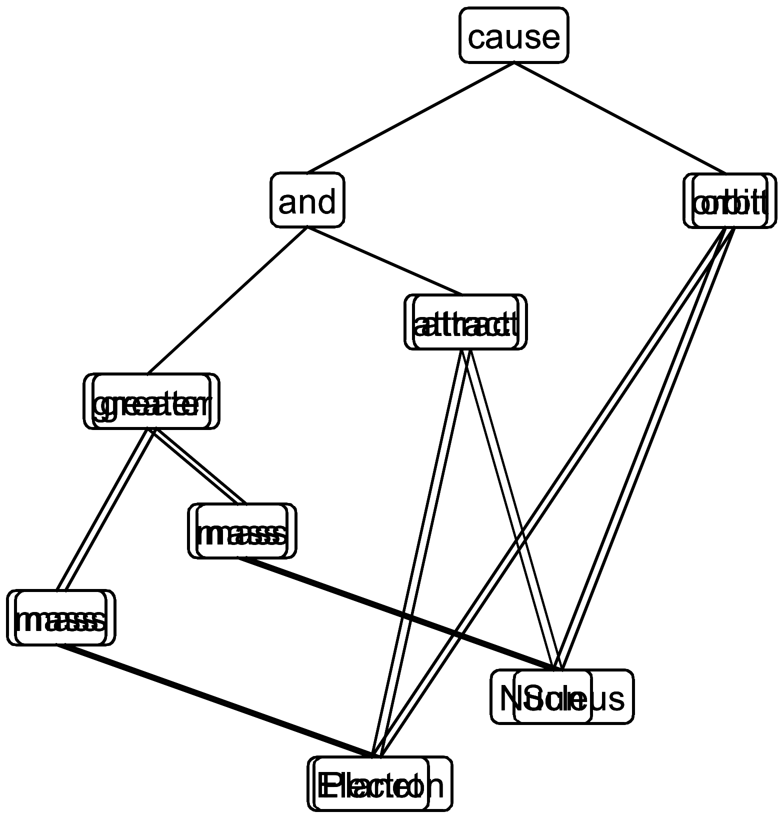
Solar system



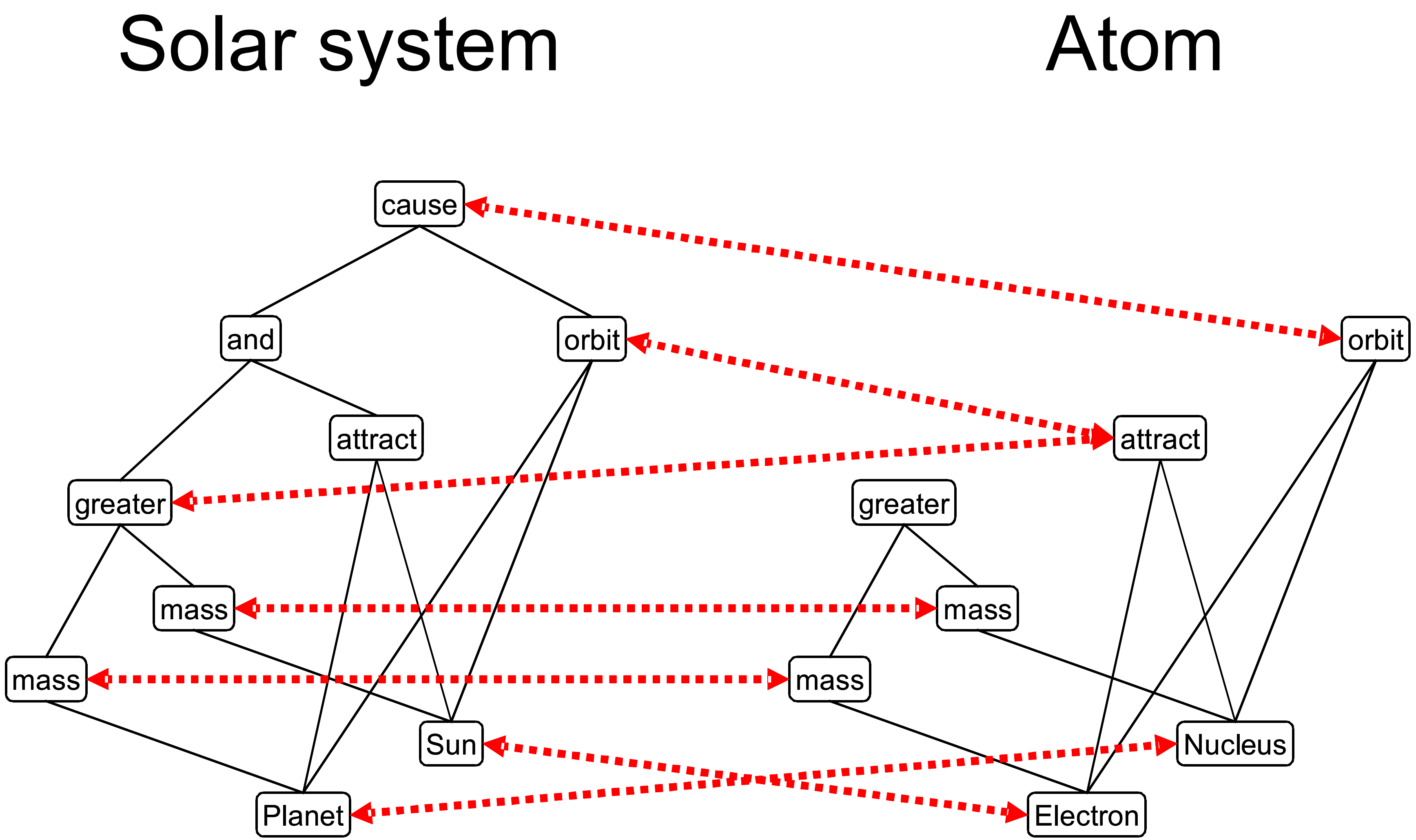
Atom



Structural consistency

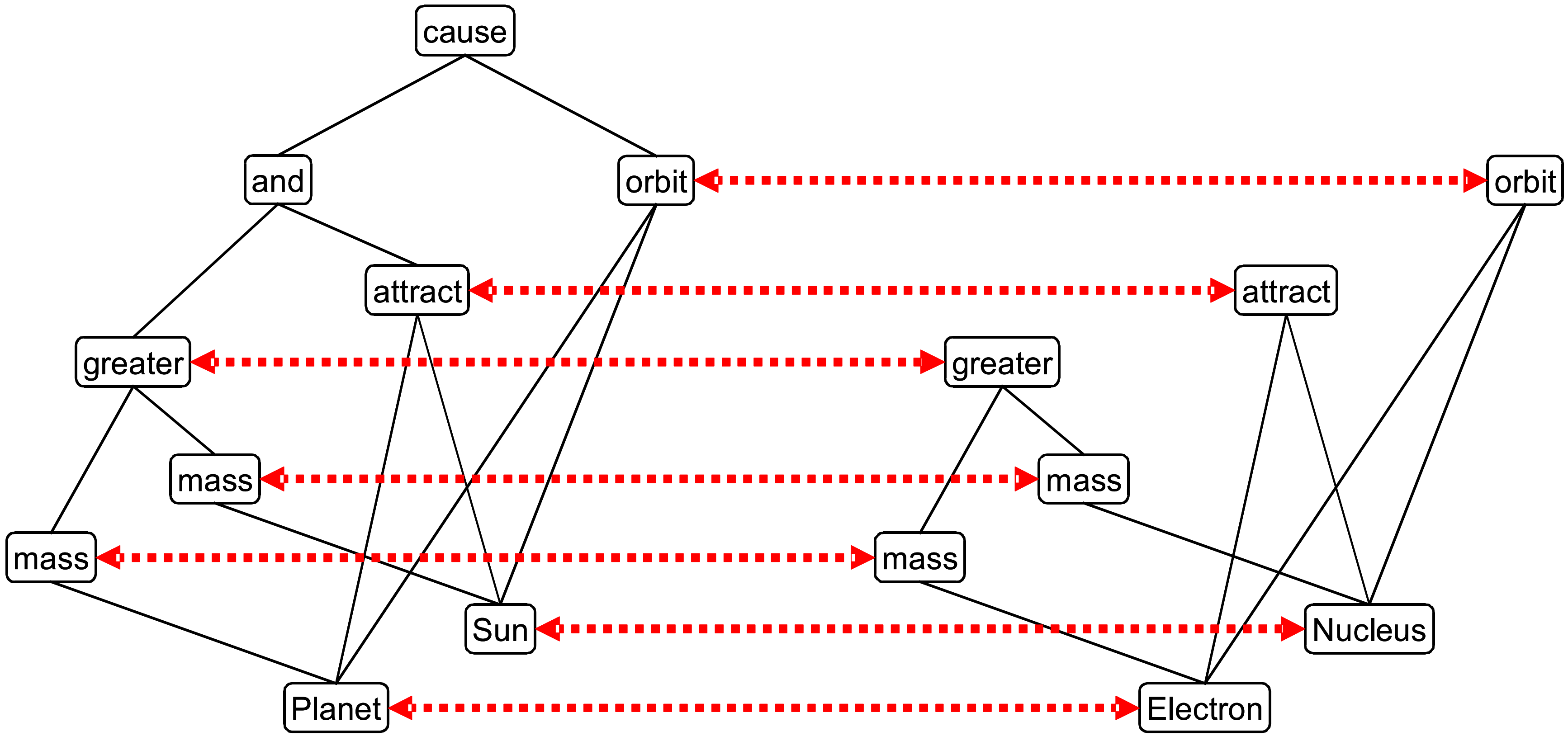


Bad mapping


Solar system
Atom
cause
and
orbit
attract
greater
mass
mass
Sun
Planet
orbit
attract
greater
mass
mass
Nucleus
Electron

Good mapping

Solar system | Atom



How to choose the mapping?

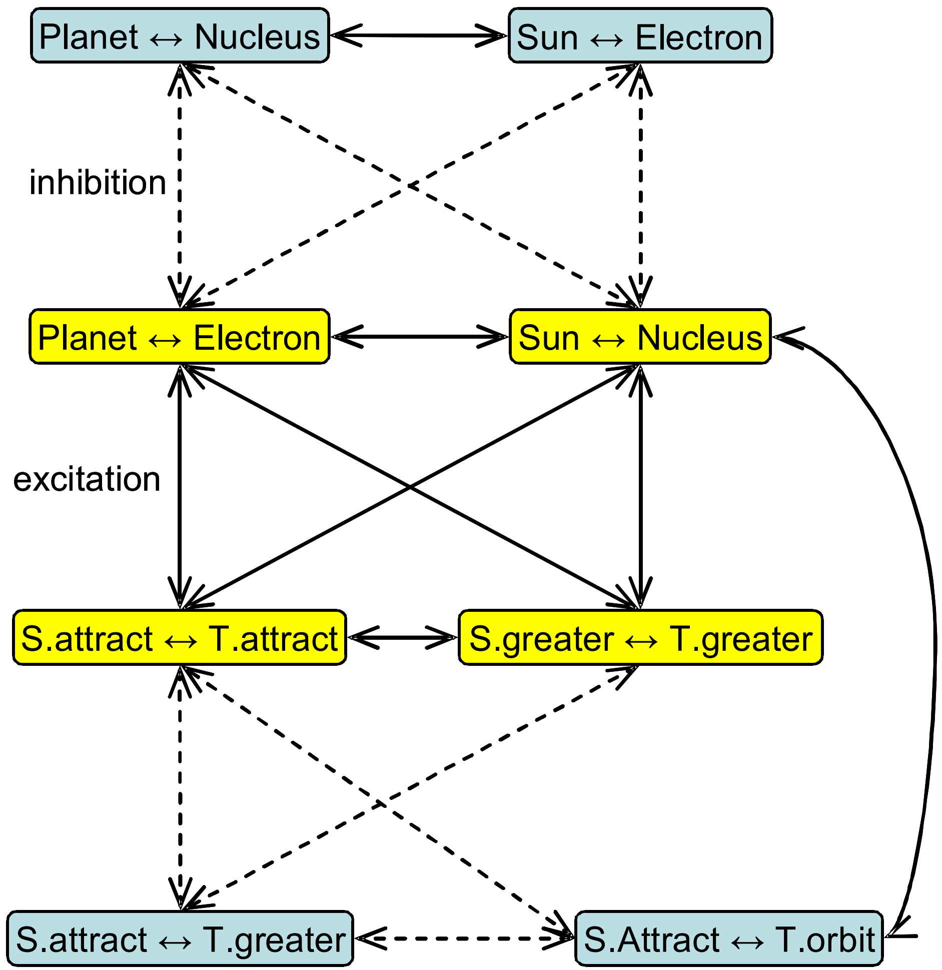
- What to encode?
- How to encode it as a graph
- Labels?
- Weights?
- Structural consistency = subgraph isomorphism
- Mechanism to find isomorphisms

ACME model of analogical mapping

- Holyoak & Thagard (1989)
- ACME is a good example of a localist connectionist model of analogical mapping
- Shows the approach and illustrates the problem with localist connectionist models

Mapping network (partial)

- Units represent possible vertex mappings (accumulators)
- Unit outputs represent support for mappings
- Connections represent compatibility between vertex mappings
- Network settles to a state representing a mapping between the graphs



Problems with mapping network as a practical mechanism

- Poor scaling: ~k^2 units , ~k^4 connections
- Represents one specific mapping problem
- Localist implementation implies creation or 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 as a connectionist system?
  – Computational cost of process

Subgraph isomorphism via replicator equations

- Won’t go directly from ACME to VSA
- Go via replicator equations

  (Interpretable as a formalised version of the ACME mapping network)
- Traditional approach to subgraph isomorphism is heuristic discrete search
- Replicator equations are a continuous maximisation approach

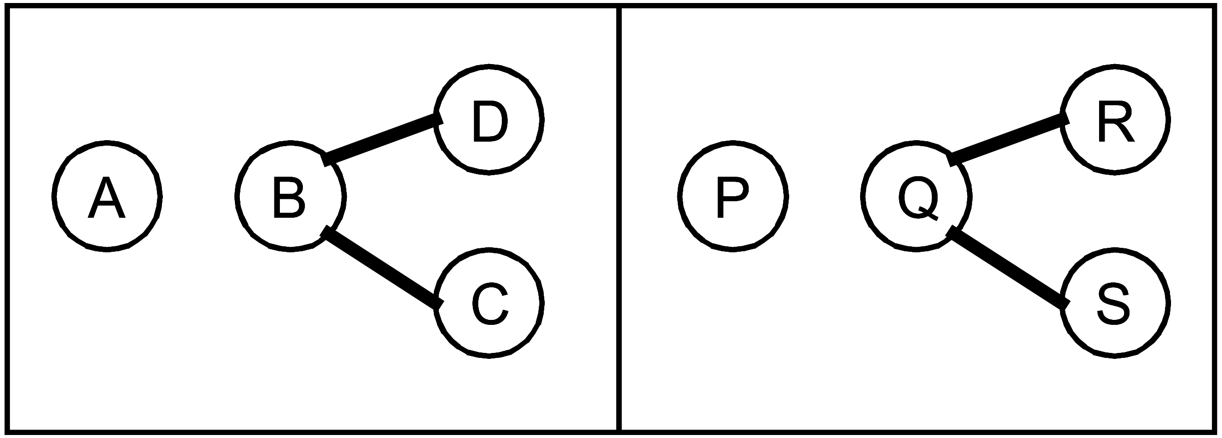
Origin of subgraph isomorphism via replicator equations

- Replicator equations arise in evolutionary game theory (extensively studied mathematically)
- Applied to isomorphism by Pelillo (1999)
- Can represent graphs by numerical matrices
- He mapped graph isomorphism to maximization of a continuous function of those matrices
- Embedded a discrete problem in continuous
- This is a heuristic solution (may not find the maximal isomorphism)

Subgraph isomorphism via replicator equations

- Vector representing support for all vertex mappings (vector of accumulators)
- Matrix representing compatibility of vertex mappings (constructed from edge information in the graphs to be matched)
- Vector-matrix multiplication is propagation of support between vertex mappings via compatibility information
- Update of the vertex mapping vector

Take two graphs

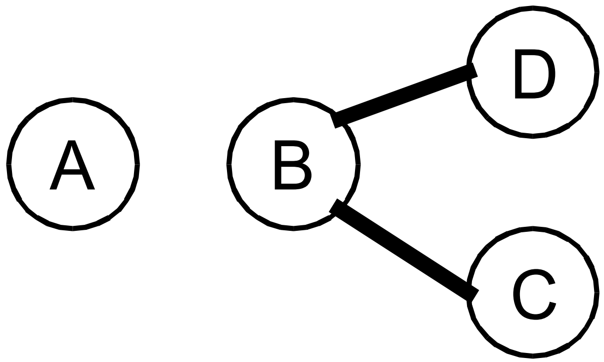


Possible solutions:

{A↔P, B↔Q, C↔R, D↔S}

{A↔P, B↔Q, C↔S, D↔R}

Encode the 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

Association graph

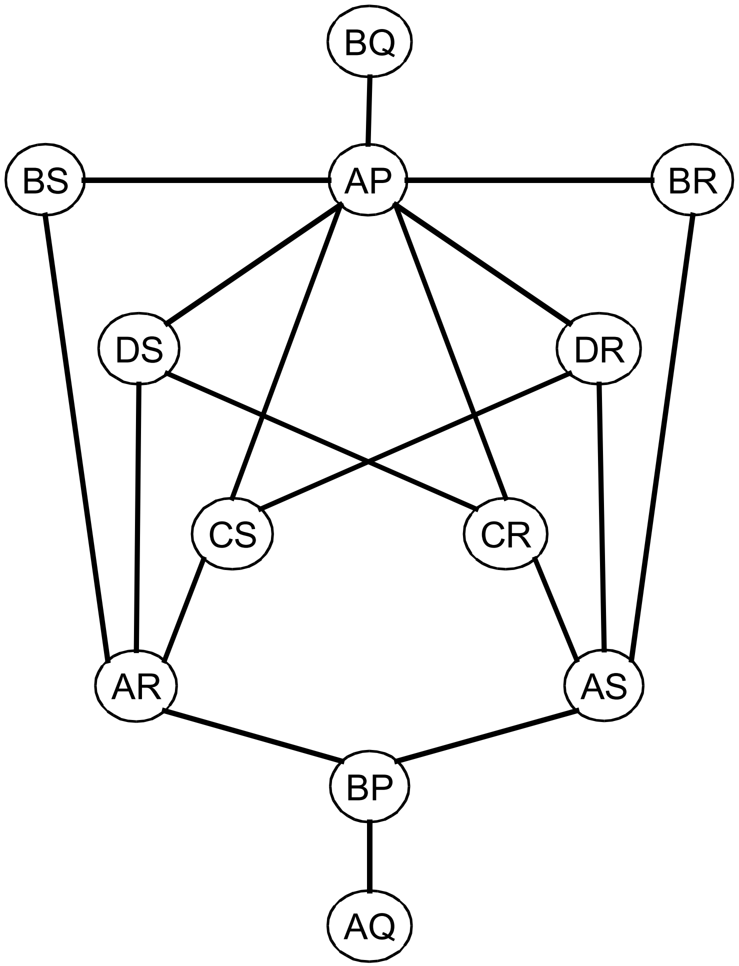
- Represents a product of the two graphs
- Vertices represent base vertex mappings
- Edges show local structural consistency

  g1:A▬B & g2:P▬Q → assoc:AP▬BQ
- Edges interpretable as inference rules

  Edge AP▬BQ

  Support for A↔P implies support for B↔Q

Association graph & matrix



	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

Replicator equations

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

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

π_i Evidence for vertex mapping *i* (x_i)

x(t) Prior support for vertex mappings

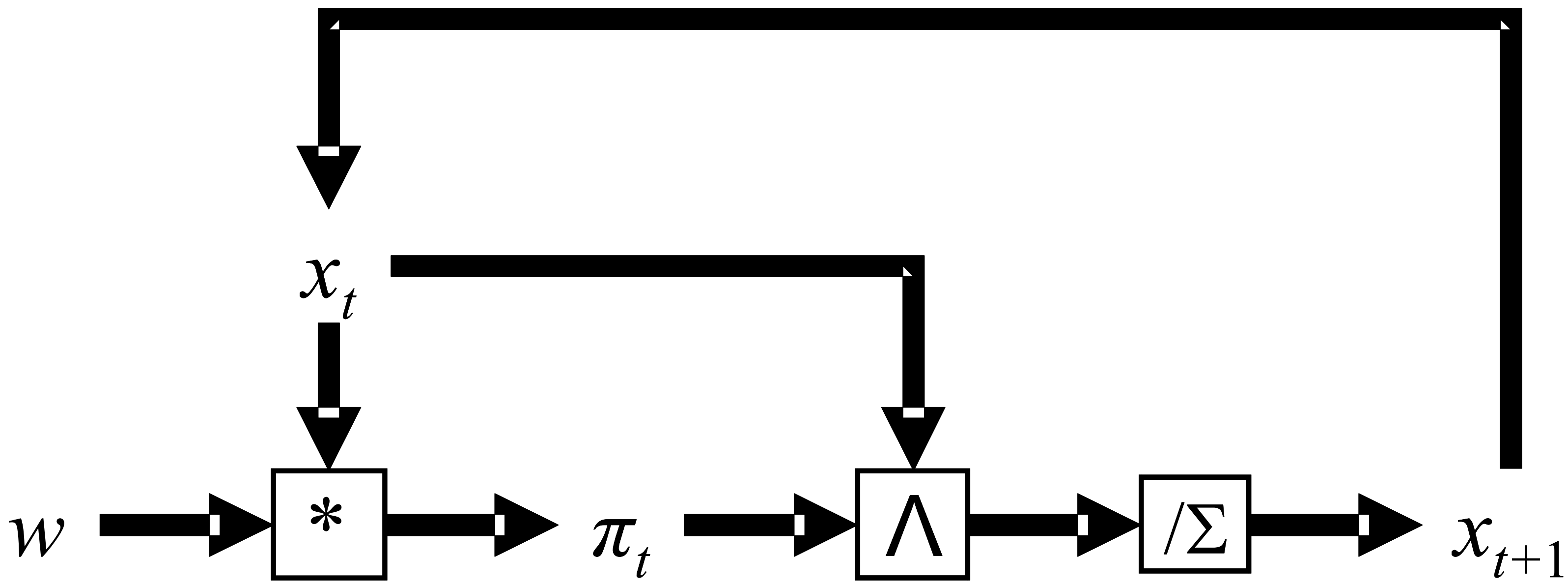
x(t+1) Posterior support for vertex mappings

W Inference rules (Association matrix)

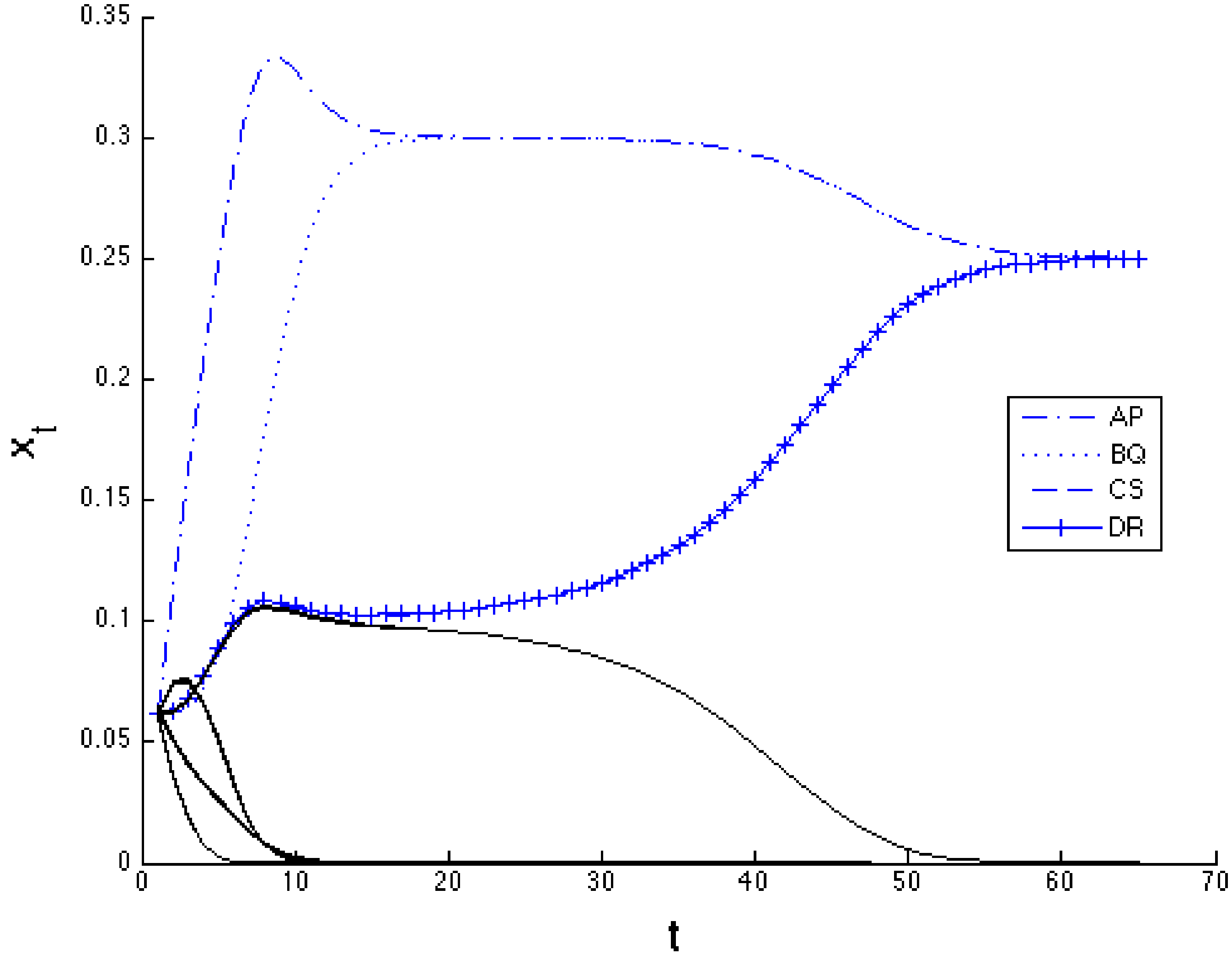
Direct connection with Bayesian inference (Harper, 2010)

Localist architecture

Each element of a vector or matrix may be implemented as a connectionist unit



Settling of localist system



Performance of replicator equations

- Tested on graphs with > 65,000 vertices
- Competitive with state of the art search
- Typically settle in < 100 iterations
- Fast parallel implementation possible
- Settling time ~ independent of graph size
- Settling time depends strongly on how constraining the graphs are
- Generalized to weighted, attributed graphs

Problems with replicator equations

- Equivalent to a localist connectionist architecture (and ACME mapping network)
- Has the same localist problems as ACME
  - Scales poorly (vector ~ k^2, matrix ~ k^4)
  - Specific to <u>one</u> problem

In localist systems functional elements are identified with physical resources (units and connections)

What are VSA?

- Family of connectionist architectures well suited to “symbolic” processing
- High-dimensional vectors (~10,000) of low resolution values
- Able to represent complex data structures
- Everything (simple or complex) is represented in a fixed-dimensional space
- Small set of fixed vector operators (MAP)

Math Psych antecedents

- Longuet-Higgins (1968) Holographic memory
- Willshaw (1971) Distributed associative memory
- Poggio (1973) Convolution and correlation algebras
- Murdock (1982) TODAM
- Metcalfe (1982) CHARM
- Pike (1984) convolution and matrix memories

Connectionist antecedents

- Smolensky (1990) Tensor product binding
- Plate (1991) Holographic Reduced Representations
- Kanerva (1996) Binary Spatter Codes
- Gayler (1998) Multiply Add Permute codes
- Rachkovskij (2001) Context Dependent Thinning

Connectionists tend to be more concerned with computational capabilities; math psychs with consequences of simple representations

How are VSA used?

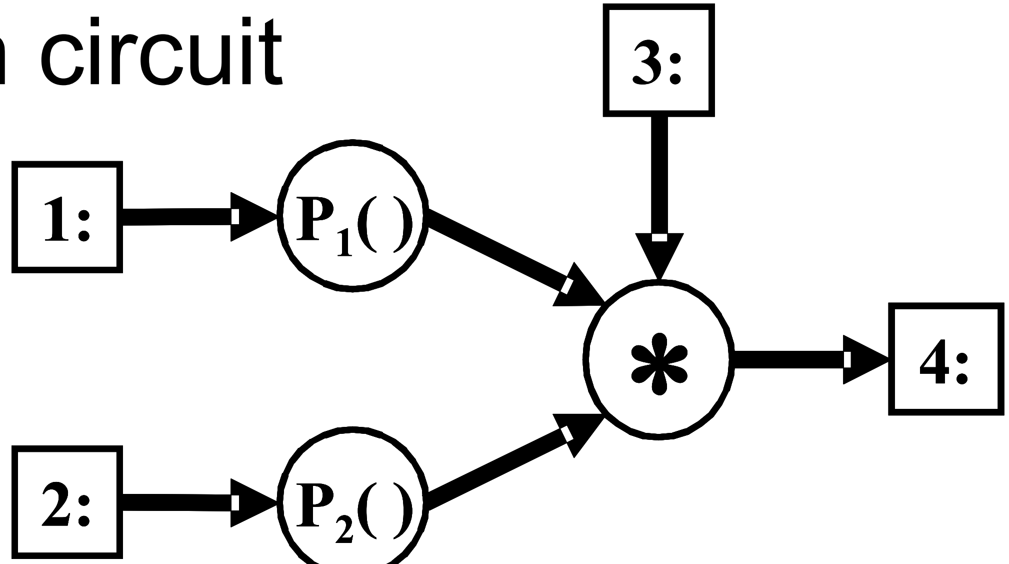
VSA vectors and operators as the bricks and mortar of computational circuits

- 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

How are VSA used?

Multiset intersection circuit



1: $k_1A + \quad k_2B + k_3C$

2: $k_4A + \quad k_5B + k_6D$

4: $k_1k_4A + k_2k_5B +$ *noise*

Why are VSA good?

- Interesting properties of high-dimensional vector spaces (Kanerva, 2009)
- Implementable as realistic connectionist systems (Eliasmith & Anderson, 2003)
- Robust to noise (30% corruption tolerable)
- Graceful degradation
- Operators not learned
- Operators blind to interpretation of vectors

Why are VSA good?

Substitution is effectively a primitive operator

- Substitution is central to symbolic computing

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)

Subgraph isomorphism via VSA

Translate the replicator equation algorithm to a VSA implementation

- How to represent the data structures?
- How to operate on 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

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

  (dynamic construction of accumulators)

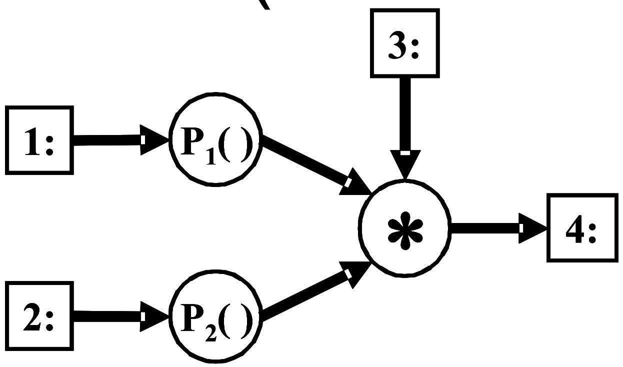
- compatibility vector

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

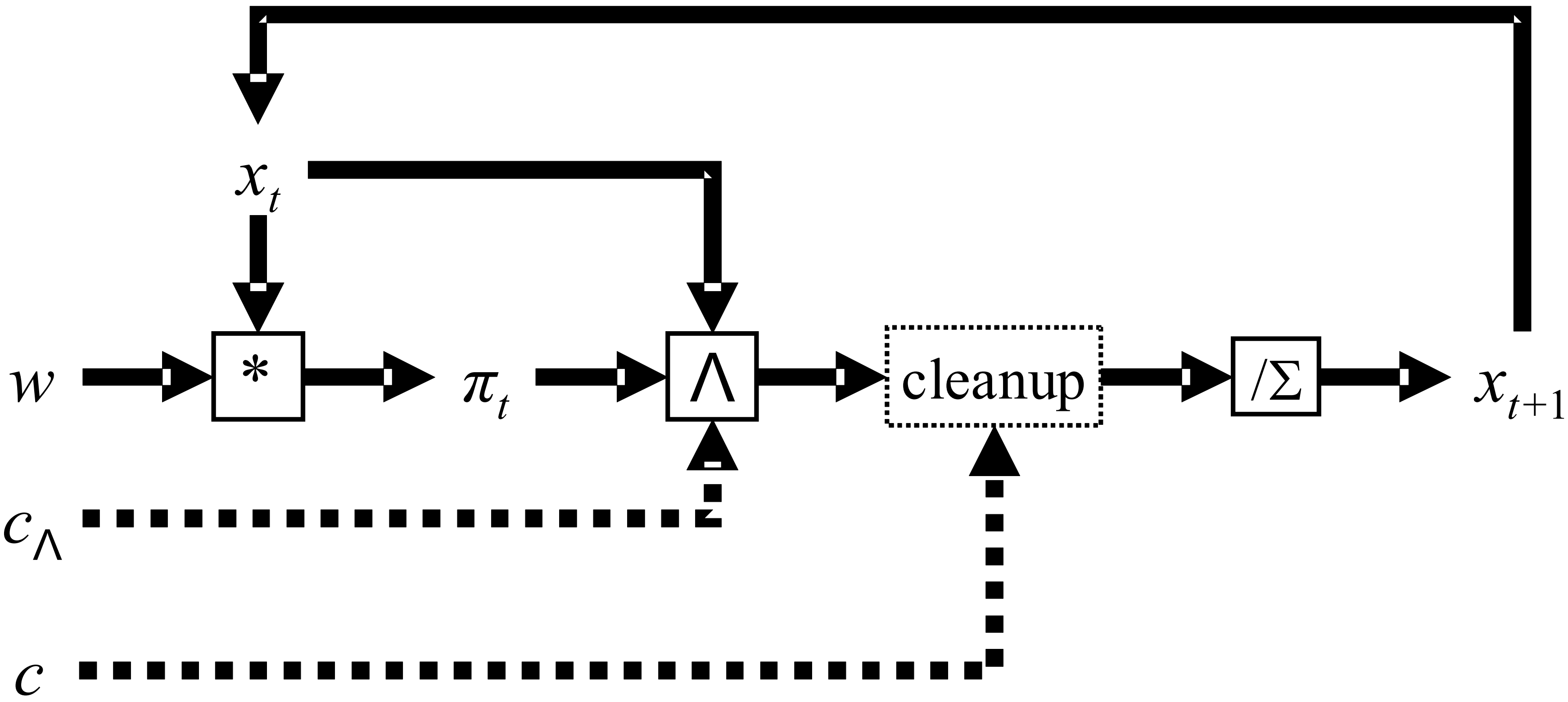
Each requires only a constant time operation (algebraic parallelism)

Evidence propagation & update

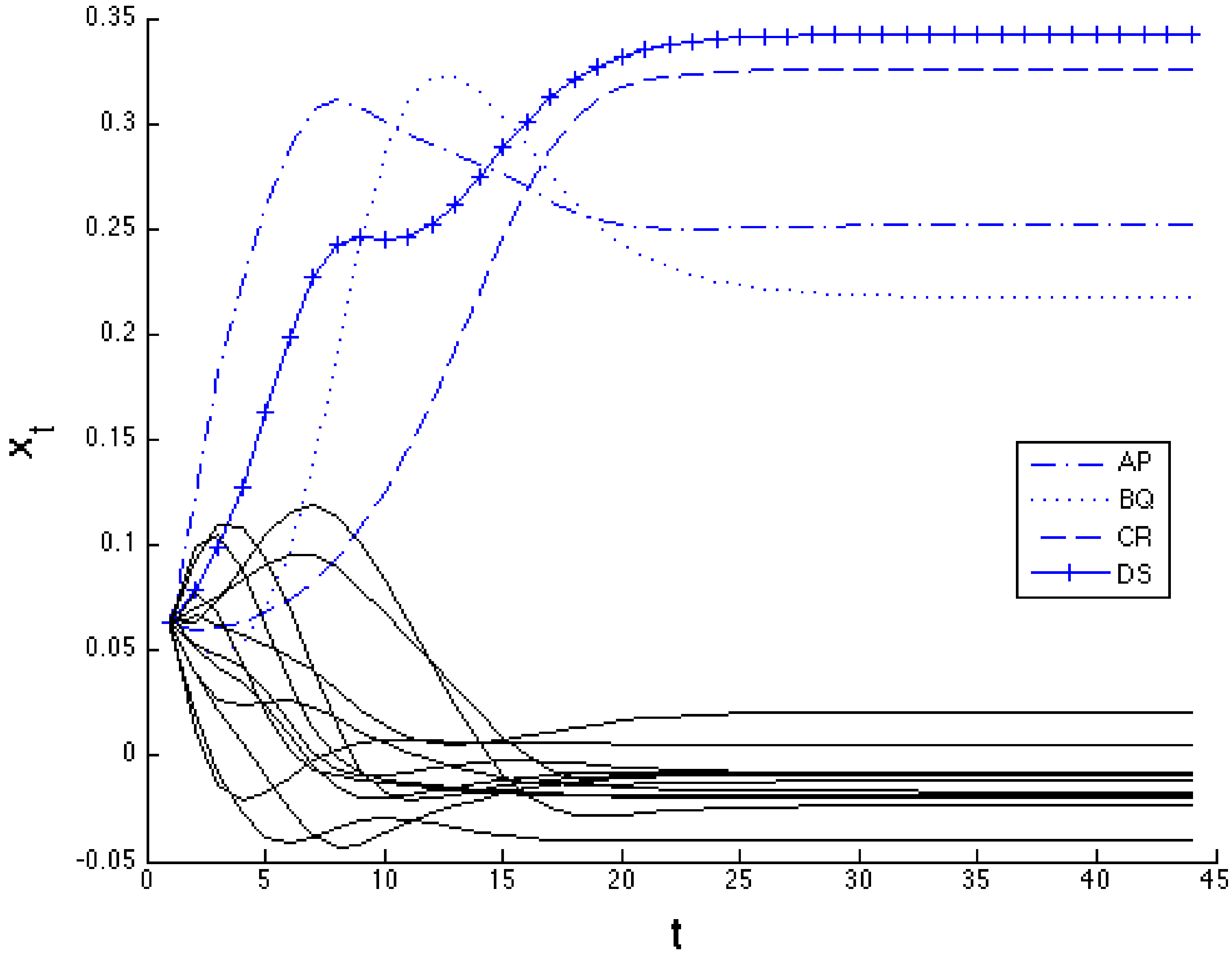
- evidence propagation
  - Product of state and compatibility vectors (constant-time operation)
- update operation
  - Apply previously introduced multiset intersection circuit (constant-time operation)



Fully distributed architecture



Settling of distributed system



Salient points

- It works! (Only tested on a few graphs)
- Hardware is fixed
- Problem is loaded as patterns of activation
- Those patterns are calculated holistically from the graph vector representations
- Many aspects of the approach seem amenable to mathematical exploration