Four facts about Tensor Product Representations (TPRs)

Paul Smolensky

Cognitive Science Dept
Johns Hopkins University

Deep Learning Group
Microsoft Research, Redmond
(Fall 2015)

NIPS Workshop — Cognitive Computation: Integrating Neural and Symbolic Approaches

Dec. 12, 2015

Neural-symbolic integration

What's the proper relationship for such an integration?

Have: a coarse-grained 'macro' theory — **symbolic computation**

- which has many successes
- is running up against its limits

Cognition, linguistics, neuroscience

states of the mind are collections of referring symbols assembled into combinatorial structures that are governed by the variable-containing well-formedness constraints of a 'grammar'

Neural-symbolic integration

What's the proper relationship for such an integration?

Have: a coarse-grained 'macro' theory — **symbolic computation**

- which has many successes
- is running up against its limits

Want: a more fine-grained 'micro' theory — **neural computation**

- which will enable us to surpass the limits of the existing theory
- without losing all of its successes

Emulate physics: mechanics

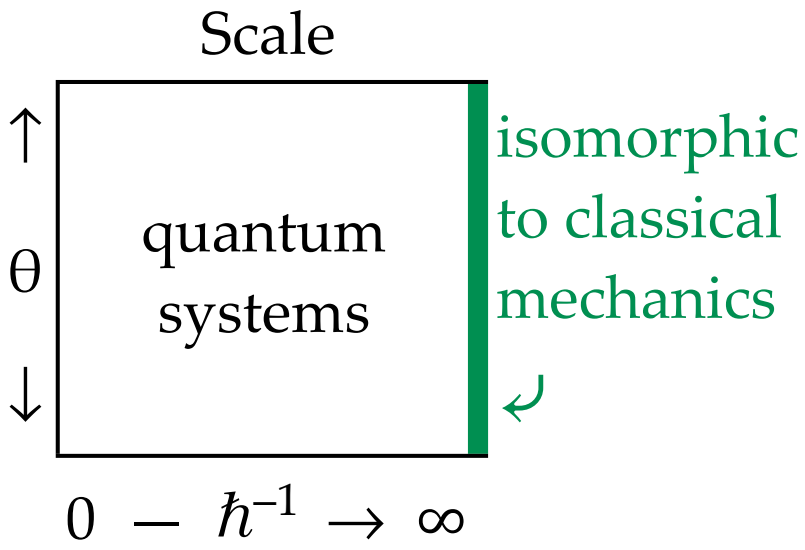
state = vector $\in \mathbb{R}^n$

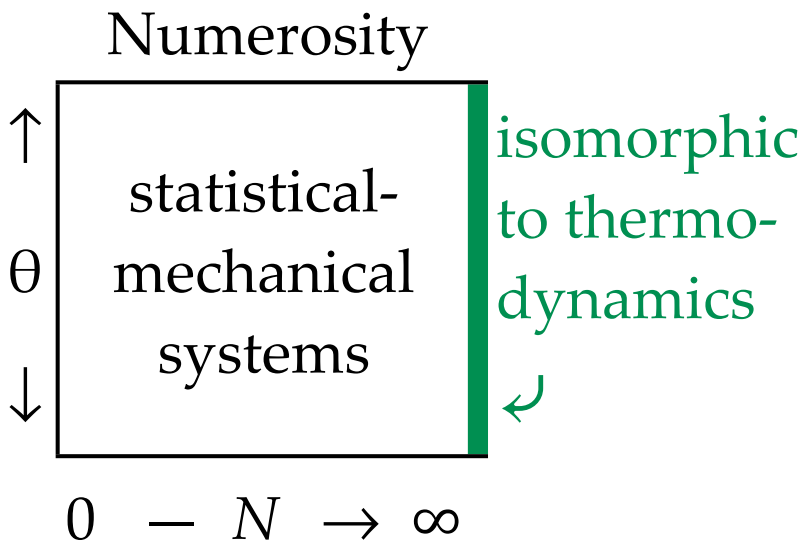
state transitions: continuous (esp. quasi-linear) dynamical system

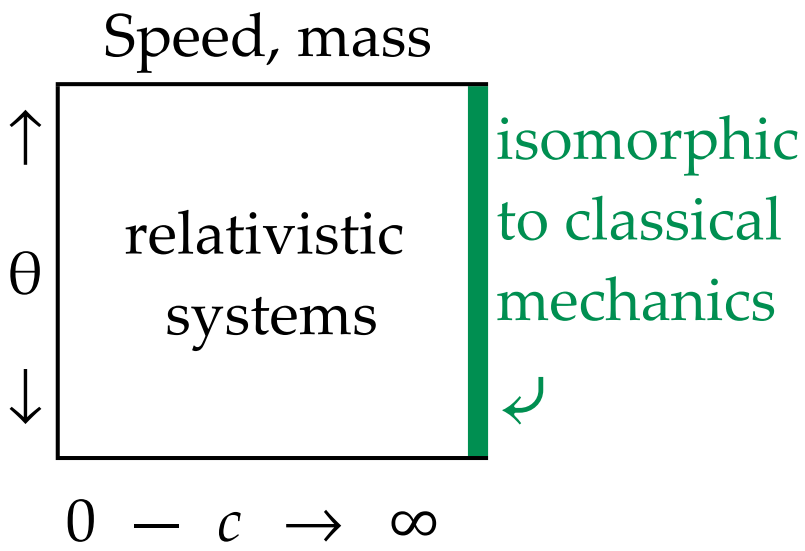
semantics: distributed representations (activity vectors, not units, have conceptual interpretation)

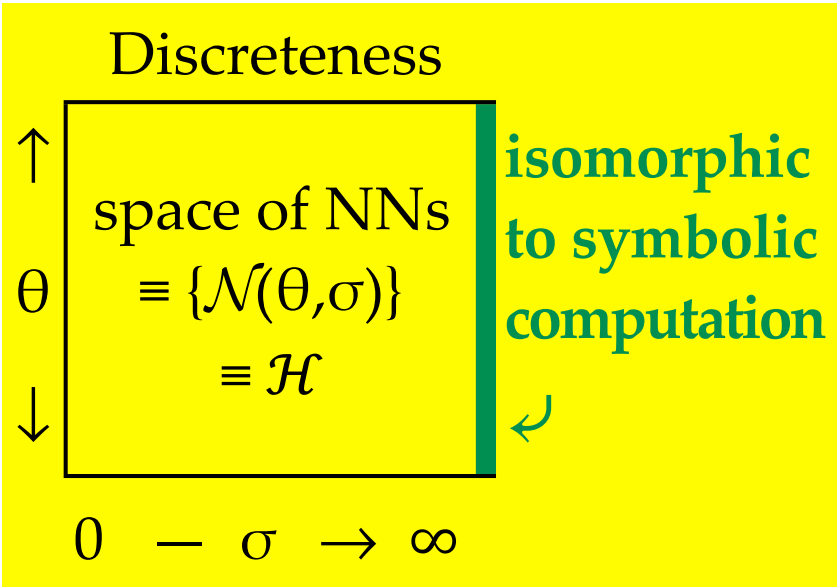
there ***is*** a challenging gap between symbolic & neural computation

From mechanics to cognition







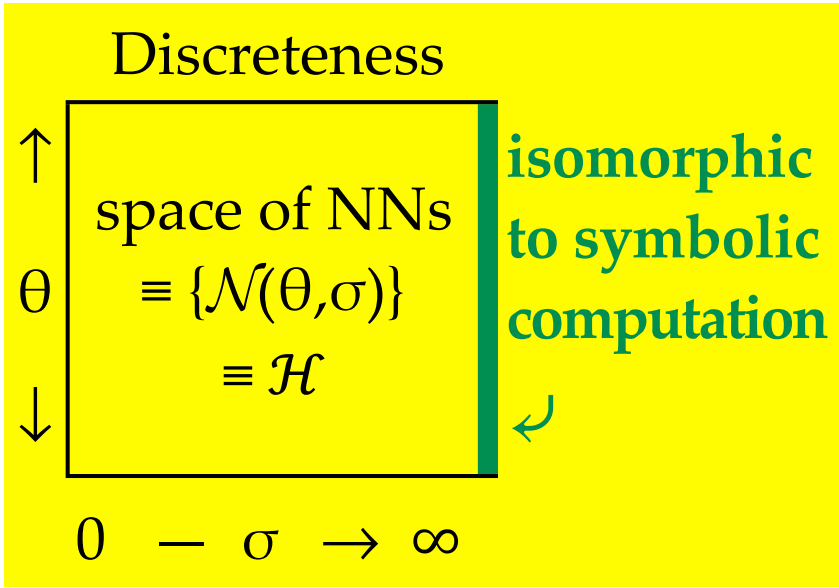


From mechanics to cognition

Want: a general NN architecture $\mathcal{H}$ which

- converges to symbolic computation in the limit of some discreteness hyperparameter vector $\sigma \to \infty$, e.g.:
  - ➤ dimensionality of activation vectors ($|\mathrm{NN}|$) encoding symbols $\to \infty$
  - ➤ similarity^{-1} of activation vectors encoding symbols structures $\to \infty$
  - ➤ non-randomness $T^{-1} \to \infty$
  - ➤ processing time $\to \infty$
- can improve upon symbolic cognitive theory
  - ➤ w.r.t. 'macro data'
  - ➤ w.r.t. 'micro data'

☞ As a hypothesis space $\mathcal{H}$ for learning, can always do at least as well as symbolic theory: can choose $\sigma \to \infty$



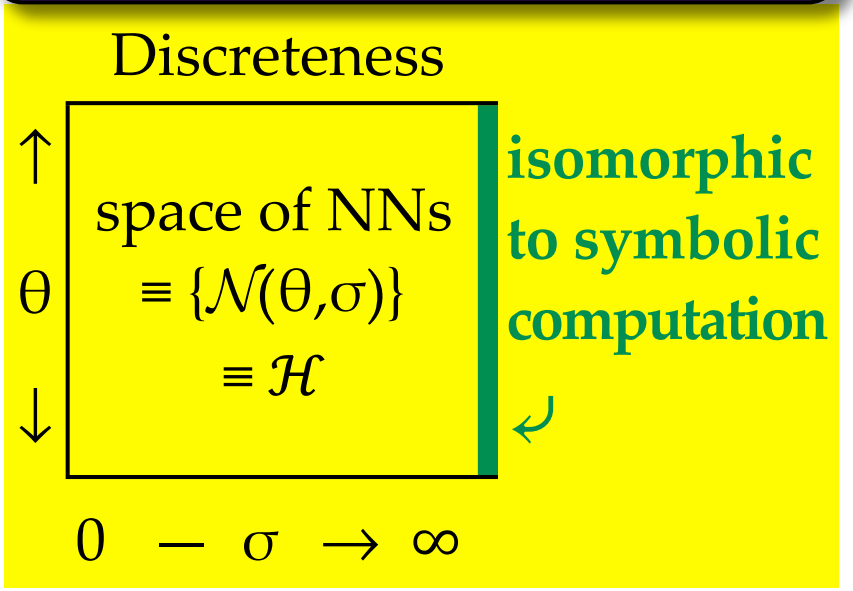
From mechanics to cognition

Want: a general NN architecture $\mathcal{H}$ which

- converges to symbolic computation in the limit of some discreteness hyperparameter vector $\sigma \to \infty$, e.g.:
  - dimensionality of activation $\to \infty$
  - similarity^{-1} of activation ve $\to \infty$
  - non-randomness $T^{-1} \to \infty$
  - processing time $\to \infty$
- can improve upon symbolic cognitive theory
  - w.r.t. 'macro data'
  - w.r.t. 'micro data'

☞ As a hypothesis space $\mathcal{H}$ for learning, can always do at least as well as symbolic theory: can choose $\sigma \to \infty$

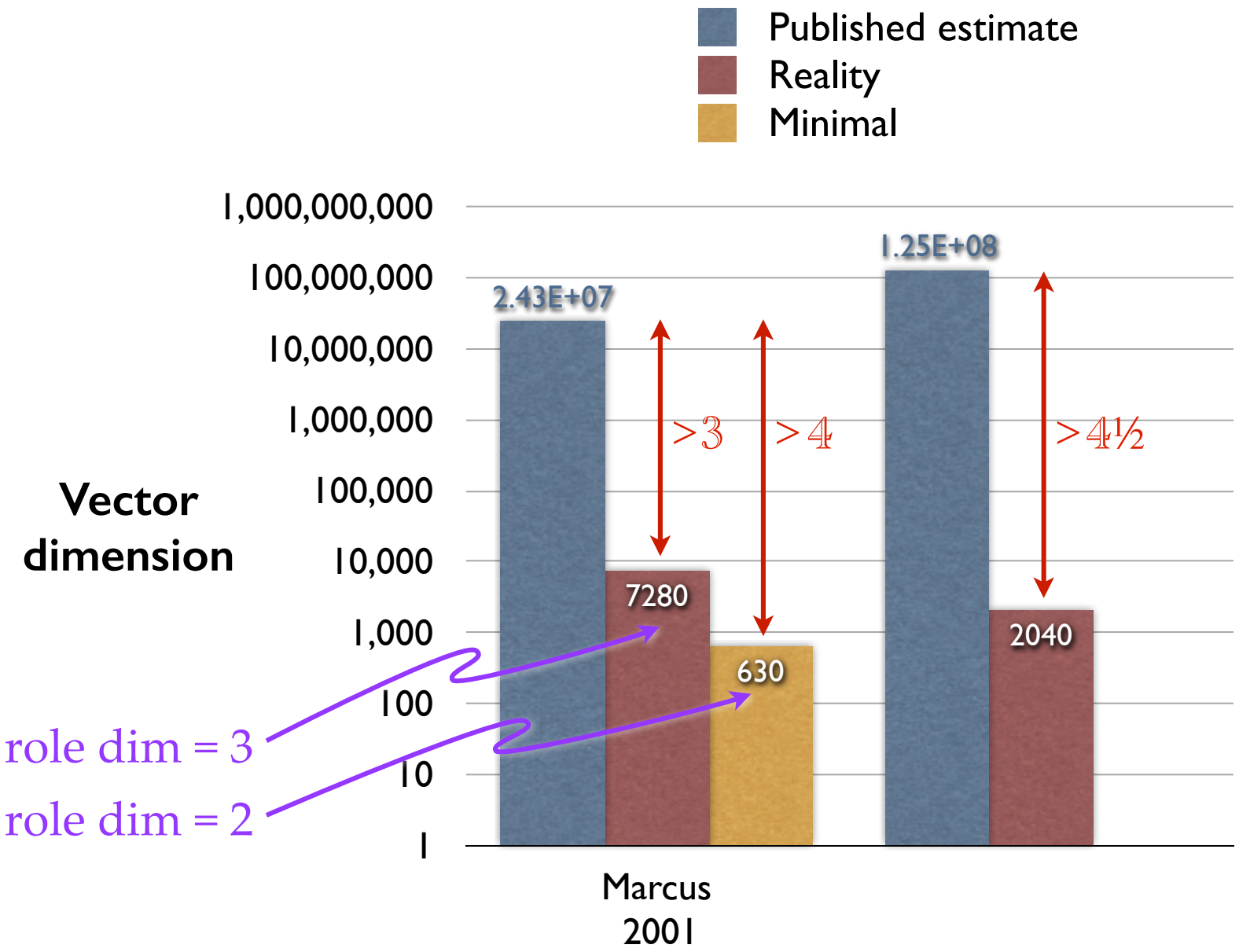
☞ $\mathcal{H}$ built on Tensor Product Representations (TPRs) ⇒ can *program* (not yet *learn*) symbolic computations



4 facts about Tensor Product Representations (TPRs)

1. **Not larger**
   - In cognitive models examined to date, (noisy) alternative schemes utilize at least as many neurons as their (noise-free) proper TPR counterparts
2. No alternatives
   - Known proposed solutions to the problem of encoding symbol structures as activity vectors (which subsumes the variable binding problem) are all special cases of Generalized TPRs.
3. Full-blown symbol processing
   - … can be carried out on fully distributed TPRs via multilinear operations.
4. Grammatical competence
   - The competence to generate binary trees that are grammatical according to a given Harmonic Grammar is implementable in fully parallel, fully distributed recurrent networks.

TPR size: Myth & reality



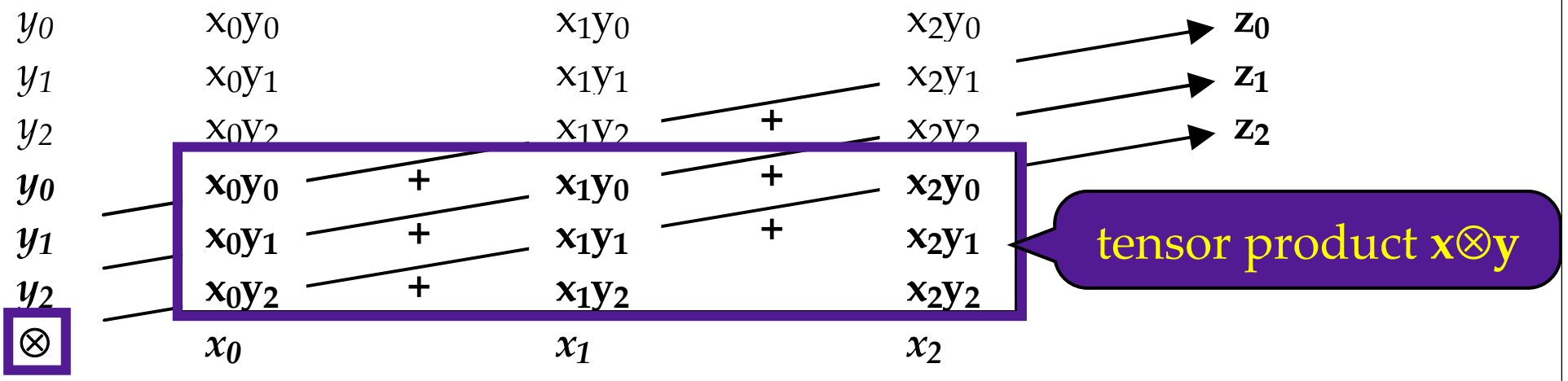
error: orders of magnitude

Error (?): Assuming TPRs are assembled identically to alternatives

So what approach do Marcus and Eliasmith favor?

Plate 1991: Holographic Reduced Representations (HRRs)

Circular convolution: $\mathbf{z} = \mathbf{x} \circledast \mathbf{y}$



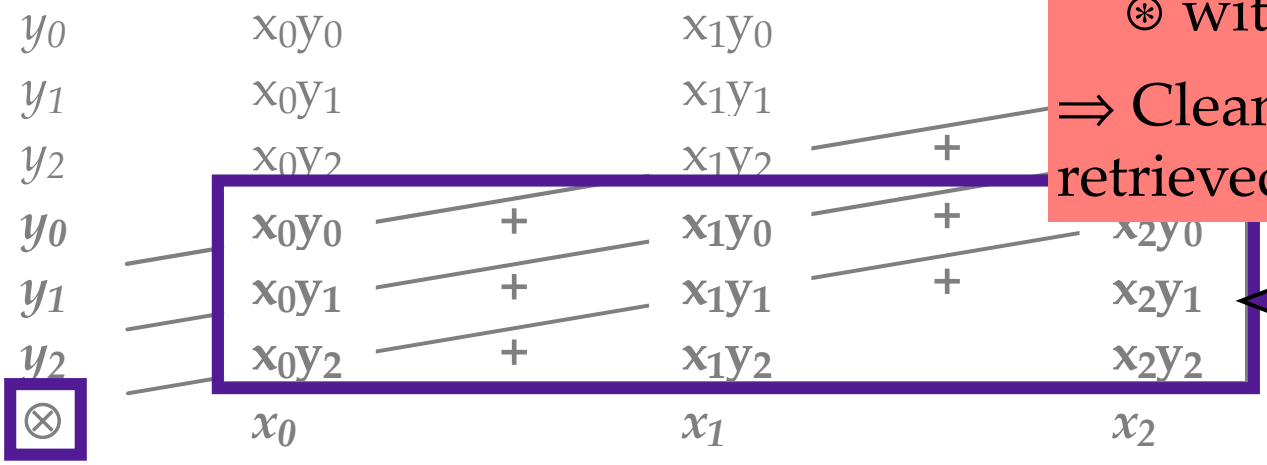
$$\mathbf{z} = \mathbf{x} \circledast \mathbf{y} \qquad \mathbf{z}, \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$$

$$z_\lambda = \sum_{\alpha\beta} T_{\lambda\alpha\beta} x_\alpha y_\beta$$

$$T_{\lambda\alpha\beta} = \begin{cases} 1 & \text{if } \lambda = \alpha + \beta \pmod{d} \\ 0 & \text{otherwise} \end{cases}$$

Plate 1991: Holographic Reduced Representations (HRRs)

Circular convolution: $\mathbf{z} = \mathbf{x} \circledast \mathbf{y}$



Unbinding ($\mathbf{z} \rightarrow \mathbf{x} \mid \mathbf{y}$) is noisy:

$\circledast$ with pseudo-inverse vector

$\Rightarrow$ Clean-up is essential (replace retrieved vector with actual one)

$$\mathbf{z} = \mathbf{x} \circledast \mathbf{y} \qquad \mathbf{z}, \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$$

$$z_\lambda = \sum_{\alpha\beta} T_{\lambda\alpha\beta} x_\alpha y_\beta$$

$$T_{\lambda\alpha\beta} = \begin{cases} 1 & \text{if } \lambda = \alpha + \beta \pmod{d} \\ 0 & \text{otherwise} \end{cases}$$

Typical type of HRR encoding:

[B C] $\rightarrow$ **B** $\circledast$ **C**

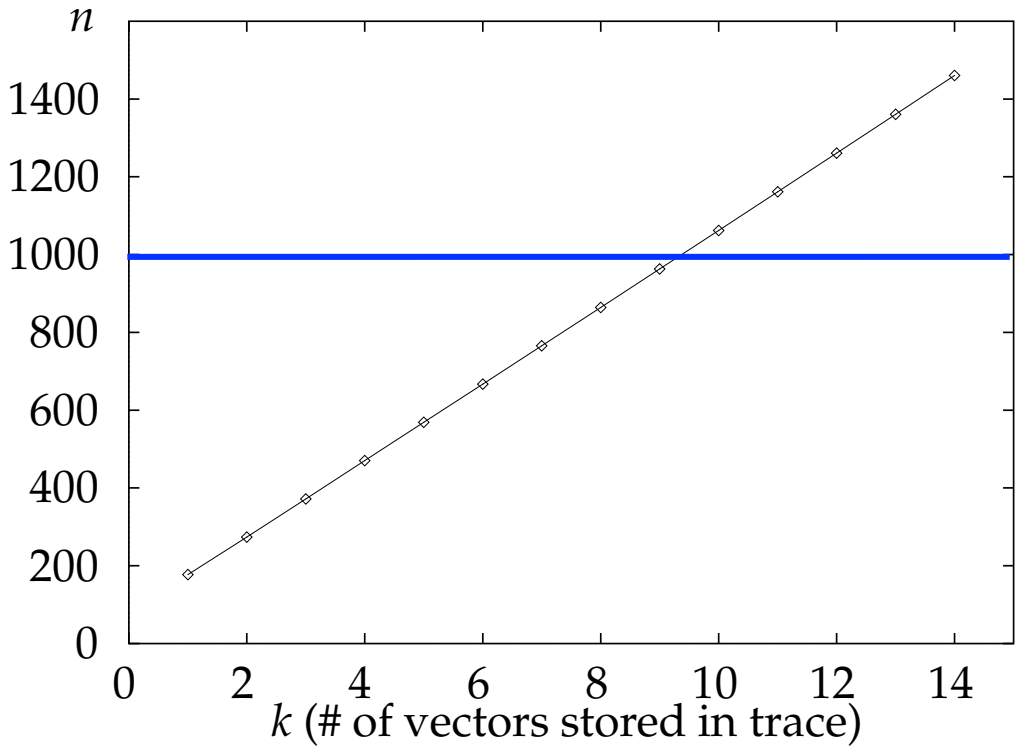
[A [B C]] $\rightarrow$ **A** $\circledast$ **[B** $\circledast$ **C]**

So how much benefit do we get in smaller representations by putting up with significant noise?

Plate 1994: Superpostion of pairs (x_i, y_i)

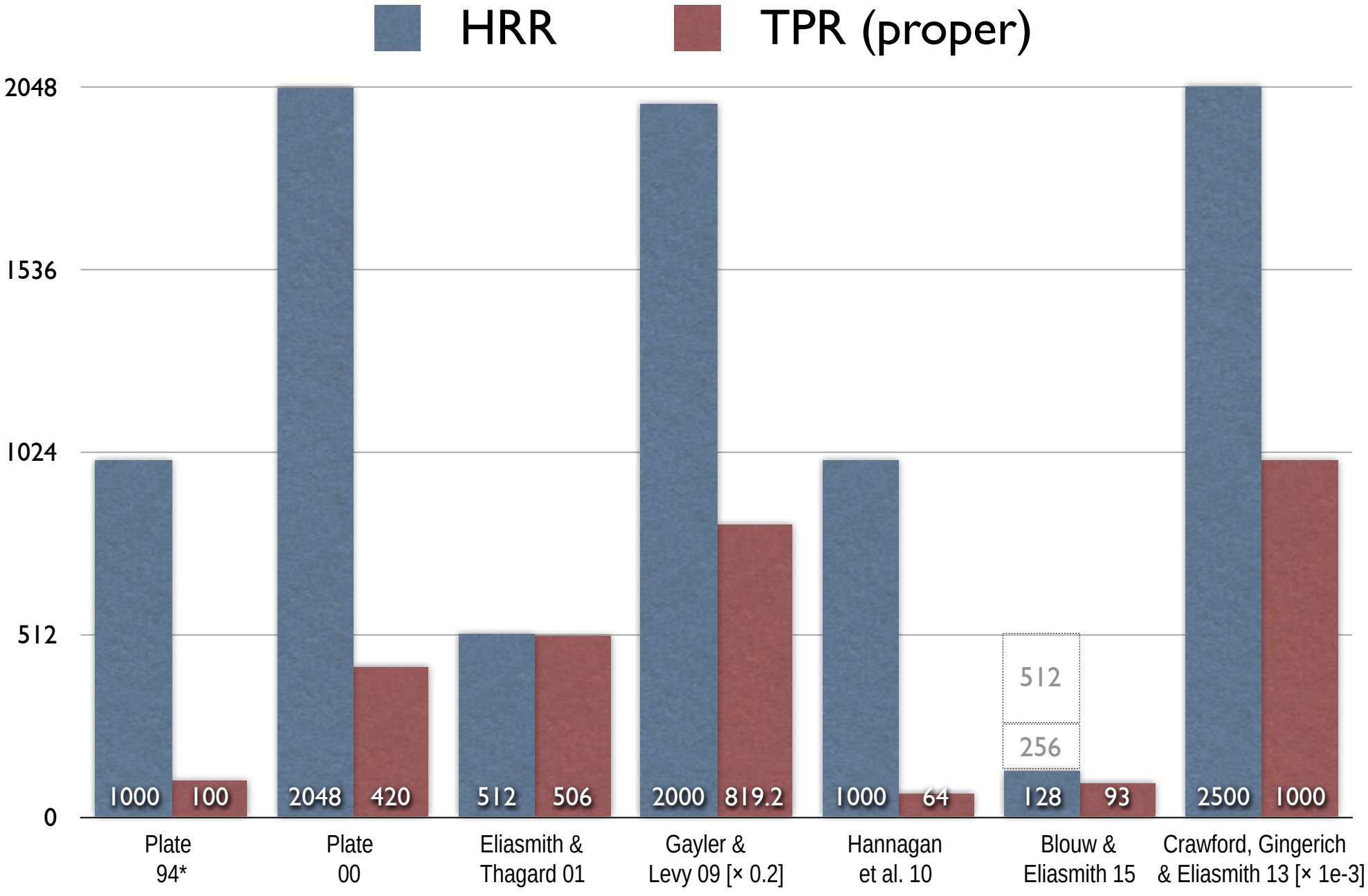
with 1000-dim (random) vectors, HRR pairs $\mathbf{x}_i \circledast \mathbf{y}_i$ superimposed for $i = 1{:}k$

can unbind accurately up to $k < 10$

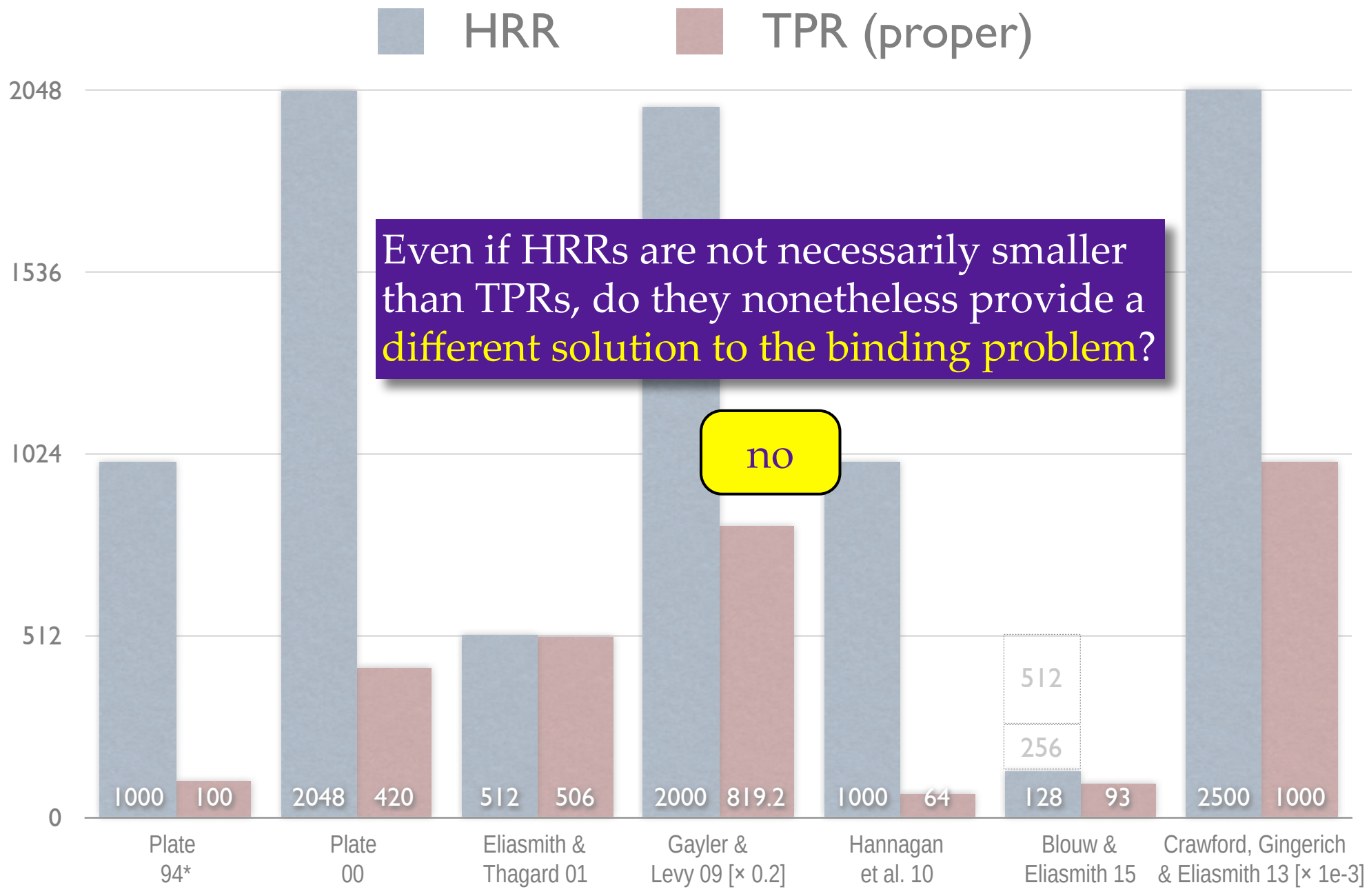


with TPR, (linearly independent) vectors of dim $d = \sqrt{1000}$, can have pairs $\mathbf{x}_i \otimes \mathbf{y}_i$ (dim = 1000) superimposed up to $k = d = 31$ & have *perfect* unbinding (would require > 3100 units given trend in plot above)

HRR vs. TPR: Size of representations



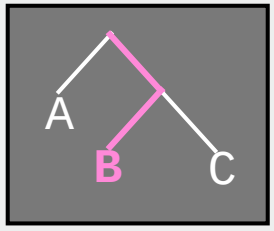
HRR vs. TPR: Size of representations



Tensor Product Representations (TPRs)

① Define a structural type as {roles}, structural token = {fillers/roles}

Two general classes of roles: [fillers here: J(ay), K(ay)]

EXAMPLE	CONTEXTUAL ROLES	POSITIONAL ROLES
K runs	{K/runner}	{K/agent, run/V}
	K⊗run	**K⊗agent + run⊗V**
J loves K	**J⊗loves⊗K**	**J⊗agent + love⊗V + K⊗patient**
[A B]	**A⊗B**	**$A\otimes r_0 + B\otimes r_1$**
[A [B C]]	**A⊗(B⊗C)** 	**$A\otimes r_0 \oplus (B\otimes r_0 + C\otimes r_1)\otimes r_1 \equiv$** **$A\otimes r_0 \oplus B\otimes r_{01} + C\otimes r_{11}$** **$r_{01} \equiv r_0\otimes r_1$**

② Encode filler & role types as vectors, and encode a structural token

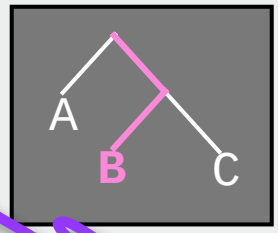
$S = \{filler_k/role_k\}$ as $\psi(S) \equiv \sum_k filler_k \otimes role_k$

Tensor Product Representations (TPRs)

① Define a structural type as {roles}, structural token = {fillers/roles}

Two general classes of roles: [fillers here: J(ay), K(ay)]

EXAMPLE	CONTEXTUAL ROLES	POSITIONAL ROLES
K runs	{K/runner}	{K/agent, run/V}
	K⊗run	**K⊗agent + run⊗V**
J loves K	**J⊗loves⊗K**	**J⊗agent + love⊗V + K⊗patient**
[A B]	**A⊗B**	**$A\otimes r_0 + B\otimes r_1$**
[A [B C]]	**A⊗(B⊗C)**	**$A\otimes r_0 \oplus (B\otimes r_0 + C\otimes r_1)\otimes r_1 \equiv$** **$A\otimes r_0 \oplus B\otimes r_{01} + C\otimes r_{11}$** **$r_{01} \equiv r_0\otimes r_1$**



TPR size estimate errors (?): assuming while for linguistic trees I use

Tensor Product Representations (TPRs)

① Define a structural type as {roles}, structural token = {fillers/roles}

Two general classes of roles: [fillers here: J(ay), K(ay)]

EXAMPLE	CONTEXTUAL ROLES	POSITIONAL ROLES
K runs	{K/runner}	{K/agent, run/V}

proper TPR: vectors for all fillers and for all roles are linearly independent

⇒ perfect unbinding by inner product: $(\sum_k [\mathbf{filler}_k][\mathbf{role}_k]^T)[\mathbf{role}_j^+] = \mathbf{filler}_j$

where the ***unbinding vectors*** satisfy: $[\mathbf{role}_k]^T[\mathbf{role}_j^+] = \delta_{kj} = [1 \text{ if } k{=}j \text{ else } 0]$

$A \otimes r_0 \oplus B \otimes r_{01} + C \otimes r_{11}$

$r_{01} \equiv r_0 \otimes r_1$

TPR size estimate errors (?): assuming while for linguistic trees I use

Application: Semantic Parsing

On-going project at Microsoft Research, with:

- Hao Cheng, Li Deng, Jianfeng Gao, Xiaodong He, Mari Ostendorf

Learn to map Sentence to Meaning = AMR **graph**

- edge of graph: sing $-\text{ARG0}\rightarrow$ J encoded as **sing⊗ARG0⊗J**
- graph: sum of all edges (triples)

stay tuned …

4 facts about Tensor Product Representations (TPRs)

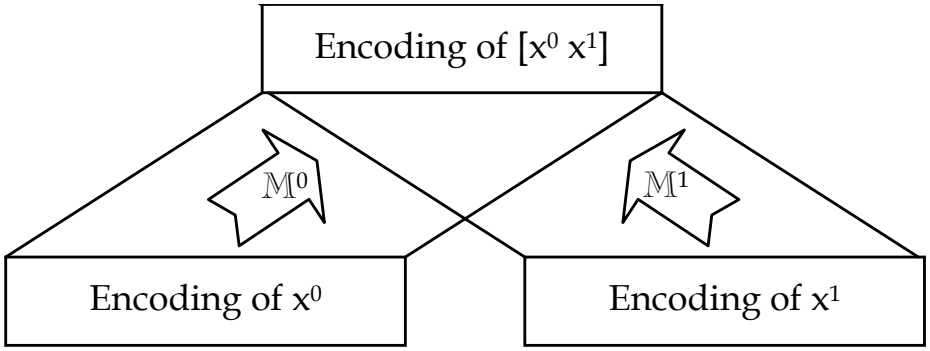
1. Not larger
   - In cognitive models examined to date, (noisy) alternative schemes utilize at least as many neurons as their (noise-free) proper TPR counterparts
2. **No alternatives**
   - Known proposed solutions to the problem of encoding symbol structures as activity vectors (which subsumes the variable binding problem) are all special cases of Generalized TPRs.
3. Full-blown symbol processing
   - … can be carried out on fully distributed TPRs via multilinear operations.
4. Grammatical competence
   - The competence to generate binary trees that are grammatical according to a given Harmonic Grammar is implementable in fully parallel, fully distributed recurrent networks.

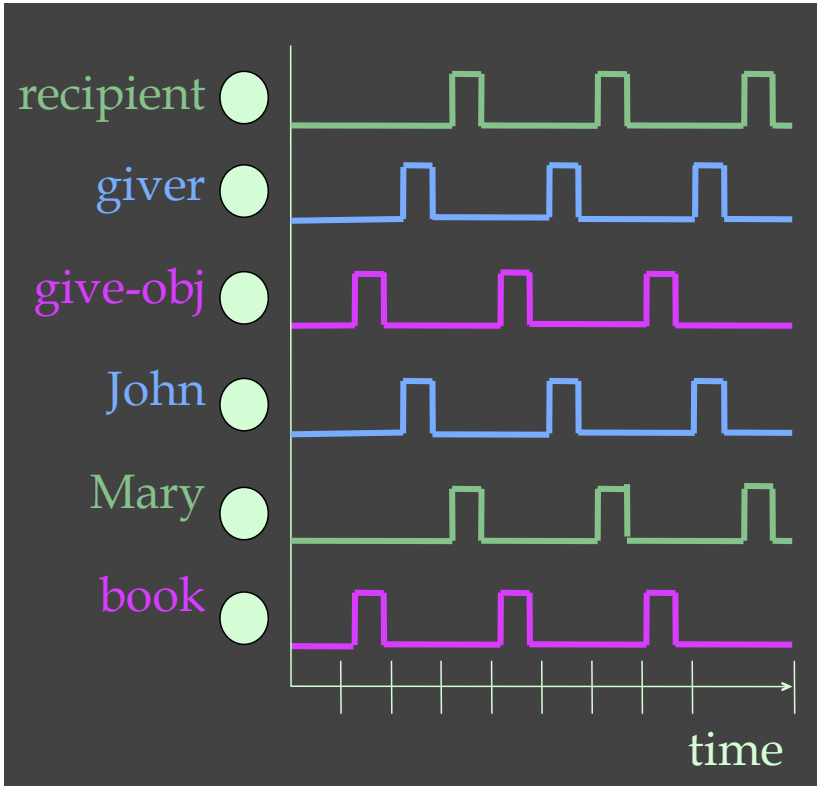
All vectorial symbol structures are Generalized TPRs

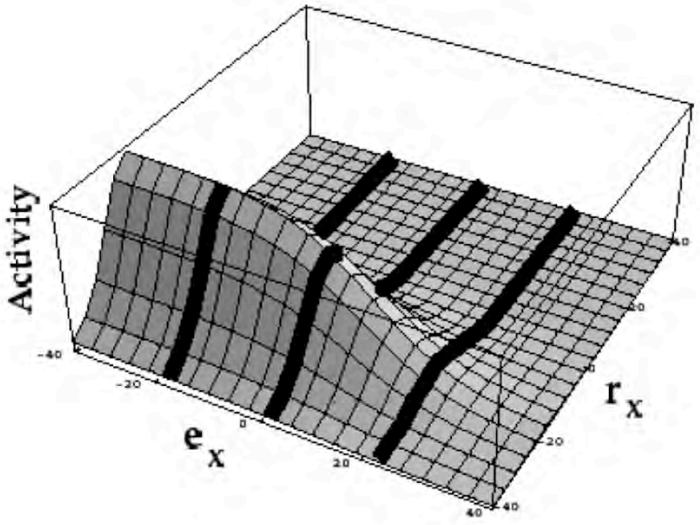
includes:

- HRRs
- RNNs/RAAM
- Temporal synchrony
- Gain receptive fields

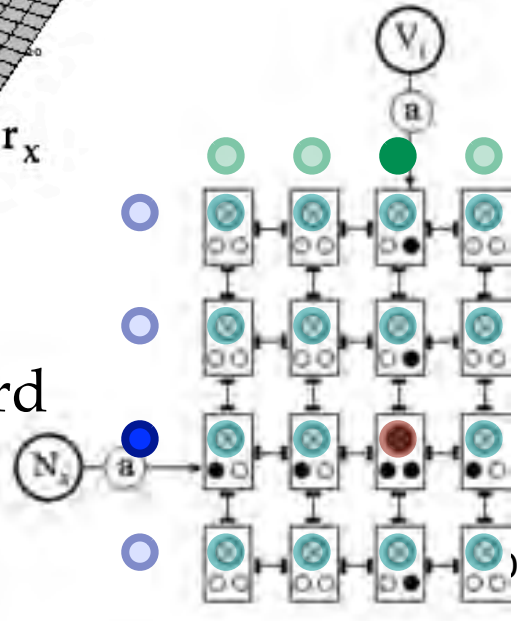
} Neuroscience







- *connections:* Neural Blackboard Architecture



4 facts about Tensor Product Representations (TPRs)

1. Not larger
   - In cognitive models examined to date, (noisy) alternative schemes utilize at least as many neurons as their (noise-free) proper TPR counterparts
2. **No alternatives**
   - Known proposed solutions to the problem of encoding symbol structures as activity vectors (which subsumes the variable binding problem) are all special cases of Generalized TPRs.

Just as the notion 'computable function' in symbolic computation displays its naturalness by reappearing in many guises, superficially seemingly very different by underlyingly isomorphic, so this result lends credence to the naturalness of the notion 'vectorial encoding of symbolic structure through tensor product representations'

4 facts about Tensor Product Representations (TPRs)

1. Not larger
   - In cognitive models examined to date, (noisy) alternative schemes utilize at least as many neurons as their (noise-free) proper TPR counterparts
2. No alternatives
   - Known proposed solutions to the problem of encoding symbol structures as activity vectors (which subsumes the variable binding problem) are all special cases of Generalized TPRs.
3. **Full-blown symbol processing**
   - … can be carried out on **fully distributed** TPRs via multilinear operations.

   a. Linearly computable functions
   b. Function-argument application in the λ-calculus
   c. Tree Adjunction as in Tree Adjoining Grammar
   d. bAbI reasoning

4 facts about Tensor Product Representations (TPRs)

1. Not larger
   - In cognitive models examined to date, (noisy) alternative schemes utilize at least as many neurons as their (noise-free) proper TPR counterparts
2. No alternatives
   - Known proposed solutions to the problem of encoding symbol structures as activity vectors (which subsumes the variable binding problem) are all special cases of Generalized TPRs.
3. **Full-blown symbol processing**
   - … can be carried out on **fully distributed** TPRs via multilinear operations.

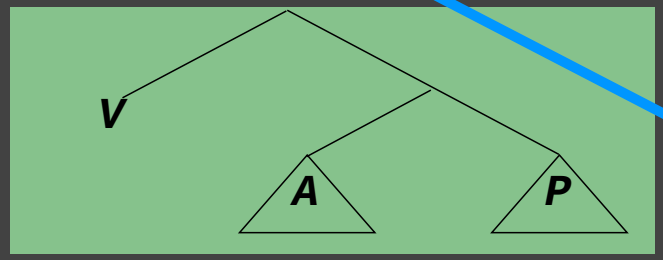
   a. **Linearly computable functions**
   b. Function-argument application in the λ-calculus
   c. Tree Adjunction as in Tree Adjoining Grammar
   d. bAbI reasoning

Few leaders are admired by George Bush $\xrightarrow{f}$ admire(George Bush, few leaders)

$f(s) = \mathbf{cons}(\mathbf{ex}_1(\mathbf{ex}_0(\mathbf{ex}_1(s))),$
$\mathbf{cons}(\mathbf{ex}_1(\mathbf{ex}_1(\mathbf{ex}_1(s))), \mathbf{ex}_0(s)))$

f is linearly neurally computable

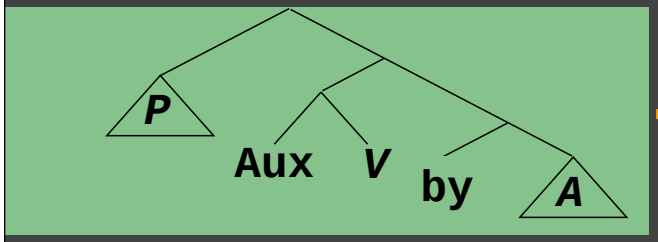
$$\mathbb{W} = \mathbb{W}_{\mathbf{cons}_0}[\mathbb{W}_{\mathbf{ex}_1}\mathbb{W}_{\mathbf{ex}_0}\mathbb{W}_{\mathbf{ex}_1}] + \mathbb{W}_{\mathbf{cons}_1}[\mathbb{W}_{\mathbf{cons}_0}(\mathbb{W}_{\mathbf{ex}_1}\mathbb{W}_{\mathbf{ex}_1}\mathbb{W}_{\mathbf{ex}_1}) + \mathbb{W}_{\mathbf{cons}_1}(\mathbb{W}_{\mathbf{ex}_0})]$$



Meaning (LF)

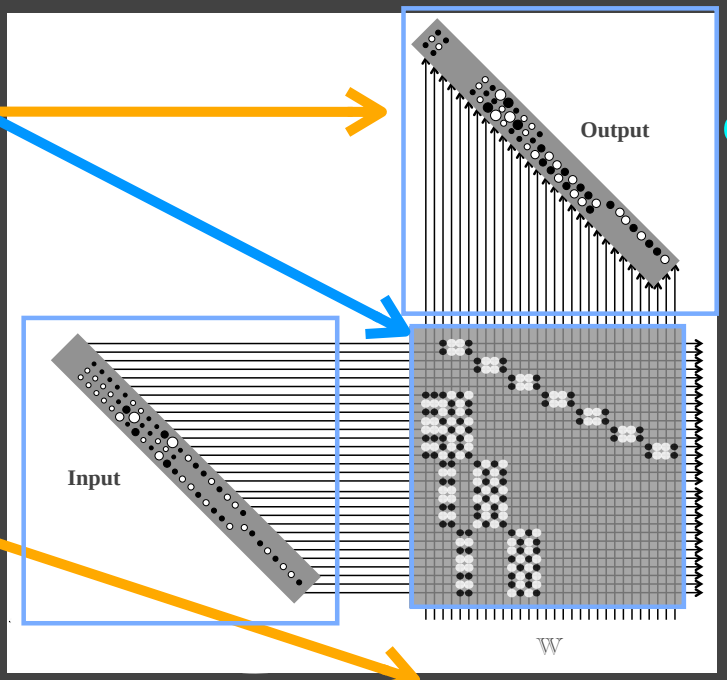
f

symbolic ↘ isomorphism subsymbolic



"Passive sentence"

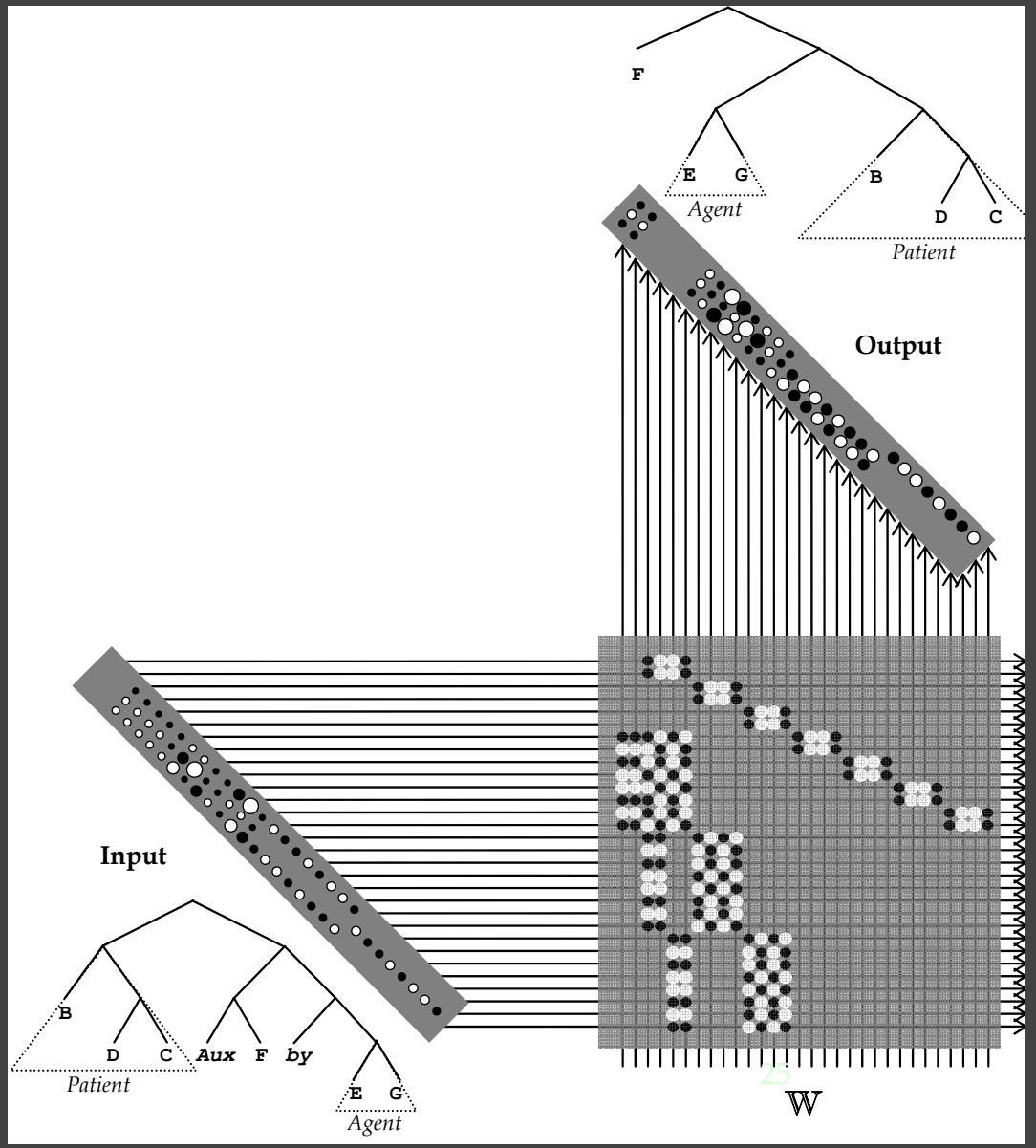
$\mathbf{a}_{Passive}$



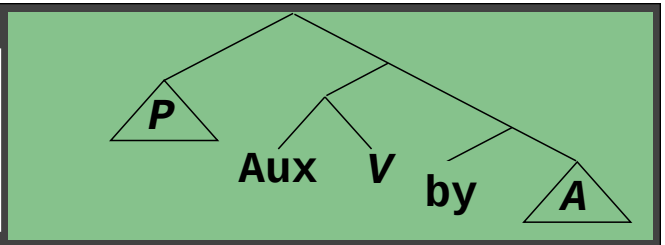
$\mathbf{a}_{LF}$

$\mathbb{W}\,\mathbf{a}_{Passive}$

$$\mathbb{W} = \mathbb{W}_{cons_0}[\mathbb{W}_{ex_1}\mathbb{W}_{ex_0}\mathbb{W}_{ex_1}] + \mathbb{W}_{cons_1}[\mathbb{W}_{cons_0}(\mathbb{W}_{ex_1}\mathbb{W}_{ex_1}\mathbb{W}_{ex_1}) + \mathbb{W}_{cons_1}(\mathbb{W}_{ex_0})]$$

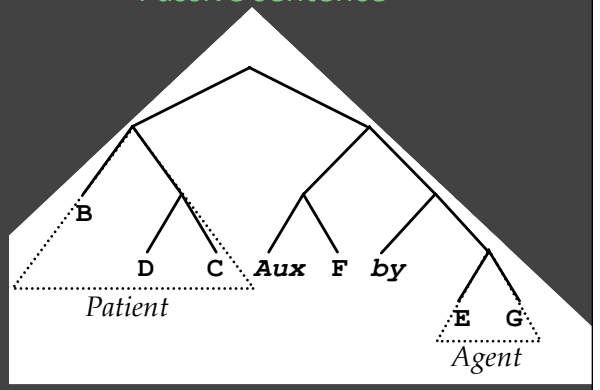


$$\mathbb{W} = \mathbb{W}_{cons_0}[\mathbb{W}_{ex_1}\mathbb{W}_{ex_0}\mathbb{W}_{ex_1}] + \mathbb{W}_{cons_1}[\mathbb{W}_{cons_0}(\mathbb{W}_{ex_1}\mathbb{W}_{ex_1}\mathbb{W}_{ex_1}) + \mathbb{W}_{cons_1}(\mathbb{W}_{ex_0})]$$




```
PassiveLF0 = FM(Wex1F, FM(Wex0F, Wex1F))
PassiveLF1 = WconsF(FM(Wex1F, FM(Wex1F, Wex1F)), Wex0F)
WPassiveLF = WconsF(PassiveLF0, PassiveLF1)
```

"Passive sentence"

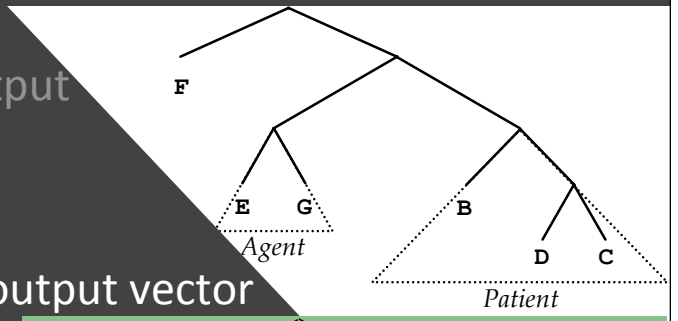


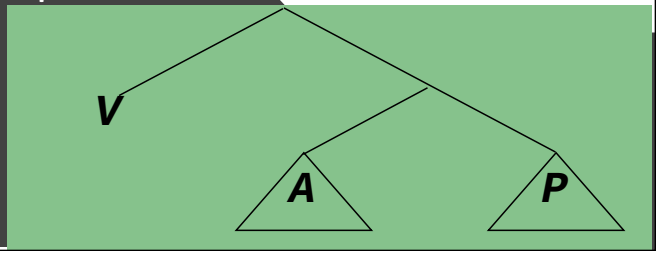

```
PassiveInput = '[[B [D C]] [[A F] [H [E G]]]]'
PIf = tree2vec(PassiveInput)                    # PIf = Fock space vector realizing PassiveInput
PrintTree(PIf)          # -> [ [ B [ D C ] ] [ [ A F ] [ H [ E G ] ] ] ]
```

```
LFf = FM(WPassiveLF, PIf)       # [output Fock vector] = [Fock transf matrix] times [input Fock vector]
PrintTree(LFf)          # -> [ F [ [ E G ] [ B [ D C ] ] ] ]
```

```
PassiveOutput = '[F [[E G] [B [D C]]]]'                    # DESIRED output
POf = tree2vec(PassiveOutput)
PrintTree(POf)          # -> [ F [ [ E G ] [ B [ D C ] ] ] ]

testEqual(LFf, POf)     # -> True  computed output vector = desired output vector
```





Few leaders are admired by George Bush $\xrightarrow{f}$ admire(George Bush, few leaders)

$f(s) = \mathbf{cons}(\mathbf{ex}_1(\mathbf{ex}_0(\mathbf{ex}_1(s))),$
$\mathbf{cons}(\mathbf{ex}_1(\mathbf{ex}_1(\mathbf{ex}_1(s))), \mathbf{ex}_0(s)))$

f is linearly neurally computable

$$\mathbb{W} = \mathbb{W}_{\mathbf{cons}_0}[\mathbb{W}_{\mathbf{ex}_1}\mathbb{W}_{\mathbf{ex}_0}\mathbb{W}_{\mathbf{ex}_1}] + \mathbb{W}_{\mathbf{cons}_1}[\mathbb{W}_{\mathbf{cons}_0}(\mathbb{W}_{\mathbf{ex}_1}\mathbb{W}_{\mathbf{ex}_1}\mathbb{W}_{\mathbf{ex}_1}) + \mathbb{W}_{\mathbf{cons}_1}(\mathbb{W}_{\mathbf{ex}_0})]$$

The functions in the following classes are linearly neurally computable

$\mathcal{B}$ = base of in-place symbol mappings

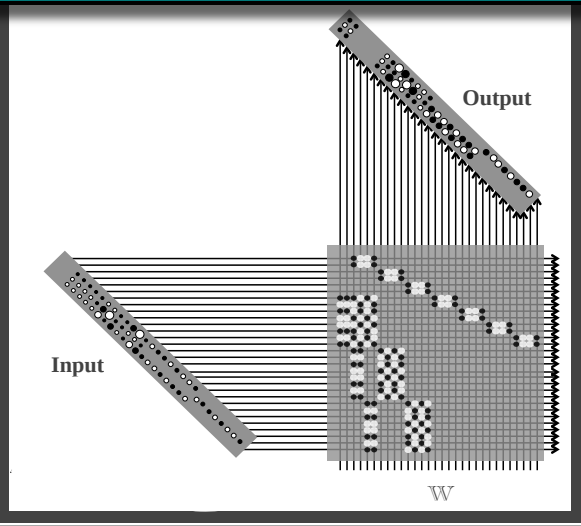
$\mathcal{C}$ = closure under composition of tree-manipulating primitives + $\mathcal{B}$

$\mathcal{P}$ ~ "primitive recursive"

$\mathcal{C} \subset \mathcal{P}$

$g, h \in \mathcal{P} \Rightarrow f \in \mathcal{P}$ when

$$f(s) = \begin{cases} g(s) & \text{if } \mathtt{atom}(s) \\ h(f(\mathbf{ex}_0(s)), f(\mathbf{ex}_1(s))) & \text{otherwise} \end{cases}$$



4 facts about Tensor Product Representations (TPRs)

1. Not larger
   - In cognitive models examined to date, (noisy) alternative schemes utilize at least as many neurons as their (noise-free) proper TPR counterparts
2. No alternatives
   - Known proposed solutions to the problem of encoding symbol structures as activity vectors (which subsumes the variable binding problem) are all special cases of Generalized TPRs.
3. **Full-blown symbol processing**
   - … can be carried out on **fully distributed** TPRs via multilinear operations.

   a. Linearly computable functions

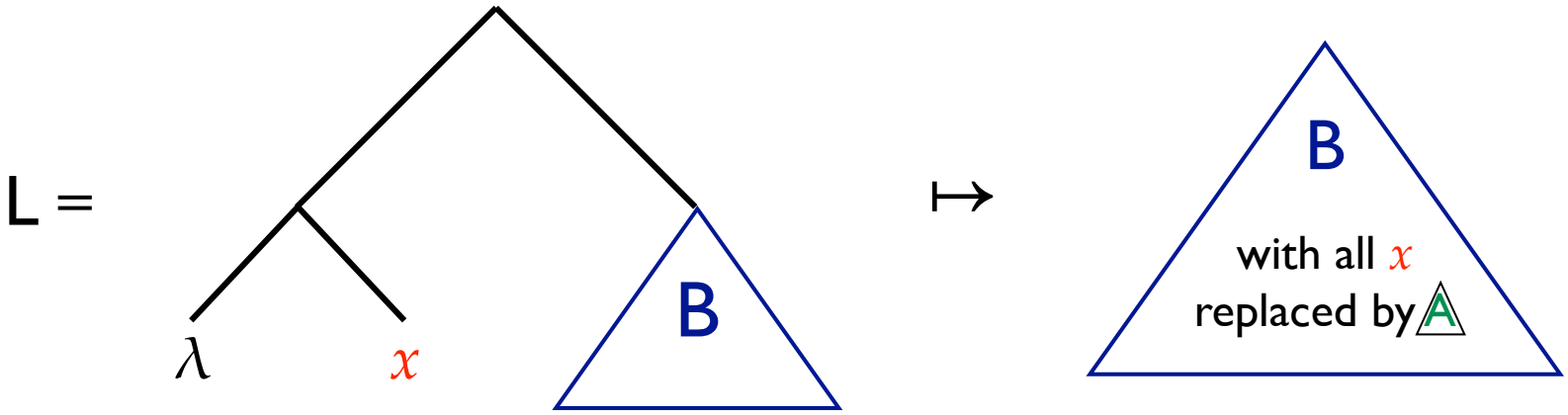
   b. Function-argument application in the λ-calculus

   c. Tree Adjunction as in Tree Adjoining Grammar

   d. bAbI reasoning

β Reduction

Basic operation of λ-calculus [function application]: $(\lambda x.\mathsf{B})\mathsf{A}$

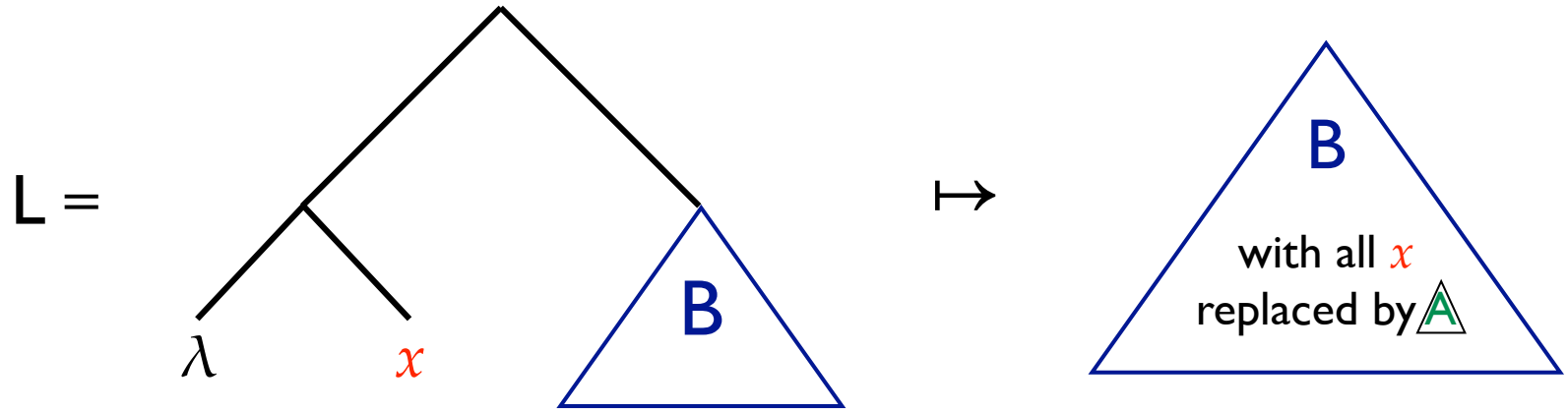


$$\beta\text{-reduce}(\mathsf{L},\mathsf{A}) = \left[\mathbb{1} + (\mathbb{A} - \mathbb{L}\cdot \mathrm{r}_{10}^{+})(\mathbb{L}\cdot \mathrm{r}_{10}^{+})^{+}\cdot\right](\mathbb{L}\cdot \mathrm{r}_{1}^{+})$$

degree 3 in $\mathbb{L}$

β Reduction

Basic operation of λ-calculus [function application]: $(\lambda x.\mathsf{B})\mathsf{A}$



$$\mathbb{x} \equiv \mathbb{L}\cdot \mathbb{r}_{10}^{+} \quad \mathbb{B} \equiv \mathbb{L}\cdot \mathbb{r}_{1}^{+}$$

extracts x

extracts B

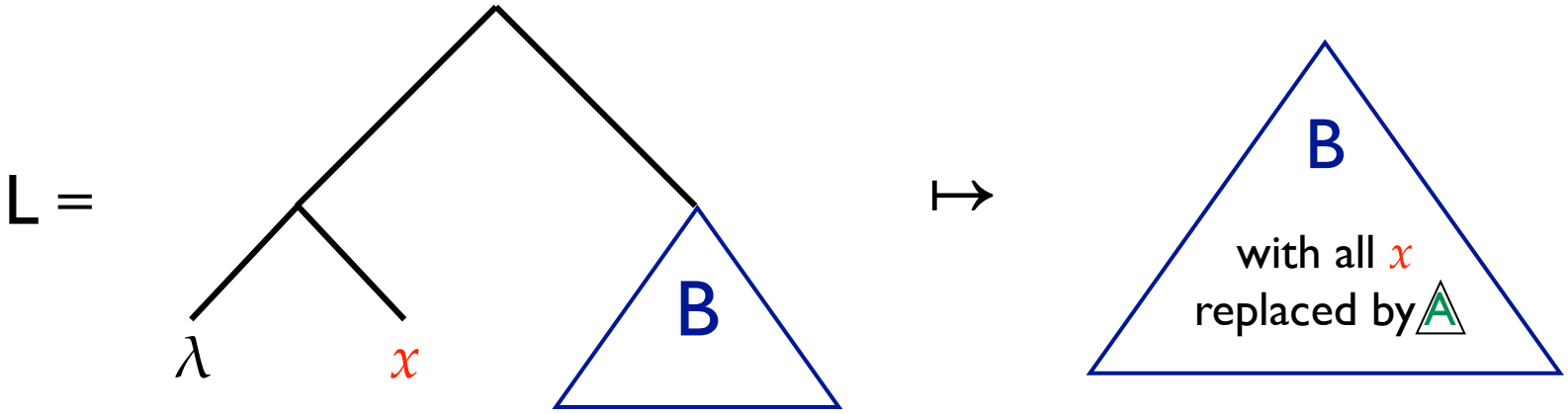
$$\text{educe}(\mathsf{L},\mathsf{A}) = \left[\, \mathbb{1} + (\mathbb{A} - \mathbb{x})\mathbb{x}^{+}\cdot \,\right]\mathbb{B}$$

$$\text{uce}(\mathsf{L},\mathsf{A}) = \left[\, \mathbb{1} + (\mathbb{A} - \mathbb{L}\cdot \mathbb{r}_{10}^{+})(\mathbb{L}\cdot \mathbb{r}_{10}^{+})^{+}\cdot \,\right](\mathbb{L}\cdot \mathbb{r}_{1}^{+})$$

degree 3 in $\mathbb{L}$

β Reduction

Basic operation of λ-calculus [function application]: $(\lambda x.\mathsf{B})\mathsf{A}$



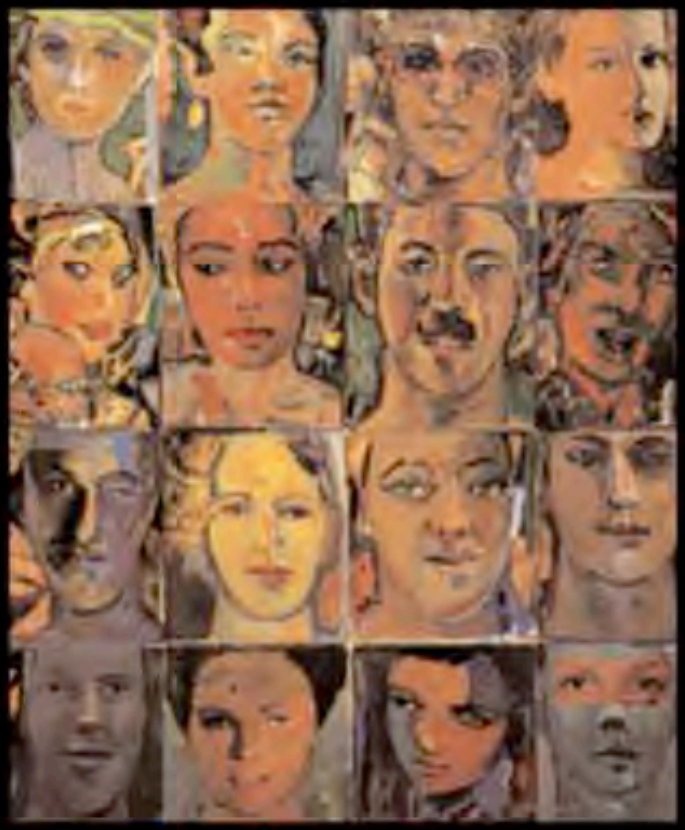
$$\mathbb{x} \equiv \mathbb{L} \cdot \mathbb{r}_{10}^{+} \quad \mathbb{B} \equiv \mathbb{L} \cdot \mathbb{r}_{1}^{+}$$

$$\beta\text{-reduce}(\mathsf{L},\mathsf{A}) = \left[\mathbb{1} + (\mathbb{A} - \mathbb{x})\mathbb{x}^{+} \cdot \right] \mathbb{B}$$

$$\beta\text{-reduce}(\mathsf{L},\mathsf{A}) = \left[\mathbb{1} + \cdots \cdot \right] (\mathbb{L} \cdot \mathbb{r}_{1}^{+})$$

- reproduces B
- inserts A at each such location
- deletes x at each such location
- returns location of all x's

one candidate neurocognitive ontology

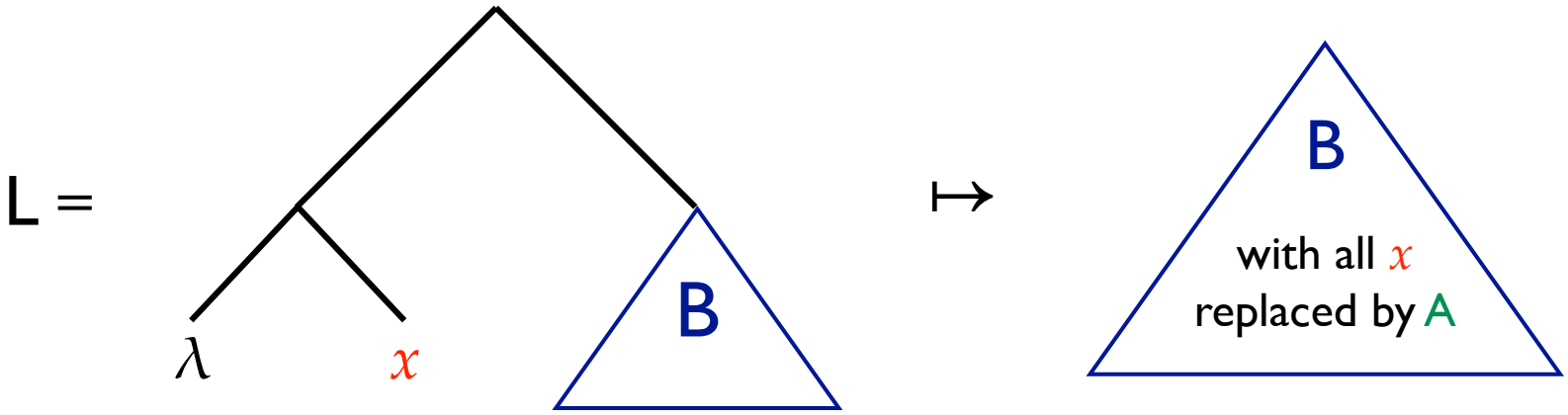


Marcus, 2001, MIT Press

1. ∃ a neurally-realized way of representing **symbols**
2. ∃ a neurally-realized way of representing **variables**
3. ∃ a neurally-realized way of representing **operations over variables**
4. ∃ a neurally-realized way of representing **distinguishing types from tokens**
5. ∃ a neurally-realized way of representing **ordered pairs (AB ≠ BA)**
6. ∃ a neurally-realized way of representing **structured units** (treelet C composed of elements A and B)
7. ~~∃ a neurally-realized way of representing **arbitrary trees**~~

β Reduction

Basic operation of λ-calculus [function application]: $(\lambda x.\mathsf{B})\mathsf{A}$



$$\beta\text{-reduce}(\mathsf{L},\mathsf{A}) = \left[\, \mathbb{1} + (\mathbb{A} - \mathbb{x}) \otimes \mathbb{x}^{+} \cdot \right] \mathbb{B}$$

$$\beta\text{-reduce}(\mathsf{L},\mathsf{A}) = \left[\, \mathbb{1} + (\mathbb{A} - \mathbb{L} \cdot \mathbb{r}_{10}^{+}) \otimes (\mathbb{L} \cdot \mathbb{r}_{10}^{+})^{+} \cdot \right] (\mathbb{L} \cdot \mathbb{r}_{1}^{+})$$

Here: B encodes a string ⇒ atoms label terminal nodes only.
Atom x is replaced by an entire tree A (encoding a string).
Next: An atom at an *internal* (non-terminal) node is replaced by an entire structure.

4 facts about Tensor Product Representations (TPRs)

1. Not larger
   - In cognitive models examined to date, (noisy) alternative schemes utilize at least as many neurons as their (noise-free) proper TPR counterparts
2. No alternatives
   - Known proposed solutions to the problem of encoding symbol structures as activity vectors (which subsumes the variable binding problem) are all special cases of Generalized TPRs.
3. **Full-blown symbol processing**
   - … can be carried out on **fully distributed** TPRs via multilinear operations.

   a. Linearly computable functions
   b. Function-argument application in the λ-calculus
   c. **Tree Adjunction as in Tree Adjoining Grammar**
   d. bAbI reasoning

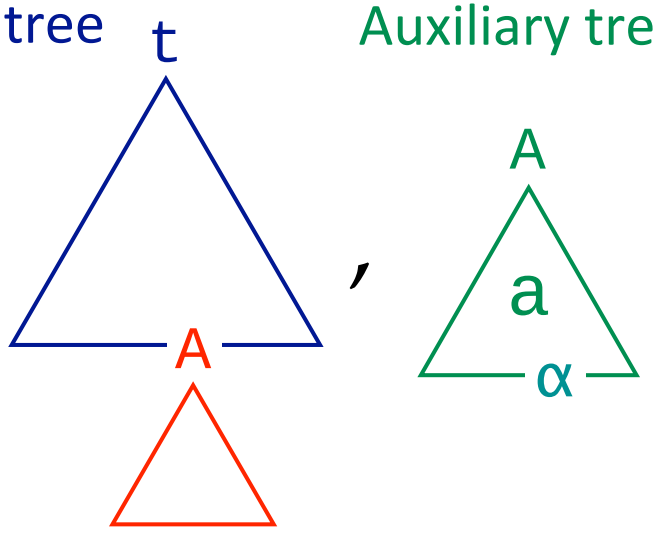
the operation that gives Tree-Adjoining Grammars higher complexity than context-free grammars, as needed for natural language syntax.

Tree Adjoining

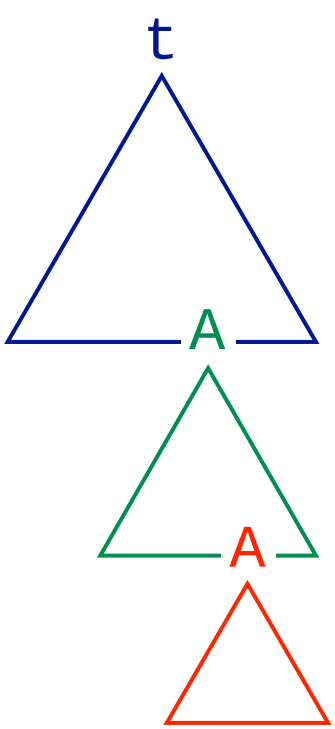
Initial tree

Auxiliary tree

TA:



↦



$\mathbb{A}_{\varepsilon} \equiv \mathbb{a} \cdot \mathbb{r}_{\varepsilon}^{+}$ $\mathbb{R}_{\mathbf{A}} \equiv \mathbb{A}_{\varepsilon}^{+} \cdot \mathbb{t}$

root symbol of a ("A")

location of A in t

$\mathbb{A}_{\varepsilon} \equiv \mathbb{a} \cdot \mathbb{r}_{\varepsilon}^{+}$ root symbol of **a** ("**A**")

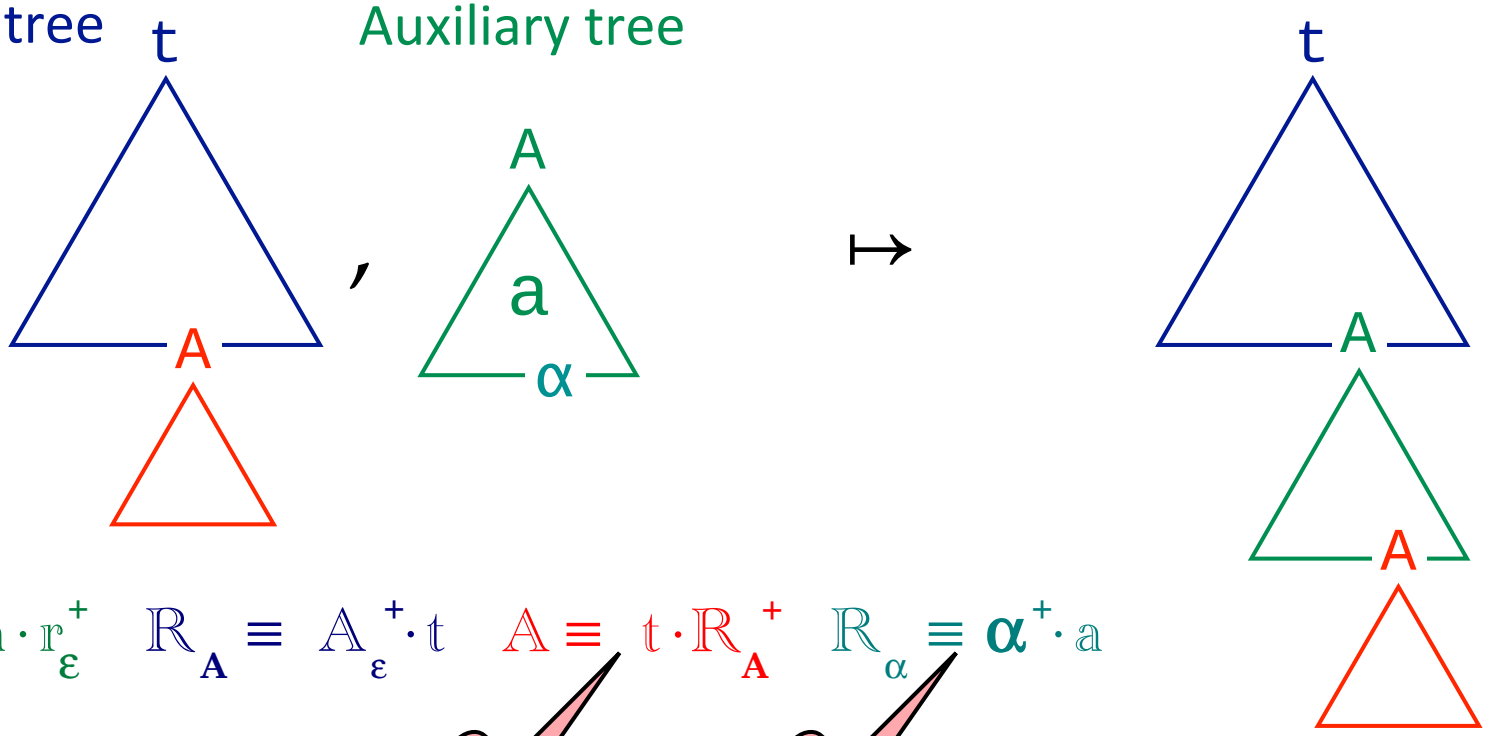
$\mathbb{R}_{\mathbf{A}} \equiv \mathbb{A}_{\varepsilon}^{+} \cdot \mathbb{t}$ location of **A** in **t**

Tree Adjoining

Initial tree

Auxiliary tree

TA:



$\mathbb{A}_{\varepsilon} \equiv a \cdot r_{\varepsilon}^{+}$ $\mathbb{R}_{A} \equiv \mathbb{A}_{\varepsilon}^{+} \cdot t$ $\mathbb{A} \equiv t \cdot \mathbb{R}_{A}^{+}$ $\mathbb{R}_{\alpha} \equiv \alpha^{+} \cdot a$

subtree A in t

location of foot node α in a

$\mathbb{A}_{\varepsilon} \equiv a \cdot r_{\varepsilon}^{+}$ root symbol of **a** ("**A**")

$\mathbb{R}_{A} \equiv \mathbb{A}_{\varepsilon}^{+} \cdot t$ location of **A** in **t**

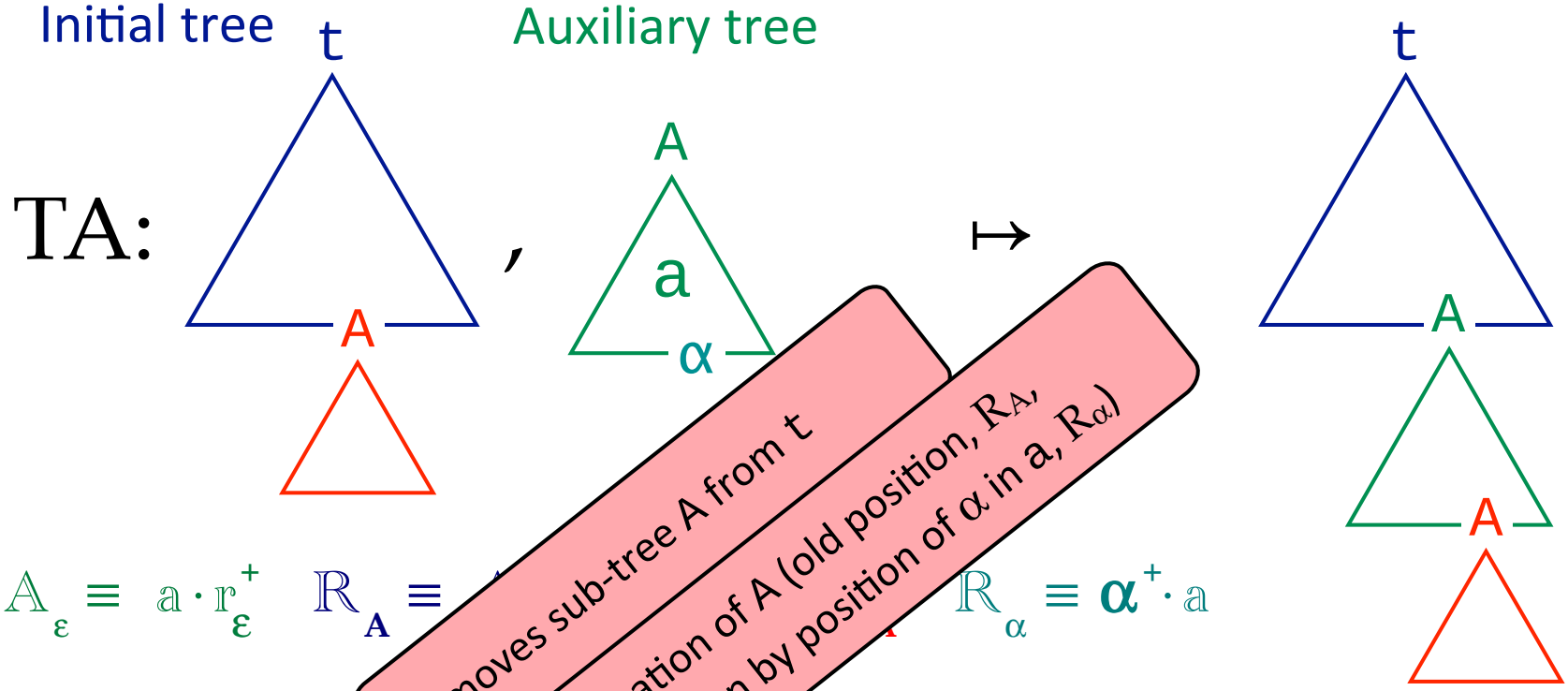
$\mathbb{A} \equiv t \cdot \mathbb{R}_{A}^{+}$ subtree **A** in **t**

$\mathbb{R}_{\alpha} \equiv \alpha^{+} \cdot a$ location of α in **a**

Tree Adjoining

Initial tree

Auxiliary tree



$\mathbb{A}_\varepsilon \equiv a \cdot r_\varepsilon^+$ $\mathbb{R}_A \equiv$ $\mathbb{R}_\alpha \equiv \alpha^+ \cdot a$

$\mathbb{TA}(t,a) =$

$t - A \otimes \mathbb{R}_A$ retain all of **t** unaffected by adj.

$+ A \otimes [\mathbb{R}_\alpha \otimes \mathbb{R}_A]$ reposition $[_A \cdots]$ in **t**

$\mathbb{A}_\varepsilon \equiv a \cdot r_\varepsilon^+$ root symbol of **a** ("**A**")

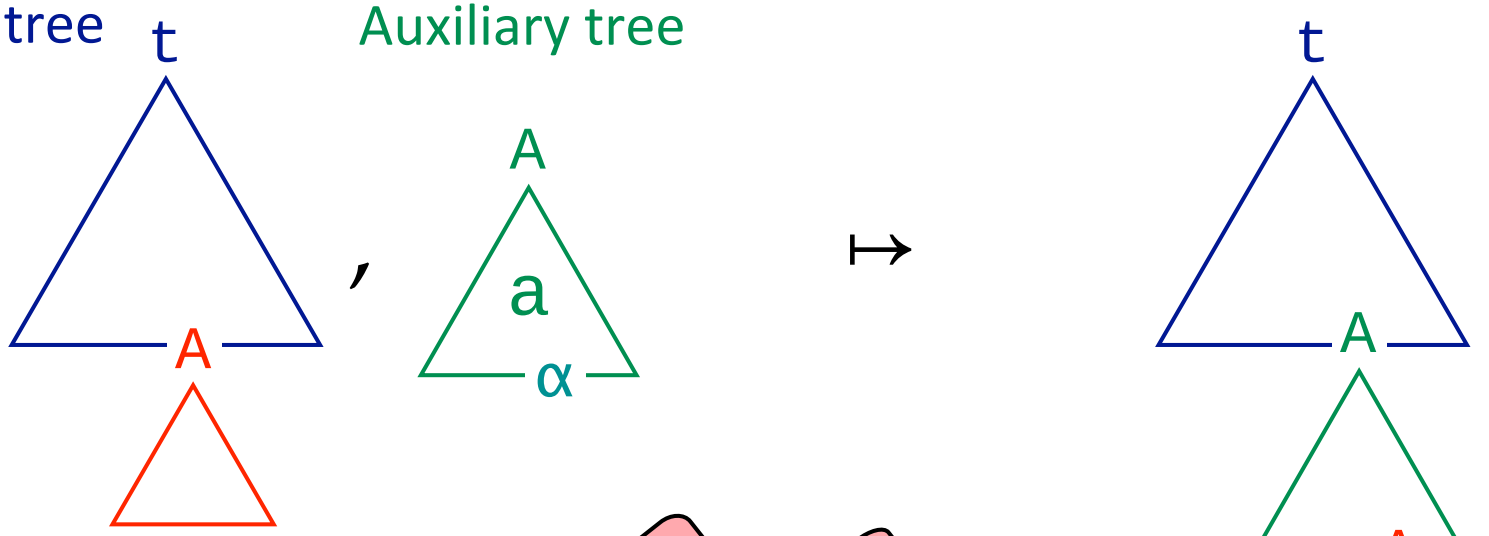
$\mathbb{R}_A \equiv \mathbb{A}_\varepsilon^+ \cdot t$ location of **A** in **t**

$A \equiv t \cdot \mathbb{R}_A^+$ subtree **A** in **t**

$\mathbb{R}_\alpha \equiv \alpha^+ \cdot a$ location of α in **a**

Tree Adjoining

Initial tree t Auxiliary tree

TA: 

$\mathbb{A}_{\varepsilon} \equiv a \cdot r_{\varepsilon}^{+}$ $\mathbb{R}_{A} \equiv \mathbb{A}_{\varepsilon}^{+} \cdot t$ $\mathbb{R}_{\alpha} \equiv \alpha^{+} \cdot a$

embeds a in t at location of A

removes 'α' at new location of t's subtree

$\mathbb{TA}(t,a) =$

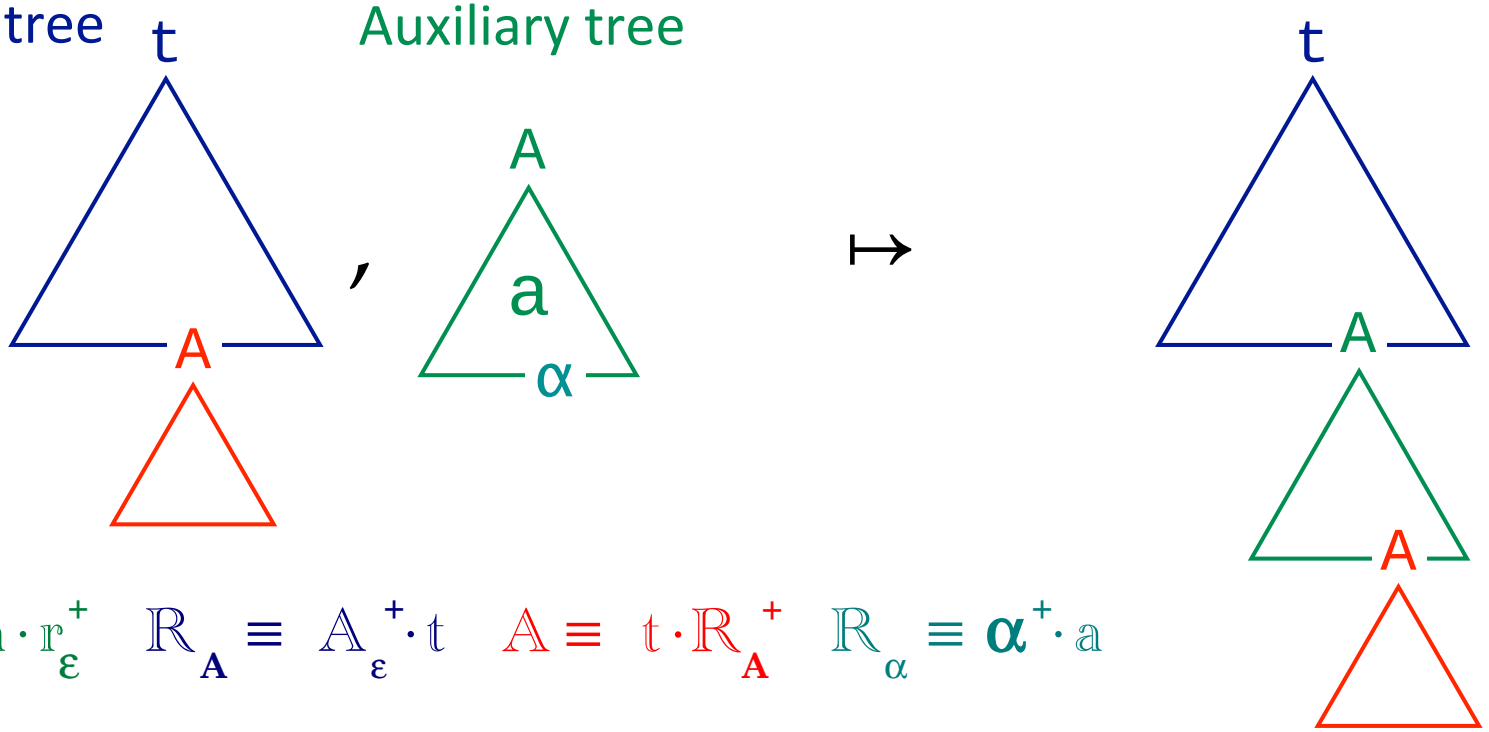
$t - A \otimes [\mathbb{R}_{A} \otimes$	...l of **t** unaffected by adj.	$\mathbb{A}_{\varepsilon} \equiv a \cdot r_{\varepsilon}^{+}$	root symbol of **a** ("**A**")
$+ A \otimes [\mathbb{R}_{\alpha}$	reposition $[_{A} \cdots]$ in **t**	$\mathbb{R}_{A} \equiv \mathbb{A}_{\varepsilon}^{+} \cdot t$	location of **A** in **t**
$+ a \otimes \mathbb{R}_{A}$	embed **a** in place of **A**	$\mathbb{A} \equiv t \cdot \mathbb{R}_{A}^{+}$	subtree **A** in **t**
$- \alpha \otimes [\mathbb{R}_{\alpha} \otimes \mathbb{R}_{A}]$	remove α from embedded **a**	$\mathbb{R}_{\alpha} \equiv \alpha^{+} \cdot a$	location of α in **a**

Tree Adjoining

Initial tree

Auxiliary tree

TA:



$$A_{\varepsilon} \equiv a \cdot r_{\varepsilon}^{+} \quad R_{A} \equiv A_{\varepsilon}^{+} \cdot t \quad A \equiv t \cdot R_{A}^{+} \quad R_{\alpha} \equiv \alpha^{+} \cdot a$$

$$TA(t,a) =$$

$$t - A \otimes R_{A}$$
retain all of **t** unaffected by adj.

$$+ A \otimes [R_{\alpha} \otimes R_{A}]$$
reposition $[_{A} \cdots]$ in **t**

$$+ a \otimes R_{A}$$
embed **a** in place of **A**

$$- \alpha \otimes [R_{\alpha} \otimes R_{A}]$$
remove α from embedded **a**

outer products

$$A_{\varepsilon} \equiv a \cdot r_{\varepsilon}^{+}$$
root symbol of **a** ("**A**")

$$R_{A} \equiv A_{\varepsilon}^{+} \cdot t$$
location of **A** in **t**

$$A \equiv t \cdot R_{A}^{+}$$
subtree **A** in **t**

$$R_{\alpha} \equiv \alpha^{+} \cdot a$$
location of α in **a**

inner products

Results in a nutshell

Single-step operation, massively parallel, from distributed encoding of I to distributed encoding of O. But third- rather than first-order functions of the input (higher-order [multipicative] connections required in net).

Key observation: With a single operation simultaneously performing inner & outer products, can:

- *extract* all roles containing a given filler (e.g., a variable symbol X)
- *insert* a given filler (e.g., tree) into all such roles

4 facts about Tensor Product Representations (TPRs)

1. Not larger
   - In cognitive models examined to date, (noisy) alternative schemes utilize at least as many neurons as their (noise-free) proper TPR counterparts
2. No alternatives
   - Known proposed solutions to the problem of encoding symbol structures as activity vectors (which subsumes the variable binding problem) are all special cases of Generalized TPRs.
3. **Full-blown symbol processing**
   - … can be carried out on **fully distributed** TPRs via multilinear operations.

   a. Linearly computable functions
   b. Function-argument application in the λ-calculus
   c. Tree Adjunction as in Tree Adjoining Grammar
   d. bAbI reasoning

Reasoning with TPRs (Lee et al. 2015)

Addressing Facebook's bAbI 20 question-types task (Bordes, yesterday):

Under review as a conference paper at ICLR 2016

REASONING IN VECTOR SPACE: AN EXPLORATORY STUDY OF QUESTION ANSWERING

Moontae Lee*
Department of Computer Science
Cornell University University
Ithaca, NY 14850, USA
moontae@cs.cornell.edu

Xiaodong He, Wen-tau Yih, Jianfeng Gao & Li Deng
Microsoft Research
Redmond, WA 98052, USA
{xiaohe,scottyih,jfgao,deng}@microsoft.com

Paul Smolensky
Department of Cognitive Science
Johns Hopkins University
Baltimore, MD 21218, USA
smolensky@jhu.edu

arXiv 1511.06426

Reasoning with TPRs (extension of Lee et al. 2015)

Early work in progress!*

Developing a general isomorphism:

logical expressions → tensor expressions

- 'A is-at J at time t_1' → @(A, J, t_1) → **@⊗A⊗J⊗$\mathbf{t_1}$**

Applied to bAbI problems, e.g.:

John picked up an apple
John went to the office
John went to the kitchen
John dropped the apple

Where was the apple before the kitchen? → the office

Reasoning with TPRs (extension of Lee et al. 2015)

knowledge base $\mathcal{B}$ = {facts from story} ∪ {inferred facts}

$\mathbb{B} = \mathbf{@}\otimes\mathbf{A}\otimes\mathbf{J}\otimes\mathbf{t} + \prec\otimes\mathbf{t}_1\otimes\mathbf{t}_2\otimes\mathbf{ø} + \cdots$

to construct $\mathcal{B}(t_k)$ / $\mathbb{B}(t_k)$, loop over sentences k in story:

$\mathcal{B}(t_{k-1})$ given

$\mathbb{B}(t_{k-1})$ already computed

inferences from Persistence Axiom

$\forall k,p\in \mathcal{P}.\ p(a_1, \ldots, a_m; t_{k-1}) \Rightarrow p(a_1, \ldots, a_m; t_k)$

$\mathbb{B}(t_k) \leftarrow \mathbb{B}(t_{k-1}) + \mathbf{P}(t_k)\mathbb{B}(t_{k-1})$

$\mathbf{P}(t_k)$ = Persistence matrix (over tensors)

update $\prec$

add $\prec(t_{k-1}, t_k)$

$\mathbb{B}(t_k) \leftarrow \mathbb{B}(t_k) + \mathbb{T}_k$

$\mathbb{T}_k = \prec\otimes\mathbf{t}_{k-1}\otimes\mathbf{T}\mathbf{t}_{k-1}$

$\mathbf{T}$ = time-update matrix; $\mathbf{t}_k = \mathbf{T}\mathbf{t}_{k-1}$

add LF of new sentence

e.g., *John picked up an_Apple*

$\mathbb{B}(t_k) \leftarrow \mathbb{B}(t_k) + \mathbb{L}(t_k)$

e.g. $\mathbf{@}\otimes\mathbf{A}\otimes\mathbf{J}\otimes\mathbf{t}_1 = \mathbb{L}(t_1)$

repeat until no change:

inferences from Transitivity Axiom:

$\forall x,y,z,t,p\in \mathcal{T}.$

$p(x, y, t)$ & $p(y, z, t) \Rightarrow p(x, z, t)$

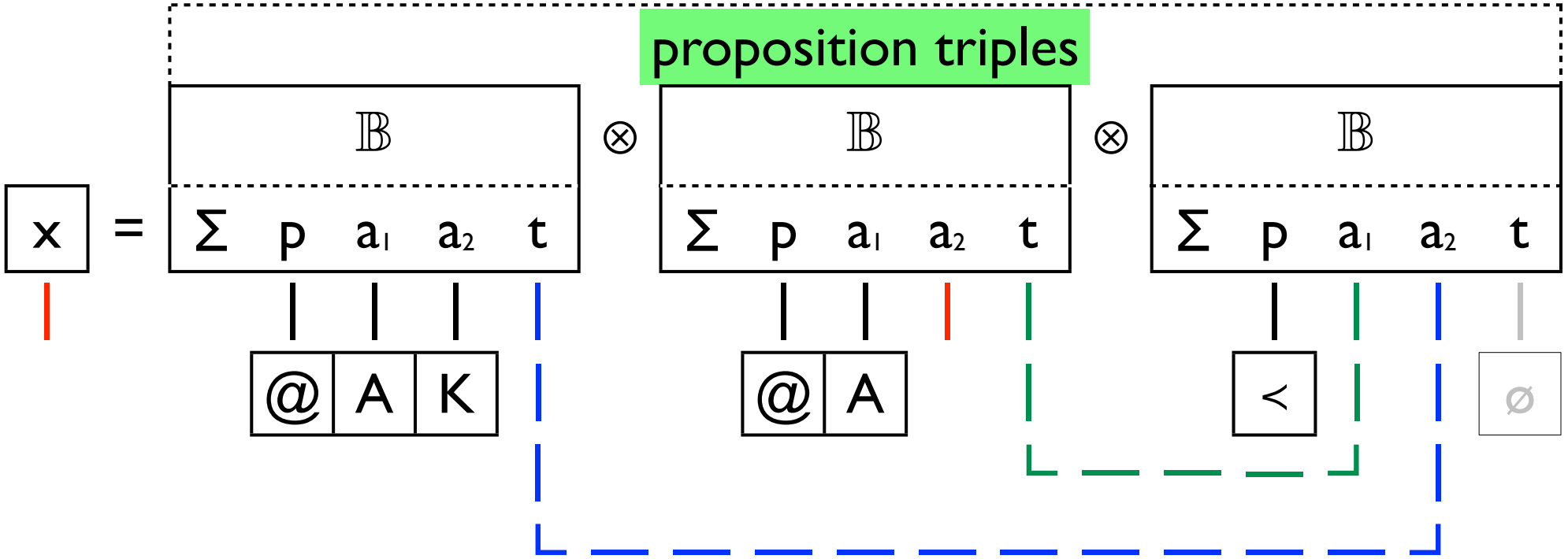
$\mathbb{B}(t_k) \leftarrow \mathbb{B}(t_k) + \mathbf{V}[\mathbb{B}(t_k), \mathbb{B}(t_k); t_k]$

$\mathbf{V}$ = multilinear tensor operation of Transitive Inference

Answering a query

"where is the apple before the kitchen?" →
the x s.t. ∃ times t, t' s.t. Apple is at Kitchen at t', A is at x at t, and t precedes t'

Penrose Tensor Diagram

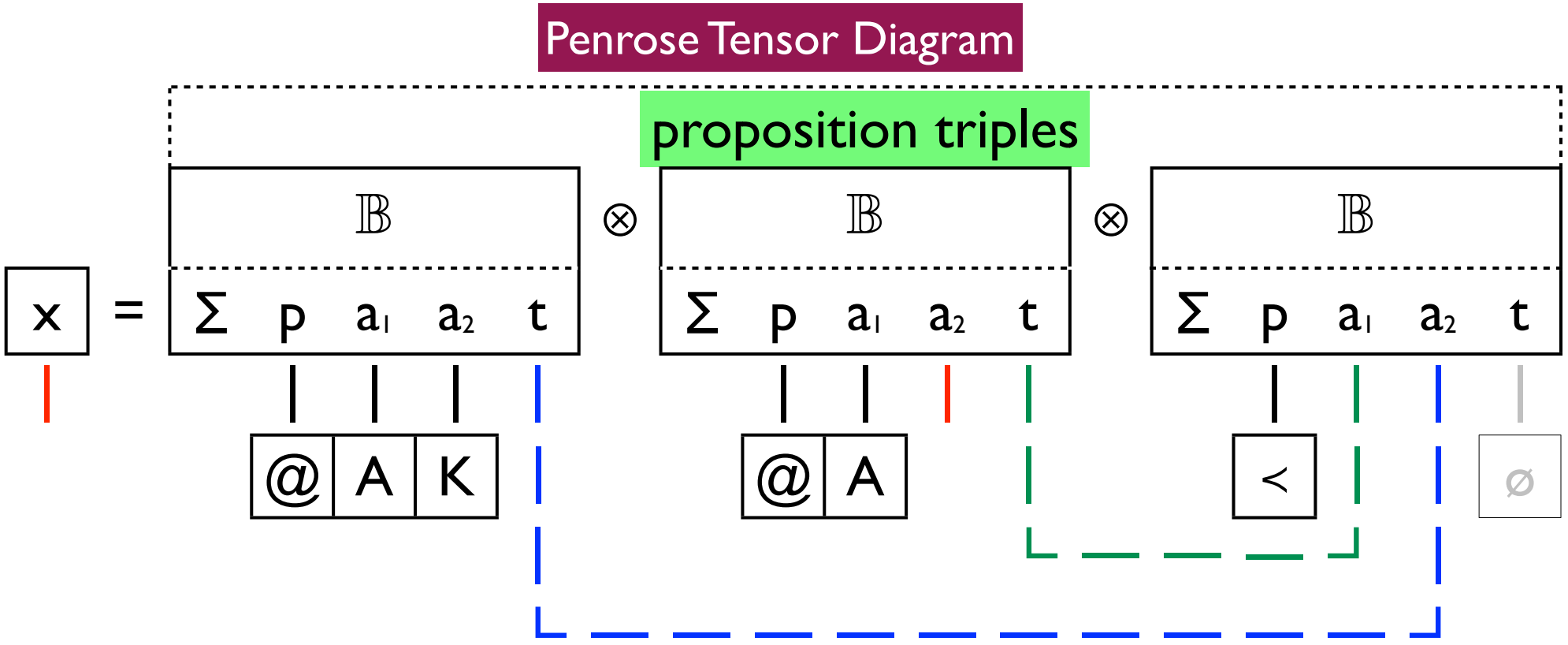


$$\mathcal{O}x.\exists t,t'.\ @(A, K; t') \ \&\ @(A, x; t) \ \&\ <(t, t')$$

the x s.t. ∃ times t, t'	Apple is at Kitchen at t'	&	Apple is at x at t	&	t precedes t'

Answering a query

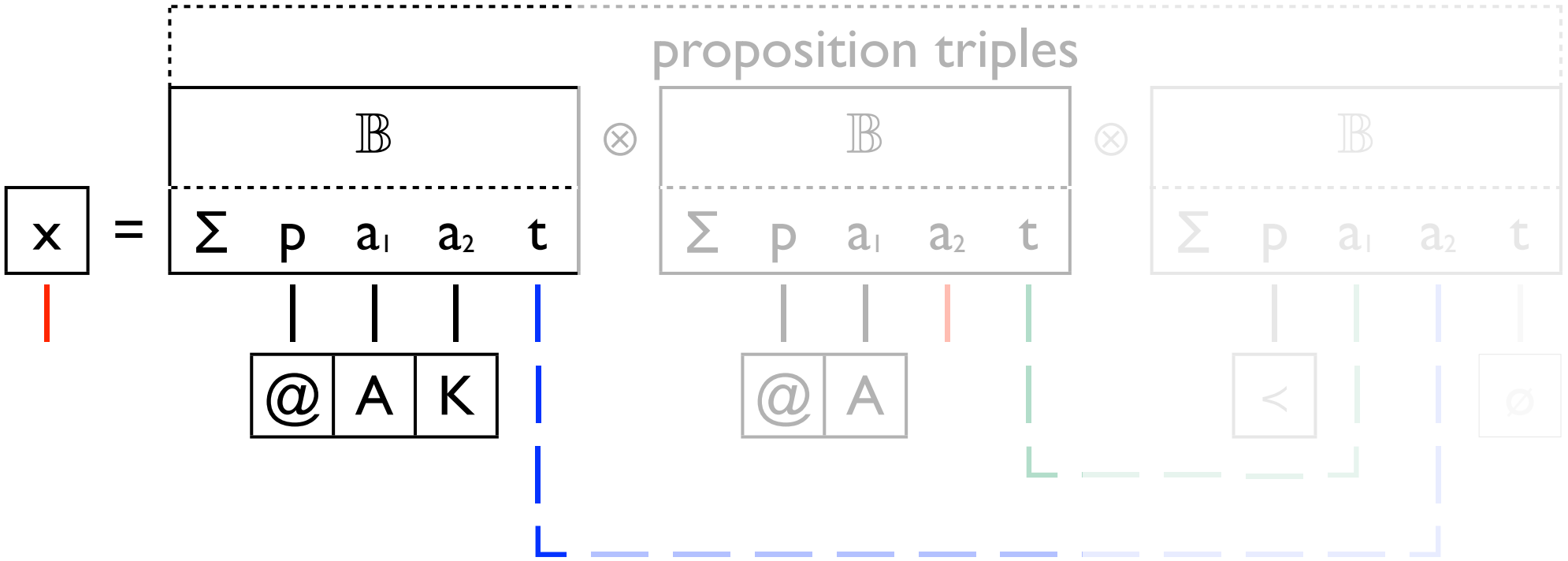
"where is the apple before the kitchen?" →
the x s.t. ∃ times t, t' s.t. Apple is at Kitchen at t', A is at x at t, and t precedes t'



$$x_{\beta_2} = \mathbf{B}_{\pi_1\alpha_1\beta_1\gamma_1}\ \mathbf{B}_{\pi_2\alpha_2\beta_2\gamma_2}\ \mathbf{B}_{\pi_3\alpha_3\beta_3\gamma_3}\ @_{\pi_1}\mathbf{A}_{\alpha_1}\mathbf{K}_{\beta_1}\ @_{\pi_2}\mathbf{A}_{\alpha_2}\ \prec_{\pi_3}\boldsymbol{\varnothing}_{\gamma_3}\ \delta_{\gamma_1\beta_3}\ \delta_{\gamma_2\alpha_3}$$

Answering a query

"where is the apple before the kitchen?" →
the x s.t. ∃ times t, t' s.t. Apple is at Kitchen at t', A is at x at t, and t precedes t'

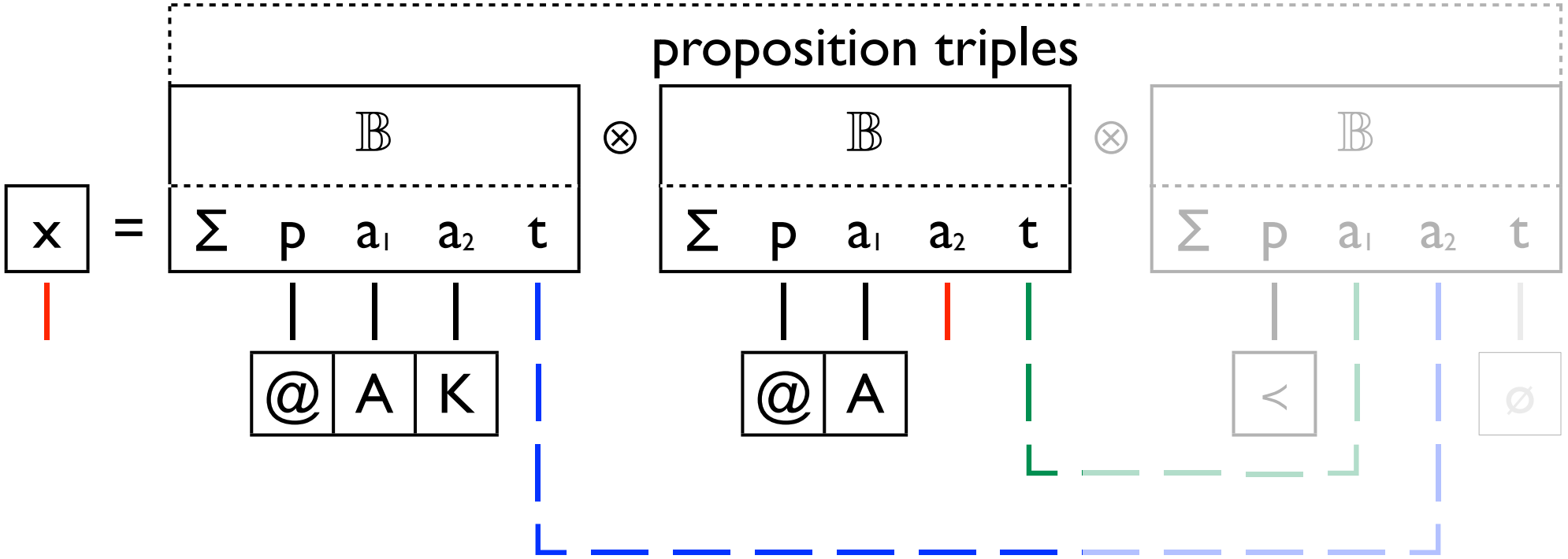


$\mathcal{O}x.\exists t,t'.\ @(\ A,\ K;\ t')$ & $@(\ A,\ x;\ t)$ & $<(\ t,\ t')$

the x s.t. ∃ times t, t'	Apple is at Kitchen at t'	&	Apple is at x at t	&	t precedes t'

Answering a query

"where is the apple before the kitchen?" →
the x s.t. ∃ times t, t' s.t. Apple is at Kitchen at t', A is at x at t, and t precedes t'

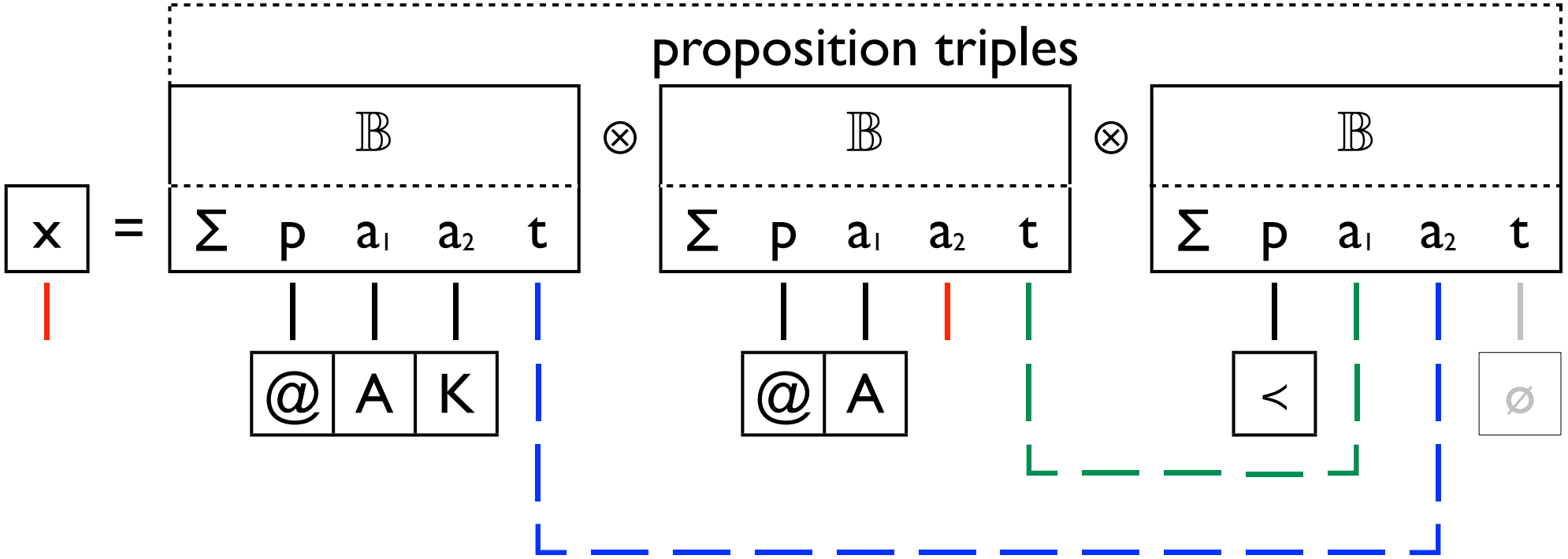


$$\mathcal{O}x.\exists t,t'.\ @(A, K; t') \ \&\ @(A, x; t) \ \&\ <(t, t')$$

the x s.t. ∃ times t, t'	Apple is at Kitchen at t'	&	Apple is at x at t	&	t precedes t'

Answering a query

"where is the apple before the kitchen?" →
the x s.t. ∃ times t, t' s.t. Apple is at Kitchen at t', A is at x at t, and t precedes t'



$$\sigma x.\exists t,t'.\ @(A, K; t') \ \&\ @(A, x; t) \ \&\ <(t, t')$$

the x s.t. ∃ times t, t'	Apple is at Kitchen at t'	&	Apple is at x at t	&	t precedes t'

Simplification for bAbI: Lee et al. 2015

For the case of the 20 question categories in bAbI, simplifications are possible (Moontae Lee): matrix versions of outer/inner product suffice

- @(A, J, t1) → $\mathbf{@\otimes A\otimes J\otimes t_1}$ becomes $\mathbf{A\otimes J = AJ^T}$ for certain questions because can do without *t*, and @ is the only predicate needed

enables virtually 100% performance on all 20 types of questions

- previous SOA: 100% except:
  - ➤ C5: 99.3%, C7: 96.9%, C10: 99%, C17: 72%, C18: 95%, C19: 36%
- from Lee et al. arXiv 1511.06426 paper: 100% except
  - ➤ C5: 99.8%, C16: 99.5%

The good results are less the point than that we can ***understand how*** the good results are obtained and see how the right answers derive from the ***right reasons*** — facilitated by the fact that there is no learning (so far; → representation learning)

Josh Tenenbaum (yesterday): "NNs will not have a role to play in reasoning"

Q_1: Does this kind of work bear on that conjecture?

Q_2: How might it interface with NN representation learning?

4 facts about Tensor Product Representations (TPRs)

1. Not larger
   - In cognitive models examined to date, (noisy) alternative schemes utilize at least as many neurons as their (noise-free) proper TPR counterparts
2. No alternatives
   - Known proposed solutions to the problem of encoding symbol structures as activity vectors (which subsumes the variable binding problem) are all special cases of Generalized TPRs.
3. Full-blown symbol processing
   - … can be carried out on **ful** operations.
4. **Grammatical competence**
   - The competence to generate binary trees that are grammatical according to a given Harmonic Grammar is implementable in fully parallel, fully distributed recurrent networks.

☞ TPRs were an important ingredient in the development of Optimality Theory & Harmonic Grammar

Any rewrite-rule grammar can be implemented as a Harmonic Grammar, including Turing-equivalent Type 0 grammars.

From mechanics to cognition

Want: a general NN architecture $\mathcal{H}$ which

- converges to symbolic computation in the limit

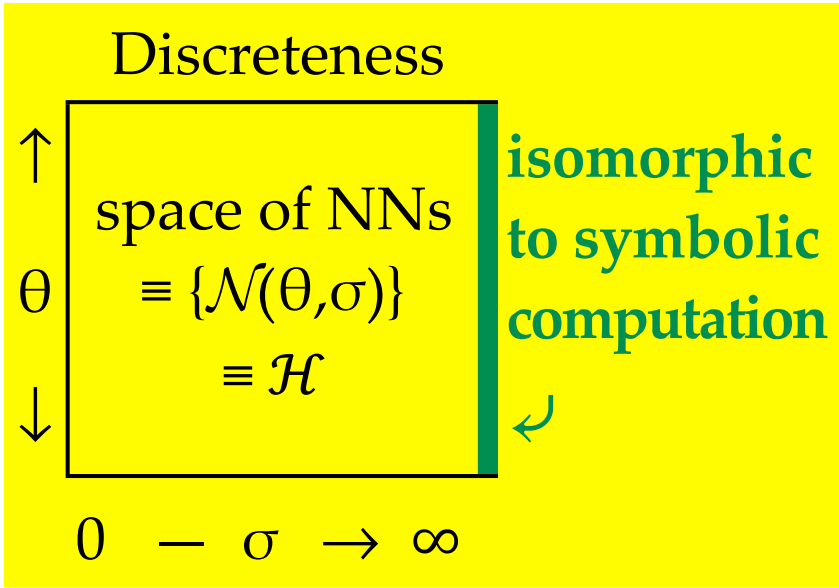
states of the mind are collections of referring symbols assembled into combinatorial structures that are governed by the variable-containing well-formedness constraints of a 'grammar'

From mechanics to cognition

Want: a general NN architecture $\mathcal{H}$ which

- converges to symbolic computation in the limit of some discreteness hyperparameter vector $\sigma \to \infty$, e.g.:
  - ➤ dimensionality of activation vectors ($|\mathrm{NN}|$) encoding symbols $\to \infty$
  - ➤ similarity^{-1} of activation vectors encoding symbols structures $\to \infty$
  - ➤ non-randomness $T^{-1} \to \infty$
  - ➤ processing time $\to \infty$
- can improve upon symbolic cognitive theory
  - ➤ w.r.t. 'macro data'
  - ➤ w.r.t. 'micro data'

☞ As a hypothesis space $\mathcal{H}$ for learning, can always do at least as well as symbolic theory: can choose $\sigma \to \infty$



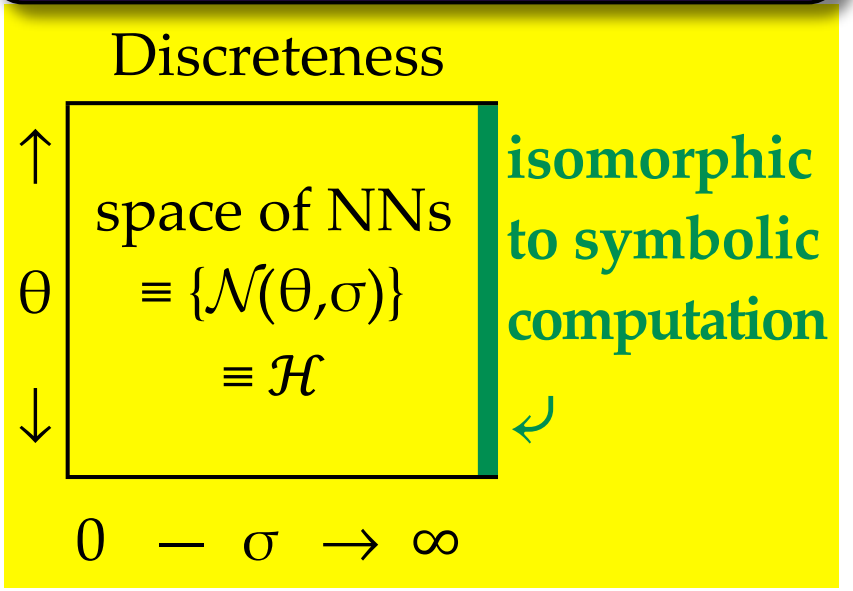
From mechanics to cognition

Want: a general NN architecture $\mathcal{H}$ which

- converges to symbolic computation in the limit of some discreteness hyperparameter vector $\sigma \to \infty$, e.g.:
  - dimensionality of activation $\to \infty$
  - similarity^{-1} of activation ve $\to \infty$
  - non-randomness $T^{-1} \to \infty$
  - processing time $\to \infty$
- can improve upon symbolic cognitive theory
  - w.r.t. 'macro data'
  - w.r.t. 'micro data'

☞ As a hypothesis space $\mathcal{H}$ for learning, can always do at least as well as symbolic theory: can choose $\sigma \to \infty$

☞ $\mathcal{H}$ built on Tensor Product Representations (TPRs) ⇒ can *program* (not yet *learn*) symbolic computations



References

Smolensky, P., Goldrick, M. & Mathis, D. 2014. Optimization and quantization in gradient symbol systems: A framework for integrating the continuous and the discrete in cognition. *Cognitive Science*.

Eliasmith, C. 2013. *How to build a brain: A neural architecture for biological cognition*. Oxford University Press.

Smolensky, Paul. 2012. Symbolic functions from neural computation. *Philosophical Transactions of the Royal Society — A: Mathematical, Physical and Engineering Sciences*.

Marcus, G. F. 2001. *The algebraic mind: Integrating connectionism and cognitive science*. MIT Press.

Smolensky, P. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist networks. *Artificial Intelligence*. [Reprinted in G. Hinton, (Ed.), 1990, *Connectionist symbol processing*, Elsevier/MIT Press.]

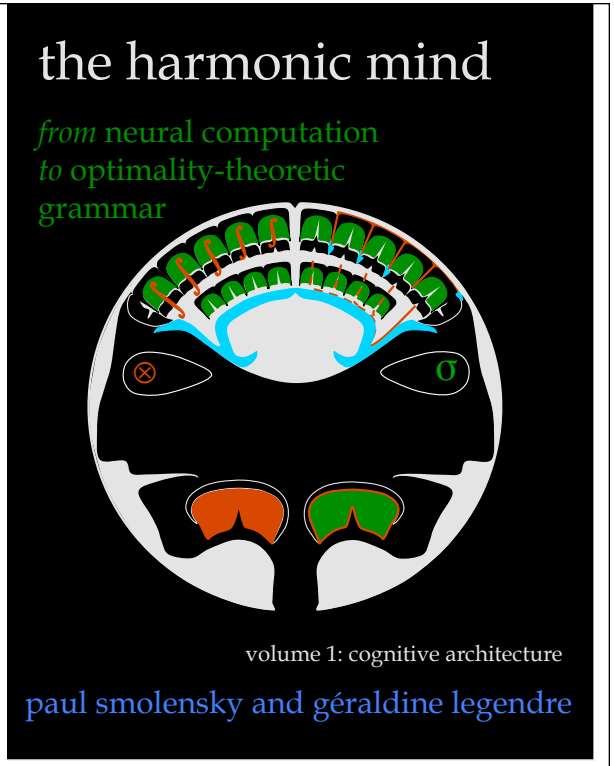
NEOLITHIC NIPS

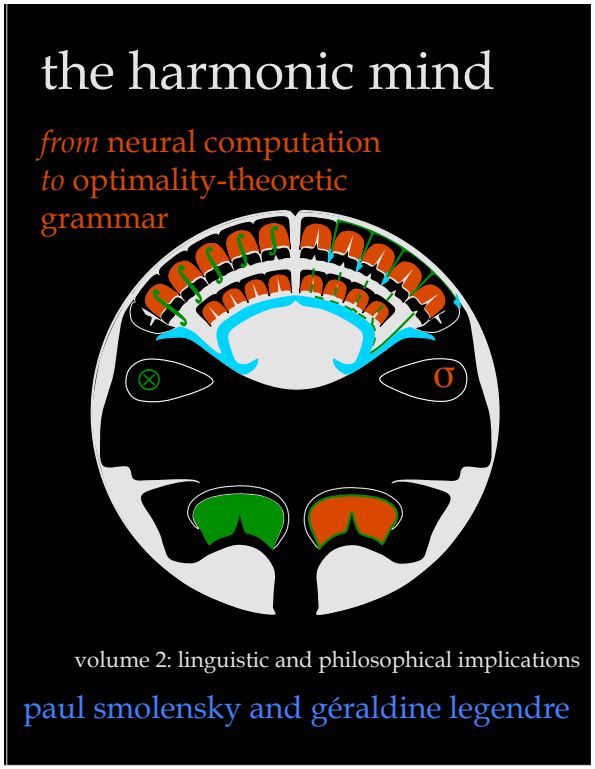
Plate, T. A. 1993. Holographic recurrent networks. In *NIPS 5*.

Smolensky, P. 1993. Harmonic Grammars for formal languages. *NIPS 5*.

Legendre, G., Miyata, Y., & Smolensky, P. 1991. Distributed recursive structure processing. *NIPS 3*.

Smolensky, P. 1988. Analysis of distributed representation of constituent structure in connectionist systems. *NIPS 0*.





That's all folks!