PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

Phil. Trans. R. Soc. A (2012) **370**, 3543–3569
doi:10.1098/rsta.2011.0334

Review

Symbolic functions from neural computation

By Paul Smolensky*

Cognitive Science Department, Johns Hopkins University, 237 Krieger Hall, 3400 North Charles Street, Baltimore, MD 21218, USA

Is thought computation over ideas? Turing, and many cognitive scientists since, have assumed so, and formulated computational systems in which meaningful concepts are encoded by symbols which are the objects of computation. Cognition has been carved into parts, each a function defined over such symbols. This paper reports on a research program aimed at computing these symbolic functions without computing over the symbols. Symbols are encoded as patterns of numerical activation over multiple abstract neurons, each neuron simultaneously contributing to the encoding of multiple symbols. Computation is carried out over the numerical activation values of such neurons, which individually have no conceptual meaning. This is massively parallel numerical computation operating within a continuous computational medium. The paper presents an axiomatic framework for such a computational account of cognition, including a number of formal results. Within the framework, a class of recursive symbolic functions can be computed. Formal languages defined by symbolic rewrite rules can also be specified, the subsymbolic computations producing symbolic outputs that simultaneously display central properties of both facets of human language: universal symbolic grammatical competence and statistical, imperfect performance.

Keywords: cognitive science; neural networks; connectionism; linguistics; language; optimality theory

1. Introduction

The classical theory of computation initiated by Turing [1] has provided the foundation for mainstream theories of cognition and intelligence since the inception of cognitive science and artificial intelligence in the 1950s [2]. This paper presents another type of computation that has developed more recently within cognitive science, driven primarily by two goals: improved formal characterization of human mental processing—including mental grammars—and improved reduction to neural computation. These newer methods develop a distinction, absent in earlier computational theory, between three levels

*smolensky@jhu.edu

Electronic supplementary material is available at http://dx.doi.org/10.1098/rsta.2011.0334 or via http://rsta.royalsocietypublishing.org.

One contribution of 18 to a Theme Issue 'The foundations of computation, physics and mentality: the Turing legacy'.

of description, which I will here call *symbolic*, *vectorial* and *neural*. These distinguish the formal characterization of the mind, using recursive functions over discrete symbols, from the characterization of the brain, using continuous, parallel, numerical computation. The vectorial level provides an interlingua. The descriptions at all three levels characterize a single computer, the mind/brain [3].

A key departure from earlier theory (including Turing's work on network machines [4]) is that machine computations operate only at a level lower than that of meaning. A conceptually meaningful entity is realized as an activation pattern—an activation vector—over many abstract neurons, each neuron simultaneously participating in the realization of multiple meaningful entities. Knowledge is realized in connections between elements (neurons) that are individually meaningless but collectively meaningful. The objects of computational manipulation are not meaningful (the neural-level description), and the meaningful elements are not objects of computational manipulation (the symbolic-level description) [5–7]. Nonetheless, we will see that the proposed vectorial computation provides the technical bridge for computing meaningful recursive symbolic functions using subsymbolic, neural computation.

2. Overview

The theory of human cognition owes much of its progress to the principle (formalized in §3) that cognition within some domain (e.g. arithmetic) consists in a system of functions that take inputs, and produce outputs, which are structures built of symbols that typically refer to elements in the domain (e.g. numbers).[1] This raises two basic questions—and a meta-question: How should these functions be specified by cognitive scientists? How are they computed within the brain? How are these two questions connected? The answers developed here are roughly these.

How should cognitive functions be specified? In §6, we adopt the conventional assumption that cognitive functions can usefully be specified via recursive equations; but in §7, we conclude that—at least in one key domain, the theory of human language—an alternative, specification by constraints on (input, output) pairs, has important advantages.

How are cognitive functions computed? In §6, we challenge the assumption that cognitive functions specified via recursive equations should be computed in the conventional manner, by sequentially computing the primitive functions in terms of which those equations are ultimately defined. Instead, we show how cognitively relevant functions in a certain class can be computed in one, massively parallel step, in which neural activations encoding the input symbol structure are transformed by a simple linear transformation to output activations that encode the output symbol structure specified by the target function. In this approach, symbolic functions are computed, but not by symbolic computation: there is no algorithm over symbols that describes the internal processes by which input becomes output. There *is* a formal specification of the process, however, using the primitives of neural computation. If this account of cognition is on the right track, then the kinds of symbol-manipulating algorithms sought by traditional artificial intelligence and cognitive theories do not exist, despite the existence of symbolic descriptions of cognitive functions.

[1]In accepting this principle, we differ from most research programmes on neural network (or 'connectionist') theories of cognition, which deny the validity of our symbolic level [8].

How are these questions connected? If symbolic cognitive functions are computed by neural, not symbolic, computation, then natural descriptions of the functions computed by neural networks provide a promising candidate framework for specifying cognitive functions. This leads, in §7, to the description of these functions in terms of optimization over constraints, at all three levels.

Addressing these questions in §§6 and 7 requires laying considerable groundwork, the foundation of which is a mapping from a space of symbol structures to a vector space. This embedding is presented in §4, and the relation between the similarity of symbol structures and the similarity of their embeddings in the vector space is then analysed in §5.

The paper provides a concise synopsis of the formal core of *The harmonic mind* [9], in which many of the results presented here are derived. Integrated into this synopsis are new results (§§5 and 7) and current research (§8). The paper initiates a research programme developing axiomatic theory in cognitive science, motivated by the need for more precise characterizations of the central notions of cognitive theory, which are highly abstract and therefore in need of formal support to escape crippling vagueness.

(*a*) *Summary of results*

In this paper, we take up the following topics: the decomposition of cognition into component functions (§3); the representation of the inputs and outputs of these cognitive functions (§4); the similarity structure of these representations (§5); a recursive-function-theoretic approach to specifying cognitive functions, and neural computation of those functions (§6); a formal-language-theoretic approach to specifying cognitive functions, and neural computation of those functions (§7); and the challenge of producing symbolically interpretable outputs from neural computation (§8).

Each of these topics (except the last) is treated at three levels of description, in separate subsections: (*a*) the abstract symbolic level; (*c*) the neural level; and, mediating between them, (*b*) the vectorial level.[2]

The main results, simplified, are approximately these (numbers identify the corresponding theorems):

Theorem 4.8. *The fundamental symbolic data structure we assume, binary trees, can be embedded in an infinite-dimensional vector space in such a way that the basic data construction and data access operations are linear transformations.*

Theorem 5.7. *Dissimilarity between two symbolic representations—as measured by differences in their constituents and the structural relations between their constituents—is given by the distance in the vector space between the two vectors 'embedding' those representations.*

Theorem 6.5. *Each function in the class defined as the closure under composition of the basic binary tree and symbol-mapping operations can be computed by a linear transformation over the vectors embedding the function's input and output.*

[2]A notational challenge arises because most elements of the theory need three symbols, one for each level. The choice here is to use essentially the same symbol for all three, but to distinguish them by font and/or style (italics, bold, etc.).

Theorem 6.6. *The infinite matrices implementing these linear transformations have a particularly simple form, the tensor product of the infinite identity matrix and a finite matrix that specifies the particular recursive function.*

Theorem 7.2. *A formal language—defined standardly by sequentially applied symbol-rewriting rules—can be specified as the structures that optimize (maximize) a numerical measure of symbolic well-formedness called (symbolic) Harmony.*

Theorem 7.5. *A natural language specified in a new grammatical framework called Optimality Theory can also be characterized as the optima of symbolic Harmony.*

Theorem 7.7. *The symbolic Harmony of a structure can be computed as the value, for the vector that embeds that structure, of a quadratic form called (neural) markedness Harmony.*

Theorem 7.11. *In evaluating errors, the dissimilarity between an output symbolic structure and an input symbolic structure considered as the desired output can be computed as the value, at the vectors embedding the input and output structures, of a bilinear form called neural faithfulness Harmony.*

Theorem 7.12. *The local optima of total network Harmony—the sum of markedness and faithfulness network Harmony—can be computed by a deterministic neural network.*

Theorem 7.13. *The global optima of total network Harmony can be computed by a stochastic neural network.*

Theorem 8.1. *The requirement that the output of a neural computation be a vector that is the embedding of a symbol structure can be met via a network dynamics that creates an attractor at each such vector.*

To obtain (or merely state) these results as theorems, a certain degree of formalization is necessary.

Through the course of the paper, it may prove useful to refer back to the following synopsis, in which organization of results is orthogonal to that of the paper: it gives the total picture at each level separately. It also identifies all the principal elements of the theory, providing an informal glossary of their notation.

At this point, the reader may skip directly to §3 without loss of continuity.

(*b*) *Synopsis by level*

At the most abstract level, the different types of information encoded mentally (and their mutual relations) are located within a cognitive macro-architecture graph $\mathfrak{A}$ (§3*a*); these mental representations are characterized as systems $\mathcal{S}$ of structures built of symbols filling specified structural roles (§4*a*); dissimilarity metrics, motivated by empirical patterns of cognitive performance, are defined with respect to (w.r.t.) a set of abstract relations $\{\mathsf{R}_i\} = \mathcal{R}$ among constituents of mental representations (§5*a*); the symbolic functions computed over $\mathcal{S}$ within $\mathfrak{A}$ are characterized, including a set $\mathcal{P}$ of recursive functions (§6*a*) and the set of formal languages $\mathcal{L}_{\mathcal{G}}$ generated through sequential derivation by rewrite-rule grammars $\mathcal{G}$; these languages and dissimilarity metrics are cast in the form of numerical Harmony functions $H_{\mathcal{G}}$ (theorem 7.2) and $H_{\mathcal{R}}$ (definition 5.4), the correct mental representations being those that maximize Harmony; Harmony

functions H_{G} can be used to characterize the grammars of natural languages, including Optimality-Theoretic grammars $\mathbb{G}$ within a system $\mathfrak{O}$ in terms of which the typology of possible human languages can be formally calculated (§7*a*).

At a level intermediate between the symbolic and the neural, linear and quasi-linear classes of dynamical systems in vector spaces are defined (§3*b*); a model of the symbol system of mental states $\mathcal{S}$ is specified abstractly by a realization mapping $\boldsymbol{\Psi}$ of symbol structures into a vector space S (§4*b*); symbolic structural-relation dissimilarity $H_{\mathcal{R}}$ is realized as vector distance within S (§5*b*); recursive functions $g \in \mathcal{P}$ and Harmony functions H_{G} are reduced to linear transformations on S, W_g (§6*b*) and W_{G} (theorem 7.7).

At the neural level, S is modelled with $\mathbb{R}^n$, the states of a neural network $\mathcal{N}$, by positing a distinguished neural basis (§3*c*) in terms of which: mental representations $\mathbf{s}$ and the realization mapping $\boldsymbol{\Psi}$ are explicitly specified numerically (§4*c*); Euclidean distance is explicitly computable, and $H_{\mathcal{R}}$ is reduced to neural faithfulness Harmony H_{F} (theorem 7.11); the linear transformations W_g and W_{G} are instantiated as numerical matrices $\mathbf{W}_g$ (§6*c*) and $\mathbf{W}_{\mathrm{G}}$ (§7*c*); $H_{\mathcal{G}}$ is reduced to neural markedness Harmony H_{M} (definition 7.6); deterministic spreading activation can be used to compute local maxima of the network Harmony $H_{\mathcal{N}} = H_{\mathrm{F}} + H_{\mathrm{M}}$ (theorem 7.12); and stochastic spreading activation dynamics $\mathcal{D}_{\mathrm{opt}}$ within a diffusion network $\mathcal{N}^{\mathrm{T}}$ can be used to compute global maxima of network Harmony (theorem 7.13); but a (deterministic) quantization dynamics $\mathcal{D}_{\mathrm{quant}}$ is needed to output an activation pattern $\mathbf{s}$ that is interpretable as a discrete symbolic state s, with $\mathbf{s} = \boldsymbol{\Psi}(s)$ (theorem 8.1); a total dynamics combining $\mathcal{D}_{\mathrm{opt}}$ and $\mathcal{D}_{\mathrm{quant}}$ yields the λ-diffusion dynamics $\mathcal{D}_{\lambda}$ (definition 8.2) of a network $\mathfrak{N}$ that computes symbolic mental representations, enabling models simultaneously capturing central features of both the idealized computational capacity of human linguistic competence and the errorful real-time computation of human linguistic performance.

3. Cognitive macro-architecture

At a coarse level of description, a cognitive architecture is an interconnected collection of components, operating in parallel, each of which processes information of a certain type (e.g. for language, these types include orthographic, phonological, syntactic and semantic information: see ① in figure 1). Knowledge of the structure of a type of information is contained within the corresponding component, and knowledge of the relations between the types is contained in the links connecting components. A component computes a function: the input comes along inward-directed links from neighbouring components, and the output is, in turn, processed by knowledge in the outward-directed links to generate input to other components. The flowcharts ('box and arrow' diagrams) conventionally used to depict such an architecture instantiate a kind of graph structure.

(*a*) *Symbolic graph structure*

Definition 3.1. A cognitive macro-architecture $\mathfrak{A}$ is a directed graph together with:

a. For each node ('cognitive component' or 'module') $\mathcal{M}^{\gamma}$ in $\mathfrak{A}$,

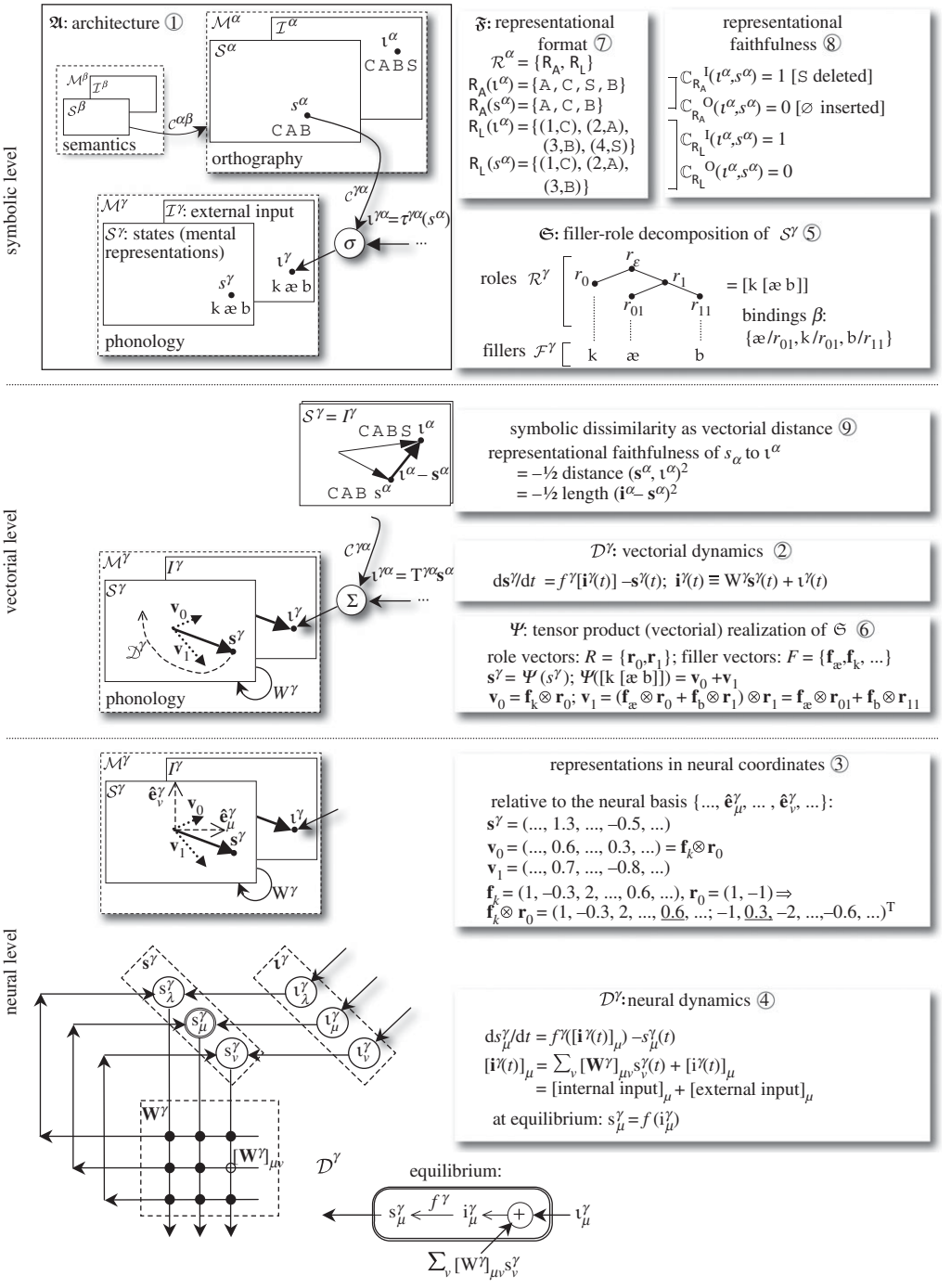

Figure 1. A schematic of the proposed theory for neural computation of symbolic cognitive functions.

(i) a set $\mathcal{S}^\gamma$, the 'state space' of $\mathcal{M}^\gamma$; an $s^\gamma \in \mathcal{S}^\gamma$ is called a 'mental representation';

(ii) a set $\mathcal{I}^\gamma$, the 'input space' of $\mathcal{M}^\gamma$;

(iii) a function $\sigma^\gamma : 2^{\mathcal{I}^\gamma} \to \mathcal{I}^\gamma$ for aggregating 'partial inputs' to $\mathcal{M}^\gamma$ (see b(ii)).

b. For each edge ('connection' or 'pathway') $\mathcal{C}^{\gamma\alpha}$, from $\mathcal{M}^\alpha$ to $\mathcal{M}^\gamma$, in $\mathfrak{A}$,

(i) a transformation $\tau^{\gamma\alpha}$: $\mathcal{S}^\alpha \to \mathcal{I}^\gamma$ (the 'inter-component transformation');

(ii) $\iota^{\gamma\alpha} \equiv \tau^{\gamma\alpha}(s^\alpha) \in \mathcal{I}^\gamma$ is called the 'partial input' from $\mathcal{M}^\alpha$ to $\mathcal{M}^\gamma$ when $\mathcal{M}^\alpha$ is in state $s^\alpha \in \mathcal{S}^\alpha$.

c. $\iota^\gamma \equiv \sigma^\gamma(\{\iota^{\gamma\alpha} \mid \text{ edge } \mathcal{C}^{\gamma\alpha} \text{ is in } \mathfrak{A}\}) \in \mathcal{I}^\gamma$ is the 'external input' to $\mathcal{M}^\gamma$.

Henceforth, we assume given some specific architecture $\mathfrak{A}$ in which, for all $\mathcal{M}^\gamma \in \mathfrak{A}$, the spaces $\mathcal{I}^\gamma$ and $\mathcal{S}^\gamma$, while conceptually distinct, are formally identical; $\mathcal{I}^\gamma$ functions as a 'target' towards which the components external to $\mathcal{M}^\gamma$ drive its state s^γ. Knowledge within $\mathcal{M}^\gamma$ will generally partially resist this pressure.

(*b*) *Vectorial model*

The vectorial level models each $\mathcal{S}^\gamma = \mathcal{I}^\gamma$ as a vector space $S^\gamma = I^\gamma$ (figure 1 ②). The vector space S^γ constitutes an abstract information-encoding medium—a space of cognitive states, of mental representations—that resides in continuous mathematics, rather than the discrete mathematics of symbolic encodings. We study two example types of vectorial models here.

In quasi-linear models, the transformations between components are linear, combining by addition; a potentially nonlinear function $\boldsymbol{f}$ and a linear transformation of S^γ to itself determines how the state of S^γ at one time, and the external input to S^γ from other components at that time, move the state forward.

Example 3.2. In a *quasi-linear dynamics* $\mathcal{D}$ [10] for the architecture $\mathfrak{A}$, for all cognitive components $\mathcal{M}^\gamma$ and $\mathcal{M}^\alpha$:

a. the operation $\tau^{\gamma\alpha}$ on the link to $\mathcal{M}^\gamma$ from $\mathcal{M}^\alpha$ is a linear transformation $\mathrm{T}^{\gamma\alpha}$: $S^\alpha \to I^\gamma$, $\boldsymbol{\iota}^{\gamma\alpha} = T^{\gamma\alpha}\mathbf{s}^\alpha$;

b. σ^γ is summation: the external input to $\mathcal{M}^\gamma$, $\boldsymbol{\iota}^\gamma$, is the sum of the partial inputs $\boldsymbol{\iota}^{\gamma\alpha}$ from neighbouring components $\mathcal{M}^\alpha$: $\boldsymbol{\iota}^\gamma = \sum_\alpha \boldsymbol{\iota}^{\gamma\alpha}$;

c. a linear transformation W^γ: $S^\gamma \to S^\gamma$ (the 'intra-component input transformation') and a function $\boldsymbol{f}^\gamma$: $S^\gamma \to S^\gamma$ (the 'activation function') define $\mathcal{D}^\gamma$, the internal dynamics of $\mathcal{M}^\gamma$, by

$$\mathrm{d}\mathbf{s}^\gamma/\mathrm{d}t = \boldsymbol{f}^\gamma[\mathbf{i}^\gamma(t)] - \mathbf{s}^\gamma(t), \quad \mathbf{i}^\gamma(t) \equiv \mathrm{W}^\gamma\mathbf{s}^\gamma(t) + \boldsymbol{\iota}^\gamma(t).$$

$\mathrm{W}^\gamma\mathbf{s}^\gamma$ is the internally generated input to $\mathcal{M}^\gamma$; it combines linearly with the external input $\boldsymbol{\iota}^\gamma$.

Note that, at equilibrium, we must have $\mathbf{s}^\gamma(t) = \boldsymbol{f}^\gamma[\mathbf{i}^\gamma(t)]$; $\boldsymbol{f}^\gamma$ gives the equilibrium relation between input $\mathbf{i}^\gamma$ and activation $\mathbf{s}^\gamma$.

A simpler type of model we will use (ultimately, as exemplified in figure 2) is as follows.

Example 3.3. A *linear associator* [11] is a simple sub-architecture consisting of two components, $\mathcal{M}^\alpha$ and $\mathcal{M}^\gamma$, with a single (uni-directional) pathway, from $\mathcal{M}^\alpha$ to $\mathcal{M}^\gamma$, in which

a. the operation $\boldsymbol{\tau}^{\gamma\alpha}$ is a linear transformation $\mathrm{T}^{\gamma\alpha}: S^\alpha \to I^\gamma$, producing external input $\boldsymbol{\iota}^\gamma = \mathrm{T}^{\gamma\alpha}\mathbf{s}^\alpha$ and
b. the state $\mathbf{s}^\gamma$ equals the external input $\boldsymbol{\iota}^\gamma : \mathbf{s}^\gamma = \mathrm{T}^{\gamma\alpha}\mathbf{s}^\alpha$.

Both types of model are spelled out more explicitly and intuitively at the neural level.

(*c*) *Neural computation: cognitive micro-architecture*

The neural level (figure 1 ③) invokes a coordinate system (with axes defined by a 'neural basis' $\{\hat{\mathbf{e}}^\gamma_\mu\}^n_{\mu=1}$ [12]) for S^γ in which the list of coordinates for any vector $\mathbf{s}^\gamma \in S^\gamma$, $(s^\gamma_1, s^\gamma_2, \ldots, s^\gamma_n)$, is the list of activation values of an enumerated set of n 'abstract neurons' in a network $\mathcal{N}^\gamma$; in this model of the vectorial theory, the mental state s^γ is realized by an 'activation pattern' described by a vector in $\mathbb{R}^n$. We assume given a neural basis for every S^γ.

Generally, neural computation is a type of analogue computation: a dynamical system in $\mathbb{R}^n$, fundamentally continuous in time, defined by differential equations $\mathcal{D}^\gamma$ that describe how the n neurons function as parallel processors (how they 'spread activation'; figure 1 ④) [13]. A particular computation is specified by values for the parameters in $\mathcal{D}^\gamma$; these parameter values are interpreted in parallel by the machine that evolves in accordance with $\mathcal{D}^\gamma$. In simpler cases, similar to the linear associator (example 3.3), there may be no dynamics, just static relations between component states.

Definition 3.4. Given a quasi-linear dynamics $\mathcal{D}^\gamma$ for S^γ, we define the following, relative to the neural basis.

a. The inter-component linear transformation $\mathrm{T}^{\gamma\alpha}$ is realized as the matrix $\mathbf{T}^{\gamma\alpha}$, the element $[\mathbf{T}^{\gamma\alpha}]_{\mu\nu}$ being the 'connection strength' or 'weight', of a 'connection' from neuron ν in $\mathcal{N}^\alpha$ to neuron μ in $\mathcal{N}^\gamma$. (This applies to a linear associator as well.)
b. The linear intra-component input transformation W^γ is realized as the weight matrix $\mathbf{W}^\gamma$, the element $[\mathbf{W}^\gamma]_{\mu\nu}$ being the weight of a connection from neuron ν in $\mathcal{N}^\gamma$ to neuron μ in $\mathcal{N}^\gamma$.
c. The *total input* to the neurons in $\mathcal{N}^\gamma$ is $\mathbf{i}^\gamma \equiv \mathbf{W}^\gamma\mathbf{s}^\gamma + \boldsymbol{\iota}^\gamma$, where $\boldsymbol{\iota}^\gamma$ is the external input from other components, while $\mathbf{W}^\gamma\mathbf{s}^\gamma$ (the matrix–vector product of $\mathbf{W}^\gamma$ and $\mathbf{s}^\gamma$) is the input generated internally within $\mathcal{N}^\gamma$. At neuron μ of $\mathcal{N}^\gamma$, the total input is $[\mathbf{i}^\gamma]_\mu = \sum_\nu [\mathbf{W}^\gamma]_{\mu\nu}[\mathbf{s}^\gamma]_\nu + [\boldsymbol{\iota}^\gamma]_\mu$, i.e. the weighted sum of the activation values of μ's neighbours ν within $\mathcal{N}^\gamma$ plus the external input to μ.

Definition 3.5. To the requirements defining a quasi-linear dynamics at the vectorial level (example 3.2) we now add:

a. the *locality* requirement $\boldsymbol{f}^\gamma: \mathbf{i}^\gamma = (i_1, i_2, \ldots, i_n) \mapsto (f^\gamma(i_1), f^\gamma(i_2), \ldots, f^\gamma(i_n))$ for some $f^\gamma: \mathbb{R} \to \mathbb{R}$ called the 'neuron activation function'; and

b. the *monotonicity* requirement that the activation function f^γ be non-decreasing.
If, in addition, f^γ is the identity function, $f^\gamma(i) = i$, then the dynamics $\mathcal{D}^\gamma$ is *linear*.

Owing to locality, the differential equation for neuron μ depends only on the activation levels of units connected to μ, the weights on connections to neuron μ and the external input to unit μ:

$$\mathrm{d}\mathrm{s}_\mu^\gamma/\mathrm{d}t = f^\gamma([\mathbf{i}^\gamma(t)]_\mu) - \mathrm{s}_\mu^\gamma(t); \quad [\mathbf{i}^\gamma(t)]_\mu = \sum_\nu [\mathbf{W}^\gamma]_{\mu\nu}\mathrm{s}_\nu^\gamma(t) + [\boldsymbol{\iota}^\gamma(t)]_\mu.$$

Monotonicity implies that, at equilibrium, the higher the input i_μ^γ to neuron μ, the greater its activation value $\mathrm{s}_\mu^\gamma = f(\mathrm{i}_\mu^\gamma)$.

Henceforth, we focus primarily on a single component $\mathcal{M}^\gamma$, and generally drop γ from the notation.

4. Symbolic functions

A component $\mathcal{M}$ of an architecture $\mathfrak{A}$ computes a function, producing a mental representation as output.

(*a*) *Mental representations as symbol structures*

At the most abstract level, mental representations in $\mathcal{S}$ are symbol structures [14] (figure 1 ⑤); e.g. in an orthographic component, a string of mental letters such as CABS; in a phonological component, a string of mental phonemes such as kæbz (or less simplistically, the binary tree [k [æ [b z]]]); in a syntactic component, a phrase-structure tree such as [S [NP Frodo] [VP [V lives]]] (simplifying greatly). The key idea now is to regard a symbol structure as a set of internal roles, each filled by a constituent symbol structure (see ⑤).

Definition 4.1. A *filler-role decomposition* $\mathfrak{S} = (\mathcal{S}, \mathcal{F}, \mathcal{R}, \beta)$ consists of three sets, the 'structures' $\mathcal{S}$, the 'fillers' $\mathcal{F}$ and the 'roles' $\mathcal{R}$, and a one-to-one function $\beta: \mathcal{S} \to 2^{\mathcal{F}\times\mathcal{R}}$; the N pairs constituting $\beta(s)$, written as $\{f_k/r_k\}_{k=1}^N$, are the 'filler/role bindings', or 'constituents', of $s \in \mathcal{S}$ [15].

A symbolic data structure that is widely applicable for mental representations is the binary tree; the artificial-intelligence language Lisp deploys it exclusively [16]. Two filler-role decompositions are as follows [17].

Example 4.2. The following defines the *canonical filler-role decomposition of binary trees*, $\mathfrak{T}_\mathrm{t}$. Let the structure set $\mathcal{S}_\mathrm{t}$ be the set of binary trees labelled with 'atomic symbols' in the set $\mathcal{A} \equiv \{\mathsf{æ}, \mathsf{b}, \mathsf{k}\}$, e.g. $s_\mathrm{cab} \equiv [\mathsf{k}\ [\mathsf{æ}\ \mathsf{b}]] \in \mathcal{S}_\mathrm{t}$. Define the role set $\mathcal{R}_\mathrm{t} \equiv \{r_x \mid x \in \{0,1\}^*\}$; we think of, e.g., r_{01} as the binary-tree position 'left child [0] of the right child [1] of the root', and analogously for any string x of 0s and 1s. In s_cab, the position r_{01} is labelled with the symbol æ. Let the filler set be $\mathcal{F}_\mathrm{t} \equiv \mathcal{A}$. Then, $\mathsf{æ}/r_{01}$ is a filler-role binding of s_cab; the bindings function β_t pairs roles with their labels: $\beta_\mathrm{t}(s_\mathrm{cab}) = \{\mathsf{æ}/r_{01}, \mathsf{k}/r_0, \mathsf{b}/r_{11}\}$. The root position is r_ε, where ε is the empty string; for any atomic symbol $\mathsf{X} \in \mathcal{A}$, $\beta_\mathrm{t}(\mathsf{X}) = \{\mathsf{X}/r_\varepsilon\}$.

Example 4.3. Continuing the example of $\mathcal{S}_t$, binary trees labelled with $\mathcal{A}$, we can also define the *recursive* filler-role decomposition $\mathfrak{T}_r = (\mathcal{S}_t, \mathcal{F}_r, \mathcal{R}_r, \beta_r)$. The roles are simply $\mathcal{R}_r \equiv \{r_\varepsilon, r_0, r_1\}$, whereas the fillers are $\mathcal{F}_r \equiv \mathcal{S}_t$. Now, for s_{cab} above, $\beta_r(s_{cab}) = \{\mathsf{k}/r_0, [\mathsf{æ\ b}]/r_1\}$. Here, the filler of $r_1, f_1 \equiv [\mathsf{æ\ b}]$, is a non-atomic element of $\mathcal{F}_r = \mathcal{S}_t$, and itself has bindings $\beta_r(f_1) = \{\mathsf{æ}/r_0, \mathsf{b}/r_1\}$; the atomic filler k has binding set $\beta_r(\mathsf{k}) = \{\mathsf{k}/r_\varepsilon\}$, as in β_t. As in Lisp, we denote by cons the binary-tree constructor function: $s_{cab} = \mathsf{cons}(\mathsf{k}, [\mathsf{æ\ b}]) = \mathsf{cons}(\mathsf{k}, \mathsf{cons}(\mathsf{æ}, \mathsf{b}))$; thus $\beta_r(\mathsf{cons}(s_0, s_1)) = \{s_0/r_0, s_1/r_1\}$ for all $s_0, s_1 \in \mathcal{S}_t$. The access function that extracts the left (right) child of a tree will be denoted ex_0 (ex_1): $\mathcal{S}_t \to \mathcal{F}_r \cup \{\emptyset\}$ (both return a special 'null filler' $\emptyset$ when applied to an atom); thus, for non-atomic $s \in \mathcal{S}_t$, $\beta_r(s) = \{\mathsf{ex}_0(s)/r_0, \mathsf{ex}_1(s)/r_1\}$. $\mathsf{ex}_\varepsilon(s)$ is the symbol bound to the root of the tree s (or $\emptyset$ if there is no such symbol).

Henceforth, unless stated otherwise, we assume that $\mathcal{S}$ is a given set of binary trees under the canonical filler-role decomposition $\mathfrak{T}_t$. We also assume throughout that, in general, the sets $\mathcal{F}$ and $\mathcal{R}$ are denumerable (not necessarily finite); henceforth, let $\{\hat{f}_j\}$ and $\{\hat{r}_k\}$ be given enumerations of them.

(*b*) *Symbol structures as vectors*

The vectorial level of description springs from the following fundamental definition [15]: it asserts that the vector realizing (or modelling, or instantiating, or embedding) a symbol structure is the sum of vectors that realize each constituent filler-role binding of the structure, and that the vector realizing a filler-role binding is the tensor (generalized outer) product of vectors that realize the filler and the role.

Definition 4.4. A *tensor-product realization* $(\mathfrak{S}, F, R, \psi_F, \psi_R)$ consists of a filler-role decomposition $\mathfrak{S} = (\mathcal{S}, \mathcal{F}, \mathcal{R}, \beta)$, two real vector spaces F and R—the 'filler vector space' and the 'role vector space', with dimensions $\dim(F)$ and $\dim(R)$, respectively—and two 'realization' functions $\psi_F: \mathcal{F} \to F$ and $\psi_R: \mathcal{R} \to R$. Here, we require that each of the ranges of ψ_F and ψ_R be linearly independent sets.

The associated *realization mapping* $\Psi: \mathcal{S} \to S \equiv F \otimes R$ is defined by

$$\Psi(s) \equiv \sum_{k=1}^{N} \mathbf{f}_k \otimes \mathbf{r}_k, \quad \text{where } \beta(s) = \{f_k/r_k\}_{k=1}^{N} \text{ and } \mathbf{f}_k \equiv \psi_F(f_k), \mathbf{r}_k \equiv \psi_R(r_k).$$

The vector space S, containing the range of Ψ, is the tensor product of spaces F and R: $S \equiv F \otimes R$. [Given respective bases $\{\hat{\mathbf{f}}_j\}$ and $\{\hat{\mathbf{r}}_k\}$ for F and R, $\{\hat{\mathbf{f}}_j \otimes \hat{\mathbf{r}}_k\}$ is a basis for S, and the mapping $(\mathbf{f}, \mathbf{r}) \mapsto \mathbf{f} \otimes \mathbf{r}$ from $F \times R$ to S is bilinear: linear in each of $\mathbf{f}$ and $\mathbf{r}$ independently; $\dim(S) = \dim(F)\dim(R)$.]

Corresponding to our given enumerations $\{\hat{f}_j\}$ and $\{\hat{r}_k\}$ of $\mathcal{F}$ and $\mathcal{R}$, we have their vectorial realizations $\{\hat{\mathbf{f}}^j \equiv \psi_F(\hat{f}_j)\} \subset F$ and $\{\hat{\mathbf{r}}_k \equiv \psi_R(\hat{r}_k)\} \subset R$.

Given a vector $\mathbf{v} \in S$ realizing some symbol structure in $\mathcal{S}$, we can determine that structure from $\mathbf{v}$.

Proposition 4.5. *The linear independence of the ranges of ψ_F and ψ_R entails that ψ_F and ψ_R are invertible. Furthermore, Ψ is invertible: given $\mathbf{v} = \Psi(s)$, s can be recovered as the unique element of $\mathcal{S}$ with bindings $\{f_k/\hat{r}_k\}$, where $f_k \equiv {\psi_F}^{-1}(\mathbf{f}_k)$, and $\{\mathbf{f}_k\}$ is the unique sequence in F such that $\mathbf{v} = \sum_k \mathbf{f}_k \otimes \hat{\mathbf{r}}_k$* [15].

Within the continuous space of vectorial mental representations S, distance can be used to model cognitive dissimilarity, once S is endowed with a metric structure.

Definition 4.6. A *metric vectorial realization* is a tensor-product realization in which each vector space $V \in \{F, R\}$ has an *inner product*; the inner product of two vectors $\mathbf{u}$, $\mathbf{v} \in V$ is written as $\mathbf{u} \cdot \mathbf{v}$ [$(\mathbf{u}, \mathbf{v}) \mapsto \mathbf{u} \cdot \mathbf{v}$ is bilinear, symmetric and positive-definite: $\mathbf{u} \neq \mathbf{0} \Rightarrow \mathbf{u} \cdot \mathbf{u} > 0$]. The *length* of $\mathbf{u}$ is $\|\mathbf{u}\| \equiv (\mathbf{u} \cdot \mathbf{u})^{1/2}$; $\mathbf{u}$ is *normalized* iff $\|\mathbf{u}\| = 1$. The *distance* between $\mathbf{u}$ and $\mathbf{v}$ is $\|\mathbf{u} - \mathbf{v}\|$. $\mathbf{u}$ and $\mathbf{v}$ are *orthogonal* iff $\mathbf{u} \cdot \mathbf{v} = 0$. Inner products on F, R induce an inner product on $F \otimes R$ satisfying $(\mathbf{f}_1 \otimes \mathbf{r}_1) \cdot (\mathbf{f}_2 \otimes \mathbf{r}_2) = (\mathbf{f}_1 \cdot \mathbf{f}_2)(\mathbf{r}_1 \cdot \mathbf{r}_2)$.

Returning to the case of binary trees, we can now relate the two decompositions in example 4.2 and example 4.3:

Definition 4.7. A *recursive realization of binary trees* is a metric vectorial realization of $\mathfrak{T}_\mathrm{t}$, built from a vector space $R_{(1)}$, in which:

a. $\psi_R(r_0) \equiv \mathbf{r}_0 \in R_{(1)}$, $\psi_R(r_1) \equiv \mathbf{r}_1 \in R_{(1)}$;
b. $\mathbf{r}_{x0} = \mathbf{r}_x \otimes \mathbf{r}_0$ and $\mathbf{r}_{x1} = \mathbf{r}_x \otimes \mathbf{r}_1$ for all $x \in \{0,1\}^*$ (where $x0$ is the concatenation of string x and 0);
c. the roles for tree positions at depth d lie in the vector space $R_{(d)} \equiv R_{(1)} \otimes R_{(1)} \otimes \cdots \otimes R_{(1)}$ (d factors); letting $R_{(0)} \equiv \mathbb{R}$, the total role space R is the direct sum of all the vector spaces $R_{(d)}$:
d. $R \equiv R_{(0)} \oplus R_{(1)} \oplus R_{(2)} \oplus R_{(3)} \oplus \cdots$ [an $\mathbf{r} \in R$ can be represented as $\mathbf{r} = (\mathbf{r}_{(0)}, \mathbf{r}_{(1)}, \ldots)$, each $\mathbf{r}_{(\mathrm{d})} \in R_{(\mathrm{d})}$].[3]

Theorem 4.8. *There are four linear transformations on $S = F \otimes R$ (namely W_{cons0}, W_{cons1}, W_{ex0}, W_{ex1}) such that if $s = \mathsf{cons}(p, q)$, and we define $\mathbf{s} \equiv \Psi(s)$, $\mathbf{p} \equiv \Psi(p)$, $\mathbf{q} \equiv \Psi(q)$, then* [19]:

$$\mathbf{s} = \mathrm{W}_{\mathrm{cons0}}\mathbf{p} + \mathrm{W}_{\mathrm{cons1}}\mathbf{q}; \quad \mathbf{p} = \mathrm{W}_{\mathrm{ex0}}\,\mathbf{s}; \quad \mathbf{q} = \mathrm{W}_{\mathrm{ex1}}\mathbf{s}; \quad \mathbf{s} = \mathbf{p} \otimes \mathbf{r}_0 + \mathbf{q} \otimes \mathbf{r}_1.$$

This recursive realization of the canonical filler-role decomposition of definition 4.2 makes it possible to write $\mathbf{s}$ either in the form corresponding to the canonical decomposition, $\mathbf{s} = \sum_x \mathbf{f}_x \otimes \mathbf{r}_x$, where $\mathbf{f}_x$ is the realization of the atomic symbol at tree position x, or as $\mathbf{s} = \mathbf{p} \otimes \mathbf{r}_0 + \mathbf{q} \otimes \mathbf{r}_1$, just as it would be expressed using the recursive decomposition $\mathfrak{T}_\mathrm{r}$, $\mathbf{r}_0$ and $\mathbf{r}_1$ realizing its two roles (left/right child), and $\mathbf{p}$ ($\mathbf{q}$) realizing the entire left (right) sub-tree. In this sense, definition 4.7*b* renders the canonical representation recursive.

Later, the linear transformations of theorem 2.8 will enable computation of an interesting class of functions.

Henceforth, we assume given metric vectorial realizations Ψ^S and Ψ^I of $\mathcal{S}$ and $\mathcal{I}$ which are defined over the same filler and role vector spaces F and R, and which obey the following *input scaling condition*.

Condition 4.9. For all $f \in \mathcal{F}$, $\psi^I_F(f) = \psi^S_F(f)$; there exists a function ρ: $\mathcal{R} \to \mathbb{R}$ such that for all $r \in \mathcal{R}$, $\psi^I_R(r) = \rho(r)\psi^S_R(r)$.

[3]Physicists will recognize this construction of an infinite-dimensional Hilbert space as isomorphic to Fock space, where $R_{(d)}$ corresponds to the subspace of d particles, and $\mathbf{f} \otimes \mathbf{r}$ corresponds to the tensor product binding, for example, of the spin of an electron ($\mathbf{f}$) to its orbital ($\mathbf{r}$) in an atom [18].

Section 5*b* uses ρ to control the length of each input $\boldsymbol{\iota}$ in computing its distance to a state $\mathbf{s}$.

(*c*) *Symbolic structures as neural activation patterns*

Each vector space S^γ has a given neural basis; the coordinates of $\mathbf{s} = \Psi_S(s)$ w.r.t. this basis are the neural activation values realizing state $s \in \mathcal{S}$ (§3*c*). If the individual vectors $\{\hat{\mathbf{f}}_j\}$ and $\{\hat{\mathbf{r}}_k\}$ realizing the individual symbolic fillers and roles each lie along a neural basis vector, then the presence of a given symbolic constituent corresponds to the activation of a single neuron: this is 'local representation'. We will assume the general case, *distributed representation*, in which an individual constituent is realized by a vector that is a linear combination of multiple neural basis vectors: the presence of that constituent is encoded by an activation pattern distributed over multiple neurons, and each neuron participates in the encoding of multiple constituent s. (Distributed representations: allow many more representations per n neurons; allow encoding of similarity among constituents; are what results from neural network learning; and are ubiquitous in the brain [20–22].) The $[\varphi \times \varrho]$th activation value in the tensor-product realization of a symbol structure is the product [φth activation in the realization of constituent k's filler] $\times$ [ϱth activation in the realization of constituent k's role], summed over all the constituents k.

Definition 4.10. A 'neural basis' $\{\underline{\mathbf{f}}_\varphi\}$, $\{\underline{\mathbf{r}}_\varrho\}$ for the spaces F, R of a metric vectorial realization is a distinguished orthonormal basis. The neural coordinates, or activation pattern, realizing a symbol structure $s \in \mathcal{S}$ is the point $\mathbf{s} \in \mathbb{R}^{nm}$ with elements $\mathrm{s}_{\varphi\varrho}$ such that $\mathbf{s} = \sum_{\varphi\varrho} \mathrm{s}_{\varphi\varrho} \underline{\mathbf{f}}_\varphi \otimes \underline{\mathbf{r}}_\varrho$.

Proposition 4.11. *For each binding of s, $\beta(s) = \{f_k/r_k\}$, let $[\mathbf{f}_k]_\varphi$ be the φth neural coordinate of $\mathbf{f}_k \equiv \psi_F(f_k)$ (i.e. $\mathbf{f}_k = \sum_\varphi [\mathbf{f}_k]_\varphi \underline{\mathbf{f}}_\varphi$) and similarly let $[\mathbf{r}_k]_\varrho$ be the ϱth neural coordinate of $\mathbf{r}_k \equiv \psi_R(r_k)$. Then $s_{\varphi\varrho} = \sum_k [\mathbf{f}_k]_\varphi [\mathbf{r}_k]_\varrho$* [15].

5. Dissimilarity as representational distance: relational faithfulness

The total input $\iota \in \mathcal{I}$ to a component $\mathcal{M}$ is the target towards which the state of $\mathcal{M}$ is driven under the combined influence of those components that send partial input to $\mathcal{M}$. As we see in §7, *all else equal*, the greater the distance $d(s, \iota)$ of a state $s \in \mathcal{S}$ to the total input ι—i.e. the more dissimilar or 'unfaithful' s is to ι—the less likely $\mathcal{M}$ is to be in state s. Experimental data provide evidence for uncovering the (component-specific) dissimilarity metric d in $\mathcal{S}$; cognitive scientists formulate hypotheses concerning d in terms of the format of representations in $\mathcal{S}$: the format determines the dimensions of unfaithfulness relevant for d. We propose to formalize the problematic notion of representational format by making explicit those relations between constituents that are understood to be defined in that format (figure 1 ⑦).

(*a*) *Representational format as relational faithfulness*

Definition 5.1. A *representational format* $\mathfrak{F}$ is a filler-role decomposition in which the roles are a set of *relations* $\mathcal{R} = \{\mathsf{R}_i\}$. Each R_i has an 'arity' n_i, and a collection of 'argument domains' $\mathsf{A}_{i1}, \ldots, \mathsf{A}_{in_i}$. Then for any particular symbol

structure $s \in \mathcal{S}$, $\mathsf{R}_i(s) \subset \mathsf{A}_{i1} \times \cdots \times \mathsf{A}_{in_i}$ is the set of elements which in s stand in the relation R. The filler/role bindings are defined as $\beta(s) = \bigcup_i \{(\mathrm{a}_1, \ldots, \mathrm{a}_{n_i})/\mathsf{R}_i \mid (\mathrm{a}_1, \ldots, \mathrm{a}_{n_i}) \in \mathsf{R}_i(s)\}$.

Example 5.2. Let $\mathcal{S}_s \subset \mathcal{A}^*$ be the set of strings of symbols from the alphabet $\mathcal{A}$ such that no symbol appears more than once in any string. For any $s \in \mathcal{S}_s$, define the arity-1 relation $\mathsf{R}_\mathsf{A}(s) \equiv \{\mathsf{X} \in s\} \subset \mathcal{A} \equiv \mathsf{A}_{\mathsf{A}1}$, i.e. $\mathsf{R}_\mathsf{A}(s)$ is the set of symbols that occur (once) in the string s. Define the arity-2 relation $\mathsf{R}_\mathsf{L}(s) \equiv \{(l, \mathsf{X}) \mid \mathsf{X}$ is the symbol of s in position l (relative to the left edge)$\} \subset \mathbb{N} \times \mathcal{A} \equiv \mathsf{A}_{\mathsf{L}1} \times \mathsf{A}_{\mathsf{L}2}$; e.g. $\mathsf{R}_\mathsf{L}(\mathsf{CAB}) = \{(2, \mathsf{A}), (1, \mathsf{C}), (3, \mathsf{B})\}$. $\mathcal{R}_\mathrm{s} \equiv \{\mathsf{R}_\mathsf{A}, \mathsf{R}_\mathsf{L}\}$ is a representational format for $\mathcal{S}_s$, yielding a bindings function β_s; e.g. $\beta_s(\mathsf{BA}) = \{\mathsf{A}/\mathsf{R}_\mathsf{A}, \mathsf{B}/\mathsf{R}_\mathsf{A}, (1, \mathsf{B})/\mathsf{R}_\mathsf{L}, (2, \mathsf{A})/\mathsf{R}_\mathsf{L}\}$.

The distance $d(\iota, s)$ is measured in terms of the degree of unfaithfulness of s to ι w.r.t. $\mathcal{R}$ (figure 1 ⑧).

Definition 5.3. Given a representational format $\mathfrak{F}$ including some relation R, the *relational faithfulness constraints* for R are the following functions $\mathcal{I} \times \mathcal{S} \to \mathbb{N}$ ($|X|$ denotes the number of members of set X):

a. $\mathbb{C}^{\mathrm{I}}_\mathsf{R}(\iota, s) \equiv |\mathsf{R}(\iota) \backslash \mathsf{R}(s)| \equiv |\{a \in \mathsf{R}(\iota) \text{ s.t. } a \notin \mathsf{R}(s)\}|$,
b. $\mathbb{C}^{\mathrm{O}}_\mathsf{R}(\iota, s) \equiv |\mathsf{R}(s) \backslash \mathsf{R}(\iota)| \equiv |\{a \in \mathsf{R}(s) \text{ s.t. } a \notin \mathsf{R}(\iota)\}|$.

$\mathbb{C}^{\mathrm{I}}_\mathsf{R}(\iota, s)$ is the number of elements for which R is true of the input $\iota \in \mathcal{I}$ but not of the output $s \in \mathcal{S}$; $\mathbb{C}^{\mathrm{O}}_\mathsf{R}(\iota, s)$ is the reverse. So for the string example $\mathcal{S}_s$ earlier mentioned, for $\mathsf{R} = \mathsf{R}_\mathsf{A}$, $\mathbb{C}^{\mathrm{I}}_\mathsf{R}(\iota, s)$ is the number of symbols that have been deleted, and $\mathbb{C}^{\mathrm{O}}_\mathsf{R}(\iota, s)$ is the number of symbols that have been inserted (irrespective of position in the string, which R_A ignores). So, for example, $\mathbb{C}^{\mathrm{I}}_{\mathsf{R}_\mathsf{A}}(\mathsf{CAB}, \mathsf{AB}) = 1$. For $\mathsf{R} = \mathsf{R}_\mathsf{L}$, $\mathbb{C}^{\mathrm{I}}_\mathsf{R}(\iota, s)$ is the number of symbols in the input that are not in the same position (relative to the left edge) in the output; e.g. $\mathbb{C}^{\mathrm{I}}_{\mathsf{R}_\mathsf{L}}(\mathsf{CAB}, \mathsf{AB}) = 3$, although $\mathbb{C}^{\mathrm{I}}_{\mathsf{R}_\mathsf{L}}(\mathsf{CAB}, \mathsf{CA}) = 1$.

It is relations such as these that are implicit in arguments (e.g. [23]) that the mental representation for letter strings uses a format in which position is reckoned from the right as well as the left edge (i.e. that, in addition to R_L, the mental format involves an analogous relation R_R in which positions are counted from the right edge). Under neurological damage, a patient spelling a list of words may erroneously intrude letters from previous words into a given word, more likely errors being those more faithful to the previous-word target. Because these errors tend to preserve the position of letters relative to right as well as left word edges, the relevant faithfulness constraints must penalize discrepancies w.r.t. R_R as well as R_L.

An overall measure of the discrepancy between an input (target) ι and an output s—which plays the role here of a distance metric $d(\iota, s)$—is given by a weighted sum of the relational faithfulness constraints for R (definition 5.3). Faithfulness is one facet of well-formedness assessed by a measure called 'Harmony'.

Definition 5.4. A pair of positive *weights* $(w^{\mathrm{I}}_\mathsf{R}, w^{\mathrm{O}}_\mathsf{R})$ defines an R-*faithfulness Harmony function* H_R:

$$H_\mathsf{R}(\iota, s) = -w^{\mathrm{I}}_\mathsf{R}\mathbb{C}^{\mathrm{I}}_\mathsf{R}(\iota, s) - w^{\mathrm{O}}_\mathsf{R}\mathbb{C}^{\mathrm{O}}_\mathsf{R}(\iota, s).$$

The total relational-faithfulness Harmony of format $\mathfrak{F}$ with relation set $\mathcal{R}$ is $H_{\mathcal{R}}(\iota, s) \equiv \sum_{\mathsf{R}\in\mathcal{R}} H_{\mathsf{R}}(\iota, s)$.

The faithfulness Harmony $H_{\mathcal{R}}$ is increasingly negative as ι and s become more disparate along the dimensions encoded in the relations $\{\mathsf{R}_i\} = \mathcal{R}$; the maximum faithfulness Harmony value is 0, attained when $\forall \mathsf{R} \in \mathcal{R}$, $\mathsf{R}(s) = \mathsf{R}(\iota)$ (e.g. when $s = \iota$).

(*b*) *Relational unfaithfulness as vector distance*

Given a representational format $\mathfrak{F}$ with relation set $\mathcal{R}$, the dissimilarity or relational-faithfulness Harmony between an input ι and a state s, $H_{\mathcal{R}}(\iota, s)$, is directly related to the distance, in the metric vector space S, between their vectorial realizations $\boldsymbol{\iota}$ and $\mathbf{s}$: $H_{\mathcal{R}}$ is the negative of the distance squared (see ⑨). This result assumes the following type of tensor-product realization of the representations in $\mathcal{S}$.

The tensor-product realization of a representational format $\mathfrak{F}$ including an arity-n relation R involves a mapping ψ_F from a filler set $\mathcal{F} = \mathsf{A}_1 \times \cdots \times \mathsf{A}_n$ to a vector space F. Treating such a filler—an element such as $(\mathsf{a}, \mathsf{b}, \mathsf{c}) \in \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3$—as a structure in its own right, and then recursively applying tensor-product realization to $\mathcal{F}$ (using contextual roles [15]) leads to the following type of mapping ψ_F: $(\mathsf{a}, \mathsf{b}, \mathsf{c}) \mapsto \mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}$.

Definition 5.5. Given a representational format $\mathfrak{F}$ (assuming all the notation of definition 5.1), a role realization mapping ψ_R: $\{\mathsf{R}_i\} \to R$, and a set of filler realization mappings $\psi_{F_{im}}: \mathcal{A}_{im} \to F_{im}$, the *induced* tensor-product realization is defined by

$$\Psi(s) \equiv \sum_i \sum \{\psi_{F_{i1}}(\mathrm{a}_1) \otimes \cdots \otimes \psi_{F_{in_i}}(\mathrm{a}_{n_i}) \otimes \psi_R(\mathsf{R}_i) \mid (\mathrm{a}_1, \ldots, \mathrm{a}_{n_i}) \in \mathsf{R}_i(s)\}.$$

Definition 5.6. A tensor-product realization is called *orthogonal* iff the ranges of ψ_F and ψ_R are each orthogonal sets: $\forall f, f' \in \mathcal{F}, f \neq f' \Rightarrow \psi_F(f) \cdot \psi_F(f') = 0$, and similarly for ψ_R. The realization is *orthonormal* iff, in addition, $\forall f \in \mathcal{F}$, $\forall r \in R$, $\|\psi_F(f)\| = \|\psi_R(r)\| = 1$.

Note that orthogonal realizations are in general distributed: they need not be local (§4*c*). The result is as follows.

Theorem 5.7. *Let Ψ^S be a tensor-product realization of a representational format $\mathfrak{F}$ induced by orthonormal filler and role realizations ψ_F^S, ψ_R^S. Assume given faithfulness weights $w_{\mathsf{R}}^{\mathrm{I}}$, $w_{\mathsf{R}}^{\mathrm{O}}$ for each $\mathsf{R} \in \mathcal{R}$; these define the faithfulness Harmony function H_{R} (definition 5.4). For each R, replace the unit-length role vector $\psi_R^S(\mathsf{R}) \equiv \hat{\mathbf{r}}_{\mathsf{R}}^S$ by the rescaled vector $\mathbf{r}_{\mathsf{R}} \equiv (2w_{\mathsf{R}}^{\mathrm{O}})^{1/2}\hat{\mathbf{r}}_{\mathsf{R}}^S$. For the scaling of Ψ^I (condition 4.9), $\psi_R^I(\mathsf{R}) = \rho(\mathsf{R})\psi_R^S(\mathsf{R})$ define*

$$\rho(\mathsf{R}) \equiv \tfrac{1}{2}(w_{\mathsf{R}}^{\mathrm{O}} + w_{\mathsf{R}}^{\mathrm{I}})/w_{\mathsf{R}}^{\mathrm{O}}.$$

Given any $\iota \in \mathcal{I}$, $s \in \mathcal{S}$, let $\boldsymbol{\iota} \equiv \Psi^I(\iota)$ and $\mathbf{s} \equiv \Psi^S(s)$. Then, up to a constant term depending on ι, the total relational-faithfulness Harmony of s to ι decreases as the

square of the distance between $\mathbf{s}$ *and* $\boldsymbol{\iota}$*:*

$$H_R(\iota, s) = -\tfrac{1}{2}\|\boldsymbol{\iota} - \mathbf{s}\|^2 + \kappa(\iota),$$

where $\kappa(\iota) \equiv \frac{1}{4}\sum_{\mathsf{R}\in\mathcal{R}} \kappa_{\mathsf{R}}(\iota)$*;* $\kappa_{\mathsf{R}}(\iota) \equiv n^{\mathrm{I}}_{\mathsf{R}}(w^{\mathrm{I}}_{\mathsf{R}} - w^{\mathrm{O}}_{\mathsf{R}})^2/w^{\mathrm{O}}_{\mathsf{R}}$*; and* $n^{\mathrm{I}}_{\mathsf{R}} \equiv |\mathsf{R}(\iota)|$.

Note that the complexities arise only when there is an asymmetry $w^{\mathrm{I}}_{\mathsf{R}} \neq w^{\mathrm{O}}_{\mathsf{R}}$; otherwise, $\rho(\mathsf{R}) = 1$, $\kappa_{\mathsf{R}}(\iota) = 0$. (For a proof, see the electronic supplementary material.)

(*c*) *Relational faithfulness as network weights*

We will see in theorem 7.11 how relational faithfulness Harmony $H_{\mathcal{R}}$ is naturally realized at the network level.

6. Recursive functions as linear transformations

Having considered the representational states of the components of the cognitive architecture $\mathfrak{A}$, we turn to the functions computed over these representations.

(*a*) *Primitives'-closure functions*

Mental processes compute recursive functions; for concreteness (without compromising generality), we take the inputs and outputs of these functions to be binary trees. We now see that an interesting class of such recursive functions, denoted here $\mathcal{P}$, can be computed at the neural level in an extremely simple architecture: the linear associator (example 3.3). $\mathcal{P}$ is the closure under composition of the primitive tree-manipulating functions defined in §4*a* (figure 2 gives an example). It is convenient to work directly with the filler-role bindings of binary trees, assuming the canonical filler/role decomposition $\mathfrak{T}_{\mathrm{t}}$ (example 4.2).

Definition 6.1. A *pseudo-tree* t is a set of binary-tree filler/role bindings with at most one filler bound to each role; $\emptyset$ is the null pseudo-tree with no bindings; $\mathcal{T}$ is the set of pseudo-trees. The *unification* $\mathrm{t} \sqcup \mathrm{t}'$ of two pseudo-trees is the union of their bindings, provided this yields a pseudo-tree (at most one filler bound to each role); otherwise, $\mathrm{t} \sqcup \mathrm{t}' \equiv \emptyset$. Any $\mathrm{t} \in \mathcal{T}$ can be represented as $\bigsqcup_k \{\mathsf{A}_k/r_k\}$ for some sequence $\{\mathsf{A}_k\} \subset \mathcal{A}$, the alphabet of filler symbols, and $\{r_k\} \in \mathcal{R}$.

A function $g: \mathcal{T} \to \mathcal{T}$ is *first order* iff $g(\emptyset) = \emptyset$ and $g(\bigsqcup_k \{\mathsf{A}_k/r_k\}) = \bigsqcup_k \{g(\mathsf{A}_k/r_k)\}$.

First-order functions process each binding separately, with no interactions among bindings.

Proposition 6.2. *The primitive binary-tree functions* ex_0, ex_1, ex_ε, cons_0, cons_1 (*example 4.3*) *are first order* [19, p. 320].

As for manipulation of fillers, the basis of the set of recursive functions we consider are those in $\mathcal{B}$.

Definition 6.3. $\mathcal{B}$ is the set of functions h: $\mathcal{T} \to \mathcal{T}$ satisfying the following conditions:

a. h is first order;
b. for all $\mathrm{t} \in \mathcal{T}$, $h(\mathrm{t}) = \mathsf{cons}(h(\mathsf{ex}_0(\mathrm{t})), h(\mathsf{ex}_1(\mathrm{t}))) \sqcup h(\mathsf{ex}_\varepsilon(\mathrm{t}))/r_\varepsilon$;

c. there is a partial function $g_{\mathcal{A}}: \mathcal{A} \to \mathcal{A}$ such that $\forall \mathsf{X} \in \mathcal{A}$, $h(\mathsf{X}/r_\varepsilon) = g_{\mathcal{A}}(\mathsf{X})/r_\varepsilon$; if X is not in the domain of the partial function $g_{\mathcal{A}}$, then $h(\mathsf{X}/r_\varepsilon) = \emptyset$.

A function $h \in \mathcal{B}$ is indeed very simple: it simply replaces every filler X in the domain of $g_{\mathcal{A}}$ by the filler $g_{\mathcal{A}}(\mathsf{X})$, and deletes all other fillers. Finally, we join the filler-manipulating functions in $\mathcal{B}$ with role manipulation in the form of the closure, under composition, of the primitive binary-tree functions.

Definition 6.4. The class of *PC functions* (primitives'-closure) $\mathcal{P}$ has the following recursive definition:

a. base case: $h \in \mathcal{B} \Rightarrow h \in \mathcal{P}$
b. recursion
   (i) if $h \in \mathcal{B}$ and $g \in \mathcal{P}$, then $h \circ g \equiv h(g) \in \mathcal{P}$
   (ii) if $g \in \mathcal{P}$, then $\mathsf{ex}_0 \circ g \in \mathcal{P}$ and $\mathsf{ex}_1 \circ g \in \mathcal{P}$
   (iii) if g, $g' \in \mathcal{P}$, then $\mathrm{t} \to \mathsf{cons}(g(\mathrm{t}), g'(\mathrm{t})) \in \mathcal{P}$
c. no other functions are in $\mathcal{P}$.

While further development of the theory to address a larger class of recursive functions is in progress, the class $\mathcal{P}$ already includes many of the functions hypothesized to be computed in cognition, for example, recursive functions like that which maps a syntactic representation such as [[`Few` [`world leaders`]] [[`are admired`] [`by` [`George Bush`]]]] to a semantic representation like `admire`(`George Bush`, `few world leaders`): see the caption of figure 2 for the definition of this function g.

(*b*) *Primitives'-closure functions as linear transformations*

Because the primitive binary-tree functions can be realized as linear transformations (theorem 4.8), and because the set of linear transformations is closed under composition and addition, we get straightforwardly the following result.

Theorem 6.5. *Any PC function $g \in \mathcal{P}$ is realized by a linear transformation W_g on S; that is,*

$$\forall \mathrm{t}, \mathrm{t}' \in \mathcal{T}, \quad g: \mathrm{t} \mapsto \mathrm{t}' \text{ iff } \mathrm{W}_g \Psi(\mathrm{t}) = \Psi(\mathrm{t}').$$

(*c*) *Primitives'-closure functions as neural networks*

It follows immediately that a PC function can be computed by a simple type of network, a linear associator (example 3.3); the weight matrix turns out to have a form that enables finite specification of an infinite matrix.

Theorem 6.6. *For any $g \in \mathcal{P}$, the linear transformation W_g is realized by a linear associator with weight matrix*

$$\mathbf{W}_g = \mathbf{I} \otimes \underline{\mathbf{W}}_g,$$

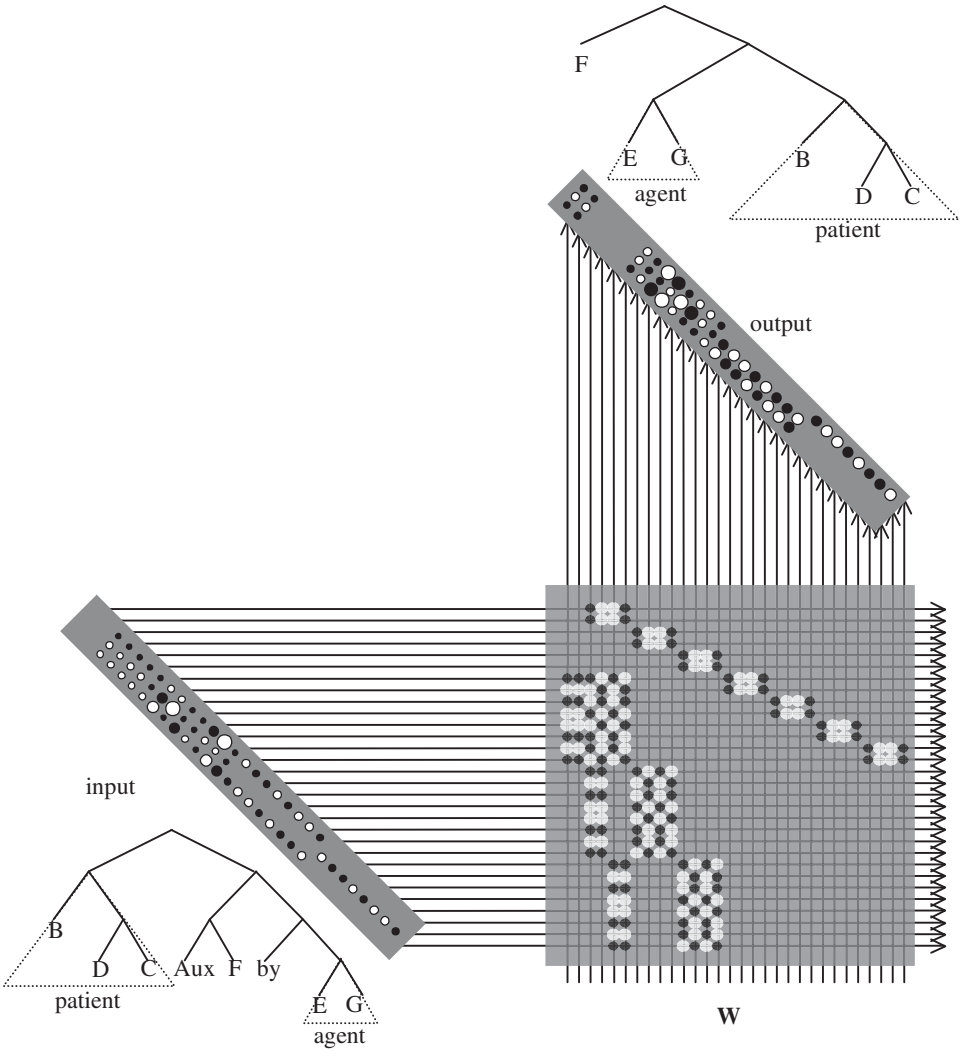

Figure 2. Applying the technique of theorem 6.5, this linear associator network computes the function

$$g(s) = \mathsf{cons}(\mathsf{ex}_1(\mathsf{ex}_0(\mathsf{ex}_1(s))), \mathsf{cons}(\mathsf{ex}_1(\mathsf{ex}_1(\mathsf{ex}_1(s))), \mathsf{ex}_0(s)))$$

with the corresponding weight matrix

$$\mathbf{W}_g = \mathbf{W}_{\mathrm{cons}0}[\mathbf{W}_{\mathrm{ex}1}\mathbf{W}_{\mathrm{ex}0}\mathbf{W}_{\mathrm{ex}1}] + \mathbf{W}_{\mathrm{cons}1}[\mathbf{W}_{\mathrm{cons}0}(\mathbf{W}_{\mathrm{ex}1}\mathbf{W}_{\mathrm{ex}1}\mathbf{W}_{\mathrm{ex}1}) + \mathbf{W}_{\mathrm{cons}1}(\mathbf{W}_{\mathrm{ex}0})].$$

Disc area displays magnitude of activations and weights; black/white denote positive/negative. The unbounded competence of the network results from the unbounded weight matrix $\mathbf{W}_g$, which is finitely specified as the tensor product of a finite matrix $\underline{\mathbf{W}}_g$ and the unbounded identity matrix $\mathbf{I}$ (§6c). (Reprinted with permission from [9].)

where $\mathbf{I}$ *is the identity matrix on* (*infinite-dimensional*) R (*definition 4.7*) *and* $\underline{\mathbf{W}}_g$ *is a finite matrix.*

The map $g \mapsto \underline{\mathbf{W}}_g$ is compositional: it can be defined constructively [19, p. 324] (see figure 2 for an example).

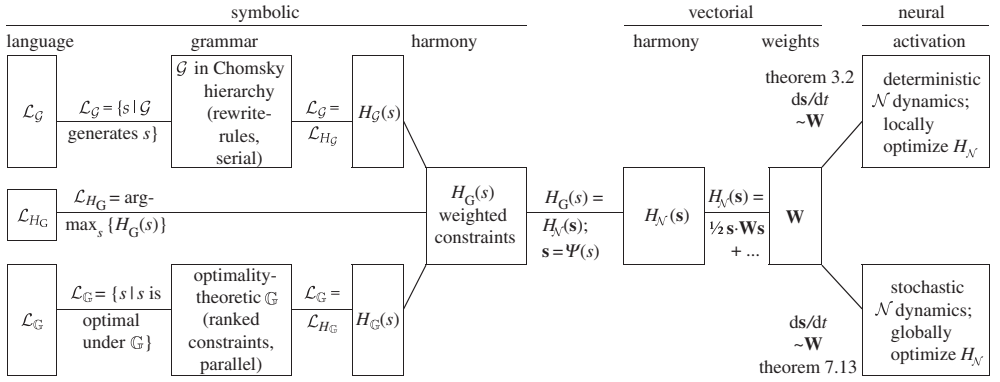

Figure 3. A roadmap showing the chain of inference in §7.

7. Grammar as optimality

Having seen that a significant class of recursive functions can be computed in one massively parallel step (fully specified at the neural level in terms of multiplication and addition operations), in order to analyse linguistic cognition, we turn from recursive function theory to another general approach for specifying functions over symbols: formal language theory. This approach conceives of the function to be computed as a kind of language, and deploys rewrite-rule grammars, interpreted sequentially. These grammars can be classified in various ways, e.g. the Chomsky hierarchy, with its corresponding hierarchy of formal sequential automata, culminating in Turing machines. It turns out that a different approach to specifying languages, more directly reducible to neural computation, deploys Harmony. In §5, we encountered one facet of Harmony, faithfulness; specifying languages introduces the other facet, markedness. Markedness reduces the Harmony of symbol structures that violate grammatical constraints, and may reward structures that meet grammatical desiderata. The structures of the formal language are those with maximal Harmony.

Because of the extended chain of inference in this section, a roadmap is provided in figure 3.

To see the basic intuition, consider a context-free rewrite-rule grammar $\mathcal{G}$ in Chomsky normal form; a legal derivation D produced by such a grammar can be represented as a binary tree t_D. For example, use of the rule $\mathsf{A} \rightarrow \mathsf{B}\ \ \mathsf{C}$ in a derivation D contributes to t_D a local tree $[_{\mathsf{A}}\ \mathsf{B}\ \mathsf{C}]$: a pair of sister nodes labelled B and C immediately dominated by a mother node labelled A. We can take the language generated by $\mathcal{G}$, $\mathcal{L}_{\mathcal{G}}$, to be the set of all binary trees representing legal derivations. To evaluate whether a given tree $\mathrm{t} \in \mathcal{L}_{\mathcal{G}}$, we must check that every local tree $[_{\mathsf{X}}\ \mathsf{Y}\ \mathsf{Z}]$ in t is sanctioned by a rule $\mathsf{X} \rightarrow \mathsf{Y}\ \ \mathsf{Z}$ in $\mathcal{G}$. We can do this numerically, computing a Harmony value $H(\mathrm{t})$ for t, as follows. We assess a markedness Harmony penalty of -3 (lower $H(\mathrm{t})$ by 3) for every symbol in t. Then for every rule $\mathsf{X} \rightarrow \mathsf{Y}\ \mathsf{Z}$ in $\mathcal{G}$, we add $+2$ to $H(\mathrm{t})$ for every pair in t consisting of a mother node labelled X with a left (right) child node labelled Y (Z); we conceptually split this into a reward of $+1$ for each node in the pair. Now, consider a node x labelled by some symbol V: x will lower $H(\mathrm{t})$ by 3. Then if x is dominated by a legal parent node for V, x increases $H(\mathrm{t})$ by 1; if x has two legal daughter

nodes for V, x increases $H(\mathrm{t})$ by 1 for each. Thus if x is fully sanctioned by the grammar (both above and below), it will contribute 0 to $H(\mathrm{t})$; if it is not legal, its penalty -3 will not be fully offset and its net contribution to $H(\mathrm{t})$ will be negative. The result is that if t is legal, $H(\mathrm{t}) = 0$, otherwise, $H(\mathrm{t}) < 0$. The maximal Harmony trees are those with $H = 0$: exactly the trees of $\mathcal{L}_{\mathcal{G}}$.[4]

(*a*) *Rewrite-rule and Optimality-Theoretic grammars as Harmonic grammars*

Definition 7.1. A *Harmonic grammar* over $\mathcal{S}$ [24] is a function $H_{\mathrm{G}} : \mathcal{S} \to \mathbb{R}$; $H_{\mathrm{G}}(s)$ is the '(markedness) Harmony' of $s \in \mathcal{S}$. The *language specified by* H_{G}, $\mathcal{L}_{H_{\mathrm{G}}} \subset \mathcal{S}$, is

$$\mathcal{L}_{H_{\mathrm{G}}} \equiv \mathrm{argmax}_{s \in \mathcal{S}} H_{\mathrm{G}}(s) \equiv \{s \in \mathcal{S} \mid \nexists s' \in \mathcal{S} \text{ s.t. } H_{\mathrm{G}}(s') > H_{\mathrm{G}}(s)\}.$$

An $s \in \mathcal{L}_{H_{\mathrm{G}}}$ has maximum Harmony, and is said to be *optimal*.

A Harmonic grammar H_{G} is *second-order* iff there exists a function $H_{\mathrm{b}} : B \times B \to \mathbb{R}$ (where $B \equiv F \times R$) such that if $\beta(s) = \{f_k/r_k\} \equiv \{b_k\}$, then $H_{\mathrm{G}}(s) = \sum_j \sum_k H_{\mathrm{b}}(b_j, b_k)$ (H_{G} depends only on *pairs* of constituents).

Theorem 7.2. *Given a formal language $\mathcal{L}_{\mathcal{G}}$ generated by a grammar $\mathcal{G}$ in the Chomsky hierarchy, there is a second-order Harmonic grammar $H_{\mathcal{G}}$ that specifies the same language: $\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{H_{\mathcal{G}}}$.*

As sketched intuitively earlier, H_{G} can be constructed compositionally from the rewrite rules of $\mathcal{G}$ [25, p. 399]. This shows that formal languages can be *specified*; we consider later how languages can be *computed*.

We turn now to the grammars relevant to cognition, those of human natural languages. It turns out that for generative linguistics—the formal theory of human language—analysing the set of grammatical expressions as a set of optimal structures is quite fruitful, thanks primarily to Optimality Theory [26,27].

Definition 7.3. An *Optimality-Theoretic system* $\mathfrak{O} = (\mathcal{I}, \mathcal{S}, Gen, Con)$ consists of the following:

a. a set $\mathcal{I}$ of symbol structures called 'inputs' (e.g. for syntax, a meaning to be expressed);
b. a set $\mathcal{S}$ of symbol structures called 'outputs' (e.g. for syntax, a parse tree of a sentence);
c. a function Gen: $\mathcal{I} \to 2^{\mathcal{S}}$; $Gen(\iota)$ is the set of 'candidate outputs' for $\iota \in \mathcal{I}$ (e.g. all possible parses);
d. a finite set Con of functions $\mathbb{C}$: $\mathcal{I} \times \mathcal{S} \to \mathbb{N} \cup \{0\}$ called 'constraints'; $\mathbb{C}(\iota, s)$ is the number of 'violations of $\mathbb{C}$ by (ι, s)', and if $\mathbb{C}(\iota, s_1) < \mathbb{C}(\iota, s_2)$ we say $\mathbb{C}$ 'prefers s_1 to s_2 given ι'
   (i) Con includes *faithfulness* constraints; each such constraint $\mathbb{C}_{\mathrm{F}}$ evaluates a dimension of structure, being violated by each deviation of s from ι w.r.t. that dimension
   (ii) Con includes *markedness* constraints; each such constraint $\mathbb{C}_{\mathrm{M}}$ evaluates, independently of ι, the inherent well-formedness of s with respect to some structural dimension.

[4]This glosses over details of the full analysis, which requires special treatment of the tree root and the start symbol, as well as terminal nodes, and conversion of $\mathcal{G}$ to 'Harmonic normal form', which enables a second-order Harmony function (definition 7.1) [24].

An $\mathfrak{O}$-*grammar* $\mathbb{G}$ is a total order $\gg$ on *Con*: a 'constraint ranking'. An input–output pair (ι, s), $s \in Gen(\iota)$, is *optimal* w.r.t. the $\mathfrak{O}$-grammar $\mathbb{G} = \gg$ iff the following holds:

$\nexists o' \in Gen(\iota)$ s.t. (ι, o') is preferred to (ι, o) by the $\gg$-maximal constraint that prefers one of them

The language $\mathcal{L}_{\mathbb{G}}$ specified by an $\mathfrak{O}$-grammar $\mathbb{G}$ is the set of all $\mathbb{G}$-optimal input–output pairs.

The *typology* specified by $\mathfrak{O}$ is the set of all languages specified by some $\mathfrak{O}$-grammar.

The empirical hypothesis is that the space of possible human grammars is an $\mathfrak{O}$-typology for some $\mathfrak{O}$ called 'universal grammar'—all languages possess the same grammatical constraints or preferences: differences arise only in how conflicts among those preferences are resolved; i.e. differences are limited to how the grammar of each language ranks the universal constraints.

Example 7.4. The Italian counterpart of English *it rains* is *piove*. An Optimality-Theoretic analysis goes as follows [28]. Our meaning is $\iota_0 =$ [rain(tense = present)]; we assume here a faithfulness constraint $\mathbb{C}_{\mathrm{F}}$ that requires the verb form to be *piove* in Italian, or *rains* in English. This constraint outranks all others, ensuring that only an expression with the correct verb can be optimal. A markedness constraint $\mathbb{C}_{\mathrm{M1}}$ requires that every sentence have a subject; the Italian candidate output sentence *piove* (or English *rains*) violates $\mathbb{C}_{\mathrm{M1}}$; *esso piove* (or *it rains*) satisfies $\mathbb{C}_{\mathrm{M1}}$. Another markedness constraint $\mathbb{C}_{\mathrm{M2}}$ requires that every word in an expression contribute to the meaning: *esso piove* (or *it rains*) violates $\mathbb{C}_{\mathrm{M2}}$ (the subject is meaningless) while *piove* (or *rains*) satisfies $\mathbb{C}_{\mathrm{M2}}$. In the Italian grammar, $\mathbb{C}_{\mathrm{M2}} \gg \mathbb{C}_{\mathrm{M1}}$: avoiding meaningless words is more important than providing a subject; so $(\iota_0, piove)$ is optimal, hence grammatical. In English, the ranking is reversed, so $(\iota_0, it\ rains)$ is optimal.

Despite being violated in these optimal expressions, $\mathbb{C}_{\mathrm{M1}}$ is active in Italian and $\mathbb{C}_{\mathrm{M2}}$ is active in English: in English, $\mathbb{C}_{\mathrm{M2}}$ can only be violated when overruled by a higher-ranked constraint such as $\mathbb{C}_{\mathrm{M1}}$. For example $(\iota_0, it\ rains\ it)$ is not grammatical—it has lower Harmony than $(\iota_0, it\ rains)$—because of the second violation of $\mathbb{C}_{\mathrm{M2}}$ incurred by the meaningless object *it*: its presence (unlike the subject *it*) is not required to satisfy any higher-ranked constraint. The difference between English and Italian is not that the former prefers and the latter disprefers meaningless items: *all* languages (according to Optimality Theory) have the same grammatical preferences—differences arise only in how to resolve conflicts among those preferences, i.e. in the ranking of constraints.

Optimality Theory arguably provides the first general formal theory of linguistic typology. This has enabled formal analysis of typologies, as well as individual languages, at all linguistic levels: phonology, syntax, semantics and pragmatics [29–32] (see also the archive http://roa.rutgers.edu/). The empirical successes of Optimality Theory are cases when what is universally shared by human languages are *preferences* that outputs of the grammar should respect, as opposed to *processes* that generate those outputs. Rewrite rules characterize grammatical knowledge as procedures, while Optimality Theory characterizes grammatical knowledge as preferences: this is the force of the idea, 'grammars as optimization'.

Given the potential value of Optimality Theory for understanding linguistic cognition, we turn to how neural computation might be used to compute languages specified by Optimality-Theoretic grammars, rather than languages specified by serial rewrite rules or functions specified by recursive equations.

For reduction of an $\mathfrak{O}$-grammar $\mathbb{G}$ to the neural level, we translate $\mathbb{G}$ into a Harmonic grammar $H_{\mathbb{G}}$.

Theorem 7.5. *Let $\mathbb{G} =\gg$ be an $\mathfrak{O}$-grammar s.t. every constraint $\mathbb{C}_k \in Con$ is bounded above: $\exists M \in \mathbb{N}$ s.t. $\forall \mathbb{C}_k \in Con$, $\forall \iota \in \mathcal{I}$, $\forall s \in Gen(\iota)$, $\mathbb{C}_k(\iota, s) < M$. Then, there is a Harmonic grammar $H_{\mathbb{G}}$ over $\mathcal{I} \times \mathcal{S}$ that specifies the same language: $\mathcal{L}_{\mathbb{G}} = \mathcal{L}_{H_{\mathbb{G}}}$. $H_{\mathbb{G}} = -\sum_k w_k \mathbb{C}_k$ for a set of weights $\{w_k\}$ with $w_k > w_j \Leftrightarrow \mathbb{C}_k \gg \mathbb{C}_j$. (In this context, (ι, s^*) is optimal for an H_{G} iff $s^* \in \mathrm{argmax}_s\{H_{\mathrm{G}}(\iota, s) \mid s \in Gen(\iota)\}$.)* [26, §10.2.2; 33; 34, p. 463].

The basic idea of the construction is that if $Con \equiv \{\mathbb{C}_k\}_{k=0}^{N}$ is ordered by $\gg$ so that $\mathbb{C}_{k+1} \gg \mathbb{C}_k$, then the constraint weights can be $w_k \equiv M^k$ ($M-1$ being the largest possible number of violations of any constraint). The Harmony cost incurred by a single violation of $\mathbb{C}_k$ ($w_k = M^k$) exceeds the total maximal cost that can be incurred by violations of all lower-ranked constraints ($\sum_{j<k} w_j[M-1] = \sum_{j=0}^{k-1} M^{j+1} - \sum_{j=0}^{k-1} M^j = M^k - 1$). This means that $H_{\mathbb{G}}(\iota, o) > H_{\mathbb{G}}(\iota, o')$ if and only if the highest-ranking constraint that has a preference between (ι, o) and (ι, o') prefers the former: optimality under $\mathbb{G}$ and under $H_{\mathbb{G}}$ are equivalent.

(*b*) *Harmonic grammars as linear input transformations*

The motivation for the following definition is given shortly (theorem 7.12).

Definition 7.6. With respect to a linear intra-component input transformation $\mathrm{W}: S \to S$ on the vector space S realizing $\mathcal{S}$, the *markedness Harmony* of a vector $\mathbf{s} \in \mathcal{S}$ is

$$H_{\mathrm{M,W}}(\mathbf{s}) \equiv \tfrac{1}{2}\mathbf{s} \cdot \mathbf{W}\mathbf{s}.$$

A linear intra-component input transformation $\mathrm{W_G} : S \to S$ realizes a Harmonic grammar H_{G} iff

$$\forall s \in \mathcal{S}, H_{\mathrm{G}}(s) = H_{\mathrm{M,W_G}}(\mathbf{s}), \text{ where } \mathbf{s} \equiv \boldsymbol{\Psi}(s).$$

Theorem 7.7. *A second-order Harmonic grammar H_{G} can be realized by a linear input transformation $\mathrm{W_G}$.*

Because a formal language $\mathcal{L}_{\mathcal{G}}$ specified by a rewrite-rule grammar $\mathcal{G}$ can also be specified by a corresponding second-order Harmonic grammar $H_{\mathcal{G}}$ (theorem 7.2), we immediately get the following corollary.

Corollary 7.8. *Given a rewrite-rule grammar $\mathcal{G}$, there is a linear input transformation $\mathrm{W}_{\mathcal{G}}$ that realizes the Harmonic grammar counterpart of $\mathcal{G}$, $H_{\mathcal{G}}$* [19, p. 333].

Moving from rewrite-rule grammars to Optimality-Theoretic grammars realized in Harmonic grammars requires not only markedness Harmony but also faithfulness Harmony, introduced next.

(c) Harmony optimization as neural computation

Given a Harmonic grammar H_G realized as a linear intra-component input transformation W_G on S (definition 7.6), and a neural realization of S in a network $\mathcal{N}$, the elements of W_G w.r.t. the neural basis constitute the internal weight matrix of $\mathcal{N}$. The utility of this is that $\mathcal{N}$ can perform Harmony optimization, computing maximum Harmony representations (theorem 7.12): these should be the optimal states constituting the language $\mathcal{L}_{H_G}$. (In order to achieve this, two obstacles will need to be overcome.)

Definition 7.9. Given a quasi-linear network $\mathcal{N}$ (example 3.2, definition 3.4) with weight matrix $\mathbf{W}$, external input $\boldsymbol{\iota}$ and unit activation function f, the *network Harmony* $H_{\mathcal{N}}: I \times S \to \mathbb{R}$ is the sum of markedness Harmony $H_{M,\mathbf{W}}$ (definition 7.6) and *faithfulness Harmony* H_F:

$$H_{\mathcal{N}}(\boldsymbol{\iota}, \mathbf{s}) = H_{M,\mathbf{W}}(\mathbf{s}) + H_F(\boldsymbol{\iota}, \mathbf{s}); \quad H_{M,\mathbf{W}}(\mathbf{s}) = \tfrac{1}{2}\mathbf{s} \cdot \mathbf{W}\mathbf{s}, \quad H_F(\boldsymbol{\iota}, \mathbf{s}) \equiv \mathbf{s} \cdot \boldsymbol{\iota} + H_1(\mathbf{s}),$$

where the *unit Harmony* H_1 is

$$H_1(\mathbf{s}) \equiv \sum_{\mu} h([\mathbf{s}]_\mu), \quad h(a) \equiv -\int_0^a f^{-1}(x)\,\mathrm{d}x.$$

Example 7.10. Let $\mathcal{N}$ be linear (definition 3.5), i.e. have units with activation function $f(z) = z$. Then, the unit Harmony is

$$H_1(\mathbf{s}) = \sum_{\mu} h([\mathbf{s}]_\mu),$$

where

$$h(a) \equiv -\int_0^a f^{-1}(x)\,\mathrm{d}x = -\int_0^a x\,\mathrm{d}x = -\tfrac{1}{2}a^2$$

$$\Rightarrow H_1(\mathbf{s}) = \sum_{\mu}[-\tfrac{1}{2}([\mathbf{s}]_\mu)^2] = -\tfrac{1}{2}\|\mathbf{s}\|^2.$$

What is the relation between the neural-level faithfulness Harmony H_F of the network $\mathcal{N}$ and the symbolic-level relational faithfulness H_{R} that evaluates the dissimilarity of symbol structures (definition 5.4)?

Theorem 7.11. *Let Ψ^S be a tensor-product realization in a vector space S of a representational format $\mathfrak{F}$ satisfying the conditions of theorem 5.7. Suppose given, for each $\mathsf{R} \in \mathcal{R}$, a pair of weights $(w^{\mathrm{I}}_{\mathsf{R}}, w^{\mathrm{O}}_{\mathsf{R}})$ that define the R-faithfulness Harmony function H_{R}, and hence the total relational faithfulness $H_{\mathcal{R}}$ (definition 5.4). Let S be realized in a linear network $\mathcal{N}$, and let $\iota \in \mathcal{I}$, $s \in \mathcal{S}$; write $\boldsymbol{\iota} \equiv \Psi^I(\iota) \in I$, $\mathbf{s} \equiv \Psi^S(s) \in S$. Then:*

$$H_{\mathcal{R}}(\iota, \mathrm{s}) = H_F(\boldsymbol{\iota}, \mathbf{s}) - \kappa'(\iota),$$

where

$$\kappa'(\iota) \equiv \sum_{\mathsf{R}} w^{\mathrm{I}}_{\mathsf{R}} n^{\mathrm{I}}_{\mathsf{R}} \quad (n^{\mathrm{I}}_{\mathsf{R}} \equiv |\mathsf{R}(\iota)|).$$

(For a proof, see the electronic supplementary material.) Thus, the relational Harmony between the symbol structures (ι, s) can be computed as the network faithfulness Harmony between the vectors $(\boldsymbol{\iota}, \mathbf{s})$ realizing (ι, s), up to a constant

depending on ι; this constant has no effect on determining the optimal representation s for a given input ι. By theorem 7.7, the well-formedness of a symbolic state s as assessed by a Harmonic grammar H_G, $H_G(s)$, can be computed as the network markedness Harmony. The network Harmony $H_{\mathcal{N}}$ combines both these evaluations. The computational utility of $H_{\mathcal{N}}$ derives from theorem 7.12.

Theorem 7.12. *Given a network $\mathcal{N}$ with quasi-linear dynamics (example 3.2) and external input $\boldsymbol{\iota}$, suppose the weight matrix is symmetric $(\forall \mu, \nu, \mathrm{W}_{\mu\nu} = \mathrm{W}_{\nu\mu})$ and scaled so that network Harmony $H_{\mathcal{N}}$ is bounded above.*[5] *Then, as activation spreads, the Harmony of the network state $\mathbf{s}$, $H_{\mathcal{N}}(\mathbf{s})$, is non-decreasing, and $\mathbf{s}$ converges to a local maximum of $H_{\mathcal{N}}$ (a state $\mathbf{s}$ such that $H_{\mathcal{N}}(\mathbf{s}) \geq H_{\mathcal{N}}(\mathbf{s} + \boldsymbol{\varepsilon})$, for all small $\boldsymbol{\varepsilon}$ [35–37]).*

We are now close to showing that neural computation can compute the grammatical expressions of a symbolic rewrite-rule grammar $\mathcal{G}$ or an Optimality-Theoretic grammar $\mathbb{G}$: these expressions are the maxima of a symbolic second-order Harmonic grammar $H_G = H_{\mathcal{G}}$ (theorem 7.2) or $H_G = H_{\mathbb{G}}$ (theorem 7.5), and H_G can be realized in network weights $\mathbf{W}_G$ as the Harmony $H_{\mathcal{N}}$ of a network $\mathcal{N}$ (theorem 7.7), and spreading activation in $\mathcal{N}$ can compute the maxima of $H_{\mathcal{N}}$ (theorem 7.12). But two obstacles remain.

The first is that the network computes *local* Harmony maxima, but the Harmonic grammar demands *global* maxima (indeed, the Harmonic grammar recognizes no such notion as 'local maximum'). The quasi-linear dynamics drives the network state constantly uphill in Harmony, so the network ends up at the peak of whatever Harmony mountain it happens to start on: the peak is higher than all neighbouring points, but need not be the highest peak of the mountain range, which is what the Harmonic grammar requires us to find. To compute the global Harmony maximum, the network needs some probability of moving downhill, providing a chance to pass through valleys in order to arrive at the highest mountain. Global optimization requires some randomness in activation spreading [38–40].

Theorem 7.13. *The neural network $\mathcal{N}^T$ with dynamics $\mathcal{D}^T_{\mathrm{opt}}$ defined by the stochastic differential equation*[6]

$$\mathrm{d}\mathrm{s}_\mu = (\partial H_{\mathcal{N}}/\partial \mathrm{s}_\mu)\,\mathrm{d}t + (2T)^{1/2}\mathrm{d}B, \quad \begin{array}{l} B \text{ a Wiener process,} \\ H_{\mathcal{N}} : S \to \mathbb{R} \text{ a Harmony function} \end{array}$$

(or the corresponding stochastic difference equation

$$\Delta \mathrm{s}_\mu = (\partial H_{\mathcal{N}}/\partial \mathrm{s}_\mu)\Delta t + (2T\Delta t)^{1/2}\underline{B} \quad \begin{array}{l} \underline{B} \text{ a random variable with standard} \\ \text{normal distribution } N(0,1)) \end{array}$$

[5]In order that $H_{\mathcal{N}}$ have a maximum, as $\mathbf{s}$ gets large, it is necessary that the H_1 term get small faster than the $H_{M,\mathbf{W}}$ gets large (if it does). For the linear case, of interest here, H_1 goes to $-\infty$ quadratically; $H_{\mathrm{M},\mathbf{W}}$ might go to $+\infty$ quadratically. But if we scale $\mathbf{W}$ appropriately, by multiplying it by a sufficiently small constant, we can ensure that H_1 dominates, so that $H_{\mathcal{N}}$ has a maximum. Rescaling $\mathbf{W}$ does not affect the location of the maxima of $H_{\mathrm{M},\mathbf{W}}$. We can ensure that the vectors realizing all symbol structures have the same length, so adding H_1 does not affect the relative Harmonies of the symbolic states.

[6]The derivative $\partial H_{\mathcal{N}}/\partial \mathrm{s}_\mu$ is simply $[\mathbf{W}\mathbf{s} + \boldsymbol{\iota} - \boldsymbol{f}^{-1}(\mathbf{s})]_\mu$ which is just $[\mathbf{W}\mathbf{s} + \boldsymbol{\iota} - \mathbf{s}]_\mu$ in the linear case.

converges to the distribution $p_T(\mathbf{s}) \propto e^{H_{\mathcal{N}}(\mathbf{s})/T}$ [41,42]. *Thus, as* $T \to 0$, $p_T(\mathbf{s}) \to 0$ *except for globally optimal* $\mathbf{s}$.

In $\mathcal{N}^T$, the *computational temperature* T is a function of computational time t: the initial temperature $T(0)$ is high, and $T(t)$ decreases to 0 during the course of computation. In principle, if this is carried out sufficiently slowly, the network's probability of being in state $\mathbf{s}$ at time t will remain approximately proportional to $e^{H(\mathbf{s})/T(t)}$ [43]; then at time t, the probability of a non-globally optimal state $\mathbf{s}'$, with Harmony H', relative to the probability of the globally optimal state $\mathbf{s}^*$, with Harmony $H_{\max}$, is, as $t \to \infty$ hence $T(t) \to 0$,

$$p_t(\mathbf{s}')/p_t(\mathbf{s}^*) = e^{H'/T(t)}/e^{H_{\max}/T(t)} = e^{-[H_{\max}-H']/T(t)} \to 0,$$

because $H_{\max} - H' > 0$. Thus the probability of any non-globally optimal state s' goes to zero as $t \to \infty$.

8. Discreteness

The stochastic neural network $\mathcal{N}^T$ (theorem 7.13) computes, in the limit, a state $\mathbf{s} \in S$ with globally maximal Harmony. The Harmonic grammar $H_{\mathcal{G}}$ requires a *symbolically interpretable* state $\mathbf{s}$ with globally maximal Harmony: a state $\mathbf{s}$ realizing a symbol structure s that maximizes $H_{\mathcal{G}}(s)$ over the set of all symbol structures $\mathcal{S}$. Because H_{M} realizes $H_{\mathcal{G}}$ (definition 7.6), we know that $H_{\mathrm{M}}(\mathsf{s}) = H_{\mathcal{G}}(s)$ for every $\mathsf{s} = \Psi(s)$ that realizes a symbol structure $s \in \mathcal{S}$; these s comprise a discrete subset $\boldsymbol{S}$ of the continuous vector space S. Our second obstacle is that the global maximum of $H_{\mathcal{N}}$ in S is generally *not* in $\boldsymbol{S}$: conflicts between the constraints encoded in H entail that optima constitute compromises that interpolate between the alternative discrete states favoured by the conflicting constraints. Harmony optimization at the neural level does not yield realizations of symbolic states. To achieve that, in addition to the optimization dynamics $\mathcal{D}_{\mathrm{opt}}$, another dynamics is required: *quantization*.

Theorem 8.1. *There is a deterministic dynamics* $\mathcal{D}_{\mathrm{quant}}$ *on* S, $\mathrm{d}\mathbf{s}/\mathrm{d}t = \boldsymbol{Q}(\mathbf{s}(t))$, *which has an attractor at every vector* $\mathsf{s} \in \boldsymbol{S}$, *i.e. at every vector* $\mathsf{s} = \Psi(s)$ *that realizes a symbol structure* $s \in \mathcal{S}$ [44, §2.6].

Definition 8.2. Given a Harmony function H, a λ-*diffusion network* $\mathfrak{N}$ is defined by the dynamics

$$\mathcal{D}_\lambda(t) \equiv \lambda(t)\mathcal{D}_{\mathrm{opt}}^{T(t)} + (1-\lambda(t))\mathcal{D}_{\mathrm{quant}}$$

$$\text{i.e. } \mathrm{d}\mathbf{s} = [\lambda \nabla H(\mathbf{s}) + (1-\lambda)\mathbf{Q}(\mathbf{s})]\,\mathrm{d}t + \lambda(2T)^{1/2}\,\mathrm{d}\mathbf{B},$$

where λ and T are functions of computational time t such that $\lambda(t)$ and $T(t)$ decrease to 0 as $t \to \infty$.

The λ-diffusion dynamics $\mathcal{D}_\lambda$ (definition 8.2) linearly combines the Harmony-optimization dynamics $\mathcal{D}_{\mathrm{opt}}^T$ (theorem 7.13), which ignores discreteness, and the quantization dynamics $\mathcal{D}_{\mathrm{quant}}$ (theorem 8.1), which ignores Harmony [45]. Although formal results have not yet been achieved, in a range of (simple) applications to modelling human language processing, λ-diffusion networks have

proved capable of reliably computing the vectorial realizations of globally optimal symbolic states, when the speed at which $T(t)$ and $\lambda(t)$ go to zero is sufficiently slow. When too fast, errors are made; the outputs are symbolic states, but not necessarily globally optimal ones. In fact, in a number of cases, the probability of outputting a symbolic state s is approximately proportional to $e^{H_{\mathcal{N}}(s)/T}$ (for some T): the relative Harmonies of error states s govern their relative probabilities. In these applications, the distribution of errors captures the key properties of human performance. For example, because faithfulness Harmony formalizes relational similarity of an error to a correct response (theorem 7.11), the probability of an error type decreases appropriately as the similarity between the error and the correct response decreases; and because markedness and faithfulness Harmony together formalize grammatical knowledge (theorems 7.5 and 7.7), the probability of ungrammatical errors is appropriately small.

9. Conclusion

A detailed summary was provided in §2*b*; only a few concluding remarks are given here.

This work seeks not to go beyond classical computability, but rather to explore what restrictions may be placed on cognitive functions assuming that they are computed in accord with a certain conception of neural computation. In this conception, cognition can be characterized by functions over meaningful symbols, but not as computation over such symbols. Vector spaces (of neural network states) are seen to provide a powerful representational medium for the computation of symbolic cognitive functions, even restricting to linear processing. In such vector spaces, a cognitively significant class of recursive functions can be computed in a single massively parallel step. A natural characterization of the functions computed by certain neural networks is through optimization, and as a result neural computation has led to powerful insights into one of the deepest realms of symbolic cognitive science, the theory of universal grammar. Formal languages can be specified in such neural computational terms, as well as natural languages; and while effective computability has not been proved, simulations suggest that these models simultaneously capture central features of both the idealized computational capacity of human linguistic competence and the errorful real-time computation of human linguistic performance.

I warmly thank my collaborators Alan Prince, Géraldine Legendre, Matthew Goldrick, Donald Mathis, Bruce Tesar, John Hale and Yoshiro Miyata in this work; special thanks to Don and Matt for comments on an earlier draft of this paper (I am solely responsible for any errors, of course). For helpful discussion of the work I thank Colin Wilson, James McClelland, Robert Frank, Robert Cummins and most of all the late David E. Rumelhart. I gratefully acknowledge partial financial support for this work from the National Science Foundation, Johns Hopkins University, les Chaires internationales de recherche Blaise Pascal, la Laboratoire de Sciences Cognitives et Psycholinguistique/CNRS and the Robert J. Glushko and Pamela Samuelson Foundation. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation or other sponsors.

References

1 Turing, A. M. 1936 On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* **42**, 230–265.
2 Pylyshyn, Z. W. 1984 *Computation and cognition: toward a foundation for cognitive science*. Cambridge, MA: MIT Press.
3 Smolensky, P. 1988 On the proper treatment of connectionism. *Behav. Brain Sci.* **11**, 1–74. (doi:10.1017/S0140525X00052432)
4 Copeland, B. J. & Proudfoot, D. 1996 On Alan Turing's anticipation of connectionism. *Synthese* **108**, 361–377.
5 Hofstadter, D. R. 1985 Waking up from the Boolean dream, or, subcognition as computation. In *Metamagical themas*, pp. 631–665. New York, NY: Basic Books.
6 Cummins, R. & Schwarz, G. 1991 Connectionism, computation, and cognition. In *Connectionism and the philosophy of mind* (eds T. E. Horgan & J. Tienson), pp. 60–73. Dordrecht, The Netherlands: Kluwer.
7 Smolensky, P. 2006 Computational levels and integrated connectionist/symbolic explanation. In *The harmonic mind: from neural computation to optimality-theoretic grammar*, vol. 2, pp. 503–592. Cambridge, MA: MIT Press.
8 McClelland, J. L. 2010 Letting structure emerge: connectionist and dynamical systems approaches to cognition. *Trends Cogn. Sci.* **14**, 348–356. (doi:10.1016/j.tics.2010.06.002)
9 Smolensky, P. & Legendre, G. 2006 *The harmonic mind: from neural computation to optimality-theoretic grammar*, vol. 1: *Cognitive architecture*, vol. 2: *Linguistic and philosophical implications*. Cambridge, MA: MIT Press.
10 Grossberg, S. 1982 *Studies of mind and brain: neural principles of learning, perception, development, cognition, and motor control*. Boston, MA: Reidel.
11 Kohonen, T. 1977 *Associative memory: a system-theoretical approach*. New York, NY: Springer.
12 Smolensky, P. 1986 Neural and conceptual interpretations of parallel distributed processing models. In *Parallel distributed processing: explorations in the microstructure of cognition*, vol. 2 (eds J. L. McClelland, D. E. Rumelhart & the PDP Research Group), pp. 390–431. Cambridge, MA: MIT Press.
13 Smolensky, P. 1996 Dynamical perspectives on neural networks. In *Mathematical perspectives on neural networks* (eds P. Smolensky, M. C. Mozer & D. E. Rumelhart), pp. 245–270. Mahwah, NJ: Erlbaum Associates.
14 Fodor, J. A. 1975 *The language of thought*. Cambridge, MA: Harvard University Press.
15 Smolensky, P. 1990 Tensor product variable binding and the representation of symbolic structures in connectionist networks. *Artif. Intell.* **46**, 159–216.
16 Abelson, H., Sussman, G. J. & Sussman, J. 1985 *Structure and interpretation of computer programs*. Cambridge, MA: MIT Press.
17 Smolensky, P. 2006 Formalizing the principles I: representation and processing in the mind/brain. In *The harmonic mind: from neural computation to optimality-theoretic grammar*, vol. 1, pp. 147–205. Cambridge, MA: MIT Press.
18 Messiah, A. 1961 *Quantum mechanics*. Amsterdam, The Netherlands: Elsevier Science.
19 Smolensky, P. 2006 Tensor product representations: formal foundations. In *The harmonic mind: from neural computation to optimality-theoretic grammar*, vol. 1, pp. 271–344. Cambridge, MA: MIT Press.
20 Hinton, G. E., McClelland, J. L. & Rumelhart, D. E. 1986 Distributed representation. In *Parallel distributed processing: explorations in the microstructure of cognition*, vol. 1 (eds D. E. Rumelhart, J. L. McClelland & the PDP Research Group), pp. 77–109. Cambridge, MA: MIT Press.
21 Churchland, P. S. & Sejnowski, T. J. 1992 *The computational brain*. Cambridge, MA: MIT Press.
22 Abbott, L. & Sejnowski, T. J. (eds) 1999 *Neural codes and distributed representations*. Cambridge, MA: MIT Press.
23 Fischer-Baum, S. 2010 Position representation: general principles or domain-specificity? Doctoral dissertation. Johns Hopkins University, Baltimore, MD.

24 Legendre, G., Miyata, Y. & Smolensky, P. 1990 Harmonic grammar—a formal multi-level connectionist theory of linguistic well-formedness: theoretical foundations. In *Proc. Cognitive Science Society*, pp. 388–395. Cambridge, MA: Erlbaum.

25 Hale, J. & Smolensky, P. 2006 Harmonic grammars and harmonic parsers for formal languages. In *The harmonic mind: from neural computation to optimality-theoretic grammar*, vol. 1, pp. 393–415. Cambridge, MA: MIT Press.

26 Prince, A. & Smolensky, P. 1993/2004 *Optimality theory: constraint interaction in generative grammar*. Technical Report, Rutgers University and University of Colorado at Boulder, 1993. ROA 537, 2002. Revised version published by Blackwell, 2004.

27 Prince, A. & Smolensky, P. 1997 Optimality: from neural networks to universal grammar. *Science* **275**, 1604–1610.

28 Grimshaw, J. & Samek-Lodovici, V. 1998 Optimal subjects and subject universals. In *Is the best good enough? Optimality and competition in syntax* (eds P. Barbosa, D. Fox, P. Hagstrom, M. McGinnis & D. Pesetsky), pp. 193–219. Cambridge, MA: MIT Press.

29 Kager, R. 1999 *Optimality theory*. Cambridge, UK: Cambridge University Press.

30 McCarthy, J. J. (ed.) 2004 *Optimality theory in phonology: a reader*. Malden, MA: Blackwell.

31 Legendre, G., Grimshaw, J. & Vikner, S. (eds) 2001 *Optimality-theoretic syntax*. Cambridge, MA: MIT Press.

32 Blutner, R., De Hoop, H. & Hendriks, P. 2006 *Optimal communication*. Stanford, CA: CSLI Publications.

33 Prince, A. 2002 Anything goes. In *A new century of phonology and phonological theory* (eds T. Honma, M. Okazaki, T. Tabata & S. Tanaka), pp. 66–90. Tokyo, Japan: Kaitakusha.

34 Soderstrom, M., Mathis, D. W. & Smolensky, P. 2006 Abstract genomic encoding of universal grammar in optimality theory. In *The harmonic mind: from neural computation to optimality-theoretic grammar*, vol. 2, pp. 403–471. Cambridge, MA: MIT Press.

35 Cohen, M. A. & Grossberg, S. 1983 Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Trans. Syst. Man Cybern.* **13**, 815–825.

36 Hopfield, J. J. 1984 Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl Acad. Sci. USA* **81**, 3088–3092.

37 Golden, R. M. 1986 The 'brain-state-in-a-box' neural model is a gradient descent algorithm. *Math. Psychol.* **30–31**, 73–80. (doi:10.1016/0022-2496(86)90043-X)

38 Kirkpatrick, S., Gelatt Jr, C. D. & Vecchi, M. P. 1983 Optimization by simulated annealing. *Science* **220**, 671–680. (doi:10.1126/science.220.4598.671)

39 Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. 1985 A learning algorithm for Boltzmann machines. *Cogn. Sci.* **9**, 147–169.

40 Smolensky, P. 1986 Information processing in dynamical systems: foundations of harmony theory. In *Parallel distributed processing: explorations in the microstructure of cognition*, vol. 1 (eds D. E. Rumelhart, J. L. McClelland & the PDP Research Group), pp. 194–281. Cambridge, MA: MIT Press.

41 Movellan, J. R. 1998 A learning theorem for networks at detailed stochastic equilibrium. *Neural Comput.* **10**, 1157–1178.

42 Movellan, J. R. & McClelland, J. L. 1993 Learning continuous probability distributions with symmetric diffusion networks. *Cogn. Sci.* **17**, 463–496.

43 Geman, S. & Geman, D. 1984 Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 721–741.

44 Baird, B. & Eeckman, F. 1993 A normal form projection algorithm for associative memory. In *Associative neural memories* (ed. M. H. Hassoun), pp. 135–166. New York, NY: Oxford University Press.

45 Smolensky, P., Goldrick, M. & Mathis, D. W. In press. Optimization and quantization in gradient symbol systems: a framework for integrating the continuous and the discrete in cognition. *Cogn. Sci.*

PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A MATHEMATICAL, PHYSICAL & ENGINEERING SCIENCES

Symbolic functions from neural computation

Paul Smolensky

Phil. Trans. R. Soc. A 2012 **370**, 3543-3569
doi: 10.1098/rsta.2011.0334