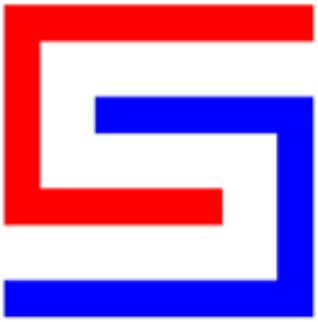
ESSLLI 2022

Proceedings of the 3rd Natural Logic Meets Machine Learning Workshop (NALOMA III)

8–12 August, 2022
University of Galway
Galway, Ireland



Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-959429-39-5

Introduction

Welcome to the 3rd edition of the Natural Logic and Machine Learning workshop (NALOMA). NALOMA aims to bridge the gap between ML/DL and symbolic/logic-based approaches to NLU and lay a focus on hybrid approaches. NALOMA'22 took place in August 8-12, 2022, during ESSLLI 2022, organized at the National University of Ireland Galway.

The workshop run over 5 days of 1.5 hour slots. It included two invited talks, one by Robin Cooper and one by Oana Camburu, and 8 regular talks. A panel discussion followed at the end with Stergios Chatzikyriakidis, Valeria de Paiva and Christos Papadimitriou moderated by Larry Moss. Of the 8 submissions corresponding to the regular talks, 5 of them were short or long papers and are included in the proceedings, and three of them were extended abstracts and are not included in the proceedings.

The workshop was sponsored by the Special Interest Group on Computational Semantics (SIGSEM). We are grateful for their support.

Stergios Chatzikyriakidis and Aikaterini-Lida Kalouli

Crete & Munich

October 2022

Organisers:

Stergios Chatzikyriakidis and Aikaterini-Lida Kalouli

Program Committee:

Stergios Chatzikyriakidis, Aikaterini-Lida Kalouli, Lasha Abzianidze, Katrin Erk, Hai Hu, Thomas Icard, Lawrence S. Moss, Valeria de Paiva, Hitomi Yanaka

Invited Speakers:

Robin Cooper, University of Gothenburg
Oana Camburu, University College, London

Table of Contents

Strings from neurons to language 1
Tim Fernando

Classification Systems: Combining taxonomical and perceptual lexical meaning 11
Bill Noble, Staffan Larsson and Robin Cooper

Learning Knowledge with Neural DTS 17
Daisuke Bekki, Ribeka Tanaka and Yuta Takahashi

Center-Embedding and Constituency in the Brain and a New Characterization of Context-Free Languages 26
Daniel Mitropolsky, Adiba Ejaz, Mirah Shi, Christos Papadimitriou and Mihalis Yannakakis

A Philosophically-Informed Contribution to the Generalization Problem of Neural Natural Language Inference: Shallow Heuristics, Bias, and the Varieties of Inference 38
Reto Gubelmann, Christina Niklaus and Siegfried Handschuh

Strings from Neurons to Language

Tim Fernando
School of Computer Science and Statistics
Trinity College Dublin, Ireland
Tim.Fernando@tcd.ie

Abstract

Chains of transitions by finite automata originally conceived to analyze neural events are described at different granularities by strings. The granularities are refined, and transformed into increasingly elaborate structures, against which to understand the events recorded in the strings. Choosing the correct structure is a problem of induction and learning. The events and strings studied arise in natural language semantics.

1 Introduction

Natural language inference concerns connections that may or may not exist between say, (i) and (ii).

(i) Facebook bought Instagram.

(ii) Facebook owns Instagram.

If (ii) follows from (i), we might add the link

$$\boxed{\text{facebook}} \xrightarrow{\text{owns}} \boxed{\text{instagram}}$$

to a knowledge graph which already has the link

$$\boxed{\text{facebook}} \xrightarrow{\text{bought}} \boxed{\text{instagram}}.$$[1]

But does (ii) follow from (i)? What if an event happened after the purchase (i) transferring ownership of Instagram away from Facebook (perhaps to Meta)? Rather than insisting that buy(X,Y) entails own(X,Y) by leaving out such troublesome events, the present paper proposes that buy(X,Y) entails *Become*(own(X,Y))

$$\boxed{\text{buy(X,Y)}} \Rightarrow \boxed{\textit{Become}(\text{own(X,Y)})} \qquad (1)$$

as pictured by a transition

$$\boxed{\neg\text{own(X,Y)}} \xrightarrow{\text{buy(X,Y)}} \boxed{\text{own(X,Y)}} \qquad (2)$$

(associating the precondition ¬own(X,Y) and postcondition own(X,Y) with the act buy(X,Y)) which may (or may not) follow (or precede) a transition such as

$$\boxed{\text{own(X,Y)}} \xrightarrow{\text{sell(X,Y)}} \boxed{\neg\text{own(X,Y)}}$$

(swapping the preconditions and postconditions in (2)) to describe further changes in ownership. The operator *Become* in (1) can be found in the aspectual calculus of Dowty (1979) and characterized by entailments

$$\boxed{\textit{Become}(\text{A})\,\boxed{\ }} \Rightarrow \boxed{\neg\text{A}}\boxed{\text{A}} \qquad (3)$$

and

$$\boxed{\neg\text{A}}\boxed{\text{A}} \Rightarrow \boxed{\textit{Become}(\text{A})\,\boxed{\ }} \qquad (4)$$

using the same binary connective ⇒ in (1) to map regular languages L and L' to a regular language $L \Rightarrow L'$ (see, for example, §3.4 of Fernando (2015)).[2]

Whatever semantics is (or is not) attached to ⇒ in (1), it is clear that there is more to a *buy*-event than the change in ownership expressed in (1),(2); no mention is made, for instance, of a payment that is part of any *buy*-event. While this omission does not diminish the entailment (1), it suggests there is more to the pre-states and post-states of a buy(X,Y)-transition than is on display in the boxes

$$\boxed{\neg\text{own(X,Y)}} \quad \text{and} \quad \boxed{\text{own(X,Y)}}$$

[1] A recent work on link prediction and entailment graphs is Hosseini et al. (2019).

[2] That is, (1), (3) and (4) are not unlike constraints in finite-state morphology (e.g., Beesley and Karttunen, 2003), except that the symbols constituting the alphabet of the languages for ⇒ are assumed throughout to be sets. These sets are drawn with boxes (rather than the customary curly braces {, } and ∅) to distinguish sets *qua* symbols (as in the string $\boxed{\ }$ of length 1) from sets *qua* languages (e.g., the language ∅ without any strings, not to mention the empty string ϵ of length 0).

in (2). To salvage (2), let us bring out the (bounded) granularity Σ underpinning (2), and assert that if

$$q \overset{\text{buy(X,Y)}}{\longrightarrow} q'$$

then the states q and q' are Σ-approximated by

$$\boxed{\neg\text{own(X,Y)}} \quad \text{and} \quad \boxed{\text{own(X,Y)}}$$

respectively. That is, (2) becomes

$$q \overset{\text{buy(X,Y)}}{\longrightarrow} q' \quad \text{with } \pi_\Sigma(q) = \boxed{\neg\text{own(X,Y)}} \quad \text{and } \pi_\Sigma(q') = \boxed{\text{own(X,Y)}} \tag{5}$$

where π_Σ maps a state to its Σ-approximation. But what exactly is this granularity Σ and map π_Σ? And how can we refine Σ to establish an entailment

$$\boxed{\text{buy(X,Y)}} \Rightarrow \boxed{\text{pay(X,Y)}} \tag{6}$$

injecting an ingredient, pay(X,Y), missing from (1),(2)?

To answer these and related questions, the present work defines three notions, a *transition signature* Σ, a Σ*-strip* and an X*-projection relative to* Σ, under which a chain

$$q_0 \overset{a_1}{\to} q_1 \overset{a_2}{\to} q_2 \overset{a_3}{\to} \cdots \overset{a_n}{\to} q_n \tag{7}$$

of transitions $q_i \overset{a_{i+1}}{\to} q_{i+1}$ from state q_i to state q_{i+1} over a_{i+1} can be formulated as strings of varying granularities, capturing finite fragments of q_i and of a_i. The somewhat surprising suggestion here is that there are strings other than $a_1 a_2 \cdots a_n$ to extract from the chain (7), and proper fragments of q_i and of a_i to describe. This suggestion becomes less surprising when we turn to the source (Kleene, 1956) of finite automata; in the application there to nerve nets, a_i is a set and q_i is a record. This is explained in section 2, where transition signatures Σ and Σ-strips are defined, under which the transition (2) can be encoded as the string

$$\boxed{\text{(own(X,Y),0), buy(X,Y)} \mid \text{(own(X,Y),1)}}$$

of length 2; its first symbol is the box consisting of the act buy(X,Y) and the ordered pair (own(X,Y),0) saying own(X,Y) is 0/false; its second symbol is the box consisting of the single ordered pair (own(X,Y),1) saying own(X,Y) is 1/true. More generally, (7) becomes a string $\alpha_1 \alpha_2 \cdots \alpha_k$ of boxes α_i formed by adding acts to records, or better still (when varying a signature within a category), record types (applied to linguistic semantics in Cooper and Ginzburg (2015)). As for the entailment (6), this is addressed through X-projections (relative to Σ), defined in section 3, where Russell-Wiener event structures (Kamp and Reyle, 1993, pages 667–674) and interval relations from Allen (1983) are revisited. Section 4 outlines how to deploy the three notions defined within logical settings for learning finite automata from strings associated with signatures. The claim is that the step from strings to automata tracks the move from episodic reports (such as (i), *Facebook bought Instagram*) to generic statements such as

(iii) Facebook spreads lies.

The ideas described below, including the connection to neural nets, are intended to make this claim plausible[3] and intriguing.

2 Nerve nets and beyond

Finite automata go back to Kleene (1956)'s analysis of a nerve net from McCulloch and Pitts (1943) consisting of finite numbers k and m of

(i) *input cells*, $\mathcal{N}_1, \ldots, \mathcal{N}_k$, described at different times by different symbols from a finite alphabet A, and

(ii) *inner cells*, $\mathcal{M}_1, \ldots, \mathcal{M}_m$, described at different times by different states from a finite set Q.

In Rabin and Scott (1959), (i) and (ii) are put aside in favor of a "black box" perspective on finite automata, moving them away from nerve nets. Widely adopted in textbook accounts of finite automata, this perspective has proved enormously fruitful. It has, however, also resulted in some ideas from Kleene (1956) being sidelined, including the possibility from (i) and (ii) above that

(†) a state q is an m-tuple $(v_1, \ldots, v_m)$ and a transition $q' \overset{a}{\to} q$ combines m simpler relations $\to_1, \ldots, \to_m$

$$q' \overset{a}{\to} q \quad \text{iff} \quad q' \overset{a}{\to}_i v_i \text{ for all } i \in \{1, \ldots, m\}$$

where $q' \overset{a}{\to}_i v_i$ depends on only certain parts of q' and of a (for $1 \leq i \leq m$).

[3] Generics have been linked to causation (e.g., Carlson, 1995); automata are obvious candidates for causal structures.

The m in (†) is the same number m in (ii) of inner cells $\mathcal{M}_1, \ldots, \mathcal{M}_m$; the state q in (†) assigns values $v_1, \ldots, v_m$ to $\mathcal{M}_1, \ldots, \mathcal{M}_m$, respectively. Apart from k and m, Kleene (1956) assumes each inner cell $\mathcal{M}_i$ can be assigned any of $s_i \geq 2$ different values (of which v_i is one), leading to $\prod_{i=1}^{m} s_i$ many m-tuples $(v_1, \ldots, v_m)$ in the state set Q. The all-or-none assumption of nervous activity in McCulloch and Pitts (1943) is applied to the k input cells in (i) for an alphabet A consisting of the 2^k subsets a of $\{\mathcal{N}_1, \ldots, \mathcal{N}_k\}$, the intention being that a describes a time where an input cell is active (firing) if and only if it is in a. The transition $q' \xrightarrow{a}_i v_i$ in (†) is subject to activation laws specifying how to update the value of $\mathcal{M}_i$ when the inner cells have values q' and a is the set of active input cells.[4] If $q' = (v'_1, \ldots, v'_m)$, then $q' \xrightarrow{a}_i v_i$ does not depend on any v'_j describing an inner cell $\mathcal{M}_j$ that does not feed into $\mathcal{M}_i$ nor on any input cell in a that does not feed into $\mathcal{M}_i$. (The nerve nets may or may not be fully recurrent.)

Extracting the string $a_1 a_2 \cdots a_n$ from the chain

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots \xrightarrow{a_n} q_n \qquad (7)$$

of transitions leaves out the states $q_0, q_1, \ldots, q_n$, which in Kleene (1956) describe m inner cells at $n+1$ times. This reflects a focus on the external environment that is connected to inner cells via inputs cells (described at n times by $a_1 a_2 \cdots a_n$). Away from the particularities of nerve nets, however, no such separation between external and internal matters need keep us from extracting instead the string $q_0 q_1 \cdots q_n$ from (7). In line with Dowty (1979)'s use of stative predicates as the basis for his aspectual calculus, we apply strings $q_0 q_1 \cdots q_n$ in section 3 to represent the finitely many events mentioned in a (finite) discourse. Some middle ground between strings $q_0 q_1 \cdots q_n$ of states and the usual strings $a_1 a_2 \cdots a_n \in A^*$ is staked out by strings $\alpha_0 \alpha_1 \cdots \alpha_n$ of finite sets α_i that provide information about states q_i and symbols a_{i+1} (for $i < n$) alike. That information may, in (†) above, zero in on the parts of q' and of a on which $q' \xrightarrow{a}_i p_i$ depends.

Two basic intuitions shape the work on strings $\alpha_0 \alpha_1 \cdots \alpha_n$ below. The first is that

(∗) a string which represents a chain (7) of transitions is a data point that is to be explained (alongside other data points) with more complex structures

and the second is that

(∗∗) to keep the structures in (∗) managable, we associate a string $\alpha_0 \alpha_1 \cdots \alpha_n$ with a finite granularity which can be refined as information about it accumulates.

Mention of data in (∗) calls for a reference to *grammatical inference* (e.g., Heinz and Sempere, 2016; de la Higuera, 2010). The examples considered here, however, are from natural language semantics. Under (∗), the step from accounts of events that happen (in actuality) up to general statements (including causal claims, counterfactuals and potentiality) is an inductive generalization over strings which demands richer structures. As for (∗∗), the main thrust of the present paper is to formulate granularities as signatures (or vocabularies) familiar in model theory, preparing the ground for logical systems based on signatures called *institutions* (Goguen and Burstall, 1992). The finite signatures formed below keep the structures finite-state (connected in a precise sense with Kleene (1956)), making significant bits of the reasoning decidable (a theme from Rabin and Scott (1959)).

Getting down to business, let us package key aspects of Kleene (1956) in a signature, following Table 1 above.

Kleene (1956)	transition signature
inner cell $\mathcal{M}_i$	label $\in \mathbf{L}$
$(s_1, \ldots, s_m)$	value-sets $\{\mathcal{V}(l)\}_{l \in \mathbf{L}}$
input cell $\mathcal{N}_i$	act $\in \mathbf{Act}$
neural connections	$\mathsf{af} : \mathbf{Act} \to 2^{\mathbf{L}}$

Table 1: Transition signatures in Kleene (1956)

Definition 1. A *transition signature* is a 4-tuple $\Sigma = (\mathbf{L}, \mathcal{V}, \mathbf{Act}, \mathsf{af})$, where

(i) $\mathbf{L}$ is a finite set of labels,

(ii) $\mathcal{V}$ is a function with domain $\mathbf{L}$ assigning each label l a finite set $\mathcal{V}(l)$ of l-values,

(iii) $\mathbf{Act}$ is a finite set of acts distinct from pairs (l, v) of labels l and l-values v

$$\mathbf{Act} \cap \{(l, v) \mid l \in \mathbf{L} \text{ and } v \in \mathcal{V}(l)\} = \emptyset$$

and

[4] These involve thesholds and two types of connections, inhibitory and excitatory (McCulloch and Pitts, 1943), or in the case of perceptrons, weights, biases and activation functions.

(iv) $\mathsf{af} : \mathbf{Act} \to 2^{\mathbf{L}}$ is a function specifying the set $\mathsf{af}(a)$ of labels that an act a can affect.

In Kleene (1956), labels are inner cells, acts are input cells, and af maps every input cell to the set of inner cells it is connected to. For the transition

$$\boxed{\neg\text{own(X,Y)}} \overset{\text{buy(X,Y)}}{\longrightarrow} \boxed{\text{own(X,Y)}} \qquad (2)$$

(from the previous section), let

$$\Sigma \;=\; (\{\text{own(X,Y)}\},\ \mathcal{V},\ \{\text{buy(X,Y)}\},\ \mathsf{af})$$

where

$$\mathcal{V}(\text{own(X,Y)}) = \{0, 1\}$$

and

$$\mathsf{af}(\text{buy(X,Y)}) = \{\text{own(X,Y)}\}$$

to encode (2) as the string

$$\boxed{\text{(own(X,Y),0), buy(X,Y)}\,|\,\text{(own(X,Y),1)}}.$$

In general, a transition signature Σ has a *stative part* $Q(\Sigma)$ equal to the set of $\mathcal{V}$-records, where a *$\mathcal{V}$-record* is a function q with domain $\mathbf{L}$ mapping each $l \in \mathbf{L}$ to an l-value $q(l) \in \mathcal{V}(l)$. The disjointness in clause (iii) of Definition 1 prevents any confusion when forming a string $\alpha_0\alpha_1 \cdots \alpha_n$ of subsets α_i of

$$\mathbf{Act} \cup \{(l, v) \mid l \in \mathbf{L} \text{ and } v \in \mathcal{V}(l)\}$$

to specify a chain

$$q_0 \overset{a_1}{\to} q_1 \overset{a_2}{\to} q_2 \overset{a_3}{\to} \cdots \overset{a_n}{\to} q_n \qquad (7)$$

of transitions where q_i is the part of α_i with*out* acts

$$q_i := \alpha_i \setminus \mathbf{Act} \quad \text{(i.e., } \{a \in \alpha_i \mid a \notin \mathbf{Act}\})$$

and a_{i+1} is the subset of α_i consisting of acts

$$a_{i+1} := \alpha_i \cap \mathbf{Act} \quad (\text{for } i < n).$$

We can sidestep the disjointness requirement by turning each set α_i in $\alpha_0\alpha_1 \cdots \alpha_n$ into an ordered pair (q_i, a_{i+1}) of a $\mathcal{V}$-record q_i and subset a_{i+1} of **Act**; we opt here instead for the union

$$q_i \cup a_{i+1} = \alpha_i.$$

Let us define a *Σ-box* α to be the union of a $\mathcal{V}$-record with a subset of **Act**. Given a Σ-box α and a label $l \in \mathbf{L}$, let us agree that the *value of l at α* is the unique l-value v such that $(l, v) \in \alpha$. We say α and α' are *l-equivalent* and write $\alpha =_l \alpha'$ if l has the same value at α and α'. To express the idea that adjacent boxes in a string are l-equivalent unless the boxes are linked by an act affecting l, let $\overline{\mathsf{af}} : 2^{\mathbf{Act}} \to 2^{\mathbf{L}}$ be the function mapping each $\alpha \subseteq \mathbf{Act}$ to the set

$$\overline{\mathsf{af}}(\alpha) := \mathbf{L} \setminus \bigcup_{a \in \alpha} \mathsf{af}(a)$$

of labels *not* in $\mathsf{af}(a)$ for any $a \in \alpha$. For example,

$$\overline{\mathsf{af}}(\emptyset) = \mathbf{L}$$

as there is no act in $\emptyset$ to affect a label. A label is said to be *unaffected by* α if it belongs to $\overline{\mathsf{af}}(\alpha)$. Next we define strings basic to this paper.

Definition 2. Given a transition signature $\Sigma = (\mathbf{L}, \mathcal{V}, \mathbf{Act}, \mathsf{af})$, a *$\Sigma$-strip* is a string $\alpha_1 \cdots \alpha_n$ of Σ-boxes α_i such that $\alpha_n \cap \mathbf{Act} = \emptyset$ and for all i such that $1 \leq i < n$, $\alpha_i \cap \mathbf{Act} \neq \emptyset$ and

$$\alpha_i =_l \alpha_{i+1} \text{ for each } l \in \overline{\mathsf{af}}(\alpha_i \cap \mathbf{Act}). \qquad (8)$$

Line (8) in Definition 2 comes with a slogan

no change without force

on the understanding that $\alpha_i =_l \alpha_{i+1}$ means "no change" and that **Act** covers all relevant forces. (8) gives us a handle on change and the tendency to infer (ii) from (i) in the absence of any act affecting own(facebook,instagram) after a buy(facebook,instagram)-event.

(i) Facebook bought Instagram.

(ii) Facebook owns Instagram.

More on af and on what it says about refinements of Σ in the next section.

3 Events from intervals to strings

"An important part" of interpreting "a piece of discourse" is representing the "comparatively few events" mentioned in it, according to Kamp (2013). An event e is assumed in Allen (1983) and Kamp and Reyle (1993) to stretch over a temporal interval, leaving times before and after e. Under this assumption, a set E of events induces a notion of time as follows. Let us define an *E-state* $q = (U, A, D)$ to be a triple of subsets U, A, D of E that are pairwise disjoint and cover E

$$U \cap A = \emptyset \text{ and } D = E \setminus (U \cup A).$$

The idea is that (U, A, D) describes a time that is

(i) before every event in U (making U the set of *u*nborn events in E),

(ii) during every event in A (making A the set of *a*live events in E), and

(iii) after every event in D (making D the set of *d*ead events in E).

To capture the order implicit in this idea, we let Q_E be the set of E-states, and we represent the passage of time by a binary relation $\rightarrow_E$ on Q_E such that

$$(U, A, D) \rightarrow_E (U', A', D')$$

means

$$U' \subseteq U \text{ and } A \neq A' \text{ and } D \subseteq D' \subseteq D \cup A. \quad (9)$$

(9) says unborn events were in the past unborn ($U' \subseteq U$), the set of alive events changes ($A \neq A'$), and dead events stay dead, having at the previous moment been alive or dead ($D \subseteq D' \subseteq D \cup A$). To associate a transition signature $\Sigma = (\mathbf{L}, \mathcal{V}, \mathbf{Act}, \mathsf{af})$ with $\rightarrow_E$, we let $\mathbf{L} = E$, and fix a set {u,a,d} of three values to which $\mathcal{V}$ maps every event in E, identifying an E-state $q = (U, A, D)$ with the function $\hat{q} : E \rightarrow$ {u,a,d} mapping $e \in E$ according to which of U, A, D has e

$$\hat{q}(e) := \begin{cases} \mathrm{u} & \text{if } e \in U \\ \mathrm{a} & \text{if } e \in A \\ \mathrm{d} & \text{otherwise (i.e., } e \in D). \end{cases}$$

The three e-values (u,a,d) are more than the two (0,1) needed by a label l such as own(facebook,instagram) to say l is true or false. For transitions such as $\rightarrow_E$ that do not specify any acts, we can express that non-specification through an anonymous act ☺ that can affect any of the labels. Putting

$$\mathbf{Act} = \{☺\} \text{ and } \mathsf{af}(☺) = \mathbf{L}$$

completes the $\rightarrow_E$-column of Table 2. An alternative to ☺ is to associate every event $e \in E$ with a left border l_e and a right border r_e for a set

$$E^{l,r} := \{l_e \mid e \in E\} \cup \{r_e \mid e \in E\}$$

of borders of E.[5] From the definition (9) of $(U, A, D) \rightarrow_E (U', A', D')$ above, we can then extract a non-empty subset

$$\{l_e \mid e \in A' \cap U\} \cup \{r_e \mid e \in D' \cap A\} \quad (10)$$

Σ	$\rightarrow_E$	actions (10)	synthesis
$\mathbf{L}$	E	$\emptyset$	E
$\mathcal{V}$	$\lambda e.\{\mathrm{u}, \mathrm{a}, \mathrm{d}\}$	$\emptyset$	$\lambda e.\{\mathrm{u}, \mathrm{a}, \mathrm{d}\}$
$\mathbf{Act}$	$\{☺\}$	$E^{l,r}$	$E^{l,r}$
af	$\{(☺, E)\}$	$\lambda a.\emptyset$	af_E

Table 2: Transition signatures for E as interval-strings

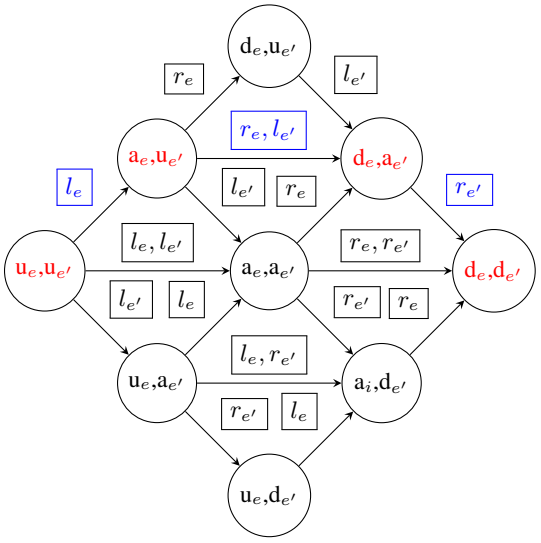


Figure 1: The relation $\rightarrow_{\{e,e'\}}$ labelled by actions (10)

of $E^{l,r}$, to express the transition from (U, A, D) to (U', A', D'). We can also use an event e as a subscript on the value an E-state q associates with e, under the repackaging $\hat{q} = \{v_e \mid e \in E\}$ where v_e abbreviates the pair (e, v) in $\hat{q}$. For instance, for $E = \{e, e'\}$, we can shorten the E-state $(E, \emptyset, \emptyset)$ to $\{\mathrm{u}_e, \mathrm{u}_{e'}\}$, the E-state $(\{e'\}, \{e\}, \emptyset)$ to $\{\mathrm{a}_e, \mathrm{u}_{e'}\}$, the E-state $(\emptyset, \{e'\}, \{e\})$ to $\{\mathrm{d}_e, \mathrm{a}_{e'}\}$, and the E-state $(\emptyset, \emptyset, E)$ to $\{\mathrm{d}_e, \mathrm{d}_{e'}\}$. These four E-states appear in red in Figure 1, with the sets (10) as boxes over arrows given by $\rightarrow_E$. The three blue boxes in Figure 1 form the string

$$\boxed{l_e}\boxed{r_e, l_{e'}}\boxed{r_{e'}} \quad (11)$$

corresponding to the Allen interval relation *e meets* e' (called abutment in Kamp and Reyle (1993)). All 13 interval relations in Allen (1983) are expressed in Figure 1 as strings labelling transitions from $\{\mathrm{u}_e, \mathrm{u}_{e'}\}$ to $\{\mathrm{d}_e, \mathrm{d}_{e'}\}$. The 13 strings over the 8 symbols $\boxed{l_e}$, $\boxed{l_e, l_{e'}}$, $\boxed{l_e, r_{e'}}$, $\boxed{r_e}$, $\boxed{r_e, l_{e'}}$, $\boxed{r_e, r_{e'}}$, $\boxed{l_{e'}}$, $\boxed{r_{e'}}$ appear in Durand and Schwer (2008) with*out* E-states. The derivation (10) of l_e and r_e from E-states supports the intuition de-

[5] An event e is, as it were, born with the injunction **l**ive, l_e, and dies with the injunction **r**est, r_e.

fended in Allen (1983) that intervals are conceptually prior to points such as l_e and r_e.

Indeed, we can construe l_e as *Become*(a_e) and r_e as *Become*(d_e), where *Become* is one of the "three or four sentential operators and connectives" through which David Dowty explains "the different aspectual properties of the various kinds of verbs" on the basis of "a single homogeneous class of predicates — stative predicates" (Dowty, 1979, page 71). The pairs a_e and d_e in *Become*(a_e) and *Become*(d_e) are stative insofar as they make up an E-state $\hat{q}$, changes to which are trigerred by *actions* made up of l_e and r_e.

Strings of actions such as

$$\boxed{l_e}\boxed{r_e, l_{e'}}\boxed{r_{e'}} \tag{11}$$

differ from strings of E-states such as

$$\boxed{\mathrm{u}_e, \mathrm{u}_{e'}}\boxed{\mathrm{a}_e, \mathrm{u}_{e'}}\boxed{\mathrm{d}_e, \mathrm{a}_{e'}}\boxed{\mathrm{d}_e, \mathrm{d}_{e'}} \tag{12}$$

(red in Figure 1) in an important respect that is revealed when reducing the set E of events to a smaller set. For this, a definition is helpful. Given a set X and a string $s = \alpha_1 \cdots \alpha_n$ of sets α_i, the *X-reduct $\rho_X(s)$ of s* is s intersected componentwise with X

$$\rho_X(\alpha_1 \cdots \alpha_n) \;:=\; (\alpha_1 \cap X) \cdots (\alpha_n \cap X)$$

(Fernando, 2015). For example, the $\{l_e, r_e\}$-reduct of (11) is

$$\rho_{\{l_e, r_e\}}(\boxed{l_e}\boxed{r_e, l_{e'}}\boxed{r_{e'}}) \;=\; \boxed{l_e}\boxed{r_e}\boxed{} \tag{13}$$

while the $\{\mathrm{u}_e, \mathrm{a}_e, \mathrm{d}_e\}$-reduct of (12) is

$$\boxed{\mathrm{u}_e}\boxed{\mathrm{a}_e}\boxed{\mathrm{d}_e}\boxed{\mathrm{d}_e}. \tag{14}$$

Strings (13) and (14) can be extracted from the chain

$$\boxed{\mathrm{u}_e} \xrightarrow{\boxed{l_e}} \boxed{\mathrm{a}_e} \xrightarrow{\boxed{r_e}} \boxed{\mathrm{d}_e} \xrightarrow{\boxed{}} \boxed{\mathrm{d}_e} \tag{15}$$

of transitions, which we can truncate to

$$\boxed{\mathrm{u}_e} \xrightarrow{\boxed{l_e}} \boxed{\mathrm{a}_e} \xrightarrow{\boxed{r_e}} \boxed{\mathrm{d}_e} \tag{16}$$

in accordance with the Aristotelian dictum

no time without change

where change is observed through the elements of X. Truncating (15) to (16) removes the empty box $\boxed{}$ in (13) and the stutter $\boxed{\mathrm{d}_e}\boxed{\mathrm{d}_e}$ in (14). This suggests forming the X-projection of a string s by compressing its X-reduct $\rho_X(s)$; that compression is Durand and Schwer (2008)'s deletion $d^{\square}$ of $\boxed{}$

$$d^{\square}(\epsilon) \;:=\; \epsilon \quad \text{(empty string)}$$
$$d^{\square}(\alpha s) \;:=\; \begin{cases} d^{\square}(s) & \text{if } \alpha = \boxed{} \\ \alpha d^{\square}(s) & \text{otherwise} \end{cases}$$

and Fernando (2015)'s elimination $\mathbb{k}$ of stutters

$$\mathbb{k}(s) \;:=\; s \qquad \text{if length}(s) < 2$$
$$\mathbb{k}(\alpha\alpha' s) \;:=\; \begin{cases} \mathbb{k}^A(\alpha' s) & \text{if } \alpha = \alpha' \\ \alpha \mathbb{k}^A(\alpha' s) & \text{otherwise.} \end{cases}$$

Returning to the transition signatures in Table 1, the two middle columns (in blue and red) agree in allowing every act to affect every label

$$\mathsf{af}(a) = \mathbf{L} \quad \text{for every } a \in \mathbf{Act} \tag{17}$$

(as ☺ is the only act in the $\rightarrow_E$-column, and there are no labels in the column next to it). For the fourth component af of a signature Σ to do any work (i.e., for line (8) in Definition 2 to be non-vacuous), neither its stative part $Q(\Sigma)$ nor its active part $\mathbf{Act}$ should be trivial. This brings us to the rightmost column of Table 1, where the specificity of the acts l_e and r_e is captured by the equation

$$\mathsf{af}_E(l_e) = \{e\} = \mathsf{af}_E(r_e) \quad \text{for } e \in E$$

which, if $|E| = 1$, reduces to (17) but is quite different otherwise.

The question arises: how do we define the X-projection of a string s of sets with non-trivial stative and non-stative parts? We compress its X-reduct $\rho_X(s)$ by splitting X between its intersections with $\mathbf{Act}$ and with the complement of $\mathbf{Act}$

$$A = X \cap \mathbf{Act} \;\text{ and }\; B = X \setminus \mathbf{Act}.$$

In case $B = \emptyset$, we remove all occurrences of the empty box from $\rho_A(s)$ for $d^{\square}(\rho_A(s))$. Otherwise, we eliminate stutters $\alpha\alpha$ whenever α does not intersect A, as carried out by κ^A

$$\kappa^A(s) \;:=\; s \qquad \text{if length}(s) < 2$$
$$\kappa^A(\alpha\alpha' s) \;:=\; \begin{cases} \kappa^A(\alpha' s) & \text{if } \alpha = \alpha' \text{ and} \\ & \quad \alpha \cap A = \emptyset \\ \alpha \kappa^A(\alpha' s) & \text{otherwise} \end{cases}$$

(so that $\not\kappa$ is just $\kappa^{\emptyset}$). Putting these two cases together, let the (A, B)-*projection* $\kappa_{A,B}(s)$ *of* s be

$$\kappa_{A,B}(s) := \begin{cases} d^{\Box}(\rho_A(s)) & \text{if } B = \emptyset \\ \kappa^A(\rho_{A \cup B}(s)) & \text{otherwise.} \end{cases}$$

For the record, we have

Definition 3. Given a set X and a string s of sets, the X*-projection of* s *relative to* a transition signature $\Sigma = (\mathbf{L}, \mathcal{V}, \mathbf{Act}, \mathsf{af})$ is the (A, B)-projection $\kappa_{A,B}(s)$, where A is $X \cap \mathbf{Act}$ and B is $X \setminus \mathbf{Act}$. When it is clear what Σ is, we shorten $\kappa_{A,B}(s)$ to $\kappa_X(s)$ and refer to it simply as the X*-projection of* s.

If $\hat{s}$ is the string

$\mathrm{u}_e, \mathrm{u}_{e'}, l_e$	$\mathrm{a}_e, \mathrm{u}_{e'}, r_e, l_{e'}$	$\mathrm{d}_e, \mathrm{a}_{e'}, r_{e'}$	$\mathrm{d}_e, \mathrm{d}_{e'}$

then $\kappa_{\{l_e, r_e, \mathrm{u}_e, \mathrm{a}_e, \mathrm{d}_e\}}(\hat{s})$ is the string

u_e, l_e	a_e, r_e	d_e

depicting the chain (16) above. In this particular case, the X-projection of a Σ-strip is a Σ_X-strip, where Σ_X is the transition signature that X reduces Σ to. In general, however, the X-projection of a Σ-strip need not be a Σ_X-strip. Such projections can be viewed as disfavored models, where we might find counterexamples to *Facebook owns Instagram* even though *Facebook bought Instagram*.

Let us summarize this section. Transitions $\rightarrow_E$ based on a set E of events-as-intervals are strung out to Σ-strips, turning E into a set of labels of records, boxed alongside acts. The sortal distinction between acts and statives (built into the transition signature Σ) is applied to the compression of X-reducts, yielding X-projections at a granularity coarser than Σ.[6]

4 Finite-state elaborations

Definitions 1-3 from sections 2 and 3 above are part of an attempt to work out (over strings) a basic aspectual difference between *buy* and *own*, glossed over by links

$$\boxed{\text{facebook}} \xrightarrow{\text{owns}} \boxed{\text{instagram}} \qquad (18)$$

and

$$\boxed{\text{facebook}} \xrightarrow{\text{bought}} \boxed{\text{instagram}} \qquad (19)$$

(from a knowledge graph), but represented by a transition

$$\boxed{\neg\text{own(X,Y)}} \xrightarrow{\text{buy(X,Y)}} \boxed{\text{own(X,Y)}} \qquad (2)$$

(in a finite automaton). The transition (2) suggests that inferring (18) from (19) is (to put it gently) complicated. But if we are to take the transition (2) seriously as a tool for lexical semantics, we must acknowledge too that buy(X,Y) is more complicated than (2), involving, as it does, acts such as pay(X,Y) left out of (2). Accordingly, we take pains to associate a certain transition signature $\Sigma_\circ$ with (2), which we encode as the $\Sigma_\circ$-strip

(own(X,Y),0), buy(X,Y)	(own(X,Y),1)

(20)

(see section 2). This $\Sigma_\circ$-strip can be obtained from transition signatures with larger vocabularies through X-projections, where X is the set

$$\{\text{(own(X,Y),0)}, \text{buy(X,Y)}, \text{(own(X,Y),1)}\}$$

from which the boxes in (20) are formed (see section 3). In particular, we may add an act pay(X,Y) to the transition signature $\Sigma_\circ$ for a more refined transition signature on which to impose the entailment

$$\boxed{\text{buy(X,Y)}} \Rightarrow \boxed{\text{pay(X,Y)}} \qquad (6)$$

adding pay(X,Y) to the first box in (20) because that box has buy(X,Y). Fleshing out the preconditions and postconditions of pay(X,Y) may require further expansions of the transition signature's vocabulary. Each expansion is finite and is (with any luck) not the last, reflecting the open-endedness of events described in natural language. Refining Σ may not only fill boxes in a Σ-strip; it may also lengthen the Σ-strip, as one transition follows another. This is why we consider chains of more than a single transition, and why (i) does *not* entail (ii).

(i) Facebook bought Instagram.

(ii) Facebook owns Instagram.

The increase in string length is turned into a decrease when, in section 3, the X-reduct $\rho_X(s)$ of a string s is compressed to form its X-projection $\kappa_X(s)$ (relative to a signature distinguising acts from the label-value pairs of records). This is because a projection moves to a coarser granularity, rather than (as in the case of an embedding) a finer

[6] With the stative/non-stative distinction in place, events as intervals can be refined to Vendler classes (e.g., Moens and Steedman, 1988; Fernando, 2020).

one. More precisely, given a category **Sign** of signatures where a morphism $\sigma : \Sigma \to \Sigma'$ embeds a signature, Σ, into a finer one, Σ', a functor **Mod** that is *contra*variant on **Sign** returns a projection $\mathbf{Mod}(\sigma)$ coarsening $\mathbf{Mod}(\Sigma')$ down to $\mathbf{Mod}(\Sigma)$. The category **Sign** and functor **Mod** constitute part of a logical system

$$(\mathbf{Sign}, \mathbf{Mod}, \mathit{Sen}, \{\models_\Sigma\}_{\Sigma \in |\mathbf{Sign}|})$$

called an *institution* (Goguen and Burstall, 1992) in which

(i) the functor **Mod** maps Σ contravariantly to a category $\mathbf{Mod}(\Sigma)$ of Σ-models,

(ii) a covariant functor Sen maps Σ to a set $\mathit{Sen}(\Sigma)$ of Σ-sentences, and

(iii) for each signature Σ, $\models_\Sigma$ is a binary relation between Σ-models and Σ-sentences that meets a certain *Satisfaction Condition* discussed below.

But how is a string of sets to be understood as a model of predicate logic? For any set U and string $s = \alpha_1\alpha_2 \cdots \alpha_n$ of subsets α_i of U, let $M_U[s]$ be the U*-structure*

$$M_U[s] \;=\; ([n], <_n, \{[\![P_u]\!]\}_{u \in U})$$

over a universe $[n] \;=\; \{1, 2, \ldots, n\}$ of string positions with $<_n$ as the usual $<$ restricted to $[n]$, interpreting, for every $u \in U$, a unary relation symbol P_u as the set

$$[\![P_u]\!] = \{i \in [n] \mid u \in \alpha_i\}$$

of positions in s where u occurs. Forming unary predicate symbols P_u from elements u of a string symbol α is "unconventional" (Vu et al., 2018), the custom being instead to name unary predicates P_α after the string symbol α in its entirety (not generally assumed to be a set with noteworthy elements). This shift from α to an element $u \in \alpha$ is consequential, but preserves the Büchi-Elgot-Trakhtenbrot theorem characterizing regular languages as the sets of strings definable in *Monadic Second-Order Logic* over strings (e.g., Libkin, 2004, Theorem 7.21). For any subset X of U, the X-structure $M_X[\rho_X(s)]$ associated with the X-reduct $\rho_X(s)$ of s is the U-structure $M_U[s]$ with P_u restricted to $u \in X$.

Transition signatures add a bit more information about the set U of subscripts u on unary predicates

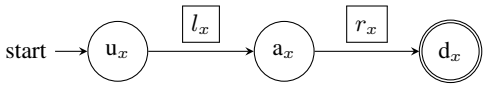


Figure 2: Interval $\boxed{\mathrm{u}_x, l_x}\boxed{\mathrm{a}_x, r_x}\boxed{\mathrm{d}_x}$ as an automaton

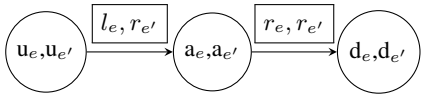


Figure 3: The shortest (middle) path in Figure 1

P_u, separating acts a from label-value pairs, and specifying the set $\mathsf{af}(a)$ of labels whose values an act a can *af*fect. Linked in section 2 to connections in nerve nets from input cells (acts) to inner cells (labels), the function af motivates the compression of X-reducts $\rho_X(s)$ of a string s, based on two dicta that bring up inertia

- *no time without change*

- *no change without force*

(meaning: *no* stuttering stative boxes *nor* empty boxes of acts). Compressing reducts deviates from the convention in institutions of using reducts for the contravariant functor **Mod**, altering a model's universe (of string positions) and damaging a property called amalgamation that is of some interest (e.g., Diaconescu, 2012; Sannella and Tarlecki, 2015). That damage is illustrated dramatically by the thirteen Allen interval relations from the conjunction of two Allen intervals; in pictures, Figure 1 from section 3 arises from Figure 2 with $x \in \{e, e'\}$ (e.g., Fernando, 2020). Without compression, Figure 1 would collapse to its shortest path, Figure 3, with e and e' marching in lockstep (born at the same time, and died at the same time).

Initial and final states are designated in Figure 2 to form a finite automaton, pointing more generally to the matter of computing constraints on strings beyond the reach of af. The clues from $\mathsf{af}(a)$ fall short of a specification of a's effects, never mind its preconditions. This is where the Σ-sentences φ from the functor Sen come in, each of which defines, via the relation $\models_\Sigma$, a set

$$\mathbf{Mod}_\Sigma(\varphi) \;:=\; \{s \in \mathbf{Mod}(\Sigma) \mid s \models_\Sigma \varphi\}$$

of strings that we can assume is accepted by some finite automaton, provided we are careful enough with our choice of $\mathit{Sen}(\Sigma)$. The aforementioned Büchi-Elgot-Trakhtenbrot theorem provides an obvious candidate, but a number of representations of

regular languages (beginning with Kleene (1956)'s regular expressions) are known. The pay-off from working with such representations is that the entailment from φ to ψ given by the inclusion

$$\mathbf{Mod}_\Sigma(\varphi) \subseteq \mathbf{Mod}_\Sigma(\psi)$$

of two regular languages is decidable. (Inclusion between say, context-free languages is not.)

There is no shortage of finite-state toolkits about. Mechanical support for interval reasoning in temporal annotation in TimeML (e.g., Pustejovsky et al., 2010) is described in Woods and Fernando (2018), based on a simplification of the string in Figure 2 to $\boxed{\ }\boxed{x}\boxed{\ }$, construable here as

$$\boxed{(x,0)}\boxed{(x,1)}\boxed{(x,0)}$$

with two values (0,1), rather than three (u,a,d). To represent acts such as buy(X,Y) along with their preconditions and effects, it is natural to box records and acts, connected by more interesting choices of af than those explored in section 3. But already with a simple interval x, its different representations raise the problem of semantic interoperability. We can formulate translations between representations two ways:

(i) within an institution, the **Sign**-morphisms in which may go beyond inclusions $\subseteq$ that **Mod** turns into X-projections, or

(ii) between institutions, each of which can be kept simple, if (as with signatures in **Sign**) there can be another to improve it.

The possibility in (ii) of multiple institution*s* points to logical pluralism (e.g., Kutz et al., 2010), cautioning against turning Definitions 1-3 into a single institution where all signatures can be found (and justifying some vagueness about what **Sign**, **Mod** and *Sen* precisely are). That said, any institution must meet a *Satisfaction Condition* asserting that for any **Sign**-morphism $\sigma : \Sigma \to \Sigma'$, Σ'-model s' and Σ-sentence φ,

$$s' \models_{\Sigma'} Sen(\sigma)(\varphi) \iff \mathbf{Mod}(\sigma')(s') \models_\Sigma \varphi.$$

For the special case of $\Sigma = (\mathbf{L}, \mathcal{V}, \mathbf{Act}, \mathsf{af})$ and $\Sigma' = (\mathbf{L}', \mathcal{V}', \mathbf{Act}', \mathsf{af}')$ where

$$\mathbf{L} \subseteq \mathbf{L}' \text{ and } \mathcal{V}' \restriction \mathbf{L} = \mathcal{V} \text{ and } \mathbf{Act} \subseteq \mathbf{Act}' \quad (21)$$

we can set $\mathbf{Mod}(\sigma)(s')$ to $\kappa_{voc(\Sigma)}(s')$ where the *vocabulary* $voc(\Sigma)$ *of* Σ is the set

$$voc(\Sigma) := \mathbf{Act} \cup \{(l, v) \mid l \in \mathbf{L} \text{ and } v \in \mathcal{V}(l)\}$$

of acts and label-value pairs, some subsets of which go into the set

$$\mathcal{B}_\Sigma := \{a \cup r \mid a \subseteq \mathbf{Act} \text{ and } r \in Q(\Sigma)\}$$

of Σ-*boxes* that are strung together into Σ-models $s \in {\mathcal{B}_\Sigma}^+$. Construing a Σ'-model s' as the $voc(\Sigma')$-structure $M_{voc(\Sigma')}[s']$ defined above, we can apply the translation scheme machinery in Makowsky (2004) to analyze $\kappa_{voc(\Sigma)}(s')$ as well as the Σ'-sentence $Sen(\sigma)(\varphi)$, abbreviated $\langle\sigma\rangle\varphi$, such that

$$s' \models_{\Sigma'} \langle\sigma\rangle(\varphi) \iff \kappa_{voc(\Sigma)}(s') \models_\Sigma \varphi.$$

The idea is $\kappa_{voc(\Sigma)}(s')$ restricts $M_{voc(\Sigma')}[s']$'s universe to string positions x satisfying the disjunction

$$\phi_\Sigma(x) := \chi_{\mathbf{Act}}(x) \vee \chi'_{voc(\Sigma)\setminus\mathbf{Act}}(x) \vee \exists y(xSy \wedge \chi_{\mathbf{Act}}(y))$$

where $\chi_{\mathbf{Act}}(x)$ says an act is done at x

$$\chi_{\mathbf{Act}}(x) := \bigvee_{a \in \mathbf{Act}} P_a(x)$$

while $\chi'_B(x)$ says some binding from B holds at x but not at x's successor

$$\chi'_B(x) := \bigvee_{u \in B} (P_u(x) \wedge \neg\exists y(xSy \wedge P_u(y))$$

(amounting to a B-discernible change at x), where S is the usual successor relation definable from $<$

$$xSy := x < y \wedge \neg\exists z(x < z \wedge z < y). \quad (22)$$

It is convenient here that $<$, rather than S, is primitive, as $\kappa_{voc(\Sigma)}(s')$ simply restricts $<$ to ϕ_Σ, and similarly with P_u, for $u \in voc(\Sigma)$. Not so with S, which the translation machinery analyzes as (22).

What about **Sign**-morphisms $\sigma : \Sigma \to \Sigma'$ for which the inclusions in (21) above do not hold? It suffices that σ come with a function $f_\sigma : \mathcal{B}_{\Sigma'} \to \mathcal{B}_\Sigma$ reducing a Σ'-box α' to a Σ-box $f_\sigma(\alpha')$, which we can extend homomorphically to ${\mathcal{B}_{\Sigma'}}^* \to {\mathcal{B}_\Sigma}^*$ before compressing by either $d^\Box$ provided $voc(\Sigma) \subseteq \mathbf{Act}$, or $\kappa^{\mathbf{Act}}$ otherwise. The resulting composition is the $voc(\Sigma)$-projection $\kappa_{voc(\Sigma)}$ in case $f_\sigma(\alpha') = \alpha' \cap voc(\Sigma)$. In general, the point is to apply $\kappa_{voc(\Sigma)}$ after a map f_σ which adds *no* information in that for every Σ'-box α',

$$\alpha' \vdash f_\sigma(\alpha')$$

where $\vdash$ is a suitable notion of entailment. I hope to write elsewhere about some interesting examples of $\vdash$ (as well as f_σ), and what these have to do with Kleene (1956), in particular, with changes to the m-tuple $(s_1, \ldots, s_m)$ specifying the number of values that the inner cells $\mathcal{M}_1, \ldots, \mathcal{M}_m$ can take.

Bibiliographic note Connections between the present work and action signatures in M. Gelfond and V. Lifschitz 1998 (Action languages, Linköping Electronic Articles in Computer and Information Science, 3:16) are described in a companion paper, T. Fernando 2022 (Action signatures and finite-state variations, Proc ESSLLI Workshop: AREA II, Annotation, Recognition and Evaluation of Actions), where the relation (21) above between transition signatures Σ and Σ' is generalized to incorporate a notion of blurring (turning the records making up the stative part $Q(\Sigma)$ into record types).

References

J.F. Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

K.R. Beesley and L. Karttunen. 2003. *Finite State Morphology*, volume 2900 of LNCS. CSLI, Stanford.

G.N. Carlson. 1995. Truth conditions of generic sentences: Two contrasting views. In G.N. Carlson and F.J. Pelletier, editors, *The Generic Book*, pages 224–237. University of Chicago Press.

R. Cooper and J. Ginzburg. 2015. TTR for natural language semantics. In S. Lappin and C. Fox, editors, *Handbook of Contemporary Semantic Theory*, second edition, pages 375–407. Wiley-Blackwell.

R. Diaconescu. 2012. Three decades of institution theory. In J-Y Béziau, editor, *Universal Logic: An Anthology*, pages 309–322. Springer.

D.R. Dowty. 1979. *Word Meaning and Montague Grammar*. Reidel.

I.A. Durand and S.R. Schwer. 2008. A tool for reasoning about qualitative temporal information: the theory of S-languages with a Lisp implementation. *J. Univers. Comput. Sci.*, 14(20):3282–3306.

T. Fernando. 2015. The semantics of tense and aspect: a finite-state perspective. In S. Lappin and C. Fox, editors, *Handbook of Contemporary Semantic Theory*, second edition, pages 203–236. Wiley-Blackwell.

T. Fernando. 2020. Temporal representations with and without points. In R. Loukanova, editor, *Logic and Algorithms in Computational Linguistics 2018*, pages 45–66. Springer.

J.A. Goguen and R.M. Burstall. 1992. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146.

J. Heinz and J. Sempere, editors. 2016. *Topics in Grammatical Inference*. Springer-Verlag.

C. de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

M.J. Hosseini, S.B. Cohen, M. Johnson, and M. Steedman. 2019. Duality of link prediction and entailment graph induction. In *Proc 57th ACL*, pages 4736–4746. Florence, Italy.

H. Kamp. 2013. The time of my life. Available at https://lucian.uchicago.edu/blogs/elucidations/files/2013/08/KampTheTimeOfMyLife.pdf.

H. Kamp and U. Reyle. 1993. *From Discourse to Logic*. Kluwer Academic Publishers.

S.C. Kleene. 1956. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press.

O. Kutz, T. Mossakowski, and D. Lücke. 2010. Carnap, Goguen, and the Hyperontologies: Logical pluralism and heterogeneous structuring in ontology design. *Logica Universalis*, 4:255–333.

L. Libkin. 2004. *Elements of Finite Model Theory*. Springer.

J.A. Makowsky. 2004. Algorithmic uses of the Feferman-Vaught theorem. *Annals of Pure and Applied Logic*, 126:159–213.

W. S. McCulloch and W. H. Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5:115–133.

M. Moens and M. Steedman. 1988. Temporal ontology and temporal reference. *Computational Linguistics*, 14(2):15–28.

J. Pustejovsky, K. Lee, H. Bunt, and L. Romary. 2010. ISO-TimeML: An international standard for semantic annotation. In *Proc 7th International Conference on Language Resources and Evaluation (LREC'10)*, pages 394–397.

M.O. Rabin and D.S. Scott. 1959. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125.

D. Sannella and A. Tarlecki. 2015. The foundational legacy of ASL. In *Software, Services and Systems: Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering*, volume 8950 of LNCS, pages 253–272. Springer.

M.H. Vu, A. Zehfroosh, K. Strother-Garcia, M. Sebok, J. Heinz, and H.G. Tanner. 2018. Statistical relational learning with unconventional string models. *Frontiers in Robotics & AI*, 5.

D. Woods and T. Fernando. 2018. Improving string processing for temporal relations. In *Proc 14th Joint ISO-ACL Workshop on Interoperable Semantic Annotation (ISA-14)*, pages 76–86.

Classification systems: Combining taxonomical and perceptual lexical meaning

Bill Noble **Staffan Larsson** **Robin Cooper**
Centre for Linguistic Theory and Studies in Probability (CLASP)
Dept. of Philosophy, Linguistics and Theory of Science
University of Gothenburg
{bill.noble@, sl@ling, cooper@ling}.gu.se

Abstract

Lexical meaning includes both perceptual and logical aspects. We present a method for combining a taxonomy with perceptual classifiers, and show that in the few-shot setting, it outperforms other methods of injecting taxonomical information in image classification. We use this method to define witness conditions for types in a rich type system with probabilistic type judgments and suggest how such a type system can be used as the basis for a new type of hybrid NLU architecture.

For words like *red*, *apple*, and *hug*, part of what it means for a person—or indeed an artificial NLU system—to understand the word's meaning is the ability to recognize that some object is red, or an apple, or that some event is one in which hugging is taking place. Marconi (1997) calls this **referential competence**. Another mode of understanding is supported by **inferential competence**, which has to do with the relationship that certain lexical items have with one another—a system that infers that John is not married from the sentence *John is a bachelor* demonstrates inferential competence with the words *bachelor* and *married*. Marconi (1997) argues that neither of these competencies are reducible to the other, meaning that a comprehensive theory of lexical meaning must explain both referential and inferential ability.

In this paper, we propose a framework for combining taxonomical information, which supports an inferential competence, with perceptual classifiers, which implement referential competence. This *classification system* is formalized in a rich type theory with probabilistic type judgments, meaning it can be integrated in a formal semantics based on Type Theory with Records (Cooper et al., 2015).[1]

[1]A PyTTR implementation of a classification systems based on convolutional visual classifiers is available online here: https://github.com/GU-CLASP/classification-systems. We also make available the code for the experiments conducted in Section 4.

1 Classifier-based perceptual meaning

While distributional methods of representing meaning have achieved a lot of success, many have argued that that relying on exclusively *ungrounded* meaning representations has fundamental limitations (Harnad, 1990; Bender and Koller, 2020; Bisk et al., 2020).

Classifier semantics offers a way to ground lexical meaning, operating on the intuition that part of what it means to understand the meaning of a word is to be able to identify instances of it based on perceptual input.

In one approach to classifier semantics (e.g., Schlangen et al., 2016; Silberer et al., 2017), the parameters of a learned classifier (for example, the relevant row of a liner classifier's weight matrix) are regarded as a distributed representation of the meaning of the word. Alternatively, it is possible to regard the classifier itself, as a function of type $f : PerceptualData \rightarrow [0, 1]$, that provides the semantics of the relevant word (e.g., Larsson, 2020a). Here, both the parameters of the classifier and the classification algorithm are considered to be part of the perceptual meaning, whereas in the distributed approach, the classification algorithm is simply a means by which a distributed representation is learned.

In this work, we take a functional approach to classifier semantics. Because they can (at least for one-place predicates) be considered analogous to Montague's $e \rightarrow t$ type, it is natural to integrate classifiers-as-functions in a type-theoretic approach to compositional meaning. Furthermore, classifiers have the nice theoretical property that they can distinguish between intentional identity and extensional equivalence (Muskens, 2005; Lappin, 2012; Larsson, 2020b).

A *multi-class classifier*, C for a set of labels, L is a function that takes an input and produces a prediction among the labels in the form of a prob-

ability distribution. We will consider multi-class classifiers that take perceptual data as input:[2]

$$C : \textit{PerceptualData} \rightarrow (L \rightarrow [0, 1]),$$

subject to the restriction that for any input a, $\sum_{l \in L} C(a)(l) = 1$.

2 Folk taxonomies

A *folk taxonomy* is a hierarchically structured collection of conceptual categories that is *common ground*, in the sense of Clark (1996), in a certain speech community. We wish to invoke a more general notion than that of scientific or technical taxonomies that rely on an authoritative reference for their common ground status. Folk taxonomies by contrast can be informal, emerging from the communicative needs of a particular community and changing in response to changes in the environment. Such a taxonomy can also be established in an *ad hoc* way between a group of speakers, grounded in a particular interaction.

For now, we define a taxonomy in set theoretic terms. A taxonomy takes the form

$$\mathrm{Tax} := \langle \mathrm{Taxon}, \mathrm{Set}(\mathrm{Set}(\mathrm{Tax})) \rangle,$$

where Taxon is the label for a taxonomical category. A taxonomy bottoms out in pairs of the form $\langle \mathrm{Taxon}, \varnothing \rangle$, which we refer to as *leaf taxons*.

Notice that the second element of Tax is a set of *sets* of taxonomies. To see why this is, we will first introduce the notion of a *distinction*, which is a pair that takes the following form:

$$\mathrm{Dist} : \langle \mathrm{Taxon}, \mathrm{Set}(\mathrm{Taxon}) \rangle.$$

Consider this taxonomy:

$\langle \mathit{object}, \{\{\langle \mathit{animal}, \{$
$\{\langle \mathit{mammal}, \{...\}\rangle, ..., \langle \mathit{bird}, \{...\}\rangle\},$
$\{\langle \mathit{herbivore}, \varnothing\rangle, \langle \mathit{omnivore}, \varnothing\rangle, \langle \mathit{carnivore}, \varnothing\rangle\}\},$
$\langle \mathit{vegetable}, \{...\}\rangle, \langle \mathit{mineral}, \{...\}\rangle\}\}\rangle$

Here, *animal* is subject to two distinctions: the distinction based on diet, and the one that categorizes animals as *mammal*s, *bird*s, and so on.

[2] In the remainder of the paper, we restrict our attention to classifiers and taxonomies of individuals, so we assume that *PerceptualData* is of a kind that corresponds to entities of type *Ind*. In general, however, we can also classify other kinds of entities (events, relations between individuals, etc.).

In the following, we let $\mathit{dist} : \mathrm{Tax} \rightarrow \mathrm{Set}(\mathrm{Dist})$ be the function from a taxonomy to its distinctions.

A *genus-species* relation holds between a taxon and the (first component of) an element of one of its distinctions. In the above example, both *mammal* and *herbivore* are species of *animal*.[3] Conceptually, the key feature of a distinction is that it implies an exhaustive partition of the genus into a set of mutually exclusive species. Note however that we need not assume every species is associated with a lexical item—there can, for example, be a catch-all species in cases where the named alternatives don't cover the entire genus.[4]

This leaves us with two main desiderata for when we start giving content to our taxonomy in the next section.

1. An instance of a species is an instance of its corresponding genus.

2. An instance of a genus is an instance of exactly one species in each of its distinctions.

3 Classification systems

By associating a word with a prediction class of a classifier, a system can be endowed with at least some referential competence. Similarly, associating a word with taxon gives a system some inferential competence in relation to other words embedded in the taxonomy. In this section, we describe a *classification system*, which combines classifiers and a taxonomy to integrate these two kinds of competence.

With this in mind, we will formalize a classification system as a rich Martin-Löf (1984)-style type system that allows for probabilistic type judgments (as in Cooper et al., 2015). Furthermore, we will assume that we can provide basic types with *witness conditions* that ground type judgments. From the perspective of an agent, a type's witness conditions are the methods by which an agent may judge something to be of that type (Cooper, forthc).

[3] For word senses, this is referred to as a *hypernym-hyponym* relation.

[4] Generally we would expect a conventionalized taxonomy to make distinctions in a systematic way; that is, where the species within a distinction are differentiated along some common dimension or set of dimensions. This intuition can be traced back at least to Aristotle's *Categories*. However, his is not a formal requirement of a taxonomy at this stage and nor could it be since, taxons are not yet associated with any kind of content that could be considered as features or establish differentia. Such content will come by way of classifiers in Section 3.

Suppose we have a taxonomy $\mathbf{T}$, and a classifier, C_d, for each distinction $d \in dist(\mathbf{T})$. For each taxon, t, in the taxonomy, we want to define a type, T_t, with the appropriate witness conditions such that $p(a : T_t)$ estimates the probability that a belongs to the taxon, according to the classifiers.

Intuitively, the classifiers give content to the distinctions of the taxonomy by *distinguishing* between species. The classifier is thus premised on the assumption that the object of classification certainly belongs to *one* the species, s_i, among which it distinguishes, meaning that it must in turn belong to the associated genus, g. In practice, this means that the classifier for a given distinction is trained on the subset of labeled data from the associated genus. The classifier's prediction, $C_d(a)(s_i)$, can thus be interpreted as the conditional probability that a has belongs to s_i, given that it belongs to g.

There is one taxon in the taxonmy—the root taxon—that is not a species in any distinction. Let T_{t^*}, which we will refer to as the *domain* classification system, be the type associated with the root taxon. We will assume that T_{t^*} is *universal* in the sense that it is witnessed by any object:[5]

$$p(a : T_{t_*}) = 1 \tag{1}$$

Every other taxon is a species in some distinction, meaning that we have a classifier associated with it. Let $d = \langle g, \{s_1, ..., s_n\}\rangle \in dists(\mathbf{T})$ be a distinction. We define auxiliary types, $T'_{s_1} ... T'_{s_n}$ with witness conditions as follows:

$$p(a : T'_{s_i}) = C_d(a)(s_i). \tag{2}$$

That is, an object a is judged to be of type T'_{s_i} with probability equal to the probability assigned by the classifier for the corresponding distinction.

The interpretation of the classifier as providing a conditional probability suggests that we should define T_{s_i} such that:[6]

$$p(a : T'_{s_i}) = p(a : T_{s_i} \mid a : T_g) \tag{3}$$

We also want T_{s_i} to satisfy the desiderata from the end of Section 2, which can be restated as follows:

$$p(a : T_{s_i}) \leq p(a : T_g) \tag{4}$$

and

$$p(a : T_{s_i} \mid T_g) = 1 - \sum_{j \neq i} p(a : T_{s_j} \mid a : T_g) \tag{5}$$

With this in mind, we let the witness conditions for T_{s_i} be defined as the product of the probability assigned to T'_s and T_g:[7]

$$p(a : T_{s_i}) = p(a : T'_{s_i}) \cdot p(a : T_g) \tag{6}$$

By induction on the taxonomy and the base case of T_{t*}, this gives us well-defined witness conditions for for every taxon t.

Briefly, we will show that this definition meets each of our desiderata. In the following, let $\langle g, \{s_1, ..., s_n\}\rangle$ be a distinction. Without loss of generality, we consider the case of T_{s_i}.

We get (4) directly from (6), since $0 \leq p(a : T'_{s_i}) \leq 1$. As a result of (4) we may write $T_{s_i} \sqsubseteq T_g$—i.e., that T_{s_i} is a *subtype* of T_g (Cooper et al., 2015). Furthermore, this has the consequence that

$$p(a : T_g | a : T_{s_i}) = 1 \tag{7}$$

From Bayes Theorem and (7), we can prove (3):

$$\begin{aligned} &p(a : T_{s_i} \mid a : T_g) \\ &\quad = \frac{p(a : T_g \mid a : T_{s_i}) \cdot p(a : T_{s_i})}{p(a : T_g)} \\ &\quad = \frac{p(a : T_{s_i})}{p(a : T_g)} \\ &\quad = \frac{p(a : T'_{s_i} \cdot p(a : T_g)}{p(a : T_g)} \\ &\quad = p(a : T'_{s_i}) \end{aligned}$$

Finally, (5) follows from (3) and the fact that $\sum_{i \leq n} C_d(a)(s_i) = 1$.

4 Empirical comparison

To investigate how well the classification system performs in practice, we compare it with two other plausible methods of combining classification with taxonomical hierarchy. We put aside type theory

[5] This assumption is convenient for simplicity, but it also works if T_{t_*} is given some constant prior or well-defined witness conditions as part of some larger type system in which the classification system is embedded.

[6] This corresponds to the probability that a is of type T_{s_i} given that it is of type T_g, though other notions of conditional judgments are possible in probabilistic type theory. See Larsson and Cooper (2021).

[7] Note that T_{s_i} has different witness conditions from that of the meet type $T'_{s_i} \wedge T_g$, as defined in Cooper et al. (2015), since the witness condition for the meet type is defined by the classical Kolmogorov (1950) equation for conjunction:

$$p(a : T'_{s_i} \wedge T_g) = p(a : T'_{s_i}) \cdot p(a : T_g \mid a : T'_{s_i}),$$

which is different since we can't assume that $C_d[s_i]$ is probabilistically independent from $C_{d'}[g]$, where d' is the distinction of which g is a species.

	Precision	Recall	F1
per-distribution	**0.93**	**0.90**	**0.90**
marginalization	0.90	0.86	0.82
hierarchy-agnostic	0.80	0.84	0.81

Table 1: Macro-averaged precision, recall, and F1 score for the three methods of incorporating hierarchy in classification.

for the moment and make a comparison based on metrics that are traditionally used for machine learning classification.

Dhall et al. (2020), proposes several possible methods of incorporating hierarchical information, including the *hierarchy agnostic* and *marginalization* methods that we compare against.[8]

The **hierarchy agnostic** method is the simplest and most common way of dealing with a taxonomically organized label set. Every label is considered by a single *multi-label* classifier, without respect to taxonomical hierarchy. There is thus no guarantee that the predicted probabilities will be consistent—the probability assigned to a genus label could be lower than the probability assigned to one of its species, for example. Hopefully the hierarchical relations inherent in the data encourages the classifier to learn a function that approximates the taxonomy.

In the **marginalization** method, a *bottom-up* classifier, is trained on the leaf nodes in the taxonomy. Labels at higher levels are predicted by marginalizing the leaf node probabilities—the probability of a genus label is computed as the sum of the probability of its species labels. Note that this method assumes that the leaf labels are disjoint, meaning that it only works for taxonomies in which there is on distinction per genus.

The system described in Section 3 is will be referred to as the **per-distinction** method. As described there, we train a classifier for each distinction and compute the probability of a given label as the product of the classifier output and the probability assigned to its parent label.

We test each method on a simple synthetic dataset shapes with different colors and sizes. The data was generated with a hierarchical stochastic process reflected in the taxonomy of the labels given to each item. Images were encoded with a convolutional autoencoder, which was pre-trained on images from a larger unstructured sample space.

Each method used simple single-layer linear classifiers trained by stochastic gradient descend through backpropagation. The marginalization and per-distinction classifiers use softmax activations with categorical cross-entropy as the loss function, and the hierarchy agnostic classifier uses a sigmoid activation and binary cross entropy with the indicator function of the item's actual label set. Table 1 gives a summary of the results of the classifiers in a few-shot classification scenario with 5 training instances and 100 testing instances for each leaf label. A separate set of 100 development items were used to choose the best model after 10 epochs of training. For the precision, recall and F1 metrics, the predicted classes were chosen in a greedy fashion from the top of the taxonomy, taking the label with the highest probability consistent with the label chosen at the previous level.

Consistent with Dhall et al. (2020), we find that both methods that explicitly take the label hierarchy into account out-perform the hierarchy agnostic method. In the few-shot experiment reported here, our per-distribution method performed best, though we note that this advantage is less pronounced with more training examples.

5 Conclusion

In this paper we have focused on the problem of integrating perceptual and logical meaning on a lexical level. To do this, we have embedded perceptual classifiers as witness conditions for types in a type system that respects a taxonomical structure. Our method for doing this is based on the intuition that such a taxonomy gives rise to a collection of *distinctions*, whose content can be defined by multi-class classifiers. We have compared our method of embedding classifiers at each node in a taxonomy to other strategies for classifying in a taxonomically structured label space suggested by (Dhall et al., 2020). Future work should also consider the possibility of learning the label hierarchy on the fly, as Bengio et al. (2010) does. Embedding such a hierarchy in a type system may present additional challenges, but allowing for changes to the taxonomy would be necessary to full model the plasticity of the lexical semantic structures used by natural language speakers.

[8] Dhall et al. (2020) also tests a *per-level* and *masked per-level* method, which are arguably most similar to what we propose here. We do not reproduce those tests because marginalization tended to out-perform them in Dhall et al. (2020)'s experiments. Like marginalization, the per-level and masked per-level methods assume that there is a single distinction per genus.

We have also left open the looming question of compositional semantics. We presented classification systems as a rich type system in order to suggest a way forward in this regard. Our proposal is compatible with Type Theory with Records (TTR), which can be used to define version of compositional semantics (Cooper et al., 2015). Indeed, TTR has been used for compositional semantics with perceptual classifier-based meaning (Larsson, 2013, 2017). The issue remains, however of how to compose the types we define in Section 3.

Composing classifiers-as-functions is no easy task.[9] For a given object a, one can compute the probability that a witnesses both T_1 and T_2 simply by taking judgments for T_1 and T_2 separately. The difficulty comes when one needs to reason hypothetically, as is necessary in NLI. What is the likelihood that *some* object of type T_1 is also of type T_2? One way forward is to find a way to compose the classifiers for T_1 and T_2 directly, as Monroe et al. (2017) does for color terms. Another option is to use the classifiers to to sample from conditioned space of objects. Something like this is the basis of the system proposed by Bernardy et al. (2019), though it is not perceptually grounded. In order for that to work, the embedding space of $PerceptualData$ would have to be regularized in such a way that admits sampling, which could potentially be achieved by using a variational autoencoder (Kingma and Welling, 2014).

Aside from compositionality, there remain many questions on the side of lexical representation, such as that of polysemy. It would seem that certain words may appear in multiple places in a taxonomy. The meaning of a word may be ambiguous among a *set* of such corresponding types. So far we have only discussed predicative nouns. Adjectives, and verbs, including transitive verbs admit a similar treatment, but that leaves quantifiers and function words, among others.

Finally, we only discuss perceptual and taxonomical aspects of meaning, but there are other aspects of meaning, including other inferential aspects. How would we represent, for example, that *being from the Champagne region* is an aspect of the meaning of *champagne* (the beverage)? In Marconi (1997)'s schema, this fact would be treated as an aspect of inferential competence. Certainly we should not expect the inference to be derivative of a perceptual classifier for champagne, but it does not fit neatly as taxonomical information either. A more sophisticated type system is needed to incorporate lexical information of this kind.

[9] Importantly, it is a different task from the one of composing distributed representations learned through classification. See Moro et al. (2019) for more on that task.

Acknowledgements

This work was supported by grant 2014-39 from the Swedish Research Council (VR) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

References

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In *ACL 2020*.

Samy Bengio, Jason Weston, and David Grangier. 2010. Label Embedding Trees for Large Multi-Class Tasks. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Jean-Philippe Bernardy, Rasmus Blanck, Stergios Chatzikyriakidis, Shalom Lappin, and Aleksandre Maskharashvili. 2019. Bayesian Inference Semantics: A Modelling System and A Test Suite. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 263–272, Minneapolis, Minnesota. Association for Computational Linguistics.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020. Experience Grounds Language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, Online. Association for Computational Linguistics.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

Robin Cooper. forthc. *From Perception to Communication: A Theory of Types for Action and Meaning*. Oxford University Press.

Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson. 2015. Probabilistic Type Theory and Natural Language Semantics. In *Linguistic Issues in Language Technology, Volume 10, 2015*. CSLI Publications.

Ankit Dhall, Anastasia Makarova, Octavian Ganea, Dario Pavllo, Michael Greeff, and Andreas Krause. 2020. Hierarchical Image Classification using Entailment Cone Embeddings. *arXiv:2004.03459 [cs, stat]*.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Conference Proceedings: Papers Accepted to the International Conference on Learning Representations (ICLR)*, Calgary.

A. N. Kolmogorov. 1950. *Foundations of the Theory of Probability*. New York: Chelsea Pub. Co.

Shalom Lappin. 2012. An Operational Approach to Fine-Grained Intensionality. *UCLA Working Papers in Linguistics, Theories of Everything*, 17:180–186.

Staffan Larsson. 2013. Formal semantics for perceptual classification. *Journal of Logic and Computation*, 25(2):335–369.

Staffan Larsson. 2017. Compositionality for perceptual classification. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short Papers*.

Staffan Larsson. 2020a. Discrete and Probabilistic Classifier-based Semantics. In *Proceedings of the Probability and Meaning Conference (PaM 2020)*, pages 62–68, Gothenburg. Association for Computational Linguistics.

Staffan Larsson. 2020b. Extensions are Indeterminate if Intensions are Classifiers. In *SemDial 2020 (WatchDial) Workshop on the Semantics and Pragmatics of Dialogue*, page 10, Waltham, MA and online.

Staffan Larsson and Robin Cooper. 2021. Bayesian Classification and Inference in a Probabilistic Type Theory with Records. In *Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA)*, pages 51–59, Groningen, the Netherlands (online). Association for Computational Linguistics.

Diego Marconi. 1997. *Lexical Competence*. Language, Speech, and Communication. MIT Press, Cambridge, Mass.

Per Martin-Löf. 1984. *Intuitionistic Type Theory*. Bibliopolis, Naples.

Will Monroe, Robert X.D. Hawkins, Noah D. Goodman, and Christopher Potts. 2017. Colors in Context: A Pragmatic Neural Model for Grounded Language Understanding. *Transactions of the Association for Computational Linguistics*, 5:325–338.

Daniele Moro, Stacy Black, and Casey Kennington. 2019. Composing and Embedding the Words-as-Classifiers Model of Grounded Semantics. *arXiv:1911.03283 [cs]*.

Reinhard Muskens. 2005. Sense and the Computation of Reference. *Linguistics and Philosophy*, 28(4):473–504.

David Schlangen, Sina Zarrieß, and Casey Kennington. 2016. Resolving References to Objects in Photographs using the Words-As-Classifiers Model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1213–1223, Berlin, Germany. Association for Computational Linguistics.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2017. Visually Grounded Meaning Representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2284–2297.

Learning Knowledge with Neural DTS

Daisuke Bekki and **Ribeka Tanaka** and **Yuta Takahashi**
Ochanomizu University
{bekki|tanaka.ribeka|takahashi.yuta}@is.ocha.ac.jp

Abstract

Neural DTS is a framework that combines logic and machine learning by fusing dependent type semantics (DTS) and deep neural networks. In this paper, we propose a learning algorithm for Neural DTS, in which a collection of semantic representations of DTS is obtained from text through syntactic parsing and semantic composition, and the collection of positive predicates is generated through the deduction of DTS. We will also discuss the advantages of this method and the challenges it shall face.

1 Introduction

In the field of natural language understanding (NLU), type-logical semantics and neural language processing exhibit complementary benefits: The former has advantages in the systemic prediction of complex linguistic phenomena such as negation, conditional, quantification, anaphora, presupposition, and modality as they appear in various syntactic structures including embeddings, while the latter has a wide range of applications including similarity calculations, summarization, translation, dialogue processing, and even multimodal inference, as well as being robust and able to quickly process real texts. In recent years, hybrid methods that utilizes both of the techniques have been explored in neighboring fields of NLU, and methods adopted in previous studies can be roughly classified as follows:

1. Emulating symbolic reasoning by embedding into neural networks:
   - knowledge graphs (Guu et al., 2015)(Das et al., 2017)(Takahashi et al., 2018)
   - SAT problems (Selsam et al., 2019)
   - first-order logic (FoL) (Demeester et al., 2016)(Šourek et al., 2018)
2. Introducing a similarity measure between symbols by using distributional representations instead of symbols (Lewis and Steedman, 2013)(Rocktäschel and Riedel, 2017).
3. Controlling the direction of proof search guided by neural networks (Wang et al., 2017).

One method that differs from these approaches is Neural DTS (Bekki et al., 2021). Compared with previous studies, Neural DTS is unique in its use of DTS (Bekki and Mineshima, 2017), a higher-order type-logical semantic framework. Since DTS is based on Martin-Löf type theory (MLTT; Martin-Löf (1984)), a framework for intuitionistic mathematics, it allows for the construction of the real numbers necessary for developping deep neural networks.[1] Neural DTS replaces all of the predicates of DTS with neural classifiers, which provide the soft symbols that are required for soft reasoning. Nevertheless, the entire system is still within the framework of MLTT, and all of the components of the neural networks, such as loss functions, have proof terms.

In this paper, we propose a learning algorithm for Neural DTS that fits the parameters for the names and predicates of Neural DTS to the data generated by the proof system of DTS from the propositions obtained from the real texts. Using this algorithm, we investigate the claim of Bekki et al. (2021) that the symbols (names and predicates) of Neural DTS are learnable, one of the criteria for soft reasoning systems.

1.1 Dependent Type Semantics (DTS)

DTS is a framework for the proof-theoretic semantics of natural language. Unlike similar proof-theoretic attempts[2], DTS takes a verificationist approach, in which the meaning of a proposition in a given context is a collection of proofs of that proposition in that context. This position shows sharp

[1] See Appendix A.4 for details.

[2] cf. Francez and Dyckhoff (2010); Francez (2014).

contrasts to most of the standard model-theoretic semantics, but it provides a more fine-grained notion of meaning than the model-theoretic approaches in the following sense.

1. In model-theoretic semantics, all tautologies have the same meaning (i.e., the set consisting of all models), but in DTS, tautologies with different proofs have different meanings. The same is true for contradictions.

2. There are pairs of sentences that are indistinguishable in their truth conditions but which have different anaphoric potentials. For example (cf. Kamp et al. (2011)):

   (1) a. Some student did not show up. They must have overslept.

   b. Not every student showed up. *They must have overslept.

   To explain this contrast, DRT uses an extra layer called Discourse Representation Structure (DRS), where the difference in the anaphoric potential is explained via the notion of accessibility defined on DRS. In DTS, on the other hand, the difference is explained by the constructivity of the proofs, without introducing another layer to the semantic theory (Bekki, 2014). It has been pointed out that the extra-representational layer in DRT complicates the compositionality problem (Yana et al., 2019), while compositionality is preserved in DTS.

With these features, DTS opens up the analyses of many linguistic phenomena, some of which were previously unexplained by model-theoretic semantics, based on a proof-theoretic perspective where one can refer to a proof as an object.[3] There have also been studies on implementations of type checking (Bekki and Sato, 2015) and proof search (Daido and Bekki, 2020) in DTS, with the goal of developing a research program in which the predictions of formal semantics can be verified through implementation.

To supplement the DTS specifications that are relevant to the main purpose of this paper, first, the enumeration type[4] **entity** plays the role of type e in the standard semantics. Second, the role of type t (or type $prop$) in the standard semantics is played by the type type (namely, propositions, under the Curry-Howard isomorphism between types and propositions). Consequently, an n-place predicate has a type $\mathbf{entity}^n \to \mathsf{type}$. For example, **dog** is a unary predicate with type $\mathbf{entity} \to \mathsf{type}$ and the name $john$ has type **entity**. The $\mathbf{dog}(john)$ has type type, which is a collection of proofs that John is a dog. The truth condition in DTS tells us that $\mathbf{dog}(john)$ is true if and only if it is inhabited by at least one proof.

From the perspective of fusing symbolic and soft reasoning, however, this kind of specification by DTS shares a crucial property with the standard semantics, despite the differences between type e and type **entity**, and type t and type type: the symbols in DTS are neither comparable nor learnable when compared with the distributional representations that are often found in neural networks, which have become a trend in recent neural language processing technology. These properties are often considered to be shortcomings of symbolic reasoning but the advantages of soft reasoning.

1.2 Neural DTS

Bekki et al. (2021) proposed Neural DTS as a framework for combining DTS with some aspects of soft reasoning. Neural DTS is obtained by replacing n-place predicates in DTS (that is, constant symbols of type $\mathbf{entity}^n \to \mathsf{type}$) with neural classifiers, which are also DTS terms of type $\mathbf{entity}^n \to \mathsf{type}$. Since the descriptive power of DTS allows for the construction of real numbers and real functions (or complex numbers and complex functions, if necessary) internally through the notion of setoids[5], it also allows for the implementation of neural networks and neural classifiers in the following way.

Consider DTS with a given signature where:

1. the type **entity** has n-introduction rules (namely, **entity** has the form $\{a_1, \ldots, a_n\}$).

2. there are k-many unary predicates $P_1, \ldots, P_k$, each of which is of type $\mathbf{entity} \to \mathsf{type}$.

Let $\mathbb{ENT}$ be a setoid $(\mathbf{entity}, =_{\mathbf{entity}})$, and let onehot be a setoid function from $\mathbb{ENT}$ to $\mathbb{R}^n$

[3] For recent developments regarding the semantic analyses in DTS, see the discussions on the overwriting problem (Yana et al., 2019), generalized quantifiers (Tanaka, 2021), proviso problems (Yana et al., 2021), and weak crossover (Bekki, 2021).

[4] See Appendix A.2 for the definition of the enumeration type.

[5] See Appendix A.3 for the definition of setoid in DTS.

$$\Gamma,\ p_{\mathsf{emb\_ent}} \in \mathbb{R}^{n\times m},\ p_{\mathsf{emb\_pred}} \in \mathbb{R}^{k\times l},\ p_{\mathsf{hidden}} \in \mathbb{R}^{(m+l)\times o},\ p_{\mathsf{out}} \in \mathbb{R}^{o\times 1}$$
$$\vdash\ \mathsf{pred}(a_i, j) \stackrel{def}{\equiv} \mathsf{sigmoid}(W_{\mathsf{out}}(\mathsf{sigmoid}(W_{\mathsf{hidden}}(W_{\mathsf{emb\_ent}}(\mathsf{onehot}(a_i)) \oplus W_{\mathsf{emb\_pred}}(e_j))))) \in \mathbb{R}$$

Figure 1: Neural predicate in DTS

$$\Gamma,\ p_{\mathsf{emb\_ent}} \in \mathbb{R}^{n\times m},\ p_{\mathsf{emb\_pred}} \in \mathbb{R}^{k\times l},\ p_{\mathsf{hidden}} \in \mathbb{R}^{(m+l)\times o},\ p_{\mathsf{out}} \in \mathbb{R}^{o\times 1}$$
$$\vdash\ \lambda x.\mathsf{pred}(x, j) \geq \mathsf{threshold} : \textbf{entity} \to \mathsf{type}$$

Figure 2: Neural classifier in DTS

that maps each entity (namely, each element a_i of $\{a_1, \ldots, a_n\}$) to the i-th element in the standard basis $e_1, \ldots, e_n$ of an n-dimensional real vector space (defined as a product setoid). Additionally, let W be a linear setoid function from $\mathbb{R}^n$ to $\mathbb{R}^m$ that maps $e_1, \ldots, e_n$ to their respective embeddings in an m-dimensional real vector space.

Next, let assume the following parameters:

$$\begin{aligned} p_{\mathsf{emb\_ent}} &\in \mathbb{R}^{n\times m} \\ p_{\mathsf{emb\_pred}} &\in \mathbb{R}^{k\times l} \\ p_{\mathsf{hidden}} &\in \mathbb{R}^{(m+l)\times o} \\ p_{\mathsf{out}} &\in \mathbb{R}^{o\times 1} \end{aligned}$$

by which we define the following matrices as linear setoid functions:[6]

$$\begin{aligned} W_{\mathsf{emb\_ent}} &\in \mathbb{R}^n \to \mathbb{R}^m \\ W_{\mathsf{emb\_pred}} &\in \mathbb{R}^k \to \mathbb{R}^l \\ W_{\mathsf{hidden}} &\in \mathbb{R}^{(m+l)} \to \mathbb{R}^o \\ W_{\mathsf{out}} &\in \mathbb{R}^o \to \mathbb{R} \end{aligned}$$

Non-linear functions such as $\mathsf{sigmoid} \in \mathbb{R} \to \mathbb{R}$ can also be defined as setoid functions. By combining these components, a simple neural predicate with only one hidden layer can be defined as in Figure 1[7]. Note that neural predicates with more complicated structures such as RNNs and Transformers, are also definable in the same manner as the setoid functions in DTS.

Let us call this setoid function $\mathsf{pred}(a_i, j)$, where a_i is an entity and j indicates that this is the j-th unary predicate ($1 \leq j \leq k$) in the signature. Suppose that $\mathsf{threshold}$ is a real number hyperparameter between 0 and 1. Then the inequality between $\mathsf{pred}(a_i, j)$ and $\mathsf{threshold}$ is a DTS relation. Thus, a neural classifier of type $\textbf{entity} \to \mathsf{type}$ obtained, as in Figure 2. Since $\textbf{entity} \to \mathsf{type}$ is a type for one-place predicates in DTS, we may safely replace it with a corresponding neural classifier. It is straightforward to implement n-ary classifiers in a similar manner.

2 Learning Algorithm for Neural DTS

The next step for Neural DTS, which is absent in previous work, is to *fit* the parameters to data. The main contribution of this paper is to propose such an algorithm. We consider a setting where knowledge is given via real texts with credible content, and where "true" (or grounded) propositions can be extracted from the syntactic-semantic theory of natural language (this will be done in practice by CCG parsers). The algorithm optimizes the parameters in Neural DTS to make this set of propositions true. DTS is the best framework candidate since it has a proof theory (unlike most of the model-theoretic semantics) that enables this kind of proposition extraction, which has already been implemented, as in Bekki and Sato (2015) and Daido and Bekki (2020). The learning algorithm is described as follows.

2.1 Preprocessing

First, we prepare a collection of texts $(\mathsf{TXT}_d)_{d\in D}$ whose contents we assume to be correct. This will

[6] From the theoretical point of view, it is worth noting that all of these are defined within the framework of MLTT. However, from the perspective of implementation, we may simply import an off-the-shelf deep learning library such as pytorch (`https://pytorch.org/`) or tensorflow (`https://www.tensorflow.org/`), and use arrays instead of product setoids. In other words, we don't have to implement setoids on top of the implementation of the DTS's type system.

[7] $\mathsf{sigmoid}$ is a sigmoid function defined as a setoid function. The one applied to the hidden layer is broadcasted to $\mathbb{R}^o$. The $\oplus$ operator is the vector concatenation.

be used as the training data for Neural DTS. Also, let each

$$p_e \in \mathbb{R}^{(n\times m)+(k\times l)+((m+l)\times o)+(o\times 1)}$$

be the parameter of the neural network at epoch e.

1. Perform syntactic analysis on each $(\mathsf{TXT}_d)_{d\in D}$ by using CCG parser. Let $(\mathsf{SYN}_d)_{d\in D}$ be the resulting (1-best) syntactic structures.

2. Perform semantic composition on each $(\mathsf{SYN}_d)_{d\in D}$. Taking DTS to be the semantic theory, it is ensured that a syntactic structure will yield a well-formed semantic representation via syntax-semantics transparency. Let $(\mathsf{SEM}_d)_{d\in D}$ be the resulting DTS representations, where each SEM_d is a type that contains the parameter p_1 (the initial parameter) as a free variable.

2.2 Training Loop

Let e be the epoch. Loop the following steps, starting from $e = 1$.

1. Obtain a collection of *positive* predicates $(\mathsf{pred}(a,j))_{(a,j)\in T^+}$, where T^+ is a subset of **entity** $\times \{1,\ldots,k\}$, each of which being deduced from one of the $(\mathsf{SEM}_d)_{d\in D}$ by using only the elimination rules (such as (ΠE) and (ΣE)).

2. Prepare a collection of *negative* predicates $(\mathsf{pred}(a,j))_{(a,j)\in T^-}$, where T^- is a randomly selected subset of $(\textbf{entity}\times\{1,\ldots,k\}) - T^+$ such that $|T^-| = |T^+|$.

3. The loss function loss_e is a setoid function defined for each epoch e as follows:

$$\begin{aligned}\mathsf{loss}_e \stackrel{def}{\equiv} &\sum_{(a,j)\in T^+} |1-\mathsf{pred}(a,j)|^2 \\ &+ \sum_{(a,j)\in T^-} |0-\mathsf{pred}(a,j)|^2\end{aligned}$$

   loss_e is minimized when each of $(\mathsf{pred}(a,j))_{(a,j)\in T^+}$ is 1 and each of $(\mathsf{pred}(a,j))_{(a,j)\in T^-}$ is 0.

   Notice that loss_e contains a free variable p_e. We need to show that $\Gamma \vdash \lambda p_e.\mathsf{loss}_e \in \mathbb{R}$ is a differentiable function, which is a repetition of the standard argument in analysis but in terms of setoids.

4. Update the parameters. Stop the loop if loss_e falls below a certain threshold; or if e reaches a certain value. Otherwise set $e = e + 1$ and go to the step 1.

   We refer to this set of procedures as *one round*.

The reason for using only the elimination rules in step 1 is that it is computationally faster, in the sense that substitution does not arise in the premise part of the elimination rules. At the same time, however, this restricts the deduction power of DTS at the minimum level. Looser restrictions with greater computational burden should also be investigated.

Note that in this algorithm, the value of the loss function is not guaranteed to converge since the logical deduction of DTS is used in the training loop to generate the training data for each epoch. This feature of the algorithm is a drawback from an engineering viewpoint, but from a cognitive viewpoint, it may reflect the process of how humans learn knowledge, where our beliefs do not always converge by the increase of knowledge. Also, this algorithm proposes a particular method for the interaction of logical deduction and machine learning at the level of the training loop. These are left as topics for further study.

2.3 Evaluation Methods

One way to evaluate this learning algorithm is to check its soundness and completeness: the direct computation is compared with the training set. In this case, precision corresponds to soundness and recall to completeness.

Next, for the embedding of predicates, we would evaluate whether empirically similar predicates have similar embeddings on the Neural DTS. This would follow the standard methods, where measures like 5best, 1best, among others, are valid.

3 Discussion

The advantage of Neural DTS over other representation learning methods using DNN is that it is sensitive to polarities such as negation and conditional clauses in the text. For example, when the proposition A is included in the scope of the negation in one of $(\mathsf{SEM}_d)_{d\in D}$, as in the example below, it is not added to the training data since A is not deduced from $\neg(A\times B)$.

$$\neg(A\times B) \quad \not\vdash \quad A$$

Also, A is not added to the training data when it is included in the antecedent part of the conditional

sentence, as in the example below, since A is not deduced from $A \to B$.

$$A \to B \quad \not\vdash \quad A$$

In other words, due to the soundness of the logical deduction, once the mapping from text to semantic representation is performed, the set of positive predicates obtained from it by deduction is guaranteed to be correct, as long as the text is correct. This is a major advantage of having a logic system like DTS at the core of the knowledge learning system.

On the other hand, the reliability of the training data is sensitive to the validity of the syntactic theory that generates the semantic representations in DTS; here, the CCG parser and the lexicon do the job. Given that the accuracy of off-the-shelf CCG parsers is not yet sufficient, one might rather consider methods that use only dependency parsers more prospective, or even an end-to-end neural system that does not assume the division of labor among these modules from the beginning, that is, a language model such as an RNN or a transformer whose final layer is trained as classifier for predicates. These approaches may seem more robust for some researchers, but less precise for the rest of the researchers. Comparing these approaches is methodologically difficult since they are based on different views on language faculty.

However, using the CCG parser and the approach of making lexical semantics more precise is the study of formal syntax and of formal semantics itself. In other words, in this enterprise, the improvement of the formal syntax and the formal semantics is directly related to the improvement of the lexical semantics and the cognitive capacity.

4 Conclusions and Future Work

This paper proposed a learning algorithm for Neural DTS for acquiring knowledge representations from text, and discussed its features, expected advantages, and difficulties.

The next step is to implement, experiment with, and evaluate this algorithm. Some difficulties are expected to be caused by errors in the syntactic parsing, by ambiguity of predicate symbols, and by the problem of entity size (namely, the number of entities forming the enumeration type).

Moreover, many of the philosophical issues raised in Bekki et al. (2021) are left open. For example, unlike DTS, all predicates have canonical proofs in Neural DTS, the implication of which is still an open question.

We would like to leave these issues, both computational and philosophical, for future discussion.

Acknowledgments We sincerely thank the anonymous reviewers of NALOMA22 for their comments. This work was partially supported by the Japan Science and Technology Agency (JST), CREST Grant Number JPMJCR20D2.22.

Appendix

A Dependent Type Theory (DTT)

A.1 Syntax

Definition A.1 (Alphabet) *An alphabet is a pair $(\mathcal{V}ar, \mathcal{C}on)$ where $\mathcal{V}ar$ is a collection of variables and $\mathcal{C}on$ is a collection of constant symbols.*

Definition A.2 (Preterms) *The collection of preterms of DTT (notation Λ) under an alphabet $(\mathcal{V}ar, \mathcal{C}on)$ is defined by the following BNF grammar, where $x \in \mathcal{V}ar$ and $c \in \mathcal{C}on$.*

$$\begin{array}{l}
\Lambda := x \mid c \mid \mathsf{type} \\
\mid (x:\Lambda) \to \Lambda \mid \lambda x.\Lambda \mid \Lambda\Lambda \\
\mid (x:\Lambda) \times \Lambda \mid (\Lambda,\Lambda) \mid \pi_1(\Lambda) \mid \pi_2(\Lambda) \\
\mid \Lambda \oplus \Lambda \mid \iota_1(\Lambda) \mid \iota_2(\Lambda) \mid \mathsf{unpack}_L\,(M,N) \\
\mid \{a_1,\ldots,a_n\} \mid a_1 \mid \ldots \mid a_n \mid \mathsf{case}^{\Lambda}_{\Lambda}\,(\Lambda,\ldots,\Lambda) \\
\mid \Lambda =_{\Lambda} \Lambda \mid \mathsf{refl}_{\Lambda}(\Lambda) \mid \mathsf{idpeel}^{\Lambda}_{\Lambda}\,(\Lambda) \\
\mid \mathbb{N} \mid \mathsf{0} \mid \mathsf{s}(\Lambda) \mid \mathsf{natrec}^{\Lambda}_{\Lambda}(\Lambda,\Lambda)
\end{array}$$

Free variables, substitutions, β-reductions are defined in the standard way. The full version of DTT also employs well-ordered types and universes, as adopted in Martin-Löf (1984), the detail of which I omit here for the sake of space.

Definition A.3 (Vertical/Box notation)

$$\left[\begin{array}{l} x:A \\ B \end{array}\right] \stackrel{def}{\equiv} (x:A) \times B$$

Definition A.4 (Logical operators) [8]

$$\begin{array}{rcll}
A \to B & \stackrel{def}{\equiv} & (x:A) \to B & where\ x \notin fv(B). \\
\left[\begin{array}{l} A \\ B \end{array}\right] & \stackrel{def}{\equiv} & \left[\begin{array}{l} x:A \\ B \end{array}\right] & where\ x \notin fv(B). \\
\bot & \stackrel{def}{\equiv} & \{\} & \\
\neg A & \stackrel{def}{\equiv} & A \to \bot &
\end{array}$$

[8] $\bot$ is defined as an empty enumeration type.

A.2 Type System

Definition A.5 (Signature) *A collection of signatures (notation σ) for an alphabet $(\mathcal{V}ar, \mathcal{C}on)$ is defined by the following BNF grammar:*

$$\sigma \quad ::= \quad () \mid \sigma, c : A$$

where () is an empty signature, $c \in \mathcal{C}on$ and $\vdash_\sigma A : \mathsf{type}$.

Definition A.6 (Context) *A collection of contexts under a signature σ (notation Γ) is defined by the following BNF grammar:*

$$\Gamma \quad ::= \quad () \mid \Gamma, x : A$$

where () is an empty context, $x \in \mathcal{V}ar$ and $\Gamma \vdash_\sigma \mathsf{type}$.

Definition A.7 (Judgment) *A judgment of DTT is the following form*

$$\Gamma \quad \vdash_\sigma \quad M : A$$

where Γ is a context under a signature σ and M and A are preterms, which states that there exists a proof diagram of DTT from the context Γ to the type assignment $M : A$. The subscript σ may be omitted when no confusion arises.

Definition A.8 (Truth) *The judgment of the form $\Gamma \vdash A\ true$ states that there exists a term M that satisfies $\Gamma \vdash M : A$.*

Definition A.9 (Structural Rules)

$$\frac{A : \mathsf{type}}{x : A}\,(VAR)$$

$$\frac{}{c : A}\,(CON) \quad where\ \sigma \vdash c : A.$$

$$\frac{}{\mathsf{type} : \mathsf{kind}}\,(\mathsf{type}F)$$

$$\frac{M : A \quad N : B}{M : A}\,(WK)$$

$$\frac{M : A}{M : B}\,(CONV) \quad where\ A =_\beta B.$$

Definition A.10 (Π-types)

$$\frac{\begin{array}{c}\overline{x : A}^{\,i} \\ \vdots \\ A : \mathsf{s}_1 \quad B : \mathsf{s}_2\end{array}}{(x : A) \to B : \mathsf{s}_2}\,(\Pi F),i$$

$$where\ (\mathsf{s}_1, \mathsf{s}_2) \in \left\{ \begin{array}{l} (\mathsf{type}, \mathsf{type}), \\ (\mathsf{type}, \mathsf{kind}) \end{array} \right\}.$$

$$\frac{\begin{array}{c}\overline{x : A}^{\,i} \\ \vdots \\ M : B\end{array}}{\lambda x.M : (x : A) \to B}\,(\Pi I),i$$

$$\frac{M : (x : A) \to B \quad N : A}{MN : B[N/x]}\,(\Pi E)$$

Definition A.11 (Σ-types)

$$\frac{\begin{array}{c}\overline{x : A}^{\,i} \\ \vdots \\ A : \mathsf{type} \quad B : \mathsf{type}\end{array}}{(x : A) \times B : \mathsf{type}}\,(\Sigma F),i$$

$$\frac{M : A \quad N : B[M/x]}{(M, N) : (x : A) \times B}\,(\Sigma I)$$

$$\frac{M : (x : A) \times B}{\pi_1(M) : A}\,(\Sigma E)$$

$$\frac{M : (x : A) \times B}{\pi_2(M) : B[\pi_1(M)/x]}\,(\Sigma E)$$

Definition A.12 (Disjoint Union Types)

$$\frac{A : \mathsf{type} \quad B : \mathsf{type}}{A \uplus B : \mathsf{type}}\,(\uplus F)$$

$$\frac{M : A}{\iota_1(M) : A \uplus B}\,(\uplus I)$$

$$\frac{N : B}{\iota_2(N) : A \uplus B}\,(\uplus I)$$

$$\frac{\begin{array}{l} L : A \uplus B \\ P : (A \uplus B) \to \mathsf{type} \\ M : (x : A) \to P(\iota_1(x)) \\ N : (x : B) \to P(\iota_2(x)) \end{array}}{\mathsf{unpack}_L^P\,(M, N) : P(L)}\,(\uplus E),i$$

Definition A.13 (Enumeration Types)

$$\frac{}{\{a_1, \ldots, a_n\} : \mathsf{type}}\,(\{\}F)$$

$$\frac{}{a_i : \{a_1, \ldots, a_n\}}\,(\{\}I)$$

$$\frac{\begin{array}{l} M : \{a_1, \ldots, a_n\} \\ P : \{a_1, \ldots, a_n\} \to \mathsf{type} \\ N_1 : P(a_1) \\ \ldots \\ N_n : P(a_n) \end{array}}{\mathsf{case}_M^P\,(N_1, \ldots, N_n) : P(M)}\,(\{\}E)$$

Definition A.14 (Intensional Equality Types)

$$\frac{A : \mathsf{type} \quad M : A \quad N : A}{M =_A N : \mathsf{type}}\ (=F)$$

$$\frac{M : A}{\mathsf{refl}_A(M) : M =_A M}\ (=I)$$

$$\frac{\begin{array}{l} E : M =_A N \\ P : (x : A) \to (y : A) \to (x =_A y) \to \mathsf{type} \\ R : (x : A) \to Pxx(\mathsf{refl}_A(x)) \end{array}}{\mathsf{idpeel}^P_E(R) : PMNE}\ (=E)$$

Definition A.15 (Natural Number Types)

$$\frac{}{\mathbb{N} : \mathsf{type}}\ (\mathbb{N}F)$$

$$\frac{}{\mathsf{0} : \mathbb{N}}\ (\mathbb{N}I) \qquad \frac{n : \mathbb{N}}{\mathsf{s}(n) : \mathbb{N}}\ (\mathbb{N}I)$$

$$\frac{\begin{array}{l} n : \mathbb{N} \\ P : \mathbb{N} \to \mathsf{type} \\ e : P(\mathsf{0}) \\ f : (k : \mathbb{N}) \to P(k) \to P(\mathsf{s}(k)) \end{array}}{\mathsf{natrec}^P_n(e, f) : P(n)}\ (\mathbb{N}E)$$

A.3 Setoids

Definition A.16 *A setoid is a pair $(\underline{X}, \sim_X)$ consisting of a type $\underline{X}$ and an equivalence relation $\sim_X$ on $\underline{X}$.*

Definition A.16 can be rewritten in the form of the formation rule as follows.

Definition A.17 (Setoid formation)

$$\frac{\begin{array}{l} \underline{X} : \mathsf{type} \\ \sim_X : \underline{X} \times \underline{X} \to \mathsf{type} \\ equiv(\sim_X)\ true \end{array}}{(\underline{X}, \sim_X)\ setoid}$$

Definition A.18 (Setoid membership)

$$\frac{(\underline{X}, \sim_X)\ setoid \quad x : \underline{X}}{x \in (\underline{X}, \sim_X)}$$

Definition A.19 (Setoid function) *A setoid function f from a setoid X to a setoid Y is a pair $(\underline{f}, ext_f)$ consisting of a function $\underline{f} : \underline{X} \to \underline{Y}$ and a proof term ext_f that proves the extensionality of $\underline{f}$:*

$$ext_f \ : \ (x, y : \underline{X}) \to (x \sim_X y) \to (\underline{f}x \sim_Y \underline{f}y)$$

Definition A.20 (Exponential of setoids) *The exponential of setoids $X \equiv (\underline{X}, \sim_X)$ and $Y \equiv (\underline{Y}, \sim_Y)$ is $(\underline{X \to Y}, \sim_E)$ (notation: $X \to Y$), where $\underline{X \to Y}$ is a type defined as:*

$$\underline{X \to Y} \stackrel{def}{\equiv} \big(\underline{f} : \underline{X} \to \underline{Y}\big) \times (x, y : \underline{X}) \to (x \sim_X y) \to (\underline{f}x \sim_Y \underline{f}y)$$

and $\sim_E$ is a binary relation defined as:

$$(\underline{f}, \_) \sim_E (\underline{g}, \_) \stackrel{def}{\equiv} (x : \underline{X}) \to \underline{f}x \sim_Y \underline{g}x$$

A function application operator ev is defined for each domain-codomain pair of setoids.

$$\frac{(\underline{f}, \_) \in X \to Y \quad a \in X}{ev_{X,Y}((\underline{f}, \_), a) \stackrel{def}{\equiv} \underline{f}a \in Y}$$

Definition A.21 (Product of setoids) *The product of setoids $X \equiv (\underline{X}, \sim_X)$ and $Y \equiv (\underline{Y}, \sim_Y)$ is the pair $(\underline{X \times Y}, \sim_P)$ (notation: $X \times Y$), where $\underline{X \times Y}$ is a type defined as:*

$$\underline{X \times Y} \stackrel{def}{\equiv} \underline{X} \times \underline{Y}$$

and $\sim_P$ is a binary relation defined as:

$$(x, y) \sim_P (u, v) \stackrel{def}{\equiv} (x \sim_X u) \times (y \sim_Y v)$$

Projections work as expected.

$$\frac{p \in X \times Y}{\pi_1(p) \in X} \qquad \frac{p \in X \times Y}{\pi_2(p) \in Y}$$

Definition A.22 (Relation on setoids) *A binary relation R between setoids X and Y is a pair $(\underline{R}, ext_R)$ consisting of a binary relation $\underline{R}$ such that $x : \underline{X}, y : \underline{Y} \vdash \underline{R}(x, y) : \mathsf{type}$ together with a proof term ext_R that proves the extensionality of $\underline{R}$:*

$$ext_R \ : \ \big(x, x' : \underline{X}\big) \to \big(y, y' : \underline{Y}\big) \to x \sim_X x' \to y \sim_Y y' \to \underline{R}xy \to \underline{R}x'y'$$

Definition A.23 (Quotient setoids) *Let $X \equiv (\underline{X}, \sim_X)$ be a setoid and $\sim$ be a binary relation on $\underline{X}$ such that $x : \underline{X}, y : \underline{X} \vdash x \sim y : \mathsf{type}$. If $\sim$ is an equivalence relation on $\underline{X}$, we define a quotient setoid $X/\sim$ as:*

$$X/\sim \stackrel{def}{\equiv} (\underline{X}, \sim)$$

with a setoid function $q : X \to X/\sim$ defined by identity function on $\underline{X}$ and its extensionality.

Remark A.24 *By the extensionality of the relation, the following holds for any $x, y \in X$:*

$$x \sim_X y \to x \sim y$$

Thus the equivalence relation $\sim_X$ is finer than $\sim$ on the type $\underline{X}$.

Definition A.25 (Subsetoids) *Let X be a setoid. A subsetoid of X is a pair $(\partial S, i_S)$, where ∂S is a setoid and $i_S : \partial S \to X$ is an injective setoid function.*

Definition A.26 (Subsetoid membership) *Let X be a setoid. An element $a \in X$ is a member of the subsetoid S of X if there exists an element $s : \partial S$ such that $a \sim_X i(s)$, namely:*

$$a \in_X (\partial S, i_S) \stackrel{def}{\equiv} (s : \partial S) \times (a \sim_X i_S(s))$$

Note that $a \in_X (\partial S, i_S)$ is a type. If A and B are subsetoids of X, then

$$A \subseteq_X B \stackrel{def}{\equiv} (x : X) \to x \in_X A \to x \in_X B$$

Definition A.27 (Separation of subsets) *Let $X = (\underline{X}, \sim_X)$ be a setoid and A a type. A subset $\{\ x \in X \mid A\ \}$ of X is defined as:*

$$\{\ x \in X \mid A\ \} \stackrel{def}{\equiv} (((x : \underline{X}) \times A, \sim_S), i)$$

where $(x, \_) \sim_S (y, \_) \stackrel{def}{\equiv} x \sim_X y$

$$i(x, \_) \stackrel{def}{\equiv} x$$

A.4 Setoids of Natural Numbers, Integers, Rationals, and Reals

Definition A.28 *The setoid of natural numbers $\mathbb{N}$ is obtained by embedding the natural number type N with its intensional equality.*

$$\mathbb{N} \stackrel{def}{\equiv} (N, =_N)$$

Definition A.29 *The setoid of integers $\mathbb{Z}$ is defined as:*

$$\mathbb{Z} \stackrel{def}{\equiv} (\mathbb{N} \times \mathbb{N}) / \sim_{\mathbb{Z}}$$

where $\sim_{\mathbb{Z}}$ is defined as $(m, n) \sim_{\mathbb{Z}} (p, q) \stackrel{def}{\equiv} m + q =_N p + n$.

Definition A.30 *The setoid of rational numbers $\mathbb{Q}$ is defined as:*[9]

$$\mathbb{Q} \stackrel{def}{\equiv} (\mathbb{Z} \times \{\ z \in \mathbb{Z} \mid \neg(z \sim_{\mathbb{Z}} 0_{\mathbb{Z}})\ \}) / \sim_{\mathbb{Q}}$$

where $\sim_{\mathbb{Q}}$ is defined as

$$(a, (b, \_)) \sim_{\mathbb{Q}} (c, (d, \_)) \stackrel{def}{\equiv} a \cdot d =_{\sim_{\mathbb{Z}}} c \cdot b$$

Definition A.31 (Cauchy sequence) $\mathsf{Cauchy}(seq)$ *is a proposition that a sequence $seq \in \mathbb{N} \to \mathbb{Q}$ is a Cauchy sequence, defined as follows:*[10]

$$\mathsf{Cauchy}(seq) \stackrel{def}{\equiv}$$
$$(i : \mathbb{N}) \to (i \neq 0) \to (k : \mathbb{N}) \times$$
$$(j_1, j_2 : \mathbb{N}) \to (j_1 > k) \to (j_2 > k)$$
$$\to |(\underline{seq}(j_1)) - (\underline{seq}(j_2))| < \frac{1}{\mathsf{asRational}^*(i)}$$

Definition A.32 *The setoid of real numbers $\mathbb{R}$ is defined as:*

$$\mathbb{R} \stackrel{def}{\equiv} \{\ seq \in \mathbb{N} \to \mathbb{Q} \mid \mathsf{Cauchy}(seq)\ \} / \sim_{\mathbb{R}}$$

where $\sim_{\mathbb{R}}$ is defined as:

$$s_1 \sim_{\mathbb{R}} s_2 \stackrel{def}{\equiv}$$
$$(i : \mathbb{N}) \to (i \neq 0) \to (k : \mathbb{N}) \times (j : \mathbb{N}) \to (j > k)$$
$$\to |\underline{\pi_1(s_1)}(j) - \underline{\pi_1(s_2)}(j)| < \frac{1}{\mathsf{asRational}^*(i)}$$

[9] $0_{\mathbb{Z}}$ is defined as $(0, 0) : \underline{\mathbb{Z}}$.

[10] The function $\mathsf{asRational}^*$ is defined as a compostion:

$$\mathsf{asRational}^* \stackrel{def}{\equiv} \mathsf{asRational} \circ \mathsf{asInteger} : \underline{\mathbb{N}} \to \underline{\mathbb{Q}}$$

where each *casting* function is defined as follows:

$$\mathsf{asInteger} \stackrel{def}{\equiv} \lambda n.(n, 0) : \underline{\mathbb{N}} \to \underline{\mathbb{Z}}$$
$$\mathsf{asRational} \stackrel{def}{\equiv} \lambda z.(z, 1_{\mathbb{Z}}) : \underline{\mathbb{Z}} \to \underline{\mathbb{Q}}$$

References

Daisuke Bekki. 2014. Representing anaphora with dependent types. In *Logical Aspects of Computational Linguistics (8th international conference, LACL2014, Toulouse, France, June 2014 Proceedings), LNCS 8535*, pages 14–29. Springer, Heiderburg.

Daisuke Bekki. 2021. Proof-theoretic analysis of weak crossover. In *Logic and Engineering of Natural Language Semantics 18 (LENLS18)*, pages 75–88.

Daisuke Bekki and Koji Mineshima. 2017. *Context-passing and Underspecification in Dependent Type Semantics*, Studies of Linguistics and Philosophy, pages 11–41. Springer.

Daisuke Bekki and Miho Sato. 2015. Calculating projections via type checking. In *TYpe Theory and LExical Semantics (TYTLES), ESSLLI2015 workshop*.

Daisuke Bekki, Ribeka Tanaka, and Yuta Takahashi. 2021. Integrating deep neural network with dependent type semantics. In *the Symposium Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021)*, page p.37. Stockholm University, 2021, DiVA Portal for Digital Publications.

Hinari Daido and Daisuke Bekki. 2020. Development of an automated theorem prover for the fragment of dts. In *the 17th International Workshop on Logic and Engineering of Natural Language Semantics (LENLS17)*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 132–141. Association for Computational Linguistics.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Lifted rule injection for relation embeddings. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1389–1399. Association for Computational Linguistics.

Nissim Francez. 2014. *The Granularity of Meaning in Proof-Theoretic Semantics*, pages 96–106. Springer, Toulouse.

Nissim Francez and Roy Dyckhoff. 2010. Proof-theoretic semantics for a natural language fragment. *Linguistics and Philosophy*, 33(6):447–477.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 318–327. Association for Computational Linguistics.

Hans Kamp, J. van Genabith, and Uwe Reyle. 2011. *Discourse Representation Theory*, volume 15, pages 125–394. Springer, Doredrecht.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Per Martin-Löf. 1984. *Intuitionistic Type Theory*, volume 17. Italy: Bibliopolis, Naples.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *the 31st International Conference on Neural Information Processing Systems*. Curran Associates, Inc.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. 2019. Learning a SAT solver from single-bit supervision. In *ICLR (Poster)*.

Ryo Takahashi, Ran Tian, and Kentaro Inui. 2018. Interpretable and compositional relation learning by joint training with an autoencoder. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2148–2159. Association for Computational Linguistics.

Ribeka Tanaka. 2021. *Natural Language Quantification and Dependent Types*. Doctoral dissertation.

Gustav Šourek, Vojtěch Aschenbrenner, Filip Železný, Steven Schockaert, and Ondřej Kuželka. 2018. Lifted relational neural networks: efficient learning of latent relational structures. *J. Artif. Int. Res.*, 62(1):69–100.

Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. 2017. Premise selection for theorem proving by deep graph embedding. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2783–2793. Curran Associates Inc.

Yukiko Yana, Koji Mineshima, and Daisuke Bekki. 2019. Variable handling and compositionality: Comparing drt and dts. *Journal of Logic, Language and Information*, 28(2):261–285.

Yukiko Yana, Koji Mineshima, and Daisuke Bekki. 2021. The proviso problem from a proof-theoretic perspective. In *Logical Aspects of Computational Linguistics (LACL) 2021*, pages 159–176.

Center-Embedding and Constituency in the Brain and a New Characterization of Context-Free Languages

Daniel Mitropolsky, Adiba Ejaz, Mirah Shi, Mihalis Yannakakis, Christos H. Papadimitriou
Dpeartment of Computer Science
Columbia University
NewYork, NY 10027

Abstract

A computational system implemented exclusively through the spiking of neurons was recently shown capable of syntax, that is, of carrying out the dependency parsing of simple English sentences. We address two of the most important questions left open by that work: constituency (the identification of key parts of the sentence such as the verb phrase) and the processing of dependent sentences, especially center-embedded ones. We show that these two aspects of language can also be implemented by neurons and synapses in a way that is compatible with what is known, or widely believed, about the structure and function of the language organ[1]. Surprisingly, the way we implement center embedding points to a new characterization of context-free languages.

1 Introduction

How does the brain make language? Even though it is universally accepted that language is produced through the activity of the brain's molecules, neurons, and synapses, there has been extremely slow progress over the past decades in the quest for pinpointing the *neural basis of language,* that is, the precise biological structures and processes which result in the generation and comprehension of language — see Friederici (2017) for an excellent overview of a major direction in the theory of the language organ. In a recent advance in this direction, a *parser of English* was implemented (Mitropolsky et al., 2021) in the computational system known as the Assembly Calculus (AC) (Papadimitriou et al., 2020), a biologically plausible computational framework for implementing cognitive functions. The basic data structure of the AC is the *assembly of neurons*, a large set of neurons representing an idea, object, episode, word, etc. — a brief description of the AC and its brain-like execution environment is given in Section 2.1. The Parser is implemented through assembly operations, and thus ultimately by the actual spiking of stylized neurons. Its input is a sequence of words, and in response to it the Parser produces, as a detectable substructure of the neural circuit, a correct *dependency parse* of the sentence. For this to happen, neural representations of the words are assumed to be already in place in a brain area called LEX (for *lexicon*). For each new word, all neurons in the corresponding representation spike, and the representations contain enough grammatical information to cause a cascade of neural activity which results in the correct parsing of the sentence. It was shown through experiments in Mitropolsky et al. (2021) that several simple classes of sentences can be parsed this way. It is important to remember that the Parser in Mitropolsky et al. (2021) works exclusively through the spikes of biologically plausible neurons, and hence it can be seen as a proof of concept — a concrete hypothesis even — about the way syntactic analysis of language happens in the brain.

Several research directions were left open in Mitropolsky et al. (2021), and preeminent among them were these two: (a) how can the parser be extended so that dependent clauses, and especially center-embedded ones, are parsed correctly? and (b) how could the *constituency* representation of the sentence (the tree with "Sentence" and "Verb Phrase" as its internal nodes and the "Subject," "Verb," and "Object" as leaves) be produced — there is experimental evidence that the main constituents of a sentence are indeed created in Broca's area during sentence processing (e.g. Zaccarella and Friederici (2015)).

Parsing center-embedded sentences presents a serious conceptual difficulty: the parsing of the embedding sentence must be interrupted, and then be continued after the embedded sentence is parsed. This is the nature and essence of recursion. A mechanism for recovering the state of the parser at the

[1]Code available https://www.github.com/dmitropolsky/assemblies.

moment of the interruption seems thus necessary. It would be tempting to posit that recursion in the brain happens as it does in software, through the creation of a stack of activation records, but as we shall discuss this is not biologically plausible.

In this paper we pursue these two unresolved research goals and make significant progress on both. We start our description with constituents, which is the simpler narrative. The Parser in Mitropolsky et al. (2021) consists of the brain area LEX where the lexicon resides (believed to be a part of the medial temporal lobe, MTL), as well as several other areas labeled SUBJ, VERB, OBJ, DET etc. corresponding to syntactic roles, believed to be subareas of Wernicke's area; this is where the dependency parse is created. We show that this architecture can be augmented by new areas and fibers in such a way that the basic constituency tree of the sentence (the three-leaf tree that has inner nodes Sentence and Verb Phrase) can be also built "on the side" while the sentence is parsed. This entails two new brain areas denoted S for sentence and VP for verb phrase. The brain areas in our model are intended to correspond to two well known subareas of Broca's area, BA 43 and BA 44, respectively, where such activity is thought to take place (Friederici, 2017).

Coming now to the problem of dependent clauses, to handle clauses that are not embedded (say"if dogs are angry, they chase cats"), the Parser needs only one extra brain area beyond the areas needed by the original (Mitropolsky et al., 2021), namely an area labeled DS for "dependent segment."

To see the problem with embedded sentences, consider the variant "dogs, when they are angry, chase cats." The Parser would recognize "dogs" as the subject of the sentence and project the assembly for "dogs" to the SUBJ area and change the state as appropriate (inhibit a fiber). Upon encountering the first comma, the parsing of the sentence is curtailed, and the parsing of a new clause will be initiated. This is not a problem, since brain areas can contain many assemblies at the same time, and the two verbs, subjects, etc. will not interfere with each other. However, upon the second comma (for simplicity we assume that these clues are always available– in speech, they may, for instance, by indicated using prosodic or pausal cues), the Parser needs to continue the parsing of the outer clause (that is, the main sentence), and for this *it needs to restore the state at the moment the parsing was interrupted by the embedding sentence.* We could have the state saved in a pushdown store and retrieve it at this point, but we can see no biologically plausible implementation of either Parser state records or a pushdown store. As we shall point out, the Parser so far is a finite automaton, and thus it is not surprising that it has trouble handling embedded clauses.

The question is, *which biologically plausible departure from finite state machines, and from the model in Mitropolsky et al. (2021), can handle embedding?* We propose the following: The Parser stores the part of the utterance already parsed in some *working memory* (as in Awh et al. (2006), for example). When the embedded clause has been parsed, the Parser returns to the beginning of the outer sentence in the working memory and reprocesses it (up to the comma) to restore the state. We only need to execute the action sets of the words in the first part of the sentence, an activity that we we call *touching,* which is, in terms of elapsed time, an order of magnitude faster than parsing the first time. Once the first comma is seen, the embedded clause is skipped (without state changes) until the second comma, and parsing of the outer sentence is resumed from the recovered state. Any embedded clause can be handled with the touching maneuver — non-embedded dependent clauses are much simpler.

We show by processing several examples that several forms of dependent and embedded clauses can be correctly parsed (that is, a correct dependency graph of the whole sentence can be retrieved from the device after processing). If the embedded clause has its own embedded clause, the trick can be repeated.

Very surprisingly to us, this extension of the Parser, arrived at strictly through considerations of biological plausibility and ease of implementation, yields a rather unexpected theorem in formal language theory: if one defines an extension of nondeterministic finite-state automata to capture the operation of the enhanced Parser, with the extra capabilities of (a) marking the current input symbol and (b) reverting from the current input symbol to the previously marked symbol that is closest to the current one, then the class of languages accepted by these devices — call them *fallback automata* — coincides with the context-free languages!

2 Background

2.1 The Assemblies Model

How does the brain beget the mind? How do molecules, neurons, and synapses effect reasoning, planning, emotions, language? Despite tremendous progress in the two extremes of scale — cognitive science and neuroscience — we do not know how to bridge the scales. According to Nobel laureate Richard Axel (A, 2018), "We don't have a Logic for the transformation of neuronal activity to thought and action. I consider discerning (this) Logic as the foremost research direction in neuroscience." Notice the use of the word "Logic" whereby a distinguished experimentalist dreams of a formal system...

Recently, a computational framework called the Assembly Calculus (AC) was proposed whose precise intention is to be this Logic: to model the brain at the level of cognitive function through the control of a *dynamical system of spiking neurons.* This section describes the variant of the AC used in Mitropolsky et al. (2021) and the present paper.

We start with a mathematical model of the brain: a finite number a of brain *areas* $A, B, ...$ each containing n excitatory neurons. Every two neurons i, j in each area have a probability p of being connected by a synapse. Each synapse (i, j) has a nonnegative weight $w_{ij} > 0$, initially 1, say, which changes dynamically. For certain *unordered* pairs of areas (A, B), $A \neq B$, there is a *fiber* connecting them, a random directed *bipartite* graph connecting neurons in A to neurons in B and back, again with probability p independently for each possible synapse. Thus, the brain is a large directed graph with an nodes and random weighted edges.

Time is discrete (in the brain each time step is throught to be about 20 ms). The state of the dynamical system at time t has two components: the weights of the synapses w_{ij}^t, and the set of neurons that *fire* at time t. That is, for each neuron i we have a state variable $f_i^t \in \{0, 1\}$ denoting whether or not i fires at time t. The state transition from time t to time $t + 1$ is computed thus:

1. For each neuron i compute its *synaptic input* $SI_i^t = \sum_{(j,i)\in E, f_j^t=1} w_{ji}^t$, that is, the sum total of all weights from pre-synaptic neurons that fired at time t.

2. For each neuron $f_i^{t+1} = 1$ — that is, i fires at time $t+1$ — if i is among the k neurons *in its area* with the highest SI_i^t (breaking any ties arbitrarily).

3. For each synapse $(i, j) \in E$, $w_{ij}^{t+1} = w_{ij}^t(1 + f_i^t f_j^{t+1}\beta)$; that is, a synaptic weight increases by a factor of $1 + \beta$ if and only if the post-synaptic neuron fires at time $t + 1$ and the pre-synaptic neuron had fired at time t.

These are the equations of the dynamical system. The AC also has commands for the high-level *control* of the system. A fiber can be *inhibited* (that is, prevented from carrying synaptic input to other areas) and *disinhibited* (inhibition is canceled). Also, a set x of k neurons in an area can be made to fire by the command fire(x) (this is most relevant in connection to *assemblies,* defined next.

The *state* of the system contains the firing state of each neuron, the edge weights w_{ij}, and inhibition information.

Assemblies of neurons are a critical emergent property of the system. An *assembly* is a special set of k neurons, all in the same area, that are *densely interconnected* — that is, these k neurons have far more synapses between them than random, and these synapses have very high weights. This renders assemblies *stable representations* for representing in the brain objects, words, ideas, etc.

How do assemblies emerge? Suppose that at time 0, when nothing else fires, we execute fire(x) for a fixed subset of k neurons x in area A (these k neurons will always correspond to a previously created assembly[2]), and suppose that there is an adjacent area B (connected to A through a disinhibited fiber) where no neurons currently fire. Since assembly x in area A fires at times $0, 1, 2, \ldots$ (and ignoring all other areas), it will effect at times $1, 2, \ldots$ the firing of an evolving set of k neurons in B, call these sets $y^1, y^2, \ldots$. It is shown in Papadimitriou et al. (2020) that, with high probability (where the probability space is the random connectivity of the system), the sequence $\{y^t\}$ eventually converges to a stable assembly y in B, called *the projection of x in B*. The new assembly will be strongly interconnected, and also has strong connections from with x: If one of the two assemblies henceforth fires, the other will follow suit.

In Mitropolsky et al. (2021) this operation was generalized for the paper's purposes. Suppose an

[2]Initially, assemblies are created by projection from stimuli from the outside world coded in the sensory cortex.

assembly x in area A fires repeatedly and there are many areas downstream — not just a single area B as before — and these areas are connected by disinhibited fibers in a way that forms a *tree*. Then a sequence of project operations, denoted project*, creates a tree of assemblies, with strong synaptic connections between them. In (Mitropolsky et al., 2021), it is these synaptic connections between projected word assemblies that constitute the valid dependency parse tree created by the Parser.

2.2 The Parser

The basic architecture of the Parser in Mitropolsky et al. (2021) consists of several brain areas connected by fibers. LEX special, and contains representations of all words. Upon input of a new word (in brain reality, read or heard) the corresponding representation is excited, and upon firing it executes the *action set* of the word, commands which capture the grammatical role of the word. These commands open and close (disinhibit and inhibit) certain fibers. Then the project* operation is executed: all active assemblies fire. By this scheme, it was shown in Mitropolsky et al. (2021) that many categories of English sentences (and Russian as well) can be parsed correctly (the correctness can be verified because the running of the parser on an input sentence leaves a retrievable graph structure within and between brain areas, which constitute a correct dependency graph of the sentence). This completes the description of the Parser's architecture. We provide a figure with an example sentence parsed, and refer the interested reader to Mitropolsky et al. (2021).

2.3 Neuroscience

We believe that our Parser represents a reasonable hypothesis for parsing in the brain. The neurobiological underpinnings of Assembly Calculus, with which our model is built, and the original Parser of Mitropolsky et al. (2021), which ours extends, are presented more fully in that paper. Briefly, Assembly Calculus is based on established tenets of neuron biology, including that neurons fire when they receive sufficient excitatory input from other neurons, the atomic nature of neuron firing, and a simplified narrative of synaptic Hebbian plasticity (see for instance Kandel et al. (1991), Chapters 7, 8, and 67). Assemblies, in turn, are an increasingly popular hypothesis for the main unit of higher-level cognition in modern neuroscience– first hypothesized decades ago by Hebb, they have been identified experimentally (Harris, 2005) displacing previously dominant theories of information encoding in the brain, see e.g. Eichenbaum (2018). With what regards the higher-level Parser architecture, language processing appears to start with access to a *lexicon*, a look-up table of word representations thought to reside in the left medial temporal lobe (MTL), motivating the inclusion of an area LEX. After word look-up, activity in the STG is thought to signify the identification of syntactic roles. Overall, the Parser generates a hierarchical dependancy-based structure that from a sentence that is processed incrementally, which we believe models something like the creation of hierarchical structures in Broca's areas in experiments such as Ding et al. (2016).

3 Constituency

Constituency parsing revolves around the idea that words may lump into a single assembly. The noun subject of a sentence — along with its dependent adjective(s) and determinant(s) if any — form the "Subject," while the verb and object form the "Verb Phrase." At the coarsest level, the "Subject" and "Verb Phrase" then form the "Sentence." We modify the underlying framework of the Parser to assign this syntactic structure to a sentence.

We add two new brain areas that hold the "Verb Phrase" and "Sentence," VP and S respectively, as well as fibers between VERB and VP, OBJ and VP, SUBJ and S, and VP and S. These fibers remain disinhibited throughout parsing so that the constituency tree is built concurrently as we parse the sentence. That is, when the verb is processed, a corresponding assembly is formed in VP, and upon encountering the object, the assemblies in VERB, OBJ, and VP fire together to form in VP what is now the merge of assemblies representing the verb and object. Parallel to this process, assemblies fire along the SUBJ to S and VP to S fibers so that the final assembly in S represents the joining of the "Subject" and "Verb Phrase."

Experiments. We extend the implementation of the Parser in Python to incorporate these new abilities. Additionally, we tailor the readout algorithm (which in Mitropolsky et al. (2021) recovers the dependency tree) to output the desired tree rooted in S. To verify that the Parser produces the correct constituency tree, we provide a test set of 40 sentences constructed from 20 syntactic patterns that include variations in word orderings; additions

of determinants, adjectives, adverbs, and prepositional phrases; and both transitive and intransitive verbs. The Parser generates the correct constituency trees on all of our given test cases. Importantly, the constituency Parser can handle any sentence structure that the original Parser can handle.

As in the original parser, we execute 20 firing epochs of project* to allow the dynamical system to stabilise. The multiple concurrent projections into S and VP cause a slowdown by a factor of 2.5 relative to the dependency parser, resulting in a frequency of 0.5-1.3 seconds/word.

4 Embedded Sentences

To handle embedded sentences, the Parser requires a new area DS (for dependent segment) to handle dependent and embedded clauses. Additionally, we need modifications that recover the state *before* an embedded clause (which we recover when we finish parsing an embedded clause, i.e., upon a "right comma").

In particular, we assume that there is a *working memory area* which holds the words which have already been processed, in sequence. We assume for simplicity that the parser can always recognize the beginning of a dependent sentence — that this is always possible through simple clues such as a comma (in text), prosodic cues (in speech), and/or a *complementizing* pronoun or preposition, such as "who", "if", or "that". We also assume that there is a unique sentence or clause at each depth, though with minor modifications we can handle the more general case.

To handle center-embedded clauses, the parser utilizes a limited working memory: it must remember the sentence up to the point when an embedded clause begins in order to efficiently reprocess these words after parsing the embedded clause. More concretely, when a center embedding is detected, the Parser "cleans the slate"; that is, the Parser state (the fiber and area states) is reinitialized in order to parse the new embedded clause, which is parsed normally until its end is detected. At this point the Parser has to restore its last state when it was parsing the outer clause. To do so, it reinitializes the state again and reprocesses the sentence from the beginning of the outer clause up to the interruption. However, this re-processing is special: the parser only "touches" the words, by which we mean that for each word we apply its action sets (inhibiting or disinhibiting areas/fibers), fire the entire system exactly once to reactivate existing assemblies that were formed in the initial parse of the fragment (when we initially parsed this part of the sentence, we used project*, which fires the entire system 20+ times in order for assemblies to form and converge). In this way, touching is by an order of magnitude faster than parsing on initial input of the word — both in our simulation and in the hypothesized language organ — since it is only recovering the *preexisting* structure. Note that when the first comma is scanned, the parsing of the outer sentenced is resumed from the second comma, skipping the (already parsed) embedded clause.

The remaining difficulty is in linking the outer and inner clauses. The last word of the outer clause before the inner sentence functions as a "signature" for the outer clause. This signature may reside in SUBJ or OBJ, for instance. After parsing the fragment of the outer clause pre-interruption, we project from the relevant signature area to DS. Then, on the verb of the inner clause, we project from both LEX and DS into VERB simultaneously. This way, we can later recover the root verb of the inner clause via the signature assembly of the outer sentence (through projection into DS and subsequently VERB). to DS, and then to VERB.

Experiments. We extend the implementation of the dependency Parser in Python to incorporate touching, linkage, and recursion within the main Parser loop. In principle, the developed Parser can handle arbitrarily many levels of embedding. We test on 20 sentences sampled from 5 different embedding structures, with depths 0, 1, and 2 (informed by the lack of three or greater depth sentences in ordinary language). Our test cases feature center-embedding, edge-embedding, mixed embedding, and relative clauses modifying the subject or the object. We assume there is a unique clause at each depth. The Parser generates the expected dependency tree on all of our given test cases.

Again, we execute 20 firing epochs of project*. The modified parser preserves the speed of the original, with a negligible increase in time for linkage projections and touching. Despite several challenges created by the added complexity of sentence embedding, the linkages between projected assemblies correspond to the correct dependency graph in all sentences.

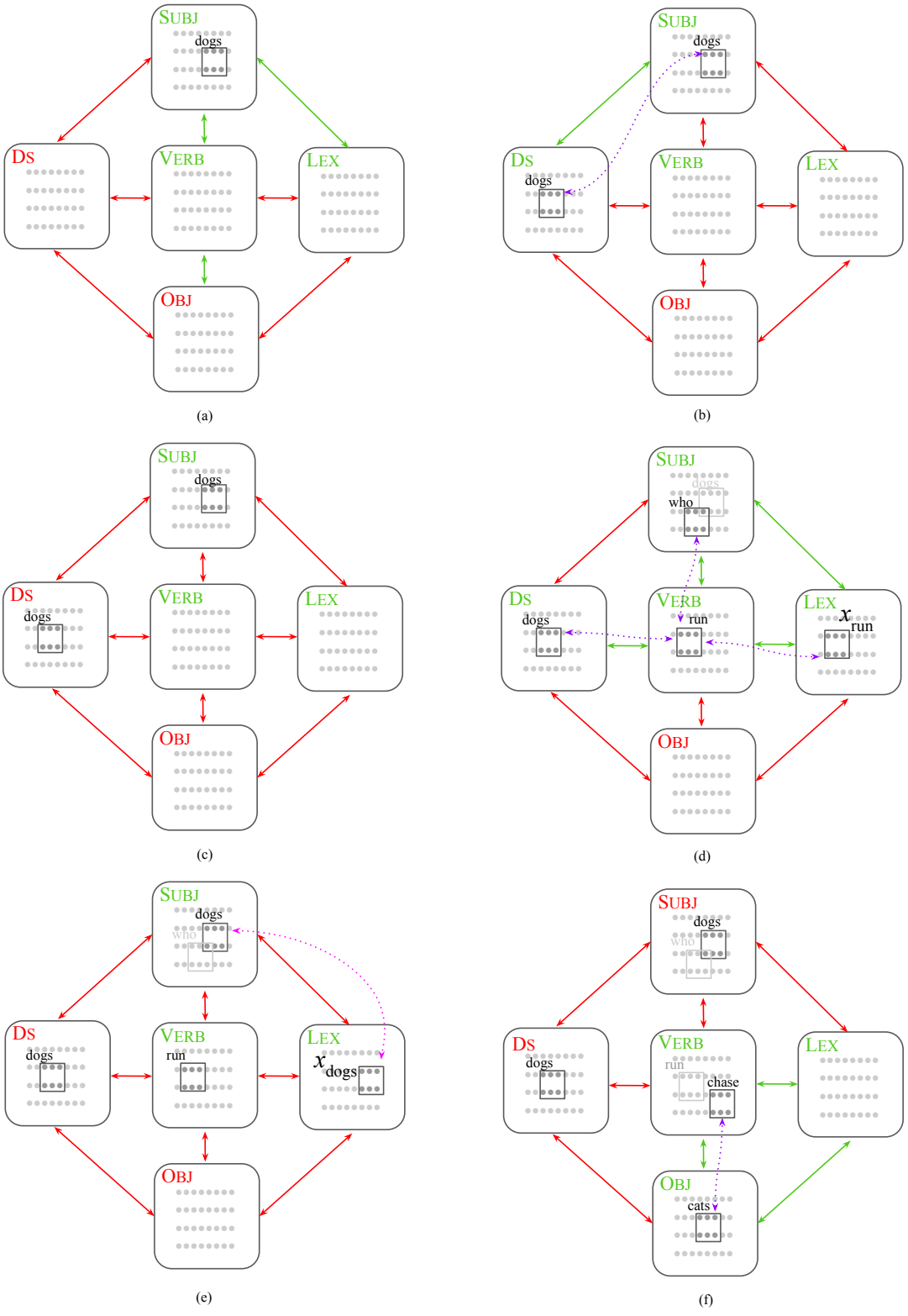


Figure 1: Snapshots of the Parser while processing the center-embedded sentence "*dogs, when they run, chase cats*." Green arrows represent fibers that have been disinhibited between each stage; red arrows represent inhibited fibers. Purple dotted arrows indicate assemblies that have been connected through project*. Pink dotted arrows indicate assemblies touched through activating their corresponding assemblies in LEX. (a) Outer sentence, up until beginning of inner sentence, parsed; (b) Assemblies in the signature and DS areas linked; (c) Fiber and area states reset in anticipation of a new sentence; (d) Inner sentence parsed and linked to outer sentence through the DS area; (e) Outer sentence revisited: each word prior to dependent sentence is touched, restoring Parser state of stage (a); (f) Remainder of outer sentence parsed.

Algorithm 1: Enhanced Parser, main loop

input : a sentence s, depth d
output : representation of dependency parse of s, rooted in VERB
Function *parse* (s, $d \leftarrow 0$) **:**
  foreach word w in s **do**
    if $(d+1)$-depth clause begins after w **then**
      disinhibit(DS) ;
      disinhibit(DS, AREA(w)) ;
      *project** ;
      inhibit(DS, AREA(w)) ;
      inhibit(DS) ;
    else if w begins (d+1) depth clause **then**
      $d \leftarrow d+1$;
      clear the slate ;
    else if w ends embedded sentence **then**
      $d \leftarrow d-1$;
      foreach word y before w **do**
        activate y in LEX ;
        fire DISINHIBITED AREAS ;
    if $d > 0$, AREA(w) = VERB **then**
      disinhibit(DS) ;
      disinhibit(DS, VERB) ;
    execute w actions and *project** ;
    inhibit(DS) ;
    inhibit(DS, VERB) ;

5 A Little Formal Language Theory

The Parser, without the embedded sentence module, is a *finite state device.* The reason is that the Parser's state is an acyclic subgraph of the graph of brain area and fibers (excluding LEX). It is no wonder then that sentence embedding (intuitively, a feature involving recursion and therefore moving us out of the realm of regular languages) requires an extension, and among many other options that seemed to us less biologically plausible we chose to revisit the outer sentence and restore the state of the Parser. It is now natural to ask: Can the operation of the Parser when handling embedded sentences be seen as a more powerful genre of automaton? This is the motivation for the results in this section.

Definition: A *fallback automaton (FBA)* is a tuple $A = (\Sigma, K, I, F, \Delta)$, where (as in a nondeterministic finite-state automaton) Σ is a nonempty finite set of symbols, K is a set of states, I is the set of initial states, F is the set of accepting states, and Δ is the transition relation.

Define the *type set* $T = \{f, s\} \cup K$. Whereas in nondeterministic FSAs $\Delta \subseteq (\Sigma \times K) \times K$, the transition relation of FBAs is more complex.

$$\Delta \subseteq ((\Sigma \times T \times K) \times (K \times \{s, \checkmark, \leftarrow\})$$

The automaton is *nondeterministic,* reflecting the nondeterministic nature of parsing in general, due to ambiguity and polysemy. The transition relation can be understood thus: symbols on the tape are marked by a type, either f (for *fresh*), s (for *seen*), or by a state $q \in K$. Initially, all symbols are fresh and hence marked f. When a symbol is marked s or by a state of K, it is not being scanned for the first time — the automaton may scan a symbol multiple times. The first time a symbol is scanned, its type is changed from f to s, or, if the rule outputs $\checkmark$, it is set to the current state q. Subsequently, after a symbol marked by a state is scanned, the type reverts to s.

To formalize the operation of the automaton, at each step, there is a *tape* $x \in (\Sigma \times T)^*$; we denote its ith symbol as $x_i = (\sigma_i, t_i)$, where $\sigma_i \in \Sigma$ and $t \in T$. The automaton is in a state $q \in K$ and the ith symbol x_i is scanned. The overall configuration is thus (x, q, i). In the initial configuration, the type of every symbol is f, $q \in I$ and $i = 1$.

If the next configuration after (x, q, i) is (y, r, j) then:

- y is identical to x except that the type of the ith symbol may have changed: f must become s or the machine's current state q, s stays s, and $q \in K$ always become s.
- unless this is a fallback step, $j = i + 1$.
- *Fallback.* When scanning a fresh symbol, the automaton may return to a position $j < i$, where j is the largest position j that was marked — that is, $t_j \in K$. Notice that in the next step, $t_j = s$— the symbol is unmarked.
- When scanning a symbol with type $q \in K$, i.e., a symbol fallen back to, the transition can map to only s or $\leftarrow$– that is, it can fallback again, but it cannot mark the symbol again for fallback.
- In all cases, the corresponding pair $((x_i, q)(r, \sigma))$ must be in Δ. $\sigma = s$ means that the i-th symbol's type is changed to s, $\sigma = \checkmark$ means it is changed to the state q, and $\sigma = \leftarrow$ means the step is a fallback.

This concludes the definition of the FBA. We say that a string x in Σ^* is accepted by FBA A if there is a sequence of legal steps from a configuration with state in I and tape x with all symbols fresh to a configuration in which the state is in F.

Note that when the FBA falls back to a previous tape location $j < i$, it then passes again over the seen symbols (marked s) between x_j and x_i, and may do meaningful computation upon this revisiting. Furthermore, an FBA can fallback repeatedly, immediately after a fallback move. Our Parser more closely corresponds to an FBA without this abilities. Hence it is interesting to define a weaker model without this ability:

A *weak-FBA* is an FBA with the additional requirements that 1) for any symbol marked s, transitions cannot change the state (that is, for all $\alpha \in \Sigma$, $q \in K$ and $(\alpha, s, q) \times (q', s) \in \Delta$, we require $q' = q$, in effect, skipping over all s symbols) and 2) for any symbol marked with $q \in K$, the transition must output (q', s) (that is, there are no repeated fallbacks).

It may seem that FBAs can do more than weak-FBAs. Consider the following language over $\Sigma = \{0, 1, \alpha, \beta\}$, $L = \{\alpha^n x \beta^n \ : \ x \in \{0, 1\}^*\}$. By marking every α and falling back on every β, an FBA can read through x at least n times, a linear dependence. However, a weak-FBA can read each symbol in the tape exactly 1 or 2 times.

Perhaps surprisingly, it turns out that the ability to do additional computation on revisited symbols offers no additional power. More importantly, it turns out that both models recognize a fundamental class of formal language theory. Denote the class of languages accepted by FBAs as **FBA**, that by weak-FBAs as **weak-FBA**, and the class of context-free languages as **CFL**. We can prove the following:

Theorem: weak-FBA = FBA = CFL

Proof outline: To show that **weak-FBA** $\supseteq$ **CFL**, we recall the classic theorem of Chomsky and Schützenberger (Chomsky and Schützenberger, 1963) stating that any context-free language L can be written as $L = R \cap h(D_k)$, where R is a regular language, D_k denotes the Dyck language of balanced parentheses of k kinds, and h is a homomorphism, mapping any symbol in the alphabet of D_k to a string in another alphabet. Let us take a context-free language in this form. Note that FBAs are ideal for accepting $h(D_k)$. The machine uses non-determinism to guess which symbol is represented by the next sequence of characters. When it guesses that it will see the image of a left parenthesis, say '{', it checks each symbol of $h(\{)$ (and rejects if the sequence of symbols is not $h(\{)$), and marks the final character (with the state $q_{\{}$)). For a right parenthesis, after checking for the sequence $h(\})$, it falls back and checks that the symbol fallen back to is marked $q_{\{}$). Intersection with the regular language R is done by simultaneously maintaining the state of the automaton accepting R in a separate component of the FBA's state (in fact, one can show that the languages accepted by FBAs are closed under intersection with regular languages).

To show that **FBA** $\subseteq$ **CFL**, we emulate the execution of a FBA with a *push-down automaton* (that is, a non-deterministic finite state automaton with the additional computational power of one stack). By the classic result proved independently by Chomsky (1962); Schützenberger (1963); Evey (1963), the languages recognized by push-down automata are exactly **CFL**. The emulation uses the following trick: the stack is composed of *vectors* of states of length $|K|$. These vectors keep track of the execution of the FBA on every possible state on the sequence of symbols between consecutive pairs of marked symbols, and between the most recent marked symbol and the head. Whenever the FBA falls back and is in state q where q corresponds to the i-th coordinate in the stack vectors, the emulation pops the top vector on the stack and jumps

to the state in the i-th coordinate, as this would be the resulting state *had the machine re-read the seen symbols starting in state* q. The full proof is technical, and is given in the Appendix.

6 Discussion

The Parser in Mitropolsky et al. (2021) can be seen as a concrete hypothesis about the nature, structure, and operation of the language organ. Here we elaborate on this hypothesis: First, rough constituency parsing (the creation of the two highest layers of the syntactic — or constituency — tree of the sentence) can be carried out simultaneously with the main dependency parsing. Second, dependent sentences can also be parsed. For center-embedded sentences, a significant extension of the Parser is required: A working memory area stores the whole utterance, and the parser returns to the beginning of the utterance to recover the state of the Parser after processing the outer sentence, and and then skips the embedded sentence and continues parsing the outer one.

Even though this maneuver was motivated by biological realism and programming necessity, we showed that it transforms the device from one that handles only regular languages to one capable of accepting *all context free languages* — and *just* these. We find this quite surprising, and possibly significant for the history of linguistic theory: Seven decades ago, Noam Chomsky sought to formalize human language and in the mid 1950s introduced CFLs expressly for this purpose. In the following two decades, this choice was criticised as too generous (not all features of CFLs are needed) and also as too restrictive (some aspects of natural language are not covered by CFLs). Arguably, this criticism was accepted by Chomsky's school of thought: Grammar remained important, of course, but context-free rules besides $S \to NPVP$ (right-hand side unordered) were not used often. Much of NLP centered around the dependency formulation of syntax. Two-thirds of a century later, computer scientists speculating about syntax in the brain came up with a computational trick in order to handle center recursion. And this maneuver, when formalized properly, leads to a device that can recognize all CFLs.

Besides speculating on the meaning of this theoretical result, our work suggests a major open problem: If we assume that syntax in the brain is handled in a way similar to the one suggested by the Parser[3], and all humans are born with a system of brain areas and fibers in their left hemisphere capable of such operation, *how do babies learn to use this device?* How are words learned and projected, presumably from the hippocampus, where they are associated with world objects and episodes, to the LEX in the medial temporal lobe? And how is each of them attached to the correct system of interneurons that are capable of changing the inhibited/disinhibited status of fibers and possibly of brain areas?

7 Acknowledgments

The authors thank Tal Malkin for helpful discussions about the proof of the theorem, and the anonymous NALOMA reviewers for helpful feedback.

References

Q & A. 2018. Richard Axel. *Neuron*, 99:1110–1112.

E. Awh, E.K. Vogel, and S.-H. Oh. 2006. Interactions between attention and working memory. *Neuroscience*, 139(1):201–208.

N. Chomsky. 1962. *Context-free Grammars and Pushdown Storage*.

N. Chomsky and M.P. Schützenberger. 1963. The algebraic theory of context-free languages*. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, volume 35 of *Studies in Logic and the Foundations of Mathematics*, pages 118–161. Elsevier.

Nai Ding, Lucia Melloni, Hang Zhang, Xing Tian, and David Poeppel. 2016. Cortical tracking of hierarchical linguistic structures in connected speech. *Nature neuroscience*, 19(1):158.

Howard Eichenbaum. 2018. Barlow versus hebb: When is it time to abandon the notion of feature detectors and adopt the cell assembly as the unit of cognition? *Neuroscience letters*, 680:88–93.

R. James Evey. 1963. Application of pushdown-store machines. In *Proceedings of the November 12-14, 1963, Fall Joint Computer Conference*, AFIPS '63 (Fall), page 215–227, New York, NY, USA. Association for Computing Machinery.

Angela D Friederici. 2017. *Language in our brain: The origins of a uniquely human capacity*. MIT Press.

Kenneth D Harris. 2005. Neural signatures of cell assembly organization. *Nature Reviews Neuroscience*, 6(5):399.

[3] A far-fetched assumption, of course, but one that is somewhat justified by the fact that there is no competing theory that we are aware of.

Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell, editors. 1991. *Principles of Neural Science*, fifth edition. Elsevier, New York.

Daniel Mitropolsky, Michael J. Collins, and Christos H. Papadimitriou. 2021. A Biologically Plausible Parser. In *Transactions of the Association for Computational Linguistics*. ArXiv: 2108.02189.

Christos H. Papadimitriou, Santosh S. Vempala, Daniel Mitropolsky, Michael Collins, and Wolfgang Maass. 2020. Brain computation by assemblies of neurons. *Proceedings of the National Academy of Sciences*, 117(25):14464–14472.

M.P. Schützenberger. 1963. On context-free languages and push-down automata. *Information and Control*, 6(3):246–264.

Emiliano Zaccarella and Angela D. Friederici. 2015. Merge in the human brain: A sub-region based functional investigation in the left pars opercularis. *Frontiers in Psychology*, 6:1818.

8 Appendix

Here we give the full proof of our main theoretical result:

Theorem: weak-FBA=FBA=CFL.

Note: We say a symbol on the tape is "marked' whenever its type is some $q \in K$.

Lemma: For any strong-FBA, there exists a strong-FBA recognizing the same language that is deterministic whenever the input symbol is marked s (that is, for every state and symbol pair q, α, there is at most one rule $((\alpha, s, q), (q', s)) \in \Delta$.

Proof: this is shown using essentially the same reduction from non-deterministic to deterministic finite state automata (FSA), since when at a seen symbol, the FBA cannot mark or fallback and is hence in a FSA-like regime. Concretely, if K is the original state set, the state set of the new FBA is 2^K. Δ contains the same rules when the input is in state f or is marked (we represent states of the original FBA with the singleton of that state in 2^K)– whenever the input is in state s, it transitions to the ϵ-closure of the subset represented by the state (i.e., $(\alpha, s, S) \rightarrow (S', s)$ iff S' is the set of all states that can be reached from a state of S on symbol α, before or after any epsilon transitions). Additionally, whenever the FBA reads symbol marked with anything other than s, or reaches the end of the tape, if the current state-set S is a non-singleton, it non-deterministically transitions to the singleton of any $q \in S$ (thereby returning to a "regular" state of FBA, and moving all the non-determinism away from s symbols). The new FBA is deterministic on s-inputs and recognizes the same language: any transition through a sequence of s-states corresponds to a specific non-deterministic transition to a single state from the final state-set at the first non$-s$ symbol. □

The main technical result is showing that a push-down automaton (PDA) can simulate a **strong-FBA**, i.e., that **strong-FBA** $\subseteq$ **CFL**.

Proof of theorem: By the lemma, without loss of generality we can assume that the strong-FBA is deterministic when the input has type s. Let $K = \{q_1, \ldots, q_t\}$ be the state set of the strong-FBA. The PDA will have state set K, and stack alphabet $\Sigma \times K \times K^{|K|}$, that is, tuples of a symbol, a state, and *state vectors*. We define the "s-transition on α" of a state q to mean the (deterministic) FBA transition rule with left-side (α, s, q).

The PDA simulates the execution of the strong-FBA. When in state q' and on fresh tape symbol α, the PDA will:

1) Non-deterministically select a rule with left side (α, f, q') on the left-hand side. Let (q, σ) be the right-hand side. That is, $\sigma \in \{s, \checkmark, \leftarrow\}$.

2) Update the state to q.

3) If the stack is non-empty, pop the top element $(\beta, p, (r_1, \ldots, r_t))$ from the stack. For each vector coordinate $i \in [t]$, apply the s-transition on α to each r_i– push the updated tuple $(\beta, p, (r'_1, \ldots, r'_t))$ back to the stack.

4) if $\sigma = \checkmark$, push $(\alpha, q, (q_1, \ldots, q_t))$ to the stack.

5) if $\sigma = \leftarrow$, the corresponding PDA transition does nothing else, but enters the following loop:

5.1) Pop the top element $(\beta, p, (r_1, \ldots, r_t))$ of the stack and sample a rule of the FBA that transitions on the marked symbol, that is, a rule of the form $(\beta, p, q) \rightarrow (q', \sigma)$. Note that $\sigma \in \{s, \leftarrow\}$. If the stack is non-empty, pop the next element, $(\gamma, w, (u_1, \ldots, u_t))$ and "apply" $(r_1, \ldots, r_t)$ to the state vector– that is, for each $i \in [t]$, if $u_i = q_j$, replace u_i with $u'_i := r_j$. Push the resulting pair $(\gamma, w, (u'_1, \ldots, u'_t))$ back onto the stack.

5.2) if $\sigma = \leftarrow$ (which we call a "fallback-again" rule of the FBA), the PDA updates the state to q', and returns to the beginning of 5.1). If $\sigma = s$ (which we call a non-"fallback-again" rule), the PDA updates the state to r_i (that is, the i-th coordinate in the stack-vector from 5.1), where i is the index of q', i.e. $q' = q_i$. Because the FBA

can have “fallback-again" rules, the PDA can repeat 5.1 multiple times, and stops when it selects a non-“fallback-again" rule (or rejects).

For any execution of the PDA, it will have $|x| + F + M$ transitions, or steps, where x is the input, F is the number of times the PDA *repeated* step 5.1 (i.e. executed a “fallback-again" rule of 5.2), and M is the number of times it ends the loop of 5), i.e. selects a non-“fallback-again" rule. Each step i will correspond to a “strong"-step of an equivalent FBA execution, which is a transition where the input symbol has type not equal to s (in other words, a step where the FBA either processes a fresh symbol, or falls back to a marked symbol).

Claim: There is a one-to-one correspondence between executions of the FBA on x with executions of the PDA on x, such that, for each such pair, the PDA simulation maintains the following invariant at each step i with respect to the FBA:

a) the PDA state *after* the i-th step is equal to the FBA state immediately before the $i+1$-th strong step (or the final state, if $i = |x|$), and the position of the PDA head is equal to the position of the last f-symbol seen by the FBA.

b) the PDA stack has l elements, where l is the number of marked symbols after the i-th strong-step of the FBA, and

c) the k-th stack element consists of the k-th marked symbol and type in the FBA execution, and a vector that contains for each state q_j, the execution of s-transitions starting from the symbol immediately after the k-th marked symbol, up to and including either the $(k+1)$-th marked symbol, or the head, whichever comes first.

Note that if the claim is true, by part a) of the invariant, the final state of the PDA is an accept state iff the equivalent FBA execution accepts x, so the theorem is proved.

We prove the claim recursively. It is trivially true at the beginning. Now, assume that after $i-1$ steps of execution, the FBA corresponds to an execution of $i-1$ strong-steps of the PDA, and that the invariant is true.

At step i, we consider all possible strong-steps of the FBA.

First, for any rule of the type $((\alpha, f, q'), (q, s)) \in \Delta$: by a) of the invariant the FBA and PDA must both be scanning a new symbol α and are in state q'. The PDA simulation can sample this rule by step 1), which results in updating the state q' to q, satisfying a). Since no marked symbols were added or removed by this kind of FBA transition, b) is trivially satisfied. Finally, 2) updates the top state vector on the stack with the s-transitions on α– since it previously represented the s-transitions from the most recent marked symbol to the previous tape symbol, it now represents an execution up to and including the current symbol, that is, c) is maintained.

For any rule of the type $((\alpha, f, q'), (q, m)) \in \Delta$: similarly, the PDA simulation can select this rule in 1), updating the state to q', giving a). The number of marked symbols in the FBA execution changes if this rule were applied, so invariants b) and c) must be checked. By step 3), we push $(\alpha, q, (q_1, \ldots, q_t))$ to the stack, immediately satisfying b). Note that the penultimate element of the stack is now “frozen", showing an execution of s-transitions from the previous marked-symbol up to and including α, the new marked symbol. The vector of the top element of the stack, $(q_1, \ldots, q_t)$, trivially represents an execution of every state from the new marked symbol to the head (since it is empty). Hence c) is satisfied.

For any rule of the type $((\alpha, f, q'), (q, \leftarrow)) \in \Delta$: again both the PDA and FBA are at a fresh tape symbol, but after this step, the FBA head will be at the previously marked symbol. By step 1) the PDA can sample this rule, updating the state to q, which satisfies a). Note that b) and c) are trivially satisfied.

Next for any strong rule of the type $((\beta, p, q'), (q, \leftarrow))$, the FBA head must be at a marked symbol, and immediately before the next strong step, it will be at the previous marked symbol in state q. Indeed, the FBA can sample this rule in 5.1, updates state to q satisfying a), pops the top stack vector satisfying b), and as for c), the PDA “applies" the state vector of the popped tuple, $(r_1, \ldots, r_t)$, to the *next* state vector on the stack, $(\gamma, w, (u_1, \ldots, u_t))$. If u_i is the s-transition of q_i from the symbol after γ up to and including β (this is guaranteed by the invariant) and $u_i = q_j$, then r_j is exactly the s-transition of q_i from the symbol after β up to and including the head, ensuring c).

Finally if the rule is of the type $((\beta, p, q'), (q, s)$, the FBA head must be at a marked symbol, and immediately before the next strong step it *passes through every s-symbol between the marked symbol and the fresh symbol*. The corresponding PDA transition satisfies b) and c) as in the previous case (a marked symbol is removed and the other vectors in

the stack are updated), but this time, we also update the current state based on the popped vector; since, by b), it contained the s-transition of each state from the marked symbol to the head, it correctly yields the state of the FBA before the next fresh symbol.

A Philosophically-Informed Contribution to the Generalization Problem of Neural Natural Language Inference: Shallow Heuristics, Bias, and the Varieties of Inference

Reto Gubelmann
University of St.Gallen
Rosenbergstrasse 30
9000 St.Gallen
{reto.gubelmann,

Christina Niklaus
University of St.Gallen
Rosenbergstrasse 30
9000 St.Gallen
christina.niklaus,

Siegfried Handschuh
University of St.Gallen
Rosenbergstrasse 30
9000 St.Gallen
siegfried.handschuh}@unisg.ch

Abstract

Transformer-based pre-trained language models (PLMs) currently dominate the field of Natural Language Inference (NLI). It is also becoming increasingly clear that these models might not be learning the actual underlying task, namely NLI, during training. Rather, they learn what is often called bias, or shallow heuristics, leading to the problem of generalization. In this article, building on the philosophy of logics, we discuss the central concepts in which this problem is couched, we survey the proposed solutions, including those based on natural logic, and we propose or own dataset based on syllogisms to contribute to addressing the problem.

1 Introduction

Current natural language inference (NLI) is typically conceived as a three-way classification problem. With samples such as (1), consisting of a premise (P) and a hypothesis (H), the PLMs are tasked to categorize their relationship as either one of *contradiction* (P and H cannot both be true), of *entailment* (If P is true, then H must be true as well), or as being *neutral* (neither of the two).

(1) (P) The streets are wet. (H) It has rained.

As we will show below (see section 2), transformer-based pre-trained language models (PLMs) are currently the standard to approach this task of NLI. What is emerging as neural NLI's most pressing problem is the fact that these neural PLMs might almost outperform the crowdworker-based human baseline for the dataset on which they were fine-tuned, but perform worse than random at out-of-dataset-samples. We call this, following standard usage, the problem of generalization.

In this article, we focus on this problem of generalization, contributing a perspective that is informed by the philosophy of logic. More specifically, our article makes three contributions. First, after developing a conceptual background from the philosophy of logic, we give a comprehensive and systematic view on the extent of the problem of generalization in NLI, and we survey the different extant proposals to address this problem. Second, we propose and make publicly available a new fine-tune and challenge dataset that is based on syllogistic. Third, we evaluate the performance of both neural NLI models (including models fine-tuned on our syllogistic dataset) and a symbolic approach on this dataset. In the remainder of this section, we introduce the philosophical concept of inference.

The first and central distinction to be drawn regarding the concept of valid inference is the one between deductively valid inferences and defeasibly valid inferences (see Koons 2021 for an introduction to the distinction and to the concept of defeasible reasoning).[1] An inference is deductively valid if it is not possible that the premises are true while the conclusion is false (for the concept of necessity involved here, see Plantinga 1974, 1ff.). With defeasible inference, this condition does not hold: for such inferences, it is possible that the premises are true, while the conclusion is wrong. Example (1) is a case of defeasible inference: the streets could be wet, but this could have other causes than rain.

Within the domain of deductively valid inferences, it is common to distinguish inferences that are deductively valid due to the form of the propositions that constitute the inference, and others that are valid due to the content of these propositions (see Quine 1980 [1951] for a critical discussion of the distinction). Example (2) is a case of a formally deductively valid inference: It does not matter what

[1] For an early discussion of the distinction between deductively valid inferences, especially as opposed to conventional and conversational implicatures, see Zaenen et al. (2005).

you plug in for "Germans", "childcare workers", and "fingerprint collectors", you will always get a deductively valid inference (note that the truth of either premise or hypothesis is not required for an inference to be deductively valid. The concept of validity applies only to the truth-functional relationship between premise and hypothesis. A deductively valid inference with true premises is called a *sound* inference).

(2) (P) All Germans are childcare workers and all childcare workers are fingerprint collectors. (H) All Germans are fingerprint collectors.

In contrast, example (3) is deductively valid because of the content, the meaning of "bachelor" and "unmarried": replacing these concepts with others will likely result in an invalid inference.

(3) (P) Peter's marital status is that of a bachelor. (H) Peter is unmarried.

Formally valid inferences can further be classified according to the formal logical apparatus that is needed to prove its validity: propositional calculi, propositional calculi of different orders, and modal calculi are the most common options (see Smullyan 1968 for introductions to propositional and first-order logic, Garson 2006 for modal logic). Briefly, natural logic can be understood as the program to successively cover all of these areas without having to resort to translation into a formal language (for details, see section 2.2 and appendix, section C).

There are different proposals to systematize the domain of defeasible inferences. Currently, a prominent one is that defeasible inferences are inferences to the best explanation, that is, abductive inferences (for an excellent introduction to the concept, see Lipton 2004). Example (1) evinces the plausibility of this perspective: It is reasonable to conceive the hypothesis there as an explanation for the premise. The inference is defeasible because there could emerge a better explanation for the premise (in example (1), this could be the information that a street cleaning crew just passed through the street). An alternative conception is that such inferences are inductive in nature, that is, based on a number of previous observations of similar situations. Ever since Hume, it has been painfully clear that, without further metaphysical argument, such inductive inferences are not deductively valid. Figure 1 gives an overview on these kinds of valid inference.

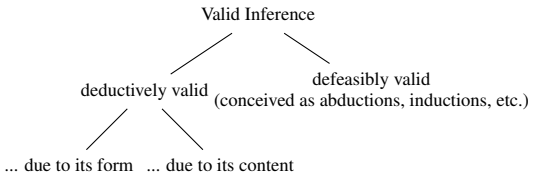


Figure 1: Kinds of valid inferences.

We will see that current practice oscillates between deductively and defeasibly valid inferences. Our own dataset focuses on the area of formal deductive validity.

To conclude our terminological survey, we mention that we will propose to distinguish between bias and shallow heuristics in a way suggested by Blodgett et al. (2020): We use bias as preconceptions that are potentially harmful, intrinsically normative, and always couched in a wider worldview. In contrast, a shallow heuristic is a local tactic to succeed at a given task without any understanding or mastery of the actual task that is explicitly not part of an intrinsically normative worldview.

2 State of the Art in Neural NLI

2.1 Neural NLI: Models & Datasets

In this section, we introduce the state of the art in neural NLI. As the focus of critical attention increasingly shifts to the datasets, we consider them in detail as well.

The Models Transformer-Based pre-trained language models (PLMs) have become the *de facto* standard in a variety of natural language processing tasks, including NLI. Based on the encoding part of the transformer (Vaswani et al., 2017), researchers have proposed a number of highly successful NLU architectures, starting with BERT (Devlin et al., 2019), quickly followed by others, including RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019), DeBERTa (He et al., 2020), and smaller versions such as DistilBERT (Sanh et al., 2019) and Albert (Lan et al., 2019). Additionally, a number of sequence-to-sequence architectures have been proposed that are more similar to the original transformer than to BERT in that they directly try to transform one sequence to another, much like the basic set-up of neural machine translation. These include T5 (Raffel et al., 2019) and BART (Lewis et al., 2020).

These PLMs are then fine-tuned on specific

datasets, such as MNLI, which means that, while predicting labels on the dataset in question, a part of their parameters is being optimized. Fine-tuning usually takes several thousand times less computations than pre-training.

Such transformer-based PLMs fine-tuned to specific datasets perform impressively at standard natural language understanding (NLU) benchmarks, which include natural language inference (NLI) tasks. The MNLI Leaderboard, for instance, shows that the top ten PLMs are without exception transformer-based. Notably, in contrast to GLUE as a whole, the PLMs did not yet manage to outperform the human baseline at MNLI (as of June 15, 2022).

The Datasets Given the importance of fine-tuning for the entire method as it is currently practiced, it is clear that this method is squarely based on the availability – and quality – of large NLI datasets. Table 1 gives an overview on the currently most widely used datasets.[2]

Name of Dataset	Total Size	Genre
RTE (Wang et al., 2018)	6k	News, Wikipedia
QNLI (Wang et al., 2018)	116k	Wikipedia
WNLI (Wang et al., 2018)	852	hand-written
SICK (Marelli et al., 2014a)	9.8k	video & image captions
SNLI (Bowman et al., 2015)	570k	image captions
MNLI (Williams et al., 2018)	433k	10 genres, written & spoken

Table 1: Overview on Datasets used. Under "size", we report the total number of samples in train, test, and validation splits.

Thanks to their sheer size, SNLI and MNLI have come to dominate the field, as their size is suitable for fine-tuning large PLMs for NLI. As a consequence, as we shall see in the following section 2.2, most of the research on generalization issues focuses on these datasets.

There is a number of studies that critically assess the SNLI and MNLI datasets for their bias and thereby provides the groundwork for proposals following option 1 below (section 2.3). Williams et al. (2018) themselves note that their dataset contains a negation bias: if the hypothesis contains a negation, then it is more likely to be part of a contradiction pair (most likely, because simply negating the premise provides an efficient way for annotators to create contradiction pairs). Poliak et al. (2018) systematically investigate the prospects of hypothesis-only approaches (methods that only consider the hypothesis for predicting the label) to NLI in different datasets, finding better-than-random performance at most of them, which suggests the broad presence of statistical irregularities. Gururangan et al. (2018) show that SNLI and, to a lesser extent, MNLI, contain clues that make hypothesis-only approaches quite successful. Chien and Kalita (2020) focus on syntactic bias for PLMs fine-tuned on SNLI and MNLI, also finding that these bias are strong. Bernardy and Chatzikyriakidis (2019) argue that both SNLI and MNLI only cover a part of the entire range of human reasoning. In particular, they suggest that they do not cover quantifiers, nor strict logical inference.

The dataset that we will present in this study is intended to remedy both the lack of quantifiers and the lack of strict logical inference, given its focus on formally valid inferences.

Furthermore, we emphasize that, thanks to their near-ubiquitous use for fine-tuning, SNLI and to a greater extent MNLI determine the precise shape of the concept of inference that state-of-the-art models employ. On the one hand, the instructions given to crowdworkers are such that it seems reasonable to conclude that MNLI is about deductively valid inference: given a premise, crowdworkers are asked to "[w]rite a sentence that is definitely correct about the situation or event in the line [containing the premise]" (for the full instructions, see the appendix, section A). Requiring that the hypothesis be definitely correct given the correctness of the premise seems to require that it is not possible that the hypothesis could be false, given that the premise is true.

This reading, however, is contradicted by the fact that the creators of MNLI deliberately selected bits of text at random, not filtering for grammaticality, etc. These bits then served as prompts for the crowdworkers: they were tasked to write other bits of text for each prompt that either contracts, is entailed by, or is neutral vis-à-vis this prompt.

[2]Note that the RTE ("Recognizing textual entailment") dataset has been compiled from RTE1 (Dagan et al., 2005), RTE2 (Bar Haim et al., 2006), RTE3 (Giampiccolo et al., 2007), and RTE5 (Bentivogli et al., 2009). The QNLI ("Question-answering Natural Language Inference") dataset was created based on Rajpurkar et al. (2016). The WNLI ("Winograd Natural Language Inference") dataset was created based on Rahman and Ng (2012).

A consequence of the diversity of genres and this near-absence of preprocessing in MNLI is that the corpus contains premises such as (4).

(4) iuh-huh how about any matching programs

It is incoherent to say that questions entail any other statements: to entail something, a statement has to have determinate truth conditions; questions are textbook cases of sentences that have no determinate truth conditions. So, it is simply not possible for (4) to be part of any valid inference, let alone a deductively valid one. This issue stems from the very idea of MNLI, which is to represent the full variety of American English, using only minimal pre-processing.

Furthermore, crowdworkers are incentivized to produce large number of samples, which makes it rational to assume that a number of samples they produce are like example (5). Intended as a case of contradiction, it is clear that the premise does not contradict the conclusion in any logical sense: The speaker could simply have been lying, and no contradiction between premise and hypothesis would exist.

(5) (P) Oh, my friend, have I not said to you all along that I have no proofs. (H) I've always had the proof that he did it.

In sum, from a philosophical perspective, a qualitative inspection of the MNLI dataset shows that there might be some deeper problems in the set-up of the dataset. Furthermore, despite appearances to the contrary as per crowdworker-instructions, MNLI itself focuses on defeasible reasoning, that is, samples where the premise gives grounds to believe the hypothesis but does not entail it.

2.2 The Generalization Problem of Neural NLI

The basic problem that begins to emerge with this currently dominant approach to NLI is the problem of generalization. By this, we understand the inability of the PLMs to transfer the impressive performance on datasets on which they have been fine-tuned to out-of-dataset samples. Of course, a drop in performance is natural (even for humans) if the PLM is asked to perform the same task on substantially different data. If, however, the performance of a PLM simply collapses entirely when applied to out-of-dataset-samples, then it is *conceptually wrong* to say that the PLM has learned the task, namely correctly predicting logical relationships between statements, in the first place during fine-tuning: The task itself remains stable regardless of whether the samples are in or out of dataset. Together with the PLM's performance's lack of stability, this implies that it has learned something other than the task itself.

The problem of generalization in NLI is broadly acknowledged in the literature, see Zhou and Bansal (2020), Bras et al. (2020), Utama et al. (2020), Asael et al. (2021), He et al. (2019), Mahabadi et al. (2019), and Bernardy and Chatzikyriakidis (2019). It is generally assumed that the underlying cause of the problem of generalization is the PLMs' overfitting (see Goodfellow et al. 2016) on the training set. This overfitting, so the assumption goes, leads to the PLMs' picking up on spurious idiosyncrasies of the datasets, leading to the use of shallow heuristics and ultimately to a lack of generalization. Romanov and Shivade (2018) detail the generalization problems of pre-transformer PLMs in a highly specialized domain, namely medical history reports used by doctors.

If the models do not learn the central logical concepts during fine-tuning, what are they learning? The dominant view in the field is that they are learning so-called shallow heuristics, or bias: rules of thumb that work for the dataset due to some kind of bias in the data, but which do not apply to out of dataset samples, causing performance to collapse. In a much-discussed study, McCoy et al. (2019) conduct experiments to the conclusion that state-of-the-art PLMs use three kinds of syntactic heuristic at NLI tasks, which they call the lexical overlap, the subsequence, and the constituent heuristics. McCoy et al. (2019) also present a new stress test dataset called HANS ("Heuristic Analysis for NLI systems") that is built so that PLMs' use of the three heuristics will come to light in cases where the heuristics suggest entailment, but where the true label is not entailment.

2.3 Two Options to Address the Generalization Problem

In this section, we will consider the two main options that researchers have explored to address the problem of generalization in NLI.

Option 1: Debias the Dataset or the PLM The first option represents the mainstream of current thinking on NLI: It accepts the diagnosis that the models are merely picking up shallow heuristics

because there is a technical shortcoming in the method, and it tries to solve the problem by debiasing the datasets or the PLMs themselves. In table 2,[3] we list the papers, we mention whether their approach is based on a priori knowledge about the bias that one should tackle, and we report performance gains on the target dataset specified. Whenever available, we report performance gains on HANS, as this dataset has established itself as the *de facto* standard in the debiasing literature. As a consequence, these figures lend themselves best to comparisons between different approaches.

Paper	a priori knowl?	Target dataset	Acc.
He et al. (2019)	Yes	HANS	n.a.
Clark et al. (2019)	Yes	HANS	66.15 (+3.7)
Mahabadi et al. (2020)	Yes	HANS	71.95 (+10.1)
Yaghoobzadeh et al. (2019)	Yes	HANS	70.5 (+7.4)
Zhou and Bansal (2020)	Yes	Custom	+4.5
Belinkov et al. (2019)	Yes	Various	no gain
Dranker et al. (2021)	Yes	Various	no gain
Bras et al. (2020)	No	Various	+3.6
Utama et al. (2020)	No	HANS	69.7 (+8.2)
Nie et al. (2020)	(Yes)	Various	appr. +2
Bowman et al. (2020)	(Yes)	MNLI	no gain

Table 2: Overview on the extant approaches in option 1. Where no performance figures are given (n.a.), the paper doesn't report overall figures per dataset and it was not possible to extract these figures with simple, undisputable computations; "no gain" is a shorthand for "no significant gains".

Option 2: Hybrid Approaches Given the current generalization problem faced by purely neural approaches, some champions of symbolic methods have seen a chance to reinsert symbolic methods into the mainstream by combining neural with symbolic approaches. All current hybrid approaches rely on natural logic, an alternative to classical translation of natural language sentences into some formal langue. For details and references, see the appendix, section C. Hu et al. (2020) deliberately propose a lightweight, almost simplistic system that does not aim at setting a new state of the art, but rather at mapping out the lower bound performance of such a model. They explore its uses to provide training data for BERT.

An early approach at combining the two approaches is Raina et al. (2005). They combine classical formal logic with statistical learning for abductive reasoning (i.e., inference to the best explanation, a kind of non-monotonic inference, see (Lipton, 2004)).

Angeli and Manning (2014) introduce a seminal approach combining natural logic, monotonicity structures, WordNet and learned word probabilities as well as embeddings to conceive of NLI as a search problem. Kalouli et al. (2020) combine a classical symbolic system with a transformer-based neural PLM to achieve state-of-the-art performance on many standard datasets. Chen et al. (2021) adopt a different approach, conceiving of NLI as a path planning problem with the premise as the start and the hypothesis as the goal to be reached. They develop a system called NeuralLog that combines classical symbolic approaches using monotonicity notation (Hu et al., 2020) with, among others, Sentence-BERT embeddings to score the candidate hypotheses (Reimers and Gurevych, 2019). They report state of the art performance on both the SICK (Marelli et al., 2014b) as well as the MED (Yanaka et al., 2019) test sets; however, from among the neural approaches, they only consider BERT base. We report the results of these two only hybrid approaches post-HANS in table 3.

Paper	Target dataset	Acc.
Kalouli et al. (2020)	HANS	68.9 (+7.4)
Chen et al. (2021)	MED	93.4 (+21.8)

Table 3: Overview on the performance of the two most recent hybrid approaches. the MED dataset has been developed by Yanaka et al. (2018).

3 Dataset

In our experiment, we build on the insight gained from the qualitative inspection of MNLI as well as from research by Bernardy and Chatzikyriakidis (2019) that current NLI datasets lack samples that center on quantifiers as well as deductively valid inferences by providing a dataset that focuses on these very domains. Furthermore, our dataset provides a simple way to distinguish two properties of

[3]Note that the papers do not always report identical baseline performance, e.g., for BERT-base. We have reproduced these figures all the same, as the differences are small enough so that they do not affect our overall argument.

models that are often conflated: bias and shallow heuristics. As we have seen above (section 2.2), it is often said that the datasets or the models contain various biases. However, following Blodgett et al. (2020), we propose to use **bias** only for evaluations that are inherently normative and part of a larger worldview that is viewed critical. For instance, if a model expects that doctors are always men and therefore fails to correctly predict some logical relationships between sentences, one should attribute this to a bias: the model represents doctors as men, which is a clear case of a gender stereotype.

In contrast, a **shallow heuristic** is something that the models use irrespective of any such worldview, simply to succeed at a given task without fully learning it. The so-called negation bias is a clear case for such a shallow heuristic: It is not connected to any larger and problematic worldview but a simple instance of a rule of thumb.

While it has so far not been used to assess NLI capacities of NLU models, the systematic behind our dataset dates back to Aristotle. In his *Prior Analytics* (composed around 350 BC), Aristotle (1984, book 1) diligently analyzes the possible combinations of subject-, predicate-, and middle-term *via* quantifiers and negations to form a number of formally valid inferences. He deduces 24 formally valid patterns of inferences, so-called syllogisms. Example (2) is an instance of such a syllogism, belonging to the mood of the first figure that goes by the name of "BARBARA", the capital "A" signifying affirmative general assertions ("All X are Y.").

Now, consider the formal logical relationship in (6). By starting out with (2) and changing one single word, three letters in total, we have switched the relationship from entailment to contradiction.

(6) (P) All Germans are childcare workers and all childcare workers are fingerprint collectors. (H) No Germans are fingerprint collectors.

Finally, consider the formal logical relationship in (7). By changing one word, four letters, we switched the relationship from entailment to neutral.

(7) (P) All Germans are childcare workers and some childcare workers are fingerprint collectors. (H) All Germans are fingerprint collectors.

We are using a total of 12 formally valid syllogisms – called BARBARA, CELARENT, DARII, FERIO, CESARE, CAMESTRES, FESTINO, BAROCO, DISAMIS, DATISI, BOCARDO, FERISON – and we manually develop 24 patterns that are very similar to these 12 syllogisms, but where the first and the second sentence together contradict or are neutral to the third sentence. This yields a total of 36 patterns, 12 of which are valid syllogisms, 12 are contradictory, and 12 are neutral. To fit the premise-hypothesis structure expected by the models, we combine premise one and two to form a single premise.

We then use a pre-compiled list of occupations, hobbies, and nationalities to fill the subject- middle- and predicate-terms in these patterns. Using 15 of each of them and combining them with the 36 pattern yields 121500 test cases in total, each consisting of a premise and a hypothesis.[4] This variation allows us to capture the influence of any bias on model prediction, that is, any expectations of the models that certain nationalities are only likely to entertain certain hobbies and certain jobs, regardless of any valid inferences suggesting otherwise. Furthermore, it allows us to systematically distinguish it from shallow heuristics, rules of thumb that are not connected to any general worldviews or racial biases, but merely local attempts to succeed at the tasks without understanding it.

4 Experiment

We run a total of seven models on our test dataset, all of which are fine-tuned on standard NLI datasets, namely SNLI and MNLI (see table 4 for details: PLMs marked with one star "*" have only been fine-tuned on MNLI, PLMs marked with two stars have been fine-tuned on both SNLI and MNLI). The models are hosted by by Huggingface (Wolf et al., 2019), three of them are fine-tuned by Morris et al. (2020), prefixed with "textattack", and four by Reimers and Gurevych (2019), prefixed with "crossencoder".

The models' performances on MNLI, per our own evaluation (not all of the models provide evaluation scores, and we did not find precise documentation on how the scores were obtained), are given in table 4, for details of the evaluation, see the appendix, section B.

The basic idea behind the experiment is to assess

[4] The datasets can be found on the following github-repo: retoj/philo_nli.

PLM	N-Par.	MNLI-M
textattack-facebook-bart-large-MNLI*	406M	0.8887
crossencoder-deberta-base**	123M	0.8824
crossencoder-roberta-base**	123M	0.8733
crossencoder-MiniLM2-L6-H768**	66M	0.86602
textattack-bert-base-uncased-MNLI*	109M	0.8458
crossencoder-distilroberta-base**	82M	0.8364
textattack-distilbert-base-uncased-MNLI*	66M	0.8133

Table 4: Performance of the models in focus on the MNLI-Matched validation set. PLMs marked with one star "*" have only been fine-tuned on MNLI, PLMs marked with two stars have been fine-tuned on both SNLI and MNLI.

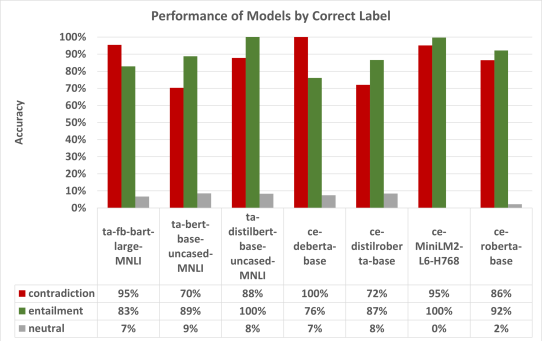


Figure 2: Performance on our syllogistic dataset by correct label.

whether the PLMs' performance on our dataset reveals any shallow heuristics learned by the models during fine-tuning on MNLI and SNLI.

5 Results

The results of our experiments are shown in figure 2. For instance, the model whose performance is represented on the very left, textattack's fine-tuned version of BART large, predicts the correct label in only 7% of cases for neutral labels, while doing so in 95% for entailment samples and still 83% for contradiction labels.

Figure 2 shows clearly that the models' predictions are quite accurate for labels *entailment* and *contradiction*, but very poor for *neutral*.

6 Discussion & Further Probes

6.1 Discussion of Experimental Results

Overall, figure 2 shows that textattack's distilbert leads the field with a accuracy of 65%, which might surprising just because it was among the smallest models evaluated here. However, there is growing evidence that NLI, and its more formal-deductive parts in particular, cannot be solved by simply increasing model size. Researchers at DeepMind find that larger models tend to generalize worse, not better, when it comes to tasks involving logical relationships. The large study by Rae et al. (2021, 23) strongly suggests that, in the words of the authors, "the benefits of scale are nonuniform", and that logical and mathematical reasoning does not improve when scaling up to the gigantic size of Gopher, a model having 280B parameters (in contrast, Gopher sets a new SOTA with many other NLU tasks such as RACE-h and RACE-m, where it outperforms GPT-3 by some 25% in accuracy).

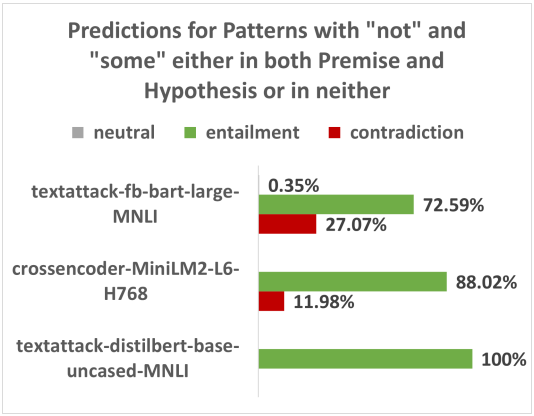


Figure 3: Predicted labels for patterns that are symmetric between premise and hypothesis regarding existential quantifier and negation.

Furthermore, figure 2 also shows that all of the models perform very poorly with neutral samples; indeed, none of the models is able to recognize such neutral relationships with a accuracy of more than 10%. Given that pure chance would still yield an accuracy of some 33%, this is a very poor performance.

We have therefore further probed the heuristics that the models might be using that could cause the poor performance with neutral labels. Manual inspection showed that they respond strongly to symmetries regarding quantifiers and negations between premises and hypotheses. In particular, if either both or none of the premise and the hypothesis contain a "some" (existential quantifier) or a negation (the symmetric conditions), then the models are strongly biased to predict *entailment* (see figure 3).

Conversely, if the pattern contains an asymmetry regarding existential quantifier and negation between premise and hypothesis, then the models are very strongly inclined to predict contradiction (see figure 4).

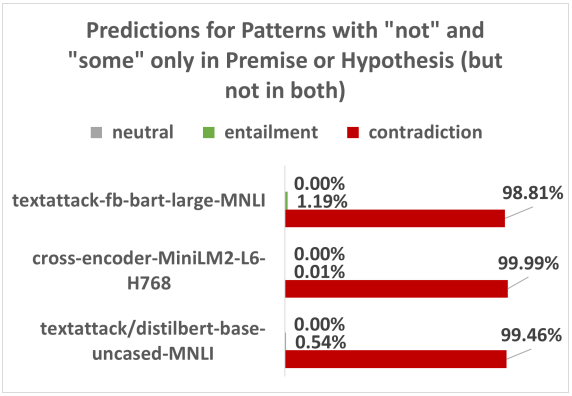


Figure 4: Predicted labels for patterns that are asymmetric between premise and hypothesis regarding existential quantifier and negation.

In the case of the contradiction and entailment pairs, these heuristics serve the models very well in our dataset, resulting in impressive performance. However, when applied to the neutral samples, the heuristics break down, performance falls far below simple guessing.

We conclude this part of our discussion by noting that the experiments did not show any significant bias in the behavior of the PLMs: Their accuracy did not change depending on existing preconceptions, say, that Germans are always engineers and like to collect stamps. What we have found, in contrast, is heavy use of shallow heuristics, as the figures 3 and 4 evince.

6.2 Fine-Tuning & Re-Evaluation, Comparison with a Symbolic Approach

In a next step, we assessed whether the models' poor performance with neutral samples in our dataset can be remedied with fine-tuning. We conducted two different fine-tuning runs, FT1 and FT2. Their sole difference consists in the way that we split up the 121k samples. For FT1, we used 110k samples for training and validation, and we tested on the neutral subset of the 10k remaining samples, which is about 3k samples ("3k" in figure 5). For FT2, we used 71k samples for training and validation, leaving the neutral subset of the remaining 50k samples, about 13k samples, for testing.[5]

We fine-tuned crossencoderMiniLM2-L6-H768 and textattack-distilbert-base-uncased-MNLI (BART-large from facebook exceeded our capacities). Furthermore, we also evaluated one of the currently leading symbolic NLI systems on both test datasets, namely GKR4NLI, introduced in (Kalouli et al., 2020). The results of all of these evaluations is shown in table 5.

Model	Neutr.	MNLI-M
FT1-crossencoder-MiniLM2-L6-H768	100%	72%
FT2-crossencoder-MiniLM2-L6-H768	62%	70%
FT1-textattack-distilbert-base-uncased-MNLI	100%	38%
FT2-textattack-distilbert-base-uncased-MNLI	61%	53%
GKR4NLI	89/23%	n.a.

Table 5: Accuracies of fine-tuned models and GKR4NLI on different test sets; For FT1-fine-tuned models, "Neutr." consists of 3k neutral samples from the syllogistic dataset, for FT2-fine-tuned models, it consists of 13k neutral samples from the same source. MNLI-M is MNLI-matched.

The results shown in table 5 show that fine-tuning does indeed help. In the first fine-tuning split FT1, both models achieve 100% accuracy; this, however, comes at rather high cost in terms of accuracy on MNLI-matched (14% and 43% respectively). GKR4NLI also performs well at this test set with 89% out of the box. With regard to the second fine-tuning split FT2, GKR4NLI's performance drops to 23%, while the two fine-tuned models achieve accuracies of around 62%, again at the cost of significantly reduced accuracy in MNLI.

These results suggest that it is not easy for the models tested to combine the representations needed to perform well at MNLI-matched with those needed to do well in our neutral samples. In particular, the results suggest that a large number of training samples is needed, as in FT1. We note that our results leave open the possibility that larger models can accommodate both kinds of sample.

At this point, we would like to compare our results with those obtained by Richardson et al. (2020). They use a cleverly chosen roaster of semantic fragments (i.e. subsets of a language translatable into formal logic, in particular first-order predicate logic) to test the models' understanding of the logical relationships of contradiction, entailment and neutral. They find that the models tested perform poorly on these tasks, but that this performance can be remedied with fine-tuning the models on sufficient amounts of training data that

[5]We adapted a huggingface-notebook found here, letting run each fine-tuning process for three epochs with a batch size of 16 on one GPU of a DGX-2.

has been synthetically generated from these fragments. In contrast to the semantic fragments used by Richardson et al. (2020), our datasets seem to pose a more difficult challenge for the models that we have tested (despite the fact that Richardson et al. 2020 only considered BERT-base, while we have also included larger and more recent models). Perhaps we have made some progress towards what Richardson et al. (2020) explicitly ask for, namely more difficult fragments?

We take these results to confirm that our dataset can make a valuable contribution to the field, as it presents a challenge for both neural and symbolic systems. Indeed, in light of these results, one could wonder whether it is not unfair to expect any NLI system to master our syllogistic dataset, as samples such as (2), (6), and (7) might be said to be very far away from ordinary language use. In response to this, we point out that, as a matter of logical fact, these are formally valid inferences which should be covered by any NLI system that aspires to cover the full extent of NLI. Furthermore, students of logics have acquired their concepts of formal validity through such examples for millennia, making it a rather natural stepping stone for AI systems. Finally, as already mentioned, it might very well be that large models could accommodate both the defeasible kinds of inferences in MNLI and our deductively valid ones.

7 Conclusion

We have detailed the problem of generalization that current neural approaches to NLI face from the background of philosophical logic. We have suggested that current datasets are light on deductively valid inferences, proposed a distinction between bias and shallow heuristics, and we have proposed our own syllogistic dataset. This dataset allows to distinguish between bias and shallow heuristic, it focuses on formally valid inferences, and our results suggest that it can help to improve both neural and symbolic approaches.

References

Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 534–545.

Aristotle. 1984. Prior analytics. In Jonathan Barnes, editor, *The Complete Works of Aristotle*, pages 39–113. Oxford University Press.

Dimion Asael, Zachary Ziegler, and Yonatan Belinkov. 2021. A generative approach for mitigating structural biases in natural language inference. *arXiv preprint arXiv:2108.14006*.

R Bar Haim, I Dagan, B Dolan, L Ferro, D Giampiccolo, B Magnini, and I Szpektor. 2006. The second pascal rte challenge. *Proceedings of the 2nd PASCAL Challenge on RTE*.

Yonatan Belinkov, Adam Poliak, Stuart Shieber, Benjamin Van Durme, and Alexander Rush. 2019. Don't take the premise for granted: Mitigating artifacts in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 877–891. Association for Computational Linguistics.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Jean-Philippe Bernardy and Stergios Chatzikyriakidis. 2019. What kind of natural language inference are nlp systems learning: Is this enough? In *ICAART (2)*, pages 919–931.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635.

Johan Bos and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesn't).

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 632–642. Association for Computational Linguistics (ACL).

Samuel R Bowman, Jennimaria Palomaki, Livio Baldini Soares, and Emily Pitler. 2020. New protocols and negative results for textual entailment data collection. In *EMNLP (1)*.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. *arXiv preprint arXiv:2002.04108*.

Zeming Chen, Qiyue Gao, and Lawrence S Moss. 2021. Neurallog: Natural language inference with joint neural and logical reasoning. *arXiv preprint arXiv:2105.14167*.

Tiffany Chien and Jugal Kumar Kalita. 2020. Adversarial analysis of natural language inference systems. *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 1–8.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4069–4082. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Ernest Davis. 2017. Logical formalizations of commonsense reasoning: a survey. *Journal of Artificial Intelligence Research*, 59:651–723.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yana Dranker, He He, and Yonatan Belinkov. 2021. Irm—when it works and when it doesn't: A test case of natural language inference. *Advances in Neural Information Processing Systems*, 34.

Gottlob Frege. 1892. Über sinn und bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, 100:25–50.

James W Garson. 2006. *Modal logic for philosophers*. Cambridge University Press.

Gerhard Gentzen. 1935. Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 35.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112.

He He, Sheng Zha, and Haohan Wang. 2019. Unlearn dataset bias in natural language inference by fitting the residual. *arXiv preprint arXiv:1908.10763*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Hai Hu, Qi Chen, Kyle Richardson, Atreyee Mukherjee, Lawrence S Moss, and Sandra Kübler. 2020. Monalog: a lightweight system for natural language inference based on monotonicity. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 284–293.

Thomas F Icard III and Lawrence S Moss. 2014. Recent progress on monotonicity. In *Linguistic Issues in Language Technology, Volume 9, 2014-Perspectives on Semantic Representations for Textual Inference*.

Stanislaw Jaskowski. 1934. On the rules of suppositions in formal logic. *Studia Logica*, 1(1).

Aikaterini-Lida Kalouli, Richard Crouch, and Valeria de Paiva. 2020. Hy-NLI: a hybrid system for natural language inference. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5235–5249, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Robert Koons. 2021. Defeasible Reasoning. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, Fall 2021 edition. Metaphysics Research Lab, Stanford University.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ArXiv*, abs/1910.13461.

Vladimir Lifschitz, Leora Morgenstern, and David Plaisted. 2008. Knowledge representation and classical logic. *Foundations of Artificial Intelligence*, 3:3–88.

Peter Lipton. 2004. *Inference to the Best Explanation*, 2 edition. Routledge.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Bill MacCartney and Christopher D Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200.

Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the eight international conference on computational semantics*, pages 140–156.

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. 2019. End-to-end bias mitigation by modelling biases in corpora. *arXiv preprint arXiv:1909.06321*.

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. 2020. End-to-end bias mitigation by modelling biases in corpora. In *ACL*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014a. The SICK (Sentences Involving Compositional Knowledge) dataset for relatedness and entailment.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014b. A sick cure for the evaluation of compositional distributional semantic models. In *Lrec*, pages 216–223. Reykjavik.

John McCarthy. 1959. Programs with common sense. *Proceedings of the Symposium on Mechanization of Thought Processes*, (1).

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901.

Francis Jeffry Pelletier and Allen Hazen. 2021. Natural Deduction Systems in Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, Winter 2021 edition. Metaphysics Research Lab, Stanford University.

Alvin Plantinga. 1974. *The Nature of Necessity*. Oxford University Press.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191. Association for Computational Linguistics.

Willard Van Orman Quine. 1980 [1951]. Two dogmas of empiricism. In *From a Logical Point of View*, pages 20–46. Harvard University Press.

Jack W. Rae, Sebastian Borgeaud, and Trevor Cai et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. DeepMind Company Publication.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The Winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789. Association for Computational Linguistics.

Rajat Raina, Andrew Y Ng, and Christopher D Manning. 2005. Robust textual inference via learning and abductive reasoning. In *AAAI*, pages 1099–1105.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Kyle Richardson, Hai Hu, Lawrence Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8713–8721.

Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596. Association for Computational Linguistics.

Bertrand Russell. 1905. On denoting. *Mind*, 14(56):479–493.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Raymond M. Smullyan. 1968. *First-Order Logic*. Dover.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Towards debiasing nlu models from unknown biases. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7597–7610.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yadollah Yaghoobzadeh, Remi Tachet, Timothy J Hazen, and Alessandro Sordoni. 2019. Robust natural language inference models with example forgetting. *arXiv e-prints*, pages arXiv–1911.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019. Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 31–40.

Hitomi Yanaka, Koji Mineshima, Pascual Martínez-Gómez, and Daisuke Bekki. 2018. Acquisition of phrase correspondences using natural deduction proofs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 756–766. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Annie Zaenen, Lauri Karttunen, and Richard Crouch. 2005. Local textual inference: can it be defined or circumscribed? In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 31–36.

Xiang Zhou and Mohit Bansal. 2020. Towards robustifying nli models against lexical dataset biases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8759–8771.

A Full Instructions Given to Crowdworkers

Williams et al. (2018, 1114) specifies the following tasks for the crowdworkers:

"This task will involve reading a line from a non-fiction article and writing three sentences that relate to it. The line will describe a situation or event. Using only this description and what you know about the world:

- Write one sentence that is definitely correct about the situation or event in the line.
- Write one sentence that might be correct about the situation or event in the line.
- Write one sentence that is definitely incorrect about the situation or event in the line. "

B Method used for evaluation of Models on MNLI

To evaluate the models, we have used Huggingface's trainer API, see Huggingface (Wolf et al., 2019). In particular, we followed the instructions in the notebook here. We evaluated the models using the API out-of-the-box, with the following exceptions:

1. The textattack-models had as labels "LABEL_0, LABEL_1, LABEL_2", which could not be read by the function that ensures that the labels are used equivalently by both model and dataset; hence, we reconfigured the models to use as labels "contradiction, entailment, neutral".
2. facebook-bart-large-mnli by textattack posed two additional challenges.
   (a) Due to out of memory issues, we had to split up processing of the validation set into three chunks, averaging the accuracy received afterwards.
   (b) The logits containing the predictions issued by facebook-bart-large-mnli could not be processed by the evaluation function, which caused the need to select only the first slice of the tensor that the model was issuing, ensuring that the metric function got a 1-dimensional tensor to compute accuracy.

C From First-Order Representations to Natural Logic

Traditionally, the topic of common-sense reasoning, and later of NLI, as we understand it, was approached by the use of formal logic, predominantly first-order logic,[6] see Davis (2017) and Lifschitz et al. (2008) for extensive surveys of this approach, and McCarthy (1959) for the pioneering paper in this tradition. Bos and Markert (2005) and Bos and Markert (2006) are two typical cases in this tradition. In the latter, the authors find that, overall, adding logical processing to a shallow word-overlap approach actually hinders rather than boosts performance.

More recently, the once-dominant approach of representing premise and hypothesis in a formal language such as first-order predicate logic has been superseded by attempts to recover the logical structure of a sentence and the logical relationship between two sentences by directly annotating the natural language sentence. In particular, the so-called monotonicity calculus has been popular in a number of approaches. Icard III and Moss (2014) present an accessible and thorough review of recent theoretical work on this monotonicity approach.

The calculus stands in the tradition of natural logic (pioneered by Gentzen 1935 and Jaskowski 1934, for an overview, see Pelletier and Hazen 2021) is used by the NatLOG system developed by MacCartney and Manning (2007), MacCartney and Manning (2008), and MacCartney and Manning (2009). The basic idea behind the monotonicity calculus is to use low-level structural properties of quantifiers and predicates to assess the validity of an inference. For instance, the validity of the inference from "Every dog is a mammal" to "Every poodle is a mammal" is explained by a bottom-up combination of properties from the quantifier as well as the predicate involved – and not by translating the entire sentence into predicate calculus.

[6] As it has been pioneered by Frege (1892) and developed by Russell (1905).