Morpheme Induction for Emergent Language

Brendon Boldt and **David Mortensen**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
{bboldt,dmortens}@cs.cmu.edu

Abstract

We introduce CSAR, an algorithm for inducing morphemes from emergent language corpora of parallel utterances and meanings. It is a greedy algorithm that (1) weights morphemes based on mutual information between forms and meanings, (2) selects the highest-weighted pair, (3) removes it from the corpus, and (4) repeats the process to induce further morphemes (i.e., Count, Select, Ablate, Repeat). The effectiveness of CSAR is first validated on procedurally generated datasets and compared against baselines for related tasks. Second, we validate CSAR's performance on human language data to show that the algorithm makes reasonable predictions in adjacent domains. Finally, we analyze a handful of emergent languages, quantifying linguistic characteristics like degree of synonymy and polysemy.

1 Introduction

Emergent languages—communication systems invented by neural networks via reinforcement learning—are fascinating entities. They give us a chance to experiment with the processes underlying the development of human language to which we would not otherwise have access. A perennial problem in this field, though, is that emergent languages are difficult to interpret. The strategies emergent languages use to convey meaning do not always align with those known from human language (Kottur et al., 2017; Chaabouni et al., 2019; Kharitonov and Baroni, 2020). Yet a lack of general-purpose methods for investigating the structure of emergent communication prevents us from systematically investigating how they encode meaning.

As an essential step towards understanding emergent languages, we introduce CSAR, an algorithm for *morpheme induction* on emergent language. That is, given an input corpus of parallel data: utterances and their associated meanings, find the smallest meaningful components of utterances with their accompanying meaning. Simply put, this task is to jointly segment utterances and align them with their meanings. The output of this algorithm, then, is a mapping between the forms and meanings of the emergent language (example shown in Fig. 1). Furthermore, the proposed algorithm is easily applicable to almost any emergent language due to the simplicity of the input format. In fact, the algorithm is general purpose enough to produce reasonable results in other domains, as we demonstrate with human language-based image captioning, machine translation, and word segmentation data. Validating CSAR's performance on human language data as well as synthetic data is critical to demonstrating its effectiveness since there is no way of obtaining ground truth morphemes for emergent languages.

Form	Meaning
3, 6	{not, gray}
7, 7	{not, blue}
32	{circle}
4, 5	{not, yellow}
6, 12, 6, 12	{green, or, yellow}
3, 12, 3	{blue, or, yellow}

Figure 1: Example of morphemes extracted from a signalling game with pixel observations.

An inventory of the morphemes of an emergent language is the foundation of many further linguistic analyses. Existing studies of compositionality (Korbak et al., 2020), word boundaries (Ueda et al., 2023), and grammar induction (van der Wal et al., 2020) could be validated and augmented with information on the morphology of emergent languages, and new directions would also be made possible, including analyses of the morphosyntactic patterns and typological properties of emergent languages. Ultimately, such studies form one of the pillars of emergent communication research: learning what emergent language can tell us about human language (Boldt and Mortensen, 2024).

In Section 2 we define the task of morpheme induction and discuss related work. Section 3 presents the proposed algorithm, CSAR. Section 4 validates CSAR's performance on procedurally generated and human language data. Section 5 applies CSAR to emergent languages. And we discuss the paper's findings and limitations and conclude in Sections 6 to 8.

Contributions This work: (1) Introduces an algorithm for inducing morphemes, applicable to a wide variety of emergent language corpora. (2) Offers an easy-to-use Python implementation for executing the proposed algorithm on arbitrary emergent language data. (3) Provides a first look into the morphology of emergent languages including phenomena such as polysemy and synonymy.

2 Problem Definition

In this section we give a precise definition of the problem and the terms we will use throughout the paper.

2.1 Task: morpheme induction

We define the task of *morpheme induction* as follows: Given a corpus of *utterances* and their corresponding *complete meanings*, identify *minimal, well-founded form–meaning pairs* (i.e., *morphemes*) present in the corpus. This collection of pairs is the *morpheme inventory* of the corpus.

form a sequence of (form) tokens; represented as an integer sequence in emergent language.

utterance a complete sequence of tokens produced by an agent; forms are subsequences of utterances.

meaning a set of meaning tokens (i.e., atomic meanings grounded in the environment).

complete meaning a meaning which represents the entire meaning of an utterance.[1]

well-founded a form–meaning pair is well-founded when the particular form corresponds with a particular meaning.

minimal a well-founded form–meaning pair is minimal when there is no way to decompose the pair while maintaining continuity of meanings.

It is important to note that we make two assumptions about the complete meanings. First, complete meanings are assumed to be *abstracted* already, hence the reason we can represent them as a set of atomic meanings. That is to say that the "raw semantics" of the utterances are already broken down into individual components of interest; this task does not entail automatically finding meaning in arbitrary data (cf. clustering). Second, since complete meanings are sets, they are not able to represent more complex phenomena that might require graph structures, for example (cf. abstract meaning representations).

Additionally, we note that *well-founded correspondence* is a concept subject to a variety of philosophical accounts. Sometimes these accounts hold that the meaning is derived from either the behavior or state of mind of a language user. Yet in this task, we only have access to a corpus, not to the language users themselves; thus, we employ a notion of "well-founded correspondence" most akin to a statistical view semantics (e.g., as in the distributional hypothesis).

[1]atomic meaning $\in$ meaning $\subseteq$ complete meaning

2.2 Related work

Emergent language Lipinski et al. (2024) serves as the inspiration for this paper through its application of normalized pointwise mutual information to probe emergent languages for certain kinds of form–meaning relationships, though it stops short of providing full morpheme inventories over arbitrary data. Ueda et al. (2023) introduces a method of form-only segment induction for emergent language based on token-level entropy patterns in utterances.

Finally, Brighton (2003) introduce methods for inducing morphemes from simulations of language evolution. In particular, the algorithm is based on finite state transducers and the minimum description length principle. The key difference, though, is that the FST-based method assumes a strict form–meaning correspondence that does not appear to hold in emergent languages generated by deep neural networks. Thus, CSAR is designed to handle noise and looser form–meaning correspondence.

Statistical word alignment The task of morpheme induction resembles the task of statistical word alignment for machine translation insofar as it involves learning a mapping between two modalities. Well-known algorithms for this task include the IBM alignment models (Brown et al., 1993). While morphemes can be extracted from the alignments, the alignments themselves are not intended to represent morphemes as such.

Segment induction Segment induction is similar to morpheme induction, except that it deals only with the forms; these methods, then, cannot provide a mapping between form meaning because they are meaning-unaware. Sometimes this task is called "morpheme induction" since the segments are supposed to correspond to morphemes, but they are not morphemes in the particular sense we use for this paper, that is: explicit form–meaning pairs. An example of an algorithm which addresses this task is Morfessor (Creutz and Lagus, 2002; Virpioja et al., 2013) or the submissions to the SIGMORPHON 2022 Shared Task Batsuren et al. (2022). Narasimhan et al. (2015) introduce a semi-supervised segment induction algorithm that uses semantic features to guide segmentation (viz. groups of morphologically related words), although meanings are not modelled explicitly and "word" is not a well-defined concept for emergent languages. The discovery of valid segments by tokenization methods based on statistics—such as BPE (Sennrich et al., 2016; Gage, 1994) and Unigram LM (Kudo, 2018)—is largely an epiphenomenon, not a design goal.

3 Algorithm

In this section we introduce the algorithm for morpheme induction: CSAR (Count, Select, Ablate, Repeat). CSAR comprises the following steps:

1. Collect form and meaning candidates from the corpus.
2. While form and meaning candidates remain.
   (a) Count co-occurrences of form and meaning candidates.
   (b) Select form–meaning pair with the highest weight.
   (c) Remove instances of the form–meaning pair from the corpus.
3. Selected form–meaning pairs constitute the morpheme inventory of the corpus.

The code implementing CSAR as well as the experiments discussed later is available under a free and open source license at `https://github.com/brendon-boldt/csar`.

3.1 Representation and preprocessing

Input data The input data to CSAR is a parallel *corpus* of utterances and their meanings. Each record in the corpus is a tuple of form and meaning where a form is a list of (form) tokens and a meaning is a set of (meaning) tokens.

Candidate collection Given the corpus, we can identify and count the *form* and *meaning candidates* to produce their corresponding *occurrence matrices*. A form candidate is any substring of form tokens under consideration for inducing morphemes. A meaning candidate is any subset of meaning tokens under consideration for inducing morphemes. The most straightforward approach is to simply consider every non-empty substring of forms and subset meanings, although CSAR is not constrained to this approach in theory (cf. Section A.1).

Having defined the universe of forms and meanings, we can build a binary *occurrence matrix* for forms and one for meanings, where each row corresponds to a record and each entry corresponds to the presence (1) or absence (0) of a form/meaning in that record. Thus, the form occurrence matrix has the shape $O_{\mathcal{F}} : |\mathcal{R}| \times |\mathcal{F}|$ and the meaning matrix $O_{\mathcal{M}} : |\mathcal{R}| \times |\mathcal{M}|$, where $\mathcal{R}$ is the list of records, $\mathcal{F}$ is the set of all forms candidates, and $\mathcal{M}$ is the set of all meanings candidates.

Example If we had a simple corpus with records ("s", $\square$), ("st", $\boxtimes$), ("ct", $\otimes$), the corresponding occurrence matrices would be:

$$\mathcal{O}_{\mathcal{F}} = \begin{bmatrix} \cdot & \mathrm{s} & \cdot & \cdot & \cdot \\ \cdot & \mathrm{s} & \mathrm{t} & \cdot & \mathrm{st} \\ \mathrm{c} & \cdot & \mathrm{t} & \mathrm{ct} & \cdot \end{bmatrix} \quad \mathcal{O}_{\mathcal{M}} = \begin{bmatrix} \square & \cdot & \cdot & \cdot & \cdot \\ \square & \times & \cdot & \boxtimes & \cdot \\ \cdot & \times & \bigcirc & \cdot & \otimes \end{bmatrix}, \quad (1)$$

where entries with value 1's are shown with the occurring symbols, and entries with value 0's with $\cdot$ for clarity.

3.2 Main loop

Weighting and co-occurrences Given the occurrence matrices, the next step is to compute the weights of all potential pairs. The pair with the highest weight will be selected and added to the morpheme inventory. The weight of a form–meaning pair is the mutual information of the binary variables representing the corresponding form and meaning. The mutual information of a particular form–meaning pair is given by

$$I(F; M) = \sum_{x \in F} \sum_{y \in M} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)}, \quad (2)$$

where $F = \{f, \neg f\}$, $p(f)$ is the probability of f appearing in a record, $p(\neg f)$ is the probability of f not appearing, and the rest are defined analogously. The key term of the mutual information expression is the joint probability between a form and a meaning, $p(f, m)$: since f and m are binary variables, all other joint probabilities can be computed from their joint probability and the marginal probabilities. The joint probability can be computed by normalizing the sum of co-occurrences of given forms and meanings, namely:

$$p(f, m) = \frac{1}{|\mathcal{R}|} \sum_{j=1}^{|\mathcal{R}|} O_{\mathcal{F}}[j, i_f] \wedge O_{\mathcal{M}}[j, i_m] \quad (3)$$

where i_f and i_m are the indices of f and m in their respective matrices. More succinctly, co-occurrences can be computed with matrix multiplications, yielding

$$p(f, m) = \frac{1}{|\mathcal{R}|} \cdot \left(O_{\mathcal{F}}^{\top} O_{\mathcal{M}} \right) [i_f, i_m] \quad (4)$$

Other weighting methods were explored including joint probabilities, pointwise mutual information, and normalized pointwise mutual information, though mutual information was found to perform best empirically.

The above weighting function results in ties which we break with the following heuristics: (1) higher initial weight, (2) fewer selected pairs with this form, (3) larger form size, and (4) smaller meaning size.

Remove pair from corpus The final step of the algorithm's main loop is ablating the pair from the corpus. That is, once we select a form–meaning pair, we want to remove all co-occurrences of the form and meaning in order to determine what form–meaning correspondences remain to be explained. For example, after ablating the pair ("t", $\times$), the corpus from above would comprise ("s", $\square$), ("s", $\square$), and ("c", $\bigcirc$); the occurrence matrices would then be updated to reflect this. In cases where ablating a pair is ambiguous, we apply a heuristic (see Section A.2).

Repeating and stopping After ablating the selected form–meaning pair, the algorithm repeats the main loop, beginning again at the weight-computation step (with the updated occurrence matrices). The one difference is that—in subsequent weight computations—the weight of a pair cannot go up, preventing spurious correlations from arising in later steps.

This loop continues until form or meaning occurrences are exhausted or some other criterion is met (e.g., time limit, inventory size limit). In this way, CSAR is an "anytime" algorithm since it can be stopped after an arbitrary number of iterations and still produce a sensible result. This is because the most heavily weighted morphemes can be considered the highest *confidence* morphemes, meaning that stopping the algorithm before completion will only leave out the lowest confidence morphemes.

3.3 Implementation

The implementation of CSAR introduced in this paper is written in Python making use of sparse matrices from `scipy` (Virtanen et al., 2020, BSD 3-Clause license) and JIT compilation with `numba` (Lam et al., 2015, BSD 2-Clause license) to speed up execution. CSAR is conceptually simple. Most of the implementation complexity lies in efficiently handling the occurrence matrices, especially when removing a form–meaning pair from the corpus. For example, the co-occurrence matrix has the shape $|\mathcal{F}| \times |\mathcal{M}|$ which is massive considering that $\mathcal{F}$ and $\mathcal{M}$ are already accounting for the universes of all possible forms and meanings in the corpus. Nevertheless, there are a wide range of heuristics that can be applied to greatly speed up execution while maintaining performance (see Section A.3).

4 Empirical Validation

To validate the ability of CSAR to find well-founded morpheme inventories, we test it against procedurally generated datasets as well as human languages. Since we do not have access to ground truth morphemes for emergent languages, we gauge the effectiveness of CSAR's morpheme induction in the next best way: by testing its performance in these adjacent domains. Procedurally generated datasets (described in Section 4.1) both give us access to the "ground truth" morphemes and allow us to vary particular facets of the languages. Having ground truth morphemes allows us to quantitatively evaluate CSAR against baseline methods (Section 4.2). Fine-grained control over the facets of the languages permits us to identify particular phenomena that are challenging for CSAR to induce correctly (Section 4.3). We also test CSAR against human language data (Section 4.4) in order to give a qualitative sense of the effectiveness of the algorithm.

4.1 Procedural datasets

The dataset-generating procedure has the following basic structure: (1) Meanings are sampled according to some structure (viz. a fixed attribute–value vector). (2) An utterance is produced from this meaning according to a mapping of meaning components to form components. (3) The form–meaning pairs that were used to generate the utterance are added to the set of ground truth morphemes. In the basic case, for example, each particular attribute and value is associated with a unique sequence of tokens which are concatenated to form an utterance, creating a one-to-one mapping from meanings to forms.

Variations Such languages are trivial to induce morphemes from, so we introduce the following variations to produce more complex datasets:

Synonymy Multiple forms may correspond to the same meaning.

Polysemy Multiple meanings may correspond to the same form.

Multi-token forms A form may comprise more than one token, possibly overlapping with other forms.

Vocab size Number of unique tokens.

Sparse meanings Meanings occur independently of each other with no additional structure (i.e., not structured as attribute–value pairs).

Distribution imbalance Meanings are sampled from non-uniform distributions.

Dataset size Number of records in the dataset.

Number of meanings Total number of meanings (e.g., varying number of attributes and values).

Noise forms Form tokens not corresponding to any meanings are added.

Shuffle form Inter-form order is varied randomly (while maintaining intra-form order).

Non-compositionality A given form may correspond to multiple meanings simultaneously.

For the following analyses, we report values for a collection of procedural datasets built from the Cartesian product of two values for each of the above variations (one where the variation is inactive and one where it is). See Section B.1 for details.

Evaluation metric We use F_1 score (harmonic mean of precision and recall) to assess the quality of an induced morpheme inventory given the ground truth inventory. We define precision as

$$\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\max_{g\in\mathcal{G}} s(i,g), \tag{5}$$

where $\mathcal{I}$ is the set of induced morphemes, $\mathcal{G}$ is the set of ground truth morphemes, and s is the similarity function. For exact F_1, the similarity function is 1 if the morphemes are identical and 0 otherwise. In fuzzy F_1, the similarity function is the minimum of form similarity (normalized insertion–deletion ratio[2]) and meaning similarity (Jaccard index). Recall is defined similarly to precision except that the roles of $\mathcal{I}$ and $\mathcal{G}$ from Eq. (5) are reversed.

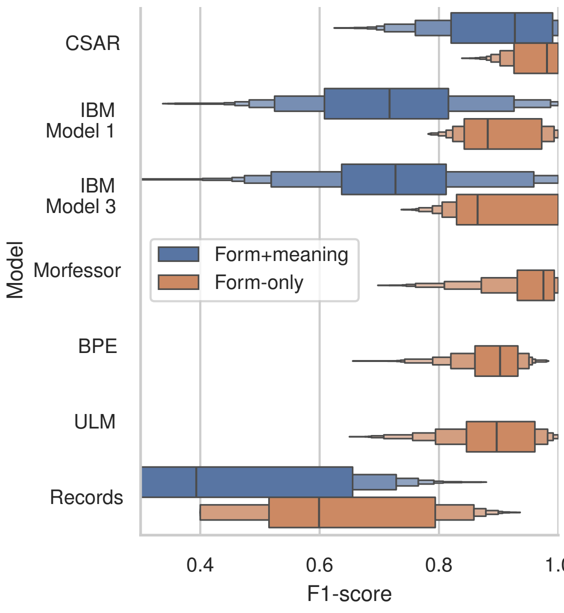

Figure 2: Fuzzy F_1 scores for CSAR and baseline methods across procedural datasets. Results reported for form–meaning inventories and form-only inventories.

4.2 Comparison with baselines

Below we describe the baseline methods we use for comparison.

IBM Model 1 Simple expectation-maximization approach to machine translation primarily through aligning words in a sentence-parallel corpus. (Brown et al., 1993)

IBM Model 3 Built on top of the IBM Model 1 to handle phenomena such as allowing a form to align to no meaning.

Morfessor A form-only segmentation algorithm built to handle human language; also uses an EM algorithm.

Byte pair encoding A greedy form-only tokenization method which recursively merges frequently occurring pairs of tokens. Vocabulary size is selected according to a simple heuristic (see Section B.2). (Gage, 1994; Sennrich et al., 2016)

Unigram LM An EM-based form-only tokenization method which starts with a large vocabu-

[2] $1-(\text{insertions}+\text{deletions})/(|s_1|+|s_2|)$

lary and iteratively removes tokens contributing least to the likelihood of the data. Vocabulary size is selected according to a simple heuristic (see Section B.2). (Kudo, 2018)

Record A trivial baseline where the inventory is just the set of all records.

For the baseline methods which do not handle meanings and only produce forms, we report the form-only F_1 score (i.e., $s(i, g)$ only takes the form into account), though CSAR and IBM models still have access to meanings. For form-only metrics, we exclude datasets which include noise forms as form-only methods cannot identify which forms are noise.

Results The results of CSAR and the baselines on the procedural datasets are presented in Fig. 2, which shows the distributions of mean scores for each hyperparameter setting for the procedural datasets. Each setting was repeated over 3 random seeds. Additional results are given in Section B.3. For inducing full morphemes (form and meaning), CSAR performs the best by a large margin over the baselines (and even greater margin when considering exact F_1). The IBM alignment models perform better than the trivial record-based baseline but still perform noticeably worse than CSAR. While CSAR yields roughly equal precision and recall, the IBM models' precision is lower than their recall suggesting that they are more prone to inducing spurious morphemes than CSAR.

When evaluating the forms only, we find that CSAR is the best method with Morfessor exhibiting comparable performance. The IBM alignment models exhibit roughly the same performance as the tokenization methods (BPE and Unigram LM). As with the full morpheme results, CSAR is the only method to achieve comparable precision and recall with all other baselines having precisions lower than their recalls.

4.3 Error Analysis

For the most part, the errors CSAR makes are "edit errors": identifying a correct morpheme but adding or removing a form or meaning token. This is reflected in the near-parity between precision and recall. This is in contrast to the baseline methods which are more prone to inducing too many morphemes, leading to lower precision.

Generally speaking, as more variations are added to a dataset, the performance degrades further. In particular, CSAR's performance decreases the most with small corpus sizes, overlapping multi-token forms, and non-compositional mappings. On the other hand, using sparse meanings, shuffling the forms, and using a non-uniform meaning distribution have relatively little effect.

Dataset	Induced Morpheme
Morphology	(“ed$”, {PAST}) (“ ’ ”, {POSSESSIVE})
Image captions	(“stop sign”, {stop sign}) (“woman”, {person}) (“skier”, {person, skis})
Translation	(“Member States”, {Mitgliedstaaten})

Figure 3: Examples of morphemes induced from various human language datasets and tasks.

4.4 Human language data

In this section we discuss the results of running CSAR on three different human language datasets. While these datasets are not the intended domain of CSAR—and CSAR is certainly not the best algorithm for the tasks—the point of these experiments is to demonstrate the general effectiveness of the algorithm qualitatively (examples shown in Fig. 3). Since these datasets are larger, we employed heuristic optimizations to CSAR to reduce their runtime (described in Section A.3). The top 100 induced morphemes for each human language dataset are given in Section E.1.

Morpho Challenge The first human language dataset we use is from the Morpho Challenge (Kurimo et al., 2010). This dataset is a human language approximation of the task of morpheme induction for emergent language. Concretely, the utterances are single English words, divided up at the character level, while the meanings are the constituent morphemes.

CSAR is able to recover a wide variety of morphemes including: roots like (“ˆfire”, {fire}), prefixes like (“ˆre”, {re-}), suffixes like (“ed$”, {PAST}), and other affixes like (“ ’ ”, {POSSESSIVE}). While the vast majority of morphemes CSAR induces are accurate, a handful of the lowest-weighted morphemes are spurious (e.g., (“s$”, {boy})) likely due to inaccurate decoding earlier in the process (i.e., part of the true form for a given meaning was included in a prior meaning).

Image captions The next dataset we employ is the MS COCO dataset (Lin et al., 2015, CC BY 4.0). In particular, we take the image captions to be the utterances, treating words as atomic units, and the meaning to be the labeled objects in the image (e.g., person, cat).

The bulk of highest weighted induced morphemes are direct equivalents of the objects they describe (e.g., (“cat”, {cat})). We find instances of synonymy (e.g., (“bicycle”, {bicycle}) and (“bike”, {bicycle})) as well as polysemy (e.g., (“animals”, {cow}) and (“animals”, {sheep})). Finally, we also observe compound forms like (“stop sign”, {stop sign}) as well as compound meanings such as (“skier”, {person, skis}). As we go beyond the top 100 or so, the associations between forms and meanings remain reasonable but become looser such as (“bride”, {dining table, tie}) or (“sink”, {toothbrush}).

Machine translation For machine translation, we use the WMT16 dataset and the English–German split, in particular (Bojar et al., 2016). In this case, the English text is considered to be the utterance and the German text to be the meaning, with words being the atomic units on both sides.

As with the image caption results, the bulk of induced morphemes are direct equivalents (e.g., (“and”, {und})). Beyond these simple one-to-one mappings, CSAR induces the polysemic relationship (“the”, {der}) and (“the”, {die}). Finally, CSAR also picks up on multi-token forms like (“Member States”, {Mitgliedstaaten}).

5 Analysis of Emergent Languages

5.1 Datasets

We apply CSAR to two different signalling game environments: one with vector-based observations and one with image-based observations.

Vector observations In the vector observation signalling game the agents directly observe one-hot vectors which directly correspond to the information to be communicated (Kharitonov et al., 2021, MIT license). Specifically, we use two variants: (1) the standard attribute–value setting where each of 4 attributes can take on 4 distinct values and (2) the “sparse” setting where there are 8 binary attributes and only attributes which are “true” are included in the meanings given to CSAR. Hyperparameters for both environments are given in Section C.1.

ShapeWorld observations The second environment is introduced by Mu and Goodman (2021, MIT license) with the following differences: (1) observations are images, and (2) employs variations which increase the level of abstraction of the game to encourage generalization. First, this environment uses the ShapeWorld tool for generating observations (Kuhnle and Copestake, 2017); namely, underlying concepts are particular shapes (e.g., red square) while the observations passed to the agents in the signalling game are pixel-based images. Second, Mu and Goodman (2021) provide three variants with increasing levels of abstraction: (1) *reference*: the sender indicates a single image, (2) *set reference*: the sender indicates a set of images with a common attribute, and (3) *concept*: as in *set reference* but the receiver's observations are different instances sharing the common attribute (referenced in Fig. 1).

5.2 Metrics

We present the following metrics to give analyze the morpheme inventories induced from the emergent language data:

Inventory size Number of morphemes in the inventory.

Inventory entropy Entropy (in bits) of the morphemes according to their prevalence.

Synonymy Entropy across forms for a given meaning.

Polysemy Entropy across meanings for a given form.

Form size Mean number of tokens in a form.

Meaning size Mean number of tokens in a meaning.

Topographic similarity Correlation (Spearman's ρ) between distances in the utterance space and complete meaning space (Brighton and Kirby, 2006; Lazaridou et al., 2018).

With the exception of inventory size and toposim, the above metrics are weighted by *prevalence* which is the proportion of records from which the morpheme was ablated.

5.3 Results

Table 1 shows the results (induced morphemes from each emergent language are given in Section E.2). Looking at form size, while the forms of morphemes do tend towards smaller values, many comprise more than one token, suggesting that assuming that each token can be analyzed as a word

	|Inv.|	Inv. H	|Form|	|Meaning|	Synonymy	Polysemy	Toposim
Vector, AV	223	6.81	3.07	1.37	1.52	0.58	0.35
Vector, sparse	156	6.09	2.08	1.55	1.91	0.62	0.39
SW, ref	1124	6.52	1.76	1.01	2.99	1.64	0.04
SW, setref	396	6.14	1.54	1.38	1.43	0.74	0.15
SW, concept	351	5.86	1.89	1.43	1.04	0.95	0.17

Table 1: Morpheme inventory metrics (described in Section 5.2) across various emergent languages. (AV: attribute–value, SW: ShapeWorld, Inv.: Inventory)

or independent unit of meaning is not a safe assumption. Addressing the mapping between forms and meanings, we see that synonymy (forms per meaning) is higher than polysemy (meanings per form). The fact that there is a higher degree of synonymy than polysemy makes sense insofar as the optimization penalizes ambiguity (polysemy) while it does not penalize merely inefficient encoding (synonymy). This is concordant with finding such as Chaabouni et al. (2019) which finds that emergent languages, in the absence of addition pressures, do not develop efficient encoding schemes.

Compositionality The meaning size metric, in particular, is interesting insofar as it relates to compositionality. In the simplest case of compositionality, morphemes comprise singleton meanings which can be combined to form compound meanings. More holistic languages, on the other hand, assign multiple atomic meanings per morpheme resulting in in larger meaning sizes. The fact that the emergent languages tend towards a meaning size of 1 suggests a non-trivial degree of compositionality under this interpretation. Yet when we compare meaning sizes values to topographic similarity values computed across records (i.e., not involving CSAR), we find that there is no obvious correlation between toposim values and meaning sizes. This could be due to the fact that individual form tokens could have "partial meanings" and need to be combined to comprise an atomic meaning. Although our sample size is too small to make any definitive claims.

6 Discussion

Due to CSAR's strong performance and easy application to a wide variety of emergent language environments, it would be a valuable addition to the standard toolkit of emergent language analyses. In particular, it helps fill a gap of environment-agnostic methods for interpreting the ways that emergent languages convey meaning—a perennial question in the field. Down the road, this opens up research questions concerning the evolution of meaning in emergent language, such as those discussed in Brighton (2003), but with the ability to deal with the larger scale and particular difficulties of *deep learning-based* emergent communication.

Furthermore, morpheme inventories are a foundation for higher-level linguistic analyses of emergent language like inducing their syntactic structure. To skip the morpheme induction step would be comparable to attempting to understand the grammatical role of the letter *C* in English. Such analyses of the syntax of emergent language and beyond are critical to understanding how emergent and human language are similar and how they are different.

7 Conclusion

CSAR presents a strong platform for investigating the morphology of emergent language, demonstrating the ability to find minimal form–meaning pairs in both procedural and human language data. Given the morpheme inventory of an emergent languages we can not only analyze phenomena like synonymy and polysemy but also the typological features of emergent languages, determining which human languages they most closely resemble, if they resemble any. Such a study of morphology forms the foundation for the more general study of the linguistic features of emergent language and unlocks the door to the insights they can provide us about human language.

8 Limitations

Greed is not always good While the greediness of CSAR does simplify induction (conceptually and implementation-wise), improve runtime, and provide good partial inventories, it suffers from the same limitation inherent to greedy algorithms: it can get trapped in local optima. For example, it is possible to construct corpora for which a greedy approach is "misled" since certain heuristics require revision based on information encountered later in the induction process (e.g., preferring smaller versus larger forms).

We did consider non-greedy approaches to morpheme induction but ultimately decided not to pursue them in this work because (1) the greedy approach itself demonstrated strong performance and (2) initial attempts at non-greedy approaches (e.g., tree search) yielded intractable runtimes. For example, an error due to greediness might select morpheme B before morpheme A because B had a higher weight while A was ultimately correct. To select A instead of B, the morpheme candidates would have to be reordered which, without an efficient way to propose these order, worsens the time complexity from $O(n^c)$ to $O(n!)$. Related algorithms use iterative approaches (IBM models and Morfessor) or search (Brighton, 2003) to avoid the local minima that trap greedy approaches. Future work could incorporate such methods to improve upon the performance of CSAR for morpheme induction.

Limited emergent language data The other limitation of this paper relates to the type and breadth of emergent language data. In terms of type, since we do not have ground truth morpheme inventories for emergent language, we cannot directly evaluate CSAR's performance on the target domain (hence the validation with procedurally generated and human languages). In terms of breadth, without a larger and more representative sample of more systematically generated data we are unable to make definitive claims about the patterns and trends of morpheme inventories in emergent languages.

References

Khuyagbaatar Batsuren, Gábor Bella, Aryaman Arora, Viktor Martinovic, Kyle Gorman, Zdeněk Žabokrtský, Amarsanaa Ganbold, Šárka Dohnalová, Magda Ševčíková, Kateřina Pelegrinová, Fausto Giunchiglia, Ryan Cotterell, and Ekaterina Vylomova. 2022. The SIGMORPHON 2022 shared task on morpheme segmentation. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 103–116, Seattle, Washington. Association for Computational Linguistics.

Ond rej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, and 2 others. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Brendon Boldt and David R Mortensen. 2024. A review of the applications of deep learning-based emergent communication. *Transactions on Machine Learning Research*.

Henry Brighton. 2003. *Simplicity as a driving force in linguistic evolution*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Henry Brighton and Simon Kirby. 2006. Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial Life*, 12(2):229–242.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Rahma Chaabouni, Eugene Kharitonov, Emmanuel Dupoux, and Marco Baroni. 2019. *Anti-efficient encoding in emergent communication*. Curran Associates Inc., Red Hook, NY, USA.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Eugene Kharitonov and Marco Baroni. 2020. Emergent language generalization and acquisition speed are not tied to compositionality. *arXiv*, 2004.03420.

Eugene Kharitonov, Roberto Dessì, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. 2021. EGG: a toolkit for research on Emergence of lanGuage in Games. `https://github.com/facebookresearch/EGG`.

Tomasz Korbak, Julian Zubek, and Joanna Rączaszek-Leonardi. 2020. Measuring non-trivial compositionality in emergent communication. *Preprint*, arXiv:2010.15058.

Satwik Kottur, José Moura, Stefan Lee, and Dhruv Batra. 2017. Natural language does not emerge 'naturally' in multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967, Copenhagen, Denmark. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Alexander Kuhnle and Ann Copestake. 2017. Shapeworld - a new test methodology for multimodal language understanding. *arXiv*, 1704.04517.

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010. Morpho challenge 2005-2010: Evaluations and results. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala, Sweden. Association for Computational Linguistics.

Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. 2015. Numba: a llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA. Association for Computing Machinery.

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. 2018. Emergence of linguistic communication from referential games with symbolic and pixel input. *ArXiv*, abs/1804.03984.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2015. Microsoft coco: Common objects in context. *arXiv*, 1405.0312.

Olaf Lipinski, Adam Sobey, Federico Cerutti, and Timothy J. Norman. 2024. Speaking your language: Spatial relationships in interpretable emergent communication. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Jesse Mu and Noah Goodman. 2021. Emergent communication of generalizations. In *Advances in Neural Information Processing Systems*, volume 34, pages 17994–18007. Curran Associates, Inc.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3:157–167.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ryo Ueda, Taiga Ishii, and Yusuke Miyao. 2023. On the word boundaries of emergent languages based on harris's articulation scheme. In *The Eleventh International Conference on Learning Representations*.

Oskar van der Wal, Silvan de Boer, Elia Bruni, and Dieuwke Hupkes. 2020. The grammar of emergent languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3339–3359, Online. Association for Computational Linguistics.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor baseline. Technical Report ISBN 978-952-60-5501-5, Aalto University, Helsinki, Finland.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, and 16 others. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

A Algorithm

A.1 Candidate generation

For simplicity's sake (and inductive bias), we limit the candidate generation functions to all non-empty substrings for forms and all non-empty subsets for meanings. Nevertheless, we could extend form candidate generation to non-contiguous forms to detect non-concatenative morphology (e.g., the form "x.z" matching "xyz" and "xwz"). In fact, we could could use arbitrary regular expressions to represent forms (or meanings) such as "^..x" or "x+" to represent absolute position and optional repetitions, respectively. We could consider empty forms and empty meanings to explicitly identify forms and meanings which do not have mappings (as opposed to implicitly not including them in the morphology).

Of course, part of the difficulty of extending the complexity of the candidate generation is that it expands the already (sometimes intractably) large search space. One method of making this tractable, though, is adding heuristics that determine which form candidates should be considered rather than considering every possible candidate.

A.2 Ambiguous pair application

In some cases of applying a morpheme to record in the dataset, there are multiple applications possible. Say we have the utterance "x y z x y" meaning

$\{A, B\}$ and we want to apply the morpheme ("x y", $\{A\}$). The form matches two substrings in the utterance, so there are two possible ways to apply the morpheme. As a heuristic for selecting the best application, CSAR break ties by selecting the substring least likely to be a morpheme (as determined by the morpheme weights). Going back to the above example, if it is the case the morpheme ("z x y", $\{B\}$) has a higher weight than ("x y z", $\{B\}$), then CSAR will apply ("x y", $\{A\}$) to the first instance of "x y" instead of the second.

This search can be very computationally expensive since it can entail going through a large number of morpheme candidates. Thus for the experiments with human language data, we do not perform this search and select the best form quasirandomly.

A.3 Heuristic optimizations

Below we include a summary of heuristic optimizations available in CSAR:

max input records Only consider a certain number of records from the input data; 20 000 for machine translation, image captions, and ShapeWorld.

max inventory size Stop after inducing a certain number of morphemes; 300 for image captions and machine translation settings.

***n*-gram semantics** Treat complete meanings as ordered and generate meaning candidates identically to forms (i.e., as n-grams); used for machine translation data where the "meanings" are sentences.

max form/meaning size Only consider form/meaning candidates up to a certain size; 3 for machine translation (form and meaning) and image captions (form only), 2 for image captions meaning.

no search best form When ablating a form with multiple matches in an utterance, do not search for best form, simply choose it randomly; no search for image captions and machine translation.

form/meaning vocabulary size Only consider the most common form/meaning candidates; 100 000 for image captions and machine translation.

token vocabulary size Only consider the most common form/meaning tokens and ignore an form meaning candidates which contain an unknown token; 1000 for image captions and 500 for machine translation.

co-occurrence threshold Zero out any co-occurrences which fall below a certain threshold (e.g., if a form and meaning candidate only occur once, treat it as never co-occurring); 1 for ShapeWorld, 10 for image captions, and 100 for machine translation.

B Empirical Validation

B.1 Procedural dataset hyperparameters

The following hyperparameters were used for generating the procedural datasets. Each dataset uses 4 attributes and 4 values except for the sparse setting which uses 8 independent values.

Synonymy $\{1, 3\}$; forms per meaning

Polysemy $\{0, 0.15\}$; proportion of meanings mapped to an already-used form

Multi-token forms $\{\{1\}, \{1, 2, 3, 4\}\}$; possible tokens per form

Vocab size $\{10, 50\}$; only applies to non-unity multi-token forms

Sparse meanings $\{\text{true}, \text{false}\}$

Distribution imbalance $\{\text{true}, \text{false}\}$; non-uniform distribution is based on the ramp function, i.e., probability of given value for an attribute is proportional to its index + 1.

Dataset size $\{50, 500\}$

Noise forms $\{0, 0.5\}$; $1 - p$ of parameter of geometric distribution

Shuffle form $\{\text{true}, \text{false}\}$

Non-compositionality $\{\text{true}, \text{false}\}$

Random seeds 3 per hyperparameter setting

Non-unity polysemy and synonymy rates for the non-compositional dataset implementation were not implemented and are excluded from the above grid.

B.2 Tokenizer vocabulary size

The heuristic for the tokenizer vocabulary size is as follows:

$$|V| = \left\lfloor \frac{|\mathcal{T}_{\text{meaning}}|}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \frac{|r_{\text{form}}|}{|r_{\text{meaning}}|} \right\rfloor + |\mathcal{T}_{\text{form}}|, \quad (6)$$

where $\mathcal{T}_{\text{meaning}}$ is the set of all meaning tokens in the dataset (likewise for $\mathcal{T}_{\text{form}}$), $\mathcal{R}$ is the multiset of records in dataset, r_{form} is the particular form (utterance) for an individual record (likewise for r_{meaning}. This heuristic can be interpreted as the mean form tokens per meaning tokens times the number unique meaning tokens added to the number of unique form tokens (since each of them will automatically be included in the vocabulary).

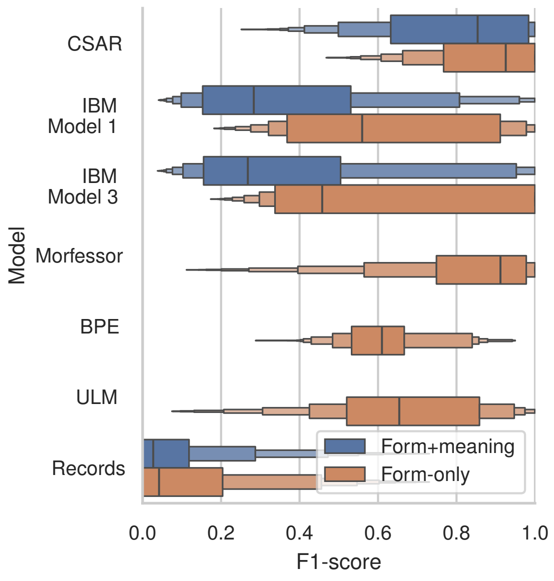

Figure 4: Exact F_1 scores of baseline methods on the procedural datasets

B.3 Additional procedural dataset results

Table 2 shows all results of baseline methods on the procedural datasets. Figure 4 visualizes the results of the baseline methods with exact F_1 score.

C Analysis of Emergent Languages

C.1 Emergent language hyperparameters

The following hyperparameters were used for the vector observation environment:

***n* values** 4, 2 (sparse)
***n* attributes** 4, 8 (sparse)
***n* distractors** 3
vocab size 32
max sequence length 10
dataset size (CSAR input) 10 000 records

The ShapeWorld observation environment uses the following hyperparameters

observations 5 shapes, 6 colors, 3 operators (and, or, not); *and* or *or* may only be used once
***n* examples** 20 total; 10 correct targets, 10 distractors
vocab size 32
max sequence length 8
dataset size (CSAR input) 20 000 records

Both environments had any beginning-of-sentence and end-of-sentence tokens removed before being fed into CSAR. Running the above experiments requires about 25 GPU-hours on NVIDIA GeForce RTX 2080Ti.

D Morfessor Results on Emergent Language

In Table 3 we show the results of running Morfessor on various emergent language corpora. Compared to the metrics for CSAR's output on the same corpora (Table 1), Morfessor's results do not match or even differ consistently (although Morfessor's forms do not have prevalence weighting like CSAR's). For the vector environments, Morfessor yields smaller inventories than CSAR yet larger inventories for ShapeWorld. Form lengths are similar for the vector environment, but for ShapeWorld, CSAR yields shorter forms than the vector environment while Morfessor yields much longer forms. Since we do not have ground truth morphemes for these emergent language corpora, we cannot definitively say one algorithm has performed better than the other. Yet Morfessor here is at a disadvantage here as it is not able to use the meanings of the utterances to guide its induction.

E Morpheme Inventories

Top 100 morphemes induced by CSAR from human and emergent language datasets.

E.1 Human languages

Morpho Challenge (“””, {+GEN}) (“ing$”, {+PCP1}) (“ed$”, {+PAST}) (“s”, {+PL}) (“er”, {er_s}) (“ly$”, {ly_s}) (“s$”, {+3SG}) (“ist”, {ist_s}) (“iz”, {ize_s}) (“ness”, {ness_s}) (“ion”, {ion_s}) (““ˆre”, {re_p}) (““ˆde”, {de_p}) (“ation”, {ation_s}) (“est$”, {+SUP}) (““ˆun”, {un_p}) (“less”, {less_s}) (“ful”, {ful_s}) (““ˆmis”, {mis_p}) (“head”, {head_N}) (“way”, {way_N}) (“ment”, {ment_s}) (“al”, {al_s}) (“it”, {ity_s}) (““ˆfire”, {fire_N}) (“ency$”, {ency_s}) (“hook”, {hook_N}) (“ish$”, {ish_s}) (“mind”, {mind_N}) (““ˆin”, {in_p}) (“at”, {ate_s}) (“if”, {ify_s}) (“able$”, {able_s}) (“ically$”, {ally_s}) (““ˆinter”, {inter_p}) (““ˆphoto”, {photo_p}) (““ˆhand”, {hand_N}) (““ˆscho”, {school_N}) (“house”, {house_N}) (“ical$”, {ical_s}) (“hold”, {hold_V}) (“long”, {long_A}) (“work”, {work_V}) (“up”, {up_B}) (“ag”, {age_s}) (“ant”, {ant_s}) (“ib”, {ible_s}) (“line”, {line_N}) (“ed$”, {ed_s}) (“er$”, {+CMP}) (““ˆover”, {over_p}) (““ˆdis”, {dis_p}) (““ˆsea”, {sea_N}) (““ˆim”, {im_p}) (“or”, {or_s}) (“pos”, {pose_V}) (“ence”, {ence_s}) (““ˆcardinal”, {cardinal_A}) (““ˆrational”, {rational_A})

	CSAR	IBM Model 1	IBM Model 3	Morfessor	BPE	ULM	Records
Exact F_1, form	0.868	0.616	0.595	0.827	0.624	0.670	0.133
Fuzzy F_1, form	0.960	0.899	0.893	0.949	0.890	0.891	0.637
Fuzzy prec., form	0.954	0.855	0.850	0.933	0.852	0.853	0.597
Fuzzy recall, form	0.967	0.952	0.946	0.967	0.934	0.938	0.701
Exact F_1	0.788	0.375	0.379	0.000	0.000	0.000	0.101
Fuzzy F_1	0.899	0.721	0.726	0.000	0.000	0.000	0.441
Fuzzy prec.	0.881	0.641	0.640	0.000	0.000	0.000	0.390
Fuzzy recall	0.921	0.855	0.866	0.000	0.000	0.000	0.543

Table 2: Results of baseline methods on the procedural datasets.

	|Inv.|	|Form|
Vector, AV	94	3.59
Vector, sparse	126	3.51
SW, ref	2898	6.93
SW, setref	2920	8.13
SW, concept	1565	7.77

Table 3: Metrics for form-only morpheme inventories generated by Morfessor across various emergent languages.

(“^shoplift”, {shop_N}) (“conciliat”, {conciliate_V}) (“^manicur”, {manicure_N}) (“^predict”, {predict_V}) (“dressing”, {dressing_V}) (“^buffet”, {buffet_V}) (“^crimin”, {crime_N}) (“^entitl”, {entitle_V}) (“^frivol”, {frivolous_A}) (“^heartb”, {heart_N}) (“^maroon”, {maroon_A}) (“^ribald”, {ribald_A}) (“^spread”, {spread_V}) (“^squeak”, {squeak_V}) (“^squint”, {squint_V}) (“^statue”, {statue_N}) (“^summar”, {summary_A}) (“whisper”, {whisper_V}) (“^blink”, {blink_V}) (“^carri”, {carry_V}) (“^cheer”, {cheer_V}) (“^four-”, {four_Q}) (“^hitch”, {hitch_V}) (“^louvr”, {louvre_N}) (“^muzzl”, {muzzle_N}) (“^nihil”, {nihilism_N}) (“^tooth”, {tooth_N}) (“^waist”, {waist_N}) (“guard\$”, {guard_N}) (“^bull”, {bull_N}) (“^rail”, {rail_V}) (“^seri”, {series_N}) (“^test”, {test_N}) (“^two-”, {two_Q}) (“ance\$”, {ance_s}) (“board”, {board_N}) (“chain”, {chain_N}) (“eroom”, {room_N}) (“grand”, {grand_A}) (“order”, {order_V}) (“power”, {power_N})

Image captions (“tennis”, {tennis racket}) (“cat”, {cat}) (“train”, {train}) (“dog”, {dog}) (“pizza”, {pizza}) (“toilet”, {toilet}) (“man”, {person}) (“bus”, {bus}) (“clock”, {clock}) (“baseball”, {baseball glove}) (“frisbee”, {frisbee}) (“bed”, {bed}) (“horse”, {horse}) (“skateboard”, {skateboard}) (“laptop”, {laptop}) (“cake”, {cake}) (“giraffe”, {giraffe}) (“table”, {dining table}) (“bench”, {bench}) (“motorcycle”, {motorcycle}) (“bathroom”, {sink}) (“elephant”, {elephant}) (“umbrella”, {umbrella}) (“kitchen”, {oven}) (“kite”, {kite}) (“people”, {person}) (“ball”, {sports ball}) (“sheep”, {sheep}) (“zebra”, {zebra}) (“phone”, {cell phone}) (“surfboard”, {surfboard}) (“hydrant”, {fire hydrant}) (“zebras”, {zebra}) (“teddy”, {teddy bear}) (“truck”, {truck}) (“stop sign”, {stop sign}) (“sandwich”, {sandwich}) (“boat”, {boat}) (“street”, {car}) (“bat”, {baseball bat}) (“bananas”, {banana}) (“giraffes”, {giraffe}) (“living”, {couch}) (“snow”, {skis}) (“bird”, {bird}) (“elephants”, {elephant}) (“vase”, {vase}) (“cows”, {cow}) (“broccoli”, {broccoli}) (“computer”, {keyboard}) (“woman”, {person}) (“tie”, {tie}) (“horses”, {horse}) (“bear”, {bear}) (“desk”, {mouse}) (“plane”, {airplane}) (“luggage”, {suitcase}) (“airplane”, {airplane}) (“person”, {person}) (“hot”, {hot dog}) (“refrigerator”, {refrigerator}) (“wii”, {remote}) (“kites”, {kite}) (“boats”, {boat}) (“couch”, {couch}) (“traffic”, {traffic light}) (“plate”, {fork}) (“surf”, {surfboard}) (“umbrellas”, {umbrella}) (“wine”, {wine glass}) (“skate”, {skateboard}) (“bowl”, {bowl}) (“stuffed”, {teddy bear}) (“room”, {tv}) (“cow”, {cow}) (“scissors”, {scissors}) (“snowboard”, {snowboard}) (“chair”, {chair}) (“car”, {car}) (“banana”, {banana}) (“bicycle”, {bicycle}) (“birds”, {bird}) (“vegetables”, {broccoli}) (“microwave”, {microwave}) (“donuts”, {donut}) (“video”, {remote}) (“batter”, {baseball bat, person}) (“skateboarder”, {person, skateboard}) (“surfer”, {person, surfboard}) (“skis”, {skis}) (“motorcycles”, {motorcycle}) (“meter”, {parking meter}) (“suitcase”, {suitcase}) (“sink”, {sink}) (“bike”, {bicycle}) (“chairs”, {chair}) (“food”, {bowl}) (“dogs”, {dog}) (“oven”, {oven}) (“court”, {sports ball})

Machine translation (“and”, {und}) (“Commission”, {Kommission}) (“not”, {nicht}) (“Union”, {Union}) (“we”, {wir}) (“I”, {ich}) (“that”, {daß}) (“Mr”, {Herr}) (“I”, {Ich}) (“Parliament”, {Parlament}) (“President”, {Präsident}) (“Member States”, {Mitgliedstaaten}) (“report”, {Bericht}) (“European”, {Europäischen}) (“We”, {Wir}) (“or”, {oder}) (“in”, {in}) (“Europe”, {Europa}) (“the”, {der}) (“Council”, {Rat}) (“between”, {zwischen}) (“is”, {ist}) (“2000”, {2000}) (“Commissioner”, {Kommissar}) (“EU”, {EU}) (“for”, {für}) (“the”, {die}) (“The”, {Die}) (“also”, {auch}) (“with”, {mit}) (“like to”, {möchte}) (“you”, {Sie}) (“1999”, {1999}) (“directive”, {Richtlinie}) (“only”, {nur}) (“proposal”, {Vorschlag}) (“European”, {Europäische}) (“Madam”, {Präsidentin}) (“Mrs”, {Frau}) (“Kosovo”, {Kosovo}) (“but”, {aber}) (“new”, {neuen}) (“Group”, {Fraktion}) (“have”, {haben}) (“behalf”, {Namen}) (“Mr”, {Herrn}) (“women”, {Frauen}) (“has”, {hat}) (“regions”, {Regionen}) (“years”, {Jahren}) (“all”, {alle}) (“two”, {zwei}) (“cooperation”, {Zusammenarbeit}) (“if”, {wenn}) (“1”, {1}) (“new”, {neue}) (“Article”, {Artikel}) (“because”, {weil}) (“whether”, {ob}) (“Parliament”, {Parlaments}) (“a”, {eine}) (“measures”, {Maßnahmen}) (“but”, {sondern}) (“institutions”, {Institutionen}) (“social”, {sozialen}) (“to”, {zu}) (“political”, {politischen}) (“development”, {Entwicklung}) (“national”, {nationalen}) (“today”, {heute}) (“countries”, {Länder}) (“European”, {europäischen}) (“must”, {muß}) (“our”, {unsere}) (“as”, {wie}) (“problems”, {Probleme}) (“initiative”, {Initiative}) (“work”, {Arbeit}) (“be”, {werden}) (“very”, {sehr}) (“human rights”, {Menschenrechte}) (“of the”, {des}) (“us”, {uns}) (“three”, {drei}) (“debate”, {Aussprache}) (“other”, {anderen}) (“hope”, {hoffe}) (“already”, {bereits}) (“question”, {Frage}) (“this”, {diesem}) (“debate”, {Debatte}) (“are”, {sind}) (“will”, {wird}) (“proposals”, {Vorschläge}) (“If”, {Wenn}) (“Prodi”, {Prodi}) (“Council”, {Rates}) (“rapporteur”, {Berichterstatter}) (“INTERREG”, {INTERREG}) (“role”, {Rolle})

E.2 Emergent languages

Vector, attribute–value Note that meanings are in the format *attribute_value* meaning that 1_2 means the 1st attribute has value 2.

(“15”, {3_3}) (“25 25”, {3_0}) (“3”, {2_3}) (“20 20”, {0_3, 1_0}) (“7 7”, {0_3, 1_3}) (“4”, {2_0}) (“16 16 16 16 16 16 16 16 16”, {0_3, 3_0}) (“2”, {0_0, 2_0}) (“13 13 13 13 13 13”, {2_0}) (“23 23 23 23 23 23 23”, {0_0, 2_3}) (“28”, {0_0, 1_3}) (“27 27”, {1_0}) (“17 17 17”, {0_0}) (“31”, {2_0, 3_2}) (“22 22 22 22 22”, {1_3}) (“22 25 25 25 25”, {0_1, 1_3}) (“30 30”, {2_1}) (“22 22 13”, {1_3, 3_3}) (“26 26 26 26 26 26 26 26”, {1_3, 2_0, 3_0}) (“15”, {3_1}) (“15 27 27 27 27 27 27 27”, {0_1, 3_2}) (“8”, {0_0}) (“3 3 3 30 30”, {0_1, 1_1, 2_2}) (“16 16”, {3_0}) (“3 3 3 3 30”, {0_2, 1_2, 2_2}) (“7 7”, {0_2, 3_1}) (“15 3 3 3 3”, {0_2, 3_2}) (“15 7 20 27 27 27 27 27 27 27”, {0_2, 1_1, 2_1, 3_2}) (“20 27 27 27 27 27 27”, {0_2, 3_2}) (“15 15 15 3 27 27 27 27 27 27”, {0_2, 1_1, 2_2, 3_2}) (“22 22 22 22 22 30 30 30 30 30”, {0_1, 1_2, 2_2, 3_2}) (“8 1 23”, {1_1, 3_0}) (“22 22 22 25 3 30 30 30 30 30”, {0_1, 1_2, 2_2, 3_1}) (“28 28 2 2 2 2 2 2”, {1_2, 3_1}) (“22 22 22 17 17 17 17”, {1_2, 3_3}) (“26 26 6 4 4 4”, {0_1, 3_0}) (“23”, {1_0, 2_3}) (“15 15 15 15 15 15 15 15 15”, {1_2, 2_3}) (“22 22 22 3”, {0_1, 1_2}) (“7 7 20 20 20 20 20 20 20 20”, {1_1}) (“7 7 7 7 7 20 7”, {1_2, 3_2}) (“31 31”, {0_3, 3_3}) (“28 28 28 8 8 8 12 12 12 12”, {1_2, 2_1, 3_0}) (“15 15 15 15 15 17 17 17 17”, {0_1, 1_2, 2_2}) (“3 27 27 27”, {2_2}) (“7 15 15 15 15 15 15 15 15”, {0_3, 1_2, 2_2}) (“3 3 3 3 3 3 3 3 3 3”, {0_2, 1_0, 3_0}) (“7 13”, {0_1, 1_2, 3_2}) (“28 26”, {3_0}) (“15 15 7”, {0_2, 1_3, 2_2}) (“22 22 23 23 23 23 23 23 23 23”, {1_1, 2_2, 3_1}) (“7”, {1_2}) (“13 13 13”, {3_3}) (“7 7 7 7”, {3_2}) (“15 17 17 17 17 17”, {1_1}) (“22 22 22 17”, {1_2}) (“15 15 15 13 13 13 13 13 17”, {0_1, 1_2, 2_1}) (“3 3 3 3 3 23 23 23”, {0_1, 1_1, 3_1}) (“7 16 16 16 16 4”, {0_3, 1_1, 3_1}) (“26 26 26 26 6”, {1_2, 3_0}) (“22 22 22 22”, {2_2, 3_2}) (“25 25 25 25 25”, {2_2}) (“7 25 25 25 25”, {0_2, 1_3, 2_3}) (“22 7 26 13”, {0_1, 1_3, 3_2}) (“15 15 15 15 17”, {0_1, 1_2}) (“23 23 17”, {2_2, 3_2}) (“7 26 26 26 26 26”, {0_2, 1_3, 2_1}) (“8 8 8 23 23 23”, {1_2, 2_2, 3_1}) (“7 26 26 26 26 26 26 26”, {2_0, 3_1}) (“22 7 26 26 26 26 28 28 28 28”, {0_1, 2_1, 3_1}) (“15 15 15 15 15 15 15 15”, {0_2, 2_3}) (“5 4 4 4 4”, {0_2, 1_0, 3_1}) (“26 26 26”, {2_0, 3_0}) (“22 13 13 13 13 13 13 13 2”, {1_2, 3_2}) (“15 15 31 31 31 31 31 31 31”, {0_2, 1_1, 2_1}) (“22 28 28 28 28 28 28 28 28”, {2_1, 3_1}) (“15”, {1_1}) (“13 13 13 13 2 2 2 2”, {1_1, 3_2}) (“1 1 1 1 1 1”, {1_2, 2_3}) (“8 8 30 30”, {1_1, 3_0}) (“4 4 4 27 27”, {0_1, 3_1}) (“17 17 17 17”, {1_0, 3_3}) (“23 23 23 23 23 23 23 27”, {2_2}) (“15 31 31”, {0_2, 2_1}) (“5 27 27 27 27 27”, {0_2, 2_1}) (“22”, {0_0, 2_3}) ("28 8 8 8 8", {2_2, 3_0}) ("17 2 2 2 2 2", {2_1, 3_2}) ("22 22 2", {1_1, 3_1}) ("3 3 23 23 23", {0_1, 3_1}) ("28 28 28 28", {2_1}) ("26 26 26 4 4 4 4 4", {0_1, 3_1}) ("17 17 27 27 27", {2_1, 3_2}) ("15 13 13 13 13", {1_2, 2_1}) ("25 3 25 3 25", {0_1, 1_2}) ("20 20 20 27 27 27 27 27 27 27", {2_1, 3_1}) ("3 3 3 3", {0_2}) ("31 31 31 31 31 31 31 31 31 31", {1_0}) ("17 13", {0_0, 2_1}) ("3 3 3 3 3", {1_0, 3_0})

Vector, bag of meanings ("22", {4, 7}) ("24", {0, 3, 6}) ("16", {1, 5}) ("11", {6}) ("26", {0, 4, 6}) ("1 1", {3}) ("6", {0, 7}) ("17", {5, 7}) ("28 28", {0, 2}) ("18", {0, 2}) ("21 21", {0, 6}) ("14 14 14 14 14", {1, 3, 6, 7}) ("16", {4, 7}) ("24", {1, 5}) ("1 28", {3, 4}) ("31 31", {4, 7}) ("12", {4}) ("3", {4, 5}) ("22 22 22", {0, 3, 6}) ("4", {3}) ("28", {2}) ("12 12 12 12 12", {2}) ("28 27 27 27 27 27 27", {0, 4}) ("28 9", {0, 3, 4}) ("11 11 11", {3}) ("7", {3}) ("12 12 12 12", {2}) ("24 5", {4}) ("30 30", {1, 7}) ("14 14 14 14", {1, 2, 7}) ("25 25 25 25 25 25", {1}) ("4 5 5 5 5", {2, 4}) ("26", {2}) ("25 25 27 27 27 27 27", {0, 5}) ("1 29 29 29 29 29 29 29 12 12", {1, 3, 6}) ("5 5 5 5 5 5 5 5", {4}) ("1 1 1 1 1 1 1", {1, 7}) ("1 12 12 12 12 12 12 12 12", {1, 3}) ("6 6 6 6", {2}) ("1 1 1 1", {0, 1}) ("1 30", {2}) ("1 18", {3, 5}) ("23", {7}) ("16 16 16 16", {0}) ("21 21 21 21", {7}) ("5 5 5 27 27", {1}) ("4 5", {4}) ("5 12", {2, 3}) ("1 12", {3, 5}) ("18 27", {5}) ("11 22 22", {2, 3}) ("22", {6}) ("22 22 11", {1, 2}) ("4 4 27 27", {5}) ("1 29 29", {3}) ("16 16", {0}) ("12", {1, 3}) ("9 9 9 9 9 9 9 9", {0}) ("6 1 1 1", {1, 2}) ("20 20 20 20 20 20 20 20", {1, 7}) ("26 26 26 26 28 28 28", {1, 7}) ("29 29 4 12 12 12 12 12 12", {5}) ("1 1", {1, 2}) ("29 29 4 4 4 4 4 4 4 4", {1, 6}) ("11 11 23 23 23", {3}) ("12 12 12", {2}) ("22 22", {3}) ("28 12", {0}) ("21 21 21 8", {2}) ("1 4 4", {2}) ("21 21 23", {2}) ("21 21 21 21 21 21 21 21 6", {1, 2}) ("12 16", {2}) ("10 10 10 10 10 10 10 25 25 25", {2, 7}) ("28 28 28", {1, 4}) ("24 24 24 24 24 24 24 24", {7}) ("21 21 6 6", {2}) ("21 21", {2}) ("9 5", {4}) ("31 31 31 31", {3}) ("28 28 28 28 28 28 28", {3}) ("10 10 10 10 10", {7}) ("13 13 13", {7}) ("11 14 14 22 22 22 17 17 17", {2}) ("7 7 7 7", {2}) ("22 26 26 5 5 5 5 5 5", {1}) ("22 22 22", {2, 5}) ("11 11", {7}) ("12 12 27 27 27 27 27", {2}) ("30 30 30 30 30 30 27 27 27 27", {5}) ("2 9 12 12", {2}) ("11 11 23 17 17 17", {3}) ("18 18 18 18 18 18 28 18", {1}) ("25", {0}) ("23 23 23 23 19 19 19 19", {1}) ("6 6 10 10 10 10 10 10 10", {3}) ("6 6 6 6 6 6 6 17 17", {3}) ("24 24 24 4 4 12 12", {2}) ("22 22 22 22 28 28 5", {1}) ("24 4 4 4 4 4 4 4 4 4", {2})

Shapeworld, reference ("29", {ellipse}) ("29", {gray}) ("29", {green}) ("29", {rectangle}) ("29", {triangle}) ("29", {white}) ("30 2", {ellipse}) ("30", {square}) ("29 29", {blue}) ("18 4 18", {white}) ("29 29", {circle}) ("5 3", {square}) ("5 3", {rectangle}) ("6", {square}) ("24 18", {ellipse}) ("11 4", {white}) ("30 5", {triangle}) ("2 2 2 2 2", {white}) ("29", {yellow}) ("11", {ellipse}) ("2 2 3", {ellipse}) ("4", {rectangle}) ("30 30 3", {ellipse}) ("2", {square, yellow}) ("30", {circle}) ("2 2 2 2 2", {ellipse}) ("3", {gray}) ("23 5", {rectangle}) ("4", {green}) ("13 6 13 2", {ellipse}) ("23 18 23 23", {white}) ("3", {rectangle}) ("18 3 18", {white}) ("2 2 5", {ellipse}) ("24 6 24 6", {white}) ("6 2", {ellipse}) ("3", {green}) ("13 13 23", {square}) ("24 24", {white}) ("18 18 18 23 18", {white}) ("30 5 2", {ellipse}) ("23 24 23", {ellipse}) ("18 4 18 5", {ellipse}) ("4 18 4 5", {yellow}) ("13", {gray}) ("18 5 4 18", {ellipse}) ("2 18", {white}) ("4 5 4", {ellipse}) ("18 18", {yellow}) ("23 13 23", {square}) ("6 3", {triangle}) ("23 3 23 3 23", {yellow}) ("13 6 13 24", {ellipse}) ("24 24", {yellow}) ("13 13 24 13 13 13 24", {ellipse}) ("6 3", {circle}) ("23 18 23", {ellipse}) ("2 2 23 2", {white}) ("5 23", {green}) ("30 30", {red}) ("18 5 4", {yellow}) ("3 23 3 3 3", {square, yellow}) ("23 24 23", {white}) ("18 18 3 18", {ellipse}) ("3 3 3 3", {square}) ("24 6 13 6", {white}) ("30 6 30 3", {ellipse}) ("18 3 18", {yellow}) ("5 30", {blue}) ("24 2", {ellipse}) ("24 13 24 13 13", {square, white}) ("23 18 23", {square, yellow}) ("4 18 18 4", {ellipse}) ("2 2 3", {white}) ("13 6 13", {ellipse}) ("13 13 24", {white}) ("18 23 18", {white}) ("13", {rectangle}) ("13 13 24 13 13", {ellipse}) ("30 24 30", {white}) ("13 13 13 13", {white}) ("23 3 23 3 23", {ellipse}) ("5 18 18 5", {ellipse}) ("24", {green}) ("13 13 23 13 13 13 24", {circle, red}) ("30 6 30 6", {blue}) ("5 5 4", {ellipse}) ("13 24", {blue}) ("5", {circle, red}) ("30 6 30 2", {white}) ("3 3 3 3 3 3", {yellow}) ("18 5 18 5 18", {ellipse}) ("3 3 3", {white}) ("18", {square}) ("18 5 18 4 2", {ellipse}) ("13 2", {ellipse}) ("3 3 23 3 3 23 3 3 2", {circle, red}) ("18 3 18 5 18 3", {ellipse}) ("4 4", {blue}) ("13 24 13", {yellow})

Shapeworld, set reference ("3 3", {circle, not}) ("21 21", {circle}) ("23 23", {gray, not}) ("20 20", {blue, not}) ("5 5", {and, green, not}) ("28", {square}) ("26", {or, yellow}) ("28", {ellipse, not}) ("4 4", {white}) ("11", {not, rectangle}) ("11 11", {ellipse}) (“25 25 25”, {blue, red}) (“25 4”, {blue}) (“3 28”, {triangle}) (“22 26”, {not, red}) (“25 23”, {green, or, red}) (“5 4 5”, {gray, or, white}) (“12 23”, {yellow}) (“12 18”, {or, red, white}) (“23 25”, {and, gray, white}) (“5 20”, {gray, or, red}) (“4 23”, {and, red}) (“12 12”, {yellow}) (“3 11”, {and, circle}) (“21”, {circle, or}) (“28”, {rectangle}) (“23 26 23”, {green}) (“26 20”, {and, white}) (“11”, {ellipse}) (“21 22”, {and, triangle}) (“22 5”, {blue, green}) (“28 25”, {triangle}) (“5 26 5”, {gray}) (“3”, {and, circle}) (“25 20”, {white}) (“25 26 4”, {blue}) (“28 3”, {triangle}) (“21”, {square}) (“12”, {yellow}) (“11 22”, {triangle}) (“12 25”, {or, red, white}) (“21”, {and}) (“3”, {square}) (“20 12 20 12”, {blue, not}) (“20 22 22”, {or, triangle, white}) (“18”, {rectangle}) (“5 5”, {gray}) (“5”, {red}) (“23 22”, {red}) (“23 23”, {green}) (“26 22”, {and, white}) (“3”, {rectangle}) (“20 5 5”, {white}) (“22”, {or}) (“5 5”, {triangle}) (“12 12 23”, {not}) (“27”, {triangle}) (“12 5”, {green, not}) (“25 23 20”, {and, gray, white}) (“25 21 4”, {blue}) (“4 25 12”, {blue, or}) (“22 22”, {triangle}) (“20 3 20”, {white}) (“22 20”, {blue, not}) (“12 22”, {triangle}) (“28 26”, {triangle}) (“21 23”, {green}) (“20 20 20”, {and, white}) (“25”, {blue}) (“28 20”, {triangle}) (“22 22”, {not}) (“5”, {gray}) (“12”, {or}) (“5 11”, {and, green}) (“3”, {triangle}) (“4 22”, {red}) (“23”, {not}) (“23 4”, {green}) (“18”, {triangle}) (“27”, {rectangle}) (“12 20 12 20 23”, {and}) (“20 22”, {blue}) (“25 23”, {and, gray}) (“4 12 23”, {and}) (“23 3 23”, {green}) (“20 22 20”, {white}) (“12 20 23”, {and}) (“12 20 12 20”, {and}) (“8 5 12”, {and, not}) (“23 11 23”, {green}) (“23 20”, {white}) (“28 4 28 4 25”, {and, triangle, white}) (“21”, {triangle}) (“25 5 20”, {white}) (“22 26”, {gray}) (“28 4”, {triangle}) (“26 18”, {and}) (“5 4 18”, {white}) (“12 20 12”, {and}) (“28”, {or})

Shapeworld, concept (“3 6”, {gray, not}) (“7 7”, {blue, not}) (“32”, {circle}) (“4 5”, {not, yellow}) (“6 12 6 12”, {green, or, yellow}) (“3 12 3”, {blue, or, yellow}) (“3 7 3 3”, {green, white}) (“6 6”, {not, red}) (“4 7 4”, {green, or, red}) (“6 28 6 28”, {or, white, yellow}) (“25 5 25”, {blue, or, white}) (“32 32 32”, {ellipse}) (“5 5 5 5 5 5 5”, {green, not, yellow}) (“25 25 25 25 25”, {green, not, red}) (“3 4 3”, {and, white, yellow}) (“28 28 28 28”, {white}) (“32”, {rectangle}) (“5 3 5”, {blue, red}) (“12 28”, {yellow}) (“22”, {square}) (“22”, {triangle}) (“7 28”, {or, red, yellow}) (“3 6 3”, {white}) (“5 32”, {or, red, white}) (“28 28 5”, {gray, or, white}) (“7 28 7 28 7 28 7”, {blue, green}) (“3 31 3”, {blue}) (“12 4 12”, {and, green}) (“28 3”, {gray}) (“5 6 5 6 5”, {and, red, yellow}) (“25 7 25”, {green, not, or}) (“7 5 7 5 7 5 7 5”, {blue, not, yellow}) (“4 4 4”, {not, or, yellow}) (“22”, {and, ellipse, not}) (“6 12 6 6”, {and, gray}) (“31 31 31 31”, {and, blue}) (“7 12 7”, {and, white}) (“5 5 3 28 7”, {gray, or, red}) (“28 28 31”, {gray}) (“25 3”, {red, white}) (“7 6 7 6 7”, {blue, not, or}) (“32”, {triangle}) (“22 22”, {or, rectangle}) (“12 12 32”, {and, yellow}) (“5 28”, {red}) (“4 6 4”, {or}) (“5 7 5 7”, {and, yellow}) (“32”, {and, square}) (“4 12”, {green}) (“3 3 3 3”, {and, not}) (“3 3 6”, {and}) (“32 32”, {ellipse, or}) (“6 5 5”, {and, gray}) (“7”, {or, red}) (“4 3”, {and, gray}) (“12 12 12”, {yellow}) (“28 7 28 7”, {and, green}) (“25 25 25”, {and}) (“28 3 3”, {or}) (“23 23”, {blue, not}) (“4 4 32 4 32 4”, {not, or, yellow}) (“4 3 4 3”, {not}) (“6 4”, {or}) (“25 7 32”, {green, not}) (“31”, {blue, or}) (“4”, {not}) (“5 5 5 25”, {green, not, yellow}) (“28 31 28 31”, {gray}) (“12 22 12”, {rectangle, yellow}) (“3 3 3”, {and, not}) (“5 3 5”, {or}) (“28”, {or, white}) (“5 5 3”, {gray}) (“31 28”, {red}) (“6 25”, {not, red}) (“32 27 27”, {ellipse, yellow}) (“7 6 7 32 7”, {blue, not}) (“32 6”, {not}) (“4”, {green, or}) (“12 6 28”, {yellow}) (“32”, {ellipse}) (“27 27”, {yellow}) (“25 32”, {not}) (“3 6 7”, {white}) (“28 28 12 3”, {yellow}) (“4”, {circle}) (“27 32 27”, {yellow}) (“3 4”, {and, gray}) (“6 7 6”, {square}) (“6 12 25”, {red}) (“23 22 23”, {blue}) (“6 3”, {white}) (“3 3 7”, {green, white}) (“3 5 5”, {or}) (“5 27 32”, {or, red, white}) (“7”, {blue, not}) (“5”, {and, yellow}) (“7 12”, {and, white}) (“28”, {gray}) (“3 25 3”, {not})