Draw Me a Flower: Processing and Grounding Abstraction in Natural Language

Royi Lachmy[1,2] **Valentina Pyatkin**[1,2] **Avshalom Manevich**[1] **Reut Tsarfaty**[1,2]
[1]Bar-Ilan University, Ramat Gan, Israel
[2]Allen Institute for Artificial Intelligence, Tel Aviv, Israel
{royi.lachmy,reut.tsarfaty}@biu.ac.il
{valpyatkin,avshalomman}@gmail.com

Abstract

Abstraction is a core tenet of human cognition and communication. When composing natural language instructions, humans naturally evoke abstraction to convey complex procedures in an efficient and concise way. Yet, interpreting and grounding abstraction expressed in NL has not yet been systematically studied in NLP, with no accepted benchmarks specifically eliciting abstraction in NL. In this work, we set the foundation for a systematic study of processing and grounding abstraction in NLP. First, we deliver a novel abstraction elicitation method and present HEXAGONS, a 2D instruction-following game. Using HEXAGONS we collected over 4k naturally occurring visually-grounded instructions rich with diverse types of abstractions. From these data, we derive an *instruction-to-execution* task and assess different types of neural models. Our results show that contemporary models and modeling practices are substantially inferior to human performance, and that model performance is inversely correlated with the level of abstraction, showing less satisfying performance on higher levels of abstraction. These findings are consistent across models and setups, confirming that abstraction is a challenging phenomenon deserving further attention and study in NLP/AI research.

1 Introduction

As human–computer interaction in natural language (NL) becomes more and more pervasive (e.g., via smart devices and chatbots), a cognitive phenomenon known as *abstraction*, which is prevalent in human communication and cognition, is taking a central role in the way users communicate their intentions and needs to artificial agents.

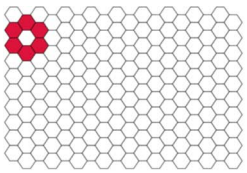
When communicating in NL with a human or an artificial agent, a human may issue a request for a single action such as ''send an email'' or ''set my alarm''. However, when engaging with more complex tasks that require multiple actions, humans often evoke *abstraction* in order to communicate their intentions in an economic-yet-precise way. Examples for evoking abstraction when issuing a complex request that consists of multiple actions may be: ''Schedule a group meeting every other Wednesday until the end of the year, unless there are holidays.'' For an autonomous car, an envisioned request might be ''Circle the block looking for a shady parking spot, park at the first spot you see. Try this four times, and one more time allowing for non-shady parking. In no luck, try the adjacent block.'' In fact, even the individual request ''send an email'' is in itself an abstraction over a sequence of multiple individual actions such as: ''open your inbox, click New Mail, select a recipient,'' and so forth.

Abstraction is defined by Wing (2011) as ''[letting] one object stand for many. It is used to capture essential properties common to a set of objects while hiding irrelevant distinctions among them''. In the calendar example, a single utterance references multiple meetings in multiple days. Likewise, in the autonomous car example, the speaker evokes some sort of control structure in order to iterate a process several times.

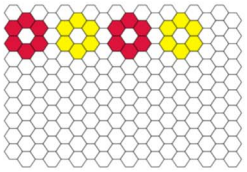
Abstraction so construed is both critical and pervasive in NL. Referring to multiple instances at once may be done by means of the shape they form, via a process that iterates them, or via an action/condition applied to select or manipulate them. To illustrate this, Figure 1 showcases an example of a natural language procedure for drawing a target image (the bottom-right image) on an empty board. The instructions start with the construction of an object, *a red flower*, covering six tiles. Then, the Instructor prescribes multiple flower patterns via *repeat* actions that realize a nested 'loop'. Finally, the instructor states the color of the flower centers via a 'condition' (*green for red, blue for yellow*). This example goes to show both the essence and power of abstraction in NL; here,

Transactions of the Association for Computational Linguistics, vol. 10, pp. 1341–1356, 2022. https://doi.org/10.1162/tacl_a_00522
Action Editor: Christopher Potts. Submission batch: 4/2022; Revision batch: 7/2022; Published 11/2022.

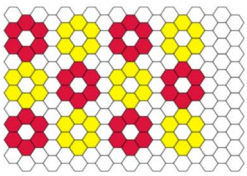
1. Make a *red flower*, by coloring in red *all tiles adjacent* to the 2nd tile from the top in the 2nd column from the left.



2. *Repeat* this *flower* pattern *across the board* to the right, *alternating* yellow and red, leaving a blank column *between every 2* flowers.



3. *Repeat* this *row of flowers* 2 more times, but *reverse* the colors *in each new row*. You should get 6 red flowers and 6 yellow flowers *in total*.



4. Paint the *centers* of the flowers using *green for red flowers* and *blue for yellow flowers*.

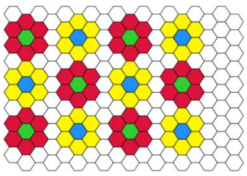


Figure 1: Abstraction in the HEXAGONS Game. On the left are instructions for drawing the target image (bottom-right), paired with their grounding on the HEXAGONS board on the right. *Italics* mark expressions of abstraction as referring to, e.g., objects (a flower), iterations, and conditions.

merely four NL utterances suffice to prescribe a complex image over a 180-tiles board.

Despite its importance and widespread use, detecting and grounding abstraction in NL has not yet been systematically studied in NLP. Previous studies on grounding instructions target linguistic phenomena as anaphora and ellipsis (Long et al., 2016), spatial relations (Jayannavar et al., 2020; Bisk et al., 2016a), and referring expressions (Haber et al., 2019) but do not specifically elicit abstraction. In studies on navigation (Anderson et al., 2018; Chevalier-Boisvert et al., 2018; Misra et al., 2018) eliciting abstract statements is also sparse. Instructions often refer to specifics of the environment rather than abstract phenomena.

In this work we aim to add a new facet to the study of natural language understanding, that of interpreting abstraction. We set out to provide a foundation for systematically studying the phenomenon of processing and grounding diverse levels of abstraction found in naturally occurring NL utterances. Achieving this goal is far from trivial. As is standard in NLP, we would first need to establish an appropriate dataset for studying this phenomenon. Specifically, we'd like to collect naturally occurring data that manifest abstraction. But how can we purposefully request for the presence of abstraction in naturally-occurring data?

To overcome this challenge, we develop an abstraction elicitation protocol by adopting practices from STEM education, specifically from *Computational Thinking* (CT) research (Wing, 2011; Grover and Pea, 2013). The idea, in a nutshell, is to develop visual stimuli that evoke, and thus cultivate (and elicit), higher-order thinking, which is then narrated in NL. We implement the proposed protocol in a novel HEXAGONS game, a situated collaborative game where an Instructor provides instructions that should be grounded and executed in a virtual world (Long et al., 2016; Bisk et al., 2016a; Kim et al., 2019; Jayannavar et al., 2020). In contrast to previous studies, we use practices from CT research to design visual triggers of abstraction. Hence, on the one hand, we implicitly call for the presence of abstraction in the instructions, but on the other hand, we provide naturally elicited abstract instructions from workers not possessing formal knowledge of what abstraction is.

Using the HEXAGONS game and the task stimuli, we collected over 4k human instructions manifesting a variety of formal abstractions (objects, control structures, and functions) expressed naturally and intuitively in NL and grounded on the HEXAGONS board. To showcase how this data may be used for studying abstraction processing in NL, we derive an *instruction-to-execution* task, where the model needs to ground and execute NL instructions on the HEXAGONS board. We propose a naïve rule-based baseline as well as two neural modeling alternatives—one based on classification, one on generation—and assess their performance on the elicited abstraction data.

Our experiments show that, while our models perform better than the naïve rule-based baseline, they are substantially inferior to human performance. Moreover, we show that model performance is inversely correlated with the level of abstraction, that is, the models execute concrete instructions quite well, but perform poorly on higher-level abstractions. This holds across

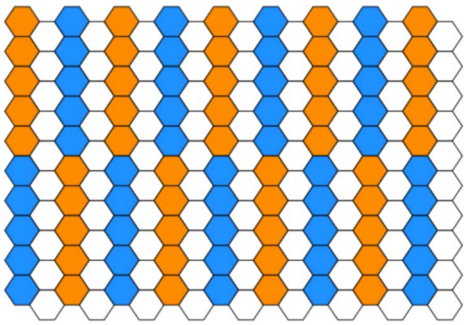
different models, setups, task conditions, amount and type of training data, and board contexts. We further observe that the instruction's history is another important factor in models' performance; the longer the history, the better the performance.

The contribution of this paper is thus manifold. First, we bring to the fore of NLP research a critical aspect of human–computer communication, namely, the ability to detect, process, and ground abstraction in natural language. Next, we devise a novel abstraction elicitation methodology and deliver the HEXAGONS dataset as a novel benchmark to explore the automatic processing of different levels of abstraction. This dataset may also serve broader communities such as AI researchers, linguists, cognitive psychologists, and STEM educators in the study of human processing of abstraction. Finally, for the instruction-to-execution task we derive from the HEXAGONS data, we show experimental evidence that unequivocally confirms that abstract instructions in NL are indeed more challenging for current systems to process, and we expose abstraction as an important and challenging dimension for further study in NLP.[1]

2 The Challenge: Eliciting and Processing Abstraction in NL

Abstraction is a cognitive phenomenon related to diverse human activities such as learning, decision making, and behavior regulation (cf. Burgoon et al., 2013). In the context of human-computer interaction, and in general in human problem-solving, abstraction is said to be one of a set of cognitive skills known as *Computational Thinking* (CT) skills, defined by Cuny, Snyder, and Wing, (2010) as ''the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.'' Abstraction in this context refers to a process of information reduction (Burgoon et al., 2013), where multiple instances are conceived as arising from a single object, ''consisting of their shared properties while discarding irrelevant distinctions'' (Wing, 2011).

Abstraction is considered by many as the most important CT skill, allowing a human to think in terms of objects and concentrating on their essential features, while ignoring irrelevant details (Dijkstra, 1972; Denning et al., 1989; Koppelman and Van Dijk, 2010; Wing, 2011, 2017). Thus, abstraction leads to a speaker's capacity for being more precise and less error-prone (Dijkstra, Accessed 1 May 2021; Haberman, 2004) and to designing more concise, elegant and efficient solutions (Ginat and Blau, 2017).



1. In the first column paint the first 5 tiles downward orange
2. In the first column paint the last 5 tiles downward blue
3. In the 3rd column paint the first 5 tiles blue
...
17. In the 17th column paint the first 5 tiles orange
18. In the 17th column paint the last 5 tiles blue

1. In first column on left of grid, start with 5 orange hexagons and complete it with 5 blue hexagons. Leave column 2 from left white.
2. Repeat these colors and this 5:5 scheme every other column but alternate the two colors so that each colored column is the opposite order of the one on either side of it.

Figure 2: Levels of Abstraction in NL. Two drawing procedures for drawing the illustrated image, manifesting low-level and high-level abstraction.

To illustrate how humans may exhibit different levels of abstraction, consider the simple example in Figure 2, where a human is requested to describe a pattern on a 2D HEXAGONS board. The first (top) NL procedure expresses low-level abstraction; it refers to each occurrence of a half-column as a unique event. This is a repetitive and lengthy procedure. In contrast, the second (bottom) NL procedure refers to all the occurrences of this half-column at once (via 'repeat but alternate'), discarding distinctions related to, for example, tiles' positions and colors. The result of this abstraction is thus concise, clear and far more efficient.

In a broader sense, in order to express abstraction speakers employ so-called *abstraction mechanisms*—such as objects, functions and control flow (Koppelman and Van Dijk, 2010)—and expertise in using them is considered an important part of humans' CT skills (Grover and Pea, 2013). Such mechanisms are invoked in Figure 1. This kind of communication is not limited to the simple HEXAGONS board used in Figures 1–2. It is

[1] The data and models, along with the collection infrastructure (HEXAGONS App, Game and tasks), are publicly available at: `https://OnlpLab.github.io/Hexagons`.

relevant in countless many other domains as in the aforementioned calendar and car examples.

Due to the prevalence of abstraction in human communication, models would need to process varied levels of abstraction in order to correctly interpret NL instructions. But in order to successfully develop such models, we have to collect data that systematically reflect such phenomena, and to the best of our knowledge, this has not yet been done in NLP. The challenge of eliciting abstraction is genuine, as we aspire to elicit natural language that reflects authentic human communication of *any* speaker. But we cannot simply request crowdworkers to employ *abstraction mechanisms* as they are not familiar with these formal concepts. On the other hand, explicitly *teaching* them to employ abstraction undermines the naturalness of expression. So, how do we break out of this loop?

3 The Proposed Methodology

In this work we are interested in creating situations where humans express abstraction in NL spontaneously and naturally, towards learning models that can interpret such abstractions. To achieve this goal, we turn to the vast research in human learning and STEM education, on cultivating (and thus eliciting) higher-order CT skills in humans (Cuny, Snyder, and Wing, 2010; Shute et al., 2017). Eliciting such higher-order thinking requires a careful task design, drawing on literature on the development of instruments that probe and assess humans' CT skills (Ructtinger and Stevens, 2017; Relkin and Bers, 2019; Basu et al., 2021).

Our designed task stimuli are carefully crafted to evoke abstraction *without* explicitly requesting workers to do so. Our elicitation methodology extends a recent trend in grounded semantic parsing, where players engage in a referential game (*situated collaborative scenarios* in terms of Jayannavar et al. [2020]) where an *Instructor* provides instructions that should be grounded and executed in a (simulated) world (Long et al., 2016; Bisk et al., 2016a; Kim et al., 2019; Jayannavar et al., 2020). The remainder of this section elaborates on the virtual environment we devise, and the task stimuli we design for elicitation.

3.1 The Hexagons App and Game

In order to collect NL descriptions which express diverse abstraction levels, we design an online drawing app that enables users to construct increasingly complex images on a Hexagons board, a two-dimensional board paved with hexagonal tiles, of the kind illustrated in Figures 1–2. The Hexagons board contains 18 columns and 10 rows, and the Hexagons App UI provides a drawing interface in which a user may paint tiles using a palette of eight colors.

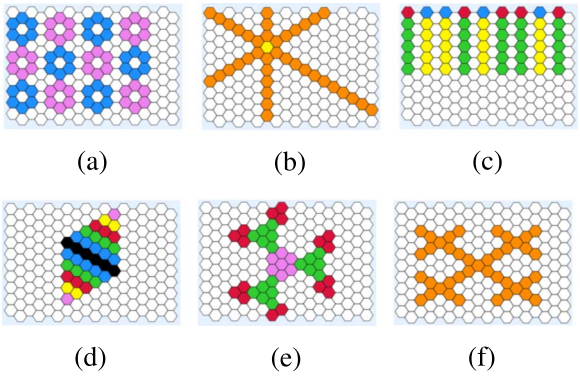


Figure 3: A Sample of the Hexagons Image Gallery.

In order to elicit NL instructions, we extended the app with an instruction-following game where a human agent is asked to describe the construction process of a given image (e.g., Figure 3) to a different user of the app, who has access to a similar but blank Hexagons board. The game has two different modes. The first mode is called **Description**, where a user is given an image from a pre-defined pool and has to provide instructions in NL on how to construct the image. Every line break in the textual description initiates a new instruction. The second mode is called **Execution**, where a user accepts a sequence of instructions one by one, and needs to execute them sequentially to reconstruct the target image on the board.

We refer to each pair of an instruction and a corresponding execution as a *drawing step*. We call the sequence of drawing steps composing the full image a *drawing procedure*.

3.2 The Task Stimuli

The Hexagons app assumes a single primitive action that corresponds to the two-place predicate `paint(position, color)`, which specifies a color for a specific tile in the 180 hexagon tiles. The key idea is to ask Instructors to construct a *complex* image that manifests some *regularity*. The regularity is intended to encourage Instructors to seek more efficient alternatives to the primitive-level operations, which then evokes

CT skills such as decomposition and abstraction in order to deliver an economic and efficient construction.

In what follows we briefly elaborate, for each abstraction mechanism that we target, how we design the form on the HEXAGONS board that potentially evokes this mechanism.

- **Objects.** Users may refer to a set of instances at once by means of the form they make (line, circle, triangle, etc.), discarding other details such as the position and color of individual tiles. Objects may be defined in one place, and referred to elsewhere (as in Figure 1).
- **Bounded Iterations** ('For' loops). To elicit bounded loops we design images that manifest periodic replication of an object. For example, Figure 3(a) shows a replication of a flower pattern 12 times.
- **Conditional Iterations** ('While' loops). To elicit conditional loops we design images that manifest a periodic replication of an object controlled by a certain condition. For example, in Figure 3(b), to replicate the lines with different length one may use the condition 'extend the lines out *up to the boundaries of the board*.'
- **Conditional Statements** ('if-then'). We design images that manifest random replication of steady variants of an object, where employing a condition enables users to capture all variants at once. For example, to capture the two variants of five-tile-long lines in Figure 3(c), one cannot simply use repetition, as the replication is not periodic. However, noticing that the red and blue 'tops' go with the green and yellow 'tails' respectively, enables a user to achieve an economic description using a condition on the 'top' tiles.
- **Functions.** We design images that manifest replication of objects in different colors or positions, to encourage defining a 'block' and then applying it with different parameters. Moreover, we use a particular set of visual functions, of *symmetrical* operations, and particularly reflection and rotation (e.g., Figures 3(d,e)).
- **Recursion.** This is a unique type of functions which is challenging to evoke. We approach this challenge by designing three types of stimuli: growing patterns, spirals, and self-similarity patterns, for example, fractals (Figure 3(f)).

We note that the association between images and targeted abstraction mechanisms has been defined *a priori*. However, in effect, users may generate instructions with no abstraction or use a different abstraction mechanism to achieve the same result. All in all, since users (and in particular, crowdworkers) aim to be efficient, they tend (even if not explicitly told) to employ abstractions.

4 Data Collection and Curation

For the data collection we employed English-speaking workers from Amazon Mechanical Turk (MTurk) and adopted the methodology of controlled crowdsourcing (Roit et al., 2020; Pyatkin et al., 2020) to ensure a high quality corpus. Specifically, the process includes four stages: **pilot, recruitment, annotation, and consolidation**.

We collect drawing procedures for the task stimuli in a process that comprises of two steps:

(1) In the *Collection* phase an Instructor writes instructions for drawing a given image, step by step via the **Description** mode of the game. Following this, the Instructor *aligns* each instruction she has written to its respective execution on the board via the **Execution** mode of the game. The result of this process is a *drawing procedure* where instructions are coupled with their actions grounded on the HEXAGONS board.

(2) In the *Verification* phase each drawing procedure from the *Collection* phase is given to two Verifiers who do not have access to the original image. The Verifiers are shown the instructions one by one in the **Execution** mode. Their task is to execute the instructions step by step until reconstructing the full image.

This two-phase process is intended to reveal faulty instructions and to ensure the quality and executability of the collected procedures, by making the Instructor's intentions explicit (step 1), and by exposing disagreements with Verifiers (step 2).

Pilot and Recruitment We checked the flow, clarity, and feasibility of the data collection in a pilot study, followed by two separate rounds of recruiting Instructors and Verifiers.

To recruit Instructors, we screened workers by examining understanding and engagement in the Instructor task. Appropriate candidates had to complete three Instructor tasks, that is, repeat the *Collection* phase with three randomly selected images from our pool. In no stage did we formally teach workers what abstraction is. Instead we engage the workers in several tasks and encourage them to write their instructions *efficiently* and to avoid tiresome repetitions of the primitive `paint`. Out of the 34 candidates, we assembled a group of 28 Instructors exhibiting *diverse* levels of abstraction, out of which 24 took an active role during the annotation procedure.

We follow a separate process in recruiting Verifiers using the *Verification* phase, instructing candidates to be as accurate as possible while executing drawing procedures. Out of 27 candidates, we recruited 16 workers exhibiting the most precise work. The groups of Instructors and Verifiers are disjoint, so drawing procedures are verified by workers other than those who generate them.

Annotation Procedure The annotation procedure is based on a *Generation-Validation* cycle, which is similar to previous protocols for constructing large-scale corpora by untrained crowdworkers (FitzGerald et al., 2018). Specifically, based on images from our crafted stimuli, drawing procedures are first generated and verified by the Instructors in the *Collection* phase, and then each procedure is given to two *additional* Verifiers, that work through the *Verification* phase to check the understandability and executability of the procedures.

The annotation process itself consisted of two rounds. In the first round we gave each image from our pool to three Instructors in order to generate three different drawing procedures for each image. Each of the generated procedures was verified as usual. In the second round, we presented Instructors with the opportunity to draw new images on a blank HEXAGONS board. The goal is to scale-up the extension of the image-pool with interesting compositions using crowdsourcing. Indeed, the collected images in this round reflect similar rationale to our own set of images (e.g., Figure 4(a–b)) yet demonstrate more complex interactions between structures and patterns (Figure 4(c–d)), with both abstract and figurative images (Figure 4(e–f)). This new pool of images then passed through the *Collection* and *Verification* phases as usual.

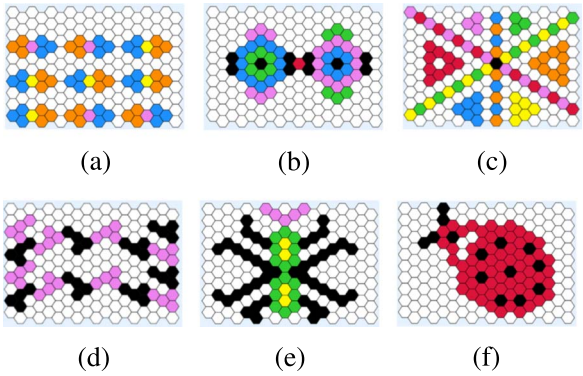


Figure 4: A Sample of Crowdsourced HEXAGONS Images by MTurk Workers in the Second Round.

Consolidation Having collected the raw dataset, we manually inspected all drawing procedures that had at least one disagreement between an Instructor and each of the Verifiers. Then, we developed a protocol to (i) detect Instructors' errors, (ii) classify the types of errors, and (iii) fix the Instructor execution. The protocol was applied to the data by the two first authors. The reported agreement on error classification was 0.95 and 0.98 Cohen's Kappa for the first two tasks and 95% agreement for the last one. Following this protocol, we detected cases where the Instructor's execution is not properly aligned with the instruction, and manually corrected the execution to match the instruction. Types of errors include: Over-/Under-execution where Instructors executes more/less than the instructions require; miscounting of positions on the board; error propagation from previous steps, and others such as using wrong colors. All in all we inspected 1461 drawing steps out of which 20.8% were identified as having Instructors' error and subsequently were manually corrected. This process results in data with fully-aligned instruction-execution pairs for each of the instructions in each of the drawing procedures.

Annotation Costs We used Amazon Mechanical Turk to recruit English-speaking workers for this study. Participants in the data collection rounds were paid higher rates than in Pilot and Recruitment. The payments for *Collection* and *Verification* were \$1.50 and \$0.50, respectively.[2]

[2]We never rejected results or blocked workers. Rather, we used MTurk qualifications as part of our controlled data collection methodology described here.

	Steps	Procedures	Tokens	Unique Images
Recruitment	1045	123	21K	17
Round I	1909	304	43K	101
Round II	1223	193	36K	63
Total	4177	620	100K	175

Table 1: Dataset Statistics.

5 The HEXAGONS Dataset

Our finalized dataset is the collection of all drawing procedures, composed of instructions and their aligned executions collected in both annotation rounds and some from the recruitment stage, after having passed our quality assurance and consolidation process. In total, we collected 620 drawing procedures yielding 4177 drawing steps, that is, 4177 instructions with aligned executions, circa 100K tokens. Table 1 shows the data statistics.

Quantitative Analysis In order to quantitatively evaluate the resulting dataset, we define Board-Based and Action-Based metrics that compare two different executions of an instruction.

Let us define a function $f(x)$ that accepts a board state x, and translates it into a set of elements $\langle position, color \rangle$ that indicate the colored tiles. Now let b and b' be two states of the board, where b and b' are considered *gold* and *hypothesis*, respectively.

Based on the $f(x)$ function, we can define Precision and Recall as in Equations (1) and (2), respectively, where precision is the percentage of tiles correctly colored from all those colored in the hypothesis, and recall is the percentage of tiles correctly colored from all those colored in gold. We then define F1 as usual as the harmonic mean of the two (see illustration in Figure 5).

$$Precision(b, b') = \frac{|f(b) \cap f(b')|}{|f(b')|} \quad (1)$$

$$Recall(b, b') = \frac{|f(b) \cap f(b')|}{|f(b)|} \quad (2)$$

The metrics we report come in two flavors. In Board-Based Metrics, $f(b)$ picks up all colored tiles in the resulting image after each step. In Action-Based Metrics, $f(b)$ focuses *only* on the tiles that changed color in the current step, that is, on the instruction's *denotation* (rather than the entire board state).

For assessing the quality of the dataset, we compare for each drawing step, the board states (or actions) of the Instructor considered as gold, to the board states (or actions) of a Verifier, considered the hypothesis. We report the Mean F1 and Exact Match (EM) averaged over the entire dataset. In addition we report the Max(/Min) Mean F1 and EM which takes into account only the higher(/lower)-scoring Verifier for each step.

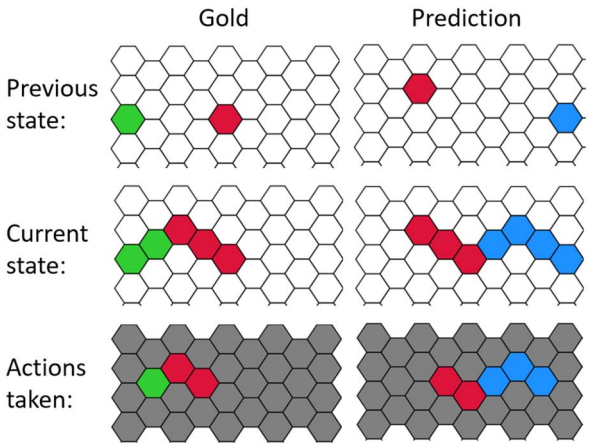


Figure 5: Board-Based and Action-Based Metrics. In the current state (second row) the intersection between Gold and Prediction is the three red tiles. Therefore, recall and precision are $\frac{3}{5}$, $\frac{3}{7}$, respectively and Board-Based F1 is $\frac{3}{6} = 0.5$. There are three actions taken in Gold and five in Prediction (third row) with one red tile in the intersection of both sets. Thus, recall and precision are $\frac{1}{3}$, $\frac{1}{5}$, respectively and Action-Based F1 score is $\frac{1}{4} = 0.25$. EM for both metrics is 0.

	Mean F1	Mean Exact Match
Board-Based	91.11 [85.85, 96.37]	72.32 [58.07, 86.57]
Actions-Based	84.46 [74.71, 94.21]	75.98 [62.58, 89.38]

Table 2: Dataset Evaluation. Min/Max Mean F1 and EM scores are in brackets.

Table 2 shows the dataset evaluation. The EM metric is more strict than Mean F1. The Max Mean-F1 score of circa 96 indicates that the images can most of the times be reproduced by at least one human following the instructions, despite the complexity of the instructions.

Qualitative Analysis: Overall Phenomena In order to understand the distribution of the elicited NL phenomena in our dataset we sampled 24 drawing procedures with a total of 194 drawing steps (instruction), preserving the internal distribution of stimuli types and annotation rounds. We then manually categorized utterances according

General Phenomenon	Type	#	# Steps	# Drawing Procedures
Abstraction	Object	77	95 (48.97%)	22 (91.67%)
	Control	61		
	Functions	49		
Goal/Result	Goal	18	33 (17.01%)	13 (54.17%)
	Result	17		
Linguistic	Anaphora	63	88 (45.36%)	15 (62.5%)
	Ellipsis	17		
	Comparatives	23		
Spatial		225	132 (67.69%)	23 (95.83%)

Table 3: Abstract, Linguistic and Spatial Phenomena in the HEXAGONS Dataset.

to abstract, linguistic and spatial phenomena, as summarized in Table 3.

Qualitative Analysis: Levels of Abstraction To probe further into the levels of abstraction that are manifested in the dataset, the first two authors annotated all the instructions in the *dev set* to one of four levels of abstraction we identified.

- *No abstraction (0):* In this level Instructors generate concrete instructions which show no abstraction. The instructions specify the coordinates and colors to be painted, in an absolute (e.g., ''Paint the third hexagon in the sixth column green'') or a relative (e.g., ''Paint the tile below this tile red'') fashion.
- *Low-Level Abstraction (1)*: In this level Instructors refer to a collection of tiles as a single object by means of the topographic shape they form in cases they form vertical or diagonal lines, which are endemic to the HEXAGONS board (e.g., ''paint the first column from the left green'', ''connect a diagonal line between the two tiles you just painted'').
- *Mid-Level Abstraction (2)*: In this level, Instructors refer to multiple tiles as defining an object (above and beyond lines), by applying an abstraction mechanism on multiple tiles (e.g., first step in Figure 1).
- *High-Level Abstraction (3)*: In this level Instructors use diverse abstraction mechanisms (Table 4) applied to multiple objects which themselves can be complex or abstract (as illustrated in the last three steps in Figure 1).

The annotation to levels was conducted in two stages. In the first stage the dev set was annotated into three levels where the last two levels are combined into a single category (Mid-to-High). In the second stage, Mid-to-High cases were split into two levels. The inter-annotator agreement for the first stage is 0.923 Krippendorff's Alpha (Krippendorff, 2004) and for the second stage it is 0.94 Krippendorff's Alpha. For both coefficient's calculations we use the weighted scheme for ordinal variables (Gwet, 2015).

Abs. Mechanisms	Examples
Bounded Iterations	''Make four more caterpillars below the original leaving an empty space between every two caterpillar.''
Cond. Iterations	''Repeat this pattern until you run out of room on grid.''
Cond. Statements	''Directly beneath the painted tiles, paint green and yellow vertical columns of five touching tiles, using green below the red tiles and yellow below the blue tiles.''
Objects	''make a blue dog bone shape''
Symmetry	''Reflect the multi-colored triangle you made in the previous two steps symmetrically over the black diagonal.''
Recursion	''Form an X by . . . At each corner, . . . form 4 smaller X shapes''

Table 4: Abstraction Mechanisms in the HEXAGONS Dataset.

	No (0)	Low (1)	Mid (2)	High (3)	Total
# Instructions	174 (39%)	47 (10.5%)	104 (23.3%)	121 (27.1%)	446 (100%)

Table 5: Levels of Abstraction in the Dev Set.

Table 5 shows the distribution of abstraction levels we defined within the drawing steps in the dev set. This analysis shows that most of the drawing steps (>60%) contain abstract instructions, where 50% contain Mid-to-High abstraction level.

6 Experiments

The Task Given the HEXAGONS dataset, we aim to devise models that interpret NL instructions and mimic an Executor's role, in order to assess how standard Pre-trained Language Models (PLM) interpret these utterances.

To this end, we define a computational task as follows. Let $D = d_1 \dots d_n$ be a sequence of NL instructions of a drawing procedure and let $b = t_1 \dots t_{180}$ be a board state naming all tiles' colors on the board at a given state. We aim to induce a function $f(d_1 \dots d_n) = b_1 \dots b_n$ that maps a given sequence of instructions to the sequence of board states that indicate the instructions' denotation on the board. Since such an f

	#Proc.	#Steps	Agreed Procedures	Agreed Steps
Train	496	3278	392 (79.51%)	87.19%
Dev	62	446	49 (79.09%)	85.43%
Test	62	453	43 (69.35%)	83.22%
Total	620	4177	484 (78.44%)	86.57%

Table 6: Data Splits Statistics.

is overly complex, we model each drawing procedure as a sequence of *instruction-to-execution* steps, where, given an instruction d_i in a procedure, we seek a model for $g(d_i) = (\{aij\}_{j=1}^{l_i})$ to predict the l_i denoted *actions* $a_{i1}...a_{il_i}$, where $a_{ij} = \langle row_{ij}, column_{ij}, color_{ij} \rangle$. Intuitively, this means that in the execution of each drawing step, we classify each tile to a color label or no_action.

Input Configurations For each of the models, we experiment with several input settings that differ in the type and extent of *context* provided: *(i) No-History*. The input is only the instruction to be executed (current drawing step). *(ii) 1-Previous*. The input contains the instruction to be executed (current step) and one previous instruction. *(iii) Full-History.* The input contains the instruction to be executed and all previous instructions in that drawing procedure. *(iv) Oracle Board-state*. The input contains the instruction to be executed and the gold board-state that is obtained prior to the current drawing step. No previous instruction history is included. *(v) Predicted Board-state*. The input contains the instruction to be executed and the predicted board-state, predicted for all steps so far. No previous instruction history is included. *(vi) Full-History + Oracle/Predicted Board-state*, a combination of (iv) and (v) with full history (iii).

Data Splits We split the dataset into train/dev/test with an 80/10/10 ratio of the drawing procedures (Table 6). Following up on recent practices (Finegan-Dollak et al., 2018; Herzig and Berant, 2020; Goldman et al., 2022), we randomly split the data in a way that *avoids* shared stimuli between the train, dev, and test sets. Specifically, (i) we make sure that there is no image overlap between the three sets (recall that each image is delivered to at least three Instructors; see Section 4), and (ii) we keep the same distribution of images in terms of the abstraction mechanisms they are designed to elicit (Section 3) as well as the same proportion between the images collected in different annotation rounds.

6.1 Models

We design two neural models based on two types of PLMs, a BERT-style encoder-only model and a generative encoder-decoder model. For each of these architectures, we modelled the task in a way that is most compatible with it, a classification task for the encoder model (DeBERTa) and a generation task with the encoder-decoder model (T5). We describe here our two architectures in turn.[3]

Classification-Based: For the classification model we fine-tune DeBERTa[4] (He et al., 2020) with a classification head to predict an action/no_action for each of the tiles, resulting in 180 prediction steps for each instruction. The output of each prediction is one of 9 classes (8 colors and 1 no_action). We define the inputs for the task as follows: The current instruction is prepended with a given tile's coordinates, to indicate for which of the tiles the model is making a prediction (e.g., <*row number*> <*column number*> <*current instruction*>). In the *Full-History* setting, the previous instructions are concatenated with a delimiter. When adding the *board-state*, we represent it as a sequence of 180 colors and we additionally mark the given tile with delimiters (e.g., *..blue, white, TARGET_S, red, TARGET_E*, red..).

Generation-Based: T5 (Raffel et al., 2020) is a generative transformer architecture which uses a text-to-text framework for a variety of NLP tasks.[5]

We formulate our text-to-actions task as a text-to-text task using a straightforward input/output scheme: The task's input is put in a template that consists of a prefix and a suffix (*simplify instructions:* <*current instruction*>. *simplified instructions:*) and fed as input to the model.[6] The gold output actions are formatted into text by first transforming them into triplets (<*row*

[3]We fine-tune all models for 20 epochs with early stopping and a batch size of 4. During inference, when performing conditional sequence generation (with T5) we use beam search with a beam size of 3, without sampling.

[4]Specifically we use DebertaForSequenceClassification from the Huggingface Transformers library (Wolf et al., 2020), with the *microsoft/mdeberta-v3-base* model.

[5]We use the T5 V1.1 Model Adapted checkpoint, which we found in preliminary experiments to perform better for our task: `https://huggingface.co/google/t5-base-lm-adapt`.

[6]The prefix we use is inspired by T5's pre-training scheme; using a natural language task prefix. This template may be viewed as a ''prompt'' (Liu et al., 2021), but we did not engage with extensive prompt-engineering.

	F1	EM
Baseline	13.15	5.96
Skyline	82.06	72.3

Table 7: Baseline and Skyline on the Test Set.

number> <*column number*> <*color*>), which we then combine into a longer comma separated string (e.g., *0 4 red, 0 5 blue, 1 0 green*), ordering the actions by row and column. During inference, we generate the most likely continuation for the input at hand. We then take the generated sequence and parse it into actions, discarding malformed token sequences. Due to the generative nature of the process, the model's current prediction is conditioned on all previously predicted actions for a given step.

Baseline and Skyline Our naïve baseline model is a deterministic rule-based model based on pattern-matching, in line with previous work (Pišl and Mareček, 2017). The model we design detects patterns that reflect the basic predicate `paint(position, color)`, where the position assumes coordinates on a (top-down, left-right) grid. For example, given the sentence ''In the first column, color the 2nd tile blue'', this model extracts the action *Paint((2, 1), blue)*. The naïve model refers only to the current instruction. As a skyline we use human performance on the task, presented in terms of the Action-Based Mean F1/EM for the dev and test sets.

Evaluation Metrics To evaluate models' performance, we report Action-Based Mean F1/EM of the predicted actions compared to gold actions. That is, the Action-Based F1/EM (Section 5) are averaged over all instructions in the test set.

7 Results and Analysis

Table 7 shows results of the naïve baseline and the human skyline performance and Table 8 shows the performance of the neural models across the different input configurations, on the test set.

The results show that all models perform substantially better than the naïve rule-based baseline, where the lowest results (obtained by the No-History condition) are still 23.23 F1 and 15.23 EM points over this baseline on the test set. At the same time, all models are substantially inferior to human performance, where the best model performance (Full-History) is 35.91 F1 and 45.81 EM points below human performance on the test set.

	DeBERTa		T5	
	F1	EM	F1	EM
No-History	36.38	21.19	36.84	21.63
1-Previous	37.51	20.31	38.94	23.17
Full-History	**46.15**	25.39	43.6	**26.49**
Predicted Board	40.7	24.28	40.21	26.49
Predicted + Full	40.47	21.19	38.56	23.4
Oracle Board	43.52	22.74	43.31	28.03
Oracle + Full	49.55	25.61	48.11	31.35

Table 8: Results of DeBERTa and T5 on the Test Set.

DeBERTa and T5 both show the same trends for the different input configurations, with the generative model (T5) often performing better at EM and the classifier (DeBERTa) having higher F1.

Our ablated experiments on input configurations are designed to empirically assess the contribution of two kinds of contexts, textual and board-state contexts. We observe that textual context (previous instructions) is an important factor in model performance; the longer the context is, the better the model performs. Performance is lowest when predicting executions on the HEXAGONS board with only the current instruction as input (No-History). Adding more context proves to be beneficial, with the Full-History condition having the best realistic (non-oracle) performance. This result corroborates previous findings which show that models benefit from textual history (Haber et al., 2019; Xu et al., 2022).

A different way of providing context for the execution of an instruction is via the state of previous executions on the board. Here, we experiment with either providing an oracle board-state at each step, or iteratively feeding the predicted board-states from the previous step to the current step. While providing the oracle board-state improves performance upon the No-History condition, our results show that it is not as informative as including the full instruction history.

A possible reason may be that textual instructions often refer back to previously introduced (or decomposed) objects, while board states do not explicitly name these decomposed concepts. Adding both the oracle board-state and all previous

We reserve the investigation of prompts and templates for future work.

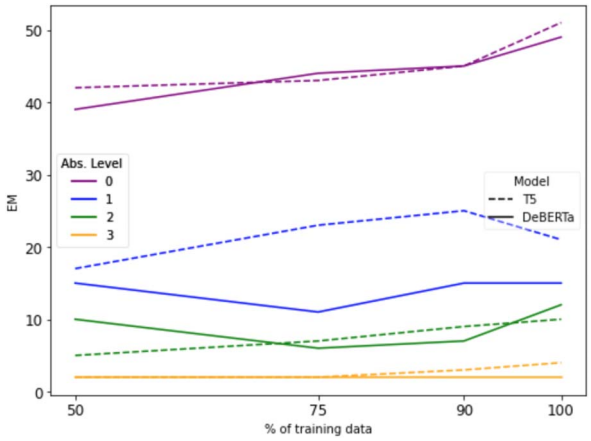


Figure 6: Qualitative Learning Curve of DeBERTa and T5 on Full History Models, Mean EM.

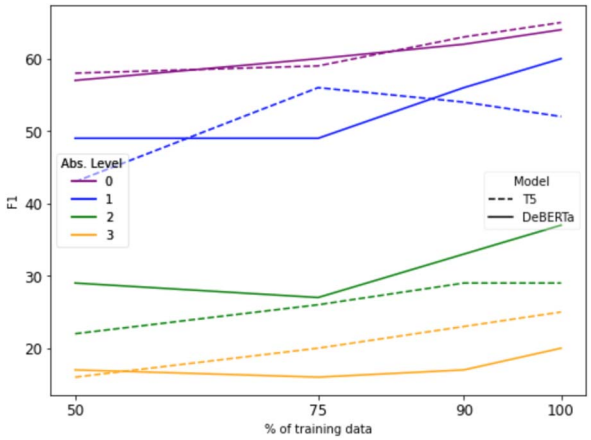


Figure 7: Qualitative Learning Curve of DeBERTa and T5 on Full-History Models, Mean F1.

instructions as input results in the best performance, however this is not a realistic setup. The more realistic context setting is that of a predicted board and all previous instructions, but it performs worse than only providing full history, due to error propagation in the predicted states.

Abstraction Levels Figures 6–7 present the models' performance by abstraction levels (Section 5) across increasing train set sizes. The results on the full train set show that models' performance is inversely correlated with the abstraction-level of the instructions; models' performance on executions of concrete primitive-like instructions exceeds those of Mid-to-High level of abstraction. This result is significant across models, metrics, and input configurations.[7]

Abs.	F1				
Levels	No	Low	Mid	High	All
Baseline	24.38	20.55	7.25	3.59	14.34
Skyline	87.77	93.55	85.02	83.52	86.59
	EM				
Baseline	14.37	12.77	2.88	0.83	7.85
Skyline	81.32	86.17	75.96	69.42	77.35

Table 9: Baseline and Skyline on the Dev Set by Levels of Abstraction.

Comparing these results with baseline and skyline by the levels of abstraction (Table 9), we observe that model performance resides between these two boundaries while substantially inferior to skyline across *all* four levels of abstraction.

The results on the gradually increasing train set size show that although in general all levels of abstraction benefit from larger train sets, still model performance on non-abstract instructions is consistently better than instructions exhibiting Mid-to-High-level of abstraction, keeping a mean gap of circa 38 Mean F1. Noticeably, the increase for the highest abstraction level is very mild. especially for EM. This hints that no substantial learning is happening at the highest abstraction level, and a different architecture or training regime, geared towards abstraction, is needed.

We manually inspected executions of our models with respect to the levels of abstraction of the instruction. Looking at the successful executions of high-level instructions, we observe that those instructions are mainly instances where no actions should be performed, for example, Goal/Result declarations, or instances where very common objects (e.g., flowers) are defined and drawn. More complex functions, such as *repetitions with conditions*, are harder for the models to interpret.

An example of how executions differ between different abstraction levels is displayed in Figure 8. The model correctly executes the first instruction which contains no abstraction. The next instruction is of high abstraction including replication of the triangle object. The model does not manage to identify the spots where to attach the new triangles, or to generate appropriate triangles.

[7] We checked significance by conducting several Kruskal-Wallis tests to compare model performances by abstraction levels for all possible combinations of different models (T5 and DeBERTa), scores (F1 and EM) and input configurations (Full-History and No-History). For brevity, Figures 6–7 show only the input configurations of Full-History models.

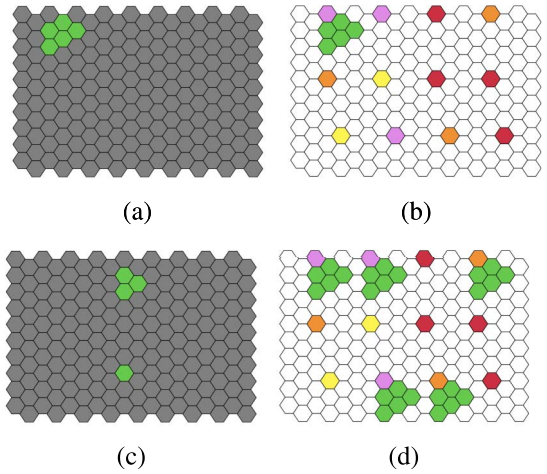


Figure 8: Processing Different Abstraction Levels. DeBERTa's executions (left) and gold board-states (right) for different levels of abstraction. Instruction (a)–(b): ''Use green to fill in the 2nd and 3rd spots on the 3rd column, 1st and 2nd spots on the 4th column, and 2nd spot on the 5th column.'', F1 = 1; (c)–(d): ''Create the same shape with green on all the purple and orange spots.'', F1 = 0.08.

These findings are all consistent with the claim that abstract instructions pose a challenge for current NLP technology, *orthogonally* to data size and various other factors.

8 Related Work

Studying abstraction in collaborative communication is related to previous studies on collaborative games that focus on how interlocutors generate referring expressions accepted and understood by both the speaker and hearer (Clark and Wilkes-Gibbs, 1986; Khani et al., 2018; Haber et al., 2019; Udagawa and Aizawa, 2019). In such situations a speaker attempts to generate the shortest refererring expression that will sufficiently communicate their intention. This phenomenon of minimizing speakers' effort is inline with Grice's (1975) maxim of *quantity*, stating that speakers will give as much information as needed and not more.

The settings of collaborative games are very common in creating datasets for grounded semantic parsing as navigation tasks (Anderson et al., 1991; MacMahon et al., 2006; Anderson et al., 2018; Chevalier-Boisvert et al., 2018; Misra et al., 2018; Chen et al., 2019; Paz-Argaman and Tsarfaty, 2019; Suhr et al., 2019), the 2-D/3-D blocks world (Bisk et al., 2016a,b, 2018; Jayannavar et al., 2020), and other instruction-following scenarios (Long et al., 2016; Kim et al., 2019). Some of these studies observe abstraction as a phenomenon that indeed occurs in NL instructions (e.g., Anderson et al., 2018), implying that abstraction is a cross-domain and hence critical phenomenon for natural language understanding. However, eliciting naturally-occurring NL instructions that reflect a variety of abstraction levels in a systematic way is novel to the HEXAGONS data.

To confirm this, we inspected the 2-D Blocks dataset (Bisk et al., 2016a,b; Pišl and Mareček, 2017), which most resembles our setting. Sampling 594 instructions (5% of the train set), we found that almost all the instructions (96.5%) map to actions of shifting a single block to some location, with some spatial expressions (e.g., ''place box 17 three spaces above box 20''). Such instructions are labeled ''no abstraction'' in our protocol (see Section 5). Notably, a fraction of the sample does express some low-level (2.7%) and mid-level (0.8%) abstraction.

Two other studies that are particularly related to our work are by Wang et al. (2017) and Wang et al. (2016). In the VoxeLurn study (Wang et al., 2017), a community of users is interacting with a computerized agent to deliver constructions in a 3-D blocks world. The community gradually and collaboratively builds increasingly complex and more *abstract* language from a core programming language via a process called ''naturalization''. SHRDLURN (Wang et al., 2016) exhibits similar constructions but on an individual rather than a community effort. Both studies indeed address abstraction, but from an *opposite* direction to ours; while these works assume a strict narrow and synthetic language and build abstractions bottom-up, our work aims to tackle the opposite direction, uncovering abstractions that are expressed in unrestricted informal NL and grounding them in an executable 'backend'. Thus, these studies and ours exhibit orthogonal ways to address abstraction.

9 Conclusion

We bring to the fore of NLP a novel and critical aspect of human–computer communication, namely, the ability to automatically detect, interpret, and ground abstraction in NL. We devise an abstraction elicitation methodology and deliver a novel benchmark, HEXAGONS, manifesting

the denotation of instructions rich and diverse in their levels of abstraction. Our results on the *instruction-to-execution* task derived from these data show that the models' performance is significantly *inversely* correlated with the level of abstraction, and this holds across models, contexts, and data sizes. This work opens a manifold of directions for future research such as generating human-like abstractions or detecting the level of abstraction, as well as studying abstraction in adjacent fields as linguistics, cognitive science and NL programming.

Acknowledgments

We thank the audience of the BIU-NLP Seminar, the BIU Linguistics Colloquium, and the TAU-NLP Seminar for fruitful discussion of this work. We specifically thank Yoav Goldberg for his critical comments on an earlier draft. We would also like to thank our action editor and the anonymous reviewers for their invaluable suggestions and feedback. This research is funded by the European Research Council, ERC-StG grant no. 677352, for which we are grateful.

References

Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, Henry S. Thompson, and Regina Weinert. 1991. The HCRC map task corpus. *Language and Speech*, 34(4):351–366. https://doi.org/10.1177/002383099103400404

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3674–3683. https://doi.org/10.1109/CVPR.2018.00387

Satabdi Basu, Daisy W. Rutstein, Yuning Xu, Haiwen Wang, and Linda Shear. 2021. A principled approach to designing computational thinking concepts and practices assessments for upper elementary grades. *Computer Science Education*, 31(2):169–198. https://doi.org/10.1080/08993408.2020.1866939

Yonatan Bisk, Daniel Marcu, and William Wong. 2016a. Towards a dataset for human computer communication via grounded language acquisition. In *Symbiotic Cognitive Systems, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 13, 2016*, volume WS-16-14 of *AAAI Workshops*. AAAI Press.

Yonatan Bisk, Kevin J. Shih, Yejin Choi, and Daniel Marcu. 2018. Learning interpretable spatial operations in a rich 3d blocks world. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2–7, 2018*, pages 5028–5036. AAAI Press.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. 2016b. Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 751–761, San Diego, California. Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1089

Erin M. Burgoon, Marlone D. Henderson, and Arthur B. Markman. 2013. There are many ways to see the forest for the trees: A tour guide for abstraction. *Perspectives on Psychological Science*, 8(5):501–520. https://doi.org/10.1177/1745691613497964, PubMed: 26173209

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12530–12539. https://doi.org/10.1109/CVPR.2019.01282

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2018. BabyAI: A platform to study the

sample efficiency of grounded language learning. In *International Conference on Learning Representations*.

Herbert H. Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22(1):1–39. https://doi.org/10.1016/0010-0277(86)90010-7

Jan Cuny, Larry Snyder, and Jeannette M. Wing. 2010. *Demystifying Computational Thinking for Non-computer Scientists*. Unpublished manuscript in progress, referenced in http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, Paul R. Young, and Peter J. Denning. 1989. Computing as a discipline. *Communication of the ACM*, 32(1):9–23. https://doi.org/10.1145/63238.63239

Edsger W. Dijkstra. 1972. The humble programmer. *ACM Turing Award Lectures*. https://doi.org/10.1145/1283920.1283927

Edsger W. Dijkstra. Accessed 1 May 2021. Ew dijkstra quotes.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia, Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1033

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale QA-SRL parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2051–2060, Melbourne, Australia. Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1191

David Ginat and Yoav Blau. 2017. Multiple levels of abstraction in algorithmic problem solving. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 237–242, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3017680.3017801

Omer Goldman, David Guriel, and Reut Tsarfaty. 2022. (un)solving morphological inflection: Lemma overlap artificially inflates models' performance. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 864–870, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-short.96

Herbert P. Grice. 1975. Logic and conversation. In *Speech Acts*, pages 41–58. Brill. https://doi.org/10.1163/9789004368811_003

Shuchi Grover and Roy Pea. 2013. Computational thinking in k–12: A review of the state of the field. *Educational Researcher*, 42(1):38–43. https://doi.org/10.3102/0013189X12463051

Kilem L. Gwet. 2015. On Krippendorff's alpha coefficient. Accessed: 1 June 2022.

Janosch Haber, Tim Baumgärtner, Ece Takmaz, Lieke Gelderloos, Elia Bruni, and Raquel Fernández. 2019. The PhotoBook dataset: Building common ground through visually-grounded dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1895–1910, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1184

Bruria Haberman. 2004. High-school students' attitudes regarding procedural abstraction. *Education and Information Technologies*, 9(2):131–145. https://doi.org/10.1023/B:EAIT.0000027926.99053.6f

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

Jonathan Herzig and Jonathan Berant. 2020. Span-based semantic parsing for compositional generalization. *arXiv preprint arXiv:2009.06040. Version 2*.

Prashant Jayannavar, Anjali Narayan-Chen, and Julia Hockenmaier. 2020. Learning to execute instructions in a Minecraft dialogue. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2589–2602, Online. Association

for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.232

Fereshte Khani, Noah Goodman, and Percy Liang. 2018. Planning, inference, and pragmatics in sequential language games. *Transactions of the Association for Computational Linguistics*, 6:543–555. https://doi.org/10.1162/tacl_a_00037

Jin-Hwa Kim, Nikita Kitaev, Xinlei Chen, Marcus Rohrbach, Byoung-Tak Zhang, Yuandong Tian, Dhruv Batra, and Devi Parikh. 2019. CoDraw: Collaborative drawing as a testbed for grounded goal-driven communication. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6495–6513, Florence, Italy. Association for Computational Linguistics.

Herman Koppelman and Betsy Van Dijk. 2010. Teaching abstraction in introductory courses. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, pages 174–178. https://doi.org/10.1145/1822090.1822140

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology (2nd ed.), Chapter 11*. Beverly Hills, CA: Sage.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586. Version 1*.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465, Berlin, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1138

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1475–1482. AAAI Press.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1287

Tzuf Paz-Argaman and Reut Tsarfaty. 2019. RUN through the streets: A new dataset and baseline models for realistic urban navigation. 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 6450–6456, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1681

Bedřich Pišl and David Mareček. 2017. Communication with robots using multilayer recurrent networks. In *Proceedings of the First Workshop on Language Grounding for Robotics*, pages 44–48. https://doi.org/10.18653/v1/W17-2806

Valentina Pyatkin, Ayal Klein, Reut Tsarfaty, and Ido Dagan. 2020. QADiscourse - discourse relations as QA pairs: Representation, crowdsourcing and baselines. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2804–2819, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.224

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Emily Relkin and Marina Umaschi Bers. 2019. Designing an assessment of computational thinking abilities for young children. In Lynn E. Cohen and Sandra Waite-Stupiansky, editors, *STEM in Early Childhood Education: How Science, Technology, Engineering, and Mathematics Strengthen Learning*, chapter 5. Routledge, New York. https://doi.org/10.4324/9780429453755-5

Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. Controlled crowdsourcing for high-quality QA-SRL annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7008–7013, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.626

Liliana Ructtinger and Robert Stevens. 2017. Computational thinking task design and assessment. *Scan: The Journal For Educators*, 36(1):34–41.

Valerie J. Shute, Chen Sun, and Jodi Asbell-Clarke. 2017. Demystifying computational thinking. *Educational Research Review*, 22:142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Alane Suhr, Claudia Yan, Jack Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. 2019. Executing instructions in situated collaborative interactions. 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 2119–2130, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1218

Takuma Udagawa and Akiko Aizawa. 2019. A natural language corpus of common grounding under continuous and partially-observable context. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7120–7127. https://doi.org/10.1609/aaai.v33i01.33017120

Sida I. Wang, Samuel Ginn, Percy Liang, and Christopher D. Manning. 2017. Naturalizing a programming language via interactive learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 929–938, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1086

Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2368–2378, Berlin, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1224

Jeannette Wing. 2017. Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, 25(2):7–14.

Jeannette M. Wing. 2011. Computational thinking—what and why? *CMU Research Notebook*.

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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-demos.6

Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland. Association for Computational Linguistics.