# PACE: Procedural Abstractions for Communicating Efficiently

## Title
PACE: Procedural Abstractions for Communicating Efficiently



Journal
Proceedings of the Annual Meeting of the Cognitive Science Society, 47(0)

Authors
Thomas, Jonathan
Silvi, Andrea
Dubhashi, Devdatt
et al.

Publication Date
2025

# PACE: Procedural Abstractions for Communicating Efficiently

**Jonathan Thomas** **Andrea Silvi** **Devdatt Dubhashi** **Moa Johansson**
Chalmers University of Technology and University of Gothenburg

## Abstract

A central but unresolved aspect of problem-solving in AI is the capability to introduce and use abstractions, something humans excel at. Work in cognitive science has demonstrated that humans tend towards higher levels of abstraction when engaged in collaborative task-oriented communication, enabling gradually shorter and more information-efficient utterances. Several computational methods have attempted to replicate this phenomenon, but all make unrealistic simplifying assumptions about how abstractions are introduced and learned. Our method, Procedural Abstractions for Communicating Efficiently (PACE), overcomes these limitations through a neuro-symbolic approach. On the symbolic side, we draw on work from library learning for proposing abstractions. We combine this with neural methods for communication and reinforcement learning, via a novel use of bandit algorithms for controlling the exploration and exploitation trade-off in introducing new abstractions. PACE exhibits similar tendencies to humans on a collaborative construction task from the cognitive science literature, where one agent (the architect) instructs the other (the builder) to reconstruct a scene of block-buildings. PACE results in the emergence of an efficient language as a by-product of collaborative communication. Beyond providing mechanistic insights into human communication, our work serves as a first step to providing conversational agents with the ability for human-like communicative abstractions.

**Keywords:** efficient communication; reinforcement learning; abstractions learning.

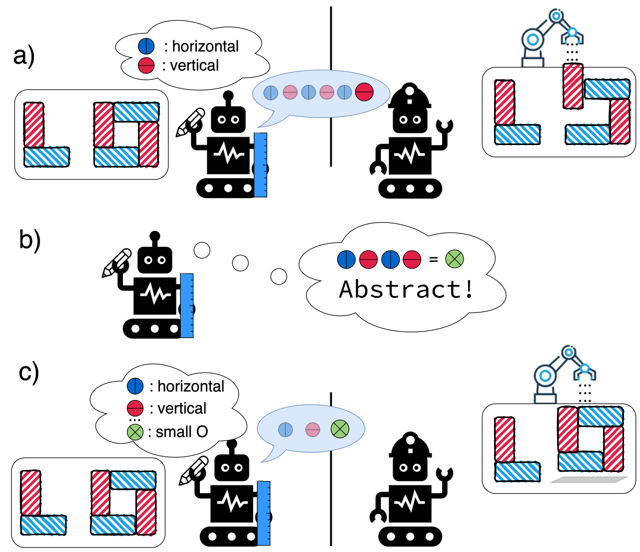



Figure 1: Two artificial agents playing the architect-builder game, starting from a small artificial language. Initially, the architect messages refer to horizontal or vertical blocks (a). After multiple interactions, the architect tries to introduce an abstraction (b), which after a learning period allows for shorter communication to solve the task (c).

## Introduction

Procedural tasks such as cooking and programming require executing a sequence of actions to achieve a desired goal. A natural approach to reduce their complexity and improve generalisation to new tasks is to introduce abstractions for common sequences of actions (Solway et al., 2014). For example, in cooking, techniques such as sautéing or kneading serve as foundational building blocks that simplify complex recipes. Similarly, abstractions emerge in repeated communication between human dyads when collaborating on shared tasks (Krauss and Weinheimer, 1964; Hawkins et al., 2020; McCarthy et al., 2021). Over time, as new abstractions are introduced into the shared language, communication becomes more concise. This can improve cooperation and allow to reach the goal more easily.

How communication shapes abstractions is of much interest within the AI and Cognitive Science communities (Lee, 1996; Ho et al., 2019). A compelling principle is *Efficient Communication*, which argues that languages are under pressure to be informative whilst minimising cognitive load (Kemp and Regier, 2012; Gibson et al., 2017; Zaslavsky et al., 2019; Gibson et al., 2019). This provides insights into the abstractions humans converge to in semantic domains such as colour, kinships and others (Xu et al., 2020; Regier et al., 2015; Kemp and Regier, 2012; Yin et al., 2024). Here we want to explore the principle of Efficient Communication with artificial agents in collaborative procedural tasks: how do the pressures for efficient communication manifest and impact abstraction introduction and use?

The architect-builder game introduced in McCarthy et al. (2021) provides a simple framework to study the use of abstractions in collaborative tasks. It is a repeated game with two participants, the architect and the builder, who must complete a collaborative building task. In each round, the architect observes the goal-scene on their screen and conveys some instructions in natural language to the builder. The builder, who cannot see the original scene, interprets these instruc-

Correspondence to: `silvi@chalmers.se`.



In D. Barner, N.R. Bramley, A. Ruggeri and C.M. Walker (Eds.), *Proceedings of the 47th Annual Conference of the Cognitive Science Society* 

tions and attempts to reconstruct the shapes by placing blocks within the grid. The architect may send further instructions to help the builder. McCarthy et al. (2021) showed that in early rounds human participants use lengthy utterances, while later realizing it is beneficial to introduce abstractions for commonly occurring shapes (e.g. an L-shape). By doing this they move towards a more compact language, thereby leading to more efficient communication. Creating such abstractions is a key feature of human collaboration. In this work, we develop a computational framework of artificial agents to solve this kind of collaborative task (Figure 1). Previous work provides computational models of this process (McCarthy et al., 2021; Jergéus et al., 2022), but rely on simplifying assumptions, limiting the ability to capture the collaborative dynamics of language learning, which we address in this work.

We propose a novel multi-agent neuro-symbolic method called Procedural Abstractions for Communicating Efficiently (PACE). We integrate both neural and symbolic learning methods from the computer science literature. On the symbolic side, library learning (Ellis et al., 2021; Bowers et al., 2023), a method from program synthesis that aims to abstract common subprograms into new, more easily reusable terms. On the neural side, emergent communication (EC) (Foerster et al., 2016; Lazaridou and Baroni, 2020), and reinforcement learning (RL) (Sutton and Barto, 1998), enable the development of a flexible, learnable communication language. PACE offers a unified framework for studying the formation and evolution of abstractions across multi-round interactions.

In previous work, EC has been used to explain how communicative pressures for efficient communication shape the language structure in other settings, such as Carlsson et al. (2021,0). Here, we apply EC techniques to study how pressures manifest within abstraction learning.

We evaluate PACE on the Architect-Builder game, which we extend to artificial agents. In our extension, the architect composes programs to describe the goal-scene in an artificial symbolic language (Figure 1 a). The programs are neurally encoded as messages, following conventions from EC, and are then interpreted by the builder, who tries to reconstruct the scene. After several rounds, the architect's internal symbolic language is extended with a new abstraction for a commonly occurring subprogram (Figure 1 b). Each new abstraction provides alternative shorter ways to express goal-scenes, but require the agents to learn how to communicate and understand them. This contention between shorter but unestablished programs and more verbose established programs is handled via RL over repeated rounds of interaction (Figure 1 c). We find that it exhibits similar tendencies to humans — a development toward a richer language that allows for more concise utterances. Interestingly, after a number of abstractions have been introduced, our model naturally converges to a stable language, after which no more abstractions are introduced. Moreover, we show that in this setting languages that are closer to optimality in terms of the trade-off between average morphosyntactic complexity and language size are easier to learn, connecting our work with the Efficient Communication literature. Our approach addresses limitations of existing approaches and provides a valuable framework for future exploration in this area.

## Set-up: Architect-Builder Game

In the Architect-Builder Game, the architect is provided with a set of goal-scenes depicting two adjacent shapes on a (9x9) grid. Each shape is a variable-size combination of 2x1 horizontal and 1x2 vertical blocks. The architect needs to communicate instructions to the builder (who does not know the goal state) that allow it to construct the goal-scene starting from a blank grid. This is inspired by the human experiments in McCarthy et al. (2021).

**Dataset** We extend the dataset from McCarthy et al. (2021) for our experiments. We increase the size going from 3 to 31 unique shapes, with multiple sub-shapes reoccurring in different shapes. Our shapes are of different sizes and resemble either uppercase or lowercase letters from the English alphabet. As before, the dataset consists of scenes composed of two shapes placed side by side. Our dataset contains 961 goal-scenes (compared to the original 9), which enables us to use it for training neural agents. We split the dataset into 930 training scenes and 31 test scenes. These splits are constructed to ensure the distribution over shapes is the same. However, in the test set, the ordered pair of shapes constituting each goal-scene, does *not* appear in the training set.

**The agents** The *architect* is a neuro-symbolic agent: building instructions are constructed symbolically and encoded and communicated neurally. The *builder* is a purely neural agent and its task is to learn to decode instructions to reconstruct the scenes step-by-step. The architect has an internal symbolic action language $\mathcal{A}$ for constructing programs $p$ of building instructions for goal scenes. Initially, the language has only two primitives: $\mathcal{A}_{init} = \{a_{horiz}, a_{vert}\}$, corresponding to placing either a horizontal or vertical block. A *program* of length $l$ takes the form $p = [a_1, ..., a_l]$, where initially, each $a_i$ is one of $\{a_{horiz}, a_{vert}\}$ [1].

## PACE: Selection, Communication and Abstraction

There are three phases to PACE, **Selection**, **Communication** and **Abstraction** (see Figure 2). The architect must first choose a program to communicate. For this purpose the architect is initialised with a table of programs written in $\mathcal{A}_{init}$ for each scene in the training set. Initially there is one program per scene. Secondly, the architect learns how to communicate (neurally) instructions to the builder via EC, allow-

[1] We note that along with each action $a_i$ we also attach positional information. As our goal is to learn abstractions of shapes, positions are irrelevant. Hence we assume positional encodings are predetermined and omit them from further notation.

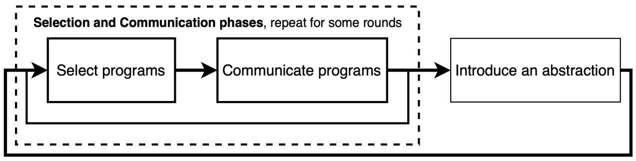



Figure 2: Interaction between the architect and builder in PACE proceeds as follows: (1) given a goal-scene the architect chooses the program to communicate, and (2) the architect and builder communicate via EC. After multiple interactions, an abstraction is introduced. Then the loop repeats.

ing them to learn a language for collaboration. After some rounds, the architect enters the Abstraction phase, where it uses a library learning mechanism to identify common subsequences to abstract. The new abstraction is then used to add additional (shorter) programs for the relevant scenes. In the next Communication phase, the architect thus has multiple programs to choose from. We use reinforcement learning techniques to learn to choose between alternative programs. We next describe these three phases in more detail.

**Selecting programs** The architect initially has one program to construct each scene from an empty grid. As abstractions are introduced, there will be multiple alternative programs of different lengths available to the architect. They may differ in reconstruction accuracy, in particular, programs containing a new abstraction will initially have lower accuracy, as the builder has not been exposed to them. Hence, this is a classic *exploration* vs. *exploitation* problem for which reinforcement learning with *bandit techniques*[2] are well-suited (Sutton and Barto, 1998; Lattimore and Szepesvári, 2020) More specifically, we view the selection between programs as a contextual multi-armed bandit with combinatorial actions: in general each goal-scene has multiple programs to choose from, corresponding to the arms of the bandit. To estimate the quality of each program, we maintain a table of Q-values, one for each action in $\mathcal{A}$, with the quality of the program being the product of the Q-values of its actions: $Q(p) = \prod_{i=1}^{|p|} \gamma Q(a_i)$.[3] This captures the trade-off between program length and the communicative accuracy of instructions. We empirically determine a value of $\gamma = 0.99$, which results in a small bias for shorter programs. To ensure that the architect also explores new programs, we adopt an ε-greedy strategy, where with probability $1-\varepsilon$ a random program (arm) is selected. We fix ε at a constant value of 0.1, ensuring a constant level of exploration. After the communication of the program (see below) we update the estimated value of its component actions as $Q(a) \leftarrow Q(a) + \alpha(r - Q(a))$, where $r$ is the reconstruction accuracy which is 1 if this instruction was successfully interpreted by the builder and 0 otherwise. This approach ensures that the new programs introduced after each Abstraction phase will be explored by the architect.

**Communicating a program** Having selected a program, $p$, the architect and builder play a one-step signalling game for each of its instructions $a_i$. This takes the form of $(x_i, a_i, x_{i+1})$: the grid state $x_i$ is transformed into $x_{i+1}$ by action $a_i$. The architect learns a neural communication policy $\pi_{comm}$, which produces a message $m_i = \pi_{comm}(a_i)$. Similarly, the builder learns a policy $\pi_{bldr}$ which estimates the next grid-state, and is defined as $\widehat{x}_{i+1} = \pi_{bldr}(x_i, m_i)$. These are both implemented as fully-connected neural networks. Since our messages are discrete, we use the gumbel-softmax relaxation to sample from discrete messages which makes our model end-to-end differentiable (Jang et al., 2017). The policies are jointly trained to minimise the binary cross-entropy loss between the target next state $x_{i+1}$ and the builder's output $\widehat{x}_{i+1}$. We also introduce a bias for positive signalling Eccles et al. (2019) in the architect's loss to encourage the architect's messages to carry meaningful information about the instruction they represent.

**Introducing Abstractions** After some rounds of communication, the architect enters the abstraction phase, where it searches for novel abstractions allowing for shorter programs for describing goal scenes. Abstractions are constructed by an improved version of the procedure used in McCarthy et al. (2021), where we are able to remove the explicit upper-size limit on the library size.

The architect evaluates the set of candidate abstractions extracted from their programs and picks the one which maximises (1):

$$P(\mathcal{A} \cup \{a_{cand}\} | \{p_i\}_{i=1}^{N}) \propto \prod_{i=1}^{N} P(p_i | \mathcal{A} \cup \{a_{cand}\}) \quad (1)$$

where $a_{cand}$ is the new candidate abstraction, $\{p_i\}_{i=1}^{N}$ are the known programs so far. The right side of the equation is further defined in 2. It provides a measure of the expected reduction in program length, which is defined in terms of the minimum description length (*MDL*) –the shortest program achievable using also the new candidate abstraction.

$$P(p_i | \mathcal{A} \cup \{a_{cand}\}) = exp(-MDL(p_i | \mathcal{A} \cup \{a_{cand}\})) \quad (2)$$

## Efficient Communication in PACE

As in other settings like recursive numeral systems (Denić and Szymanik, 2024), programs in the Architect-Builder game can be perfectly informative even with a minimal vocabulary – precisely of size two, with one term referring to the horizontal primitive block and the other referring to the vertical one. With just the two initial action-words the architect can describe infinitely complex goal-scenes, given enough time! This is because the semantics of the environment are

[2]an analogy coming from a gambler repeatedly having to select which arm of a slot machine to play.

[3]We empirically found that initialising the Q-values to 0 works best in practice.

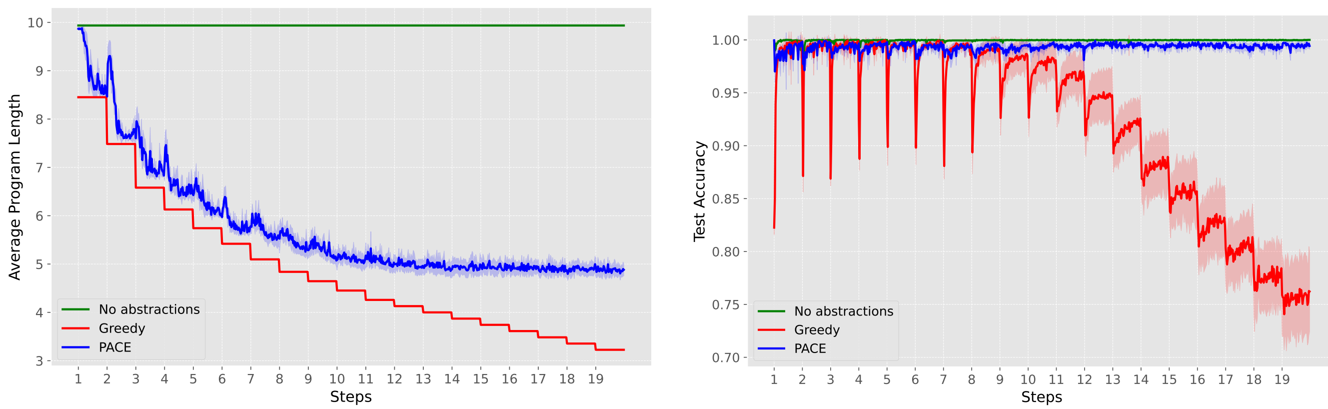



Figure 3: Comparison between PACE, *Greedy* and *No abstractions* in terms of program length and test reward over time. Line indicates mean value and shaded regions indicate the 95% confidence interval.

compositional. Thus, as Denić and Szymanik (2024) show for recursive numeral systems, we expect that the pressures in competition to shape the agents' languages in the Architect-Builder game are the average morphosyntactic complexity of the programs describing the goal-scenes and the language size. In our case this corresponds to a trade-off between the average length of the programs communicated by the architect, and the number of lexicalised terms, i.e. the number of primitives plus abstractions in $\mathcal{A}$, the architect's lexicon.

## Implementation Details

The neural agents are deep neural networks which have 1 and 2 hidden layers of size 200, respectively. The architect's output layer is of size 30, its maximum vocabulary size. While the output layer of the builder is of size 81, the size of the grid. We use the gumbel-softmax relaxation with the *hard* parameter so that the networks can be differentiated end-to-end. We use a learning rate of 0.0009 in conjunction with the ADAM optimiser (Kingma and Ba, 2014). The bandit hyperparameters $\alpha$, $\gamma$, $\varepsilon$ and $q_{init}$ are set to 0.5, 0.99, 0.1, and 0.0 respectively. Note, that before an abstraction is introduced we prune all but the 3 best programs (determined empirically) for each goal-scene in the architects symbolic table to not incur an exponential growth in the number of programs for each goal-scene. We also set the number of epochs $e$ to 40. All hyperparameters chosen are experimentally determined through grid search.

## Results & Discussion

Through empirical experimentation we want to investigate the following questions: 1) Does PACE display conversational tendencies which are similar to humans, allowing for more concise communication after multiple interactions? 2) Does communicative pressures impact what abstractions get adopted? If so, which abstractions are adopted and which are discarded? The behaviour of PACE is compared to two naïve baselines, to assess the impact of each of PACE components better. The first, *No abstractions*, is PACE without the abstraction phase. The second, *Greedy*, always picks the shortest program, eliminating the bandit. We use reward (also referred to as reconstruction accuracy) and average morphosyntactic complexity to compare these approaches. Unless otherwise stated results obtained are averaged over 16 runs.

**PACE reduces average morphosyntactic complexity** PACE does indeed reduce morphosyntactic complexity as seen in Figure 3 (left). Starting from an average morphosyntactic complexity of 9.95, over successive interaction steps, PACE is able to reduce average morphosyntactic complexity consistently and finally converges around a value of $4.92 \pm 0.20$ . When we compare PACE to the *No abstraction* variant, the average morphosyntactic complexity is halved, going from an average of 10, indicating that a more efficient language has been derived. The *Greedy* variant reduces program lengths even more, however, this comes at a price. In Figure 3 (right) we see drastic drops in accuracy for *Greedy* every time an abstraction is introduced, whereas PACE's slower introduction leads to smaller drops in accuracy. For a while, both strategies manage to recover near full accuracy, but after some time, the *Greedy* communication fails to recover and becomes progressively worse.

**Why does PACE converge?** As more terms are introduced and the average morphosyntactic complexity reduces, we find that it becomes harder to introduce new abstract terms. This explains why the *Greedy* approach diverges as it is forced to use new abstractions in its programs. This raises a profound question which is of interest within efficient communication: why does PACE introduce some abstractions whereas others are omitted? To understand this we conduct an experiment where at each step, before the communicative phase, we gather all candidate abstractions, group them for their size (as in how big the shape the abstraction refers to is in terms of primitive blocks) and frequency in our dataset, and then introduce them one by one into $\mathcal{A}$. For each candidate abstraction, we let the architect and builder communicate until

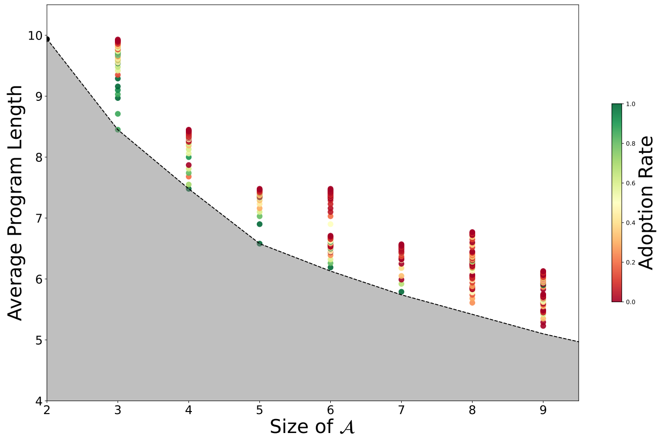



Figure 4: Adoption rate of possible abstractions by language size. As the language grows, fewer new abstractions are adopted. The dashed line represents the interpolation of the (discrete) Pareto Frontier calculated as the trade-off between average program length and size of the language $\mathcal{A}$. The grey area represents unachievable languages.

convergence and then probe the final language to check if the tested abstraction has been adopted or not. [4] This provides an exhaustive search of differing abstractions and allows us to analyse their ease of adoption or *learnability* into the current language (Steinert-Threlkeld and Szymanik, 2020), based on how much reduction in average morphosyntactic complexity they can potentially lead to. At the next step, the experiment is repeated with the new symbolic language $\mathcal{A}$ augmented with the new abstraction introduced at the previous Abstraction phase, making each step of this experiment conditioned on the current language of PACE.

We plot the results in Figure 4. We find that when $\mathcal{A}$ is only composed of primitive terms (the language is small), abstractions are generally always adopted but as $\mathcal{A}$ grows abstractions which would result in languages that are further away from the Pareto Frontier become harder to learn. Eventually, introducing new abstractions becomes infeasible, resulting in PACE's convergence. This demonstrates how communicative efficiency impact language formation within PACE, and connects with Steinert-Threlkeld and Szymanik (2020) which show that simplicity and ease of learning are intertwined.

**Which Abstractions are Adopted?** As discussed, PACE's final language does not include all abstractions suggested. In Figure 5, we show some that experience different fates. Figure 5 (left) shows the Q-value and frequency for the tower abstraction, consisting of two stacked vertical blocks, which is retained in PACE's final language. Figure 5 (right) instead shows the same for a more complex abstraction resembling an H rotated by 90 degrees, which after a trial period was not retained, judged as too hard to be learnt by the builder and thus discarded by the architect via the bandit. This is connected to the different frequencies of these two shapes in our dataset, with the tower being more than 20 times more common than the rotated H, resulting in the first one being easier for the builder to understand and ultimately adopted by the architect into their language.

In Figure 6, we show how the composition of our language changes throughout the repeated interactions. The relative proportion of primitives is reduced in favour of abstractions referring to either one of the 31 discrete shapes in the dataset (roughly 40%), or a sub-shape (roughly 20%). We show how this change in distribution impacts the composition of programs in Figure 7 with two goal-scenes examples. We see that the introduction of new abstractions enables for programs to be rewritten much more compactly.

## Related Literature

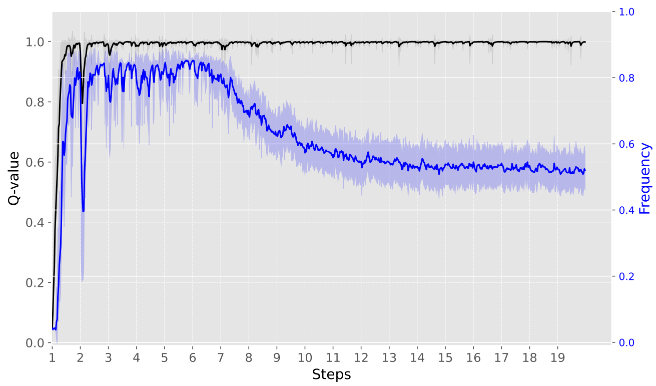
McCarthy et al. (2021) propose a Bayesian model of procedural abstractions that combines library learning (Ellis et al., 2021) with social reasoning-based communication (Goodman and Frank, 2016). While effective in capturing human-like trends, this model assumes a predefined mapping from instructions to builder's actions, essentially defining a priori the meaning of messages that might not have been introduced yet. They also explicitly constrain language size to prevent adding further abstractions after a certain number have been introduced. Jergéus et al. (2022) take a Deep Reinforcement Learning approach but oversimplify interactions by assuming the builder immediately understands abstractions, collapsing the setup into a single-agent framework and removing pressures for efficient communication. By using EC within PACE, our approach enables both agents to learn interactively, introducing abstractions organically and letting communicative pressures naturally limit language size.

Emergent Communication has been widely used to study various human language phenomena, such as compositionality (Mordatch and Abbeel, 2018; Chaabouni et al., 2020; Ren et al., 2020), the impact of populations (Kim and Oh, 2021; Chaabouni et al., 2022; Rita et al., 2022), and naming conventions in semantic categories (Carlsson et al., 2024). The architect-builder game presents a very challenging setting, being a repeated game with multiple interactions per round and sparse rewards. By incorporating symbolic methods with emergent communication, we provide a more interpretable, easy-to-use approach for modelling the introduction and use of procedural abstractions into conversational AI dyads.

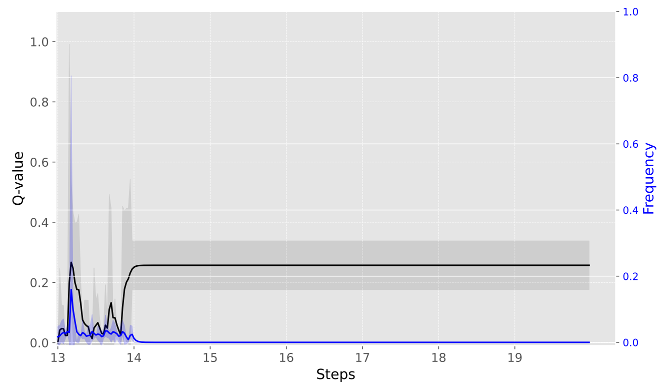
## Conclusion

In collaborative task-orientated communication humans tend towards more concise utterances by introducing procedural abstractions. In this work we propose a novel neuro-symbolic algorithm called PACE which displays similar tendencies. Our work serves as a bridge between procedural abstraction learning and efficient communication (Kemp and

[4] Note that to make this experiment computationally tractable we limit to sampling 3 abstractions for each frequency and size, wherever we have more than 3.




(a) The tower abstraction being introduced successfully.




(b) The rotated H abstraction gets eventually discarded.

Figure 5: Different abstractions being introduced. We show the mean and 95% confidence interval for Q-value (black) and frequency (blue) versus epochs (of which there are 40 in a step).

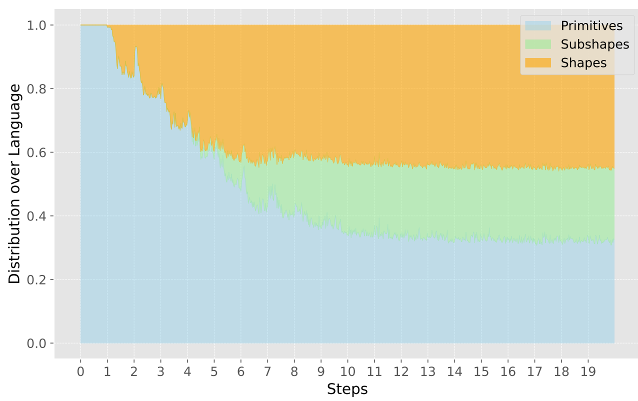



Figure 6: Relative proportion of actions as the language changes over training. Primitives refer to the initial actions, shapes refer to one of the 31 discrete shapes appearing in goal-scenes, and sub-shapes refer to anything else.

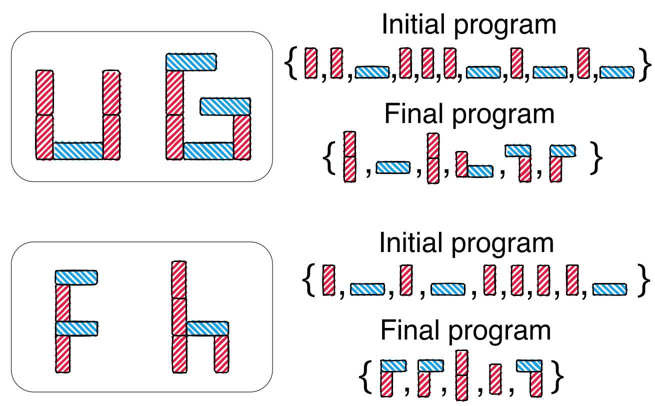



Figure 7: Two goal-scenes with representations of their initial and final programs chosen by PACE.

Regier, 2012; Gibson et al., 2017; Zaslavsky et al., 2019; Gibson et al., 2019; Denić and Szymanik, 2024; Carlsson et al., 2024). We demonstrate that more optimal languages are easier to learn reinforcing ideas relating learnability with efficient communication.

In future work, we intend to extend our analysis to consider other collaborative domains and deepen the comparison to human behaviour. We also intend to explore how Large Language Models understand and reason about these conversational dynamics. Significant research effort is being invested into exploring the role of natural language as a mechanism for humans to provide instructions to intelligent agents (Brohan et al., 2023; Shi et al., 2024). Providing these systems with the capability of handling novel procedural abstractions can facilitate improved cooperation between humans and agents. We believe that PACE represents a step towards equipping intelligent agents with flexible and extendable languages.

## Acknowledgments

This work was supported by funding from from the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and the Swedish Research Council. The computations in this work were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC). We also extend our thanks to Emil Carlsson and Vikas Garg for insightful discussions at various points throughout the project and to Sandro Stucki for valuable comments on a previous version of the manuscript that lead to this final one.

## References

Matthew Bowers, Theo X Olausson, Lionel Wong, Gabriel Grand, Joshua B Tenenbaum, Kevin Ellis, and Armando Solar-Lezama. Top-down synthesis for library learning. *Proceedings of the ACM on Programming Languages*, 7 (POPL):1182–1213, 2023.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. PMLR, 2023.

Emil Carlsson, Devdatt P. Dubhashi, and Fredrik D. Johansson. Learning approximate and exact numeral systems via reinforcement learning. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 43, 2021.

Emil Carlsson, Devdatt Dubhashi, and Terry Regier. Cultural evolution via iterated learning and communication explains efficient color naming systems. *Journal of Language Evolution*, 9(1-2):49–66, 2024.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. Compositionality and generalization in emergent languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4427–4442. Association for Computational Linguistics, 2020.

Rahma Chaabouni, Florian Strub, Florent Altché, Eugene Tarassov, Corentin Tallec, Elnaz Davoodi, Kory Wallace Mathewson, Olivier Tieleman, Angeliki Lazaridou, and Bilal Piot. Emergent communication at scale. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=AUGBfDIV9rL.

Milica Denić and Jakub Szymanik. Recursive numeral systems optimize the trade-off between lexicon size and average morphosyntactic complexity. *Cognitive Science*, 48 (3):e13424, 2024. 10.1111/cogs.13424.

Tom Eccles, Yoram Bachrach, Guy Lever, Angeliki Lazaridou, and Thore Graepel. Biases for emergent communication in multi-agent reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACpM sigplan international conference on programming language design and implementation*, pages 835–850, 2021.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Edward Gibson, Richard Futrell, Julian Jara-Ettinger, Kyle Mahowald, Leon Bergen, Sivalogeswaran Ratnasingam, Mitchell Gibson, Steven T. Piantadosi, and Bevil R. Conway. Color naming across languages reflects color use. *Proceedings of the National Academy of Sciences*, 2017. ISSN 0027-8424.

Edward Gibson, Richard Futrell, Steven P. Piantadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. How efficiency shapes human language. *Trends in Cognitive Sciences*, 23(5):389 – 407, 2019. ISSN 1364-6613.

Noah D Goodman and Michael C Frank. Pragmatic language interpretation as probabilistic inference. *Trends in cognitive sciences*, 20(11):818–829, 2016.

Robert D Hawkins, Michael C Frank, and Noah D Goodman. Characterizing the dynamics of learning in repeated reference games. *Cognitive science*, 44(6):e12845, 2020.

Mark K Ho, David Abel, Thomas L Griffiths, and Michael L Littman. The value of abstraction. *Current Opinion in Behavioral Sciences*, 29:111–116, 2019. ISSN 2352-1546. https://doi.org/10.1016/j.cobeha.2019.05.001. URL https://www.sciencedirect.com/science/article/pii/S2352154619300026. Artificial Intelligence.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2017. URL https://arxiv.org/abs/1611.01144.

Erik Jergéus, Leo Karlsson Oinonen, Emil Carlsson, and Moa Johansson. Towards learning abstractions via reinforcement learning. In *8th International Workshop on Artificial Intelligence and Cognition*, 2022. URL https://doi.org/10.48550/arXiv.2212.13980.

Charles Kemp and Terry Regier. Kinship categories across languages reflect general communicative principles. *Science (New York, N.Y.)*, 336:1049–54, 05 2012.

Jooyeon Kim and Alice Oh. Emergent communication under varying sizes and connectivities. *Advances in Neural Information Processing Systems*, 34:17579–17591, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Robert M Krauss and Sidney Weinheimer. Changes in reference phrases as a function of frequency of usage in social interaction: A preliminary study. *Psychonomic Science*, 1: 113–114, 1964.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Angeliki Lazaridou and Marco Baroni. Emergent multi-agent communication in the deep learning era, 2020.

Penny Lee. *The Whorf theory complex: A critical reconstruction*, volume 81. John Benjamins Publishing, 1996.

W. McCarthy, RD. Hawkins, C. Holdaway, H. Wang, and J. Fan. Learning to communicate about shared procedural abstractions. In *Proceedings of the 43rd Annual Conference of the Cognitive Science Society*, 2021.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Terry Regier, Charles Kemp, and Paul Kay. Word Meanings across Languages Support Efficient Communication. *The Handbook of Language Emergence*, (January 2015):237–263, 2015.

Yi Ren, Shangmin Guo, Matthieu Labeau, Shay B. Cohen, and Simon Kirby. Compositional languages emerge in a neural iterated learning model. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HkePNpVKPB`.

Mathieu Rita, Florian Strub, Jean-Bastien Grill, Olivier Pietquin, and Emmanuel Dupoux. On the role of population heterogeneity in emergent communication. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=5Qkd7-bZfI`.

Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections, 2024. URL `https://arxiv.org/abs/2403.12910`.

Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G. Barto, Yael Niv, and Matthew M. Botvinick. Optimal behavioral hierarchy. *PLOS Computational Biology*, 10(8):1–10, 08 2014. 10.1371/journal.pcbi.1003779. URL `https://doi.org/10.1371/journal.pcbi.1003779`.

Shane Steinert-Threlkeld and Jakub Szymanik. Ease of learning explains semantic universals. *Cognition*, 195:104076, 2020.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 1998.

Yang Xu, Emmy Liu, and Terry Regier. Numeral Systems Across Languages Support Efficient Communication: From Approximate Numerosity to Recursion. *Open Mind*, 4:57–70, 2020.

Kayo Yin, Terry Regier, and Dan Klein. Pressures for communicative efficiency in american sign language. In *Annual Conference of the Association for Computational Linguistics (ACL)*, August 2024.

Noga Zaslavsky, Charles Kemp, Naftali Tishby, and Terry Regier. Color naming reflects both perceptual structure and communicative need. *Topics in Cognitive Science*, 11(1): 207–219, 2019.