arXiv:2006.07043v1 [cs.LG] 12 Jun 2020

Language-Conditioned Goal Generation: a New Approach to Language Grounding in RL

Cédric Colas*[1] Ahmed Akakzia*[2] Pierre-Yves Oudeyer[1] Mohamed Chetouani[2] Olivier Sigaud[2]

Abstract

In the real world, linguistic agents are also embodied agents: they perceive and act in the physical world. The notion of *Language Grounding* questions the interactions between language and embodiment: how do learning agents connect or *ground* linguistic representations to the physical world ? This question has recently been approached by the Reinforcement Learning community under the framework of *instruction-following* agents. In these agents, behavioral policies or reward functions are conditioned on the embedding of an instruction expressed in natural language. This paper proposes another approach: using language to condition goal generators. Given any goal-conditioned policy, one could train a language-conditioned goal generator to generate language-agnostic goals for the agent. This method allows to decouple sensorimotor learning from language acquisition and enable agents to demonstrate a diversity of behaviors for any given instruction. We propose a particular instantiation of this approach and demonstrate its benefits.

1. Introduction

语言基础理论认为，语言习得强烈受到个体对物理世界的经验所影响。

Language Grounding describes the idea that language acquisition is strongly shaped by one's experience of the physical world. This idea emerged in Cognitive Science (Harnad, 1990; Glenberg & Kaschak, 2002; Zwaan & Madden, 2005) and quickly inspired Artificial Intelligence approaches in natural language processing (Roy & Reiter, 2005), human-machine interactions (Dominey, 2005; Madden et al., 2010) and more recently deep Reinforcement Learning (deep RL) (Luketina et al., 2019).

In the RL community, this has taken the form of *language-conditioned* agents (Hermann et al., 2017; Chan et al., 2019; Bahdanau et al., 2018; Cideron et al., 2019; Jiang et al., 2019; Zhong et al., 2019; Waytowich et al., 2019; Colas et al., 2020b). Language can be used to build representations (Waytowich et al., 2019) or to characterize the dynamics of the environment (Zhong et al., 2019). However, it is mostly used to represent instructions or goals (Hermann et al., 2017; Chan et al., 2019; Bahdanau et al., 2018; Cideron et al., 2019; Jiang et al., 2019; Colas et al., 2020b). ✓ In these approaches, natural language (NL) sentences are embedded through recurrent networks and merged with the agent's state to form the input of the policy or reward function. The language encoder is then trained jointly with either the former (Hermann et al., 2017; Chan et al., 2019; Jiang et al., 2019; Cideron et al., 2019; Hill et al., 2019) or the latter (Bahdanau et al., 2018; Colas et al., 2020b). These approach demonstrate benefits over traditional goal-conditioned methods:

1. *Targeting abstract goals.* Language can express abstract goals characterized by sets of properties the scene should verify (e.g. *block A* above *block B*). This contrasts with previous goal-as-state approaches, where goals are specific states of the agent (e.g. target pixel image, target block positions).
2. *Systematic generalization.* Previous language-conditioned approaches demonstrate strong generalization capabilities including generalizations to new combinations of action verbs and object attributes (colors, shapes, categories etc.) (Hermann et al., 2017; Bahdanau et al., 2018; Hill et al., 2019; Colas et al., 2020b).

However, this comes with drawbacks:

1. *Language becomes a pre-requisite for sensorimotor learning.* Pre-verbal infants are known to demonstrate goal-oriented behavior (Wood et al., 1976). Language-conditioned agents require language inputs to act in the world and, thus, cannot account for this decoupling.
2. *Lack of behavioral diversity.* Because policies are directly conditioned on language embeddings, an instruction in a given context only generates a low diversity of behaviors: a unique behavior for a deterministic policy and minor noise-induced behavioral variations for stochastic policies.

*Equal contribution [1]Flowers Team, Inria Bordeaux, FR. [2]Université Paris-Sorbonne, Paris, FR.. Correspondence to: Cédric Colas <cedric.colas@inria.fr>.

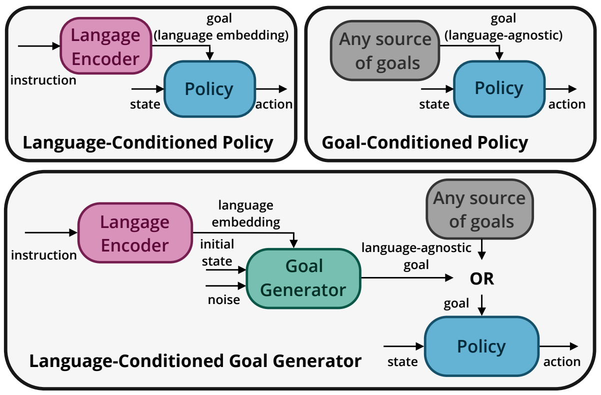

Figure 1. Language-conditioned policy, goal-conditioned policy and language-conditioned goal generator.

This paper proposes to leverage the abstraction and generalization capacities of language while 1) decoupling language acquisition from sensorimotor learning; 2) allowing behavioral diversity. This is achieved via *language-conditioned goal generation* (Figure 1). Instead of using language to condition policies directly, we use it to condition a generator of language-agnostic goals.

Language-agnostic goals can be represented by specific states (e.g. pixels inputs) (Péré et al., 2018; Laversanne-Finot et al., 2018; Nair et al., 2018; Warde-Farley et al., 2018; Nair et al., 2019; Pong et al., 2019) or handcrafted representations (e.g. target block coordinates) (Schaul et al., 2015; Andrychowicz et al., 2017; Florensa et al., 2018; Racaniere et al., 2019; Fournier et al., 2019; Colas et al., 2019). These can come from any source: uniform sampling of the goal space (Schaul et al., 2015), sampling from low-density areas (Pong et al., 2019), from high learning progress areas (Fournier et al., 2019; Colas et al., 2019), using generative models of states (Nair et al., 2018; 2019) or generative models targeting intermediate difficulty (Florensa et al., 2018; Racaniere et al., 2019). In our approach, language-conditioned goal generators are a source of language-agnostic goals among others. As a result, pure sensorimotor training does not require any language input: agents can simply use any other source of goals.

Language-conditioned goal generators are generative models of language-agnostic goals conditioned on instructions (e.g. Variational Auto-Encoders, Generative Adversarial Networks). For any given instruction, agent can thus sample a set of matching goals. This results in behavioral diversity, a set of diverse behaviors matching any given instruction.

Contributions. This paper introduces a novel approach to the problem of language grounding in RL agents: language-conditioned goal generation. This leverages the capability of language to represent abstract goals and to generalize, while avoiding the lack of behavioral diversity and the dependence to language inputs usually associated with language-conditioned agents. This paper presents a particular implementation of this approach and disentangles language acquisition from policy learning by assuming access to pre-trained goal-conditioned policies. Our policies are pre-trained to reach high-level configurations characterizing spatial relations between objects in the scene.

2. Methods

This paper presents a particular implementation of *language-conditioned goal generation*. Section 2.1 describes the learning environment and the pre-trained goal-conditioned policies. Assuming these, Section 2.2 describes an implementation of a language-conditioned goal generator and how it is used for language grounding.

2.1. Behavioral Policies Conditioned on Abstract Semantic Representations

The agents considered in this paper were pre-trained without language input by an algorithm presented in a companion paper.[1]

The *Fetch Manipulate* environment Agents evolve in the *Fetch Manipulate* environment: a robotic manipulation domain based on MUJOCO (Todorov et al., 2012) and adapted from the Fetch tasks (Plappert et al., 2018). Agents are 4 DoF robotic arms that face 3 colored blocks on a table (see Figure 2). They are given innate representations called *semantic configurations* that characterize spatial relations between blocks. During sensorimotor training, agents discovered all reachable configurations in that space and learned to master them.

Semantic configurations. In contrast to traditional approaches, goals are not defined as particular targets for each block but as high-level *semantic configurations*. These configurations are based on two spatial predicates infants demonstrate early in their development (Mandler, 2012): the *close* and the *above* binary predicates. These two predicates are applied to all permutations of object pairs, i.e. 6 permutations for the 3 objects we consider. Because the *close* predicate is order invariant, we only need to evaluate it on 3 object combinations. The *above* predicate being order dependent, we need all 6 permutations. The resulting binary vector of size 9 forms the *semantic configuration*. It represents the spatial relations between objects in the scene. In the resulting semantic configuration space $\{0,1\}^9$, the agent can reach 35 physically valid configurations, including stacks of 2 or 3 blocks and pyramids. Supplementary

[1]This paper is not cited here to respect anonymity. It will be after reviews.

Section 5 provides formal definitions and properties of predicates and semantic configurations. Figure 2 displays visual representations of some example configurations.

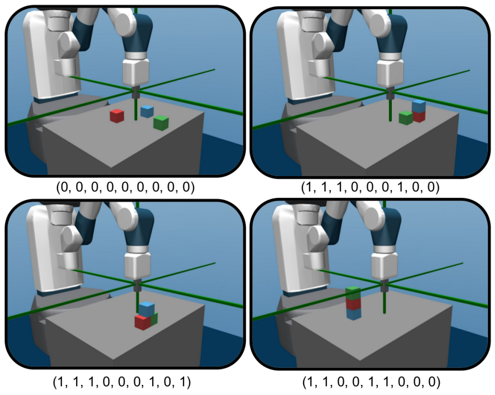

Figure 2. Some semantic configurations in *Fetch Manipulate*.

2.2. Language-Conditioned Goal Generation

The language-conditioned goal generator, or *language module*, generates semantic configurations matching the agent's initial configuration and a sentence describing an expected transformation of a relation between a pair of objects. This section explains its training and use for language grounding.

Training the language module. The language-conditioned goal generator is implemented by a conditional Variational Auto-Encoder (C-VAE) (Sohn et al., 2015) trained in a supervised setting. The training data is collected via interactions between a trained agent and a social partner. For each goal-directed trajectory the agent performs, the social partner provides the description of one of the resulting transformations in the object relations. The set of possible descriptions contains 102 sentences, each describing in a simplified language a positive or negative shift for each of the 9 predicates (e.g. *get red above green*). NL descriptions are encoded via a recurrent network that is jointly trained with the C-VAE. Supplementary Section 6 provides the list of sentences and implementation details.

Language grounding. At test time, agents are instructed by one of the 102 sentences to transform a relation between objects. The trained language module acts as a translator: agent can sample a goal configuration matching their current state and instruction. This module effectively enables agents to ground NL in their internal semantic representations and set of sensorimotor skills. We consider three evaluation settings: 1) performing a single instruction; 2) performing a sequence of instructions; 3) performing a logical combination of instructions. As the agent can generate a set of goals matching any instruction, it can easily combine these sets to perform logical functions of instructions: *and* is an intersection, *or* is an union and *not* is the complement within the set of goals the agent discovered during sensorimotor training. Given sets of compatible goal configurations, agents can also *try again*: find other goal configurations that match the required instruction when previous attempts failed.

3. Experiments

We first train a language-conditioned goal generator from a training dataset $\mathcal{D}$ collected via interactions between the agent and a social partner (Section 2.2). For a given initial configuration and a given sentence, we want the language module to generate all compatible final configurations, and just these.

Language-conditioned goal generation performance. To evaluate the language module, we construct a synthetic, oracle dataset $\mathcal{O}$ of triplets $(c_i,\ s,\ \mathcal{C}_f(c_i,\ s))$, where c_i is the initial configuration, s is the sentence describing the expected transformation and $\mathcal{C}_f(c_i,\ s)$ is the set of all final configurations compatible with $(c_i,\ s)$. Note that, on average, $\mathcal{C}_f$ in $\mathcal{O}$ contains 16.7 configurations, while the training dataset $\mathcal{D}$ only contains 3.4 (20%). We are interested in two metrics: 1) The *Compatibility Probability* (CP) is the probability that a goal sampled from the generator belongs to $\mathcal{C}_f$; 2) The *Coverage* (Cov) is the size of the intersection between $\mathcal{C}_f$ and the set resulting from sampling the language-conditioned generator 100 times. We compute these metrics on 5 different sets of input pairs (c_i, s), each calling for a different type of generalization:

1. Pairs found in $\mathcal{D}$, except pairs removed to form the following test sets. This calls for the extrapolation of known initialization-effect pairs (c_i, s) to new final configurations c_f ($\mathcal{D}$ contains only 20% of $\mathcal{C}_f$ on average).
2. Pairs that were removed from $\mathcal{D}$, calling for a recombination of known effects s on known c_i.
3. Pairs for which the c_i was entirely removed from $\mathcal{D}$. This calls for the transfer of known effects s on unknown c_i.
4. Pairs for which the s was entirely removed from $\mathcal{D}$. This calls for generalization in the language space, to generalize unknown effects s from related sentences and transpose this to known c_i.
5. Pairs for which both the c_i and the s were entirely removed from $\mathcal{D}$. This calls for the generalizations 3 and 4 combined.

Our language module demonstrates these 5 types of generalization (see Table 1). Agents can generate goals from situations they never encountered (Test 3). They can generalize the meaning of sentences they never heard (Test 4) and even apply the latter to unknown situations (Test 5). We detail the testing sets in Supplementary Section 6.

Table 1. Language module average metrics over 10 seeds. Std is below 0.07 for Cov and 0.06 for CP.

Metr.	Test 1	Test 2	Test 3	Test 4	Test 5
CP	0.93	0.94	0.95	0.90	0.92
Cov	0.97	0.93	0.98	0.99	0.98

Grounding language in sensorimotor behavior. We investigate how the language module interacts with the sensorimotor skills of the agent. We consider three evaluation settings. In the *transition* setup, we look at the average success rate of the agent when asked to perform the 102 instructions 5 times each, resetting the environment each time. In the *expression* setup, we evaluate the agent on 500 randomly generated logical functions of sentences. In both setups, we give the agent 5 attempts, enabling it to resample new compatible goals when the previous failed (without reset). Success rates after 1 (SR1) and 5 (SR5) attempts are reported in Table 2. In the *sequence* setup, we ask the agent to execute 20 random sequences of instructions without reset and report the average number of successes before the agent fails: $N_s = 14.9 \pm 5.7$ (std). The 10 RL agents are evaluated with the 10 C-VAE models evaluated above. These results show that the language module efficiently implements language grounding. Agents achieve instructed transitions almost all the time, resampling alternative goals when previous ones failed. They only fail when a previous trajectory kicked the blocks out of reach.

Table 2. Language grounding performance metrics over 10 seeds (av±std).

Metr.	Transition	Expression
SR1	0.89 ± 0.05	0.74 ± 0.08
SR5	0.99 ± 0.01	0.94 ± 0.06

4. Discussion

This paper introduces *language-conditioned goal generation*: a new approach to the problem of language grounding in RL agents. It shows the following advantages over traditional language-conditioned RL agents:

- **Decoupled language grounding.** Our approach enables agents to decouple language acquisition from sensorimotor learning, as it is observed in infants (Wood et al., 1976). However, nothing in the architecture prevents language from being grounded during sensorimotor learning. This would result in "overlapping waves" of sensorimotor and linguistic development (Siegler, 1998).
- **Behavioral diversity.** The language module generates a diversity of goals matching any instruction (see Coverage metrics in Table 1). This results in a behavioral diversity that language-conditioned agents cannot demonstrate. Indeed, while a language-conditioned agent trained on *put red close_to green* would only push the red block towards the green one, our agent can generate many matching goal configurations. It could build a pyramid, make a blue-green-red pile or target a dozen other compatible configurations.
- **Trying again.** Generating a diversity of goals matching any instruction enables agents to *try again*: to find alternative approaches to satisfy a same instruction when previous attempts failed. Table 2 shows that benefiting from several attempts significantly improves the chances of success. This could also improve robustness to perturbed environments where former successful behaviors fail.
- **Logical expressions.** Generating sets of compatible goals makes it easy to scale language understanding to any logical combination of instructions. This cannot be achieved with language-conditioned policies.

One strength of the language-conditioned approach is their capability of *systematic generalization* (Hermann et al., 2017; Bahdanau et al., 2018; Hill et al., 2019; Colas et al., 2020b). Our language module also seems to demonstrate generalization abilities: agents can generate goals from situations they never encountered (Type 3). They can generalize the meaning of sentences they never heard (Type 4) and even apply the latter to unknown situations (Type 5). Type 4, especially, involves recombinations of action verbs and attributes similar to Hill et al. (2019); Colas et al. (2020b).

C-VAE were already used to generate goals matching initial states in Nair et al. (2019). However, the additional condition on instructions brings the benefits of language use: 1) the representation of abstract goals; 2) the systematic generalization capacities. In addition instructions enable agents to control their goal generation. Because image-based goal generation was shown to work in Nair et al. (2018); Pong et al. (2019); Nair et al. (2019), we believe our language-conditioned goal generator could be trained to generate imaged-based goals.

Further work could investigate the extension of language-conditioned goal generator to diverse goal-conditioned settings such as, for example, *Quality-Diversity* algorithms. These algorithms train populations of diverse and high-performing solutions to a problem (Cully & Demiris, 2017). Each solution is associated with its *behavioral characterization*: a low-dimensional description of its behavior. Our language-conditioned goal generator could be used to map language instructions to that behavioral space, enabling a language-based control of these diversity-seeking evolutionary algorithms (Mouret & Clune, 2015; Lehman & Stanley, 2011; Conti et al., 2018; Colas et al., 2020a).

Acknowledgments

Cédric Colas is partly funded by the French Ministère des Armées - Direction Générale de l'Armement.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight Experience Replay. *arXiv preprint arXiv:1707.01495*, 2017.

Bahdanau, D., Hill, F., Leike, J., Hughes, E., Hosseini, A., Kohli, P., and Grefenstette, E. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*, 2018.

Chan, H., Wu, Y., Kiros, J., Fidler, S., and Ba, J. Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *arXiv preprint arXiv:1902.04546*, 2019.

Cideron, G., Seurin, M., Strub, F., and Pietquin, O. Self-educated language agent with hindsight experience replay for instruction following. *arXiv preprint arXiv:1910.09451*, 2019.

Colas, C., Oudeyer, P.-Y., Sigaud, O., Fournier, P., and Chetouani, M. CURIOUS: Intrinsically motivated multi-task, multi-goal reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 1331–1340, 2019.

Colas, C., Huizinga, J., Madhavan, V., and Clune, J. Scaling map-elites to deep neuroevolution. *arXiv preprint arXiv:2003.01825*, 2020a.

Colas, C., Karch, T., Lair, N., Dussoux, J.-M., Moulin-Frier, C., Dominey, P. F., and Oudeyer, P.-Y. Language as a cognitive tool to imagine goals in curiosity-driven exploration. *arXiv preprint arXiv:2002.09253*, 2020b.

Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K., and Clune, J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, pp. 5027–5038, 2018.

Cully, A. and Demiris, Y. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017.

Dominey, P. F. Emergence of grammatical constructions: evidence from simulation and grounded agent experiments. *Connection Science*, 17(3-4):289–306, sep 2005. ISSN 0954-0091. doi: 10.1080/09540090500270714.

Florensa, C., Held, D., Geng, X., and Abbeel, P. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pp. 1514–1523, 2018.

Fournier, P., Colas, C., Chetouani, M., and Sigaud, O. Clic: Curriculum learning and imitation for object control in non-rewarding environments. *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

Glenberg, A. M. and Kaschak, M. P. Grounding language in action. *Psychonomic Bulletin & Review*, 9(3):558–565, sep 2002. ISSN 1069-9384. doi: 10.3758/BF03196313.

Harnad, S. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., Teplyashin, D., et al. Grounded language learning in a simulated 3D world. *arXiv preprint arXiv:1706.06551*, 2017.

Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A. Emergent systematic generalization in a situated agent, 2019.

Jiang, Y., Gu, S. S., Murphy, K. P., and Finn, C. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 9414–9426, 2019.

Laversanne-Finot, A., Péré, A., and Oudeyer, P.-Y. Curiosity driven exploration of learned disentangled goal spaces. *arXiv preprint arXiv:1807.01521*, 2018.

Lehman, J. and Stanley, K. O. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 211–218. ACM, 2011.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

Madden, C., Hoen, M., and Dominey, P. F. A cognitive neuroscience perspective on embodied language for human–robot cooperation. *Brain and Language*, 112(3):180–188, mar 2010. ISSN 0093-934X. doi: 10.1016/J.BANDL.2009.07.001.

Mandler, J. M. On the spatial foundations of the conceptual system and its enrichment. *Cognitive science*, 36(3):421–451, 2012.

Mouret, J.-B. and Clune, J. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Nair, A., Bahl, S., Khazatsky, A., Pong, V., Berseth, G., and Levine, S. Contextual imagined goals for self-supervised robotic learning. *arXiv preprint arXiv:1910.11670*, 2019.

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9191–9200, 2018.

Péré, A., Forestier, S., Sigaud, O., and Oudeyer, P.-Y. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. *arXiv preprint arXiv:1803.00781*, 2018.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Pong, V. H., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Racaniere, S., Lampinen, A. K., Santoro, A., Reichert, D. P., Firoiu, V., and Lillicrap, T. P. Automated curricula through setter-solver interactions. *arXiv preprint arXiv:1909.12892*, 2019.

Roy, D. and Reiter, E. Connecting language to the world. *Artificial Intelligence*, 167(1-2):1–12, 2005.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

Siegler, R. S. *Emerging minds: The process of change in children's thinking*. Oxford University Press, 1998.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Warde-Farley, D., Van de Wiele, T., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

Waytowich, N., Barton, S. L., Lawhern, V., Stump, E., and Warnell, G. Grounding natural language commands to starcraft ii game states for narration-guided reinforcement learning. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 110060S. International Society for Optics and Photonics, 2019.

Wood, D., Bruner, J. S., and Ross, G. The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100, 1976.

Zhong, V., Rocktäschel, T., and Grefenstette, E. Rtfm: Generalising to novel environment dynamics via reading. *arXiv preprint arXiv:1910.08210*, 2019.

Zwaan, R. and Madden, C. Embodied sentence comprehension. *Grounding Cognition: The Role of Perception and Action in Memory, Language, and Thinking*, pp. 224–245, 2005. doi: 10.1017/CBO9780511499968.010.

Supplementary Material

This supplementary material includes:

- Section 5: a formal definition of semantic configurations and the definiton of the semantic configurations used in the *Fetch Manipulate* environment.
- Section 6: further details about the training of our language-conditioned goal generation module.

5. Formal Definition of Semantic Configurations

Semantic configurations are based on a collection of formal systems known as *predicate logic* or *first-order logic*. They use *k-ary relations* to describe possible connections between k quantified variables. This paper focuses on *spatial binary predicates* characterizing spatial relations between pairs of physical objects. We provide formal definitions, properties and examples below.

Binary predicates Consider a finite set of objects $O = \{o_1,\ o_2,\ ...,\ o_M\}$. A binary predicate p associated with a semantic relation **r** is an expression that takes as input any ordered pair of objects $(o_i,\ o_j) \in O^2$. $p(o_i,\ o_j)$ is said *true* if and only if "o_i **r** o_j" is verified. For simplicity, we refer to p and **r** interchangeably.

Examples of binary predicates. We consider the objects o_1 and o_2.

- The expression "o_1 is **close** to o_2" describes the predicate *close* evaluated on (o_1, o_2).
- The expression "o_2 is **above** o_1" describes the predicate *above* evaluated on (o_2, o_1).

Semantic mapping functions. To achieve symbol grounding into non-symbolic sensorimotor interactions using predicates, we define a *semantic mapping function* f associated with the binary predicate p as the probability that p is true given the states of the considered objects. Formally, if we consider the objects o_i, o_j and their respective states s_i, s_j, then:

$$f(s_i,\ s_j) = \mathrm{P}\left(p(o_i,\ o_j) \mid s_i,\ s_j\right).$$

This paper assumes oracle deterministic semantic mapping functions, i.e. f is a Boolean function in $\{0,1\}$. Practically, we hard-code a function, assumed internal to the agent, that uses predefined fixed thresholds to determine whether a predicate is true or false given the states of the considered objects. For example, for the *close* predicate, it outputs 1 if and only if the Euclidean distance between the two considered objects is below a defined threshold. For the sake of simplicity, we omit the word *deterministic*.

Symmetry and asymmetry. Consider a finite set of objects $O = \{o_1,\ o_2,\ ...,\ o_M\}$ and a binary predicate p. The predicate p is said to be *symmetric* if and only if, for any ordered pair of objects $(o_i,\ o_j) \in O^2$, "o_i **r** o_j" and "o_j **r** o_i" are equivalent. As a result, the corresponding semantic mapping function f needs to be symmetric, i.e. $f(o_i,\ o_j) = f(o_j,\ o_i)$. The predicate p is said to be *asymmetric* iff, for any ordered pair $(o_i,\ o_j) \in O^2$, "o_i **r** o_j" implies **not** "o_j **r** o_i".

Examples. We consider the objects o_1 and o_2.

- *close* is symmetric: "o_1 is **close** to o_2" $\Leftrightarrow$ "o_2 is **close** to o_1". The corresponding semantic mapping function is based on the Euclidean distance, which is symmetric.
- *above* is asymmetric: "o_1 is **above** o_2" $\Rightarrow$ **not** "o_2 is **above** o_1". The corresponding semantic mapping function evaluates the sign of the difference of the object Z-axis coordinates.

Effective number of predicate relations. Consider a finite set of M objects $O = \{o_1, o_2, ..., o_M\}$ and a binary predicate p.

- If p is not symmetric, then the effective number of relations K_p that can be described without redundancy is equal to the number of **permutations** of 2 objects among M, i.e. $K_p = A_{M,2} = M(M-1)$.
- If p is symmetric, then the effective number of relations K_p is equal to the number of **combinations** of 2 objects among M, i.e. $K_p = \binom{M}{2} = \frac{M(M-1)}{2}$.

Semantic configurations based on spatial relations. Let $(p_i)_{i\in[1..P]}$ be a list of P binary predicates. The concatenation of the evaluations of the semantic mapping functions f_i on the K_{pi} pairs of objects forms a *semantic configuration*. It is an abstract representation of a scene which characterizes all relations defined by the (p_i) predicates among the M objects. This defines a binary *semantic configuration space* $\mathcal{C}_p = \{0,1\}^{K_c}$, where $K_c = \sum_{i=1}^{P} K_{p_i}$. If any world configuration can be mapped to $\mathcal{C}_p$, not all configurations are reachable (e.g. o_1 cannot be *above* and *below* o_2 at the same time).

Semantic representation space in *Fetch Manipulate*. In the *Fetch Manipulate* environment, we restrict semantic representations to the use of the *close* and *above* binary predicates applied on $M = 3$ objects. The resulting semantic configurations are formed by:

$$\begin{aligned} c_p = [&c(o_1,o_2),\ c(o_1,o_3),\ c(o_2,o_3),\ a(o_1,o_2),\\ &a(o_2,o_1),\ a(o_1,o_3),\ a(o_3,o_1),\ a(o_2,o_3),\ a(o_3,o_2)], \end{aligned}$$

where *c()* and *a()* refer to the *close* and *above* predicates respectively and $(o_1,\ o_2,\ o_3)$ are the red, green and blue blocks respectively.

6. Language-Conditioned Goal Generator Training

We use a conditional Variational Auto-Encoder (C-VAE) (Sohn et al., 2015). Conditioned on the initial configuration and a sentence describing the expected transformation of one object relation, it generates compatible goal configurations. After the first phase of goal-directed sensorimotor training, the agent interacts with a hard-coded social partner as described in Main Section 2.2. From these interactions, we obtain a dataset of 5000 triplets: initial configuration, final configuration and sentence describing one change of predicate from the initial to the final configuration. The list of sentences used by the synthetic social partner are provided in Table 3. Note that *red*, *green* and *blue* refer to objects o_1, o_2, o_3 respectively.

Content of test sets. We describe the 5 test sets:

1. Test set 1 is made of input pairs $(c_i,\ s)$ from the training set, but tests the coverage of all compatible final configurations

$\mathcal{C}_f$, 80% of which are not found in the training set. In that sense, it is partly a test set.

2. Test set 2 contains two input pairs: {[0 1 0 0 0 0 0 0 0], *put blue close_to green*} and {[0 0 1 0 0 0 0 0 0], *put green below red*} corresponding to 7 and 24 compatible final configurations respectively.

3. Test set 3 corresponds to all pairs including the initial configuration c_i = [1 1 0 0 0 0 0 0 0] (29 pairs), with an average of 13 compatible final configurations.

4. Test set 4 corresponds to all pairs including one of the sentences *put green on_top_of red* and *put blue far_from red*, i.e. 20 pairs with an average of 9.5 compatible final configurations.

5. Test set 5 is all pairs that include both the initial configuration of test set 3 and one of the sentences of test set 4, i.e. 2 pairs with 6 and 13 compatible goals respectively. Note that pairs of set 5 are removed from sets 3 and 4.

Table 3. List of instructions. Each of them specifies a shift of one predicate, either from false to true ($0 \rightarrow 1$) or true to false ($1 \rightarrow 0$). **block A** and **block B** represent two different blocks from {red, blue, green}.

Type	Sentences
Close $0 \rightarrow 1$	*Put* ***block A*** *close_to* ***block B***,
	Bring ***block B*** *and* ***block A*** *together*,
	Put ***block B*** *close_to* ***block A***,
	Bring ***block A*** *and* ***block B*** *together*,
	Get ***block B*** *and* ***block A*** *close_from each_other*,
	Get ***block A*** *close_to* ***block B***
	Get ***block A*** *and* ***block B*** *close_from each_other*,
	Get ***block B*** *close_to* ***block A***.
Close $1 \rightarrow 0$	*Put* ***block A*** *far_from* ***block B***,
	Get ***block A*** *far_from* ***block B***,
	Put ***block B*** *far_from* ***block A***,
	Get ***block B*** *far_from* ***block A***,
	Get ***block A*** *and* ***block B*** *far_from each_other*,
	Bring ***block A*** *and* ***block B*** *apart*,
	Get ***block B*** *and* ***block A*** *far_from each_other*,
	Bring ***block B*** *and* ***block A*** *apart*.
Above $1 \rightarrow 0$	*Put* ***block A*** *above* ***block B***,
	Put ***block A*** *on_top_of* ***block B***,
	Put ***block B*** *under* ***block A***,
	Put ***block B*** *below* ***block A***.
Above $1 \rightarrow 0$	*Remove* ***block A*** *from_above* ***block B***,
	Remove ***block A*** *from* ***block B***,
	Remove ***block B*** *from_below* ***block A***,
	Put ***block B*** *and* ***block A*** *on_the_same_plane*,
	Put ***block A*** *and* ***block B*** *on_the_same_plane*.

Testing on logical expressions of instructions. To evaluate our agents on logical functions of instructions, we generate three types of expressions:

1. 100 instructions of the form "A and B" where A and B are basic instructions corresponding to shifts of the form *above* $0 \rightarrow 1$ (see Table 3). These intersections correspond to stacks of 3 or pyramids.

2. 200 instructions of the form "A and B" where A and B are *above* and *close* instructions respectively. B can be replaced by "not B" with probability 0.5.

3. 200 instructions of the form "(A and B) or (C and D))", where A, B, C, D are basic instructions: A and C are *above* instructions while B and D are *close* instructions. Here also, any instruction can be replaced by its negation with probability 0.5.

Hyperparameters. The encoder is a fully-connected neural network with two layers of size 128 and *ReLU* activations. It takes as input the concatenation of the final binary configuration and its two conditions: the initial binary configuration and an embedding of the NL sentence. The NL sentence is embedded with an recurrent network with embedding size 100, *tanh* non-linearities and biases. The encoder outputs the mean and log-variance of the latent distribution of size 27. The decoder is also a fully-connected network with two hidden layers of size 128 and *ReLU* activations. It takes as input the latent code z and the same conditions as the encoder. As it generates binary vectors, the last layer uses *sigmoid* activations. We train the architecture with a mixture of Kullback-Leibler divergence loss (KD_{loss}) w.r.t a standard Gaussian prior and a binary Cross-Entropy loss (BCE_{loss}). The combined loss is loss $= BCE_{\text{loss}} + \beta \times KD_{\text{loss}}$ with $\beta = 0.6$. We use an Adam optimizer with a learning rate of 5×10^{-4}, a batch size of 128 and optimize for 150 epochs. As training is fast ($\approx$2 min on a single cpu), we conducted a quick hyperparameter search over β, layer sizes, learning rates and latent sizes (see Table 4). We found robust results for various layer sizes, various β below 1 and latent sizes above 9.

Table 4. Language module hyperparameter search. In bold are the selected hyperparameters.

Hyperparam.	Values.
β	[0.5, **0.6**, 0.7, 0.8, 0.9, 1.]
layers size	[**128**, 256]
learning rate	[0.01, **0.005**, 0.001]
latent sizes	[9, 18, **27**]