





# Environment Descriptions for Usability and Generalisation in Reinforcement Learning

Dennis J. N. J. Soemers<sup>1</sup><sup>a</sup>, Spyridon Samothrakis<sup>2</sup><sup>b</sup>, Kurt Driessens<sup>1</sup><sup>c</sup>  
and Mark H. M. Winands<sup>1</sup><sup>d</sup>

<sup>1</sup>*Department of Advanced Computing Sciences, Maastricht University, Maastricht, The Netherlands*

<sup>2</sup>*Institute for Analytics and Data Science, University of Essex, Colchester, U.K.*

**Keywords:** Environment Descriptions, Generalisation, Reinforcement Learning.

**Abstract:** The majority of current reinforcement learning (RL) research involves training and deploying agents in environments that are implemented by engineers in general-purpose programming languages and more advanced frameworks such as CUDA or JAX. This makes the application of RL to novel problems of interest inaccessible to small organisations or private individuals with insufficient engineering expertise. This position paper argues that, to enable more widespread adoption of RL, it is important for the research community to shift focus towards methodologies where environments are described in user-friendly domain-specific or natural languages. Aside from improving the usability of RL, such language-based environment descriptions may also provide valuable context and boost the ability of trained agents to generalise to unseen environments within the set of all environments that can be described in any language of choice.


## 1 INTRODUCTION


Reinforcement learning (RL) (Sutton and Barto, 2018) researchers have largely converged on common APIs for the development of benchmark domains used to evaluate RL algorithms for sequential decision-making problems. New environments (problems) are customarily written in general-purpose programming languages such as C++ or Python, implementing a Gym-like (Brockman et al., 2016) API for algorithms to interface with environments.


We may distinguish two broad categories of RL research. On the one hand, there is research focusing on the development of (modifications of) training algorithms, typically not focused on any specific task. There may be a focus on certain categories of tasks (single-agent RL, multi-agent RL, RL for partially observable environments, and so on), but existing and established frameworks with a suite of applicable domains are typically used for empirical evaluations. Researchers typically aim to demonstrate a high level of generality, by showing that an algorithm can effectively learn on a large collection of different environments within such a suite, as opposed to only a single environment. These environ-


ments are often games or other simulations (Machado et al., 2018; Tassa et al., 2018; Cobbe et al., 2020; Ellis et al., 2023), with arguably limited direct real-world impact outside of their use as benchmarks for RL research. On the other hand, there is research in which a concrete, high-impact “real-world” task is selected, and RL is used to improve performance on that one task. Substantial engineering effort is often dedicated towards implementing and optimising a simulator for such a task. This engineering effort often requires specialised knowledge of, for example, programming for GPUs or other hardware accelerators, and of the inner workings of deep learning and RL algorithms.

While discussions on experimental methodologies in RL have been on the rise (Henderson et al., 2018; Agarwal et al., 2021; Jordan, 2022; Patterson et al., 2023; Jordan et al., 2024; Voelcker et al., 2024), we see little discussion on how (or by whom) tasks (or environments) are described or implemented in the first place. In this position paper, we argue that the standard assumption that environments can be implemented (and heavily optimised) in general-purpose programming languages, by engineers familiar with machine learning, (i) poses a challenge to widespread adoption of RL for real-world use cases, and (ii) also leads the research community to miss out on interesting research directions with respect to generalisa-

<sup>a</sup> <https://orcid.org/0000-0003-3241-8957>

<sup>b</sup> <https://orcid.org/0000-0003-1902-9690>

<sup>c</sup> <https://orcid.org/0000-0001-7871-2495>

<sup>d</sup> <https://orcid.org/0000-0002-0125-0824>

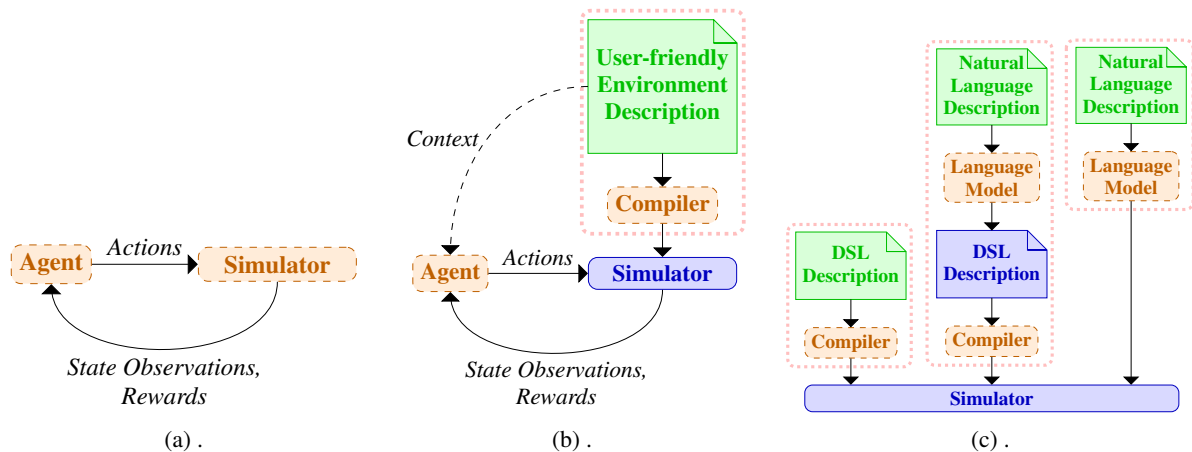


Figure 1: Orange boxes with dashed lines represent components that require substantial engineering or RL expertise. The green components can be provided by users with little to no engineering and RL expertise. **(a)** A depiction of the customary setting in most current RL research. **(b)** The approach for which this paper posits that increased research attention is warranted (Section 2), which also leads to interesting avenues for improving (zero-shot) generalisation in RL (Section 3). **(c)** User-friendly environment descriptions may be written in a DSL, or in a natural language, where the latter approach may or may not also generate an intermediate DSL description.

tion and transfer in RL. While it may be acceptable to invest substantial engineering resources for the implementation of environments for large-scale projects with high potential impact, it impedes the application of RL by smaller organisations or private individuals. We posit that more widespread applications of RL will be greatly aided if the latter groups can express their tasks in user-friendly domain-specific languages (DSLs) (Mernik et al., 2005; Aram and Neumann, 2015), or even in natural language. There are many possible definitions and interpretations of the term “user-friendly” (Stevens, 1983), but as a working definition, we will say that *a language is user-friendly if it is easy to use for users who may be experts in their application domain of interest, but may not have any RL, AI, or programming expertise, and is designed with their needs in mind.*

Once we adopt a methodology where environments are represented in explicit forms that can be provided as inputs to an agent (e.g., DSL or natural language snippets), we can also explore new forms of generalisation or transfer in RL, where effective generalisation or zero-shot transfer to unseen environments may become feasible given sufficient understanding of the task descriptions. Figure 1a depicts the setting where the environment is implemented directly in a general-purpose programming language, and Figure 1b depicts the proposed settings, with Figure 1c providing three examples of how the translation from a user-friendly environment description to a simulator may work. For tasks that take place in the physical world, such as non-simulated robotics tasks, a description of the reward function can suf-

fice, as hardware and the real world already define aspects such as the action space and transition dynamics. However, even in these cases, the ability to automatically generate a sufficiently accurate simulator from user-friendly descriptions would, in combination with sim-to-real transfer (Zhao et al., 2020), still be highly beneficial (Yang et al., 2024).

## 2 DESCRIPTION LANGUAGES FOR ENVIRONMENTS

Subsection 2.1 describes the established practice where RL research uses environments implemented, conforming to a standardised API, in general-purpose programming languages. As an initial step towards more user-friendly descriptions, Subsection 2.2 discusses the use of DSLs for describing environments used in RL research. Subsection 2.3 explores the possibility of using natural languages to define environments—arguably one of the most user-friendly modalities. Finally, Subsection 2.4 presents the central position of this paper: a call for more (research attention for) benchmarks in which environments are described in DSLs or natural language.

### 2.1 Defining Environments in Programming Languages

Outside of robotics work applied directly in the physical world, it is customary to implement the environments used for RL research in programming lan-

guages such as C++ or Python. In a recent trend, more specialised toolkits, such as CUDA or JAX (Bradbury et al., 2018) are used to enable the environments themselves—and not just DNN forward and backward passes—to make efficient use of hardware accelerators (Dalton and Frosio, 2020; Freeman et al., 2021; Lange, 2022; Koyamada et al., 2023). This can provide dramatic speed increases, but also imposes additional constraints on programming style and requires more specialised engineering skills.

Most developers of RL environments have converged to the API popularised by OpenAI Gym (Brockman et al., 2016). This API requires developers to implement:

- A definition of the *observation space*. For any state that an agent may ever reach in an environment, it will receive an observation from this space as input.
- A definition of the *action space*  $\mathcal{A}$ . It is typically assumed that agents must select any one element from this space as their action in each non-terminal state.
- A function to *reset* the environment to an initial state.
- A *step* function, which takes an action from  $\mathcal{A}$  as input, transitions from a current state  $s \in \mathcal{S}$  to a successor state  $s' \in \mathcal{S}$ , and returns a real-valued reward  $r$  and an observation of  $s'$ .

## 2.2 DSLs for Environments

A potential alternative to the standard practice of programming environments, is to use DSLs to describe sets of environments. This approach still requires significant engineering effort to develop a compiler that can translate descriptions from the DSL to a runnable simulator with an API for (learning) agents. However, once this compiler has been built, users with little to no programming experience may—depending on the complexity and user-friendliness of the DSL in question—use it to describe new environments that fit within the overarching domain supported by the DSL.

Numerous examples of DSLs for describing sequential decision-making problems already exist, though their adoption as benchmarks in the RL community is limited compared to benchmarks such as the Arcade Learning Environment (Bellemare et al., 2013; Machado et al., 2018) or the DeepMind Control Suite (Tassa et al., 2018), which are not based on DSLs. Examples include PDDL (McDermott et al., 1998) for planning problems, and the Stanford Game Description Language (Love et al., 2008; Genesereth and Thielscher, 2014), Ludii (Piette et al., 2020), and

MiniHack (Samvelyan et al., 2021) for various ranges of games. PDDLGym (Silver and Chitnis, 2020) provides Gym environment wrappers around PDDL problems.

## 2.3 Describing Environments in Natural Language

While DSLs may already be considered a more user-friendly alternative to general-purpose programming languages (Mernik et al., 2005; Aram and Neumann, 2015) for describing environments, natural language would be even more accessible to a wider userbase. Although the state of the art of large language models (LLMs) is highly impressive (Zhao et al., 2023), there are still concerns surrounding reliability and correctness (Marcus et al., 2023). Ambiguities typically present in natural languages, as well as the tendency for humans to underspecify task descriptions (e.g., rules of games), present challenges that require further research. Recently, Afshar and Li (2024) demonstrated promising initial results for an LLM generating executable environment code from natural language descriptions, but it still requires an expert human who is able to interpret the generated code and provide feedback on potential mistakes. In the short term, it may be more realistically feasible to use a combination of natural language and DSLs, where an LLM first translates a natural language description to a DSL-based description (Desai et al., 2016; Oswald et al., 2024; Zuo et al., 2024), and a user can inspect the generated description and make corrections if necessary. In the long term, if LLMs can be made sufficiently reliable, natural languages would likely be the most accessible modality for describing environments.

## 2.4 Research Focus on Description Languages for Environments

Before formally stating the central position of this paper, we make two assumptions relating to the user-friendliness of DSLs and natural languages (Assumption 1), and the desirability of this user-friendliness (Assumption 2).

**Assumption 1.** Defining environments in DSLs or natural languages can be more user-friendly than general-purpose programming languages.

Increasing user-friendliness and lowering barriers to entry is a well-established motivation for the use of DSLs (Mernik et al., 2005; Aram and Neumann, 2015). Note that there may also be other reasons for using DSLs, and there can be DSLs that do not

substantially lower barriers to entry: this depends on the design of the DSL in question. For example, the logic-based Stanford Game Description Language (Love et al., 2008; Genesereth and Thielscher, 2014) arguably still requires substantial technical expertise, and writing games in it may be considered error-prone due to the large file size required for many games. In contrast, allowing for clear and succinct descriptions that are easy to read and write was an explicit design goal for Ludii’s description language (Piette et al., 2020). Likely in no small part due to the language’s level of accessibility, Ludii has amassed a library of over 1200 distinct official game descriptions,<sup>1</sup> including third-party contributions from game designers with little or no programming experience.<sup>2</sup>

In the case of natural languages, if any concerns around ambiguities and underspecification of environments can be adequately addressed, we see little reason to doubt that many users would indeed find them more accessible than programming languages. If procedures translating natural language descriptions directly into executable simulations cannot be made sufficiently reliable, a potential solution may be to use DSLs as an intermediate step. Users could first describe their tasks in natural languages, and ideally only have to verify or fix small issues in automatically generated DSL descriptions afterwards.

**Assumption 2.** Enabling environments to be defined in more user-friendly ways is desirable.

First, we will acknowledge that lowering the barrier to entry for defining environments is not necessarily *always* of importance. For example, when RL is applied to an individual, specific domain with a high degree of scientific, societal, economic or other form of impact, it will often be worth investing substantial engineering effort into the environment definition. However, running such projects tends to be restricted to groups with direct access to RL experts.

A survey among AI engineers, AI designers, and RL engineers from AAA video game studios, independent developers, and industrial research labs—most of which do have direct access to a substantial amount of engineering expertise—revealed, among other concerns, an overreliance on engineering support, and difficulties in designing tasks for RL agents, as challenges for the adoption of RL and other AI techniques in video game development (Jacob et al., 2020). While not focused on RL (but, rather, AI in general) or environment descriptions, a recent study by Simkute et al. (2024) reveals a substantial disconnect between the technical know-how that AI design-

ers expect users will have, and what they actually tend to have, as a core barrier to adoption of AI in practice. These studies point to the relevance of improving the user-friendliness of any aspect of the RL (or any AI) pipeline.

If we wish to democratise the use of AI (Seger et al., 2023) to the extent that users with little expertise in RL—or even programming—can apply it to their problems of interest, enabling environments to be defined in more user-friendly ways would be a requirement. Outside of RL, in the landscape of generative artificial intelligence (AI), substantial value is generated not necessarily just by the models themselves, but also by the release of user-friendly tools and interfaces to access the trained models. Famous examples include OpenAI’s ChatGPT (OpenAI, 2022), and Gradio apps (Abid et al., 2019). We envision that a comparable workflow for RL would have a convenient interface for a user to describe their problem, after which a policy—ideally without requiring any further training (see Section 3)—would be able to start taking actions and solve the problem. In addition to easing the deployment of RL by non-engineers for their tasks of interest, domain experts of novel problems would also become able to create interesting new benchmark domains for RL researchers. This leads to our position as follows:

#### Position

The RL research community should place greater focus on benchmarks with environments defined in user-friendly DSLs or natural languages.

Two clear lines of research that follow from Assumptions 1 and 2 are the design of user-friendly DSLs for relevant application domains, and generating reliable translations from natural language to executable simulators. However, beyond these challenges related to getting simulators to run in the first place, we also argue that they should be used extensively as benchmarks in general RL research, and that existing benchmarks—with environments implemented directly in general-purpose programming languages—are not sufficient to evaluate how different algorithms and approaches might perform when later applied to environments defined in more user-friendly languages.

For example, consider the common assumption that the full action space  $\mathcal{A}$  can be defined in advance, as described in Subsection 2.1. While standardised APIs such as Gym’s (Brockman et al., 2016) have undoubtedly accelerated RL research, there is a risk that convergence of the community on such an API may

<sup>1</sup><https://ludii.games/library.php>

<sup>2</sup><https://ludii.games/forum/forumdisplay.php?fid=23>

have inadvertently entrenched this assumption in the community. The ubiquity of this assumption may also be due to its convenience in deep learning research. There is some early deep RL (DRL) work (Riedmiller, 2005; Lange and Riedmiller, 2010) where actions were treated as inputs of neural networks—hence requiring separate forward passes for every legal action to compute policies or state-action values. However, it quickly became common practice—especially after the work on Deep  $Q$ -Networks by Mnih et al. (2013)—to have outputs for all actions. Requiring only a single DNN forward pass per state greatly improves computational efficiency, at the cost of requiring prior knowledge of the full action space.

In practice, the full action space (or a reasonably sized superset thereof) cannot always be automatically inferred from environment descriptions written in languages that prioritise aspects such as usability over support for robust automated inference. In relatively verbose, logic-based DSLs this may be possible, and it can be straightforward to build policy networks accordingly (Goldwasser and Thielscher, 2020). In contrast, in the DSL of Ludii, which is substantially more succinct and arguably user-friendly (Piette et al., 2020), this does not appear to be feasible. The root of the issue is that succinct descriptions of, for example, game rules, are generally descriptions of procedures that may be used in any game state to generate the set of legal actions for that particular game state. Determining the full action space of the environment requires combinations of this information with an inference of what the entire state space may look like, and this is challenging if the semantics of the DSL are not readily available in a logic-based format. For example, the rule that legal moves consist of players placing one of their pieces on any empty cell in the game of *Hex* is formulated as `(play (move Add (to (sites Empty))))` in Ludii’s DSL. In combination with knowledge of the size of the board (which is defined in a different rule), knowledge that there are no rules that can ever change the size of the board, and knowledge that there are no other rules for other types of moves, it is easy for humans to infer that the action space of this game must be equal to the number of cells on the board. However, without explicit, direct access to formal semantics of the many hundreds of keywords in Ludii’s DSL (Browne et al., 2020), there is no clear way to make this inference in an automated and general manner that works for any game described in the language. Practical attempts at using deep learning with Ludii have therefore faced challenges such as *action aliasing*, where a single output node of a policy network may end up getting shared by multiple distinct legal actions (Soe-

mers et al., 2022)—an issue that is rarely considered possible in other DRL research. Maras et al. (2024) opted to forgo training a policy head altogether, sticking only to a state value function, for games written in another DSL. A similar problem surfaces in PDDL-Gym (Silver and Chitnis, 2020), which also requires careful treatment of action spaces due to a mismatch between PDDL and the customary assumptions about action spaces in RL.

These examples of multiple existing DSLs that conflict with the otherwise common assumption of prior knowledge of the full action space may be merely one example of an important issue that is largely overlooked by current research. It cannot be ruled out that other types of issues, which are not adequately accounted for by the currently prevailing research methodologies, may surface as the community shifts focus to more benchmarks based on environments defined in DSLs or natural language.

### 3 DESCRIPTIONS AS CONTEXT FOR GENERALISATION

The previous section argues for the importance of developing and benchmarking RL techniques that can operate on environments defined in DSLs or natural language, as opposed to general-purpose programming languages, and potential issues that may surface and are underexplored in the current research landscape. However, in addition to potential issues, we also see opportunities. In particular, succinct—but complete—environment descriptions may serve as a powerful tool to improve (zero-shot) generalisation (Kirk et al., 2023) across the set of all environments that may be described in the language of choice.

#### 3.1 Generalisation in RL

The most straightforward setting in RL is to have an agent training in a single environment for some time, and to subsequently evaluate its performance in the same environment. This approach has a high risk of producing agents that overfit, in the sense that they may become overly reliant on spurious features, largely ignore state observations altogether and simply memorise trajectories of states or actions, or otherwise be incapable of handling even minor variations on the environment after training (Whiteson et al., 2011; Machado et al., 2018; Zhang et al., 2018, 2020).

A popular category of RL research with a higher degree of generalisation involves training agents on a subset of one or more closely-related environments,

and evaluating them in the same set, or a different set of similar environments (Farebrother et al., 2018; Justesen et al., 2018; Nichol et al., 2018; Cobbe et al., 2019, 2020; Stone et al., 2021). Different environments in this case may be different levels of the same video game, or subtle variants of an environment with, for example, modified background or foreground colours or patterns, different values for the velocities of certain entities or other numeric parameters, or different reward functions. While prior research collectively covers variation along all dimensions of environments (variation in transition dynamics, in colours used in state observations, in goals or reward functions, etc.), the work described in each publication individually tends to be restricted to a smaller subset of these dimensions. Soemers et al. (2023) used DSL-based environment descriptions for (zero-shot) transfer learning between different board games, but only to a relatively small degree, where the transfer mechanism was not automatically learnt. Banerjee and Stone (2007); Kuhlmann and Stone (2007) automatically identified mappings or transferable features between games, but they used a low-level logic-based DSL, which is arguably lacking in user-friendliness.

### 3.2 Generalisation Through Context

Theoretical work suggests that, in the worst case, strong assumptions on the similarity between different environments are required for efficient generalisation to be possible (Malik et al., 2021). One reason for the difficulty of generalisation to unseen environments, without strong restrictions on the degree of variation, is that epistemic uncertainty about relevant parameters of the current environment essentially turns the collection of all environments that the agent may face into a partially observable environment—even if the current state of each individual environment is fully observable (Ghosh et al., 2021).

The notion of such a collection of environments, each of which may be identified by certain parameters (a *context*), of which some may never be used for training and only appear at test time, may be formalised as a *contextual* (Kirk et al., 2023) Markov decision process. Contexts may be as simple as just the value of a random seed that is used for procedural level generation, or take a more complex form such as a vector of parameters that describe important properties of the environment. Contexts may or may not be observable to the agent(s), although the ability to observe contexts—which should also carry sufficient information to enable disambiguation between environments—is required to resolve partial observ-

ability (Ghosh et al., 2021) induced by epistemic uncertainty.

Research on multi-task RL often involves providing contexts as inputs to agents, but these contexts tend to be far from full environment descriptions. For example, Deisenroth et al. (2014) provide goal coordinates for robotic control tasks as context. It is common to provide short, language-based instructions or hints to guide the agent (Luketina et al., 2019; Lifschitz et al., 2023; Kharyal et al., 2024), but such instructions do not (fully) describe the environment. Lee et al. (2023); Raparthy et al. (2023); Reed et al. (2023) prompt agents with demonstrations of interactions by experts for disambiguation between environments and in-context learning, which is a form of context that is arguably more difficult to acquire than environment descriptions (requiring an environment-specific expert to have already been trained), whilst simultaneously carrying less information (it does not reveal information about any parts of the environment that are not explored in the demonstration). Sun et al. (2020) use a DSL to prescribe policies that an agent should execute, as opposed to describing the environment itself. The textual descriptions provided to agents by Zhong et al. (2020) are perhaps closest to what we propose, although their descriptions are not sufficiently detailed to the extent that they could be compiled into a correct simulator, and are not meant to serve as a substitute for implementing the environment in a programming language.

### 3.3 Environment Descriptions as Context

If it is often desirable to describe environments in succinct DSLs or in natural language, as posited in Section 2, then these descriptions may also be used to improve generalisation by serving as contexts. Leveraging such descriptions as context should not be viewed as a reduction in generality, or being restricted to a particular DSL, as the general workflow of providing environments in such a language is arguably more accessible and more general than using a programming language for many potential end users. An important property of such environment descriptions is that they come from a shared language, and it ought to be possible for humans as well as programs to generate novel environment descriptions in the same language. We cannot only generate contexts from environments, but also generate environments (in the form of fully executable simulators) from contexts. From the researchers' point of view, this is valuable as it makes environments easily controllable and enables a wide variety of evaluation protocols (Kirk et al., 2023).

From the learning agent’s point of view, this property may also be valuable in that a program could actively learn about the description language that is used by procedurally generating new descriptions (Browne, 2009; Todd et al., 2024), translating them into executable simulators, and learning in them—effectively forming their own curriculum of environments (Dennis et al., 2020; Rigter et al., 2024).

Furthermore, it could be argued that contexts that completely describe an environment—to the extent that they could be translated into executable simulators—are likely to be a prerequisite for unrestricted, zero-shot generalisation in RL (Irpan and Song, 2019). Consider the generalisation abilities of humans. In some cases, humans can effectively generalise to unseen situations without relying on explicit task descriptions, but in others they cannot. For example, if a human plays a new video game for the first time, in which there is something that looks like fire, they can infer that they should likely avoid the fire—based on their related experience in the physical world and other video games. However, if a human is faced with a brand new board game, they cannot be expected to play it well if they are not explained the rules of the game. Once the rules are explained, they may be able to play well immediately—based on their experience with related board games and ability to reason—without any direct experience with the game in question.

## 4 RELATED WORK

Mannor and Tamar (2023) caution against excessive focus of the research community on algorithms in existing benchmarks, with little attention for deploying to novel problems, but they do not discuss ease of use, or user-friendly environment description languages as a potential solution. Rodriguez-Sanchez et al. (2023) introduce RLang as a DSL that can be used to provide background knowledge on any aspect of an environment. However, they propose for such descriptions to be provided *in addition to* environment implementations in general-purpose programming language, rather than as a replacement. Nevertheless, this could be an example of a DSL that could be used for our proposed research agenda. Jothimurugan et al. (2019) describe a DSL used to specify reward functions via, e.g., goals and constraints, but no other aspects of the environment. Focusing specifically on the problem of representing goals (rather than full environments), and not necessarily from the perspective of users who are not engineering or RL experts, Davidson and Gureckis (2024) also consider goal represen-

tations based on programs (essentially DSLs) (Davidson et al., 2024) and natural languages, among other solutions.

## 5 CONCLUSION

It is common practice in reinforcement learning (RL) research to implement environments in general-purpose programming languages or frameworks such as CUDA or JAX for hardware acceleration. Such implementations require engineering skills and effort, and often leverage RL expertise to handcraft efficient representations of the state and action spaces. In this position paper, we have argued that this established workflow is not accessible to smaller organisations or private individuals who may not have access to this expertise, and therefore hinders widespread adoption of RL for real-world applications outside of larger projects by teams with substantial resources.

We envision a path to addressing this concern based on using more user-friendly languages, ranging from domain-specific languages to natural languages, for describing environments. Such languages may democratise the ability to apply policies trained with RL to novel problems. This research agenda is expected to involve numerous aspects. DSL-based solutions will require studies of how to develop user-friendly DSLs for describing RL problems, evaluations of their user-friendliness (e.g., via user studies), and the development of efficient compilers or even JAX wrappers for such DSLs. Solutions based on natural languages will require advances in the reliability and consistency of LLMs. Improving the sample efficiency of RL may become even more crucial than it already is. Our focus in this paper has been on democratising the ability to describe RL problems, but this will likely need to be paired up with advances in e.g. AutoRL (Parker-Holder et al., 2022) to also democratise the ability to effectively train policies. Finally, using succinct, information-rich descriptions of environments as context may open up new opportunities for generalisation and transfer in RL.

## REFERENCES

- Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., and Zou, J. (2019). Gradio: Hassle-free sharing and testing of ML models in the wild. In *2019 ICML Workshop on Human in the Loop Learning*.
- Afshar, A. and Li, W. (2024). DeLF: Designing learning environments with foundation models. In *AAAI 2024 Workshop on Synergy of RL and LLMs*.

- Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. (2021). Deep reinforcement learning at the edge of the statistical precipice. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Adv. Neural Inf. Proc. Syst.*, volume 34, pages 29304–29320. Curran Associates, Inc.
- Aram, M. and Neumann, G. (2015). Multilayered analysis of co-development of business information systems. *Journal of Internet Services and Applications*, 6(13).
- Banerjee, B. and Stone, P. (2007). General game learning using knowledge transfer. In *20th Int. Joint Conf. Artif. Intell.*, pages 672–677.
- Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal Artif. Intell. Research*, 47(1):253–279.
- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI gym. <https://arxiv.org/abs/1606.01540>.
- Browne, C., Soemers, D. J. N. J., Piette, É., Stephenson, M., and Crist, W. (2020). Ludii language reference. [ludii.games/downloads/LudiiLanguageReference.pdf](https://ludii.games/downloads/LudiiLanguageReference.pdf).
- Browne, C. B. (2009). *Automatic Generation and Evaluation of Recombination Games*. Phd thesis, Faculty of Information Technology, Queensland University of Technology, Queensland, Australia.
- Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. (2020). Leveraging procedural generation to benchmark reinforcement learning. In Daumé III, H. and Singh, A., editors, *Proc. 37th Int. Conf. Mach. Learn.*, volume 119 of *PMLR*, pages 2048–2056.
- Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2019). Quantifying generalization in reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proc. 36th Int. Conf. Mach. Learn.*, volume 97 of *PMLR*, pages 1282–1289. PMLR.
- Dalton, S. and Frosio, I. (2020). Accelerating reinforcement learning through GPU Atari emulation. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Adv. Neural Inf. Proc. Syst.*, volume 33, pages 19773–19782. Curran Associates, Inc.
- Davidson, G. and Gureckis, T. M. (2024). Toward complex and structured goals in reinforcement learning. In *Finding the Frame workshop @ Reinforcement Learn. Conf.*
- Davidson, G., Todd, G., Togelius, J., Gureckis, T. M., and Lake, B. M. (2024). Goals as reward-producing programs. <https://arxiv.org/abs/2405.13242>.
- Deisenroth, M. P., Englert, P., Peters, J., and Fox, D. (2014). Multi-task policy search for robotics. In *Proc. 2014 IEEE Int. Conf. Robotics and Automation*, pages 3876–3881.
- Dennis, M., Jaques, N., Vinitzky, E., Bayen, A., Russell, S., Critch, A., and Levine, S. (2020). Emergent complexity and zero-shot transfer via unsupervised environment design. In *Adv. Neural Inf. Proc. Syst.*, volume 33, pages 13049–13061.
- Desai, A., Gulwani, S., Hingorani, V., Jain, N., Karkare, A., Marron, M., R, S., and Roy, S. (2016). Program synthesis using natural language. In *Proc. 38th Int. Conf. Software Eng.*, page 345–356. ACM.
- Ellis, B., Cook, J., Moalla, S., Samvelyan, M., Sun, M., Mahajan, A., Foerster, J. N., and Whiteson, S. (2023). SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. In *Adv. Neural Inf. Proc. Syst.* Accepted.
- Farebrother, J., Machado, M. C., and Bowling, M. (2018). Generalization and regularization in DQN. <https://arxiv.org/abs/1810.00123>.
- Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. (2021). Brax - a differentiable physics engine for large scale rigid body simulation.
- Genesereth, M. and Thielscher, M. (2014). *General Game Playing*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.
- Ghosh, D., Rahme, J., Kumar, A., Zhang, A., Adams, R. P., and Levine, S. (2021). Why generalization in RL is difficult: Epistemic POMDPs and implicit partial observability. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Adv. Neural Inf. Proc. Syst.*, volume 34, pages 25502–25515. Curran Associates, Inc.
- Goldwaser, A. and Thielscher, M. (2020). Deep reinforcement learning for general game playing. In *34th AAAI Conf. Artif. Intell.*, pages 1701–1708. AAAI Press.
- Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep reinforcement learning that matters. In *Proc. 32nd AAAI Conf. Artif. Intell.*, pages 3207–3214. AAAI.
- Irpan, A. and Song, X. (2019). The principle of unchanged optimality in reinforcement learning generalization. In *ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning*.
- Jacob, M., Devlin, S., and Hofmann, K. (2020). “it’s unwieldy and it takes a lot of time” — challenges and opportunities for creating agents in commercial games. In *Proc. 16th AAAI Conf. Artif. Intell. Interactive Digital Entertainment*, pages 88–94.
- Jordan, S. M. (2022). Scientific experiments in reinforcement learning. <https://nips.cc/virtual/2022/63891>. Opinion talk contributed to the Deep Reinforcement Learning Workshop at NeurIPS 2022.
- Jordan, S. M., White, A., da Silva, B. C., White, M., and Thomas, P. S. (2024). Position: Benchmarking is limited in reinforcement learning research. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors, *Proc. 41st Int. Conf. Mach. Learn.*, volume 235 of *PMLR*, pages 22551–22569. PMLR.
- Jothimurugan, K., Alur, R., and Bastani, O. (2019). A composable specification language for reinforcement learning tasks. In *Adv. Neural Inf. Proc. Syst.*, volume 32, pages 13041–13051.



- Justesen, N., Torrado, R. R., Bontrager, P., Khalifa, A., Torgelius, J., and Risi, S. (2018). Illuminating generalization in deep reinforcement learning through procedural level generation. In *NeurIPS 2018 Workshop on Deep RL*.
- Kharyal, C., Krishna Gottipati, S., Kumar Sinha, T., Das, S., and Taylor, M. E. (2024). GLIDE-RL: Grounded language instruction through DEmonstration in RL. <https://arxiv.org/abs/2401.02991>.
- Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. (2023). A survey of zero-shot generalisation in deep reinforcement learning. *Journal Artif. Intell. Research*, 76:201–264.
- Koyamada, S., Okano, S., Nishimori, S., Murata, Y., Habara, K., Kita, H., and Ishii, S. (2023). Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In *Adv. Neural Inf. Proc. Syst.*
- Kuhlmann, G. and Stone, P. (2007). Graph-based domain mapping for transfer learning in general games. In Kok, J., Koronacki, J., Mantaras, R., Matwin, S., Mladenić, D., and Skowron, A., editors, *Mach. Learn.: ECML 2007*, volume 4071 of *LNCS*, pages 188–200. Springer, Berlin, Heidelberg.
- Lange, R. T. (2022). gymnax: A JAX-based reinforcement learning environment library.
- Lange, S. and Riedmiller, M. (2010). Deep auto-encoder neural networks in reinforcement learning. In *Neural Networks. Int. Joint Conf. 2010*, pages 1623–1630. IEEE.
- Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. (2023). Supervised pretraining can learn in-context reinforcement learning. <https://arxiv.org/abs/2306.14892>.
- Lifschitz, S., Paster, K., Chan, H., Ba, J., and McIlraith, S. (2023). Steve-1: A generative model for text-to-behavior in Minecraft. <https://arxiv.org/abs/2306.00937>.
- Love, N., Hinrichs, T., Haley, D., Schkufza, E., and Genesereth, M. (2008). General game playing: Game description language specification. Technical Report LG-2006-01, Stanford Logic Group.
- Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. (2019). A survey of reinforcement learning informed by natural language. In *Proc. 28th Int. Joint Conf. Artif. Intell., IJCAI-19*, pages 6309–6317.
- Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal Artif. Intell. Research*, 61:523–562.
- Malik, D., Li, Y., and Ravikumar, P. (2021). When is generalizable reinforcement learning tractable? In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Adv. Neural Inf. Proc. Syst.*, volume 34, pages 8032–8045. Curran Associates, Inc.
- Mannor, S. and Tamar, A. (2023). Towards deployable RL – what’s broken with RL research and a potential fix. <https://arxiv.org/abs/2301.01320>.
- Maras, M., Keça, M., Kowalski, J., and Szykuła, M. (2024). Fast and knowledge-free deep learning for general game playing (student abstract). In *Proc. AAAI Conf. Artif. Intell.*, volume 38, pages 23576–23578.
- Marcus, G., Leivada, E., and Murphy, E. (2023). A sentence is worth a thousand pictures: Can large language models understand human language? <https://arxiv.org/abs/2308.00109>.
- McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL—the planning domain definition language. Technical Report CVC TR98003/DCS TR1165, New Haven, CT: Yale Center for Computational Vision and Control.
- Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4):316–344.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. <https://arxiv.org/abs/1312.5602>.
- Nichol, A., Pfau, V., Hesse, C., Klimov, O., and Schulman, J. (2018). Gotta learn fast: A new benchmark for generalization in RL. <https://arxiv.org/abs/1804.03720>.
- OpenAI (2022). Introducing ChatGPT. <https://openai.com/blog/chatgpt>. Accessed: 2024-01-02.
- Oswald, J., Srinivas, K., Kokel, H., Lee, J., Katz, M., and Sohrabi, S. (2024). Large language models as planning domain generators. In *Proc. Int. Conf. Automated Planning and Sched.*, volume 34, pages 423–431.
- Parker-Holder, J., Rajan, R., Song, X., Biedenkapp, A., Miao, Y., Eimer, T., Zhang, B., Nguyen, V., Calandra, R., Faust, A., Hutter, F., and Lindauer, M. (2022). Automated reinforcement learning (autoRL): A survey and open problems. *Journal Artif. Intell. Research*, 74:517–568.
- Patterson, A., Neumann, S., White, M., and White, A. (2023). Empirical design in reinforcement learning. <https://arxiv.org/abs/2304.01315>.
- Piette, É., Soemers, D. J. N. J., Stephenson, M., Sironi, C. F., Winands, M. H. M., and Browne, C. (2020). Ludii – the ludemic general game system. In Giacomo, G. D., Catala, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., and Lang, J., editors, *Proc. 24th Eur. Conf. Artif. Intell.*, volume 325 of *Frontiers in Artificial Intell. and Appl.*, pages 411–418. IOS Press.
- Raparthi, S. C., Hambro, E., Kirk, R., Henaff, M., and Raileanu, R. (2023). Generalization to new sequential decision making tasks with in-context learning. <https://arxiv.org/abs/2312.03801>.
- Reed, S., Żoła, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Giménez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. (2023). A generalist agent. *Trans. Mach. Learn. Research*.
- Riedmiller, M. (2005). Neural fitted Q iteration - first experiences with a data efficient neural reinforcement

- learning method. In *Mach. Learn.: ECML 2005*, volume 3720 of *LNCS*, pages 317–328. Springer.
- Rigter, M., Jiang, M., and Posner, I. (2024). Reward-free curricula for training robust world models. In *Int. Conf. Learn. Representations*.
- Rodriguez-Sanchez, R., Spiegel, B. A., Wang, J., Patel, R., Tellex, S., and Konidaris, G. (2023). RLang: A declarative language for describing partial world knowledge to reinforcement learning agents. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proc. 40th Int. Conf. Mach. Learn.*, volume 202 of *PMLR*, pages 29161–29178. PMLR.
- Samvelyan, M., Kirk, R., Kurin, V., Parker-Holder, J., Jiang, M., Hambro, E., Petroni, F., Küttler, H., Grefenstette, E., and Rocktäschel, T. (2021). Mini-hack the planet: A sandbox for open-ended reinforcement learning research. In *Adv. Neural Inf. Proc. Syst.*
- Seger, E., Ovadya, A., Siddarth, D., Garfinkel, B., and Dafoe, A. (2023). Democratising AI: Multiple meanings, goals, and methods. In *Proc. 2023 AAAI/ACM Conf. AI, Ethics, and Society*, pages 715–722.
- Silver, T. and Chitnis, R. (2020). PDDL Gym: Gym environments from PDDL problems. In *ICAPS Workshop on Bridging the Gap Between AI Planning and RL (PRL)*.
- Simkute, A., Luger, E., Evans, M., and Jones, R. (2024). “it is there, and you need it, so why do you not use it?” achieving better adoption of AI systems by domain experts, in the case study of natural science research. <https://arxiv.org/abs/2403.16895>.
- Soemers, D. J. N. J., Mella, V., Browne, C., and Teytaud, O. (2022). Deep learning for general game playing with Ludii and Polygames. *ICGA Journal*, 43(3):146–161.
- Soemers, D. J. N. J., Mella, V., Piette, É., Stephenson, M., Browne, C., and Teytaud, O. (2023). Towards a general transfer approach for policy-value networks. *Trans. Mach. Learn. Research*.
- Stevens, G. C. (1983). User-friendly computer systems?: A critical examination of the concept. *Behaviour & Inf. Tech.*, 2(1):3–16.
- Stone, A., Ramirez, O., Konolige, K., and Jonschkowski, R. (2021). The distracting control suite – a challenging benchmark for reinforcement learning from pixels. <https://arxiv.org/abs/2101.02722>.
- Sun, S.-H., Wu, T.-L., and Lim, J. J. (2020). Program guided agent. In *Int. Conf. Learn. Representations*.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition.
- Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. (2018). DeepMind control suite.
- Todd, G., Padula, A., Stephenson, M., Piette, É., Soemers, D. J. N. J., and Togelius, J. (2024). GAVEL: Generating games via evolution and language models. In *Adv. Neural Inf. Proc. Syst.*, volume 37. Accepted.
- Voelcker, C., Hussing, M., and Eaton, E. (2024). Can we hop in general? A discussion of benchmark selection and design using the Hopper environment. In *Finding the Frame workshop @ RLC*.
- Whiteson, S., Tanner, B., Taylor, M. E., and Stone, P. (2011). Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE Symp. Adaptive Dynamic Programming and Reinforcement Learn.*, pages 120–127.
- Yang, M., Du, Y., Ghasemipour, K., Tompson, J., Kaelbling, L., Schuurmans, D., and Abbeel, P. (2024). Learning interactive real-world simulators. In *Int. Conf. Learn. Representations*.
- Zhang, A., Ballas, N., and Pineau, J. (2020). A dissection of overfitting and generalization in continuous reinforcement learning. <https://arxiv.org/abs/1806.07937>.
- Zhang, C., Vinyals, O., Munos, R., and Bengio, S. (2018). A study on overfitting in deep reinforcement learning. <https://arxiv.org/abs/1804.06893>.
- Zhao, W., Queralta, J. P., and Westerlund, T. (2020). Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *2020 IEEE Symp. Ser. Comput. Intell. (SSCI)*, pages 737–744.
- Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2023). A survey of large language models. <https://arxiv.org/abs/2303.18223>. Accessed: 25-01-2024.
- Zhong, V., Rocktäschel, T., and Grefenstette, E. (2020). RTFM: Generalising to novel environment dynamics via reading. In *Int. Conf. Learn. Representations*.
- Zuo, M., Velez, F. P., Li, X., Littman, M. L., and Bach, S. H. (2024). Planetarium: A rigorous benchmark for translating text to structured planning languages. <https://arxiv.org/abs/2407.03321>.