A Research Agenda for Usability and Generalisation in Reinforcement Learning

Dennis J.N.J. Soemers[1], Spyridon Samothrakis[2], Kurt Driessens[1], and Mark H.M. Winands[1]

[1] Department of Advanced Computing Sciences, Maastricht University, Maastricht, the Netherlands
{dennis.soemers, kurt.driessens, m.winands}@maastrichtuniversity.nl
[2] Institute for Analytics and Data Science, University of Essex, Colchester, United Kingdom
ssamot@essex.ac.uk

Abstract. It is common practice in reinforcement learning (RL) research to train and deploy agents in bespoke simulators, typically implemented by engineers directly in general-purpose programming languages or hardware acceleration frameworks such as CUDA or JAX. This means that programming and engineering expertise is not only required to develop RL algorithms, but is also required to use already developed algorithms for novel problems. The latter poses a problem in terms of the usability of RL, in particular for private individuals and small organisations without substantial engineering expertise. We also perceive this as a challenge for effective generalisation in RL, in the sense that is no standard, shared formalism in which different problems are represented. As we typically have no consistent representation through which to provide information about any novel problem to an agent, our agents also cannot instantly or rapidly generalise to novel problems. In this position paper, we advocate for a research agenda centred around the use of user-friendly description languages for describing problems, such that (i) users with little to no engineering expertise can formally describe the problems they would like to be tackled by RL algorithms, and (ii) algorithms can leverage problem descriptions to effectively generalise among all problems describable in the language of choice.

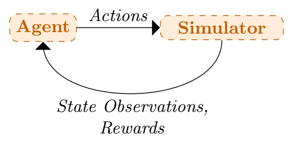
Keywords: Environment Descriptions · Generalisation · Reinforcement Learning.

1 Introduction

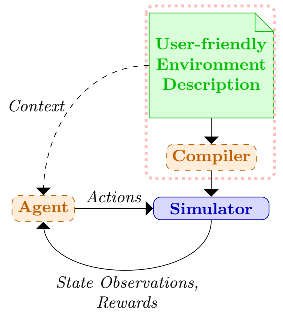
Researchers in the field of reinforcement learning (RL) have largely converged on a small number of common APIs for the development of benchmark domains that are used to evaluate the performance of RL algorithms. New environments are customarily written in general-purpose programming languages such as `C++` or `Python`, and implement a Gym-based [22] API for learning algorithms to interface with environments. There are variants on this approach, such as PettingZoo [154] for multi-agent RL, but the general workflow remains similar.

At a high level, we may distinguish roughly two categories of RL research. On the one hand, there is research where the focus is on the development of new and improved (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 domains with vision-based inputs, RL for partially observable environments, RL for continuous action spaces, 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 environments are often games or other simulations, with arguably relatively limited direct real-world impact outside of their use as benchmarks for RL research. Examples of such suites of environments include the Arcade Learning Environment [13,87], ProcGen [25], SMACv2 [37], and the DeepMind Control Suite [152]. On the other hand, there is research in which a single, 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 such a task for the purpose of this research. This engineering effort often requires highly specialised knowledge of, for example, programming for GPUs or other hardware accelerators, or of the inner workings of deep learning (DL) and RL algorithms, such that the task and its state and action representations can be implemented in such a way that the existing DL and RL techniques can be applied to their fullest potential. Examples include research focused on applications such as resource balancing for logistics [78], flight control for stratospheric balloons [12], compiler optimisations [27,159], automated chip floorplanning [96], power grid management [167], alignment of large language models with human preferences [110,64], and magnetic tokamak controllers [31].

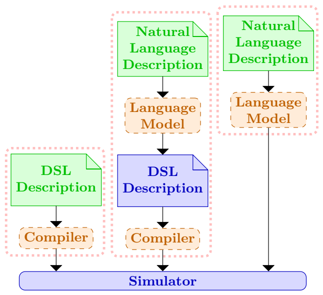
While discussions on experimental methodologies and statistical analyses of empirical evaluations in RL have been on the rise within the research community [53,3,60,113,59], as well as discussions of how to select which environment(s) to use in experiments [113,160], we see little to no discussion on *how* or *by whom* tasks (or environments) are described or implemented in the first place. In a recent position paper [146], we argued 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 at large to miss out on interesting research directions with respect to generalisation 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) [95,8], or even in natural language.



(a) A simulator of the environment is implemented in a general-purpose programming language. This is the most common setting in current research.


User-friendly Environment Description
Context
Compiler
Agent
Actions
Simulator
State Observations, Rewards

(b) The environment is described in a user-friendly language, and compiled into a simulator. The description may serve as additional input to the agent.



(c) A DSL description can be converted into a simulator by a compiler. A natural language description can first be translated into an intermediate DSL description by a language model, or can be translated directly with a language model serving as compiler.

Fig. 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 expertise. **(a)** A depiction of the customary setting in RL research. **(b)** The approach for which we posit that increased research attention is warranted (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 generate an intermediate DSL description. Image source: [146].

Once we adopt a methodology in which 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. Fig. 1a depicts the setting where the environment is implemented directly in a general-purpose programming language, and Fig. 1b depicts the proposed settings, with Fig. 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 suffice, as hardware and the real world with its laws of physics already dictate 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 [170], still be highly beneficial [166].

This paper is an extension of a previous position paper [146]. The primary extensions consist of (i) a discussion of barriers to widespread adoption of RL other than the one that is our main focus (Section 5), and (ii) a detailed description of the research agenda we envision following from our position (Section 6). Furthermore, other sections throughout the paper have been updated to provide additional context, examples, and to reflect and relate our work to other recent developments in the field, such as the Ludax framework [158] and other related position papers [127,17,52].

2 Background

This section provides background information on (partially observable) Markov decision processes [55] and Markov games [161] (Subsection 2.1)—two frequently used formalisations of sequential decision-making problems, which we argue users should be able to describe in more accessible languages than programming languages—and deep reinforcement learning (Subsection 2.2).

2.1 Markov Decision Processes & Markov Games

A Markov decision process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, \rho \rangle$ is a formal description of a single-agent sequential decision-making problem, in which $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ a function that defines transition probabilities, and $\rho : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R}$ a reward function. More precisely, $0 \leq P(s' \mid s, a) \leq 1$ gives the probability of observing a transition from a current state $s \in \mathcal{S}$ to a successor state $s' \in \mathcal{S}$ after executing an action $a \in \mathcal{A}$, and $\rho(r \mid s, a, s')$ denotes the probability of observing a real-valued reward $r \in \mathbb{R}$ after the same event. In some MDPs, it may be the case that only certain subsets of the full action space $\mathcal{A}$ are legal in certain states.

The behaviour of an agent may be described as a policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, such that $0 \leq \pi(a \mid s) \leq 1$ denotes the probability with which the policy selects an action $a \in \mathcal{A}$ whenever the current state is $s \in \mathcal{S}$. At discrete time steps $t = 0, 1, \dots,$ the agent observes the current state $S_t \in \mathcal{S}$, samples an action $A_t \sim \pi(\cdot \mid S_t)$ from its policy π, transitions into a successor state $S_{t+1} \sim P(\cdot \mid S_t, A_t)$, and receives a reward $R_t \sim \rho(S_t, A_t, S_{t+1})$. The most common objective is to select actions such that the returns G_0 are maximised, where $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ denotes sum of discounted rewards collected from time t onwards. Temporally distant rewards are discounted, relative to short-term rewards, by the discount factor $0 < \gamma \leq 1$.

The *state value function* $V^\pi : \mathcal{S} \mapsto \mathbb{R}$ gives the expected returns when action according to a policy π from any input state s onwards: $V^\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$. Similarly, the *state-action value function* $Q^\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ gives the expected returns of executing an input action a in an input state s, and sampling any subsequent actions from π: $Q^\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right]$.

Partially observable Markov decision processes (POMDPs) additionally feature an observation space $\mathcal{O}$, and a mapping $\phi : \mathcal{S} \mapsto \mathcal{O}$ from states to observations. In a POMDP, the agent cannot necessarily observe the current state S_t, but only an observation $O_t = \phi(S_t)$, where it is possible for multiple different states to map to the same observation.

The concept of Markov games may be viewed as a generalisation of (PO)MDPs for multi-agent settings [82]. In a Markov game, each of $k \geq 1$ players (or agents) has its own action space from which to select actions, and its own reward function. Transition probabilities between pairs of states are defined over the joint action space of all players. This paper uses the term *environment* to refer to problems that may be tackled by RL, regardless of whether they may be (PO)MDPs or Markov games.

2.2 Deep Reinforcement Learning

One of the main goals in reinforcement learning (RL) [151] research is the development of algorithms that can automatically learn strong policies from experience gathered within a (PO)MDP or Markov game. *Tabular* RL methods learn distinct state-action values or probabilities for every individual state-action pair in $\mathcal{S} \times \mathcal{A}$. This is only feasible in relatively small problems, and often wasteful in terms of data efficiency due to the lack of generalisation between related state-action pairs. RL methods with *function approximation* address these issues, by training function approximators that use features of states and/or actions, rather than enumerating the complete space. State-action pairs that are closely related to each other may be expected to have similar features, allowing for improved data efficiency through generalisation. One of the most popular and successful forms of RL with function approximation is deep RL, in which deep neural networks (DNNs) [74] are used as function approximators.

3 Description Languages for Environments

Subsection 3.1 describes the established practice where RL research is performed using environments that have been implemented, conforming to a standardised API such as the Gym API [22], in general-purpose programming languages or more specialised frameworks for hardware accelerators such as GPUs. It identifies potential issues that may arise when the entire research community focuses solely on such environments. As an initial step towards more user-friendly descriptions, Subsection 3.2 discusses the use of DSLs for describing environments for use in RL research. Subsection 3.3 explores the possibility of using natural languages to define environments—arguably one of the most user-friendly language classes. Finally, Subsection 3.4 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.

3.1 Defining Environments in Programming Languages

Outside of cases where RL is applied directly in the physical world, such as some work in robotics [85], it is customary to implement the environments used for RL research in programming languages such as `C++` or `Python`. There is also a recent trend of using more advanced toolkits or libraries, such as

CUDA or JAX [20], to enable the environments themselves—and not just DNN forward and backward passes—to make efficient use of hardware accelerators [28,41,89,15,71,42,51,69,130,18,118]. The latter approach can provide dramatic speed increases, but also imposes additional constraints on programming style and requires more specialised and advanced engineering skills.

Most developers of RL environments have converged to the API popularised by OpenAI Gym [22]. 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. The observation space may, for example, be a discrete set of integers, each of which uniquely identifies one element of the state space $\mathcal{S}$, or it may be subset of $\mathbb{R}^d$ for some dimensionality d, such that every state is described by a d-dimensional real-valued feature vector.
- 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 at each time step.
- A *reset* function, which resets 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' (alongside several auxiliary variables). In this function, programmers essentially implement the transition and reward models P and ρ. This is typically done implicitly: the function usually implements a procedural algorithm that generates s' and r in a manner that ends up being consistent with the probabilities defined by P and ρ, without explicitly defining and sampling from the full distributions.

3.2 Domain-Specific Languages for Environments

A potential alternative to the standard practice of describing environments in general-purpose (or more advanced and complex hardware acceleration frameworks) is to use DSLs to describe sets of environments. This approach still requires significant engineering effort to develop a compiler that can translate valid 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. Note that the requirement for a compiler to be implemented suggests that it would only ever be reasonable to use a DSL for sets of multiple environments, but never for a single environment. If there is only a single environment of interest, it would likely be easier to directly implement that environment itself in a programming language.

Numerous examples of DSLs for describing sequential decision-making problems already exist, though their adoption as benchmarks within the RL community is restricted compared to benchmarks such as the Arcade Learning Environment [13,87] or the DeepMind Control Suite [152], which are not based on DSLs. Examples include the Planning Domain Definition Language (PDDL)

[93] for planning problems, and the Stanford Game Description Language [83,44], Ludi [24], Toss [63], the Video Game Description Language [134,136], Regular Boardgames [68], Ludii [115], Stratega [114], Griddly [9], MiniHack [133], and Ludax [158] for various ranges of games. PDDLGym [140] provides Gym environment wrappers around PDDL problems.

As an example, we provide a description of the game of Tic-Tac-Toe in the DSL of Ludii below. Note that some rules—such as players moving in turns, or the game ending in a draw when neither player can move—are not explicitly stated, because these are default settings and can therefore be omitted in this language. Games that deviate from these defaults can be described by explicitly including rules where appropriate.

Ludii Game Description Example

```
(game "Tic-Tac-Toe"
    (players 2)
    (equipment {
        (board (square 3))
        (piece "Disc" P1)
        (piece "Cross" P2)
    })
    (rules
        (play (move Add (to (sites Empty))))
        (end (if (is Line 3) (result Mover Win)))
    )
)
```

3.3 Describing Environments in Natural Language

While DSLs may already be considered a more user-friendly alternative to general-purpose programming languages [95,8] for describing environments, natural language would be even more accessible to a wider userbase. Here, we provide an example of what a natural language description of the game of Tic-Tac-Toe might look like:

Natural Language Description Example

The game of Tic-Tac-Toe is played by two players on a grid of 3×3 square cells. Players take turns drawing their symbol—a circle for the first player, and a cross for the second—in an empty cell of their choice. The game ends in a win for a player if that player completes an orthogonal or diagonal line of three instances of their symbol. The game ends in a draw if the board is filled up with neither player achieving their win condition.

Although the state of the art of large language models (LLMs) is highly impressive [171], there are still concerns surrounding reliability and correctness

[76]. In particular, we envision that ambiguities typically present in natural languages, as well as the tendency for humans to underspecify task descriptions (e.g., rules of games), will present substantial challenges that require further research. Recently, [2] demonstrated promising initial results for an LLM automatically generating executable environment code based on natural language descriptions, but it still requires an expert human in the loop 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 [34,109,176], and a user can inspect the generated description and make corrections if necessary before it is compiled into a simulator. In the long term, if LLMs can be made sufficiently reliable, natural languages would likely be the most accessible modality for describing environments.

3.4 Research Focus on Description Languages for Environments

Before formally stating the central position of this paper, we make two core 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, such as (PO)MDPs or Markov games, in domain-specific languages (DSLs) or natural languages can be more user-friendly than defining them in general-purpose programming languages.*

Increasing user-friendliness and lowering barriers to entry is a well-established potential motivation for the use of DSLs [95,8]. 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 [83,44] 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 interesting 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 [115]. Likely in no small part due to the language's level of accessibility, Ludii has amassed a library of over 1400 distinct official game descriptions,[3] including third-party contributions from game designers with little or no programming experience.[4]

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

[3] https://ludii.games/library.php
[4] https://ludii.games/forum/forumdisplay.php?fid=23

describe their tasks in natural languages, and they would ideally only have to verify or fix small issues in automatically generated descriptions in a user-friendly DSL 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 [58]. While not focused on RL (but, rather, AI in general) or environment descriptions, a recent study by [141] reveals a substantial disconnect between the technical know-how that AI designers 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 [138] 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 [108] and DALL·E 2 [120], Stability AI's models and interfaces for audio [147], image [117], and video [16] generation, Midjourney,[5] and Gradio apps [1]. 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 4)—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—similar to how [35] aimed to improve accessibility and facilitate further research for dynamic algorithm configuration. This leads to our position as follows:

[5] https://www.midjourney.com/

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 would be the design of user-friendly DSLs for relevant application domains, as well as generating reliable translations from natural language to exectuable 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 3.1. While standardised APIs such as OpenAI Gym's [22] have undoubtedly aided and accelerated RL research, there is a risk that convergence of the community on such an API may have entrenched this assumption in the community. The ubiquity of this assumption may also be explained by its convenience in the context of deep learning research. There exists some early deep RL work [124,72] 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 [99]—to treat actions as outputs. 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 the relatively verbose, logic-based S-GDL [83,44], this is possible, and it is straightforward to build policy networks accordingly [49]. In contrast, in Ludii's DSL, which is substantially more succinct and arguably user-friendly [115], 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 [23], 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 [143]—an issue that is rarely considered possible in other deep RL research. [92] 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 PDDLGym [140], 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 is 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.

4 Leveraging Environment Descriptions for Generalisation in RL

The previous section argues for the relevance of developing RL techniques that can operate on environments defined in DSLs or natural language, as opposed to general-purpose programming languages, the importance dedicating substantial research efforts on benchmarks following the same methodology, 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 [66] across the set of all environments that may be described in the language of choice.

4.1 Generalisation in RL

The most straightforward setting in RL [151] is to have an agent interacting, collecting experience, and 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 [162,87,169,168].

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 [164,77,39,19,32,38,62,104,26,25,149,94]. 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. Particularly in the cases of meta RL [137,40,103,119,175,11] and few-shot transfer learning [153,73,102,174], at least a small amount of environment-specific fine-tuning prior to evaluation is also assumed. While prior research collectively covers variation along all dimensions of environments (variation in transition dynamics, variation in colours used in state observations, variation in goals or reward functions, etc.), the work described in each publication individually tends to be restricted to a smaller subset of these dimensions. Research on transfer learning in RL [153,73,174] is often similarly restricted in scope [46,129,155,142,43,98,47,165]. [144] 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. [10,70] automatically identified mappings or transferable features between games, but they used a low-level logic-based DSL, which is arguably lacking in terms of user-friendliness.

Sim-to-real transfer [170]—where policies are trained in simulation, but deployed on robots in the physical world—is another major category of generalisation in RL. However, in this case, the need for generalisation is an unfortunate reality due to inevitable differences between simulators and the real world. While some degree of variation is inevitable in this setting, it is typically intentionally kept as low as possible.

4.2 Leveraging Context for Generalisation

Theoretical work suggests that, in the worst case, strong assumptions on the similarity between different environments are required for efficient generalisation to be possible: different environments must share an optimal policy [90]. 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 [45].

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* [66,14] (PO)MDP or Markov game. In the formalism of [66], 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 observability [45] 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, [32] provide a single or a handful of numeric values as context, which describe goal coordinates for robotic control tasks. It is common to provide short, language-based instructions or hints to guide the agent [88,6,97,107,139,163,84,50,79,65], but such instructions do not (fully) describe the environment. In cases where goals correspond to elements of the state space, universal value function approximators [135] can learn reusable skills and generalise by conditioning on goal states. [75,122,123] 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). CARL [14] extends several well-established RL benchmarks with contexts, but these contexts take the form of dictionaries specifying values for only a handful of variables. [128] consider more elaborate task descriptions, but not necessarily ones that fully specify complete environments. [150] use a DSL to prescribe policies that an agent should execute, as opposed to describing the environment itself. [21] leverage text from strategy guides to guide search-based game playing agents, but these texts do not fully describe the environment, and were not used to improve generalisation to unseen environments. The textual descriptions provided to agents by [172] 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.

4.3 Environment Descriptions as Context

If it is desirable to describe environments in succinct DSLs or in natural language, as posited in Section 3, then these descriptions may also be used to improve generalisation by serving as contexts. Crucially, leveraging such descriptions as context should not be viewed by peer reviewers as being 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 should be possible for humans as well as software 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 [66]. 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 [24,157], translating them into executable simulators, and learning in them—effectively forming their own curriculum of environments to learn from [33,111,57,132,125].

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 [56]. 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. This relies on an implicit assumption that fire in this particular video game works similarly to how it works in other games, or in the real world, which may be incorrect. If a human is faced with a brand new board game, they cannot be expected to play it well if the rules of the game are not explained. 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.

5 Other Barriers to Widespread Deployment of RL

While the focus of this paper is on the challenge posed to widespread adoption of RL by the customary use of programming languages for defining environments or tasks, this is not the only barrier. The choice to focus on environment descriptions and the languages in which they are described is due to our perception that this aspect of RL research and deployment is hardly if ever discussed, whereas other challenges are already more frequently acknowledged and discussed. However, we do not expect these different challenges, or solutions to them, to be completely orthogonal, and discuss how they may be related here.

5.1 Sample Efficiency and Computation Resource Requirements

Sample (in)efficiency and the requirement for substantial computation resources are frequently cited as pervasive issues in deep RL, both in terms of feasibility of deployment in the "real world" as well as research more generally [106,4,101]. This issue motivates the recent trend of implementing simulators to be entirely runnable on hardware accelerators via frameworks such as JAX, as discussed in Subsection 3.1. When applicable, such an approach is highly effective at lowering the barrier to entry in terms of hardware requirements for RL research. However, this style of implementing environments is arguably even more demanding in terms of engineering expertise than using general-purpose programming languages, and is not directly helpful in terms of lowering the barrier to entry for deployment of RL by users without such engineering expertise.

In contrast, environments implemented in commonly-used DSLs [83,136,115] tend to have substantially lower simulation speeds than similar environments implemented directly in programming languages like `Java` or `C++`. This only exacerbates the need for computation resources—to make up for low simulation speeds—if RL is to be successful in these environments. We do not view this as an argument against using DSLs, but rather as an additional reason to focus even more on improving sample efficiency of RL. Only focusing on (methodologies for implementing) environments that permit running simulations at high speeds would be an inadvisable form of tunnel vision, not unlike how the *hardware lottery* [54] describes when certain algorithms receive more attention than others depending on their compatibility with current hardware. Alternatively, it may be possible to combine the high speed of hardware-accelerated programming with user-friendly domain specific languages. The Ludax framework [158] is a recent, and to our knowledge first, example of a system that compiles user-friendly game description languages (inspired by Ludii's language [115]) into native and hardware-accelerated simulation code via JAX. To what extent this can be effectively replicated with other languages, domains other than board games, or even simply a broader and more diverse set of board games than that currently supported by Ludax, remains to be seen in future research.

5.2 Complexity of Agent Design, RL Algorithm Selection, and Hyperparameter Tuning

Another challenge for the deployment of RL is lack of reliability (e.g., instability or high variance in performance levels over different random seeds), and the degree to which RL performance depends on selecting the right algorithms, optimisers, neural network architectures, hyperparameters, and other design decisions for each specific environment [7,36,105]. All of these are choices that tend to require a substantial amount of RL and engineering expertise, or the ability to run extensive experiments (e.g., grid searches for hyperparameters). However, advances in meta RL [137,40,103,119,175,11] and AutoML (or AutoRL) [112,100,48] may help lower this barrier to entry [81]. It is plausible that the ability to provide succinct and information-rich environment descriptions as context, as discussed in Section 4, may also benefit these techniques.

6 Proposed Research Agenda

Building on the arguments and analyses from the previous sections, we propose a research agenda centred around the use of user-friendly languages to describe problems of interest (i.e., the environments in RL settings), with anticipated opportunities, new avenues for research, and challenges regarding aspects such as democratisation of the use of AI (specifically, RL), description language design, generalisation and transfer, sample efficiency, and AutoRL and meta RL.

While it is not necessarily an exhaustive list, and relative importance of each item may vary among different research and deployment contexts, we identify the following list of desiderata for environment description languages:

- *User-friendliness*: enabling end users with little programming or RL expertise to describe their optimisation problems of interest, such that RL can be applied to tackle them, is central to our proposed research agenda. This implies that any languages used to these environments ought to be user-friendly.
- *Complete and unambiguous environment specifications*: there is a substantial amount of work on providing *some* information about an environment as context to aid decision-making [88,6,97,107,139,163,84,128,150,172,50,14,79,65], but unless descriptions are complete to the extent that they can be compiled into an executable simulator, they do not help alleviate the need for engineers with programming and RL expertise. A key exception is when training will be done in the real world (e.g., directly on physical robots), in which case only specifying goals, objectives, or reward functions would suffice.
- *Generality and expressiveness*: we remark that, as the orange boxes in Fig. 1b indicate, we still expect programming and RL expertise to be required in some parts of the proposed workflow. In particular, such expertise will still be required to build compilers for any designed language. Therefore, this workflow will only pay off, in terms of democratisation of the use of RL, if the same compiler can be re-used for multiple different (but presumably related) tasks, all describable without engineering expertise in the same user-friendly language. If there is only a single task of interest, a simulator for it might as well be implemented directly in code. This suggests that a sufficient degree of generality and expressiveness of the language is crucial: we ought to be able to describe a variety of different tasks in the same language. In practice, we expect that there may be tension between the desiderata of user-friendliness and generality. While we have no quantitative evidence at this point, our own practical experience, and conversations with others among the general game playing research community in particular, suggest that description languages that specifically target a restricted subset of domains such as video games [134,136] or board games [115], using high-level keywords specific to that domain, tend to be easier to use than lower-level (e.g., logic-based) languages [83,44].
- *Computational efficiency*: as discussed in Subsection 5.1, high simulation speeds are desirable to speed up training, in particular due to the lack of sample efficiency in current RL techniques. Therefore, descriptions can ideally be compiled into computationally efficient simulators.
- *Facilitation of procedural generation*: especially in the case of highly expressive languages, it may be infeasible to manually write enough descriptions to provide sufficient coverage of the full space of all describable problems, which can in turn impair the ability to train policies that effectively generalise in a zero-shot manner across the space. For example, the t-SNE visualisation [86] in Fig. 2 depicts how even a relatively large set of 1059 different board games still leaves large uncovered areas in the underlying space, which may be important to fill up with further example games for a model to effectively learn the precise semantics of the game description language, and implications on effective playing strategy. This may be addressed by procedurally

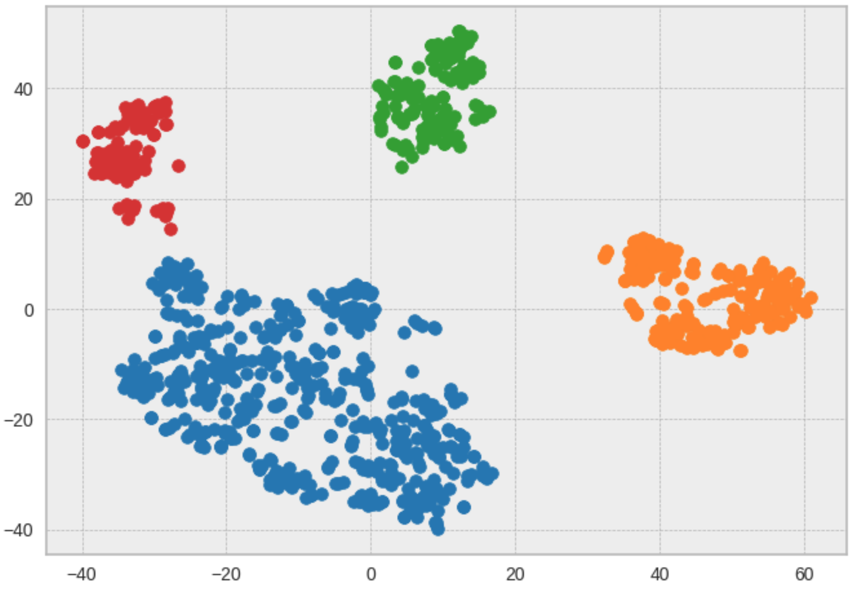


Fig. 2: A set of 1059 different board games, described in Ludii's game description language, reduced from a larger feature space [116] to two dimensions via a t-SNE embedding [86]. Image source: [148].

generating new tasks to fill up uncovered parts of the space [157]. The design of the language and underlying problem representation can have a substantial impact on how easy or difficult it is for, e.g., evolutionary algorithms to navigate and construct suitable, novel elements of this space.

All of these desiderata should feature in evaluations of designed languages in our proposed research agenda. For some aspects, such as measuring computational efficiency of frameworks [67,158] and (theoretically) analysing the expressiveness of environment and goal description languages [156,68,80,145,131], this is already commonplace. Other aspects, such as ease of use for humans or suitability of representations for procedural content generation, are sometimes listed as design considerations [115], but lacking in terms of quantitative and objective evaluations in existing research (though there is some recent work in such directions [121]).

When we have a user-friendly description language in which end users can conveniently describe a variety of problems of interest, we still require an RL solution to tackling these problems. For the same end users to be able to also execute this step, we expect to require substantial advances in (i) generalisation, and/or (ii) AutoRL and meta RL. Advances in generalisation (discussed in more detail in Section 4) would enable us to pre-train an agent in a wide variety of environments described in the chosen language, and have it automatically generalise to novel problems described in the same language. In the case of successful zero-shot generalisation, this could even circumvent the need for training time and other resources (e.g., hardware) for the end user. Similarly, advances in AutoRL and meta RL (discussed in more detail in Subsection 5.2) would circumvent the need for the end user to have the RL knowledge necessary to choose which algorithm(s) to use, but would not alleviate the need for training resources such as time and hardware.

7 Related Work

Mannor and Tamar [91] also caution against the 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. Zhu et al. [173] present Pearl as a framework meant to facilitate users applying RL to their real-world applications. However, this framework focuses on the design of agents, as opposed to environments, and does not alleviate the need for highly engineered environment implementations.

Rodriguez-Sanchez et al. [126] introduce RLang as a DSL that can be used to provide background or expert knowledge on any aspect of an MDP. However, they propose for such descriptions to be provided *in addition to* environment implementations in general-purpose programming languages, 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. [61] describe a DSL that can be used to specify reward functions via, e.g., goals and constraints, but no other aspects of the environment. In the specific case of robotics in the physical world, natural language-based instructions for robots to follow [5] may suffice to provide widespread, easy access, as there is no need for a simulator, but this does not extend to many other (virtual) applications. Focusing specifically on the problem of representing goals (rather than full MDPs or RL problems), and not necessarily from the perspective of users who are not engineering or RL experts, Davidson et al. also consider goal representations [29] based on programs (essentially DSLs) [30] and natural languages, among other solutions.

While our position and proposed research agenda revolve around ideas such as user-friendliness and democratisation of the use of RL, we still view these issues largely from a technical angle by exploring how we may technically develop techniques to facilitate broader use of RL. Other recent position papers explore related ideas, such as the inherent value of application-driven AI research, recommended changes to peer review, hiring, and teaching policies in the machine learning community, and legal and ethical considerations when shifting research focus to real-world applications [127,52]. In another recent position paper, Blili-Hamelin et al. [17] argue to shift away from an excessive focus on Artificial General Intelligence as a research goal, and rather be more specific and explicit about research goals and the exact meaning of notions such as "generality." Our proposed research agenda is in line with this, as we argue to focus on restricted subsets of problems, and build agents that can generalise specifically within the set of problems describable in any description language of choice.

8 Conclusion

The central position in this paper is to argue for a need for increased focus in the reinforcement learning (RL) research community on benchmarks in which environments are not implemented directly in general-purpose programming languages, but rather described in user-friendly domain-specific languages (DSLs) or even natural languages. The core reason for this is to empower end users with little to no programming or RL expertise to apply RL as a solution to their optimisation problems of interest: they only need to be able to describe their problems in the user-friendly language of choice. Furthermore, we see potential for advances in (zero-shot) generalisation and transfer within the set of problems describable in any such language, by using the environment descriptions as context on which to condition policies, value functions, and so on. We presented a complete outline of the research agenda we propose, accounting for desiderata and challenges such as user-friendliness, sample efficiency, and generalisation.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., Zou, J.: Gradio: Hassle-free sharing and testing of ML models in the wild. In: 2019 ICML Workshop on Human in the Loop Learning (2019)
2. Afshar, A., Li, W.: DeLF: Designing learning environments with foundation models. In: AAAI 2024 Workshop on Synergy of Reinforcement Learning and Large Language Models (2024)
3. Agarwal, R., Schwarzer, M., Castro, P.S., Courville, A., Bellemare, M.G.: Deep reinforcement learning at the edge of the statistical precipice. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 29304–29320. Curran Associates, Inc. (2021)
4. Agarwal, R., Schwarzer, M., Castro, P.S., Courville, A.C., Bellemare, M.: Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 28955–28971. Curran Associates, Inc. (2022)
5. Ahn, M., Dwibedi, D., Finn, C., Gonzalez Arenas, M., Gopalakrishnan, K., Hausman, K., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kirmani, S., Leal, I., Lee, E., Levine, S., Lu, Y., Maddineni, S., Rao, K., Sadigh, D., Sanketi, P., Sermanet, P., Vuong, Q., Welker, S., Xia, F., Xiao, T., Xu, P., Xu, S., Xu, Z.: AutoRT: Embodied foundation models for large scale orchestration of robotic agents. In: IEEE ICRA 2024 Workshop on Vision-Language Models for Navigation and Manipulation (2024)

6. Andreas, J., Klein, D., Levine, S.: Modular multitask reinforcement learning with policy sketches. In: Proceedings of the 34th International Conference on Machine Learning. vol. 70, pp. 166–175. PMLR (2017)
7. Andrychowicz, M., Raichuk, A., Stań'czyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., Bachem, O.: What matters for on-policy deep actor-critic methods? a large-scale study. In: 2021 International Conference on Learning Representations (2021)
8. Aram, M., Neumann, G.: Multilayered analysis of co-development of business information systems. Journal of Internet Services and Applications **6**(13) (2015)
9. Bamford, C., Huang, S., Lucas, S.: Griddly: A platform for AI research in games. In: AAAI-21 Workshop on Reinforcement Learning in Games (2021)
10. Banerjee, B., Stone, P.: General game learning using knowledge transfer. In: The 20th International Joint Conference on Artificial Intelligence. pp. 672–677 (2007)
11. Beck, J., Vuorio, R., Liu, E.Z., Xiong, Z., Zintgraf, L., Finn, C., Whiteson, S.: A survey of meta-reinforcement learning. https://arxiv.org/abs/2301.08028 (2023)
12. Bellemare, M.G., Candido, S., Castro, P.S., Gong, J., Machado, M.C., Moitra, S., Ponda, S.S., Wang, Z.: Autonomous navigation of stratospheric balloons using reinforcement learning. Nature **588**, 77–82 (2020)
13. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research **47**(1), 253–279 (2013)
14. Benjamins, C., Eimer, T., Schubert, F., Mohan, A., Döhler, S., Biedenkapp, A., Rosenhahn, B., Hutter, F., Lindauer, M.: Contextualize me – the case for context in reinforcement learning. Transactions on Machine Learning Research (2023)
15. Bettini, M., Kortvelesy, R., Blumenkamp, J., Prorok, A.: VMAS: A vectorized multi-agent simulator for collective robot learning. In: Proceedings of the 16th International Symposium on Distributed Autonomous Robotic Systems. DARS '22, Springer (2022)
16. Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., anda V. Voleti, Z.E., Letts, A., Jampani, V., Rombach, R.: Stable video diffusion: Scaling latent video diffusion models to large datasets. https://arxiv.org/abs/2311.15127 (2023)
17. Blili-Hamelin, B., Graziul, C., Hancox-Li, L., Hazan, H., El-Mhamdi, E.M., Ghosh, A., Heller, K., Metcalf, J., Murai, F., Salvaggio, E., Smart, A., Snider, T., Tighanimine, M., Ringer, T., Mitchell, M., Dori-Hacohen, S.: Position: Stop treating 'AGI' as the north-star goal of AI research. In: Proceedings of the 42nd International Conference on Machine Learning (2025), to appear
18. Bonnet, C., Luo, D., Byrne, D., Surana, S., Abramowitz, S., Duckworth, P., Coyette, V., Midgley, L.I., Tegegn, E., Kalloniatis, T., Mahjoub, O., Macfarlane, M., Smit, A.P., Grinsztajn, N., Bolge, R., Waters, C.N., Mimouni, M.A., Sob, U.A.M., de Kock, R., Singh, S., Furelos-Blanco, D., Le, V., Pretorius, A., Laterre, A.: Jumanji: a diverse suite of scalable reinforcement learning environments in JAX. In: Proceedings of the International Conference on Learning Representations (2024)
19. Bou Ammar, H., Eaton, E., Ruvolo, P., Taylor, M.E.: Online multi-task learning for policy gradient methods. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 1206–1214 (2014)
20. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q.: JAX:

composable transformations of Python+NumPy programs (2018), http://github.com/google/jax
21. Branavan, S.R.K., Silver, D., Barzilay, R.: Learning to win by reading manuals in a Monte-Carlo framework. Journal of Artificial Intelligence Research **43**, 661–704 (2012)
22. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI gym. https://arxiv.org/abs/1606.01540 (2016)
23. Browne, C., Soemers, D.J.N.J., Piette, É., Stephenson, M., Crist, W.: Ludii language reference. ludii.games/downloads/LudiiLanguageReference.pdf (2020)
24. Browne, C.B.: Automatic Generation and Evaluation of Recombination Games. Phd thesis, Faculty of Information Technology, Queensland University of Technology, Queensland, Australia (2009)
25. Cobbe, K., Hesse, C., Hilton, J., Schulman, J.: Leveraging procedural generation to benchmark reinforcement learning. In: Daumé III, H., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 2048–2056 (2020)
26. Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 1282–1289. PMLR (2019)
27. Cummins, C., Wasti, B., Guo, J., Cui, B., Ansel, J., Gomez, S., Jain, S., Liu, J., Teytaud, O., Steiner, B., Tian, Y., Leather, H.: Compilergym: Robust, performant compiler optimization environments for ai research. https://arxiv.org/abs/2109.08267 (2021)
28. Dalton, S., Frosio, I.: Accelerating reinforcement learning through GPU Atari emulation. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 19773–19782. Curran Associates, Inc. (2020)
29. Davidson, G., Gureckis, T.M.: Toward complex and structured goals in reinforcement learning. In: Finding the Frame workshop @ Reinforcement Learning Conference (2024)
30. Davidson, G., Todd, G., Togelius, J., Gureckis, T.M., Lake, B.M.: Goals as reward-producing programs. https://arxiv.org/abs/2405.13242 (2024)
31. Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de las Casas, D., Donner, C., Fritz, L., Galperti, C., Huber, A., Keeling, J., Tsimpoukelli, M., Kay, J., Merle, A., Moret, J.M., Noury, S., Pesamosca, F., Pfau, D., Sauter, O., Sommariva, C., Coda, S., Duval, B., Fasoli, A., Kohli, P., Kavukcuoglu, K., Hassabis, D., Riedmiller, M.: Magnetic control of tokamak plasmas through deep reinforcement learning. Nature **602**, 414–419 (2022)
32. Deisenroth, M.P., Englert, P., Peters, J., Fox, D.: Multi-task policy search for robotics. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 3876–3881 (2014)
33. Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., Levine, S.: Emergent complexity and zero-shot transfer via unsupervised environment design. In: Advances in Neural Information Processing Systems. vol. 33, pp. 13049–13061 (2020)
34. Desai, A., Gulwani, S., Hingorani, V., Jain, N., Karkare, A., Marron, M., R, S., Roy, S.: Program synthesis using natural language. In: Proceedings of the 38th International Conference on Software Engineering. p. 345–356. Association for Computing Machinery (2016)

35. Eimer, T., Biedenkapp, A., Reimer, M., Adriaensen, S., Hutter, F., Lindauer, M.: DACBench: A benchmark library for dynamic algorithm configuration. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence. pp. 1668–1674 (2021)
36. Eimer, T., Lindauer, M., Raileanu, R.: Hyperparameters in reinforcement learning and how to tune them. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 9104–9149. PMLR (2023)
37. Ellis, B., Cook, J., Moalla, S., Samvelyan, M., Sun, M., Mahajan, A., Foerster, J.N., Whiteson, S.: SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems (2023), accepted.
38. Farebrother, J., Machado, M.C., Bowling, M.: Generalization and regularization in DQN. https://arxiv.org/abs/1810.00123 (2018)
39. Fernandéz, F., Veloso, M.: Learning domain structure through probabilistic policy reuse in reinforcement learning. Progress in AI **2**(1), 13–27 (2013)
40. Finn, C., Abeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1126–1135 (2017)
41. Freeman, C.D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., Bachem, O.: Brax - a differentiable physics engine for large scale rigid body simulation (2021), http://github.com/google/brax
42. Frey, S., Li, K., Nagy, P., Sapora, S., Lu, C., Zohren, S., Foerster, J., Calinescu, A.: JAX-LOB: A GPU-accelerated limit order book simulator to unlock large scale reinforcement learning for trading. In: ICAIF '23: Proceedings of the Fourth ACM International Conference on AI in Finance. pp. 583–591 (2023)
43. Gamrian, S., Goldberg, Y.: Transfer learning for related reinforcement learning tasks via image-to-image translation. In: Proceedings of the 36th International Conference on Machine Learning. pp. 2063–2072 (2019)
44. Genesereth, M., Thielscher, M.: General Game Playing. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2014)
45. Ghosh, D., Rahme, J., Kumar, A., Zhang, A., Adams, R.P., Levine, S.: Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 25502–25515. Curran Associates, Inc. (2021)
46. Glatt, R., da Silva, F.L., Costa, A.H.R.: Towards knowledge transfer in deep reinforcement learning. In: Brazilian Conference on Intelligent Systems (BRACIS). pp. 91–96. IEEE (2016)
47. Glatt, R., Silva, F.L.D., da Costa Bianchi, R.A., Costa, A.H.R.: DECAF: Deep case-based policy inference for knowledge transfer in reinforcement learning. Expert Systems with Applications **156** (2020)
48. Goldie, A.D., Lu, C., Jackson, M.T., Whiteson, S., Foerster, J.N.: Can learned optimization make reinforcement learning less difficult? In: AutoRL Workshop @ ICML 2024 (2024)
49. Goldwaser, A., Thielscher, M.: Deep reinforcement learning for general game playing. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence. pp. 1701–1708. AAAI Press (2020)

50. Goyal, P., Niekum, S., Mooney, R.J.: PixL2R: Guiding reinforcement learning using natural language by mapping pixels to rewards. In: Proceedings of the 2020 Conference on Robot Learning. PMLR, vol. 155, pp. 485–497 (2021)
51. Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., Co-Reyes, J.D., Agarwal, R., Roelofs, R., Lu, Y., Montali, N., Mougin, P., Yang, Z., B, W., Faust, A., McAllister, R., Anguelov, D., Sapp, B.: Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (2023)
52. Hartman, S., Ong, C.S., Powles, J., Kuhnert, P.: Position: We need responsible, application-driven (RAD) AI research. In: Proceedings of the 42nd International Conference on Machine Learning (2025), to appear
53. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence. pp. 3207–3214. AAAI (2018)
54. Hooker, S.: The hardware lottery. Communications of the ACM **64**(12), 58–65 (2021)
55. Howard, R.A.: Dynamic Programming and Markov Processes. The MIT Press, Cambridge, Massachusetts (1960)
56. Irpan, A., Song, X.: The principle of unchanged optimality in reinforcement learning generalization. In: ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning (2019)
57. Jackson, M.T., Jiang, M., Parker-Holder, J., Vuorio, R., Lu, C., Farquhar, G., Whiteson, S., Foerster, J.N.: Discovering general reinforcement learning algorithms with adversarial environment design. In: Advances in Neural Information Processing Systems. vol. 36, pp. 79980–79998 (2023)
58. Jacob, M., Devlin, S., Hofmann, K.: "it's unwieldy and it takes a lot of time" — challenges and opportunities for creating agents in commercial games. In: Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. pp. 88–94 (2020)
59. Jordan, S.M., White, A., da Silva, B.C., White, M., Thomas, P.S.: Position: Benchmarking is limited in reinforcement learning research. In: Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., Berkenkamp, F. (eds.) Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 22551–22569. PMLR (2024)
60. Jordan, S.M.: Scientific experiments in reinforcement learning. https://nips.cc/virtual/2022/63891 (2022), opinion talk contributed to the Deep Reinforcement Learning Workshop at NeurIPS 2022
61. Jothimurugan, K., Alur, R., Bastani, O.: A composable specification language for reinforcement learning tasks. In: Advances in Neural Information Processing Systems. vol. 32, pp. 13041–13051 (2019)
62. Justesen, N., Torrado, R.R., Bontrager, P., Khalifa, A., Togelius, J., Risi, S.: Illuminating generalization in deep reinforcement learning through procedural level generation. In: NeurIPS 2018 Workshop on Deep Reinforcement Learning (2018)
63. Kaiser, Ł., Stafiniak, Ł.: First-order logic with counting for general game playing. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 25, pp. 791–796 (2011)

64. Kaufmann, T., Weng, P., Bengs, V., Hüllermeier, E.: A survey of reinforcement learning from human feedback. Transactions on Machine Learning Research (2025)
65. Kharyal, C., Krishna Gottipati, S., Kumar Sinha, T., Das, S., Taylor, M.E.: GLIDE-RL: Grounded language instruction through DEmonstration in RL. https://arxiv.org/abs/2401.02991 (2024)
66. Kirk, R., Zhang, A., Grefenstette, E., Rocktäschel, T.: A survey of zero-shot generalisation in deep reinforcement learning. Journal of Artificial Intelligence Research **76**, 201–264 (2023)
67. Kowalksi, J., Miernik, R., Mika, M., Pawlik, W., Sutowicz, J., Szykuła, M., Tkaczyk, A.: Efficient reasoning in regular boardgames. In: Proceedings of the 2020 IEEE Conference on Games. pp. 455–462. IEEE (2020)
68. Kowalski, J., Maksymilian, M., Sutowicz, J., Szykuła, M.: Regular boardgames. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence. vol. 33, pp. 1699–1706. AAAI Press (2019)
69. Koyamada, S., Okano, S., Nishimori, S., Murata, Y., Habara, K., Kita, H., Ishii, S.: Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In: Advances in Neural Information Processing Systems (2023)
70. Kuhlmann, G., Stone, P.: Graph-based domain mapping for transfer learning in general games. In: Kok, J., Koronacki, J., Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) Machine Learning: ECML 2007. Lecture Notes in Computer Science, vol. 4071, pp. 188–200. Springer, Berlin, Heidelberg (2007)
71. Lange, R.T.: gymnax: A JAX-based reinforcement learning environment library (2022), http://github.com/RobertTLange/gymnax
72. Lange, S., Riedmiller, M.: Deep auto-encoder neural networks in reinforcement learning. In: Neural Networks. International Joint Conference. 2010. (IJCNN 2010). pp. 1623–1630. IEEE (2010)
73. Lazaric, A.: Transfer in reinforcement learning: a framework and a survey. In: Wiering, M., van Otterlo, M. (eds.) Reinforcement Learning. Adaptation, Learning, and Optimization, vol. 12, pp. 143–173. Springer, Berlin, Heidelberg (2012)
74. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
75. Lee, J.N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., Brunskill, E.: Supervised pretraining can learn in-context reinforcement learning. https://arxiv.org/abs/2306.14892 (2023)
76. Leivada, E., Marcus, G., Günther, F., Murphy, E.: A sentence is worth a thousand pictures: Can large language models understand hum4n l4ngu4ge and the w0rld behind w0rds? https://arxiv.org/abs/2308.00109 (2024)
77. Li, H., Liao, X., Carin, L.: Multi-task reinforcement learning in partially observable stochastic environments. Journal of Machine Learning Research **10**(40), 1131–1186 (2009)
78. Li, X., Zhang, J., Bian, J., Tong, Y., Liu, T.Y.: A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network. In: Agmon, N., Taylor, M.E., Veloso, E.E.M. (eds.) Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019). pp. 980–988 (2019)
79. Lifschitz, S., Paster, K., Chan, H., Ba, J., McIlraith, S.: Steve-1: A generative model for text-to-behavior in Minecraft. https://arxiv.org/abs/2306.00937 (2023)
80. Lin, S., Bercher, P.: On the expressive power of planning formalisms in conjunction with LTL. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 32, pp. 231–240 (2022)

81. Lindauer, M., Karl, F., Klier, A., Moosbauer, J., Tornede, A., Mueller, A., Hutter, F., Feurer, M., Bischl, B.: Position: A call to action for a human-centered AutoML paradigm. In: Proceedings of the 41st International Conference on Machine Learning. PMLR, vol. 235, pp. 30566–30584 (2024)
82. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the Eleventh International Conference on Machine Learning. pp. 157–163 (1994)
83. Love, N., Hinrichs, T., Haley, D., Schkufza, E., Genesereth, M.: General game playing: Game description language specification. Tech. Rep. LG-2006-01, Stanford Logic Group (2008)
84. Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., Rocktä"schel, T.: A survey of reinforcement learning informed by natural language. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 6309–6317 (2019)
85. Luo, J., Hu, Z., Xu, C., Gadipudi, S., Sharma, A., Ahmad, R., Schaal, S., Finn, C., Gupta, A., Levine, S.: SERL: A software suite for sample-efficient robotic reinforcement learning. In: Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition @ CoRL2023 (2023)
86. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008)
87. Machado, M.C., Bellemare, M.G., Talvitie, E., Veness, J., Hausknecht, M., Bowling, M.: Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. Journal of Artificial Intelligence Research **61**, 523–562 (2018)
88. Maclin, R., Shavlik, J.W.: Creating advice-taking reinforcement learners. Machine Learning **22**, 251–281 (1996)
89. Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., State, G.: Isaac gym: High performance GPU based physics simulation for robot learning. In: Vanschoren, J., Yeung, S. (eds.) Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks. vol. 1. Curran (2021)
90. Malik, D., Li, Y., Ravikumar, P.: When is generalizable reinforcement learning tractable? In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 8032–8045. Curran Associates, Inc. (2021)
91. Mannor, S., Tamar, A.: Towards deployable RL – what's broken with RL research and a potential fix. https://arxiv.org/abs/2301.01320 (2023)
92. Maras, M., Kępa, M., Kowalski, J., Szykuła, M.: Fast and knowledge-free deep learning for general game playing (student abstract). In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 23576–23578 (2024)
93. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL—the planning domain definition language. Tech. Rep. CVC TR98003/DCS TR1165, New Haven, CT: Yale Center for Computational Vision and Control (1998)
94. Mediratta, I., You, Q., Jiang, M., Raileanu, R.: A study of generalization in offline reinforcement learning. In: 2024 International Conference on Learning Representations (2024)
95. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Computing Surveys **37**(4), 316–344 (2005)

96. Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J.W., Songhori, E., Wang, S., Lee, Y.J., Johnson, E., Pathak, O., Nazi, A., Pak, J., Tong, A., Srinivasa, K., Hang, W., Tuncer, E., Le, Q.V., Laudon, J., Ho, R., Carpenter, R., Dean, J.: A graph placement methodology for fast chip design. Nature **594**, 207–212 (2021)
97. Misra, D., Langford, J., Artzi, Y.: Mapping instructions and visual observations to actions with reinforcement learning. In: Palmer, M., Hwa, R., Riedel, S. (eds.) Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 1004–1015 (2017)
98. Mittel, A., Munukutla, P.S.: Visual transfer between Atari games using competitive reinforcement learning. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 499–501 (2019)
99. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with deep reinforcement learning. https://arxiv.org/abs/1312.5602 (2013)
100. Mohan, A., Benjamins, C., Wienecke, K., Dockhorn, A., Lindauer, M.: AutoRL hyperparameter landscapes. In: Faust, A., Garnett, R., White, C., Hutter, F., Gardner, J.R. (eds.) International Conference on Automated Machine Learning. Proceedings of Machine Learning Research, vol. 224. PMLR (2023)
101. Mohan, A., Zhang, A., Lindauer, M.: Structure in deep reinforcement learning: A survey and open problems. Journal of Artificial Intelligence Research **79**, 1167–1236 (2024)
102. Müller-Brockhausen, M., Preuss, M., Plaat, A.: Procedural content generation: Better benchmarks for transfer reinforcement learning. In: Proceedings of the 2021 IEEE Conference on Games. pp. 924–931 (2021)
103. Nagabandi, A., Clavera, I., Liu, S., Fearing, R.S., Abbeel, P., Levine, S., Finn, C.: Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In: Proceedings of the 2019 International Conference on Learning Representations (2019)
104. Nichol, A., Pfau, V., Hesse, C., Klimov, O., Schulman, J.: Gotta learn fast: A new benchmark for generalization in RL. https://arxiv.org/abs/1804.03720 (2018)
105. Obando-Ceron, J., Araú'jo, J.G.M., Courville, A., Castro, P.S.: On the consistency of hyper-parameter selection in value-based deep reinforcement learning. In: Reinforcement Learning Conference (2024), accepted
106. Obando-Ceron, J.S., Castro, P.S.: Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. pp. 1373–1383. PMLR (2021)
107. Oh, J., Singh, S., Lee, H., Kohli, P.: Zero-shot task generalization with multi-task deep reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning. pp. 2661–2670. PMLR (2017)
108. OpenAI: Introducing ChatGPT. https://openai.com/blog/chatgpt (2022), accessed: 2024-01-02
109. Oswald, J., Srinivas, K., Kokel, H., Lee, J., Katz, M., Sohrabi, S.: Large language models as planning domain generators. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 34, pp. 423–431 (2024)
110. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R.: Training language models to follow instructions with human feedback. In: Koyejo, S.,

Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 27730–27744. Curran Associates, Inc. (2022)

111. Parker-Holder, J., Jiang, M., Dennis, M., Samvelyan, M., Foerster, J., Grefenstette, E., Rocktäschel, T.: Evolving curricula with regret-based environment design. In: Proceedings of the 39th International Conference on Machine Learning. PMLR, vol. 162, pp. 17473–17498 (2022)
112. Parker-Holder, J., Rajan, R., Song, X., Biedenkapp, A., Miao, Y., Eimer, T., Zhang, B., Nguyen, V., Calandra, R., Faust, A., Hutter, F., Lindauer, M.: Automated reinforcement learning (autoRL): A survey and open problems. Journal of Artificial Intelligence Research **74**, 517–568 (2022)
113. Patterson, A., Neumann, S., White, M., White, A.: Empirical design in reinforcement learning. https://arxiv.org/abs/2304.01315 (2023)
114. Perez-Liebana, D., Dockhorn, A., Grueso, J.H., Jeurissen, D.: The design of "Stratega": A general strategy games framework. In: Osborn, J.C. (ed.) Joint Proceedings of the AIIDE 2020 Workshops co-located with 16th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2020). CEUR Workshop Proceedings (2020)
115. Piette, É., Soemers, D.J.N.J., Stephenson, M., Sironi, C.F., Winands, M.H.M., Browne, C.: Ludii – the ludemic general game system. In: Giacomo, G.D., Catala, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J. (eds.) Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020). Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 411–418. IOS Press (2020)
116. Piette, É., Stephenson, M., Soemers, D.J.N.J., Browne, C.: General board game concepts. In: Proceedings of the 2021 IEEE Conference on Games (CoG). pp. 932–939. IEEE (2021)
117. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R.: SDXL: Improving latent diffusion models for high-resolution image synthesis. https://arxiv.org/abs/2307.01952 (2023)
118. Ponse, K., Kleuker, J.F., Moerland, T.M., Plaat, A.: Chargax: A JAX accelerated EV charging simulator. In: Reinforcement Learning Conference (2025), to appear
119. Rakelly, K., Zhou, A., Quillen, D., Finn, C., Levine, S.: Efficient off-policy meta-reinforcement learning via probabilistic context variables. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 5331–5340 (2019)
120. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with CLIP latents. https://arxiv.org/abs/2204.06125 (2022)
121. Rani, S., Booth, S., Sreedharan, S.: Goals vs. rewards: A comparative study of objective specification mechanisms. In: Reinforcement Learning Conference (2025), to appear
122. Raparthy, S.C., Hambro, E., Kirk, R., Henaff, M., Raileanu, R.: Generalization to new sequential decision making tasks with in-context learning. https://arxiv.org/abs/2312.03801 (2023)
123. Reed, S., Żołna, 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., de Freitas, N.: A generalist agent. Transactions on Machine Learning Research (2023)

124. Riedmiller, M.: Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In: Machine Learning: ECML 2005. Lecture Notes in Computer Science, vol. 3720, pp. 317–328. Springer (2005)
125. Rigter, M., Jiang, M., Posner, I.: Reward-free curricula for training robust world models. In: International Conference on Learning Representations (2024)
126. Rodriguez-Sanchez, R., Spiegel, B.A., Wang, J., Patel, R., Tellex, S., Konidaris, G.: RLang: A declarative language for describing partial world knowledge to reinforcement learning agents. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 29161–29178. PMLR (2023)
127. Rolnick, D., Aspuru-Guzik, A., Beery, S., Dilkina, B., Donti, P.L., Ghassemi, M., Kerner, H., Monteleoni, C., Rolf, E., Tambe, M., White, A.: Position: Application-driven innovation in machine learning. In: Proceedings of the 41st International Conference on Machine Learning. PMLR, vol. 235, pp. 42707–42718 (2024)
128. Rostami, M., Isele, D., Eaton, E.: Using task descriptions in lifelong machine learning for improved performance and zero-shot transfer. Journal of Artificial Intelligence Research **67**, 673–703 (2020)
129. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. https://arxiv.org/abs/1606.04671 (2016)
130. Rutherford, A., Ellis, B., Gallici, M., Cook, J., Lupu, A., Ingvarsson, G., Willi, T., Khan, A., de Witt, C.S., Souly, A., Bandyopadhyay, S., Samvelyan, M., Jiang, M., Lange, R.T., Whiteson, S., Lacerda, B., Hawes, N., Rocktäschel, T., Lu, C., Foerster, J.N.: JaxMARL: Multi-agent RL environments and algorithms in JAX. https://arxiv.org/abs/2311.10090 (2023)
131. Sakçak, B., Shell, D.A., O'Kane, J.M.: Limits of specifiability for sensor-based robotic planning tasks. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA) (2025), to appear
132. Samvelyan, M., Khan, A., Dennis, M., Jiang, M., Parker-Holder, J., Foerster, J., Raileanu, R., Rocktäschel, T.: MAESTRO: Open-ended environment design for multi-agent reinforcement learning. In: International Conference on Learning Representations (2023)
133. Samvelyan, M., Kirk, R., Kurin, V., Parker-Holder, J., Jiang, M., Hambro, E., Petroni, F., Küttler, H., Grefenstette, E., Rocktäschel, T.: Minihack the planet: A sandbox for open-ended reinforcement learning research. In: Advances in Neural Information Processing Systems (2021)
134. Schaul, T.: A video game description language for model-based or interactive learning. In: Proceedings of the IEEE Conference on Computational Intelligence in Games. pp. 193–200. IEEE (2013)
135. Schaul, T., Horgan, D., Gregor, K., Silver, D.: Universal value function approximators. In: Proceedings of the 32nd International Conference on Machine Learning. JLMR: W&CP, vol. 37, pp. 1312–1320 (2015)
136. Schaul, T.: An extensible description language for video games. IEEE Transactions on Computational Intelligence and AI in Games **6**(4), 325–331 (Dec 2014). https://doi.org/10.1109/TCIAIG.2014.2352795
137. Schmidhuber, J.: On learning how to learn learning strategies. Tech. Rep. FKI-198-94, Institut für Informatik, Technische Universität München (1994)
138. Seger, E., Ovadya, A., Siddarth, D., Garfinkel, B., Dafoe, A.: Democratising AI: Multiple meanings, goals, and methods. In: Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society. pp. 715–722 (2023)

139. Shu, T., Xiong, C., Socher, R.: Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In: International Conference on Learning Representations (2018)
140. Silver, T., Chitnis, R.: PDDLGym: Gym environments from PDDL problems. In: ICAPS Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL) (2020)
141. Simkute, A., Luger, E., Evans, M., Jones, R.: "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 (2024)
142. Sobol, D., Wolf, L., Taigman, Y.: Visual analogies between atari games for studying transfer learning in rl. https://arxiv.org/abs/1807.11074 (2018)
143. Soemers, D.J.N.J., Mella, V., Browne, C., Teytaud, O.: Deep learning for general game playing with Ludii and Polygames. ICGA Journal **43**(3), 146–161 (2022)
144. Soemers, D.J.N.J., Mella, V., Piette, É., Stephenson, M., Browne, C., Teytaud, O.: Towards a general transfer approach for policy-value networks. Transactions on Machine Learning Research (2023)
145. Soemers, D.J.N.J., Piette, É., Stephenson, M., Browne, C.: The Ludii game description language is universal. In: Proceedings of the 2024 IEEE Conference on Games. pp. 1–8 (2024)
146. Soemers, D.J.N.J., Samothrakis, S., Driessens, K., Winands, M.H.M.: Environment descriptions for usability and generalisation in reinforcement learning. In: Rocha, A.P., Steels, L., van den Herik, H.J. (eds.) Proceedings of the 17th International Conference on Agents and Artificial Intelligence. vol. 3, pp. 983–992 (2025)
147. Stability AI: Stable audio: Fast timing-conditioned latent audio diffusion. https://stability.ai/research/stable-audio-efficient-timing-latent-diffusion (2023), accessed: 2024-1-4
148. Stephenson, M., Soemers, D.J.N.J., Piette, É., Browne, C.: Measuring board game distance. In: Browne, C., Kishimoto, A., Schaeffer, J. (eds.) Computers and Games. CG 2022. Lecture Notes in Computer Science, vol. 13865, pp. 121–130. Springer, Cham (2023)
149. Stone, A., Ramirez, O., Konolige, K., Jonschkowski, R.: The distracting control suite – a challenging benchmark for reinforcement learning from pixels. https://arxiv.org/abs/2101.02722 (2021)
150. Sun, S.H., Wu, T.L., Lim, J.J.: Program guided agent. In: International Conference on Learning Representations (2020)
151. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 2 edn. (2018)
152. Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., Riedmiller, M.: DeepMind control suite (2018)
153. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. In: Mahadevan, S. (ed.) Journal of Machine Learning Research. vol. 10, pp. 1633–1685 (2009)
154. Terry, J.K., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sullivan, R., Santos, L., Perez, R., Horsch, C., Dieffendahl, C., Williams, N.L., Lokesh, Y., Ravi, P.: Pettingzoo: A standard API for multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems. vol. 34, pp. 15032–15043. Curran Associates, Inc. (2021)

155. Tessler, C., Givony, S., Zahavy, T., Mankowitz, D., Mannor, S.: A deep hierarchical approach to lifelong learning in minecraft. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 1553–1561. AAAI (2017)
156. Thielscher, M.: The general game playing description language is universal. In: Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence, IJCAI-11. pp. 1107–1112 (2011)
157. Todd, G., Padula, A., Stephenson, M., Piette, É., Soemers, D.J.N.J., Togelius, J.: GAVEL: Generating games via evolution and language models. In: Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., Zhang, C. (eds.) Advances in Neural Information Processing Systems. vol. 37, pp. 110723–110745. Curran Associates, Inc. (2024)
158. Todd, G., Padula, A.G., Soemers, D.J.N.J., Togelius, J.: Ludax: A GPU-accelerated domain specific language for board games. https://arxiv.org/abs/2506.22609 (2025)
159. Trofin, M., Qian, Y., Brevdo, E., Lin, Z., Choromanski, K., Li, D.: MLGO: a machine learning guided compiler optimizations framework. https://arxiv.org/abs/2101.04808 (2021)
160. Voelcker, C., Hussing, M., Eaton, E.: Can we hop in general? A discussion of benchmark selection and design using the Hopper environment. In: Finding the Frame workshop @ Reinforcement Learning Conference (2024)
161. van der Wal, J.: Stochastic Dynamic Programming. No. 139 in Mathematical Centre tracts, Morgan Kaufmann, Amsterdam (1981)
162. Whiteson, S., Tanner, B., Taylor, M.E., Stone, P.: Protecting against evaluation overfitting in empirical reinforcement learning. In: 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). pp. 120–127 (2011)
163. Williams, E.C., Gopalan, N., Rhee, M., Tellex, S.: Learning to parse natural language to gronuded reward functions with weak supervision. In: 2018 IEEE International Conference on Robotics and Automation. pp. 4430–4436 (2018)
164. Wilson, A., Fern, A., Ray, S., Tadepalli, P.: Multi-task reinforcement learning: A hierarchical Bayesian approach. In: Proceedings of the 24th International Conference on Machine Learning. pp. 1015–1022 (2007)
165. Yang, K.: Learn dynamic-aware state embedding for transfer learning. https://arxiv.org/abs/2101.02230 (2021)
166. Yang, M., Du, Y., Ghasemipour, K., Tompson, J., Kaelbling, L., Schuurmans, D., Abbeel, P.: Learning interactive real-world simulators. In: International Conference on Learning Representations (2024)
167. Yoon, D., Hong, S., Lee, B.J., Kim, K.E.: Winning the L2RPN challenge: Power grid management via semi-Markov afterstate actor critic. In: Proceedings of the International Conference on Learning Representations (2021)
168. Zhang, A., Ballas, N., Pineau, J.: A dissection of overfitting and generalization in continuous reinforcement learning. https://arxiv.org/abs/1806.07937 (2020)
169. Zhang, C., Vinyals, O., Munos, R., Bengio, S.: A study on overfitting in deep reinforcement learning. https://arxiv.org/abs/1804.06893 (2018)
170. Zhao, W., Queralta, J.P., Westerlund, T.: Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 737–744 (2020)
171. 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., Wen, J.R.: A survey of large language models. https://arxiv.org/abs/2303.18223 (2023), accessed: 25-01-2024

172. Zhong, V., Rocktäschel, T., Grefenstette, E.: RTFM: Generalising to novel environment dynamics via reading. In: International Conference on Learning Representations (2020)
173. Zhu, Z., de Salvo Braz, R., Bhandari, J., Jiang, D., Wan, Y., Efroni, Y., Wang, L., Xu, R., Guo, H., Nikulkov, A., Korenkevych, D., Dogan, U., Cheng, F., Wu, Z., Xu, W.: Pearl: A production-ready reinforcement learning agent. https://arxiv.org/abs/2312.03814 (2023)
174. Zhu, Z., Lin, K., Jain, A.K., Zhou, J.: Transfer learning in deep reinforcement learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(11), 13344–13362 (2023)
175. Zintgraf, L.: Fast Adaptation via Meta Reinforcement Learning. Ph.D. thesis, University of Oxford, Oxford, United Kingdom (2022)
176. Zuo, M., Velez, F.P., Li, X., Littman, M.L., Bach, S.H.: Planetarium: A rigorous benchmark for translating text to structured planning languages. https://arxiv.org/abs/2407.03321 (2024)