Published in the Proceedings of the 11th Conference on Artificial General Intelligence (August 22-25, 2008 in Prague)

Task Analysis for Teaching Cumulative Learners*

Jordi E. Bieger[1,2] and Kristinn R. Thórisson[1,3]

[1] Center for Analysis and Design of Intelligent Agents, Reykjavik University, Iceland
[2] ICT group, Delft University of Technology, The Netherlands
[3] Icelandic Institute for Intelligent Machines, Iceland
{jordi13,thorisson}@ru.is

Abstract. A generally intelligent machine (AGI) should be able to learn a wide range of tasks. Knowledge acquisition in complex and dynamic task-environments cannot happen all-at-once, and AGI-aspiring systems must thus be capable of *cumulative learning:* efficiently making use of existing knowledge during learning, supporting increases in the scope of ability and knowledge, incrementally and predictably — without catastrophic forgetting or mangling of existing knowledge. Where relevant expertise is at hand the learning process can be aided by curriculum-based teaching, where a teacher divides a high-level task up into smaller and simpler pieces and presents them in an order that facilitates learning. Creating such a curriculum can benefit from expert knowledge of (a) the task domain, (b) the learning system itself, and (c) general teaching principles. Curriculum design for AI systems has so far been rather ad-hoc and limited to systems incapable of cumulative learning. We present a task analysis methodology that utilizes expert knowledge and is intended to inform the construction of teaching curricula for cumulative learners. Inspired in part by methods from knowledge engineering and functional requirements analysis, our strategy decomposes high-level tasks in three ways based on involved actions, features and functionality. We show how this methodology can be used for a (simplified) *arrival control* task from the air traffic control domain, where extensive expert knowledge is available and teaching cumulative learners is required to facilitate the safe and trustworthy automation of complex workflows.

Keywords: Artificial Intelligence · Artificial Pedagogy · Curriculum Learning · Task Theory · Trustworthy Automation

1 Introduction

In learning complex tasks humans tend to take an incremental approach (e.g. learning the meaning of traffic signs before driving in traffic, or learning to fly a single-propeller plane before flying a jumbo jet), where newly acquired knowledge and skills build on those priorly acquired. If a skill is in some way related to one

* The authors gratefully acknowledge partial funding for this project from Isavia, IIIM, Reykjavik University, Delft Univeresity of Technology, and the Netherlands Organization for Scientific Research (NWO grant 313-99-3160 / Values4Water project).

we have learned before—e.g. learning to play squash after racquetball—the first task will often help us learn the second one [8].[4].Humans can typically learn and alternate between many different tasks without completely forgetting them in the process, and often apply lessons learned across multiple domains. We consider such cumulative learning abilities to be a necessary feature of (artificial) general intelligence (AGI), but no artificial intelligence (AI) or machine learning (ML) system to date can rival humans in these regards.

The ideal (artificial) cumulative learner (CL), in our conceptualization, can acquire knowledge and skills through both experience [22] and teaching [5,6]. Their learning is 'always on' throughout their lifetime[5] (lifelong learning [19]), and happens continuously as new experiences accumulate (online learning [28]). Their knowledge is defeasible [17] ("better knowledge replaces worse knowledge"), and new knowledge is reconciled with old knowledge (old-new integration [16]: new knowledge can be used in conjunction with, and integrated with, older tasks — irrespective of overlap). Knowledge from one task or domain can be applied to speed up learning another (e.g. through analogy; transfer learning [14]), without catastrophic interference/forgetting [9,11], possibly to the point that few-shot learning is enabled [12]. A cumulative learner that fulfills all of these features will tend to grow its capabilities over time to cover a wide range of tasks (multitask learning [24]) as experience accumulates.

The order in which information, tasks, and subtasks are encountered can have a large influence on the efficiency and efficacy of the cumulative learning process. Pedagogical methodologies like *shaping* [20], *scaffolding* [27] and *part-task training* [23] take advantage of this for teaching humans and animals. In these approaches teachers use extensive domain knowledge to decompose complex high-level tasks into smaller and simpler subtasks that are manageable by the learner, and gradually introduce other subtasks or complexity. Similarly hierarchical methods have been applied to various existing AI approaches [10,13,2]. However, these methods are not developed for systems with advanced cumulative learning abilities.[6] Furthermore, in most of these cases no domain knowledge is utilized for defining the curriculum: the subdivision and presentation order is determined by the learner—which is complicated by the learner's limited (domain) knowledge and control over the environment (especially at the start)—or

[4] *Negative* transfer of training may also occur, where pre-existing knowledge interferes with learning something new — e.g. a racquetball player may take longer to get used to the way a squash ball bounces than somebody who never played racquetball. An optimal curriculum would mitigate negative transfer as much as possible.

[5] However this is measured, we expect at a minimum the 'learning cycle' (alternating learning and non-learning periods) to be free from designer intervention at runtime. Given that, the smaller those periods become (relative to the shortest perception-action cycle, for instance), to the point of being considered virtually or completely continuous, the better the "learning always on" requirement is being met.

[6] For instance, they typically require up-front full data disclosure (final data set up-front), all-at-once training (to train on the final data set from the very beginning) and learning-free deployment (the need to turn off learning before deployment to avoid unpredictable drift; cf. [15]).

by an algorithm that uses (slowly) discovered structural features of the task-environment. While this can be a strength in cases where domain knowledge is unavailable, in this paper the focus is on the case where domain expertise *does* exist, and on converting such knowledge into a subtask hierarchy or curriculum.

We present a task analysis methodology intended to aid in the instructional design of curricula to teach artificial cumulative learners, as part of broader work on *artificial pedagogy* [4] and *task theory* [25]. Here we focus on the design phase of the ADDIE model [7] from the instructional design field, with some assumptions about the analysis phase based on features of cumulative learners.[7] Guidelines are given for extracting knowledge from domain experts (Sec. 2) to decompose high-level tasks along three dimensions (Sec. 3), producing a hierarchy of (smaller and simpler) subtasks for which functional requirements are known, which can inform curriculum design (Sec. 4).

We illustrate our methodology by showing how a curriculum might be constructed for the use case of automating the task of *arrival control* (AC)[8] for the Icelandic air traffic control (ATC) agency Isavia[9] (Sec. 5), where a combined AI–human control structure could increase efficiency and safety. AC's goal is to create an optimal flow of landings by telling incoming aircraft to speed up or slow down, avoiding (near) collisions and costly holding patterns. Like other tasks in safety-critical domains, domain expertise is plentiful[10] and cumulative learning is desirable because it facilitates piecemeal introduction of functionality which minimizes disruptions to the complex and sensitive workflows of ATC operators.

2 Expert Knowledge Extraction & Representation

Creating a teaching curriculum can benefit from expert knowledge of (a) the task domain, (b) the learning system itself, and (c) general teaching principles. We assume that a prospective AI teacher knows what their AI system is (in)capable of (aside from cumulative learning) and what resources and methods are available for training/teaching. Learning (and teaching) from scratch and without guidance may be feasible (and preferable) for simple tasks, but the more complex the task, the more benefit can be derived from knowledge that can be transferred to

[7] The ADDIE model for instructional design consists of (1) **a**nalysis of the learner, learning goals, and teaching constraints, (2) **d**esign of the lesson plan or curriculum, which involves subject matter / task analysis, (3) **d**evelopment or assembly of the actual training materials, (4) **i**mplementation of the instruction with the learner (i.e. the actual teaching/training/learning), and (5) **e**valuation of learning outcomes.

[8] Due to space limitations we only describe a highly simplified version of arrival control here. A more elaborate version can be found in our tech report: http://www.ru.is/faculty/thorisson/RUTR18001_ArrivalControl.pdf

[9] Isavia is Iceland's aviation authority, managing air traffic in an area measuring 5.4 million square kilometers.

[10] A lot has even been written on task analysis for ATC (cf. https://www.eurocontrol.int/articles/atco-task-analysis), but we still need a new method for designing curricula for non-human cumulative learners.

the learner or otherwise used to inform the teaching process. Luckily, many of the tasks we want our AI/AGI systems to automate are currently being performed by humans with a great deal of domain expertise. Here we describe a method for knowledge extraction from a domain expert that results in a description of a high-level task that can inform the construction of teaching curricula.

The process begins with a common practice from requirements engineering for software development, where the goal is to produce a *"scenario"* ("user story", "use case") that describes, at a fairly high level, how a certain chunk of functionality (part of the task) is to be carried out. The interviewer (i.e. the AI teacher) starts by asking the expert to describe what they do when carrying out the job, while taking care to note each *"action"* that is taken. The concept of "action" is taken very broadly and incorporates for instance: acting in the environment, predicting outcomes, obtaining particular information, making (internal) decisions, updating current knowledge, etc.

As the scenario unfolds, the teacher should make note of each action, mark it with a unique identifier and put it in a dependency graph. It is often the case that higher-level actions (e.g. "instruct pilot to slow down") consist of multiple lower-level actions (e.g. "determine optimal aircraft speed", "connect to aircraft" and "send the message") or that one action relies on inputs from previous actions (e.g. you can't send a message without knowing what should be in it). Such dependencies should be noted (see Fig. 2).

It is likely that the domain expert does not succeed immediately in describing a scenario where all of their actions and their dependencies are explicitly mentioned. The AI teacher should check that none of the actions in the scenario can be usefully broken down further and that there are no holes in the story (missing implicit or unmentioned actions or decisions).

If an action with no dependencies can be usefully decomposed further an (inverse) laddering technique can be used where the expert is asked "how is this done?" and "what steps are involved?". The usefulness of further decompositions should be judged by the AI teacher based on their assumed knowledge of what can be (easily) learned by their AI.

If the expert doesn't know explicitly how a certain action is done it can help to have them perform the task while the teacher asks "what are you doing now?", "what are you paying attention to?" and "why?". If this is not possible—e.g. because the job is high-pressure and safety-critical and the expert should not be distracted—it may instead be possible to observe a colleague and discuss what they are doing.

If a composite action is not fully determined by sub-actions the teacher should point this out to the expert, and for each input/output ask where the data comes from / what it's used for, until the missing action(s) are found.

If a dependency A does not directly contribute to an action C a laddering technique should be used to ask "why do you do A here?" to elicit an intermediate dependency B. For instance, action A to "divide distance by velocity" does not seem to contribute directly to action C to "prevent equal arrival times", but rather to intermediate action B "estimate arrival times".

After the scenarios have been formed, we need to fill in the details of the actions. This process strongly resembles the Task analysis in CommonKADS [18] and functional requirements analysis in software engineering. The goal here is to describe, in as much detail as possible, all actions that are involved in carrying out the main task. For each action, this involves answering:

1. What is the **input**? What groups of variables / information can or must be taken into account, and what are their possible and simplest values?
2. What is the **output** or result? This can be anything, ranging from e.g. "a message to pilot X to move up/down by Y amount at time Z" to "preparation/prediction of the information for another action".
3. By what **method** do we transform input to output? Can be a straightforward series of steps/calculations, or vague descriptions of intuitive processes.
4. How can the action be **evaluated**? What variables are being optimized? What is their relative importance?

Ideally, no actions should be left implicit. There can be some redundancy due to describing actions at both high and low levels (e.g. one action may be "tell pilot what to do, based on all data", which may involve other actions like "decide which pilot to talk to, based on closeness to airport", "predict closeness to airport, based on weather", etc.).

3 Task Decomposition

The extracted high-level task needs to be decomposed so that components can be cumulatively learned and introduced piecemeal into the workflow. We use three complementary dimensions of decomposition:

Task-based decomposition (or action-based decomposition) identifies all subtasks/actions (including commands, decisions, classifications, predictions, judgments, etc.) that are part of the task, at a sufficiently low level. Lower-level actions are grouped together into higher-level ones to form a hierarchy, where a low-level action control may be (re)used by multiple higher-level actions.

Feature-based decomposition (or situation-based decomposition) in a directly-learned task (or action) attempts to identify (ideally independent) subgroups of features/variables that could be learned separately. For instance, in the "predict arrival time" action for an aircraft, we may have features for wind and precipitation, and we plan to train the system first on "no wind, no precipitation", then on "various wind conditions, no precipitation" and "no wind, various precipitation conditions", and finally on "everything combined". This is expected to lead to faster (curriculum) learning of "everything combined" than if we had started with that from the beginning. Furthermore, by allowing us to "skip" tricky situations, they no longer hold back the introduction of (partial) automation into the workflow; the system could still automate the majority of simpler cases, while warning or deferring to a human operator in trickier ones that have not been adequately learned yet.

Functionality-based decomposition is a decomposition based on the functionality that is to be introduced into the workflow, which tends to be based mostly on action-based decompositions and somewhat on feature-based ones. To create and introduce functionality, it is not sufficient that the AI system has (partially) learned the relevant tasks, it is also necessary to integrate such functionality into the larger workflow (e.g. adding certain GUI elements to the workers' software). In addition to being guided by other decompositions, which determine what functionality might be available, this is also guided by the actual workflow and identifying opportunities / situations where automation is most desired (analysis of these requirements is beyond the scope of this paper).

Based on the elicited actions in Sec. 2 we make a graphical representation of the task / action hierarchy (see Fig. 2 for an example from the arrival control task in Sec. 5). A feature-based decomposition of each action can be made based on their inputs and outputs. The graph should indicate which actions use the same features through connections or color coding. Functionality-based decompositions can be made based on the requirements of the client/user for whom the AI system is built, but will often correspond to elicited scenarios, or consist of an action with all of its sub-actions. However, in some cases the client may indicate that support for certain features/situations is not immediately crucial and feature-based segments can be maintained.

4 Curriculum Construction

A decomposition in these terms can serve as the basis for the construction of a teaching curriculum for cumulative learners.

The main philosophy behind curriculum learning is to have learning occur in what Vygotsky called the "zone of proximal development" (ZPD) [26]: the sweet spot between challenges that are too complex or novel to handle and ones that are too easy or familiar. This concept forms the basis of teaching approaches like shaping [20,3], scaffolding [27], and part-task training [23], as well as for many concepts of intrinsic motivation or "curiosity" [21]. In all cases the ZPD informs the novel stimuli that the AI sees. From the perspective of a teacher, this is achieved by making a task smaller or simpler until it enters the ZPD, and then making it larger and more complex as the learner becomes more competent. A curriculum then consists of a "lesson plan" that prescribes an order in which to teach the simplified tasks and how to complexify them.

The exact way in which (low-level) actions are taught is going to depend on (a) the learning system, (b) available training resources and (c) the nature of the task, e.g. whether it is a reinforcement learning or supervised learning task, whether it contains a lot of sequential events, and whether it is a kind of "one-shot" task. Our decomposition can greatly inform the order in which things should be taught: Within a cut-out chunk of desired functionality, we should teach actions in a roughly bottom-up manner so that the AI system can (re)use low-level functionality it already learned when learning higher-level tasks. Furthermore, the feature-based decomposition allows us to make individual tasks

Scenario S1: Separation maintenance
The Cumulative Learner (CL) is presented with IDs, velocities, and distances of a fixed number of aircraft and needs to maintain a minimal separation time between landings (A1). First, the CL must predict the time at which each aircraft is expected to arrive at each runway (A2). Based on this information, the CL needs to detect if the arrival times of any two aircraft conflict (A3). Detected conflicts must then be resolved by telling an aircraft to speed up or slow down (by ±10% in our simplification) (A4).

Action A1: Separation maintenance
See Scenario S1.
Input: IDs α $\{0, 1, ...\}$, velocities v in m/s [1–400] and distances s in m [0–4,000,000] of a fixed number of aircraft
Output: ID + speed up/slow down 10% command, or nothing
Method: predict landing times (A2), detect conflicts (A3), resolve conflicts (A4)
Evaluation: +10 per landed aircraft, -1000 per conflict

Action A2: Arrival time prediction
Predict the time at which aircraft A will arrive at the runway.

Input: aircraft info for A (ID, velocity and distance)
Output: time t in s [0–10,000]
Method: $\frac{distance}{velocity}$
Evaluation: $\left\| t_{\text{predicted}} - t_{\text{actual}} \right\|_2$

Action A3: Conflict detection
Predict whether two aircraft A and B will have conflicting landing times.

Input: estimated landing times t
Output: conflict c yes/no
Method: $\|t_A - t_B\|_1 <$ threshold
Evaluation: $\|t_A - t_B\|_1 <$ threshold

Action A4: Conflict resolution
Resolve conflict between aircraft B and C.

Input: ID, velocity, distance and arrival time of aircraft A, B, C, D, where A is directly before B, and D directly after C
Output: ID of B or C + speed up/slow down Δ 10% command, or nothing
Method: See if the conflict can be mitigated by speeding up B, without introducing conflict with A. If not, slow down C and invoke A4 for C and D if this creates a conflict.
Evaluation: $c_{\text{after}} - c_{\text{before}}$ (where global conflict cost c is the sum of all local conflict costs for aircraft pairs)

Fig. 1. Extracted task description of (simplified) arrival control.

simpler by limiting the range of values that its inputs and/or outputs can take on, or even omitting some altogether (by setting them to a default value). Because we can expect cumulative learners to positively transfer knowledge of shared features between tasks, we recommend prioritizing teaching (simplified) tasks with features that are shared by many other actions.

5 Case Study: Arrival Control

Safety-critical domains with high time- and energy sensitivity and low error tolerance, like air traffic control and human transportation, rely on complex workflows designed to result in safe processes. The arrival control (AC) task, like most others in aviation, is based on thoroughly documented procedures for achieving high levels of quality, safety and reliability. Automation is shunned in domains like these unless it can be fully trusted and understood, and new functionality can be introduced gradually to avoid disrupting the proven workflow. Cumulative learners have an advantage here, because they (a) can gradually add more functionality to their skills without deteriorating already-known tasks, (b) be understood modularly in terms of the tasks they were taught, and (c) deal

more robustly with distributional drift in the task or novel situations [1,22], by appropriate adaptation when sufficient prior knowledge is available to them, and yielding control when it's not. Since the cumulative learning capabilities of modern AI are limited at best, humans are relied on in practically all cases. The need for automatic cumulative learning, and the rich access to domain expertise, make this domain highly suitable for testing our methodology.

The primary goal of arrival control is to ensure an optimal flow of aircraft arrivals at the airport, avoiding collisions and costly holding patterns. This is a highly complex and safety-critical task that requires understanding of weather patterns, aircraft specifications, communication issues and the delicate coordination between many pilots and ATC operators with different roles. To illustrate our task analysis methodology we present the extracted task description (Fig. 1), decomposition and curriculum for a version of arrival control that is significantly simplified due to space limitations.[11]

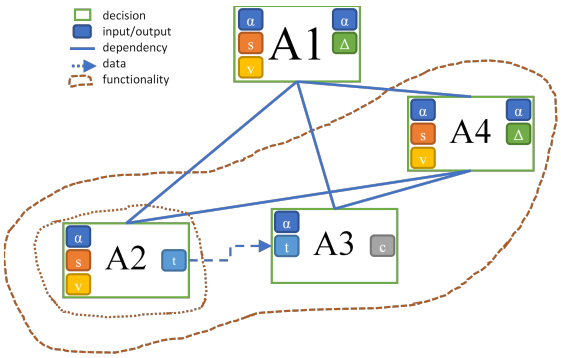

Fig. 2. Extracted action hierarchy for simplified arrival control. The relations (solid lines) between actions represent an task-based decomposition.

Decomposition As described above, action A1 makes use of A2, A3 and A4, while action A4 also makes use of the functionality of A2 and A3 (see Fig. 2). We can also see that A3 depends on data from A2. Since A2 does not have any dependencies, we can extract it as a single chunk of functionality, that can eventually be expanded into a chunk that provides the functionality of A4 (and its dependencies).

Curriculum The order in which arrival control should be taught, according to our methodology, would no doubt be A2 → A3 → A4 → A1. The reasoning is as follows: A2 doesn't depend on any other actions, and can be learned alone. A3 requires A2's output as its input, and could therefore benefit from knowledge of A2, although we could also train A3 with fake data to remove this dependency. A4 requires both A2 and A3 and cannot really be trained without them, and A1 requires all the others. The individual tasks can be further simplified (and gradually made harder again) by e.g. changing the allowable values for the input and output features. These should be modified in the same way across actions.

[11] A more elaborate version can be found in our tech report: http://www.ru.is/faculty/thorisson/RUTR18001_ArrivalControl.pdf

6 Conclusion

We have presented a task analysis methodology to inform the design of teaching curricula, when both domain expertise and cumulative learners are available. We envision this to often be the case for AGI systems, who are by definition capable of cumulative learning, and will often be used to automate complex workflows that are currently being done by human experts. Especially in safety-critical domains with complex overlapping tasks, such as air traffic control, we find that extensive knowledge and documentation of processes is typically available. Furthermore, in proven workflows that are highly sensitive to time-pressure and errors, disruptions by the abrupt wholesale introduction of monolithic automation are unacceptable, and having cumulative learners that are taught to gradually expand their functionality is highly desirable. Our presented methodology takes a step in the direction of making curriculum design more systematic, using any available domain knowledge.

Future work will be needed to compare the proposed knowledge elicitation and task analysis methods with reasonable alternatives, in terms of ease-of-use and required (time) investment for both the teacher and domain expert, as well as quality of the produced analysis. Knowledge extraction can furthermore be augmented by utilizing data from other (written) sources, and we are interested to know how different (expert or written) sources can lead to different task decompositions and how this affects subsequent curricula. A better theory is needed for constructing teaching curricula based on the presented task analysis, but also on characteristics of the learning system and available training resources. The benefits of the produced curricula should be evaluated and compared to alternate approaches like "no curriculum" (i.e. training on the full monolithic task), "alternate / random order curricula" and curricula arrived at through different curriculum construction methods and task analyses (e.g. where decomposition is done using a variety of existing automated methods).

References

1. Amodei, D., Olah, C., Steinhardt, J., Christiano, P.F., Schulman, J., Mané, D.: Concrete problems in AI safety. CoRR **abs/1606.06565** (2016)
2. Barry, A.: A hierarchical XCS for long path environments. In: Proceedings of GECCO-2001 (2001)
3. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of ICML-26 (2009)
4. Bieger, J.: Artificial Pedagogy: A Proposal. In: HLAI 2016 Doctoral Consortium. New York, NY (2016)
5. Bieger, J., Thórisson, K.R., Garrett, D.: Raising AI: Tutoring Matters. In: Proceedings of AGI-14. Quebec City, Canada (2014)
6. Bieger, J., Thórisson, K.R., Steunebrink, B.R.: The Pedagogical Pentagon: A Conceptual Framework for Artificial Pedagogy. In: Proceedings of AGI-17 (2017)
7. Branson, R.K., Rayner, G.T., Cox, J.L., Furman, J.P., King, F.J.: Interservice Procedures for Instructional Systems Development: Phase 4 and 5. Tech. rep., Florida State University (1975)

8. Burke, L.A., Hutchins, H.M.: Training transfer: An integrative literature review. Human resource development review **6**(3) (2007)
9. Hasselmo, M.E.: Avoiding catastrophic forgetting. Trends in cognitive sciences **21**(6), 407–408 (2017)
10. Hengst, B.: Hierarchical approaches. In: Reinforcement learning. Springer (2012)
11. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A.: Overcoming catastrophic forgetting in neural networks. PNAS **114**(13), 3521–3526 (2017)
12. Lake, B., Salakhutdinov, R., Gross, J., Tenenbaum, J.: One shot learning of simple visual concepts. In: Proceedings of CogSci 2011. vol. 33 (2011)
13. Looks, M.: Competent Program Evolution. Ph.D. thesis, Washington University (2006)
14. Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S., Zhang, G.: Transfer learning using computational intelligence: a survey. Knowledge-Based Systems **80**, 14–23 (2015)
15. Marcus, G.: Deep learning: A critical appraisal. CoRR **abs/1801.00631** (2018)
16. Nivel, E., Thórisson, K.R., Steunebrink, B.R., Dindo, H., Pezzulo, G., Rodriguez, M., Hernandez, C., Ognibene, D., Schmidhuber, J., Sanz, R., Helgason, H.P., Chella, A., Jonsson, G.K.: Bounded Recursive Self-Improvement. Technical RUTR-SCS13006, Reykjavik University, Reykjavik, Iceland (2013)
17. Pollock, J.L.: Defeasible reasoning and degrees of justification. Argument and Computation **1**(1), 7–22 (2010)
18. Schreiber, G.: Knowledge engineering and management: the CommonKADS methodology. MIT press (2000)
19. Silver, D.L., Yang, Q., Li, L.: Lifelong Machine Learning Systems: Beyond Learning Algorithms. In: AAAI Spring Symposium: Lifelong Machine Learning (2013)
20. Skinner, B.F.: The behavior of organisms: An experimental analysis. Appleton-Century-Crofts Inc., New York (1938)
21. Steunebrink, B.R., Koutnık, J., Thórisson, K.R., Nivel, E., Schmidhuber, J.: Resource-Bounded Machines are Motivated to be Effective, Efficient, and Curious. In: Proceedings of AGI-13. Beijing (2013)
22. Steunebrink, B.R., Thórisson, K.R., Schmidhuber, J.: Growing recursive self-improvers. In: Proceedings of AGI-16. New York, NY, USA (2016)
23. Teague, R.C., Gittelman, S.S., Park, O.c.: A review of the literature on part-task and whole-task training and context dependency. Tech. rep., ARI, US (1994)
24. Teh, Y.W., Bapst, V., Czarnecki, W.M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., Pascanu, R.: Distral: Robust Multitask Reinforcement Learning. CoRR **abs/1707.04175** (2017)
25. Thórisson, K.R., Bieger, J., Thorarensen, T., Sigurðardóttir, J.S., Steunebrink, B.R.: Why Artificial Intelligence Needs a Task Theory — And What it Might Look Like. In: Proceedings of AGI-16. New York, NY, USA (2016)
26. Vygotsky, L.S.: Interaction between Learning and Development. In: Mind in society: The development of higher psychological processes. Harvard University Press, Cambridge, MA (1978)
27. Wood, D., Bruner, J.S., Ross, G.: The role of tutoring in problem solving. Journal of child psychology and psychiatry **17**(2), 89–100 (1976)
28. Zhan, Y., Taylor, M.E.: Online transfer learning in reinforcement learning domains. CoRR **abs/1507.00436** (2015)