Position paper

Problem Solving Through Human-AI Preference-Based Cooperation

Subhabrata Dutta
TU Darmstadt

Timo Kaufman
LMU Munich, MCML

Goran Glavaš
University of Würzburg

Ivan Habernal
RU Bochum

Kristian Kersting
TU Darmstadt

Frauke Kreuter
LMU Munich, MCML

Mira Mezini
TU Darmstadt

Iryna Gurevych
TU Darmstadt

Eyke Hüllermeier*
LMU Munich, MCML

Hinrich Schütze*
LMU Munich, MCML

While there is a widespread belief that artificial general intelligence (AGI) – or even superhuman AI – is imminent, complex problems in expert domains are far from being solved. We argue that such problems require human-AI cooperation and that the current state of the art in generative AI is unable to play the role of a reliable partner due to a multitude of shortcomings, including difficulty to keep track of a complex solution artifact (e.g., a software program), limited support for versatile human preference expression and lack of adapting to human preference in an interactive setting. To address these challenges, we propose `HAI-Co`^{2}*, a novel human-AI co-construction framework. We take first steps towards a formalization of* `HAI-Co`^{2} *and discuss the difficult open research problems that it faces.*

1. Introduction

Despite the impressive advances of generative AI (Cao et al. 2025), especially for natural language (large language models), vision (vision language models) and code (code models), recent investigations have pointed out a lack of competence in dealing with complex generation problems that require intricate planning (Kambhampati et al. 2024) and task adherence while keeping track of multiple constraints (Xie et al. 2024a). A broad class of such complex problems, especially complex problems in expert domains, requires active human participation. Therefore, although the recent focus in generative AI has mostly been on complete automation (Hong et al. 2024; Brown et al. 2020), we believe that human-AI cooperation is a more promising approach.

* Shared senior authorship

To address complex problems of this kind, we propose **H**uman-**AI Co-Co**nstruction (`HAI-Co`^{2}), a novel framework for human-AI cooperative problem solving that builds on preference-based learning and search methodology and relies on natural language to facilitate interaction.

We acknowledge that many substrands of the NLP, ML, AI and HCI communities have noticed and addressed problems that are closely related to the problem `HAI-Co`^{2} is addressing. Moreover, many of the components of `HAI-Co`^{2} are also components of this prior work. The differentiator of our proposal lies (i) in the specifics of the problem statement, (ii) in the specifics of the holistic framework that we call `HAI-Co`^{2} and (iii) in a clear articulation of the research challenges and unmet needs that follow from the way we define problem and framework. We will lay out the research challenges in Section 4 and review related work in detail in Section 5. To set the context for the reader, we highlight here the most important points that make our work novel.

First, we address problems in **expert domains that require complex solutions or artifacts**.

- We posit that an explicit and persistent representation of the solution space is crucial for systematic solution construction and that this solution space must be equipped to represent the complexities of expert domains, including through an abstraction hierarchy.
- We propose search as the paradigm for constructing solutions: construction proceeds from an initial draft to a satisfactory solution step by step where each step consists of a search for an appropriate extension or modification of the current artifact.

Second, `HAI-Co`^{2} is designed for **complex problem solving by a team of an expert and an AI agent**.

- In contrast to many other approaches, `HAI-Co`^{2} is set up for a cooperation of expert and agent as equal partners, each contributing their complementary strengths.
- In complex domains, the exact goal of the cooperation is often underspecified in the beginning. This means that the cooperation is not only about constructing the solution, but also about constructing the precise objective of the solution. This raises ethical concerns (e.g., influencable reward functions) that need to be addressed.
- We posit natural language as the primary medium of communication. One of the challenges in `HAI-Co`^{2} is that the agent has to learn effectively from implicit human feedback in natural language, but also from other signals in the complex co-construction environment, e.g., multimodal information and human edits.
- Finally, a new evaluation methodology needs to be developed for `HAI-Co`^{2} due to the difficulty of assessing the quality of solutions in complex expert domains and due to the open-endedness and non-uniqueness of solutions to complex problems in expert domains.

In this article, we first give a general introduction to `HAI-Co`^{2} (Section 2), including its ethical challenges, and take first steps towards a formalization (Section 3). We outline open research challenges posed by `HAI-Co`^{2} in Section 4. Section 5 discusses related work. Section 6 concludes the paper.

2. `HAI-Co`^{2}: Human-AI co-construction through preference-based search

In this article, we propose **H**uman-**AI Co-Co**nstruction (`HAI-Co`^{2}), a novel framework for human-AI cooperative problem solving. The four defining characteristics of `HAI-Co`^{2} are (i) solution of complex problems in expert domains that require active human participation, (ii) co-construction of the solution by human and AI agent, (iii) co-construction of the objective by human and AI agent by means of preference learning and (iv) the use of natural language as the main communication medium, which makes it possible for human and AI agent to be equal partners, with complementary strengths, in the co-construction. We now describe these four characteristics in more detail.

First, we target applications in **expert domains** where the task is to construct a solution to a **complex problem**. Since expert domains are our focus, we use "expert" and "human" interchangeably in this article. Examples of complex problems in expert domains include writing a computer program in software engineering; constructing a machine learning pipeline in automated machine learning (AutoML); writing a related work section in scientific research; and developing a formalization of a problem described in natural language in mathematics.

Second, we conceive of problem solving as **co-construction of the solution to the complex problem by a human and an AI agent** or – more generally – by a team of humans and agents. As detailed in Section 3, the problem solving process is formalized as a process of *systematic search* in a *construction space* $\mathcal{X}$ of *candidate solutions* on several hierarchical levels of abstraction. In this co-constructive process, candidate solutions are modified step by step until a sufficiently good solution has been found.

We draw inspiration from our understanding of how humans collectively devise solutions to complex problems. Humans often tackle such problems by iteratively co-constructing a solution step by step, revising and refining draft solutions while transitioning between different levels of abstraction and exchanging information about preferences and potential improvements in natural language. Our primary motivation for `HAI-Co`^{2} comes from natural language processing and computer science; see Section 5 for a brief discussion of related fields that have conducted extensive research on human-human cooperation on solving tasks.

Human-human cooperation would make less sense as a promising template for human-AI cooperation if current AI systems could solve complex expert-domain problems on their own. However, current AI capabilities are limited for complex expert-domain problems, e.g., due to insufficient knowledge and reasoning capabilities, bias and lack of trustworthiness. Scaling generative AI systems, particularly language models, has demonstrated improved performance across a range of tasks, e.g., math word problems and commonsense reasoning. However, even the most powerful LLMs show a lack of robustness under different semantic perturbations that would not have fooled an otherwise robust reasoner (Li et al. 2024b). While it is unwise to rule out future improvements, their current limitations call for interventions beyond scaling. Thus, in order to be able to effectively solve complex expert-domain problems, we believe it is necessary for the AI agent to work closely with human experts.

Third, cooperative problem solving often involves the **co-construction of the objective** – or **objective co-construction** – alongside the co-construction of the solution itself. As part of the co-construction process, the requirements for the solution are often changed and refined as the collaborators understand the details of the complex problem better and revise their initial assumptions. As we will see in Section 3, we formalize this process of objective co-construction through interactive preference learning: we define a utility function over the construction space that reflects preferences of user and agent, i.e., which artifacts are better and which are worse candidate solutions. Thus, the objective is encoded in a preference model. Or, stated differently, the objective is first described informally on a general level – e.g., write a computer program performing a particular task – and then formalized in terms of the preference model.

One inspiration for making objective co-construction part of our framework for solution co-construction comes from the field of multi-criteria decision aiding (MCDA), a branch of operations research (Roy 2000, 1996). "MCDA underlines the 'aiding' in a process involving the DMs [decision makers] in the co-construction of their preferences ... It assumes that preferences of the DM with respect to considered alternatives do not pre-exist in the DM's mind." (Hüllermeier and Slowinski 2024a) MCDA (see also Hüllermeier and Slowinski (2024b)) makes assumptions that are close to objective co-construction; in particular, the user's objective is only partly determined in the beginning and further developed in the course of the decision aiding process.

It is important to note that the two types of co-construction in `HAI-Co`^{2} – solution co-construction and objective co-construction – are quite different. Solutions are artifacts whereas objectives are encoded as preference models. The solution is co-constructed through search in the construction space whereas the objective is co-constructed through preference learning. Still, at the highest level, both solutions and objectives are the result of a cooperation of human and agent, with each being a contributor even though their contributions may differ in nature and scope.

Fourth, in `HAI-Co`^{2}, **natural language is the main communication medium**. This makes it possible for human and AI agent to be **equal partners**, with complementary strengths, in the co-construction process. In our view, language-based communication is a key enabler of an equal partnership. The ability to express oneself fully and on the same level is a prerequisite for making equal contributions to problem solving. Previous communication technology was a limiting factor in this regard: only with the advent of LLMs do we now have AI agents available that comprehend and generate natural language at a human level of capability. Such human-level capabilities are required for the complex communication needs that occur during cooperation on complex expert-domain problems.

There are certainly problem-solving scenarios in which the human manages the process and the agent's role is reduced to handling low-level tasks (e.g., "tool" tasks like internet search or copy-editing) – or, conversely, scenarios where the human's role is reduced to providing input when prompted by the agent (e.g., in active learning). In contrast, the type of human-AI cooperation scenario we are interested in is one in which the two partners are equal.

Equality here does not mean identical roles. On the contrary, the roles are complementary: each partner has skills or knowledge that the other lacks. An example for complementarity is that the human may understand the context better in which the complex problem arises (e.g., the requirements and personalized preferences of human stakeholders) whereas the AI agent may be able to more efficiently access vast information resources and make more effective use of tools such as compilers and unit testing. Just as human-human cooperation excels at problem solving if the collaborators complement each other, so is human-AI cooperation most beneficial if each partner can contribute their unique strengths. This aligns with recent evidence showing that human-AI cooperation alone does not guarantee superior performance; effective integration and task-appropriate division of labor are key to realizing the benefits of cooperation (Vaccaro, Almaatouq, and Malone 2024).

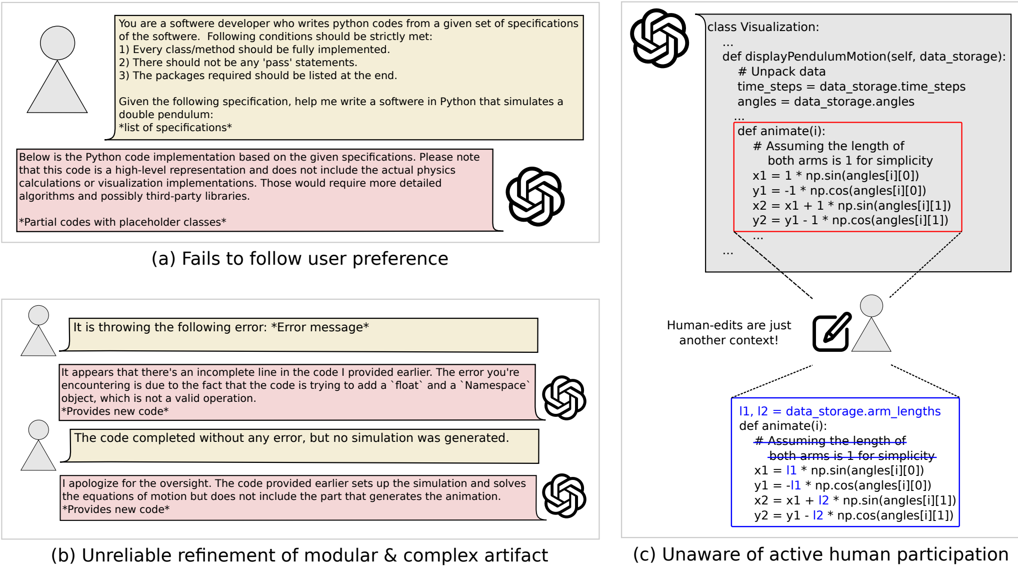

Figure 1
Existing generative AI lacks proficiency in key aspects of co-construction of solutions to complex problems. We give a code synthesis example. (a) GPT-4 Turbo fails to follow preferences explicitly stated by the human expert. (b) Due to the lack of a persistent object representation, a modification request targeted toward one feature of the desired solution leads to the unwanted (and erroneous) modification of another feature. (c) The human expert modifies the generated code directly to remove inline assumptions and introduces general variables; such active participation is not demarcated and recorded by the AI and there is no facility to extract the implicit preferences and follow them elsewhere.

Scenarios that `HAI-Co`^{2} **does not address.** To more clearly delineate which class of scenarios we want `HAI-Co`^{2} to address, we now give some examples where `HAI-Co`^{2} is not a good fit for solving problems. (i) If the problem is simple (as opposed to a complex expert-domain problem), then AI models probably can solve it autonomously. (ii) Even many complex problems may be solvable autonomously by AI (now or in the future) if a full specification is available that can be checked automatically. Games like Go and chess are such examples in which objective co-construction is not necessary, i.e., a full specification of the objective can be easily obtained. (iii) Another class of scenarios may be best addressed by an expert managing the solution process and utilizing the agent for solving specific subtasks. Somebody working through their email inbox after coming back from vacation may want to closely oversee this process (e.g., not letting the agent send email autonomously), but may be happy to give specific subtasks to the agent (e.g., ask correspondents that requested reviews to re-request them).

In contrast to scenarios in which the cooperation is dominated by either the agent (ii) or the expert (iii), `HAI-Co`^{2} is intended for equal-partner scenarios in which the two work together as partners leveraging their complementary strengths.

Is the current state of generative AI enough? As a running example, we will use a code generation example that illustrates some of the bottlenecks of GPT-4 Turbo, an update[1] of the original GPT-4 model (OpenAI 2023). In Figure 1 (a), GPT-4 Turbo ignores

1 We use `gpt-4-1106-preview`.

the human expert's explicit instructions to generate a complete Python code with the required module specifications, echoing Xie et al. (2024a)'s observation that current language-based AI agents lack task adherence. After repeated prompting with partial code snippets, the process produces a complete – albeit faulty – code. This limitation is even more serious when more varied and realistic expressions of human preferences are taken into account – for the human expert to contribute productively, one must allow preferences expressed via explicit instructions, binary choice, ranking, etc. Current generative AI solutions do not facilitate such multi-modal preference incorporation. Figure 1 (b) shows unreliable debugging attempts. Specifically, the LLM performs unrelated (and faulty) edits to address a bug and even introduces new errors. This demonstrates that existing LLMs struggle with handling complex, modular software code (Jiang et al. 2024). The common practice is that the human (as a knowledgeable expert who keeps track of overall context) identifies faulty output and repeatedly prompts the model to guide it to the correct generation – this is implicitly adopting a co-construction paradigm. However, Figure 1 (c) shows that current modes of human-AI interaction cannot unlock the full potential of co-construction – direct modification of the co-constructed candidate solution by the human expert does not bear any special significance to the LLM, and it treats it as just another context. There is no explicit mechanism for the AI to learn implicit preferences expressed by the human through active participation. On the other hand, forcing humans to take a passive role and making them review AI-generated code affects productivity negatively (Xu, Vasilescu, and Neubig 2022; Bird et al. 2023). Meta-analyses by Simkute et al. (2024), with a focus on coding assistants, identify four major axes of AI-mediated productivity loss: shift of human roles from production to evaluation, unhelpful workflow restructuring, task interruptions, and, easy tasks becoming easier while hard tasks become harder. Recent advances in interactive coding have sought to address the first challenge by providing users with evaluation tools — AI-generated tests or static analysis-based — to decrease the cognitive load of reviewing AI-generated code. A key takeaway from these advancements is the need to redefine the human is supposed to do, what the AI needs to do, and how they are going to complement their respective expertise and limitations.

While these examples are focused on code synthesis, there is evidence of similar shortcomings in other domains such as planning (Kambhampati et al. 2024), radiology (Lecler, Duron, and Soyer 2023) and clinical decision making (Hager et al. 2024). Carroll et al. (2019), for example, demonstrate the necessity of incorporating explicit "human awareness" in a version of the collaborative game Overcooked, providing evidence that agents fail to coordinate with human subjects without such awareness. Code synthesis in particular – and the experience from day-to-day use of generative AI for solving complex problems in general – points toward co-construction as a naturally evolving problem solving paradigm where the human expert tries to search for the optimal solution by interacting with the AI. However, the current state of generative AI hinders its role as a reliable partner in successful co-construction. This is because the "one-directional" interaction between human and AI typical of how AI agents are used today often fails to steer the co-construction towards a solution that satisfies the user's constraints.

In summary, prior work has laid out the inherent shortcomings of present-day generative AI for complex problem solving (see also Section 5 for a much more detailed discussion of prior work). This motivates our alternative emphasis on human-AI co-construction as a paradigm for solving complex problems in expert domains.

Ethical considerations. One aspect of `HAI-Co`^{2} carries considerable risk: the co-construction of the objective. This opens the door to manipulation by the agent. For example, LLMs that fail to solve a goal have been observed to redefine it to be easier (Anthropic 2025a).

Influenceable reward functions have been studied by Carroll et al. (2024). They write: "We show that despite its convenience, the static-preference assumption may undermine the soundness of existing alignment techniques, leading them to implicitly reward AI systems for influencing user preferences in ways users may not truly want." and: "... suggesting that a straightforward solution to the problems of changing preferences may not exist."

Even if there is no general solution to the problem of influenceable reward functions, it is possible to build in safeguards in the context of `HAI-Co`^{2}. Specifically, if there is a conflict between user preferences and AI preferences, then we can mandate that user preferences prevail. This can be implemented by a "monitor" agent – a secondary agent that cannot be manipulated by the primary agent and is responsible for alerting the expert to objective changes that were not clearly communicated. Alternatively, we can employ the methodology of alignment to discourage unwanted manipulation of the objective by the agent. For example, we can devise a set of rules that should govern objective co-construction (e.g., "a change to the objective must be clearly communicated to the human"), create a synthetic dataset that embodies the rules and then train the agent on this dataset using supervised finetuning or reinforcement learning.

More generally, we believe that a paradigm of close cooperation is a promising approach to addressing many of the hard ethical problems that AI faces. If the agent takes an initial problem statement, goes off and comes back with a complete solution, then that means that the human cannot make any course corrections. This is true both for initial decisions (if she were part of the co-construction process, the human may realize that some initial decisions were based on wrong assumptions and correct them) and for decisions made by the agent (in which the human is not involved in autonomous problem solving and therefore cannot influence). Our co-construction paradigm ensures consistent human participation in shaping the solution as it is being constructed. Similarly, if the human is involved in the step-by-step co-construction of the solution, then she will have a good understanding of its inner workings and the motivation for its parts; thus, cooperation can be effective in bringing about some measure of explainability and (by extension) transparency with respect to ethical concerns.

In summary, making the objective influenceable by the agent is a risk. But there are promising solutions for addressing it (e.g., adding a monitoring agent for supervision). In addition, the close cooperation of human and agent in `HAI-Co`^{2}– which in contrast to an autonomous approach ensures that the human is involved in all aspects of the co-construction process – is a form of artificial intelligence that addresses some ethics problems of AI by design, e.g., it supports transparency with respect to ethical concerns.

Our contribution. In this article, we take steps towards formalizing co-constructive problem solving and thereby aim to address important limitations of current generative AI models. In contrast to approaches in which problems are solved autonomously by AI or – conversely – in which AI is an assistant without autonomy that is limited to executing tasks clearly defined by the human, we view co-constructive problem solving as a process that involves the two parties as equal partners, each contributing complementary strengths. Concretely, we present `HAI-Co`^{2}, a conceptual framework that facilitates human-AI co-construction. `HAI-Co`^{2} introduces multiple levels of abstraction to the candidate solution, providing a seamless interface for the human expert

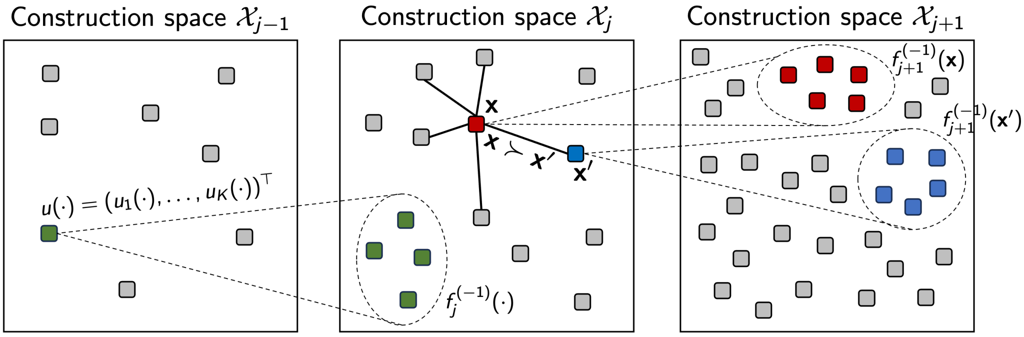

Figure 2
Illustration of the hierarchy of construction spaces in HAI-Co^2. Each point $\boldsymbol{x}$ symbolizes a candidate solution (on a certain level of abstraction), e.g., a software program. The topology of the space is specified by a suitable neighborhood structure (as illustrated for point $\boldsymbol{x}$). Each point is associated with a latent utility u^t, possibly multi-dimensional and comprised of local utilities $u_1^t, \dots, u_K^t$, and preferential information (e.g., $\boldsymbol{x} \succ \boldsymbol{x}'$: solution $\boldsymbol{x}$ is better than $\boldsymbol{x}'$) that provides information about promising regions in the space. The relationship between the different abstraction levels is specified by the abstraction mappings f_j resp. the (inverse) refinement mappings $f_j^{(-1)}$.

and the AI agent to modify and keep track of the complex, modular, co-constructed candidate solution. HAI-Co^2 supports objective co-construction – i.e., both the solution and the objective are co-constructed by human and agent – by allowing multi-modal preference input from the human expert, with natural language as the central mediator to capture information-rich guidance signals, along with other forms of active expert participation, such as categorical choice-based preference. Solution co-construction in HAI-Co^2 is conceived as search where the candidate solution (represented on multiple levels of abstraction) is iteratively revised to maximize the its utility, modeled by the preference model. While several components of HAI-Co^2 have been explored in prior research independently across different domains (see Section 5), we are the first to bring them together under a unified conceptual umbrella and to show that, enabled by current generative AI, they have the potential to address the major challenges in solving complex expert-domain problems.

3. Towards a formalization of HAI-Co^2

In this section, we propose HAI-Co^2, a framework for cooperative problem solving. Broadly speaking, HAI-Co^2 is meant to formalize an interactive problem solving scenario, in which a human expert seeks to (co-)construct a *solution* – such as a computer program – with the help of an AI agent. The problem solving process is conceived as a process of *systematic search* in a space $\mathcal{X}$ of *candidate solutions*, i.e., as a co-constructive process, in which candidate solutions are modified or extended step by step until a (sufficiently) good solution has been found. Therefore, we also refer to the search space $\mathcal{X}$ as the *construction space*. The construction space, its hierarchical organization and its topology (or neighborhood structure) are depicted in Figure 2.

Actions taken by the AI agent during the search (e.g., adapting a candidate solution or asking the expert a question regarding where to move next) depend on its *informational state* $\mathcal{I}$, which comprises its experience so far, e.g., about the expert's preferences, any relevant information about the current context, the solutions considered so far and the best solution constructed so far. Formally, the behavior of the AI agent can be determined by a *policy* π that maps informational states to actions.

As a running example, we provide the code generation use case in Figure 3 as an illustration of how different formal aspects of `HAI-Co`^{2} can be implemented (see Appendix 1 for more details). A second use case – in the expert domain of related work section generation – can be found in Appendix 2. The choice of code generation as a use case is motivated by recent advancements in automation in this domain and the subsequent development of knowledge regarding the limitations of these advancements. On the other hand, related work generation is a natural use case for the scientific community, irrespective of the individual researcher's particular domain.

`HAI-Co`^{2}, as we envision, is not limited to these two choices of use cases. For example, in case of travel planning problems (Xie et al. 2024b), one can conceptualize an abstract hierarchy similar to multi-agent planning systems (Li et al. 2025) and a plan-space search (Lee et al. 2025) guided by human preferences. Objective co-construction under `HAI-Co`^{2} allows the AI and the human to define the utility criteria of a plan together; e.g., an AI agent with access to large number of online reviews might inform the human about certain emergent patterns in hotel booking refusals, whereas an expert travel agent might guide the AI to book trips via certain transports. Similar analogies can be drawn for tasks pertaining to medical decision making. Recent studies (Tikhomirov et al. 2024) have indicated inherent differences in the clinical reasoning process adopted by humans and AI systems. This further strengthens our argument for the goal of AI to become complementary to human, not mimicking humans. One can formalize the solution space as space of concrete diagnosis, with the abstractions highlighting different aspects of the diagnosis that construct a hierarchical plan (Fdez-Olivares et al. 2011).

3.1 Construction space and abstraction hierarchy

The construction space will typically be large, most often even (countably) infinite. For example, the construction space may consist of all computer programs in a specific programming language. Spaces of this kind cannot be specified in an explicit way. Instead, they will be defined implicitly and may even be adapted or designed on-the-fly in the course of the problem solving process. In this regard, the *formal representation* of candidate solutions is of major importance and will strongly influence the efficacy and efficiency of `HAI-Co`^{2}. Moreover, it is also clear that the representation of solutions will not be universal but rather specific to the expert domain. For example, a computer program will not be represented in the same way as a machine learning pipeline or data science workflow. It should be noted that we do not make any assumption of *completeness* for candidate solutions: at any stage of the search, a candidate solution $\boldsymbol{x} \in \mathcal{X}$ can be partial or incomplete (i.e., an incomplete codebase, an incomplete ML pipeline, etc.).

During problem solving, it is often useful to look at (candidate) solutions on multiple levels of abstraction. In many cases, for example, a rough draft of the solution is found in a first phase of the process, and this draft is then worked out in more detail in a second phase. More generally, one can imagine a search process that switches back and forth between different levels of abstraction whenever appropriate. Therefore, we assume that the construction space $\mathcal{X}$ is equipped with a hierarchy of abstraction levels.

Formally, this can be modeled by a sequence $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_J$ of spaces, where $\mathcal{X}_j$ is a refinement of $\mathcal{X}_{j-1}$ – or, vice versa, $\mathcal{X}_{j-1}$ an abstraction of $\mathcal{X}_j$. We describe the abstraction

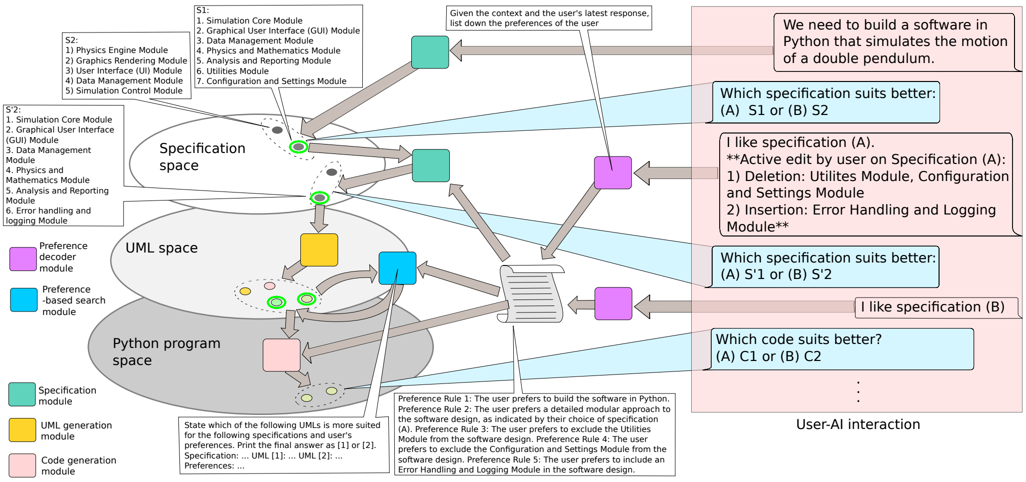

Figure 3
Running example. An example instantiation of HAI-Co[2] for the problem of building a double pendulum simulation. A solution is co-constructed through human-AI cooperation as follows. The interaction between the human and the AI is shown in the red box on the right, top to bottom. We define the co-construction space on three levels of hierarchy: Specification space, UML space, and Python program space. The user starts by specifying the (potentially incomplete) problem to solve. The Specification module (green rectangle) generates a pair of candidate specifications of the software to build (S1 and S2) and presents them to the user. The user expresses their preference in two manners: i) they choose one of the candidate solutions (S1) as better than the other, and ii) provide partial edits to the specification directly. The Preference decoder module (purple rectangle) extracts preference rules from the interaction context. Based on the decoded preferences, the Specification module generates a new pair of candidate specifications (only one shown for space reasons), from which the user chooses one. The UML generation module serves as a generator of refinements from specification space to UML space and generates a set of four UMLs from the specification selected. The Preference-based search module then runs a tournament-based search among these UML candidates: a pair of UMLs are compared against the specification and the decoded user preferences and one is chosen. Two finalist UMLs from the tournament are then used by the Code generation module (pink rectangle) to generate two candidate Python programs. These programs are presented to the user again for their feedback.

process from $\mathcal{X}_j$ to $\mathcal{X}_{j-1}$ as a surjection $f_j : \mathcal{X}_j \to \mathcal{X}_{j-1}$ such that $\boldsymbol{x}' = f_j(\boldsymbol{x}) \in \mathcal{X}_{j-1}$; that is, $\boldsymbol{x}'$ is the abstraction of $\boldsymbol{x}$ on the abstraction level modeled by $\mathcal{X}_{j-1}$. We denote by $f_j^{(-1)}(\boldsymbol{x}') = \{\boldsymbol{x} \in \mathcal{X}_j \mid f_j(\boldsymbol{x}) = \boldsymbol{x}'\}$ the set of all refinements of $\boldsymbol{x}' \in \mathcal{X}_{j-1}$ on abstraction level $\mathcal{X}_j$. Note that refinements are not unique, which is why a transition from $\mathcal{X}_{j-1}$ to $\mathcal{X}_j$ may come with a certain arbitrariness.

In our running example on code generation presented in Appendix 1, we implement the construction space on three levels of abstraction, considering a Python program as a refinement of a UML diagram, which in turn is a refinement of a natural language specification. The refinement maps are implemented by suitably prompted LLMs that take a representation in a higher abstraction space as input and produce a solution representation in the lower level of abstraction as output. The one-to-many nature of the refinement is reflected via stochastic sampling in the LLM inference: multiple UML representations are sampled stochastically from one natural language specification. In case of related work section generation, these abstractions can be conceptualized as list of papers, semantic graph/outline of related work, and the actual related work section (the final artifact/solution, see Appendix 2). Note that in either case, the abstraction hierarchy can facilitate better utilization of different expertise of the human and the AI in the co-construction process; for example, modern code-language models are great at stylizing a piece of code (variable naming, commenting, etc.), while human understanding of requirement engineering is needed to reach the optimal specification document. Similarly, an LLM paired with a search engine can possibly result in an efficient paper-finder agent in the use case of related work generation, while the human scientist defines the utility function that evaluates a candidate related work section.

3.2 Latent utility

We assume that the construction space is equipped with a latent *utility function* reflecting the preferences of the expert, i.e., the quality of solutions as perceived by the expert. But the utility function also reflects input the AI agent may have given on which solutions are to be preferred. Thus, the utility function embodies the current state of the co-construction of the objective that we introduced in Section 2. This also means that the utility function is not static, but changes as expert and agent refine and change their goals during the co-construction process.

What exactly do we mean by "quality" of the solution? In general, "quality" may refer to various dimensions or criteria, and different objectives might be pursued at the same time; we formalize this with a *multidimensional* utility function $u^t(\boldsymbol{x}) = (u_1^t(\boldsymbol{x}), \ldots, u_K^t(\boldsymbol{x}))^\top$ comprised of local utility functions u_i^t where the time index t indicates the temporal dependency and dynamic nature of the utility function. For example, a computer program could be rated by average runtime or memory consumption. The local utility functions can be combined into a scalar utility function $U^t : \mathcal{X} \to \mathbb{R}$ via a suitable aggregation operator.

Various factors influencing the quality of candidate solutions can be distinguished, notably hard and soft constraints. *Hard constraints* refer to (functional) properties that qualify a candidate as a valid solution. For example, a computer program should properly compile and not contain any syntax errors. Even if invalid solutions should normally be considered useless, the abstract notion of utility is flexible enough to distinguish different levels of invalidity. For example, a non-executable computer program may still have a non-zero utility if the error can easily be fixed by the expert. In any case, hard constraints will normally not identify a solution uniquely. For example, there are many computer programs that are functionally equivalent in the sense of having the same input-output behavior. *Soft constraints* refer to criteria that make a solution more or less desirable such as the length of a computer program and its time and memory consumption.

In general, the utility (be it in the form of the multidimensional utility function u^t or the scalar utility function U^t) is not known to the expert and the AI agent, nor are they explicitly aware of it. Rather, the utility is latent and underlies the expert's preference feedback, potentially taking into account the AI agent's input on what makes a good solution. Based on this, the AI can learn an approximation $\hat{U}^t$. The AI's goal is then to construct a solution $\boldsymbol{x}^*$ that maximizes $\hat{U}^t$, or which is at least close to the maximizer, while simultaneously improving the approximation quality of $\hat{U}^t$. The utility U^t also induces utilities on higher levels of abstraction. For example, one way to "lift" a utility function from level $\mathcal{X}_j$ to the more abstract level $\mathcal{X}_{j-1}$ is via aggregation: $U^t(\boldsymbol{x}') = \alpha(\{U^t(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \mathcal{X}_j, f_j(\boldsymbol{x}) = \boldsymbol{x}'\})$, where α is an appropriate aggregation function (Grabisch et al. 2009).

In our running example, we represent the utilities as user preferences in natural language, extracted from the interaction where both the user and the AI agent can choose between presented options, provide explicit instructions, or actively edit parts of the solution.

3.3 Interaction and preference-based search

Search through the construction space is guided by an underlying *search strategy* – in principle, any heuristic search method (properly balancing exploration of the construction space and exploitation of acquired knowledge) may serve as a point of departure. However, in HAI-Co^2, the search is also interactive and largely controlled by the human-AI cooperation.

To guide the search, human and AI can communicate via natural language; e.g., the AI agent may ask the expert for feedback or explicit advice. Alternatively, the expert may actively intervene, for example by critiquing or modifying a candidate solution. A third type of interaction, particularly important in the context of HAI-Co^2, is driven through *preferential feedback*: By informing the AI agent about the quality of candidate solutions, the expert provides hints at presumably more promising (and, likewise, less promising) regions of the construction space, and hence suggests promising "search directions" to the AI agent. To give an example, the expert can compare two competing candidate solutions with each other (e.g., whether a modification has improved a solution or made it worse) and provide this feedback (in natural language) to the AI agent for the next iteration. The AI agent utilizes the feedback to improve its approximation $\hat{U}^t$ of the latent utility function, which is an important element of its informational state.

In our running example, for instance, we implement a preference-based search strategy that identifies promising solutions via a tournament of pairwise comparisons. Besides, we realize a search policy that refines an existing solution guided by the expert's preferences (see Appendix 1 for details). Additionally, we present a conceptual application of HAI-Co^2 to another expert domain in Appendix 2: the task of related work generation (Hu and Wan 2014; Nishimura et al. 2024).

The way in which the AI agent and the human expert cooperate with each other is defined in the form of a *protocol*. Among other things, the protocol clarifies the types of queries and responses on the two sides (AI agent and human expert) and the (preference) feedback that can be given by the expert.

In summary, the specification of a concrete *instantiation of* HAI-Co^2 includes the following elements:

– (Hierarchical) representation of candidate solutions (domain-specific)
– Structure of construction space $\mathcal{X}$, refinement/abstraction mappings, neighborhood structure
– Search operators (for modification of candidate solutions, refinement, abstraction, etc.) and strategy
– Natural language methods and protocol for cooperation
– Representation of informational states, the AI agent's action space and policy
– Utility as the formalization of the co-constructed objective: soft/hard constraints, preference relations/predicates (i.e., what type of preferences can in general be expressed, and in which form)

While some of these components can be specified by hand, others could be subject to (machine) learning and data-driven adaptation.

`HAI-Co`^{2} comprises a broad variety of human-AI cooperation in natural language, as well as categorical choices and active modification of the candidate solution by the human expert. The search policy π is designed to generate a (locally) optimal candidate solution based on the immediate as well as historical feedback, thereby adapting to the preference signals from the human expert user. The hierarchical abstraction of the search space facilitates a modular modification of the complex candidate solution. As we can see in the running example (Appendix 1), `HAI-Co`^{2} also allows for incorporating creative components into the generation of candidate solutions, for example, through the injection of randomness in the heuristic search process or the refinement of abstract into more concrete solutions.

4. Challenges and future research

Our characterization of `HAI-Co`^{2} implies multiple challenges that need to be addressed to realize co-construction effectively. In the following, we briefly describe these challenges with reference to the current state of research. Additionally, we outline possible future research along these directions.

We start with the structural components of `HAI-Co`^{2} in terms of the construction space and interaction between the AI system and the human expert.

Specification of abstraction hierarchy. Two core components of `HAI-Co`^{2} are (i) the abstraction hierarchy of the construction space and (ii) the neighborhood structure that facilitates preference-based search. A synergistic implementation of (i) and (ii) poses a non-trivial research challenge; several related criteria must be fulfilled, e.g., expressive and abstraction power of the hierarchical representations, ease of expressing human preference across different abstractions, aggregation of utility along the abstraction. In the domain of code generation, Le et al. (2024) propose a modular approach to circumvent this challenge: they generate a chain-of-thought style intermediate description of the subtasks followed by modular codes implementing each of them. Such a hierarchical generation approach can be extended to solution co-construction. However, relying on purely natural language-based intermediate representations limits the utility of the hierarchy – structured, semi-symbolic representations (e.g., UML diagrams for software) can provide better abstraction and facilitate ease of modification. Multi-agent systems adopting hierarchical workflows (Qian et al. 2024; Phan, Nguyen, and Bui 2024) often simulate multi-level representation similar to human organizations. However, such abstractions are not formalized with the goal of co-construction. Such multi-agent systems suffer from limitations in specification and system design. We stress that future research toward designing the co-construction space needs to consider the strengths and limitations of the human expert and the AI system to best utilize the efforts of both. Simultaneously, the formal elements of the co-construction space (e.g., abstraction and refinement maps, utility aggregation, neighborhood structure) must be suitable for expressing expert preference.

Communication in natural language. When humans co-construct a solution, communication in natural language plays an important role. Natural language is a powerful and at the same time succinct medium for conveying information. Given the expressivity of natural language, human and AI agent can easily communicate different options of how to improve the current solution, both at a detailed level and in more abstract terms (Qian et al. 2024). Similarly, preference learning is facilitated by natural language, since many preferences are easily specified in natural language. The challenge here is that the language capabilities of LLMs have advanced to an impressive level for the general domain, but this does not apply to complex expert domains (Magesh et al. 2024; Hager et al. 2024; Anand et al. 2024). It is essential for the AI agent to *understand* what the human partner (as well as other AI agents in case of a multi-agent setup) asks and describe what it needs from them, which remains an important open problem in present human-AI interaction research (Bansal et al. 2024). In this regard, co-construction of objective and solution can serve as a point of fundamental rethinking: by formalizing a neighborhood and imposing a utility structure, it is possible to calibrate the effects of natural language-based interaction on the problem-solving process.

Preferences of the human expert constitute the most important functional component of `HAI-Co`^{2}. Next, we outline the challenges related to preference extraction in complex problem solving with co-construction.

Learning from active human edits. The role of the AI agent as a co-construction partner is central to `HAI-Co`^{2}. This entails the possibility of active participation from the human expert and the need to learn expert preferences from such participation. Current generative AI lacks the necessary structures of the solution space, primarily represented in its input context, that could delineate the changes introduced by the expert and, subsequently, be the basis for learning from it. `HAI-Co`^{2} provides a plausible alternative to "put everything in context" that can solve this challenge, as we argue in the following. Let $\boldsymbol{x} \in \mathcal{X}_j$ and $\boldsymbol{x}' \in \mathcal{X}_j$ be the solution before and after the edit from the human expert, respectively. The neighborhood structure imposed by `HAI-Co`^{2} on $\mathcal{X}_j$ requires learning the changes in the utility function upon moving from a candidate solution to a neighboring one. If one ensures a vector space structure formed by the utility $U^t(\boldsymbol{x})$, then the expert preference is equivalent to $U^t(\boldsymbol{x}') - U^t(\boldsymbol{x})$. Even without learning to map the solutions to the utility, one can simply seek to learn the mapping from $\boldsymbol{x}' - \boldsymbol{x}$ to expert preferences under the assumption of a (locally) linear utility. Gao et al. (2024) previously showed that learning preferences from such changes is superior to prompting-based methods in terms of aligning LLMs to user edits. However, their experiments are focused on simpler, general-purpose natural language generation tasks. In expert-domain applications where the utility of a solution includes multiple hard constraints (e.g., executability of code) along with stylistic preferences, learning a structured representation of the utility space is essential and a challenge on its own.

Multimodal human-AI interaction. Natural language-based interaction is not the ideal channel for all types of preferences. Categorical preference can be communicated more simply by pointing towards a better solution. Thus, we would like to incorporate multiple types of preference into `HAI-Co`^{2}. Similar to deciphering the preference from natural language, different modalities and modeling approaches have their own sets of challenges and require non-trivial research efforts. For example, inferring a preference-based global ranking from pairwise comparisons can be challenging. Popular methods like the Bradley-Terry model (Bradley and Terry 1952) have their own limitations, such as strong assumptions on the preference structure. Prior literature tackling such hurdles (Mao, Weed, and Rigollet 2018; Shah et al. 2016) paves the way for research under the umbrella of `HAI-Co`^{2}. Additionally, incorporation of expert preferences across multiple modalities poses the challenge of aligning these multiple modes of feedback with each other. For example, the human expert may express the need for a security feature in a software engineering problem explicitly, or they can express it implicitly by choosing a candidate solution that includes the feature over one that does not. The AI needs to extract equivalent preference information in these two scenarios. Contemporary research in recommender systems that deal with modeling user preferences on multiple item modalities (Guo et al. 2018; Xu et al. 2021) can serve as a starting point.

However, the relative complexity and nuances of preferences in the case of `HAI-Co`^{2} hinder a trivial extension of recommendation-oriented solutions.

Integration of vision-language models. The rapid advancement of vision-language models (VLMs) (Li et al. 2024a) presents opportunities for extending how human experts communicate preferences within `HAI-Co`^{2} beyond natural language. While visual representations in construction spaces (e.g., UML diagrams) are already achievable through text-based specifications and external rendering, VLMs could expand this ability and also enhance how experts communicate preferences about solutions at any abstraction level. VLMs would enable preference extraction from visual annotations, sketches, and spatial manipulations – providing an additional channel for the expert to convey utility information, particularly useful in inherently visual domains such as UI development. The key challenges mirror those already present in the text domain but with distinct complexities: (i) defining neighborhood structures for visual construction spaces requires formalizing continuous visual operations into discrete semantic transformations, and (ii) visual feedback signals must be decoded into the same utility learning framework that processes textual preferences, ensuring consistency across modalities. Future research should investigate how `HAI-Co`^{2}'s mechanisms – the informational state, preference decoder, and hierarchical consistency constraints – can be extended to incorporate visual feedback while maintaining the systematic search methodology central to the framework.

Dynamic user preferences. Current techniques of aligning neural AI systems to human preferences, broadly referred to as RLHF (Reinforcement Learning from Human Feedback), typically involve a two-stage process: learning a reward model on preference data followed by fine-tuning a foundation model (often an LLM or a diffusion model) on reward supervision from the reward model (Kaufmann et al. 2023). This setup is fundamentally limited to static adaptation in the regime of expert-domain co-construction; a single model of human preferences is imitated by the agent that cannot adapt to the personalized preferences of the human expert. This is a fundamental challenge in co-construction problems, where the AI agent must adapt to the evolving preferences of the human expert. Multi-turn RLHF (Zhou et al. 2024), although it extends the context of preference-adherence to an iterative, conversational regime, cannot solve the challenge of dynamically evolving user preferences. The PAL framework, proposed by Chen et al. (2024), provides a partial solution to our problem via personalized modeling of static human preferences. Unlike traditional policy learning, `HAI-Co`^{2} motivates a reward-free exploration of the solution space (Jin et al. 2020). In-context reinforcement learning can pave the way towards handling dynamic preference signals (Yang et al. 2024; Lee et al. 2024). However, the action space in the scope of `HAI-Co`^{2} overlaps with the generation of multiple hierarchical views of the candidate solution, rendering the problem much harder than existing work on in-context RL. Prior work with LLMs showcases the possibilities of using them as in-context agents, though exploration abilities will need fine-tuning-based interventions (Krishnamurthy et al. 2024).

Guardrails for objective co-construction. Co-construction of the objective opens the door to manipulation to AI agents (Carroll et al. 2019; Hong, Levine, and Dragan 2023); see (Anthropic 2025a) for a real-world example. As we discussed in Section 2, even if there is no general solution to this problem, we see promising avenues of research in the specific context of `HAI-Co`^{2}, including introducing a secondary monitoring agent that alerts the human when there is a suspicion of manipulative behavior by the primary agent and employing the methodology of AI alignment for training the agent to refrain from manipulative behavior – a methodology that is an active area of research with many challenges (Casper et al. 2023), but has nevertheless been successful in reducing the risks of generative AI (Ji et al. 2023; Anthropic 2025b). Another perspective is to broaden the scope of observation. As Miehling et al. (2025) argue in the context of multi-agent systems, focusing on the individual abilities of isolated agents in a multi-agent ecosystem can lead to underestimation – newer phenomena can emerge from the inter-agent and agent-environment interactions. We extend this argument to human-AI interactions and call for agent integrity research under the broad vision of co-construction of solution and objective.

Finally, the search-based methodology outlined under `HAI-Co`^{2} entails certain technical challenges inherent to the present-day AI, which we discuss in the following. Additionally, with the multi-turn feedback-driven process of problem solving, we discuss the non-triviality of evaluation as opposed to standard automatic evaluation techniques.

Specification of informational state. `HAI-Co`^{2} utilizes an informational state to keep track of relevant information in the interaction history. Given that the search policy is conditioned on it, an efficient representation of the informational state is a core challenge of `HAI-Co`^{2}. Typically, such interactive co-constructions are expected to span a long sequence context. While we have observed a significant surge in the context-size of present-day generative AI (e.g., GPT-4 Turbo can handle up to 128K tokens in the input prompt), recent research has questioned the effective usability of such very long context information (Liu et al. 2024). The representation of the informational state needs to be compatible with the abstraction specification of the construction space as well as the choice of how preference signals from the human expert are encoded. This is particularly important as the reflection of any preference signal upon the candidate solution is manifested via the informational state – an unreliable update of the informational state subsequently worsens the solution quality and may result in a divergent search.

Creativity-correctness dilemma. The specific class of co-construction problems that we seek to address requires creative generation. At the same time, in most expert-domain applications, the solution needs to fulfill objective correctness criteria. With generative models, the two requirements of creativity and correctness become counteractive. Creative generation typically emerges in highly stochastic regimes, e.g., in high temperature decoding (Wang, Liu, and Awadallah 2023). However, increased stochasticity carries the risk of hallucination (Aithal et al. 2024). For problems with definite answers, it has already been shown that more robust reasoning can be achieved by stochastic exploration of the generation space and identifying the subset of solutions that are most consistent (Wang et al. 2023). However, such self-consistency methods are limited to problem classes with definitive answers and cannot be readily applied to the co-construction problems that we characterize in this paper. In `HAI-Co`^{2}, this can be generalized into a broader learning problem of exploration-exploitation trade-off. In the early iterations of co-construction, when the preference input from the human expert is likely to be vague, the AI may bias towards exploration of the construction space in search for a creative solution backbone. As the co-construction proceeds, the human expert fixates on the feature requirements and the AI must refrain from abrupt modifications and build on the preference model developed from the early exploration.

Evaluation of co-construction techniques. Due to the dynamics of the co-constructed objective and the complexity and modularity of the solution, the evaluation of co-construction is a difficult challenge. We identify multiple dimensions of evaluation that need to be addressed:

- *Quality of the solution* should be evaluated using domain-specific measures; irrespective of the process of co-construction (of solution and objective), the solution must fulfill some objective criteria of correctness.
- *Preference-adherence* is an essential criterion of the co-construction problem; across multi-iteration co-construction, the generation should be compatible with the human expert's preference input.
- *Self-consistency* is another key aspect of HAI-Co^2, as it allows multiple levels of abstraction along with multiple modes of human preference input; it is essential to quantize how consistently the hierarchical abstraction is represented and different modes of preference input are aligned.
- *Complexity* of co-construction includes the computational complexity of generation and preference-based search – resulting in potentially high computational cost – and the cognitive complexity of the framework – resulting in cognitive load for the expert user. The latter demands significant research efforts from a multidisciplinary approach to ensure that automated assistants truly bring value to the expert.

Given that human experts are costly and have limited time, LLM-based simulation of human-AI interaction may facilitate large-scale evaluation (Tamoyan, Schuff, and Gurevych 2024). Even though our four evaluation criteria seem to demand human evaluation, we conjecture that the development of artificial critic models (McAleese et al. 2024), with human value alignment, will be an important research direction in the future.

5. Related work

We group relevant prior work into several major strands: human-AI cooperation, reinforcement learning from human feedback (RLHF), assistance games, learning from natural language interactions, search- and evolution-driven construction, inference-time compute scaling, LLM agents for complex problem solving, persistent solution spaces for iterative construction and human in the loop. We discuss how these approaches attempt to address (versions of) expert-AI co-construction. However, as we saw in Section 4, they fail to comprehensively tackle these challenges.

Human-AI Cooperation. Recent research emphasizes enhancing human-AI cooperation to support designers in complex, creative tasks. A notable example is the AI-assisted design (AIAD) framework proposed by De Peuter, Oulasvirta, and Kaski (2023), which mirrors our approach in emphasizing cooperation over automation and aiming to support designers' creativity by inferring their goals. AIAD addresses the challenge of goal communication by using generative user models to understand designer reasoning and capabilities from their behavior. This allows AI to provide helpful recommendations and learn from the designer's active participation and corrections. More broadly, this line of work aligns with the vision of Hybrid Intelligence (Akata et al. 2020), which advocates for human-AI systems that interact as co-evolving collaborators with complementary skills and adaptive behaviors. While both AIAD and Hybrid Intelligence share our goal of leveraging complementarity between human and machine, HAI-Co^2 differs from them in several aspects. First, our approach explicitly adopts a hierarchical view encompassing multiple abstraction levels, like specifications, UML diagrams, and code, which is not a core element of AIAD or Hybrid Intelligence. Second, the search in our framework operates across this multi-level structure rather than within a monolithic or unstructured design space, enabling systematic refinement and revision. Third, we place stronger emphasis on actively learning not just from user feedback to AI suggestions, but also from direct human edits to evolving solution artifacts. Fourth, HAI-Co^2 focuses on natural language communication for intuitive and flexible interaction between users and AI. Finally, whereas AIAD emphasizes minimally disruptive assistance with the aim of AI unobtrusively supporting designers, our framework envisions a more proactive and equal partnership, giving the AI system more responsibility in the co-construction process, including proposing refinements to the objective.

Reinforcement Learning from Human Feedback. RLHF focuses on learning a policy preferred by humans, most commonly relying on comparisons between candidate solutions (Kaufmann et al. 2023). The goal is to learn a policy that maximizes a reward or utility function that is consistent with the human feedback. Originating in classical reinforcement learning domains such as games and continuous control (Christiano et al. 2017), RLHF has been extended to a variety of domains, most notably fine-tuning generative models such as LLMs (Stiennon et al. 2020; Ouyang et al. 2022), eventually leading to the development of AI models such as ChatGPT that can generate human-preferred responses in natural language.

RLHF for generative AI is typically employed in a single-turn setting, where the agent generates an immediate response to a query, evaluated by a human expert. This contrasts with expert-AI co-construction, which involves multi-turn interactions where agent and expert collaboratively construct a solution. Multi-turn interactions introduce challenges such as extended time horizons and large action spaces. Extensions to RLHF have been proposed that address these issues (Zhou et al. 2024).

Even multi-turn RLHF, however, is not well suited to expert-AI co-construction without further extension: It does not maintain an explicit representation of the solution space, which is crucial for systematic solution construction. In principle, RLHF could be used to learn the AI agent's policy in HAI-Co^2, but it is challenging to do so interactively as required in HAI-Co^2.

Assistance Games. Originally introduced as cooperative inverse reinforcement learning (Hadfield-Menell et al. 2016), assistance games (Shah et al. 2021; Laidlaw et al. 2024) model human-AI interaction as a game under partial information where the AI strives to learn and maximize the human's underlying reward function. A key assumption is that humans have pre-defined objectives, even if initially unknown to the AI. While HAI-Co^2 shares this overarching aim of AI-driven assistance for human experts in tackling complex problems via iterative engagement and the integration of human preferences, it fundamentally differs in its formalization and underlying assumptions.

An important distinction is the concept that the objective itself is not fixed in HAI-Co^2 but co-constructed through interaction. In complex domains, expert preferences rarely remain static; they evolve through exploration and iterative refinement. This dynamic formation of objectives stands in contrast to the assistance game paradigm, where the AI primarily infers a static, pre-existing human reward function. Beyond this conceptual distinction, assistance games offer elegant theoretical properties but face practical challenges in real-world deployment due to computational complexity arising from the many possible combinations of AI and human policies as well as human objectives.[2] HAI-Co^2 addresses these challenges by structuring the search process

2 Laidlaw et al. (2024) present steps toward scalably solving assistance games in more complex environments. While promising, our approach differs fundamentally through its emphasis on objective co-construction, structured search spaces, and natural-language communication.

at multiple abstraction levels where humans provide feedback at their most intuitive level, while leveraging pretrained language models to guide the search process. While not suitable for all domains in which assistance games apply, we argue that this combination of structured solution spaces, natural language communication, and emphasis on objective co-construction provides significant advantages in the domain of expert-AI cooperation.

Learning from natural language interactions. Natural language interaction between the expert and the AI agent is central to `HAI-Co`^{2}. A popular approach towards facilitating human-AI interactive problem solving involves training the agent to follow instructions in natural language (Branavan et al. 2009; Tellex et al. 2011). This paradigm of learning to follow instructions has found attention in the LLM era as well (Wei et al. 2022a). However, directly mapping language-specified goals to actions has limited applicability to novel tasks. Alternatively, learning rewards from natural language interaction to successfully align the AI agent with the human user has been explored (Fu et al. 2019; Sumers et al. 2021) – instead of learning to map language-defined goals to actions directly, they seek to learn the reward function from the language-defined goals that can be generalized to novel tasks. The findings of Sumers et al. (2022) suggest that while instructions typically perform well in low-autonomy settings, high-autonomy regimes favor the reward-learning paradigm. Instead of learning a language-conditioned policy or reward, it is also possible to use conversational cues as rewards themselves (Jaques et al. 2020), which can be combined with the other approaches discussed here. Yet another approach, deployed in OpenAI's ChatGPT, is to enable the language model to save information about the user and their interactions with the model functioning as a natural-language 'memory', which may include information about the user's preferences.[3] Most of these approaches do not consider the need to actively elicit and co-construct the expert's preferences, a key aspect of `HAI-Co`^{2}. This necessity is supported by Co-Reyes et al. (2019) and Lin et al. (2022), who identify that inferring the correct behavior or reward from a single utterance is non-trivial given the multidimensionality of language. Querying the human user and estimating their preferences in an interactive setup is also a key component of Peng et al. (2024)'s framework. As a step in this direction, Li et al. (2023a) use active elicitation to strengthen preference understanding – the AI agent is trained to elicit and infer human preferences by actively interacting with the user. This is crucial prior work for an implementation of `HAI-Co`^{2}, forming an important pillar of future research under the abstract umbrella it provides.

Search- and Evolution-Driven Construction. Our framework emphasizes iterative search within the construction space, a process akin to evolutionary optimization, which iteratively generates and evaluates candidate solutions (Bäck 1996). This evolution can be viewed as a form of search-based construction. Interactive evolutionary computation, a preference-based extension, is particularly relevant to our work as it involves human evaluation of candidate solutions (Takagi 2001; Wang and Pei 2024). For example, these methods have been applied to search-based procedural content generation in video games (Togelius et al. 2011). Our approach differs in the core approach to the search process: Traditional evolutionary methods maintain a population of candidate solutions and generate new ones through mutation and recombination. In contrast, in our framework, each iteration ends with a single candidate solution that is then the basis for the next iteration. In addition, we leverage the extensive prior knowledge of

3 `https://openai.com/index/memory-and-new-controls-for-chatgpt/`

pretrained language models to guide the search and use natural language communication to facilitate cooperation between the AI agent and the human expert.

Inference-time compute scaling. Allocating additional compute at inference-time can significantly enhance language models' performance on complex tasks. Strategies vary along two axes: (1) **depth vs. breadth** (sequential refinement vs. independent exploration) and (2) **structured vs. learned** process control (externally imposed rules and algorithms vs. autonomous model capabilities guided by prompts or training). Depth-based methods encompass *structured refinement*, like systematic revision (Qu et al. 2024), and *learned refinement*, such as autonomous reasoning steps via chain-of-thought (Wei et al. 2022b; Kojima et al. 2022). Breadth-based methods often involve structured parallel generation with verification (e.g., best-of-N, Cobbe et al. 2021; Lightman et al. 2024). Recognizing complementary strengths of depth- and breadth-based approaches (Snell et al. 2024), recent work advocates combined approaches, either explicitly structuring integration (Wang et al. 2022; Yao et al. 2023; Snell et al. 2024) or relying on models' learned or emergent self-reflection and adaptive search-like behaviors (OpenAI et al. 2024; DeepSeek-AI et al. 2025). Although both breadth-based parallel efforts and depth-based iterative revisions implicitly or explicitly search the solution space, this search is typically conducted non-interactively. This limits the ability to adapt generation to nuanced or shifting user goals *during* the construction process, relying solely on predefined, static objectives or verifiers, with minimal incorporation of explicit human preferences or feedback during search.

In contrast to these predominantly non-interactive methods optimized for static objectives, our approach introduces an *interactive, preference-guided search paradigm* explicitly designed to handle underspecified and evolving goals. While learned reasoning techniques like chain-of-thought excel in domains with clear objective correctness, such as mathematical or symbolic reasoning (Sprague et al. 2025), our method targets complex expert tasks where subjective preferences and evolving requirements critically shape optimal solutions. Instead of seeking solutions based on fixed objectives, we iteratively co-construct solutions aligned with dynamic expert preferences by explicitly integrating iterative feedback. This allows the search process to dynamically reshape the model's reasoning trajectory in response to evolving goals, enabling alignment with expert decision-making in nuanced tasks. Beyond this distinction, many of the approaches discussed above are complementary to our framework and could help produce better base models or be used as inspiration for the search process itself.

LLM agents for complex problem solving. The rapid increase in the capabilities of LLMs has triggered multiple recent efforts to integrate them at the core of autonomous agents that interact with the environment, plan, and act to solve complex problems (Wang et al. 2024). Typical approaches adopt integrating different tool-usage capabilities into LLMs via efficient prompting, often with multimodal capabilities (Chen et al. 2023). A single agent is often insufficient to solve complex problems; thus, multiple agents with different capabilities have to be integrated. Recent efforts in LLM-based multi-agent systems seek to mimic such cooperative problem solving by role-playing LLMs via in-context examples (Li et al. 2023b) or fine-tuning (Juneja, Dutta, and Chakraborty 2024). Multi-agent systems have started gaining traction in expert domain applications as well, e.g., software development (Qian et al. 2024), finance (Ganesh et al. 2024), chemistry (Song et al. 2025). The ChatDev workflow proposed by Qian et al. (2024) showcases the effectiveness of using natural language as the primary communication medium between roleplaying agents for software development. This demonstration in the context of AI-AI interaction strengthens our argument for natural language-based human-AI cooperation. These frameworks of LLM-based autonomous agents are largely designed toward AI autonomy. However, the target problems are not isolated applications – they require interactions between human organization(s), and as a result, these autonomous agents end up as incomplete assistants. Recently, arguments in favor of strategically allocating tasks between humans and LLM-based agents to exploit their distinct strengths have been put forward (He, Treude, and Lo 2024), which aligns with our approach of leveraging the strengths of both humans and AI agents in co-construction. We argue that the open challenges in efficient interaction between a human and an agentic ecosystem, as outlined by Bansal et al. (2024), share some of the characteristics of human-AI co-construction as we envision it. An alternative view of our framework[4] is hence an extension of LLM-based multi-agent systems with a human agent as a key component, focusing on the co-construction of solutions.

Persistent Solution Space for Iterative Construction. A fundamental component of our proposed framework is the explicit representation of the construction space for systematic solution search. Such a persistent memory can be useful for LLM agents. Sumers et al. (2024) propose a cognitive architecture for language agents that connects LLMs to internal memory and external environments, grounding them in existing knowledge or external observations. Similarly, Modarressi et al. (2025) introduce a structured memory component that LLM agents can use for storage and retrieval. In terms of deployed products, Anthropic's artifacts[5] add an explicit representation of an LLM-constructed artifact to the Claude series of language models, which can be iterated on through further interaction. Although these approaches do not directly address expert-AI co-construction challenges, they relate to our approach by providing agents with persistent memory to store intermediate solutions and relevant information for problem solving.

Human in the Loop (HIL). In its original form, HIL refers to a type of interactive machine learning system in which the human has the role of an annotator, advisor or provider of feedback that the system can solicit input from on specific questions and requests (Mosqueira-Rey et al. 2023). For example, in active learning, the machine learning system identifies informative examples for the human to label and retrains the machine learning model iteratively until a termination criterion has been met. While there is no generally accepted definition of HIL and quite heterogeneous approaches have been grouped under this umbrella term by different authors (McCarthy 1959; Settles 2009; Towell and Shavlik 1994; Schramowski et al. 2020; Dejong and Mooney 1986; Natarajan et al. 2014; Maclin et al. 2005; Boutilier 2002), HIL work includes some form of human feedback in contrast to completely automatic forms of machine learning.

In traditional HIL, the human's role is limited: it is confined to the role of an oracle that is consulted with specific requests. The human is not involved in higher-level decisions or co-construction of a solution.

In contrast, as we discussed in Section 2, `HAI-Co`^{2} is a framework in which human and agent cooperate as equal partners, each fully engaged in co-constructing the solution to the problem and in co-constructing the objective. Thus, the cooperation between human and agent is more symmetric in `HAI-Co`^{2} than in HIL.

The need for a more equal partnership between human and AI agent – going beyond a narrow definition of HIL – is being recognized more generally. For example, Natarajan et al. (2025) argue that while there are problems in which the limited role of

4 Indeed, our running example for `HAI-Co`^{2} is closely analogous to multi-agent systems.
5 `https://support.anthropic.com/en/articles/9487310-what-are-artifacts-and-how-do-i-use-them`

humans typical of traditional HIL is appropriate, many other problems (similar to the complex expert-domain problems we target) require a more equal partnership.

The term co-construction. We have borrowed the term co-construction from other disciplines, namely from the social sciences and humanities, where it usually refers to the "joint creation of form, interpretation, stance, action, activity, identity, [..] or other culturally meaningful reality" (Jacoby and Ochs 1995). Although the definition allows for various possible interpretations, co-construction is interactional at its core. However, as the theory of co-construction is beyond the scope of this article and also out of our expertise, we would refer to Robertson et al. (2024, Section 3) who thoroughly discuss constructivist principles and human-AI knowledge co-construction from a theoretical perspective. Beyond this theoretical basis, our scope is substantially different from Robertson et al. (2024) who solely consider efficient techniques for hand-crafting prompts for business managers.

6. Conclusion

Our position is that existing generative AI agents require active human participation to successfully construct solutions to complex expert-domain problems, but cannot effectively serve as reliable partners in human-AI cooperation due to their current limitations. We find evidence for this position in prior research across a broad set of domains. Our running example focuses on software generation using GPT-4 Turbo, a strong proprietary LLM, and exemplifies the major drawbacks of current LLMs such as inability to follow human preferences, unreliable refinement of complex solution artifacts and limitations to facilitate active human participation. We observed that although day-to-day usage of generative AI tends to adopt a type of human-AI co-construction paradigm in an uninformed manner, the challenges that LLMs face confine such interactions to a much weaker form.

As a remedy, we introduce HAI-Co^2, a framework that is motivated by the effectiveness of collective human problem solving. HAI-Co^2 facilitates a solution construction space with multiple levels of abstractions, in which human and AI iteratively refine the candidate solution through search guided by human preference. HAI-Co^2 allows active human participation along with versatility in preference expression. After presenting steps towards a formalization of HAI-Co^2, we discussed the research challenges – including ethical challenges – for this new approach as well as possible future directions for addressing them.

References

Aithal, Sumukh K., Pratyush Maini, Zachary C. Lipton, and J. Zico Kolter. 2024. Understanding hallucinations in diffusion models through mode interpolation. *CoRR*, abs/2406.09358.

Akata, Zeynep, Dan Balliet, Maarten de Rijke, Frank Dignum, Virginia Dignum, Guszti Eiben, Antske Fokkens, Davide Grossi, Koen Hindriks, Holger Hoos, Hayley Hung, Catholijn Jonker, Christof Monz, Mark Neerincx, Frans Oliehoek, Henry Prakken, Stefan Schlobach, Linda van der Gaag, Frank van Harmelen, Herke van Hoof, Birna van Riemsdijk, Aimee van Wynsberghe, Rineke Verbrugge, Bart Verheij, Piek Vossen, and Max Welling. 2020. A Research Agenda for Hybrid Intelligence: Augmenting Human Intellect With Collaborative, Adaptive, Responsible, and Explainable Artificial Intelligence. *Computer*, 53(8):18–28.

Anand, Abhinav, Shweta Verma, Krishna Narasimhan, and Mira Mezini. 2024. A critical study of what code-llms (do not) learn. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 15869–15889, Association for Computational Linguistics.

Anthropic. 2025a. Claude 3.7 sonnet system card. https://assets.anthropic.com/m/785e231869ea8b3b/original/claude-3-7-sonnet-system-card.pdf.

Anthropic. 2025b. Responsible scaling policy. `https://www-cdn.anthropic.com/17310f6d70ae5627f55313ed067afc1a762a4068.pdf`.

Bäck, Thomas. 1996. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press.

Bansal, Gagan, Jennifer Wortman Vaughan, Saleema Amershi, Eric Horvitz, Adam Fourney, Hussein Mozannar, Victor Dibia, and Daniel S. Weld. 2024. Challenges in human-agent communication. *CoRR*, abs/2412.10380.

Bird, Christian, Denae Ford, Thomas Zimmermann, Nicole Forsgren, Eirini Kalliamvakou, Travis Lowdermilk, and Idan Gazit. 2023. Taking flight with copilot: Early insights and opportunities of ai-powered pair-programming tools. *Queue*, 20(6):35–57.

Boutilier, Craig. 2002. A pomdp formulation of preference elicitation problems. In *Eighteenth National Conference on Artificial Intelligence*, page 239–246, American Association for Artificial Intelligence, USA.

Bradley, Ralph Allan and Milton E. Terry. 1952. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4):324–345.

Branavan, S.R.K., Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90, Association for Computational Linguistics, Suntec, Singapore.

Brooks, Ethan, Logan A Walls, Richard Lewis, and Satinder Singh. 2023. Large Language Models can Implement Policy Iteration. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, Curran Associates, Inc.

Cao, Yihan, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip Yu, and Lichao Sun. 2025. A Survey of AI-Generated Content (AIGC). *ACM Comput. Surv.*, 57(5):125:1–125:38.

Carroll, Micah, Davis Foote, Anand Siththaranjan, Stuart Russell, and Anca Dragan. 2024. Ai alignment with changing and influenceable reward functions. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, JMLR.org.

Carroll, Micah, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. 2019. On the utility of learning about humans for human-ai coordination. In *Advances in Neural Information Processing Systems*, volume 32, Curran Associates, Inc.

Casper, Stephen, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Tong Wang, Samuel Marks, Charbel-Raphaël Ségerie, Micah Carroll, Andi Peng, Phillip J. K. Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biyik, Anca D. Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. 2023. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback. *Trans. Mach. Learn. Res.*, 2023.

Chen, Daiwei, Yi Chen, Aniket Rege, and Ramya Korlakai Vinayak. 2024. PAL: pluralistic alignment framework for learning from heterogeneous preferences. *CoRR*, abs/2406.08469.

Chen, Liang, Yichi Zhang, Shuhuai Ren, Haozhe Zhao, Zefan Cai, Yuchi Wang, Peiyi Wang, Tianyu Liu, and Baobao Chang. 2023. Towards end-to-end embodied decision making via multi-modal large language model: Explorations with gpt4-vision and beyond. *CoRR*, abs/2310.02071.

Christiano, Paul, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, Curran Associates, Inc.

Co-Reyes, John D, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, John DeNero, Pieter Abbeel, and Sergey Levine. 2019. Meta-learning language-guided policy learning. In *International Conference on Learning Representations*.

Cobbe, Karl, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.
De Peuter, Sebastiaan, Antti Oulasvirta, and Samuel Kaski. 2023. Toward AI assistants that let designers design. *AI Magazine*, 44(1):85–96.
DeepSeek-AI et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *CoRR*, abs/2501.12948.
Dejong, Gerald and Raymond Mooney. 1986. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176. Copyright: Copyright 2017 Elsevier B.V., All rights reserved.
Dutta, Subhabrata, Ishan Pandey, Joykirat Singh, Sunny Manchanda, Soumen Chakrabarti, and Tanmoy Chakraborty. 2024. Frugal LMs Trained to Invoke Symbolic Solvers Achieve Parameter-Efficient Arithmetic Reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17951–17959.
Fdez-Olivares, Juan, Luis Castillo, Juan A. Cózar, and Oscar García Pérez. 2011. Supporting clinical processes and decisions by hierarchical planning and scheduling. *Computational Intelligence*, 27(1):103–122.
Friedrich, Felix, Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. 2023. A typology for exploring the mitigation of shortcut behaviour. *Nature Machine Intelligence*, 2:1–12.
Fu, Justin, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. 2019. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *International Conference on Learning Representations*.
Ganesh, Sumitra, Leo Ardon, Daniel Borrajo, Deepeka Garg, Udari Madhushani Sehwag, Annapoorani Lakshmi Narayanan, Giuseppe Canonaco, and Manuela M Veloso. 2024. Generative ai agents for knowledge work augmentation in finance. *Annual Review of Control, Robotics, and Autonomous Systems*, 8.
Gao, Ge, Alexey Taymanov, Eduardo Salinas, Paul Mineiro, and Dipendra Misra. 2024. Aligning LLM agents by learning latent preference from user edits. *CoRR*, abs/2404.15269.
Gehring, Jonas, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. 2024. RLEF: grounding code llms in execution feedback with reinforcement learning. *CoRR*, abs/2410.02089.
Grabisch, M., J.L. Marichal, R. Mesiar, and E. Pap. 2009. *Aggregation Functions*. Cambridge University Press.
Guo, Yangyang, Zhiyong Cheng, Liqiang Nie, Xin-Shun Xu, and Mohan Kankanhalli. 2018. Multi-modal preference modeling for product search. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 1865–1873, Association for Computing Machinery, New York, NY, USA.
Hadfield-Menell, Dylan, Stuart J Russell, Pieter Abbeel, and Anca Dragan. 2016. Cooperative Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29, Curran Associates, Inc.
Hager, Paul, Friederike Jungmann, Robbie Holland, Kunal Bhagat, Inga Hubrecht, Manuel Knauer, Jakob Vielhauer, Marcus Makowski, Rickmer Braren, Georgios Kaissis, and Daniel Rueckert. 2024. Evaluation and mitigation of the limitations of large language models in clinical decision-making. *Nature Medicine*, 30(9):2613–2622.
He, Junda, Christoph Treude, and David Lo. 2024. Llm-based multi-agent systems for software engineering: Vision and the road ahead. *CoRR*, abs/2404.04834.
Hong, Joey, Sergey Levine, and Anca Dragan. 2023. Learning to influence human behavior with offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:36094–36105.
Hong, Sirui, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.
Hu, Yue and Xiaojun Wan. 2014. Automatic generation of related work sections in scientific papers: An optimization approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1633, Association for Computational Linguistics, Doha, Qatar.
Hüllermeier, Eyke and Roman Slowinski. 2024a. Preference learning and multiple criteria decision aiding: differences, commonalities, and synergies—part i. *4OR-Q J Oper Res*,

22:179–209.
Hüllermeier, Eyke and Roman Slowinski. 2024b. Preference learning and multiple criteria decision aiding: differences, commonalities, and synergies—part ii. *4OR-Q J Oper Res*, 22:313–349.
Jacoby, Sally and Elinor Ochs. 1995. Co-Construction: An Introduction. *Research on Language and Social Interaction*, 28(3):171–183.
Jaques, Natasha, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2020. Human-centric dialog training via offline reinforcement learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3985–4003, Association for Computational Linguistics, Online.
Ji, Jiaming, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, Fanzhi Zeng, Kwan Yee Ng, Juntao Dai, Xuehai Pan, Aidan O'Gara, Yingshan Lei, Hua Xu, Brian Tse, Jie Fu, Stephen McAleer, Yaodong Yang, Yizhou Wang, Song-Chun Zhu, Yike Guo, and Wen Gao. 2023. AI alignment: A comprehensive survey. *CoRR*, abs/2310.19852.
Jiang, Juyong, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *CoRR*, abs/2406.00515.
Jin, Chi, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. 2020. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pages 4870–4879, PMLR.
Juneja, Gurusha, Subhabrata Dutta, and Tanmoy Chakraborty. 2024. LM2: A simple society of language models solves complex reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 16473–16484, Association for Computational Linguistics.
Kambhampati, Subbarao, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. *arXiv preprint arXiv:2402.01817*.
Kaufmann, Timo, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. 2023. A survey of reinforcement learning from human feedback. *CoRR*, abs/2312.14925.
Kojima, Takeshi, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems*, volume 35.
Krishnamurthy, Akshay, Keegan Harris, Dylan J. Foster, Cyril Zhang, and Aleksandrs Slivkins. 2024. Can large language models explore in-context? *CoRR*, abs/2403.15371.
Laidlaw, Cassidy, Eli Bronstein, Timothy Guo, Dylan Feng, Lukas Berglund, Justin Svegliato, Stuart Russell, and Anca Dragan. 2024. Scalably Solving Assistance Games. In *ICML 2024 Workshop on Models of Human Feedback for AI Alignment*.
Le, Hung, Hailin Chen, Amrita Saha, Akash Gokul, Doyen Sahoo, and Shafiq Joty. 2024. CodeChain: Towards Modular Code Generation Through Chain of Self-revisions with Representative Sub-modules. In *The Twelfth International Conference on Learning Representations*.
Le, Hung, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu-Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
Lecler, Augustin, Loïc Duron, and Philippe Soyer. 2023. Revolutionizing radiology with gpt-based models: Current applications, future possibilities and limitations of chatgpt. *Diagnostic and Interventional Imaging*, 104(6):269–274.
Lee, Jonathan, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. 2024. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36.
Lee, Kuang-Huei, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. 2025. Evolving deeper LLM thinking. *CoRR*, abs/2501.09891.
Li, Ao, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. 2025. Agent-oriented planning in multi-agent systems. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, OpenReview.net.
Li, Belinda Z., Alex Tamkin, Noah D. Goodman, and Jacob Andreas. 2023a. Eliciting human preferences with language models. *CoRR*, abs/2310.11589.

Li, Chunyuan, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng Gao. 2024a. Multimodal Foundation Models: From Specialists to General-Purpose Assistants. *Foundations and Trends® in Computer Graphics and Vision*, 16(1-2):1–214.

Li, Guohao, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023b. CAMEL: Communicative Agents for ”Mind” Exploration of Large Language Model Society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Li, Qintong, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024b. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984, Association for Computational Linguistics, Bangkok, Thailand.

Lightman, Hunter, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Lin, Jessy, Daniel Fried, Dan Klein, and Anca Dragan. 2022. Inferring rewards from language in context. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8546–8560, Association for Computational Linguistics, Dublin, Ireland.

Liu, Jiate, Yiqin Zhu, Kaiwen Xiao, QIANG FU, Xiao Han, Yang Wei, and Deheng Ye. 2023. RLTF: Reinforcement learning from unit test feedback. *Transactions on Machine Learning Research*.

Liu, Nelson F, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Maclin, Richard, Jude Shavlik, Lisa Torrey, Trevor Walker, and Edward Wild. 2005. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, page 819–824, AAAI Press.

Magesh, Varun, Faiz Surani, Matthew Dahl, Mirac Suzgun, Christopher D. Manning, and Daniel E. Ho. 2024. Hallucination-free? assessing the reliability of leading AI legal research tools. *CoRR*, abs/2405.20362.

Maninger, Daniel, Krishna Narasimhan, and Mira Mezini. 2024. Towards Trustworthy AI Software Development Assistance. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, ICSE-NIER'24, page 112–116, Association for Computing Machinery, New York, NY, USA.

Mao, Cheng, Jonathan Weed, and Philippe Rigollet. 2018. Minimax rates and efficient algorithms for noisy sorting. In *ALT*, pages 821–847.

McAleese, Nat, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. 2024. LLM critics help catch LLM bugs. *CoRR*, abs/2407.00215.

McCarthy, John. 1959. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, Her Majesty's Stationary Office, London.

Miehling, Erik, Karthikeyan Natesan Ramamurthy, Kush R. Varshney, Matthew Riemer, Djallel Bouneffouf, John T. Richards, Amit Dhurandhar, Elizabeth M. Daly, Michael Hind, Prasanna Sattigeri, Dennis Wei, Ambrish Rawat, Jasmina Gajcin, and Werner Geyer. 2025. Agentic AI Needs a Systems Theory. *CoRR*, abs/2503.00237.

Modarressi, Ali, Abdullatif Köksal, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2025. MemLLM: Finetuning LLMs to use an explicit read-write memory. *Transactions on Machine Learning Research*. To appear.

Mosqueira-Rey, Eduardo, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. 2023. Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, 56(4):3005–3054.

Natarajan, Sriraam, Saurabh Mathur, Sahil Sidheekh, Wolfgang Stammer, and Kristian Kersting. 2025. Human-in-the-loop or ai-in-the-loop? automate or collaborate? In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 28594–28600, AAAI Press.

Natarajan, Sriraam, Jose Picado, Tushar Khot, Kristian Kersting, Christopher Ré, and Jude W. Shavlik. 2014. Effectively creating weakly labeled training examples via approximate domain knowledge. In *Inductive Logic Programming - 24th International Conference. International Conference on Inductive Logic Programming (ILP-2014), September 14-16, Nancy, France*, volume 9046 of *Lecture Notes in Computer Science*, pages 92–107, Springer.

Nishimura, Kazuya, Kuniaki Saito, Tosho Hirasawa, and Yoshitaka Ushiku. 2024. Toward structured related work generation with novelty statements. In *Proceedings of the Fourth Workshop on Scholarly Document Processing (SDP 2024)*, pages 38–57, Association for Computational Linguistics, Bangkok, Thailand.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

OpenAI et al. 2024. OpenAI o1 System Card. *CoRR*, abs/2412.16720.

Ouyang, Long, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35.

Peng, Andi, Andreea Bobu, Belinda Z. Li, Theodore R. Sumers, Ilia Sucholutsky, Nishanth Kumar, Thomas L. Griffiths, and Julie A. Shah. 2024. Preference-conditioned language-guided abstraction. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '24, page 572–581, Association for Computing Machinery, New York, NY, USA.

Phan, Huy Nhat, Phong X. Nguyen, and Nghi D. Q. Bui. 2024. Hyperagent: Generalist software engineering agents to solve coding tasks at scale. *CoRR*, abs/2409.16299.

Qian, Chen, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15174–15186, Association for Computational Linguistics.

Qu, Yuxiao, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024. Recursive Introspection: Teaching Language Model Agents How to Self-Improve. In *Advances in Neural Information Processing Systems*, volume 37, pages 55249–55285, Curran Associates, Inc.

Robertson, Jeandri, Caitlin Ferreira, Elsamari Botha, and Kim Oosthuizen. 2024. Game Changers: A Generative AI Prompt Protocol to Enhance Human-AI Knowledge Co-Construction. *Business Horizons*, 67(5):499–510.

Ross, Andrew Slavin, Michael C. Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2662–2670.

Roy, B. 1996. *Multicriteria Methodology for Decision Aiding*. Kluwer Academic Publishers, Dordrecht.

Roy, B. 2000. Decision science or decision-aid science. *European Journal of Operational Research*, 66:184–203.

Schramowski, Patrick, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. 2020. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence*, 2(8):476–486.

Settles, Burr. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison, Department of Computer Sciences. `https://minds.wisconsin.edu/handle/1793/60660`.

Shah, Nihar, Sivaraman Balakrishnan, Aditya Guntuboyina, and Martin Wainwright. 2016. Stochastically transitive models for pairwise comparisons: Statistical and computational issues. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 11–20, PMLR, New York, New York, USA.

Shah, Rohin, Pedro Freire, Neel Alex, Rachel Freedman, Dmitrii Krasheninnikov, Lawrence Chan, Michael D. Dennis, Pieter Abbeel, Anca Dragan, and Stuart Russell. 2021. Benefits of Assistance over Reward Learning. Preprint.

Simkute, Auste, Lev Tankelevitch, Viktor Kewenig, Ava Elizabeth Scott, Abigail Sellen, and Sean Rintel. 2024. Ironies of generative AI: understanding and mitigating productivity loss in human-ai interactions. *CoRR*, abs/2402.11364.

Snell, Charlie, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314.

Song, Tao, Man Luo, Xiaolong Zhang, Linjiang Chen, Yan Huang, Jiaqi Cao, Qing Zhu, Daobin Liu, Baicheng Zhang, Gang Zou, Guoqing Zhang, Fei Zhang, Weiwei Shang, Yao Fu, Jun

Jiang, and Yi Luo. 2025. A multiagent-driven robotic ai chemist enabling autonomous chemical research on demand. *Journal of the American Chemical Society*. PMID: 40056128.
Sprague, Zayne Rea, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations*.
Stiennon, Nisan, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, Curran Associates, Inc.
Sumers, Theodore, Robert Hawkins, Mark K Ho, Tom Griffiths, and Dylan Hadfield-Menell. 2022. How to talk so AI will learn: Instructions, descriptions, and autonomy. In *Advances in Neural Information Processing Systems*, volume 35, pages 34762–34775, Curran Associates, Inc.
Sumers, Theodore, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. 2024. Cognitive Architectures for Language Agents. *Transactions on Machine Learning Research*.
Sumers, Theodore R, Mark K Ho, Robert D Hawkins, Karthik Narasimhan, and Thomas L Griffiths. 2021. Learning rewards from linguistic feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6002–6010.
Takagi, H. 2001. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.
Tamoyan, Hovhannes, Hendrik Schuff, and Iryna Gurevych. 2024. LLM roleplay: Simulating human-chatbot interaction. *CoRR*, abs/2407.03974.
Tellex, Stefanie, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 25, pages 1507–1514.
Teso, Stefano and Kristian Kersting. 2019. Explanatory interactive machine learning. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES 2019)*, pages 239–245.
Tikhomirov, Lana, Carolyn Semmler, Melissa McCradden, Rachel Searston, Marzyeh Ghassemi, and Lauren Oakden-Rayner. 2024. Medical artificial intelligence for clinicians: the lost cognitive perspective. *The Lancet Digital Health*, 6(8):e589–e594.
Togelius, Julian, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186.
Towell, Geoffrey G. and Jude W. Shavlik. 1994. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1):119–165.
Vaccaro, Michelle, Abdullah Almaatouq, and Thomas Malone. 2024. When combinations of humans and AI are useful: A systematic review and meta-analysis. *Nature Human Behaviour*, 8(12):2293–2303.
Wang, Chi, Xueqing Liu, and Ahmed Hassan Awadallah. 2023. Cost-effective hyperparameter optimization for large language model generation inference. In *International Conference on Automated Machine Learning, 12-15 November 2023, Hasso Plattner Institute, Potsdam, Germany*, volume 224 of *Proceedings of Machine Learning Research*, pages 21/1–17, PMLR.
Wang, Lei, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).
Wang, Xuezhi, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.
Wang, Xuezhi, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.
Wang, Yanan and Yan Pei. 2024. A comprehensive survey on interactive evolutionary computation in the first two decades of the 21st century. *Applied Soft Computing*, page 111950.
Wei, Jason, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net.

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Curran Associates Inc., Red Hook, NY, USA.

Xie, Jian, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024a. TravelPlanner: A Benchmark for Real-World Planning with Language Agents. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Xie, Jian, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024b. Travelplanner: A benchmark for real-world planning with language agents. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, OpenReview.net.

Xu, Cai, Ziyu Guan, Wei Zhao, Quanzhou Wu, Meng Yan, Long Chen, and Qiguang Miao. 2021. Recommendation by users' multimodal preferences for smart city applications. *IEEE Transactions on Industrial Informatics*, 17(6):4197–4205.

Xu, Frank F., Bogdan Vasilescu, and Graham Neubig. 2022. In-ide code generation from natural language: Promise and challenges. *ACM Trans. Softw. Eng. Methodol.*, 31(2).

Yang, Rui, Xiaoman Pan, Feng Luo, Shuang Qiu, Han Zhong, Dong Yu, and Jianshu Chen. 2024. Rewards-in-context: Multi-objective alignment of foundation models with dynamic preference adjustment. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 56276–56297, PMLR.

Yao, Shunyu, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36.

Zhou, Yifei, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62178–62209, PMLR.

1. An exemplary simulation of `HAI-Co`^{2}

In this section, we present an example implementation of the major elements of `HAI-Co`^{2}, tailored to code generation as a co-construction problem. This example does not claim scientific rigor on its own; instead, we use it to demonstrate what prior findings (see Section 1 and Section 5) already establish. We do not provide a complete implementation of `HAI-Co`^{2}; in particular, the following are not included in the case study: neighborhood structure of the solution space, preference extraction from actual human participation, dedicated utility function tailored toward the expert problem. Instead, we emulate the intended behavior of a complete implementation using prompted LLMs, with the goal of motivating the practicality of `HAI-Co`^{2}.

The initial problem description is underspecified. During the cooperation, the user can introduce new requirements, ask for modifications to the already generated code, and so on. We approximate different aspects of `HAI-Co`^{2} (the surjective mappings between different abstraction hierarchies of the construction space, policy and heuristic search strategy) using baseline implementation strategies for ease of demonstration. Future research endeavors should be directed to more in-depth implementation of these features.

Problem. The user wants to develop a modular Python codebase for simulating a double pendulum. Modules should include components such as I/O interface, visualization and physics engine.

In this example, the construction space $\mathcal{X}$ consists of the set of all Python programs. Three distinct levels of abstraction are implemented. (i) Specification space. A specification of the simulation software in natural language ($\mathcal{X}_0$). (ii) UML space. A UML description of the software ($\mathcal{X}_1$). (iii) Python program space. The Python program

($\mathcal{X}_2$) itself. The abstraction refinements $f_1^{(-1)}$ and $f_2^{(-1)}$ (as introduced in Section 3) are implemented using suitably prompted instances of GPT-4 Turbo that we denote as *UML generation module* and *Code generation module*, respectively; while the former produces a (stochastic) set of refinements in UML given a natural language specification, the latter generates Python implementations of a given UML description. To decode the user's preferences from the interaction, we use a *Preference decoder module*, implemented using prompted GPT-4 Turbo. Following the focus on natural language, the informational state $\mathcal{I}$ is realized primarily as the interaction history in natural language, along with an explicit list of preference rules decoded from this interaction. One can impose a geometric structure on $\mathcal{I}$ by introducing explicit metric space structure on the different abstraction spaces (e.g., edit distance), rendering $\mathcal{I}$ to behave like a trajectory. However, introduction of such structures will be dependent on the expert domain application.

To facilitate the exploration of the candidate solution space, we generate multiple solution representations on different abstraction levels by setting a high decoding temperature in the respective generation modules and sampling multiple responses. Intuitively, we seek to exploit earlier findings that a highly stochastic generation regime facilitates better novelty (Wang, Liu, and Awadallah 2023). Furthermore, this imposed stochasticity can be interpreted as the notion of neighborhood in the respective spaces: one can treat two solutions sampled from the same input context of the generation module as neighbors; distance between two different input contexts can be measured by edit distance. Although we do not explicitly specify such geometric structure in this example, the search strategy uses it implicitly.

We do not implement a concrete realization of the search policy π; instead, we rely on the limited abilities of LLM instances to explore and implement policy iterations (Krishnamurthy et al. 2024; Brooks et al. 2023). While the notion of search is present across all three levels of abstraction, we perform explicit search in the UML space using the *Preference-based search module*, which runs a tournament among candidate UML solutions, guided by the decoded preferences.

The implementation, as depicted in Figure 3, instantiates `HAI-Co`^{2} as follows. The co-construction starts with the user providing an underspecified description of the task (in this example, building a simulation software in Python). The specification module (green rectangle in Figure 3) generates the natural language abstraction of the candidate solution as a list of possible components of the software along with their functionalities. This serves as a transparent interface in natural language that provides a layout of the construction. A pair of candidate specifications are generated using a high-temperature stochastic generation regime. The user chooses one of them as better. Additionally, they can state any explicit modification request. Moreover, they can directly edit the specification if they have specific requirements in mind (*preemptive reviewer*), or choose to continue with the workflow and decide on the specifics upon observing the final program (*lazy reviewer*). The Preference decoder module (purple rectangle in Figure 3) lists down the preferences decoded from the user's actions. If the user introduces any new modification (e.g., in the presented example, they remove certain modules from the specification and insert new modules), the specification module generates a new pair of specifications for the user to provide feedback on. This continues till a suitable specification is obtained.

Next, the UML generation module generates a set of stochastic refinements of the natural language specification into UML descriptions. The UML description of the software forces the subsequent code generation module to generate a final program that consists of multiple, independent components (in this case, Python classes) and well-defined dependencies among such components. Micro-level changes to the code (e.g., changing the design of the GUI, choice of numerical algorithms in the simulator, etc.) can be facilitated now without changing the complete codebase – a desirable property of our implementation that is closer to real-life software engineering. This addresses the challenge monolithic code LLMs face in scenarios in which persistent editing is required. However, generating the Python programs from all such candidate UMLs and verifying them one by one is both computationally expensive and infeasible for the human user.

The preference-based search among the candidate UMLs is implemented as a tournament by iteratively declaring one among a pair as the winner of a round. After a logarithmic order of such rounds ($\log_2 n$ being the depth of the tournament tree for n candidate UMLs), the Preference-based search module comes up with a final pair and a summary of preference justifications.[6] Note that although we seek to minimize the cost of human intervention in this step by automating preference-based ranking, one can envision the human expert providing their judgement on these UMLs. In such a setup, the Preference-decoding module can be used to explicitly adapt to such gold preference examples.

Next, we utilize the code generation module to translate each of the two selected UML candidates into a candidate Python program that will be used for human feedback post-execution. Aligned to the goal of co-construction, in this last stage, the user provides their binary judgment on the relative quality of the two generated Python programs along with (optionally) natural language feedback. Such feedback can incorporate the errors found in the program execution (if any), additional requirements, etc. This feedback, along with the summary of the tournament generated by the preference learning module, are together used as a context for the next iteration of refinement. This iterative process continues until the user is satisfied with the solution.

Comparison with monolithic LLMs. We perform offline human evaluation to compare `HAI-Co`^{2} against a vanilla LLM (in this case, GPT-4 Turbo) in terms of their effectiveness as co-construction partners for expert domain problems. The problem to be solved is to generate code for the double pendulum simulation. Multiple co-construction episodes are generated by specifying different initial preferences and mid-episode preference switching (e.g., choosing different specifications generated by the Specification module; asking for different functionalities in the software; and editing different segments in the specification as well as the codes). Each human evaluator is presented with a pair of co-construction episodes – one using `HAI-Co`^{2}, one using GPT-4 Turbo – and asked to compare them in terms of the following criteria:

Q1. Which assistant has better incorporated the initial preferences of the user?
Q2. Which assistant better adapted to preference switching?
Q3. Which assistant is more precise in terms of iterative refinement? 'Precision of modification' means that the changes are relevant to the request.
Q4. Which assistant is more complete in terms of iterative refinement? 'Completeness of modification' means that all the necessary changes have been made.
Q5. Overall, which assistant seems more suitable for software-level code generation?

We recruited a total of 14 participants for this evaluation; each is a doctoral student in Natural Language Processing, so that expert preferences in code generation can be

6 See `https://subha0009.github.io/ExAIC-Interactions/PreferenceLayer.html` for the tournament on the candidate UMLs generated

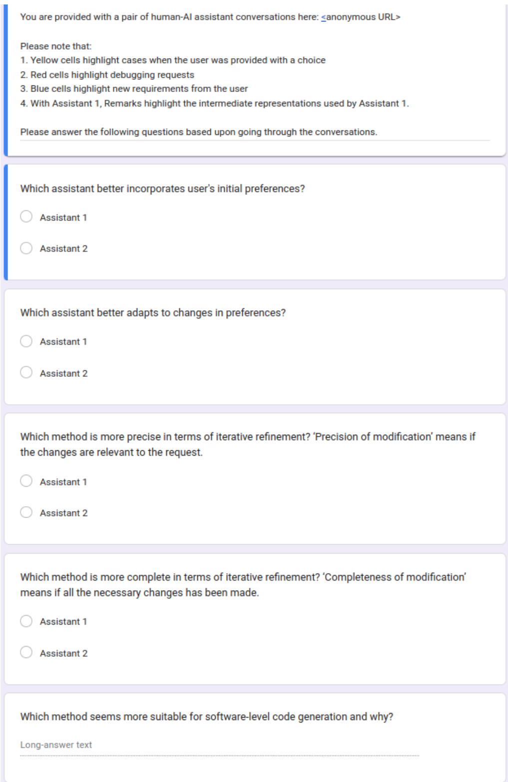

Figure 4
Interface used for human evaluation.

understood. We provide each evaluator with a pair of interactions between i) a user and GPT-4 Turbo [7] and ii) a user and `HAI-Co`^{2} [8]. Figure 4 shows the interface used to collect the users' responses. We do not collect any personal information from the evaluators.

On all five criteria, the majority of evaluators rate `HAI-Co`^{2} higher than GPT-4 Turbo. 12 (85.7%) evaluators find that `HAI-Co`^{2} better captures the initial preferences of the user (**Q1**). 11 (78.6%) agree that it can adapt better to preference switching (**Q2**). 11 (78.6%) see `HAI-Co`^{2} as superior on precision (**Q3**) and completeness of modification (**Q4**). 12 (85.7%) suggest `HAI-Co`^{2} is better suited for software development (**Q5**). Detailed responses are available here: `https://subha0009.github.io/ExAIC-Interactions/FormResponses.html`.

Limitations and further improvements. The immediate improvements we observe are prevalent without any dedicated implementation – neither of the two refinement maps (from natural language to UML and from UML to Python) nor of the preference-based search policy. In this implementation, we do not equip the co-construction space with explicit neighborhood structures. We posit that development along these directions

7 Example interaction can be found at `https://subha0009.github.io/ExAIC-Interactions/Assistant2.html`
8 Example interaction can be found at `https://subha0009.github.io/ExAIC-Interactions/Assistant1.html`

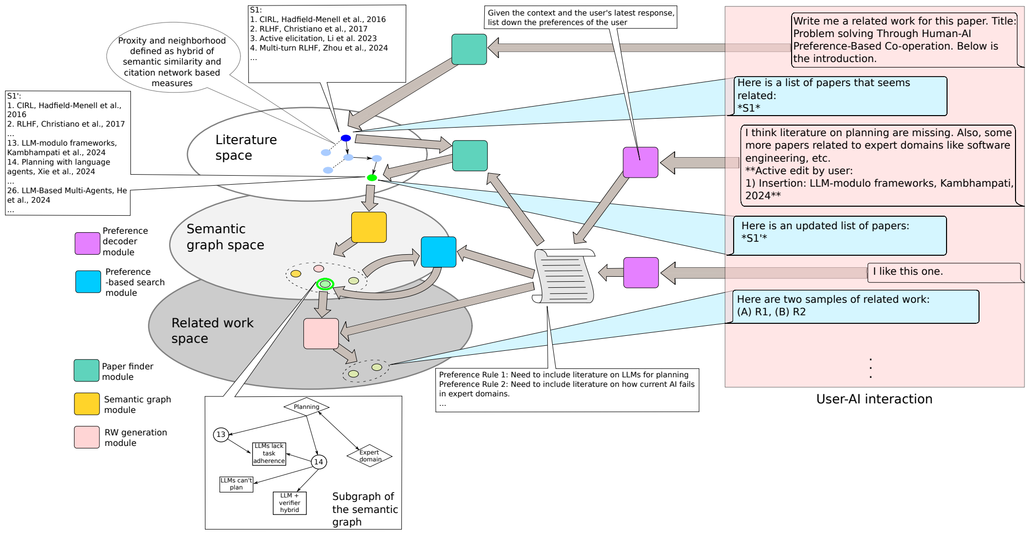

Figure 5
A conceptual application of HAI−Co^2 for the problem of generation of a related work section of a research paper. The user provides the title and the introduction of the research paper for which the related work section is to be co-constructed (i.e., written). Three abstraction hierarchies are envisioned. The Literature space consists of lists of papers. The Semantic graph space depicts the papers, their findings, and their interrelations using a directed graph. The Related work space contains related work sections (written in natural language using a writing style appropriate for this genre) that describe the semantic graph. The Paper finder module lists down the papers that should be incorporated into the related work. Neighborhood structure is imposed upon this space using semantic similarity and hops over the citation network. The Preference decoding module keeps track of the user's preferences. Upon deciding on a suitable list of papers, the Semantic graph module translates them into a semantic graph. Finally, the Related Work (RW) generation module writes related work sections based on the semantic graph.

will further improve the quality of co-construction and confirm HAI−Co^2's potential as an effective framework for the class of co-construction problems we aim to address. Modern software development relies on software engineering tools such as type systems, test drivers, static program analysis tools, monitoring and debugging tools and security vulnerability detectors. Realistic software artifacts are complex, and their full evaluation by humans without these tools is infeasible. Our implementation of HAI−Co^2– not intended as a systematic evaluation of HAI−Co^2's effectiveness in the software domain – will need to be extended with many of the elements that are standard in the DevOps pipeline (see (Le et al. 2022; Maninger, Narasimhan, and Mezini 2024) for examples of how to integrate such standard tools). We use the LLM's generative capacity as is, e.g., when it explains why one generated solution is better than another. An interesting research direction would be *explanatory interactive learning* (Ross, Hughes, and Doshi-Velez 2017; Teso and Kersting 2019; Friedrich et al. 2023), where more faithful explanations are produced through interactively constraining model explanations. The search strategy can be further refined by implementing reinforcement learning from execution feedback (Gehring et al. 2024; Liu et al. 2023; Dutta et al. 2024).

2. Possible implementation in Related Work Generation

To showcase the applicability of HAI-Co^2 for expert domains other than software engineering, we present a workflow for the problem of related work generation in Figure 5. Note that this is not an actual implementation, rather a proposal on how HAI-Co^2 can be adapted for this problem. Similar to our case study on software engineering, a solution construction space with three levels of hierarchy is defined. The highest level of abstraction (Literature space) is the space of lists of relevant papers; each point (represented as a list of papers) is intended to capture the literature relevant to the research paper (in this case "Problem solving through Human-AI Preference-Based Co-operation"). Based on the user's problem specification, the Paper finder module (which can be an LLM-web search hybrid) lists the possible papers that are relevant to the research paper. This is a classical search problem. Next, these papers are used to construct a semantic graph that relates different papers according to their domain of focus, methodology, findings, prescriptions, etc. Such a graph is inherently heterogeneous. Multiple semantic graphs can be generated from a given list of papers. One can define a neighborhood over the space of these semantic graphs via edit distance. The search strategy, in this case, can again be realized through a tournament (as in the software engineering domain presented in Section 1) or through another mechanism (e.g., a specialized module that evaluates semantic graphs based on their graph-theoretical properties). Finally, the RW (related work) generation module translates these semantic graphs into Related work sections. Locally valid neighborhood structures can be constructed using the neural representations of textual differences. The Preference decoder module extracts the preferences expressed by the user to guide the search in different spaces. In this example, one can define refinement maps between the different abstraction levels in a straightforward manner: The semantic graph can be mapped to the list of papers directly as the former has nodes that are members of the latter. Similarly, each pair of nodes and their connecting edge in the semantic graph can be translated to a sentence in the related work section in the Related work space.