

Strategy assessment for solving rich physical problems

Chiheb Boussema

CMU-RI-TR-23-68

August 10, 2023



The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:

Christopher Atkeson, *Chair*
David Held
Oliver Kroemer
Leonid Keselman

*Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Robotics.*

Copyright © 2023 Chiheb Boussema. All rights reserved.

Abstract

Every day humans face challenges that require ingenious solutions and making do with limited resources. From opening a stuck jar to re-floating a sunken ship to improvising a cooling system or an air/water filtration system, the examples abound. Often, coming up with a good strategy calls upon an understanding of a rich set of physics that spans multiple domains. While humans excel at solving such problems, this remains a challenge for robots. Arguably a lack of a rich physics engine that can be quickly and flexibly instantiated, as well as the combinatorial complexity of identifying what physical concepts are relevant to a particular situation are key roadblocks. This work constitutes an exploratory investigation towards addressing these issues.

In particular, we propose an “intuitive physics reasoner” to assess strategies that are proposed to solve a given problem. We focus on problems where the physics take a precedence over geometric considerations. We believe the ability to quickly determine whether a strategy is worth considering and allocating further resources to planning using that strategy is useful.

As a major design consideration, we required that the input to our system (i.e., a strategy) be as simple and convenient as possible. This is why we use strategies expressed in natural language as the input to our system. This not only allows humans to easily provide candidate strategies, but also makes for a convenient bridge with the recent developments in generative AI, notably large language models (LLMs).

We build a physics knowledge library, with an explicit encoding of knowledge, such that it facilitates targeted knowledge retrieval and the building of knowledge graphs with the physical concepts that are relevant to a given situation. Such graphs form the substrate for subsequent reasoning steps, such as forward simulations and parameter optimization.

On the other hand, we leverage LLMs’ versatility to translate a strategy described in natural language into a computationally-usable form, which is a graph, similar to the ones described above.

The reasoner can then verify that a strategy is consistent with its prior knowledge of physics, and that inputs, states, and outputs are within set limits. It is also capable of finding optimal parameters according to a given criterion. We construe the verification of the validity of the physics (at an abstract level) as graph matching or a correspondence problem. Then, a numerical reasoning step leverages the graph structure to propagate information (forward for simulating, or backward when computing Jacobians), and verify constraints satisfaction. This ultimately allows the reasoner to assess the feasibility of a strategy, both on an abstract physics front (in terms of idealized and simplified physics) and numerically, with the option of finding optimal quantities when not specified by the natural language strategy.

We demonstrate the system’s capability on a number of problems involving a diverse set of physical and chemical concepts, with favorable outcomes in terms

of assessment accuracy.

Acknowledgments

I would like to thank my advisor Chris Atkeson for his support and advice, as well as all my friends and family who provided support during these years.

I would also like to thank our research sponsors who helped fund our work: Tata Consultancy Services, and CMU's RI department.

Contents

1	Introduction	1
2	Related Work	5
2.1	Cognitive sciences	5
2.2	Task and Motion Planning	6
2.3	Knowledge representation	6
2.4	Embodied task planning ML	7
3	An Intuitive Physics Reasoner	9
3.1	Physical knowledge library	9
3.2	Reasoning Capabilities	11
3.2.1	Knowledge graphs	11
3.2.2	Grounding phase	12
3.2.3	Optimization of intermediate parameter values	14
3.3	From natural language to a suitable graph	15
3.3.1	Graph matching problem	17
3.4	Overall system for strategy assessment	18
4	Results	21
4.1	Overall success rate	22
4.2	Graph matching	22
4.3	Numerical reasoning	24
5	Discussion	27
6	Conclusion	31
A	Scenarios and Strategies for Thermodynamics	33
A.1	Improvising a desalination system	33
A.2	Improvising a cooling system	34
A.3	Improvising a water heating system	35
A.4	Getting a fire started	36
A.5	Improvising an air conditioning system	36

List of Figures

3.1	Summary of the overall system.	10
3.2	Natural language description turned into a graph by the LLM.	17
4.1	Failed graph correspondence because of a bad natural language-to-graph transcription.	23
4.2	Successful graph correspondence.	23
4.3	Successful graph correspondence.	24

Chapter 1

Introduction

Challenges for robotics include generating and then selecting strategies to solve problems, and improvising when unexpected events happen. The problems that we, humans, face in everyday life or may face in mission-critical situations such as space exploration, often necessitate reasoning beyond tool mechanics [5] and geometric reachability and retrieval issues [37, 42]. Opening a jar with a stuck lid for instance calls for a physics-based reasoning: running it under hot water increases temperature, which in turns increases the pressure inside, thereby relieving the vacuum that is making it difficult to open the lid. Recently, Large Language Models [4] have demonstrated that, given a description of a problem or situation and a goal, they can generate alternative strategies. However, some strategies can be “hallucinated” and incorrect or not applicable, and others may not be possible with the resources available or too costly. Verifying that a specific strategy will work, and estimating its costs and benefits, can be done using detailed planning and policy optimization in simulation and, more expensively, using a real robot and problem setup. This is what NASA does over many years and at great expense when planning a space mission [29]. Detailed planning and policy optimization is computationally expensive, and also costly in terms of human engineering effort. It is often not clear how to allocate computational and human resources to transform a set of vague strategies expressed in natural language into detailed plans and assess the proposed plans to most efficiently select a good or even optimal plan.

The hypothesis we are testing is whether reasoning, planning, and simulation at a variety of levels of simplification and abstraction, based on stored (and possibly learned) knowledge of physics and models of the effects of robot actions, is useful to bridge the gap between proposed strategies expressed in natural language and planning and simulation using the most accurate models available to assess those proposed strategies. Our specific research goal is to automate and reduce the computation necessary to find good strategies, whether they are proposed by humans or language models, or are generated from a stored knowledge base.

We will use the goal of floating an object in a liquid as one of our example problems. Powerful methods have already been developed for motion planning, so we are focusing on problems where physical knowledge is useful but geometric issues are secondary. When asked, GPT4 [1] suggested the following strategies: 1) Add solvents to the liquid to increase

1. Introduction

its density until the liquid is more dense than the object. 2) Inflate the object or attach inflatable cells ("floaties") to reduce the object's density. 3) Use magnets. 4) Use surface tension. 5) Use hydrodynamics. We humans can come up with some additional strategies, which we believe could be automatically generated from a knowledge base: a) Put the object on another object that already floats, such as a raft or boat-like object. b) Freeze the water completely. Or just make a big iceberg and use (a). c) Build a structure that supports the object. The structure's foundation is at the bottom of the liquid on whatever surface is below the liquid. d) Add a vertical thruster to the object. e) Hang the object from a balloon, kite, helicopter, or other aerial vehicle. We have certainly not exhausted the space of possible strategies, and that space may be infinite.

In this study we aim to build a system that can verify the validity and feasibility of proposed strategies. This system should accommodate rich physics that go beyond geometric considerations and may be expensive to accurately simulate. In particular, we avoid considering geometric path planning, which typically involves trajectory optimization for multiple degrees of freedom. We focus on quickly evaluating the physical validity of a strategy before having to involve difficult and expensive trajectory optimizations: assessing opening a stuck jar by putting it under hot water should not initially involve reasoning about the entire trajectory and motor commands. Assessing the validity of a strategy in this work implies verifying that it is consistent with prior knowledge, and that any intermediate values are attainable. We also include the possibility of optimizing such values (e.g., quantities) and therefore complementing vaguely proposed strategies with a set of feasible parameters. The component challenges that we address are the following: (1) acquiring a knowledge base that can be used to simulate rich physics, (2) transforming a strategy proposed in natural language into a computationally-usable form for an intuitive physics reasoner, and (3) assessing the feasibility or validity of the proposed strategy.

This study builds a first prototype, which defines the form of the knowledge and how it might be used, as well as how one might leverage the versatility of LLMs to facilitate the transition from a natural language description of a strategy to a physics checker and reasoner (i.e., a sort of intuitive physics engine with reasoning capabilities). We restrict ourselves to manually building a physical knowledge library, where we encode knowledge at the level of basic, first principles physics. We believe encoding simple physics is conducive to maximum re-usability of information, as the same physical concept may be implicated in various phenomena. Notably, we restrict the scope of physical reasoning to quasi-statics and equilibria, and we leave flows and time evolutions typical of dynamic phenomena for future work. We describe pilot studies to flesh out this approach, using floating an object and other case studies.

The contributions of the present work are the following: (a) we formulate a way to build a general library of simple and commonly-occurring physics spanning different physical and application domains (noting that the library can be further expanded and/or used as a precursor for further learning), (b) we present a way to generate an "intuitive" physics reasoner from a natural language description, from which we can (c) evaluate a given strategy.

A discussion including work still under exploration as well as future work ideas is presented

at the end of this manuscript.

1. Introduction

Chapter 2

Related Work

2.1 Cognitive sciences

There has been a great deal of interest in the fields of cognitive science and psychology on understanding and modeling human and animal physical problem-solving capabilities and generalizable reasoning [5, 7, 9, 18, 19, 24, 28, 34, 38, 46, 47]. In [7, 24, 38], the authors posit the notion of built-in intuitive physics, which allow infants to make inferences on plausible outcomes and constrain learning. In our work, we endow a robotic agent with a physical knowledge library, where physical concepts may relate to one another via simple mathematical expressions. The authors in [24, 38] further propose that the ability to compose knowledge to build causal models of the world that generalize the reasoning capabilities is essential for human-level intelligence. Structuring one’s knowledge base as communicating vessels requires years of learning and experience and is at the heart of one’s cognitive agility and expertise [17].

While this statement can be intuitively understood, it is also supported by the structure of the brain’s neural connections, where each neuron contacts and is contacted by a large number of other neurons (noting that most connections do not actually fit into a hierarchical scheme). A recent study [32] has shown that information in a biological nervous system flows through topologically highly complex neural structures, a process that appears to be a fundamental property of decision-making, contrasting with the topologically-impooverished artificial neural networks [40]. These insights inspired us to impose connectivity capabilities when designing the structure of our knowledge base. In other words, the knowledge base should provide the possibility to quickly connect physical concepts that may interact (for example, pressure and temperature for gases). Relatedly, the authors in [5] endow a reasoning agent with rich prior knowledge on what objects and which actions to use to solve a mechanical reasoning problem. A stochastic intuitive physics engine is then used to simulate outcomes before committing to a strategy. Finally, we also designed our knowledge library and reasoning algorithm so that objects may be considered at different levels of abstraction [38] depending on the physical properties of interest to a task (i.e., identifying affordances to objects [30]).

2.2 Task and Motion Planning

Solving physical puzzles in robotics has traditionally been the subject of Task and Motion Planning (TAMP), which combines both discrete and continuous-space subproblems [16]. Usually, the robot comes with a knowledge base set in the form of a set of action primitives and semantic predicates that include pre-conditions (states and predicates that must hold true), effects (indicating which variables change) [21], and constraints (for example imposing kinematic constraints on a motion primitive, or a particular placement or configuration of objects relative to one another). The most common approach to forming a plan skeleton is then to sample action sequences, often interleaved with constraint satisfaction and action parameter-finding [16, 20, 36, 37]. This approach can be negatively impacted by the length of the planning horizon as well as the breadth of the total knowledge base, especially when the constraint satisfaction problem is complex. In [37] for example, the constraint satisfaction problem is akin to a trajectory optimization problem subject to dynamics constraints, which is computationally burdensome especially when the objective is to (a) find multiple strategies or action skeletons, and (b) ultimately only execute one. Our framework is inspired by [37], but we specifically sought to encode knowledge in a way that allows purposeful knowledge graph generation using relevant information, as opposed to a sampling-based approach. Additionally, a large set of the TAMP literature focuses on problems related to geometric reasoning, i.e., identifying and using objects whose geometry is the key to task completion, as in object retrieval or placement [10, 23, 27, 37, 42, 43]. In this manuscript, we sought to expand the problem-solving capabilities by allowing more general physics-based reasoning.

2.3 Knowledge representation

A knowledge base representing the world is necessary to perform reasoning, and in fact shapes the types of reasoning that is possible. In [35], the authors use an extensive knowledge library, represented as an ontology relating various levels of actions, objects, temporal events, and mathematical concepts, to retrieve information (e.g., possible locations of ketchup) required by pre-programmed plans or reasoning methods (e.g., a *clear-table* routine). Extensive work is presented in [35] where they also show the possibility to acquire new knowledge, e.g., from the web, and map it onto their ontology. In our work, we focus on the ability to assess novel (not predetermined) plans by composing relevant knowledge. In particular, we are interested in sequencing small quantitative models in contrast to end-to-end task models [3, 6, 44, 45]. In [44], the authors represent a task (e.g., pouring a liquid into a container) as a graph-structured dynamical system potentially subject to discrete selection decisions that lead to different dynamics. The objective is to find good selections at bifurcation steps of the task graph, as well as optimize continuous parameter values of actions with a stochastic differential dynamic programming approach. Similar to [44], each strategy or plan in this manuscript is represented in a tree structure that allows both the forward propagation of relevant knowledge and variables, as well as the computation of a task Jacobian in a back-propagation-like recursive manner. This Jacobian

then allows for parameter value optimization (possibly for action primitives), and for a qualitative assessment of tools (although this point is not explored in this work). This is related to an older approach to AI - qualitative reasoning, or qualitative physics - where variables and their relationships are represented using qualitative values such as increasing, decreasing, or constant, and system behavior can be inferred using some qualitative algebra [14, 22]. While in the above works, the symbolic concepts that describe the world or task are pre-determined, [21] presents a framework to automatically build a symbolic representation given an agent’s environment and actions by identifying the set of pre-conditions and effects of skills following a training phase where an agent executes feasible options or skills at random.

2.4 Embodied task planning ML

While complex problem-solving has traditionally been tackled by a TAMP approach, recent strides in machine learning foundational models have generated impressive results. In [4], the authors employ an LLM to suggest actions to a scoring module learned via reinforcement learning. The robot is endowed with a skills library, each having a natural language tag and a value function indicating the likelihood of success at a given state. The robot picks and executes the action that has the highest combined likelihood of being generated by the LLM and of being successful (given its value function). While impressive, the framework is only demonstrated on object fetching scenarios. Additionally, it struggles with certain task formulations such as negation, and requires an enormous amount of data and time for skill acquisition. In [13], the authors introduce a multi-modal language model that embeds continuous observations (e.g., images) into the same embedding space as language tokens. Given a text description of a task, it generates a text-based response, which, in addition to an image observation, maps onto a low-level policy. These steps can be repeated, thus generating and carrying out long-horizon mobile manipulation tasks, with one-shot and zero-shot generalization capabilities. These frameworks demonstrate impressive results, but only generate plans an action at a time with no guarantee of success or long-horizon feasibility. Additionally, LLM outputs can be factually wrong and difficult to correct. In [25], the authors present an LLM specialized for solving quantitative reasoning tasks, typical of STEM fields. Solving such problems is related to our reasoning framework, which has to solve mathematically-expressed physical constraints to find parameter values (e.g., quantities) and assess feasibility of strategies. However, it is shown in [25] that this specialized LLM (a) suffers from a non-negligible failure rate, and (b) half the failures are due to reasoning errors. This aspect of LLMs (hallucinations, mathematical, physical, and logical errors) justifies the need for a separate verification process that can catch and possibly correct the inconsistencies of LLMs. In our work, verification arises from a library of prior, human-vetted, physical knowledge.

Another type of planning where LLMs were used is that of object placement and rearrangement. In [12], an LLM generates symbolic then geometric (with numerical values) spatial relationships between objects, information that gets transmitted to a task and motion planner to optimize plans.

2. Related Work

As with the TAMP paragraph above, most works have focused on object fetching and geometric reasoning scenarios. In contrast, our work is oriented towards more general physics-based reasoning, where geometric knowledge alone is not sufficient to find solutions. Additionally, we believe that cognitive models like this work, and model-building capabilities [9, 38] will be complementary capabilities to LLMs.

Chapter 3

An Intuitive Physics Reasoner

Solutions to complex problems that may require multiple steps and call on a rich set of physics are often easily and naturally described in natural language. This is the case for both humans and AI with the recent developments in large language models [1]. However, such natural language based solutions may be fraught with problems, especially if they are LLM-generated (hallucinations, mathematical and logical errors and incoherences). Therefore, an added verification process based on vetted human knowledge is needed. In this study, we build such a process, which serves as an intuitive physics reasoner. This reasoner makes use of prior knowledge encoded in a physics knowledge library. This knowledge is used to generate graphs relating the physical concepts involved in a particular situation, allowing further numerical calculations to take place. On the other hand, we leverage the versatility of LLMs in order to transform a strategy given in natural language into a corresponding graph. The latter is checked against the graphs that were built using prior knowledge in order to find a match.

The overall system acts as follows: first a natural language solution is transcribed into a graph, then we solve a graph matching problem with the graphs that were generated from prior knowledge (this step verifies the physical consistency of the proposed solution at a high level), then a numerical verification phase (akin to a forward simulation in its simplest form) with the possibility of finding optimal parameters takes place. This is summarized in Fig. 3.1.

In this section, we will explain the various components that make up our physical reasoning framework. We will start by describing the physical knowledge library and its structure. Then we will explain the strategy assessment algorithm, which is comprised of a high level physics checking cast as a graph matching problem, and a numerical verification phase, possibly complemented by an optimization phase. Finally, we describe how we transition from the natural language description of a strategy to its verification.

3.1 Physical knowledge library

A crucial part upon which the framework we present in this manuscript relies is the presence of an explicit knowledge library that includes a way to connect “pieces of knowledge” when

3. An Intuitive Physics Reasoner

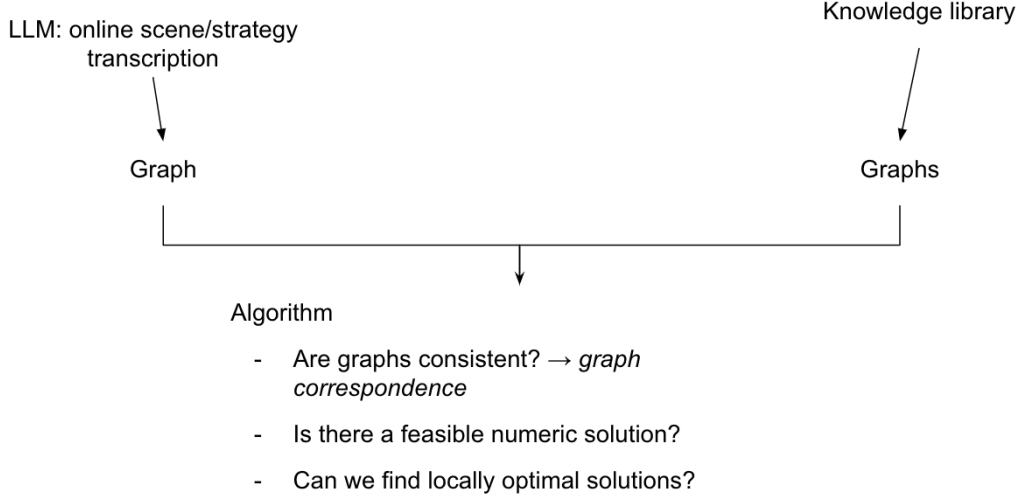


Figure 3.1: Summary of the overall system: a scene/strategy is turned into a graph, whose validity is verified from prior knowledge.

they are related, as well as the assumptions or constraints, if any, that define the existence of particular relationships or particular expressions. In other words, the physical knowledge library not only stores a number of physical concepts or properties, but also elucidates the connections that relate different concepts to each other.

These concepts or properties can be material properties (e.g., stiffness, density, surface energy), states (e.g., position, velocity, concentration, temperature), forces, stresses, ... All concepts are introduced in a symbolic manner, and are computationally structured as objects (or classes) having inputs and outputs. A property may be ‘packaged’ as a graph node for the various algorithmic processings described later.

A physical object (e.g., a table) is then simply viewed as a collection of physical properties. The outputs of these properties are a function of the inputs, and this function can vary depending on assumptions or constraints.

Illustrative example: consider the properties $\mathcal{P}_1, \mathcal{P}_2$ and \mathcal{P}_3 in the knowledge library. \mathcal{P}_1 has the inputs $In_{\mathcal{P}_1} = \{X, Y, Z\}$ whereby X, Y are themselves outputs of \mathcal{P}_2 (i.e., $X, Y \in O_{\mathcal{P}_2}$, where $O_{\mathcal{P}_i}$ is the set of outputs of property \mathcal{P}_i), and Z is an output of \mathcal{P}_3 . \mathcal{P}_1 has a single output W , but its expression depends on assumptions as follows:

$$W = \begin{cases} f(X, Y); & \text{if } c_1(Z) \geq 0 \\ g(X, Y); & \text{if } c_2(Z) \geq 0 \end{cases}$$

where f, g are different functions, and $c_{1,2}$ define different constraints.

Finally, a property’s inputs further carry the information of whether they are *internal* to the object of interest (e.g., the object’s own density or Young’s modulus), or *external*, meaning that they represent information that results from an interaction with other objects, for instance by direct contact (e.g., the temperature of another body for heat transfer processes), as well as which property \mathcal{P} they came from.

We give here an example to illustrate the above description. Let us consider the density, ρ , of a liquid. This property has a nominal value, ρ_0 , which is valid most of the time. However, when some material is dissolved in the liquid, the solution’s density changes to be a weighted average of both substances¹. The description of the assumptions can be made in terms of predicates that may be related to known action primitives (e.g., X is dissolved in Y, in which case ‘dissolved’ should refer to a known action primitive in the robot’s actions library), or simply describe a particular set of constraints (e.g., position of the material is inside the container that holds the liquid).

An additional important piece of information that each property \mathcal{P} holds is its internal Jacobian, $J_{\mathcal{P}}$, relating its outputs to its inputs: $J_{\mathcal{P}} = \frac{\partial \text{outputs}}{\partial \text{inputs}} \in \mathbb{R}^{N_{out} \times N_{in}}$, where N_{out}, N_{in} are the number of outputs and inputs, respectively. In case there are some assumptions that constrain the outputs, the property \mathcal{P} will have the corresponding number of Jacobians, each corresponding to a particular assumption.

The following definitions summarize the above explanations.

Definition 1 *An assumption is a set of constraints. It may directly map onto an action primitive.*

Definition 2 *A property \mathcal{P} is a computational object, corresponding to a physical concept, which includes a set of inputs, $In_{\mathcal{P}}$, and outputs, $Op_{\mathcal{P}}$. Assumptions may dictate the relationship between inputs and outputs. The inputs further carry the information of which property they came from, and whether this property is external or internal. A property \mathcal{P} can furthermore compute its Jacobian, $J_{\mathcal{P}}$, relating its outputs to its inputs.*

3.2 Reasoning Capabilities

The knowledge library described above allows for several reasoning capabilities: connecting various pieces of relevant knowledge in the form of graphs, and numerical calculations including assessment and/or suggestion of tools and parameter optimization.

3.2.1 Knowledge graphs

Given a goal (expressed in the form of a property to act upon with a related constraint to be achieved), the physical reasoning process starts by identifying the physical concepts involved and building knowledge graphs. This is achieved by back-chaining the connected concepts. While doing this back-chaining, we also compute the Jacobian of the goal (output) with respect to the inputs, or the task Jacobian, using the chain rule in a process similar to automatic differentiation [8] and backpropagation. As an illustrative example, if B is an input to A, and C an input to B, then $J_{A \leftarrow C} = J = \frac{\partial A}{\partial C} = \frac{\partial A}{\partial B} \cdot \frac{\partial B}{\partial C}$.

Definition 3 *A knowledge graph is a graph where each node represents a property, the assumption constraining its outputs (if any), and the corresponding Jacobian. A directed edge from node n_1 to node n_2 exists when a subset of the outputs of n_1 ’s property, \mathcal{P}_1 , are inputs to n_2 ’s property, \mathcal{P}_2 , under n_2 ’s assumption.*

¹This is a simplification, as in some cases the volume and density changes due to the dissolution of a solute may not be trivially computed.

In some cases, the output of a physical concept may depend on some assumptions or constraints, whereby different conditions may result in qualitatively and quantitatively different outcomes, usually with a discontinuity in the outputs or the Jacobian (e.g., heat transfer by conduction, convection, or radiation). In this situation, the back-chaining splits into different branches corresponding to the assumption under which the subsequent back-chaining and calculations occur. This is what results in the creation of different knowledge graphs (that can usually be associated with alternative strategies) connecting the relevant physical concepts and keeping tabs of the assumptions that define the relationships and properties’ outputs.

We identify when inputs to a physical concept come from an “external” source or interaction with another object. When this is the case, it is possible to create a virtual object defined by the physical properties or concepts that it contains. This step is relevant when the reasoning algorithm is tasked with identifying and proposing objects in the environment that may serve as adequate tools for a subtask.

We present a pseudo-code, Algo. 1, implementing the steps explained above for generating knowledge graphs from a given goal property to modify. The function ‘*create_and_populate_virtual_tools()*’ identifies when a property is of an external origin, and creates a virtual tool containing that property as well as its children properties (properties feeding into it, as long as they remain of internal origin).

At the end of this step, we have the following: (i) a knowledge graph representing the physical properties or concepts involved, and the assumptions under which they hold; the graph therefore becoming the computational representation of the plan or strategy. (ii) a list of virtual objects or tools that are characterized by specific physical properties and which are going to be useful to the plan. (iii) the explicit task Jacobian relating the goal to the inputs that can be directly acted upon by an agent (e.g., a robot), either by selecting a tool that has the best values for these inputs (e.g., to break a glass panel, it is best to use an object with a high Young’s modulus) or performing an action.

3.2.2 Grounding phase

Given the symbolic physical understanding from the step above (in the form of a strategy’s knowledge graph), and the accompanying list of required abstract tools, the next step is to concretize the reasoning in the context of the environment in which the robot is placed. This means either verifying whether the tools that were proposed in the strategy are adequate, or, if no tools were specified (i.e., the proposed strategy was vague in this regard), identify objects in the environment that may serve as adequate tools.

First, as each abstract object is characterized by a number of properties, we can easily verify whether a tool proposed by the strategy also holds those properties (e.g., electric conductivity).

In the case where no tool was specified, we form a list of suitable items for each abstract tool. Next, we select the best tool among the list of suitable candidates. To do this, we refer to the Jacobian we computed above. It is often possible to assess whether the Jacobian

Algorithm 1 Pseudo-code for knowledge graphs generation

Require: goal property \mathcal{P}_g , knowledge graph \mathcal{G} (empty), a working queue Q_w , (empty), a final queue Q_f (empty)
 first knowledge graph (\mathcal{G}) node n .property $\leftarrow \mathcal{P}_g$
 initialize Q_w with \mathcal{G}
while Q_w is not empty **do**
 $\mathcal{G} \leftarrow Q_w$.pop()
 for each leaf of \mathcal{G} **do**
 node \leftarrow leaf
 if node is not terminal **then** \triangleright i.e., the set of properties feeding into this property is not empty
 if node has assumptions **then**
 duplicate \mathcal{G} for the corresponding number of assumptions
 # (do what's below for each assumption)
 end if
 node.assumption \leftarrow assumption \triangleright only if there is an assumption
 node.J \leftarrow get_Jacobian(node)
 compute task Jacobian, J_{task} , using chain rule
 input_props \leftarrow get_input_props(node) \triangleright Get properties feeding into this node's property
 for input_prop \in input_props **do**
 add node (i.e., new_node.property \leftarrow input_prop) and edge
 create_and_populate_virtual_tools(input_prop)
 end for
 end if
 end for
 if all leaves are terminal **then**
 add \mathcal{G} to the final queue Q_f \triangleright The final queue contains the alternative knowledge graphs
 end if
end while
 return Q_f

3. An Intuitive Physics Reasoner

with respect to a particular input is positive or negative. A positive Jacobian implies that the output increases with the input, and vice versa. It then becomes possible to construct a scoring function to assess which tool is the best. A naive scoring function would be to simply add the value of the input if the Jacobian is positive, and to subtract it if the Jacobian is negative. In other words, for each input for which $J \neq 0$, an object's score $s = s + in \cdot \mathbf{1}_{J>0} - in \cdot \mathbf{1}_{J<0}$ (this assumes that the input in is positive, which is generally the case). It may be possible to normalize these input values if there exists a prior on the range of values one might encounter.

Note that this grounding phase assumes that the agent knows the physical characteristics of the objects around it. This may be achieved via prior learning, internet search, or possibly carrying out appropriate tests to determine whether an object holds a physical property of interest (e.g., electrical conductivity) and qualitatively assess its numerical value relative to the other objects around (e.g., Young's modulus). This latter option will be the subject of future work as we believe our proposed framework is well-suited to accommodate purposeful system identification.

At the end of this phase, the selected strategy graph is grounded in the sense that the properties involved are assigned to actual objects in the environment, the symbols are replaced by their numerical value (if provided by the strategy, or supplied by prior knowledge of objects), and the resulting mathematical expressions are forward-propagated along the strategy's graph. This is akin to a forward simulation.

To illustrate this forward propagation, consider \mathcal{P}_1 with $In_{\mathcal{P}_1} = \{D, C\}$ and $O_{\mathcal{P}_1} = \{E = D \cdot C\}$, \mathcal{P}_2 with $In_{\mathcal{P}_2} = \{A, B\}$ and $O_{\mathcal{P}_2} = \{C = A + B\}$, and finally \mathcal{P}_3 with $In_{\mathcal{P}_3} = \{D_0\}$ and $O_{\mathcal{P}_3} = \{D = D_0\}$. There is an object in the environment that holds property \mathcal{P}_3 and is characterized by $D_0 = 1$ (e.g., the nominal density of a solid), and there is an object in the environment that holds property \mathcal{P}_2 and is characterized by $A = 2$ and $B = 3$. The forward propagation step, applied recursively, ultimately results in: $E = D \cdot C = D_0 \cdot (A + B) = 1 \cdot (2 + 3)$.

3.2.3 Optimization of intermediate parameter values

For a grounded knowledge graph, it is possible that some parameters remain symbolic (i.e., with no corresponding numerical value), for example parameters relating to action primitives. In this case, it is possible to find optimal values for these parameters.

Remember that during graph building, we identified the different assumptions that constrain the tree of connected physical concepts. These constraints relate some of the inputs of the physical property with equality and/or inequality constraints (e.g., $\rho_1 > \rho_2$ where ρ refers to density, or $z_1 > z_2$, where z is a position). In this phase, we compile these constraints and solve an optimization problem, where the parameters that remained symbolic form the decision variables. For example, a strategy that includes dissolving a solute in a liquid will have the following constraints supplied by the dissolving assumption (corresponding to a

dissolving action primitive):

$$\begin{cases} s &> 0 \\ V_L &> 0 \\ m_d &> 0 \\ m_d &\leq M_s \\ m_d &\leq \bar{s}V_L \end{cases} \quad (3.1)$$

where the decision variable is the mass, m_d , to be dissolved, and the inequalities indicate that: the material used as solute needs to be soluble, the liquid’s volume has to be positive, the dissolved mass needs to be positive, not exceed the total mass, M_s , of the material used as solute, and not exceed its saturation limit in the liquid.

Once all the constraints along the strategy’s graph are compiled, the following constrained optimization can be solved:

$$\begin{aligned} \min_x & f(x) \\ \text{s.t.} & \underline{c} \leq c(x) \leq \bar{c} \end{aligned}$$

where x is the vector of decision variables, $f(x)$ is the objective function (we used $f(x) = \frac{1}{2}\|x\|^2$ unless otherwise specified), $c(x)$ are the constraints compiled above. Note that owing to the explicit symbolic expressions, the Jacobian of the constraints can be easily supplied to the optimizer. We used IPOPT [39] to solve the above problem.

3.3 From natural language to a suitable graph

One of the important aspects of the intuitive physics reasoner we aim to build is the possibility to quickly and flexibly generate a scenario of a “simulation”. Since strategy generation is most easily done in natural language for both humans and current generative AI [1], we want our system to take in a strategy expressed in natural language as input. However, as explained above, our system manipulates and reasons on knowledge graphs. We therefore seek a way to go from a natural language description to a knowledge graph of physical concepts representing the strategy.

To do this, we leverage the capability of LLMs to do few-shot learning and generate code [26]. We prompt ChatGPT [1] by giving it a few examples of how to transform a strategy expressed in natural language into a knowledge graph. This then forms the input to the rest of the system described above. We used GPT-3.5.

Turning the natural language description of a strategy into a suitable graph depends on the prompt given to the LLM. In our implementation, the prompt included the following items:

- (i) a set of physical concepts to use if possible; this ensures the LLM-produced graph shares the same concepts as the knowledge library.
- (ii) a set of instructions or constraints on how to generate the graph.
- (iii) one or more examples of a natural language strategy and a corresponding graph.

3. An Intuitive Physics Reasoner

- (iv) the natural language strategy to turn into a graph; the LLM is asked to write a Python program that produces this graph.

An example of the above is given below, with the output graph being presented in Fig. 3.2.

Example prompt: *I'm going to describe a solution to a MacGyver-style physical problem. I want you to turn the steps of the solution into a graph with the following properties:*

- *each node represents a physical concept involved in the solution steps*
- *each node further contains as attribute the name of the object linked to the node's physical concept*
- *a directed edge from node 1 to node 2 exists if the physical concept of node 1 directly affects the physical concept of node 2.*
- *each node can only feed into only one other node.*

Use the following physical concepts where needed: float, liquid_density, solid_density, volume, granular_material, solute, mass.

Here is an example:

I want to make a lump of clay float on water. So the goal is to change the floating property of the clay. The water is in a beaker and has a volume of 300ml. Here are the solution steps:

- *place the clay on top of a piece of styrofoam*
- *place the clay-styrofoam on the water*

Here is the expected output graph:

- *node n1 has name "float", and object attribute "clay"*
- *node n2 has name "solid_density" and has object attribute "clay"*
- *node n3 has name "liquid_density" and has object attribute "water"*
- *node n4 has name "solid_density" and has object attribute "styrofoam"*
- *a directed edge goes from n2 to n1*
- *a directed edge goes from n3 to n1*
- *a directed edge goes from n4 to n2*

Write a python script outputting a similar graph for the following situation:

I want to make a lump of clay float on water. So the goal is to change the floating property of the clay. The water is in a beaker and has a volume of 300ml. Here are the solution steps:

- *dissolve salt in the water*
- *put the clay on the water*

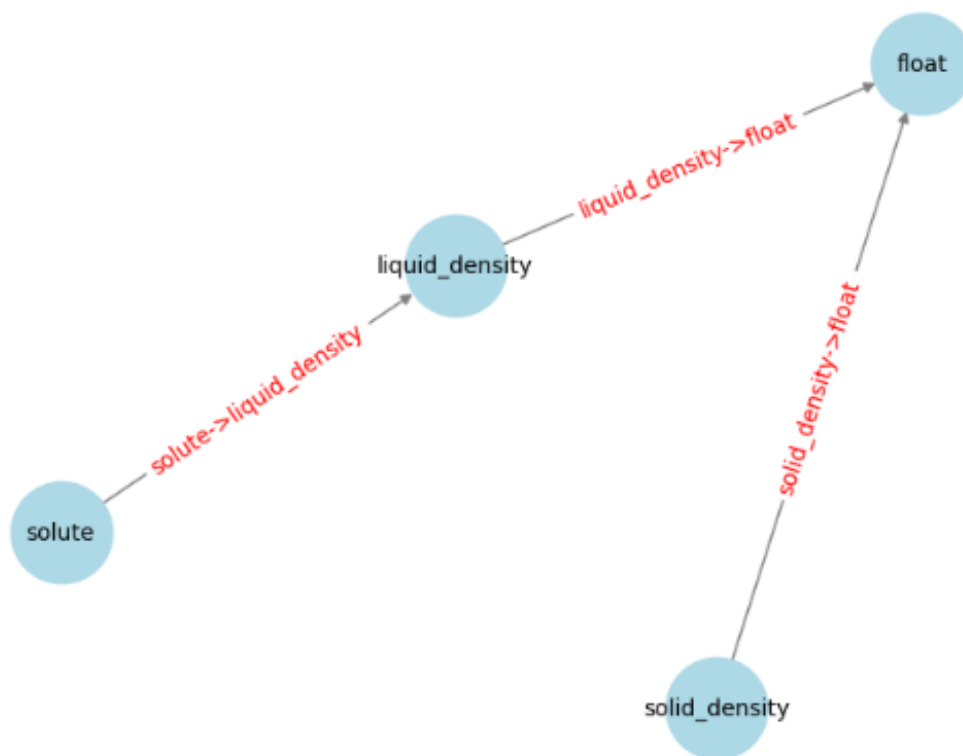


Figure 3.2: Natural language description (see text) turned into a graph by the LLM.

3.3.1 Graph matching problem

Transcribing a strategy described in natural language into a graph is the first step of our intuitive reasoner. The next step in assessing the validity of the strategy is to assess whether the generated graph is consistent with prior knowledge. As a reminder, our reasoner can leverage its knowledge library to generate multiple “ground truth” graphs connecting the physical concepts relevant to the situation. We would therefore like to find the “ground truth” (GT) graph that most likely corresponds to the LLM-generated graph (NL graph). We address this issue by finding nodes that correspond to each other between the NL and GT graphs, and score the GT graphs based on how many NL nodes found a correspondence. Then, from the set of GT graphs with highest score, the one with the least number of additional nodes with respect to the NL graph is selected as the most likely match from prior knowledge (see Alg. 2).

We note that an NL graph may have omissions, i.e., nodes or physical concepts not present in the NL graph but are present in the more complete GT graph.

Conversely, it is also possible for the NL graph to have nodes that find no correspondence in any GT graph, thereby indicating a piece of knowledge that does not exist in the knowledge library. In this case, the extra knowledge is ignored and subsequent numerical verifications are carried out using only prior knowledge. If the system still identifies the strategy as

viable, then the extra piece of knowledge is deemed as non-essential. And in case the system identifies the strategy as non-viable, then it is not immediately possible to identify whether this is because of a lacking knowledge library, or whether the strategy itself was not feasible from the beginning.

3.4 Overall system for strategy assessment

In the previous sections, we described individual system components. We put everything together in this section (see Fig. 3.1).

Given a strategy described in natural language:

1. ChatGPT is asked to transcribe it into a graph and “ground truth” graphs are generated from the knowledge library.
2. A graph correspondence problem is solved to check that the proposed physics are consistent with prior knowledge at a high level. Inconsistent pieces of knowledge (nodes) are flagged and ignored for the subsequent steps.
3. After an optional tool verification step, if all numerical values are given, then forward numerical calculations (akin to a simulation) are carried out while checking for constraints satisfaction. If all constraints are satisfied then the strategy is deemed feasible. If some parameters are not given then an optimization ensuring constraints satisfaction is carried out. If it reaches a solution then the strategy is feasible.

Algorithm 2 Pseudo-code for graph matching

Require: NL graph, GT_graphs

G.root.corresponding_node = NL_graph_root ▷ both graphs start with the same root

num_correspondences = 1

max_corr = len(NL_graph.descendants)+1 ▷ max. number of correspondences

possible_graphs ← list() ▷ initialization

while GT_graphs not empty **do**

 G ← GT_graphs.pop()

 ordered_groups ← get_ordered_groups(G) ▷ organize nodes of G into groups based on depth

for group **in** ordered_groups **do**

for nl_node **in** group **do**

 # identify corresponding nodes based on name (equivalent to a node's property or physical concept)

 potentially_corresponding_nodes ← [x for x in G.descendants if nl.name == x.name and not hasattr(x, "corresponding_node")]

if len(potentially_corresponding_nodes) == 0 **then**

break

else

 # potentially corresponding nodes with lower depth are considered first

 sorted_nodes ← sort_based_on_depth(potentially_corresponding_nodes)

for s_node **in** sorted_nodes **do**

 # Find the ancestors of each node

 anc_corr_nodes = [x.corresponding_node for x in s_node.ancestors if hasattr(x, 'corresponding_node')]

if len(anc_corr_nodes) == 0 **then**

break

else

if nl_node.parent **in** anc_corr_nodes **then**

 s_node.corresponding_node ← nl_node

 num_correspondances += 1

break

end if

end if

end for

end if

end for

end for

if num_correspondances == max_corr **then**

return G

else

 # if there are less correspondences from natural language to prior knowledge, add the graph to a list of possible graphs, till we find a perfect correspondence or exhaust possible prior knowledge

 possible_graphs.append(G) 19

end if

return possible_graphs.popleft() ▷ return most plausible graph ignoring the knowledge added by natural language

end while

3. An Intuitive Physics Reasoner

Chapter 4

Results

To determine the effectiveness of the proposed reasoner, we tested it on 20 scenarios with varying strategies and quantities. These scenarios were related to the problem of making a solid object float on a liquid. Below are some examples of these scenarios. A human ultimately judged whether the reasoner's conclusions (feasibility assessment and optimized quantities) were correct.

Example scenarios:

Each time, the objective is to make a particular object float on some liquid.

Example 1:

- object to float has density $\rho = 2636.1\text{kg/m}^3$,
- liquid of density $\rho = 1259.82\text{kg/m}^3$ and volume 500ml,
- strategy: dissolve a soluble substance of density $\rho = 3349.1\text{kg/m}^3$ and the total quantity available is 2.39kg.

Example 2:

- object to float has density $\rho = 5812.3\text{kg/m}^3$,
- liquid of density $\rho = 1079.34\text{kg/m}^3$,
- strategy: place the object on another available object of density $\rho = 782.48\text{kg/m}^3$, has a mass of 4.14kg.

Example 3:

- object to float has density $\rho = 4938.54\text{kg/m}^3$ and mass 1.5kg,
- liquid of density $\rho = 943.77\text{kg/m}^3$ and volume 500ml,
- strategy: (1) dissolve some quantity of a soluble substance of density $\rho = 6365.84\text{kg/m}^3$ with total available mass of 300g, (2) place the object on another available object of density $\rho = 782.11\text{kg/m}^3$ and mass 2.5kg, (3) place the assembly on the water.

4.1 Overall success rate

We found that our intuitive physics reasoner was correctly able to assess strategies $\sim 84\%$ of the time, whereby failures were mainly due to a faulty transition from the natural language description into a graph.

We will show in what follows examples of success and failure of the graph matching component of our reasoner, followed by examples of the numerical component (including optimizing for unknown quantities).

4.2 Graph matching

The first reasoning goal of the intuitive physics reasoner after transcribing natural language into a graph (NL graph) is to determine whether the latter is consistent with its knowledge of physics. This was done by identifying to which knowledge graph (generated from the knowledge library) it most likely corresponded to. We found that this step, i.e. language-to-graph transcription then graph matching, was the limiting factor in our system. We present some examples below.

Example 1 [failed matching]: In this example, the strategy that was given to the LLM to transcribe into a graph was: (1) mix salt with cesium chloride (CsCl), (2) dissolve the mix in the water, (3) place the clay on the water. The LLM then generated a graph that contained an inaccuracy that could not be adequately resolved by Alg. 2. Indeed, the LLM made it such that the salt-CsCl mix fed its ‘granular_material’ property into the ‘solute’ property of CsCl, and at the same time had a salt ‘solute’ as input. These nodes did not find a correspondence with knowledge graphs generated from the knowledge library and were therefore ignored (as mentioned in Sec. 3.3.1). The best corresponding graph was then found to be simply dissolving a single substance as opposed to a mix (two or more substances). This contrast is shown in Fig. 4.1.

We note that because of this mismatch and depending on the quantities explicitly given in the strategy, the reasoner may ultimately correctly or incorrectly assess the validity of the strategy. For example, suppose the strategy asked for mixing x grams of CsCl with y grams of salt, and that these quantities were sufficient to make the strategy feasible. Because of the graph mismatch however, the reasoner ends up working with a strategy that involves the dissolution of a single substance, in particular here CsCl. If x grams of CsCl alone are insufficient, the reasoner will wrongly conclude that the strategy was not feasible.

Example 2 [matching with correction]: In this example, the strategy that was given to the LLM to transcribe into a graph was: (1) place the clay on top of a piece of styrofoam, (2) mix salt with cesium chloride (CsCl), (3) dissolve the mix in the water, (4) place the clay-styrofoam on the water.

The LLM then generated a graph that contained a slight difference relative to how the graph would have been organized from the knowledge library: a node explicating the mix was created, whereas the algorithm that generates graphs from the knowledge library does

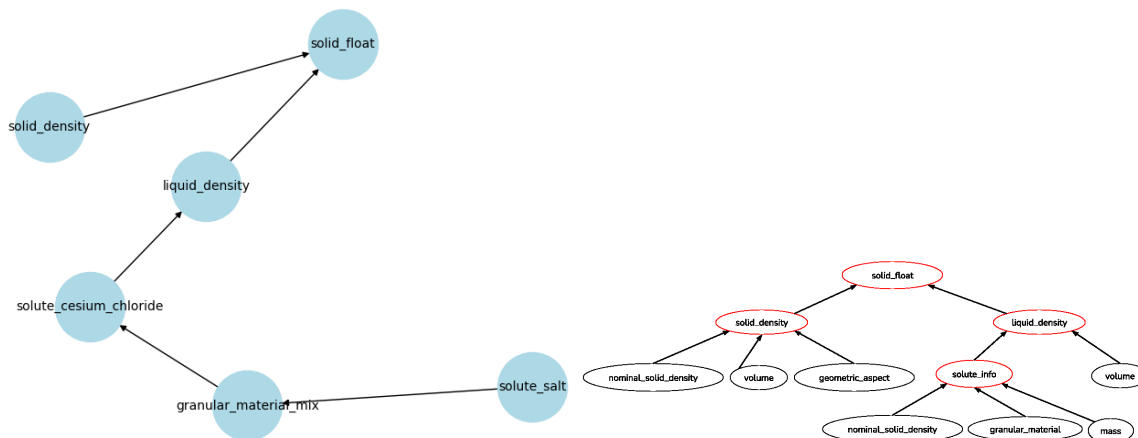


Figure 4.1: Failed graph correspondence. The bad natural language-to-graph transcription (left) was matched to a knowledge graph (right) that did not exactly correspond to the original meaning of the strategy. Corresponding nodes between the two graphs are highlighted in red in the graph on the right.

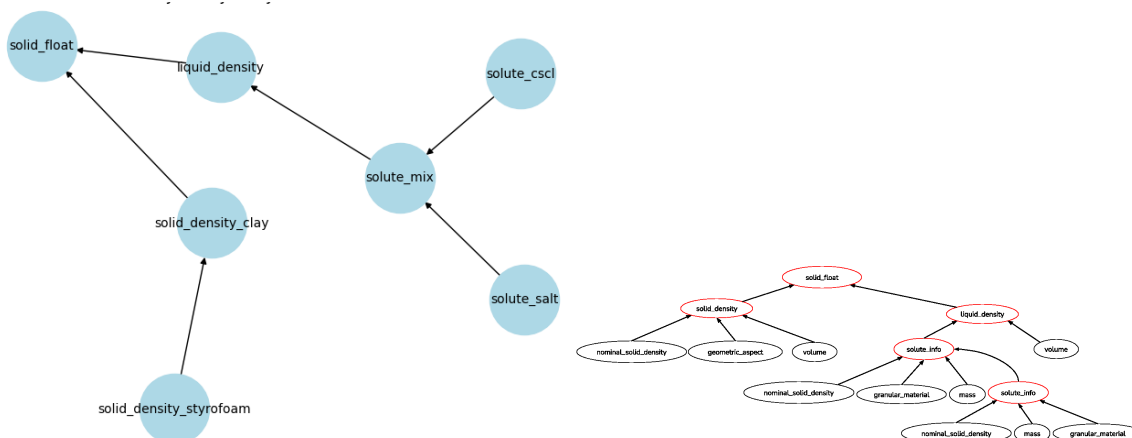


Figure 4.2: Successful graph correspondence. The natural language-to-graph transcription (left) was matched to a knowledge graph (right), indicating that the strategy was consistent with the reasoner’s prior knowledge. Corresponding nodes between the two graphs are highlighted in red in the graph on the right.

not create a new node because the properties of the substance in question simply take a new, effective value. A trivial post-processing step was put in place to catch such cases before calling the graph matching algorithm.

In this case, Alg. 2 was able to correctly identify the corresponding graph from its prior knowledge, thereby indicating that the strategy was consistent with its knowledge of physics. This is shown in Fig. 4.2.

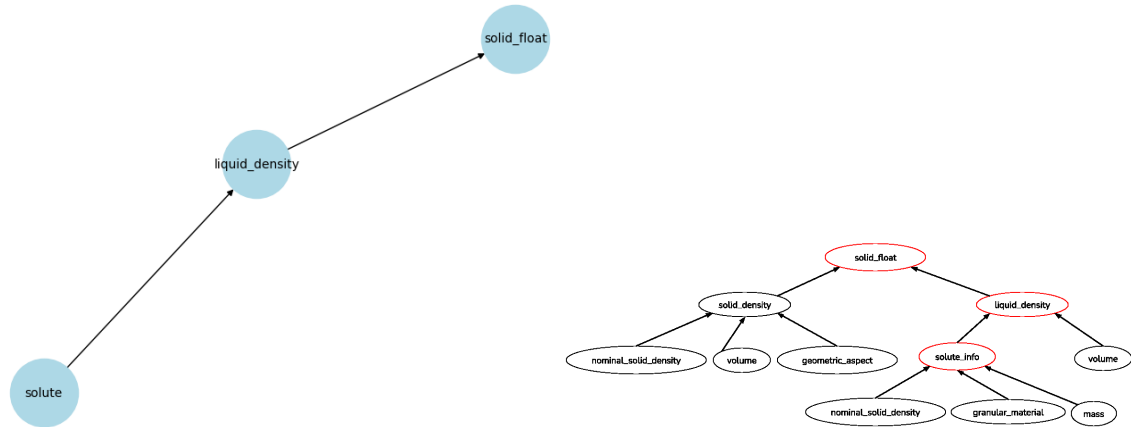


Figure 4.3: Successful graph correspondence. The natural language-to-graph transcription (left) was matched to a knowledge graph (right), indicating that the strategy was consistent with the reasoner’s prior knowledge. Corresponding nodes between the two graphs are highlighted in red in the graph on the right.

Example 3 [matching despite omission]: In this example, the strategy that was given to the LLM to transcribe into a graph was: (1) dissolve cesium chloride (CsCl) in the water, (2) place the clay [object to float] on the water. The LLM then generated a graph that omitted the influence of the clay’s density on its buoyancy. However, Alg. 2 was still able to correctly identify the corresponding graph from its prior knowledge. Indeed, given the other node correspondences, shown in red in Fig. 4.3, and using the heuristic that the simplest graph is the most likely one, the correct graph was identified.

4.3 Numerical reasoning

Given a graph, the next reasoning goal was to establish whether the strategy was numerically feasible, with the option of computing a set of optimal parameters (quantities). In the various scenarios we tested, this was always performed accurately, indicating that the algorithms for forward simulation and optimization worked as intended. We give below some examples, where we note that because the objects and substances’ property values (e.g., densities) were randomly generated for the sake of testing, these substances may not exist in the real world.

Example 1:

- object to float has density $\rho = 2636.1\text{kg/m}^3$,
- liquid of density $\rho = 1259.82\text{kg/m}^3$ and volume 500ml,
- strategy: dissolve a soluble substance of density $\rho = 3349.1\text{kg/m}^3$ and the total quantity available is 2.39kg.

In this example, the strategy expressed in natural language did not specify how much of

the substance to dissolve. The reasoner therefore had to choose that on its own. This is done as a constrained optimization, with the objective function minimizing the quantities used. We note that other objective functions could have been used.

The reasoner found that the constrained optimization was infeasible, and therefore assessed the whole strategy as infeasible. Indeed, it would have taken 3.23kg of that substance, which is greater than the total quantity available.

Example 2:

- object to float has density $\rho = 3025.84\text{kg/m}^3$,
- liquid of density $\rho = 953.63\text{kg/m}^3$ and volume 500ml,
- strategy: dissolve a soluble substance of density $\rho = 6247.68\text{kg/m}^3$ the total quantity available is 3.23kg, and the substance's solubility limit in the liquid (before precipitation) is 2898.7g/L.

In this example, the strategy expressed in natural language did not specify how much of the substance to dissolve. The reasoner therefore had to find on its own. This is done as a constrained optimization, with the objective function minimizing the quantities used.

The reasoner found that the constrained optimization was infeasible, and therefore assessed the whole strategy as infeasible. Interestingly, because we have access to the outputs of the optimization problem, we can identify which constraints were impossible to satisfy. In this particular case, the solubility limit could not be satisfied. Indeed, it would have taken 2kg of that substance, which, while less than than the total quantity available, is greater than the solubility limit of 1.45kg ($= 2.9\text{g/mL} \times 500\text{mL}$).

Example 3:

- object to float has density $\rho = 4938.54\text{kg/m}^3$ and mass 1.5kg,
- liquid of density $\rho = 943.77\text{kg/m}^3$ and volume 500ml,
- strategy: (1) dissolve some quantity of a soluble substance of density $\rho = 6365.84\text{kg/m}^3$ with total available mass of 300g, and solubility limit of 537.8g/L, (2) place the object on another available object of density $\rho = 782.11\text{kg/m}^3$ and mass 2.5kg, (3) place the assembly on the water.

In this example, the strategy expressed in natural language did not specify how much of the substance to dissolve. The reasoner therefore had to find on its own. This is done as a constrained optimization, with the objective function minimizing the quantities used.

The reasoner found that the constrained optimization was feasible, and therefore assessed the whole strategy as infeasible. It correctly computed that we need a minimum of 0.12kg of solute to dissolve. This also indicates that placing the object of interest on top of the lower density object is insufficient on its own, and that the liquid's density still has to be raised.

Example 4:

- object to float has density $\rho = 7490\text{kg/m}^3$ and mass 2kg,
- liquid of density $\rho = 1170\text{kg/m}^3$ and volume 500ml,
- strategy: dissolve 500g of a substance of density $\rho = 7490.55\text{kg/m}^3$ with total available mass of 760g, and is not soluble.

4. Results

Performing a forward simulation with constraints verification, the reasoner found that at least one constraint was not satisfied. In particular here, because the substance is not soluble, an error is thrown indicating constraint violation. The strategy as therefore deemed infeasible.

Example 5: Towards robot implementability

As explained in Sec. 3.1, the outputs of properties (physical concepts) may be subject to constraints. In particular, it is possible to conceive of constraints that relate to robotic actions. For example, the effective density of a material may be altered if it is reshaped (e.g., to include pockets of air). In such cases, we may include constraints that are imposed by an action that is performed by a human or a machine and which affects the physical properties of objects.

We show in this example such a scenario. Notably, we make use of additional physical concepts that were not introduced in the preceding examples, for instance the fracture and yield strengths of a material.

- object to float has density $\rho = 1500\text{g/L}$ and mass 350g, yield strength of 20kPa, and fracture strength of 1MPa,
- liquid of density $\rho = 1000\text{g/L}$ and volume 3L,
- strategy: reshape the object by hollowing it out such that its bounding box volume is 440mL, the hollowed out volume is 207mL. The surface on which to apply force is 46cm^2 , and the robot's maximum force is 50N.

Performing a forward simulation with constraints verification, the reasoner found that at least one constraint was not satisfied. In particular here, the force to exert to reach the plastic regime (above yield strength) exceeds the robot's motor limits. Hence, the strategy is deemed infeasible. Had the yield strength been 10kPa instead, this strategy would have been feasible.

Similarly, a condition imposing that the yield strength be smaller than the fracture strength is also verified. Were it not the case, the material would have fractured before undergoing permanent deformation.

We note here the reasoner did not account for whether this material reshaping causes any dimensions to become impossibly thin, or whether the robot had the accuracy of performing such an action. The reasoner is limited by its knowledge library: the less knowledge it has, the rougher the feasibility assessment will be. However, we also note that our original goal was not to assess strategies with perfect accuracy, but rather to quickly determine whether a strategy was worth considering and allotting further resources to its planning and execution.

Chapter 5

Discussion

In this work we have presented a framework to assess strategies expressed in natural language. Describing potential solutions to physically-rich problems in natural language is perhaps the easiest way for human operators. The recent advances in generative AI enable natural language specification of robot plans as well. However, there are no guarantees as to the physical, logical, and mathematical consistency of such strategies, especially if they were generated by an LLM. We believe the ability to quickly determine whether a strategy is worth considering and allocating further resources to its execution is an important faculty robots should have, especially in mission-critical situations that require quick thinking and where failing multiple times is not an option. Because of a lack of a suitably quick and flexible simulator, problems that involve a rich set of physics beyond rigid body mechanics and geometric considerations are generally under-explored in robotics.

The framework we have presented here is our first attempt to handle this. Critical to our endeavor was the building of a simple physics knowledge library, forming a pool of human-vetted prior knowledge with which strategies to novel situations can be scrutinized. At a high level, building a knowledge library is similar to building a physics engine, except that it is more geared towards (symbolic and numerical) reasoning rather than just forward simulation; which explains the explicit encoding of knowledge and reliance on graphs.

The physical knowledge library we presented represents physical concepts or properties as symbolic objects that have inputs, outputs, functions relating inputs to outputs, and possibly constraints on those outputs (essentially defining the domain of applicability of the functions). In building this library, we have taken advantage of the fact that physics are amenable to modularity. Many phenomena share the same underlying physical causes, but simply have different boundary conditions, are taken at different limits, or have different combinations of causes.

An important feature of our knowledge library and knowledge encoding is that they can accommodate diverse domains of physics and chemistry at once (e.g., stress, buoyancy, solubility, ...), so long as relationships can be functionalized and organized into properties. We acknowledge however that this knowledge encoding may not be suitable to all domains of

physics and chemistry, especially when geometry becomes an important factor. A versatile way of encoding geometric considerations has remained elusive.

That said, initial work in tool use for object retrieval (not shown here), similar to [37], showed promise. The task was to use a tool to reach an otherwise unreachable object and pull it towards a desired location. In this chiefly geometric problem, the geometric knowledge of objects, in particular the tools to be used to retrieve an unreachable object, was encoded as a parameterization $\gamma(t)$ of the tangent, $\Gamma \subset \mathbb{R}^n$ (i.e., contour) of the object, along with the matching exterior normal $\sigma(t)$. This was explored for 2D shapes, so $n = 2$, but one can envision the extension to 3D shapes. In particular, Γ can be a piecewise continuous curve, $\Gamma = \cup_k \Gamma_k$, each piece parameterized by $\gamma_k : [a, b] \rightarrow \Gamma_k \subset \mathbb{R}^n$.

Given a tool and robot kinematic limits, we were able to optimize for the robot-tool contact point, tool-object (to be retrieved) contact point, tool orientation, and direction of motion. The main component of the objective function was to maximize $\sigma(t_c) \cdot d$, where $\sigma(t_c)$ is the exterior normal at the tool-object contact point, and d the normalized direction from the object to the desired location. The optimization was carried out continuously as the agent moved the object in the direction of $\sigma(t_c)$.

Proper use of geometric features, such as concavities, emerged as a result of this method, although inconsistently so.

While we tested our framework on scenarios involving multiple domains of physics, relationships and equations remained simple and reasoning was performed on equilibria. Flows and time evolutions typical of dynamic phenomena and expressed by (ordinary or partial) differential equations were not explored. While the ability to reason in terms of quasi-statics and equilibria is relevant and important to assess various strategies (e.g., engineers often use such back-of-the-envelope reasoning to quickly assess the merits of claims), the ability to reason in terms of flows and dynamic evolutions is likely to be important for a subset of problems. This therefore constitutes an area of future research.

Additional work is also needed to implement this on a robot. As mentioned at the end of Chap. 4, constraints on the outputs of properties (physical concepts) may be related to actions from humans or robots (e.g., action primitives). These actions may be parameterizable, and since they are part of the constraints, the values of these parameters can also be optimized during the numerical reasoning component of our intuitive reasoner. This may open the door towards implementation on an actual robot.

Of note, we remark that our framework would allow a robot to reason about objects at different levels: since objects are viewed as collections of properties, the utility or affordance of an object is determined by the needs of the robot. A metal bar for instance can thus be used to transmit force (in which case it is simply viewed as a generic rigid object), or to conduct electricity, or to conduct heat. On a robot, the values of these properties may be obtained using large vision-language models to detect and recognize objects, and answer questions about them [11, 33].

Additionally, the introduction in our framework of a Jacobian allows for the possibility to perform a form of qualitative reasoning [14, 22]. While not shown here, it is possible to perform a discrete optimization in order to select the best tool for a task among

candidate objects. Indeed, given the explicit encoding of knowledge, we have access to the Jacobian of physical concepts with respect to their inputs (other physical concepts), allowing one to score candidate tools based on the values of their relevant physical properties.

A great number of works on reasoning have previously been proposed with impressive results. However, these are most often concerned with rigid-body, geometric problems, and often a focus on motion optimization [4, 10, 12, 23, 27, 37, 42, 43]. In this work, we focused on problems with a richer set of physics, involving liquids, aspects of continuum mechanics, granular materials, solubility, which are often hard to accurately simulate. Problems involving thermodynamics are currently under exploration, and we present in this regard, in Appendix A, a number of scenarios we believe are interesting to investigate.

The lack of a general physics engine that can be used to flexibly reason about rich physics, balancing accuracy and computational expense, may be part of the reason why previous work has mainly focused on rigid body geometric problems. Nvidia’s Omniverse [2] is an impressive physics engine, but requires time and expertise to set a simulation up. In our work, we leveraged the capability of LLMs to produce code to flexibly encode a natural language description of a strategy into a computationally-useful representation that forms the input of our physics reasoner. This is one of the key elements of our work. Our goal was to intuitively reason about complex problems, whose solutions need to be found relatively quickly and are difficult to express in mathematical or logical expression. We therefore imposed the design criterion of having an easy entry point (input) to our reasoner, hence choosing to describe solutions in natural language as an input to the reasoner. This required the finding of a suitable interface to translate this natural language into a computational input (graph) to the physics reasoner.

Interestingly, owing to the versatility of LLMs and their generalization power, the LLM we used (GPT-3.5) was able, in most cases, to adequately identify the physical concepts to use (from the set given to it) when building a graph despite a very limited example (see Sec. 3.3). That said, we empirically noticed that the LLM’s output was sensitive to the prompt, in particular the graph building rules that were given in the prompt. More thorough prompt engineering may have yielded better results.

It is clear that LLMs are powerful tools with strong reasoning capabilities and can provide high-level strategies [4, 12, 13]. They are however prone to “hallucination”, providing incorrect information, a feature that may be unacceptable for certain applications (e.g., space exploration). For example, when asked about strategies to make something float, ChatGPT [1] suggested reducing the weight of the object (as in removing material), which is wrong. Even when pointing out the error, it still insisted on this strategy. In order to fix such failures, cognitive models like this work, and model-building capabilities [24] will prove to be complementary to LLMs.

Importantly, we believe that establishing a feedback loop from the intuitive reasoner to the LLM may not only further improve the accuracy results (i.e., the veracity of the feasibility assessment) but also to actively alter and refine strategies. In particular, violated constraints, omitted knowledge, superfluous steps, and related knowledge may all be pointed out to the LLM, which can then take this information, refine the strategy, then transcribe it into a graph for another round of verification.

5. Discussion

The notion of a quick “intuitive” physics engine has been previously applied to obtain excellent results on tool use and physical reasoning [5], although again for rigid body geometric problems. The authors in [15] also tackle the problem of getting robots to reason about a rich set of physics and propose a rapid simulator that accounts for a number of physical and chemical phenomena. They use the simulator to train an RL agent to build a target world (given as an image), destroy it, or manipulate certain elements (e.g., water or sand) to target locations. The above physics engines [2, 5, 15] can be considered as “black-box” simulators where one can only set some initial conditions and observe the evolution of outcomes. In contrast, because our intuitive physics reasoner has direct, explicit access to its knowledge base, it can offer insights and be interrogated differently. Where a sampling-based approach is required to find local optima or bifurcations in qualitative behavior when using a black-box simulator, these insights can be directly calculated and offered up by our explicit physics engine. Examples of this include calculating how much of a substance to be dissolved in a liquid is needed, and whether this amount will lead to precipitation and how it impacts the overall strategy.

Finally, we do not argue that knowledge of physics alone is sufficient to make robots useful in all situations of everyday life. Additional cognitive processes are needed, such as remembering where the plates are stored, or inferring that milk (perishable) should be stored in the fridge (cool storage). In [35], the authors propose an extensive semantic knowledge library (KnowRob) with an ontology that permits such types of inference. It is used to obtain impressive results that outline the holistic nature of their approach: perception, reasoning, and actions are interleaved via this knowledge library. In essence, the latter is leveraged to ground pre-programmed abstract plans.

Chapter 6

Conclusion

In this work, we argued for the need for a flexible intuitive physics engine or reasoner for rich physics to quickly determine strategy feasibility before allocating further computational resources for planning (e.g., trajectory planning). We took advantage of LLMs' versatility to make our system's input easy and convenient: a natural language description of strategies, which is turned into a computationally-usable form. This was then the input to our reasoner which uses human knowledge and machine reasoning to overcome LLM caveats (hallucinations, unnecessary steps, logical and mathematical errors), and can identify physical consistency (relative to prior knowledge), assess numerical feasibility, as well as determine parameter values when the natural language strategies do not specify them. We demonstrated the system's capability on a number of problems involving a diverse set of physical and chemical concepts, with favorable outcomes in terms of assessment accuracy.

6. Conclusion

Appendix A

Scenarios and Strategies for Thermodynamics

Thermodynamics provide an interesting set of physics that are relevant to day-to-day, engineering, and survival problems. They are also underexplored in the domain of robotic problem-solving. As they are currently under exploration, thermodynamic problems did not make into this thesis document. However, we wish to provide here a number of problem scenarios and strategies that we believe are interesting and could provide a starting point for other researchers to use. The strategies were proposed by ChatGPT (GPT-3.5) [1] and are thus subject to inconsistencies.

A.1 Improvising a desalination system

Problem: You find yourself stranded on a deserted island with no fresh drinking water. However, you have access to a limited supply of compressed gas (e.g., a canister of compressed air), a few hollow bamboo stems, and some basic tools. Your objective is to devise a makeshift desalination system using these items and your knowledge of physics and fluid mechanics to obtain potable water.

Objective: Your objective is to create an improvised desalination system to desalinate seawater using the available items.

Strategy:

1. Find a suitable location near the shoreline where you can partially submerge the bamboo stems in seawater.
2. Use the tools to create a seal at one end of the bamboo stem to prevent water from entering.
3. Insert the other end of the bamboo stem into the compressed gas canister, creating a sealed connection.

A. Scenarios and Strategies for Thermodynamics

4. Apply pressure to the canister, releasing a controlled burst of compressed gas into the bamboo stem.
5. As the compressed gas expands into the bamboo stem, it undergoes an adiabatic process, causing its temperature to decrease.
6. Submerge the sealed end of the bamboo stem into the seawater, ensuring that it remains below the water surface.
7. As the gas expands inside the bamboo stem, it creates a pressure differential that draws seawater into the stem through small openings or pores in the bamboo.
8. Close the bamboo stem, trapping the seawater inside.
9. Repeat the process of releasing bursts of compressed gas into the bamboo stem, allowing seawater to enter and creating a series of alternating pressure differentials.
10. The pressure differentials created by the expanding gas force the seawater through the small openings or pores in the bamboo, creating a fine mist or spray.
11. Collect the mist or spray in a clean container or surface.
12. The mist or spray undergoes the process of condensation, where the water vapor cools and converts back into liquid form, leaving behind the salt and impurities.
13. Collect the condensed water, which is now desalinated, in a separate container for drinking or other purposes.

A.2 Improvising a cooling system

Problem: You find yourself in a remote location with no access to electricity, but you need to cool down perishable food items or medication that require refrigeration. You have a small insulated box, water, a cloth or towel, and a can of compressed gas (e.g., compressed air). Your objective is to devise a makeshift cooling system using these items and your knowledge of physics and thermodynamics.

Objective: Your objective is to create an improvised cooling system to cool down the interior of the insulated box using the available items.

Strategy:

1. Wet the cloth or towel with water, making sure it is thoroughly damp but not dripping.
2. Wrap the damp cloth or towel around the items you want to cool, ensuring good contact between the cloth and the items.
3. Place the wrapped items inside the insulated box.
4. Use the compressed gas to release a controlled burst of gas into the box through a small opening or puncture.
5. As the compressed gas expands inside the box, it undergoes an adiabatic process, causing its temperature to decrease.

6. Close the opening or puncture in the box to retain the expanded gas inside.
7. The cool gas inside the box will absorb heat from the surroundings, including the items and the damp cloth.
8. Evaporative cooling will occur as the water on the cloth evaporates, drawing heat from the items and further cooling them down.
9. Repeat the process of releasing bursts of compressed gas into the box periodically to maintain a steady cooling effect.
10. If available, placing the box in a shaded or breezy area can enhance the cooling effect by facilitating heat transfer and increasing evaporation.
11. Monitor the temperature inside the box and adjust the frequency of gas releases as needed to maintain the desired cooling.

A.3 Improvising a water heating system

Problem: You are on a camping trip and need to heat water for cooking or making hot beverages. Unfortunately, you don't have access to a stove or any conventional heating equipment. However, you have a metal container, a small piece of glass, and a can of compressed gas (e.g., compressed air). Your objective is to devise a makeshift water heating system using these items and your knowledge of physics and thermodynamics.

Objective: Your objective is to create an improvised water heating system to heat water using the available items.

Strategy:

1. Fill the metal container with the water you want to heat.
2. Place the small piece of glass at the bottom of the container, creating a reflective surface.
3. Position the container in a sunny area, ensuring that the glass is facing the sun and the reflective surface is directing sunlight onto the water.
4. Use the compressed gas to release a controlled burst of gas into the container through a small opening or puncture on the side.
5. As the compressed gas expands, it undergoes an adiabatic process, causing its temperature to decrease.
6. Place the container in a safe location and wait for a few minutes.
7. The cool gas inside the container will absorb heat from the surroundings, including the water, through heat transfer.
8. Repeat the process of releasing bursts of compressed gas into the container periodically, maintaining a steady flow of cooled gas.
9. As the cool gas absorbs heat from the water, the water's temperature gradually

increases over time.

10. Continue the process until the water reaches the desired temperature for cooking or making hot beverages.

A.4 Getting a fire started

Problem: You are on a deserted tropical island, you need to get a fire started. You have damp twigs that won't easily catch fire, you have a plastic bottle with liquid gasoline, and additional trash items brought in by the currents. Your maximum force is 100N.

Strategy:

1. Make a syringe-like contraption.
2. Put some gasoline inside.
3. Compress quickly.

A.5 Improvising an air conditioning system

Problem: You are trapped in a room with a malfunctioning air conditioning system during a heatwave. The temperature inside the room is rapidly rising, and you need to cool it down using whatever items are available. You have access to a small canister of compressed gas (e.g., compressed air), a rubber balloon, a drinking straw, and a roll of aluminum foil. Your objective is to devise a makeshift cooling system using these items and the concepts of thermodynamics.

Objective: Your objective is to cool down the room by creating an improvised air conditioning system using the provided items.

Strategy:

1. Use the drinking straw to inflate the rubber balloon partially with the compressed gas. Pinch the opening of the balloon to keep the gas inside.
2. Wrap the balloon with a layer of aluminum foil, making sure to leave a small opening for the drinking straw.
3. Position the balloon in front of a fan or near a window where air can flow.
4. Insert the drinking straw through the aluminum foil into the balloon to release a controlled amount of compressed gas into the balloon.
5. As the compressed gas expands inside the balloon, it cools down due to the adiabatic expansion process. This cooling effect is a result of the gas doing work against the external pressure.
6. The cooled air inside the balloon can be directed towards the room, providing a localized cooling effect.

Bibliography

- [1] Openai, chatgpt. <https://openai.com/blog/chatgpt>. Accessed: 2023-04-25. 1, 3, 3.3, 5, A
- [2] Nvidia, omniverse. <https://www.nvidia.com/en-us/omniverse/>. Accessed: 2023-06-16. 5
- [3] Eric W Aboaf, Christopher G Atkeson, and David J Reinkensmeyer. Task-level robot learning: Ball throwing. 1987. 2.3
- [4] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 1, 2.4, 5
- [5] Kelsey R Allen, Kevin A Smith, and Joshua B Tenenbaum. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences*, 117(47):29302–29310, 2020. 1, 2.1, 5
- [6] Christopher G Atkeson. Memory-based techniques for task-level learning in robots and smart machines. In *1990 American Control Conference*, pages 2815–2820. IEEE, 1990. 2.3
- [7] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013. 2.1
- [8] A Güneş Baydin, BA Pearlmutter, A Andreyevich Radul, and J Mark Siskind. Automatic differentiation in machine learning: A survey. arxiv preprint arxiv: 150205767. 2015. 3.2.1
- [9] AMP von Bayern, Samara Danel, AMI Auersperg, Berenika Mioduszewska, and A Kacelnik. Compound tool construction by new caledonian crows. *Scientific reports*, 8(1):15676, 2018. 2.1, 2.4
- [10] Jake Brawer, Meiying Qin, and Brian Scassellati. A causal approach to tool affordance learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8394–8399. IEEE, 2020. 2.2, 5
- [11] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852*,

2021. [5](#)
- [12] Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. Task and motion planning with large language models for object rearrangement. *arXiv preprint arXiv:2303.06247*, 2023. [2.4](#), [5](#)
 - [13] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023. [2.4](#), [5](#)
 - [14] Kenneth D Forbus. Qualitative physics: Past, present, and future. In *Exploring artificial intelligence*, pages 239–296. Elsevier, 1988. [2.3](#), [5](#)
 - [15] Kevin Frans and Phillip Isola. Powderworld: A platform for understanding generalization via rich task distributions. *arXiv preprint arXiv:2211.13051*, 2022. [5](#)
 - [16] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021. [2.2](#)
 - [17] Jeff Hawkins. *A thousand brains: A new theory of intelligence*. Basic Books, 2021. [2.1](#)
 - [18] Sarah A Jelbert, Alex H Taylor, Lucy G Cheke, Nicola S Clayton, and Russell D Gray. Using the aesop’s fable paradigm to investigate causal understanding of water displacement by new caledonian crows. *PloS one*, 9(3):e92895, 2014. [2.1](#)
 - [19] Hadar Karmazyn-Raz and Linda B Smith. Sampling statistics are like story creation: a network analysis of parent–toddler exploratory play. *Philosophical Transactions of the Royal Society B*, 378(1870):20210358, 2023. [2.1](#)
 - [20] Beomjoon Kim, Kyungjae Lee, Sungbin Lim, Leslie Kaelbling, and Tomás Lozano-Pérez. Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9916–9924, 2020. [2.2](#)
 - [21] George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014. [2.2](#), [2.3](#)
 - [22] Benjamin Kuipers. *Qualitative reasoning: modeling and simulation with incomplete knowledge*. MIT press, 1994. [2.3](#), [5](#)
 - [23] Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 33(14):1726–1747, 2014. [2.2](#), [5](#)
 - [24] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40: e253, 2017. [2.1](#), [5](#)
 - [25] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *arXiv preprint*

- arXiv:2206.14858*, 2022. [2.4](#)
- [26] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022. [3.3](#)
- [27] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3684–3691. IEEE, 2014. [2.2](#), [5](#)
- [28] Lydia V Luncz, Tiago Falótico, Alejandra Pascual-Garrido, Clara Corat, Hannah Mosley, and Michael Haslam. Wild capuchin monkeys adjust stone tools according to changing nut properties. *Scientific reports*, 6(1):33089, 2016. [2.1](#)
- [29] Andre Makovsky, Peter Ilott, and Jim Taylor. Mars science laboratory telecommunications system design. *DESCANSO Design and Performance Summary Series, Article*, 14:32–35, 2009. [1](#)
- [30] François Osiurak and Arnaud Badets. Tool use and affordance: Manipulation-based versus reasoning-based approaches. *Psychological review*, 123(5):534, 2016. [2.1](#)
- [31] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.
- [32] Michael W Reimann, Max Nolte, Martina Scolamiero, Katharine Turner, Rodrigo Perin, Giuseppe Chindemi, Paweł Dłotko, Ran Levi, Kathryn Hess, and Henry Markram. Cliques of neurons bound into cavities provide a missing link between structure and function. *Frontiers in computational neuroscience*, page 48, 2017. [2.1](#)
- [33] Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2210.05663*, 2022. [5](#)
- [34] Linda B Smith and Hadar Karmazyn-Raz. Episodes of experience and generative intelligence. *Trends in Cognitive Sciences*, 2022. [2.1](#)
- [35] Moritz Tenorth and Michael Beetz. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research*, 32(5): 566–590, 2013. [2.3](#), [5](#)
- [36] Wil Thomason and Ross A Knepper. A unified sampling-based approach to integrated task and motion planning. In *Robotics Research: The 19th International Symposium ISRR*, pages 773–788. Springer, 2022. [2.2](#)
- [37] Marc A Toussaint, Kelsey Rebecca Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. 2018. [1](#), [2.2](#), [5](#)
- [38] Tomer D Ullman and Joshua B Tenenbaum. Bayesian models of conceptual development: Learning as building models of the world. *Annual Review of Developmental Psychology*, 2:533–558, 2020. [2.1](#), [2.4](#)

- [39] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006. [3.2.3](#)
- [40] Asim Waqas, Hamza Farooq, Nidhal C Bouaynaya, and Ghulam Rasool. Exploring robust architectures for deep artificial neural networks. *Communications Engineering*, 1(1):46, 2022. [2.1](#)
- [41] James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.
- [42] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *arXiv preprint arXiv:1904.05538*, 2019. [1](#), [2.2](#), [5](#)
- [43] Danfei Xu, Ajay Mandlekar, Roberto Martín-Martín, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Deep affordance foresight: Planning through what can be done in the future. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6206–6213. IEEE, 2021. [2.2](#), [5](#)
- [44] Akihiko Yamaguchi and Christopher G Atkeson. Differential dynamic programming for graph-structured dynamical systems: Generalization of pouring behavior with different skills. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 1029–1036. IEEE, 2016. [2.3](#)
- [45] Akihiko Yamaguchi and Christopher G Atkeson. Model-based reinforcement learning with neural networks on hierarchical dynamic system. In *the Workshop on Deep Reinforcement Learning: Frontiers and Challenges in the 25th International Joint Conference on Artificial Intelligence (IJCAI2016)*, 2016. [2.3](#)
- [46] Lei Yuan, Violet Xiang, David Crandall, and Linda Smith. Learning the generative principles of a symbol system from limited examples. *Cognition*, 200:104243, 2020. [2.1](#)
- [47] Lei Yuan, Richard Prather, Kelly Mix, and Linda Smith. The first step to learning place value: A role for physical models? In *Frontiers in Education*, volume 6, page 683424. Frontiers Media SA, 2021. [2.1](#)
- [48] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.