Creativity, Compositionality, and Common Sense in Human Goal Generation

Guy Davidson
guy.davidson@nyu.edu
Center for Data Science
New York University

Todd M. Gureckis
todd.gureckis@nyu.edu
Department of Psychology
New York University

Brenden M. Lake
brenden@nyu.edu
Department of Psychology and
Center for Data Science
New York University

Abstract

Inspired by notions of intrinsic motivation (Schmidhuber, 2010) and play as proposing and solving arbitrary problems (Chu and Schulz, 2020b), we report initial progress toward computational modeling of playful goal generation. We create an embodied, 3D environment resembling a child's room, and ask study participants to play in the environment and then create a scorable game. We propose to model games using a domain-specific language, which represents each game as a computer program. These programs act as reward-generating functions, mapping states visited by an agent as they play a game to the score they should receive. We then analyze our corpus of program representations to highlight four key aspects of human games that would contribute to constructing effective computational models of game generation: creativity, compositionality, common sense, and context sensitivity.

Keywords: goals, play, intrinsic motivation, exploration, program synthesis, domain-specific language, program induction

Introduction

People are internally motivated to establish their own goals and then act to fulfill them (see Schmidhuber, 2010, on intrinsic motivation). This behavior is readily evident in children – Figure 1A depicts a child playing in a room full of toys, where they may continue stacking blocks in a tower, or perhaps throw objects at the tower in attempt to knock it over. Chu and Schulz (2020b) characterize these types of play as creating and solving arbitrary and unusual problems, hypothesizing such play aids in generating new ideas and exploring novel search spaces. However, the computational basis of this ability remains poorly understood. In this paper, we take the first steps towards developing a computational model of creative, playful goal generation for the purpose of both understanding these human abilities and driving exploration in AI systems.

Our modeling is informed by a novel data set we collected of humans creating games for themselves to play. Inspired by work on structured imagination (Ward, 1994) and conceptual combination (Murphy, 1988), we sought an experimental approach that allows us to capture the rich, playful, and creative nature of how children and adults can generate goals in everyday scenarios. Using AI2-THOR (Kolve et al., 2017)—a rich, embodied, 3D platform—we set up a children's bedroom-like environment full of blocks, balls, and other objects (Figure 1B). We invited (adult) participants to explore the room environment through an interactive, web-based platform, and then describe (in natural language) a game that could be played in the room (Figure 1C). We posed few constraints on the games our participants created, in an attempt to simulate the scenario of a child playing with a collection of available objects.

Our modeling approach operationalizes games as reward-generating functions. Here we model the underlying structure of these functions, rather than automatically generating them

Figure 1: Overview. Inspired by playful goal generation **(A)**, we designed an interactive environment **(B)**, used to collect games from our participants in natural language **(C)**, enabling the creation of a novel domain-specific language for games **(D)**.

or finding policies to solve them (both focuses of future work). Formally, we represent games as short computer programs in a domain-specific language (DSL), which receive as input a series of environment states, generated as an agent is playing a game, and return as an output the score the person receives—as a person acting as a scorekeeper would do (Figure 1D). This approach relates to a growing body of work in cognitive science that considers program-like knowledge representation (Rule et al., 2020). We constructed our DSL using an empirical, bottom-up approach: we studied the games participants created, and designed a representation that captured as much as possible of the semantics of games in our dataset. In the longer term, we hope the structured representation in the DSL (as opposed to unstructured natural language) facilitates generating new goals and automatically evaluating how well agents (human and artificial) can learn to play games.

In the shorter term, these structured representations are amenable to statistical analyses, revealing several key attributes of human cognition in the context of proposing new games. For one, we observed a high degree of reuse and recombination (i.e., compositionality) across games created by independent participants. In addition, games reflected a substantial degree of creativity (high variability across subjects) and relied heavily on common sense via intuitive physics (throwing balls and stacking blocks, rather than vice versa). Participants also demonstrate context sensitivity, creating games using salient objects in different rooms, rather than trivial games that would work regardless of the affordances of the environment.

Related Work

We build on a tradition of studying children's play and exploratory behavior (Chu and Schulz, 2020b; Sim and Xu,

2017; Buchsbaum et al., 2012; Siegel et al., 2021; Chu and Schulz, 2020a; Pelz and Kidd, 2020), but in a more open-ended environment than most experiments (and instead with adults). Our ultimate aim is to provide goal representations that drive exploratory behavior in unrewarded environments. It has been long recognized that unstructured play aids children's learning. For example, exploration is highlighted as one of a variety of proposed developmental roles for play (see Chu and Schulz, 2020b, for a review). Siegel et al. (2021) directly study exploratory play, finding that children play longer when adjudicating between less discriminable hypotheses. Chu and Schulz (2020a) study children performing the same goals during and outside of play, and find that they take on costs during play that they avoid when more directly fulfilling the goal. They remark that for children, play involves "setting up problems where they incur needless costs to achieve arbitrary rewards."

How can we formalize this notion of self-constructed problems? In reinforcement learning (RL), the reward function specifies the external, environmentally-defined signal the agent attempts to maximize: formally, it maps each state and each action to the scalar reward the agent experiences for taking an action at a particular state. Agent-generated goals are used to supplement the environment's reward function and provide an alternative signal that can guide exploration (see Colas et al., 2020b; we will omit discussion of other exploration approaches, see Weng, 2020 for a review). Most current approaches generate goals that can be evaluated on only a single world state, and as such, fall far short of the richness of real-world goals (Colas et al., 2020b, section 7.1). Some approaches consider goals comprised of (x, y) positions in the environment (e.g., Florensa et al., 2018; Campero et al., 2021), while others construct richer goals by sampling from limited grammars (Colas et al., 2020a; Akakzia et al., 2021). While these grammars enable some compositionality, they express only a small number of conditions on a single state, without capturing the temporally-extended nature of human goals that we study. We represent temporally-extended goals as programs operating over sequences of world states, not unlike reward machines (Toro Icarte et al., 2022), using a domain-specific language designed by modeling human games.

Data Collection

In our task, participants were virtually placed in an unfamiliar room full of objects and asked to propose a single-player game to be played in the room. We use game generation as a means of studying how children, or adults playing alongside children, can produce playful goals, in the context of a specific set of objects and affordances. The concreteness of game-based goals, as opposed to more abstract goals, offers the advantage of a more tractable setting for formal specification.

Environment. We built our experiment using the AI2-THOR (Kolve et al., 2017) framework, which offers an embodied, first-person environment including various room scenes and household objects (Figure 1B). We modified a bedroom scene by adding some toys, such as various balls and blocks,

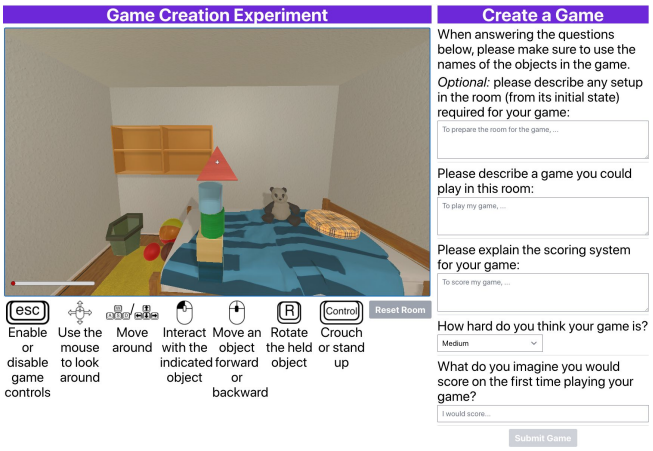

Figure 2: Interactive experiment platform. The main part of the screen presents the AI2-THOR-based experiment room. Below it, we depict the available controls. To the right, we show the text prompts for creating a new game. Our experiment is available online here.

in place of some of the furniture to offer more playful objects. We created three versions of this room, varying both the variety and quantity of objects available, to examine how an environment's richness impacts game generation. We also changed the controls and affordances available to human participants from the AI2-THOR defaults: agents can move in the room, look around, crouch and stand up, pick up objects, rotate held objects, freeze objects[1], and put them down or throw them. Items such as lights and blinds can also be toggled on and off, and other items such as drawers can be opened and closed.

Interactive Experiment Procedure. The interactive, web-based experiment (Figure 2) took place in several phases: *Instructions:* we presented participants with detailed instructions and an attention check quiz. *Tutorial:* we provided a tutorial on the controls and the various affordances different objects offer. To continue to the experiment, participants had to successfully use all actions introduced by the tutorial. *Exploration:* we placed participants in one of the three variations of the environment (randomly assigned), and allowed them to explore in the room. *Creation:* once the participant indicated they had a game ready, they described their game in a form that opened next to the environment screen (Figure 2, right-hand side). Participants optionally provided any setup required to prepare the room for their game to be played, and then specified the gameplay of their game and how it is scored. Participants were also asked to imagine how many points they would score on their first time playing their game, and to rate its difficulty on a five-point Likert scale. *Gameplay:* participants played their game at least once to experience it for themselves. *Edit and debrief:* participants were offered a chance to edit their game after playing it, before answering several debrief questions.

Constraints on Games. Participants were informed that games should be for a single player, require no space beyond the room or additional objects, and include a scoring system.

[1] An option provided to help participants perform dexterous manipulations without knocking over other objects.

The scoring constraint may seem limiting, but any arbitrary goal can be scored by rewarding the achievement of the goal.

Participants. We contacted 192 participants through Prolific, of whom 114 finished the experiment and another 12 were paid due to technical difficulties. Participants were paid a base rate of $10, and received $2 bonus if their game satisfied the required constraints. Successful participants took 44.4 minutes on average, with a standard deviation of 23.3 minutes.

Translation to the DSL. We manually translated the games from the natural language descriptions to the DSL. We made minimal assumptions, and tried to use participants' gameplay replays to clarify their intentions. After excluding 8 games that did not satisfy the constraints, 6 duplicates, and 6 unclear or underconstrained games, we arrived at a dataset of 98 games[2].

Dataset Variety. Considering that participants were neither prompted to nor compensated for their creativity, we were encouraged by the variety and creativity displayed by our participants. While most games in our dataset include some element of throwing, participants varied what was thrown, what it was thrown to, where it was thrown from, and instantiated other constraints (bouncing off a wall or ramp, placing objects on other objects, and many others). We also observed different structure-building games (for height or for number of objects), all manner of different organization games (sorting objects to predetermined locations), and various other games that defy classification (for example, dropping blocks one at a time and adjacent to each other to build a path from the rug to the desk).

Goals as Reward-Generating Programs

We represent games as reward-generating functions, mapping sequences of environment states generated by an agent playing a game to the score received. While this representation is unlikely to perfectly match the cognitive primitives humans use to reason about goals, it allows us to formalize people's productive capacity to generate goals and evaluate progress towards them. The specification of our DSL is available here.

Motivation. The decision to translate games described in natural language to a domain-specific language (DSL) is inspired by both the cognitive science literature as well as artificial intelligence research. On the cognitive science side, we subscribe to the benefits of program representations as outlined by Rule et al. (2020). Programs are compositional, allowing them to reuse elements of previous examples when tackling new challenges. Programs are also expressive, allowing a small number of base types and primitives to solve a wide array of problems. Simultaneously, we draw inspiration from the use of structured description languages in the AI community, and specifically, in representing planning problems using PDDL (Ghallab et al., 1998). PDDL specifies initial and terminal conditions to a planning problem, as well as optional preferences that encourage solving the planning problem in particular fashions. Whereas in planning problems preferences are optional and terminal conditions are mandatory, games are the opposite—key gameplay details are captured by preferences and scoring rules (see sections below), and under 50% of our participants chose to specify an exact end to the game.

We will now briefly outline the different components of our representations, and see Figure 3 for two worked examples of how participants' games can be expressed in the DSL.

Gameplay Preferences. Gameplay preferences capture *how* games should be played by specifying temporal constraints over sequences of states. Each **preference** is named (e.g. *throwBallToBin* in Figure 3), allowing to refer to it as part of the scoring rules or terminal conditions. A gameplay preference can quantify (existentially – **exists** , or universally – **forall**) over one or more object types (the dodgeball and the bin in the first preference in Figure 3 on the right, and the dodgeball in the second preference). Using the quantified variables, a preference then specifies one of two temporal options. A preference could include a series of temporal modals expressed in our syntax under the **then** operator. Temporal reasoning is required because most goals cannot be expressed over single world-states, using a trivial state representation (e.g., the positions and velocities of all objects at a given timestep). Consider the throwing games described in Figure 3: given only a single state, we can check that a ball is currently stationary and in the bin—but we would not know if it was thrown or placed there, and from what distance it might have been thrown. We support a variety of temporal modals, such as **once** (a predicate happens for a single world state) and **hold** (a predicate holds for a sequence of world states). These temporal modals can be mapped onto linear temporal logic (LTL) operators acting on the same predicates[3]. LTL (Manna and Pnueli, 1992) is a temporal modal logic allowing to reason about the validity of prepositions over time, using modal operators such as next (a preposition holds at the next timestep), until (a preposition holds until another holds), and always (a preposition holds for all remaining timesteps). A preference could also consist of a predicate that must hold at the end of gameplay, expressed using the **at-end** operator.

Scoring. The scoring specification describes how to count the different preferences, how much satisfactions of the different preferences are worth, and how to combine these into a final score. The most commonly used counting method (**count-nonoverlapping**) is to count how many times each preference is fulfilled over the entire state trace generated as an agent plays the game. Other counting methods include counting a preference at most once (**count-once**), or counting a preference once for each unique set of objects used to fulfill it (**count-once-per-objects**). Once preferences are counted, the rest of the scoring specification describes the arithmetic to combine the counted preferences to a final score.

Terminal Conditions. A game can specify terminal conditions. These usually take the place of a comparison between a preference counting, as used in the scoring, and a target

[2]We did not model two games due to their complexity, and are missing data from several other participants due to technical issues.

[3]See Section 3 of our full DSL specification here for the mapping. We provide the mapping to facilitate reasoning about the expressivity of preferences in our DSL and to relate to existing work using similar LTL variations to represent RL tasks (e.g., Littman et al., 2017)

First Throwing Game Example

Setup: N/A
Gameplay: "Throwing w Dogball to the can form 1 meter distance"
Scoring: "1 Dogball in the bin = 1 point."

```
01: (:setup )
02: (:constraints
03:  (preference throwBallToBinFromOneMeter
04:    (exists (?d - dodgeball ?h - hexagonal_bin)
05:      (then
06:        (once (and (agent_holds ?d) (= (distance agent ?h) 1)))
07:        (hold (and (not (agent_holds ?d)) (in_motion ?d)))
08:        (once (and (not (in_motion ?d)) (in ?h ?d)))
09: ))))
10: (:scoring maximize
11:   (count-nonoverlapping throwBallToBin)
12: )
```


Second Throwing Game Example

Setup: N/A
Gameplay: "Throw a dodgeball into the bin to score points."
Scoring: "Everytime you score a dodgeball into the bin you get 1 point and you lose 1 point every 5 throws."

```
01: (:setup )
02: (:constraints (and
03:   (preference throwBallToBin
04:     (exists (?d - dodgeball ?h - hexagonal_bin)
05:       (then
06:         (once (agent_holds ?d))
07:         (hold (and (not (agent_holds ?d)) (in_motion ?d)))
08:         (once (and (not (in_motion ?d)) (in ?h ?d)))
09:   )))
10:   (preference throwAttempt
11:     (exists (?d - dodgeball)
12:       (then
13:         (once (agent_holds ?d))
14:         (hold (and (not (agent_holds ?d)) (in_motion ?d)))
15:         (once (not (in_motion ?d)))
16:   )))
17: ))
18: (:scoring maximize (+
19:   (count-nonoverlapping throwBallToBin)
20:   (- (/ (count-nonoverlapping throwAttempt) 5)
21: ))
```

Figure 3: Example games. We provide two participant-generated games to illustrate our DSL and how a more complex game (right) can be built compositionally from a simpler one (left). The images (bottom) illustrate the creator of the more complex game playing their game, showing a ball being held and then successfully thrown into the bin. We reference line numbers in square brackets. **Left:** The game specifies no setup [01]. The game requires specifying a single **preference** [02-09], named *throwBallToBinFromOneMeter* [03] which quantifies a dodgeball and the bin ("can") [04]. It quantifies sequences of states using the **then** operator [05], starting from a single state (**once**) in which the agent holds the dodgeball, and is a distance unit away from the bin [06]. It then seeks a contiguous sequence of states (**hold**), immediately following the previous one, in which the ball is in motion and unheld by the agent [07]. The sequence ends with a single state (or more) in which the ball is stationary and in the bin [08]. The scoring [10-12] seeks to maximize the number of times the preference occurs, counting non-overlapping sequences of states [11]. **Right:** We focus on deviations from the previous game. The *throwBallToBin* preference is slightly simpler in this case, since the participant did not provide a throwing distance constraint [06]. However, they count attempts, which requires specifying a second preference, named *throwAttempt* [10-16], to count attempts regardless of whether or not they land in a bin (compare [15] to [08]). The scoring also requires counting both preferences, dividing the counting of the *throwAttempt* preference by 5 and negating it [20].

number. For example, to specify "You get 5 chances to shoot the dodgeball into the bin from the same location," we would create a preference for a throw attempt (regardless of whether it landed in the bin or not), and terminate after the fifth one.

Setup. In many cases, participants provided instructions for how to set up for their game, in terms of repositioning objects. In the DSL, the setup consists of quantifications over objects, predicates over objects, and marking of these predicates as **game-optional** (can be changed during play) or **game-conserved** (cannot be changed).[4]

Results and Analysis

We report three analyses highlighting four aspects of human games we believe would aid developing computational models of game generation: common sense, compositionality, creativity, and context sensitivity. Our analyses operate over the DSL representations of the games we collected, leveraging the grammar to systematically extract types, predicates, and repeated structures, all of which would be arduous to analyze from natural language. To contextualize the analyses, we attempted to categorize the types of games our participants created into high level classes. We observe three types of gameplay elements participants reused: throwing (handheld objects into or onto other objects), building (stacking objects), and organizing (moving objects to predetermined positions)[5]. 92 of the 98 games in our dataset only use one element, mostly throwing (75), followed by organizing (10), and building (8). Five games combine multiple types of elements, and a single game does not fall into any of these categories.

Common Sense

We begin our analyses by investigating the roles common sense and intuitive physics play in game creation. In our experiment environment, all handheld objects provide the same affordances—they can be picked up, dropped, or thrown—a priori making throwing games involving balls as likely as throwing games involving blocks, for example. To systematically explore this notion, we examine what sorts of objects are used with which predicates across different types of games. We analyze every predicate in the DSL representations of our games, and count how many times it appears with each object type[6]. We then categorize objects into higher level types, such as balls, blocks, furniture, buildings constructed by the agent, and several others. Figure 4 depicts these object-predicate co-occurrence counts, separating between games involving throwing and games not involving throwing.

[4]Consider a setup with "the bin on the bed and the two dodgeballs on the table," with the goal of throwing balls into the bin. The first clause, placing the bin on the bed, should be conserved during gameplay—if a player moved the bin off the bed, they would be violating the game described. The second clause, putting the dodgeballs on the table, must be violated when the agent throws them.

[5]We categorize games based on their scoring preferences. For instance, a game whose setup includes moving items around to enable playing the game, but only scores successful throws, would be labeled as a throwing game, rather than a throwing and organizing game.

[6]For example, line [08] in the examples in Figure 3 would count the predicate *in_motion* appearing with the type *dodgeball*, and the predicate *in* appearing with the types *dodgeball* and *hexagonal_bin*.

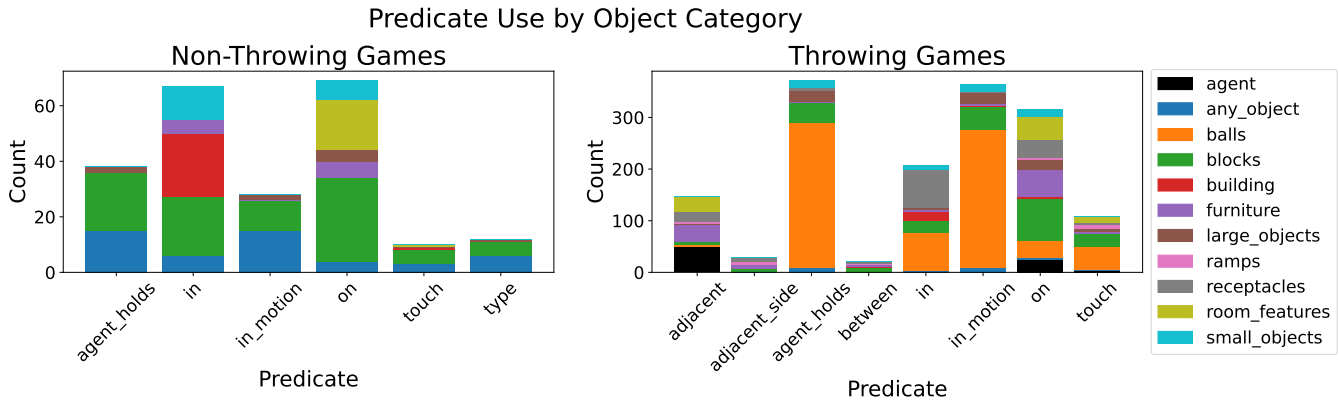

Figure 4: Common sense through object and predicate usage. We categorize object and predicate co-occurrence across games in our dataset, splitting between games with throwing elements and games without them. We coarsen objects types to the categories plotted in the legend, where the 'building' category refers to structures of touching objects created by the participant. For clarity, we omit rare predicates and object categories: types appearing under 10 times are omitted from the non-throwing panel (left), and types appearing under 20 times are omitted from the throwing panel (right). We threshold the panels differently as 18 games contribute to the left-hand panel and 80 to the right-hand one.

First, we observe markedly different distributions of object categories between games types: throwing games tend to involve balls and receptacles, whereas building and organizing games make relatively more use of blocks, small objects, and room features (such as the walls and doors). As all moveable objects present the same affordances, a model that sampled objects uniformly by their affordances would arrive at a distinctly different distribution than the on we observed. Participants seem to rely on their intuitive physical understanding that some objects have straight edges, and so are easier to stack, while other objects are round, and so are easier to throw.

Second, we find a difference in the object-predicate co-occurrence distributions between game types. Throwing games concern themselves with the position of the agent (the black bar segments for *adjacent* and *on* in Figure 4), often to specify constraints on where the agent should throw from to modulate game difficulty). Non-throwing games never refer to the agent's position, instead focusing the constraints on how and where objects should be stacked, as evidenced by the higher frequencies of the *in* and *on* predicates in relation to blocks (green) and room features (yellow). We do not find this application of common sense by our participants to be particularly surprising, but we note that models sampling objects and predicates from a uniform distribution would fail to recreate such patterns. Thus, a model equipped with a rich description language for goals is unlikely to be sufficient on its own; it will also need to represent and utilize basic principles of physical reasoning.

Compositionality and Creativity

Next, we study reuse of recurring motifs (to probe for compositionality) and unique motifs (to quantify creativity) as a means of analyzing the generative structure of games. To that end, we examine all predicate structures that appear in our dataset under temporal operators (**once**, **hold**, etc.), coarsening out the specific variables or referent objects in the predicates to examine the abstract structures (e.g., the expressions (**once** agent_holds ?d) and (**once** agent_holds ?b) would be equivalent, even if the variables ?d and ?b are bound to

Compositionality in Gameplay Preferences

```
01: (preference throwBallToBinFromDesk
02:   (exists (?d - dodgeball ?h - hexagonal_bin)
03:     (then
04:       (once (and (agent_holds ?d) (adjacent agent desk)))
05:       (hold (and (not (agent_holds ?d)) (in_motion ?d)))
06:       (once (and (not (in_motion ?d)) (in ?h ?d)))
07: ))))
08:
09: (preference throwBallFromOffRugToRug
10:   (exists (?d - dodgeball)
11:     (then
12:       (once (and (agent_holds ?d) (not (on rug agent))))
13:       (hold (and (not (agent_holds ?d)) (in_motion ?d)))
14:       (once (and (not (in_motion ?d)) (on rug ?d)))
15: ))))
16:
17: (preference throwOrBounceFromEdgeOfRug
18:   (exists (?d - dodgeball ?h - hexagonal_bin)
19:     (then
20:       (once (and (agent_holds ?d) (adjacent rug agent)))
21:       (hold (and (not (agent_holds ?d)) (in_motion ?d)))
22:       (once (in ?h ?d))
23: ))))
24:
25: (preference blockFromRugToDeskWithoutBreaking
26:   (exists (?c - cube_block)
27:     (then
28:       (once (and (agent_holds ?c) (on rug agent)))
29:       (hold (and
30:         (not (agent_holds ?c)) (in_motion ?c) (on rug agent)
31:         (not (exists (?o - (either lamp laptop desktop)) (broken ?o)))
32:       ))
33:       (once (and (on rug agent) (on desk ?c) (not (in_motion ?c))))
34: ))))
35:
```

Figure 5: Constructing modified preferences through minimal compositional changes. These example preferences, taken from various games in our corpus, build on the structure of the *throwBallToBin* and *throwAttempt* preferences in Figure 3 (right). The *throwBallToBinFromDesk* preference adds a constraint that the ball is thrown with the agent next to the desk [04], while the *throwBallFromOffRugToRug* preference indicates the agent should throw from off the rug [12] and land the ball on the rug [14]. *throwOrBounceFromEdgeOfRug* omits the requirement that the ball end up not in motion in the bin [22], allowing for it to either land inside or bounce in and out. Finally, *blockFromRugToDeskWithoutBreaking* requires the agent to throw a block from the rug [28], avoid leaving the rug while the block is in motion [30], not break any of the objects on the desk [31], and also requires that the block lands and stops on the desk [33].

different types). We then counted how many times each such abstract structure occurs over our entire dataset, and sort by the number of occurrences descending. Figure 6 depicts the cumulative proportion covered by the most common structure, the two most common ones, the four most common ones, and so forth. **Compositionality:** we discover that four recurring structures, including the ones used in lines [07] and

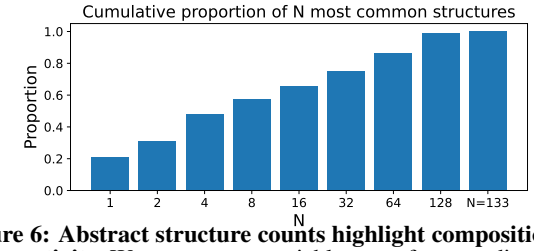

Figure 6: Abstract structure counts highlight compositionality and creativity. We coarsen out variable types from predicate arguments to extract structures, and count the occurrences of each abstract structure across our dataset. Each bar indicates the cumulative proportion covered by the *N* most common structures.

[08] in Figure 3 (right), cover 47.9% of all occurrences in our dataset. We take that as evidence of a heavily compositional structure: combining these recurring structures with varying objects and additional preferences comes up quite often, far more often than would be expected if predicates were drawn arbitrarily from the DSL. Figure 5 depicts the capacity of compositionality in our DSL, offering four examples of how minimal changes to a gameplay preference can alter its semantic meaning, enabling different games. **Creativity:** conversely, we find evidence of creativity on the other end of the count distribution. 56.9% of unique structures appear exactly once in our dataset, and combine to account for almost one in seven occurrences in the entire dataset. We take this as evidence of creativity—just as people are capable of being endlessly creative in play (in our dataset of 98 games, only two were identical), a computational model aiming to generate human-like games would be required to display a similar ability to conceive of unlikely yet plausible instances.

Context Sensitivity

Table 1: Context sensitivity

Version	Few	Medium	Many
Few	100%	80%	100%
Medium	23%	100%	60%
Many	36%	48%	100%

We evaluate how participants tailor goals to the specific objects and affordances presented by the environment. To assess this sort of context sensitivity, we enumerate which objects each game refers to, and check whether or not they exist in the other two versions of the environment. Table 1 plots summarizes this information, where each entry indicates the percentage of games created in the version marked by the row that could be played in the version marked by the column. Room versions are named by how many objects are in each one. There is overlap between the objects in each room (every object in the 'Few' room exists in the 'Many' room, which is not true for the other two pairs). Unsurprisingly, games more often transfer from sparser versions of the environment to more object-dense ones, than vice versa. Although this could be partially accounted for by the overlap in terms of which objects exists, many of the movable objects, and all furniture and architectural room features, exist in all three versions. It would be trivial to construct games that use the shared objects, and in fact, to create games that use no objects at all, albeit they might be less fun and interesting[7]. We hypothesize that to effectively model human game creation, and generate games that are compelling, a model would need to be sensitive to salient objects in its environment.

[7]"Walk in circles around the room. You get a point for each lap."

Discussion

Inspired by intrinsic motivation and playful exploration, this work takes initial steps towards a formal characterization of human game generation. We created a web-based experiment to collect games in an embodied, interactive environment. We formalized a notion of temporally extended goals representing reward-generating functions, and use our dataset to propose a DSL capturing the semantics of the games we collected. We used the representations of games in our DSL to systematically analyze our dataset and highlight four key aspects of human game generation any computational account must address: common sense, compositionality, creativity, and context sensitivity. The DSL is critical for analyzing games in terms of their underlying compositional and correlational structures, which are readily computable from the semantic representations but not from the natural language descriptions. We see the DSL as forming the foundation of future computational modeling efforts to automatically generate human-like goals.

Our work builds upon language of thought (LOT, Fodor, 1979) models in cognitive science, treating knowledge as programs and learning as program induction within this language (Bramley et al., 2018). Rule et al. (2020) offer an extensive of discussion of this approach. Program-like representations have been used to model cognitive processes in a variety of domains: causal models (Chater and Oaksford, 2013), concepts (Goodman et al., 2015), handwritten character generation (Lake et al., 2015), and question asking (Rothe et al., 2017), among many others. Our work extends this tradition by showing how programs can capture playful goals. Looking forward, techniques for synthesizing programs within our DSL will provide guidance in extending our approach to goal generation, although goals are distinguished by the additional challenges of recombining parts in ways that reflect common sense constraints we identified here. Treating goals as programs also provides an avenue for developing models that act on goals, building on the planning literature that inspired our DSL.

We view this work as a stepping stone toward models that can generate, reason with, and pursue playful goals. Beyond the ability to score successful achievement, we are excited about modeling two other human cognitive capacities. One is the ability to reason about goals: could we devise and train a model to predict whether a particular goal is easy or hard? Boring or fun? A second is the capacity to generate new, human-like goals: can a model propose goals that are indistinguishable from human-generated ones? Or tailor goals to use a specific object or affordance? We also hope to investigate the relationship between environment complexity and creativity: how would simpler environments (with fewer objects or affordances) change the breadth of games participants create? Finally, we hope to develop reinforcement learning methods that use playful goals as exploration objectives, or that learn how to pursue many goals and then generalize to new ones.

Acknowledgements

The authors would like to thank Graham Todd and Julian Togelius for helpful discussions at various stages of the project, and Emily Liquin and Pat Little for feedback on earlier versions of the manuscript. This work was supported by NSF Award 1922658 NRT-HDR: FUTURE Foundations, Translation, and Responsibility for Data Science.

References

Akakzia, A., Colas, C., Oudeyer, P.-Y., Chetouani, M., and Sigaud, O. (2021). "Grounding Language to Autonomously-Acquired Skills via Goal Generation". In: *The Ninth International Conference on Learning Representations (2021)* (p. 2).

Bramley, N., Schulz, E., Xu, F., and Tenenbaum, J. (2018). "Learning as program induction" (p. 6).

Buchsbaum, D., Bridgers, S., Weisberg, D. S., and Gopnik, A. (2012). "The power of possibility: causal learning, counterfactual reasoning, and pretend play". *Philosophical Transactions of the Royal Society B: Biological Sciences* 367.1599, p. 2202 (p. 2).

Campero, A., Raileanu, R., Küttler, H., Tenenbaum, J. B., Rocktäschel, T., and Grefenstette, E. (2021). "Learning with AMIGo: Adversarially Motivated Intrinsic Goals". In: *The Ninth International Conference on Learning Representations (2021)* (p. 2).

Chater, N. and Oaksford, M. (2013). "Programs as causal models: speculations on mental programs and mental representation". *Cognitive science* 37.6, pp. 1171–1191 (p. 6).

Chu, J. and Schulz, L. (2020a). "Exploratory play, rational action, and efficient search" (p. 2).

Chu, J. and Schulz, L. E. (2020b). "Play, Curiosity, and Cognition". *Annual Review of Developmental Psychology* 2.1, pp. 317–343 (pp. 1, 2).

Colas, C., Karch, T., Lair, N., Dussoux, J. M., Moulin-Frier, C., Dominey, P. F., and Oudeyer, P. Y. (2020a). "Language as a Cognitive Tool to Imagine Goals in Curiosity-Driven Exploration". *Advances in Neural Information Processing Systems* 2020-December (p. 2).

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y. (2020b). "Autotelic Agents with Intrinsically Motivated Goal-Conditioned Reinforcement Learning: a Short Survey" (p. 2).

Florensa, C., Held, D., Geng, X., and Abbeel, P. (2018). "Automatic Goal Generation for Reinforcement Learning Agents". *35th International Conference on Machine Learning, ICML 2018* 4, pp. 2458–2471 (p. 2).

Fodor, J. A. (1979). *The language of thought*. Harvard University Press, p. 214 (p. 6).

Ghallab, M., Howe, A., Knoblock, C., et al. (1998). "PDDL – The Planning Domain Definition Language" (p. 3).

Goodman, N. D., Tenenbaum, J. B., and Gerstenberg, T. (2015). "Concepts in a Probabilistic Language of Thought". In: *The Conceptual Mind: New Directions in the Study of Concepts*. Ed. by E. Margolis and S. Laurence. MIT Press (p. 6).

Kolve, E., Mottaghi, R., Han, W., et al. (2017). "AI2-THOR: An Interactive 3D Environment for Visual AI" (pp. 1, 2).

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). "Human-level concept learning through probabilistic program induction." *Science (New York, N.Y.)* 350.6266, pp. 1332–8 (p. 6).

Littman, M. L., Topcu, U., Fu, J., Isbell, C., Wen, M., and MacGlashan, J. (2017). "Environment-Independent Task Specifications via GLTL". *arXiv* (p. 3).

Manna, Z. and Pnueli, A. (1992). *The Temporal Logic of Reactive and Concurrent Systems*. Springer New York (p. 3).

Murphy, G. L. (1988). "Comprehending Complex Concepts". *Cognitive Science* 12 (4), pp. 529–562 (p. 1).

Pelz, M. and Kidd, C. (2020). "The elaboration of exploratory play". *Philosophical Transactions of the Royal Society B* 375.1803 (p. 2).

Rothe, A., Lake, B. M., and Gureckis, T. M. (2017). "Question asking as program generation". In: *Advances in Neural Information Processing Systems 30* (p. 6).

Rule, J. S., Tenenbaum, J. B., and Piantadosi, S. T. (2020). "The Child as Hacker". *Trends in Cognitive Sciences* 24.11, pp. 900–915 (pp. 1, 3, 6).

Schmidhuber, J. (2010). "Formal theory of creativity, fun, and intrinsic motivation (1990-2010)". *IEEE Transactions on Autonomous Mental Development* 2.3, pp. 230–247 (p. 1).

Siegel, M. H., Magid, R. W., Pelz, M., Tenenbaum, J. B., and Schulz, L. E. (2021). "Children's exploratory play tracks the discriminability of hypotheses". *Nature Communications 2021 12:1* 12.1, pp. 1–9 (p. 2).

Sim, Z. L. and Xu, F. (2017). "Learning higher-order generalizations through free play: Evidence from 2- and 3-year-old children". *Developmental Psychology* 53.4, pp. 642–651 (p. 1).

Toro Icarte, R., Klassen, T. Q., Valenzano, R., and McIlraith, S. A. (2022). "Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning". *Journal of Artificial Intelligence Research 73 (2022)* 73, pp. 173–208 (p. 2).

Ward, T. B. (1994). "Structured Imagination: the Role of Category Structure in Exemplar Generation". *Cognitive Psychology* 27 (1), pp. 1–40 (p. 1).

Weng, L. (2020). *Exploration Strategies in Deep Reinforcement Learning* (p. 2).