

Interactive Acquisition of Fine-grained Visual Concepts by Exploiting Semantics of Generic Characterizations in Discourse

Jonghyuk Park and Alex Lascarides and Subramanian Ramamoorthy

School of Informatics, University of Edinburgh

10 Crichton Street, Edinburgh EH8 9AB, UK

jay.jh.park@ed.ac.uk, alex@inf.ed.ac.uk, s.ramamoorthy@ed.ac.uk

Abstract

Interactive Task Learning (ITL) concerns learning about unforeseen domain concepts via natural interactions with human users. The learner faces a number of significant constraints: learning should be online, incremental and few-shot, as it is expected to perform tangible belief updates right after novel words denoting unforeseen concepts are introduced. In this work, we explore a challenging symbol grounding task—discriminating among object classes that look very similar—within the constraints imposed by ITL. We demonstrate empirically that more data-efficient grounding results from exploiting the truth-conditions of the teacher’s generic statements (e.g., “Xs have attribute Z.”) and their implicatures in context (e.g., as an answer to “How are Xs and Ys different?”, one infers Y lacks attribute Z).

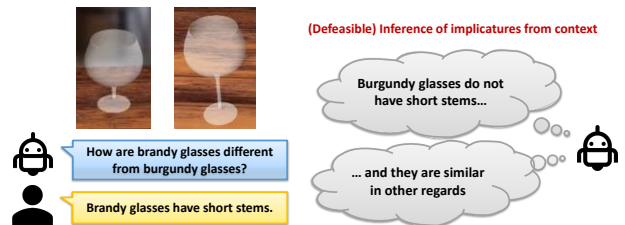
1 Introduction

Consider a general-purpose robot assistant purchased by a restaurant, which must acquire novel domain knowledge to operate in this particular venue. For example, the agent must learn to distinguish brandy glasses from burgundy glasses (Fig. 1a), but these subcategories of glasses are entirely absent from the agent’s domain model in its factory setting. Learning to distinguish among fine-grained visual subcategories is a nontrivial feat (Wei et al., 2021); most current approaches require careful engineering by ML practitioners, making them unsuitable for lay users to readily inspect and update the robot’s domain knowledge.

There are also challenges regarding data efficiency, which using natural language can potentially address (Laird et al., 2017). A single *generic statement*—e.g., “Brandy glasses have short stems”—expresses content that would take many visual examples to infer. Such statements, given their dialogue context, may also carry additional meaning that is linguistically implicit. For



(a) 3D models of fine-grained types of glasses.



(b) Example interaction between a teacher and a learner discussing generic knowledge about types of glasses.

Figure 1: Learning via embodied dialogue in a simulated tabletop domain.

instance, if the statement “Brandy glasses have short stems” is given as an answer to the contrastive question “How are brandy glasses and burgundy glasses different?”, then it implies that burgundy glasses don’t have short stems, and also, defeasibly, that these two types of glasses are similar in other conceivable respects (Grice, 1975; Asher, 2013). Vision processing models that exploit natural language data exist (He and Peng, 2017; Xu et al., 2018; Chen et al., 2018; Song et al., 2020), but they generally treat language as supplementary signals for augmenting training examples, rather than leveraging a range of symbolic inferences licensed by purposeful utterances in dialogue.

In this work, we develop an *interactive* symbol grounding framework, in which the teacher presents to the learner evidence for grounding during embodied dialogues like those illustrated in Fig. 1b. The framework is based on a highly modular neurosymbolic architecture, in which subsym-

bolic perceptual inputs and symbolic conceptual knowledge obtained during dialogues gracefully combine. We run proof-of-concept experiments to show that agents that exploit semantic and pragmatic inferences from generic statements in discourse outperform baselines that don’t exploit semantics and pragmatics, or don’t exploit symbolic inference at all.

2 Related Work

In fine-grained image analysis (FGIA), a model learns to distinguish (patches of) images of sub-categories that belong to the same basic category. FGIA is challenging because images exhibit small inter-class variance and large intra-class variance, and labeling often requires specific domain expertise, hence high annotation costs (Wei et al., 2021).

A natural approach to FGIA is to utilize information of different modalities, including unstructured text descriptions (He and Peng, 2017; Song et al., 2020), structured knowledge bases (Xu et al., 2018; Chen et al., 2018) and human-edited attention maps (Duan et al., 2012; Mitsuhashi et al., 2021). However, to our knowledge, no existing FGIA models exploit NL generic statements provided *in vivo* during natural dialogues. Existing interactive FGIA methods (Branson et al., 2010; Wah et al., 2011, 2014; Cui et al., 2016) query humans to refine predictions from off-the-shelf vision models at inference time but do not update the grounding models. In contrast, our framework supports continuous learning, updating the grounding model as and when the teacher says something noteworthy.

Our use case, described in §1, can be subsumed under the framework of Interactive Task Learning (ITL; Laird et al., 2017). Motivated by scenarios where unforeseen changes may happen to the domain after deployment, the core goal of ITL is to acquire novel concepts that the learner is unaware of but are critical to task success. ITL systems gather evidence from *natural embodied interactions* with a teacher that take place while the learner tries to solve its task. Thus a key desideratum in ITL is that learning should be online and incremental: the learner should change its beliefs and behaviours whenever the teacher provides guidance.

Natural language is a common mode of teacher-learner interaction in ITL (Kirk et al., 2016; She and Chai, 2017). Accordingly, several ITL works draw inspiration from linguistic theories to make learning more effective and efficient. While the for-

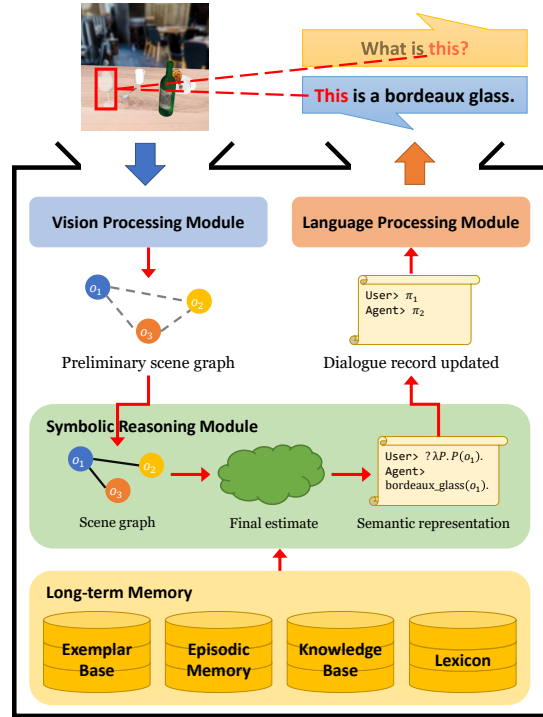


Figure 2: Overview of the architecture in inference mode, in which the component modules interact to generate an answer to a user question.

mal semantics of quantifiers and negation (Rubavicius and Lascarides, 2022) and of discourse coherence (Appelgren and Lascarides, 2020) has been explored in ITL settings, none of the works in the ITL literature have investigated the utility of exploiting the logical inferences licenced by the semantics and pragmatics of contrastive questions and their generic statement answers.

3 Agent architecture

Fig. 2 illustrates our neurosymbolic architecture for situated ITL agents that can engage in extended dialogues with a teacher. Its design enables both subsymbolic-level learning of visual concepts from perceptual inputs (“This looks like a X”) and symbolic-level learning and exploitation of relational knowledge between concepts (“Xs generally have attribute Z”) *during* task execution. Here we stress that our proposed approach is not in direct competition with wide-coverage neural vision-language models, but actually complements them. As an ITL framework, we offer a coping mechanism, to be employed when an existing pre-trained model is deployed in a domain where concepts are frequently introduced and changed, requiring the model to quickly adapt with only a few exemplars

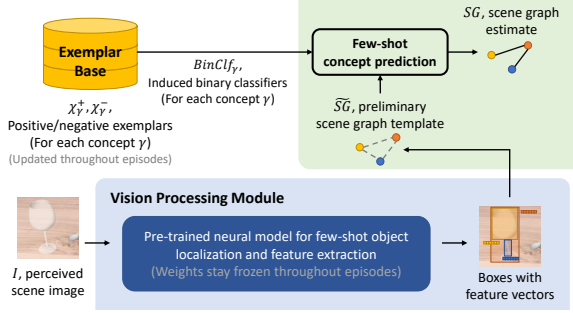


Figure 3: Abridged illustration of the few-shot scene graph generation process (Full version in Appendix A)

of unforeseen concepts.

3.1 Vision processing module

Given a visual scene perceived by the vision sensor, the agent first summarizes the raw input into a graph-like data structure (*scene graph* hereafter). A scene graph SG encodes a set of salient objects in the scene with their distinguishing features and their pairwise relationships, serving as the agent’s internal, abstracted representation of the scene.

Our architecture makes exemplar-based few-shot predictions to generate scene graphs, so as to quickly learn novel visual concepts after a few training instances in an online, incremental fashion. Specifically, our vision processing module employs a neural model extended from Deformable DETR (Zhu et al., 2021), trained to learn distinct low-dimensional metric spaces for each concept type (object class/attribute/relation). The module makes binary concept predictions based on similarity distances between embedded vectors. As illustrated in Fig. 3, the role of the vision module is to process an RGB image input I into a preliminary scene graph template $\tilde{S}G$. The template is further processed along with the agent’s store of concept exemplars in its long-term memory (§3.3) to yield SG . For further details about the inner working of the neural vision module and the translation process from $\tilde{S}G$ to SG , refer to Appendix A.

3.2 Language processing module

The language processing module parses natural language utterances into formal semantic representations, maintains dialogue records, and generates natural language utterances as needed. For controlled experiments, we constrain our attention to a class of simple sentences that discuss primarily two types of information: 1) instance-level descriptions about conceptual identities of scene objects (e.g.,

“This is a brandy glass”, “This has a wide bowl”); and 2) relational knowledge about generic properties shared across instances of the same concepts (e.g., “Brandy glasses have short stems”).

More formally, we represent the propositions expressed by NL sentences via a simple antecedent-consequent pair (PROP hereafter). PROPS draw on a first-order language \mathcal{L} which includes constants referring to objects in the visual scene and predicate symbols for their classes, attributes and pairwise relations (i.e., visual concepts from §3.1). Indicative NL sentences are generally represented with a PROP $\psi = Ante \Rightarrow Cons$, where *Ante* and *Cons* are each a \mathcal{L} -formula (for ψ , we refer to these as $Ante(\psi)$ and $Cons(\psi)$). $Ante(\psi)$ is empty (and thus omitted) if ψ represents a non-conditional, factual statement. Further, we notate a PROP that stands for a generic characterization with a ‘generic quantifier’ \mathbb{G} . For example, the sentences “ o is a brandy glass” and “Brandy glasses have short stems” are translated into PROPS respectively as $brandyGlass(o)$ and $\mathbb{G}O.brandyGlass(O) \Rightarrow haveShortStem(O)$.¹

We represent questions (QUES hereafter) following notation similar to Groenendijk and Stokhof (1982). The answer to a polar question, represented as $?\psi$, is ψ (if true) or $\neg\psi$ (if false). Answers to a *wh*-question $?\lambda X.\psi(X)$ provide values a of X that make $\psi[X/a]$ true (i.e., all occurrences of X in ψ are substituted with a). For the question “How are p_1 and p_2 different?”, we avoid the complexity of higher-order formal languages and simply introduce a reserved formalism $?conceptDiff(p_1, p_2)$, which our implemented dialogue participants can handle by invoking a dedicated procedure. The answer to $?conceptDiff(p_1, p_2)$ is the set of attributes that all objects of class p_1 have and p_2 lack, and *vice versa*.

The language processing module is implemented as a pipeline with two components: an off-the-shelf large-coverage parser of the English Resource Grammar (Copestake and Flickinger, 2000) followed by manual heuristics that map the parser’s outputs to the above forms, as required by the symbolic reasoner (see §3.4). The module also keeps

¹In the interest of brevity and simplicity, we have translated “have short stems” into an ‘agglomerate’ predicate $haveShortStem$ in this text. This is contrary to the actual implementation, where we introduced the concepts *stem*, *short* (unary predicates) and *have* (binary predicate) as elementary units. See Appendix B for a more accurate exposition.

track of the current dialogue history as a sequence of utterances: each one logged as a PROP or QUES, its NL surface form and its speaker.

3.3 Long-term memory module

Our agent stores new knowledge acquired over the course of its operation in its long-term memory. We implement four types of knowledge storage: visual exemplar base (XB), symbolic knowledge base (KB), episodic memory and lexicon.

Visual XB For each visual concept γ , the visual XB stores χ_γ^+ and χ_γ^- , a set of positive/negative exemplars worth remembering. The exemplars serve as the basis of the agent’s few-shot prediction capability as mentioned in §3.1. The visual XB is expanded each time the agent makes an incorrect prediction. Specifically, when the learner incorrectly states “This is $\tilde{\gamma}$ ”, the teacher provides a corrective response, saying “This is not $\tilde{\gamma}$, this is γ ”, thereby augmenting χ_γ^+ and χ_γ^- . New sets $\chi_\gamma^{+/-}$ are created whenever the teacher introduces a novel concept γ via a neologism.

Symbolic KB The symbolic KB is a collection of generic PROPs describing relations between symbolic concepts, such as $\mathbb{G}O.brandyGlass(O) \Rightarrow haveShortStem(O)$. Each KB entry is annotated with the source of the knowledge: a generic rule may be explicitly uttered by the teacher or inferred as an implicature, given the dialogue context. We’ll discuss how the learner can extract unstated knowledge in §4.2.2 in further detail.

Episodic memory The episodic memory stores the summary of each episode of situated interactions between the agent and the teacher.

Lexicon The lexicon stores a set of content words the teacher introduces into the discourse, along with linguistic metadata like part-of-speech.

3.4 Symbolic reasoning

For symbolic reasoning, we employ a probabilistic variant of a logic programming² technique known as answer set programming (ASP; Lifschitz, 2008). The formalism of ASP represents a reasoning problem as a *normal logic program* that consists of rules of the following form:

$$a \leftarrow b_1, \dots, b_m, \text{not } c_1, \dots, \text{not } c_n. \quad (1)$$

²In contrast to first-order logic, logic programming is based on the notion of *minimal models*, where any true atom must be justified (founded) by a clause in the logic program.

where the rule head atom a and the rule body atoms $\{b_i\}_{i=1}^m, \{c_j\}_{j=1}^n$ can be propositional or (quantifier-free) first-order logic formulas. An intuitive reading of the rule, by itself, is that a is logically justified if and only if all of the positive body atoms $\{b_i\}_{i=1}^m$ hold and none of the negative body atoms $\{c_j\}_{j=1}^n$ are proven to hold. For instance, the ASP rule $fly(X) \leftarrow bird(X), \text{not } abnormal(X)$ would roughly correspond to the meaning of the generic NL statement “Birds (generally) fly”. A rule whose head is empty (\perp) represents an *integrity constraint* that its rule body should not hold in answer models.

In probabilistic ASP (Lee and Wang, 2016), each rule is associated with a weight, such that possible worlds satisfying a set of rules with higher total weights are assigned greater probability. Thus a rule may be violated at the expense of its weight. Formally, a probabilistic ASP program $\Pi = \{w : R\}$ is a finite set of weighted rules where R is a rule of the form (1) and w is its associated weight value. The probability of a possible world I according to Π is computed via a log-linear model on the total weight of rules in Π_I , where Π_I is the maximal subset of Π satisfiable by I .

$$W_\Pi(I) = \exp\left(\sum_{w:R \in \Pi_I} w\right) \quad (2)$$

$$P_\Pi(I) = \frac{W_\Pi(I)}{\sum_{J \in \text{possible worlds by } \Pi} W_\Pi(J)} \quad (3)$$

For more rigorous technical definition, refer to Lee and Wang (2016).

Each symbol grounding problem is cast into an appropriate program as follows. First, serialize the learner’s visual observations contained in the scene graph SG into $\Pi_O = \{\text{logit}(s) : \gamma(o_1, \dots)\}$, where each $\gamma(o_1, \dots)$ is a visual observation in SG with confidence score $s \in [0, 1]$. Then we export the KB into a program Π_K , built as follows:

- For each KB entry κ , add to Π_K :

$$\text{logit}(U_d) : \perp \leftarrow \text{Ante}(\kappa), \text{not } \text{Cons}(\kappa).$$

which penalizes ‘deductive violation’ of κ .

- For each set of KB entries $\{\kappa_i\}$ that share identical $\text{Cons}(\kappa_i)$, add to Π_K :

$$\text{logit}(U_a) : \perp \leftarrow \text{Cons}(\kappa_i),$$

$$\bigwedge_{\kappa_i} \{\text{not } \text{Ante}(\kappa_i)\}$$

which penalizes failure to explain $Cons(\kappa_i)$.

Here, $U_d, U_a \in [0, 1]$ are parameters encoding the extent to which the agent relies on its symbolic knowledge; we use $U_d = U_a = 0.95$ in our experiments. For instance, the KB consisting of a single PROP parsed from “Brandy glasses have short stems” will be translated into Π_K consisting of the two rules (7) and (8) in Example 1 below. Finally, the program $\Pi = \Pi_O \cup \Pi_K$ is solved using a belief propagation algorithm (Shenoy, 1997) modified to accommodate the semantics of logic programs.

Example 1. *The program Π below encodes a scenario where the agent sees an object o_1 and initially estimates o_1 is equally likely to be a brandy or burgundy glass. The agent also notices with high confidence it has a short stem, and knows brandy glasses have short stems:*

$$\text{logit}(0.61) : \text{brandyGlass}(o_1). \quad (4)$$

$$\text{logit}(0.62) : \text{burgundyGlass}(o_1). \quad (5)$$

$$\text{logit}(0.90) : \text{haveShortStem}(o_1). \quad (6)$$

$$\text{logit}(0.95) : \perp \leftarrow \text{brandyGlass}(O), \\ \text{not haveShortStem}(O). \quad (7)$$

$$\text{logit}(0.95) : \perp \leftarrow \text{haveShortStem}(O), \\ \text{not brandyGlass}(O). \quad (8)$$

This results in $P_\Pi(\text{brandyGlass}(o_1)) = 0.91$, whereas $P_\Pi(\text{burgundyGlass}(o_1)) = 0.62$. Thus the agent forms a stronger belief that o_1 is a brandy glass than it is a burgundy glass.

See Appendix C for more examples.

4 Interactive Visual Concept Acquisition

4.1 Task description

In our symbol grounding task, each input is a tuple $x_i = (\mathcal{I}_i, b_i)$, where $\mathcal{I}_i \in [0, 1]^{3 \times H \times W}$ is an RGB image and b_i is a specification of a bounding box encasing an object in \mathcal{I}_i . That is, x_i is essentially reference to an object in an image. The task output y_i is dependent on two possible modes of querying the agent about the identity of the object referenced by x_i . The first ‘polar’ mode amounts to testing the agent’s knowledge of a concept in isolation (i.e., “Is this a X?”; so y_i is yes or no). The second ‘multiple-choice’ mode demands the agent selects a single object class y_i describing the object among possible candidates (i.e., “What is this?”, and y_i is a class). The teacher’s response to y_i is dependent on

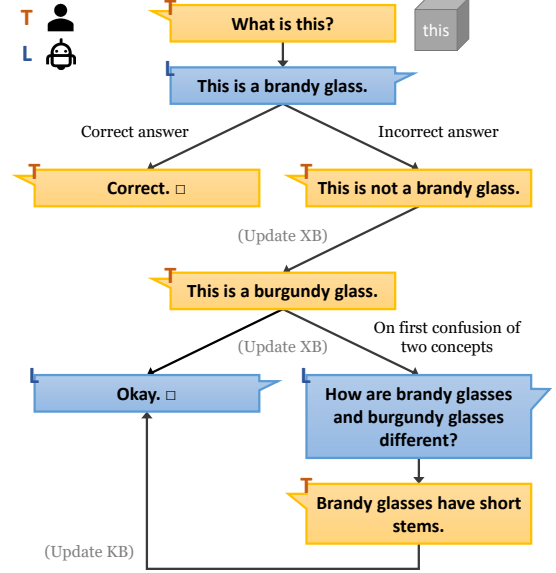


Figure 4: Flowchart covering the range of training dialogues modeled in this study. □ signals termination of an interaction episode.

the content of y_i and the teacher’s dialogue strategy as described in §4.2.1. The learner updates its symbol grounding model from the teacher’s moves using the methods described in §3 and §4.2.2.

As mentioned earlier, the agent’s domain model may entirely lack the concept of interest for labelling x_i . The agent acquires unforeseen concepts via teacher utterances. For example, if “This (x_i) is a brandy glass” introduces the agent to the unforeseen concept “brandy glass”, then $BrandyGlass$ is added to \mathcal{L} and the visual XB is augmented with newly generated sets $\chi_{BrandyGlass}^+ = \{x_i\}$ and $\chi_{BrandyGlass}^- = \emptyset$.

4.2 Flow of dialogues

We focus on a family of dialogues illustrated in Fig. 4. As depicted, each interaction episode is initiated by a teacher query. Dialogues will proceed according to the learner’s responses and the teacher’s strategy. In this research, we want to investigate how different interaction and learning strategies affect learning efficiency.

4.2.1 Teacher’s strategy options

The teacher starts off each interaction episode by presenting an instance o of some visual concept p , querying the learner with a probing QUES “ $? \lambda P.P(o)$ ”.³ If the learner provides the correct

³Note that the expression $? \lambda P.P(o)$ does not fully capture the intended meaning of “What is this?” in its own right, since the discourse contexts set up additional semantic/pragmatic

answer as the PROP “ $p(o)$ ”, the teacher responds “correct” and the episode terminates without agent belief updates. Otherwise, if the learner provides an incorrect answer, “ $\tilde{p}(o)$ ” or “I am not sure”, the teacher needs to provide some corrective information so that the learner can adjust its beliefs. We implement and compare the following variations in the teacher’s response, in increasing order of information content:

- **minHelp**: Provides only boolean feedback to the learner’s answer, i.e., “ $\neg\tilde{p}(o)$ ”.
- **medHelp**: In addition to **minHelp**, provides the correct answer label, saying “ $p(o)$ ”.
- **maxHelp**: In addition to **medHelp**, provides a set of generic PROPs that characterize p or \tilde{p} . The feedback is provided after the learner’s QUES “ $?conceptDiff(p, \tilde{p})$ ”, asked once on the first confusion between p and \tilde{p} .

The generic PROPs provided by **maxHelp** teachers originate from the teacher’s domain knowledge, which we assume here to be correct and exhaustive. The set of PROPs to be delivered is computed as the symmetric difference between the set of properties of p versus that of \tilde{p} (see Appendix D for an example). **minHelp** and **medHelp** serve as vision-only baselines since only concept exemplars with binary labels are communicated as teaching signal.

4.2.2 Learner’s strategy options

Another dimension of variation we model is the learner’s strategy for interpreting generic statements within dialogue contexts. Note that the variation in this dimension is meaningful only when the teacher deploys the **maxHelp** strategy, thereby allowing exploitation of generic statements.

In human dialogues, interlocutors infer, and speakers exploit, *implicatures* that are validated by linguistically explicit moves, given the context of utterance (Grice, 1975). As a core contribution of this study, we model how generic statements given as an answer to a question about similarities and differences give rise to certain implicatures (Asher, 2013) that can be exploited for more data-efficient learning.

Suppose a question “How are X and Y different?” is answered with a generic statement “Xs have attribute Z”. The following implicatures can arise

constraints on what counts as acceptable answers. We have approximated those constraints via our pre-defined dialogue strategies

Situation	
Confusion	brandy glass vs. burgundy glass
Teacher input	“Brandy glasses have short stems.”
Current KB	$\mathbb{G}O.brandyGlass(O) \Rightarrow haveWideBowl(O)$
New KB entries added	
semOnly	$\mathbb{G}O.brandyGlass(O) \Rightarrow haveShortStem(O)$
semNeg	$\mathbb{G}O.brandyGlass(O) \Rightarrow haveShortStem(O)$ $\mathbb{G}O.burgundyGlass(O) \Rightarrow \neg haveShortStem(O)$
semNegScal	$\mathbb{G}O.brandyGlass(O) \Rightarrow haveShortStem(O)$ $\mathbb{G}O.burgundyGlass(O) \Rightarrow \neg haveShortStem(O)$ $\mathbb{G}O.burgundyGlass(O) \Rightarrow haveWideBowl(O)$

Table 1: An example of how different learner strategies update their KBs from the teacher’s generic statement feedback after the learner has confused a burgundy glass for a brandy glass. The learner has already learned that burgundy glasses have wide bowls. PROPs in black is obtained from the teacher’s NL utterance; PROPs in red from ‘negative’ implicatures (ψ^{neg} from ψ) as demanded by coherence; and PROPs in blue from scalar implicatures (κ^{scl} from κ).

from this discourse context: 1) “Ys do not have attribute Z”, and 2) “X and Y are otherwise similar”. The former follows from the assumption that the generic is a coherent answer to a contrastive question (Asher and Lascarides, 2003). The latter, which arguably is more defeasible (Grice, 1975), is what’s known as a scalar implicature: if there were other important differences that the learner should know, then Gricean maxims of conversation predict that the teacher would have included them in the answer as well.

For a PROP ψ , let $\psi^{p \leftrightarrow q}$ denote a PROP which is identical to ψ except that occurrences of the predicate p are all substituted with the predicate q and *vice versa*. We consider the following strategies the learner can take when interpreting a set of generic PROPs $\{\psi_i\}$ provided during an episode:

- **semOnly**: Simply add all ψ_i ’s to KB.
- **semNeg**: In addition to **semOnly**, infer a generic PROP $\psi_i^{neg} = Ante(\psi_i^{p \leftrightarrow \tilde{p}}) \Rightarrow \neg Cons(\psi_i^{p \leftrightarrow \tilde{p}})$ for each ψ_i given as answer to $?conceptDiff(p, \tilde{p})$.
- **semNegScal**: In addition to **semNeg**, infer a generic PROP $\kappa^{scl} = Ante(\kappa^{p \leftrightarrow \tilde{p}}) \Rightarrow Cons(\kappa^{p \leftrightarrow \tilde{p}})$ for each KB entry κ that has either p or \tilde{p} mentioned, only if κ^{scl} is not inconsistent with any of ψ_i ’s or ψ_i^{neg} ’s.

For example, consider the example situation illustrated in Tab. 1. The **semNeg** learner adds “Bur-

gundy glasses do not have short stems” to its KB, and `semNegScal` in addition adds “Brandy glasses have X” for every property X that, according to its KB, burgundy glasses have (e.g., wide bowls).

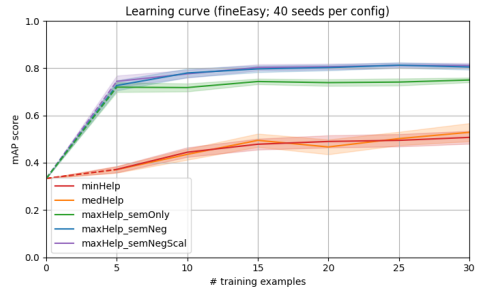
While the `semNeg` inference stems from the demand that the teacher’s move is a coherent answer (Asher and Lascarides, 2003), the scalar implicatures inferred by `semNegScal` are defeasible presumptions (Grice, 1975). That is, `semNegScal` risks misunderstanding the teacher’s intended meaning, inferring general rules that are incorrect—yet cancellable (see Appendix E for an example failure case). Subsequent pieces of refuting evidence may falsify the inferred implicatures without rendering the conversation incoherent. Therefore, we equip our agents with some risk management faculty that can assess and reject contents of scalar implicatures. This is achieved by periodically testing KB entries whose origin is solely from scalar implicatures, rejecting those whose counterexamples can be found in the episodic memory.

5 Experiments

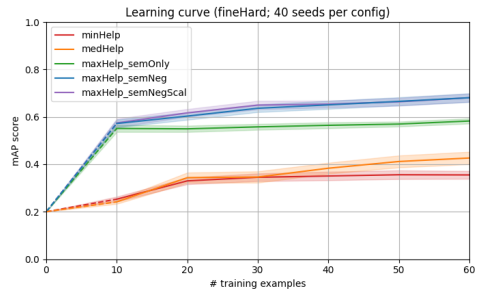
5.1 Evaluation Scheme

We run a suite of experiments that evaluate the data efficiency of the learner’s and teacher’s strategies from §4.2. Results are averaged over multiple sequences of interaction episodes for each of five combinations of teacher’s and learner’s strategies: `minHelp`, `medHelp`, `maxHelp+semOnly`, `maxHelp+semNeg` and `maxHelp+semNegScal`. Each episode-initial probing question “ $?\lambda P.P(o)$ ” is associated with a randomly selected instance o of a concept selected from a round-robin of the target concepts to be acquired. For controlled random selections of concept instances and shuffling of the round-robin, 40 seeds are shared across different configurations. Each sequence continues until the learner makes N_t mistakes in total.

As is common in ITL scenarios, training and inference are fully integrated. Learning has to take place during use whenever the teacher imparts information. In this work, we evaluate our learners by having them take ‘mid-term exams’ on a separate test set after every N_m mistakes made ($N_m \leq N_t$). The mid-term exams comprise binary prediction problems “ $?p(o)$ ” asked per every target concept p for each test example o , and we collect confidence scores between 0 and 1 as response. The primary evaluation metric reported is mean average preci-



(a) fineEasy difficulty (three glass types)



(b) fineHard difficulty (five glass types)

Figure 5: Averaged learning curves (with 95% confidence intervals): effective training examples vs. mAP.

sion (mAP)⁴; we do not use an F1 score because we are more interested in relative rankings between similar-looking concepts than the learners’ absolute performances at some fixed confidence threshold. We also report averaged confusion matrices collected for the sequence-final exams (partially in Fig. 6, fully in the supplementary material).

5.2 Setup

The learner agents start with relatively good, but still error-prone, priors of what bowls and stems and their attributes (e.g., “short stem”) look like, but completely lacks the vocabulary, concepts and related visual features for the various glass types. The prior knowledge is injected into the learner agents by exposing them to the full set of positive examples of stems and bowls in our data set, and randomly sampled non-instances for negative examples. The average binary classification accuracies on balanced test sets were 98.11% for the part concepts and 86.12% for their attributes.

Our training and testing images are randomly generated from a simulation framework CopeliaSim (Rohmer et al., 2013), using a toolkit for controlled sampling of 3D environments (Innes and Ramamoorthy, 2021). Each image features a

⁴Mean of areas under interpolated precision-recall curves.

scene of several objects from the restaurant domain laid on a tabletop (e.g., the image in Fig. 2). Each type of glass in our tabletop domain can be characterized by its parts having different attributes; see Appendix D for the complete list. We implement simulated teachers in place of real human users for the experiments, which perform rule-based pattern matching just sufficient for participating as a teacher in our training dialogues. Our implementation and datasets are publicly released in <https://github.com/itl-ed/ns-arch>.

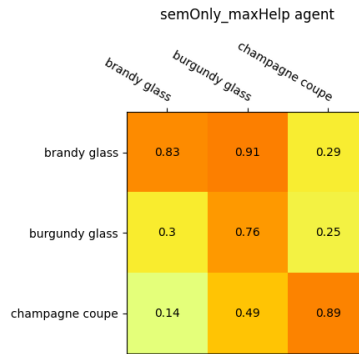
The more distractor concepts we have, the more difficult the task becomes; difficulty scales roughly quadratically with respect to the number of concepts, since C concepts enable $\binom{C}{2}$ different pairwise confusions. Our experiments cover two levels of difficulty: `fineEasy` and `fineHard`. For `fineEasy`, we set $(C, N_t, N_m) = (3, 30, 5)$, where target concepts are {brandy glass, burgundy glass, champagne coupe}. For `fineHard`, we set $(C, N_t, N_m) = (5, 60, 10)$, where target concepts as those for `fineEasy` plus {bordeaux glass, martini glass}.

5.3 Results and Discussion

Fig. 5 and Tab. 2 display the averaged learning curves for the five strategy combinations in each task difficulty setting, along with 95% confidence intervals. It is obvious that learners exploiting the semantics of generic statements from `maxHelp` teachers are significantly faster in picking up new concepts, compared to the vision-only baseline configurations with `minHelp` or `medHelp` teachers. Among the `maxHelp` results, the learners which extract and exploit additional, unstated information from the context (i.e., `semNeg` and `semNegScal`) outperform the learner `semOnly`, which doesn't exploit pragmatics.

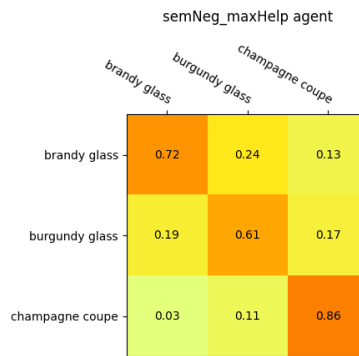
Our error analysis reveals that the significant performance boosts enjoyed by `semNeg` and `semNegScal` learners comes from the ability to infer non-properties from property statements (i.e. ψ^{neg} from ψ). The confusion matrices reported in Fig. 6 allow us to study the mechanism. Specifically, notice how the `maxHelp_semOnly` learner in Fig. 6a frequently misclassifies brandy glasses as burgundy glasses, whereas it is considerably less likely to make such mistakes in the opposite direction: 91% vs. 30%. We can see this is because `semOnly` learners do not have access to

Confusion matrix for (fineEasy; 40 seeds per config)



(a) `maxHelp_semOnly` on `fineEasy` difficulty.

Confusion matrix for (fineEasy; 40 seeds per config)



(b) `maxHelp_semNeg` on `fineEasy` difficulty.

Figure 6: Averaged confusion matrices taken from the sequence-final evaluations for two configurations.

the negative property of burgundy glasses of not having short stems ($\mathbb{G}O.burgundyGlass(O) \Rightarrow \neg haveShortStem(O)$). Therefore, while `semOnly` learners can confidently dismiss instances of burgundy glasses as non-instances of brandy glasses, they are not able to dismiss instances of brandy glasses as non-instances of burgundy glasses. We can observe in Fig. 6b that this is precisely remedied by `semNeg` and `semNegScal` learners, which are able to reliably distinguish the two types in both directions: 24% vs. 19%.

The difference between `semNeg` and `semNegScal` learner is more subtle. Although their performances generally tend to converge after sufficient training, learners that exploit scalar implicatures seem to show higher data efficiency at earlier stages, especially in the `fineHard` task. Nonetheless, the two learning curves have largely overlapping confidence intervals; we cannot make a strong scientific claim based on these results, and we will have to conduct experiments at a larger scale to corroborate this difference.

Task difficulty	fineEasy			fineHard		
# training examples	5	15	30	10	30	60
minHelp	0.372	0.478	0.507	0.253	0.345	0.355
medHelp	0.371	0.494	0.529	0.241	0.346	0.426
maxHelp_semOnly	0.719	0.743	0.750	0.551	0.558	0.582
maxHelp_semNeg	0.727	0.797	0.805	0.572	0.636	0.681
maxHelp_semNegScal	0.744	0.803	0.811	0.574	0.649	0.681

Table 2: Task performances of agents by mAP scores after different numbers of effective training examples.

6 Conclusion and Future Directions

In this research, we have proposed an interactive symbol grounding framework for ITL, along with a neurosymbolic architecture for the learner agent. We empirically showed that learners who can comprehend and exploit valid inferences from generic statements, including pragmatic content given their context of use, can learn to ground novel visual concepts more data-efficiently. Our findings confirm it pays to study human-AI natural language interactions through the lens of discourse semantics, not only the truth conditions of isolated sentences but also their coherent connections to their context.

In future, we plan to relax some of many simplifying assumptions we made for controlled experiments, possibly exploring other domains. For instance, the ideal assumption that teachers are infallible and communication is noise-free does not hold in most real-world scenarios (Appelgren and Lascarides, 2021). Further, the set of linguistic constructions we have studied in this work is very constrained (as intended), and a natural next step is to accommodate a wider range of diverse and free-form NL constructions. It is also a strong assumption that the learner agent already has relatively reliable beliefs about object part and concept attributes. For example, if the learner does not know what the “stem” of a wine glass means, the absence of the concept must be resolved before communicating any generic characterizations involving stems. Finally, our approach does not fully exploit the semantics of generic statements, which express qualitative rules that admit exceptions (Pelletier and Asher, 1997). The generic quantifier \mathbb{G} did not play any significant role in this work. One major strength of ASP is that it is well suited for modeling non-monotonic inferences, and it would be interesting to study how to model ITL scenarios that can robustly address exceptions to generic rules.

Acknowledgements

This work was supported by the UKRI Research Node on Trustworthy Autonomous Systems Governance and Regulation (grant EP/V026607/1) and by Informatics Global PhD Scholarships, funded by the School of Informatics at The University of Edinburgh. We thank the anonymous reviewers for their feedback on an earlier draft of this paper, and Mattias Appelgren, Rimvydas Rubavicius and Gautier Dagan for continued feedback over the course of this research.

References

- Mattias Appelgren and Alex Lascarides. 2020. Interactive task learning via embodied corrective feedback. *Autonomous Agents and Multi-Agent Systems*, 34(2):1–45.
- Mattias Appelgren and Alex Lascarides. 2021. Symbol grounding and task learning from imperfect corrections. In *Proceedings of Second International Combined Workshop on Spatial Language Understanding and Grounded Communication for Robotics*, pages 1–10.
- Nicholas Asher. 2013. Implicatures and discourse structure. *Lingua*, 132:13–28.
- Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.
- Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. 2010. Visual recognition with humans in the loop. In *European Conference on Computer Vision*, pages 438–451. Springer.
- Tianshui Chen, Liang Lin, Riquan Chen, Yang Wu, and Xiaonan Luo. 2018. Knowledge-embedded representation learning for fine-grained image recognition. *arXiv preprint arXiv:1807.00505*.
- Ann A Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage english grammar using hpsg. In *LREC*, pages 591–600. Athens.

- Yin Cui, Feng Zhou, Yuanqing Lin, and Serge Belongie. 2016. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1153–1162.
- Kun Duan, Devi Parikh, David Crandall, and Kristen Grauman. 2012. Discovering localized attributes for fine-grained recognition. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3474–3481. IEEE.
- Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. 2004. Neighbourhood components analysis. *Advances in neural information processing systems*, 17.
- H. P. Grice. 1975. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics Volume 3: Speech Acts*, pages 41–58. Academic Press.
- J. Groenendijk and M. Stokhof. 1982. Semantic analysis of wh-complements. *Linguistics and Philosophy*, 5(2):175–233.
- Xiangteng He and Yuxin Peng. 2017. Fine-grained image classification via combining vision and language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5994–6002.
- Craig Innes and Subramanian Ramamoorthy. 2021. ProboScene: A probabilistic specification language for 3d robotic manipulation environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9446–9452. IEEE.
- James Kirk, Aaron Mininger, and John Laird. 2016. Learning task goals interactively with visual demonstrations. *Biologically Inspired Cognitive Architectures*, 18:1–8.
- Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73.
- J. Laird, K. Gluck, J. Anderson, K. Forbus, O. Jenkins, C. Lebiere, D. Salvucci, M. Scheutz, A. Thomaz, J. Trafton, Robert. Wray, S. Mohan, and J. Kirk. 2017. Interactive task learning. *IEEE Intelligent Systems*, 32:6–21.
- Joohyung Lee and Yi Wang. 2016. Weighted rules under the stable model semantics. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*.
- Vladimir Lifschitz. 2008. What is answer set programming? AAAI’08, page 1594–1597. AAAI Press.
- Masahiro Mitsuhashi, Hiroshi Fukui, Yusuke Sakashita, Takanori Ogata, Tsubasa Hirakawa, Takayoshi Yamashita, and Hironobu Fujiyoshi. 2021. Embedding human knowledge in deep neural network via attention map. In *VISIGRAPP*.
- Francis Jeffrey Pelletier and Nicholas Asher. 1997. Generics and defaults. In *Handbook of logic and language*, pages 1125–1177. Elsevier.
- John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.
- Eric Rohmer, Surya PN Singh, and Marc Freese. 2013. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE.
- Rimvydas Rubavicius and Alex Lascarides. 2022. Interactive symbol grounding with complex referential expressions. In *2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.
- Lanbo She and Joyce Chai. 2017. Interactive learning of grounded verb semantics towards human-robot communication. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1634–1644.
- Prakash P Shenoy. 1997. Binary join trees for computing marginals in the shenoy-shafer architecture. *International Journal of approximate reasoning*, 17(2-3):239–263.
- Kaitao Song, Xiu-Shen Wei, Xiangbo Shu, Ren-Jie Song, and Jianfeng Lu. 2020. Bi-modal progressive mask attention for fine-grained recognition. *IEEE Transactions on Image Processing*, 29:7006–7018.
- Catherine Wah, Steve Branson, Pietro Perona, and Serge Belongie. 2011. Multiclass recognition and part localization with humans in the loop. In *2011 International Conference on Computer Vision*, pages 2524–2531. IEEE.
- Catherine Wah, Grant Van Horn, Steve Branson, Subhansu Maji, Pietro Perona, and Serge Belongie. 2014. Similarity comparisons for interactive fine-grained categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–866.
- Xiu-Shen Wei, Yi-Zhe Song, Oisín Mac Aodha, Jianxin Wu, Yuxin Peng, Jinhui Tang, Jian Yang, and Serge Belongie. 2021. Fine-grained image analysis with deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Huapeng Xu, Guilin Qi, Jingjing Li, Meng Wang, Kang Xu, and Huan Gao. 2018. Fine-grained image classification by visual-semantic embedding. In *IJCAI*, pages 1043–1049.

Nick Zangwill. 2011. Negative properties. *Noûs*, 45(3):528–556.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2021. [Deformable {detr}: Deformable transformers for end-to-end object detection](#). In *International Conference on Learning Representations*.

A Vision processing module: implementational details

Our implementation of the few-shot neural vision processing module is based on the pretrained model of two-staged Deformable DETR (Zhu et al., 2021). We train new lightweight multilayer perceptron (MLP) blocks for embedding image regions into low-dimensional feature spaces. The MLP blocks replace the existing pretrained prediction heads that have fixed number of output categories, enabling metric-based few-shot predictions of incrementally learned visual concepts.

Let C , A and R denote open sets of visual concepts of different types: object classes⁵, attributes⁶ and pairwise relations⁷. In principle, we need one metric space for each concept type for their separate handling, hence three MLP blocks to train. But for this work, $|R| = 1$, where the only relation concept we need to capture is ‘have’ (whole-part relationship). We can make proxy predictions for the concept by the ratio of the area of bounding box intersection to the area of the candidate object part’s bounding box. Therefore, in the interest of simplicity, we prepare only two embedder blocks for C and A respectively; in future extensions where we need to deal with a truly open R , we will have to implement a relation-centric embedder block for R as well.

Fig. 7 depicts how our vision module summarizes the raw RGB image input $\mathcal{I} \in [0, 1]^{3 \times H \times W}$ into a preliminary scene graph \tilde{SG} , and then makes few-shot predictions to finally yield a scene graph SG . \mathcal{I} is first passed through the feature extractor backbone to produce $\{f_l\}_{l=1}^L$, a set of feature maps $f^l \in \mathbb{R}^{C \times H^l \times W^l}$ at L different scales. $\{f_l\}_{l=1}^L$ are flattened into a single sequence of input tokens (thus in $\mathbb{R}^{C \times \sum_l H^l \cdot W^l}$), combined with appropriate positional encodings and fed into the encoder.

⁵Intuitively corresponding to concepts denoted by nouns—e.g., ‘brandy glass’, ‘stem’.

⁶intuitively corresponding to concepts denoted by adjectives—e.g., ‘wide’, ‘short’.

⁷intuitively corresponding to concepts denoted by transitive verbs and adpositions—e.g., ‘have’, ‘of’

We obtain from the encoder an objectness logit score s_i and a proposal bounding box coordinate $\mathbf{b}_i \in [0, 1]^4$ for the input tokens, out of which the top k proposals with the highest s_i scores are selected. The selected proposals are fed into the decoder along with corresponding feature vectors to generate $\mathbf{f}_i^c, \mathbf{f}_i^a \in \mathbb{R}^D$, the class/attribute-centric embeddings of each input token, in addition to the (refined) bounding box coordinates \mathbf{b}_i . The decoder outputs are collated into the preliminary scene graph template $\tilde{SG} = (\tilde{N}, \tilde{E})$. \tilde{N} is the node set containing $\mathbf{b}_i, \mathbf{f}_i^c, \mathbf{f}_i^a$ for each detected object. \tilde{E} is the edge set that *would* contain pairwise relation-centric embeddings $\mathbf{f}_{i,j}^r$ for each pair of detected objects. However, \tilde{E} is essentially empty in our current implementation since as mentioned above, we fall back to proxy prediction by area ratio for the only relation concept of interest ‘have’.

For each visual concept $\gamma \in C, A, R$, the agent’s visual XB stores $\chi_\gamma^{+/-}$, a set of positive/negative exemplars, which together naturally induce a binary classifier $BinClf_\gamma$. We are free to choose any binary classification algorithm as long as it can return probability scores for concept membership from χ_γ^+ and χ_γ^- . We use Platt-scaled SVM with RBF kernel (Platt et al., 1999) in our implementation. Then, $SG = (N, E)$ is generated from \tilde{SG} and a set of $BinClf_\gamma$ ’s, where N and E are each the scene graph node set and the scene graph edge set. For each scene object, N contains $\mathbf{c}_i \in [0, 1]^{|C|}$ and $\mathbf{a}_i \in [0, 1]^{|A|}$, each a vector designating the probabilistic beliefs of whether the object classifies as an instance of concepts in C and A , as well as the box specification \mathbf{b}_i . E contains information about binary relationships between ordered pairs of objects, namely $\mathbf{r}_{i,j} \in [0, 1]^{|R|}$, the probabilistic beliefs of whether the pair (i, j) is an instance of concepts in R . As a reminder, in our setting, N is computed from \tilde{N} and $BinClf_\gamma$ for each $\gamma \in C, A$, whereas E is computed from bounding box area ratios.

Our new embedder blocks are trained on 50% of the Visual Genome dataset (Krishna et al., 2017) with NCA loss objective (Goldberger et al., 2004) for metric learning, for 80,000 steps using SGD optimizer with the batch size of 64, the learning rate of 3×10^{-4} and the momentum factor of 0.1. The prediction heads are then fine-tuned on our tabletop domain dataset⁸ for 2,000 steps using Adam optimizer, with the batch size of 16, the initial learning

⁸Excluding the fine-grained types of drinking glasses.

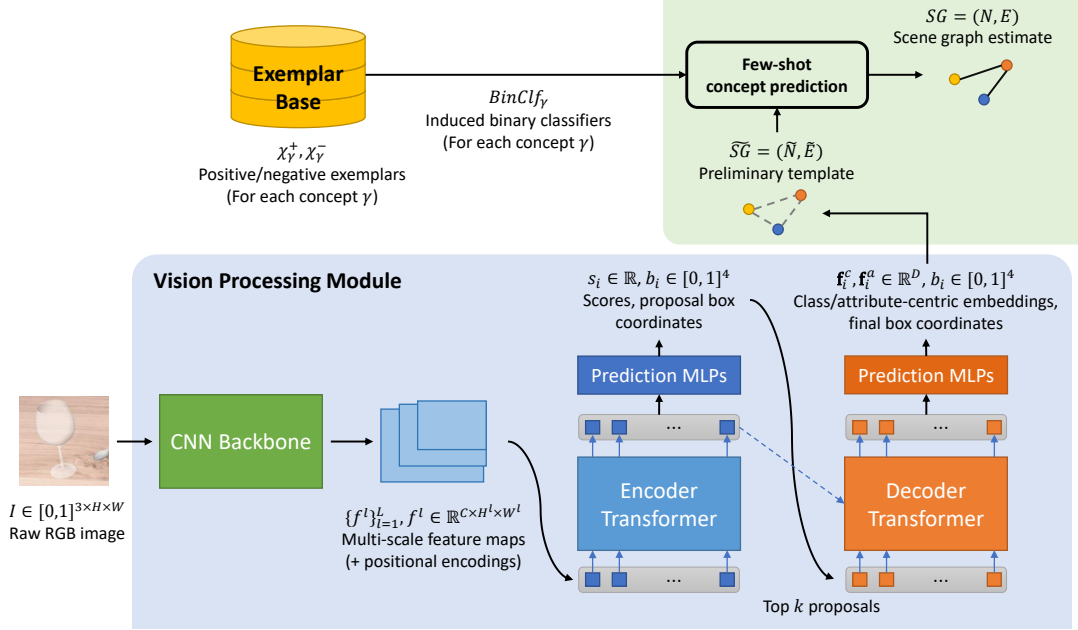


Figure 7: A schematic of the structure of the vision processing module component in our agent architecture, and the pipeline through which raw visual inputs are processed into the final scene graph estimate.

rate of 2×10^{-4} and PyTorch default values⁹ for the hyperparameters $\beta_1, \beta_2, \epsilon$.

B FOL representation of concept properties

In the main paper, we have represented the NL predication “have short stems” with an agglomerate predicate *haveShortStem* for the sake of brevity, so that “Brandy glasses have short stems” would be translated into the PROP $\mathbb{G}O.brandyGlass(O) \Rightarrow haveShortStem(O)$. However, this is an oversimplification of what is actually happening under the hood in our implementation. The predication “have short stems” ought to be broken down into its constituent meanings delivered by the individual tokens “have”, “short” and “stem” respectively, for primarily two reasons: 1) they are the elementary units of concepts handled by the vision module and included in the output scene graphs, and 2) the object parts should be explicitly acknowledged as entities separate from the objects they belong to, and the generic PROPs should model relations between objects and their parts (plus their attributes).

In light of this, we choose to read NL sentences of the form “{object}s have {attribute} {part}s” as follows: “If O is an object, there exists an entity P such that O has P as its part, and P is a part that is attribute”. Then, for exam-

⁹<https://pytorch.org/docs/stable/generated/torch.optim.Adam.html>

ple, the sentence “Brandy glasses have short stems” would be represented by the following PROP:

$$\mathbb{G}O.brandyGlass(O) \Rightarrow (\exists P.have(O, P), short(P), stem(P))$$

or alternatively,

$$\mathbb{G}O.brandyGlass(O) \Rightarrow have(O, f(O)), short(f(O)), stem(f(O))$$

where f is a skolem function that maps from the instance of brandy glass to its (only) short stem. We opt for the latter option because it is more compliant with the formalism commonly used by logic programming methods, in which existential quantifiers are not admitted and variables are all implicitly universally quantified.

C More examples of grounding problems as probabilistic ASP programs

Example 2 below illustrates how lack of high-confidence observation of a short stem of a glass o_1 results in a weaker belief that o_1 is a brandy glass.

Example 2. *The agent sees an object o_1 and initially estimates it’s equally likely to be a brandy or burgundy glass. The agent also notices with high confidence it does NOT have a short stem, and*

knows brandy glasses have short stems:

$$\text{logit}(0.61) : \text{brandyGlass}(o_1). \quad (9)$$

$$\text{logit}(0.62) : \text{burgundyGlass}(o_1). \quad (10)$$

$$\text{logit}(0.10) : \text{haveShortStem}(o_1). \quad (11)$$

$$\text{logit}(0.95) : \perp \leftarrow \text{brandyGlass}(O), \\ \text{not haveShortStem}(O). \quad (12)$$

$$\text{logit}(0.95) : \perp \leftarrow \text{haveShortStem}(O), \\ \text{not brandyGlass}(O). \quad (13)$$

This results in $P_{\Pi}(\text{brandyGlass}(o_1)) = 0.20$, whereas $P_{\Pi}(\text{burgundyGlass}(o_1)) = 0.62$.

Example 3 shows how knowledge of *negative* properties of an object class can affect symbolic reasoning. The example supposes the agent’s KB only consists of the knowledge “Burgundy glasses do *not* have short stems”, namely the PROP $\mathbb{G}O.\text{burgundyGlass}(O) \Rightarrow \neg\text{haveShortStem}(O)$. Note how we translate a generic PROP whose *Cons* is a negation of some \mathcal{L} -formula into probabilistic ASP rules. Only rules penalizing deductive violations are generated, in which the negation (\neg) that wraps around *Cons* ‘cancels out’ the default negation `not`. We do not generate rules for penalizing failures to explain *Cons* from negative PROPS, as abductive inferences of object classes from lack of properties would give rise to far-fetched conclusions: e.g., inferring something might be a banana because it does not have wheels.

Example 3. *The agent sees an object o_1 and initially estimates it’s equally likely to be a brandy or burgundy glass. The agent also notices with high confidence it has a short stem, and knows burgundy glasses do NOT have short stems:*

$$\text{logit}(0.61) : \text{brandyGlass}(o_1). \quad (14)$$

$$\text{logit}(0.62) : \text{burgundyGlass}(o_1). \quad (15)$$

$$\text{logit}(0.90) : \text{haveShortStem}(o_1). \quad (16)$$

$$\text{logit}(0.95) : \perp \leftarrow \text{burgundyGlass}(O), \\ \text{haveShortStem}(O). \quad (17)$$

This results in $P_{\Pi}(\text{brandyGlass}(o_1)) = 0.61$, whereas $P_{\Pi}(\text{burgundyGlass}(o_1)) = 0.19$.

Note that knowledge about brandy glasses do not affect the likelihood of o_1 being a burgundy glass, and *vice versa*: i.e., for an object, the events of being a brandy glass vs. a

burgundy glass are independent. This is because the KBs in the examples do not introduce any type of dependency between the two glass types. For instance, if we inject mutual exclusivity relation between the two types in the KB, both probability values $P_{\Pi}(\text{brandyGlass}(o_1))$ and $P_{\Pi}(\text{burgundyGlass}(o_1))$ would be affected by knowledge about either.

D Task domain: Fine-grained types of drinking glasses to distinguish






Type	Properties	Sample image
bordeaux glass	Bowl: elliptical, tapered.	
brandy glass	Bowl: wide, tapered, round. Stem: short.	
burgundy glass	Bowl: wide, tapered, round.	
champagne coupe	Bowl: broad, round.	
martini glass	Bowl: broad, conic.	

Table 3: Fine-grained types of drinking glasses modeled in our tabletop domain. (Note only brandy glasses have characteristic stems, whereas bowls of all glass types can be characterized by some set of attributes.)

Tab. 3 lists the set of fine-grained types of drinking glasses that are modeled in our simulated tabletop domain, along with their properties and sample images. 3D meshes of the glasses are obtained from a website that lists stock models made by third-party providers,¹⁰ then imported into the simulation environment.

As illustrated, properties of each fine-grained type comprise its part attributes. For instance, the full set of properties of a brandy glass could be expressed as a set $\{(\text{wide, bowl}), (\text{tapered, bowl}), (\text{round, bowl}), (\text{short, stem})\}$. When asked, our simulated teacher computes the answer to `?conceptDiff QUES` as pairwise symmetric differences between property sets: e.g., for `?conceptDiff(brandyGlass, burgundyGlass)`,

¹⁰<https://www.turbosquid.com/3d-models/wine-glasses-3d-1385831>

Confusion	champagne coupe - burgundy glass	burgundy glass - bordeaux glass
KB state	$\mathbb{G}O.champagneCoupe(O) \Rightarrow haveBroadBowl(O)$ $\mathbb{G}O.burgundyGlass(O) \Rightarrow$ $haveWideBowl(O), haveTaperedBowl(O)$ $\mathbb{G}O.burgundyGlass(O) \Rightarrow \neg haveBroadBowl(O)$ $\mathbb{G}O.champagneCoupe(O) \Rightarrow$ $\neg(haveWideBowl(O), haveTaperedBowl(O))$	$\mathbb{G}O.burgundyGlass(O) \Rightarrow$ $haveWideBowl(O), haveRoundBowl(O)$ $\mathbb{G}O.bordeauxGlass(O) \Rightarrow haveEllipticalBowl(O)$ $\mathbb{G}O.bordeauxGlass(O) \Rightarrow$ $\neg(haveWideBowl(O), haveRoundBowl(O))$ $\mathbb{G}O.burgundyGlass(O) \Rightarrow \neg haveEllipticalBowl(O)$ $\mathbb{G}O.bordeauxGlass(O) \Rightarrow$ $haveWideBowl(O), haveTaperedBowl(O)$ $\mathbb{G}O.bordeauxGlass(O) \Rightarrow \neg haveBroadBowl(O)$

Table 4: An example illustration of how `semNegScal` learners can infer incorrect and unintended knowledge. The underlined PROP denotes a generic rule which is neither correct nor intended by the teacher.

we obtain $\{(short, stem)\}$ for brandy glasses and \emptyset for burgundy glasses.

These properties of glasses did not ship with the 3D models; instead, we hand-coded them based on information available on the internet. We have put effort to prepare an annotation scheme that is faithful to properties of the glasses in the reality, yet the domain knowledge may still have inconsistency against the ‘ground-truth’—any error in that regard remains our own.

E Rule acquisition by inference of implicatures and failure case analysis

In this work, we assume that all generic NL statements given by the teacher are characterizations of object classes by their positive properties (those described in Appendix D), and statements of *negative* properties are never explicitly provided. This reflects the fact that we usually characterize things by their positive properties rather than by their negative properties because the former generally have more determining power (Zangwill, 2011). Therefore, in our experiments, negative properties can be obtained only by virtue of inference of implicatures. That is, for example, only `semNeg` or `semNegScal` learners have access to the negative PROP $\mathbb{G}O.burgundyGlass(O) \Rightarrow \neg haveShortStem(O)$.

Nevertheless, `semNegScal` learners risk acquisition of incorrect and unintended knowledge when they make inferences of scalar implicatures. To see this, study the example illustrated in Tab. 4, where two successive confusions take place in the order of brandy glass vs. burgundy glass and then burgundy glass vs. champagne coupe. In the example, the underlined PROP successfully infiltrates into the learner’s KB without being suppressed by explicitly stated PROPs or their negative implicature counterparts. This is why in-

ference of the scalar implicatures should be cancellable, so that they can be retracted in the face of contradictory evidence. In our implementation, this is achieved by periodically inspecting the KB entries against the episodic memory, removing any rules whose counterexamples are found.