Neuro-Symbolic Visual Reasoning: Disentangling "Visual" from "Reasoning"

Saeed Amizadeh [1] **Hamid Palangi** [*2] **Oleksandr Polozov** [*2] **Yichen Huang** [2] **Kazuhito Koishida** [1]

Appendix A: Proofs

Proof. **Lemma 3.1:** Let X be the left most variable appearing in formula $\mathcal{F}(X, ...)$, then depending on the quantifier q of X, we will have:

$$\begin{aligned}
\text{If } q = \forall:\ \alpha(\mathcal{F}) &= \Pr(\mathcal{F} \Leftrightarrow \top \mid \mathcal{V}) \\
&= \Pr\Big(\bigwedge_{i=1}^{N} \mathcal{F}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V}\Big) \\
&= \prod_{i=1}^{N} \Pr(\mathcal{F}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V}) \\
&= \prod_{i=1}^{N} \alpha(\mathcal{F} \mid x_i) = \mathcal{A}_{\forall}\big(\boldsymbol{\alpha}(\mathcal{F} \mid X)\big)
\end{aligned}$$

$$\begin{aligned}
\text{If } q = \exists:\ \alpha(\mathcal{F}) &= \Pr(\mathcal{F} \Leftrightarrow \top \mid \mathcal{V}) \\
&= \Pr\Big(\bigvee_{i=1}^{N} \mathcal{F}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V}\Big) \\
&= 1 - \prod_{i=1}^{N} \Pr(\mathcal{F}_{X=x_i} \Leftrightarrow \bot \mid \mathcal{V}) \\
&= 1 - \prod_{i=1}^{N} \big(1 - \alpha(\mathcal{F} \mid x_i)\big) = \mathcal{A}_{\exists}\big(\boldsymbol{\alpha}(\mathcal{F} \mid X)\big)
\end{aligned}$$

$$\begin{aligned}
\text{If } q = \nexists:\ \alpha(\mathcal{F}) &= \Pr(\mathcal{F} \Leftrightarrow \top \mid \mathcal{V}) \\
&= \Pr\Big(\bigwedge_{i=1}^{N} \mathcal{F}_{X=x_i} \Leftrightarrow \bot \mid \mathcal{V}\Big) \\
&= \prod_{i=1}^{N} \Pr(\mathcal{F}_{X=x_i} \Leftrightarrow \bot \mid \mathcal{V}) \\
&= \prod_{i=1}^{N} \big(1 - \alpha(\mathcal{F} \mid x_i)\big) = \mathcal{A}_{\nexists}\big(\boldsymbol{\alpha}(\mathcal{F} \mid X)\big)
\end{aligned}$$

Note that the key underlying assumption in deriving the above proofs is that the binary logical statements $\mathcal{F}_{X=x_i}$ for all objects x_i are independent random variables *given* the visual featurization of the scene, which is a viable assumption. □

Proof. **Lemma 3.2:** $\boldsymbol{\alpha}(\mathcal{F} \mid X) = \big[\Pr(\mathcal{F}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V})\big]_{i=1}^{N} = \big[\Pr(\top \Leftrightarrow \top \mid \mathcal{V})\big]_{i=1}^{N} = \mathbf{1}$ □

Proof. **Lemma 3.3:**

(A) If $\mathcal{F}(X, Y, Z, ...) = \neg\mathcal{G}(X, Y, Z, ...)$:

$$\begin{aligned}
\boldsymbol{\alpha}(\mathcal{F} \mid X) &= \big[\Pr(\mathcal{F}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V})\big]_{i=1}^{N} \\
&= \big[\Pr(\mathcal{G}_{X=x_i} \Leftrightarrow \bot \mid \mathcal{V})\big]_{i=1}^{N} \\
&= \big[1 - \alpha(\mathcal{G} \mid x_i)\big]_{i=1}^{N} = \mathbf{1} - \boldsymbol{\alpha}(\mathcal{G} \mid X)
\end{aligned}$$

(B) If $\mathcal{F}(X, Y, Z, ...) = \pi(X) \wedge \mathcal{G}(X, Y, Z, ...)$ where $\pi(\cdot)$ is a unary predicate:

$$\begin{aligned}
\boldsymbol{\alpha}(\mathcal{F} \mid X) &= \big[\Pr(\mathcal{F}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V})\big]_{i=1}^{N} \\
&= \big[\Pr(\pi(x_i) \wedge \mathcal{G}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V})\big]_{i=1}^{N} \\
&= \big[\Pr(\pi(x_i) \Leftrightarrow \top \wedge \mathcal{G}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V})\big]_{i=1}^{N} \\
&= \big[\Pr(\pi(x_i) \Leftrightarrow \top \mid \mathcal{V}) \cdot \Pr(\mathcal{G}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V})\big]_{i=1}^{N} \\
&= \big[\alpha(\pi \mid x_i) \cdot \alpha(\mathcal{G} \mid x_i)\big]_{i=1}^{N} \\
&= \boldsymbol{\alpha}(\pi \mid X) \odot \boldsymbol{\alpha}(\mathcal{G} \mid X)
\end{aligned}$$

(C) If $\mathcal{F}(X, Y, Z, ...) = \big[\bigwedge_{\pi \in \Pi_{XY}} \pi(X, Y)\big] \wedge \mathcal{G}(Y, Z, ...)$ where Π_{XY} is the set of all binary predicates defined on variables X and Y in $\mathcal{F}$ and Y is the left most variable in $\mathcal{G}$ with quantifier q:

$$\begin{aligned}
\boldsymbol{\alpha}(\mathcal{F} \mid X) &= \big[\Pr(\mathcal{F}_{X=x_i} \Leftrightarrow \top \mid \mathcal{V})\big]_{i=1}^{N} \\
&= \left[\Pr\Bigg(\underbrace{\Big[\bigwedge_{\pi \in \Pi_{XY}} \pi(x_i, Y)\Big]}_{\mathcal{R}_{x_i}(Y)} \wedge \mathcal{G} \Leftrightarrow \top \mid \mathcal{V}\Bigg)\right]_{i=1}^{N} \\
&\overset{\text{L3.1}}{=} \big[\mathcal{A}_q\big(\boldsymbol{\alpha}(\mathcal{R}_{x_i} \wedge \mathcal{G} \mid Y)\big)\big]_{i=1}^{N} \\
&\overset{\text{L3.3B}}{=} \big[\mathcal{A}_q\big(\boldsymbol{\alpha}(\mathcal{R}_{x_i} \mid Y) \odot \boldsymbol{\alpha}(\mathcal{G} \mid Y)\big)\big]_{i=1}^{N}
\end{aligned}$$

[*]Equal contribution [1]Microsoft Applied Sciences Group (ASG), Redmond WA, USA [2]Microsoft Research AI, Redmond WA, USA. Correspondence to: Saeed Amizadeh <saamizad@microsoft.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

$$\overset{\text{L3.3B}}{=} \left[\mathcal{A}_q\left(\left[\bigodot_{\pi\in\Pi_{XY}} \boldsymbol{\alpha}(\pi_{X=x_i} \mid Y)\right] \odot \boldsymbol{\alpha}(\mathcal{G} \mid Y)\right)\right]_{i=1}^{N}$$

$$= \left[\mathcal{A}_q\left(\left[\bigodot_{\pi\in\Pi_{XY}} \boldsymbol{\alpha}(\pi \mid x_i, Y)\right] \odot \boldsymbol{\alpha}(\mathcal{G} \mid Y)\right)\right]_{i=1}^{N}$$

$$= \left[\bigodot_{\pi\in\Pi_{XY}} \boldsymbol{\alpha}(\pi \mid X, Y)\right] \times_q \boldsymbol{\alpha}(\mathcal{G} \mid Y)$$

Note that the key underlying assumption in deriving the above proofs is that all the unary and binary predicates $\pi(x_i)$ and $\pi(x_i, y_i)$ for all objects x_i and y_j are independent binary random variables *given* the visual featurization of the scene, which is a viable assumption. □

Appendix B: The Language System

Our language system defines the pipeline to translate the questions in the natural language (NL) all the way to the DFOL language which we can then run to find the answer to the question. However, as opposed to many similar frameworks in the literature, our translation process takes place in two steps. First, we *parse* the NL question into the *task-dependent*, high-level, domain-specific language (DSL) of the target task. We then *compile* the resulted DSL program into the *task-independent*, low-level DFOL language. This separation is important because the ∇-FOL core reasoning engine executes the task-independent, four basic operators of the DFOL language (i.e. **Filter**, **Relate**, **Neg** and $\mathcal{A}_{\{\forall,\exists,\nexists\}}$) and *not* the task specific DSL operators. This distinguishes ∇-FOL from similar frameworks in the literature as a *general-purpose* formalism; that is, ∇-FOL can cover any reasoning task that is representable via first-order logic, and not just a specific DSL. This is mainly due to the fact that DFOL programs are equivalent to FOL formulas (up to reordering) as shown in Section 3.3. Figure 1 shows the proposed language system along with its different levels of abstraction. For more details, please refer to our PyTorch code base: https://github.com/microsoft/DFOL-VQA.

For the GQA task, we train a neural semantic parser using the annotated programs in the dataset to accomplish the first step of translation. For the second step, we simply use a *compiler*, which converts each high-level GQA operator into a composition of DFOL basic operators. Table 1 shows this (fixed) conversion along with the equivalent FOL formula for each GQA operator.

Most operators in the GQA DSL are parameterized by a set of NL tokens that specify the arguments of the operation (e.g. *"attr"* in **GFilter** specifies the attribute that the operator is expected to filter the objects based upon). In addition to the NL arguments, both terminal and non-terminal operators take as input the attention vector(s) on the objects present in the scene (except for **GSelect** which does not take any input attention vector). However, in terms of their outputs, terminal and non-terminal operators are fundamentally different. A terminal operator produces a scalar likelihood or a list of scalar likelihoods (for "query" type operators). Because they are "terminal", terminal operators have logical quantifiers in their FOL description; this, in turn, prompts the aggregation operator $\mathcal{A}_{\{\forall,\exists,\nexists\}}$ in their equivalent DFOL translation. Non-terminal operators, on the other hand, produce attention vectors on the objects in the scene without calculating the aggregated likelihood.

Appendix C: Some Examples from the Hard and the Easy Sets

In this appendix, we visually demonstrate a few examples from the hard and the easy subsets of the GQA Test-Dev split. Figures 2,3,4 show a few examples from the hard set with their corresponding questions, while Figures 5,6 show a few examples from the easy set. In these examples, the green rectangles represent where in the image the model is attending according to the attention vector $\boldsymbol{\alpha}(\mathcal{F} \mid X)$. Here the formula $\mathcal{F}$ represents either the entire question for the easy set examples or the partial question up until to the point where the visual system failed to produce correct likelihoods for the hard set examples. We have included the exact nature of the visual system's failure for the hard set examples in the captions. As illustrated in the paper, the visually hard-easy division here is with respect to the original Faster-RCNN featurization. This means that the "hard" examples presented here are *not* necessarily impossible in general, but are hard with respect to this specific featurization.

Furthermore, in Figure 7, we have demonstrated two examples from the hard set for which taking into the consideration the context of the question via the calibration process helped to overcome the imperfectness of the visual system and find the correct answer. Please refer to the caption for the details.

GQA OP	T	Equivalent FOL Description	Equivalent DFOL Program
GSelect$(name)[]$	N	$name(X)$	$\mathbf{Filter}_{\mathsf{name}}\big[\mathbf{1}\big]$
GFilter$(attr)[\boldsymbol{\alpha}_X]$	N	$attr(X)$	$\mathbf{Filter}_{\mathsf{attr}}\big[\boldsymbol{\alpha}_X\big]$
GRelate$(name, rel)[\boldsymbol{\alpha}_X]$	N	$name(Y) \wedge rel(X, Y)$	$\mathbf{Filter}_{\mathsf{name}}\big[\mathbf{Relate}_{rel,\exists}[\boldsymbol{\alpha}_X]\big]$
GVerifyAttr$(attr)[\boldsymbol{\alpha}_X]$	Y	$\exists X : attr(X)$	$\mathcal{A}_{\exists}\big(\mathbf{Filter}_{\mathsf{attr}}[\boldsymbol{\alpha}_X]\big)$
GVerifyRel$(name, rel)[\boldsymbol{\alpha}_X]$	Y	$\exists Y \exists X : name(Y) \wedge rel(X, Y)$	$\mathcal{A}_{\exists}\big(\mathbf{Filter}_{\mathsf{name}}\big[\mathbf{Relate}_{rel,\exists}[\boldsymbol{\alpha}_X]\big]\big)$
GQuery$(category)[\boldsymbol{\alpha}_X]$	Y	$\big[\exists X : c(X) \text{ for } c \text{ in } category\big]$	$\big[\mathcal{A}_{\exists}\big(\mathbf{Filter}_{\mathsf{c}}[\boldsymbol{\alpha}_X]\big) \text{ for } c \text{ in } category\big]$
GChooseAttr$(a_1, a_2)[\boldsymbol{\alpha}_X]$	Y	$\big[\exists X : a(X) \text{ for } a \text{ in } [a_1, a_2]\big]$	$\big[\mathcal{A}_{\exists}\big(\mathbf{Filter}_{\mathsf{a}}[\boldsymbol{\alpha}_X]\big) \text{ for } a \text{ in } [a_1, a_2]\big]$
GChooseRel$(n, r_1, r_2)[\boldsymbol{\alpha}_X]$	Y	$\big[\exists Y \exists X : n(Y) \wedge r(X, Y) \text{ for } r \text{ in } [r_1, r_2]\big]$	$\big[\mathcal{A}_{\exists}\big(\mathbf{Filter}_{\mathsf{name}}\big[\mathbf{Relate}_{rel,\exists}[\boldsymbol{\alpha}_X]\big]\big) \text{ for } r \text{ in } [r_1, r_2]\big]$
GExists$()[\boldsymbol{\alpha}_X]$	Y	$\exists X ...$	$\mathcal{A}_{\exists}(\boldsymbol{\alpha}_X)$
GAnd$()[\boldsymbol{\alpha}_X, \boldsymbol{\alpha}_Y]$	Y	$\exists X ... \wedge \exists Y ...$	$\mathcal{A}_{\exists}(\boldsymbol{\alpha}_X) \cdot \mathcal{A}_{\exists}(\boldsymbol{\alpha}_Y)$
GOr$()[\boldsymbol{\alpha}_X, \boldsymbol{\alpha}_Y]$	Y	$\exists X ... \vee \exists Y ...$	$1 - \big(1 - \mathcal{A}_{\exists}(\boldsymbol{\alpha}_X)\big) \cdot \big(1 - \mathcal{A}_{\exists}(\boldsymbol{\alpha}_Y)\big)$
GTwoSame$(category)[\boldsymbol{\alpha}_X, \boldsymbol{\alpha}_Y]$	Y	$\exists X \exists Y \bigvee_{c \in category} \big(c(X) \wedge c(Y)\big)$	$\mathcal{A}_{\exists}\big(\big[\mathcal{A}_{\exists}\big(\mathbf{Filter}_{\mathsf{c}}[\boldsymbol{\alpha}_X]\big) \cdot \mathcal{A}_{\exists}\big(\mathbf{Filter}_{\mathsf{c}}[\boldsymbol{\alpha}_Y]\big) \text{ for } c \text{ in } category\big]\big)$
GTwoDifferent$(category)[\boldsymbol{\alpha}_X, \boldsymbol{\alpha}_Y]$	Y	$\exists X \exists Y \bigwedge_{c \in category} \big(\neg c(X) \vee \neg c(Y)\big)$	$1 - \mathbf{GTwoSame}(category)[\boldsymbol{\alpha}_X, \boldsymbol{\alpha}_Y]$
GAllSame$(category)[\boldsymbol{\alpha}_X]$	Y	$\bigvee_{c \in category} \forall X : ... \rightarrow c(X)$	$1 - \prod_{c \in category} \mathcal{A}_{\exists}\big(\boldsymbol{\alpha}_X \odot \mathbf{Neg}\big[\mathbf{Filter}_{\mathsf{c}}[\boldsymbol{\alpha}_X]\big]\big)$

Table 1. The GQA operators translated to our FOL formalism. Here the notation $\boldsymbol{\alpha}_X$ is the short form for the attention vector $\boldsymbol{\alpha}(\mathcal{F} \mid X)$ where $\mathcal{F}$ represents the formula the system has already processed up until the current operator. For the sake of simplicity, we have not included all of our GQA DSL here but the most frequent ones. Also the "Relate"-related operators are only shown for the case where the input variable X is the "subject" of the relation. The formalism is the same for the "object" role case except that the order of X and Y are swapped in the relation. The column **T** in the table indicates whether the operator is terminal or not. The full DSL can be found at our code base: https://github.com/microsoft/DFOL-VQA.

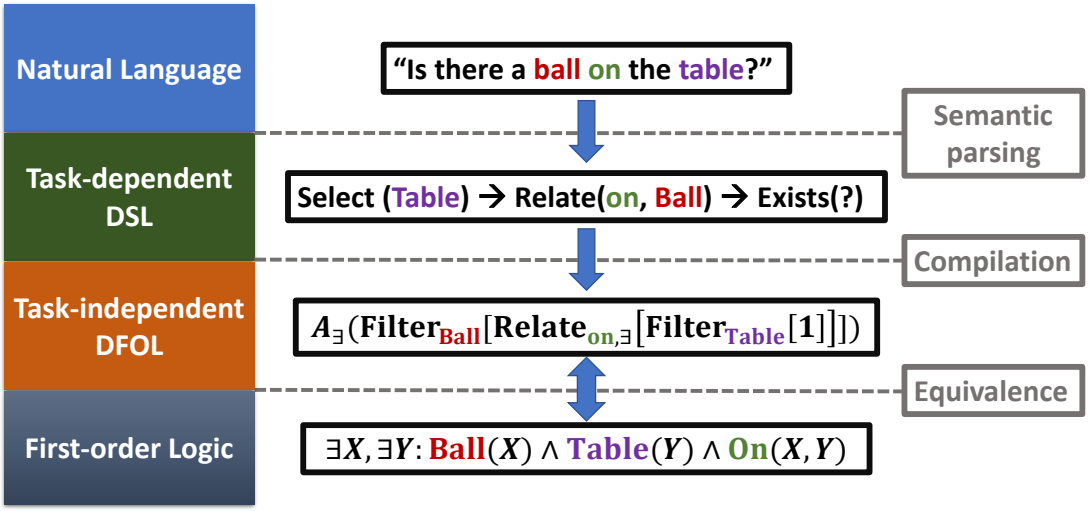
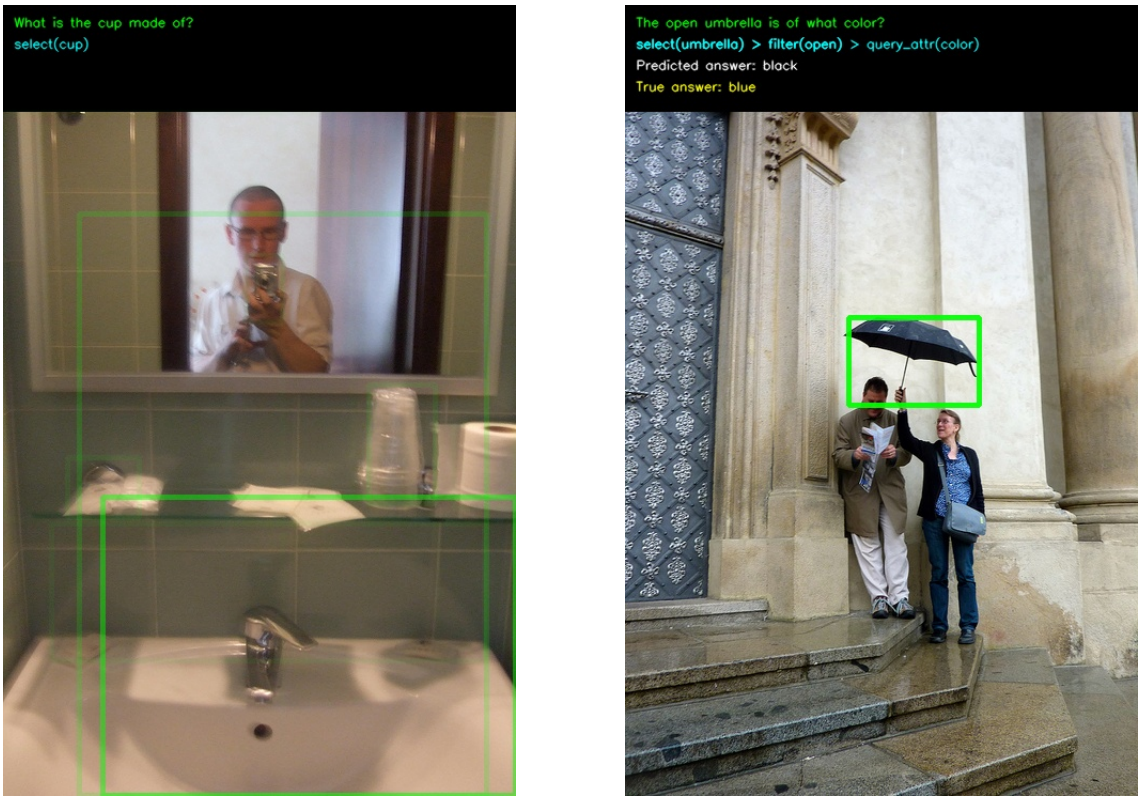


Figure 1. The language system: natural language question $\xrightarrow{\text{semantic parser}}$ DSL program $\xrightarrow{\text{compiler}}$ DFOL program $\rightleftharpoons$ FOL formula.

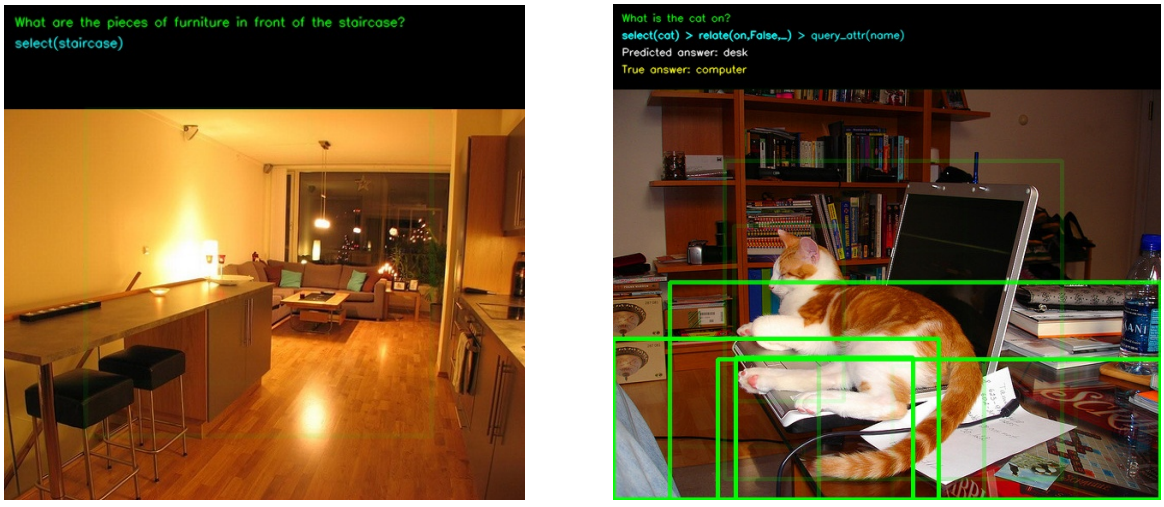
(a) (b)

Figure 2. **Hard Set: (a)** Q: *"What are the rackets are lying on the top of?"* As the attention bounding boxes show, the visual system has a hard time detecting the rackets in the first place and as a result is not able to reason about the rest of the question. **(b)** Q: *"Does the boy's hair have short length and white color?"* In this example, the boy's hair are not even visible, so even though the model can detect the boy, it cannot detect his hair and therefore answer the question correctly.



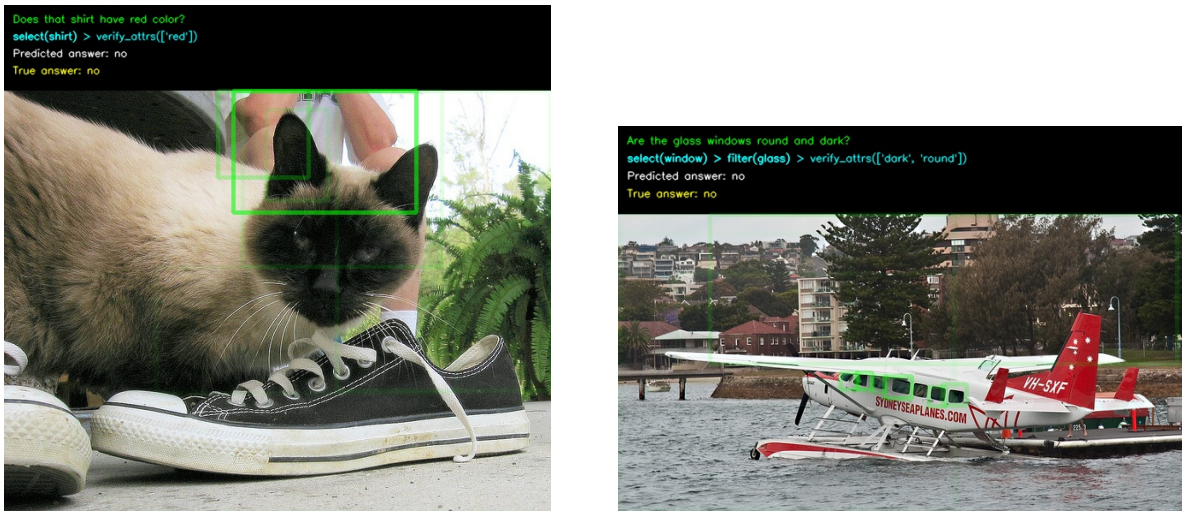
(a) (b)

Figure 3. **Hard Set: (a)** Q: *"What is the cup made of?"* As the attention bounding boxes show, the visual system has a hard time finding the actual cups in the first place as they are pretty blurry. **(b)** Q: *"The open umbrella is of what color?"* In this example, the visual system was in fact able to detect an object that is *both* "umbrella" and "open" but its color is ambiguous and can be classified as "black" even by the human eye. However, the ground truth answer is "blue" which is hard to see visually.



(a) (b)

Figure 4. **Hard Set: (a)** Q: *"What are the pieces of furniture in front of the staircase?"* In this case, the model has a hard time detecting the staircase in the scene in the first place and therefore cannot find the correct answer. **(b)** Q: *"What's the cat on?"* In this example, the visual system can in fact detect the cat and supposedly the object that cat is "on"; however, it cannot infer the fact that there is actually a laptop keyboard invisible between the cat and the desk.



(a) (b)

Figure 5. **Easy Set: (a)** Q: *"Does that shirt have red color?"* **(b)** Q: *"Are the glass windows round and dark?"*

(a) (b)

Figure 6. **Easy Set: (a)** Q: *"What side of the photo is umpire on?"* **(b)** Q: *"Are the sandwiches to the left of the napkin triangular and soft?"*

(a)

(b)

Figure 7. **(a)** Q: *"Are there any lamps next to the books on the right?"* Due to the similar color of the lamp with its background, the visual oracle assigned a low probability for the predicate ’lamp’ which in turn pushes the answer likelihood below 0.5. The calibration, however, was able to correct this by considering the context of ’books’ in the image. **(b)** Q: *"Is the mustard on the cooked meat?"* In this case, the visual oracle had a hard time recognizing the concept of ’cooked’ which in turn pushes the answer likelihood below 0.5. The calibration, however, was able to alleviate this by considering the context of ’mustard’ and ’meat’ in the visual input and boosts the overall likelihood.