Appendix to "Think before You Simulate: Symbolic Reasoning to Orchestrate Neural Computation for Counterfactual Question Answering"

Section A describes more details about the datasets. Section B details implementation of CRCG^{approx} in ASP. Section C gives the examples of the three methods for the CRAFT experiment (Section 6). Section D describes the details of how we achieve the SOTA performance on all four types of the CLEVRER questions accompanied by ablation studies. Section E presents the full ASP programs we wrote.

All experiments were done on Ubuntu 18.04.2 LTS with two 10-core CPU Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz and four GP104 [GeForce GTX 1080]. We use Clingo version 5.3.0. ASP solving time is instant.

A. Datasets

A.1. CLEVRER

The CLEVRER dataset [32] consists of 20K synthetic videos of colliding objects and more than 300K questions about objects in motion and their interactions. The videos are five seconds with a total of 127 frames and contain various objects moving and colliding with each other. The questions about the videos are divided into four categories. *Descriptive questions* are about intrinsic attributes of objects, such as color, shape, and material, and the events related to the objects, such as collision and entering/exiting the scenes. *Explanatory questions* are about if an object or a collision event is a cause for another collision event or an object exiting the scene. *Predictive questions* are about whether a collision will happen after the video ends. *Counterfactual questions* are about collisions that would or would not happen if some object in the video were removed. Except for descriptive questions, which require short answers, all the other questions are multiple-choice questions requiring one to select all true choices.

A.2. CRAFT

The CRAFT dataset [2] consists of 10K videos involving causal relations between falling and sliding objects, along with 57K questions. Compared to CLEVRER, CRAFT introduces additional events and has many different environments, including various configurations of immovable objects such as ramps and the basket. The three main categories of questions are descriptive, causal, and counterfactual. We focus only on counterfactual questions, which have 6 types. These question types are generally more complex than CLEVRER counterfactual questions, sometimes requiring multiple counterfactual simulations (e.g., "Will the large green square enter the basket if any of the other objects are removed?"), or counting (e.g., "How many objects fall to the ground if the small blue box is removed?").

The counterfactual questions in the CRAFT dataset are comprised of 6 types. Examples of each are as follows:

```
Type 1: Removing one object, does enter event happen?
If the small brown triangle is removed, will the big green
     circle fall into the bucket?
Will the large cyan circle end up in the basket if the tiny
      cyan triangle is removed?
Will the tiny purple circle fall into the container if the
     tiny purple triangle is removed?

Type 2: Removing one object, does ground collision event
     happen?
will the small purple triangle hit the floor if the large
     gray circle is removed?
will the large gray circle hit the ground if the tiny
     purple triangle is removed?
will the small purple circle fall to the ground if the big
     gray circle is removed?

Type 3: Removing one object, how many enter events happen?
How many objects go into the bucket if the tiny gray circle
      is removed?
If the tiny yellow triangle is removed, how many objects
     get into the basket?
How many objects get into the basket if the large green
     triangle is removed?

Type 4: Removing one object, how many ground collision
     events?
If the small gray circle is removed, how many objects hit
     the ground?
How many objects fall to the ground if the big red triangle
      is removed?
If the large brown triangle is removed, how many objects
     fall to the ground?

Type 5: Removing any object, does enter event happen?
Will the tiny purple triangle fall into the bucket if any
     of the other objects are removed?
Will the small gray circle fall into the bucket if any of
     the other objects are removed?
If any of the other objects are removed, will the large
     yellow triangle get into the basket?

Type 6: Removing any object, does ground collision happen?
If any one of the other objects are removed, will the tiny
     purple circle fall to the floor?
Will the large blue triangle hit the floor if any of the
     other objects are removed?
If any one of the other objects are removed, will the big
     gray cube hit the ground?
```

B. Implementation of CRCG^{approx} (Sec 3.3) using ASP

If the frame-by-frame simulator is not accessible, we use the approximation of Algorithm 1 to answer. For this, we need a few more ASP rules. For a counterfactual question $Q = \langle \mathbb{O}_c, o_1, o_2 \rangle$, and its prediction $p_Q \in \{yes, no\}$ from any baseline, let's represent the prediction p_Q as `predict(Q, yes)` or `predict(Q, no)` according to its value and represent each collision $\langle i, j, t \rangle$ in $\mathbb{C}$ with `event(i,collide,j,t)`.

Determined The result of a counterfactual question $Q = \langle \mathbb{O}_c, o_1, o_2 \rangle$ is determined to be `yes` if the collision happened in the video at a time when the states of o_1 and o_2 are not affected by the removed object.

```
determined(Q, yes) :- query(Q, qobj(I1), Event, qobj(I2)),
        same(qobj(I1), O1), same(qobj(I2),O2),
        event(O1, Event, O2, F),
        not affected(O1,F), not affected(O2,F).
```

Here, `query(Q,qobj(1),Event,qobj(2))` represents "query `Q` is about the `Event` happening between objects o_1 and o_2," and `same(qobj(1),O)` represents "o_1 is the same as the `O`-th object in the video."

The result is determined to be `no` if o_1 or o_2 is removed,

```
determined(Q, no) :- query(Q, qobj(I1), Event, qobj(I2)),
        same(qobj(I1), O1), same(qobj(I2),O2),
        removed(O1): not removed(O2).
```

or if the collision didn't happen in the video and the states of o_1 and o_2 are not affected by the removed objects.

```
determined(Q, no) :- query(Q, qobj(I1), Event, qobj(I2)),
        same(qobj(I1), O1), same(qobj(I2),O2),
        not event(O1, Event, O2, _),
        not affected(O1,_), not affected(O2,_).
```

Then, given a prediction p_Q, the answer to Q is `Res` if (i) the result of Q is determined to be `Res`, or (ii) the value of p_Q is `Res`, and the result of Q is not determined.

```
answer(Q, Res) :- determined(Q, Res).
answer(Q, Res) :- predict(Q, Res), not determined(Q, _).
```

The answer to $Q = \langle \mathbb{O}_c, o_1, o_2 \rangle$ is directly obtained from the value of `Res` in the `answer` atoms that are derived.

C. Example Flow of CRAFT Experiment

The GPT-x model used in our experiments is "text-davinci-002" and the temperature is set to 0 for the reproductivity of all experiments. A description-query pair for example video #1,179 (shown in Figure 6) from the CRAFT test set is:

```
Start. Large cyan circle collides with small yellow circle.
     Small purple triangle enters basket. Large cyan
     circle collides with small yellow circle. Small purple
     triangle collides with basket. End.

Will the tiny purple triangle end up in the basket if the
     large cyan circle is removed?
```

C.1. GPT-x Baseline

In the GPT-x baseline, a counterfactual question is asked directly in a prompt, consisting of an instruction, a video description (from the CRAFT dataset), and the counterfactual question itself. An example prompt for the example in Figure 6 is shown below.

```
Instructions: A description of a scene of moving objects
     and their physical dynamics is presented. A question
     is then asked about hypothetical changes in the scene
     and their outcomes.

Description: Start. Large cyan circle collides with small
     yellow circle. Small purple triangle enters basket.
     Large cyan circle collides with small yellow circle.
     Small purple triangle collides with basket. End.

Will the tiny purple triangle end up in the basket if the
     large cyan circle is removed? (yes or no)
```

For this example, the GPT-x responds "No", which is incorrect.

C.2. $\text{CRCG}^{approx}_{GPTx}$: Enhance GPT-x Baseline with CRCG^{approx}

We describe the whole process with the above example. From the description-query pair

```
Start. Large cyan circle collides with small yellow circle.
     Small purple triangle enters basket. Large cyan
     circle collides with small yellow circle. Small purple
     triangle collides with basket. End.

Will the tiny purple triangle end up in the basket if the
     large cyan circle is removed?
```

we first use a python script to extract the following atomic facts into an ASP program '`input.lp`'.

```
size(0,large).color(0,cyan).shape(0,circle).
size(1,small).color(1,purple).shape(1,triangle).
size(2,small).color(2,yellow).shape(2,circle).
size(95,large).color(95,black).shape(95,ground).
size(97,large).color(97,black).shape(97,basket).
collision(0,2,0).
collision(0,2,2).
collision(1,97,3).
enter(1,97,1).

counterfact(remove,qobj(0)).
feature(qobj(0),large).
feature(qobj(0),cyan).
feature(qobj(0),circle).

option(1, qobj(1), enter, qobj(2)).
feature(qobj(1),small).feature(qobj(1),purple).feature(qobj
     (1),triangle).
feature(qobj(2),large).feature(qobj(2),black).feature(qobj
     (2),basket).
```

The background knowledge about moving dynamics is encoded in '`causal.lp`' which is the one presented in Section 4. The full ASP program is given in Appendix E. The answer set of `input.lp` + `causal.lp` is:

```
sim(2,0), determined(1, yes), answer(1,yes), ...
```

The answer set contains the fact `answer(1,yes)`, meaning that the answer to the query "Will the tiny purple triangle end up in the basket if the large cyan circle is removed?" is yes, which is the correct answer.

C.3. GPT-x with CRCG guided prompt

For GPT-x with CRCG guided prompt, when CRCG derives `determined(Q,R)` for a counterfactual question `Q`, the prompt for GPT-x is replaced with the following perception question.

```
Instructions: A description of a scene of moving objects
     and their physical dynamics is presented. A question
     is then asked about the scene description.

Description: Start. Large cyan circle collides with small
     yellow circle. Small purple triangle enters basket.
     Large cyan circle collides with small yellow circle.
     Small purple triangle collides with basket. End.

According to the scene description, did the tiny purple
     triangle end up in the basket? (yes or no)
```

The GPT-x responds "Yes," which is correct.

D. Other Enhancements to Neuro-Symbolic Models for CLEVRER Tasks

D.1. Improved Object Detection (IOD)

For the descriptive and explanatory questions in the CLEVRER dataset, it is important to recognize the events in the video correctly. To help improve the accuracy of the detected results by the perception model $\mathcal{M}_p$, we implement a simple method called IOD (Improved Object Detection).

IOD is a post-processor for the perception model $\mathcal{M}_p$ to reduce its output noise and errors through two functions: *trajectory smoothing* and *topmost as center*. The input to IOD is the object trajectories output from the perception model $\mathcal{M}_p$. Since there are missing trajectories of objects at some frames due to occlusion and errors, *trajectory smoothing* draws a virtual line to connect the trajectories and uses interpolation for the frames an object is missing. *Trajectory smoothing* is applied to both NS-DR and VRDP, yielding better accuracy on all question types.

In the case of NS-DR, the Mask-RCNN outputs the mask of each object, and PropNet uses the *center* of the mask as the object's position. However, this method has a defect because occlusion could make the center of the mask move abruptly, leading to wrong answers for questions like how many objects are moving. To address this issue, the IOD module for NS-DR also applies *topmost as center*, which uses an object's topmost point (instead of center) to trace its trajectory as the topmost position is less likely to be occluded.

D.2. Simple Physics Simulator (SPS)

To better detect collisions, we introduce a Simple Physics Simulator (SPS) with two functions. The first is to predict the linear trajectory of each object after some frame. The second is collision detection.

Linear Trajectory Predictions Using several kinematic equations, the simulated object trajectory is computed. Estimations for the coefficient of friction and direction of motion for each object are computed from the perception results from $\mathcal{M}_p$ with or without IOD. This information is later used in equations to calculate the linear distance traveled and positions of each object o in each frame after o is removed from $\mathbb{O}_p$ (i.e., the set of objects whose perception states can be used at the moment) in step 6 of Algorithm 1.

From the visual perception module in Figure 1, we know x and y coordinates of objects in each video frame. We denote the position of an object to have the form of a vector $\mathbf{x} = x\hat{\boldsymbol{\imath}} + y\hat{\boldsymbol{\jmath}}$. $\hat{\boldsymbol{\imath}}$ and $\hat{\boldsymbol{\jmath}}$ are unit vectors representing the x and y direction. We can compute the approximate velocity of an object in some frame up to the end of the video as:

$$\mathbf{v}_{i+t} = (x_{i+t} - x_i)\hat{\boldsymbol{\imath}} + (y_{i+t} - y_i)\hat{\boldsymbol{\jmath}}$$

where $i + t \leq max_v$; max_v is the last frame in the video and is 127 as CLEVRER videos contain 127 frames; $\mathbf{v}_i$ represents the approximate velocity of this object in frame i; t is the temporal resolution, that is, the number of frames between every two consecutive positional information for x (or y). In general, the higher the temporal resolution, the less accurate the simulation. For NS-DR, since the position of each object is available every 5 frames in the perception results, we set $t = 5$. When using IOD or VRDP, $t = 1$, since positional information is available for every frame.

The magnitude of the velocity of an object with friction over time is modeled with the following equation:

$$|\mathbf{v}_{i+t}| = |\mathbf{v}_i| - g\sigma t$$

where $|\cdot|$ denotes the vector norm, σ is the coefficient of friction, and g is the gravitational constant. We use the last two perception data (i.e., the two perception states for each object at frames i and $i+t$) and solve for $g\sigma$ for each object. With the frictional term $g\sigma$ and the velocity v_i of each object, we can approximate the magnitude of velocity $|\mathbf{v}_{i+t}|$ at frames before the object comes to a stop. Furthermore, we can approximate the distance traveled by an object in t frames with:

$$\boldsymbol{\Delta}\mathbf{x_i} = |\mathbf{v_i}| \cdot t$$

Finally, we get the simulated trajectory of all objects by splitting up the distance traveled into the x and y components. Figure 7 shows an example simulated trajectory in the 2D space. Note that all objects need to be simulated either from the frame identified by a sim node (e.g., the yellow object in Figure 7) or from frame max_v (e.g., the red object in Figure 7).

Collision Detection To detect a collision, we check the distance of each object relative to every other object at each time frame. If they are within some threshold, then we detect a collision. The threshold we use is 23.0 units, learned from the validation set.

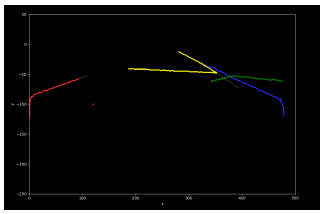

Figure 7. Simple Physics Simulator (SPS) output for the question "Without the green cylinder, what will happen?" The colored lines are in-video trajectories detected by the perception model $\mathcal{M}_p$ with IOD. The gray dashed lines are the full counterfactual trajectories computed by SPS without the green object. The yellow object hits the green object in the video, but SPS simulates the yellow object's physics trajectory from frame 46, just before this collision would happen, according to the `sim(0,46)` fact present in the answer set.

D.3. Achieving SOTA for the CLEVRER Challenge

The IOD module serves as a post-processor of a perception model to improve its perception accuracy, thus is applicable to all question types. The SPS module is only applicable to predictive and counterfactual questions.

Descriptive and Explanatory Question Answering

Table 6 shows the improvements due to IOD on the 54,990 descriptive questions in the validation set. NS-DR achieves 88.02% accuracy for descriptive questions, and with IOD, which makes the prediction to adhere to physical constraints, the accuracy is improved to 95.46%. VRDP's 93.79% accuracy is improved to 96.64% with IOD.

Table 6. Ablation study on descriptive questions in the validation set. (IOD: improved object detection)

Model	Per Ques.(%)
NS-DR	88.02
VRDP	93.79
NS-DR + IOD	95.46
VRDP + IOD	**96.64**

Explanatory questions ask about the cause of an event (collision or object exiting) that happens in the video, so like descriptive questions, they do not require physics simulation. We apply the same enhancement IOD. Table 7 shows the result of each enhancement on the 8,488 explanatory questions and 30,697 options in the validation set. Overall, we achieve 99.68% question accuracy over baseline NS-DR's 79.31%.

The improvements we made for descriptive and explanatory questions are relatively simple. However, it is also worth noting that such simple improvements achieve near 100% accuracy. Such is not the case with end-to-end models.

Predictive Query Answering Recall that predictive questions are about collision events after the video ends. When NS-DR evaluates predictive questions, the symbolic executor calls the functional module *unseen_events*, which returns the post-video collision events that PropNet generates. It then checks if any of these predicted collision events match one of the collision options in the question. Like counterfactual QA, it is essential to use a simulation to predict the movement, but we observe that the PropNet prediction is the primary source of errors in the validation set, and the errors are overwhelmingly false negatives ($\approx$ 89.06%). We find that VRDP also has a high false negative rate of 91.8% (269 out of 293 errors in the validation set) though the errors are significantly less than NS-DR.

To alleviate the false negative issue, we use the simple physics simulator (SPS) (Section D.2) that computes objects' linear trajectories and collision events using physics equations. The inputs to SPS are the trajectories and collision events generated from the IOD module (Section D.1). These additional collision events found by SPS are appended to the set of post-video collision events predicted in the baseline. Finally, the answer is computed by the program executor with the functional programs generated from the baseline question parser.

Table 8 shows the results of applying IOD and/or SPS on the baselines NS-DR and VRDP for the 3,557 predictive questions and 7,114 options in the validation set. Starting from NS-DR, our best enhancements yield an additional 9.31% question accuracy and, starting from VRDP, an additional 1.04% question accuracy. Adding only the SPS module to the baseline shows a noticeable improvement because the module could find a significant number of missing predictions.

Table 7. Ablation study on explanatory questions in the validation set. (IOD: improved object detection)

Model	Per Opt.(%)	Per Ques.(%)
NS-DR	87.19	79.31
NS-DR + IOD	**99.90**	**99.68**
VRDP	92.98	89.00
VRDP + IOD	99.28	98.82

Table 8. Ablation study on predictive questions in the validation set. (IOD: Improved object detection, SPS: Simple Physics Simulator)

Model	Per Opt.(%)	Per Ques.(%)
NS-DR	83.68	70.03
NS-DR + SPS	86.91	76.61
NS-DR + IOD + SPS	89.50	79.34
VRDP	95.88	91.90
VRDP + SPS	**96.43**	**92.94**
VRDP + IOD + SPS	95.83	91.82

Counterfactual Query Answering NS-DR and VRDP address counterfactual questions by the simulation to predict collision events when some object is removed. For the simulator, NS-DR uses PropNet and VRDP uses an impulse-based differentiable rigid-body simulator. As with predictive questions, the physics simulation often makes mistakes. For NS-DR baseline, out of the total 7,337 errors in the validation options, we find that 3,050 (41.57%) of them are false positives (i.e., the collision in the option is incorrectly predicted in the set of collision events produced by PropNet) and 4,287 (58.43%) are false negatives (i.e., the collision in the choice should be predicted by PropNet but not). VRDP performs much better, with 1756 total errors, where 49.1% are false positives and 50.9% are false negatives.

Table 9 shows an ablation study on different modules, including our main method CRCG and two simple modules IOD and SPS, on the 9,333 counterfactual questions and 33,051 choices in the validation set.

For NS-DR, we applied CRCG^{approx} to directly enhance the predictions of NS-DR, which we denote by CRCG$^{approx}_{NSDR}$. We also applied IOD and SPS to improve the accuracy of the perception states and the simulated states. Table 9 shows that each enhancement could improve the baseline NS-DR's performance and the best accuracy, 91.49% per option and 75.39% per question, is achieved when all enhancements are applied. Consider the best combination, i.e., row (e) in Table 9, where we use CRCG$^{approx}_{NSDR}$ with IOD and SPS. Among 33,051 question-option pairs in the validation dataset, the results for 20,234 are `determined` by CRCG^{approx} with an accuracy of 97.07%. For the remaining 12,817 question-option pairs, the prediction is the same as NS-DR enhanced with IOD and SPS and the accuracy is 82.75%.

For VRDP, we applied CRCG using the perception model and the simulation model in VRDP, which we denote by CRCG$_{VRDP}$. For comparison, we also applied CRCG^{approx} to directly enhance the predictions of VRDP, which we denote by CRCG$^{approx}_{VRDP}$. Table 9 shows that, while CRCG^{approx} could improve the accuracy of the predictions from VRDP, the accuracy can be further improved with CRCG, which models the influence from other simulated objects with the frame-by-frame simulation in Algorithm 1. Consider row (h) in Table 9 where CRCG^{approx} is applied to VRDP. Among 33,051 question-option pairs in the validation dataset, the results for 20,776 are `determined` by CRCG^{approx} with an accuracy of 97.38%. For the remaining 12,275 question-option pairs, the prediction is the same as the baseline VRDP and the accuracy is 94.10%. (The number of `determined` question-option pairs is slightly different for NS-DR and VRDP because they use different perception model $\mathcal{M}_p$ to detect objects $\mathbb{O}$ and in-video collisions $\mathbb{C}$.)

Note that when the answer is determined the accuracy is significantly better than when it is not because perception result is being used and is more reliable. This is evidenced by (a) vs. (c); (b) vs. (d); and (f) vs. (h). For the best results with each respective baseline, even when the answer is not determined, we do not blindly apply simulation, which yields the performance gain evidenced by (d) vs. (e) with the use of SPS and (h) vs. (i) with the use of Algorithm 1 in CRCG$_{VRDP}$. The result justifies the model's prioritization of using perception results rather than the more error-prone baseline counterfactual simulation.

Table 9. Ablation study on counterfactual questions in the validation set. The components are (1) NSDR, (2) VRDP, (3) causal reasoning with ASP (CRCG), (4) Improved object detection (IOD), and (5) the simple physics module (SPS)

Model	Opt.(%)	Ques.(%)
(a) NS-DR	73.98	41.55
(b) NS-DR + IOD	75.99	43.05
(c) CRCG$^{approx}_{NSDR}$	86.22	64.59
(d) CRCG$^{approx}_{NSDR}$ + IOD	88.95	70.78
(e) CRCG$^{approx}_{NSDR}$ + IOD+ SPS	91.49	75.39
(f) VRDP	94.69	84.11
(h) CRCG$^{approx}_{VRDP}$	95.80	87.21
(i) CRCG$_{VRDP}$	**96.16**	**88.13**

E. ASP Programs

E.1. ASP Input Generation

The first step in CRCG in Figure 1 is to turn (i) questions and options extracted by the Question Parser and (ii) object features and collision events extracted by the Visual Perception module into ASP facts.

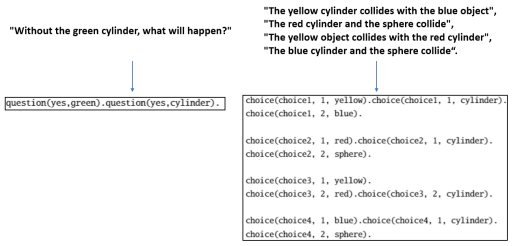

Figure 8. Conversion of a question and choices into ASP facts.

For instance, for the question "Without the green cylinder, what will happen?" and its four options in the CLEVRER dataset, Figure 8 shows the conversion into ASP facts, which is done by a Python script.

Figure 9 shows the conversion from IOD improved object detection results (from the visual perception module) into ASP facts. These ASP facts along with any post-video collisions detected by the simulation module (without removing any objects) make up the `input.lp` file in Appendix E.2. The post-video collisions as input are neces-

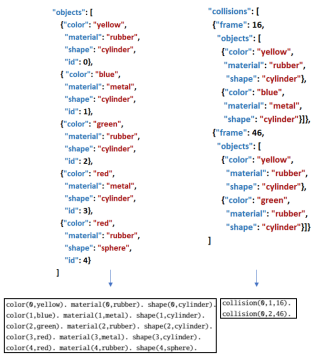

Figure 9. Conversion of improved object detection (video frame parser) results into ASP facts.

sary because the counterfactual questions in the CLEVRER dataset ask about hypothetical events that may happen during the counterfactual simulation for the duration of the video *and* some time after (we simulate/reason about up to 185 frames, 58 more than the video has).

E.2. The Full ASP Program for CRCG

Below we show the full ASP programs, including "`input.lp`" and "`causal.lp`", that realize our methods in Sections 3.2 and 3.3. The ASP program "`input.lp`" is constructed from video #147, question #15 in the validation set of CLEVRER dataset. The ASP program "`causal.lp`" is general and is applied to all examples.

E.2.1 `input.lp`

`input.lp` encodes the information from a question, its 4 choices, and the corresponding video.

```
color(0,yellow). material(0,rubber). shape(0,cylinder).
color(1,blue). material(1,metal). shape(1,cylinder).
color(2,green). material(2,rubber). shape(2,cylinder).
color(3,red). material(3,metal). shape(3,cylinder).
color(4,red). material(4,rubber). shape(4,sphere).

collision(0,1,16).
collision(0,2,46).
collision(3,0,155).

question(yes,green).question(yes,cylinder).

choice(1, 1, yellow).choice(1, 1, cylinder).
choice(1, 2, blue).

choice(2, 1, red).choice(2, 1, cylinder).
choice(2, 2, sphere).

choice(3, 1, yellow).
choice(3, 2, red).choice(3, 2, cylinder).

choice(4, 1, blue).choice(4, 1, cylinder).
choice(4, 2, sphere).
```

E.2.2 `causal.lp`

"`causal.lp`" encodes general knowledge about the causal graph and related definitions.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rules as interface to turn the atoms in input.lp into
    more general forms
%   * New atoms:
%     counterfact(remove, qobj(I))
%     option(OptionIdx, qobj(I1), Event, qobj(I2))
%     feature(qobj(I), Feature)
%     query(negated) --- which represents "the question is
    asking about something not happening"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

counterfact(remove, qobj(0)).

option(C, qobj(C*10 + 1), collide, qobj(C*10 + 2)) :-
    choice(C,_,_).

feature(qobj(0), Feature) :- question(_,Feature).
feature(qobj(C*10 + I), Feature) :- choice(C, I, Feature).

query(negated) :- question(no,_).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Supress warnings of "atom does not occur in any rule head
    "
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#defined size/2.
#defined enter/3.
#defined query/3.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Helper atoms
%   * turn size/2, color/2, shape/2 into feature/3 and
    feature/2
%   * immovable/1 denotes "background" objects that will
    never move
%   * event/4 denotes the events in {collide, enter}
%   * pos_result/1 denotes the possible result in {yes, no}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

feature(O, size, V) :- size(O, V).
feature(O, color, V) :- color(O, V).
feature(O, shape, V) :- shape(O, V).
feature(O, material, V) :- material(O, V).

feature(O, V) :- feature(O, _, V).

immovable(O) :- feature(O, shape, basket).
immovable(O) :- feature(O, shape, ground).

event(O1, collide, O2, Frame) :- collision(O1, O2, Frame).
event(O1, enter, O2, Frame) :- enter(O1, O2, Frame).

pos_result(yes; no).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rules for the causal graph
%   * same/2 identify the objects in query with the objects
    in video
%   * removed/1 denotes the removed object(s)
%   * timestamp/1 denotes the frames with event happening
%   * ancestor/4 determines the ancestor relationships
    between 2 collisions
```

```
%   * affected/2 denotes which nodes in the graph are
      affected by the removed object
%   * sim/2 represents sim node, which denotes when to
      start simulation for an object
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Each object in query should be the same as an object in
      video

same(qobj(I), O) :- feature(O,_,_), feature(qobj(I),_),
      feature(O,_,T): feature(qobj(I),T).

% Define removed object(s)

removed(O) :- counterfact(remove, qobj(I)), same(qobj(I), O
      ).

% Find all timestamps to be considered in the causal graph

timestamp(T) :- collision(_,_,T).
timestamp(T) :- enter(_,_,T).

% Collision is symmetric

collision(O2,O1,T) :- collision(O1,O2,T).

% The ancestor relation is introduced either by the same
      object with different frames, or by a collision, or

ancestor(O,T1,O,T2) :- feature(O,_,_), timestamp(T1),
      timestamp(T2), T1<T2, not immovable(O).
ancestor(O1,T,O2,T) :- collision(O1,O2,T), not immovable(O1
      ), not immovable(O2).
ancestor(O1,T1,O2,T2) :- ancestor(O1,T1,O3,T3), ancestor(O3
      ,T3,O2,T2), (O1,T1)!=(O2,T2).

% Find the nodes (i.e., object states) in the graph that
      are affected

affected(O,T) :- removed(O), collision(_,_,T).
affected(O,T) :- removed(O'), ancestor(O',T',O,T).
% If we can remove "anything", every node that has an
      ancestor is affected
affected(O,T) :- counterfact(remove, any), ancestor(O',_,O,
      T), O!=O'.

% Find the sim nodes in the graph

sim(O,T) :- not removed(O), affected(O,T), T<=T': affected(
      O,T').

% The result is determined to be yes if the collision
      happened in the video at a time when the states of o1
      and o2 are not affected by the removed object.

determined(Q, yes) :- option(Q, qobj(I1), Event, qobj(I2)),
    same(qobj(I1), O1), same(qobj(I2),O2),
    event(O1, Event, O2, F),
    not affected(O1,F), not affected(O2,F).

% The result is determined to be no if O1 or O2 is removed.

determined(Q, no) :- option(Q, qobj(I1), Event, qobj(I2)),
    same(qobj(I1), O1), same(qobj(I2),O2),
    removed(O1): not removed(O2).

% The result is determined to be no if the collision did
      not happen in the video and the states of O1 and O2
      are not affected by the removed objects.

determined(Q, no) :- option(Q, qobj(I1), Event, qobj(I2)),
    same(qobj(I1), O1), same(qobj(I2),O2),
    not event(O1, Event, O2, _),
    not affected(O1,_), not affected(O2,_).

% we answer negated result if the query is asking about an
      event not happening

answer(Idx, Ans) :- determined(Idx, Res),
    pos_result(Res), pos_result(Ans),
    Res = Ans: not query(negated);
    Res != Ans: query(negated).

% we answer the count if the query is about counting events

answer(N) :- query(counting, Event, qobj(I)),
    same(qobj(I), O),
    N = #count{Ox: event(Ox,Event,O,_), not removed(Ox)},
    not sim(Ox,_): feature(Ox,_,_).

% we answer tbd if no result is predicted

{answer(Idx, tbd)} :- option(Idx,_,_,_).
:- option(Idx,_,_,_), #count{Res: answer(Idx, Res)} = 0.
:- option(Idx,_,_,_), answer(Idx, tbd), #count{Res: answer(
      Idx, Res)} > 1.

#show sim/2.
#show answer/2.
```