Simulation-based Temporal Projection of Everyday Robot Object Manipulation

Lars Kunze, Mihai Emanuel Dolha, Emitza Guzman, Michael Beetz
Intelligent Autonomous Systems Group
Technische Universität München
80333 Munich, Germany
{kunzel,dolha,ortega,beetz}@cs.tum.edu

ABSTRACT

Performing everyday manipulation tasks successfully depends on the ability of autonomous robots to appropriately account for the physical behavior of task-related objects. Meaning that robots have to predict and consider the physical effects of their possible actions to take.

In this work we investigate a simulation-based approach to naive physics temporal projection in the context of autonomous robot everyday manipulation. We identify the abstractions underlying typical first-order axiomatizations as the key obstacles for making valid naive physics predictions. We propose that temporal projection for naive physics problems should not be performed based on abstractions but rather based on detailed physical simulations. This idea is realized as a temporal projection system for autonomous manipulation robots that translates naive physics problems into parametrized physical simulation tasks, that logs the data structures and states traversed in simulation, and translates the logged data back into symbolic time-interval-based first-order representations. Within this paper, we describe the concept and implementation of the temporal projection system and present the example of an egg-cracking robot for demonstrating its feasibility.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Design, Experimentation

Keywords

Temporal Projection, Naive Physics, Simulation, Cognitive Robots

1 Introduction

Accomplishing everyday manipulation tasks successfully requires robots to predict the consequences of actions before committing to them: the robot has to decide where and how hard to hit an egg in order to open it without damaging its content (see Figure 1). Or, it should reason about whether it is necessary to hold a cup upright to avoid spilling the coffee inside. To make such decisions the robot has to predict the changes of the physical state caused by its actions.

To compute the consequences of picking up a cup of coffee too rapidly we can model the setting as a fluid dynamics problem and solve the respective equations for the variables of interest. This way the robot could compute the fluid flow caused by the robot action and derivated variables. However, these computations do not readily provide the information needed to choose the appropriate action parametrization. It is more informative to predict whether or not a given action parametrization will cause coffee to be spilled. Abstracting the reality into a small qualitative state space, such as coffee spilled or not spilled will also cut down the search space for action selection and thereby make the search more tractable.

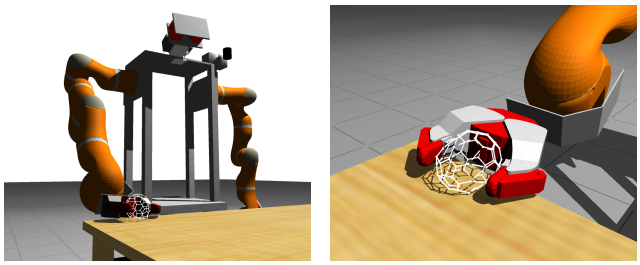

Figure 1: Robot TUM-Rosie manipulates egg in simulation.

Researchers in Artificial Intelligence have investigated approaches to represent and reason about such knowledge under the notion of *naive physics* and *commonsense reasoning*. The attempt to formulate and automate this knowledge using first-order logic has received most of the attention so far. Researchers in qualitative reasoning [18] have formalized various physics problems. The objectives of this approach are most comprehensively stated in Hayes's Naive Physics Manifesto [8]. More recently, the Common Sense Problem Page [14] lists challenge problems. Most relevant are attempts to so-called mid-size axiomatizations [15].

The basic idea is to formalize the laws of physics and situations as logical axioms in an abstract and qualitative language and then deduce the predictions of what will happen from these axiomatizations. Unfortunately, the formalizations tend to become very lengthy and often it is difficult to make the right predictions based on axiomatizations of qualitative physics. One of the main reasons is that logical axiomatizations often quantify over the values of state variables or abstract away from some state variables assuming that they are not relevant for valid predictions. However, when considering actions such as cracking an egg the effects of actions can vary largely with small changes of action parametrizations. The effects depend on where exactly and how hard the egg is hit, how strong and where it is held, and on the exact state of the egg's yolk, etc. Without exactly knowing the values of all state variables it might be impossible to predict the action effects.

Indications of these difficulties are the number and the restrictions of action logics that try to capture phenomena such as concurrent actions, the size of axiomatizations of simple physical phe-

Cite as: Simulation-based Temporal Projection of Everyday Robot Object Manipulation, Lars Kunze, Mihai Emanuel Dolha, Emitza Guzman, and Michael Beetz, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 107-114.

nomena for problems on the Common Sense Problem Page, or, the impossibility to perform certain predictions in a qualitative representations, such as predicting whether a robot will see a certain object when it navigates through the environment while at the same time turning its camera.

These problems do not occur in physics simulations where physics engines (such as ODE[1] or Bullet[2]) can simulate such phenomena without problems because they apply accurate dynamics models at a fine level of granularity.

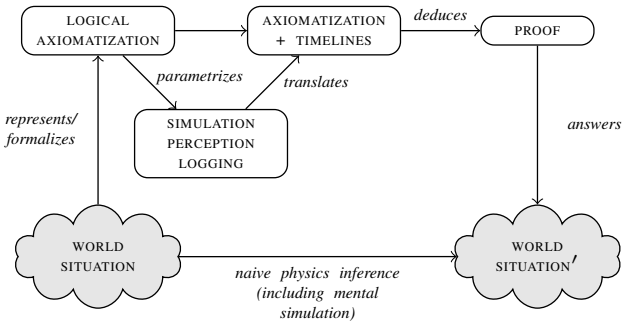

Figure 2: Naive physics inference scheme.

In this paper we combine the ideas of qualitative reasoning about courses of action and their physical effects and having accurate and realistic modeling as depicted in Figure 2. We do so by translating qualitative physics problem formalizations into a parametrized simulation problem, performing a detailed physics-based simulation, logging the state evolution into appropriate data structures and then translating these subsymbolic data structures into an interval-based first-order symbolic/qualitative representation of the respective episode. The resulting fact-base is then used to infer the answers to the qualitative reasoning problems.

The key contribution of this paper is the combination of first-order symbolic representation with physics-based simulation as a inference mechanism for predicting the effects of actions. This combination provides the best of both worlds: it provides the structure and compactness of symbolic representations *and* the realism and accuracy of physics-based simulation. Taken together the robot can predict consequences of actions such as whether an egg will break when the robot performs a specific parametrized move, whether a table will be clean after wiping it with a sponge or whether the sponge needs to be pressed out before, whether using a specific parametrization of a pick up action would cause the coffee in a cup be spilled. The point is that while these predictions are symbolic they are computed from realistic models. Combining the simulation with sampling in the state space as well as in the parametrization space of actions also allows for probabilistic predictions.

In the remainder of the paper we proceed as follows: First, we shortly revisit a well-known problem in naive physics, namely *cracking an egg*. Second, we explain how our approach addresses problems of this kind by tightly integrating logic-based reasoning and physics-based simulation. Third, we demonstrate the feasibility of our approach through experiments. Finally, we conclude after discussing related work.

2 Cracking An Egg

In this paper we take the cracking of an egg as our running example. Egg cracking has been proposed by [3] as a challenge problem for logical formalization and reads as follows:

> *"A cook is cracking a raw egg against a glass bowl. Properly performed, the impact of the egg against the edge of the bowl will crack the eggshell in half. Holding the egg over the bowl, the cook will then separate the two halves of the shell with his fingers, enlarging the crack, and the contents of the egg will fall gently into the bowl. The end result is that the entire contents of the egg will be in the bowl, with the yolk unbroken, and that the two halves of the shell are held in the cook's fingers."*

Solutions to this problem should not only characterize aspects mentioned above but also account for variants of the problem:

> *"What happens if: The cook brings the egg to impact very quickly? Very slowly? The cook lays the egg in the bowl and exerts steady pressure with his hand? The cook, having cracked the egg, attempts to peel it off its contents like a hard-boiled egg? The bowl is made of looseleaf paper? of soft clay? The bowl is smaller than the egg? The bowl is upside down? The cook tries this procedure with a hard-boiled egg? With a coconut? With an M & M?"*

The cracking an egg problem poses many challenges, especially in the context of everyday robot manipulation. In order to solve it we regard the following aspects to be substantial: First, the abstraction level of a formalization should reflect the sensing and acting capabilities of the manipulating robot. Second, variants should be handled without the need of explicit modeling. And third, concurrent actions and events should be taken into account.

3 Temporal Projection

Let's now consider how simulation-based temporal projection infers answers to naive physics problems like cracking an egg or pouring coffee to a cup. After giving a short overview of the overall system, we present each part in more detail.

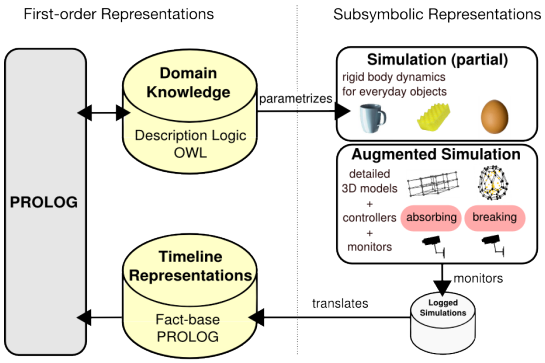

Figure 3: Simulation-based temporal projection system.

3.1 System Overview

An overview of the proposed simulation-based temporal projection system is shown in Figure 3. We formalize task-relevant domain knowledge about object classes and individuals within an ontology. Assertions about individual objects are stored in a knowledge base and used for automatically parametrizing a physics-based simulation. Within the simulator we describe everyday objects with detailed 3D models, augment the descriptions with object controllers that compute physical phenomena not covered by the rigid body dynamics, e.g. breaking of objects, and attach monitoring routines to objects in order to collect object specific data. For a task like egg

[1] http://www.ode.org
[2] http://www.bulletphysics.com

cracking, we run simulations with differently parametrized control programs, whereby states of objects are monitored and logged. Logged simulations are then translated into time-interval-based representations, called timelines. Finally, we use PROLOG for reasoning about the generated timelines. As can be seen in Figure 3, for its reasoning PROLOG also accesses the domain knowledge.

3.2 Domain Knowledge

We use first-order representations to formalize domain knowledge. Within our approach, we describe general physical knowledge about object types and their properties as well as specific knowledge about individuals in Description Logic (DL). In the following we explain what kind of knowledge we represent.

For representing domain knowledge in DL, we use the semantic web ontology language OWL[3]. We build our representations on OpenCyc's[4] upper-ontology and extend type and property descriptions whenever necessary. For example, let's have a closer look at the physically relevant knowledge about eggs and how it can be formalized in DL. We consider an egg as consisting of an eggshell and its content, i.e. egg white and egg yolk. An eggshell is a solid rigid (but fragile) container that has a shape, a mass, and extensions in space. Since the eggshell is fragile it can break. The egg's content is a liquid which has a viscosity and a mass. Figure 4 depicts a simplified excerpt of the ontology that shows type, relation and property information about eggs. For describing a specific situation individuals of relevant objects and their properties are explicitly asserted, e.g., an individual of type *Egg*, *egg3*, has *eggshell3* and *yolk3* as its parts, where *eggshell3* and *yolk3* have a mass of 0.01 and 0.04 respectively. Other properties and relations are specified similarly. For a specific task like egg cracking information about all relevant objects is asserted in the knowledge base. These assertions build the basis for parametrizing the physics-based simulation which is explained in the next section.

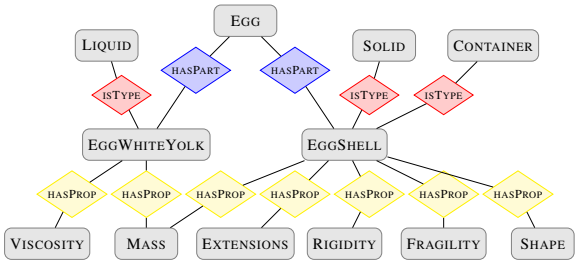

Figure 4: Ontology showing physical aspects of eggs.

3.3 Physics-based Simulation

Within our approach, we utilize a physics-based simulator, namely Gazebo[5], for computing the effects of robot actions, object interactions and other physical events.

For the computation, we parametrize the simulator on the basis of the logical axiomatization, i.e. the domain knowledge, run simulations and log data of features like position, velocity, forces, and contact points between objects over time. After explaining shortly how a physics-based simulator computes physical effects generally, we present how the Gazebo simulator can be configured and how we derive a configuration based on the assertions in the knowledge base.

Generally a physics-based simulator works as follows: the simulator starts its computation of physical effects based on an initial configuration. Then it periodically receives motor control commands which are translated into forces and updates the state of the simulated world according to physical laws. Within each tiny update step, forces are applied to affected objects by considering both the object's current dynamic state and its properties like mass and friction. Later we explain how we augment the simulation in order to account for physical phenomena like breaking or absorbing.

The initial configuration of the Gazebo simulator is based on an XML file, called *world file*. The world file describes properties of the simulation, specifies parameters for the physics engine (ODE) and describes all things occurring in the world, including robots, sensors and everyday objects. The following excerpt of a world file shows entries for the physics engine and the objects *eggshell3* and *yolk3*.

```
<gazebo:world ...>
  <physics:ode>
    <stepTime>.006</stepTime>
    <gravity>0 0 -9.8</gravity>...
  </physics:ode>
  <model:physical name="eggshell3">
    <xyz>0 0 1.23</xyz>
    <rpy>0 0 0</rpy>
    <include embedded="true">
      <xi:include href="../models/eggshell3.model" />
    </include>
  </model:physical>
  <model:physical name="yolk3">...
</gazebo:world>
```

Within a world file each object has its own model description. Such model descriptions comprise mainly the object's shape and a set of physical properties like size, mass, and rigidity. Figure 5 visualizes some parameters for both models: *eggshell3* and *yolk3*. These models configurations are derived from the information stored in the knowledge base. When properties are not explicitly specified within the knowledge base, we simply assume default values.

To simulate physical phenomena like breaking objects we augment the model descriptions, how this is realized is presented in the next section.

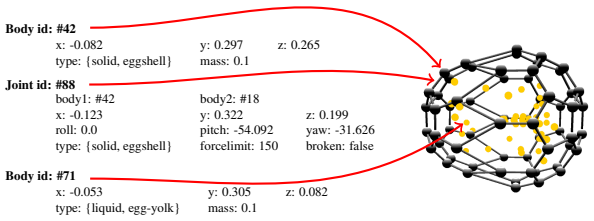

Figure 5: Modeling shape and physical properties of an egg. The shape of the egg is modeled with a graph-based structure of bodies which are linked by joints. The physical properties of these individual bodies and joints, which are shown exemplarily on the left side, determine the physical properties of the whole egg, e.g. its mass and fragility.

3.4 Augmented Simulation

The Gazebo simulator is designed for simulating robots, sensors and objects, whereby physical aspects of objects and their interactions are more or less limited to rigid body dynamics. Since we want to simulate naive physics problems with phenomena like breaking, cutting, mixing, cooking, baking, or melting we augment object model descriptions with detailed shape models, controllers for simulating physical phenomena, and monitors for logging states of objects. The extended model descriptions are collected in a library for simulating phenomena of everyday physics.

[3] http://www.w3.org/2004/OWL
[4] http://www.opencyc.org
[5] http://playerstage.sourceforge.net/gazebo/gazebo.html

Instead of modeling objects as rigid bodies, we describe the shape of objects similar to work by [9] with graph-based structures which allow us to inspect physical aspects at a more detailed level. Figure 5 visualizes the shape of an eggshell with egg yolk inside. The basic entities for modeling the shape of an object are *bodies* and *joints*, which are mutually connected. Properties of an object like type, mass, spatial extensions, and rigidity determine the attributes of these basic entities.

In order to simulate new classes of objects, e.g. objects that are breakable, cutable, or objects that absorb liquids we add controllers to the object model descriptions. These controllers are called within each simulation step and perform some specialized computation. The computation can be based on physical properties calculated by the simulator or on results computed by other controllers. Thereby object attributes like temperature, being wet, being dirty, or being broken can be computed. This allows us to simulate a new range of processes like filling, cutting, or breaking.

In addition to controllers, we add monitoring routines to object model descriptions to log the object's state at each simulation step. The data that is monitored and logged is specified for each object individually.

3.5 From Logged Simulations to Timelines

In this section we explain how we ground first-order representations in logged data structures of the simulator. Before we explain how log files are translated into logic, we will present the representation formalism for temporal knowledge and shortly discuss its relation to domain knowledge.

For representing temporal knowledge, i.e. object configurations and events at given time points, we make use of notations common in the *event calculus* [10] and its extensions. In the following we present predicates relevant for temporal reasoning.

The notation is based on two concepts, namely fluents and events. Fluents are conditions that change over time, e.g., a cup contains coffee: *contains(cup,coffee)*. Events (or actions) are temporal entities that have effects and occur at specific points in time, e.g., consider the action of pouring coffee: *pourTo(coffee,pot,cup)*. Logical statements about both fluents and events are expressed mainly by two predicates:

- *Holds(f,t)* and
- *Occurs(ev,t)*,

where f denotes a fluent, *ev* denotes an event and t simply denotes a point in time. The statement *Holds(f,t)* represents that fluent f holds at time t, whereas *Occurs(ev,t)* represents an occurrence of event *ev* at time t. Although fluents and events look as if they were predicates themselves, they are not: both fluents and events are reified as functions returning respective instances. Thus, by treating them as 'first-class citizens' in a first-order representation allows us to state at what points in time they hold or occur.

The relation of domain and temporal knowledge is straight forward. Domain knowledge, in particular the assertions about individual objects, characterize the initial conditions for the temporal reasoning. From the temporal reasoning point of view, the assertional knowledge holds at time point *0.0*, i.e. *Holds(f,0.0)*. That the assertional knowledge describes the initial conditions for the temporal reasoning perfectly makes sense since it is also used for parametrizing (or initializing) the simulation as we explained earlier.

Logged simulations are translated into interval-based timeline representations by using the predicates *Holds* and *Occurs*. Whenever a fluent or event is recognized an instance of its corresponding type is generated and either the *Holds* or the *Occurs* predicate is asserted for the observed timepoint. We reuse a generated instance only if the fluent or event is also valid in successive timesteps. Thereby we get an interval-based representation of timelines. Table 1 and Table 2 list examples of implemented fluents and events for which we assert predicates from the logged simulations.

Table 1: Fluents for static physical configurations.

fluent	intuitive description
contacts(o_1, o_2)	object o_1 and object o_2 contact each other
attached(o_1, o_2)	object o_2 is attached to object o_1
supports(o_1, o_2)	object o_1 supports object o_2
contains(o_1, o_2)	container o_1 contains object (or stuff) o_2
broken(o_1)	object o_1 is broken
spilled(o_1)	object o_1 is spilled

Table 2: Fluents for physical events.

fluent	intuitive description
colliding(o_1, o_2)	object o_1 and object o_2 are colliding
falling(o_1)	object o_1 is falling
moving(o_1)	object o_1 is moving
openingGripper(o_1)	robot is opening gripper o_1
closingGripper(o_1)	robot is closing gripper o_1
breaking(o_1)	object o_1 is breaking
spilling(o_1)	object o_1 is spilling over a surface

How fluents and events are grounded in the data structures of the simulator is exemplarily explained for the fluents *contacts*(o_1, o_2) and *supports*(o_1, o_2) and the events *moving*(o_1) and *breaking*(o_1).

A contact between objects is directly reported by the simulator:

$$\begin{aligned}&\mathit{Holds}(\mathit{contacts}(o_1, o_2), t_i) \Leftrightarrow \\ &\quad \mathit{Collisions} = \mathit{SimulatorValueAt}(\mathit{Collisions}, t_i) \wedge \\ &\quad \mathit{Member}(\langle o_1, o_2 \rangle, \mathit{Collisions})\end{aligned}$$

Object o_1 supports an object o_2 when there exists a contact between both objects and the maximum value of o_1's bounding box within z-dimension is slightly less or equal than the minimum value of o_2's bounding box and o_2's center of mass lies within the spatial extensions of object o_1 regarding the x-y-dimensions. The later condition is captured by the *isDirectlyBelow* predicate. Furthermore the gravity force of o_2 has to be canceled out:

$$\begin{aligned}&\mathit{Holds}(\mathit{supports}(o_1, o_2), t_i) \Leftrightarrow \\ &\quad \mathit{Holds}(\mathit{contacts}(o_1, o_2), t_i) \wedge \\ &\quad p_1 = \mathit{SimulatorValueAt}(\mathit{Pose}(o_1), t_i) \wedge \\ &\quad p_2 = \mathit{SimulatorValueAt}(\mathit{Pose}(o_2), t_i) \wedge \\ &\quad \mathit{isDirectlyBelow}(p_1, p_2) \wedge \\ &\quad \mathit{gravityForceIsCanceledOut}(o_2)\end{aligned}$$

An object o_1 is moving when its pose has changed between two successive timesteps t_j and t_i:

$$\begin{aligned}&\mathit{Occurs}(\mathit{moving}(o_1), t_i) \Leftrightarrow \\ &\quad p_1 = \mathit{SimulatorValueAt}(\mathit{Pose}(o_1), t_i) \wedge \\ &\quad p_2 = \mathit{SimulatorValueAt}(\mathit{Pose}(o_1), t_j) \wedge \\ &\quad \mathit{previousTimestep}(t_j, t_i) \wedge \\ &\quad p_1 \neq p_2\end{aligned}$$

An object o_1 is breaking in timestep t_1 when one of its joints is detached within that timestep. The controller that realizes the breaking phenomenon of objects directly reports which joints are detached in a timestep:

$$\begin{aligned}&\mathit{Occurs}(\mathit{breaking}(o_1), t_i) \Leftrightarrow \\ &\quad j_1 = \mathit{SimulatorValueAt}(\mathit{Detached}(\mathit{joint}_1), t_i) \wedge \\ &\quad \mathit{Member}(j_1, \mathit{GetJoints}(o_1))\end{aligned}$$

The next section explains how we do reasoning on the grounded fluents and events asserted in timelines.

3.6 Reasoning on Timelines

Figure 6 shows a sequence of images of a simulated egg dropped onto the floor. By examining this simple example we show how PROLOG can be used for reasoning about both timelines derived from logged simulations and domain knowledge. Let's consider the PROLOG query:

```
?- holds(F1,T1), fluentT(F1,supports), objOf(F1,egg1),
   after(T2,T1), occurs(E1,T2), objOf(E1,egg1).
```

where $F1$ and $E1$ are variables for a fluent and event respectively, $T1$ and $T2$ are time intervals, *supports* is the type of fluent $F1$, and $egg1$ denotes an individual. The query basically asks for all events $E1$ that hold for $egg1$ after $egg1$ is no longer supported. For the dropped egg example, $E1$ is bound to *falling*(*egg1*), *colliding*(*egg1*,*floor*), and *breaking*(*egg1*). The *after* relation used in the query above is one of the thirteen possible temporal relationships between time intervals [1] which we have implemented as predicates for reasoning about timelines.

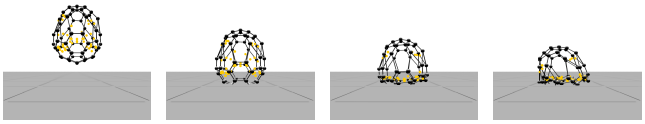

Figure 6: Simulation of an egg dropped onto the floor. From left to right: falling, colliding, breaking, and broken.

Figure 7 illustrates the complete process of naive physics reasoning for a situation where a robot wants to grasps an egg but his hand is wet. Similar to the example query above, the robot could ask what will happen if (after) it grasps the egg with its wet hand. The initial conditions describing the actual situation are taken from the knowledge base to parametrize the simulation, the fact that the hand is wet would reduce the friction of the hand. Then the simulation is run whereby states of objects are monitored and logged. After the logged simulations are translated into interval-based first-order representations the query will be answered based on the resulting timelines. Depending on the reduced friction of the hand the egg might slip away and fall onto the floor which cause the eggshell to break and the egg yolk to be spilled.

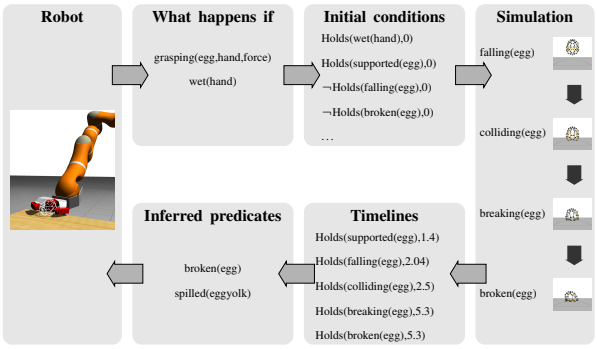

Figure 7: Complete process of the temporal projection.

In addition to fluents and events that are grounded within the data structures of the simulator, more complex events like picking up an object or an overflowing container can be defined by utilizing grounded fluents, events and additional temporal constraints. For the description of complex events we follow the notion of chronicles [7].

4 Experiments

For showing the feasibility of our approach we have conducted several robot manipulation experiments for the problem of cracking an egg as described in Section 2. In these experiments we addressed the requirements posed in the problem formulation. Furthermore we have conducted experiments for the problem of pouring and absorbing liquids.

The robot model used in our experiments is the PR2 robot platform developed by Willow Garage[6]. The PR2 has an omnidirectional base, a telescoping spine and a pan-tilt head. Each of the two compliant arms of the platform have four degrees of freedom (DOF) with an additional three DOF in the wrist and one DOF gripper. The sensor setup is comprised of a laser sensor on the base, a tilting laser sensor for acquiring 3D point clouds, two stereo camera setups and a high resolution camera in the head. The hands also have cameras in the forearms, while the grippers have three-axis accelerometers and fingertip pressure sensor arrays. The entire setup is realistically modeled and ready to use in the Gazebo simulator.

4.1 Cracking an Egg

Cracking an egg against another object and then separating (splitting) it requires a robot to be able to grasp an egg at all. Therefore we start our experiments with a scenario where a robot is supposed to simply grasp an egg lying on a table.

The first experiment consists of several trials in which a robot, in this case the PR2, is using different values of gripper force to grasp an egg. The experiment underlines the importance of a physical simulation since it allows to determine an appropriate force for grasping an egg which would not be possible by pure symbolic reasoning.

The simulation setup for this experiment is simple and consists of the PR2 robot model and the egg model lying on a table being spawned in a Gazebo environment. The robot is trying to pick up the egg by applying different forces with his gripper. It starts with the lowest force level and after each try the force is increased, the old egg is unspawned and a new one is created as the old one might be damaged during the experiment. In each trial the robot tries to pick the egg up and hold it up for a period of time. During the experiments we found three possible outcomes (see Figure 8): the egg slips out of the robot's gripper, the egg is held by the robot successfully, and the egg is crashed by the robot (Figure 9).

Within the experiment we identified four force levels as being too low for grasping the egg, three force levels as appropriate and five force levels as being too high. A video showing this experiment is available online[7].

The data structures of the simulation were logged and translated into interval-based first-order representations. Thereby the results of the experiment are made available in the logical programming environment PROLOG which is demonstrated by the following queries

```
?- occurs(E,T1), eventT(E,'ClosingGripper'),
   argsOf(E, [rightGripper, force7]), holds(F,T2),
   after(T2,T1), fluent(F,Type), argsOf(F,Args).

   E = closingGripperEvt7,
   T1 = 3.25,
   F = attachedFl1,
   T2 = 5.0,
   Type = 'Attached',
   Args = [rightGripper, egg1].

?- occurs(E,T1), eventT(E,'ClosingGripper'),
   argsOf(E, [rightGripper, force12]), holds(F,T2),
```

[6] http://www.willowgarage.com/pages/pr2/overview
[7] http://www.youtube.com/watch?v=MzMnTooXyCc

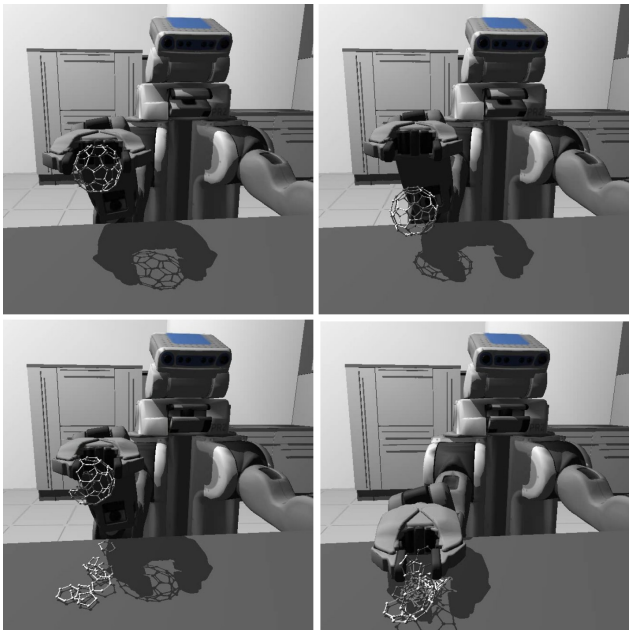

Figure 8: PR2 robot picking up an egg with different force levels (upper left: successful; upper right: egg slipping; bottom left: egg crashed, parts of the eggshell fell onto the table; bottom right: egg crashed).

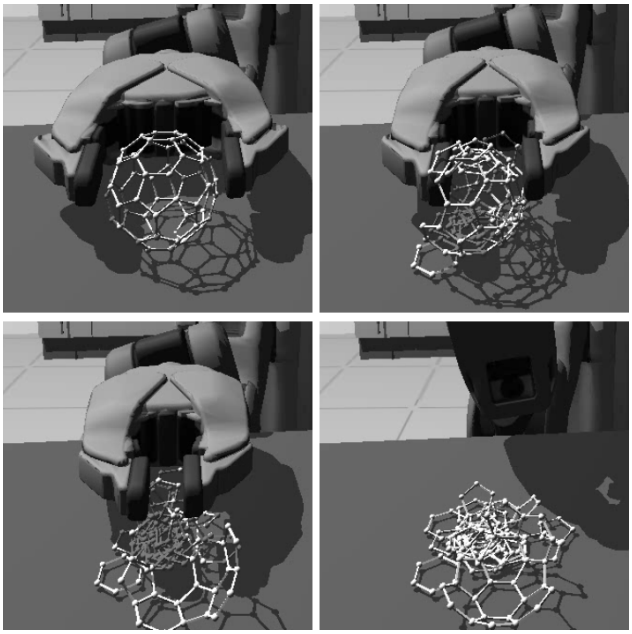

Figure 9: The egg model crashing in a grasp trial because of too much gripper force.

```
after(T2,T1), fluent(F,Type), argsOf(F,Args).

E = closingGripperEvt12,
T1 = 3.0,
F = brokenF11,
T2 = 4.25,
Type = 'Broken',
Args = [egg1]
```

where the first and the second query ask for fluents that hold after grasping the egg with *force7* and *force12*, respectively. Whereas using *force7* result in a successful trial where the egg is attached to the robot's gripper, using *force12* result in a situation where the egg is broken. Given the simulation-based temporal projections of what will happen if the robot grasps an egg with a particular force it is possible to determine an appropriate value for the grasp force parameter.

In the second experiment we used a valid grasping force to pick up an egg and test its behavior when hitting it against obstacles and tables. The egg is picked up and then is hit or pressed against an obstacle. The results here are of course dependent on the forces that affect the egg model: while hitting the egg against another object very gently would not break it, hitting it stronger or pressing it firmly against the table would produce breaking. Figure 10 shows the egg model being cracked after being hit against an obstacle. This experiment can be used to gather information on how to safely manipulate such a fragile object and how the robot's actions influence the forces applied to the object.

The last experiment was focused on egg splitting. The robot is grasping the egg from the table that's lying on the table and, after hitting it against an obstacle and cracking it, is trying to split it using his other gripper (Figure 11). This experiment was not entirely successful as the egg to be too fragile for the PR2 grippers. The result of most of the trials involved in this experiment was the cracked egg being completely crashed by the two grippers.

In contrast to the logical formalization that has been proposed by [15] our approach is able to make temporal projections about almost all aspects of the problem specification and its variants. Their theory [15] is based on roughly 70 axioms, but variations such as

- the cook brings the egg to impact very quickly or very slowly
- the bowl is upside down
- the cook tries the procedure with a hard-boiled egg, coconut, or an M&M
- the cook puts the egg in the bowl and exerts steady pressure with his hand
- the cook, having cracked the egg, attempts to peel it off its contents like a hard-boiled egg

cannot be handled without further extensions. All these variations, except the last, seem to be feasible with our simulation approach. We simply have to adapt the robot control program to induce a different manipulation behavior, to change the configu-

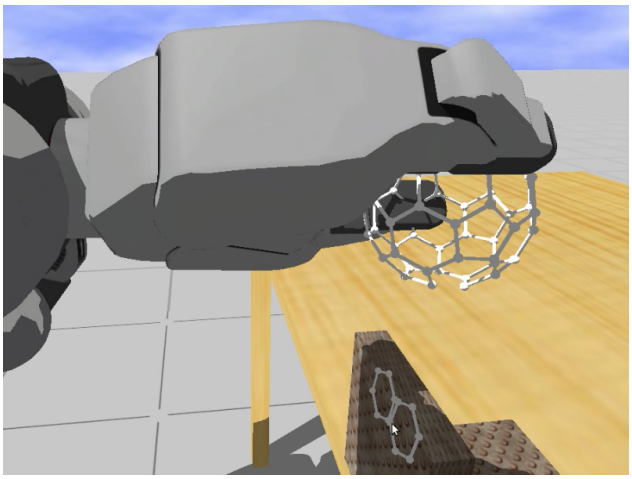

Figure 10: An egg model cracked by hitting an obstacle.

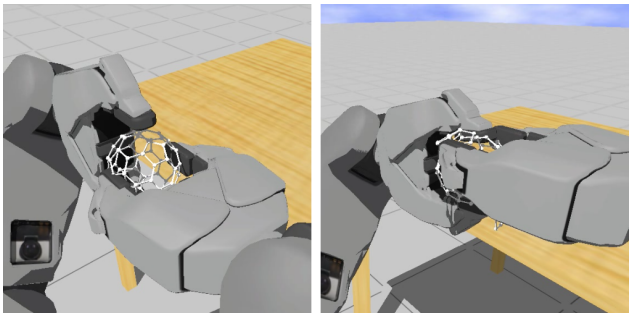

Figure 11: An egg splitting trial using the PR2 robot model.

ration of the environment, or to adjust the physical parameters of the object models, e.g. size, structure and/or fragility of objects. Although the adjustment of the physical parameters is not trivial, it seems to be much easier than the extension of a logical theory since machine learning techniques can be applied for finding the appropriate physical models.

Figure 12 shows the robot moving its arm away from the egg after breaking it. In the beginning, parts of the eggshell stuck to the gripper and fell off at a later point in time. It is impossible to model such phenomena within logical abstractions, whereas in detailed simulations they simply emerge from the laws of physics. This example strongly emphasizes the benefit of combining first-order representations with physics simulations.

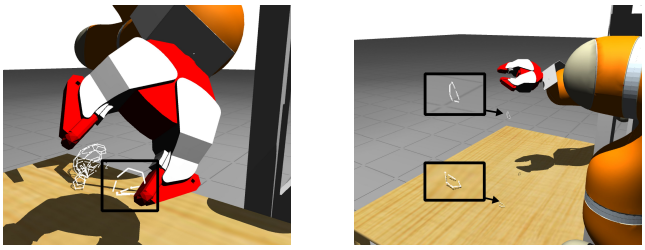

Figure 12: Grasping an egg. (1) eggshell stuck to gripper (2) eggshell fell off the gripper.

4.2 Pouring Liquids

In this scenario the robot picks up a filled container from the table, moves it above a second container, and then pours the water from the first to the second container. We looked at several variants of the problem: (a) the second container has holes, (b) the second container is upside down, (c) honey (instead of water) is poured to an upside down container (d) the task is performed successfully, meaning that the liquid completely ends up in the second container, (e) the second container is already filled which causes it to overflow as soon as the robot pours further liquid to it, meaning that some liquid is spilled on the table.

The following query asks for the conditions that hold after the pouring action:

```
?- occurs(E,T1), eventT(E,pouringTo),
   argsOf(E,[cup1,liq1,cup2]),
   after(T2,T1), holds(F,T2).
```

where in variant (a) and (e) F is bound to *contains(cup2,liq1)*, *spilled(liq1)* and *supports(table1,liq1)*, in variant (b) F is bound to *spilled(liq1)*, *supports(table1,liq1)* and *supports(cup2,liq1)*, in (c) F is bound to *spilled(liq1)* and *supports(cup2,liq1)*, and in (d) F is bound to *contains(cup2,liq1)* (Figure 13). The notably difference between variants (b) and (c) result from the fact that honey has a

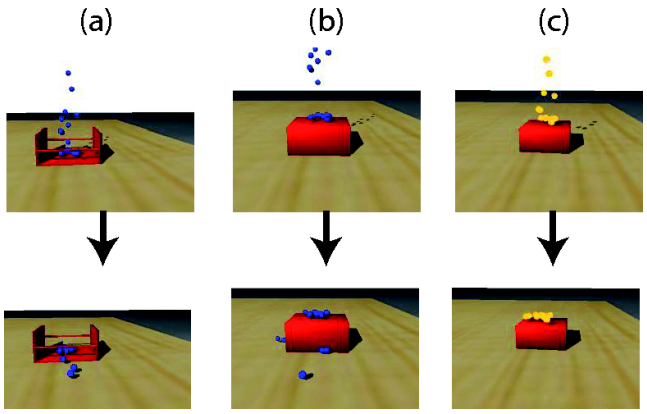

Figure 13: Pouring liquids to containers (a) Pouring liquid to container with holes (b) Pouring water to an upside down container (c) Pouring honey to an upside down container.

higher viscosity than water. Within the simulation this is reflected by the different friction values for water and honey.

In a follow-up experiment the robot cleans the table with a sponge (Figure 14). The corresponding query looks as follows:

```
?- occurs(E,T1), eventT(E,wiping),
   argsOf(E,[sponge1,table1]),
   after(T2,T1), holds(F,T2).
```

where F is bound to *absorbs(sponge1,liq1)*.

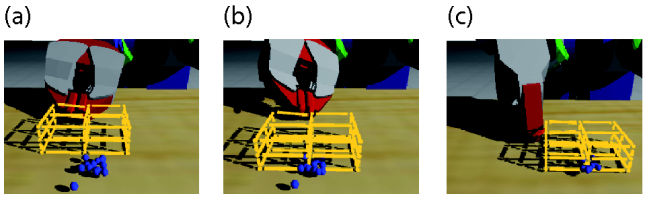

Figure 14: Wiping table with sponge (teleoperated) (a) Gripper approaches sponge (b) Gripper pushes sponge over liquid which in turn is absorbed (c) Gripper pushes sponge and both absorbed liquid and sponge move together.

5 Related Work

Solutions to a naive physics problem, namely egg cracking [3], were formulated by [11, 15] based on logical axiomatizations. Limitations of these approaches are mainly that physical details are abstracted away and that variants cannot be handled very flexibly. To overcome such limitations this work proposes a simulation-based approach: we take a logical axiomatization and translate it into a parametrized simulation problem, simulate and log simulation data, translate logged simulation data into an interval-based first-order representation which is used for answering queries about a qualitative reasoning problem.

The integration of numerical simulation and qualitative methods has been investigated before, for example, work on qualitative-numeric simulation [2] and self-explanatory simulations [5]. Work by [12] has shown an integration of numerical simulation and qualitative modeling based on the Qualitative Process Theory [4] for virtual interactive environments. But none of the approaches, we are aware of, have investigated a simulation-based approach for making predictions in the context of every robot object manipulation.

Our simulation-based approach is in a similar line of work by [9] who integrated logic and simulation for commonsense reasoning. Whereas they use a general purpose simulation, we utilize a physics-based simulator augmented with phenomena of everyday physics since we are particularly interested in naive physics reasoning for robot manipulation. Instead of looking at isolated problems, we aim for a tight integration between the our proposed reasoning system and other processes like planning, e.g., to predict whether a meal is edible when executing a specific plan for cooking pasta.

The grounding of logical predicates like *contacts*(o_1, o_2) in data of logged simulations is done similar to work by [17] who grounded semantics in visual perception. Similarly, we ground only primitive predicates in logged simulations. Complex predicates are formulated in PROLOG and are based on primitive or other complex predicates similar to definitions of symbolic chronicles [7].

The underlying idea of our approach is not restricted to problems in naive physics, but it can effectively be applied to tasks like the visibility of objects in scenes, perspective taking [13], physics-based motion planning [19], navigation in environments with non-rigid objects [6], and the prediction of interfering effects of continuous and concurrent actions [16].

6 Conclusions

In this paper we presented a simulation-based approach to naive physics temporal projection in the context of robot manipulation. Instead of making predictions based on logical axiomatizations, we propose an inference system based on physics simulations.

We developed techniques for transferring qualitative reasoning problems to physics simulations, for monitoring states of objects and logging them to appropriate data structures, for translating logs into first-order representations, and for answering queries on the resulting representations. We successfully conducted experiments that show that it is feasible to infer answers to naive physics problems based on physical simulations. We think that the effort needed to get a functionality in simulation-based inference is less than for axiomatizing the respective functionality. Additionally, details of investigated problems are not abstracted away within detailed simulations. Furthermore, inference tasks that are notoriously difficult to axiomatize are doable in simulation-based reasoning, for example, tasks including concurrent actions, soft bodies, liquids, and physical processes like mixing, overboiling, or scorching. In this work we are neither aiming for high-performance nor high-fidelity simulations, but rather exploiting simulation technologies for answering questions about naive physics problems which a abstracted in a reasonable small qualitative state space. We expect that issues related to performance and realistic models will be addressed by the game and animation film industry.

Thereby, we believe that this system provides a functionality needed for successfully accomplishing complex everyday robot manipulation tasks. In future work, we will address how robots can employ the simulation-based temporal projection approach for determining the appropriate parametrizations for their actions.

7 Acknowledgments

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys*[8].

8 References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. In *Readings in qualitative reasoning about physical systems*, pages 361–372. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[2] D. Berleant and B. Kuipers. Qualitative-numeric simulation with q3. In *RECENT ADVANCES IN QUALITATIVE PHYSICS*, pages 3–16. The MIT Press, 1992.

[3] E. Davis. Cracking an Egg. Commmon Sense Problem Page, `http://www-formal.stanford.edu/leora/commonsense/#eggcracking`, 9 1997.

[4] K. D. Forbus. Qualitative process theory. *Artif. Intell.*, 24(1-3):85–168, 1984.

[5] K. D. Forbus and B. Falkenhainer. Self-explanatory simulations: An integration of qualitative and quantitative knowledge. In *AAAI*, pages 380–387, 1990.

[6] B. Frank, C. Stachniss, R. Schmedding, M. Teschner, and W. Burgard. Real-world robot navigation amongst deformable obstacles. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 617–622, Piscataway, NJ, USA, 2009. IEEE Press.

[7] M. Ghallab. On Chronicles: Representation, On-line Recognition and Learning. In *Principles of Knowledge Representation and Reasoning-International Conference-*, pages 597–607. Morgan Kaufmann Publishers, 1996.

[8] P. Hayes. The second naive physics manifesto. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 1–36. Ablex, Norwood, NJ, 1985.

[9] B. Johnston and M. Williams. Comirit: Commonsense Reasoning by Integrating Simulation and Logic. In *Artificial General Intelligence 2008: Proceedings of the First AGI Conference*, page 200. IOS Press, 2008.

[10] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Gen. Comput.*, 4(1):67–95, 1986.

[11] V. Lifschitz. Cracking an Egg: An Exercise in Commonsense Reasoning. Presented at the 4th Symposium on Logical Formalizations of Commonsense Reasoning, 1998.

[12] J.-L. Lugrin and M. Cavazza. Making sense of virtual environments: action representation, grounding and common sense. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 225–234, New York, NY, USA, 2007. ACM.

[13] L. Marin, E. A. Sisbot, and R. Alami. Geometric tools for perspective taking for human-robot interaction. In *MICAI 2008*, 2008.

[14] R. Miller and L. Morgenstern. Common Sense Problem Page. `http://www-formal.stanford.edu/leora/commonsense`, 2009.

[15] L. Morgenstern. Mid-sized axiomatizations of commonsense problems: A case study in egg cracking. *Studia Logica*, 67(3):333–384, 2001.

[16] L. Mösenlechner and M. Beetz. Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior. In *ICAPS 2009*, 2009.

[17] J. M. Siskind. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *J. Artif. Intell. Res. (JAIR)*, 15:31–90, 2001.

[18] D. S. Weld and J. d. Kleer, editors. *Readings in qualitative reasoning about physical systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[19] S. Zickler and M. Veloso. Efficient physics-based planning: sampling search via non-deterministic tactics and skills. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 27–33, Richland, SC, 2009. IFAAMAS.

[8] `http://www.cotesys.org`