Mid-Sized Axiomatizations of Commonsense Problems: A Case Study in Egg Cracking

Leora Morgenstern
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
leora@watson.ibm.com

July 30, 2000

All true believers shall break their eggs at the convenient end: and which is the convenient end, seems, in my humble opinion, to be left to every man's conscience, or at least in the power of the chief magistrate to determine.

– Jonathan Swift, *Gulliver's Travels*

Abstract

We present an axiomatization of a problem in commonsense reasoning, characterizing the proper procedure for cracking an egg and transferring its contents to a bowl. The axiomatization is mid-sized, larger than toy problems such as the Yale Shooting Problem or the Suitcase Problem, but much smaller than the comprehensive axiomatizations associated with CYC and HPKB. This size of axiomatization permits the development of non-trivial, reusable core theories of commonsense reasoning, acts as a testbed for existing theories of commonsense reasoning, and encourages the discovery of new problems in commonsense reasoning.

We present portions of core theories of containment, falling, and pouring, integrated into Shanahan's circumscriptive event calculus, and show how these can serve as the basis of an axiomatization that partly characterizes egg cracking. We discuss several commonsense reasoning problems encountered during this research, such as the Initial Specification Problem (a relative of the frame problem that occurs in theories in which fluents can trigger actions), and the Unobtainable State Problem (the problem of determining whether or not a theorem stating that one cannot get from one state to another is meaningful).

1 Introduction

The goal of formalizing commonsense reasoning within logic is a long-standing goal of Artificial Intelligence (McCarthy 1959, McCarthy and Hayes 1969; Hayes 1985b). There are two well-known but contrasting approaches to the task: (1) the careful and painstaking formalization of very small, toy commonsense reasoning problems, in which meticulous attention is paid to details; and (2) the broad but all-encompassing formalization of large chunks of world knowledge, where the focus is on breadth rather than depth, and inaccuracy (and sometimes inconsistency) is expected and tolerated.

The first approach is exemplified by the work on temporal reasoning problems such as the Yale Shooting Problem (Hanks and McDermott 1987), the Russian Roulette Problem (Sandewall 1994), and the Suitcase Problem (Lin 1995). The scenarios of these problems often involve only a handful of axioms but have been very fruitful. Careful study has led to the characterization and analysis of

difficult problems like the frame, ramification, and qualification problems, and to the development of temporal theories in which such problems can be solved (Sandewall 1994; Shanahan 1997).

The second approach is characterized by the CYC program (Lenat and Guha 1990) and more recently, by the DARPA High-Performance Knowledge Bases (HPKB) project (Pease et al. 2000), which demonstrate that shallow but broad representations can be successfully used for solving commonsense problems.

This paper presents a formalization of a commonsense reasoning problem that in approach and scale lies somewhere between these two approaches. In contrast to toy commonsense formalizations, the idea is to develop formalizations that capture with a greater degree of accuracy the workings of the world, and which are robust enough to be extendible and reused. In contrast to large-scale formalizations, where there is a great deal of pressure to formalize large bodies of knowledge in very short periods of time, the idea is to take enough care with the formalization so that important logical and conceptual issues are not ignored or massaged away.

In addition to the emphasis on mid-sized axiomatizations, our approach is characterized first, by selecting as the domain of inquiry a reasonably complex benchmark commonsense reasoning problem, and second, by the attempt to integrate the axiomatization, to the extent possible, with existing work in formalizations of commonsense reasoning. For example, we have chosen to use Shanahan's (1997) circumscriptive event calculus (itself an integration of nonmonotonic logic with Kowalski and Sergot's (1986) event calculus) as the core temporal theory of the axiomatization.

We believe that there are several advantages to our approach to formalization.

First, the careful formalization of a non-toy problem of commonsense reasoning should result in the development of core, reusable theories of commonsense reasoning. For example, for the formalization of the egg-cracking problem, we developed portions of theories of containment, falling, and pouring, that should carry over to other problems of commonsense reasoning.

While reuse is also a goal of large-scale formalization, the speed at which such formalizations are constructed and the resulting compromise in accuracy may make reuse difficult. Indeed, the difficulty in reusing theories during the HPKB project is one of the problems that HPKB's successor project, RKF (Rapid Knowledge Formation), is meant to address. [1] We believe that our emphasis on developing reusable core theories ties in well with that aspect of the RKF project.

Second, integrating existing work in commonsense reasoning into a new formalization tests the limit of the existing theory, and possibly points out inadequacies of the existing theory. Again, scale is an important issue here. Existing theories which have thus far been used only in axiomatizations of toy problems may never have had their limits tested. On the other hand, the pressure of large projects like HPKB may entail that theories that still have kinks to work out are never candidates for integration; or, difficulties that may arise are either not noticed or passed over.

Third, the analysis of a mid-sized problem could result in the discovery of new representational issues and problems that might not appear in an artificially small toy problem. The investigations into toy problems have been very fruitful. But focusing on such problems to the exclusion of all others can lead to our ignoring problems which are crucial for progress in knowledge representation and logic. Our exercise highlighted two such problems which arise in Shanahan's circumscriptive event calculus: the Initial Specification Problem and the Unobtainable State Problem. The Initial Specification Problem arises in temporal languages that (like Shanahan's) are powerful enough to represent triggered actions. Closely related to the frame problem, it is the problem of having to specify, in an initial state, all relevant fluents, so that one can determine that no unexpected actions are triggered in the initial state. The Unobtainable State Problem is the problem of determining whether a theorem stating that there is no action to get from one state to another is meaningful, or reflects the impoverishment of the theory. Both problems are discussed in detail in Section 5.

It is of course possible, once one has discovered these problems, to create toy domains in which these

[1] See, for example, the proposal at http://reliant.teknowledge.com/RKF/proposals/SRI/SRIproposal.htm.

problems arise. The reality is, however, that unless one is lucky in one's choice of toy domain, one may not discover these issues. Tackling a larger and more realistic domain makes discovery of such problems more likely.

Finally, we should not ignore the educational and public relations role of mid-sized formalizations. The Yale Shooting Problem and its relations and descendants were invaluable in the development, during the last fifteen years, of theories of causal reasoning. However, if we cannot move past problems of this ilk, we risk losing our credibility as researchers who can advance the state of the art, and we risk losing the attention of a new generation of students and of our fellow AI researchers. Demonstrating that we can make headway on larger problems without giving up our use of logic as a tool of precision can reignite interest in and emphasize the relevance of careful logical formalization in AI.

This paper is structured as follows. In Section 2, we give the problem statement and discuss the methodology used for formalization. Section 3 presents the formalization. The focus is on developing portions of theories of containment, falling, pouring, and breaking. Section 4 presents general theorems about successful and unsuccessful pouring actions, a theorem stating that the scenario described in the problem description will result in the successful cracking of an egg, and a discussion of variants of the original problem. Detailed proofs are given. In Section 5, we discuss in detail the Initial Specification Problem and the Unobtainable State Problem.

2 Problem Statement and Methodology

2.1 Problem Statement

The egg-cracking problem (Davis, 1997) is one of a set of benchmark commonsense reasoning problems collected several years ago by members of the commonsense reasoning community. (These can be found in Miller and Morgenstern, 1997.)

The problem reads:

Characterize the following:

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.

What happens if: The cook brings the egg to impact very quickly? Very slowly? The cook places 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?

2.2 The Approach Taken Here

Davis (1998) has argued that given the current state of the field of logicist knowledge representation, it is impossible to adequately formalize such problems. Initial attempts at formalizing the egg-cracking problem by Lifschitz (1998), Morgenstern (1998), and Shanahan (1998) demonstrated some of the difficulties faced. Our approach in this paper will be to simplify the problem and the domain so that formalization will be possible. However, we will stay clear of a simplification that is so severe

that the real-world problem is turned into a toy problem. [2]

Specifically, we make the following simplifications: We ignore the role and position of the hands during egg cracking. No distinction is made between egg whites and egg yolks. Egg insides are simply considered to be liquid. The viscosity of the egg is not specified (though the viscosity of a liquid is considered in the discussion of pouring and leaking). The shape of the crack in an eggshell, and its role in the egg-cracking process, are ignored. The method of breaking open a container discussed here is a sequence of forming an initial break, or crack, and then enlarging that opening, as is done with a cardboard box. A container does not separate into two (or more) fragments. The only rotation considered is a 180-degree rotation. Most properties of liquids, such as the fact that dry objects in contact with liquids become wet, are not formalized.

The simplified problem is still quite complex. We focus on formalizing issues of containment and pouring. There is also a sketchy formalization of materials and breaking objects.

The ultimate goal is the development of a theory that solves both the egg-cracking problem and its stated *elaborations*. The theory is said to solve the egg-cracking problem if it is a theorem of the theory that the procedure described above results in the egg being safely contained by the bowl.

Certain elaborations or *variants* of the egg-cracking problem are given in the second paragraph of the problem statement. In this case, the elaborations presented are ways in which the egg-cracking process can go awry. A theory Λ is said to handle a particular elaboration if it is *not* a theorem of Λ that the variant egg-cracking scenario will result in a successful egg cracking. (It is preferable, of course, if it is a theorem of Λ that the variant scenario will *not* result in a successful egg cracking. In sufficiently powerful nonmonotonic theories of action, if it is not a theorem that the variant scenario will be successful, it will in fact be a theorem that the variant scenario will not be successful.)

Considering elaborations on a problem can be useful for restricting a theory so that it is neither too strong (allowing all sorts of odd actions to result in successful egg crackings) nor too weak (e.g., unable to make the transition from large to jumbo eggs). Elaborations can be used to test the limits of an existing theory, or to guide the development of the theory.

A theory is said to be *elaboration tolerant* (McCarthy 1988) if it successfully handles elaborations of the primary problem. The term is often used in a binary sense, where theories fail to be elaboration tolerant if they cannot solve every variant of a problem. It is unrealistic, however, to expect formalizations of commonsense theories to be elaboration tolerant in the absolute sense. A variant that seems to be only slightly changed from the primary problem can in fact involve many other issues. Consider, for example, the variant of the egg-cracking scenario where the cook presses his hand down on the egg and crushes it, presumably winding up with a mixture of eggshell fragments and egg on his hand. To correctly formalize what happens, one would at least need to develop theories of fragmentation and of mixtures, theories that are rather far removed from the formalization of the simple egg-cracking scenario. Furthermore, it is impossible to consider, or even to think, of every possible variant of a problem, so that unless one has formalized all of commonsense reasoning, it is almost inevitable that there is some variant that one will not be able to handle.

It may be more useful to consider the degree of elaboration tolerance of a theory, so that we can speak of a theory as being more or less elaboration tolerant. [3] Given the difficulty of formalizing egg cracking even in the simple case, we concentrate on developing a theory that can characterize the successful egg-cracking process. We can, in fact, handle some of the elaborations. In particular, we can handle the variants which deal with different materials and too-small bowls.

[2] Note that Davis's egg-cracking problem is itself an idealization. Eggshells do not divide precisely in half; there are usually some fragments of eggshell hanging off the shell; and some of the eggwhite stays inside the shell.

[3] It is unclear how we can measure the elaboration tolerance of a particular theory, although this question has been examined by (Amir 2000).

2.3 Microworlds

How does the approach outlined above compare with the microworlds approach? Both Davis (1998) and Sandewall (2000) are proponents of using microworlds to formalize commonsense reasoning, and in particular, suggest using microworlds to move beyond the axiomatization of toy problems. Davis contrasts the microworlds methodology with Hayes's (1985a,b) "naive physics" approach, which advocates directly formalizing large chunks of commonsense knowledge without attempting to develop an underlying model. However, neither Davis nor Sandewall has given a formal or precise definition of microworlds. Davis gives as examples of the microworld methodology the roller coaster world of deKleer (1977), Forbus's (1980) formalization of motion, and de Kleer and Brown's (1985) formalization of component-based electronics; his own work on funnels and cutting objects (Davis 1988; 1993) are also exemplars. The idea is to focus on a specific domain of commonsense reasoning, to simplify the domain in a principled manner, ideally resulting in a model which can be formally characterized independent of the axioms. The model can then be analyzed using logical and mathematical tools. In particular, one can "check for the soundness and consistency of (one's) axioms by establishing that they are all true in the model." (Davis 1999)

Sandewall (1999) has constructed several microworlds for research in formalizing commonsense reasoning. One example is the zoo microworld, in which there are humans, animals, and cages. Cages may be open or closed; animals may be fed, and/or may eat other animals; animals and humans can go into and out of cages. Such microworlds have the flavor of toy problems but are considerably more complex. Sandewall's microworlds deliberately focus on integrating several domains, such as motion, space, and agents, but are clearly (and intentionally) artificial. In contrast, Davis's microworlds arise from natural problems, but they generally focus on single domains. This can affect the reuse of commonsense formalizations: Davis (1998) has pointed out the difficulties that can arise from integrating theories based on microworlds developed for single domains.

The approach described in this paper differs from both Davis's and Sandewall's microworld approaches. In contrast to Sandewall's artificial microworlds, we are examining a problem that arises in everyday reasoning. In contrast to Davis's preferred method of focusing on a single restricted domain, the problem — the egg-cracking "domain" — really spans several domains (shape, liquids, pouring, materials, collisions). Also in contrast to Davis, there is little attempt to define a microworld that can be characterized formally independently of the axioms. That approach, although it has obvious advantages, may simply not be viable when one tries to formalize the messy realities of the world. This point has been made even by proponents of the microworlds methodology, for inherently scruffy domains like emotions [4]. It may also hold when independent formal characterizations are possible but non-intuitive — that is, the formal model has little to do with commonsense reasoning.

If one's model cannot or should not be formally characterized independently of the axioms (that is, the best commonsense characterization is a clear English description of the model), then it is the role of the axioms to characterize the model, in the standard Tarskian manner. In this case, there is no point in establishing the truth of an axiom in the model; the axioms are true by definition. That is, there is no way to use the model to independently assess the correctness of one's axioms.

Although it is clear that developing an independent formal characterization of a domain is a very elegant approach, we believe that we ought not to let a desire for an elegant methodology curtail our choice of domain. The point of logicist commonsense reasoning is to formalize commonsense domains in logic, rather than to choose a particular methodology.

Nevertheless, we do follow the microworlds methodology in that we explicitly simplify the problem statement and the chunk of the world that we aim to formalize. We call the simplified chunk of the world a microworld, but do not subscribe to all previous assumptions about the use of the term.

[4] see, e.g, (Davis 1999)

Time	
e	events
f	fluents
t,n,d	time points (real numbers)
Shape	
r	regions of space
Objects and Containers	
en, loc	entities, locations
o, fo, do	objects, flat objects, destination objects (where objects fall)
c, cc, oc	containers, closed containers, open containers
p	packages
os	object sets
pt	object parts or fragments
egg	eggs
es	eggshells
ei	egg insides (an egg inside includes both white and yolk)
locations, objects, and object parts are entities; eggs, egg shells, egg insides, packages, containers, flat objects, and destination objects are objects; closed containers and open containers are containers	
General	
m	materials
x,y,q	real numbers (quantities, rates — number of units per time unit)

Table 1: Variables and Sorts

3 The Formalization

As discussed in the previous section, we develop portions of reusable core theories which can be used to characterize the process of egg cracking. We focus on theories of containment and pouring, with some attention to materials and breaking objects, and to the particulars of the egg.

In the definitions and axioms below, all variables are assumed to be universally quantified unless otherwise specified. We use a sorted logic. All sorts, predicates, and functions, including fluents, will be summarized in tables in the paper; most will be explicitly introduced in the text as well. The sorts used in the paper can be found in Table 1.

3.1 Temporal Theory: Shanahan's Circumscriptive Event Calculus

It is assumed that the reader has a basic familiarity with a temporal language like the event calculus or the situation calculus. The following account is intended as a summary or brief introduction. It is largely borrowed from Shanahan's (1998) summary.

We use a slight modification of the circumscriptive event calculus developed in (Shanahan 1997). This theory is based on the *event calculus* (Kowalski and Sergot 1986). The basic entities are actions (or events) and instants of time. As in the situation calculus (McCarthy and Hayes 1969), *fluents* — functions whose domain is the set of time instants — are used to represent time-dependent properties, such as the color of a block or the president of the United States. A propositional fluent is a fluent which maps onto the set of values {true, false}. If *f* is a fluent and *t* is a time point, and *f* maps *t* onto *true*, we say that *Holds(f, t)* is true. An example of a propositional fluent is *On(BlockA, BlockB)*. If Block A is on Block B at time *t1* but not at time *t4*, we say that *Holds(On(BlockA,*

BlockB), t1) but ¬ *Holds(On(BlockA, BlockB), t4)*. An example of a non-propositional fluent is *Color(BlockA)*. In the event calculus, the fact that *BlockA* is red at time *t3* can be expressed as *Holds(Color(BlockA)=Red, t3)*.

Syntactic Sugar: *In general, if fn(α) is a fluent, where fn is a fluent function and α represents a set of arguments to fn, and v is the value of fn(α), then we may use interchangeably the expressions Holds(fn(α)=v, t) and fn(α,t) = v.*
An example of the second usage, which is often much more compact, can be seen in Axiom 18.

A distinguished time point $t = 0$ is used to describe what is true at some initial time point. *InitiallyP(f)* is true iff *Holds(f,0)*; *InitiallyN(f)* is true iff ¬ *Holds(f, 0)*.

Actions can *initiate*, *terminate*, or *release* fluents. For example, the action *TurnOn(switch)* initiates the fluent *On(Bulb)*; the action *TurnOff(switch)* terminates that same fluent. Assuming these statements hold true for all time points *t*, they could be represented in the event calculus as *Initiates(TurnOn(switch), On(Bulb),t)* and *Terminates(TurnOff(switch), On(Bulb),t)*.

The *Releases* predicate is weaker than either *Initiates* or *Terminates*; it is used when after an action, the value of a fluent is no longer known. For example, if a single chamber of a revolver, the active chamber, is loaded with a bullet, and the gun is spun, as in Russian Roulette, we no longer know whether the active chamber is loaded. The spinning action does not *terminate* the gun's being loaded, but it *releases* the loaded fluent. This could be represented as *Releases(Spin(gun), Loaded(gun),t)* [5].

The *Initiates*, *Terminates*, and *Releases* predicates form the core of the causal theory. Event narration is accomplished using the *Happens* predicate: *Happens(e,t)* means that event *e* happens at time *t*. Continuous change is represented through *trajectories*. *Trajectory(f1,t,f2,d)* means that if fluent *f1* starts to hold at time *t*, then fluent *f2* holds at time *t+d*. The *Trajectory* predicate is used, for example, to describe the filling of a container with liquid, or the position of a moving car on the highway.

The central concern of temporal reasoning is determining when fluents change, and when they stay the same. We say that a fluent is *clipped* during an interval if an action happens during the interval that terminates or releases the fluent. A fluent is *declipped* during an interval if an action happens that initiates or releases the fluent. If a fluent is initiated during an interval, and not subsequently clipped, it holds at the end of the interval; if a fluent is terminated, and not subsequently declipped, it does not hold at the end of the interval.

The following axioms, which formalize this brief description of the event calculus, are taken directly from (Shanahan 1997):

Axiom ECF1:
$(InitiallyP(f) \wedge \neg Clipped(0,f,t2)) \vee (Happens(a,t1) \wedge Initiates(a,f,t1) \wedge t1 < t2 \wedge \neg Clipped(t1,f,t2)) \Rightarrow Holds(f,t2)$

Axiom ECF2:
$(InitiallyN(f) \wedge \neg Declipped(0,f,t2)) \vee (Happens(a,t1) \wedge Terminates(a,f,t1) \wedge t1 < t2 \wedge \neg Declipped(t1,f,t2)) \Rightarrow \neg Holds(f,t2)$

Axiom ECF3:
$Clipped(t1,f,t2) \Leftrightarrow \exists a,t\ (Happens(a,t) \wedge (Terminates(a,f,t) \vee Releases(a,f,t)) \wedge t1 < t \wedge t < t2)$

Axiom ECF4:
$Declipped(t1,f,t2) \Leftrightarrow \exists a,t\ (Happens(a,t) \wedge (Initiates(a,f,t) \vee Releases(a,f,t)) \wedge t1 < t \wedge t < t2)$

The *frame problem* (McCarthy and Hayes 1969) in AI is the problem of correctly predicting that most fluents remain unchanged as actions occur, without having to explicitly specify that an action does not cause a fluent to change. In the classic blocks-world planning example, if three blocks A, B, and C are on the table, and one wishes to make a tower with A on top of B and B on top of C,

[5] The *Releases* predicate will not be used in the formalization presented in this paper, but could easily be used in an extension. For example, there could be an axiom stating that transporting a whole egg in the basket of a bicycle releases the fluent *WholeEgg(egg)*.

one first places B on C. It is assumed that in the resulting state, A is still on the table; that is has not moved during the first action.

The tendency of fluents to persist is often referred to as the commonsense law of inertia. Shanahan's solution to the frame problem formalizes the two essential assumptions underlying the law of inertia:
(1) actions do not generally have unexpected consequences; that is, if an action has a consequence, it is implicit in the causal theory.
(2) Unexpected actions generally do not happen; that is, if an action happens, it is usually explicit in the narrative, or can be derived from the narrative and the causal theory.

(We note that although most fluents obey the commonsense law of inertia, some fluents are inherently *unstable* in some circumstances: they are generally guaranteed not to persist [6]. For example, a liquid object inside a porous container leaves the container.)

Shanahan formalizes these assumptions by circumscribing (minimizing) over *Happens*, *Initiates*, *Terminates*, and *Releases*. Intuitively, minimizing *Initiates*, *Terminates*, and *Releases* formalizes the first assumption; that and minimizing *Happens* formalizes the second assumption.

Formally, if Σ is a domain causal theory (a set of *Initiates*, *Releases* and *Terminates* formulas), Δ is a narrative description (a collection of *InitiallyP*, *InitiallyN*, and *Happens* formulas), then we are interested in the theory:

Circ(Δ; *Happens*) $\wedge$ Circ(Σ; *Initiates*, *Terminates*, *Releases*) $\wedge\Lambda$,

where Λ is the set of event calculus axioms ECF1–ECF4, together with relevant state constraints, unique names axioms, and domain closure axioms.

Predicate	**Intuitive Meaning**
Holds(f,t)	Fluent *f* holds at time *t*
InitiallyP(t)	Fluent *f* starts to hold at time 0
InitiallyN(t)	Fluent *f* does not hold starting at time 0
Happens(e,t)	Action *e* occurs at time *t*
Initiates(e,f,t)	Fluent *f* holds after action *e* at time *t*
Terminates(e,f,t)	Fluent *f* does not hold after action *e* at time *t*.
Releases(e,f,t)	Fluent *f* is not subject to the commonsense law of inertia after action *e* at time *t*; it is not generally known whether *f* holds after time *t*
Trajectory(f1,t,f2,d)	If *f1* starts to hold at *t*, then *f2* holds at time *t+d*
Clipped(t1,f,t2)	Some event terminates fluent *f* between times *t1* and *t2*
Declipped(t1,f,t2)	Some event initiates fluent *f* between times *t1* and *t2*

Table 2: Theory of Time

3.2 Liquid and Solid Objects; Materials

It is useful to characterize objects by their shape, their state of liquidity, and the materials they are made of. We discuss at length certain useful object shapes — in particular, objects that are shaped like open and closed containers — in Section 3.3.

Liquids

In this microworld, an object can be in a solid or liquid state.

Axiom 1 *Holds(Solid(o), t)* $\dot{\vee}$ *Holds(Liquid(o), t)*

We do not concern ourselves here with the fact that liquids divide often and easily. A liquid object, once identified, remains an object even if it is divided among multiple locations. To the extent possible, we use the notions of volume, quantity, and capacity to formalize the facts in our domain.

Note that what we call a liquid object is identical to what Hayes (1985a) calls a *piece of liquid*. (Hayes used the term "liquid object" to characterize some liquid entity over a period of time.) Our

[6] Note that this usage differs from Shanahan, who uses "unstable" in a somewhat different sense.

representation of liquids is drastically simpler than Hayes's. We discuss the possibility of integrating these theories in the conclusion.

Materials

We consider only objects that are *uniform*; that is, each object consists, at a particular point of time, of a single material. The material of an object can change over time. Thus, *MaterialOf* is a time-dependent function (fluent). Materials may have various properties. Materials can be hard or soft; fragile, unbreakable, or hard to break; porous or non-porous. In some cases, properties of a material are mutually exclusive; we list those below. In other cases, properties may co-exist; thus, for example, a material can be simultaneously hard and porous. Note that hard and hard-to-break are related loosely, if at all. In particular, an object may be hard and fragile, or soft and hard-to-break or unbreakable.

Axiom 2 (Mutual Exclusivity of Material Properties)
$Hard(m) \dot{\vee} Soft(m)$
$Fragile(m) \dot{\vee} Hard\text{-}to\text{-}Break(m) \dot{\vee} Unbreakable(m)$

One material may be harder to break than another. The axioms below state that unbreakable objects are harder to break than hard-to-break objects and that hard-to-break objects are harder to break than fragile objects, and that harder to break is a transitive relation.

Axiom 3 $Unbreakable(m1) \wedge Hard\text{-}to\text{-}Break(m2) \Rightarrow HardertoBreak(m1,m2)$
$Hard\text{-}to\text{-}Break(m1) \wedge Fragile(m2) \Rightarrow HardertoBreak(m1,m2)$

Axiom 4 $HardertoBreak(m1,m2) \wedge HardertoBreak(m2,m3) \Rightarrow HardertoBreak(m1,m3)$

In particular, we posit that glass is hard and relatively hard to break (thus, neither fragile nor unbreakable):

Axiom 5 $Glass(m) \Rightarrow Hard\text{-}to\text{-}Break(m)$

Axiom 6 $Glass(m) \Rightarrow Hard(m)$

Formula	Intuitive Meaning (and examples)
MaterialOf(o)	Fluent. Returns the material of an object *o* (at a particular moment in time).
Liquid(o)	Propositional fluent. Object *o* is a liquid (water, raw egg).
Solid(o)	Propositional fluent. Object *o* is a solid (chair, bowl).
Hard(m)	Predicate. *m* is hard (wood, glass).
Soft(m)	Predicate. *m* is soft (e.g. dough, clay).
Fragile(m)	Predicate. *m* is easily breakable / brittle (eggshells, dried leaves)
Hard-to-break(m)	Predicate. Possible, but hard, to break *m* (china, glass)
Unbreakable(m)	Predicate. Impossible to break *m* (steel)
Harder(m1,m2)	Predicate. *m1* is harder than *m2*
Porous(m)	Predicate. *m* is porous (paper, towels, mesh)
Glass(m)	Predicate. The material *m* is glass.

Table 3: Solids, Liquids, and Materials

3.3 Containers

We focus on two types of containers: closed containers and open containers. Of open containers, we consider only those that have exactly one opening: for example, bowls or broken eggshells. We do not consider open containers that have more than one opening, such as salt shakers or funnels.

3.3.1 Closed Containers

We begin by characterizing the shape of a closed container. This definition is adapted from Davis (1988), who labels this concept a box.

Definition 1 *ClosedContainerShape(r)* is true iff r is an ordinary point set and the complement of r has exactly two connected components.

Note that one component will be finite (the component "inside" the container shape); the other will be infinite (the component "outside" the container shape). In other words, the *ClosedContainerShape* defines a hollowed out container-like shape.

Definition 2 *insideof(r)* is the closure of the finite component of the complement of r.

Definition 3 *outsideof(r)* is the closure of the infinite component of the complement of r.

The function *shape* maps an object and a point in time to the region of space which the object occupies at that point of time. A closed container is any object which has the closed container shape. Since this concept is time dependent, it is a fluent.

Definition 4 *Holds(ClosedContainer(c),t)* $\Leftrightarrow$*(Holds(shape(c)=r,t)* $\wedge$ *ClosedContainerShape(r))*

3.3.2 Open Containers

Open containers are represented as closed containers that are missing a relatively small piece. [7] Thus, we have:

Definition 5 *OpenContainerShape(r1)* $\Leftrightarrow$
$\neg$ *ClosedContainerShape(r1)* $\wedge\exists$*r3, r2 (r3 = r1* $\cup$ *r2* $\wedge$*ClosedContainerShape(r3)* $\wedge$*smaller(r2,r1))*

As above, we extend this definition to objects:

Definition 6 *Holds(OpenContainer(c),t)* $\Leftrightarrow$*(Holds(shape(c) = r, t)* $\wedge$*OpenContainerShape(r))*

It will be useful to talk about *rims*. The rim of an open container is a part of the container.

Axiom 7 *Holds(ObjPart(Rim(oc),oc),t)*

3.3.3 Enclosing and Containing:

An object is *enclosed* by a closed container if it lies entirely within the space inside the closed container.

Definition 7 *Holds(Encloses(cc,o),t)* $\Leftrightarrow$*(Holds(shape(o) = r1, t)* $\wedge$*Holds(shape(o) = r2)* $\wedge$*subregion(r1,r2))*

Containment is more complex. Obviously, closed containers always contain their contents, and being contained within a closed container is the same as enclosure:

Axiom 8 *Holds(Contains(cc,o),t)* $\Leftrightarrow$*Holds(Encloses(cc,o),t)*

[7] An alternate characterization, with a heavy emphasis on topology, can be found in (Davis 1988).

Intuitively, an open container contains an object if the object fits into the container's cavity, or the inside of an OpenContainerShape. To formalize this, we introduce the *VirtualLid* of the OpenContainershape.

Definition 8 If r is an open container shape, *VirtualLid(r)* is the smallest *r'* such that
(1) $r \cup r'$ is a closed container shape
(2) for any two points *x,y* in *r'*, the straight line drawn between *x* and *y* contains points only in *r'* or in *insideof*$(r \cup r')$.
(Condition (2) excludes from consideration virtual lids that curve downward (concave lids), crowding the space inside the open container.)

The intuition is that the virtual lid of an open container shape is a region in the shape of a lid that could "close" the open container shape.

The containing space of an open container shape can now be defined as the inside of the virtual closed container that is formed by the open containershape and the virtual lid.

Definition 9 *OpenContainerShape(r)* $\Rightarrow$ *VirtualClosing(r)* $= r \cup$ *VirtualLid(r)*

Definition 10 *OpenContainerShape(r)* $\Rightarrow$ *ContainingSpace(r)* $=$ *insideof(VirtualClosing(r))*

An object is *contained* by an open container if the object lies entirely within the space defined by the virtual closing of the open container shape.

Axiom 9 *Holds(Contains(oc,o),t)* $\Leftrightarrow$
(Holds(shape(o) $=$ *r1, t)* $\wedge$ *Holds(shape(oc)=r2), t)* $\Rightarrow$ *subregion(r1, ContainingSpace(r2)))*

Note that this definition makes no reference to the orientation of an open container. Even if the cavity of an open container faces downward, the container can still contain the object, if only instantaneously. Of course, we wish to predict that in such a case the object will fall from the container, as we discuss below.

It is often useful to refer to the entity consisting of a closed container and the object or objects that it encloses. We will use the concept *package* for this purpose.

Definition 11 *Holds(Package(p, cc, os), t)* $\Leftrightarrow$ *(p* $=$ *cc* $\cup$ *os)* $\wedge \forall o \in$ *os Holds(Encloses(cc,o), t)*

We introduce some syntactic sugar when the package contains only one object. In this case, we allow the third argument to be the object, rather than the singleton set whose member is the object.

3.4 When Objects Leave Containers

Shanahan's solution to the frame problem (minimizing over *Initiates*, *Terminates*, *Releases*, and *Happens*) guarantees that if an object is enclosed or contained by a container, it remains enclosed or contained unless an action that causes a change in enclosement or containment happens.

We focus on two types of ways in which objects leave containers: First, when objects fall out or are poured out from containers, and second, when objects leak out of containers. The first case can occur only with open containers; the object may be solid or liquid. The second case can occur with either open or closed containers, but only with liquid objects. Because the occurrence of the falling and pouring actions depends on the orientation of open containers, we begin this subsection with a discussion of orientation and rotation.

We are particularly interested in characterizing "successful" falls; those in which the object rests on or is contained in a destination object at the end of its fall. In the case of liquid objects, successful falls occur when the destination object is an open container (as opposed to a flat object) that has enough space to accommodate the object. To formalize this, we axiomatize volume and capacity in section 3.4.2. In successful falls, it is also the case that the object is directly above its destination. We axiomatize the relative location of objects in section 3.4.3.

3.4.1 The Orientation of Open Containers

In this microworld, an open container shape may have its opening upward or downward (exclusive or). We introduce the *Upwardfacing* and *Downwardfacing* predicates, which range over regions of space, as primitive.

Definition 12 $Holds(Upward(oc),t) \Leftrightarrow (Holds(shape(oc)=r,t) \wedge Holds(Upwardfacing(r)))$

Definition 13 $Holds(Downward(oc),t) \Leftrightarrow (Holds(shape(oc)=r,t) \wedge Holds(Downwardfacing(r)))$

Axiom 10 $Holds(Downward(oc),\ t)\ \dot{\vee}\ Holds(Upward(oc),\ t)$

Orientation changes through the rotation action. (Given a richer theory of the orientation of objects, these axioms could be generalized and simplified.)

Axiom 11 $Holds(Upward(oc),t) \Rightarrow Initiates(Rotate(oc),Downward(oc),t)$

Axiom 12 $Holds(Upward(oc),t) \Rightarrow Terminates(Rotate(oc),Upward(oc),t)$

Axiom 13 $Holds(Downward(oc),t) \Rightarrow Initiates(Rotate(oc),Upward(oc),t)$

Axiom 14 $Holds(Downward(oc),t) \Rightarrow Terminates(Rotate(oc),Downward(oc),t)$

3.4.2 Volume and Capacity

The function *volume* designates the standard volume measure of a 3-D region of space. We introduce the fluent *ovolume* (*object* volume) which designates the volume of the space occupied by an object at a particular instant of time. This function allows us to speak about the volume of an object in accordance with standard usage.

The total capacity of a closed container is equivalent to the volume of the inside space:

Axiom 15 $Holds(shape(cc)=r,\ t) \Rightarrow Holds(Capacity(cc) = volume(inside(r),\ t))$

The total capacity of an open container is equal to the volume of the containing space of the virtual closing of the open container shape:

Axiom 16 $Holds(shape(oc)=r,\ t) \Rightarrow Holds(Capacity(oc) = volume(inside(ContainingSpace(r))),\ t)$

At any moment in time, an open container may already contain one or more objects. It is generally important to consider a container's available capacity. The following axioms formalize the fact that a container's available capacity is reduced by the volume of the objects already in it.

Definition 14 *Holds(TotalObjectVolume(c) = x, t)* $\Leftrightarrow x=$

$$\sum_{o|Holds(Contains(c,o),t)} x_i \mid Holds(ovolume(o,t) = x_i, t)$$

Axiom 17 *Holds(TotalObjectVolume(c) = x1,t)* $\wedge$ *Holds(Capacity(c) = x2, t)* $\Rightarrow x1 \leq x2$

Axiom 18 *AvailCapacity(c,t) = Capacity(c,t) - TotalObjectVolume(c,t)* [8]

Definition 15 *Holds(Full(c), t)* $\Leftrightarrow$ *Holds(AvailCapacity(c)=0, t)*

Definition 16 *Holds(Empty(c),t)* $\Leftrightarrow \neg \exists o$ *Holds(Contains(c,o),t)*

It is an immediate (monotonic) conclusion that the available capacity of an empty container is equal to its total capacity:

Lemma 1 *Holds(Empty(c),t)* $\Rightarrow$ *Holds(AvailCapacity(c)=Capacity(c), t)*

3.4.3 The Relative Position of Objects

When an object falls from an open container, it generally falls to whatever is nearest below it. We will need to speak about one object being above, or directly above, another. We introduce the fluent *Above*, with its usual meaning, as primitive. *Above* ranges over objects and is transitive. We introduce the concept of a *location*, which is a kind of entity in accordance with its common usage. One location may be above another; a location can be above an object; and an object can be above a location. An object may be *at* a location; and the *Move* action moves an object to a location.

Axiom 19 *Holds(Above(en1,en2), t)* $\wedge$ *Holds(Above(en2,en3), t)* $\Rightarrow$ *Holds(Above(en1,en3), t)*

Axiom 20 *(Holds(At(o1,loc), t)* $\wedge$ *Holds(Above(loc,o2), t))* $\Rightarrow$ *Holds(Above(o1,o2), t)*

Axiom 21 *(Holds(At(o2,loc), t)* $\wedge$ *Holds(Above(o1,loc), t))* $\Rightarrow$ *Holds(Above(o1,o2), t)*

Axiom 22 *Initiates(Move(o,loc), At(o,loc), t)*

Axiom 23 *Holds(At(o, loc1), t)* $\wedge loc1 \neq loc2 \Rightarrow$ *Terminates(Move(o,loc2), At(o,loc1), t)*

An object is directly above another object if there is no third object in between the two.

Definition 17 *Holds(DirectlyAbove(o1,o2), t)* $\Leftrightarrow$
Holds(Above(o1,o2), t) $\wedge \neg \exists$ *o3 (Holds(Above(o1,o3), t)* $\wedge$ *Holds(Above(o3,o2), t))*

We make the following simplification to our model: It is never the case that one container is directly above two distinct containers.

Axiom 24 *Holds(DirectlyAbove(c1,c2), t)* $\wedge$ *Holds(DirectlyAbove(c1,c3), t)* $\Rightarrow c2 = c3$

[8] See note on syntactic sugar in Section 3.1.

3.4.4 Falling and Pouring

An object can fall from an open container when its opening faces downward. Specifically, if an object is either in a liquid state or is small enough to fit through the opening, then whenever the container's opening faces downward and the container contains the object, a falling action is triggered.

The general *Fall* action takes two arguments: the object, and the open container from which the object falls. The *FallTo* action is a special kind of *Fall* action, namely, where the object falls to a destination object. The *Pour* action is also a special kind of *Fall* action: the falling object is liquid. The *PourTo* action is a *FallTo* action in which the falling object is liquid.

It is often useful to associate with an action a fluent that describes the action process, particularly when one must reason about continuous change. We associate with each of the actions above an action fluent. The action fluent is initiated by its corresponding action.

Thus, we have the following axioms:

Axiom 25 *Holds(Contains(oc,o),t)* $\wedge$ *Holds(Downward(oc,t))* $\wedge$
(Holds(Liquid(o),t) $\vee$ *Holds(smaller(shape(o),VirtualLid(shape(oc))),t))* $\Rightarrow$ *Happens(Fall(o,oc),t)*

Axiom 26 *Initiates(Fall(o,oc), Falling(o,oc), t)*

Definition 18 *Holds(DirectlyAbove(o,do),t)* $\wedge$ *Happens(Fall(o,oc), t)* $\Leftrightarrow$ *Happens(FallTo(o,oc,do),t)*

Axiom 27 *Initiates(FallTo(o,oc,do), FallingTo(o,oc,do), t)*

Definition 19 *Holds(Liquid(o),t)* $\wedge$ *Holds(Upward(oc),t)* $\wedge$ *Happens(Fall(o,oc),t)* $\Leftrightarrow$ *Happens(Pour(o,oc),t)*

Axiom 28 *Initiates(Pour(o,oc), Pouring(o,oc), t)*

Definition 20 *Happens(Pour(o,oc1),t)* $\wedge$ *Happens(FallTo(o,oc1,oc2),t)* $\Leftrightarrow$ *Happens(PourTo(o,oc1,oc2),t)*

Axiom 29 *Initiates(PourTo(o,oc1,oc2), PouringTo(o,oc1,oc2), t)*

As soon as an object falls from a container, it is no longer contained by that container:

Axiom 30 *Terminates(Fall(o,oc), Contains(oc,o), t)*

We can use the *trajectory* predicate to describe how falling changes the location of objects, and how pouring changes the quantity of a liquid object in a container.

If an object is falling from a container to a destination object, its distance from the container becomes larger as its distance from the destination object becomes smaller.

Axiom 31 *(y=rateoffall(n)* $\wedge$ *Holds(Distance(o,do,x),t)* $\wedge$ *f(n)* $\leq$ *x)* $\Rightarrow$
Trajectory(FallingTo(o,oc,do),t,Distance(o,oc,y),n) $\wedge$
Trajectory(FallingTo(o,oc,do),t,Distance(o,oc,x-y),n)

When one pours a liquid from one open container to another, the quantity of liquid in the source container decreases as the quantity of liquid in the destination container increases. The total object volume in the destination container likewise increases.

Axiom 32 *(x= rateofqfall(o,oc1)*$\cdot$*d* $\wedge$ *x* $\leq$ *ovolume(o,t))* $\Rightarrow$
Trajectory(PouringTo(o,oc1,oc2), t, Quantity(o,oc2,x), d)

Axiom 33 *(x= rateofqfall(o,oc1)·d ∧Holds(ovolume(o) = y, t) ∧x ≤ y) ⇒*
Trajectory(PouringTo(o,oc1,oc2), t, Quantity(o,oc1,y - x), d)

Axiom 34 *(Holds(TotalObjectVolume(oc2) = x1, t) ∧Holds(Capacity(oc2) = x2, t) ∧*
x3 = x1 + rateofqfall(o,oc1)·d ∧x3 ≤ x2) ⇒
Trajectory(PouringTo(o,oc1,oc2),t,TotalObjectVolume(oc2) = x3,d)

This axiom assumes that the capacity of the destination container is sufficient to hold the liquid object. Indeed, if the entire quantity of a liquid object is held by a container, the container *contains* that object:

Axiom 35 *(Holds(Liquid(o),t) ∧Holds(ovolume(o) = x, t)) ⇒*
(Holds(Contains(oc,o), t) ⇔Holds(Quantity(o,oc,x), t))

If the capacity of the destination container is not sufficient to hold the liquid object, an overflow will ensue:

Axiom 36 *Holds(Full(oc2), t) ∧Holds(PouringTo(o,oc1,oc2), t) ⇒Happens(Overflow(oc2),t)*

Action fluents like *FallingTo* and *PouringTo* are terminated when the solid or liquid object has reached its destination. There is no direct way, in the event calculus, to say that a condition terminates a fluent. Rather, we must say that a condition triggers an action, and that the action terminates the fluent. We introduce the actions *StopFallTo* and *StopPourTo*. *StopFallTo* is triggered when the falling object hits the destination object, and terminates the *Falling* fluent. *StopPourTo* is triggered when the liquid object is contained by the destination open container [9]. Note that these actions are somewhat artificial, serving essentially as bridges between conditions and fluents in the event calculus framework.

Axiom 37 *Holds(Contains(oc2,o), t) ∧Holds(PouringTo(o,oc1,oc2), t) ⇒*
Happens(StopPourTo(o,oc1,oc2), t)

Axiom 38 *Terminates(StopPourTo(o,oc1,oc2), PouringTo(o,oc1,oc2), t)*

Axiom 39 *Holds(Distance(o,do,0), t) ∧Holds(FallingTo(o,oc,do), t) ⇒*
Happens(StopFallTo(o,oc,do),t)

Axiom 40 *Terminates(StopFallTo(o,oc,do),FallingTo(o,oc,do), t)*

The previous discussion has considered the case of pouring liquid into an open container. A liquid may also be poured onto a flat object; in this case, it begins spreading.

Axiom 41 *Holds(Liquid(o),t) ⇒Initiates(Fall(o,oc,fo),Spreading(o,fo),t).*

Both overflows and spreadings are considered the results of unsuccessful pours. We do not further axiomatize overflowing and spreading.

[9] Note that an overflow action will not necessarily stop a *PourTo* action. If one is pouring a quart of milk into a bowl that is too small, the pouring may continue even after the bowl has overflowed. It would not, however, be accurate to say that the *PourTo* stops when the source container is empty, since in general, there will be a time gap between the time that the source container is empty and the destination container is filled. An analysis that gives all sufficient conditions for the occurrence of the *StopPourTo* action is beyond the scope of this work.

Axiomatizing the falling of solid objects is much more difficult, because one must reason about the shape (and weight) of the falling object. If the destination object is an open container, then even if the volume of an object is smaller than the available capacity of the destination container, it does not follow that the object will successfully fall into the destination container. The relative dimensions of the object and the destination container must also be taken into account: a thin 8 X 11 notebook will not fit into a standard large kitchen bowl, for example. If the destination object is a flat object, then one might say that a successful fall occurs if the surface area of the largest cross section of the falling object can be projected onto a subregion of the top of the flat object (this would ensure that the falling object winds up lying on the flat object at the end of the fall). However, this state of affairs might hold for only an instant: if the object is round, it might roll (e.g., a marble); if oddly shaped and/or weighted, it might tip over, past the edges of the destination object (e.g., a floppy stuffed giraffe). A productive line of inquiry might be to characterize classes of falling objects which have "stable" falls; they do not move from the destination flat object as soon as they have landed. Such an inquiry is beyond the scope of this paper.

3.4.5 Leaking Containers

Commonsense experience tells us that liquids leak out of containers if the containers have holes (e.g., a punctured milk container) or are made of porous material (e.g., a bowl made out of paper towels). An otherwise closed container that has one hole in it can be represented as an open container, since there is only one opening. But we have at this point no way of representing containers with multiple openings (e.g., a milk container with several punctures, a plastic bowl with a hole at the bottom). Representing this sort of leaking would first require extending the theory of containers so that we can handle containers with multiple openings. This would seem to be relatively easy for objects whose multiple holes are relatively close together, such as salt shakers. Objects whose multiple openings are spaced further apart, such as a bowl with a hole in the side or bottom, present a more difficult challenge.

We restrict the scope of inquiry in this paper to leaking porous containers (either open or closed).

The salient fact about leaking containers is that their containment of liquids is unstable: the liquid object inside the container begins leaving the container immediately:

Axiom 42 *(Holds(Contains(c,o),t) ∧Holds(Liquid(o),t) ∧Holds(Porous(material(c)),t))*
⇒Happens(Leak(o,c),t)

The leak action initiates the leaking fluent, and terminates the contains fluent.

Axiom 43 *Initiates(Leak(o,c), Leaking(o,c), t)*

Axiom 44 *Terminates(Leak(o,c), Contains(c,o), t)*

We assume that leaking occurs continuously until all the liquid is gone from the container. The amount of liquid that leaks out is a function of the viscosity of the liquid, the porosity of the container, and time. The specification of this function is beyond the scope of this work [10].

Axiom 45 *(Holds(Quantity(o,c,x1),t) ∧*
x2=x1 -rateofleak(porosity(c), viscosity(o), d) ∧x2 > 0)
⇒Trajectory(Leaking(o,c), t1, Quantity(o,c,x2), d)

[10] Note that more liquid leaks out from a porous container during the beginning of the leak action. As there is less liquid in the container, there is less contact between the remaining liquid and the material of the container from which the liquid leaks. Similarly for containers with multiple openings: when the liquid goes beneath the level of some of these openings, the rate of leaking declines. In any extension of this theory to containers with multiple openings, the location of the openings would also be an argument to the function.

The *Leak* action ends when the liquid object has entirely left the leaking container.

Axiom 46 *Holds(Quantity(o,c,0), t)* $\wedge$ *Holds(Leaking(o,c), t)* $\Rightarrow$ *Happens(StopLeak(o,c), t)*

Axiom 47 *Terminates(StopLeak(o,oc), Leaking(o,oc), t)*

Analogously with the analysis above of source destinations for falling objects, it is possible to reason about the containers into which liquid objects leak. For example, a porous container may be directly above a non-porous container (consider a sieve suspended over a glass jar); in this case, if the capacity of the non-porous container is sufficient, the leaking liquid will eventually be contained in the non-porous container. The decision to introduce *FallTo* and *PourTo* actions, but no *LeakTo* action, reflects the fact that falls usually happen over a short period of time. Thus, one frequently talks about a single destination object. Leaks, on the other hand, usually happen over an extended period of time, and there may not be a single destination object (consider a leaking milk container that is carried home from the supermarket). Adding a *LeakTo* action, however, if desired, would be entirely straightforward.

Formula	**Intuitive Meaning**
volume(r)	function. Returns the volume (real number) of a region of space.
subregion(r1,r2)	predicate. r1 is a subregion of r2
insideof(r)	function on closed container shapes. Returns closure of finite component of r
outsideof(r)	function, as above. Returns closure of infinite component of complement of r
smaller(r1,r2)	predicate on regions of space. Obvious meaning.
ClosedContainerShape(r)	predicate. *r* is in the shape of a closed container.
OpenContainerShape(r)	predicate. *r* is in the shape of an open container.
VirtualLid(r)	function on open container shapes. Returns a shape that could act as a lid to *r*.
VirtualClosing(r)	function on open container shapes. Returns a closed container shape consisting of the open container and its virtual lid.
ContainingSpace(r)	function on open container shapes. Returns space inside the open container and its virtual lid.
UpwardFacing(r)	predicate. True if *r* faces up. (*r* must be an open container shape).
DownwardFacing(r)	predicate. True if *r* faces down. (*r* must be an open container shape.)

Table 4: Container Shapes

Formula	Intuitive Meaning
shape(o)	Fluent. Maps object *o* to a region of space
ClosedContainer(c)	Propositional fluent. *c* is a closed container.
OpenContainer(c)	Propositional fluent. *c* is an open container.
Rim(oc)	Fluent. Returns the rim (object part) of an open container.
Encloses(cc, o)	Propositional fluent. Closed container entirely encloses object.
Contains(c, o)	Propositional fluent. Container (open or closed) contains object.
Package(p, cc, os)	Prop. fluent. Package *p* consists of a closed container and all objects inside. Third argument can be set of objects or single object.
ObjectPart(pt, o)	Propositional fluent. Object part *pt* is a part of object *o*.
Downward(oc)	Propositional fluent. Open container faces downward.
Upward(oc)	Propositional fluent. Open container faces upward.
ovolume(o)	Fluent. Returns volume (real number) of an object.
Capacity(c)	Fluent. Returns the capacity of a container: the volume an object can hold.
TotalObjectVolume(c)	Fluent. Returns the total volume of all objects already in container *c*.
AvailCapacity(c)	Fluent. Returns the available capacity of a container.
Empty(c)	Propositional fluent. Container has no objects inside.
Full(c)	Propositional fluent. Container is full; available capacity = total capacity.
Distance(o1,o2,x)	Propositional fluent. Distance between objects *o1* and *o2* is x.
Porosity(c)	Fluent. Returns a real number representing the porosity of a container.
Viscosity (o)	Fluent. Returns a real number representing the viscosity of a liquid object.
Quantity(o,c,x)	Propositional fluent. The volume of the portion of liquid object *o* in *c* is *x*.
Rotate(o)	Action. Object rotates 180 degrees.
Fall(o, oc)	Action. Object *o* falls from open container *oc*.
FallTo(o, oc, do)	Action. Object *o* falls from open container *oc* onto destination object *do*.
Pour(o, oc)	Action. Liquid object *o* pours (falls) from open container *oc*.
PourTo(o, oc1, oc2)	Action. Liquid object *o* pours from open container *oc1* onto open container *oc2*.
Overflow(oc)	Action. Open container *oc* overflows.
Leak(o,c)	Action. Liquid object *o* leaks from container *c*.
Falling(o, oc)	Action fluent, initiated by *Fall* action.
FallingTo(o, oc)	Action fluent, initiated by *FallTo* action.
Pouring(o, oc)	Action fluent, initiated by *Pour* action.
PouringTo (o, oc1, oc2)	Action fluent, initiated by *PourTo action*.
Leaking(o, c)	Action fluent, initiated by *Leak* action.
Spreading(o, do)	Action fluent, initiated by a fall (or, presumably, a leak) onto a flat object.
StopFallTo(o, oc, do)	Action which terminates the *FallingTo* fluent.
StopPourTo(o, oc1, oc2)	Action which terminates the *PouringTo* fluent.
StopLeak(o, oc1, oc2)	Action which terminates the *Leaking* fluent.
rateoffall(o)	Fluent. Returns the rate at which an object falls over time(real number).
rateofqfall(o, oc)	Fluent. Returns rate at which quantity of liquid leaves container during pour.
rateofleak(o, c, n)	Fluent. Returns the rate at which a liquid object leaks from a porous container.

Table 5: Containers: Falling, Pouring, and Leaking

3.5 Breaking Objects

We are interested in reasoning about breaking closed containers. Breaks occur when the object is breakable and is hit with sufficient force. If one wishes to break a closed container in order to get at its contents, it is usually not advisable to indiscriminately attack the container with a great amount of force. Rather, one wishes to control the break. One can do this by introducing a small opening into the container, and then enlarging that opening. Often one introduces the small opening by scoring a line along the container: e.g., scoring a line on a cardboard box with a box cutter. For objects such as nuts or eggs one introduces the small opening by *cracking*: hitting the object against a hard

object with just enough force that a crack is formed. In this idealization, we assume that the crack is not really an opening but a "pre-opening": that is, the cracked object remains a closed container until the enlargement, or opening-up action, occurs. This assumption simplifies the axiomatization, because one does not have to reason about what might happen to the contents of the container if it is moved while the crack is face down. However, it would be relatively simple to extend this theory so that cracks were represented as small openings.

We introduce the fluent *crackforce*, which takes as argument the object to be cracked, the part of that object which is to be cracked, and the part of the second object which collides with the first object part, and returns the proper (range of) force at which such a collision will result in a crack (as opposed to a more substantial break, or a fragmentation). Note that the *parts* of the objects at which the collision occurs are arguments to *crackforce*: One needs less force, for example, to crack an egg against the rim of a bowl than against the side of the bowl because the sharpness of the edge plays a factor. The protruding part of the egg is likewise more vulnerable than other parts of the egg. Sharpness and shape are not explicit arguments to *crackforce* but are implicit in the parts of the objects. In general, one must make sure that both the object to be cracked and the hitting object are hard. The hitting object must also be harder to break than the object to be cracked; otherwise, one risks cracking the wrong object.

Axiom 48 *Holds(HardertoBreak(MaterialOf(o2),MaterialOf(o1)), t)* $\wedge$
Holds(Hard(MaterialOf(o2)),t) $\wedge$*Holds(Hard(MaterialOf(o1)), t)* $\wedge$
Holds(ObjPart(pt1,o1), t) $\wedge$*Holds(ObjPart(pt2,o2), t)* $\Rightarrow$
Initiates(Hit(pt1, pt2, crackforce(o1,pt1, pt2)), CrackedContainer(o1),t)

In this axiomatization cracked containers are still closed containers. (We do not need to add an axiom stating this fact: the circumscriptive policy ensures that closed containers remain closed unless there are explicit axioms that entail otherwise.) However, unlike other (standard) closed containers, they can, if they are made of an easily breakable material, be broken. We are especially interested in the *OpenUp* action, which takes a cracked container and turns it into an open container. Breaking and fragmenting actions that result in the closed container being broken into a number of different parts are not discussed here. Thus, the discussion of breaking open a cracked egg will assume an idealized model of egg cracking, e.g., where the agent carefully turns back the eggshell parts along the crack, resulting in an eggshell that still holds together (consider the moment just before the eggshell separates into two pieces).

Axiom 49 *Holds(CrackedContainer(c),t)* $\wedge$*Holds(Fragile(MaterialOf(c),t)*
$\Rightarrow$*Initiates(OpenUp(c), OpenContainer(c),t)*

Axiom 50 *Terminates(OpenContainer(c), ClosedContainer(c), t)*

It is assumed for simplicity that cracks are relatively small, and do not extend too far along the container. Thus, one can speak of a crack being face up or face down.

Axiom 51 *Holds(UpCrack(c), t)* $\dot{\vee}$*Holds(DownCrack(c), t)*

If the crack is face down at the beginning of the opening action, the open container resulting from the action is face down; if the crack is face up, the open container will be face up:

Axiom 52 *Hold(CrackedContainer(c),t)* $\wedge$*Holds(UpCrack(c),t)* $\Rightarrow$
Initiates(OpenUp(c), Upward(c), t)

Axiom 53 *Holds(CrackedContainer(c),t)* $\wedge$*Holds(DownCrack(c), t)* $\Rightarrow$
Initiates(OpenUp(c), Downward(c), t)

Rotating a cracked object will change the orientation of its crack:

Axiom 54 *Holds(DownCrack(c), t) ⇒Initiates(Rotate(c), UpCrack(c), t)*

Axiom 55 *Holds(DownCrack(c), t) ⇒Terminates(Rotate(c), DownCrack(c), t)*

Axiom 56 *Holds(UpCrack(c), t) ⇒Initiates(Rotate(c), DownCrack(c), t)*

Axiom 57 *Holds(UpCrack(c), t) ⇒Terminates(Rotate(c), UpCrack(c), t)*

Formula	Intuitive Meaning
CrackedContainer(c)	Propositional fluent. *c* is a cracked closed container.
UpCrack(c)	Propositional fluent. *c*'s crack faces up.
DownCrack(c)	Propositional fluent. *c*'s crack faces down.
ObjPt(pt, o)	Propositional fluent *pt* is some part or fragment of object *o*.
crackforce(o1,pt1,pt2)	Function. Returns the proper force (represented as real number) needed to crack *o1* as part *pt1* of *o1* collides with *pt2*, a part of some other object .
Hit(pt1, pt2, x)	Action. Hit object part *pt1* against object part *pt2* with force *x*.
OpenUp(c)	Action. Open up cracked closed container *c*.

Table 6: Hitting, Cracking, and Opening Objects

3.6 The Egg

From the point of view of the logicist who typically axiomatizes small domains with clearly defined objects, the egg raises several interesting issues. First, eggs do not always exist. For each egg (excluding possibly the *ur*-egg), there existed some period in time when it did not yet exist. Second, an egg goes through various stages where its properties may radically change. Third, while we often think of an egg as a single object, it really consists of at least two pieces, both of which we reason about: the eggshell and the inside of the egg. (The inside of the egg itself can be thought of as consisting of both an egg yolk and an egg white, but this distinction is beyond the scope of this paper).

Standard logic does not allow representation of objects that do not always exist. However, we can achieve the representational power we need by positing that there are different stages of the egg, including the stage where the egg has not yet been laid. We introduce the predicate NotYetLaid which refers not only to the time period in which the egg is inside the hen, but also to the time period in which the egg has not yet been formed by the hen. Indeed, it refers to the period before the laying hen exists herself.

The egg itself can go through many stages, including (in alternate courses of events), hatching, boiling, breaking, and scrambling. For the purposes of this paper, we will focus only on the stages in which the egg is not yet laid or is whole, cracked, or broken.

Axiom 58 *Holds(NotYetLaid(egg), t) ∨̇Holds(WholeEgg(egg), t) ∨̇*
Holds(CrackedEgg(egg), t) ∨̇Holds(BrokenEgg(egg), t)

These stages are ordered: an egg is always not yet laid before it is whole; whole before it is cracked; cracked before it is broken. We discuss in Section 5 how we can infer these facts using circumscription.

The laying action results in the change from the not yet laid stage to the whole egg stage.

Axiom 59 *Holds(NotYetLaid(egg), t) ⇒Initiates(Lay(egg), WholeEgg(egg), t)*

Axiom 60 *Terminates(Lay(egg), NotYetLaid(egg), t)*

Whole eggs and cracked eggs are both packages consisting of the eggshell and the inside of the egg. We introduce the functions EggshellOf and EggInsideOf, which, when applied to an egg, return, respectively, the shell and the inside of that particular egg.

Axiom 61 *Holds(WholeEgg(egg), t) ∨Holds(CrackedEgg(egg), t)*
⇒Holds(Package(egg, EggshellOf(egg), EggInsideOf(egg)), t)

The following is an immediate result:

Lemma 2 *Holds(WholeEgg(egg), t) ∨Holds(CrackedEgg(egg), t) ⇒*
Holds(ClosedContainer(EggshellOf(egg)), t)

A cracked egg is defined as one whose shell is a cracked (but still closed) container; a broken egg is defined as one whose shell is an open container. Note that in this idealization, eggshells do not break apart into pieces.

Definition 21 *Holds(CrackedEgg(egg), t) ⇔Holds(CrackedContainer(EggshellOf(egg)), t)*

Definition 22 *Holds(BrokenEgg(egg), t) ⇔Holds(OpenContainer(EggshellOf(egg)), t)*

Eggshells are fragile, but hard.

Axiom 62 *Holds(Fragile(MaterialOf(EggshellOf(egg))),t)*

Axiom 63 *Holds(Hard(MaterialOf(EggshellOf(egg))),t)*

The following conclusion follows immediately from Axioms 3, 5, and 62:

Lemma 3 *eggshell = EggshellOf(egg) ∧Holds(Glass(MaterialOf(o)), t) ⇒*
HardertoBreak(MaterialOf(o, t), eggshell)

Egg insides are liquid. (In this idealization, only raw eggs are considered.)

Axiom 64 *Holds(Liquid(EggInsideOf(egg)), t)*

The volume of the inside of an egg is between 30 ml and 70 ml.

Axiom 65 *Holds(30 ≤ ovolume(EggInsideOf(egg)) ≤ 70, t)*

The central band of an eggshell (the "waist" of the egg) is an object part.

Axiom 66 *eggshell = EggshellOf(egg) ⇒Holds(ObjPart(centralband(eggshell), eggshell), t)*

This completes the axiomatization. We will refer to the set of axioms introduced in Section 3 as the egg-cracking theory.

Formula	Intuitive Meaning
Egg(o)	Propositional fluent. *o* is an egg.
NotYetLaid(egg)	Propositional fluent. *egg* has not yet been laid.
WholeEgg(egg)	Propositional fluent. *egg* is a whole egg.
CrackedEgg(egg)	Propositional fluent. *egg* is a cracked egg.
Broken Egg(egg)	Propositional fluent. *egg* is a broken egg.
EggshellOf(egg)	Fluent. Returns eggshell (container) of egg package.
EggInsideOf(egg)	Fluent. Returns inside of an egg of egg package.
centralband(es)	Fluent. Returns the central band of the eggshell (object part).
Lay(egg)	Action. Initiates WholeEgg; terminates NotYetLaid.

Table 7: Eggs

4 The Theorems

All theorems below are proved using the techniques outlined in Chapter 16 of (Shanahan, 1997) [11]. Proofs proceed first by circumscribing the *Happens*, *Initiates* and *Terminates* predicates, and adding them to the theory, and subsequently proving theorems in the new, stronger theory. In general, *Happens*, *Initiates*, and *Terminates* axioms, together with ECF1–ECF4, are used to show that fluents change; the circumscription of *Happens*, *Initiates*, and *Terminates*, together with ECF1–ECF4, are used to show that fluents stay the same.

4.1 General Theorems

It was argued in Section 1 that one measure of the usefulness of an axiomatization is the generality of its theorems. We state and prove two general theorems about containers and pouring. These should serve as models for the proofs of other general theorems.

First, a theorem about successful pouring: If a liquid object is poured from one open container to a second open container and the available capacity of the receiving open container is larger than the volume of the liquid object, the receiving object will contain the liquid object at the end of the pouring action.

Theorem 1: Successful Pouring

Statement of Theorem 1:
Assume the following:

(1a) *InitiallyP(ovolume(O) =q1)*
(1b) *InitiallyP(AvailCapacity(OC2) = q2)*
(1c) *q2 > q1*
(1d) *Happens(PourTo(O,OC1,OC2), 0)*
(1e) *n = q1/rateofqfall(O,OC1)*

Let Λ be the union of all logical formulas in Section 3, the above assumptions, and all necessary uniqueness of names assumptions and domain closure axioms. We divide Λ into

- Φ: the conjunction of the event calculus formulas ECF1–ECF4,
- Δ: the conjunction of *Happens*, *InitiallyP*, *InitiallyN*, and temporal ordering formulas,
- Σ: the conjunction of *Initiates*, *Terminates* (and *Releases*) formulas,
- Ω: the conjunction of uniqueness-of-names axioms and domain closure axioms,
- Ψ: the conjunction of domain constraints, and
- Π: the conjunction of Trajectory formulas.

Then,
Circ(Δ; *Happens*) $\wedge$Circ(Σ; *Initiates*, *Terminates*) $\wedge\Omega$ $\wedge\Psi$ $\wedge\Pi$ $\wedge\Phi$
$\models \exists n$ *Holds(Contains(OC2,O), n)*

Proof of Theorem 1:

[11] Shanahan proves some results on separation that facilitate the proof process: To determine whether a formula is entailed by the circumscription of *Happens*, *Initiates*, *Terminates*, and *Releases* within a theory, one can examine the theory resulting from the union of the circumscription of *Happens* in the narrative portion of the axioms, the circumscription of *Initiates*, *Terminates*, and *Releases* in the causal portion of the axioms, and the other parts of theory (domain constraints, unique names assumptions, etc.). Given certain conditions, which hold in this theory, circumscription reduces to predicate completion. Details can be found in (Shanahan 1997).

Overview of Proof of Theorem 1: We begin by computing the circumscription of *Happens* in the narrative part of the theory (Δ) and the circumscription of *Initiates* and *Terminates* in the causal part of the theory (Σ). We show that the action of *PourTo* initiates the *PouringTo* fluent, and that as long as the fluent is true, the destination container fills up with liquid. The *PourTo* fluent terminates only when all the liquid in the object *O* from *OC1* is in *OC2*. Since first, the volume of the liquid object *O* is less than the volume of the container *OC2*, second, *OC2* is initially empty, and third, nothing else is filling up OC2, we can show that there will be enough space in *O2* for *O*.

Initial Setup — Predicate Completion:

We begin by completing the *Happens*, *Initiates*, and *Terminates* predicates. (The *Releases* predicate is not used in this formalization.)

The completion of the *Happens* predicate (H*), the completion of the *Initiates* predicate (I*),and the completion of the *Terminates* predicate (T*) are given below.

H* (completion of *Happens*):

Happens(a,t) $\Leftrightarrow$
$\exists$*o,oc (a = Fall(o,oc)* $\wedge$*Holds(Contains(oc,o),t)* $\wedge$*Holds(Downward(oc,t))* $\wedge$
(Holds(Liquid(o),t) $\vee$*Holds(smaller(shape(o),VirtualLid(shape(oc)),t))))*
$\vee$
$\exists$*o,oc,do*
(a= FallTo(o,oc,do) $\wedge$*Holds(Contains(oc,o),t)* $\wedge$*Holds(Downward(oc,t))* $\wedge$
(Holds(Liquid(o),t) $\vee$*Holds(smaller(shape(o),VirtualLid(shape(oc)),t))* $\wedge$*Holds(Above(o,do), t)))*
$\vee$
$\exists$*o,oc*
(a=Pour(o,oc) $\wedge$*Holds(Contains(oc,o),t)* $\wedge$*Holds(Downward(oc,t))* $\wedge$*Holds(Liquid(o),t))*
$\vee$
$\exists$*o,oc1,oc2*
(a=PourTo(o,oc1,oc2) $\wedge$*Holds(Contains(oc,o),t)* $\wedge$*Holds(Downward(oc,t))* $\wedge$
Holds(Above(o,do),t) $\wedge$*Holds(Liquid(o),t)* $\wedge$*Holds(Upward(o), t))*
$\vee$
$\exists$*o,oc1,oc2 (a=Overflow(oc2)* $\wedge$*Holds(Full(oc2),t)* $\wedge$*Holds(PouringTo(o,oc1,oc2),t))*
$\vee$
$\exists$*c,o (a=Leak(o,c)* $\wedge$*Holds(Contains(c,o),t)* $\wedge$*Holds(Liquid(o,t))* $\wedge$*Holds(Porous(MaterialOf(c),t)))*
$\vee$
$\exists$*o,oc1,oc2*
(a=StopPourTo(o,oc1,oc2) $\wedge$*Holds(Contains(oc2,o),t)* $\wedge$*Holds(PouringTo(o,oc1,oc2),t))*
$\vee$
$\exists$*o,oc,do (a=StopFallTo(o,oc,do)* $\wedge$*Holds(Distance(o,do,0),t)* $\wedge$*Holds(FallingTo(o,oc,do),t))*
$\vee$
$\exists$*o,c (a=StopLeak(o,c)* $\wedge$*Holds(Quantity(o,c,0),t)* $\wedge$*Holds(Leaking(o,oc),t))*
$\vee$
$\exists$*o,oc1,oc2 (a=PourTo(O,OC1,OC2)* $\wedge$*t=0)*

I* (The completion of the *Initiates* predicate) is:

Initiates(a,f,t) $\Leftrightarrow$
$\exists$*oc (a=Rotate(oc)* $\wedge$*Holds(Upward(oc),t)* $\wedge$*f=Downward(oc))* $\vee$
$\exists$*oc (a=Rotate(oc)* $\wedge$*Holds(Downward(oc),t)* $\wedge$*f=Upward(oc))* $\vee$
$\exists$*o,c (a=Fall(o,c)* $\wedge$*f=Falling(o,c))* $\vee$
$\exists$*o,oc,do (a=FallTo(o,oc,do)* $\wedge$*f=FallingTo(o,oc,do))* $\vee$
$\exists$*o,oc (a=Pour(o,oc)* $\wedge$*f=Pouring(o,oc))* $\vee$
$\exists$*o,oc1,oc2 (a=PourTo(o,oc1,oc2)* $\wedge$*f=PouringTo(o,oc1,oc2))* $\vee$

$\exists o,c\ (a=Leak(o,c) \wedge f=Leaking(o,c))$
$\vee$
$\exists o1,o2,pt1,pt2$
$(a= Hit(o1,o2,pt1,pt2,\ crackforce(o1,o2,pt1,pt2)) \wedge Holds(HardertoBreak(MaterialOf(o2),MaterialOf(o1)),t)$
$\wedge$
$Holds(Hard(MaterialOf(o1),t)) \wedge Holds(Hard(MaterialOf(o2),t)) \wedge$
$Holds(ObjPart(pt1,o1),t) \wedge Holds(ObjPart(pt2,o2),t) \wedge f=CrackedContainer(o1))$
$\vee$
$\exists c\ (a=OpenUp(c) \wedge f=OpenContainer(c) \wedge Holds(CrackedContainer(c),t)) \vee$
$\exists c\ (a=OpenUp(c) \wedge f=Downward(c) \wedge Holds(CrackedContainer(c),t) \wedge Holds(UpCrack(c),t)) \vee$
$\exists c\ (a=OpenUp(c) \wedge f=Upward(c) \wedge Holds(CrackedContainer(c),t) \wedge Holds(DownCrack(c),t)) \vee$
$\exists c\ (a=Rotate(c) \wedge f=UpCrack(c) \wedge Holds(DownCrack(c),t)) \vee$
$\exists c\ (a=Rotate(c) \wedge f=DownCrack(c) \wedge Holds(UpCrack(c),t)) \vee$
$\exists c\ (a=Lay(egg) \wedge f=WholeEgg(o) \wedge Holds(NotYetLaid(egg),t))$

T* (The completion of the *Terminates* predicate) is:

$Terminates(a,f,t) \Leftrightarrow$
$\exists oc\ (a=Rotate(oc) \wedge f=Upward(oc) \wedge Holds(Upward(oc),t)) \vee$
$\exists oc\ (a=Rotate(oc) \wedge f=Downward(oc) \wedge Holds(Downward(oc),t)) \vee$
$\exists o,loc1,loc2\ (a=Move(o,loc2) \wedge f=At(o,loc1) \wedge Holds(At(o,loc1),t)) \vee$
$\exists o,oc\ (a=Fall(o,c) \wedge f=Contains(oc,o)) \vee$
$\exists o,c\ (a=Leak(o,oc) \wedge f=Contains(c,o)) \vee$
$\exists c\ (a=OpenContainer(c) \wedge f=ClosedContainer(c)) \vee$
$\exists c\ (a=Rotate(c) \wedge f=DownCrack(c) \wedge Holds(DownCrack(c),t)) \vee$
$\exists c\ (a=Rotate(c) \wedge f=UpCrack(c) \wedge Holds(UpCrack(c),t)) \vee$
$\exists egg\ (a=Lay(egg) \wedge f=NotYetLaid(egg)) \vee$
$\exists o,oc1,oc2\ (a=StopPourTo(o,oc2,oc2) \wedge f=PouringTo(o,oc1,oc2)) \vee$
$\exists o,oc,do\ (a=StopFallTo(o,oc,do) \wedge f=FallingTo(o,oc,do)) \vee$
$\exists o,c\ (a=StopLeak(o,c) \wedge f=Leaking(o,c))$

Continuation of Proof of Theorem 1:

From assumption **(1d)** and Axiom 32, we have that $\forall x,\ n\ Holds(Quantity(O,OC2,x),n)$
where $x=rateofqfall \cdot n$ and $x \leq ovolume(O,0)$.
Choose $N1$ such that $rateofqfall \cdot N1 = ovolume(O,0)$. Then
(1.1) $Holds(Quantity(O,OC2,X1),N1)$ where $X1=rateofqfall \cdot N1=ovolume(O,0)$.

From assumption **(1d)**, and Definitions 19 and 20, we have that
(1.2) $Holds(Liquid(O),0)$

From H*, T*, **(1.2)**, and ECF3, we have that
$Holds(Liquid(O),\ N1)$.
(That is, since nothing happens that could terminate O's liquid state, we know that the $Liquid(O)$ fluent is not clipped.)

We can similarly show that
(1.3) $Holds(ovolume(O)=x,0) \Leftrightarrow Holds(ovolume(O)=x,N1)$.

Since the left-hand side of **(1.3)** is true by assumption, the right-hand side is true. Thus, by **(1.1)** and Axiom 35, we have that
$Holds(Contains(OC2,O),\ N1)$.
Thus, there is an n such that given the assumptions, $Holds(Contains(OC2,O),\ n)$.
QED.

Theorem 2:

Statement of Theorem 2:
Assume the following. (Note that these conditions are identical to those of Theorem 1, with the exception of **(2c)**: here, the capacity of the destination container is smaller than the volume of the object that is being poured.)

(2a) *InitiallyP(ovolume(O) =q1)*
(2b) *InitiallyP(AvailCapacity(OC2) = q2)*
(2c) *q2 < q1*
(2d) *Happens(PourTo(O,OC1,OC2), 0)*
(2e) *n = q1/rateofqfall(O,OC1)*

As in Theorem 1, Let Λ be the union of all logical formulas in Section 3, the above assumptions, and all necessary uniqueness of names assumptions and domain closure axioms. We divide Λ into Φ, Δ, Σ, Ω, Ψ, and Π, as in the statement of Theorem 1.

Then,
Circ(Δ; *Happens)* $\wedge$Circ(Σ; *Initiates*, *Terminates*) $\wedge\Omega$ $\wedge\Psi$ $\wedge\Pi$ $\wedge\Phi$
$\models \exists n$ *Happens(Overflow(OC2), n)*

Proof of Theorem 2:

Overview of Proof of Theorem 2:

As in the proof of Theorem 1, we begin by computing the circumscription of *Happens*, *Initiates*, and *Terminates*. We show that the action of *PourTo* initiates the *PouringTo* fluent, and that as long as the fluent is true, the destination container fills up with liquid. Since the destination container *OC2* has less capacity than the volume of the liquid object *O*, there will be a point when *OC2* is full, but when the *PouringTo* fluent is still true. At that point, the overflow will be triggered.

Setup of Proof of Theorem 2:

We begin by completing the *Happens*, *Initiates*, and *Terminates* predicates. These are identical to the completions of these predicates, H*, I*, and T*, which were stated in the proof of Theorem 1.

Continuation of Proof of Theorem 2:

Without loss of generality, assume that *InitiallyP(AvailCapacity(OC2) = q3)*. By Axiom 18, $q2 \geq q3$, so $q1 \geq q3$.

From premise **(2d)** and 29, we have that
(2.1) *Initiates(PourTo(OC1,OC2), 0)*

Choose *N2 = q3/rateofqfall(x)*. By Axiom 34,
(2.2) *Holds(TotalObjectVolume(OC2) = q2, N)*.

Now, by H*, T*, ECF1, and ECF3,
Holds(Capacity(OC2) = q2, N) $\Leftrightarrow$*Holds(Capacity(OC2) = q2, 0)*.
Since the right-hand side is true by assumption, we get that
(2.3) *Holds(Capacity(OC2) = q2, N2)*

From **(2.3)**, Definition 15, and Axiom 18, we have that
(2.4) *Holds(Full(OC2), N2)*

Now, from H*, T*, ECF1, and ECF3, we get that
(2.5) *Holds(PouringTo(O,OC1,OC2), N2)*.

So from **(2.4)**, **(2.5)**, and 36, we get *Happens(Overflow(OC2), N2)*. Thus, $\exists n$ *Happens(Overflow(OC2), n)*.
QED

4.2 The Egg-Cracking Theorem

The premises of the egg-cracking theorem are taken from the problem statement. There is a bowl with sufficient capacity to hold an egg (we assume a bowl with a one-liter capacity); it is made of glass, and is empty in the initial situation. There is a whole egg.

We assume that the egg is hit against the rim of the bowl; that it is moved to a location directly above the bowl; that if necessary, the egg is rotated so that its crack faces down; and that the egg is then opened up. From these premises, we want to show that the egg inside is contained by the bowl.

Statement of Theorem 3:

Formally, assume the following:

Premises about the egg and the bowl:

Premise 1 InitiallyP(WholeEgg(Egg1))

Premise 2 InitiallyP(Eggshell1 = EggshellOf(Egg1))

Premise 3 InitiallyP(EggInside1 = EggInsideOf(Egg1))

Premise 4 InitiallyP(Glass(MaterialOf(Bowl1)))

Premise 5 InitiallyP(Capacity(Bowl1) = 1000)))

Premise 6 InitiallyP(Empty(Bowl1))

Premise 7 InitiallyP(DirectlyAbove(Loc1,Bowl1))

Premise 8 InitiallyP(OpenContainer(Bowl1))

Premise 9 InitiallyP(Upward(Bowl1))

Premises about the actions that occur in this narrative:

Premise 10 Happens(Hit(centralband(Eggshell1),Rim(Bowl1),
crackforce(Eggshell1,centralband(Eggshell1),Rim(Bowl1))),0)

Premise 11 Happens(Move(Eggshell1, Loc1), 3)

Premise 12 Holds(UpCrack(Eggshell1), 5) ⇒Happens(Rotate(Eggshell1), 5)

Premise 13 Happens(OpenUp(Eggshell1), 8)

As in Theorem 1, Let Λ be the union of all logical formulas in Section 3, the above assumptions, and all necessary uniqueness of names assumptions and domain closure axioms. We divide Λ into Φ, Δ, Σ, Ω, Ψ, and Π, as in the statement of Theorem 1.

Then,

Circ(Δ; *Happens*) $\wedge$Circ(Σ; *Initiates*, *Terminates*) $\wedge\Omega$ $\wedge\Psi$ $\wedge\Pi$ $\wedge\Phi$
$\models \exists n > 8$ *Holds(Contains(Bowl1,EggInside1), n)*

Proof of Theorem 3:

Overview of Proof of Theorem 3:

As in the proof of Theorem 1, we begin by computing the circumscription of *Happens*, *Initiates*, and *Terminates*. We show that hitting the central band of the eggshell against the rim of the bowl results in a cracked eggshell. The eggshell, however, is still a closed container, and remains so during the action of moving the eggshell to the location above the bowl and rotating the egg so that its crack faces down. Thus, the egg inside remains inside the eggshell. When the eggshell is opened, however, the eggshell becomes an open container. Since the opening is facing down, the liquid egg will start to fall (pour) out. Since the eggshell was previously moved to a location directly above the bowl, and since nothing has happened that would change its location, the eggshell is still directly above the bowl when the egg inside starts pouring out. Thus, the egg inside falls into the bowl. Now, the bowl was initially empty, and until the eggshell was opened up, no action happened that would change the bowl's empty status. Thus, since the capacity of the bowl is larger than the volume of the egg, we can show that the egg will eventually be contained by the bowl, using a similar argument to that used in proving Theorem 1.

Setting up the Proof of Theorem 3:

As in Theorems 1 and 2, we begin by completing the predicates *Happens*, *Initiates*, and *Terminates*. The completions of the *Initiates* and *Terminates* predicates are identical to I* and T* of Theorems 1 and 2 (since the underlying causal theory is identical). The completion of the *Happens* predicate is somewhat different, because the actions in the narrative are different. To get the completion of the *Happens* predicate, take H* of Theorems 1 and 2, delete the last disjunct, and add the following disjuncts:

$\vee$
(a = Hit(centralband(Eggshell1),Rim(Bowl1), crackforce(Eggshell1,centralband(Eggshell1),Rim(Bowl1)))
$\wedge$*t = 0)*
$\vee$*(a = Move(Eggshell1,Loc1)* $\wedge$*t=3)*
$\vee$*(a = Rotate(Eggshell1)* $\wedge$*t=5* $\wedge$*Holds(UpCrack(Eggshell1),5))*
$\vee$*(a = OpenUp(EggShell1)* $\wedge$*t=8)*

Domain closure gives us that we have only one bowl, one egg, etc.

Continuation of Proof:

From Lemma 3, Premise 4, and Axioms 6, 63, 7, 66, and 48, we have
Initiates(Hit(centralband(Eggshell1),Rim(Bowl1), crackforce(Eggshell1,centralband(Eggshell1),Rim(Bowl1))), CrackedContainer(Eggshell1), 0).

Thus, from Premises 1, 2, and 10 and ECF1, we have that
$\forall t > 0 \neg$ *Clipped(CrackedContainer(Eggshell1),t)* $\Rightarrow$*Holds(CrackedContainer(Eggshell1), t).*

Furthermore, from ECF3, T*, and H*, we have that
(3.1) *Holds(CrackedContainer(Eggshell1), 3).*
That is, nothing happens to terminate the CrackedContainer status of Eggshell1 between 0 and 3.

We can repeatedly use ECF3, T*, and H* to show that all other fluents (with the exception of *WholeEgg*) that held at 0 still hold at 3.

From Premise 11, Axiom 22, and ECF1, we get that
$\forall t > 3 \neg$ *Clipped(3,At(Eggshell1,Loc1), t)* $\Rightarrow$*Holds(At(Eggshell1,Loc1), t)* .
Moreover, from H*, I*, and T*, we get that
$\neg$ *Clipped(3, At(Eggshell1, Loc1), 8)*; thus, from ECF1,
(3.2) $\forall t \in$ *(3,8] Holds(At(Eggshell1, Loc1), t)*

Now consider the orientation of Eggshell1. By Axiom 51 the crack in Eggshell1 is either facing up or down at time 5. If it is facing down, then from H*, I*, T*, and ECF1, we have that *Holds(DownCrack(Eggshell1), 8).* If it is facing up, then from Premise 12, Axiom 56, ECF1, H*, I*, and T*, we also get that *Holds(DownCrack(Eggshell1), 8).* Thus, in either case, we have that
(3.3) *Holds(DownCrack(Eggshell1), 8).*

From H*, I*, and T*, we get that
(3.4) *Holds(CrackedContainer(Eggshell1), 8)*,
which entails that *Holds(Package(Eggshell1,EggInside1), 8)* and thus
(3.5) *Holds(Contains(Eggshell1, EggInside1), 8)*

Now, from **(3.4)**, Premise 2 and Axioms 62 and 49, we have that
Initiates(OpenUp(Eggshell1), OpenContainer(Eggshell1), 8)

Likewise, from **(3.3)**, Premise 2, and Axiom 53, we get that
Initiates(OpenUp(Eggshell1), Downward(Eggshell1), 8)

Thus, from Premise 13 and ECF1, we get that
(3.6) $\forall t > 8 \neg$ *Clipped(8,OpenContainer(Eggshell1), t)* $\Rightarrow$*Holds(OpenContainer(Eggshell1, t))*
and **(3.7)** $\forall t > 8 \neg$ *Clipped(8,Downward(Eggshell1), t)* $\Rightarrow$*Holds(Downward(Eggshell1, t))*

Using Axiom 25 and Definitions 19 and 20, along with Premises 3 and 8 and Axiom 64 we have that for some t' > 8,
Happens(Pourto(EggInside1,Eggshell1,Bowl1), t')

(This depends upon the fact that the liquidity of the inside of the egg, the location of the bowl, etc. persist from time 8 until time t'. This is trivial to show for most properties. To see that *Eggshell1* contains *EggInside1* at time t', consider that the only way for the fluent *Contains(Eggshell1, EggInside1)* to be clipped between 8 and t' is if there were some other fall of *EggInside1* from *Eggshell1* between 8 and t'. But if this is the case, we would simply let t' be this earlier time.)

It is left only to show that there is some n such that
Holds(Contains(Bowl1, EggInside1), t'+n).
We can show this using Axiom 65, Premises 6, 9, 5, and 8, and using an argument identical to that of Theorem 1.
QED

4.3 Elaborations

Recall the second paragraph of the problem statements, giving the elaborations or variants:

What happens if: The cook brings the egg to impact very quickly? Very slowly? The cook places 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 theory presented in Section 3 can handle some of these elaborations. Given the brittleness of typical commonsense formalizations, we consider this a reasonable success. Below, we discuss which variants the theory can handle, and how the theory might be extended to handle other variants.

4.4 Elaborations that the Theory can Handle

4.4.1 The bowl is made of looseleaf paper

As desired, the egg-cracking plan will not work in this case, first because looseleaf paper is soft, and therefore, hitting the eggshell against the looseleaf will not crack the eggshell, and second, because looseleaf paper is porous, and therefore the egg will (eventually) leak out of the bowl.

This variant is easily handled by adding axioms giving the properties of looseleaf paper:

Axiom V1a: *Paper(m)* $\Rightarrow$*Soft(m)*

Axiom V1b: *Paper(m)* $\Rightarrow$*Porous(m)*

In addition, Premise 4 is replaced by the premise:

Premise V1c: *InitiallyP(Paper(MaterialOf(Bowl1)))*

We could, but do not, add axioms comparing looseleaf paper and eggshells in terms of the predicate *HardtoBreak*; the axioms above will be sufficient for our purposes.

It will not be possible to prove the egg-cracking theorem given these premises. In particular, Lemma 3 (or the analogous lemma for paper objects) will not hold. The antecedents of Axiom 48 are not satisfied. Thus, the fluent *CrackedContainer(Eggshell1)* is not initiated at time 0, and therefore the *OpenUp(Eggshell1)* action will not result in an opencontainer.

Indeed, because *Happens*, *Initiates*, and *Terminates* are circumscribed, one can prove that there is no successful egg-cracking in this narrative.

Suppose, however, that one would change the narrative in the following way. As before, we would replace Premise 4 with Premise V1c. We would also delete Premise 10 and add the following premises:

Premise V1d: *InitiallyP(Wood(MaterialOf(Table1)))*

Premise V1e: *FlatObject(Table1)*

Premise V1f:
Happens(Hit(centralband(Eggshell1), Edge(Table1), crackforce(Eggshell1, centralband(Eggshell1), Edge(Table1))), 0)

and add the following axioms to the theory:

Axiom V1g: *Wood(m) ⇒Hard(m)*

Axiom V1h: *Wood(m) ⇒Hard-to-Break(m)*

Axiom V1i: *ObjPart(Edge(fo), fo)*

In this case, the egg would be cracked (because it would be hit against a hard object which is harder than an eggshell) and broken open. The egg would fall into the bowl, and indeed be contained by the bowl, at some point after time 8. However, because the material of the bowl is paper, and paper is porous, we can predict that the egg will leak out of the bowl (Axiom 42).

4.4.2 The bowl is made of clay

There are two ways to analyze this variant. We assume, first, that soft clay stays soft indefinitely. To handle this variant, we add the following axiom to the theory:

Axiom V2a: *SoftClay(m) ⇒Soft(m)*

We replace Premise 4 by:

Premise V2b: *InitiallyP(SoftClay(MaterialOf(Bowl1)))* As is the case with the looseleaf paper bowl, the egg-cracking theorem cannot be proved because the Hit action will not result in a cracked eggshell.

Again, in this case, if the eggshell is hit against the edge of a wood table, the egg cracking will succeed. However, in this case, the egg will remain contained inside the bowl; it will not leak out, because clay is not porous.

In real life, clay starts out soft, but eventually hardens. We could formalize this fact with the following axioms

Premise V2b: *Holds(SoftClay(MaterialOf(o)), t) ∧n > 3000 ⇒Holds(HardClay(MaterialOf(o)), t + n)*

Premise V2c: *Holds(HardClay(MaterialOf(o)),t) ⇒Hard(MaterialOf(o)),t)*

If the narrative were then changed so that the first action happened at t=4000 (and similarly for the other actions), the egg-cracking theorem could be proved.

4.4.3 The bowl is smaller than the egg

As expected, the egg-cracking scenario will work. However, the egg cannot be contained by the bowl. The size of the bowl should not affect the egg cracking itself. (Although when the bowl is extremely small, the rim may be so small that hitting the eggshell against it may be impractical.) However, the egg will not be contained by the bowl; it will overflow the bowl.

We replace Premise 5 with
Premise V3a: *InitiallyP(volume(Bowl1) = 15*
(That is, the bowl has a volume of just 1 tablespoon.)

The proof will carry through most of the way as before, with the eggshell becoming an open container after the *OpenUp* action at time 8, and the egg falling out just after. Using the argument structure of Theorem 2, it can be shown that the egg will overflow the container.

4.5 Elaborations that the Theory Cannot Handle

4.5.1 The cook brings the egg to impact very quickly or very slowly

One would expect that when the egg is brought to impact very quickly, the egg will shatter; that when the egg is brought to impact very slowly, the egg may not crack at all.

The theory incorporates a concept of force. The function *crackforce*, given the object to be cracked, and the relevant object parts, returns the proper amount of force; force is one of the arguments to the *Hit* action. To handle the variants of very quick and very slow impacts, one could add the definition of force as mass times acceleration, and prove that if the mass stays the same, but the acceleration increases (resp. decreases), the force increases (resp. decreases). One would probably have to modify the function *crackforce* so that it returned the amount of force needed consistent with a "normal" speed; one could then prove that a high-speed impact would result in too great a force and a low-speed impact would result in too small a force. One could then show that the egg cracking would not be successful.

However, we would expect the theory to entail, for example, that too great a force shatters the object, under the right circumstances. There should probably be some axiomatization of shattering. Thus far, we have not spoken explicitly about an object being broken into physically discontinuous pieces (although this is what happens to a liquid object during a pour); this would need to be incorporated into the theory.

We believe that there is nothing in the theory presented in Section 3 that is not consistent with such an extension. However, it would likely involve a fair amount of work to develop the extensions needed.

4.5.2 The bowl is upside down

For simplicity, we consider the case where the bottom of the bowl is flat. If the bowl is upside down, one cannot access the edge of the bowl for egg cracking. The theory has no knowledge of feasible or infeasible actions: one would need to integrate a theory of planning into Shanahan's circumscriptive event calculus. Some initial work in this direction has been done by Shanahan (2000). In addition, one would have to axiomatize feasible and infeasible actions in this domain. In particular, one would have to describe the collisions that are feasible.

If one hits the egg against another object part (such as the edge of the table, as in Section 4.4.1), the egg will crack. However, if one breaks open the egg over the bowl, the egg will fall onto a flat surface rather than into an open container. The theory can already predict that the egg will spread (Axiom 41). However, we would also have to add axioms specifying how far a liquid will spread (this depends on the viscosity and the volume of the liquid) and specifying what happens when the

liquid reaches the boundaries of the flat object.

4.5.3 The cook tries this procedure with a hard-boiled egg, a coconut, or an M&M

One would expect that in these cases, going through the steps in the narrative — hitting the outside of the object against the rim of a glass bowl, moving the object to a location above the bowl, and "opening up" the outside of the object — will not result in the inside of the object falling into the bowl.

In all cases, one problem that arises is that the outside of the object — the eggshell, the candy coating, the coconut shell — adheres to the object inside: the hard-boiled egg, the chocolate center of the M&M, the coconut itself. Because of this, the inside object will not fall out.

The theory in Section 3 does not incorporate a notion of adhesion. As it happens, it will not predict that the inside object will fall out. But it refrains from this prediction for the wrong reasons. Hard-boiled eggs, chocolate, and coconut meat are solid; thus, in order to apply Axiom 25, one would have to know that the object is smaller than the opening of the open container. In fact, the axiom on *OpenUp* (Axiom 49) does not specify how large the opening is. So the consequent of Axiom 25 is not fired.

To correctly analyze the problem, one would need to characterize the size of the openings in open containers. One would also need to formalize the concept of adhesion. In addition, we note that even if these were formalized, we would still have to deal with situations like the coconut, where the coconut is so hard to break that a standard hitting action will not suffice, and other methods must be used.

4.5.4 The cook puts the egg in the bowl and exerts steady pressure with his hand.

In this case, one expects that the eggshell will be crushed, turning the eggshell and the inside of the egg into a messy mixture of eggshell fragments, egg yolk, and egg white.

The theory cannot handle this variation at all. The cook and the cook's hands are entirely ignored in the theory. The action of crushing has not been formalized. There is no concept of a mixture.

These are all complex concepts. To handle this variant, several core theories — on mixture, crushing, etc. — must be developed and integrated into the existing theory.

4.5.5 The cook, having cracked the egg, attempts to peel it off its contents like a hard-boiled egg

If the cook attempts this while the egg is over the bowl, the egg will wind up in the bowl, although a bit more messily than would happen otherwise. However, the eggshell would be fragmented into several pieces. The theory currently does not support a theory of fragmentation; this would need to be developed and integrated into the existing theory.

5 Other Issues in Temporal Reasoning

We claimed in Section 1 that considering in detail a non-toy problem could point out interesting commonsense reasoning problems or difficulties in an existing theory of commonsense reasoning. During the formalization of the egg-cracking problem, we came across two such issues in temporal reasoning, which we discuss below.

5.1 The Initial Specification Problem

The Initial Specification problem is a variant of the frame problem that occurs in temporal languages in which actions can be triggered by fluents.

Shanahan's circumscriptive event calculus solves the frame problem by ensuring that actions do not have unexpected consequences and that unexpected actions do not happen. Minimizing unexpected actions is not the same as minimizing action occurrences. In particular, the two are not the same in theories in which actions can be *triggered*. There are two ways of expressing the triggering of actions in a temporal language: allowing actions to trigger other actions [12], and allowing fluents to trigger actions. Shanahan (1988) uses fluents to trigger actions. We have followed his style in this paper, and thus, for example, the fluents *Full* and *PouringTo* trigger an *Overflow* action.

Using fluents to trigger actions can, however, result in a theory with unexpected consequences unless one is careful. Consider the egg-cracking theory of Section 3, augmented by an axiomatization of dripping faucets.

Axiom 67 *Holds(Broken(Washer(f)), t)* $\Rightarrow \exists q$ *Holds(Dripping(f,q),t)*

Axiom 68 *Holds(Dripping(f,q), t)* $\wedge$ *Holds(DirectlyAbove(f,oc), t)* $\wedge$ *Holds(Upward(oc), t)* $\Rightarrow$ *Holds(DrippingInto(f,oc,q), t)*

Axiom 69 *Holds(QuantityLiquid(oc,x), t)* $\Rightarrow$ *Trajectory(DrippingInto(f,oc,q), t, QuantityLiquid(oc, x+d·q), d)*

Now assume the following narrative:
InitiallyP(At(Faucet1,Loc1))
InitiallyP(At(Bowl1, Loc2))
Initially(Capacity(Bowl1) = 500)).

It would be reasonable to predict that nothing happens, since we are not told of anything that happens, and none of the fluents that trigger actions are known to be true. Thus, it would be reasonable to expect that it would be a theorem of the theory that nothing happens at the time t = n (for various values of n).

However, this is not the case. Although *Happens*, *Initiates*, and *Terminates* are minimized, this is *not* enough to minimize unexpected action occurrences. This is because triggered actions are not minimized. Specifically, although the narrative says nothing about *Faucet1* dripping, models of the narrative include those in which it is dripping, and those in which it is not dripping. Likewise, there is no way to prefer models in which *Faucet1* is not directly above *Bowl1*, *Bowl1* is not upward, etc. Thus, there are models of the circumscribed theory in which the faucet drips into the bowl, and eventually the bowl overflows.

To prevent this undesirable consequence, it would be necessary to specify all the relevant conditions in the initial situation. For example, there would be no difficulty in the case above if the premise *InitiallyP(Upward(Bowl2))* or the premise *InitiallyP(Empty(Bowl2))* were added to the theory.

In general, the Initial Specification Problem, which arises in a theory that allows actions to be triggered by fluents, is that given a problem description in which not all information is known about the initial situation, unexpected consequences may occur. One can rule out these unexpected consequences by furnishing more information about the initial situation, but this approach has difficulties. First, it is unclear how one can efficiently determine which fluents are "relevant" in a particular initial situation. Second, one may not have enough information to more fully specify the theory. What if, in the example above, one does not know what the orientation of Bowl2 is? On the one hand, one certainly does not want to posit as premises statements that are not known to be

[12] see (Morgenstern and Stein 1988) for an early discussion of actions which trigger other actions.

true. On the other hand, there are some situations, such as the dripping faucet, that are relatively uncommon. We would like some way of excluding them from consideration unless we have a good reason to believe that they do hold.

These issues seem difficult to resolve. Possibly, it is more fruitful to examine ways to avoid the Initial Specification Problem — perhaps by eschewing a language in which actions are triggered by fluents, and restricting oneself to actions triggered by other actions. It would always seem possible to refer back to the triggering action, rather than the triggering fluent. For example, one could have a *Break* action that breaks the washer of the faucet, and state that this action initiates the dripping faucet. In certain cases, restructuring the theory in this way may be somewhat unnatural.

The problem is deferred to future research.

5.2 The Unobtainable State Problem

> *Have a fresh, raw egg decorated like Humpty Dumpty. Recite nursery rhyme using the egg as Humpty Dumpty. During the second or third recitation, "accidentally" drop Humpty Dumpty. When he falls and breaks, act surprised and ask how we will repair him. Explain that all the king's horses and all the king's men couldn't put Humpty together again. Then allow each child an opportunity to try to repair the egg ... The children will learn that not all things can be fixed.*
>
> – from teachers.net, an online resource for teachers

Axiom 58 states an obvious fact about eggs: that any egg either is not yet laid, or is whole, or is cracked, or is broken, and that these stages are mutually exclusive. It is also a fact that these stages are ordered: an egg is not yet laid before it is whole, whole before it is cracked, and cracked before it is broken. Once one passes from one stage to the next, there is no going back.

We do not need to state this second fact as an axiom. The ordering of egg stages — the irreversibility of change in the stages of an egg — is a theorem in the circumscribed theory. To see that this is so, consider that circumscribing *Happens*, *Initiates*, and *Terminates* means that one effectively limits causation: if a cause — a way of getting from fluent *f1* to fluent *f2* — is not explicit in the theory, or cannot be derived from the axioms in the theory, it is not true in any of the models of the circumscribed theory. Indeed, in such cases, if the latest mentioned time point of the narrative portion of the theory [13] is *tn*, then it is a theorem that if *Holds(f1,t2)* where $t2 \geq tn$, then $\forall t3 > t2 \neg$ *Holds(f2,t3)*. In particular, the conclusion of Humpty Dumpty is a theorem — once the egg is broken, nothing in the world can make it whole again. We call such a theorem, stating that one cannot get from fluent *f1* to fluent *f2*, an *unobtainable state* theorem.

The difficulty is that one can get many similar theorems in one's theory, many of which might not be desirable. For example, in the theory there are no axioms that specify how the (total) capacity of a container might be changed. Thus, it is a theorem of the circumscribed theory that the capacity of a container never changes. But this is rather unsatisfying. There is a real difference between the theorem that there is no way to get from a broken egg to a whole egg, and the theorem that there is no way to change the capacity of a container. Intuitively, the difference is due to the fact that there is no mention at all in the theory of actions that change the capacity of a container. The theorem that there is no way to change the capacity of a container is true only because we have not introduced and axiomatized the actions that could change the capacity of a container (such as crushing a styrofoam container). On the other hand, the theorem that there is no way to get from a broken egg to a whole egg is true because, in fact, there are no actions that get you from a broken egg to a whole egg.

The issue is not that we get unexpected conclusions (as is the case in the Yale Shooting Problem). In the circumscribed event calculus, or in any theory which minimizes causation in a similar manner,

[13] One can always use the narrative to work around any limitations of one's causal theory. For example, in the narrative presented in Section 4.2, one could add the premise *Holds(WholeEgg(Egg1), 10)*. This statement would then *override* the causal theory.

it is clear that if there are two fluents *f1* and *f2* in the language, and there are no axioms describing how one can get from *f1* to *f2*, then there is no way to get from *f1* to *f2*. Rather, the issue is that some *unobtainable state* theorems are meaningful, while others are merely the result of having an impoverished theory. The Unobtainable State Problem is the problem of determining whether an unobtainable state theorem is indeed meaningful. [14]

We do not have a solution to the Unobtainable State Problem; this awaits future work. It may be possible to characterize when theories are "impoverished" with respect to a particular fluent. For example, one could say that if a fluent *f* is part of the underlying language of a theory, and if there are no axioms in the causal part of the theory mentioning *f* in their consequent, then the theory is impoverished with respect to *f*. (Given this characterization, the egg-cracking theory is impoverished with respect to *Capacity*.) One could then say that an unobtainable state theorem, stating that one cannot get from *f1* to *f2*, is not meaningful if the theory is impoverished in *f2*.

Closely related to the Unobtainable State Problem is the Limited State Change Problem. There are many theorems that say that one can go from fluent *f1* to fluent *f2* by performing one of a limited set of actions. We call such theorems Limited State Change Theorems. For example, it is a theorem that the only way to get from *WholeEgg* to *CrackedEgg* is through the *Hit* action. Given the incomplete nature of all existing formalizations of commonsense reasoning, it is clear that the set of actions specified in a Limited State Change theorem will sometimes (actually, quite often) be only a subset of the set of actions which could effect a change. (For example, in the real world, there are many ways to crack an egg, including jostling and dropping.) That is, there are more ways of getting from one state to another than are captured in any partial formalization.

This does not mean, of course, that limited state change theorems are never meaningful. Perhaps meaningfulness is best measured by degree. In addition, the way in which a formalization is to be used is also an important consideration. For a planning system, for example, a Limited State Change theorem or an Unobtainable State Change theorem may be useful, even if it does not accurately model the world. Further investigation into these issues is deferred to future research.

6 Conclusion and Future Work

We have developed a mid-sized axiomatization that gives a partial characterization of containment, falling, and pouring, and have shown that using this theory, one can prove general theorems about successful and unsuccessful pourings. We have given a very partial characterization of breaking open objects and a brief characterization of eggs, and have shown that we can handle the simplified version of the benchmark egg-cracking problem (in which the agent's hands are ignored).

The theory was built using Shanahan's circumscriptive event calculus, a powerful and expressive temporal language. The expressivity was necessary to describe the complexities of the domain. However, as we pointed out, certain features of the circumscriptive event calculus, such as the fact that fluents can trigger events, result in the Initial Specification Problem. This problem, a close relative of the frame problem, seems to arise only when we have this particular expressive power (though more investigation needs to be done). On the other hand, the Unchangeable State Problem, the problem of determining when theorems stating that one cannot get from one fluent to the next are meaningful, would seem to arise in most nonmonotonic temporal theories. It would be worthwhile to investigate both problems in the future.

It would be interesting to see whether the core theory fragments developed for solving the egg-cracking problem can be reused for solving other benchmark commonsense problems. It would

[14] It should be noted that this problem would not be avoided or solved by using the classic microworlds methodology, where one first formally characterizes one's model before one writes down any axioms. Presumably, one determines which actions allow transitions between which fluents as one defines the model, and therefore, knows a priori the types of unobtainable state theorems the model entails. There is still, however, no obvious way to determine which of these theorems are meaningful.

seem, for example, that the fragments on containment and pouring could be reused for John Bell's Eating on an Airplane example (Bell 1998). On the other hand, it is clear that to handle Davis's Falling Objects problem, the theory of falling must be extended, and a comprehensive theory of the properties of materials must be added.

We have reason to believe that the theory we have developed will prove amenable to reuse. First, it was designed with reuse in mind; second, we have been able to handle at least some of the variants, or elaborations, given. On the other hand, we know that integration and extension are never easy tasks, and we can anticipate that problems will arise.

We argued in Section 1 that it is desirable, when formalizing commonsense domains, to use existing formalizations of commonsense reasoning to the extent possible. Thus, we built our theory on Shanahan's circumscriptive event calculus. Another obvious candidate for integration is Hayes's (1985a) theory of liquids. Hayes has developed a rich and complex theory of liquids, which allows one to reason about contained spaces of liquid (such as a lake), pieces of liquid (which correspond to liquid objects in our theory), and liquid objects, which are four-dimensional liquid entities over some period of time.

Hayes's theory allows many inferences not handled by our theory, partly because we have not considered many basic properties of liquids. For example, in Hayes's theory, one can show that if a container contains some liquid, then the inside surface of the container is wet. In addition, his theory avoids some of the anomalies present in ours. In our theory, when one pours a glass of water into a leaking container, the water is first contained by the leaking container; and then the water begins leaking out. It is possible to avoid this anomaly, but one has to explicitly add axioms to do so. Hayes's theory can describe the situation elegantly; during most of the pouring-leaking history, there are two simultaneous events: a "leaving" through some imaginary portal between the glass and the container, and a "leaving" through the leaking container.

For all its advantages, however, Hayes's theory has one feature which makes integration difficult. He uses the concept of histories — a four-dimensional piece of space-time — as the underlying entity of his temporal language. One would need to rework Hayes's theory of liquids using the circumscriptive event calculus before it could be integrated with our theory. This is not a simple task; it was beyond the scope of this work, but would be a challenging future research project.

In general, we believe that research in mid-sized axiomatizations in commonsense reasoning ought to move in several directions. First, there should be an emphasis on using existing theories or integrating theories. Second, it would be helpful if the commonsense reasoning community were to work toward solving several benchmark commonsense reasoning problems. Several sets have been proposed (Miller and Morgenstern 1999; Sandewall 1999), but until these problems enter the consciousness of the commonsense reasoning community, in the way that the Yale Shooting Problem or the Suitcase Problem has, work on these problems will continue to proceed only in isolation. Third, at least in the initial stages of efforts at mid-sized axiomatizations, multiple formalizations of individual benchmark problems are very useful. We need a critical mass of reasonably sized, carefully crafted axiomatizations both to help move toward the eventual goal of developing a formalization of large chunks of the commonsense world, and to help establish a set of criteria for evaluating axiomatizations. Some brave attempts have been made (Sandewall 2000), but as Nagel (Nagel 1961) has pointed out, the establishment of evaluative criteria can happen only when there is a critical mass of work in the area.

Given the relative paucity of existing work in this area, this may take a while to happen. In the meantime, we should be motivated by the fact that embarking on a mid-sized axiomatization of this sort is no textbook exercise; a detailed investigation into a problem of reasonable scope almost always results in encountering new and challenging research problems. Of course, it is the long-term picture, the construction of axiomatizations that will eventually solve larger sets of problems, that remains the ultimate motivation for our work.

References

[1] Eyal Amir. Towards a formalization of elaboration tolerance: Adding and deleting axioms. In Mary-Anne Williams and H. Rott, editors, *Frontiers of Belief Revision*. Kluwer, 2000.

[2] John Bell. Eating on an airplane. *The Commonsense Problem Page*, 1997. http://www-formal.stanford.edu/leora/cs.

[3] Ernest Davis. The egg cracking problem. *Commonsense Problem Page*.

[4] Ernest Davis. A logical framework for commonsense predictions of solid object behavior. *Artificial Intelligence in Engineering*, 3(3):125–140, 1988.

[5] Ernest Davis. The kinematics of cutting solid objects. *Annals of Mathematics and Artificial Intelligence*, 9(3,4):253–305, 1993.

[6] Ernest Davis. Falling objects. *Commonsense Problem Page*, 1997. http://www-formal.stanford.edu/leora/cs.

[7] Ernest Davis. The naive physics perplex. *AI Magazine*, 19(4):51–79, 1998.

[8] Ernest Davis. Guide to axiomatizing domains in first-order logic. *Electronic Newsletter on Reasoning about Actions and Change*, 99002, 1999. At http://www.etaij.org/rac/.

[9] Johan de Kleer. Multiple representations of knowledge in a mechanics problem solver. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence, IJCAI-77*, pages 299–304, 1977.

[10] Johan de Kleer and John Seely Brown. A qualitative physics based on confluences. In Daniel Bobrow, editor, *Qualitative Reasoning about Physical Systems*. MIT Press, Cambridge, Massachusetts, 1985.

[11] Kenneth Forbus. Spatial and qualitative aspects of reasoning about motion. In *Proceedings of the First National Conference on Artificial Intelligence, AAAI-1980*, pages 436–442, 1980.

[12] Steven Hanks and Drew McDermott. Nonomonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1997.

[13] Patrick Hayes. Naive physics i: Ontology for liquids. In Jerry Hobbs and Robert Moore, editors, *Formal Theories of the Commonsense World*, pages 71–107. Ablex, Norwood, New Jersey, 1975.

[14] Patrick Hayes. The second naive physics manifesto. In Jerry Hobbs and Robert Moore, editors, *Formal Theories of the Commonsense World*, pages 1–36. Ablex, Norwood, New Jersey, 1975.

[15] Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.

[16] Doug Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, Reading, Massachusetts, 1990.

[17] Vladimir Lifschitz. Cracking an egg: An exercise in commonsense reasoning. Fourth International Symposium on Logical Formalizations of Commonsense Reasoning, http://www.cs.utexas.edu/users/vl/mypapers/egg.ps, 1998.

[18] Fangzhen Lin. Embracing causality in specifying the indirect effects of action. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI-95*, pages 1985–1993, 1995.

[19] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI-95*, pages 1978–1984, 1995.

[20] John McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London, 1959. Her Majesty's Stationary Office.

[21] John McCarthy. Mathematical logic in artificial intelligence. *Daedalus*, pages 297–311, 1988.

[22] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[23] Rob Miller and Leora Morgenstern. The commonsense problem page. 1997. http://www-formal.stanford.edu/leora/cs.

[24] Leora Morgenstern. Beyond toy problems: A logical formalization of the egg-cracking domain. Fourth International Symposium on Logical Formalizations of Commonsense Reasoning, http://www-formal.stanford.edu/leora/cs98/egg.a.ps, 1998.

[25] Leora Morgenstern and Lynn Andrea Stein. When things go wrong: A formal theory of causal reasoning. In *Proceedings of AAAI-1988*, pages 518–523, 1988.

[26] Ernest Nagel. *The Structure of Science*. Harcourt, Brace, and Co., New York, 1961.

[27] Adam Pease, Vinay Chaudhri, Fritz Lehmann, and Adam Farquhar. Practical knowledge representation and the DARPA high performance knowledge bases project. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*, 2000.

[28] Erik Sandewall. *Features and Fluents*. Oxford University Press, Oxford, England, 1994.

[29] Erik Sandewall. Logic modelling workshop. *Electronic Colloquium on Reasoning about Actions and Change*, 1999. http://www.ida.liu.se/ext/eta/lmw.

[30] Erik Sandewall. On the methodology of research in knowledge representation and commonsense reasoning. In Jack Minker, editor, *Logic-Based Artificial Intelligence*. Kluwer Academic Publishers, Dordrecht, 2000. Forthcoming.

[31] Murray Shanahan. *Solving the Frame Problem*. MIT Press, Cambridge, Massachusetts, 1997.

[32] Murray Shanahan. A logical formalisation of Ernie Davis's egg cracking problem. Fourth International Symposium on Logical Formalizations of Commonsense Reasoning, http://www-formal.stanford.edu/leora/cs, 1998.

[33] Lynn Andrea Stein and Leora Morgenstern. Motivated action theory. *Artificial Intelligence*, 71:1–41, 1994.