Radically Incomplete Reasoning about Containers: A First-Order Theory

Ernest Davis

September 28, 2013

This paper is a supplement to the paper, "Reasoning about Containers and Manipulation using Radically Incomplete Information" (Davis, Marcus, and Chen, 2013).[1] We present an axiomatic theory for qualitative reasoning about containers and manipulation. Motivation and larger context for this work is presented and manipulation or developing cognitive models of this reasoning is in the main paper. This supplement is a bare-bones account of the details of the first order theory and of its use in supporting sample inferences.

This supplement is a work in progress. In particular, the development of the axiomatic theory has far outrun the formalization of examples that use the axioms. This imbalance should be corrected over time. The axiomatization here is certainly not intended as a complete qualitative theory of containers.

1 Microworld

We have in mind a microworld of the structure described below. We have not given a formal characterization of the microworld, still less a formal semantics of the symbols; but it should be reasonably clear that such an account could be given in a way that makes the axioms true.

1.1 Ontology

Time is forward-braching and continuous; each time line is isomorphic to the real line $\mathbb{R}$. Forward branching corresponds to an agent's choice between actions. Branches occur *after* instants; that is, an interval that is bounded and open on the right has a unique least upper-bound, but there can be any number of non-overlapping intervals with the same lower bound.

Space is assumed to be three-dimensional Euclidean space $\mathbb{R}^3$.

Nothing that is yet in the theory requires such a high-powered spatio-temporal theory; in fact, the theory in its current form can probably be instantiated in a discrete branching model of time and a finite model of space. However, the theory is constructed with these continuous models in mind.

Objects are distinct; that is one object cannot be part of another or overlap with another. They are eternal, neither created nor destroyed. They move continuously. Distinct objects cannot overlap spatially. They are not required to be connected. An object occupies a region of some three-dimensional extent (technically, a topologically regular region); it cannot be a one-dimensional curve or two-dimensional surface.

[1] http://www.cs.nyu.edu/faculty/davise/containers/Containers.pdf

This object ontology works well with solid, indestructible objects. It works much less well for liquids, thought it does not entirely exclude them; better ontologies for liquids are developed in (Hayes, 1985) and (Davis, 2008)

For any object `O`, there is some range of regions that `O` can in principle occupy, consistent with its own internal structure; these are called the feasible regions for `O`. For instance, a rigid object can in principle occupy any region that is congruent (without reflection) to its standard shape. A string can occupy any tube-shaped region of a specific length. A particular quantity of liquid can occupy any region of a specific volume.

There is a single agent, which itself is an object. The agent is capable of moving by itself and of directly manipulating one other object.

1.2 Physical Theory

We posit the following very general physical theory.

Two distinct objects do not overlap spatially.

The trajectory of an object is a continuous function of time.

An object `O` occupies a region feasible for `O`. More detailed theories of specific object categories would give geometrical details of what regions are feasible for a given object; we do not include any such axioms here.

The agent can hold an object `O` only if the agent is in contact with `O` along an extended face, and can manipulate the object only if he is holding it.

If the agent is holding an object, he can release it at any time.

If the agent is holding an object and releases it in an unstable position, then the object will fall for a short period of time. (We do not develop any theory of stability at all; and all that is asserted in our theory of falling is that the object does not move outside any upright container that it is currently in.)

Our theory characterizes the effect of moving one object `o` on another object `ox` only in cases where this effect is a controlled one (see section 6.3 for details).

- If `o` is a closed container and `ox` is inside, then `ox` remains inside, however `o` is moved.
- If `o` is carried upright and either
  - `o` is an upright open container and `ox` is inside; or
  - `ox` is a lid, placed on box `o`; or
  - `o` is a box with a lid and `ox` is inside the box with lid.

  then this relation will be preserved.

Otherwise, we dichotomize the space of effects very crudely. If `ox` is not *possibly moved by* `o`, then a motion of `o` has no effect on `ox`. If `ox` is possibly moved by `o`, and does not fall into one of the above categories, then in the current theory the effect on `ox` of moving `o` is entirely unconstrained. We do not give any further conditions for this relation; those can be specified, either in a problem statement or in specialized axioms. For instance, if object `o` is sitting on a stable table but is otherwise isolated, then no other objects are possibly moved by it.

If no objects are being manipulated and no objects are falling, then no object other than the agent moves.

These conditions on motion are not, of course, true in general in the real world. However, it is possible for these axiom to be consistent with a model in which the objects other than the agent follow Newtonian mechanics by positing that the agent cautiously avoids moving objects in ways that would cause them to violate the axioms here Since there is no general comprehension axioms on the actions of an agent, such a restriction is consistent with our theory.

2 Logic, Sorts, Notation

We use a simple sorted first-order logic with equality.

Sorts are in *italics*. Other symbols are in `typewriter font.`

Lower case symbols are variables. Symbols beginning with upper case are constants, functions, and predicates. Free variables are implicitly universally quantified, with a scope of the entire formula.

The precedence of Boolean operators is: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$. The quantifiers $\exists$, $\forall$, and $\exists^1$ (unique existence) have scope until the end of the formula or close bracket of larger scope.

The entities in the universe are partitioned into sorts: Each entity is of exactly one sort. There are six sorts: *Time*, *Region*, *History, Object*, *ObjectSet* and *Event.* For each sort there is a corresponding unary predicate, written in typewriter font; for example, the predicate `Time(t)` corresponds to the sort *Time.*

We use italicized sortal symbols in two contexts. The first use is for restricted quantification. A quantified variable can be restricted to a sort, with the standard meanings: If μ is a variable, α is a sortal symbol and $\phi(\mu)$ is a forumla, then
$\forall_{\mu:\alpha}\, \phi(\mu)$ is equivalent to $\forall_\mu\, \alpha(\mu) \Rightarrow \phi(\mu)$; and
$\exists_{\mu:\alpha}\, \phi(\mu)$ is equivalent to $\forall_\mu\, \alpha(\mu) \wedge \phi(\mu)$.
For example
$\forall_{\mathtt{u,v}:\,Time}$ `Leq(u,v)` $\Leftrightarrow$ `Lt(u,v)` $\vee$ `u=v`
is equivalent to
$\forall_{\mathtt{u,v}}$ `Time(u)` $\wedge$ `Time(v)` $\Rightarrow$
`[Leq(u,v)` $\Leftrightarrow$ `Lt(u,v)` $\vee$ `u=v].`

The second use of sortal symbols is in the declaration of non-logical symbols. Every non-logical symbol is introduced with a declaration of the sorts of its arguments and values. In our theory, sorting of non-logical symbols is strict; every symbol except the equality and inequality signs is sorted and there is no overloading or polymorphism. Each such declaration implicitly expresses a sortal axiom governing the symbol. The syntax of declarations is modelled on the syntax of function declarations in programming languages such as Pascal or Ada. These declarations and axioms are of three types:

- **Constant symbols.** A constant symbol has a declaration of the form `Symbol` $\rightarrow$ *Sort.* The corresponding axiom states that the symbol is of the sort. For example, the declaration "`Ta`$\rightarrow$*Time*" corresponds to the axiom "`Time(Ta).`"

- **Predicate symbols**. A predicate symbol declaration declares a sort for each argument. The corresponding axiom asserts that the predicate holds on arguments only if the arguments are of the proper sorts. For instance, the declaration "`Continuous(ta,tb:`*Time*`; h:`*History*`)`" corresponds to the axiom
  "$\forall_{\mathtt{ta,tb,h}}$ `Continuous(ta,tb,h)` $\Rightarrow$ `Time(ta)` $\wedge$ `Time(tb)` $\wedge$ `History(h).`"

- **Function symbols.** A function symbol declaration declares the sorts of each argument and the sort of the result. The corresponding axiom asserts that if the arguments have the specified sorts, then the result has the specified sort. For example, the declaration

`Slice(t:`*Time*`, h:`*History*`)` $\rightarrow$ *Region* corresponds to the axiom

$\forall_{\mathtt{t,h}}$ `Time(t)` $\wedge$ `History(h)` $\Rightarrow$ `Region(t,h).`

Functions are all total over the space of arguments of the proper sort. (Presumably, the function is undefined if the sortal conditions on the arguments are not met, but we do not axiomatize that.)

Thus, the entire theory in the sorted logic can be translated into an equivalent theory in an unsorted logic — which, indeed, is exactly what we do in preprocessing this theory for input into SPASS.

In each section or subsection, we first declare the new formal symbols introduced, then enumerate the definitions, then enumerate the axioms. (The distinction between axioms and definitions is informal, but intuitively useful.) Axioms are numbered using four letters. The first two indicate the second and subsection; the third is 'D' or 'A' for definition or axiom; the fourth is just enumerative; these are not necessarily consecutive, if axioms have been deleted or rearranged. Some of the axioms are redundant; no attempt has been made to generate a minimal set of axioms (see discussion in main paper).

2.1 Sorts

We use the following sorts:

Time — An instant of time.
Region — A region of space.
History — A function from time to regions.
Object — A physical entity.
ObjectSet — A set of objects.
Event — An event or action.

A region is a non-empty, bounded, topologically regular sets of points. Additional well-behavedness conditions could certainly be added. Events take place over a duration of some non-zero length; we do not allow instantaneous events.

3 Time

3.1 Time Ordering

The time line is forward-branching. Forward branching correspond to an agent's choice between actions. Since time is forward branching, it is not totally ordered; but the times previous to any given time `z` are totally ordered (axiom T.I.A.E below).

Symbols:
`Lt(x,y:`*Time*`)` — Time `x` is earlier than time `y`.
`Leq(x,y:`*Time*`)` — Time `x` is earlier than or equal to time `y`.
`Ordered(x,y:`*Time*`).`
`Leq3(x,y,z:`*Time*`).` $\mathtt{x} \leq \mathtt{y} \leq \mathtt{z}$
`TimeIntervalOverlap(w,x,y,z).` The intervals `[w,x]` and `[y,z]` overlap;
that is, they have an interior point in common.

Definitions

T.I.D.A $\forall_{x,y:Time}$ `Leq(x,y)` $\Leftrightarrow$ `Lt(x,y)` $\vee$ `x=y`.

T.I.D.B $\forall_{x,y}$ `Ordered(x,y)` $\Leftrightarrow$ `[Lt(x,y)` $\vee$ `Lt(y,x)` $\vee$ `x=y]`.

T.I.D.C `Leq3(x,y,z)` $\Leftrightarrow$ `Leq(x,y)` $\wedge$ `Leq(y,z)`.

T.I.D.D `TimeIntervalOverlap(w,x,y,z)` $\Leftrightarrow$
$\exists_t$ `Lt(w,t)` $\wedge$ `Lt(t,x)` $\wedge$ `Lt(y,t)` $\wedge$ `Lt(t,z)`.

Axioms

T.I.A.A $\neg$(`Lt(x,y)` $\wedge$ `Lt(y,x)`). `Lt` is antisymmetric.

T.I.A.B `Lt(x,y)` $\wedge$ `Lt(y,z)` $\Rightarrow$ `Lt(y,z)`. `Lt` is transitive.

T.I.A.E `Lt(x,z)` $\wedge$ `Lt(y,z)` $\Rightarrow$ `Ordered(x,y)`.
Forward branching: The times earlier than `z` are totally ordered.

3.2 Events / Actions

`Occur(ta,tb:`*Time*: `e:`*Event*`)`.
Event `e` occurs starting at time `ta` and ending at time `tb`.
`Feasible(t:`*Time*: `e:`*Event*`)`.
It is possible to execute action `e` starting at time `t`.

Definition:

T.E.D.A `Feasible(ta,e)` $\Leftrightarrow$ $\exists_{tb}$ `Occurs(ta,tb,e)`.
If action `e` is feasible at time `ta`, then there is some branch of the time line in which `e` occurs.

Axiom:

T.E.A.A `Occurs(ta,tb,e)` $\Rightarrow$ `Lt(ta,tb)`.

4 Spatial Relations

4.1 Basic spatial relations

We use the RCC (Randell, Cui, and Cohn, 1992) binary spatial relations `P, C, O, DR, EC, OV`. We do not give a complete theory of these here; we only enumerate the definitions and axioms that we need. For a more complete discussion of the axiomatization of RCC, see (Pratt and Schoop, 1998; Pratt-Hartmann, 2007). The function `RUnion(u,v)` is the union of regions `u` and `v`.

`Symbols:`
`P(u,v:`*Region*`)` — Region `u` is a subset of `v`.
`C(u,v:`*Region*`)` — Regions `u` and `v` are in contact.
`O(u,v:`*Region*`)` — Regions `u` and `v` overlap.
`DR(u,v:`*Region*`)` — Regions `u` and `v` do not overlap.
`EC(u,v:`*Region*`)` — Regions `u` and `v` are externally connected.
`OV(u,v:`*Region*`)` — Regions `u` and `v` partially overlap.
`RUnion(u,v:`*Region*`)` $\rightarrow$ *Region*.

Definitions:

S.B.D.A $\texttt{O(u,v)} \Leftrightarrow \exists_{\texttt{z}}\ \texttt{P(z,u)} \wedge \texttt{P(z,v)}.$

S.B.D.B $\forall_{\texttt{u,v}:Region}\ \texttt{DR(u,v)} \Leftrightarrow \neg\texttt{O(u,v)}.$

S.B.D.C $\texttt{EC(u,v)} \Leftrightarrow \texttt{DR(u,v)} \wedge \texttt{C(u,v)}.$

S.B.D.D $\forall_{\texttt{u,v}:Region}\ \texttt{DC(u,v)} \Leftrightarrow \neg\texttt{C(u,v)}.$

S.B.D.E $\forall_{\texttt{u,v,w}:Region}\ \texttt{w = RUnion(u,v)} \Leftrightarrow$
$\texttt{P(u,w)} \wedge \texttt{P(v,w)} \wedge$
$[\forall_{\texttt{z}}\ \texttt{P(u,z)} \wedge \texttt{P(v,z)} \Rightarrow \texttt{P(w,z)}].$
The union of `u` and `v` is the minimal set containing both `u` and `v`.

S.B.D.F $\texttt{OV(u,v)} \Leftrightarrow \texttt{O(u,v)} \wedge \neg\texttt{P(u,v)} \wedge \neg\texttt{P(v,u)}.$

Axioms:

S.B.A.A $\texttt{P(u,v)} \wedge \texttt{P(v,u)} \Rightarrow \texttt{u=v}.$

S.B.A.B $\texttt{P(u,v)} \wedge \texttt{P(v,w)} \Rightarrow \texttt{P(u,w)}.$

S.B.A.C $\forall_{\texttt{u}:Region}\ \texttt{P(u,u)}.$

S.B.A.D $\forall_{\texttt{u}:Region}\ \texttt{C(u,u)}.$

S.B.A.E $\texttt{C(u,v)} \Rightarrow \texttt{C(v,u)}.$

S.B.A.F $\texttt{C(u,v)} \wedge \texttt{P(v,w)} \Rightarrow \texttt{C(u,w)}.$

S.B.A.G $\texttt{O(u,v)} \Rightarrow \texttt{C(u,v)}.$

S.B.A.H $\forall_{\texttt{u,v,w}:Region}\ \texttt{O(w,RUnion(v,u))} \Rightarrow \texttt{O(w,v)} \vee \texttt{O(w,u)}.$

S.B.A.I $\texttt{P(u,v)} \wedge \texttt{DC(v,w)} \Rightarrow \texttt{DC(u,w)}.$

4.2 Same Vertical

The predicate `SameVertical(ra,rb)` means that region `ra` is congruent to region `rb` and that the mapping from one to another does not involve a rotation of the vertical axis, though it may involve a rotation around the vertical axis. This is an equivalence relation. Two regions that are `SameVertical` have corresponding parts, also `SameVertical`.

Definition:
Axioms:

S.V.A.A $\forall_{\texttt{ra}:Region}\ \texttt{SameVertical(ra,ra)}.$

S.V.A.B $\texttt{SamVVertical(ra,rb)} \Rightarrow \texttt{SameVertical(rb,ra)}.$

S.V.A.C $\texttt{SameVertical(ra,rb)} \wedge \texttt{SameVertical(rb,rc)} \Rightarrow \texttt{SameVertical(ra,rc)}.$

S.V.A.C $\texttt{SameVertical(ra,rb)} \wedge \texttt{P(rc,ra)} \Rightarrow$
$\exists_{\texttt{rd}}\ \texttt{P(rd,rb)} \wedge \texttt{SameVertical(rd,rc)}.$

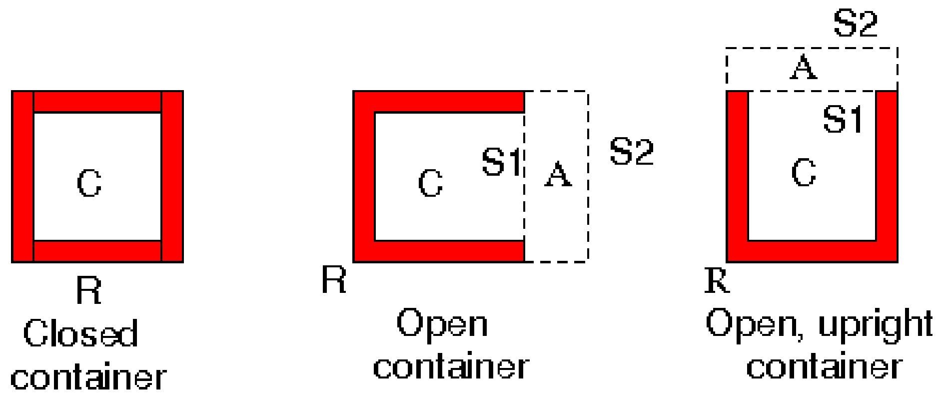


Figure 1: Closed, open, and open upright containers

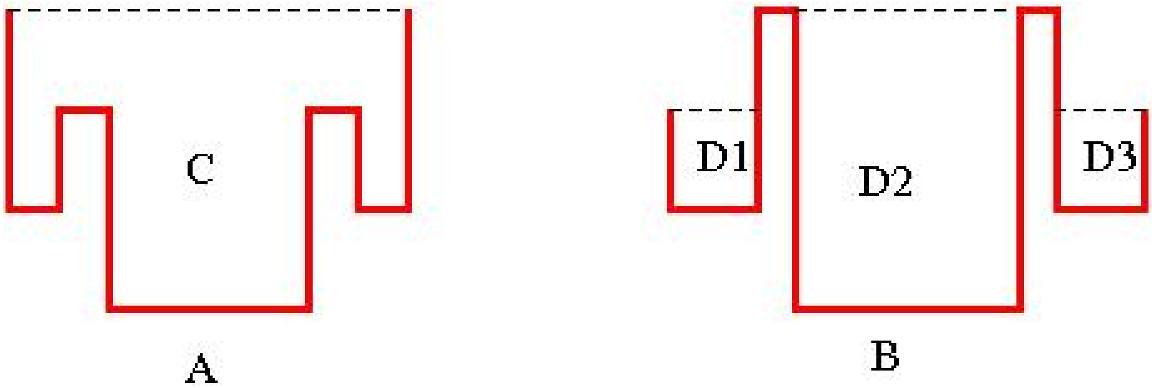
4.3 Spatial Containment

For convenience, we define symbols for what are essentially the same containment relations applying to one region containing another; to an object or set of objects containing a region (section 5.3); or to one object or set of objects containing another (section 5.4).

> Region `R` is a *closed container* for cavity `C` if `C` is an interior-connected, bounded component of the complement of `R` (figure 1).
>
> Region `R` is an *open container* for cavity `C` (a region) if there exists a region `A`, between two parallel planar surfaces `S1` and `S2` such that:
>
> - `A` and `R` do not overlap. The intersection where they meet $\mathtt{R} \cap \mathtt{A}$ is equal to the ring around `A` separating `S1` and `S2`: $\mathtt{R} \cap \mathtt{A} = \mathrm{Bd}(\mathtt{A}) \setminus (\mathtt{S1} \cap \mathtt{S2})$.
> - `C` is a cavity of the union $\mathtt{R} \cup \mathtt{A}$, but is not a cavity of either `R` or of `A` separately.
>
> Region `R` is an *upright open container* for cavity `C` if the planar surfaces `S1` and `S2` associated with `A` are horizontal and `A` is above `C`.
>
> Region `R` is a *simple upright open container* with cavity `C` if `C` is the unique maximal interior with respect to which `R` is an upright open container (figure 2).

The definition of closed container is purely topological, and therefore is expressible in our qualitative spatial language. However, expressing the conditions that the surfaces `S1` and `S2` are planar and parallel would require a much more powerful geometric theory than we are undertaking here.

Symbols:
`IntConn(r:`*Region*`)`. – Region `r` is interior connected.
`FaceConn(u,v:`*Region*`)` — Regions `u` and `v` are connected on a face.
`Cavity(u,v:`*Region*`)` — Region `u` is an interior cavity of `v`.
`Outside(u,v:`*Region*`)` — Region `u` is outside region `v`.
(`u` is a subset of the unbounded connected component of the complement of `v`).
`Contained(u,v:`*Region*`)`. Region `u` is is inside a cavity in `v`.
`CombinedContainer(ra,rb,rc:`*Region*`)`.
Region `rc` is an interior cavity of $\mathtt{ra} \cup \mathtt{rb}$.
`OpenContainerShape(rb,rc:`*Region*`)`. Region `rb` is an open container with interior `rc`.
`UprightContainerShape(rb,rc:`*Region*`)`.



A is a simple upright container: C is the unique maximal region contained.
B is an upright container that is not simple; D1, D2, D3 are each maximal contained regions.

Figure 2: Closed, open, and open upright containers

Region `rb` is an upright open container with interior `rc`.
`SimpleUprightContainerShape(rb,rc)`.
`OpenContained(ra,rb)` — Region `ra` is in the open container `rb`.
`OutsideContainer(ro,rb)` — Region `ro` is outside the container `rb`.

Definitions:

S.C.D.A $\texttt{FaceConn(u,v)} \Leftrightarrow \texttt{EC(u,v)} \land \texttt{IntConn(RUnion(u,v))}$.

S.C.D.B $\texttt{Cavity(u,v)} \Leftrightarrow$
$\texttt{IntConn(u)} \land \texttt{DR(u,v)} \land$
$\forall_{\texttt{r,rx}}\ \texttt{IntConn(r)} \land \texttt{O(r,u)} \land \texttt{P(rx,r)} \land \texttt{DR(rx,u)} \Rightarrow$
$\texttt{O(r,v)}$.
Region `u` is a cavity of `v` if it is a maximal interior-connected region disjoint from `v`. (Note that the outside of `v` does not satisfy this condition, since `u` must be a region and by definition a region is bounded.)

S.C.D.C $\texttt{Outside(u,v)} \Leftrightarrow [\texttt{DR(u,v)} \land [\forall_{\texttt{w}}\ \texttt{Cavity(w,v)} \Rightarrow \texttt{DR(u,w)}]]$.
Region `u` is outside `v` if `u` is disjoint from `v` and from every cavity of `u`.

S.C.D.D $\texttt{Contained(u,v)} \Leftrightarrow \exists_{\texttt{c}}\ \texttt{Cavity(c,v)} \land \texttt{P(u,c)}$.
Region `u` is contained in `v` if `u` is part of a cavity of `v`.

S.C.D.E $\texttt{CombinedContainer(ra,rb,rc)} \Leftrightarrow$
$\texttt{EC(ra,rb)} \land \texttt{Cavity(rc,RUnion(ra,rb))} \land \neg\texttt{Cavity(rc,ra)} \land \neg\texttt{Cavity(rc,rb)}$.

S.C.D.F $\texttt{SimpleUprightContainerShape(rb,rc)} \Leftrightarrow$
$\texttt{UprightContainerShape(rb,rc)} \land$
$\forall_{\texttt{rd}}\ \texttt{UprightContainerShape(rb,rd)} \Rightarrow \texttt{P(rd,rc)}$.

S.C.D.G $\texttt{OpenContained(ra,rb)} \Leftrightarrow$
$\exists_{\texttt{rc}}\ \texttt{OpenContainerShape(rb,rc)} \land \texttt{P(ra,rc)}$.

S.C.D.H $\texttt{OutsideContainer(ro,rb)} \Leftrightarrow$
$\texttt{DC(ro,rb)} \land \forall_{\texttt{rc}}\ \texttt{OpenContainerShape(rb,rc)} \Rightarrow \texttt{DC(ro,rc)}$.

Axioms:

S.C.A.A `Contained(u,v)` $\wedge$ `Contained(v,w)` $\Rightarrow$ `Contained(u,w)`.

S.C.A.C `UprightContainerShape(rb,rc)` $\Rightarrow$ `OpenContainerShape(rb,rc)`.

S.C.A.E `UprightContainerShape(rb,rc)` $\wedge$ `SameVertical(ra,rb)` $\Rightarrow$
$\exists_{\texttt{rd}}$ `UprightContainerShape(ra,rd)`

4.4 Much Smaller

We include a qualitative comparator on the size of regions: `MuchSmaller(ra,rb)`, meaning that `ra` is much smaller than `rb`. This comparator on region is related to the predicate `SmallSet(s,r)` (section 5.6) which in turn is used in some specialized physical axioms (e.g. A.C.A.A, section 9).

The axioms state that `MuchSmaller` is a partial ordering (S.M.A.A, .B); compatible with the part relation `P` (S.M.A.C); and invariant under the relation `SameVertical` (S.M.A.F). Also a small region cannot contain a larger region, under any kind of "containment". These axioms, and further properties stated below, are satisfied if `MuchSmaller(ra,rb)` is defined as holding if a sphere circumscribing `ra` would fit inside `rb`.

Symbols:
`MuchSmaller(ra,rb:`*Region*`)`.

Axioms:

S.M.A.A $\neg$`MuchSmaller(ra,ra)`.

S.M.A.B `MuchSmaller(ra,rb)` $\wedge$ `MuchSmaller(rb,rc)` $\Rightarrow$ `MuchSmaller(ra,rc)`.

S.M.A.C `MuchSmaller(ra,rb)` $\wedge$ `P(rc,ra)` $\wedge$ `P(rb,rd)` $\Rightarrow$
`MuchSmaller(rc,rd)`.

S.M.A.D `MuchSmaller(ra,rb)` $\Rightarrow$ $\neg$`Cavity(rb,ra)` $\wedge$ $\neg$`OpenContainerShape(ra,rb)`.

S.M.A.E `MuchSmaller(ra,rb)` $\Rightarrow$ $\exists_{\texttt{rc}}$ `SameVertical(rc,ra)` $\wedge$ `P(rc,rb)`.

S.M.A.F `MuchSmaller(ra,rb)` $\wedge$ `SameVertical(ra,rc)` $\wedge$ `SameVertical(rb,rd)` $\Rightarrow$
`MuchSmaller(rc,rd)`.

5 Objects

The theory of objects introduces two sorts *Object*, and *ObjectSet*. Objects are disjoint; they do not overlap, and one object is not part of another.

5.1 Object Sets

The relations over object sets and their definitions are standard. The sole axiom O.S.A.A is the axiom of extension, that two sets with the same elements are equal. (The usual ZF axiom of pairing is implicit in the fact that `Pair` is a function.)

Symbols:
`Element(x:`*Object*`; s:`*ObjectSet*`).` — *Object* `x` is an element of *ObjectSet* `s`.
`Null` $\rightarrow$ *ObjectSet*.
`Singleton(x:`*Object*`)` $\rightarrow$ *ObjectSet*. — $\{x\}$
`Pair(x,y:`*Object*`)` $\rightarrow$ *ObjectSet*. — $\{x, y\}$
`Subset(sa,sb:`*ObjectSet*`).`
`Disjoint(sa,sb:`*ObjectSet*`).`
`Union(sa,sb:`*ObjectSet*`)` $\rightarrow$ *ObjectSet*.

Definitions:

O.S.D.A $\forall_{x}$ $\neg$`Element(x,Null).`

O.S.D.B $\forall_{x,y:Object}$ `Element(y,Singleton(x))` $\Leftrightarrow$ `y=x.`

O.S.D.C $\forall_{sa,sb:ObjectSet}$ `Subset(sa,sb)` $\Leftrightarrow$ $\forall_{o}$ `Element(o,sa)` $\Rightarrow$ `Element(o,sb).`

O.S.D.D $\forall_{x,y:Object,z}$ `Element(z,Pair(x,y))` $\Leftrightarrow$ `z=x` $\vee$ `z=y.`

O.S.D.E $\forall_{sa,sb:ObjectSet}$ `Disjoint(sa,sb)` $\Leftrightarrow$ $\neg$ $\exists_{o}$ `Element(o,sa)` $\wedge$ `Element(o,sb).`

O.S.D.F $\forall_{sa,sb:ObjectSet;x:Object}$ `Element(x,Union(sa,sb))` $\Leftrightarrow$ `Element(x,sa)` $\vee$ `Element(x,sb).`

Axiom

O.S.A.A $\forall_{sa,sb:ObjectSet}$ [$\forall_{x}$ `Element(x,sa)` $\Leftrightarrow$ `Element(x,sb)`] $\Rightarrow$ `sa=sb.`

5.2 Objects and Object Sets: Spatio-Temporal

We next define the primitives that relate objects to the regions they occupy at a given time. The function `Place(t,o)` is the region the object `o` occupies at time `t`. The predicate `FeasiblePlace(o,r)` holds if it is physically possible to configure `o` so that it occupies `r`. The predicate `OSPlace(t,s,r)` holds if `r` is the region occupied by object set `s` at time `t`. (This is a predicate rather than a function, since the null set does not occupy any region.) `Contents(t,r)` is the set of objects that are in region `r` at time `t`.

Symbols:
`Place(t:`*Time*`; o:`*Object*`)` $\rightarrow$ *Region*.
`FeasiblePlace(o:`*Object*`; r:`*Region*`).`
`OSPlace(t:`*Time*`; s:`*ObjectSet*`; r:`*Region*`).`
`Contents(t:`*Time*`; r:`*Region*`)` $\rightarrow$ *ObjectSet*.

Axioms

O.T.A.A $\forall_{t:Time;o:Object}$ `FeasiblePlace(o,Place(t,o)).`
Every object always occupies a feasible place.

O.T.A.B $\forall_{p,q:Object;t:Time}$ `p` $\neq$ `q` $\Rightarrow$ `DR(Place(t,p), Place(t,q)).`
Any two objects are spatially disjoint.

O.T.A.C $\neg\exists_{t,r}$ `OSPlace(t,Null,r).`
The null set has no place.

O.T.A.D $\forall_{\texttt{s}:ObjectSet;\,\texttt{t}:Time}$ `s` $\neq$ `Null` $\Rightarrow$ $\exists^1_{\texttt{r}}$ `OSPlace(s,t,r)`.
Every non-empty set of objects occupies a unique region at any time.

O.T.A.E `OSPlace(t,s,r)` $\Rightarrow$
[$\forall_{\texttt{o}}$ `Element(o,s)` $\Rightarrow$ `P(Place(t,o),r)`] $\wedge$
[$\forall_{\texttt{ra}}$ [$\forall_{\texttt{o}}$ `Element(o,s)` $\Rightarrow$ `P(Place(t,o),ra)`] $\Rightarrow$ `P(r,ra)`.].
The region occupied by a set `s` is the minimal region that contains all the regions occupied by the elements of `s`.

O.T.A.F $\forall_{\texttt{o}:Object;\texttt{r}:Region;\texttt{t}:Time}$ `Element(o,Contents(t,r))` $\iff$ `P(Place(t,o),r)`.

O.T.A.G `FeasiblePlace(o,r)` $\wedge$ `SameVertical(ra,r)` $\Rightarrow$ `FeasiblePlace(o,ra)`.
`FeasiblePlace` is invariant under `SameVertical`.

5.3 Objects containing regions

We here define the containment relations between a container, which is an object or a set of objects and a region contained. Here and in section 5.4, we define closed containers in terms of a set of objects but open containers in terms of a single object, because closed containers are often composed of multiple objects (e.g. a box with a lid; a bottle with a cap; and so on) where as this is much rarer for open containers, though it does occur (e.g. cupping your two hands.)

Symbols:
`ClosedContainer(t:`*Time*`; s:`*ObjectSet*`; rc:`*Region*`)`.
`OpenContainer(t:`*Time*`; o:`*Object*`; rc:`*Region*`)`.
`UprightContainer(t:`*Time*`; o:`*Object*`; rc:`*Region*`)`.
`SimpleUprightContainer(t:`*Time*`; o:`*Object*`; rc:`*Region*`)`.

Definitions:

O.R.D.A `ClosedContainer(t,s,rc)` $\Leftrightarrow$
`Time(t)` $\wedge$ `ObjectSet(s)` $\wedge$ $\exists_{\texttt{rc}}$ `OSPlace(t,s,rs)` $\wedge$ `Cavity(rc,rs)`.

Note: A cup upside down inside a box is both an object inside a closed container and part of a closed container. A box with shelves therefore forms $n(n-1)/2$ closed containers (any pair of shelves/top/bottom determine a container) and a box with small cubby holes and dividers in two directions forms an exponential number (any interior-connected collection of cubby holes is considered a closed container) but that's the way it goes.

O.R.D.B `OpenContainer(t,o,rc)` $\Leftrightarrow$
`Time(t)` $\wedge$ `Object(o)` $\wedge$ `OpenContainerShape(Place(t,o),rc)`.

O.R.D.C `UprightContainer(t,o,rc)` $\Leftrightarrow$
`Time(t)` $\wedge$ `Object(o)` $\wedge$ `UprightContainerShape(Place(t,o),rc)`.

O.R.D.D `SimpleUprightContainer(t,o,rc)` $\Leftrightarrow$ `SimpleUprightContainerShape(Place(t,o),rc)`.

5.4 Object Containment

Same relations again, for one object or a set of objects containing another object.

Symbols:
`CContained(t:`*Time*`; ox:`*Object*`; s:`*ObjectSet*`)`.

OContained(t:*Time*; ox,ob:*Object*).
UContained(t:*Time*; ox,ob:*Object*).
CContents(t:*Time*; s:*ObjectSet*) $\rightarrow$ *ObjectSet*.
UContents(t:*Time*; ob:*Object*) $\rightarrow$ *ObjectSet*.
Outside(t:*Time*; ox:*Object*).
Empty(t:*Time*; ox:*Object*).

Definitions:

O.C.D.A CContained(t,ox,s) $\Leftrightarrow$
$\exists_{\texttt{rc}}$ ClosedContainer(t,s,rc) $\wedge$ Object(ox) $\wedge$ P(Place(t,ox),rc).

O.C.D.B OContained(t,ox,ob) $\Leftrightarrow$
$\exists_{\texttt{rc}}$ OpenContainer(t,ob,rc) $\wedge$ Object(ox) $\wedge$ P(Place(t,ox),rc).

O.C.D.C UContained(t,ox,ob) $\Leftrightarrow$
$\exists_{\texttt{rc}}$ UprightContainer(t,ob,rc) $\wedge$ Object(ox) $\wedge$ P(Place(t,ox),rc).

O.C.D.D $\forall_{\texttt{t}:Time;\ \texttt{ob}:Object;\ \texttt{s}:ObjectSet}$
s = CContents(t,ob) $\Leftrightarrow$
$\forall_{\texttt{ox}}$ Element(ox,s) $\Leftrightarrow$ CContained(t,ox,ob).

O.C.D.E $\forall_{\texttt{t}:Time;\ \texttt{ob}:Object;\ \texttt{s}:ObjectSet}$
s = UContents(t,ob) $\Leftrightarrow$
$\forall_{\texttt{ox}}$ Element(ox,s) $\Leftrightarrow$ UContained(t,ox,ob).

O.C.D.F $\forall_{\texttt{t}:Time;\texttt{r}:Region}$ Empty(t,r) $\Leftrightarrow$
$\neg\exists_{\texttt{o}:Object}$ O(Place(t,o),r).

5.5 Box with lid

The intended meaning of a box with a lid is a pair of object that form a closed container, where the lid is stably placed on the box, so that, if you move the box, the lid will follow along. We do not fully axiomatize the conditions necessary for this, which are complex, and involve both geometric and physical properties of the box and the lid. Rather, we present it as a primitive, give some necessary but hardly sufficient conditions, and use it as a primitive in some further causal axioms characterizing actions.

Symbols:
BoxWithLidShape(rb,rl,rc:*Region*).
Regions rb,rl,rc geometrically could be a box, lid, and cavity. Note that rb is not necessarily an open container with inside rc; the lid can arch over the box and enclose more space.
BoxWithLid(t:*Time*; ob,ol:*Object*).
Objects ob,ol physically form a box with lid (thus, ol will move along when ob is moved.)
BoxWithLidC(t:*Time*; ob,ol:*Object*; rc:*Region*).
Object ob and ol are a box with a lid and contain region rc.
LContents(t:*Time*; ob,ol:*Object*; s:*ObjectSet*).
Objects ob and ol form a box with lid with contents s.

Definition:

O.L.D.A BoxWithLidC(t,ob,ol,c) $\Leftrightarrow$
BoxWithLid(t,ob,ol) $\wedge$ BoxWithLidShape(Place(t,ob), Place(t,ol), rc).

O.L.D.A `LContents(t,ob,ol,s)` $\Leftrightarrow \exists_{\texttt{rc}}$ `BoxWithLidC(t,ob,ol,rc)` $\wedge$ `s=Contents(t,rc)`.

Axioms:

O.L.A.A `BoxWithLidShape(rb,rl,rc)` $\Rightarrow$
`CombinedContainer(rb,rl,rc)` $\wedge \neg$`MuchSmaller(rb,rl)`.

O.L.A.B `BoxWithLid(t,ob,ol)` $\Rightarrow \exists_{\texttt{rc}}$ `BoxWithLidC(Place(t,ob),Place(t,ol),rc)`.

5.6 Fits and Small Set

The final spatial relation between objects and regions we axiomatize is `Fits(s,r)` — object set `s` fits into region `r` and `SmallSet(s,r)` — object set `s` is collectively small as compared to region `r`. The axioms are self-explanatory.

Likewise we consider a class of *small objects*. There are objects that are much smaller than the agent, and therefore particularly easy to move.

Symbol:
`Fits(s:`*ObjectSet*`; r:`*Region*`)`.
`SmallSet(s:`*ObjectSet*`; r:`*Region*`)`.
`SmallObject(o:`*Object*`)`.

Definition:

O.F.D.A `SmallObject(o)` $\Leftrightarrow$
$\forall_{\texttt{ra,rb}}$ `FeasiblePlace(ra,Agent)` $\wedge$ `FeasiblePlace(rb,o)` $\Rightarrow$ `MuchSmaller(rb,ra)`.

Axioms:

O.F.A.A $\forall_{\texttt{r}:Region}$ `Fits(Null,r)`.

O.F.A.B $\forall_{\texttt{o}:Object;\texttt{r}:Region}$ `FeasiblePlace(o,r)` $\Rightarrow$ `Fits(Singleton(o),r)`.

O.F.A.C `OSPlace(t,s,r)` $\Rightarrow$ `Fits(s,r)`.

O.F.A.D `Fits(s,ra)` $\wedge$ `P(ra,rb)` $\Rightarrow$ `Fits(s,rb)`.

O.F.A.E $\forall_{\texttt{r}:Region}$ `SmallSet(Null,r)`.

O.F.A.F `SmallSet(s,r)` $\wedge$ `Subset(sa,s)` $\Rightarrow$ `SmallSet(sa,r)`.

O.F.A.G `SmallSet(s,r)` $\wedge$ `OSPlace(t,s,ra)` $\Rightarrow$ `MuchSmaller(ra,r)`.

O.F.A.H `SmallSet(s,r)` $\Rightarrow$ `Fits(s,r)`.

O.F.A.I `Fits(s,ra)` $\wedge$ `SameVertical(ra,rb)` $\Rightarrow$ `Fits(s,rb)`.

6 Motion and Manipulation

In this section we present the core of our physical theory, which is the theory of an agent manipulating an object.

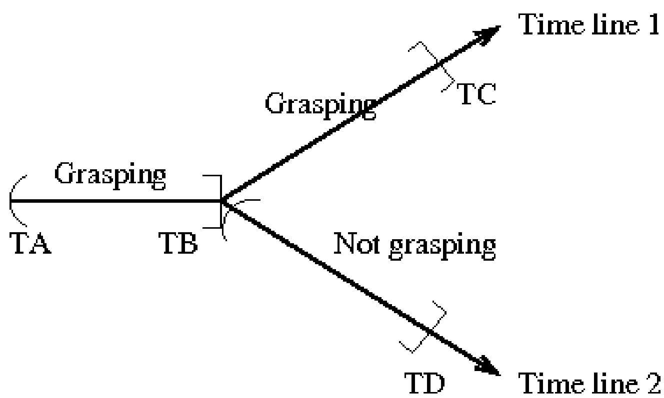


Figure 3: Grasping on a dense branching time line

6.1 Grasping an Object

We start with a very limited theory of grasping an object. There are two basic primitives here: the constant `Agent`, the hero agent who is a distinguished object, and the predicate `Grasp(t,o)` meaning that the agent is grasping object `o` at time `t`. We define some further predicates for convenient reference.

By convention we suppose that, on any time line, the agent grasps any given object over a time interval that is open on the left and closed on the right. Thus, if the agent grasps `o` from time `ta` to `tb` and then releases it, he is grasping `o` at time `tb` and is not grasping it over some closed interval (`tb`, `tc`]. For instance, in the branching structure shown in figure 3, on time line 1, the agent grasps `O` from `TA` to `TC`; on time line 2, the agent grasps O from `TA` to `TB` and then is not grasping at all times after `TB` up to `TD`.

We implicitly include in the concept `CanGrasp` that the agent `CanGrasp` object `o` only if he can move it in some way that is consistent with our restricted theory of motion; that is, he can move it without pushing other objects out of the way and without removing a support for other objects, thus causing them to fall over (either of which would be inconsistent with frame axiom M.R.A.A).

Symbols:
`Agent` → *Object*.
`Grasp(t:`*Time*`; o:`*Object*`).`
`EmptyHanded(t:`*Time*`).`
`Grasps(ta,tb:`*Time*`; o:`*Object:*`).`
`CanGrasp(t:`*Time*`; o:`*Object*`).` The agent can grasp object `o` at time `t`.
`Released(ta,tb:`*Time*`; o:`*Object:*`).`

Definitions:

M.G.D.A $\texttt{EmptyHanded(t)} \Leftrightarrow \neg\exists_{\texttt{o}}\ \texttt{Grasp(t,o)}$.
The agent is `EmptyHanded` at time `t` if he is not grasping anything at `t`.

M.G.D.B $\texttt{Grasps(ta,tb,o)} \Leftrightarrow$
$\texttt{Lt(ta,tb)} \wedge \forall_{\texttt{t}}\ \texttt{Lt(ta,t)} \wedge \texttt{Leq(t,tb)} \Rightarrow \texttt{Grasp(t,o)}$.
The agent grasps object `o` from time `ta` (non-inclusive) to time `tb` (inclusive).

M.G.D.C $\texttt{CanGrasp(t,o)} \Leftrightarrow \exists_{\texttt{tb}}\ \texttt{Grasps(t,tb,o)}$.

M.G.D.D `Released(ta,tb,o)` $\Leftrightarrow$
`Lt(ta,tb)` $\wedge$ $\forall_{\texttt{t}}$ `Lt(ta,t)` $\wedge$ `Leq(t,tb)` $\Rightarrow$ $\neg$`Grasp(t,o)`.
Object `o` is released (i.e. the agent is not grasping it) from time `ta` (not inclusive) through `tb` (inclusive).

Axioms:

M.G.A.A `Grasp(t,oa)` $\wedge$ `Grasp(t,ob)` $\Rightarrow$ `oa=ob`.
The agent can only grasp one object at a time.

M.G.A.B `[Grasp(t,o)` $\vee$ `CanGrasp(t,o)]` $\Rightarrow$ `FaceConn(Place(t,Agent),Place(t,o))`.
The agent can only grasp `o` if he has contact with `o` along an extended face.

M.G.A.C $\forall_{\texttt{ta}:\mathit{Time};\texttt{o}:\mathit{Object}}$ $\exists_{\texttt{tb}}$ `Released(ta,tb,o)`.
At any time `ta` it is possible for the agent to release object `o` (assuming that he is holding `o`).

M.G.A.D $\neg$`Grasp(t,Agent)`.
The agent does not grasp himself.

6.2 Motion

We next characterize motion. Agents move, objects move when they are carried, directly or indirectly, and objects fall when released in an unstable position.

We do not give any geometric characterization of stability.

Symbols:
`Motionless(ta,tb:`*Time*`; o:`*Object:*`)`.
`TravelTo(r:`*Region*`)` $\rightarrow$ *Event*.
The event of the agent traveling empty-handed to region `r`.
`StandStill` $\rightarrow$ *Event.*
The event of the agent standing still, not grasping anything.
`MoveTo(o:`*Object*`; r:`*Region*`)` $\rightarrow$ *Event.*
The event of the agent directly manipulating object `o` so as to move it to region `r`.
`Stable(t:`*Time*`; o:`*Object*`)`.
At time `t`, object `o` is in a position where it will be stable, if released.
`Falling(ta,tb:`*Time*`; o:`*Object*`)`.
Object `o` is falling from time `ta` to time `tb`.
`AllStable(t:`*Time*`)`.
All objects are either grasped or stable at time `t`.

Definitions:

M.O.D.D `Motionless(ta,tb,o)` $\Leftrightarrow$
`Lt(ta,tb)` $\wedge$ $\forall_{\texttt{t}}$ `Lt(ta,t)` $\wedge$ `Leq(t,tb)` $\Rightarrow$ `Place(t,o) = Place(ta,o)`.

M.O.D.E `Occurs(ta,tb,TravelTo(r))` $\Leftrightarrow$
`r = Place(tb,Agent)` $\wedge$ $\forall_{\texttt{o}:\mathit{Object}}$ `Released(ta,tb,o)`.

M.O.D.F `Occurs(ta,tb,StandStill)` $\Leftrightarrow$
`Motionless(ta,tb,Agent)`$\wedge$ $\forall_{\texttt{o}:\mathit{Object}}$ `Released(ta,tb,o)`.

M.O.D.G `Occurs(ta,tb,MoveTo(o,r))` $\Leftrightarrow$
`r = Place(tb,o)` $\wedge$ `Grasps(ta,tb,o)`.

M.O.D.J `AllStable(t)` $\Leftrightarrow \forall_{\texttt{o}:Object}$ `o=Agent` $\vee$ `Stable(t,o)` $\vee$ `Grasp(t,o)`.

Axioms:

M.O.A.A `Falling(ta,tb,o)` $\Rightarrow$ `Lt(ta,tb)`.

M.O.A.B `Falling(ta,tb,o)` $\wedge$ `Leq(ta,tx)` $\wedge$ `Lt(tx,tb)` $\Rightarrow$
`Grasp(ta,o)` $\wedge$ `Released(ta,tb,o)` $\wedge$ $\neg$`Stable(tx,o)` $\wedge$ `Stable(tb,o)`.
Object `o` is falling from time `ta` to time `tb` only if `o` is grasped at time `ta` and released over the interval (`ta`, `tb`] and unstable over the interval [`ta`, `tb`), but stable at time `tb`. (This, of course, is an simplification in our microworld; it is not true in the world in general.)

M.O.A.C `Grasp(ta,o)` $\wedge$ `Released(ta,tb,o)` $\wedge$ $\neg$`Stable(ta,o)` $\Rightarrow$
$\exists_{\texttt{tf}}$ `Ordered(tf,tb)` $\wedge$ `Falling(ta,tf,o)`.

M.O.A.D $\neg$`Stable(t,o)` $\Rightarrow$
$\exists_{\texttt{ta,tf}}$ `Leq(ta,t)` $\wedge$ `Lt(t,tf)` $\wedge$ `Falling(ta,tf,o)`.
If `o` is unstable at time `t`, then it is falling over some interval [`ta`, `tf`) containing `t`.

M.O.A.E `BoxWithLid(t,ob,ol)` $\Rightarrow$ `Stable(t,ol)`.
A lid is stably supported by the box underneath (this is a necessary condition for `BoxWithLid`.)

6.3 Effect of moving one object on another

We here partially characterize the effect of moving one object on another.

- We say that object `ox` is *possibly moved by* object `o` at time `t` if there are small motions of `o` that will immediately cause `ox` to move (this condition is not axiomatized).
- We distinguish two cases in which a motion of `o` causes a controlled motion of `ox`.
  - If either `ox=o` or `o` is a closed container containing `ox` then that relation persists under any motion. In this case we say that `ox` *always goes with* `o` (definition M.E.D.A).
  - If either `o` is an upright container containing `ox` or `o` is a box with a lid `ox` or `o` is a bid with lid `ol` and `ox` is contained in the pair { `o,ol` }, then that relation persists if `o` is moved and kept upright. In this case we say that `ox` *goes with* `o` *under upright motions.* (definition M.E.D.B)
- We say that `o` can be *safely moved* if every object that can possibly be moved by `o` either always goes with `o` or goes with `o` under upright motions (definition M.E.D.E). A motion `m` is *safe* if either
  - All objects that can possibly be moved by `o` always go with `o`. In this case, `m` can be any motion.
  - All objects that can possibly be moved by `o` either always go with `o` or go with `o` under upright motions. In this case `m` is restricted to be an upright motion.

In the current state of our theory, we do not axiomatize the relation `PossiblyMovedBy`. As the theory is extended, necessary conditions and sufficient conditions can certainly be stated; for example, if `o` is supported by a stable table and is otherwise isolated, then it can be safely moved.

Symbols:
`PossiblyMovedBy(t:`*Time*`; ox,oc:`*Object*`)`.

`AlwaysGoesWith(t:`*Time*`;ox,oc:`*Object*`)`.
Object `ox` "goes with" object `oc` at time `t`.
`UprightThroughout(ta,tb:`*Time*`; o:`*Object*`)`.
`UprightGoesWith(t:`*Time*`;ox,oc:`*Object*`)`.
`UprightMoveTo(ta,tb:`*Time*`; o:`*Object*`; r:`*Region*`)` $\rightarrow$ *Event*.
`SafelyMovable(t:`*Time*`; o:`*Object*`)`.
`SafeMoveTo(ta,tb:`*Time*`; o:`*Object*`; r:`*Region*`)` $\rightarrow$ *Event*.

Definitions:

M.E.D.A $\texttt{AlwaysGoesWith(t,ox,oc)} \Leftrightarrow$
$\texttt{Object(oc)} \wedge [\texttt{ox = oc} \vee \texttt{CContained(t,ox,Singleton(oc))}]$.

M.E.D.B $\texttt{UprightGoesWith(t,ox,oc)} \Leftrightarrow$
$\texttt{UContains(t,ox,oc)} \vee$
$\exists_{\texttt{ol}}\ \texttt{BoxWithLid(t,ob,ol)} \wedge [\texttt{ox=ol} \vee \texttt{CContained(t,ox,Pair(ob,ol))}]$.

M.E.D.C $\forall_{\texttt{ta,tb}:\mathit{Time};\texttt{o}:\mathit{Object}}\ \texttt{UprightThroughout(ta,tb,o)} \Leftrightarrow$
$\forall_{\texttt{tm}}\ \texttt{Leq3(ta,tm,tb)} \Rightarrow \texttt{SameVertical(Place(ta,o),Place(tb,o))}$.

M.E.D.D $\texttt{Occurs(ta,tb,UprightMoveTo(ta,tb,o,r))} \Leftrightarrow$
$\texttt{Occurs(ta,tb,MoveTo(o,r))} \wedge \texttt{UprightThroughout(ta,tb,o)}$.

M.E.D.E $\forall_{\texttt{t}:\mathit{Time};\texttt{o}:\mathit{Object}}\ \texttt{SafelyMovable(t,o)} \Leftrightarrow$
$\forall_{\texttt{ox}}\ \texttt{PossiblyMovedBy(t,ox,o)} \Rightarrow$
$[\texttt{AlwaysGoesWith(t,ox,o)} \vee \texttt{UprightGoesWith(t,ox,o)}]$.

M.E.D.F $\texttt{Occurs(ta,tb,SafeMoveTo(o,r))} \Leftrightarrow$
$\texttt{Occurs(ta,tb,MoveTo(o,r))} \wedge$
$\forall_{\texttt{ox}}\ \texttt{PossiblyMovedBy(t,ox,o)} \Rightarrow$
$[\texttt{AlwaysGoesWith(t,ox,o)} \vee$
$[\texttt{UprightGoesWith(t,ox,o)} \wedge \texttt{UprightThroughout(ta,tb,o)}]]$.

Axiom

M.E.A.E $\texttt{AlwaysGoesWith(t,ox,oc)} \vee \texttt{UprightGoesWith(t,ox,oc)} \Rightarrow$
$\texttt{PossiblyMovedBy(t,ox,oc)}$.

6.4 Frame axioms

Here are frame axioms: necessary conditions for change over time. These are mostly in "explanation closure" form: a change in a time-dependent state can occur only under such and such circumstances. The most important of these, discussed in the main paper, is the frame axiom for change of position (M.R.A.A). Other explanation closure axioms delimit change in `BoxWithLid` (M.R.A.B); and `Stable`. Axiom M.R.A.E is a forward-direction inference for falling: A falling object in an upright container remains in the container.

Symbol:
`Moves(ta,tb:`*Time*`; o:`*Object*`)`.
Object `o` moves some time between times `ta` and `tb`.

Definition:

M.R.D.A $\texttt{Moves(ta,tb,o)} \Leftrightarrow$
$\exists_{\texttt{tm}}\ \texttt{Leq(ta,tm)} \wedge \texttt{Lt(tm,tb)} \wedge \texttt{Place(tm,o)} \neq \texttt{Place(tb,o)}$.

Axioms:

M.R.A.A `Moves(ta,tb,o)` $\Rightarrow$
`o = Agent` $\vee$
$\exists_{\texttt{tc,td,ox,rx}}$ `TimeIntervalOverlap(tc,td,ta,tb)` $\wedge$ `PossiblyMovedBy(tc,o,ox)` $\wedge$
`[Occurs(tc,td,MoveTo(ox,rx))` $\vee$ `Falling(tc,td,ox)]`.
Frame axiom for change of position: Object `o` moves only if (a) it is the agent; (b) it is "possibly moved by" object `ox`, which in turn either is directly moved by the agent or is falling.

M.R.A.B `Lt(ta,tb)` $\wedge$ `BoxWithLid(ta,ob,ol)` $\wedge$ $\neg$`BoxWithLid(tb,ob,ol)` $\Rightarrow$
`[[`$\exists_{\texttt{tc,td,ra}}$ `IntervalOverlap(tc,td,ta,tb)` $\wedge$ `Occurs(tc,td,MoveTo(ol,ra))]` $\vee$
`[`$\exists_{\texttt{tm}}$ `Lt(ta,tm)` $\wedge$ `Leq(tm,tb)` $\wedge$ $\neg$`SameVertical(Place(tm,ob),Place(ta,ob))]]` .
A `BoxWithLid` relation between `ob` and `ol` can cease only if the lid is taken off or if the box is tilted.

M.R.A.C `Lt(ta,tb)` $\wedge$ $\neg$`BoxWithLid(ta,ob,ol)` $\wedge$ `BoxWithLid(tb,ob,ol)` $\Rightarrow$
$\exists_{\texttt{tc,td,ra}}$ `IntervalOverlap(tc,td,ta,tb)` $\wedge$ `Occurs(tc,td,MoveTo(ol,ra))` .
A `BoxWithLid` relation between `ob` and `ol` can be created only by putting the lid on the box.

M.R.A.D `Lt(ta,tb)` $\wedge$ $\neg$`[Stable(ta,o)` $\Leftrightarrow$ `Stable(tb,o)]` $\Rightarrow$ `Moves(ta,tb,o)`.
The stability of object `o` changes only if `o` moves (not of course true in general in the real world; a simplification in our microworld.) one or the other is moved.

M.R.A.E `Falling(ta,tb,o)` $\wedge$ `UContained(ta,ox,ob)` $\wedge$ `Lt(ta,tm)` $\wedge$ `Leq(tm,tb)` $\Rightarrow$
`UContained(tm,ox,ob)`.

M.R.A.F `PossiblyMovedBy(ta,ox,oc)` $\wedge$ `[`$\forall_{\texttt{o}}$ `o` $\neq$ `Agent` $\Rightarrow$ `Place(ta,o) = Place(tb,o)]` $\Rightarrow$
`PossiblyMovedBy(tb,ox,oc)`.
The `PossiblyMovedBy` relation is determined by the positions of the objects other than the agents. That is, if all objects other than the agent are in the same place at `ta` and `tb`, then all `PossiblyMovedBy` relations are the same.

6.5 Feasibility of travelling

We give an incomplete theory of the feasibility of `TravelTo`.

The predicate `Trajectory(ra,rb,rw)` means that there is a feasible trajectory for the agent from `ra` to `rb` remaining in `rw`. We give some necessary conditions for this (axiom M.F.A.B) and some combinatorial axioms (M.F.A.D – .F).

Axiom M.F.A.G and .H give conditions for the feasibility of travelling that are necessary and sufficient if no other objects are moving.[2] M.F.A.G states that, if `TravelTo(rb)` occurs from `ta` to `tb`, then there exists a region `rw` such that `Trajectory(Place(ta,Agent),rb,rw)` and the agent stays in `rw` during `[ta,tb]`. M.F.A.H states that, if `Trajectory(Place(ta,Agent),rb,rw)` and `rw` is free of obstacles, then `TravelTo(rb)` is feasible at time `ta`.

Symbols:
`NoObstacles(t:`*Time*; `r:`*Region*`)`.
No objects other than the agent are inside region `r` at time `t`.
`Trajectory(ra,rb,rw:`*Region*`)`. Discussed above.
`MiddlePos(ta,tb:`*Time*: `o:`*Object*; `r:`*Region*`)`.
Object `o` occupies region `r` some time between times `ta` and `tb`.

[2] If other objects are falling, then it would be difficult to give either necessary or sufficient conditions, since an external object may either fall so as to block the path or fall so as to clear the path.

StaysIn(ta,tb: *Time*; o: *Object*; r: *Region*).
Object o remains inside r throughout the interval [ta,tb].

Graspable(t: *Time*; o: *Object*).
At time t, the agent can move so as to grasp o.

Definitions:

M.F.D.A NoObstacles(t,r) $\Leftrightarrow$
Time(t) $\wedge$ $\forall_{\texttt{o}:Object}$ o $\neq$ Agent $\Rightarrow$ DR(Place(t,o),r).

M.F.D.C MiddlePos(ta,tb,o,r) $\Leftrightarrow$
Object(o) $\wedge$ $\exists_{\texttt{tx}}$ Leq3(ta,tx,tb) $\wedge$ r=Place(tx,o).

M.F.D.D StaysIn(ta,tb,o,r) $\Leftrightarrow$
$\forall_{\texttt{rx}}$ MiddlePos(ta,tb,o,rx) $\Rightarrow$ P(rx,r).

M.F.D.E Graspable(t,o) $\Leftrightarrow$
$\exists_{\texttt{tb,ra}}$ Occurs(t,tb,TravelTo(ra)) $\wedge$ CanGrasp(tb,o).
Object o is graspable if the agent can travel to a place ra where he can grasp o.

Axioms:

M.F.A.A $\forall_{\texttt{ta,tb,o}}$ Lt(ta,tb) $\wedge$ Object(o) $\Rightarrow$ $\exists_{\texttt{rw}}$ StaysIn(ta,tb,o,rw).
In a more powerful spatio-temporal theory this would of course be a theorem rather than an axiom.

M.F.A.B Trajectory(ra,rb,rw) $\Rightarrow$
FeasiblePlace(Agent,ra) $\wedge$ FeasiblePlace(Agent,rb) $\wedge$
IntConn(rw) $\wedge$ P(ra,rw) $\wedge$ P(rb,rw).

M.F.A.C FeasiblePlace(Agent,ra) $\Rightarrow$ Trajectory(ra,ra,ra).

M.F.A.D Trajectory(ra,rb,rw) $\Rightarrow$ Trajectory(rb,ra,rw).

M.F.A.E Trajectory(ra,rb,rw) $\wedge$ Trajectory(rb,rc,rx) $\Rightarrow$
Trajectory(ra,rc,RUnion(rw,rx)).

M.F.A.F Trajectory(ra,rb,rw) $\wedge$ P(rw,rx) $\wedge$ IntConn(rx) $\Rightarrow$
Trajectory(ra,rb,rx).

M.F.A.G EmptyHanded(ta) $\wedge$ AllStable(ta) $\wedge$ Occurs(t,tb,TravelTo(rb)) $\Rightarrow$
$\exists_{\texttt{rw}:Region}$ Trajectory(Place(ta,Agent),rb,rw) $\wedge$ StaysIn(ta,tb,Agent,rw) $\wedge$
NoObstacles(ta,rw).

M.F.A.H EmptyHanded(t) $\wedge$ AllStable(t) $\wedge$ NoObstacles(t,rw) $\wedge$
Trajectory(Place(t,Agent),rb,rw) $\Rightarrow$
$\exists_{\texttt{tb}}$ Occurs(t,tb,TravelTo(rb)) $\wedge$ StaysIn(t,tb,Agent,rw).

M.F.A.I $\forall_{\texttt{t}:Time}$ Feasible(t,StandStill).
The agent always has the option of standing still.

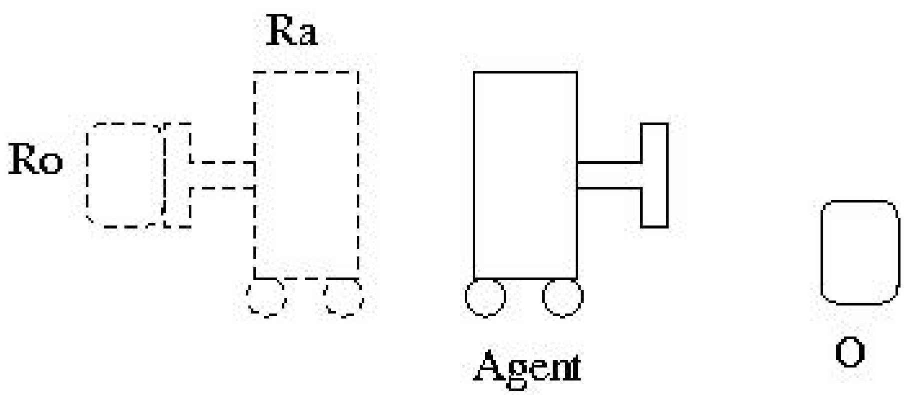


Figure 4: Moving a small object upright

6.6 Feasibility of Manipulation

We state two approximate axioms (M.M.A.A and .B) with sufficient conditions for the feasibility moving a small object. The axiom asserts that if a small object `o` can be grasped, and the agent could move empty-handed to `ra`, and region `ro` adjoins `ra`, is empty and is `SameVertical` with the current place of `o`, then the agent can execute `UprightMoveTo(o,ro)` (figure 4). Even in our restricted microworld there are numerous exceptions to this (equivalently, the microworld would need some quite fiddly tuning to make this absolutely true), so this would be better phrased as a rule of plausible inference:

- This is only true if the agent's *hand* can be made adjacent to `ro`; you cannot carry out fine manipulations with the top of your head.
- There are cases where `o` fits in `ro`, but it is impossible to get `o` into `ro` because of a bottleneck.

Axiom M.M.A.B states that, if you move an object `o` safely into an open container then you do not limit the agent's travel outside the container unless the agent himself is entirely inside the container.

Axiom:

M.M.A.A $\forall_{\texttt{o}:Object}$ `SmallObject(o)` ∧ `CanGrasp(t,o)` ∧ `Feasible(t,TravelTo(ra))` ∧
`FaceConn(ra,ro)` ∧ `NoObstacles(t,ro)` ∧ `SameVertical(ro,Place(t,o))` ∧
`AllStable(t)` ∧ `EmptyHanded(t)` ∧ `SafelyMovable(t,o)` ⇒
`Feasible(t,SafeMoveTo(o,ro)).`

M.M.A.B `Occurs(ta,tb,SafeMoveTo(o,r))` ∧ `OpenContainer(ta,ob,rd)` ∧ `P(r,rd)` ∧
`Feasible(ta,TravelTo(rx))` ∧ `DR(rx,rd)` ∧ ¬`P(Place(tb,Agent),rd)` ⇒
`Feasible(tb,TravelTo(rx)).`

7 Histories

Some spatio-temporal axioms require the use of *Histories*, functions from time to regions.

A general theory of histories would require a powerful comprehension axiom, either a second-order axiom or an axiom schema that asserts that any such function satisfying appropriate regularity conditions is indeed a *History*. Such an axiom is needed in order to prove the existence of a suitably broad class of manipulations; examples of how they are formulated and used can be found in (Davis, 2008) and (Davis, 2011). However, they are not suitable for conversion into first-order format, and they create an immense explosion of the search space in inference.[3] Instead we have a number of specialized axioms and function symbols that guarantee the existence of various histories. For instance, the fact that `HPlace(o)` is a function guarantees that the trajectory of an object `o` is a history (recall that all functions are total over their sorts). Axiom H.I.A.D guarantees the existence of a constant history for each history.

`Symbols:`
`Slice(t:`*Time*`, h:`*History*`)` $\rightarrow$ *Region*. The slice of history `h` at time `t` (a region).
`Continuous(ta,tb:`*Time*`; h:`*History*`)` —
History `h` is continuous (with respect to the Hausdorff distance (Davis, 2001)) between times `ta` and `tb`.
`HPlace(o:`*Object*`)` $\rightarrow$ *History*. The place occupied by object `o` (a history).
`HSPlace(s:ObjectSet)` $\rightarrow$ *History*. The place occupied by object set `s` (a history)
`WeaklyContinuous(ta,tb:`*Time*`; h:`*History*`)`. See below.
`Constant(t1,t2:`*Time*`; h:`*History*`)`.
History `h` has a constant value between times `t1` and `t2` (inclusive).

Definitions:

H.I.D.A `WeaklyContinuous(ta,tb,h)` $\Leftrightarrow$
`Lt(ta,tb)` $\wedge$ `History(h)` $\wedge$
$\forall_{\texttt{tm}}$ `Lt(ta,tm)` $\wedge$ `Lt(tm,tb)` $\Rightarrow$
$\exists_{\texttt{tc,td,r}}$ `Lt(tc,tm)` $\wedge$ `Lt(tm,td)` $\wedge$
$\forall_{\texttt{t}}$ `Leq3(tc,t,td)` $\Rightarrow$ `P(r, Slice(tm,h))`.
A history `h` is *weakly continuous* if it never jumps from one region to a disconnected region. Intuitively, a small marble that can knows in advance how `h` will develop can succeed in staying inside `h`. Formally, `h` is weakly continuous at time `tm` if there is an open interval `(tc,td)` containing `t` and a region `r` such that, for any time `t` in `(tc,td)`, the slice of `h` at `t` contains `r`.

H.I.D.B `Constant(t1,t2,h)` $\Leftrightarrow$
`History(h)` $\wedge$ `Lt(t1,t2)` $\wedge$ $\forall_{\texttt{t}}$ `Leq3(t1,t,t2)` $\Rightarrow$ `Slice(t,h) = Slice(t1,h)`.

Axioms:

H.I.A.A `Object(o)` $\wedge$ `Lt(ta,tb)` $\Rightarrow$ `Continuous(ta,tb,HPlace(o))`.
An object occupies a constant region.

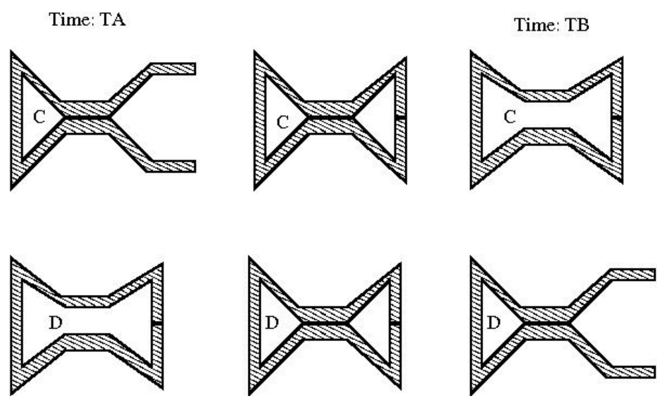
H.I.A.B $\forall_{\texttt{t:}\mathit{Time};\texttt{o:}\mathit{Object}}$ `Place(t,o) = Slice(t,HPlace(o))`.
`Place` can be defined in terms of `Slice` and `HPlace`.

H.I.A.C $\forall_{\texttt{t:}\mathit{Time};\texttt{s:}\mathit{ObjectSet}}$ `OSPlace(t,s) = Slice(t,HSPlace(o))`.
`OSPlace` can be defined in terms of `Slice` and `HSPlace`.

H.I.A.D $\forall_{\texttt{r,t1,t2}}$ `Region(r)` $\wedge$ `Lt(t1,t2)` $\Rightarrow$
$\exists_{\texttt{h}}$ `Constant(t1,t2,h)` $\wedge$ `Slice(t1,h)=r`.
For any region `r` there is a history that is constantly equal to `r`.

H.I.A.E `Constant(ta,tb,h)` $\Rightarrow$ `Continuous(ta,tb,h)`.
A constant history is continuous.

[3] Ramanchandran, Reagan and Goolsbey (2005) claim that the higher-order axioms of ResearchCyc can mostly be translated into first-order, but that there is a loss of efficiency in inference. That almost certainly does not apply here.



C is a no-exit cavity.
D is a no-entrance cavity.

Figure 5: No-exit and no-entrance cavities

H.I.A.F `Continuous(ta,tb,h)` $\Rightarrow$ `WeaklyContinuous(ta,tb,h).`
A continuous history is weakly continuous.

7.1 Dynamic containers and cavities

In a container made of flexible material, cavities can split and merge, like bubbles in liquid; they can open up to the outside world, or close themselves off from the outside world.

A history *hc* is a *dynamic cavity* of container `ho` over interval [`ta`, `tb`] if `hc` is weakly continuous and, at every time in [`ta`, `tb`], `hc` is a cavity of `ho`. We distinguish three kinds of dynamic cavities.

- `HC` is a *no-exit cavity* of `HO` if there is no way to escape from `HC`, except by going through the material of `HO` itself.
- `HC` is a *no-entrance cavity* of `HO` if there is no way to enter `HC`, except by going through the material of `HO` itself.
- `HC` is a *persistent cavity* of `HO` if it is both a no-exit and a no-entrance cavity.

Definition H.C.D.A defines persistent cavity in terms of no-exit and no-entrance cavities. H.C.A.A and .C state that no-exit and no-entrance cavities are dynamic cavities. H.C.A.B asserts that if `hc` is a no-exit cavity of `hb` and `hs` is a continuous history that starts inside `hc` at time `ta` and is outside `hc` at a later time `tb`, then `hs` overlaps with `hb` at some intermediate time. H.C.A.D makes the corresponding assertion for no-entrance cavities.

Symbols:
`NoExitCavity(t1,t2:`*Time*`; hc,ho:`*History*`)`
`NoEntranceCavity(t1,t2:`*Time*`; hc,ho:`*History*`)`
`PersistentCavity(t1,t2:`*Time*`; hc,ho:`*History*`)`

Definition:

H.C.D.A `PersistentCavity(t1,t2,hc,hb)` $\Leftrightarrow$
`NoExitCavity(t1,t2,hc,hb)` $\wedge$ `NoEntranceCavity(t1,t2,hc,hb).`

Axioms:

H.C.A.A `NoExitCavity(t1,t2,hc,ho)` $\Rightarrow$
`Lt(t1,t2)` $\wedge$ `WeaklyContinuous(t1,t2,hc)` $\wedge$
$\forall_{\mathtt{t}}$ `Leq3(t1,t,t2)` $\Rightarrow$ `Cavity(Slice(t,hc),Slice(t,ho)).`

H.C.A.B `NoExitCavity(t1,t2,hc,hb)` $\wedge$ `Continuous(t1,t2,hs)` $\wedge$
`P(Slice(t1,hs),Slice(t1,hc))` $\wedge$ $\neg$`P(Slice(t2,hs),Slice(t2,hc))` $\Rightarrow$
$\exists_{\mathtt{tm}}$ `Lt(t1,tm)` $\wedge$ `Lt(tm,t2)` $\wedge$ `O(Slice(tm,hs),Slice(tm,hb)).`
Let `hb` be the history of a container (box or bottle or bag); let `hs` be the history of some stuff; and let `hc` be a no-exit cavity in `hb`. If `hs` is inside `hc` at time `t1` and is not inside `hc` at time `t2`, then it must overlap with `hb` at some time in between.

H.C.A.C `NoEntranceCavity(t1,t2,hc,ho)` $\Rightarrow$
`Lt(t1,t2)` $\wedge$ `WeaklyContinuous(t1,t2,hc)` $\wedge$
$\forall_{\mathtt{t}}$ `Leq3(t1,t,t2)` $\Rightarrow$ `Cavity(Slice(t,hc),Slice(t,ho)).`

H.C.A.D `NoEntranceCavity(t1,t2,hc,hb)` $\wedge$ `Continuous(t1,t2,hs)` $\wedge$
$\neg$`P(Slice(t1,hs),Slice(t2,hc))` $\wedge$ `P(Slice(t2,hs),Slice(t2,hc))` $\Rightarrow$
$\exists_{\mathtt{tm}}$ `Lt(t1,tm)` $\wedge$ `Lt(tm,t2)` $\wedge$ `O(Slice(tm,hs),Slice(tm,hb)).`
Time reversed version of C.2. `hc` is a no-entrance cavity and `hs` goes from outside `hb` to inside `hc`.

H.C.A.E `Constant(t1,t2,hc)` $\wedge$ `Constant(t1,t2,ho)` $\wedge$ `Cavity(Slice(t1,hc), Slice(t1,ho))` $\Rightarrow$
`PersistentCavity(t1,t2,hc,ho).`

8 Rigid Objects

Rigid objects maintain their shape over time; they are a particularly important and well-behaved kind of object. For our purposes, the only property that we use is that any cavity of a rigid object is a persistent cavity (axiom R.G.A.B)

Symbols:
`RigidObject(o:` *Object*`).` — `o` is a rigid solid object.
`RigidHistory(h:` *History*`)` — `h` is a rigid history.

Axioms:

R.G.A.A `RigidObject(o)` $\Rightarrow$ `RigidHistory(HPlace(o)).`

R.G.A.B $\forall_{\mathtt{h}:\mathit{History},\mathtt{r}:\mathit{Cavity},\mathtt{t1},\mathtt{t2}:\mathit{Time}}$ `RigidHistory(h)` $\wedge$ `Cavity(r,Slice(t1,h))` $\wedge$ `Lt(t1,t2)` $\Rightarrow$
$\exists_{\mathtt{hc}:\mathit{History}}$ `RigidHistory(hc)` $\wedge$ `PersistentCavity(t1,t2,hc,h)` $\wedge$ `r = Slice(t1,hc).`

Cavities inside rigid histories are persistent.

9 Actions

9.1 Simple Actions

We here define a collection of specific actions in our domain, for which it is possible to state much stronger physical axioms (causal axioms, frame axioms, and preconditions) in qualitative terms than for unconstrained manipulation. We illustrate with one example: If a container is carried upright, then the contents remain inside.

`CarrySimple(o:`*Object*`; r:`*Region*`).`
`CarryClosed(o:`*Object*`; s:`*ObjectSet*`; r:`*Region*`).`
`CarryUpright(o:`*Object*`; r:`*Region*`).`
`CarryBoxWithLid(ob,ol:`*Object*`: r:`*Region*`).`
`PutLidOnBox(ob,ol:`*Object*`: r:`*Region*`).`
`TakeLidOffBox(ob,ol:`*Object*`: r:`*Region*`).`
`CloseBag(o:`*Object*`; r,rc:`*Region*`).`
`OpenBag(o:`*Object*`; r,rc:`*Region*`).`

Definitions:

A.S.D.A $\texttt{Occurs(ta,tb,CarrySimple(o,r))} \Leftrightarrow$
$\texttt{Occurs(ta,tb,MoveTo(o,r))} \wedge \texttt{UContents(ta,o) = Null} \wedge \texttt{CContents(ta,o) = Null}.$
Carry object `o` to region `r`, bringing nothing along with it.

A.S.D.B $\texttt{Occurs(ta,tb,CarryClosed(o,s,r))} \Leftrightarrow$
$\texttt{Occurs(ta,tb,MoveTo(o,r))} \wedge \texttt{CContents(ta,o)} \neq \texttt{Null} \wedge$
$\forall_{\texttt{rc}}\ \texttt{ClosedContainer(t,Singleton(o),rc)} \Rightarrow$
$\exists_{\texttt{hc}:History}\ \texttt{Slice(ta,hc) = rc} \wedge \texttt{PersistentCavity(ta,tb,hc,HPlace(ol))}.$
Carry closed container `o` to region `r`, maintaining all internal cavities. The object set `s` is the set of objects inside.

A.S.D.C $\texttt{Occurs}(\texttt{ta},\texttt{tb},\texttt{CarryUpright}(\texttt{o},\texttt{s},\texttt{r})) \Leftrightarrow$
$\texttt{Occurs}(\texttt{ta},\texttt{tb},\texttt{MoveTo}(\texttt{o},\texttt{r})) \wedge$
$\exists_{\texttt{rc},\texttt{hc}}\ \texttt{UprightContainer}(\texttt{ta},\texttt{o},\texttt{rc}) \wedge \texttt{s} = \texttt{UContents}(\texttt{ta},\texttt{rc}) \wedge$
$\texttt{Continuous}(\texttt{ta},\texttt{tb},\texttt{hc}) \wedge$
$\forall_{\texttt{tm}}\ \texttt{Leq3}(\texttt{ta},\texttt{tm},\texttt{tb}) \Rightarrow$
$\texttt{UprightContainer}(\texttt{tm},\texttt{o},\texttt{Slice}(\texttt{tm},\texttt{hc})) \wedge$
$\texttt{Fits}(\texttt{s},\texttt{Slice}(\texttt{tm},\texttt{hc})).$
Carry open container `o` to region `r`, keeping it upright, and keeping the cavity large enough to contain object set `s`, which is initially inside.

A.S.D.D $\texttt{Occurs(ta,tb,CarryBoxWithLid(ob,ol,r))} \Leftrightarrow$
$\texttt{Occurs(ta,tb,MoveTo(ob,r))} \wedge$
$\forall_{\texttt{tm}}\ \texttt{Leq3(ta,tm,tb)} \Rightarrow \quad \texttt{m}\exists_{\texttt{rc}}\ \texttt{BoxWithLid(tm,ob,ol,rc)}.$
Carry box `ob` to region `r`, maintaining `ol` as a lid.

A.S.D.E $\texttt{Occurs(ta,tb,PutLidOnBox(ob,ol,r))} \Leftrightarrow$
$\texttt{Occurs(ta,tb,MoveTo(ol,r))} \wedge \exists_{\texttt{rc}}\ \texttt{BoxWithLid(tb,ob,ol,rc)}.$
Put lid `ol` on box `ob`.

A.S.D.F $\texttt{Occurs(ta,tb,TakeLidOffBox(ob,ol,r))} \Leftrightarrow$
$\texttt{Occurs(ta,tb,MoveTo(ol,r))} \wedge$

$[\exists_{\texttt{rc}}$ `BoxWithLid(ta,ob,ol,rc)`$] \wedge [\neg\exists_{\texttt{rc}}$ `BoxWithLid(tb,ob,ol,rc)`$]$.
Take lid `ol` off box `ob`.

A.S.D.G `Occurs(ta,tb,CloseBag(o,r,rc))` $\Leftrightarrow$
`Occurs(ta,tb,MoveTo(o,r))` $\wedge$ `Outside(Place(tb,Agent),Place(tb,o))` $\wedge$
$\exists_{\texttt{hc}:History;\texttt{tm}:Time}$ `Slice(ta,hc)=rc` $\wedge$ `WeaklyContinuous(ta,tb,hc)` $\wedge$
`Leq3(ta,tm,tb)` $\wedge$
$[\forall_{\texttt{t}}$ `Leq(ta,t)` $\wedge$ `Lt(t,tm)` $\Rightarrow$
`UprightContainer(t,o,Slice(t,hc))`$] \wedge$
$[\forall_{\texttt{t}}$ `Lt(tm,t)` $\wedge$ `Leq(t,tb)` $\Rightarrow$
`ClosedContainer(t,Singleton(o),Slice(t,hc))`$]$.
Change object `o` from a upright open container to a closed container, keeping it upright until it is closed.

A.S.D.H `Occurs(ta,tb,OpenBag(o,r,rc))` $\Leftrightarrow$
`Occurs(ta,tb,MoveTo(o,r))` $\wedge$ `Outside(Place(tb,Agent),Place(tb,o))` $\wedge$
$\exists_{\texttt{hc}:History;\texttt{tm}:Time}$ `Slice(ta,hc)=rc` $\wedge$ `WeaklyContinuous(ta,tb,hc)` $\wedge$
`Leq3(ta,tm,tb)` $\wedge$
$[\forall_{\texttt{t}}$ `Leq(ta,t)` $\wedge$ `Lt(t,tm)` $\Rightarrow$
`ClosedContainer(t,Singleton(o),Slice(t,hc))`$] \wedge$
$[\forall_{\texttt{t}}$ `Lt(tm,t)` $\wedge$ `Leq(t,tb)` $\Rightarrow$
`UprightContainer(t,o,Slice(t,hc))`$]$.
Change object `o` from a closed container to a open upright container, keeping it upright after it has been opened.

Axiom:

A.S.A.A `Occurs(ta,tb,CarryUpright(o,s,r))` $\wedge$ `Leq3(ta,t,tb)` $\Rightarrow$
`UContents(tm,o) = UContents(ta,o)`.

9.2 Compound Actions

We also define specific compound actions; again, it may be possible to give stronger qualitative characterizations of these. We give a single example of loading an object `ox` into an open upright container `ob`.

Symbols:
`Reachable(t:`*Time*; `r:`*Region*) `Sequence(e1,e2:`*Event*) $\rightarrow$ *Event.*
`PutInUC(ox,ob:`*Object*) $\rightarrow$ *Event.*
`LoadUprightContainer(ox,ob:`*Object*) $\rightarrow$ *Event*

Definitions:

A.C.D.A $\forall_{\texttt{ta,tb}:Time;\texttt{e1,e2}:Event}$ `Occurs(ta,tb,Sequence(e1,e2))` $\Leftrightarrow$
$\exists_{\texttt{tx}}$ `Occurs(ta,tx,e1)` $\wedge$ `Occurs(tx,tb,e2)`.
General sequence operator; Execute `e1` then `e2`.

A.C.D.C $\forall_{\texttt{ta,tb}:Time;\texttt{ox,ob}:Object}$ `Occurs(ta,tb,PutInUC(ox,ob))` $\Leftrightarrow$
$\exists_{\texttt{rc,rx}}$ `UprightContainer(ta,ob,rc)` $\wedge$ `P(rx,rc)` $\wedge$
`Occurs(ta,tb,SafeMoveTo(ox,rx))` $\wedge$ `OV(Place(tb,Agent),rc)`.

A.C.D.B `Occurs(ta,tb,LoadUprightContainer(ox,ob))` $\Leftrightarrow$
$\exists_{\texttt{r1,r3}}$ `OutsideContainer(r3,Place(ta,ob))` $\wedge$

Occurs(ta,tb,Sequence(TravelTo(r1),Sequence(PutInUC(ox,ob), TravelTo(r3))).

Loading object ox into open upright container ob is the sequence of travelling to a place where ox can be grasped, moving ox inside ob and then withdrawing the manipulator out of ob. The container ob remains motionless throughout.

A.C.D.D Reachable(ta,r) $\Leftrightarrow$
$\exists_{rx}$ IntConn(RUnion(rx,r)) $\wedge$ Feasible(t,TravelTo(rx)).
Region r is reachable at time t if it is feasible for the agent to travel to a position rx such that r $\cup$ rx is interior connected.

Axioms:

A.C.A.A UprightContainer(ta,ob,rc) $\wedge$ CanGrasp(ta,ox) $\wedge$
SmallSet(Union(UContents(ta,ob),Singleton(ox)), rc) $\wedge$ Reachable(ta,rc) $\Rightarrow$
Feasible(ta,PutInUC(ox,ob)).
Feasibility axiom: If ob is an upright container with cavity rc, the agent can grasp ox, ox together with the current contents of rc is small as compared to rc, and the agent can reach inside rc, then the agent can load ox into rc.

10 Inferences

We come at last to our example inferences. Complete proofs of Scenarios 1-4 in a natural-deduction format may be found in the external documents

10.1 Scenario 1

Qualitative prediction. If Ob1 is a rigid object and it is a closed container container object Ox1, then Ox1 remains inside Ob1.

Symbols:
Ox1 $\rightarrow$ *Object* — Some stuff.
Ob1 $\rightarrow$ *Object* — A box.
Ta1,Tb1 $\rightarrow$ *Time* — Times.

Given:

C.1.A.A RigidObject(Ob1).

C.1.A.B CContained(Ta1,Ox1,Singleton(Ob1)).

C.1.A.C Lt(Ta1,Tb1).

C.1.A.D Ob1 $\neq$ Ox1.

Infer: CContained(Tb1,Ox1,Singleton(Ob1)).

Depends On: S.B.D.B, O.C.D.A, O.R.D.A, O.T.A.B, O.T.A.F, H.I.A.A, H.I.A.B, H.C.D.A, H.C.A.A, H.C.A.B, R.G.A.A, R.G.A.B, C.1.A.A, C.1.A.B, C.1.A.C, C.1.A.D,

10.2 Scenario 2

Infer object characteristics from behavior. If the stuff `Os2` is contained in `Ob2` at one time and outside at another, then `Ob` is not a rigid object. Note: the order of `Ta2,Tb2` is not specified.

Symbols:
`Os2` → *Object* — Some stuff.
`Ob2` → *Object* — A container.
`Ta2,Tb2` → *Time* — Times.

C.2.A.A `CContained(Ta2,Os2,Singleton(Ob2)).`

C.2.A.B `Outside(Place(Tb2,Os2),Place(Tb2,Ob2)).`

C.2.A.C `Ordered(Ta2,Tb2).`

Infer: ¬`RigidObject(Ob2).`

Depends On: T.I.D.B, S.B.D.A, S.B.D.B, S.B.A.C, S.C.D.C, S.C.D.D, O.C.D.A, O.R.D.A, O.T.A.B, O.T.A.F, H.I.A.A, H.I.A.B, H.C.D.A, H.C.A.A, H.C.A.B, H.C.A.C, H.C.A.D, R.G.A.A, R.G.A.B, C.2.A.A, C.2.A.B, C.2.A.C.

10.3 Scenario 3

Qualitative prediction. If `Ob3b` is a rigid object and a closed container containing `Ob3a`, and `Ob3a` is a closed container (not necessarily rigid) containing object `Os3`, then `Os3` will remain inside `Ob3b`.

Symbols:
`Os3` → *Object* — Some stuff.
`Ob3a` → *Object* — Inner container.
`Ob3b` → *Object* — Outer box.
`Ta3,Tb3` → *Time* — Times.

C.3.A.A `RigidObject(Ob3b).`

C.3.A.B `CContained(Ta3,Os3,Singleton(Ob3a)).`

C.3.A.C `CContained(Ta3,Ob3a,Singleton(Ob3b)).`

Infer: `CContained(Tb3,Os3a,Singleton(Ob3a)).`

Depends On: S.B.D.B, O.C.D.A, O.R.D.A, O.T.A.B, O.T.A.F, H.I.A.A, H.I.A.B, H.C.D.A, H.C.A.A, H.C.A.B, H.C.A.C, H.C.A.D, R.G.A.A, R.G.A.B, C.3.A.A, C.3.A.B, C.3.A.C

10.4 Scenario 4

If the situation depicted in figure 6 (from Smith, Dechter, Tenenbaum, and Vul, 2013) is modified so that the red region is flush against the green region, then the ball must reach the red region before it can reach the green region.

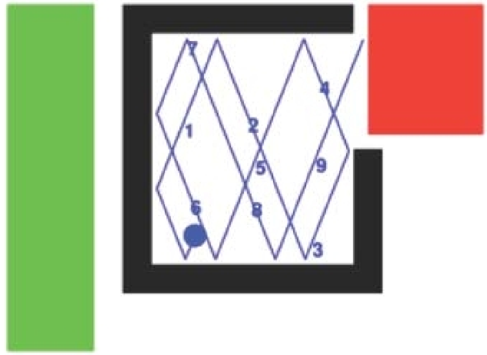


Figure 6: Reasoning about a bouncing ball (from (Smith, Dechter, Tenenbaum, and Vul, 2013))

Symbols:
`Os4` $\rightarrow$ *Object.* Movable object.
`Ob4` $\rightarrow$ *Object.* Fixed object.
`RRed, RGreen, RInside` $\rightarrow$ *Region.* Two target regions.
`Ta4, Tb4` $\rightarrow$ *Time.*

C.4.A.A $\forall_t$ `Place(t,Ob4) = Place(Ta,Ob4).` `Ob4` is fixed.

C.4.A.B `CombinedContainer(Place(Ta,Ob4), RRed, RInside)`

C.4.A.C `P(Place(Ta4,Os4),RInside).`

C.4.A.D `Outside(RGreen, RUnion(Place(Ta,Ob4),RRed)).`

C.4.A.E `P(Place(Tb4,Os4),RGreen).`

C.4.A.F `Lt(Ta4,Tb4).`

Infer: $\exists_{tm}$ `Lt(Ta4,tm)` $\wedge$ `Lt(tm,Tb4)` $\wedge$ `O(Place(tm,Os4),TRed).`

Depends On: S.B.D.A, S.B.D.B, S.B.A.A, S.B.A.H, S.C.D.E, O.T.A.B, H.I.D.B, H.I.A.A, H.I.A.B, H.I.A.D, H.C.D.A, H.C.A.B, H.C.A.E, C.4.A.A, C.4.A.B, C.4.A.C, C.4.A.D, C.4.A.E, C.4.A.F.

10.5 Scenario 5

If Ox5 is outside upright container Ob5, and the current contents of Ob5 together with Ox5 are much smaller than the interior of Ob5, and the agent can reach and move Ox5 and can reach into Ob5, then

a. The agent can load Ox5 into Ob5; and

b. If the agent does load Ox5 into Ob5, then Ox5 will be contained in Ob5.

C.5.A.A `UprightContainer(Ta5,Ob5,Rc5).`

C.5.A.B `DC(Place(Ta5,Ox5),Rc5).`

C.5.A.C `SmallSet(Union(Contents(Ta5,Rc5),Singleton(Ox5),Rc5)).`

C.5.A.D `AllStable(Ta5).`

C.5.A.E `EmptyHanded(Ta5).`

C.5.A.F `Graspable(Ta5,Ox5).`

C.5.A.G `Reachable(Ta5,Rc5).`

C.5.A.H `Ox5` $\neq$ `Agent` $\neq$ `Ob5.`

C.5.A.I `OutsideContainer(Place(Ta5,Agent),Place(Ta5,Ob5)).`

C.5.A.J `SafelyMovable(Ta5,Ox5).`

C.5.A.K `SmallObject(Ox5).`

Infer: `Feasible(Ta5,Tb5,LoadUprightContainer(Ox5,Ob5).`

DependsOn:
T.I.D.A, T.I.D.D, T.I.A.B, T.E.A.A,
S.B.D.B, S.B.D.C, S.B.D.D, S.B.D.F, S.B.A.B, S.B.A.H, S.B.A.I, S.C.D.A, S.C.D.F, S.C.D.H, S.C.A.C,
O.S.D.C, O.S.D.F, O.R.D.A, O.R.D.C, O.R.D.D, O.C.D.C, O.C.D.E,
M.G.D.A, M.G.D.B, M.G.D.C, M.G.D.D, M.G.A,B, M.O.D.D, M.O.D.E, M.O.D.F, M.O.D.G, M.O.D.J,
M.O.A.A, M.R.A.A, M.R.D.A, M.R.A.C, M.F.D.A, M.F.D.B, M.F.D.D, M.F.A.D, M.F.A.E, M.F.A.G,
M.F.A.H, M.F.A.I, M.M.A.B,
A.C.D.A, A.C.D.B, A.C.D.C, A.C.D.D, A.C.A.A,
C.5.A.A–.K

References

E. Davis (2001). Continuous shape transformation and metrics on regions. *Fundamenta Informaticae.* **46**(1-2):31-54. E. Davis (2008). Pouring Liquids: A Study in Commonsense Physical Reasoning *Artificial Intelligence,* **175,** 1540-1578.

E. Davis (2011). How Does a Box Work? A Study in the Qualitative Dynamics of Solid Objects. *Artificial Intelligence,* **175,** 299-345.

P. Hayes (1985). An Ontology for Liquids. In J. Hobbs and R. Moore (eds.) *Formal Theories of the Commonsense World,* Ablex.

I. Pratt-Hartmann (2007). First-Order Mereotopology. In J. Aiello, I. Pratt-Hartmann, and J. van Benthem (eds.), *Handbook of Spatial Logics*, Springer.

I. Pratt and D. Schoop (1998). A complete axiom system for polygonal mereotopology of the real plane. *Journal of Philosophical Logic,* **27:**6, 621-658.

D. Ramachandran, P. Reagan, and K. Goolsbey (2005). First-Orderized ResearchCyc: Expressivity and Efficiency in a Common-Sense Ontology. AAAI Workshop on Contexts and Ontologies.

D. Randell, Z. Cui, and A.G. Cohn (1992). A spatial logic based on regions and connection. *KR-92.*

L. Schubert. (1990) Monotonic Solution of the Frame Problem in Situation Calculus. In H. Kyburg, R. Loui, and G. Carlson, *Knowledge Representation and Defeasible Reasoning,* Kluwer, 23-67.

K. Smith, E. Dechter, J. Tenenbaum, and E. Vul (2013). Physical predictions over time. *Proceedings of the 35th Annual Meeting of the Cognitive Science Society.*