Artificial Intelligence 175 (2011) 346–377

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Formalising the Fisherman's Folly puzzle

Pedro Cabalar [a,*], Paulo E. Santos [b]

[a] Dept. Computación, University of Corunna, Spain
[b] AI and Automation Group (IAAA), Centro Universitário da FEI, São Paulo, Brazil

ARTICLE INFO

Article history:
Available online 3 April 2010

Keywords:
Common sense reasoning
Qualitative spatial reasoning
Reasoning about actions and change

ABSTRACT

This paper investigates the challenging problem of encoding the common sense knowledge involved in the manipulation of spatial objects from a reasoning about actions and change perspective. In particular, we propose a formal solution to a puzzle composed of non-trivial objects (such as holes and strings) assuming a version of the Situation Calculus written over first-order Equilibrium Logic, whose models generalise the stable model semantics.

1. Introduction

Real life situations where we must deal with strings tying objects and passing through holes appear from time to time in very different contexts. Examples range from tying shoelaces, to handling ropes in a sailboat or organising the cable connections map inside an office, a building or a whole city. Although humans show an amazing intuition for solving problems of this nature, a formal representation for reasoning about holes and strings is still a relatively unexplored area. To understand the complexity of this problem, note for instance that using a fully detailed mathematical model of the involved objects (assuming them as rigid) does not seem feasible for computational purposes, let alone when we consider deformable objects like a string. Moreover, humans typically describe solutions to spatial reasoning problems in terms of *qualitative* descriptions instead. This is, in fact, the orientation followed by *Qualitative Spatial Reasoning* (QSR) [1–3], a field that attempts the logical formalisation of spatial knowledge based on primitive relations defined over elementary spatial entities. In general QSR approaches, however, lack a formal treatment of actions and change (apart from a few exceptions [4,5]). The formal treatment of actions are of central interest to the Reasoning about Actions and Change (RAC) community [6,7]. In this work we formalise a spatial domain from the perspective of RAC, aiming to provide a step in the direction of a rigorous treatment of reasoning about actions, change and qualitative space. This work falls within the logic-based knowledge representation subfield of Artificial Intelligence [8,9], whose main goals are: the logical formalisation of reasoning processes capable of inferring knowledge from representations of the world; the construction of a medium for pragmatically efficient computation, in which the formal representation provides the means to organise domain knowledge allowing for efficient (and consistent) queries, updates and revisions of the knowledge base; or the rigorous treatment of ontological commitments, which provide the base rules that guide reasoning about the world (for instance, what should or should not be considered as the effects of actions, to which extent should ramifications of these effects be considered, how to assume the persistence of objects through time and so on) [10]. In particular the problem of correctly inferring the effects (and non-effects) of actions has received much attention by the AI community under the heading of the Frame Problem.

In a nut-shell, the challenge of solving the Frame Problem is to find a compact representation of the non-effects of actions, such that the complexity of the representation is at the order of the number of actions in the domain, and not of the number of domain elements times actions. Much solutions to this problem were related to the development of

* Corresponding author.
E-mail addresses: cabalar@udc.es (P. Cabalar), psantos@fei.edu.br (P.E. Santos).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.04.004

non-monotonic reasoning formalisms [11] and the definition of a *law of inertia*, which states that actions do not change properties of a domain unless explicitly written [12]. Attempts to solve the Frame Problem stumbled on the existence of multiple possible models for sound domain representations, some of which were counter-intuitive as exemplified by the so-called Yale Shooting problem [13]. In this work we tackle the formalisation of a spatial scenario within a logic that has built into it an elegant solution to the Frame Problem as one of its ontological commitments, as exemplified in the simple solution to the Yale Shooting scenario presented in Section 4.

To obtain a suitable representation of spatial domains containing strings and holes we have adopted the following methodology. We begin from specific formalisations of particular scenarios, what usually implies a more abstract and simplified description level, and advance then towards more general representations to cover different domains, what necessarily implies a more fine-grained ontology. As a starting point, puzzle-like examples constitute a good test bed, as they offer a small number of objects while keeping enough complexity for a challenging problem of knowledge representation. Thus, puzzles involving physical objects are our *drosophila*,[1] i.e. our base line from which we develop AI research.

In this paper, we take as a starting point the work developed in [16], which presented an automated solution to the classical puzzle called *Fisherman's Folly* introduced below. The approach taken in [16], however, falls short on the formal treatment of fluents, actions and the problematics surrounding these concepts (such as inertia). The present paper tackles these issues by presenting a formalisation of the *Fisherman's Folly* puzzle into an Equilibrium Logic version of the Situation Calculus (SC). Here Situation Calculus [17,6] provides the syntactical tools to formalise the domain and Equilibrium Logic [18], a suitable semantics to the problem, which identifies its minimal models by means that resemble circumscription [19, 20]. We shall deal with these concepts in more details below.

In this context, the original contributions of the present work include the following:

- a novel formalisation of the Situation Calculus in Equilibrium Logic. The main advantage of this formalisation over the traditional classic logic SC is the first-order encoding of predicate extension minimisation providing a rigorous solution to the frame problem (which in classic logic is accomplished by a fragment of the second-order calculus);
- the first logical formalisation of a scenario composed of a string and rigid objects (hosting holes or not). This formalisation allows the representation of string segments, as well as their creation (and annihilation), depending on the action evoked. It is worth mentioning that the necessity to handle fragments of objects in this way brings a new challenge to Knowledge Representation formalisms;
- in this paper we define a notion of connectivity as a fluent within an action formalism (the fluent *linked* in Section 5.1), an issue that has not been considered before in the field of Qualitative Spatial Reasoning.

1.1. The Fisherman's Folly puzzle

The elements of the Fisherman's Folly puzzle are a holed post (*Post*) fixed to a wooden base (*Base*), a string (*Str*), a ring (*Ring*), a pair of spheres (*Sphere*1, *Sphere*2) and a pair of disks (*Disk*1, *Disk*2). The spheres can be moved along the string, whereas the disks are fixed at each string endpoint. The string passes through the post's hole in a way that one sphere and one disk remain on each side of the post. It is worth pointing out that the spheres are larger than the post's hole, therefore the string cannot be separated from the post without cutting either the post, or the string, or destroying one of the spheres. The disks and the ring, in contrast, can pass through the post's hole.

In the initial state (shown in Fig. 1(a)) the post is in the middle of the ring, which in its turn is supported on the post's base. The goal of this puzzle is to find a sequence of (non-destructive) transformations that, when applied on the domain objects, frees the ring from the other objects, regardless their final configuration. Fig. 1(b) shows one possible goal state.

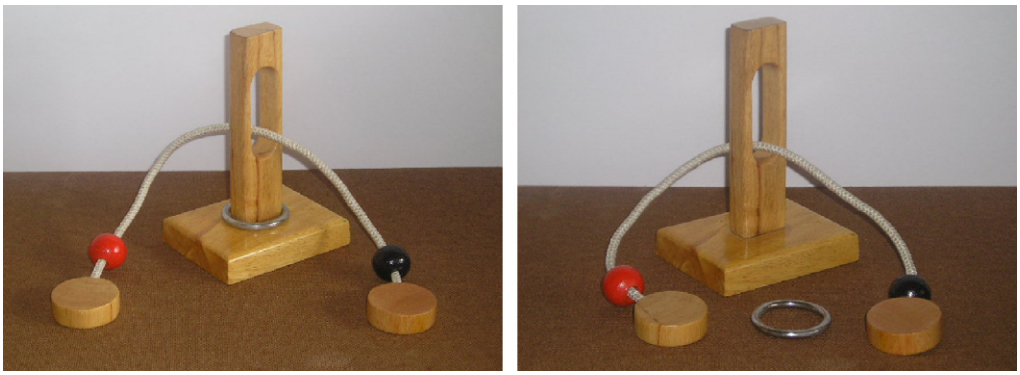
A crucial observation is that the puzzle actually deals with four holes: the post hole, the ring hole and the two sphere holes. Note that in a natural language description we would probably identify holes with their host objects, saying that "the string passes through the sphere" (hole) or that "the post passes through the ring" (hole). Furthermore, we would talk about "sliding the ring up the post," rather than "moving the post down through the ring hole".

We argue that the complexity imposed by the (possibly infinite) states of the string allied to the necessity of handling void space (holes) makes spatial puzzles such as Fisherman's Folly rich benchmark domains to be tackled from the Common Sense Knowledge Representation standpoint [21,8]. Moreover, besides the fact that this problem involves both Reasoning about Actions and Change and Qualitative Spatial Reasoning about objects (that are ubiquitous in various everyday domains), there are a number of versions of puzzles involving strings and holes.[2] Making use of these analogous puzzles we can check whether the representation and reasoning methods *developed for one puzzle could be applied to other of its variants*, thus verifying the elaboration tolerance [22] of our formalisation.

In this paper we show that the same formalism developed to solve the Fisherman's Folly puzzle solves other puzzles: the Rope Ladder puzzle – Fig. 2 (whose solution is presented in Section 7) and the Tricky Dick puzzle (cf. Appendix C). The way the proposed formalism could be easily applied on distinct domains suggests that the proposed solution is tolerant to elaborations. However, we make no claim that the work developed here can solve *any* problem involving strings, holes and

[1] Following the metaphor used in [14] and [15].

[2] E.g. http://www.puzzlethis.co.uk/products_mad_cow.htm (last accessed in 12/10/2009).



(a) Initial state (b) Goal

Fig. 1. The Fisherman's Folly puzzle.

Fig. 2. The Rope Ladder puzzle in its initial state.

rigid objects. For instance, the formalism is not capable of representing the case of a number of rings interlocking (such as in the symbol of the Olympic games), puzzles where string loops (or knots) are relevant to the solution, and so forth.

At this point it is worth mentioning some previous work on dealing (separately) with holes and strings from a formal standpoint. Reasoning about holes and holed objects have been discussed in detail in [23] and [24], whereby a formal ontology for these entities, based on their topological aspects, is developed. In the present work we assume holes as sharing the same level of existence as rigid objects, i.e., they are reified individuals that can be involved in actions. In the present paper we are mostly interested in how these entities could be engaged in actions, leaving aside some aspects of their precise ontology. A fine grained spatial ontology of the Fisherman's Folly (following the guidelines proposed in [23]) is presented in [25].

The problem of incorporating knowledge about strings and string manipulation has been tackled in [26,27] where a robotic system capable of learning to tie a knot from visual observation is proposed – this system is called *the Knot Planning from Observation* (KPO) paradigm. In KPO each state of a string is represented by a matrix encoding its segments, which are defined by the portion of the string that lies in between its endpoints and points where it crosses over itself. Actions on flexible objects in this context were defined as an extension of the Reidemeister moves in knot theory [28]. This representation is suitable for the identification of string states from a computer vision system; however, it falls short in the context of problem solving, which is the main purpose of the present paper. In contrast to the work proposed in [26], we do not take into account knots in this paper. Incorporating some of the ideas of the KPO paradigm in our work shall be investigated in the future. Nonetheless, research on knot planning and reasoning about strings and holes are some of the issues that the growing community of robot assisted needle steering are interested about [29].

This paper is organised as follows: Section 2 presents an early solution to the puzzle in order to provide the intuitions behind the present development; Section 3 introduces Equilibrium Logic; Situation Calculus theories written in Equilibrium Logic are developed in Section 4, while Section 5 presents the formalisation of Fisherman's Folly in this version of Situation Calculus. The correctness of the developed formalism is discussed in Section 6; Section 7 presents a formal solution of a more complex version of the Fisherman's Folly puzzle and Section 8 concludes this paper. Appendices A, B and C present,

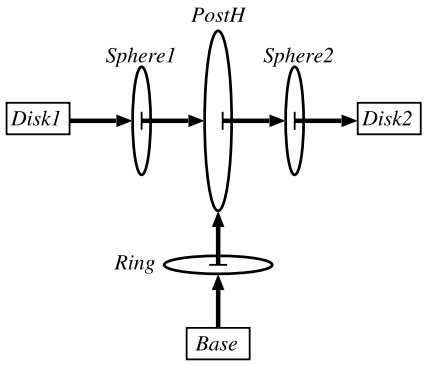


Fig. 3. Schematic representation of the initial state.

respectively, the Prolog code of the planner developed following the guide lines presented in the paper, the proofs of this paper's propositions and a formal solution of a third puzzle.

2. A simple automated solution to the Fisherman's Folly puzzle

The formalisation of Fisherman's Folly presented in this paper is based on a simple automated solution of the puzzle proposed in [16], briefly presented in this section to provide the intuitions behind this work.

The simple solution presented in [16] relies on distinguishing the puzzle's objects into three sorts: *holes* (which includes the post hole, the ring hole and the holes through the spheres), long objects (that includes the string and the post), and regular objects (including all the remainder objects). For each hole h, its faces are distinguished: h^- and h^+; and for each long object l its tips l^- and l^+ are defined.

The domain objects are represented by the following constants: *Disk*1, *Disk*2 (for the left and right disks), Base (representing the square base), *Ring*, *Sphere*1, *Sphere*2, *PostH* (representing the holed objects), and *Str* and *Post*, for the so called *long objects*: the string and the non-holed part of the post (as shall be discussed further on the paper).

For helping the reader to figure out a puzzle state, we use schematic representations like the one in Fig. 3, which shows the initial state. Arrows correspond to segments of long objects, defined between pairs of hole crossings, or between a hole crossing and a tip. These arrows point in the direction from tip l^- to tip l^+ of a same long object l. Ellipses represent holes and boxes are linked regular objects. The positive face of a hole implicitly corresponds to the "visible" side of the ellipse.

Central to this simple solution is the definition of a list data structure named $chain(X)$. This data structure represents the sequence of all hole crossings on a long object X, when traversing X from its negative tip to its positive one. For instance, the state shown in Fig. 3 is represented by the following two chains: $chain(Post) = [Ring^+]$ and $chain(Str) = [Sphere1^+, PostH^+, Sphere2^+]$. The former represents that the long object *Post* crosses the ring hole and the latter states that the string crosses the hole on the sphere 1, the post hole and the hole on the sphere 2, respectively. Note that, for brevity, only the outgoing hole faces are shown, following the direction negative to positive tip.

An action *pass* was defined[3] to represent the movements of puzzle objects. The effects of *pass* either add or delete a hole crossing from the *chain* on which it is applied.

Using these definitions, a solution to the Fisherman's Folly puzzle can be represented by the sequence of chains shown in Fig. 4, whereby each state is identified by its sequence number plus the pair of lists $chain(Post)$ and $chain(Str)$ in this order. Note that State 5 has actually reached the goal since, at this point, the ring hole *Ring* does not occur in any list, i.e., it is not crossed by any long object.

In [16] we also present an iterative deepening planning system that manipulates chains in order to find a solution to the puzzle.

This early approach, however, completely overlooks any formal aspects of the domain or the system. This lack of formality implies on a number of drawbacks. First, there was no possibility of proving the correctness of the automated solutions obtained, since we did not have available the mathematical tools to analyse the system. Second, there was no proper ontology of the domain, the simple chain definition did not allow any such formal definition. Amongst the definitions we would like our theory to capture are the spatial properties of objects as three-dimensional regions (including holes). Besides spatial properties, a formalisation of a spatial puzzle should have a rigorous treatment of actions and change, including the indirect effects of actions, the definition of non-executable actions and the efficient representation of actions' non-effects (issues that are virtually impossible to address given the early list-based representation). In contrast, the present paper proposes a rigorous logical account for the Fisherman's Folly puzzle, from which its solution can follow as a logical proof.

[3] In [16] two actions were defined, one representing the movement of rigid objects and another the movement of holed objects. As we will see later, in the current paper these two actions will be replaced by a single one for passing a bundle of elements through a hole. Fig. 4 actually shows the solution in terms of this new representation.

state	$chain(Post)$	$chain(Str)$
S_0	$[Ring^+]$	$[Sphere1^+, PostH^+, Sphere2^+]$
s_1	$[Ring^+]$	$[Sphere1^+, PostH^+, Sphere2^+, PostH^-]$
s_2	$[\,]$	$[Sphere1^+, Ring^-, PostH^+, Ring^+, Ring^-, Sphere2^+, Ring^+, PostH^-]$
s_3	$[\,]$	$[Sphere1^+, Ring^-, PostH^+, Sphere2^+, PostH^-, Ring^+]$
s_4	$[\,]$	$[Sphere1^+, PostH^+, Ring^-, Sphere2^+, Ring^+, PostH^-]$
s_5	$[\,]$	$[Sphere1^+, PostH^+, Sphere2^+, PostH^-]$

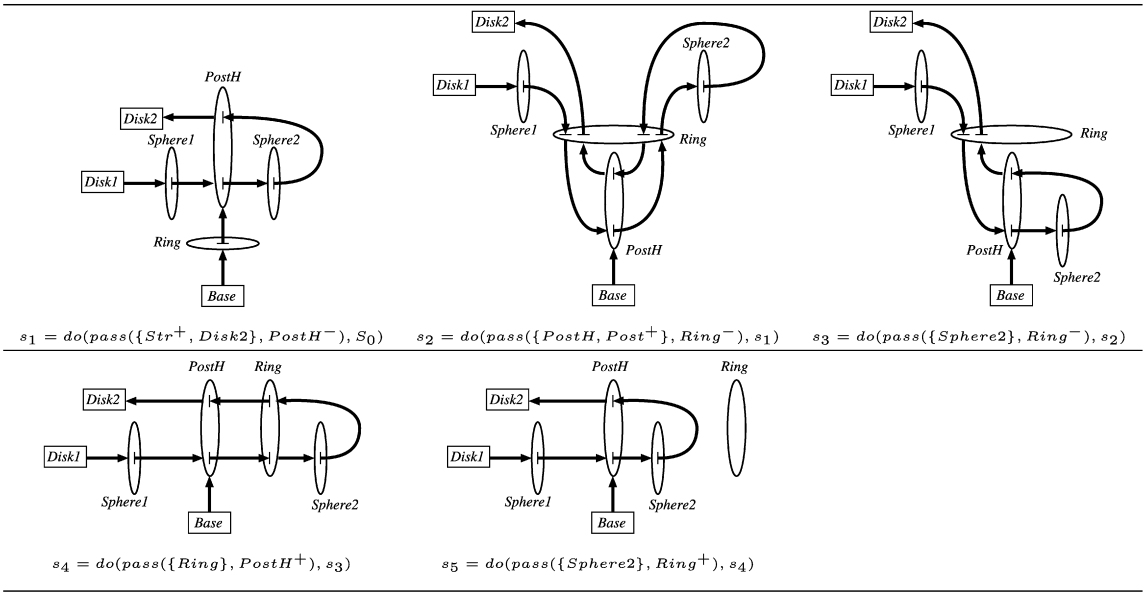


Fig. 4. A formal solution for the Fisherman's puzzle and its graphical representation.

Moreover, the relational nature of the formalism allows for the definition of the spatial extent of objects (some of which are defined and used in this paper, as we shall see). On top of all, the theory proposed is defined on a well-known reasoning about actions and change formalism, which carries built into it a complete account of the problematics surrounding rigorous definitions of dynamic domains.

3. First-order equilibrium logic

There exist several alternatives for incorporating non-monotonicity in a formalism for Reasoning about Actions and Change (RAC). One possibility is using Predicate Circumscription [19], so that, we define a set of auxiliary predicates[4] and then minimise their extent using a particular circumscription policy – that is, deciding which predicates are fixed, minimised or freely varied; deciding which parts of the theory are circumscribed and which parts are not, etc. These decisions have proved to be a non-trivial task. In fact, it can be said that the area of RAC has evolved by a continuous proposal, and later solution, of motivating examples showing counterintuitive results from an inadequate circumscription or minimisation policy.[5]

Another possibility for formalising action domains that has recently gained some popularity is the use of Logic Programming (LP) as an underlying logical devise, and in particular, the methodology proposed in [35] that relies on the *Stable Models* semantics [36]. The advantages of LP are that: (1) default negation provides a natural way for representing defaults and, in particular, the inertia law for solving the frame problem; and (2) unlike classical material implication, program rules provide a directional behaviour very convenient for representing causal effects and, in particular, for avoiding the ramification and qualification problems in a straightforward way.

Traditionally, there has not been much work in combining Situation Calculus with LP under stable models semantics (as a pair of exceptions [37,38]). Perhaps the main reason for this was the need to resort to a ground theory, i.e., stable models was essentially a propositional formalism. This limitation, however, has drastically changed in the last years, thanks

[4] Many of them have been used in the literature, like *Clipped* [30], *Abnormal* [13,31], *Occluded* [32], *Causes* [33], *Affects* [7], etc.

[5] Consider, for instance, the minimal abnormality policy in the *Yale Shooting Problem* [13] or in the *Ramification Problem* as in [34], the chronological minimisation in the *Stanford Murder Mystery* [31], etc.

to the logical characterisation of stable models in terms of *Equilibrium Logic* [39] and, in particular, to its extension to first-order theories, *Quantified Equilibrium Logic* (QEL) [40]. As a result of this characterisation, the concept of stable model is now defined for any theory of predicate calculus with equality. In fact, stable models can be alternatively described by a second-order logic operator [41] similar to Circumscription.

In this section we proceed to introduce the main definitions of QEL. For a better understanding, it is crucial to bear in mind its intuitive connection to logic programming. In what follows, when talking about logic programs, we adopt a logical syntax notation, replacing the usual '$\leftarrow$', ',' and '*not*' operators respectively by implication, conjunction and negation. In this way, for instance, a typical logic program rule like:

$$p(X) \leftarrow r(X, Y), not\, q(Y)$$

would just be written as the formula $R(x, y) \wedge \neg Q(x) \rightarrow P(x)$ where all variables are implicitly universally quantified. This interpretation is extensible to Equilibrium Logic, where operator '$\neg$' can be informally understood as *default negation*.

Let $\mathcal{L} = \langle C, F, P\rangle$ be a first-order language where C is a set of *constants*, F a set of *functions* and P a set of *predicates*. First-order formulae for $\mathcal{L}$ are built up in the usual way, with the same syntax of classical predicate calculus. As in Intuitionistic Calculus, the formula $\neg\varphi$ will actually stand for $\varphi \rightarrow \bot$. We write $Atoms(C, P)$ to stand for the set of atomic sentences built with predicates in P and constants in C. Similarly, $Terms(C, F)$ denote the set of ground terms built from functions in F and constants in C. From now on, a formula with free variables is implicitly assumed to be preceded by their universal quantifications.

The definition of Equilibrium Logic has two steps: we start first from a (monotonic) intermediate logic and then define a model selection criterion on this logic.

We recall the definition introduced in [18] of the logic of *quantified here-and-there* with static domain, decidable equality and Herbrand structures, referring to it as the logic of *here and there* (*HT*) for short.

The *HT* logic is an intermediate logic. It can be defined in terms of the Intuitionistic Calculus plus axioms:

$$\begin{aligned}
&\alpha \vee \big(\neg\beta \vee (\alpha \rightarrow \beta)\big)\\
&\forall x\, \neg\neg\alpha(x) \rightarrow \exists x\, \big(\alpha(x) \rightarrow \forall x\, \alpha(x)\big)\\
&\neg\neg\exists x\, \alpha(x) \rightarrow \exists x\, \neg\neg\alpha(x)\\
&\forall x \forall y\ (x = y \vee x \neq y) \qquad \text{(DE)}
\end{aligned}$$

where the abbreviation $x \neq y$ stands for $\neg(x = y)$. In particular, (DE) is the axiom for *decidable equality* and states that the equality predicate behaves in a "classical" way, satisfying the law of the excluded middle.

The corresponding semantics for *HT* is described as follows.

Definition 1 *(HT-interpretation).* An *HT interpretation* for a language $\mathcal{L} = \langle C, F, P\rangle$ is a tuple $\langle (D, \sigma), H, T\rangle$ where:

1. D is a non-empty set of constant names identifying each element in the interpretation universe. For simplicity, we take the same name for the constant and the universe element.
2. $\sigma : Terms(D \cup C, F) \rightarrow D$ assigns a constant in D to any term built with functions in F and constants in the extended set of constants $C \cup D$. It must satisfy: $\sigma(d) = d$ for all $d \in D$.
3. H and T are sets of ground atomic sentences such that $H \subseteq T \subseteq Atoms(D, P)$.

Satisfaction of formulae is recursively defined as follows. Given an interpretation $\mathcal{M} = \langle (D, \sigma), H, T\rangle$, the following statements are true:

- $\mathcal{M} \models p(t_1, \ldots, t_n)$ if $p(\sigma(t_1), \ldots, \sigma(t_n)) \in H$.
- $\mathcal{M} \models t_1 = t_2$ if $\sigma(t_1) = \sigma(t_2)$.
- $\mathcal{M} \not\models \bot$.
- $\mathcal{M} \models \alpha \wedge \beta$ if $\mathcal{M} \models \alpha$ and $\mathcal{M} \models \beta$. Disjunction $\vee$ is analogous.
- $\mathcal{M} \models \alpha \rightarrow \beta$ if both:
  (i) $\mathcal{M} \not\models \alpha$ or $\mathcal{M} \models \beta$ and
  (ii) $\langle (D, \sigma), T\rangle \models \alpha \rightarrow \beta$ in classical logic.
- $\mathcal{M} \models \forall x\, \alpha(x)$ if for each $d \in D$, $\mathcal{M} \models \alpha(d)$ and $\langle (D, \sigma), T\rangle \models \alpha(d)$.
- $\mathcal{M} \models \exists x\, \alpha(x)$ if for some $d \in D$, $\mathcal{M} \models \alpha(d)$.

We say that $\mathcal{M}$ is a *model* of a theory Γ if $\mathcal{M}$ satisfies all the formulae in Γ.

An interpretation like $\langle (D, \sigma), T, T\rangle$ is said to be *total* and, moreover, may be seen as a classical interpretation $\langle (D, \sigma), T\rangle$. In fact, it is easy to check that:

Proposition 1. *$\langle (D, \sigma), T, T\rangle \models \Gamma$ iff $\langle (D, \sigma), T\rangle \models \Gamma$ in classical logic.*

Non-monotonic entailment is obtained by introducing a model-minimisation criterion. Let us define the following ordering relation among interpretations.

Definition 2. We say that an interpretation $\mathcal{M} = \langle (D, \sigma), H, T \rangle$ is *smaller* than an interpretation $\mathcal{M}' = \langle (D, \sigma), H', T \rangle$, written $\mathcal{M} \leqslant \mathcal{M}'$, when $H \subseteq H'$.

That is, to be comparable, $\mathcal{M}$ and $\mathcal{M}'$ must only differ in their H component, so that $\mathcal{M} \leqslant \mathcal{M}'$ iff $H \subseteq H'$. As usual, we write $\mathcal{M} < \mathcal{M}'$ when $\mathcal{M} \leqslant \mathcal{M}'$ and $\mathcal{M} \neq \mathcal{M}'$ (that is $H \subset H'$).

The next definition introduces the idea of minimal models for the *HT* logic.

Definition 3 *(Equilibrium model).* A total model $\mathcal{M}$ of a theory Γ is an *equilibrium model* if there is no smaller model $\mathcal{M}' < \mathcal{M}$ of Γ.

Note that an equilibrium model is total, i.e., it has the form $\mathcal{M} = \langle (D, \sigma), T, T \rangle$ and can be seen as a classical interpretation $\mathcal{M} = \langle (D, \sigma), T \rangle$. We name *Quantified Equilibrium Logic* (**QEL**) the logic induced by equilibrium models.

A *Herbrand HT-interpretation* $\langle (D, id), H, T \rangle$ is such that $D = Terms(C, F)$ and $\sigma = id$, where id is the identity relation. In the rest of the paper, we implicitly assume that we handle Herbrand *HT* models and Herbrand equilibrium models.

As usual, a *literal* is an atom $p(\mathbf{t})$ or its negation $\neg p(\mathbf{t})$. A *logic program* is a conjunction of implications $\alpha \rightarrow \beta$ where α (the *body*) is a conjunction of literals, β (the *head*) is a disjunction of literals, and all variables are universally quantified.

Proposition 2. *(From [18].)* $\mathcal{M} = \langle (D, \sigma), T, T \rangle$ *is a Herbrand equilibrium model of a logic program* Π *iff* T *is a stable model of the (possibly infinite) ground program* $gr_D(\Pi)$ *obtained by replacing all variables by all terms in* D *in all possible ways.*

Although QEL is defined for any arbitrary first-order theory, we will use in most cases a fragment of QEL that fits into the syntactic class of logic program rules. Furthermore, the rest of the formulae we use that do not belong to this class can be translated into LP, although their direct QEL representation is much more compact or readable and requires fewer auxiliary predicates. When specifying these translations we will say that two theories Γ_1, Γ_2 are strongly equivalent with respect to a signature $\mathcal{L}$, written $\Gamma_1 \equiv_s^{\mathcal{L}} \Gamma_2$ iff for any theory Γ in that signature, the sets of equilibrium models of $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$, when restricted to signature $\mathcal{L}$, coincide. The idea of strong equivalence is that it guarantees replacing Γ_1 by Γ_2 and vice versa, regardless of the context Γ. In [18] it was shown that checking regular equivalence in the logic of Quantified Here and There is a necessary and sufficient condition for $\Gamma_1 \equiv_s^{\mathcal{L}} \Gamma_2$, provided that Γ_1 and Γ_2 also belong to the signature $\mathcal{L}$.

Proposition 3. $\varphi \rightarrow \psi \wedge \gamma$ *is HT-equivalent to* $(\varphi \rightarrow \psi) \wedge (\varphi \rightarrow \gamma)$.

This proposition states that we can "compress" two program rules like $\varphi \rightarrow \psi$ and $\varphi \rightarrow \gamma$ into a formula $\varphi \rightarrow \psi \wedge \gamma$ with a conjunction in the consequent.

Proposition 4. $\neg\varphi$ *is the formula* $\varphi \rightarrow \bot$ *by definition.*

In fact, this is the definition of $\neg$ as a derived operator. It asserts that a negated formula $\neg\varphi$ can be understood or replaced by a constraint with φ in its body.

Proposition 5. $\varphi \rightarrow x = y$ *is HT-equivalent to* $\varphi \wedge x \neq y \rightarrow \bot$.

This proposition is saying that a rule with an equality in the head can be understood as a constraint (a rule with $\bot$ head) where the negation of $x = y$ has been moved to the antecedent.

Proposition 6. *Let* Γ_1 *be a theory consisting of the single formula*

$$\alpha(\mathbf{x}) \wedge \neg\exists \mathbf{y}\, \beta(\mathbf{x}, \mathbf{y}) \rightarrow \gamma(\mathbf{x}) \tag{1}$$

for language $\mathcal{L}$*, being* $\mathbf{x}$ *a tuple with all the variables that occur free in the antecedent or in the consequent. Then* $\Gamma_1 \equiv_s^{\mathcal{L}} \Gamma_2$ *where* Γ_2 *is the pair of formulae:*

$$\alpha(\mathbf{x}) \wedge \neg aux(\mathbf{x}) \rightarrow \gamma(\mathbf{x}), \tag{2}$$

$$\alpha(\mathbf{x}) \wedge \beta(\mathbf{x}, \mathbf{y}) \rightarrow aux(\mathbf{x}) \tag{3}$$

and $aux(\mathbf{x})$ *is a fresh auxiliary predicate not included in* $\mathcal{L}$.

In other words, this proposition[6] asserts that we can handle the negation of an existentially quantified formula in a rule body by introducing a new auxiliary predicate. For example, the formula:

$$Person(x) \wedge \neg(\exists y)\big(Person(y) \wedge Parent(x, y)\big) \rightarrow Orphan(x)$$

is strongly equivalent to:

$$Person(x) \wedge \neg aux(x) \rightarrow Orphan(x)$$

$$Person(x) \wedge Person(y) \wedge Parent(x, y) \rightarrow aux(x)$$

where $aux(x)$ is taking the intuitive meaning "x has some parent."

The next section develops Situation Calculus in Quantified Equilibrium Logic.

4. Situation Calculus theories in equilibrium logic

We introduce next a version of the Situation Calculus which essentially maintains its standard syntax [17,6], but replaces Predicate Calculus by Quantified Equilibrium Logic as the underlying logical framework. Our formalisation will be multi-sorted. We define a finite set of action names `Action` plus a finite set of fluent names `Fluent`. A *situation* is defined in the usual way, that is, it can be the constant S_0 (the *initial* situation) or a term like $do(a, s)$ where a is an action and s a situation in its turn. We will use variables a, a' for the sort `Action`, s, s' for the sort `Situation` and f, f' for `Fluent`.

Notice that, as we deal with Herbrand models, axioms for unique names assumption are not needed. However, we do include Domain Closure axioms for all sorted variables. For instance, for any variable a ranging on sort `Action` we have:

$$(\exists \mathbf{x}_1)\big[a = A_1(\mathbf{x}_1)\big] \vee \cdots \vee (\exists \mathbf{x}_n)\big[a = A_n(\mathbf{x}_n)\big], \tag{DC}$$

where the A_i are all the action names included in `Action` and each $\mathbf{x}_i$ is a tuple of variables whose cardinality is the arity of A_i. Analogous axioms are included for the `Fluent` sort.

All fluents will be *functional*, that is, each $f(\mathbf{x})$ has a unique value v among a range of possible values we denote $range(f)$. The predicate $Holds(f(\mathbf{x}), v, s)$ represents that the fluent $f(\mathbf{x})$ is assigned a value $v \in range(f)$ at a situation s. In order to keep an homogeneous notation, a Boolean fluent f will also be considered as functional with $range(f) = \texttt{Boolean} = \{True, False\}$. As we will see later, this will be especially convenient[7] for specifying the inertia default.

Since the fluent stands for a function, it will have a unique value:

$$Holds\big(f(\mathbf{x}), v, s\big) \wedge Holds\big(f(\mathbf{x}), v', s\big) \rightarrow v = v'. \tag{UV}$$

Although in Situation Calculus we can always build formulae with *any* situation term, we will be usually interested in describing some situations that are unfeasible, as some of their occurring actions cannot be executed. This is typically accomplished by a predicate $Poss(a, s)$ to point out when an action a is *possible* in a situation s. This predicate is used in so-called *unfeasibility axioms*, intuitively describing conditions under which $Poss(a, s)$ is false, and in *effect axioms*, where $Poss(a, s)$ is used in the antecedent of an implication, acting as one more condition for the causal rule to be applied. Note that this predicate should be true by default, that is, we only want to specify when an action cannot be executed, rather than explicitly stating all the cases in which the action *can* be performed. However, as in Equilibrium Logic all predicates are false by default, we use here the predicate $Impossible(a, s)$ stating the exact opposite, that is, performing the action a is *impossible* at the situation s.

A fluent may have a designated default value $default(f) = d$, $d \in range(f)$. In order to establish the effect of this default value, we further divide the sort `Fluent` into two disjoint subsets: `Inert` (*inertial* fluents) and `NonInert` (*non-inertial* fluents or *events*). For any non-inertial fluent, $f \in \texttt{NonInert}$ with $default(f) = d$, we include the axiom:

$$\neg(\exists v)\big[v \neq d \wedge Holds\big(f(\mathbf{x}), v, s\big)\big] \rightarrow Holds\big(f(\mathbf{x}), d, s\big) \tag{DV}$$

that intuitively captures the meaning of its default value for any situation s. In the case of inertial fluents, however, the default fluent value will be its value in a previous situation (this is the *inertia* assumption):

$$\neg Impossible(a, s) \wedge Holds\big(f(\mathbf{x}), v, s\big) \wedge \neg Released\big(f(\mathbf{x}), s\big)$$
$$\wedge \neg(\exists w)\big[w \neq v \wedge Holds\big(f(\mathbf{x}), w, do(a, s)\big)\big] \rightarrow Holds\big(f(\mathbf{x}), v, do(a, s)\big). \tag{INE}$$

In other words, when an action execution is possible, $\neg Impossible(a, s)$, if a fluent $f(\mathbf{x})$ had value v in the previous situation s and there is no evidence that the fluent has a different value w in the next situation $do(a, s)$ then the fluent value v remains unchanged in $do(a, s)$. The additional condition $\neg Released(f(\mathbf{x}), s)$ is included so that, when convenient, we can

[6] See the recent works [42] and [43] for a detailed description on this technique of removing existential quantifiers from the rule bodies.

[7] Most logic programming formalisations of inertia for boolean fluents make use of a second negation called strong or explicit. In this sense, an atom like $Holds(f(\mathbf{x}), False, s)$ can be seen as the strong negation of $f(\mathbf{x})$ at situation s. The advantage of our representation is that the inertia axiom is the same for Boolean and non-Boolean fluents.

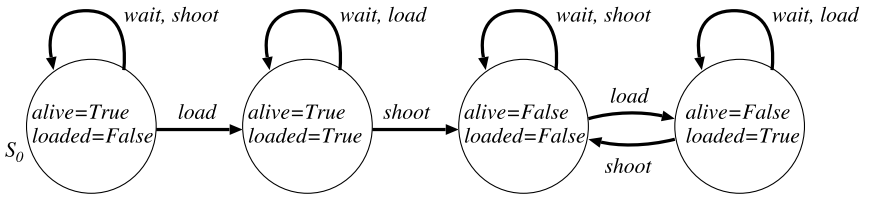


Fig. 5. State transition system for the Yale Shooting scenario.

"release" a given inertial fluent $f(\mathbf{x})$ from following the inertia default at a particular situation s. Predicate *Released* will be particularly useful for forcing a fluent $f(\mathbf{x})$ to become undefined (that is, it has no associated value), something we will represent with the negation of the following derived predicate:

$$\mathit{Defined}\big(f(\mathbf{x}), s\big) \leftrightarrow (\exists v)\mathit{Holds}\big(f(\mathbf{x}), v, s\big). \tag{4}$$

As we can see, Axiom (INE) does not establish any default assumption for the fluent value at the initial situation S_0. The fluent default value, when provided, is used for that case. In other words, when f is inertial and has some $\mathit{default}(f) = d$ we include the following version of (DV):

$$\neg(\exists v)\big[v \neq d \wedge \mathit{Holds}\big(f(\mathbf{x}), v, S_0\big)\big] \rightarrow \mathit{Holds}\big(f(\mathbf{x}), d, S_0\big). \tag{DV$_0$}$$

For a compact description of a fluent, we use the notation[8] $f : D \rightarrow R = d$ where D is the fluent domain, R the range and '$= d$' is an optional specification of a default value $\mathit{default}(f) = d$.

We write **ESC** (for *Equilibrium Situation Calculus*) to denote this multi-sorted version of **QEL** including the set of axioms seen before in this section: (DC), (UV), (DV), (INE) and (DV$_0$).

Since our spatial theory will deal with a relatively complex representation, it may be convenient to consider first a classical actions' scenario: the Yale Shooting Scenario [13], for a better understanding of notation and semantics. It is worth pointing out that the solution of this scenario in **ESC** shows how the semantics of Equilibrium Logic, together with the Situation Calculus, solves the Frame Problem in a rigorous way. This solution is inherited by our formalisation of the Fisherman's Folly puzzle, developed below.

Example 1. Consider the classical Yale Shooting Scenario [13] where we have a gun that must be previously loaded in order to shoot and kill a turkey. Initially the gun is not loaded and the turkey alive. We load the gun, wait one situation, and then shoot.

The formalisation of Example 1 would include the sorts $\texttt{Action} = \{\mathit{load}, \mathit{wait}, \mathit{shoot}\}$ and $\texttt{Inert} = \{\mathit{loaded}, \mathit{alive}\}$, with both fluents of range `Boolean` and no default value. The theory **YALE** would consist of **ESC** plus the formulae:

$$\mathit{Holds}(\mathit{loaded}, \mathit{True}, s) \rightarrow \mathit{Impossible}(\mathit{load}, s), \tag{5}$$

$$\mathit{Holds}(\mathit{loaded}, \mathit{False}, s) \rightarrow \mathit{Impossible}(\mathit{shoot}, s), \tag{6}$$

$$\neg\mathit{Impossible}(\mathit{load}, s) \wedge s' = \mathit{do}(\mathit{load}, s) \rightarrow \mathit{Holds}\big(\mathit{loaded}, \mathit{True}, s'\big), \tag{7}$$

$$\neg\mathit{Impossible}(\mathit{shoot}, s) \wedge s' = \mathit{do}(\mathit{shoot}, s) \rightarrow \mathit{Holds}\big(\mathit{alive}, \mathit{False}, s'\big) \wedge \mathit{Holds}\big(\mathit{loaded}, \mathit{False}, s'\big), \tag{8}$$

$$\mathit{Holds}(\mathit{loaded}, \mathit{False}, S_0) \wedge \mathit{Holds}(\mathit{alive}, \mathit{True}, S_0). \tag{9}$$

Proposition 7. *Theory* **YALE** *has a unique equilibrium model whose atoms for Holds corresponds to the state transition system in Fig.* 5.

Proof. See Appendix B. □

As a result, the following is a **QEL** consequence of **YALE**:

$$\mathit{Holds}\big(\mathit{alive}, \mathit{False}, \mathit{do}\big(\mathit{shoot}, \mathit{do}\big(\mathit{wait}, \mathit{do}(\mathit{load}, S_0)\big)\big)\big)$$

(it suffices to check it in Fig. 5).

Having **ESC** at our disposal, next section formalises the Fisherman's Folly puzzle in Equilibrium Situation Calculus.

[8] The current formalisation for functional fluents with default values has been adapted from [44].

5. Formalising the puzzle

In [16] the puzzle entities were classified into three different sorts: *regular objects*, *holes* (actually, single-holed objects) and *long objects* that will be respectively named as `Regular`, `Hole` and `Long`. In the Fisherman's Folly domain, these sorts respectively consist of:

$$\begin{aligned}&\texttt{Regular} = \{\mathit{Disk}1, \mathit{Disk}2, \mathit{Base}\},\\ &\texttt{Hole} = \{\mathit{Ring}, \mathit{Sphere}1, \mathit{Sphere}2, \mathit{PostH}\}, \quad \text{and}\\ &\texttt{Long} = \{\mathit{Str}, \mathit{Post}\}.\end{aligned} \tag{10}$$

Notice that, for notational simplicity, we identify a hole, like the ring hole, with its host object, like *Ring*. The whole set of *physical objects* is simply denoted as $\texttt{Object} \stackrel{def}{=} \texttt{Regular} \cup \texttt{Hole} \cup \texttt{Long}$. We will also handle additional derived sorts. The sort of *sets of long objects* $\texttt{Longset} \stackrel{def}{=} 2^{\texttt{Long}}$ will be used to specify when a given object can pass through a hole currently crossed by several long objects. To understand the need for this sort, note for instance that the *Sphere*1 can pass through the ring, provided that the latter is not being currently crossed by the *Post*. We assume we have available the set predicates '$\in$' and '$\subseteq$' with their standard meaning (note that `Longset` is finite and the whole extension of these two predicates can be provided explicitly without requiring further axiomatisation). Similarly, we will use the union operator $\cup$ which can be fixed to its usual meaning in a similar way.

We will define the two sides of a hole (we will call them *faces*) or the two *tips* of a long object as follows:

$$\begin{aligned}&\texttt{Face} \stackrel{def}{=} \big\{\mathit{face}(h,+), \mathit{face}(h,-) \;\big|\; h \in \texttt{Hole}\big\};\\ &\texttt{Tip} \stackrel{def}{=} \big\{\mathit{tip}(x,+), \mathit{tip}(x,-) \;\big|\; x \in \texttt{Long}\big\},\end{aligned}$$

where *tip* and *face* are two new constructors or function symbols.[9] For the sake of compactness, we will use the abbreviations h^+, h^-, x^+ and x^- respectively standing for $\mathit{face}(h,+)$, $\mathit{face}(h,-)$, $\mathit{tip}(x,+)$ and $\mathit{tip}(x,-)$. In this way, for instance, the string has the two tips Str^+ and Str^- whereas the *Ring* has two faces denoted as Ring^+ and Ring^-. The predicate *Opposite* will be used to assert that two different hole faces correspond to the same hole:

$$\mathit{Opposite}\big(h^+, h^-\big) \wedge \mathit{Opposite}\big(h^-, h^+\big). \tag{11}$$

Finally, we define a pair of additional (finite) derived sorts to deal with links among objects. The sort $\texttt{Node} \stackrel{def}{=} \texttt{Tip} \cup \texttt{Hole} \cup \texttt{Regular}$ captures all the possible elements that can form a node of a link. The difference with respect to `Object` is that, for linking a long object x, we must use one of its tips x^+ or x^- as the link node. We also define $\texttt{Nodeset} \stackrel{def}{=} 2^{\texttt{Node}}$ that will be used to allow moving "bundles" of linked objects, and use $\in, \subseteq, \cup$ as we did for the sort `Longset`.

5.1. Spatial predicates and fluents

We begin defining an inertial fluent $\mathit{linked} \in \texttt{Inert}$:

$$\mathit{linked} : \texttt{Node} \times \texttt{Node} \to \texttt{Boolean} = \mathit{False}$$

whose meaning is self-explanatory. In our scenario, *linked* will always relate a tip to a regular or a holed object, and this information will not vary along time. We prefer to maintain this as a fluent, and not as a static predicate, to allow a future inclusion of possible actions that change the degree of connectivity between objects.

Formulae that describe a particular Fisherman's Folly puzzle configuration will be labelled as (FFn). For instance, the initial facts for *linked* in the puzzle would be:

$$\begin{aligned}\mathit{Holds}\big(\mathit{linked}\big(\mathit{Str}^-, \mathit{Disk}1\big), \mathit{True}, S_0\big) &\wedge \mathit{Holds}\big(\mathit{linked}\big(\mathit{Str}^+, \mathit{Disk}2\big), \mathit{True}, S_0\big)\\ &\wedge \mathit{Holds}\big(\mathit{linked}\big(\mathit{Post}^+, \mathit{PostH}\big), \mathit{True}, S_0\big)\\ &\wedge \mathit{Holds}\big(\mathit{linked}\big(\mathit{Post}^-, \mathit{Base}\big), \mathit{True}, S_0\big).\end{aligned} \tag{FF1}$$

As we can see in its declaration, *linked* will be *False* by default. If we do not provide further information for this predicate apart from (INE), axiom (DV) allows deriving that all the rest will be false. The following are some axioms for the fluent *linked*:

[9] Remember we handle Herbrand models.

$$Holds\big(linked(x,y),True,s\big) \rightarrow Holds\big(linked(y,x),True,s\big), \tag{12}$$

$$Holds\big(linked(x,y),True,s\big) \wedge Holds\big(linked(y,z),True,s\big) \rightarrow Holds\big(linked(x,z),True,s\big), \tag{13}$$

$$Holds\big(linked\big(x^{+},x^{-}\big),False,s\big). \tag{14}$$

They respectively assert that the relation *linked* is commutative, transitive, and that the two tips of a long object are never linked.[10]

In principle, information about what can or cannot pass through a hole will not vary along time. For this reason, we just consider a static predicate $CannotPass(x,h,y)$ to represent that the object x cannot pass through the hole h when the latter is currently crossed by the set of long objects y. In the Fisherman's Folly example we include the following facts:

$$\begin{aligned} &CannotPass(disk,sphere,\emptyset) \wedge CannotPass(disk,Ring,\emptyset)\\ &\qquad \wedge CannotPass(Base,Ring,\emptyset) \wedge CannotPass(Base,PostH,\emptyset)\\ &\qquad \wedge CannotPass(Base,sphere,\emptyset)\\ &\qquad \wedge CannotPass(Post,sphere,\emptyset) \wedge CannotPass(Post,PostH,\emptyset)\\ &\qquad \wedge CannotPass(sphere,PostH,\emptyset) \wedge CannotPass\big(sphere,sphere',\emptyset\big)\\ &\qquad \wedge CannotPass\big(sphere,Ring,\{Post\}\big)\\ &\qquad \wedge CannotPass(PostH,sphere,\emptyset)\\ &\qquad \wedge CannotPass(Ring,sphere,\emptyset) \end{aligned} \tag{FF2}$$

with *disk* varying on $\{Disk1, Disk2\}$ and *sphere*, $sphere'$ on $\{Sphere1, Sphere2\}$. Note, for instance, that under any circumstance, any string can be passed through any hole. Note also that the spheres can pass through the *Ring*, unless the *Ring* is crossed by the *Post*. In order to guarantee this latter constraint, the following two axioms are also included:

$$CannotPass(h,h,y); \tag{15}$$

$$CannotPass(x,h,y) \wedge y \subseteq z \rightarrow CannotPass(x,h,z), \tag{16}$$

respectively stating that a single holed object h cannot pass through itself, and that once x cannot pass through h when crossed by y, the same will hold for any superset of y. We also assert that if a long object x cannot pass through a hole, then none of its tips can do so by default:

$$CannotPass(x,h,y) \wedge \neg CanPass\big(x^{v},h,y\big) \rightarrow CannotPass\big(x^{v},h,z\big), \tag{17}$$

where *CanPass* is an auxiliary predicate used to assert exceptions to this rule.

It is worth pointing out that, as predicates are false by default and *CannotPass* is a "negative" predicate, we are assuming that objects can pass through holes as a default. Apparently, this does not seem to be well chosen for the Fisherman's Folly scenario, where there actually exist more cases in which objects cannot pass through holes than those that can – in other words, (FF2) would be smaller for a *CanPass* predicate. However, in a more complex scenario, it may be the case that we do not have this information in a complete way. If so, the advantage of our representation is that an hypothetical planner could go finding "optimistic" plans that, when tried on the physical object, could perhaps be unfeasible at some point of their execution, but could help us to find out a new fact for $CannotPass(x,h,y)$ we had not realised before. In this way, we could go refining our information for *CannotPass* until some obtained plan actually solves the physical problem.

5.2. Crossings and segments

When representing a long object, the relevant information we will be interested in is the set of hole *crossings*. It is important to notice that the same long object can cross the same hole several times and in different directions (this is especially common when the long object is flexible, like a string). If we understand long objects and holes as three-dimensional regions, a *crossing* is the overlapping region between a hole and a long object. We can define a *segment* as a maximal continuous portion of a long object not overlapping with any hole. An important assumption about long objects will be made:

For any long object x, there cannot be a segment of x connecting more than two hole crossings.

[10] In this paper we disregard handling circular long objects or, similarly, linking the negative and positive tips of a same long object. This possibility is left for future study.

Our representation of the domain, and in particular, of the state of a long object in a given situation, will rely on this concept of segment. Thus, when writing formulae, we will require a way to *refer* to a given segment. The representation of segments of a long object x will be done using *segment labels*. A segment label i is a term that must uniquely identify each segment in a long object x. In the textual or graphical explanations, we will represent a segment as $x:i$ where x is a long object and i a label.

Intuitively, a good candidate sort for segment labels could be the natural numbers $0, 1, 2, \ldots$, or equivalently their Peano-like representation 0, $s(0)$, $s(s(0))$, ... so that we use their ordering to follow the sequence of crossings in the long object from tip x^- to tip x^+. In this way, for instance, the string would have the four initial segments $Str:0$ (from *Disk*1 to *Sphere*1), $Str:1$ (from *Sphere*1 to *PostH*), $Str:2$ (from *PostH* to *Sphere*2) and $Str:3$ (from *Sphere*2 to *Disk*2). An important problem, however, is that the set of segments will vary along time depending on the actions performed. When segments disappear this is not a real problem, as this just means that we will "stop using" a label in the relevant fluents that represent the long object state. Unfortunately, we also need to *create* new segments, which sometimes must be included *between* two existing segments. In this way, we should be able to partition a segment into new fragments as many times as needed. One possibility could be using rational numbers to this aim.[11] However, in order to avoid their axiomatisation, we adopt a simpler choice. We will use four new Herbrand functions or label constructors $pred(i)$, $mid1(i,j)$, $mid2(i,j)$ and $succ(i)$, so that:

- $x:pred(i)$ denotes a predecessor segment to $x:i$;
- $x:succ(i)$ denotes a successor segment to $x:i$ and
- $x:mid1(i,j)$ and $x:mid2(i,j)$ denote a pair of segments that are positioned in this same ordering, in the middle of $x:i$ and $x:j$.

We assume that there exists a way to assign a different label to each object segment occurring in the initial situation S_0. Let us call `InitLabel` this finite set of initial labels. The sort `Label` is recursively defined as: $l \in \texttt{Label}$ iff $l \in \texttt{InitLabel}$ or l has any of the forms $pred(i)$, $succ(i)$, $mid1(i,j)$ or $mid2(i,j)$ with $i, j \in \texttt{Label}$.

As said before, we will represent the list of segments and crossings following the chain from the negative tip x^- to the positive one x^+. To this aim we can imagine a linear graph where each node is a segment and each arc is labelled with the crossing that connects both segments, represented by the corresponding outgoing hole face. Following this notation, the initial situation of the string *Str* would correspond to the graph:

$$Str:0 \overset{Sphere1^+}{\longrightarrow} Str:1 \overset{PostH^+}{\longrightarrow} Str:2 \overset{Sphere2^+}{\longrightarrow} Str:3.$$

The representation of this graph will be done with a pair of inertial fluents:

$$next : \texttt{Long} \times \texttt{Label} \to \texttt{Label} \cup \{End\}$$

$$towards : \texttt{Long} \times \texttt{Label} \to \texttt{Face}$$

so that, for instance, a pair of facts like

$$Holds\big(next(Str,0),1,s\big) \wedge Holds\big(towards(Str,0),Sphere1^+,s\big)$$

represent the labelled graph edge $Str:0 \overset{Sphere1^+}{\longrightarrow} Str:1$. For convenience, the last segment in the list will further point to a special constant *End* – in our example, $Holds(next(Str,3),End,S_0)$.

It must be noticed that not any possible pair $x:i$ will form a real object segment in any situation. For instance, in the goal situation, the long object *Post* does not cross any hole and so, it has a unique segment, labelled in this case as $Post:0$. Thus, for instance, there are no segments $Post:1$ or $Post:2$ in that situation, although 1 and 2 are possible labels. The "existence" of a segment $x:i$ at a given situation s is captured by the fact that no atom $Holds(next(x,i),z,s)$ or $Holds(towards(x,i),y,s)$ is true at that situation, i.e., the segment *is not referenced as argument of any true atom*.

The initial situation in the puzzle can be represented now by the formula:

$$\begin{aligned} &Holds\big(next(Str,0),1,S_0\big) \wedge Holds\big(towards(Str,0),Sphere1^+,S_0\big) \\ &\quad \wedge Holds\big(next(Str,1),2,S_0\big) \wedge Holds\big(towards(Str,1),PostH^+,S_0\big) \\ &\quad \wedge Holds\big(next(Str,2),3,S_0\big) \wedge Holds\big(towards(Str,2),Sphere2^+,S_0\big) \\ &\quad \wedge Holds\big(next(Str,3),End,S_0\big) \\ &\quad \wedge Holds\big(next(Post,0),1,S_0\big) \wedge Holds\big(towards(Post,0),Ring^+,S_0\big) \\ &\quad \wedge Holds\big(next(Post,1),End,S_0\big) \end{aligned} \tag{FF3}$$

[11] Another possibility we have not explored is the use of existential quantifiers, but this would surely mean a more difficult reading and longer formulae to guarantee that segment labels are pairwise different.

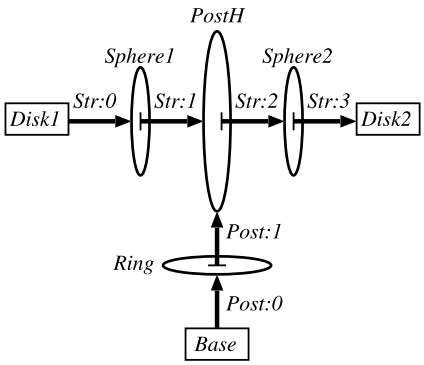


Fig. 6. Schematic representation of the initial state showing the long object segments.

where, as we can see in the fifth line of the previous statement, the post has also a crossing (towards $Ring^+$) and so it has been divided into two segments, $Post:0$ and $Post:1$. We include the following axiom, to check that in S_0 no segment $x:k$ is preceded by two different crossings:

$$Holds\big(next(x,i),k,S_0\big) \wedge Holds\big(next(x,j),k,S_0\big) \rightarrow i = j. \tag{18}$$

At this point we would like to recall the schematic representation introduced in Section 2 (Fig. 3), reproduced in Fig. 6 for convenience. Note the representation of segments: $x:i$.

An additional non-inertial fluent (*from* below) will be useful for specifying action preconditions in a more readable[12] way.

$$from : \texttt{Long} \times \texttt{Label} \rightarrow \texttt{Face}.$$

Its meaning is fixed by the axiom:

$$Holds\big(next(x,j),i,s\big) \wedge Holds\big(towards(x,j),p,s\big) \rightarrow Holds\big(from(x,i),p,s\big) \tag{19}$$

that is, *towards* and *from* respectively indicate the next and the previous hole crossing of a given segment $x:i$.

5.3. Actions

From a physical point of view, the solution to the Fisherman's Folly puzzle can be exclusively described in terms of actions for passing objects through holes in a given direction. In principle, this would lead us to consider some action like $pass(x,p)$ meaning that we pass object x through some hole towards its face p, such as for instance, $pass(Ring, PostH^+)$. Although in principle x could be of any type $\texttt{Regular} \cup \texttt{Long} \cup \texttt{Hole}$, in practice some objects are not relevant to be passed through a hole, w.r.t. the puzzle's states. For example, assume we have a disconnected disk and we decide to pass it through the post hole: it is not difficult to see that this action has *no effect* on our state representation at all. Generally speaking, passing a regular object x through a hole is only really relevant when x *is linked to some tip* of a long object. For this reason, we disregard the case $x \in \texttt{Regular}$ in favour of passing long object tips instead, $x \in \texttt{Tip}$. Note that, by passing tips, we can also disregard $pass(x,p)$ for a whole long object $x \in \texttt{Long}$, as it can be replaced by a sequence of $pass(x^+,p), pass(x^-,p)$ or $pass(x^-,p), pass(x^+,p)$. The only remaining possibility is $pass(x,p)$ for some $x \in \texttt{Hole}$, that is, some single holed object. In fact, the most difficult part of the formalisation has to do with the case in which x is currently crossed by one or more segments of one or more long objects.

Apart from individually passing tips and holed objects, we must also bear in mind one more feature that is present in the problem: linked objects. In our puzzle, we have four pairs of linked elements: three of them are linking tips to regular objects, but one is linking a tip to a holed object. It is not difficult to imagine other scenarios (see, for instance, the Tricky Dick puzzle discussed in Appendix C) where we may also link two tips of (different) long objects or two linked holed objects. When elements are linked one each other, passing them through a hole implies passing each of their individual components. We will define the concept of *object bundle* as a (maximal) set of linked objects. The non-inertial fluent:

$$bundle : \texttt{Nodeset} \rightarrow \texttt{Boolean} = True,$$

will be used to reflect that a given set of objects B forms a bundle at some situation. This fluent is *True* by default because it is easier to specify when an arbitrary set of objects b *does not form* a bundle. Its meaning is captured by the following axioms:

[12] In fact, references to this additional fluent in action preconditions could be replaced by a suitable use of quantifiers.

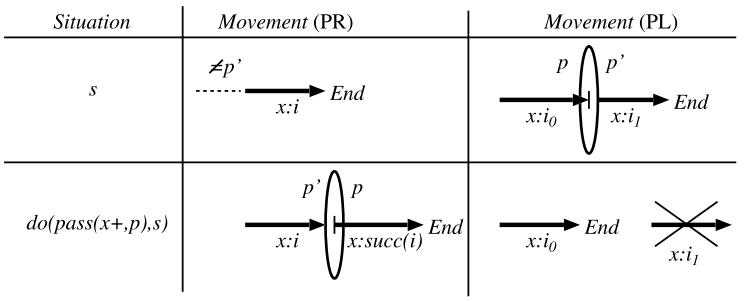


Fig. 7. Movements for passing the positive tip $pass(x^+, p)$.

$$x \in b \wedge y \in b \wedge Holds\big(linked(x, y), False, s\big) \rightarrow Holds\big(bundle(b), False, s\big), \tag{20}$$

$$x \in b \wedge \neg(y \in b) \wedge Holds\big(linked(x, y), True, s\big) \rightarrow Holds\big(bundle(b), False, s\big), \tag{21}$$

$$Holds\big(bundle(\emptyset), False, s\big). \tag{22}$$

Axiom (20) asserts that the set b does not form a bundle when it contains two unlinked objects x and y. Axiom (20) captures the bundle maximality: a bundle b cannot contain an object x linked to an external element y not in b. Finally, (22) asserts that the empty set cannot be a bundle.

Example 2. Axioms for *bundle* (20)–(22) and *linked* (12)–(14) plus the initial state of *linked* in the Fisherman's Folly puzzle (FF1) and the default values for *linked* and *bundle* allow deriving the facts:

$$\begin{aligned} &Holds\big(bundle\big(\{Str^-, Disk1\}\big), True, S_0\big) \wedge Holds\big(bundle\big(\{Str^+, Disk2\}\big), True, S_0\big) \\ &\qquad \wedge Holds\big(bundle\big(\{Post^-, Base\}\big), True, S_0\big) \\ &\qquad \wedge Holds\big(bundle\big(\{Post^+, PostH\}\big), True, S_0\big) \end{aligned}$$

and no other bundle is formed at situation S_0.

To sum up, our main action will have the form $pass(b, p)$ meaning that we pass the bundle b towards the hole face p. We will first describe the *unfeasibility axioms*, that is, conditions for which it is impossible to perform action $pass(b, p)$ at a given situation. To this aim, we define an auxiliary non-inertial fluent:

$$crossedBy : \texttt{Hole} \times \texttt{Longset} \rightarrow \texttt{Boolean} = False,$$

with axioms

$$Holds\big(towards(x, i), h^v, s\big) \rightarrow Holds\big(crossedBy\big(h, \{x\}\big), True, s\big), \tag{23}$$

$$Holds\big(crossedBy(h, a), True, s\big) \wedge Holds\big(crossedBy(h, b), True, s\big) \rightarrow Holds\big(crossedBy(h, a \cup b), True, s\big). \tag{24}$$

In other words, *crossedBy* points out all the long object sets that are crossing a given hole h at situation s. The unfeasibility conditions are then specified as follows:

$$Holds\big(bundle(b), False, s\big) \rightarrow Impossible\big(pass(b, p), s\big), \tag{25}$$

$$x \in b \wedge Holds\big(crossedBy(h, a), True, s\big) \wedge CannotPass(x, h, a) \rightarrow Impossible\big(pass\big(b, h^v\big), s\big). \tag{26}$$

Axiom (25) asserts that we cannot pass a set b that does not form a bundle. Axiom (26) says that a bundle b cannot pass through a hole h if b contains some node[13] x that cannot currently pass through h.

Let us proceed now with the *effect axioms*, that is, the causal rules describing effects of an action. To this aim, the effects and constraints for passing a tip or a holed object inside a bundle are described separately. For instance, for passing a positive tip x^+ to some face p, we get:

$$\begin{aligned} \neg Impossible\big(pass(b, p), s\big) \wedge x^+ \in b \wedge s' = {} & do\big(pass(b, p), s\big) \wedge Opposite\big(p, p'\big) \\ & \wedge Holds\big(next(x, i), End, s\big) \wedge \neg Holds\big(from(x, i), p', s\big) \\ \rightarrow {} & Holds\big(next(x, i), succ(i), s'\big) \wedge Holds\big(towards(x, i), p, s'\big) \\ & \wedge Holds\big(next\big(x, succ(i)\big), End, s\big) \wedge Released\big(towards(x, i), s\big). \end{aligned} \tag{PR}$$

[13] To be precise, we could modify *CannotPass* to reflect when a *set* of objects cannot pass altogether through a hole currently crossed by another set of objects, but this is not really needed for the current puzzles and is disregarded in this paper.

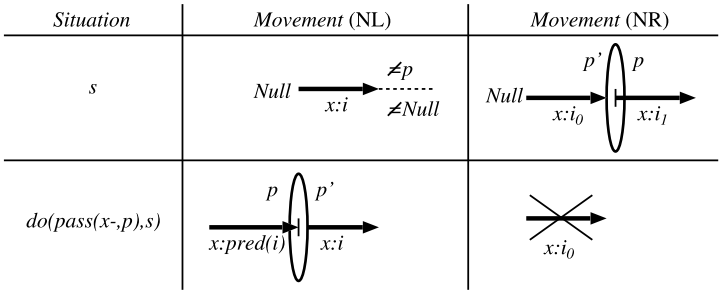


Fig. 8. Movements for passing the negative tip $pass(x^-, p)$.

Axiom name (PR) stands for *Positive tip to "Right"*. This axiom asserts that if $x : i$ is the last segment and there is no preceding crossing through the same hole in the opposite direction p', then we add a new crossing from $x : i$ passing through p to a new (last) segment $x : succ(i)$. The analogous movement for a positive tip (*Positive tip to Left*) would be covered by:

$$\begin{aligned}\neg Impossible\big(pass(b,p),s\big) \wedge x^+ \in b \wedge s' \;=\; & do\big(pass(b,p),s\big) \wedge Opposite\big(p,p'\big) \wedge Holds\big(next(x,i_0),i_1,s\big)\\ & \wedge Holds\big(towards(x,i_0),p',s\big) \wedge Holds\big(next(x,i_1),End,s\big)\\ & \rightarrow Holds\big(next(x,i_0),End,s'\big) \wedge Released\big(towards(x,i_0),s\big)\\ & \wedge Released\big(next(x,i_1),s\big) \qquad \text{(PL)}\end{aligned}$$

which states that when we execute $pass(x^+, p)$ but the last crossing of x was in the opposite direction p', we must remove this crossing and the last segment. Fig. 7 schematically shows the effects of movements (PR), (PL).

Analogous axioms are included for passing the tip x^-. Passing *Negative tip to Left* is stated as follows:

$$\begin{aligned}\neg Impossible\big(pass(b,p),s\big) \wedge x^- \in b \wedge s' \;=\; & do\big(pass(b,p),s\big) \wedge Opposite\big(p,p'\big)\\ & \wedge \neg Defined\big(from(x,i),s\big) \wedge Defined\big(next(x,i),s\big) \wedge \neg Holds\big(from(x,i),p,s\big)\\ & \rightarrow Holds\big(next\big(x,pred(i)\big),i,s'\big) \wedge Holds\big(towards\big(x,pred(i)\big),p',s'\big), \qquad \text{(NL)}\end{aligned}$$

that is, if i is the first segment (there are no previous crossings) and it is not followed by a crossing p then we add a new (first) segment $pred(i)$ passing through p' towards i. Note that we require $Defined(next(x, i), s)$ so that $x : i$ is an existing segment, i.e., it is followed by a crossing or by *End* (in this case, it would be the sole object segment). Finally, the action for passing the *Negative tip* of a long object to the *Right* of a hole is represented as follows:

$$\begin{aligned}\neg Impossible\big(pass(b,p),s\big) \wedge x^- \in b \wedge s' \;=\; & do\big(pass(b,p),s\big) \wedge Opposite\big(p,p'\big) \wedge Holds\big(next(x,i_0),i_1,s\big)\\ & \wedge Holds\big(towards(x,i_0),p,s\big) \wedge \neg Defined\big(from(x,i_0),s\big)\\ & \rightarrow Released\big(next(x,i_0),s\big) \wedge Released\big(towards(x,i_0),s\big) \qquad \text{(NR)}\end{aligned}$$

which means that, if $x : i_0$ is the initial segment (it has no previous crossing) and it points to $x : i_1$ through face p, then this crossing is removed, so $x : i_1$ implicitly becomes the first segment. Fig. 8 graphically represents the effects of (NL), (NR) – notice that the figure is almost symmetrical to Fig. 7.

We move now to consider the effects of passing a holed object h through another hole towards the outgoing face p of the latter. For instance, $pass(b, Ring^-)$ with $PostH \in b$ would pass the post hole towards the bottom side of the ring (in other words, slide the ring upwards along the post hole). The execution of this action will affect all crossings of any long object x towards any face h^v of h. Note that the same long object may cross the same hole several times and in different directions.

$$\begin{aligned}&\neg Impossible\big(pass(b,p),s\big) \wedge h \in b \wedge Opposite\big(p,p'\big) \wedge s'\\ &= do\big(pass(b,p),s\big) \wedge \neg Holds\big(from(x,i_0),p',s\big) \wedge Holds\big(next(x,i_0),i_1,s\big)\\ &\quad \wedge Holds\big(towards(x,i_0),h^v,s\big) \wedge \neg Holds\big(towards(x,i_1),p,s\big)\\ &\rightarrow Holds\big(next(x,i_0),mid1(i_0,i_1),s'\big) \wedge Holds\big(towards(x,i_0),p,s'\big) \wedge Holds\big(next\big(x,mid1(i_0,i_1)\big),mid2(i_0,i_1),s'\big)\\ &\quad \wedge Holds\big(towards\big(x,mid1(i_0,i_1)\big),h^v,s'\big)\\ &\quad \wedge Holds\big(next\big(x,mid2(i_0,i_1)\big),i_1,s'\big) \wedge Holds\big(towards\big(x,mid2(i_0,i_1)\big),p',s'\big). \qquad \text{(1R)}\end{aligned}$$

This first movement is applicable when segment $x : i_0$ crosses h towards $x : i_1$ but the former is not preceded by a crossing through p' (the opposite of p) and the latter is not followed by a crossing through p. The effect of $pass(b, p)$ in

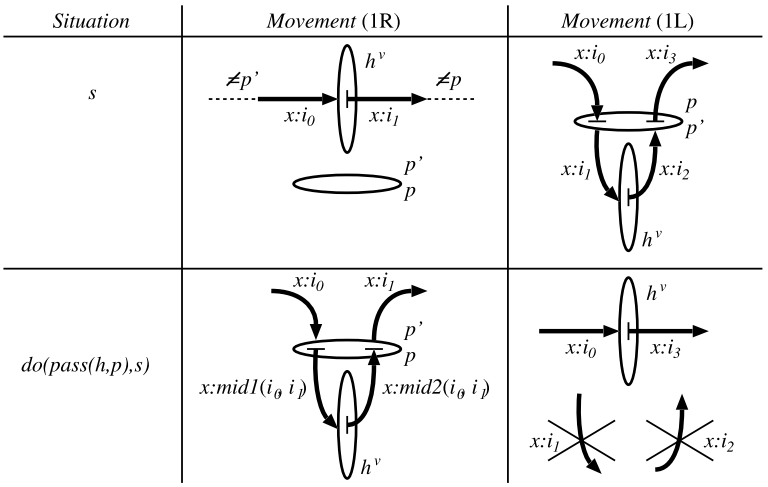


Fig. 9. Movements (1R), (1L) for action *pass*(*h*, *p*).

this part of the list is the insertion of two new crossings, one through p before h and one after h through p'. This also implies the creation of two new segments labelled with $mid1(i_0, i_1)$ and $mid2(i_0, i_1)$. The opposite movement is captured by the axiom:

$$\begin{aligned}
&\neg Impossible(pass(b, p), s) \wedge h \in b \wedge Opposite(p, p') \\
&\quad \wedge s' = do(pass(b, p), s) \wedge Holds(next(x, i_0), i_1, s) \wedge Holds(towards(x, i_0), p', s) \\
&\quad \wedge Holds(next(x, i_1), i_2, s) \wedge Holds(towards(x, i_1), h^v, s) \\
&\quad \wedge Holds(next(x, i_2), i_3, s) \wedge Holds(towards(x, i_2), p, s) \\
&\rightarrow Holds(next(x, i_0), i_3, s') \wedge Holds(towards(x, i_0), h^v, s') \\
&\quad \wedge Released(next(x, i_1), s) \wedge Released(towards(x, i_1), s) \\
&\quad \wedge Released(next(x, i_2), s) \wedge Released(towards(x, i_2), s). \qquad (1L)
\end{aligned}$$

When we have a sequence of crossings p', h^v, p (with p' opposite face of p), action *pass*(h, p) implies removing crossings p' and p from the list. Movements (1R), (1L) are schematically represented in Fig. 9.

The remaining movements do not create or destroy segments, but just swap crossings:

$$\begin{aligned}
&\neg Impossible(pass(b, p), s) \wedge h \in b \wedge Opposite(p, p') \\
&\quad \wedge s' = do(pass(b, p), s) \wedge \neg Holds(from(x, i_0), p', s) \wedge Holds(next(x, i_0), i_1, s) \\
&\quad \wedge Holds(towards(x, i_0), h^v, s) \wedge Holds(towards(x, i_1), p, s) \\
&\rightarrow Holds(towards(x, i_0), p, s') \wedge Holds(towards(x, i_1), h^v, s'), \qquad (2R)
\end{aligned}$$

$$\begin{aligned}
&\neg Impossible(pass(b, p), s) \wedge h \in b \wedge Opposite(p, p') \\
&\quad \wedge s' = do(pass(b, p), s) \wedge Holds(next(x, i_0), i_1, s) \wedge Holds(towards(x, i_0), p', s) \\
&\quad \wedge Holds(next(x, i_1), i_2, s) \wedge Holds(towards(x, i_1), h^v, s) \wedge \neg Holds(next(x, i_2), p, s) \\
&\rightarrow Holds(towards(x, i_0), h^v, s') \wedge Holds(towards(x, i_1), p', s'). \qquad (2L)
\end{aligned}$$

The graphical representation of these movements is shown in Fig. 10.

The goal can be specified as a ground situation s that satisfies the formula

$$GOAL(s) \stackrel{def}{=} \neg(\exists x, i, v)\big[Holds(towards(x, i), Ring^v, s)\big]$$

which is entailed from our scenario description. $GOAL(s)$ asserts that the *Ring* hole is not referenced in any crossing and so, is completely free.

We define the *Holes and Strings Theory*, **HST**, as the set of formulae **ESC** $\cup$ {(11)–(26), (PR), (PL), (NR), (NL), (1L), (1R), (2L), (2R)}. An **HST** *scenario* consists of axioms in **HST** plus the definition of sorts `Regular`, `Hole`, `Long` and domain dependent

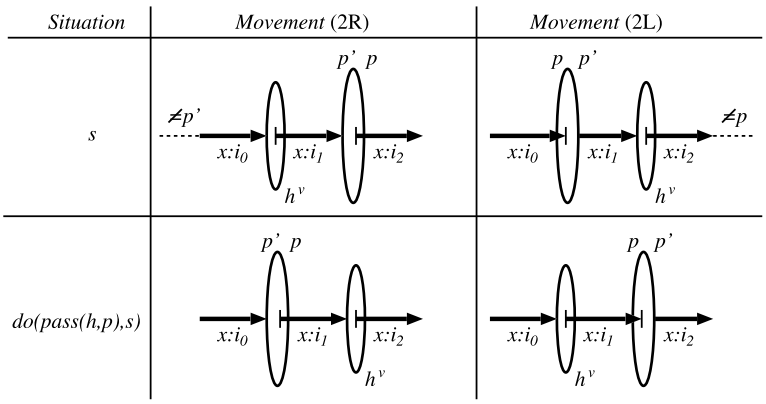


Fig. 10. Movements (2R), (2L) for action $pass(h, p)$.

axioms containing a conjunction of ground atoms for predicate *CannotPass* and initial values of *linked*, *next* and *towards*. In particular, we define the Fisherman's Folly scenario, **FOLLY**, by adding to **HST** the sorts in (10) and the domain dependent axioms (FF1), (FF2) and (FF3).

The following proposition shows that the method for labelling new segments works as expected.

Proposition 8. *Let M be an Equilibrium model of an* **HST** *scenario and let $L(s) = \{i \mid M \models Defined(next(x, i), s)\}$, that is, the labels of "existing" segments at situation s. Then, if the antecedent of* (PR) *holds in M, the composed label referred in its consequent $succ(i)$ does not belong to $L(s)$. The same happens for* (NL) *with new label $pred(i)$ and for* (1R) *with the new labels $mid1(i_0, i_1)$ and $mid2(i_0, i_1)$.*

Proof sketch. This relies on the fact that we can establish a strict ordering relation among labels that coincides with their sequence in the long object for the initial situation, while for composed labels satisfies: if $pred(i) < i$, $i < mid1(i, j) < mid2(i, j) < j$ and $i < succ(i)$. □

An important observation is that, axioms in an **HST** scenario can be seen as logic program rules, as with the formalisation of the Yale Shooting scenario in Example 1. Furthermore, **HST** also preserves the same dependence ordering among situations: a predecessor situation s never depends on a successor one $do(a, s)$. So, a similar splitting technique can be used to prove the following results.

Proposition 9. *Let M be an equilibrium model of some* **HST** *scenario. Then, M satisfies* (18) *for any situation s*:

$$Holds\big(next(x, i), k, s\big) \wedge Holds\big(next(x, j), k, s\big) \rightarrow i = j.$$

In other words, we never get into a situation where two different crossings point to the same long object segment $x : k$.

Proposition 10. *Any* **HST** *scenario has at most a unique Equilibrium model.*

Proof sketch. As with **YALE**, if we carefully look at all movements, it can be checked that they induce a deterministic state transition system. □

In the case of **FOLLY**, the theory is consistent, and so has a unique equilibrium model capturing the whole transition state system of the puzzle.

The next section discusses the correspondence of the above formalisation with respect to a model of the domain based on list manipulation. We argue that this model guarantees the correctness of our rigorous account of the Fisherman's Folly puzzle.

6. The induced transition system: operations on lists of crossings

As we have seen the previous formalisation is not too far away from other typical uses of Situation Calculus, except that classical Predicate Calculus has been replaced by First-Order Equilibrium Logic as the underlying logical formalism. The use of the Situation Calculus allows us to express non-trivial properties of the system, while providing a high degree of flexibility and elaboration tolerance. In this way, our formalisation can cope with some simple modifications of the scenario like adding new constraints, or defining new actions. On the other hand, the use of Equilibrium Logic as the underlying logical formalism allows us to inherit most of the typical constructs from action representations using *Answer Set Programming* [45]

(ASP), but removing the ASP restriction to propositional theories. As a result, we get a compact representation of inertia (INE), and we may naturally deal with indirect effects, like Axiom (19), or handle action disqualifications, as shown with respect to the predicate *Impossible*.

Analogous to our investigation of the Yale Shooting Scenario in Fig. 5 (Section 4), we can also study the transition system induced by our logical representation. This may help for a better understanding of the system behaviour, and can also be used for an efficient implementation of a planning algorithm. On the other hand, it must be taken into account that a transition system is much less elaboration tolerant than the logical representation – for instance, it cannot adequately represent default values, inertia or indirect effects, so that small elaborations of the domain may drastically affect the transition system configuration.

In order to describe the transition system states, we have a problem that did not manifest in the Yale Shooting scenario, where all fluents were Boolean – we have an infinite set of states and, furthermore, their representation is hard due to the possibility of arbitrarily creating new segments. To avoid using segments in the transition system representation, we adopt the approach we followed in a previous work [16] (briefly introduced in Section 2) where a puzzle state was represented as an ordered list of crossings for each long object. Transitions were then specified as combinations of elementary transformations on these lists, inducing a transition system for which a simple forward-driven planner was designed. We will describe next the correspondence between the list-based solution of the puzzle and our present formalisation, proving the correctness of the planner.

In this section we use Greek letters $\alpha, \beta, \gamma, \ldots$ to denote lists and adopt the usual Prolog list notation. The expression $\alpha \cdot \beta$ represents the concatenation of lists α and β. Given an **HST** interpretation M and a situation s we will define its corresponding state as a set of lists $chain(x)$ for each long object $x \in \mathtt{Long}$. The formal definition of $chain(x)$ will be done in terms of an auxiliary list $cr(x, i)$, where i is one of the segment labels of x. This list cr collects all the crossings that follow a given long object segment $x:i$ (cf. the informal definitions presented in Section 2 above).

Definition 4 *(Crossing list).* Given a model M of an **HST** scenario, a segment $x:i$, and a ground situation s, we define the list $cr(x, i, s)$ recursively as follows:

$$cr(x,i,s) \stackrel{def}{=} \begin{cases} [p] \cdot cr(x,j,s) & \text{if } M \models Holds(next(x,i),j,s) \wedge Holds(towards(x,i),p,s) \\ [\,] & \text{if } M \models Holds(next(x,i), End, s). \end{cases}$$

The list $chain(x, s)$ just appends the left and right tip of x as first and last element of the list obtained by $cr(x, i, s)$, being $x:i$ the first segment of x. This is formally specified by:

Definition 5 *(Chain list).* Given a model M of an **HST** scenario, a long object x, and a ground situation s, we define the list $chain(x, s)$ as:

$$chain(x,s) \stackrel{def}{=} \left[x^-\right] \cdot cr(x,i,s) \cdot \left[x^+\right]$$

where i is such that $M \models \neg Defined(from(x,i),s) \wedge Defined(next(x,i),s)$, that is, the first segment label of x.

With these rules, it is easy to see that the state corresponding to the initial situation would be described by the pair of lists:

$$chain(Str, S_0) = \left[Str^-, Sphere1^+, PostH^+, Sphere2^+, Str^+\right],$$
$$chain(Post, S_0) = \left[Post^-, Ring^+, Post^+\right].$$

Now, we will show that the value of $chain(x, do(a, s))$ after a transition can be directly computed from $chain(s)$ using a pair of simple list transformation operators, one for each action type: passing tips $pass(t, p)$ and passing single-holed objects $pass(h, p)$.

Definition 6 *(Operator $opT_{t,p}$).* Let x be a long object, t one of its tips $t = x^v$, p a face and p' its opposite face. The transformation on list $\alpha = chain(x)$ caused by the action $pass(t, p)$, written $opT_{t,p}$, is defined as: $opT_{t,p} \stackrel{def}{=}$

$$\beta \cdot \left[y, p, x^+\right] \quad \text{if } t = x^+,\ \alpha = \beta \cdot \left[y, x^+\right],\ y \neq p', \tag{27}$$
$$\beta \cdot \left[x^+\right] \quad \text{if } t = x^+,\ \alpha = \beta \cdot \left[p', x^+\right], \tag{28}$$
$$\left[x^-, p', y\right] \cdot \beta \quad \text{if } t = x^-,\ \alpha = \left[x^-, y\right] \cdot \beta,\ y \neq p, \tag{29}$$
$$\left[x^-\right] \cdot \beta \quad \text{if } t = x^-,\ \alpha = \left[x^-, p\right] \cdot \beta, \tag{30}$$

where $\alpha \cdot \beta$ denotes the concatenation of lists α and β.

It is easy to see that the four conditions (27)–(30) respectively correspond to movements (PR), (PL), (NL) and (NR), introduced in Section 5.1. The remaining movements of the puzzle can be stated as shown in Definition 7 below.

Definition 7 *(Operator $opH_{h,p}$).* Let h be a hole, p a face and p' the opposite face of p. Given a list of crossings $\alpha = chain(z)$ for a long object z, the transformation on α caused by an action $pass(h, p)$, written $opH_{h,p}(\alpha)$, is recursively defined as: $opH_{h,p}(\alpha) \stackrel{def}{=}$

$$[x] \quad \text{if } \alpha = [x], \tag{31}$$

$$[x \mid \gamma] \quad \text{if } \alpha = \left[x, g^{v} \mid \beta\right],\ g \neq h,\ \gamma = opH_{h,p}\left(\left[g^{v} \mid \beta\right]\right), \tag{32}$$

$$\left[x, p, h^{v}, p' \mid \gamma\right] \quad \text{if } \alpha = \left[x, h^{v}, y \mid \beta\right],\ x \neq p',\ y \neq p,\ \gamma = opH_{h,p}\left([y \mid \beta]\right), \tag{33}$$

$$\left[h^{v} \mid \gamma\right] \quad \text{if } \alpha = \left[p', h^{v}, p \mid \beta\right],\ opH_{h,p}\left([p \mid \beta]\right) = [p \mid \gamma], \tag{34}$$

$$\left[x, p, h^{v} \mid \gamma\right] \quad \text{if } \alpha = \left[x, h^{v}, p \mid \beta\right],\ x \neq p',\ opH_{h,p}\left([p \mid \beta]\right) = [p \mid \gamma], \tag{35}$$

$$\left[h^{v}, p', y \mid \gamma\right] \quad \text{if } \alpha = \left[p', h^{v}, y \mid \beta\right],\ y \neq p,\ \gamma = opH_{h,p}\left([y \mid \beta]\right). \tag{36}$$

After a careful reading, it can be observed that conditions (33)–(36) correspond to movements (1R), (1L), (2R) and (2L). As we can see, *opH* operator processes triples x, h^{v}, y in the list whose central element is a crossing through the moved hole h. Note that these movements exhaust all the possibilities: x and y could be the target face p, its opposite face p' or none of them, but all the cases are covered in (33)–(36). Note also that the last element in a triple is being used as the first element of the next triple. This means that elementary movements can be combined, as shown by the following example, extracted[14] from a similar puzzle explained in Section 7.

Example 3 *(Combined movements).* Assume we have the list $\alpha =$:

$$\left[H1^{+}, Ring^{-}, H2^{+}, Ring^{+}, H1^{-}, Ring^{-}, H2^{+}\right].$$

The result corresponding to $pass(Ring, H2^{+})$, that is, $opH_{Ring,H2^{+}}(\alpha)$, is computed as shown below. We just show relevant steps involving some change in the list. At each step, we show the movement to be applied embracing the three involved crossings. The accumulated result is shown as a second appended list. When new crossings appear, they are shown in boldface:

$$\left[H1^{+}, Ring^{-}, H2^{+}, Ring^{+}, \underbrace{H1^{-}, Ring^{-}, H2^{+}}_{(2R)}\right] \cdot [\,]$$

$$\left[H1^{+}, Ring^{-}, \underbrace{H2^{+}, Ring^{+}, H1^{-}}_{(1R)}\right] \cdot \left[H2^{+}, Ring^{-}\right]$$

$$\left[\underbrace{H1^{+}, Ring^{-}, H2^{+}}_{(2R)}\right] \cdot \left[\mathbf{H2}^{+}, Ring^{+}, \mathbf{H2}^{-}, H1^{-}, H2^{+}, Ring^{-}\right]$$

$$[\,] \cdot \left[H1^{+}, H2^{+}, Ring^{-}, H2^{+}, Ring^{+}, H2^{-}, H1^{-}, H2^{+}, Ring^{-}\right].$$

Conjecture 1. *Let Γ be an* **HST** *scenario, M a model of Γ, x a long object and s a ground situation such that $chain(x, s)$ is defined w.r.t. M. Then*:

1. *If $M \models \neg Impossible(pass(t, p), s)$ then $chain(x, do(pass(t, p)), s) = opT_{t,p}(chain(x, s))$;*
2. *If $M \models \neg Impossible(pass(h, p), s)$ then $chain(x, do(pass(h, p)), s) = opH_{h,p}(chain(x, s))$.*

Conjecture 1 above shall be formally proved in a future work.

We have built a forward planner `holestrings` (see Appendix A) that tries all action executions in an iterative deepening strategy, using operators *opH* and *opT* and exclusively considering the lists of crossings for representing each state.

Notice that, from the planner point of view, the Fisherman's Folly goal consists now in checking that no lists of crossings contains a face like $Ring^{v}$. The planner cannot obtain any solution with less than six action executions. Fig. 11 presents the first solution obtained by the planner in six steps, and shows the executed actions together with each intermediate state for the chain lists.[15] A graphical representation is also included.

[14] List α is a portion of $chain(Str)$ at state s_3 in Fig. 15.

[15] For the sake of readability, we omitted the object tips x^{-} and x^{+} which should respectively occur as first and last element of each list $chain(x)$.

state	$chain(P)$	$chain(Str)$	next action	movements
S_0	$[Ring^+]$	$[Sphere1^+, PostH^+, Sphere2^+]$	$pass(\{Str^+, Disk2\}, PostH^-)$	(PR)
s_1	$[Ring^+]$	$[Sphere1^+, PostH^+, Sphere2^+, PostH^-]$	$pass(\{Post^+, PostH\}, Ring^-)$	(PL)
s_2	[]	$[Sphere1^+, Ring^-, PostH^+, Ring^+, Ring^-, Sphere2^+, Ring^+, PostH^-]$	$pass(\{Sphere2\}, Ring^-)$	(1R) × 2 (1L)
s_3	[ ]	$[Sphere1^+, Ring^-, PostH^+, Sphere2^+, PostH^-, Ring^+]$	$pass(\{Ring\}, PostH^+)$	(2R)+(2L)
s_4	[ ]	$[Sphere1^+, PostH^+, Ring^-, Sphere2^+, Ring^+, PostH^-]$	$pass(\{Sphere2\}, Ring^+)$	(1L)
s_5	[ ]	$[Sphere1^+, PostH^+, Sphere2^+, PostH^-]$		

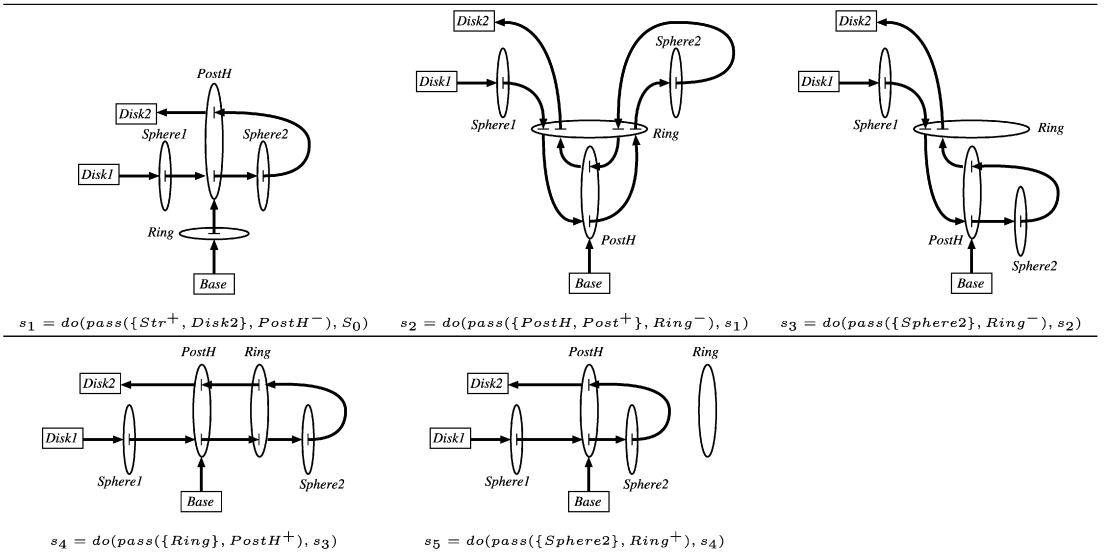


Fig. 11. A formal solution for the Fisherman's puzzle and its graphical representation.

7. A second puzzle: Rope Ladder

In this section we briefly present a second *strings* & *holes* puzzle, named Rope Ladder (Fig. 12). Rope Ladder can be seen as a more complex variant of Fisherman's Folly, containing the same sort of objects, entangled in a rigid U-shaped object. The ring in the initial state of this puzzle is trapped in the horizontal edge of the U-shaped figure, whose vertical edges are holed posts, each of which resembles the single post in the Fisherman's puzzle. The string makes a *zig-zag* from one post hole to the other (see state S_0 in Fig. 13), crossing each hole twice in the initial state. There are two holed spheres (that can slide through the string), initially lying near the tips of the string, each of these tips is linked to a disk. The constraints are analogous to those stated in the previous domain: the spheres cannot pass through either post-holes, where the ring and disks have free passage, if the hole isn't already crossed by another object.

A formalisation of the Rope Ladder puzzle, along with its automated solution, is analogous to that developed for Fisherman's Folly. We have in this case the sorts:

$$\begin{aligned}&\texttt{Regular} = \{\mathit{Disk}1, \mathit{Disk}2\},\\ &\texttt{Hole} = \{\mathit{Ring}, \mathit{Sphere}1, \mathit{Sphere}2, H1, H2\}, \quad \text{and}\\ &\texttt{Long} = \{\mathit{Str}, \mathit{Post}\}\end{aligned} \tag{37}$$

where $H1$ and $H2$ stand for the two post holes, *Post* represents the U-shaped object (without its holes) that is of long object sort. Instead of (FF1), (FF2) and (FF3), we include now the domain dependent axioms:

$$\begin{aligned}\mathit{Holds}\big(\mathit{linked}\big(\mathit{Str}^-, \mathit{Disk}1\big), \mathit{True}, S_0\big) \wedge \mathit{Holds}\big(\mathit{linked}\big(\mathit{Str}^+, \mathit{Disk}2\big), \mathit{True}, S_0\big) \wedge \mathit{Holds}\big(\mathit{linked}\big(\mathit{Post}^-, H1\big), \mathit{True}, S_0\big)\\ \wedge \mathit{Holds}\big(\mathit{linked}\big(\mathit{Post}^+, H2\big), \mathit{True}, S_0\big),\end{aligned} \tag{RL1}$$

Fig. 12. The Rope Ladder puzzle in its initial state.

$$\begin{aligned}
&\mathit{CannotPass}(\mathit{disk}, \mathit{sphere}, \emptyset) \wedge \mathit{CannotPass}(\mathit{disk}, \mathit{Ring}, \emptyset)\\
&\quad \wedge \mathit{CannotPass}(\mathit{Post}, \mathit{sphere}, \emptyset) \wedge \mathit{CannotPass}(\mathit{Post}, \mathit{postH}, \emptyset)\\
&\quad \wedge \mathit{CannotPass}(\mathit{sphere}, \mathit{postH}, \emptyset) \wedge \mathit{cannotPass}\big(\mathit{sphere}, \mathit{sphere}', \emptyset\big)\\
&\quad \wedge \mathit{CannotPass}\big(\mathit{sphere}, \mathit{Ring}, \{\mathit{Post}\}\big) \wedge \mathit{CannotPass}(\mathit{postH}, \mathit{sphere}, \emptyset)\\
&\quad \wedge \mathit{CannotPass}(\mathit{Ring}, \mathit{sphere}, \emptyset) \qquad \text{(RL2)}
\end{aligned}$$

with *disk* varying on $\{\mathit{Disk}1, \mathit{Disk}2\}$, *sphere*, sphere' on $\{\mathit{Sphere}1, \mathit{Sphere}2\}$, and *postH* on $\{H1, H2\}$.

$$\begin{aligned}
&\mathit{Holds}\big(\mathit{next}(\mathit{Str}, 0), 1, S_0\big) \wedge \mathit{Holds}\big(\mathit{towards}(\mathit{Str}, 0), \mathit{Sphere}1^{+}, S_0\big)\\
&\quad \wedge \mathit{Holds}\big(\mathit{next}(\mathit{Str}, 1), 2, S_0\big) \wedge \mathit{Holds}\big(\mathit{towards}(\mathit{Str}, 1), H1^{+}, S_0\big)\\
&\quad \wedge \mathit{Holds}\big(\mathit{next}(\mathit{Str}, 2), 3, S_0\big) \wedge \mathit{Holds}\big(\mathit{towards}(\mathit{Str}, 2), H2^{+}, S_0\big)\\
&\quad \wedge \mathit{Holds}\big(\mathit{next}(\mathit{Str}, 3), 4, S_0\big) \wedge \mathit{Holds}\big(\mathit{towards}(\mathit{Str}, 3), H1^{-}, S_0\big)\\
&\quad \wedge \mathit{Holds}\big(\mathit{next}(\mathit{Str}, 4), 5, S_0\big) \wedge \mathit{Holds}\big(\mathit{towards}(\mathit{Str}, 4), H2^{+}, S_0\big)\\
&\quad \wedge \mathit{Holds}\big(\mathit{next}(\mathit{Str}, 5), 6, S_0\big) \wedge \mathit{Holds}\big(\mathit{towards}(\mathit{Str}, 5), \mathit{Sphere}2^{+}, S_0\big)\\
&\quad \wedge \mathit{Holds}\big(\mathit{next}(\mathit{Str}, 6), \mathit{End}, S_0\big)\\
&\quad \wedge \mathit{Holds}\big(\mathit{next}(\mathit{Post}, 0), 1, S_0\big) \wedge \mathit{Holds}\big(\mathit{towards}(\mathit{Post}, 0), \mathit{Ring}^{+}, S_0\big)\\
&\quad \wedge \mathit{Holds}\big(\mathit{next}(\mathit{Post}, 1), \mathit{End}, S_0\big). \qquad \text{(RL3)}
\end{aligned}$$

The reduced number of changes in the formalisation of Fisherman's Folly to account for Rope Ladder is an indication that our formal solution to the former respects Elaboration Tolerance [22], at least in what concerns the representation of different physical configurations.

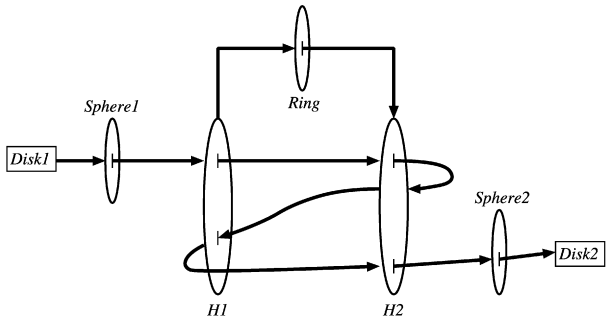
The solution of the Rope Ladder puzzle is sketched in Figs. 13 and 14 below, a list-based description of the states in this solution is shown in Fig. 15.

8. Discussion

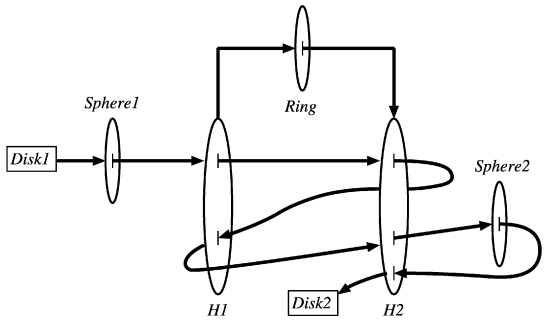
In this work we investigated a logic-based formalisation of a domain that involves qualitative spatial reasoning about non-trivial physical objects (such as strings and holes) and reasoning about actions and change on these objects. The domain chosen is the spatial puzzle named Fisherman's Folly, whose goal is to find a sequence of actions that applied to the puzzle's objects will disentangle a ring from a complex arrangement of a string, rigid objects and holes through them.

The endeavour of formalising the Fisherman's Folly puzzle is in line with the challenges for common sense reasoning[16] such as the *egg cracking problem* [46–48], whose logical formalisations were motivated by McCarthy's seminal papers [21, 17]. From a very abstract level, automated solutions to such domains could be obtained from a simple search on a state

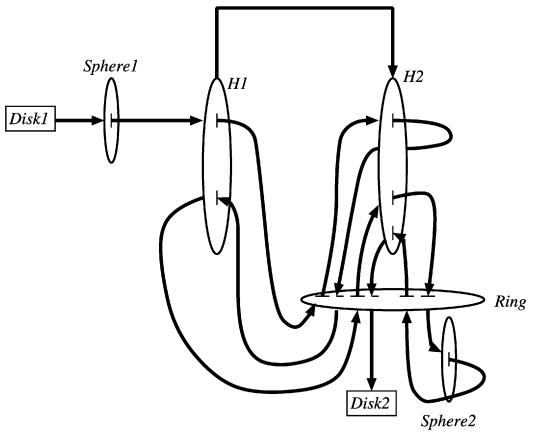
[16] A list of such challenges can be found on the *Common sense problem page*: http://www-formal.stanford.edu/leora/commonsense/ (last accessed in 10/12/2009).



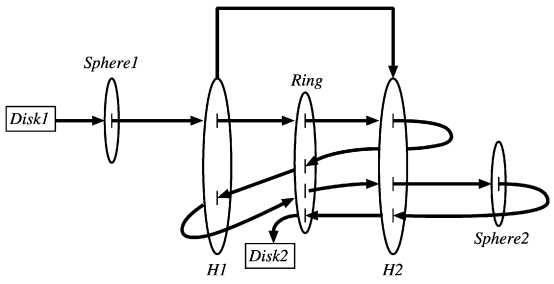
S_0



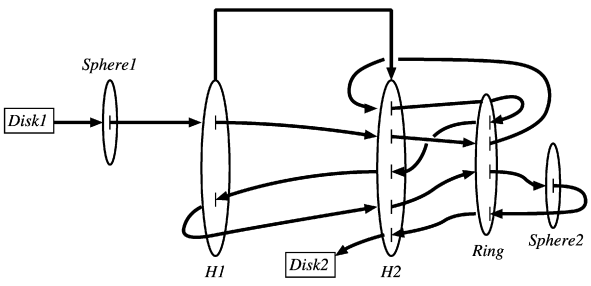
$s_1 = do(pass(\{Str^+, Disk2\}, H2^-), S_0)$



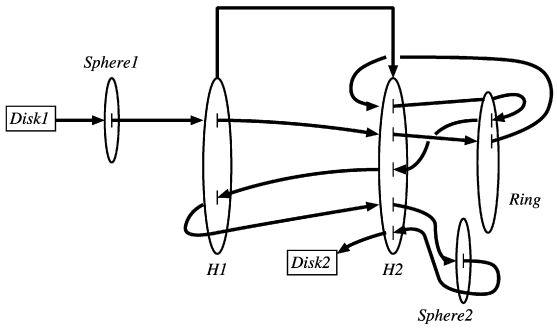
$s_2 = do(pass(\{H2, Post^+\}, Ring^-), s_1)$



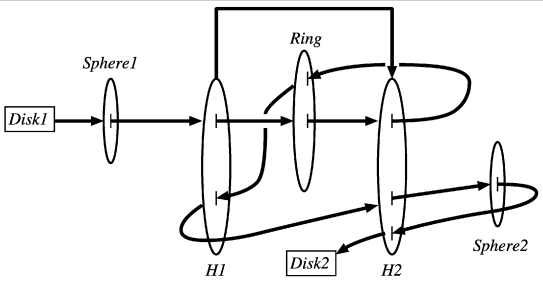
$s_3 = do(pass(\{Sphere2\}, Ring^-), s_2)$



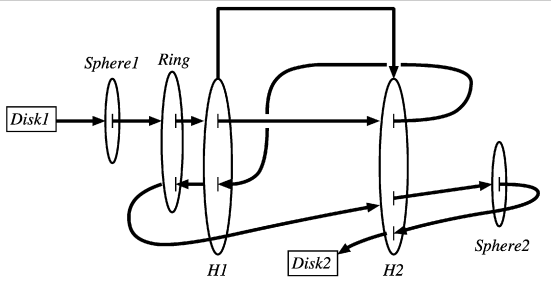
$s_4 = do(pass(\{Ring\}, H2^+), s_3)$



$s_5 = do(pass(\{Sphere2\}, Ring^+), s_4)$

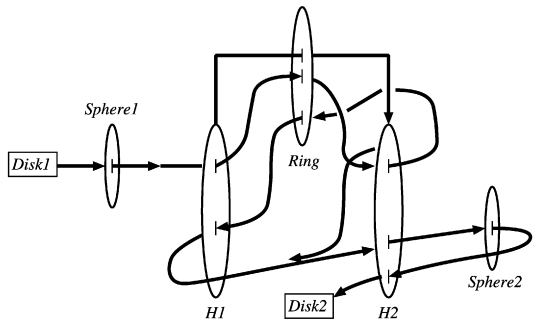


$s_6 = do(pass(\{Ring\}, H2^-), s_5)$

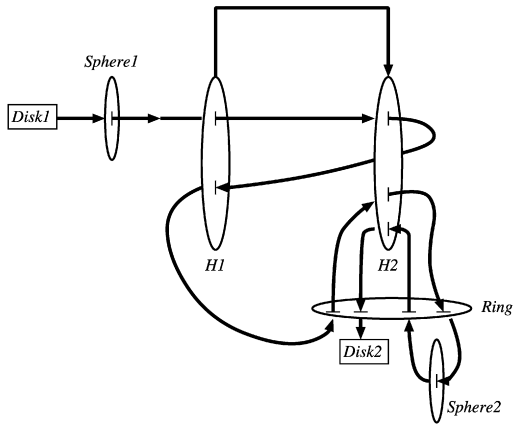


$s_7 = do(pass(\{Ring\}, H1^-), s_6)$

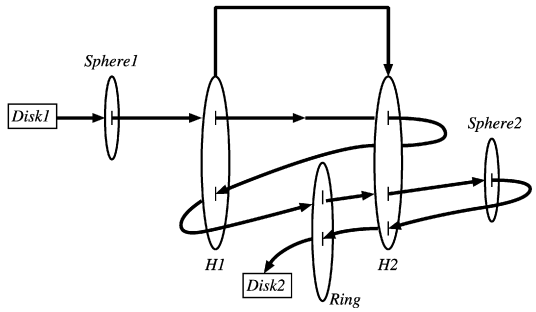
Fig. 13. Solution of Rope Ladder: first part.



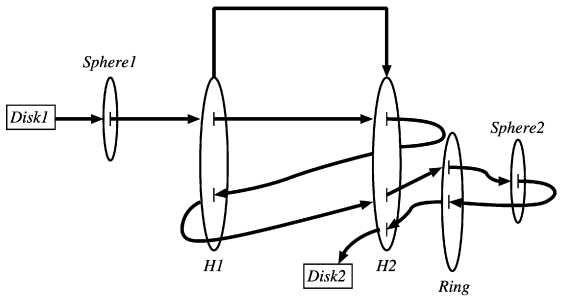
$s_8 = do(pass(\{H1, Post^-\}, Ring^+), s_7)$



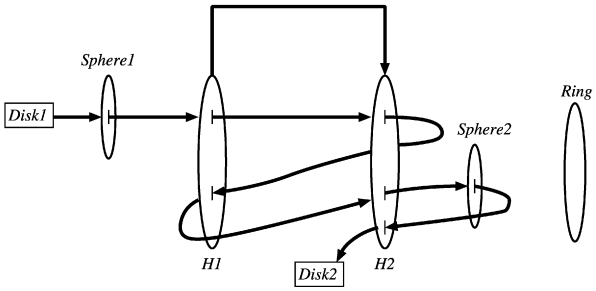
$s_9 = do(pass(\{H2, Post^+\}, Ring^+), s_8)$



$s_{10} = do(pass(\{Sphere2\}, Ring^+), s_9)$



$s_{11} = do(pass(\{Ring\}, H2^+), s_{10})$



$s_{12} = do(pass(\{Sphere2\}, Ring^-), s_{11})$

Fig. 14. Solution of Rope Ladder: second part.

state	*chain*(*p*)	*chain*(*str*)	next action(s)
S_0	[$Ring^+$]	[$Sphere1^+, H1^+, H2^+, H1^-, H2^+, Sphere2^+$]	$pass(\{Str^+, Disk2\}, H2^-)$
s_1	[$Ring^+$]	[$Sphere1^+, H1^+, H2^+, H1^-, H2^+, Sphere2^+, H2^-$]	$pass(\{H2, Post^+\}, Ring^-)$
s_2	[ ]	[$Sphere1^+, H1^+, Ring^-, H2^+, Ring^+, H1^-, Ring^-, H2^+, Ring^+, Sphere2^+, Ring^-, H2{\cdot}Ring^+$]	$pass(\{Sphere2\}, Ring^-)$
s_3	[ ]	[$Sphere1^+, H1^+, Ring^-, H2^+, Ring^+, H1^-, Ring^-, H2^+, Sphere2^+, H2^-$]	$pass(\{Ring\}, H2^+)$
s_4	[ ]	[$Sphere1^+, H1^+, H2^+, Ring^-, H2^+, Ring^+, H2^-, H1^-, H2^+, Ring^-, Sphere2^+, Ring^+, H2^-$]	$pass(\{Sphere2\}, Ring^+)$
s_5	[ ]	[$Sphere1^+, H1^+, H2^+, Ring^-, H2^+, Ring^+, H2^-, H1^-, H2^+, Sphere2^+, H2^-$]	$pass(\{Ring\}, H2^-)$
s_6	[ ]	[$Sphere1^+, H1^+, Ring^-, H2^+, Ring^+, H1^-, H2^+, Sphere2^+, H2^-$]	$pass(\{Ring\}, H1^-)$
s_7	[ ]	[$Sphere1^+, Ring^-, H1^+, H2^+, H1^-, Ring^+, H2^+, Sphere2^+, H2^-$]	$pass(\{H1, Post^-\}, Ring^+)$
s_8	[$Ring^-$]	[$Sphere1^+, H1^+, Ring^-, H2^+, Ring^+, H1^-, H2^+, Sphere2^+, H2^-$]	$pass(\{H2, Post^+\}, Ring^+)$
s_9	[ ]	[$Sphere1^+, H1^+, H2^+, H1^-, Ring^+, H2^+, Ring^-, Sphere2^+, Ring^+, H2{\cdot}Ring^+$]	$pass(\{Sphere2\}, Ring^+)$
s_{10}	[ ]	[$Sphere1^+, H1^+, H2^+, Ring^-, H1^-, Ring^+, H2^+, Sphere2^+, H2^-$]	$pass(\{Ring\}, H2^+)$
s_{11}	[ ]	[$Sphere1^+, H1^+, H2^+, H1^-, H2^+, Ring^+, Sphere2^+, Ring^-, H2^-$]	$pass(\{Sphere2\}, Ring^-)$
s_{12}	[ ]	[$Sphere1^+, H1^+, H2^+, H1^-, H2^+, Sphere2^+, H2^-$]	

Fig. 15. A formal solution to the Rope Ladder puzzle.

space, resembling automated solutions to classical domains such as the 8-puzzle [49]. However, this sort of solutions carry a number of implicit (hard-wired) assumptions on reasoning about the domain, which are brought to light by a logical representation. An example of such assumptions is present in the so-called Yale Shooting Scenario [13], a simple domain that proved to be an elusive problem in the formalisation of commonsense knowledge, due to the fact that its formal descriptions allowed the existence of counter-intuitive models. Nevertheless, it is possible to hand code a correct state-transition solution for this scenario (such the one shown in Fig. 5). However, it turns out that this solution is a consequence of sound reasoning about the domain, as it is the intended model of a correct logical encoding of the scenario. The problem here is how a logic could derive only the intended models (and not possible counter-intuitive ones), a problem that has motivated the development of non-monotonic logics [11]. We presented in Section 4 a formalism that solves this problem. In Sections 5 and 6 we used the proposed formalism to solve the Fisherman's Folly puzzle and, further, this solution is illustrated with a list-based model of the logical reasoning process. The connection between the list-based model and the logic formalisation of the puzzle is analogous to that relating the state-transition description of the Yale Shooting Scenario with its formal description: the list-based solution of the Fisherman's Folly puzzle carry strong assumptions on reasoning about the domain, which its logical description makes clear. In particular, the logic formalisation of the puzzle solves the problem of defining which parts of a list are not changed, by incorporating a commonsense law of inertia within the stable model semantics underlying the formalism developed.

Another advantage of using logic (over the direct application of a state transition system) for problem solving is its flexibility in handling elaborations of the domain: including a new transition in an automaton may imply on changing completely its structure (let alone the inclusion of indirect effects or continuous change, for instance), whereas – in principle – logic allows this to be done by adding the related axioms. In fact, much research has been done to equip logic with the proper tools to accomplish robust and efficient knowledge representation [7,6,8,9]. In this paper we follow the streamline of this research that tackles the solution of the various problems in knowledge representation assuming Situation Calculus (handling fluents and actions) and Equilibrium Logic (for correctly dealing with non-monotonicity) [18] as *lingua franca*.

Although the approach presented in this paper could solve a few puzzles of distinct complexities, there are situations where a more complex treatment is needed. For instance, the current approach has difficulties on the representation of a holed object crossing another holed object. Instead we assume in this work that the state of a hole *crossing* another hole is an intermediate state that exists momentarily (i.e. the state transition is not complete until this crossing holds). Relaxing this restriction is a matter for future investigations.

Another interesting subject for future work is how to relate the current formalisation to a lower-level representation based on spatial primitives, opening questions like the "creation" and "destruction" of regions (crossings and segments) or how to deal with the concept of inertia for spatial change. Clarifying the spatial properties of the puzzle's objects, such as the string's flexibility, and providing the precise reification of holes as real objects is the subject of our current research [25].

It is worth mentioning that the importance of reasoning about strings and holes goes beyond pure theoretical interest, embracing application domains such as: *cabling*, where reasoning about flexible and perforated objects is needed for optimising the spatial arrangements of networks of cables; *autonomous maintenance of mechanical machines*, from which devices have to be extracted without damaging the surrounding parts. But, perhaps, the most challenging application of reasoning about strings rely in the field of *robotic surgery*, where autonomous machines have to deal (and plan how to handle) sutures in situations involving very distinct and delicate structures.

Acknowledgements

We would like to close this paper by pointing out the influence of John McCarthy's ideas, adopted by us as our main motivations, and which have inspired most part of this work. This was a report on our investigations on logical formalisation of a common sense domain, an issue that was first proposed by McCarthy in [21] as a project, and later stated as a philosophical stance in [17]. Our formalisation was developed on top of the Situation Calculus and Equilibrium Logic. The former was first proposed by McCarthy in [17], and the latter is a novel non-monotonic formalism generalising stable models which surprisingly, have been recently characterised as a simple modification of Circumscription, also first proposed by McCarthy in [19]. Thus, it wouldn't be much far from the truth if we say that every sentence, every concept, in this paper was inspired by John's ideas.

Paulo Santos acknowledges support from FAPESP and CNPq, Brazil. Pedro Cabalar acknowledges support from Spanish Ministry MEC project TIN2006-15455-C03-02 and Xunta de Galicia project INCITE08-PXIB105159PR.

We also want to thank the reviewers for their thorough work and useful suggestions that have helped to improve the paper.

Appendix A. Planner

In this section we include the Prolog source for the planner. The program does not contain any domain dependent data, so it is applicable to Fisherman's Folly but also to similar puzzles, like the one in the next section (the Rope Ladder). The additional data for the Fisherman's Folly scenario is omitted, as it can be easily deduced from (FF1), (FF2) and (FF3).

The planner contains an additional action (currently commented) for cutting a link. Of course, enabling this action allows shorter solutions, like for instance, cutting the link between the post and its base and then passing $Post^{-}$ to $Ring^{+}$.

```
% SORTS ------------------------------------------------------
object(X) :- long(X); regular(X); hole(X).
sign(+). sign(-).
opposite(X^ +,X^ -). opposite(X^ -,X^ +).
tip(X^V) :- long(X), sign(V).
pole(H^V) :- hole(H), sign(V).
action(pass_t(T,P)):- tip(T),pole(P).
action(pass_h(H,G^V)):- hole(H),pole(G^V), G \= H.
% action(cut(T)):-tip(T).

% UNFEASIBILITY ----------------------------------------------
crossedBy(_H,[],_S):-!.
crossedBy(H,[X|Xs],S):- member(chain(X)=L,S),
    member(H^ _,L),!,crossedBy(H,Xs,S).

imposs1(pass_t(X^_,H^_), S):- crossedBy(H,A,S), cannotPass(X,H,A),!.
imposs1(pass_t(T,H^_), S):-
    member(linked(T,Y)=true,S), cannotPass(Y,H,A), crossedBy(H,A,S), !.
imposs1(pass_h(H,H2^_), S) :- cannotPass(H,H2,A),crossedBy(H2,A,S), !.
imposs1(cut(T), S) :- member(linked(T,_)=false,S),!.
imposs2(pass_h(H,_), S) :- member(linked(_,H)=true,S),!.

imposs(A,S) :- imposs1(A,S),!; imposs2(A,S).

% EFFECTS ----------------------------------------------------
replace(S0,X=V,S1):- append(Pre,[X=_|Suf],S0), append(Pre,[X=V|Suf],S1).

trans(S0, cut(X^ V), S1):-
    !, member(linked(X^ V,Y)=true,S0),
    replace(S0,linked(X^ V,Y)=false,S1).

trans(S0, pass_t(X^V,P), S1) :-
    !,member(chain(X)=A,S0), opT(X^V,P,A,B),
    replace(S0,chain(X)=B,S1).

trans([], pass_h(_,_), []):-!.
trans([chain(X)=A|S0], pass_h(H,P), [chain(X)=B|S1]):-
    !, opH(H,P,A,B), trans(S0,pass_h(H,P),S1).
trans([Fact|S0], pass_h(H,P), [Fact|S1]):-
    !, trans(S0,pass_h(H,P),S1).

opT(X^ +,P,A,G) :- !, append(B,[Y,X^ +],A),
    (opposite(P,Y),!, append(B,[X^ +],G)                    % (PR)
    ; append(B,[Y,P,X^ +],G)).                              % (PL)
opT(X^ -,P,[X^ -,P|B],[X^ -|B]) :- !.                       % (NL)
opT(X^ -,P,[X^ -,Y|B],[X^ -,P1,Y|B]) :- !,opposite(P,P1).   % (NR)

opH(_,_, [X],[X]) :- !.
opH(H,P, [X,G^V|B],[X|D]) :- G\=H,!,opH(H,P, [G^V|B],D).
opH(H,P, [X,H^V,Y|B],[X,P,H^V,P1|G]) :-
    opposite(P,P1), X \= P1, Y \= P, !, opH(H,P, [Y|B],G).  % (1R)
opH(H,P, [P1,H^V,P|B], [H^V|G]) :-
    opposite(P,P1),!,opH(H,P, [P|B],L), L=[P|G].            % (1L)
opH(H,P, [X,H^V,P|B],[X,P,H^V|G]) :-
    opposite(P,P1), X \= P1, !, opH(H,P, [P|B],L), L=[P|G]. % (2R)
opH(H,P, [P1,H^V,Y|B], [H^V,P1|G]) :-                       % (2L)
    opposite(P,P1),Y \= P,!,opH(H,P, [Y|B],G).
```

```
% TEMPORAL PROJECTION ----------------------------------------
execute(S,[]) :- !, write_state(S).
execute(S,[A|As]) :- !,
write_state(S),nl,write('ACTION: '),write(A),nl,
(imposs1(A,S),!,write(imposs(A)),nl,fail
; trans(S,A,S1),execute(S1,As)
).

% PLANNING: DEPTH-FIRST ITERATIVE DEEPENING -------------------
find_plan(MinDepth,MaxDepth,Sol):-
initial(S0),iterate_depth(MinDepth,MaxDepth,S0,Sol).

iterate_depth(N,Limit,_,_):-N>Limit,!,fail.
iterate_depth(N,Limit,S0,Sol):-
    write('Trying depth '),write(N),nl,
    get_plan(N,[S0],[],Sol)
  ; M is N+1, iterate_depth(M,Limit,S0,Sol).

get_plan(_N,[S0|Ss],Plan,([S0|Ss],Plan)):- is_goal(S0),nl.
get_plan(N,_,P,_):-length(P,M),M>=N,!,fail.
get_plan(N,[S0|Ss],Plan0,Sol):-
  action(A),
  check_undo_actions(A,Plan0),
  \+ imposs(A,S0),
  check_linked_action(S0,A,S1,A2),
  trans(S1,A,S2),
  \+ violated_constraint(S2),
  get_plan(N,[S2,S0|Ss],[A2|Plan0],Sol).

% If we passed an object tip linked to a hole, we pass the hole too
check_linked_action(S1,pass_t(T,P),S2,pass_t(T,P)+pass_h(H,P)):-
  member(linked(T,H)=true,S1),hole(H),!,trans(S1,pass_h(H,P),S2).
check_linked_action(S1,A,S1,A).

% Reject states with two repeated crossings... h+,h+
violated_constraint(S) :-
  member(chain(_Y)=L,S), append(_Pre,[X,X|_Suf],L).

opposite_action(pass_t(T,P),pass_t(T,P1)):-opposite(P,P1).
opposite_action(pass_h(X,P),pass_h(X,P1)):-opposite(P,P1).

check_undo_actions(A,[A1|_]):-!,\+ opposite_action(A,A1).
check_undo_actions(_,_).

write_solution((T,P)):-
  reverse(T,T1),reverse(P,P1),
  write_states(T1,P1).
write_states([],_):-!.
write_states([S],[]):-write_state(S).
write_states([S|Ss],[A|As]):-
  write_state(S),write_action(A),write_states(Ss,As).
write_state(S):-member(chain(X)=L,S),write(chain(X)=L),nl,fail.
write_state(_):-nl.
write_action(A):-write('ACTION: '),write(A),nl.

goal :- find_plan(0,7,K),write_solution(K).
```

Appendix B. Proofs

For proving Proposition 7 we will first use some properties of QEL that will allow us to convert **YALE** into an equivalent theory with the syntactic form of a normal logic program containing function symbols and variables. This normal program

can be transformed afterwards into an infinite ground program that can be inductively split to compute the transition system shown in Fig. 5.

Proof of Proposition 3. Proved in [50]. □

Proposition 11. *For any interpretation* $\langle (D,\sigma), H, T\rangle$ *and any formula* φ*, we have* $\langle (D,\sigma), H, T\rangle \models \varphi$ *implies* $\langle (D,\sigma), T\rangle \models \varphi$.

Proof. It easily follows by structural induction. □

Proposition 12. $\langle (D,\sigma), H, T\rangle \models \neg\varphi$ *is equivalent to* $\langle (D,\sigma), T\rangle \models \neg\varphi$.

Proof. By simply checking the conditions for satisfaction in *HT*. □

Lemma 1. *The formula* $\alpha : (\psi \to \varphi) \to \neg(\psi \wedge \neg\varphi)$ *is an HT-tautology.*

Proof. For any interpretation $M = \langle (D,\sigma), H, T\rangle$ we must prove both $\langle (D,\sigma), T\rangle \models \alpha$ and that $M \models (\psi \to \varphi)$ implies $M \models \neg(\psi \wedge \neg\varphi)$. The first condition follows from the fact that α is a classical tautology. Assume now that $M \models (\psi \to \varphi)$. This implies $\langle (D,\sigma), T\rangle \models \psi \to \varphi$ in classical logic, and this is equivalent to $\langle (D,\sigma), T\rangle \models \neg(\psi \wedge \neg\varphi)$ which, in its turn, is equivalent to $M \models \neg(\psi \wedge \neg\varphi)$ by Proposition 12. □

Lemma 2. *The following formula is an HT-tautology*:

$$\varphi \vee \neg\varphi \to \big((\psi \to \varphi) \leftrightarrow \neg(\psi \wedge \neg\varphi)\big). \tag{B.1}$$

Proof. For any interpretation $M = \langle (D,\sigma), H, T\rangle$ we would have to prove that both: $\langle (D,\sigma), T\rangle \models$ (B.1); and $M \models \varphi \vee \neg\varphi$ implies $M \models (\psi \to \varphi) \leftrightarrow \neg(\psi \wedge \neg\varphi)$. The first condition is trivial, since (B.1) is a classical tautology. To prove the second condition, assume M satisfies $\varphi \vee \neg\varphi$. By Lemma 1, the left to right direction of the equivalence $(\psi \to \varphi) \to \neg(\psi \wedge \neg\varphi)$ is an HT-tautology. For the right to left direction, it is clear that $\langle (D,\sigma), T\rangle \models (\psi \to \varphi) \leftarrow \neg(\psi \wedge \neg\varphi)$ since this is a classical tautology. Assume now that $M \models \neg(\psi \wedge \neg\varphi)$. By Proposition 12 this is equivalent to $\langle (D,\sigma), T\rangle \models \neg(\psi \wedge \neg\varphi)$ that in its turn is equivalent to $\langle (D,\sigma), T\rangle \models (\psi \to \varphi)$. We remain to prove that $M \models \psi$ implies $M \models \varphi$. Assume $M \models \psi$ but $M \not\models \varphi$. From the latter and $M \models \varphi \vee \neg\varphi$ we conclude $M \models \neg\varphi$. By Proposition 12 this is equivalent to $\langle (D,\sigma), T\rangle \models \neg\varphi$ and, as we had $\langle (D,\sigma), T\rangle \models (\psi \to \varphi)$ we get $\langle (D,\sigma), T\rangle \not\models \psi$ that by Proposition 11 implies $M \not\models \psi$ reaching a contradiction. □

Proof of Proposition 5. Since (DE) asserts that equality satisfies the excluded middle $x = y \vee \neg(x = y)$, we can apply Lemma 2 to obtain that $\varphi \to x = y$ is HT-equivalent to $\neg(\varphi \wedge x \neq y)$. But, by Proposition 4, the latter is HT-equivalent to $\varphi \wedge x \neq y \to \bot$. □

Proof of Proposition 6. Independently shown in [42,43]. □

Proof of Proposition 7. Note first that **YALE** has the form of a logic program excepting for the use of an existential quantifier in (INE), (DV) and (DV_0) and the use of conjunction in (8) and (9). Using the propositions in Section 3 and previous results in this appendix, we can transform **YALE** into a logic program, we can then consider $\Pi = gr_D(\textbf{YALE})$, that is, the (infinite) ground logic program resulting from replacing variables by all terms in the Herbrand universe D. An important observation is that this program can be *split* in the sense of the Splitting Theorem in [51], as any ground atom for *Holds* or *Imposs* indexed by a situation s never depends[17] on an atom indexed by $do(a,s)$. Furthermore, atoms for $do(a,s)$ and $do(a',s)$ with $a \neq a'$ do not even depend on each other. This allows us to construct an inductive proof on the structure of situation terms. For instance, the initial state for situation S_0 is uniquely fixed by (9). Then, by the splitting theorem, we can independently compute the states for $do(load, S_0)$, $do(wait, S_0)$ and $do(shoot, S_0)$ considering their corresponding ground rules.

We have implemented these 1-transition programs into lparse/smodels. The resulting programs can be seen in Figs. 16 and 17. From their execution we obtain all possible single transitions leading to the successor states of S_0 in Fig. 5. Then we could move to consider all situations with two actions $do(wait, do(load, S_0))$, $do(load, do(load, S_0))$, …, and so on. The complete formal proof would actually consist in assuming that the correspondence to the figure holds for all situations terms with n actions $do(a_n, do(a_{n-1}, \ldots, do(a_1, S_0)\ldots))$ and applying splitting to prove it for terms with $n+1$ actions. Finally, note that we get a unique state for S_0 and all transitions are deterministic. This shows that **YALE** has a unique equilibrium model. □

[17] An atom p *depends* on an atom q if there exists a rule with p occurring in its head and q in its body.

```
% Equilibrium Situation Calculus (ESC) --------------------------
#domain situation(S).
#domain fluent(F).
#domain inertial(IF).
#domain noninertial(NIF).
#domain action(A).

% Types of fluents
fluent(IF).
fluent(NIF).

% We only consider 1 transition
situation(s0).
situation(do(A,s0)).

% Unique value (UV)
:- holds(F,S,V), holds(F,S,W), V != W,
   range(F,V), range(F,W).

% Inertia (INE)
holds(IF,do(A,s0),V) :- not imposs(A,s0),
holds(IF,s0,V),
not released(IF,s0),
not ab(IF,V,A,s0),
range(IF,V).

ab(IF,V,A,s0) :- holds(IF,do(A,s0),W), W!=V,
range(IF,V), range(IF,W).

% Default values (DV)
holds(NIF,S,V) :- not ab2(NIF,S,V), default(NIF,V), range(NIF,V).
ab2(NIF,S,V) :- holds(NIF,S,W), W!=V, range(NIF,V), range(NIF,W).

% Default values for inertial fluents (DV0)
holds(IF,s0,V) :- not ab3(IF,s0,V), default(IF,V), range(IF,V).
ab3(IF,s0,V) :- holds(IF,s0,W), W!=V, range(IF,V), range(IF,W).

% 'defined' predicate
defined(F,S) :- holds(F,S,V), range(F,V).
```

Fig. 16. Axioms of **ESC** for a single transition, from S_0 to $do(A, S_0)$, represented in an lparse/smodels program.

Appendix C. A third puzzle: the Tricky Dick

The *Tricky Dick*© is a puzzle designed by Richard Earson that was presented in the *International Puzzle Party*, London 1999. The puzzle, shown in Fig. 18, consists of:

- two wooden cylinders of different length connected in their bottom by a tiny piece of string,
- a pair of metallic rings of different diameter (the large one cannot pass through the small one),
- a string loop linked to the top of the small cylinder and embracing the large cylinder,
- and finally, a string linked to the top of the large cylinder, crossing through the small ring and linked to the large ring.

As in the other puzzles, the goal is removing the small ring, so it gets free from the rest of the puzzle.

In order to formalise the puzzle we will make several simplifying assumptions. For instance, although the cylinders can pass through the large ring, after playing a while with the puzzle we may soon discover that the real role of this piece is to be a top for the small ring. Thus, for the sake of simplicity, we will consider the large ring as a regular object, ignoring its hole.

A second simplifying assumption is considering the two cylinders and the tiny string that links them in their bottom as a single piece. In fact, the mobility of this part of the puzzle is practically null and so, is equivalent to a single 'U'-shaped wooden piece.

```
% YALE SHOOTING SCENARIO -----------------------------
% We define sorts for fluent values.
% In this case, just Boolean fluents
boolean(true;false).

% We define the fluents
inertial(loaded).
range(loaded,Bool) :- boolean(Bool).
inertial(alive).
range(alive,Bool) :- boolean(Bool).

% We define the set of actions
action(load).
action(shoot).
action(wait).

% Executability axioms
impossible(load,S) :- holds(loaded,S,true).
impossible(shoot,S) :- holds(loaded,S,false).

% Effect axioms
holds(loaded,do(load,s0),true) :- not impossible(load,s0).
holds(alive,do(shoot,s0),false) :- not impossible(shoot,s0).
holds(loaded,do(shoot,s0),false) :- not impossible(shoot,s0).

% Initial state generation
1 { holds(F,s0,VAL) : range(F,VAL) } 1.

hide.
show holds(X1,X2,X3).
```

Fig. 17. YALE represented in an lparse/smodels program.

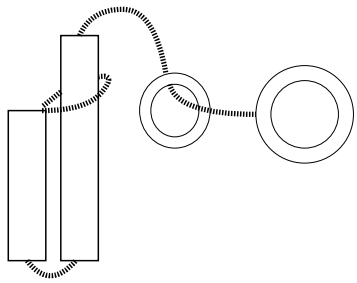


Fig. 18. The Tricky Dick puzzle.

Finally, the role of the string loop is quite different from the way in which we handle the longest string. Since this loop is permanently fixed, it can be actually seen as a (flexible) holed object.[18]

To sum up, the sorts for this scenario would correspond to:

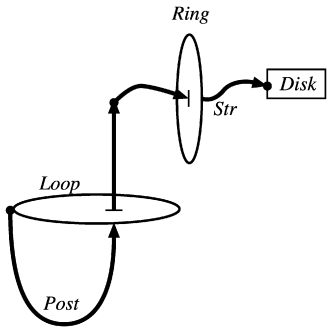
$$\begin{aligned} &\texttt{Regular} = \{\mathit{LargeRing}\},\\ &\texttt{Hole} = \{\mathit{Ring}, \mathit{Loop}\}, \quad \text{and}\\ &\texttt{Long} = \{\mathit{Str}, \mathit{Post}\} \end{aligned} \tag{C.1}$$

where *Post* represents the U-shaped object formed by the cylinders and the tiny string. The initial state of the puzzle can be schematically represented by situation S_0 in Fig. 19.

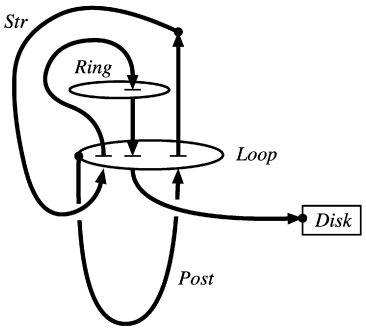
The domain dependent axioms would be:

$$\begin{aligned} \mathit{Holds}\big(\mathit{linked}\big(\mathit{Str}^-, \mathit{Post}^+\big), \mathit{True}, S_0\big) \wedge \mathit{Holds}\big(\mathit{linked}\big(\mathit{Str}^+, \mathit{LargeRing}\big), \mathit{True}, S_0\big)\\ \wedge \mathit{Holds}\big(\mathit{linked}\big(\mathit{Post}^-, \mathit{Loop}\big), \mathit{True}, S_0\big), \end{aligned} \tag{TD1}$$

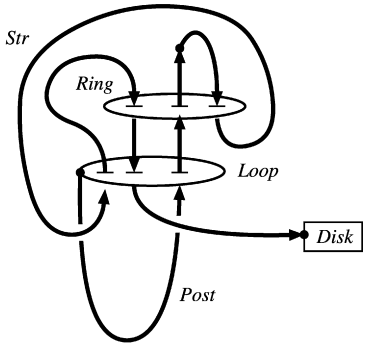
[18] In fact, this is directly related to one of the topics we are currently considering for immediate future work: linking the two tips of a string or making a knot in it forms a string loop that can be handled as a holed object (we can define two faces, pass objects through the loop, etc.).



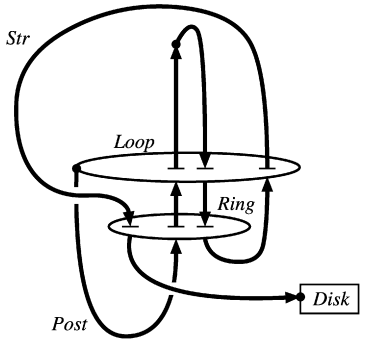
S_0



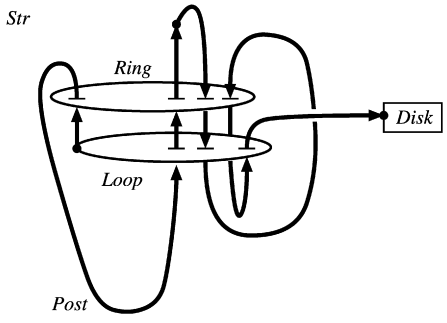
$s_1 = do(pass(\{Ring\}, Loop^+), S_0)$



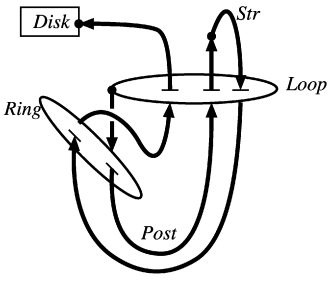
$s_2 = do(pass(\{Str^-, Post^+\}, Ring^+), s_1)$



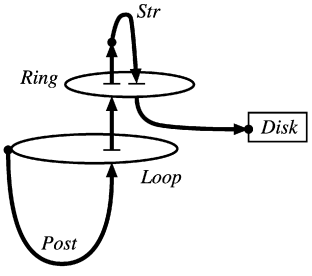
$s_3 = do(pass(\{Ring\}, Loop^-), s_2)$



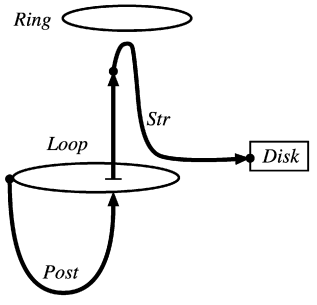
$s_4 = do(pass(\{Loop, Post^-\}, Ring^-), s_3)$



$s_5 = do(pass(\{Post^+, Str^-\}, Ring^-), s_4)$



$s_6 = do(pass(\{Ring\}, Loop^+), s_5)$



$s_7 = do(pass(\{Post^+, Str^-\}, Ring^-), s_6)$

Fig. 19. Solution of Tricky Dick.

state	*chain*(*Post*)	*chain*(*Str*)	next action(s)
S_0	$[Loop^+]$	$[Ring^-]$	$pass(\{Ring\}, Loop^+)$
s_1	$[Loop^+]$	$[Loop^+, Ring^-, Loop^-]$	$pass(\{Str^-, Post^+\}, Ring^+)$
s_2	$[Loop^+, Ring^+]$	$[Ring^-, Loop^+, Ring^-, Loop^-]$	$pass(\{Ring\}, Loop^-)$
s_3	$[Ring^+, Loop^+]$	$[Loop^-, Ring^-, Loop^+, Ring^-]$	$pass(\{Loop, Post^-\}, Ring^-)$
s_4	$[Ring^+, Loop^+, Ring^+]$	$[Ring^-, Loop^-, Ring^-, Loop^+]$	$pass(\{Post^+, Str^-\}, Ring^-)$
s_5	$[Ring^+, Loop^+]$	$[Loop^-, Ring^-, Loop^+]$	$pass(\{Ring\}, Loop^+)$
s_6	$[Loop^+, Ring^+]$	$[Ring^-]$	$pass(\{Post^+, Str^-\}, Ring^-)$
s_7	$[Loop^+]$	[ ]	

Fig. 20. A formal solution to the Tricky Dick puzzle.

$$CannotPass(LargeRing, Ring, \emptyset) \wedge CannotPass(LargeRing, Loop, \emptyset), \tag{TD2}$$

$$\begin{aligned} Holds\big(next(Str, 0), S_0, 1\big) &\wedge Holds\big(towards(Str, 0), S_0, Ring^-\big) \\ &\wedge Holds\big(next(Str, 1), S_0, End\big) \\ &\wedge Holds\big(next(Post, 0), S_0, 1\big) \wedge Holds\big(towards(Post, 0), S_0, Loop^+\big) \\ &\wedge Holds\big(next(Post, 1), S_0, End\big). \end{aligned} \tag{RL3}$$

The solution is shown in Figs. 19 and 20, that respectively contain the graphical representation and the complete intermediate states (in terms of *chain* lists).

The real interesting feature of this puzzle is that state s_5 is not actually feasible in the real device due to rigidness of the post (i.e. the cylinders). The real puzzle solution implies a combined movement: the right side of the ring is rotated pivoting on its left side and simultaneously embracing $Post^+$ and the *Loop*. However, the *effects* of this combined movement exactly correspond to state s_6 where we "virtually" decompose this action into first passing the loop and $Post^-$, and then passing $Post^+$ in a second step. □

References

[1] O. Stock (Ed.), Spatial and Temporal Reasoning, Kluwer Academic, 1997.
[2] A.G. Cohn, S.M. Hazarika, Qualitative spatial representation and reasoning: An overview, Fundamenta Informaticae 46 (1–2) (2001) 1–29.
[3] A.G. Cohn, J. Renz, Qualitative spatial representation and reasoning, in: F. van Hermelen, V. Lifschitz, B. Porter (Eds.), Handbook of Knowledge Representation, Elsevier, 2008, pp. 551–596.
[4] M. Souchanski, P. Santos, Reasoning about dynamic depth profiles, in: Proc. of the 18th European Conference on Artificial Intelligence (ECAI), IOS, 2008, pp. 30–34.
[5] M. Bhatt, S. Loke, Modelling dynamic spatial systems in the situation calculus, Spatial Cognition and Computation 8 (1–2) (2008) 86–130.
[6] R. Reiter, Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, MIT Press, Cambridge, MA, 2002.
[7] M. Shanahan, Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia, MIT Press, Cambridge, MA, 1997.
[8] E. Davis, Representations of Commonsense Knowledge, Morgan Kaufmann, 1990.
[9] F. van Harmelen, V. Lifschitz, B. Porter (Eds.), Handbook of Knowledge Representation (Foundations of Artificial Intelligence), Elsevier Science, 2007.
[10] R. Davis, H. Shrobe, P. Szolovits, What is a knowledge representation? AI Magazine 14 (1) (1993) 17–33.
[11] G. Brewka, Nonmonotonic Reasoning: Logical Foundations of Commonsense, Cambridge University Press, 1991.
[12] M. Shanahan, The frame problem, in: E.N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, Fall 2008, http://plato.stanford.edu/archives/fall2008/entries/frame-problem/.
[13] S. Hanks, D. McDermott, Nonmonotonic logic and temporal projection, Artificial Intelligence 33 (3) (1987) 379–412.
[14] H. Simon, J. Schaeffer, The game of chess, in: H. Aumann (Ed.), Handbook of Game Theory with Economic Applications, vol. 1, North-Holland, 1992, pp. 1–17.
[15] J. McCarthy, AI as sport, Science 276 (5318) (1997) 1518–1519.
[16] P. Cabalar, P. Santos, Strings and holes: an exercise on spatial reasoning, in: 10th Iberoamerican Conference on AI (IBERAMIA'06), in: Lecture Notes in Artificial Intelligence, vol. 4140, 2006, pp. 419–429.
[17] J. McCarthy, P. Hayes, Some philosophical problems from the standpoint of artificial intelligence, Machine Intelligence Journal 4 (1969) 463–512.
[18] D. Pearce, A. Valverde, Quantified equilibrium logic and the first order logic of here-and-there, Tech. Rep. MA-06-02, University of Málaga, Spain, 2006.
[19] J. McCarthy, Circumscription: A form of non-monotonic reasoning, Artificial Intelligence 13 (1980) 27–39.
[20] V. Lifschitz, Circumscription, in: D.M. Gabbay, C.J. Hogger, J.A. Robinson (Eds.), The Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3, Oxford University Press, 1994, pp. 297–352.
[21] J. McCarthy, Programs with common sense, in: Proc. of the Teddington Conference on Mechanization of Thought Processes, 1959, pp. 75–91.
[22] J. McCarthy, Elaboration tolerance, http://www-formal.stanford.edu/jmc/elaboration.ps, 1999.
[23] A.C. Varzi, Reasoning about space: The hole story, Logic and Logical Philosophy 4 (1996) 3–39.
[24] R. Casati, A.C. Varzi, Parts and Places, MIT Press, 1999.
[25] P. Santos, P. Cabalar, Playing with a puzzle in mereotopology, Spatial Cognition and Computation 8 (1–2) (2008) 47–64.
[26] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, K. Ikeuchi, Knot planning from observation, in: Proc. of the IEEE ICRA, 2003, pp. 3887–3892.
[27] J. Takamatsu, T. Morita, K. Ogawara, H. Kimura, K. Ikeuchi, Representation for knot-tying tasks, IEEE Transactions on Robotics 22 (1) (2006) 65–78.
[28] K. Reidemeister, Knot Theory, BCS Associates, 1983.
[29] N. Abolhassani, R. Patel, M. Moallem, Needle insertion into soft tissue: A survey, Medical Engineering & Physics 29 (4) (2007) 413–431.
[30] R. Kowalski, M. Sergot, A logic-based calculus of events, New Generation Computing 4 (1986) 67–95.
[31] A.B. Baker, A simple solution to the Yale shooting problem, in: Proc. of the Intl. Conf. on Knowledge Representation and Reasoning, 1989, pp. 11–20.
[32] E. Sandewall, Features and Fluents. A Systematic Approach to the Representation of Knowledge about Dynamical Systems, Oxford University Press, 1994.

[33] F. Lin, Embracing causality in specifying the indirect effects of actions, in: C.S. Mellish (Ed.), Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI), Morgan Kaufmann, Montreal, Canada, 1995.
[34] M. Thielscher, Ramification and causality, Artificial Intelligence Journal 1–2 (89) (1997) 317–364.
[35] M. Gelfond, V. Lifschitz, Representing action and change by logic programs, Journal of Logic Programming 17 (1993) 301–321.
[36] M. Gelfond, V. Lifschitz, The stable models semantics for logic programming, in: Proc. of the 5th Intl. Conf. on Logic Programming, 1988, pp. 1070–1080.
[37] C. Baral, Embedding revision programs in logic programming situation calculus, Journal of Logic Programming 30 (1) (1997) 83–97.
[38] H. Turner, Representing actions in logic programs and default theories: A situation calculus approach, Journal of Logic Programming 31 (1997) 245–298.
[39] D. Pearce, A new logical characterisation of stable models and answer sets, in: Nonmonotonic Extensions of Logic Programming, Proc. NMELP'96, in: LNAI, vol. 1216, Springer-Verlag, 1996.
[40] D. Pearce, A. Valverde, Towards a first order equilibrium logic for nonmonotonic reasoning, in: Proc. of the 9th European Conf. on Logics in AI (JELIA'04), 2004, pp. 147–160.
[41] P. Ferraris, J. Lee, V. Lifschitz, A new perspective on stable models, in: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI'07), 2007, pp. 372–379.
[42] P. Cabalar, Existential quantifiers in the rule body, in: Proc. of the 23rd Workshop on (Constraint) Logic Programming (WLP'09), 2009.
[43] J. Lee, R. Palla, System F2LP – computing answer sets of first-order formulas, in: Proc. of the 10th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'09), in: Lecture Notes in Computer Science, vol. 5753, Springer, 2009, pp. 515–521.
[44] P. Cabalar, D. Lorenzo, Logic programs with functions and default values, in: 9th European Conference on Logics in Artificial Intelligence (JELIA'04), Lisbon, Portugal, in: LNAI, vol. 3229, 2004, pp. 294–306.
[45] V. Lifschitz, Answer set programming and plan generation, Artificial Intelligence 138 (2002) 39–54.
[46] M. Shanahan, An attempt to formalise a non-trivial benchmark problem in common sense reasoning, Artificial Intelligence 153 (1–2) (2001) 141–165.
[47] L. Morgenstern, Mid-sized axiomatizations of commonsense problems: A case study in egg cracking, Studia Logica 67 (3) (2001) 333–384.
[48] V. Lifschitz, Cracking an egg: An exercise in commonsense reasoning, in: Proc. of the Symposium of Logical Formalizations of Commonsense Reasoning, 1998.
[49] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice-Hall, 2003.
[50] V. Lifschitz, L.R. Tang, H. Turner, Nested expressions in logic programs, Annals of Mathematics and Artificial Intelligence 25 (1999) 369–389.
[51] V. Lifschitz, H. Turner, Splitting a logic program, in: International Conference on Logic Programming, 1994, pp. 23–37.