Artificial Intelligence 238 (2016) 1–10

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

A qualitative spatial representation of string loops as holes

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:
Received 11 December 2015
Received in revised form 9 April 2016
Accepted 10 May 2016
Available online 13 May 2016

Keywords:
Spatial representation
Problem solving
Reasoning about actions

ABSTRACT

This research note contains an extension of a previous work by Cabalar and Santos (2011) that formalised several spatial puzzles formed by strings and holes. That approach explicitly ignored some configurations and actions that were irrelevant for the studied puzzles but are physically possible and may become crucial for other spatial reasoning problems. In particular, the previous work did not consider the formation of string loops or the situations where a holed object is partially crossed by another holed object. In this paper, we remove these limitations by treating string loops as dynamic holes that can be created or destroyed by a pair of elementary actions, respectively picking or pulling from strings. We explain how string loops can be recognised in a data structure representing the domain states and define a notation to represent *crossings through string loops*. The resulting formalism is dual in the sense that it also allows understanding any hole as a kind of (sometimes rigid) closed string loop.

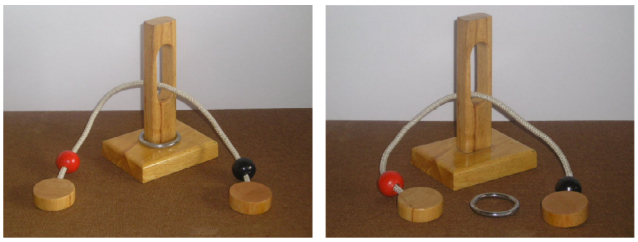
1. Introduction

The design of computer programs and machines with commonsense reasoning constitutes an important long-term goal of Artificial Intelligence. In most commonsense reasoning scenarios, the spatial component of the domain plays a fundamental role. People usually reason about spatial entities and their behaviour in their daily lives without apparent effort – it is somehow an embodied (and possibly innate) feature in the human mind. As a simple example, think about all the steps for putting on a pair of trousers and a belt. While children usually learn this process without much difficulty, scenarios like this become a real challenge for computer programs as they must deal with complex geometric figures (e.g. the pants, the zipper), measure-related constraints (such as choosing the right hole in the belt, depending on your waist size) and other object constraints related to rigidness (the belt buckle) versus flexibility (the clothes and the belt).

Research on spatial commonsense reasoning comes from two main sources in the Knowledge Representation (KR) literature. On the one hand, the area of *Reasoning about Actions and Change* comprises a family of logical languages [1–5] for the formalization of an intelligent agent operating in action domains and performing common reasoning tasks such as simulation, planning, temporal explanation or diagnosis. On the other hand, *Qualitative Spatial Reasoning* (QSR) [6,7] aims at the rigorous treatment of qualitative abstractions of spatial entities that constitute the foundations of our commonsense understanding of the external world. Although the combination of QSR and temporal reasoning is not unfrequent in the literature (see for instance [8]), in general QSR approaches have traditionally overlooked a formal treatment of actions as those involved in our previous example or tackled temporal reasoning tasks such as planning, simulation or explanation.

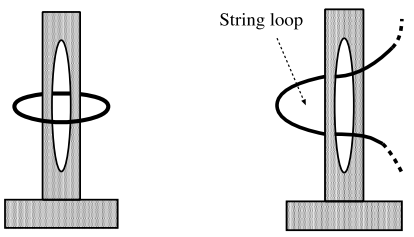
* Corresponding authors.
E-mail addresses: cabalar@udc.es (P. Cabalar), psantos@fei.edu.br (P.E. Santos).

http://dx.doi.org/10.1016/j.artint.2016.05.001
0004-3702/



(a) Initial (b) Goal: the ring is free

Fig. 1. A spatial puzzle: the Fisherman's Folly.



(a) Crossings. (b) String loop.

Fig. 2. String loops and hole-hole crossings.

Trying to fill this gap, we have concentrated our efforts in formalizing action domains that involve flexible objects and holes, as they are very common in different scenarios like our trousers example.[1] Our methodology, applied along a series of papers [9–13], has consisted in studying spatial puzzles in a bottom-up fashion, starting from restricted cases and gradually covering new puzzles with more challenging features. Puzzles constitute a good test bed, as they offer a small number of objects requiring a minimum background knowledge about unrelated features, while they keep enough complexity to constitute a challenging problem for KR. Most of these puzzles consist in releasing a rigid ring from an entanglement of strings and other objects.

Our initial efforts were put in solving the so-called Fisherman's Folly puzzle shown in Fig. 1 using a list-based representation of string crossings. All this work eventually led to an extensive paper [11] describing a complete logical formalization plus a preliminary planner capable of solving a family of related puzzles with similar features.

In [11] some issues were left open. In particular, we did not consider states where a holed object was partially crossing another hole, as in Fig. 2(a), or the formation of string loops as in Fig. 2(b). Both situations were irrelevant for solving the family of puzzles under study but, as it can be imagined, ignoring them may easily suppose a lack of elaboration tolerance for other closely related puzzles. For instance, the variation of Fisherman's Folly shown in Fig. 3(a) is essentially the *same* puzzle with the difference that the holed post has been replaced by a long metallic arc. The latter forms a hole that, in its turn, must cross the ring hole, becoming a case of Fig. 2(a). Although this feature is not essential for solving Fisherman's Folly, there are other puzzles that cannot be solved without removing these restrictions – for instance, in [14] we studied the so-called "*easy-does-it*" puzzle (Fig. 3(b)) which cannot be solved without representing (and acting upon) string loops.

The present paper shows how the recent developments reported in [14] fill up a number of gaps left open in [11] and allow removing the above mentioned limitations by considering the formation of string loops. We describe how to recognize loops in a list of string crossings and define a notation to represent *crossings through string loops*, as they actually behave as regular holes. On top of the previous approach, we identify two basic new actions on strings that may form or destroy loops: (1) *picking* a string segment through a hole, and (2) *pulling* from a string to unwind a loop. The most difficult part of the paper corresponds to the description of the direct and indirect effects of these two actions and, particularly, to the fact that a loop may be inside some larger loop. As a result, an action on a loop may imply inheriting crossings with respect to a larger loop. We also explain how a hole can also be seen as a kind of (sometimes rigid) closed string loop, allowing the representation of problems such as the one in Fig. 2(a). The next section introduces the basis upon which this work was developed.

[1] In fact, most actions in the example involve passing objects through holes: we pass our legs through the trousers sleeves, the button through the buttonhole, the belt through loops in the trousers, the belt tip through the belt buckle, and the buckle bolt through a hole in the belt.