On the Geometric Modelling of Visual Languages

John Power and Kostas Tourlas

October 10, 2002

Visual Language Modelling

Models support

- definition
- inference (i.e. resoning)

Definitional models often include concrete aspects of how diagrams are realised, typically on computers.

Yet visual languages exist independent of their computer realisations.

Geometric models

- what are they?
- assist understanding
- assist improved tool design

A Case for Geometry

Do we need geometric models?

- Not always, perhaps not in the end
- Yes, early at the VL conception stage
- Yes, when layout is semantically significant
- Yes, to underpin the design of usable, well-structured tools.

Hasn't all this been done before?

- Attributed models (grammars or algebras)
- OO (meta)models, twin-level models

These are "concrete" but still non-geometric.

Aren't geometric models intractable? Not for core visual formalisms.

A non-geometric model

A *constraint diagram* is a tuple $(C, A, \ldots)$, where

- C is a finite set **whose elements are called contours**,
- A is a set of arrows,
- ...

- this definition is "*independent of any topological and visual representations*"

- contours are such only in name

- and so $C = \{1, 2, 3, 4\}$ is a valid instance

Intrinsically geometric models

Explicitly include information as to how the structures in the abstract model are realised on the plane.

Typically, a function

$$AbstractModel \longrightarrow Plane = \mathbb{R}^2$$

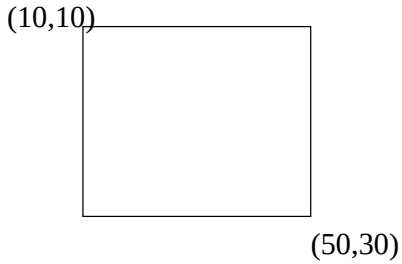
using primitives such as

- *paths*, i.e. continuous embeddings $[0, 1] \longrightarrow \mathbb{R}^2$
- *simple closed curves*, i.e. images under continuous embeddings of the unit circle.

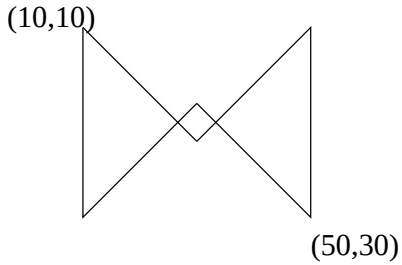
Even topological definitions are often non-geometric!

How geometric are attributes ?

Consider "rectangle" r with attributes $ul = (10, 10)$ and $lr = (50, 30)$. In the presence only of attributes,



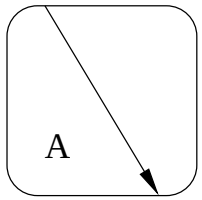
realises the "abstract" rectangle r. But, arguably, so does this:



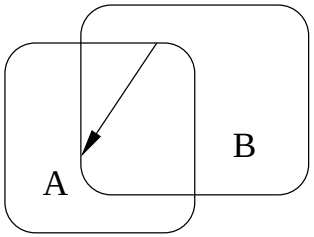
Sanity issues

But who wants twisted rectangles?

In a graph, who wants



or

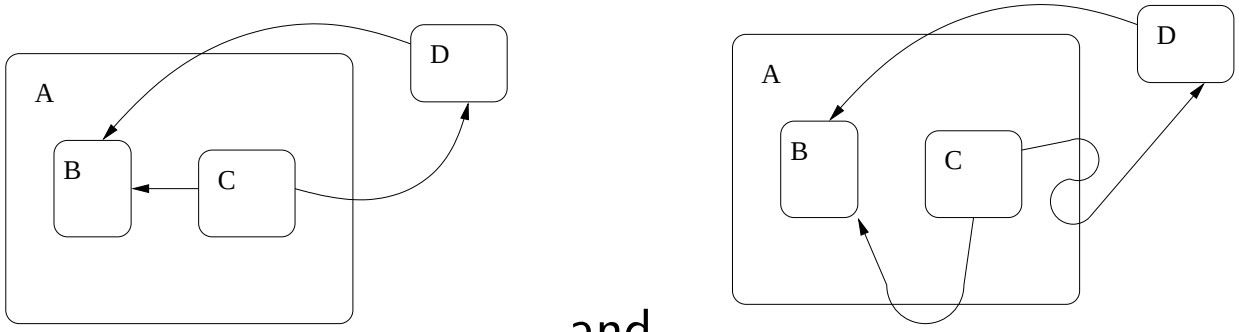


either? Geometric models must be embeddings.

Higraph Visualisation

A higraph is a tuple $(B, \leq_B, E, s, t : E \to B)$, where $\langle B, \leq_B \rangle$ a poset of "blobs", E a set of "edges".

Abstract model does not distinguish between



and

But neither do most concrete models. Harel does!

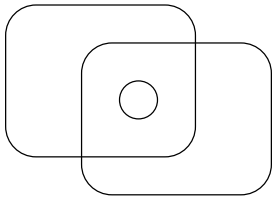
Higraph embedding

An *embedding* of a higraph the plane is a pair $\mathcal{E} = \langle \mathcal{E}_B, \mathcal{E}_E \rangle$ of functions such that:

- $\mathcal{E}_B$ sends each $b \in B$ a simple closed curve
- $\mathcal{E}_E$ sends each $e \in E$ to a path from the boundary of $\mathcal{E}_B(s(e))$ to the boundary of $\mathcal{E}_B(t(e))$.

1. $b \neq b' \implies \mathcal{E}_B(b) \cap \mathcal{E}_B(b') = \emptyset$;
2. $b' \leq b \implies$
$$\mathcal{I}(\mathcal{E}_B(b')) \cap \mathcal{I}(\mathcal{E}_B(b)) = \mathcal{I}(\mathcal{E}_B(b'))$$
3. $e \neq e' \implies \mathcal{E}_E(e) \cap E_E(e')$ finite;
4. $\mathcal{E}_E(e) \cap \mathcal{E}_B(b)$ also finite.

Some observations



Not all higraphs are embeddable:

Model is relatively simple and tractable.

Leaves no assumptions implicit.

Makes no assumption as to shape of blobs or edges.

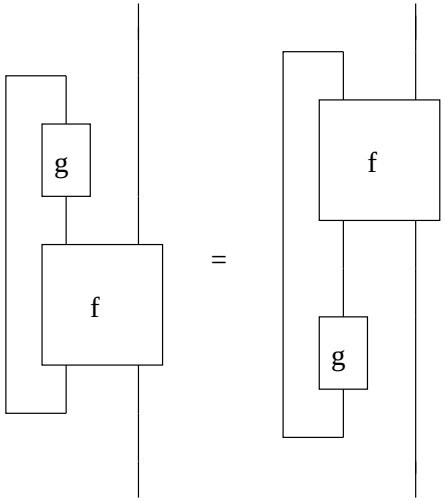
Forces inclusion of easy-to-miss conditions.

Makes embedding precise.

Not immediately implementable, however. . .

Joyal & Street Diagrams

Equation solving in the tensor calculus.



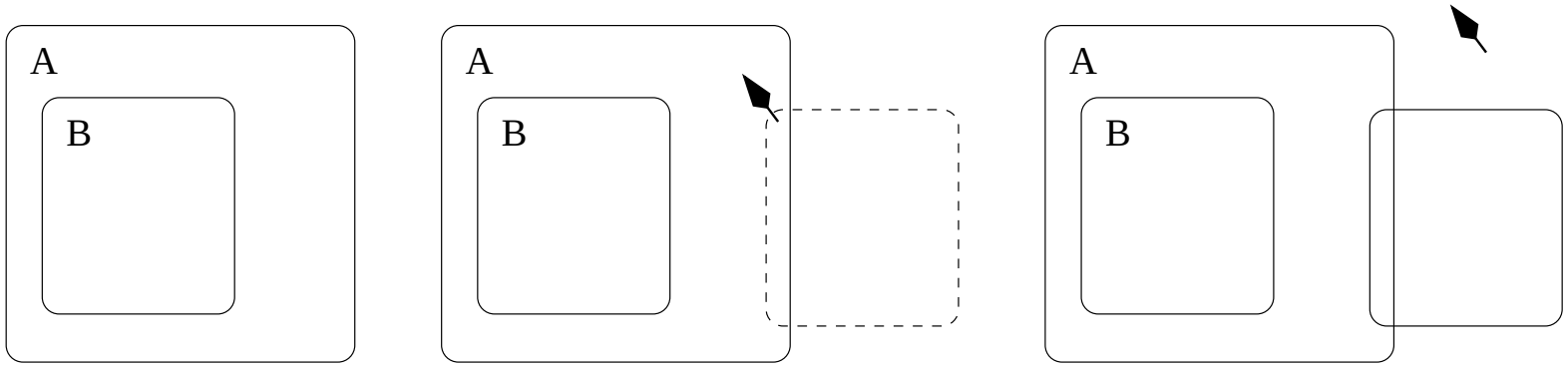
Left-right, top-bottom ordering. Crossing of lines significant.

Elegant algebraic model derived from simple geometric one.

Geometric models in tools

OO implementations distribute concrete model over "view-objects".

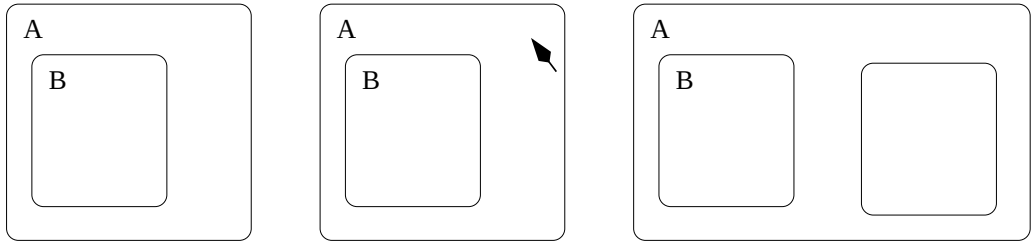
Blob insertion at the concrete level:



(a) (b) (c)

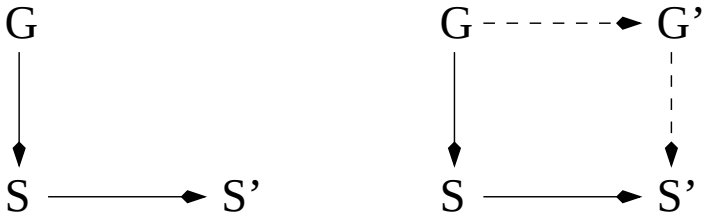
Consistency (i.e. embedding) checking is hard to maintain.

A more principled approach



(d) (e) (f)

Centralise and separate geometric model from structural:



When is G' inferrable? What makes it special among others?

Concluding remarks

Geometric models

- establish "visualisability" and "embeddability" of abstract ones
- inform language design decisions
- highlight unusual situations
- are not necessarily intractable
- should be well-sepatated from structural models in tools

Attributes are geometrically uninterpreted entities.

Attributed and other concrete models

- often leave much of what is visual out
- implicitly assume specific realisations
- force design decisions

But who wants twisted rectangles anyway?

No one, but the issues still remain.