Electronic Notes in Theoretical Computer Science 45 (2001)
URL: `http://www.elsevier.nl/locate/entcs/volume45.html` 12 pages

An Algebraic Foundation for Graph-based Diagrams in Computing

John Power [1,3] and Konstantinos Tourlas [2,4]

Division of Informatics
The University of Edinburgh
Edinburgh EH9 3JZ
United Kingdom

Abstract

We develop an algebraic foundation for some of the graph-based structures underlying a variety of popular diagrammatic notations for the specification, modelling and programming of computing systems. Using hypergraphs and higraphs as leading examples, a locally ordered category $Graph(C)$ of graphs in a locally ordered category C is defined and endowed with symmetric monoidal closed structure. Two other operations on higraphs and variants, selected for relevance to computing applications, are generalised in this setting.

1 Introduction

Recent years have witnessed a rapid, ongoing popularisation of diagrammatic notations in the specification, modelling and programming of computing systems. Most notable among them are Statecharts [4], a notation for modelling reactive systems, and the Unified Modelling Language (UML) [10], a family of diagrammatic notations for object-based modelling. Invariably, underlying such complex diagrams is some notion of graph, upon which labels and other linguistic or visual annotations are added according to application-specific needs (see e.g. [10,9,3] for a variety of examples).

Beyond ordinary graphs, the two leading examples studied here are hypergraphs and higraphs [5]. The latter underlie a number of sophisticated diagrammatic formalisms including, most prominently, Statecharts, the state

[1] This work has been done with the support of EPSRC grant GR/M56333 and a British Council grant, and the COE budget of STA Japan.

[2] Support EPSRC grant GR/N12480 and of the COE budget of STA Japan is gratefully acknowledged.

[3] Email: `ajp@dcs.ed.ac.uk`

[4] Email: `kxt@dcs.ed.ac.uk`

diagrams of UML, and the domain-specific language Argos [8] for programming reactive systems. Higraphs allow for vitally concise, economical representations of complex state-transition systems, such as those underlying realistic reactive systems, by drastically reducing the number of edges required to specify the transition relation. This is achieved by replacing a number of transitions having, say, a common target state with a *single* transition having the same target but with source a new "super-state" containing all the source states of the original transitions. The resulting reduction in complexity is of the order of n^2, where n is the number of states.

We begin our analysis by observing that graphs, hypergraphs and higraphs are all instances of the same structure, that of a graph in a category C, with C being respectively Set, Rel and $Poset$. Other variants are also considered. The case of higraphs is motivated and studied extensively and concretely in the draft paper [13]. The latter assumes only elementary knowledge of category theory on the part of the reader, so as to be accessible to a wide audience of computer scientists who have immediate scientific and practical interest in higraphs and their applications in UML and Statecharts. In the present paper, Section 2 introduces our leading examples, followed by a definition in Section 2.4 of a category $Graph(C)$ of graphs in a locally ordered category C.

Underlying Statecharts is a binary operation which given Statecharts S and S' yields a third corresponding to the semantics of S and S' operating concurrently. We show how the same applies to higraphs and hypergraphs. Here we formulate this precisely and uniformly in algebraic terms by defining a symmetric monoidal closed structure on $Graph(C)$. We do so in Section 3. It is further shown that symmetric monoidal closed adjunction linking $Graph(C)$ to $Cat(C)$ exists when the latter category bears a generalisation of the "other" symmetric monoidal closed structure on Cat.

Hierarchies of edges in higraphs are exploited in practical applications to produce concise specifications of complex reactive systems. To understand the meaning of higher-level edges we introduce in Section 4 a completion operation on higraphs. This is shown to be an instance of the right adjoint to the inclusion of $Graph(C)$ into $Graph_{opl}(C)$, the latter having oplax natural transformations as arrows. A theorem stating conditions for the existence of such right adjoints is proved.

To support users in working with large, hierarchically structured diagrams representing complex systems, one requires effective mechanisms for re-organising, abstracting and filtering the information present in diagrams [9]. The leading example studied here is of a filtering operation on higraphs, introduced and motivated by Harel in [5] under the name of *zooming out.* We show in Section 5 how it generalises to graphs in non-trivially locally ordered categories.

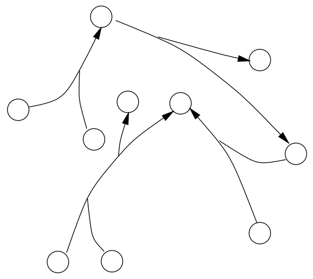


Fig. 1. A simple hypergraph.

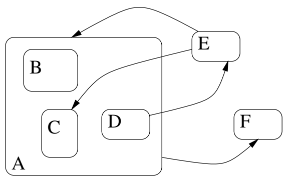


Fig. 2. A simple higraph.

2 Leading examples and main definition

We begin by recalling the standard definition of a (directed, multi-)*graph* as consisting of a set V of vertices, a set E of edges and two functions $s, t : E \longrightarrow V$ giving the source and target of each edge. That is, a graph is a pair of parallel arrows $s, t : E \longrightarrow V$ in the category *Set*.

2.1 Hypergraphs

Hypergraphs are a generalisation of graphs in which each edge may have sets of vertices as its source and target. The typical pictorial representation of this kind of directed hypergraph is illustrated in Figure 1.

Thus, a hypergraph consists of a set V of vertices, a set E of edges and two functions $s, t : E \longrightarrow 2^V$ giving sources and targets. Equivalently, s and t may be seen as relations from E to V, thus arriving at the following

Definition 2.1 *A hypergraph is a pair of parallel maps in the category Rel of (small) sets and relations.*

2.2 Higraphs

Higraph is a term coined-up by Harel[5] as short for *hierarchical graph*, but is often used to include several variants. The definitive feature of higraphs, common to all variants, is referred to as *depth*, meaning that nodes may be contained inside other nodes. Figure 2 illustrates the standard pictorial representation of a higraph consisting of six nodes and four edges, with the nodes labelled B, C and D being spatially contained within the node labelled A. It is therefore common, and we shall hereafter adhere to convention, to call the nodes of a higraph *blobs*, as an indication of their pictorial representation by convex contours on the plane. For further details the reader is referred to [13].

The containment relation on blobs is captured by requiring poset structure on the set of blobs. The notion of higraph developed here extends this requirement to the set of edges:

Definition 2.2 *A higraph is a pair of parallel arrows $s, t : E \longrightarrow B$ in the category Poset.*

In practice, a higraph typically arises as a graph (B, E, s, t) together with a partial order $\leq_B$ on B. In that case, the poset structure on E may be taken to be the discrete one. However, other choices of orders on E are often useful, e.g. for encoding the conflict resolution schemes [6] adopted in Statecharts.

In most applications of higraphs, especially Statecharts, the intuitive understanding of en edge e is as *implying* the presence of "lower-level", *implicit* edges from all blobs contained in $s(e)$ to all blobs contained in $t(e)$. The point in general is that a multitude of edges is made implicit in a single, explicitly shown higher-level edge. In Statecharts, this device is employed for representing *interrupt transitions*, thus drastically reducing the number of edges required to specify the transition relation among the states of the represented transition system.

2.3 Combinations and variants

To deal with realistic diagrams, one may additionally wish to combine features found in different notions of graph, e.g. to allow edges in higraphs to have multiple sources and targets, as is indeed allowed in some Statecharts. The resulting notion of graph, a combination of simple higraphs (as defined above) and hypergraphs, could be approached by considering the category of posets and relations between their underlying sets. The category *BSup* of posets with all binary sups (and sup-preserving monotone maps) gives a better model of depth in Statecharts. One may also consider graphs in the category ω-Cpo of ω-complete partial orders.

2.4 Graphs in locally ordered categories

Each of our leading examples of "notions of graph" has been cast in terms of a pair of parallel maps in a suitable category C. Another, less obvious commonality among our examples is that C has been a *locally ordered* category, i.e. a category enriched in the cartesian closed category *Poset* of posets, a fact of which substantial use will be made later. (The category *Set* is locally ordered in a trivial sense: each hom-object is a discrete poset.) Generalising from our situation one has:

Definition 2.3 *Let C be a locally ordered category. Let $Graph(C)$ denote the locally ordered category of graphs in C, that is the functor category $[\cdot \rightrightarrows \cdot, C]$ where the category $\cdot \rightrightarrows \cdot$ consists of two objects and two non-identity maps as shown.* □

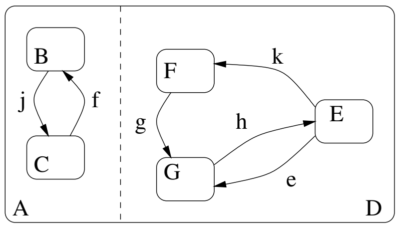


Fig. 3. A simple Statechart

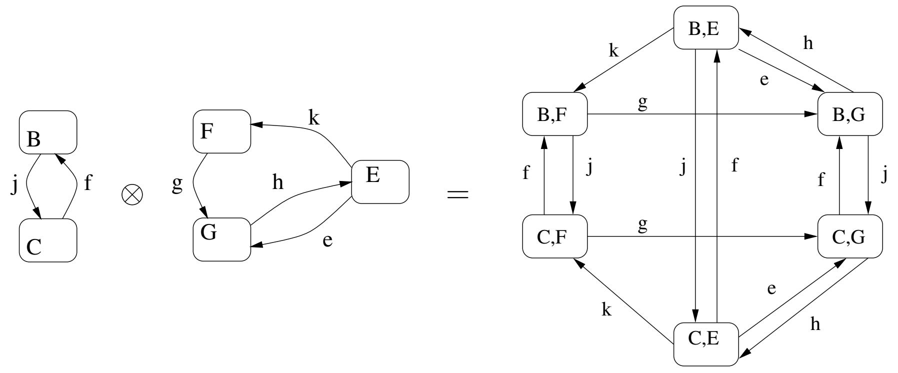


Fig. 4. Operation underlying the Statechart of Fig. 3

So, an object of $Graph(C)$ consists of a pair of objects E and V of C, together with a pair of maps $s,t : E \longrightarrow V$ in C. An arrow of $Graph(C)$ from $(E,V,s,t : E \longrightarrow V)$ to $(E',V',s',t' : E' \longrightarrow V')$ consists of maps $f_E : E \longrightarrow E'$ and $f_V : V \longrightarrow V'$ such that $f_V s = s' f_E$ and $f_V t = t' f_E$. The local order of $Graph(C)$ is generated by that of C, i.e., $(f_E, f_V) \leq (g_E, g_V)$ if $f_E \leq g_E$ and $f_V \leq g_V$.

3 A symmetric monoidal closed structure on $Graph(C)$

We now proceed to study some extra structure on $Graph(C)$, for well-behaved C. Our motivation arises from the application of higraphs in Statecharts. Specifications of complex reactive systems directly in terms of transition systems become impractical to visualise owing to the large number of states involved. Statecharts deal with this problem by allowing the modelling of reactive systems directly in terms of their identifiable concurrent subsystems:

Example 3.1 *Consider the Statechart in Figure 3 representing two subsystems A and D operating concurrently. Assuming an interleaving model of concurrency, as is the case with Statecharts, the meaning of this picture is captured precisely by the operation where the resulting transition system is exactly the intended behaviour of the complete system.* □

A consequence of our results in this section is that the above operation, which in [5] is referred to as “a sort of product of automata”, generalises smoothly to higraphs. This is an essential step in pinpointing the precise mathematical structures underpinning the semantics of Statecharts. For, more generally, the specifications of the subsystems A and D in Figure 3 typically bear higraph structure.

So for our next main result, we observe that, generalising the situation for $C = Set$ in Example 3.1, here not requiring local order structure on C, we have

Theorem 3.2 *For any cartesian closed category C with finite coproducts, the category $Graph(C)$ has a symmetric monoidal structure given as follows: given $G = (E, V, s, t)$ and $G' = (E', V', s', t')$, the graph $G \otimes G'$ has vertex object $V \times V'$ and edge object $(E \times V') + (V \times E')$, with source and target maps evident. The unit of this symmetric monoidal structure is given by $V = 1$ and $E = 0$.*

Proof. That $\otimes$ is a bifunctor follows directly from the properties of the binary products and coproducts in C. The required isomorphisms are easily deduced from those associated with the symmetric monoidal structure induced on C by its cartesian structure, and the verification of the required coherence conditions is routine. $\square$

Example 3.3 *On higraphs $\otimes$ yields a straightforward generalisation of the operation in Figure 4. Specifically $\chi \otimes \chi'$ contains an edge $\langle b_1, b' \rangle \rightarrow \langle b_2, b' \rangle$ for every edge $b_1 \rightarrow b_2$ in χ and blob b' in χ', and an edge $\langle b, b'_1 \rangle \rightarrow \langle b, b'_2 \rangle$ for every edge $b'_1 \rightarrow b'_2$ in χ' and blob b in χ. Containment is given by $\langle b_1, b'_1 \rangle \leq \langle b_2, b'_2 \rangle$ iff $b_1 \leq b_2$ and $b'_1 \leq b'_2$. In the case of hypergraphs, $H \otimes H'$ contains an edge $\{\langle x_1, x' \rangle, \ldots, \langle x_n, x' \rangle\} \rightarrow \{\langle y_1, x' \rangle, \ldots, \langle y_m, x' \rangle\}$ for each edge $\{x_1, \ldots, x_n\} \rightarrow \{y_1, \ldots, y_m\}$ in H and vertex x' in χ', and similarly for the edges in H'.*

Theorem 3.4 *For any cartesian closed category C with finite coproducts and finite limits, the symmetric monoidal structure on $Graph(C)$ given in Theorem 3.2 is closed.*

Proof. The exponential object $[G', G'']$ has object of vertices the domain of the equaliser of the two maps from $[V', V''] \times [E', E'']$ to $[E', V'] \times [E', V']$ given by $\langle [s', V'], [t', V'] \rangle \circ \pi_0$ and $\langle [E', s'], [E', t'] \rangle \circ \pi_1$ where π_0, π_1 are the projections from $[V', V''] \times [E', E'']$. The object of edges of $[G', G'']$ is the domain of the equaliser of the maps $\langle \pi_0 \circ q \circ \pi'_0, \pi_0 \circ q \circ \pi'_2 \rangle$ and $\langle [V', s''] \circ \pi'_1, [V', t''] \circ \pi'_1 \rangle$, both having domain $V \times [V', E''] \times V$ and codomain $[V', V''] \times [V', V'']$, where π'_i are the three projections out of $V \times [V', E''] \times V$. $\square$

Notice, in particular, that the exponential in the category $Graph(C)$ with the tensor product defined in the theorem is particularly natural. The object of vertices represents all graph homomorphisms from G to G', and the object of edges represents all transformations between graph homomorphisms.

3.1 A symmetric monoidal closed adjunction

It is well known that one may define categories in any category C with finite limits, the usual category Cat being isomorphic to the category of models $Cat(Set)$ in Set of an appropriate finite limit sketch [1]. We shall write $Cat(C)$ for the category of categories in C, implicitly asserting C to have finite limits as required.

While it is well known that Cat is a cartesian closed category, it is far less well known that there is precisely one other symmetric monoidal closed structure on Cat [2,12]. We refer to the other one as the *other* symmetric monoidal closed structure on Cat, which may be outlined as follows:

- The exponential $A \longrightarrow B$ is given by the set of functors from A to B, with a morphism from g to h being the assignment of an arrow $\alpha_x : gx \longrightarrow hx$ to each object x of A. The composition is obvious. We shall call an arrow of $A \longrightarrow B$ a *transformation*.
- The tensor product may be described in terms of a universal property: it is the universal D for which one has, for each object x of A, a functor $h_x : B \longrightarrow D$ and for each object y of B, a functor $k_y : A \longrightarrow D$ such that $h_x y = k_y x$ for each (x, y). The unit of the tensor product is the unit category.

Explicitly, the tensor product $A \otimes B$ of A and B has as object set $ObA \times ObB$, and an arrow from (x, y) to (x', y') consists of a finite sequence of non-identity arrows, with alternate arrows forming a directed path in A, and the others forming a directed path in B. Composition is given by concatenation, then cancellation accorded by the composition of A and B. The symmetry is obvious.

It is routine to verify that if, in addition to having finite limits, C is cocomplete and cartesian closed, the other symmetric monoidal closed structure extends to $Cat(C)$. We are now in position to state our theorem relating $Cat(C)$ to $Graph(C)$:

Theorem 3.5 *For a cocomplete cartesian closed category C with finite limits, the forgetful functor $U : Cat(C) \longrightarrow Graph(C)$ is part of a symmetric monoidal closed adjunction with respect to the other tensor product on $Cat(C)$ and the above symmetric monoidal closed structure on $Graph(C)$.*

Proof. For a proof, consider the case that C is Set and simply internalise the argument there. $\square$

Note that a corresponding result does not hold for the cartesian closed structures of $Cat(C)$ and $Graph(C)$ even in the case of $C = Set$, so we regard this result as strong evidence of the naturalness of this structure. Finally, in this vein, we observe

Theorem 3.6 *For cartesian closed C with finite coproducts, the forgetful functor from $Graph(C)$ to C is part of a symmetric monoidal closed adjunc-*

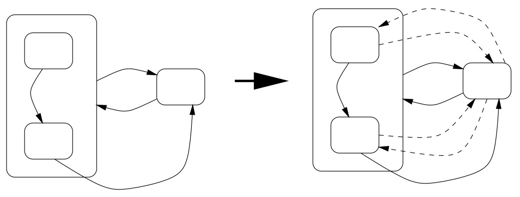


Fig. 5. Completion of a simple higraph, where the added edges are shown dashed.

tion with respect to the above symmetric monoidal structure on Graph(C).

Proof. For a proof, consider the proof in the case of $C = Set$ and routinely internalise it to C. □

Again, even in the case of $C = Set$, a corresponding result does not hold in respect of the cartesian closed structure of $Graph(C)$ as the left adjoint does not preserve the unit, i.e., it does not send 1 to the terminal object of *Graph* as the latter has an edge.

4 A completion operation

A construction useful in understanding the semantics of higraphs and variants (for instance that involving the categories *BSup* or ω-Cpo) is to explicate all edges which are understood as being implicitly present in a higraph (recall the discussion near the end of Section 2.2). This "completion" operation is illustrated in Figure 5.

Definition 4.1 *Let* $\chi = s, t : E \longrightarrow B$ *be a higraph. The higraph* $T(\chi)$*, called the* completion *of* χ*, has blobs* B *and edges the subset of* $E \times (B \times B)$ *consisting of those pairs* $\langle e, \langle b, b' \rangle \rangle$ *such that* $b \leq_B s(e)$ *and* $b' \leq_B t(e)$*, partially ordered pointwise, with source and target given by projections.* □

Definition 4.2 *Given a locally ordered category* C*, we denote by* $Graph_{opl}(C)$ *the locally ordered category whose objects are graphs in* C *and whose arrows are oplax transformations, i.e. pairs* $(f_E : E \longrightarrow E', f_V : V \longrightarrow V')$ *such that* $f_V s \leq s' f_E$ *and* $f_V t \leq t' f_E$*, with local order structure induced by that of* C*.* □

To state our theorem, it is convenient to use a little of the theory of 2-categories, specifically some finite limits. A convenient account of such limits is [7]. In particular, we need to use the notion of an oplax limit of a map. So we recall it here.

Definition 4.3 Given an arrow $f : X \longrightarrow Y$ in a locally ordered category C, an *oplax limit* of f is given by a diagram of the form

$$\begin{tikzcd} L \arrow[r, "\pi_o"] \arrow[d, "id"'] \arrow[dr, phantom, "\leq"] & X \arrow[d, "f"] \\ L \arrow[r, "\pi_1"'] & Y \end{tikzcd}$$

satisfying two properties:

- for any other diagram of the form

$$\begin{tikzcd} K \arrow[r, "h_0"] \arrow[d, "id"'] \arrow[dr, phantom, "\leq"] & X \arrow[d, "f"] \\ K \arrow[r, "h_1"'] & Y \end{tikzcd}$$

  there is a unique arrow $u : K \longrightarrow L$ such that $\pi_0 u = h_0$ and $\pi_1 u = h_1$, and
- (the two-dimensional property) for any two diagrams of the form

$$\begin{tikzcd} K \arrow[r, "h_0"] \arrow[d, "id"'] \arrow[dr, phantom, "\leq"] & X \arrow[d, "f"] \\ K \arrow[r, "h_1"'] & Y \end{tikzcd} \qquad \begin{tikzcd} K \arrow[r, "h_0^{'}"] \arrow[d, "id"'] \arrow[dr, phantom, "\leq"] & X \arrow[d, "f"] \\ K \arrow[r, "h_1^{'}"'] & Y \end{tikzcd}$$

  with $h_0 \leq h_0^{'}$ and $h_1 \leq h_1^{'}$, it follows that $u \leq u'$.

□

Theorem 4.4 *If the locally ordered category C has finite limits, then the inclusion of $Graph(C)$ into $Graph_{opl}(C)$ has a right adjoint.*

Proof. Given a graph $G = (E, V, s, t)$, the right adjoint has vertex object given by V and object of edges given by the oplax limit of the map $\langle s,\, t\rangle : E \longrightarrow V \times V$. It is a routine exercise in 2-categories to prove that this construction yields a right adjoint. □

The 2-category theory expert will observe that we have only used pie-limits in C, which may become important in due course [11]. Perhaps a more familiar expression for the oplax limit used in the proof is in terms of a comma object in C from the identity map on $V \times V$ to the map $\langle s,\, t\rangle : E \longrightarrow V \times V$. If

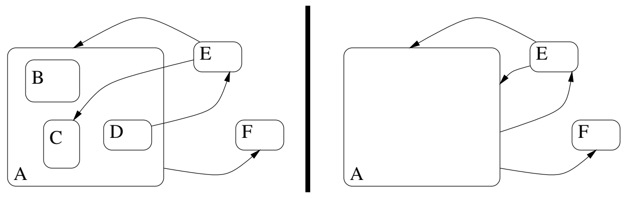


Fig. 6. Zooming out of a blob in a higraph

C were the locally ordered category *Poset*, then the right adjoint could be described explicitly by placing an edge from v to v' if there is an edge from a vertex greater than or equal to v to a vertex greater than or equal to v' in G. This matches exactly our explicit description of T in Definition 4.1.

Dually, if C has finite colimits, the inclusion of $Graph(C)$ into $Graph_{opl}(C)$ has a left adjoint.

5 Zooming out

We begin by recalling Harel's simple instance of a zooming operation on higraphs: the selection of a single blob and the subsequent removal from view of all blobs contained in it. An example is illustrated in the transition from the left to the right half of Figure 6.

To capture the notion of selecting a blob in a higraph we need the following:

Definition 5.1 *A* pointed higraph ψ *consists of an ordinary higraph* $\chi = s, t : E \longrightarrow B$ *together with a distinguished blob, given as a map* $1 \longrightarrow B$ *in Poset and called the* point *of* ψ. *The category* $\mathcal{H}_\star$ *has pointed higraphs as its objects and maps those ones which preserve points. Let* $\mathcal{H}_{\star,min}$ *be the full subcategory of* $\mathcal{H}_\star$ *consisting of all objects (pointed higraphs) in which the point is minimal wrt. the partial order on blobs; in other words, the point is an atomic blob. Let* I *be the full functor including* $\mathcal{H}_{\star,min}$ *into* $\mathcal{H}_\star$. □

Consider a pointed higraph ψ with $\chi = (s, t : E \longrightarrow B)$ and point, say, $p \in B$. The pointed higraph $Z(\psi)$, obtained by zooming out of the point in ψ, is determined by the following data:

- blobs: $B' = B \setminus \{b \mid b < p\}$ (ordered by the restriction to B' of the partial order on B);
- edges: E, with the source and target functions being $q \circ s$ and $q \circ t$ respectively, where $q : B \longrightarrow B'$ is the (obviously monotone) function mapping each $b \not< p$ in B to $b \in B'$ and each $b < p$ to $p \in B'$;
- point: p

One now has the following [13]:

Proposition 5.2 *The function* Z *extends to a functor from* $\mathcal{H}_\star$ *to* $\mathcal{H}_{\star,min}$ *which is left adjoint to the inclusion functor* I. □

This proposition will be shown an instance of Theorem 5.5 below. Generalising the essential structure underlying our leading example one has:

Definition 5.3 *Given a locally ordered category C, denote by $Graph(C)_*$ the locally ordered category for which an object consists of a graph (E, V, s, t) in C together with a map $v : 1 \longrightarrow V$ in C. The maps are pairs of maps that strictly preserve the structure.* □

Definition 5.4 *Given a locally ordered category C, denote by $Graph(C)_{*min}$ the locally ordered full subcategory of $Graph(C)_*$ such that the point $v : 1 \longrightarrow V$ is a minimal element in the poset $C(1, V)$.* □

Theorem 5.5 *If C is a cocomplete locally ordered category, then the inclusion of $Graph(C)_{*min}$ in $Graph(C)_*$ has a left adjoint.*

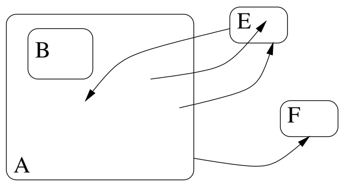
Proof. Given (E, V, s, t) and $v : 1 \longrightarrow V$, take the joint coequaliser of v with all of the elements of the poset $C(1, V)$ that are less than or equal to it. It is routine to verify that this gives the left adjoint. □

Example 5.6 *For graphs in BSup the theorem gives the expected generalisation of the zoom-out operation on graphs in Poset in the presence of the extra structure given by binary sups. However, zoom-outs do not generalise to graphs in Rel, or the category of posets and relations between their underlying sets, as the terminal object is the empty set (poset).*

6 Further work

Our aim is to develop, in an incremental and principled way, structures which bear sufficient detail to model realistic diagrammatic notations. Currently we are working towards providing such a model for a large class of Statecharts, which include features found in higraphs and hypergraphs. The work herein presented lays the abstract foundations for our approach, in which notions of graph and combinations thereof may be studied.

Another strand of our work is to study extensions to such notions of graph, as required to support users in performing specification and reasoning tasks with diagrams. For instance, a mild extension to higraphs was briefly introduced by Harel in [5], permitting edges to be "loosely" attached to nodes, the four possibilities being illustrated in



.

The rationale was to indicate transitions or relations between some as yet unspecified, or purposefully omitted (e.g. as the result of zooming out) parts of the represented system. For motivation and details the reader is referred to [13]. We conclude by noting that such graphs with "loose edges" can be added easily to our framework, provided that the locally ordered category C

has finite (pie) colimits, thereby allowing one to define tensors with the arrow poset.

References

[1] M. Barr and C. Wells. *Category Theory for Computing Science.* Prentice-Hall, 1990.

[2] F. Foltz, C.M. Kelly, and C. Lair. Algebraic categories with few monoidal biclosed structures or none. *Journal of Pure and Applied Algebra*, 17:171–177, 1980.

[3] Corin Gurr and Konstantinos Tourlas. Towards the principled design of software engineering diagrams. In *Proceedings of the 22nd International Conference on Software Engineering*, pages 509–520. ACM, IEEE Computer Society, ACM Press, 2000.

[4] David Harel. Statecharts: A visual approach to complex systems. *Science of Computer Programming*, 8(3):231–275, 1987.

[5] David Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988.

[6] David Harel and Amnon Naamad. The STATEMATE semantics of Statecharts. *ACM Transactions on Software Engineering Methodology*, 5(4), October 1996.

[7] G.M. Kelly. Elementary observations on 2-categorical limits. *Bull. Austral. Math. Soc.*, pages 301–317, 1989.

[8] F. Maraninchi. The Argos language: Graphical representation of automata and description of reactive systems. In *Proceedings of the IEEE Workshop on Visual Languages*, 1991.

[9] Bonnie M. Nardi. *A Small Matter of Programming: Perspectives on End-User Computing.* MIT Press, 1993.

[10] Rob Pooley and Perdita Stevens. *Using UML.* Addison Wesley, 1999.

[11] A.J. Power and E.P. Robinson. A characterization of pie-limits. *Math. Proc. Cambridge Philos. Soc.*, 110:33–47, 1991.

[12] John Power and Edmund Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Comp. Science*, 11, 1993.

[13] John Power and Konstantinos Tourlas. An algebraic foundation for higraphs. Submitted for publication, March 2001.