Reasoning in Higraphs with Loose Edges

Stuart Anderson John Power Konstantinos Tourlas*
Laboratory for Foundations of Computer Science, Division of Informatics
University of Edinburgh, The King’s Buildings, Edinburgh EH9 3JZ
e-mail: {soa, ajp, kxt}@dcs.ed.ac.uk

Abstract

Harel introduces the notion of zooming out as a useful operation in working with higraphs. Zooming out allows us to consider less detailed versions of a higraph by dropping some detail from the description in a structured manner. Although this is a very useful operation it seems it can be misleading in some circumstances by allowing the user of the zoomed out higraph to make false inferences given the usual transition system semantics for higraphs. We consider one approach to rectifying this situation by following through Harel’s suggestion that, in some circumstances, it may be useful to consider higraphs with edges that have no specific origin or destination. We call these higraphs loose higraphs and show that an appropriate definition of zooming on loose higraphs avoids some of the difficulties arising from the use of zooming. We also consider a logic for connectivity in loose higraphs.

1 Introduction

Our emphasis here is on making precise the issues surrounding the semantic import of common syntactic operations on diagrams. Our motivation is that, in practice, diagrams used in the design, construction and analysis of systems are highly dynamic, constantly manipulated and altered in the course of design and reasoning about computing systems. For this reason it is important to consider how precisely to define the kinds of operations that are common in this activity in order that we can begin to provide adequate tool support for the manipulation of diagrams with well-defined semantic content.

Even diagrams that are designed with economy and compactness in mind may still grow impractically large, or simply become too detailed to be effective in supporting the user perform a particular task. Often this is because the same diagram simultaneously conveys multiple aspects of the represented system, perhaps describing the system at a number of distinct levels or using an array of pragmatic devices to indicate associations that are important in the understanding of the design. Because diagrams are often very rich descriptions of systems one needs effective mechanisms, and tools to support them, for re-organising, abstracting and filtering the information present in diagrams [6].

The leading example studied here is a filtering operation on higraphs, introduced briefly and motivated by Harel in [3] under the name of zooming out. This provides an operation essential in managing the kind of layered description encouraged by the higraph approach to systems design.

Our broader goal is to provide a secure foundation for the development of tools to support the use of diagrams in systems design and analysis. Thus our emphasis is always on maintaining a clear relationship between the syntactic appearance of the diagram and its semantic interpretation while attempting to encompass the range of syntactic mechanisms that are employed to support the user in interpreting diagrams.

Section 3 introduces higraphs and their most common semantic interpretations: as relations and as state-transition systems. In Section 4 we then go on to consider a simple definition of zooming on higraphs and demonstrate in Section 5 that in some cases zooming can be misleading. In Section 6 we introduce higraphs with loosely connected edges and in Section 7 we define a more refined notion of zooming on these newly defined higraphs. Finally in Section 8 we consider logics that capture some of the expressiveness of the new definitions and demonstrate how the confusion arising in our earlier example is eliminated.

2 On the use of visual formalisms in system modelling and design

In her book on end-user programming [6], Nardi studies a variety of domain-specific (or task-specific) programming languages which feature a strong “visual” or diagrammatic flavour. Typically, the design of such languages is

* We acknowledge the support of the UK EPSRC project Domain Specific Representation and Reasoning, GR/N12480/01
centered around a “core” visual formalism, such as the notion of table, graph etc., which is appropriately extended with domain-specific features, or specialised to accommodate domain-specific constraints. Elements of the visual formalism, such as nodes, cells etc., are typically associated with domain-specific objects, either visually or by means of linguistic (i.e. textual) annotation. A spreadsheet, for instance, may be considered as a rudimentary programming language for computing with expressions arranged in a two-dimensional table of cells, the table being the underlying visual core.

However, most of the issues identified in [6] pertain not only to the use of visual formalisms in programming (i.e. to software systems) but also to the design and modelling of more general notions of computing system. Central among these issues is the need to identify, study and support the kind of common manipulations which users routinely perform on diagrams in the course of design.

2.1 Common manipulations

One aim of domain-specific notations is to provide representations that support the user in the kinds of tasks they are likely to carry out in working with the representation. Both in system design and analysis it is very common that we want to control the level of detail.

Often this is done by “chunking” some subsystem and representing it as a single object. For example, Figure 1 shows successive “chunkings” of a graph by hiding detail.

![Figure 1. Incremental “chunking” on a graph.](image1)

Our goal in this paper is to explore exactly this kind of complexity management in the context of higraphs.

2.2 The role of formal methods

Sequences of diagrams, such as the one in Figure 1, may be generated either by the user in the course of developing an argument, or by the system in providing the user with justification or a counter example wrt. some property. We need to formalise how each diagram in the sequence relates to its predecessor, how their semantics are related, and what it means for a property to be unaffected by abstracting away parts of a diagram. This is what we do here in the context of higraphs and zooming.

3 Higraphs and Statecharts

Higraphs extend graphs by permitting spatial containment among the nodes, and were originally proposed by Harel [3] as the core visual formalism underlying Statecharts [2]. Figure 2 illustrates the pictorial representation of a simple 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. A blob is called atomic if no other blobs are contained in it. The feature of spatial containment is often referred to as depth, leading to an expression of the relationship of higraphs to graphs in terms of Harel’s “equation”:

\[
\text{higraphs} = \text{graphs} + \text{depth}^1
\]

3.1 Transition semantics of higraphs

The main application of higraphs has been in the specification and visualisation of complex state-transition systems, manifested mainly in Statecharts and, more recently, in the state diagrams of UML. In such applications, depth is used both as a conceptual device, in decomposing the overall system into meaningful subsystems, and as an economical and effective representation of interrupts. In terms of our example higraph in Figure 2, the edge emanating from blob A

![Figure 2. A simple higraph.](image2)

Higraph is a term coined-up by Harel[3] as short for hierarchical graph, but often used quite liberally to include several variants. The view taken here is that depth is the most distinguishing, definitive feature of higraphs, common to all variants. Harel’s original definition includes an extra feature which he called orthogonality and which is not treated here. It is our conviction, supported by preliminary results outside the scope of the present paper, that orthogonality can, at least mathematically, be regarded as an extension to the basic, “depth-only” higraphs considered here.
may be regarded as a higher-level transition interrupting the operation of the subsystem comprising states B, C and D.

When applied at multiple levels, depth therefore facilitates the concise representation of large systems by drastically reducing the number of edges required to specify the transition relation among states. Thus, for instance, the higraph

concisely represents the following “flat” transition system:

In the preceding example, the states B, C and D contained in state A are said to be the OR-decomposition of state A. If the system is in any one of the states B, C or D (but only one of them at any given time) then it is also in state A.

The reader should note that the transition interpretation of higraphs is more general than that of Statecharts. In Statecharts, a facility exists which, in terms of our example above, allows one to annotate either B or C as the default state within A. Such annotation forces the arrow from D to A to be a transition from D to the default state in A, thereby eliminating non-deterministic choice. Although such a device may readily be added to higraphs, we have chosen not to do so in the present paper for reasons of generality and simplicity of exposition.

3.2 Relational semantics of higraphs

Higraphs have also been used in extensions of other modelling notations, such as the Entity-Relationship (ER) diagrams popular in database analysis and design [3]. In that and similar applications, a relational (or graph-theoretic) interpretation of higraphs is appropriate. Thus, blobs denote sets (subsets of some universe set or “domain” D) and containment is directly interpreted as set inclusion. That is, if \([X]\) means the set assigned to blob X and blob B is contained in blob A, then \([B]\) \(\subseteq [A]\) \(\subseteq D\). An edge from blob A to blob B means that \((a, b)\) is an instance of the represented relation for each \(a \in [A]\) and \(b \in [B]\). Further details and a formal definition of this interpretation of higraphs may be found in [3].

Of the two interpretations, transition based and relational, here we are primarily interested in the former as it is computational in nature: it describes the dynamics (i.e. temporal evolution of behaviour) of a computational system in terms of states and transitions among them. By contrast, the relational interpretation is static in nature, emphasising structural relationships rather than evolution. Below we shall contrast the two and illustrate that in relation to the transition system interpretation the zooming operation can be misleading.

3.3 Higraphs, formally

Our definition of higraph is based on Harel’s [3] but uses posets (i.e. partially ordered sets) to capture the notion of depth and extends it also to the edges:

Definition 1. A higraph is a 5-tuple \((B, \leq_B, E, \leq_E, s, t)\), where \(B\) and \(E\) are respectively the sets of blobs and edges, \(\leq_B\) is a partial order on \(B\), \(\leq_E\) is a partial order on \(E\), and \(s, t : E \rightarrow B\) are monotone functions giving, for each edge \(e \in E\), its source \(s(e)\) and target blob \(t(e)\).

Example 1. Figure 2 may be seen as the pictorial representation of a higraph with:

- blobs: \(\{A, B, C, D, E, F\}\) where \(B, C, D < A\)
- edges: \(\{e_1, e_2, e_3, e_4\}\) where \(e_2 < e_1\)
- \(s(e_1) = E, t(e_1) = A, s(e_2) = E, t(e_2) = C\), and so on.

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 or, alternatively, as induced by \(\leq_B\). One possibility of inducing an order on \(E\) is given by: \(e \leq_E e' \) iff \(e = e' \) or \((s(e) <_B s(e')\) and \(t(e) \leq_B t(e')\) \), thereby making both functions \(s\) and \(t\) monotone. Here the order induced on \(E\) partially captures the priority scheme in [4] for resolving conflicts of transitions in Statecharts.

Essentially, each higraph \(\chi\) is a pair of “parallel” monotone functions \(\chi_s = s : \chi_E \rightarrow \chi_B\) and \(\chi_t = t : \chi_E \rightarrow \chi_B\), with common domain the poset \(\chi_E = (E, \leq_E)\) and codomain \(\chi_B = (B, \leq_B)\). Hereafter we shall denote higraphs with \(\chi, \chi', \chi''\), etc., and implicitly decompose them as \(\chi = (s, t : E \rightarrow B), \chi' = (s', t' : E' \rightarrow B')\) and so on, unless specifically indicated otherwise.

4 Zooming out

We base our analysis on the simplest, and most frequently occurring in practice, instance of a zooming-out operation on higraphs: the selection of a single blob and the
subsequent removal from view of all structure (blobs and edges) contained in it. An example, in which edges are conveniently shown labelled for ease of reference, is illustrated in the transition from the left to the right half of Figure 3. Notice, in particular, how the edges attached to the blobs contained in A are subsequently fixed to A.

This filtering operation on higraphs is introduced, albeit briefly and informally, and justified in terms of its practical significance, in [2, 3]. The idea of zooming out is to obtain a higher-level, more abstract view of the represented system by eliminating detail which is deemed by the user as irrelevant to the task at hand. The task in question may be simply the visualisation or communication of a large design. In this case zooming out results in a diagram which is less cluttered than the original, and it may be possible that the missing detail can be presented separately if required. On the other hand, and perhaps more importantly, the task may be inferential in nature (i.e. a reasoning task). Then, the intention of zooming out is to simplify the reasoning argument, but without compromising its soundness, by considering a simpler, more abstract diagrammatic representation.

5 Semantic implications of zooming

In this section we discuss some of the semantic issues pertaining to the operation of zooming out, using Figure 3 as a simple example. For convenience let (in this section only) \( \chi \) and \( Z(\chi) \) denote respectively the left and right halves of Figure 3.

Imagine a user performing, typically with the aid of a software tool, the step from \( \chi \) to \( Z(\chi) \) in an attempt to obtain a less detailed representation of the same system. It is important to emphasise that \textit{a priori}, i.e. when viewed in isolation, the two diagrams may represent, in general, two \textit{different} systems. It is the user’s intention to regard them as alternative representations of the same system and we are interested in examining how zooming affects the user’s view of the system when she adopts \( Z(\chi) \) as her current working representation. We consider each interpretation of higraphs in turn.

5.1 Relational interpretation

In this case it is immediate to enforce that, at a minimum, the relations corresponding to the two halves of Figure 3 are defined over the same set(s) (i.e. over the same subset of the domain \( D \)). This is done by insisting that blob A receives the same interpretation in both diagrams: \( [A]_\chi = [A]_Z(\chi) \), and similarly also for the remaining blobs. The enforcement of this constraint may be delegated to the software tool assisting the user.

Even so, the relation defined by \( Z(\chi) \) is still finer (i.e. contains more pairs) than the one defined by \( \chi \). This is so because now all elements in \( [A] \) are related to all elements in \( [E] \), whereas previously only those in \( [D] \subset [A] \) were so related (we assume \([B], [C] \neq \emptyset \)). In any case, the user must be at least warned of working with a finer relation.

There are cases, however, where zooming out preserves the relational semantics exactly. This indeed would be so in our example if edge \( e_3 \) in \( \chi \) had its source at A rather than D.

5.2 Transition system interpretation

Under the transition system interpretation, the possibilities of introducing semantic inconsistencies by zooming out, as well as the consequences, are even more profound. This is because the specified dynamic behaviour is inferred from the representing higraph by considering paths:

**Definition 2.** A path of length \( n \) in a higraph \( (s, t : E \rightarrow B) \) is a sequence \( \langle e_0, \ldots, e_{n-1} \rangle \) of edges in \( E \) such that \( t(e_i) \leq_B s(e_{i+1}) \) or \( s(e_{i+1}) \leq_B t(e_i) \). A path is said to include a blob \( b \) if \( b \) occurs as either the source or target of at least one edge in the path.

Consider now the path consisting of edges \( e_2 \) and \( e_3 \) in \( Z(\chi) \). It suggests that the system, when in state \( E \), can perform an \( e_2 \) transition followed by an \( e_3 \) transition to return to \( E \). This sequence of transitions starting from \( E \) is, however, impossible in \( \chi \). We describe this situation by saying that the path \( \langle e_2, e_3 \rangle \) in \( Z(\chi) \) is \textit{not reflected} by a path in \( \chi \). Thus, in general, the zoomed-out view of a system may contain paths which are not reflected by (possibly longer paths) in the original view.

Most behavioural properties of the system (such as those expressible in some modal or temporal logic) depend on path connectivity. Consequently, there may be behavioural properties implied by the zoomed-out view of a system which are not implied by the original view.

Even if the user is made aware that a blob (such as A in \( Z(\chi) \)) has resulted from a zoom-out, e.g. by visually marking the blob, uncertainty still remains. In general, there is no way of knowing whether a path including the marked blob does indeed correspond to a path in the original higraph,
unless the zoom-out operation is actually undone. To make matters worse, the extent of zoom-out increases with the number of successive zoom-outs performed.

In fact the effect of marking a blob in \( Z(\chi) \) as the product of a zoom-out is no less than compelling the user to regard any path which includes the marked blob as uncertain, i.e. as not necessarily reflected by a path in \( \chi \). What one requires as a minimum is the ability to distinguish visually exactly those paths which are certain, i.e. guaranteed to be reflected by paths in \( \chi \), from those which are not.

The preceding analysis has shown that the naïve notion of single-blob zoom-out presented so far is inadequate to cope with the requirements imposed by the transition system semantics of higraphs. A mild extension to higraphs providing a solution to this problem is briefly introduced by Harel in [3] (and also used in some examples in his earlier paper [2] on Statecharts).

6 Higraphs with loosely attached edges

The required extension to higraphs permits edges to be “loosely” attached to nodes, the four possibilities being illustrated in

An edge such as the one attached to the contours of \( A \) and \( E \) is called firm. The remaining three are non-firm.

The rationale is to indicate transitions or relations between some as yet unspecified parts of the represented system. For instance, the edge from \( E \) to \( A \) represents an edge from \( E \) to some unspecified blob contained in \( A \). Similarly, the lower of the two edges from \( A \) to \( E \) indicates that some unspecified blob contained in \( A \) has an edge to \( E \), but not directly to any blob which may be contained in \( E \). Thus, loosely attached edges are employed as a visual device to indicate not only which blobs may contain some further, unspecified structure, but also which paths are guaranteed. The latter are exactly those paths in which no edge whose target is non-firmly attached is followed by one with nonfirmly attached source.

We cast such an extended higraph with blobs \( B \) as an ordinary one having the same edges but containing two distinct copies \( \langle 0, b \rangle \) and \( \langle 1, b \rangle \) of each \( b \in B \), tagged with 0’s and 1’s. In the pictorial representation of such extended higraphs the convention is that blobs tagged with 0 are not shown at all and that, for instance, an edge with target of the form \( \langle 0, b \rangle \) has its endpoint lying inside the contour picturing \( b \).

Moreover one stipulates that \( \langle 0, b \rangle < \langle 1, b \rangle \) for all \( b \), to capture the intuition underlying the pictorial representation. Any edge \( \langle i, b \rangle \rightarrow \langle j, b' \rangle \) will be called non-firm if \( i = 0 \) or \( j = 0 \). When both \( i = 0 \) and \( j = 0 \) we shall call the edge totally loose.

Let \( A \times B \), where \( A \) and \( B \) are posets, denote their Cartesian product, partially ordered pointwise: \( \langle a, b \rangle \leq \langle a', b' \rangle \) iff \( a \leq_A a' \) and \( b \leq_B b' \).

Definition 3. A higraph with loosely attached edges is a pair \( \phi = s, t : E \rightarrow (\cdot \rightarrow \times B) \) of monotone functions, where \( \cdot \rightarrow \cdot \) is the poset with two elements, 0 and 1, such that \( 0 < 1 \).

Example 2. The picture at the beginning of this section corresponds to a higraph with loosely attached edges, having:

- blobs: \( \{ A, B, E, F \} \) where \( B < A \)
- edges: \( \{ e_1, e_2, e_3, e_4 \} \), say ordered discretely
- \( s(e_1) = \langle 1, E \rangle \), \( t(e_1) = \langle 0, A \rangle \), \( s(e_2) = \langle 0, A \rangle \), \( t(e_2) = \langle 0, E \rangle \), and so on.

For brevity we shall hereafter abuse terminology and refer to higraphs with loosely attached edges as loose higraphs.

7 Zooming out on loose higraphs

A formal definition of zooming out of a single blob in an ordinary higraph was developed in [8, 7], using the methods of category theory [1]. Here we extend this definition to loose higraphs, but without the assistance of category theory so as to make it more widely accessible.

First, to capture the notion of selecting a blob in a loose higraph we introduce the following:

Definition 4. A pointed loose higraph \( \phi_* \) consists of a loose higraph \( \phi = s, t : E \rightarrow (\cdot \rightarrow \times B) \) together with a distinguished blob \( p \in B \) called the point of \( \phi_* \).

Let \( \mathcal{LH}_* \) be the set of pointed loose higraphs. With \( \mathcal{LH}_{*, \text{min}} \) we shall denote the set of all pointed loose higraphs in which the point is minimal wrt. the partial order on \( B \); in other words, the point is an atomic blob.

The operation of zooming out may now be viewed as a function \( Z \) from \( \mathcal{LH}_* \) to \( \mathcal{LH}_{*, \text{min}} \), since, in essence, it reduces the point (selected blob) of \( \phi_* \) to a minimal point in \( Z(\phi_*) \):

Definition 5. Let \( \phi_* \) be a pointed loose higraph with \( \phi = s, t : E \rightarrow (\cdot \rightarrow \times B) \) and point, say, \( p \in B \). Formally, \( Z(\phi_*) \) is determined by the following data:

- blobs: the Cartesian product of \( \cdot \rightarrow \cdot \) with \( B' = B \setminus \{ b \mid b < p \} \) (the latter being 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 $q : (\rightarrow \times B) \rightarrow (\rightarrow \times B')$ is the (obviously monotone) function mapping each $(i, b) \not< (1, p)$ to $(i, b)$ and each $(i, b) < (1, p)$ to $(0, p)$;

• point: $p$.

Thus, any edges emanating from or targeting sub-blobs of the point in $\phi_*$ have their source or target fixed accordingly to lie “inside” the minimal point in $Z(\phi_*)$, i.e. are fixed to the “invisible” blob $(0, p)$. Thus, if any edges exist between any sub-blobs of the point of $\phi_*$, they become totally loose edges lying entirely inside the (now minimal) point of $Z(\phi_*)$. In Harel’s examples, however, such edges are eliminated too. The advantage of our definition lies in its mathematical properties (discussed in [8, 7]), and the extra step of removing all totally loose edges from inside the point of $Z(\phi_*)$ may be delegated to an extra, straightforward operation on pointed loose higraphs.

8 Interpreting logics

We show how a (pointed) loose higraph may be seen as a model for a simple modal logic, in the style of Hennessy and Milner [5], capable of expressing simple path-connectivity properties in the presence of loosely attached edges.

The definitions and results in this section may be seen as a first step towards defining models in higraphs (and, later, Statecharts) of expressive modal and temporal logics [9], such as the mu-calculus. Accommodating loose higraphs right from the start allows the utility of the logics to extend from diagrams representing finished designs to diagrams representing abstract or incomplete ones.

In the rest of this section fix an arbitrary pointed loose higraph $\phi_*$, where $\phi = s, t : E \rightarrow (\rightarrow \times B)$, and let $p$ be its point. We let $\bar{t}$ and $\bar{s}$ denote the composites of $t$ and $s$ with the second projection $(\rightarrow \times B) \rightarrow B$. That is, for instance, $\bar{t}(e) = b$ iff $t(e) = (i, b)$, $i = 0, 1$.

8.1 Exiting from states

In order to define the meaning of the modal operators in the logic, we need a few auxiliary concepts.

First, for any blob $b \in B$ of $\phi_*$ we define $\text{exists}_{\phi_*}(b)$ to be the set of all edges in $E$ which are guaranteed “exists from $b$”, meaning that (seen as transitions) they are guaranteed to lead the represented system out of the state associated with blob $b$. Recalling (from Section 3) that the order on edges may be taken as a priority scheme for resolving conflicts, we define the set $\text{exists}_{\phi_*}(b)$ to consist of the maximal elements in the poset with elements

$$\{ e \in E \mid (1, b) \leq s(e) \text{ and } s(e) = (1, b') \text{ for some } b' \in B \}$$

and order relation inherited from $\leq_E$. In words, and in agreement with the intuitive semantics, all edges in $\text{exists}_{\phi_*}(b)$ have a firm source which contains $b$.

The set $\text{exists}_{\phi_*}(b)$ may now be extended to also include edges which may lead out of $b$:

$$\text{exists}_{\phi_*}(b) \overset{\text{def}}{=} \text{exists}_{\phi_*}(b) \cup \{ e \in E \mid s(e) = (0, b) \}.$$ 

Both $\text{exists}_{\phi_*}(b)$ and $\text{exists}_{\phi_*}(b)$ inherit a partial order structure from $\leq_E$.

8.2 The logic

The syntax of our logic is defined inductively thus:

$$\Psi ::= tt \mid \neg \Psi \mid \Psi_1 \land \Psi_2 \mid \langle \rangle \Psi \mid \langle ? \rangle \Psi.$$ 

Notice that our modal operators $\langle \rangle$ and $\langle ? \rangle$ do not depend on labels, since the edges of $\phi_*$ are not labelled. Although the addition of labels is unproblematic, we have chosen to exclude it from the present paper for reasons of space requirements and clarity of exposition.

Informally, a blob $b$ in $\phi_*$ satisfies $\langle \rangle \Psi$ if there exists at least one edge which is guaranteed to exit $b$ and lead to a blob satisfying $\Psi$. Similarly, $b$ satisfies $\langle ? \rangle \Psi$ if at least one edge exists which may lead out of $b$ to a blob satisfying $\Psi$.

To make this intuition precise we define, for each formula $\Psi$, the set $[\Psi]^{\phi_*}$ of blobs in $\phi_*$ which satisfy $\Psi$. With the aid of the following notation,

$$[\langle \rangle \Psi]^{\phi_*}.B' \overset{\text{def}}{=} \{ b \in B \mid \exists e \in \text{exists}_{\phi_*}(b), \bar{t}(e) \in B' \}$$

$$[\langle ? \rangle \Psi]^{\phi_*}.B' \overset{\text{def}}{=} \{ b \in B \mid \exists e \in \text{exists}_{\phi_*}(b), \bar{t}(e) \in B' \}$$

where $B' \subseteq B$, our definition proceeds by induction on the structure of formulas:

$$[tt]^{\phi_*} = B$$

$$[\neg \Psi]^{\phi_*} = B \setminus [\Psi]^{\phi_*}.$$ 

$$[\Psi_1 \land \Psi_2]^{\phi_*} = [\Psi_1]^{\phi_*} \cap [\Psi_2]^{\phi_*}.$$ 

$$[\langle \rangle \Psi]^{\phi_*} = [\langle \rangle]^{\phi_*} [\Psi]^{\phi_*}.$$ 

$$[\langle ? \rangle \Psi]^{\phi_*} = [\langle ? \rangle]^{\phi_*} [\Psi]^{\phi_*}.$$ 

9 Zooming in the context of reasoning

In Figure 3 of Section 5.2 we observed that the path $\langle e_2, e_3 \rangle$ in $Z(\chi)$ is not reflected by a path in $\chi$. In the context of loose higraphs this deficiency is rectified. In Figure 4 we have the original higraph (dropping edges $e_1$ and $e_4$) and its zoomed version as a loose higraph. Now consider the formula $\langle \langle \rangle tt \rangle$ we can see that this does not hold for vertex $E$ because we are no longer certain that the path $\langle e_2, e_3 \rangle$
is connected in the transition semantics for the zoomed hi-
graph. By contrast, the formula $\langle ? \rangle \langle ? \rangle \ell t$ does hold for ver-
tex $E$ because there may indeed be a path of length two
beginning at $E$.

![Diagram](image)

**Figure 4. Zooming out with loose edges**

This very small example does confirm Harel’s argument
for the utility of loose edges in carrying important semantic
information when we use zooming on higraphs. The lit-
tle logic describing connectivit in loose higraphs seems as
though it may form the basis for a more expressive logic of
loose higraphs.

10 Further work

This work is part of a project, drawing on cognitive,
computational and mathematical views of diagrams, to re-
search principles which will improve the design of diagram-
matic, domain-specific programming languages.

Typically the diagrams used in practice contain a mul-
titude of subtly interacting features. This situation necce-
sitates an analytic approach to identify suitable primitive
structures and ways of combining them. Higraphs, feature-
ing only an underlying graph structure and depth (hierar-
chy), appear as being an excellent point of departure in the
study of diagrammatic features and their interaction.

Specifically with respect to higraphs and their applica-
tions, we aim to consider additional features (such as la-
belling and Harel’s “orthogonality”) towards obtaining a
structured account of simple Statecharts.

This paper presents loose higraphs in their simplest form
and considers only very simple semantics. Our next steps
will involve considering extensions to the simple situation
described here to take account of edge labels, and other
structuring primitives in Statecharts. The goal is to pro-
vide sound diagrammatic tool support for Statecharts and
expressive logics for reasoning in the context of Statecharts
with zooming.

References


[4] David Harel and Amnon Naamad. The STATEMATE
semantics of Statecharts. *ACM Transactions on Soft-

[5] Robin Milner. *Communication and Concurrency*. In-
ternational Series in Computer Science. Prentice Hall,
1989.

Perspectives on End-User Computing*. MIT Press,
1993.

foundation for graph-based diagrams in computing. In
*Proceedings of the 17th Conference on the Mathematical

foundation for higraphs. In *Proceedings of the 15th An-
ual Conference of the European Association for Com-

Logic In Computer Science*, volume 2, pages 478–563.