Visualizing a Logic of Dependability Arguments

C Gurr
School of Systems Engineering
The University of Reading
Reading, UK
Email: C.A.Gurr@reading.ac.uk

Abstract

This paper offers general guidelines for the development of *effective* visual languages. That is, languages for constructing diagrams that can be easily and readily interpreted and manipulated by the human reader. We use these guidelines first to examine classical AND/OR trees as a representation of logical proofs, and second to design and evaluate a visual language for representing proofs in *LofA*: a Logic of Dependability Arguments, for which we provide a brief motivation and overview.

1 Introduction

Throughout the histories of both mathematics and engineering, diagrams have been used to model and reason about systems, whether physical or conceptual – such as logic. Both historical and more recent work in this area has given rise to a plethora of diagrammatic languages, often based upon a simple core language, such as "graphs" consisting of nodes with edges linking them. Graphs have the advantage of a very simple syntax and thus are easy to read (modulo good layout), yet are rather inexpressive and so, like many diagrammatic languages, are typically extended and embellished significantly. Such extensions often risk swamping the simplicity of the underlying graphs with overloaded symbolism and a confusion of textual annotations (for example, Fig. 1 versus Fig. 2). Ideally we would wish to know how, and how far, we might extend such attractively simple languages without losing their "salience". That is, their ability to be easily and correctly interpreted and manipulated by the human reader.

Assessing the salience of diagrams requires a theory of diagrammatic languages that explains how meaning can be attached to the components of a language both naturally (by exploiting intrinsic graphical properties) and intuitively (taking consideration of human cognition). The outlines of such a theory, constructed by analogy to theories of natural languages as studied in computational linguistics, were first introduced in [3, 4] and later drawn upon to offer guidelines for designing maximally salient diagrams and diagrammatic languages in [2]. This approach, dubbed "Computational Diagrammatics"[1], separates and clarifies issues of diagram morphology, syntax, semantics, pragmatics etc, facilitating the design of diagrammatic languages that maximise expressiveness without sacrificing readability.

[1]The author is indebted to Professor Michael A. Gilbert for first proposing this term.

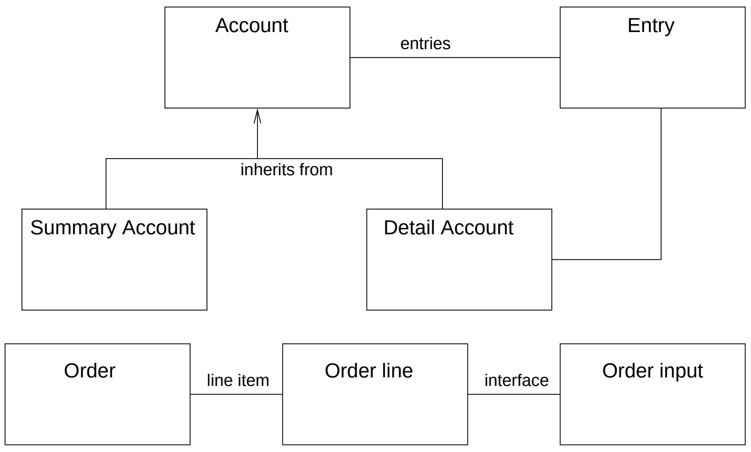

Figure 1: Simple UML Class diagram

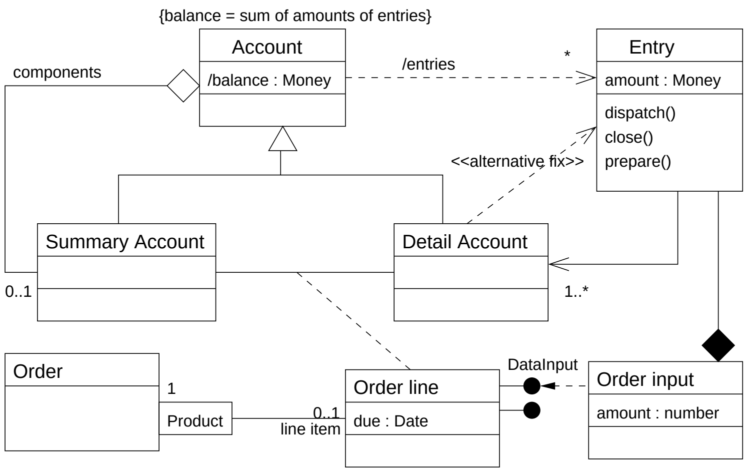

Figure 2: Enhanced UML Class diagram

There are typically many means by which a diagram, or diagrammatic language, may capture structure in a diagram. Naturally, when designing a new diagram or new diagrammatic language we would wish to determine which of the options available would be the most effective means of matching structure to content. That is, ensuring that the inherent structure in a diagram closely matches that of its semantic interpretation in a manner which is readily accessible to the reader.

This paper presents an overview of the visual language design guidelines proposed in [2] and an original analysis of the classical representation of logical AND/OR trees in the light of these guidelines, highlighting two significant issues. In Section 3 we present a specialised logic of argumentation first introduced in [1]. In section 4 we illustrate the application of our guidelines by developing an original visual representation language for this logic and discuss broader implications for for developing effective visualisations of more general logical arguments/proofs.

2 Effectiveness of Visual Representations

We may describe the syntactic components of a visual language as being comprised of basic graphical primitives and compound shapes. Graphical properties may be applied to the primitives. A part of what gives visual languages their power is the ability of: (i) categorizations of certain primitives, and (ii) the graphical properties; to carry semantic information.

Basic graphical primitives are shapes and lines. Compound graphical objects can be constructed from these primitives, two notable compound shapes being arrowed-lines (composed of a line and a shape – the arrow-head – at either end) and bordered shapes (composed of a shape and a line which traces the boundary of that shape). The latter type compound graphical object permits us to consider shapes with, for example, thick, textured and/or coloured borders.

2.1 Properties of Graphical Primitives

The primary properties of graphical objects, a variation of that suggested in [5], are: value (e.g. greyscale shading); orientation; texture (e.g. patterns); colour; and size. These are applied to lines and shapes as in Table 1. Examining these properties, we

	Valu	**Orie**	**Text**	**Colo**	**Size**
Line	lim	$\surd$		$\surd$	$\surd$
Shape	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Table 1: Properties of Graphical Objects

note that *value* and *texture* are not clearly distinguishable when applied to lines. Lines may be of variable weight (e.g. dotted, dashed or solid) which we deem to be *value*. However, texture may only be applied if all lines are of significant width, in which case they should be considered as shapes rather than lines. We consider *size* applied to lines primarily to indicate their width, although we note that it may also be applied to their length. Finally, orientation is a property over which some care must be taken. Certain tokens deemed of different shapes are in fact the same shape in a different orientation, for example squares and diamonds.

We make the following observations of the characteristics of these properties:

Value (greyscale shading) is discrete and ordered. For lines we consider value to equate to dotted, dashed or solid; and thus its use is somewhat limited ("lim"). For shapes we note that the combination of (any pair of) value, texture and colour is non-trivial.

Orientation is continuous and ordered. As noted above, some care must be taken with orientation and its usage should be minimal for points.

Texture is discrete and nominal (non-ordered).

Colour in theory, the colour spectrum is continuous and ordered. However, this is not intuitive in human perception. Hence, as graphic designers are aware, colour is best used as set of easily discernible values which are thus interpreted as discrete and nominal.

Size is continuous and ordered. However, in many cases – particularly for lines, where size=thickness – the most perceptually effective use of size is with some small number (typically 3-7) of discrete, easily discriminable values. Furthermore, we consider here that points have only a minimal ("min") variation in size, as too great a variation would suggest that they are shapes rather than points.

2.2 Guidelines for Visual Language Design

An effective visual representation is one in which the desired content is readily accessible to the reader and in which desired reasoning tasks are as simple and straightforward as possible. As indicated above, a significant benefit of visual languages is that elements of their syntax can intrinsically carry semantic information. Hence we may define a visual language that is *well-matched*, meaning that its syntax is defined to maximise the desired semantic information that it carries. However, there are a variety of elements of visual language design, over and above the syntax, which may convey semantic information. A summary of these, and a set of guidelines is introduced in [2] as follows:

1. **Morphology as types:** define a clear partitioning of basic diagram shapes into meaningful, and readily apparent, categories according to semantic type.

2. **Properties of graphical elements:** utilise graphical properties such as colour, size and texture to further partition diagram elements into a more refined type structure.

3. **Matching semantics to syntax:** select diagrammatic relations to represent semantic relations for which they have matching intrinsic properties.

4. **Extrinsic imposition of structure:** Impose constraints, or add enhancements or augmentations, to account for those cases where no direct matching of syntactic and semantic elements can be found. However, do this with care to ensure that salience is not unacceptably diminished.

5. **Pragmatics for diagrams:** Utilise, and allow diagram developers the freedom to utilise, any and all as yet unused diagrammatic morphology, properties and syntax to embody further structure in diagrams.

Clearly, there will often be cases where a choice of options exist for matching the semantics of a language. Furthermore, certain choices will conflict with others; as with the combination of any pair of colour, texture and value indicated earlier. To assist in resolving some of these choices, when constructing a visual representation (or, more generally, a visual language) that we wish to be effective, we must consider:

1. **The audience:** who are the intend readers/users of the diagram? In particular, are they commonly from some domain with existing visual languages and hence have prior conceptions of the meanings of certain symbols or other elements of diagrammatic syntax?
2. **The task:** what content needs to be identified in a diagram, and what reasoning tasks is the diagram intended to support?

Note however, that as certain graphical properties and syntactic relations may interfere, often a balance or trade-off is required when selecting the most appropriate syntactic match for some semantic aspect. Experience in graphic design (e.g [6, 7]) suggests a rule of thumb that *task* concerns outweigh *semantic* concerns; that is – where a trade-off is required, the preference should be whichever option supports greater salience of task-specific features. For example, in certain electronic or logical circuit diagrams it is often considered acceptable to duplicate the representation of some node if this improves the layout by reducing the complexity and/or number of connections to that node. In this case a semantic consideration – that each node is uniquely represented in the diagram – is over-ridden by the desire to simplify the task of identifying and tracking connections.

Finally we note that typically, for any non-trivial semantic domain and intended tasks, not all information may be captured directly through diagram syntax. Consequently the use of *labelling languages* for labels which may potentially contain significant semantic information is necessary for most practical diagrammatic languages. However, in an effort to increase the expressiveness, the unprincipled use of sophisticated labelling languages can perturb the directness of a diagrammatic language. Examples are legion of languages which are diagrammatic at core, but have had their expressiveness so enhanced through sophisticated labelling languages that any benefit to readers' interpretation of the "diagrammatic aspects" is negated. Hence we issue the warning: treat labels with care.

2.3 Issues in Visualising Logical Proof

To illustrate issues raised through the application of the above guidelines in the visualisation of logic, consider a typical AND/OR tree for visualising a logical proof, such as the example of Figure 3 which represents the proof tree for the proposition *A* in the propositional theory:

$A \Leftarrow B \vee C.$

$B \Leftarrow D \wedge E.$

$C \Leftarrow F \vee G.$

$D \Leftarrow True, E \Leftarrow True, F \Leftarrow True, G \Leftarrow True.$

Note that for brevity the tree has been simplified by omitting the *True* leaf nodes.

With respect to the above guidelines for effective visual language design, there are two notable issues with this classical representation. The first is that the significant semantic distinction between AND and OR nodes in the tree is represented only with the relatively minor syntactic distinction of having a horizontal bar across the lines leading

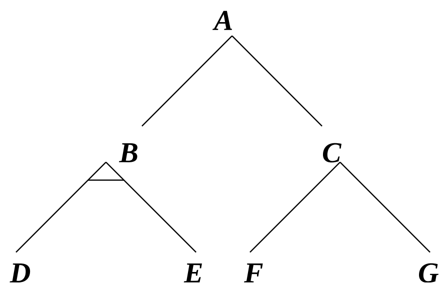

Figure 3: AND/OR Proof Tree

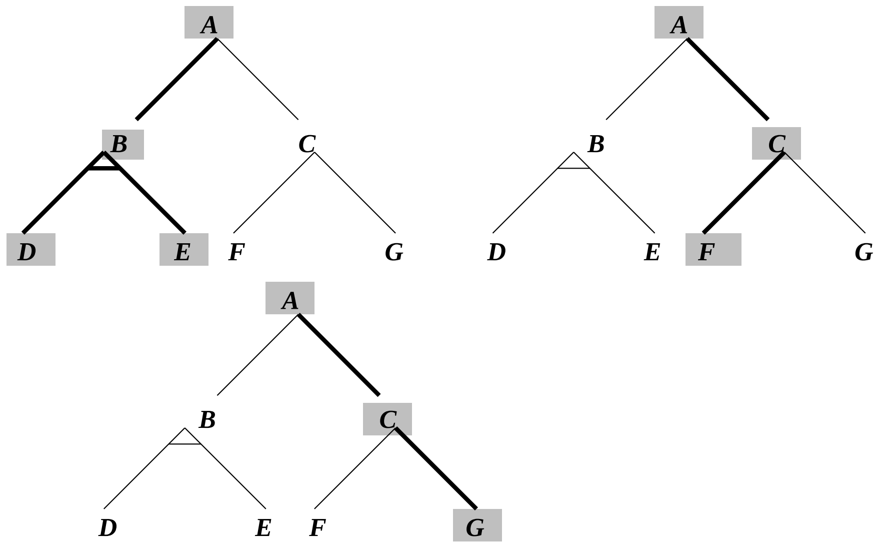

Figure 4: Disjunctive Proofs

to the children of an AND node – as with the $B \wedge E$ node in Figure 3. The second issue becomes apparent when considering the task of identifying the propositions that are active in any particular proof of the proposition A.

There are, in fact, three alternative proofs of the proposition A wrt the above theory. Visual languages are notably highly specific and generally do not lend themselves naturally to displaying such disjunctive information. Typically we have two options when displaying disjunction, either (i) we display multiple diagrams, one for each disjunct (as in Figure 4); or (ii) we introduce some new part of the notation which represents a semantic *abstraction* over the disjunctive information. As an example of this latter form, we might for example introduce a system of colour codings – where each disjunct is assigned a distinctive colour and each node and branch may be assigned multiple colours as appropriate. However, such an approach introduces further issues that make it undesirable for our purposes here.

In the example of Figure 4 we have chosen to represent disjunctive information (the alternative proofs) with multiple instances of the tree of Figure 3, each representing one of the proofs. To assist the task of identifying which propositions are active in a single proof, we have used both *value* – that is, greyscale shading – to highlight the relevant nodes; and *size* – that is, thickness of line – to highlight the relevant branches in the trees.

3 *LofA:* A Logic of (Dependability) Arguments

Many classes of highly dependable systems, both computer-based and otherwise, are required to construct an argument to satisfy some, typically independent, third party that the system achieves some minimum specification or standard measure of dependability. That is, an argument that the system can be justifiably said to meet some specified level of – for example – security, reliability, quality or safety. A system developer's motivation for providing a dependability argument may be the desire to attain some internationally recognised quality standard for a business, a process or a product. Alternatively, as is often the case with safety- and security-related systems, there may be a mandatory regulatory requirement both to attain some specified standard, and to provide a dependability argument that argues the case for the attainment of that standard.

The logic *LofA: A Logic of Arguments*, introduced in [1], was developed to assist in the representation, evaluation and negotiation of dependability arguments. At heart, *LofA* is a relatively simple propositional logic using a Horn Clause representation, which is augmented with a number of non-logical warrants (derived from ideas in the Philosophy of Argumentation [8]) and an extensible set of feature values. An argument is a proof in *LofA*. Two key non-logical warrants are: (i) Argument by Authority; and (ii) Multi-Legged Argument. The most notable feature which can be utilised in a *LofA* theory is an expression of *confidence* in the argument. Typically, one imagines the author of an argument will assign confidence values to any evidence (leaf-nodes in a proof) and the confidence of a parent node in the argument (*LofA* proof) is automatically derived from its children. Further available features include temporal and numerical values (resulting in relatively simple propositional temporal and many-valued logics respectively) and also completeness values (useful for representing partially-constructed arguments).

The full expressiveness of *LofA* was first introduced in [1]. However, for the purposes of this exposition we shall consider only the simplest variant of the language, denoted *LofA*^{0}, in which propositional, definite Horn Clauses are supplemented with the above two non-logical warrants and the single feature *confidence*. For convenience we represent a propositional atom A and its associated confidence value c (a numerical value in the range 0–1) as the unary predicate atom $A(c)$. Thus a *LofA*^{0} theory is a collection of clauses of the form: $A \Leftarrow B_1 \wedge \ldots \wedge B_n$ where A is a unary predicate atom, as above, which we refer to as the *head* of an argument, and each of $B_1, \ldots, B_n$ is a *warrant*. A warrant is any one of:

1. A unary predicate atom, such as $P(c)$
2. $Auth(s, e, c)$: representing an 'Argument by Authority', where s is an authoritative statement by (presumed) expert e, and c its confidence.
3. $Multi(L)$: representing a 'Multi-Legged Argument', where L is a list $M_1, \ldots, M_n$ of warrants, supposedly offering alternative warrants for the proposition which this multi-legged argument itself warrants (in *LofA*^{0} these sub-

warrants are simply assumed to be independent, while more sophisticated variants of *LofA* insist on a supplementary argument to justify some measurable degree of independence of the sub-warrants).

4. $Ev(P, c)$: An item of evidence, where P is the predicate name of an atom representing some piece of evidence used to warrant an argument, and c the associated confidence value of this evidence.

Clearly, leaf nodes in a *LofA*^{0} argument are either *Auth* nodes or *Ev* nodes. The latter "Evidence" form of warrant is, in fact, a simple syntactic sugar. Rather than have evidence (i.e. an item that needs no further warrant) represented as:
$P(C) \Leftarrow true$ – where C is some constant
we have replaced, in the body of each clause, each occurrence of the atom $P(x)$ with the warrant $Ev(P, C)$. That is, we have simplified the clauses by application of a single resolution step.

The confidence value of proposition A in a clause $A \Leftarrow B_1 \wedge \ldots \wedge B_n$ is $c_1 * \ldots * c_n$, where $c_1, \ldots, c_n$ are the confidence values of $B_1, \ldots, B_n$ respectively. The confidence value c_n of a multi-legged argument of legs $M_1, \ldots, M_n$ with confidence values of $l_1, \ldots, l_n$ respectively, is defined recursively as $c_i = c_{i-1} + (1 - c_{i-1}) * l_i$ and $c_0 = 0$. For example, a multi-legged argument with three legs having respective confidence values $0.5, 0.8$ and 0.6, will have derived confidence value $0.5 + (1 - 0.5) * 0.8 + (1 - (0.5 + (1 - 0.5) * 0.8)) * 0.6 = 0.5 + 0.4 + 0.06 = 0.96$.

To illustrate the above, consider the following argument that *LofA* is a valuable argumentation tool:

> *LofA* is a valuable argumentation tool, as it is: (i) a relatively simple language; (ii) sufficiently expressive for the representation of dependability arguments; and (iii) has also shown itself to be "fit for purpose". Proposition (i) we consider self-evident; proposition (ii) is asserted as being true by the author; while proposition (iii) is justified both by the fact that a range of key dependability arguments have already been represented in *LofA*, and by the fact that *LofA* has proved an effective educational tool in the teaching of dependability argumentation.

Simplifying the above propositions for brevity, and ignoring confidence values for the moment, we may encode these in *LofA*^{0} as the following three clauses:

LofA_valuable $\Leftarrow Ev($*Simple*$) \wedge$ *Sufficient* $\wedge$ *Fit_for_purpose*.
Sufficient $\Leftarrow Auth($*"I say so", 'C Gurr'*$)$.
Fit_for_purpose $\Leftarrow$
    $Multi([Ev($*Represented_key_arguments*$),\ Ev($*Effective_teaching_tool*$)]\ )$.

Suppose that we assert confidence values for the evidence and authority warrants as follows: *Simple* = 1, *"I say so"* = 0.9, *Represented_key_arguments* = 0.5, *Effective_teaching_tool* = 0.8; Then the derived confidence values of intermediary propositions will be: *Sufficient* = 0.9, *Fit_for_purpose* = 0.5 + (1-0.5)*0.8 = 0.5 + 0.4 = 0.9 and the derived confidence value for *LofA_valuable* is 1*0.9*0.9 = 0.81.

4 Visualizing *LofA* Arguments

In Section 2.3 we noted two issues with the classical representation of AND/OR proof trees: (i) that the visual distinction between AND/OR nodes was too weak; and (ii)

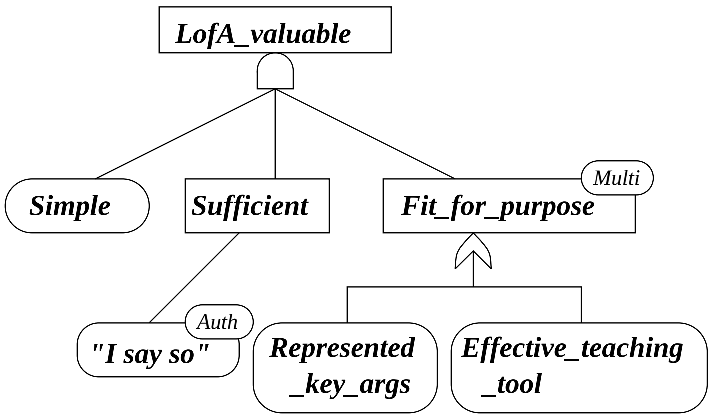

Figure 5: *Lofa* Argument Tree

that disjunctive proofs could not be represented in a single visualisation without introducing extra syntactic abstractions. Adopting the guidelines for visual language design summarised earlier, we have addressed these two issues in the visualisation of *LofA* arguments, as illustrated by Figure 5, as follows.

Firstly, to make clear the categorical distinction between differing node types, we adopt the convention of using categorically different node *shapes* to clearly distinguish the types. Hence, leaf nodes (Evidence and Authority nodes) are depicted as boxes with rounded corners, while non-leaf nodes (AND nodes and Multi nodes) are depicted by boxes with square corners. Furthermore, non-logical warrants (Authority and Multi nodes) are depicted as a compound shape, consisting of the appropriate box with a named box inset in the top-right corner. Finally, where a node has multiple children, the top of the branches leading to the children are clearly labelled with one of two shapes depending on whether they are an AND node or a Multi node – the latter being, effectively, a variant of a logical OR node. These two labelling AND- and OR-shapes are adopted from the classic representation of equivalent nodes in logical circuits. These have been chosen specifically to match the domain knowledge of the most likely readers of *LofA* arguments, who are typically familiar with this specific representation of AND and OR nodes from either logical or electrical circuits, or Fault Tree Diagrams, each of which are common in the assessment of highly dependable systems. Note also the somewhat more subtle variation in the representation of branches between AND and Multi nodes, that branches below AND nodes are drawn as (angled) straight lines with no bends, in contrast to those below Multi nodes. Thus we have used *orientation* to some effect here. In addition, for both AND and Multi nodes the branches below such nodes issue from a single point – and are subsequently split in the case of Multi nodes – rather than issuing from multiple points as is permitted in logical circuit diagrams. This is a further subtle, yet deliberate, indicator of the equally subtle point that the validity – as opposed to the confidence – of the argument represented by an AND or Multi node is equally dependent upon the validity of each of its children.

The second issue raised with AND/OR trees – that disjunctive arguments cannot be readily represented in a single diagram – is avoided in this visualisation of *LofA* arguments through the simple fact that such disjunctive arguments (where only a subset of propositions contribute to a given proof) do not arise in *LofA*. The *LofA* equivalent of

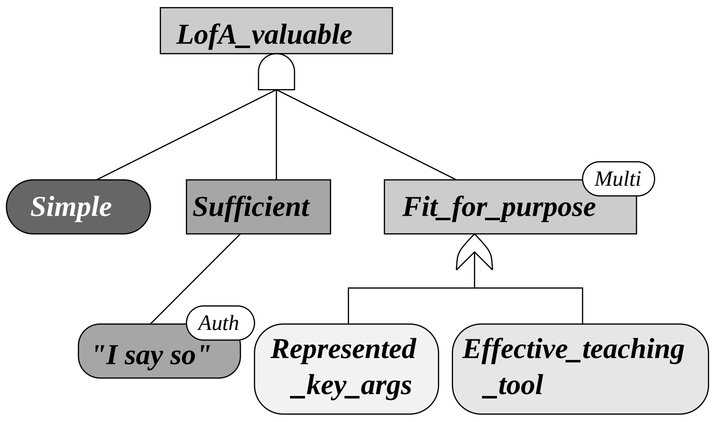

Figure 6: Visually annotated *Lofa* Argument Tree

the OR node is the Multi node, where sub-warrants each offer an alternative argument for the node, yet these are considered as a *collective* effort to increase confidence, rather than independent alternative justifications.

Having constructed this visual language for *LofA*, we may next explore how any currently unused graphical attributes – notably properties applied to graphical primitives – may be used to enhance the language wrt making certain tasks especially easy to perform. For example, consider the confidence values suggested above for our illustrative *LofA* argument. An assessor of this argument would wish to be able to readily identify both (i) what confidence is derived for the top-level argument, and (ii) how do subsidiary nodes contribute to this derived value? We may make such information readily visible in our language through application of an appropriate graphical property to the nodes. For example, using *value* (greyscale shading), where darker nodes indicate greater confidence, we may visually annotate the argument of Figure 5 to produce that of Figure 6. The choice of value to represent confidence in Figure 6 is appropriate as the primary semantic characteristics of confidence (that it is discrete and ordered) are well-matched by our choice of graphical property to represent it. Thus, for example, identifying which nodes in the argument of Figure 6 have greater or lesser confidence is an almost immediate process for the reader. Having made such a representational choice, however, constrains our further choices should we wish to extend the visual language.

Consider, for example, the feature of *completeness* with which we may extend *LofA*^{0}. In its simplest form, completeness is a binary value (complete/incomplete) applied to any leaf node in an argument. All leaf nodes in an argument are incomplete until determined to be complete by the argument's developer. A non-leaf node is complete iff all of its children are complete. This permits the argument developer to construct a *partial* argument. That is, one for which the structure of the argument is known, but for which not all evidence has been fully assembled and/or assessed. Extending the above illustrative *LofA*^{0} argument, we could assert that the evidence node *Represented_key_arguments* is currently incomplete, and hence that its parent Multi node and the top-level argument are similarly incomplete.

We have a number of means available to add such completeness information into the argument representation of Figure 6 with a well-matched visual representation.

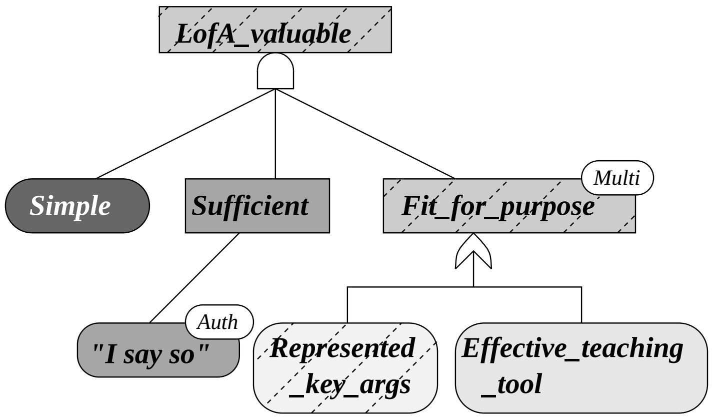

Figure 7: Further annotated *Lofa* Argument Tree

We might adopt a categorical colour coding, such as green for complete and red for incomplete. However, this choice may be problemmatic, as we have noted. Not least because colour can interfere with value, which we have already used, but there are also issues with the accessibility of colour to the human reader – both due to printing issues (this article is intended to be able to be reproduced in black and white) and concerns that not all human readers can discern all colours. Consequently we choose instead to adopt *texture* (that is, patterning) in nodes to represent incompleteness. In the argument tree of Figure 7 a value of *incomplete* for a node is indicated through a dashed patterning of the relevant node. Hence it is an easy matter for the reader to identify both which evidence nodes are incomplete, and how this impacts the (in)completeness of the remainder of the argument.

Finally, we consider briefly what further annotations could be made to argument trees such as that of Figure 7. We have applied no graphical properties to the branch-lines connecting nodes (other than limited use of orientation), so colour or size could be applied meaningfully there. Clearly, the use of colour in nodes also is a possibility – although as mentioned above this could only be used in certain situations and with the greatest of care. Making the orientation of nodes semantically meaningful would not seem wise, although we might make their size meaningful. However, care would be needed here also as employing significant variations in size of node would likely lead to issues with layout. Were the nodes to be considered compound shapes consisting of both the shaped node and its bordering line, we might apply further visual annotations to these borders – potentially both colour and size (thickness) – which would again permit us to represent further, well-matched, properties in a visually salient manner. Only last of all should we consider adding *textual* annotations to this representation; for example if we wished to depict the absolute (numerical) confidence values and not merely their relative values as currently displayed. As noted previously, adding increasingly semantically sophisticated textual labels is a very strong temptation to all visual language designers, but one which can very rapidly negate the intrinsic advantages of a *visual* representation.

5 Conclusions and Future Work

The study of dependability argumentation is both a rich and interesting area. Dependability arguments are often complex in nature, combining disparate forms and sources of evidence in detailed arguments, which must then be reviewed by, and negotiated with, a broad audience exhibiting diverse competencies and interests. The issue of *representing* dependability arguments is a pressing one, and current approaches – while often expressive – generally lack sufficient semantics or depth to provide the necessary support for the analysis and evaluation of arguments.

Adopting a formal approach to representing dependability arguments has many advantages, most notably in the precision and exactness that such an approach brings to questions of the meaning and validity of an argument. However, given the relative sophistication and detail contained in a typical dependability argument, the expressiveness required of any language for representing such arguments goes significantly beyond what typical formal logics presently offer. Ongoing work in reviewing practical dependability arguments illustrates what is required of a representation for it both to be sufficiently expressive, and to support the variety of evaluations and manipulations demanded in dependability domains. Studies of the representation of dependability arguments have much to learn from philosophical students of argumentation theory, an illustrative example being issues of validity and plausibility with regard to appeals to authority – a major aspect of many dependability cases.

Diagrams and new diagrammatic languages are frequently cited as being “natural” and “intuitive” notations, permitting “easy” and “accurate” communication of complex structures and concepts. Indeed, such claims have been readily applied to the many graph-based notations popular in dependability argumentation. However, the veracity of these claims is seldom tested and often the design of such diagrammatic languages follows no clear or obvious principles of usability, readability or effectiveness for the human user. We intend to continue to develop and evaluate our effective diagrammatic languages for representing dependability arguments. After all, it is clear that ultimate responsibility for acceptance or rejection of any dependability argument will always rest with a human agent or agency. As such, it is both in our interests and within our responsibility to ensure that such agents are presented with the most comprehensive, comprehensible and accessible representations of dependability arguments that it is within our power to provide.

References

[1] C Gurr. Argument representation for dependable computer-based systems. *Informal Logic*, 22(3):293–321, 2002.

[2] C Gurr. Computational diagrammatics: diagrams and structure. In D Besnard, C Gacek, and C.B. Jones, editors, *Structure for Dependability: Computer-Based Systems from an Interdisciplinary Perspective*, pages 143–168. Springer, 2005.

[3] C Gurr, J Lee, and K Stenning. Theories of diagrammatic reasoning: distinguishing component problems. *Mind and Machines*, 8(4):533–557, December 1998.

[4] C A Gurr. Effective diagrammatic communication: Syntactic, semantic and pragmatic issues. *Journal of Visual Languages and Computing*, 10(4):317–342, August 1999.

[5] R E Horn. *Visual Language: Global Communication for the 21st Century*. MacroVU Press, Bainbridge Island, WA, 1998.

[6] E R Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire CT, 1983.

[7] E R Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT, 1990.

[8] D Walton. *Informal Logic*. Cambridge University Press, New York, 1989.