Journal of Software and System Modeling manuscript No.
(will be inserted by the editor)

Using DAG Transformations to Verify Euler/Venn Homogeneous and Euler/Venn FOL Heterogeneous Rules of Inference

Nik Swoboda[1,2], **Gerard Allwein**[3]

[1] Departamento de Inteligencia Artificial,
Universidad Politécnica de Madrid, Spain.
e-mail: `nswoboda@fi.upm.es`

[2] ATR Media Information Science Labs
Kyoto, Japan.

[3] Naval Research Laboratory, Code 5543
4555 Overlook Avenue SW
Washington DC 20375-5337, USA
e-mail: `allwein@itd.nrl.navy.mil`

Received: date / Revised version: date

Abstract In this paper we will present a graph-transformation based method for the verification of heterogeneous first order logic (FOL) and Euler/Venn proofs. In previous work, it has been shown that a special collection of directed acyclic graphs (DAGs) can be used interchangeably with Euler/Venn diagrams in reasoning processes. Thus, proofs which include Euler/Venn diagrams can be thought of as proofs with DAGs where steps involving only Euler/Venn diagrams can be treated as particular DAG transformations. Here we will show how the characterization of these manipulations can be used to verify Euler/Venn proofs. Also, a method for verifying the use of heterogeneous Euler/Venn and FOL reasoning rules will be presented that is also based upon DAG transformations.

Key words Euler and Venn diagrams – diagrammatic reasoning – graph transformation – proof verification

1 Introduction

The basic goal of this work is to propose techniques for the mechanical verification of heterogeneous proofs involving formulas of first order logic (FOL) and Euler/Venn diagrams. These techniques involve translating Euler/Venn diagrams into a special kind of directed acyclic graph (DAG) and then working with these DAGs in the verification process. After the diagrams in a proof have been replaced with DAGs, the changes that occur to these DAGs in steps of the proof can then be used to verify the rules of inference used in those steps. It should be noted that, in this project, DAGs are playing a dual role. They are being used as data-structures in the implementation, and also as the basis of the theory that will be presented to demonstrate the validity of the graph-transformation based proof checking techniques we are proposing.

This is the last in a series of papers describing the design of a heterogeneous FOL and Euler/Venn reasoning system from both theoretical and practical points of view. This project began with the description of a technique for using DAGs to represent Euler/Venn diagrams [14]. Subsequently, theoretical issues involved in the design of the FOL and Euler/Venn logical system have been presented [15,16]. Here we build upon and extend this work by presenting the use of DAG transformations in the verification of heterogeneous FOL and Euler/Venn proofs.

2 Background

It has been shown that a special system of DAGs can be used to *capture the essential properties* of an extended version of Shin and Hammer's Venn reasoning system consisting of Euler/Venn diagrams [14]. This DAG system includes a grammar specifying syntactic requirements upon the DAGs as well as rules of inference, and a semantic interpretation for those DAGs. By saying that we can capture the essential properties of the Euler/Venn system, we mean that it is possible to translate any well-formed Euler/Venn diagram into a DAG and that there is an Euler/Venn diagram which is the translation of any DAG in the system. Furthermore, these translations preserve the semantic and inferential properties of the systems. Thus, any reasoning that can be conducted using Euler/Venn diagrams can be conducted using DAGs and vice versa.

The use of DAGs in the implementation of Euler/Venn diagram systems has a number of advantages. First, DAGs provide a discrete symbolic representation of an Euler/Venn diagram which abstracts away many of the details of a particular diagram which do not carry interpretable information. Some of these details include the exact size, shape, and placement of the diagram's curves. In fact, any two Euler/Venn diagrams having the same exact information content will have equivalent corresponding DAGs. Also, each DAG is uniquely characterized by its collection of leaf nodes. Since many calculations involving the DAGs only require the checking of the leafs, DAGs can be stored in terms of their leafs and the rest of the graph generated from these leafs only when necessary. Though theoretically the size of a DAG's leaf node set can increase exponentially with the number of curves in the diagram, we have found that in practice people do not typically draw diagrams with many curve intersections. Diagrams with many curves and curve intersections are simply difficult to interpret and use.

3 Related work

This project arose from work being done to extend Barwise and Etchemendy's *Hyperproof* [1] system to include more diagrammatic and sentential systems. In *Hyperproof*, users can write and verify proofs involving formulas of FOL and blocks world diagrams. Currently, development is under way to build a new system called *Openproof* in which multiple diagrammatic and sentential systems can be used together in the writing of proofs. One of the core diagrammatic systems that will be provided in the *Openproof* framework is a system of Euler/Venn diagrams based upon Shin and Hammer's Venn systems [13, 6]. Techniques similar to those discussed here will be used in the verification of the FOL and Euler/Venn portions of the *Openproof* system.

It should also be noted that the general goal of this project, the implementation of a heterogeneous FOL and Euler/Venn reasoning system, is not the first such attempt. A basic system called JVENN has been presented in [12]. However that implementation only allows Venn diagrams containing at most three curves (and no Euler type diagrams in which one curve can be completely contained in another). Also the system has only a very basic notion of rules similar to *Hyperproof's* Observe and Apply Rules and does not allow the construction of heterogeneous proofs.

Though one of the most common uses of Venn and Euler diagrams is in the teaching of basic set theory, more recently diagrammatic systems inspired by Venn and Euler diagrams have also been developed to assist in software design. Constraint diagrams [10, 4](and then later spider diagrams [5, 8]) were introduced as a system for visually specifying invariants in models of object-oriented software. These diagrams were intended to remedy the lack of facilities for specifying FOL formula in the Unified Modeling Language (UML) [11].[1] Using constraint diagrams, a UML diagram of an object-oriented system can be annotated with constraints representing relations between the objects of the system. One interesting difference between spider diagrams and the Euler/Venn diagrams discussed here is the addition of a richer syntax for describing constant sequences that allows the placing of both upper and lower bounds on the size of sets described in the diagrams. Though not discussed here, the techniques described in this paper could be extended to be used as the basis of the verification of proofs involving those diagrams as well.

4 Overview

In the proofs we will verify, the reasoner will be able to use homogeneous FOL, homogeneous Euler/Venn, and the heterogeneous FOL and Euler/Venn reasoning rules. The heterogeneous rules will allow the extraction and re-expression of information from Euler/Venn diagrams as formulas of FOL. They also will allow the construction of Euler/Venn diagrams based

[1] UML is a diagrammatic system commonly used in software design.

upon information contained in formulas of FOL. We generically refer to these kinds of rules of inference as Recast rules. Here, following in the tradition of Barwise and Etchemendy in their work on the *Hyperproof* system [1], we will focus on a special kind of Recast rule, known as the Observe rule. Additional background information regarding these rules can be found in Section 9.

We will begin by reviewing the systems of Euler/Venn diagrams and DAGs in Section 5 and Section 6. Using these concepts, we can then think of a proof involving Euler/Venn diagrams as a proof involving DAGs, each step consisting of a DAG and using a DAG as support can be thought of as a DAG manipulation. In Section 7 we will define the notion of a DAG delta which will be used to characterize these manipulations. Then in Section 8 we will propose algorithms for the analysis of these manipulations to determine their validity as applications of homogeneous Euler/Venn rules of inference. Finally, in Section 9 we will show how a heterogeneous rule of inference involving Euler/Venn diagrams and formulas of FOL can also be verified using DAGs.

5 A brief review of Euler/Venn diagrams

Euler and Venn diagram notions such as region, basic region, etc. will be used in a manner inspired by Hammer's definitions [6]. Here we will give a brief summary of these notions as well as quickly describe the notion of well-formed diagram used in this paper.

Let $\mathcal{L}$ be some set of predicates, each of which can be thought of as the label of some curve of an Euler/Venn diagram, and let the set Terms be the union of a set Cons of constant symbols and a set Var of variable symbols also occurring in those diagrams. For the purposes of this project, free variables and constants will be treated almost identically. Thus, free variables will be replaced by fresh constants at the point of their evaluation. We also define a collection of special variables SVar which will only be used in DAGs to denote existentially quantified variables. An Euler/Venn diagram consists of the following syntactic features: rectangles, a countably infinite set of closed curves uniquely labeled with predicate symbols from $\mathcal{L}$, shading, a countably infinite set of individual constant symbols from Terms, and lines used to connect those constant symbols into *constant sequences*.

A *region* of a diagram is any, possibly empty, area of the diagram that is completely enclosed by lines of that diagram. The collection of regions is closed under union, intersection, and complement; thus a region may contain disconnected parts. Any region of the diagram completely enclosed by a closed curve is referred to as a *basic region* and a region which is not represented in the diagram (has no area in the diagram) will be called a *missing region*. Each basic region has a unique *label*. A *minimal region* is any non-missing region which is not crossed by any of the lines of that diagram (i.e., any region that can not be thought of as the union of other non-missing regions). Two regions in two different diagrams are said to be *counterparts* if they are both interpreted as representing the same set. [2]

5.1 Well-formed Euler/Venn Diagrams

Any diagram only consisting of a rectangle is a well-formed Euler/Venn diagram. If a diagram is well-formed, that diagram with the following modifications is also well-formed:

1. The addition of any closed curve C with a label L (not already occurring in the diagram) and not extending outside the rectangle of the diagram so that all the minimal regions intersected by C are split into at most two new regions
2. The shading of any minimal region
3. The addition of an individual constant symbol that is not already in the diagram to any minimal region
4. The addition of a constant symbol that is already in the diagram to any minimal region not containing that constant symbol along with a line connecting this new constant symbol to an existing instance of that symbol

The collection of well-formed Euler/Venn diagrams will be referred to as $\mathcal{EV}$. Note that for the purposes of this paper we will be somewhat lax about the constraints placed upon the curves used in the diagram. Though we will not allow diagrams with two different minimal regions to represent the same set, we will allow curves partially or totally concurrent to one another and to the enclosing rectangle as well as points at which any number of curves can cross. This was done to ensure that every Euler/Venn diagram with no missing regions and shading could be re-drawn as an Euler/Venn diagram without shading (and with missing regions). Recently, work has been done to describe when an Euler/Venn diagram with no missing regions can be re-drawn as an Euler diagram without shading while not allowing concurrent curves nor triple cross points [3]. However, the algorithms presented there are complicated and thus for simplicity they will not be addressed here.

Before proceeding we will take a look at one example Euler/Venn diagram. From the information in Fig. 1 we can see that there are no Sloths that are both Three-toed and Two-toed due to the location of the shaded region. We can also see that Maned sloths are either Three-toed and not Two-toed or both Three-toed and Two-toed from the placement of the Maned constant symbol links.

[2] Later once we have introduced the notion of a tag, we can also give a syntactically based definition of counterparts for regions with tags saying that counterpart regions are regions with the same tag. For additional work on syntactically defining the notion of counterpart regions see [9].

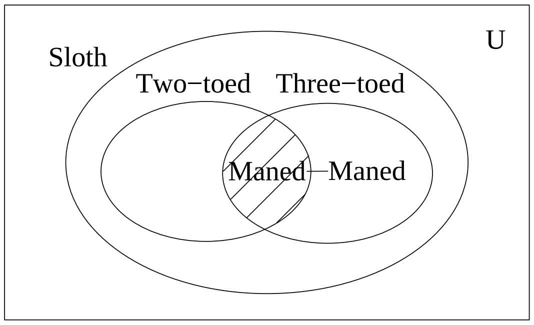


Fig. 1 An Euler/Venn diagram

5.2 Notion of a Tag

Given the set $\{L_1, \ldots, L_n\}$ of labels of a diagram $V \in \mathcal{EV}$, a *tag* is a subset of $\{L_1, \overline{L_1}, \ldots, L_n, \overline{L_n}\}$ containing at most one of L_i and $\overline{L_i}$ for each i. A tag τ is said to be *complete* if for each label L_i of V, either $L_i \in \tau$ or $\overline{L_i} \in \tau$.

Thus for each basic region labeled L there will be a tag $\{L\}$ corresponding to it and tag $\{\overline{L}\}$ corresponding to the complement of that region. Then given two regions tagged with τ_1 and τ_2 the tag for the intersection of those regions will be $\tau_1 \cup \tau_2$ (provided that this tag doesn't contain L_i and $\overline{L_i}$ for some L_i). We then see that all the complete tags consisting of labels $L_1, \ldots, L_n$ correspond exactly to all of the minimal regions of a Venn diagram (a diagram with no missing regions) having curves with each of those labels. It should be noted that in diagrams with missing regions, tags can still be used to refer to the missing regions that could be re-introduced into the diagram.

5.3 Semantics for Euler/Venn diagrams

The semantics of the system is given by the assignment of a domain to the rectangle of the diagram, subsets of this domain to basic regions of the diagram, and members of the domain to each of the diagram's individual constants. This assignment of subsets to the basic regions of the diagram can then be used to uniquely define another assignment of subsets of the domain to all the minimal regions in a diagram. Shaded and missing regions are used to denote the fact that the represented subset of the domain is empty. Constant sequences are used to express the fact that the represented individual is contained in the subset represented by the region containing all the links of that constant sequence.

5.4 Euler/Venn rules of inference

Given diagrams V and V' of $\mathcal{EV}$, V' can be inferred from V if V' is well-formed and is the result of applying any of the following rules to V:

1. Erasure of part of a constant sequence – V' is obtained by erasing a c of a constant sequence of V where that c falls within a shaded region, and provided that the possibly split c sequence is rejoined if necessary.
2. Extending a constant sequence – V' is the result of adding a new c link to a constant sequence of V in a region not already containing a link of that sequence.
3. Erasure – V' is obtained from V by erasing: an entire constant sequence, the shading of a region, or a closed curve (and possibly redrawing the remaining curves to keep the diagram well-formed) if the curve removal does not cause any counterpart regions to disagree with regard to shading (and whether they are missing) as well as the containment of links of a constant sequence.
4. Introduction of a new curve – V' is the result of adding a new curve to V which crosses all of the minimal regions of V once, and in such a way that V' is well-formed, the other labels of V are left undisturbed, and all counterparts agree with respect to shading (and whether they are missing) as well as the containment of links of a constant sequence.
5. Inconsistency – V' of any form can obtained from V if V contains a region that is both shaded and contains all the links of some constant sequence.
6. Adding shaded regions – V' is the result of adding a new (but not basic) region which is a counterpart of a missing region in V provided that this new region is shaded and is drawn so that the region is contained within the basic regions to whose intersection it is intended to correspond.
7. Removing shaded regions – V' is the result of removing a shaded (but not basic) region of V. To emphasize the fact that the region has been removed the lines enclosing the now non-existing region should be smoothed, and if possible, the remaining curves should be spaced out to remove points of unintended intersection.

Shin and Hammer also presented a Unification rule which plays an important role in the soundness of the system. Here, in the interest of space, this rule will not be discussed. Though the fact that it involves two support diagrams makes its verification a little more complicated, the verification of the Unification rule is similar to the verification techniques that will be presented for the above rules.

6 Euler/Venn diagrams and DAGs

.

The DAGs that we will be considering consist of nodes labeled with tags representing regions of a diagram along with shading and constant sequence information in the form of node properties. Directed edges connecting the

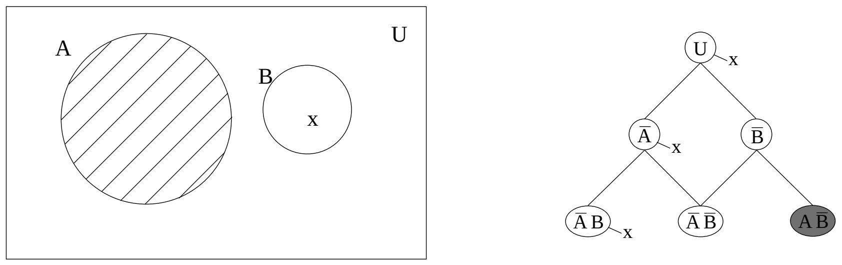


Fig. 2 A sample Euler/Venn diagram and its corresponding DAG

nodes of the DAG will be taken to represent the covers relation between regions of the diagram.[3] Each DAG will have one node representing its domain (the collection of all the objects that the diagram could explicitly represent), and all the other nodes will uniquely represent each of its corresponding diagram's non-equivalent basic regions, complements of basic regions, and non-missing regions that are the intersection of some collection of basic regions and their complements. The label of each node consists of U, for the root node, or tags constructed from some collection $L_1, \ldots, L_n$ of curve labels from $\mathcal{L}$. With the exception being the root node, each node is labeled with the most complete tag which corresponds to the region it represents. For example, the node labeled $\overline{A}B$ in Fig. 2 is taken to represent the intersection of the basic region B and the complement of the basic region A. Thus the leaf nodes of the DAG represent the minimal regions of the diagram. Nodes of the DAG are designated as shaded if the region that node represents is shaded in the diagram. Likewise, a node is designated as containing a link of a constant sequence if the region that node represents contains a link of that constant sequence. The DAG containing only one node labeled U and no edges is a member of $\mathcal{DAG}$. All other DAGs in $\mathcal{DAG}$ have the the following properties:

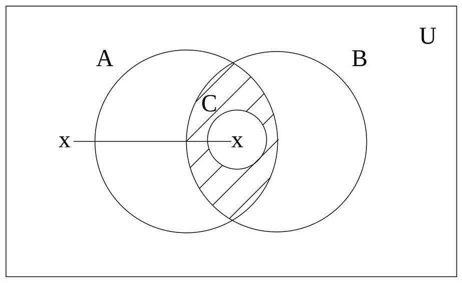
1. Every node has as its ancestor the root node U.
2. No node has a parent which is also a non-trivial ancestor of another of its parents.
3. Every leaf node is labeled with a complete tag constructed from some set of curve labels $L_1, \ldots, L_n$.
4. The tag t of any node is unique, contains all of curve labels of each of its parents tags, and provided that there are no inner nodes with only one child, has ancestor nodes labeled with every tag t' containing only curve labels in t.
5. A node is shaded iff all its descendants are shaded and contains a constant c (from Terms or SVar) iff it is a leaf node or one of its descendants which is a leaf node contains that constant.

[3] "*A covers B*" iff $B \subsetneq A$ and there is no region C such that $B \subsetneq C$ and $C \subsetneq A$.

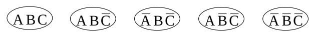
A sample Euler/Venn diagram and its corresponding DAG can be found in Fig. 2. In this diagram, the region corresponding to the intersection of the basic regions A and B is missing, and this can be interpreted to mean that there are no members of the set denoted by A that are also members of the set denoted by B. The shading of the basic region A denotes that the corresponding set is empty. The location of the constant x carries the information that the object that it denotes does belong to the set corresponding to basic region B, and does not belong to the set corresponding to the basic region A. In the diagram's DAG, note the shading of the node $A\overline{B}$, the lack of a node labeled AB, and the locations of the x constant symbols.

An example of non-inductively constructing a DAG from an Euler/Venn diagram is given in Fig. 3. First starting with the diagram in Fig. 3(a), one leaf node with the appropriate tag is made for each minimal region of the diagram resulting in the nodes shown in Fig. 3(b). Then working up the diagram, inner nodes and edges are added. First for each leaf node whose tag contains n basic curve labels, parent nodes are added for each of the n choose $n-1$ (e.g,. $\binom{n}{n-1}$), tags of length $n-1$ which can be constructed from that tag's labels. For example, for the leaf node $A\overline{B}C$ we add parent nodes $A\overline{B}$, AC, and $\overline{B}C$. If any of these new nodes has already been made, an edge is added to the already existing node. After doing this for the leaf nodes, we proceed up the DAG row by row until we construct a row of nodes with single label tags. Then we add a root node labeled U as the parent of each of these single label tag nodes. The result of this process if shown in Fig. 3(c). Then from the bottom up, we remove any inner nodes and associated edges having one or no children, and reconnecting orphaned nodes to their covering parents. Lastly we shade and add constants to the leaf nodes, and then working up the DAG shade any node with only shaded descendants and add a constant to a node which has any node as a descendant with that constant. The resulting diagram is shown in Fig. 3(d).

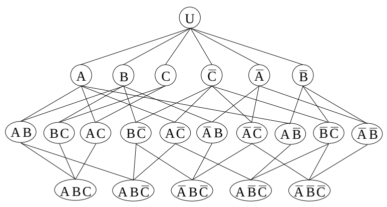
All of the Euler/Venn rules of inference described in Section 5.4 can be re-defined in terms of DAGS. Here we will just briefly summarize a few of the more interesting of these rules. To add or remove a shaded region from a DAG the shaded leaf node is either added or removed from the DAG and the inner nodes are adjusted to preserve the above properties. Removing an entire curve from a DAG involves removing that curve label and its complement from all the node tags, merging identically labeled nodes (empty labels with the node labeled U), and then again adjusting the inner nodes to preserve the above properties. Adding a curve to the diagram is done by adding two new nodes (one for the curve label and another for its complement) as children of the root node and then creating new nodes with tag labels corresponding to the union of these two curve labels with all the existing node labels in the diagram along with the appropriate edges while preserving the above properties.



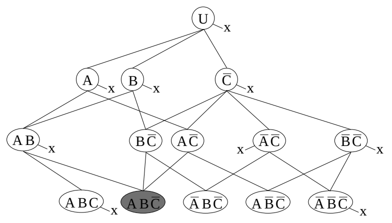
(a) Euler/Venn diagram



(b) Start with the leaf nodes



(c) Add all possible inner nodes



(d) Remove redundant nodes, then shade and add constant sequences

Fig. 3 A example of non-inductive DAG construction

7 Characterizing DAG manipulations

In the easiest case, the rules of inference that we would like to verify consist of a supporting Euler/Venn diagram and a resulting Euler/Venn diagram. To check the application of these rules, one can translate these diagrams into DAGs and study the differences that exist between them. Thus it is important for us to be able to characterize and to analyze these DAG transformations. In order to do this more easily we will introduce the notion of a *DAG delta*. Each delta will denote some meaningful transformation to a DAG. A collection of DAG deltas will be used to describe the changes made to transform one DAG into another. Once we have the collection of DAG deltas that exist between two DAGs, we can then see if the specified rule of inference allows each of those deltas. The following is the list of deltas that we will use.

Definition 1 *Given any curve label l, constant symbol c, and complete tag t all relating to some DAG $D \in \mathcal{DAG}$, the following are all valid DAG deltas:*

– `ADD CURVE` $: l$ *and* `REMOVE CURVE` $: l$

– *ADD REGION* : t *and REMOVE REGION* : t
– *ADD SHADE* : t *and REMOVE SHADE* : t
– *ADD CONSTANT LINK* : $c; t$ *and REMOVE CONSTANT LINK* : $c; t$

A three step algorithm is used to compute the collection of DAG deltas, DELTAS, between the support DAG and the resulting DAG. First we determine which curves need to be added or removed from the supporting DAG to result in a DAG representing the same collection of basic regions as the resulting DAG. For each such curve we add the appropriate ADD CURVE or REMOVE CURVE delta to DELTAS. In the case in which a curve is added we can use the Introduction of a new curve rule of inference to add the curve to the support DAG. In the case of removing a curve, we use the Erasure rule of inference to remove that curve from the support DAG. (Note that the changed support DAG is then used in the later parts of the algorithm.) Then, since the entire DAG can be constructed from its leafs, we can consider only the differences between the leafs of the resulting two DAGs. So we proceed by checking each leaf node one by one and adding any of the appropriate deltas to DELTAS as would be expected. A more detailed description of this algorithm will now be presented.

Algorithm 1 Given $D, D' \in \mathcal{DAG}$, we use the following to compute the collection of DAG deltas, DELTAS, describing the differences between D and D'. We start with DELTAS containing no members.

1. Compute curve deltas:
   (a) For each curve l in D' not in D we add ADD CURVE : l to DELTAS and we use the Introduction of a new curve rule of inference (using the inductive DAG construction algorithm) to add the curve l to D.
   (b) For each curve l in D not in D' we add REMOVE CURVE : l to DELTAS and we use the Erasure rule of inference to remove l from D.
2. Compute region deltas:
   (a) For each n' a leaf node of D' with tag t such that there is no node n in D with the tag t we add ADD REGION : t to DELTAS. If the added node in D' is shaded we add ADD SHADE : t to DELTAS and if it contains links of constant sequences we add ADD CONSTANT LINK : $c; t$ deltas to DELTAS for each constant link c.
   (b) For each n a leaf node of D with tag t such that there is no node n' in D' with the tag t we add REMOVE REGION : t to DELTAS. If the removed node in D is shaded we add REMOVE SHADE : t to DELTAS and if it contains links of constant sequences we add REMOVE CONSTANT LINK : $c; t$ deltas to DELTAS for each constant link c.
3. Compute all other deltas:
   For each n a leaf node of D and n' a leaf node of D' such that n and n' have the same tag t:
   (a) if n' is shaded and n is not then add ADD SHADE : t to DELTAS
   (b) if n is shaded and n' is not then add REMOVE SHADE : t to DELTAS
   (c) if n' has a constant c that is not in n we add ADD CONSTANT LINK : $c; t$ to DELTAS.

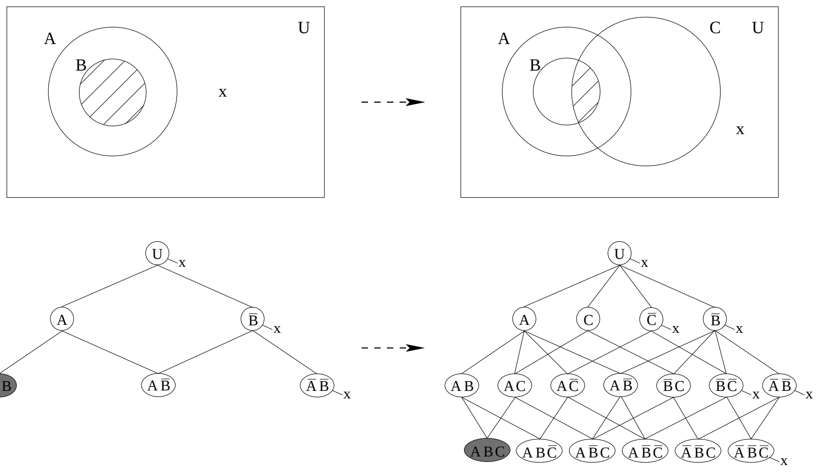


Fig. 4 An example of computing DAG deltas

(d) if n has a constant c that is not in n' we add `REMOVE CONSTANT LINK` : $c; t$ to DELTAS.

For example, when computing the deltas between the diagrams of Fig. 4, in Step 1 we need to add a curve to the support diagram resulting in the situation in Fig. 5. While doing this we also add `ADD CURVE`:C to DELTAS. Then in Step 2, we note that there are no region deltas, i.e., that both DAGs have the same leaf nodes. Lastly in Step 3, we compute the add and remove shading and constant link deltas. Thus the collection DELTAS for Fig. 4 would consist of: `ADD CURVE`:C, `REMOVE SHADE`:AB$\overline{\text{C}}$, `REMOVE CONSTANT LINK`:x;$\overline{\text{A}}\,\overline{\text{B}}$C. Note that in Fig. 4 there is no node labeled B since the basic region labeled B is the same as the minimal region AB which is represented in the DAG.

Proposition 1 *All logically significant differences between any two DAGs in $\mathcal{DAG}$ can be captured with a set of DAG deltas.*

Proof sketches for this and the following results will be given in the appendix included at the end of the paper.

Proposition 2 *Algorithm 1 for computing DAG deltas captures all logically significant differences between the two DAGs.*

8 Verifying the Euler/Venn homogeneous rules of inference using DAGs

Using these DAG deltas, the checking of the homogeneous Euler/Venn rules from the collection DELTAS can be done directly. For example, to check the Erasure of part of a constant sequence rule, we verify that the collection DELTAS between the support and the result only consists of a single `REMOVE`

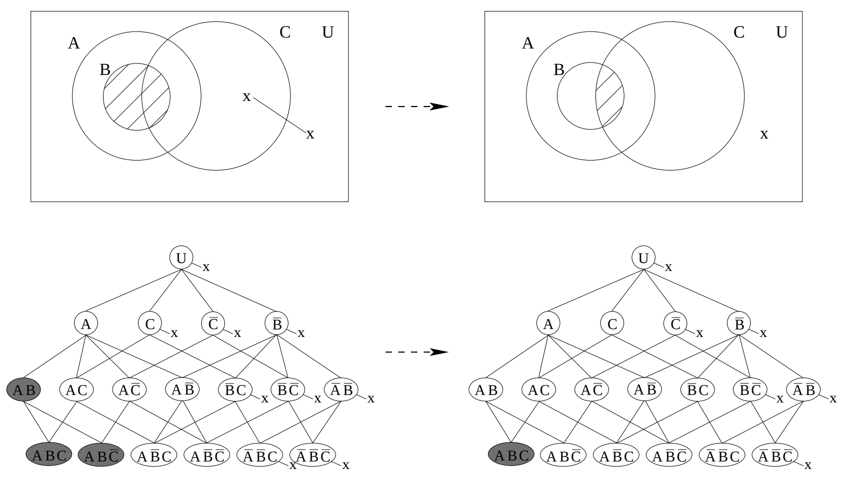


Fig. 5 The result of adding curve C to the left diagram and corresponding DAG of Fig. 4

`CONSTANT LINK` delta whose node is shaded in the supporting DAG. More detailed algorithms for the Euler/Venn homogeneous rules of inference presented in Section 5.4 will now be given.

Algorithm 2 Given $D, D' \in \mathcal{DAG}$, and the set DELTAS resulting from the application of Algorithm 1 to D and D', we do the following to check the application of one of the homogeneous rules of inference presented in Section 5.4

- Erasure of part of a constant sequence – We check to see that DELTAS only contains one delta of the form `REMOVE CONSTANT LINK` : $c; t$ and that the node tagged by t is shaded in D.
- Extending a constant sequence – We check to see that DELTAS only contains one delta of the form `ADD CONSTANT LINK` : $c; t$, that the node corresponding to t doesn't already contain the link c, and that some other node n' in D contains the link c (i.e., that a new constant sequence has not been introduced).
- Erasure – We check to see that DELTAS only contains:
  - `REMOVE SHADE` or `REMOVE CURVE` deltas
  - `REMOVE CONSTANT LINK` deltas where for each removed constant link c there is a `REMOVE CONSTANT LINK` : $c; t$ delta for every leaf node with tag t containing c (i.e., that the entire constant sequence is being removed).
- Introduction of a new curve – We check to see that DELTAS only contains one `ADD CURVE` delta.
- Adding shaded regions – We check to see that DELTAS contains only one delta of the form `ADD REGION` : t and that the node labeled t in D' is shaded.

– Removing shaded regions – We check to see that DELTAS only contains one delta of the form `REMOVE REGION` : t that the node labeled t in D is shaded.
– Inconsistency – We check to see that for some constant sequence c in D that every leaf node in D containing c is shaded.

Proposition 3 *Algorithm 2 correctly checks the use of all the Euler/Venn homogeneous rules of inference presented in Section 5.4.*

9 Verifying heterogeneous rules of inference using DAGs

The heterogeneous rule of inference that we will check is the Observe rule. Though inspired by the work of Barwise and Etchemendy, its use in this context will be slightly different. Here the rule's use will be symmetrical, it can be used to transfer information from Euler/Venn diagrams to FOL and from FOL to Euler/Venn diagrams. Also, in our case the rule will only be used to extract explicit information. This notion of being explicit is based upon Dretske's notion of *secondary seeing that* [2]. Dretske attempted to define a notion of epistemic seeing, or "seeing that", which has a fundamental relation to a notion of non-epistemic seeing. He took very seriously the idea that any genuine instance of "seeing that" should have as its basis a visual event. This was done to preserve, as he says, the "visual impact" of "seeing that" and thus to exclude such natural language uses of the phrase "to see that", as "Mary could see that the eggs were completely hard boiled (from the ringing of the timer)", from his notions of seeing. Part of the motivation for doing this was to ensure that the "How?" justification of any information gathered through an act of "seeing that" involves an act of non-epistemic seeing in an essential way. For our purposes, this locution is important because it will help us to characterize information which can be observed, acquired through fundamentally visual means, from a diagram. These kinds of rules are further discussed in [16], and will be minimally presented here. An example of a valid use of the Observe rule with Fig. 2 as support would be the formula $\neg\exists x\ (A(x) \wedge B(x))$.

9.1 Observing formulas of FOL from Euler/Venn diagrams

Before we define a notion of observation from Euler/Venn diagrams to formulas of FOL we will first introduce the notion of an Euler/Venn observational formula(EVOF). These formulas have been specially selected so that each formula of EVOF carries information which corresponds directly to a syntactic feature of an Euler/Venn diagram. (Please note that where $\varphi(t)$ is written in the following definitions it will mean that *all* of the predicates in the formula φ contain the term t.)

Definition 2 *Euler/Venn Observational Formulas (**EVOF**)*

1. *Basic formulas: For every predicate P in $\mathcal{L}$, and term t in Term, $P(t)$ is in **EVOF**.*
2. *Negations, Conjunctions, and Disjunctions: For every $\varphi_1(t), \ldots, \varphi_n(t)$ in **EVOF**, the following are also in **EVOF**:*
   – $\neg\varphi_1(t)$
   – $(\varphi_1(t) \wedge \ldots \wedge \varphi_n(t))$
   – $(\varphi_1(t) \vee \ldots \vee \varphi_n(t))$
3. *Quantifiers: For every unquantified $\varphi(x)$ in **EVOF**, the following are also in **EVOF**:*
   – $\mathsf{N}x\ \varphi(x)$ *(read as "There is no x such that $\varphi(x)$.")*
   – $\exists x\ \varphi(x)$

Definition 3 *Given a diagram $V \in \mathcal{EV}$ containing curves labeled $P_1, \ldots, P_n$, the partial region assignment function $region_V$ from **EVOF** to the regions of V will be defined as follows:*

1. *For each basic region r labeled P in V $region_V(P(t)) = r$.*
2. *If $region_V(\varphi_1(t)) = r_1, \ \ldots, \ region_V(\varphi_n(t)) = r_n$ then:*
   – $region_V(\neg\varphi_1(t)) = \overline{r_1}$.[4]
   – $region_V((\varphi_1(t) \wedge \ldots \wedge \varphi_n(t))) = r_1 \cap \ldots \cap r_n$
   – $region_V((\varphi_1(t) \vee \ldots \vee \varphi_n(t))) = r_1 \cup \ldots \cup r_n$
   – $region_V(\mathsf{N}x\ \varphi(x)) = region_V(\exists x\ \varphi(x)) = region_V(\varphi(x))$

Definition 4 *The relations of strong observation, $V |\!\!\approx^{+} \varphi(t)$ and $V |\!\!\approx^{-} \varphi(t)$, will be defined between diagrams of $\mathcal{EV}$ and formulas of **EVOF** by induction on the complexity of $\varphi(t)$ as follows:*

– *For unquantified formulas:*
  – $\boldsymbol{V |\!\!\approx^{+} \varphi(t)}$ *if the term symbol t appears in $region_V(\varphi(t))$.*[5]
  – $\boldsymbol{V |\!\!\approx^{-} \varphi(t)}$ *if the term symbol t appears in the complement of $region_V(\varphi(t))$.*
– *For quantified formulas:*
  – $\boldsymbol{V |\!\!\approx^{+} \mathsf{N}x\ \psi(x)}$ *if the region $region_V(\psi(x))$ is shaded or missing.*
  – $\boldsymbol{V |\!\!\approx^{-} \mathsf{N}x\ \psi(x)}$ *if $V |\!\!\approx^{+} \exists x\ \psi(x)$.*
  – $\boldsymbol{V |\!\!\approx^{+} \exists x\ \psi(x)}$ *if some term symbol t appears in $region_V(\psi(x))$.*
  – $\boldsymbol{V |\!\!\approx^{-} \exists x\ \psi(x)}$ *if $V |\!\!\approx^{+} \mathsf{N}x\ \psi(x)$.*

$V |\!\!\approx^{?} \varphi$ will be written if neither $V |\!\!\approx^{+} \varphi$ nor $V |\!\!\approx^{-} \varphi$.

To define observation for all formulas in monadic first order logic (MFOL), we introduce a special normal form called EVCNF. In this form each of the formula's conjuncts carries information regarding some feature of an Euler/Venn diagram. Sentences of EVCNF have only unary predicates, atomic negation, minimized quantifier scope, and are in the form of conjuncts of

[4] Here $\overline{r}$ will be taken as denoting the region bound by the diagram's rectangle with r removed.

[5] The term symbol t appears in the region r if t is not part of a constant sequence and t appears in r or if t is part of a constant sequence and the entire sequence appears in r.

disjuncts or quantified expressions (using $\exists$ and N) where every predicate in the scope of the quantifier contains the quantified variable. In this form each conjunct is either a quantified formula of EVOF, a disjunction in EVOF or a mixed term disjunction.

Definition 5 *Given a Euler/Venn diagram $V \in \mathcal{EV}$ and a formula φ in MFOL, we will define the relations of observation as follows:*

- $V|\!\!\approx^{+}\varphi$ *if when φ is transformed into EVCNF every conjunct is in EVOF, and for each conjunct ψ it is the case that* $V|\!\!\simeq^{+}\psi$
- $V|\!\!\approx^{-}\varphi$ *if when φ is transformed into EVCNF every conjunct is in EVOF, and there is some conjunct ψ such that $V|\!\!\simeq^{-}\psi$, and for the rest either $V|\!\!\simeq^{-}\psi$ or $V|\!\!\simeq^{+}\psi$*

$V|\!\!\approx^{?}\varphi$ *will be written if neither $V|\!\!\approx^{+}\varphi$ nor $V|\!\!\approx^{-}\varphi$.*

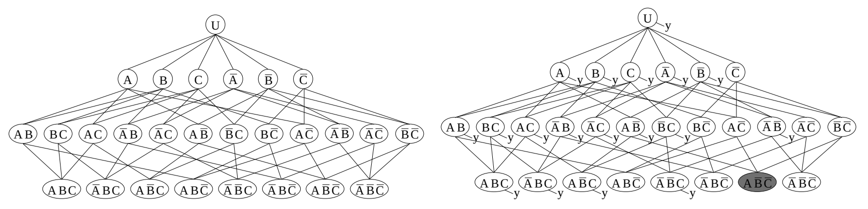
9.2 Using DAGs to verify this rule

We begin with a support Euler/Venn diagram and a resulting formula of FOL. Due to the nature of Euler/Venn diagrams, we begin by assuming that the formula only contains monadic predicates. First we convert the formula of FOL into EVCNF and then using this formula we construct a DAG containing all the information in that formula that could be expressed in an Euler/Venn diagram. Here we will focus on just the positive part of the observe relation, $V|\!\!\approx^{+}\varphi$, as the negative part can be addressed analogously.

Algorithm 3 Given a formula φ of MFOL, we use the following algorithm to extract all pertinent Euler/Venn information from the formula and to produce a DAG. We begin with D as the empty DAG.

1. For each predicate P occurring in φ we add to D, using the Introduction of a new curve rule, a basic curve with the label P.
2. Convert φ into φ' in EVCNF and group non-quantified conjuncts sharing the same term. If any of the conjuncts of φ' is not in EVOF then the construction fails.
3. For each group of non-quantified conjuncts all with the term t, $\psi_1(t), \ldots \psi_n(t)$, in φ' we make a new constant sequence in D placing the term t in every leaf node of D that represents a subregion of $region_V(\psi_1(t) \wedge \ldots \wedge \psi_n(t))$. [6] If $region_V(\psi_1(t) \wedge \ldots \wedge \psi_n(t))$ is empty the construction fails.
4. For all other conjuncts ψ in φ', if ψ is of the form:
   - $\theta_1(t) \vee \ldots \theta_n(t)$ - We make a new constant sequence in D placing the term t in every leaf node of D that represents a subregion of $region_V(\theta_1(t) \vee \ldots \theta_n(t))$.

[6] Here and later when using $region_V$ without the explicit mention of which diagram V is intended, we assume that V will be a Venn diagram with the appropriate number of curves.



(a) A DAG representing a Venn diagram with three curves

(b) Shading and constant sequences added

Fig. 6 An example of constructing a DAG from a sentence of FOL

  – $\mathsf{N}x\ \theta(x)$ - We shade every leaf node of D that represents a subregion of $region_V(\theta(x))$.
  – $\exists x\ \theta(x)$ - We make a new constant sequence in D placing the fresh special term x_k (from SVar) in every leaf node of D that represents a subregion of $region_V(\theta(x))$.
5. Lastly, working from the leaf nodes row by row up the DAG we shade any node with only shaded descendants and add a constant to a node which has any node as a descendant with that constant.

So now to verify the observation of a formula of MFOL from an Euler/Venn diagram we first use Algorithm 3 to generate a DAG from that formula. As before we then need to calculate the deltas between the support and the result. Then the collection of deltas is examined to insure that no inappropriate changes have occurred. The swapping of shaded regions for missing regions and constant sequences for special term sequences, as well as the weakening of information is permitted. However, any changes which introduce new information are not permitted.

Before proceeding, we will present a simple example of the use of Algorithm 3 to generate a DAG from the sentence $\forall x(A(x) \to (B(x) \lor C(x)) \land C(y)$. In Steps 1 and 2, we begin by constructing the DAG in Fig. 6(a), and converting the sentence into EVCNF, $\mathsf{N}x\ (A(x) \land \neg B(x) \land \neg C(x)) \land C(y)$ Then in step 3, we then place a y constant into every leaf node which represents a subregion of the basic region C and shade the leaf node with the tag $A\overline{B}\,\overline{C}$. Step 4 does not require any further changes to the DAG.

Algorithm 4 Given a DAG D in $\mathcal{DAG}$ and a formula φ of MFOL we use the following procedure to verify the observation of φ from D:

1. Using φ and Algorithm 3 construct the DAG D'. If the construction fails then the rule fails.
2. Use Algorithm 1 to compute the set DELTAS between D and D'.
3. We check that for each delta $d \in$ DELTAS that d is:
  – a `REMOVE CURVE`, `ADD REGION` or `REMOVE SHADE` delta.

- a `REMOVE CONSTANT LINK` delta and that for the removed link c there is a `REMOVE CONSTANT LINK` delta for every link of that sequence and also for some special term x_k a `ADD CONSTANT LINK` : $x_k; t$ for each link in the same leaf nodes. More simply put, that there is a sequence of deltas which demonstrate the substitution of a possibly longer special term sequence for some constant sequence.
- a `REMOVE CONSTANT LINK` delta and that for the removed link c there is a `REMOVE CONSTANT LINK` : $c; t$ delta for every leaf node in D with tag t containing c (i.e., that the entire constant sequence is being removed).
- `ADD CONSTANT LINK` delta and that there is a leaf node labeled t containing a c for which there are no `ADD CONSTANT LINK` : $c; t$ deltas (i.e., that an entire new constant sequence isn't being added).
- a `ADD SHADE` delta and d's node was missing in D.
- otherwise the use of the Observe rule fails. (Note that it is not possible for a `REMOVE REGION` delta to occur, since Algorithm 3 does not produce DAGs with nodes representing missing regions.)

Proposition 4 *The verification technique given in Algorithm 4, correctly verifies the* Observe *rule presented in Definition 5.*

9.3 Observing from formulas of FOL Euler/Venn diagrams

Due to space constraints, we are unable to give a detailed description of the verification of the observation of formulas of FOL from Euler/Venn diagrams. However, after providing the appropriate definitions regarding when a diagram can be observed from a formula, an analogous procedure can be used to the one given in the last section (switching the roles of the diagram and the formula).

10 Conclusion and further work

The main goal of this paper was to present methods for the verification of heterogeneous Euler/Venn and FOL proofs. To accomplish this, we have considered the steps of a heterogeneous Euler/Venn and FOL proof as containing DAGs. From this point of view, a proof can then be thought of as a sequence of DAG transformations. We have shown how the characterization of these transformations can be used in the verification of the proofs. Furthermore, these verification methods were shown to be correct with respect to a certain mathematical system of Euler/Venn and FOL reasoning. Though the details of these techniques are specific to Euler/Venn diagrams the general technique of thinking of diagrammatic proofs as graph transformations could be applied to other diagrammatic systems, including constraint and spider diagrams discussed in Section 3. Other natural applications could include systems with different kinds of hierarchical diagrams such as family trees and company hierarchy diagrams.

Acknowledgements The research reported here was supported in part by a contract with the Telecommunications Advancement Organization of Japan entitled, "A study of innovational interaction media for the development of a highly functional networked society" and a Ramón and Cajal research grant provided by the Spanish Ministry of Science and Technology.

References

1. Barwise, J. and J. Etchemendy, "Hyperproof," CSLI Publications, Stanford, 1994.
2. Dretske, F. I., "Seeing and Knowing," Chicago University Press, Chicago, 1969.
3. Flower, J. and J. Howse, *Generating euler diagrams*, in: Hegarty et al. [7] pp. 61–75.
4. Gil, J. Y., J. Howse and S. Kent, *Constraint diagrams: a step beyond uml*, in: *Proceedings TOOLS USA* (1999), pp. 453–463.
5. Gil, J. Y., J. Howse and S. Kent, *Formalising spider diagrams*, in: *Proceedings IEEE Symposium on Visual Languages (VL99)* (1999), pp. 130–137.
6. Hammer, E. M., "Logic and Visual Information," CSLI and FOLLI, Stanford, 1995.
7. Hegarty, M., B. Meyer and N. H. Narayanan, editors, "Diagrammatic Representation and Inference," Number 2317 in Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, 2002.
8. Howse, J., F. Molina, J. Taylor, S. Kent and J. Y. Gil, *Spider diagrams: A diagrammatic reasoning system*, Journal of Visual Languages and Computing **12** (2001), pp. 299–324.
9. Howse, J., G. Stapleton, J. Flower and J. Taylor, *Corresponding regions in euler diagrams*, in: Hegarty et al. [7] pp. 76–90.
10. Kent, S., *Constraint diagrams: visualizing invariants in object-oriented models*, in: *Proceedings of the 1997 ACM SIGPLAN conference on Object-oriented programming systems, languages and applications* (1997), pp. 327–341.
11. Rumbaugh, J., I. Jacobson and G. Booch, "Unified Modeling Language Reference Manual," Addison Wesley Professional, 1999.
12. Sawamura, H. and K. Kiyozuka, *Jvenn: A visual reasoning system with diagrams and sentences*, in: M. Anderson, P. Cheng and V. Haarslev, editors, *Theory and Application of Diagrams*, number 1889 in Lecture Notes in Artificial Intelligence (2000), pp. 271–285.
13. Shin, S.-J., "The Logical Status of Diagrams," Cambridge University Press, Cambridge, 1995.
14. Swoboda, N., *Implementing Euler/Venn reasoning systems*, in: M. Anderson, B. Meyer and P. Olivier, editors, *Diagrammatic Representation and Reasoning*, Springer-Verlag, London, 2002 pp. 371–386.
15. Swoboda, N. and G. Allwein, *A case study of the design and implementation of heterogeneous reasoning systems*, in: L. Magnani, N. J. Nersessian and C. Pizzi, editors, *Logical and Computational Aspects of Model-Based Reasoning*, Kluwer Academic, Dordrecht, 2002 pp. 3–20.
16. Swoboda, N. and G. Allwein, *Modeling heterogeneous systems*, in: Hegarty et al. [7] pp. 131–145.

Appendix: Proof sketches for the results reported in this paper

Proposition 5 *Any DAG in $\mathcal{DAG}$ can be uniquely determined by its collection of leaf nodes and leaf node properties.*

Proof sketch of Proposition 5:

In [14], it was shown that the non-inductive construction technique for building DAGs produces a unique DAG from a collection of DAG leaf nodes.

Proof sketch of Proposition 1:

We begin by observing that diagrams $V, V' \in \mathcal{EV}$ are syntactic variations of one another (both having the same information content) if and only if:

- they both contain the same collection of labeled curves
- For each shaded minimal region r with tag t in V or V' there is a region r' with the tag t in the other diagram that is also shaded
- For each missing region r with a complete tag t in V or V' there is a region r' with the tag t in the other diagram that is also missing
- For each constant link c in a minimal region r with tag t in V or V' there is a region r' containing the link c with the tag t in the other diagram

Thus informationally significant differences between two Euler/Venn diagrams consist of adding or removing curves, missing regions, shading or constant links. Since the system of DAGs captures the essential properties of the Euler/Venn system this same property holds of DAGs. The collection of DAG deltas has one delta for each of these possible changes.

Proof sketch for Proposition 2:

The algorithm begins by resolving the curve differences, so we know that all appropriate `ADD CURVE` and `REMOVE CURVE` deltas will be included in DELTAS. While adding curves from the support DAG, the algorithm uses the Introduction of a new curve rule of inference, which we know does not add nor remove any information from the DAG. To remove curves from the diagram the Erasure rule of inference is used, which soundly removes inappropriate information from the support DAG. Using the Proposition 5 we know that a DAG is uniquely determined by its leaf nodes and thus only considering all the possible DAG deltas between the leaf nodes captures all appropriate differences.

Proof sketch for Proposition 3:

We proceed rule by rule checking that the kinds of DAG deltas allowed by Algorithm 2 are justified by the rules of inference and that any valid application of a rule of inference is allowed by Algorithm 2. Since the relation between the rules of inference and the DAG deltas is very close we will show both directions at the same time. Given $D, D' \in \mathcal{DAG}$ and the set DELTAS resulting from the application of Algorithm 1 to D and D':

- Erasure of part of a constant sequence – This rule allows the erasure of a constant link of a constant sequence of D where that link falls within a shaded node. To verify this rule, we check to see that DELTAS only contains one delta of the form `REMOVE CONSTANT LINK` : $c; t$ and that the leaf node tagged by t is shaded in D.
- Extending a constant sequence – This rule allows the addition of a new c link to a constant sequence of D in a leaf node of D' that does not already contain a link of c. To verify this rule, we check to see that DELTAS only contains one delta of the form `ADD CONSTANT LINK` : $c; t$, that the node corresponding to t doesn't already contain the link c, and that some other node n' in D contains the link c (to prevent the possible introduction of a new constant sequence).
- Erasure – D' is obtained from D by erasing:
  - an entire constant sequence;
  - the shading of a node and the shading of all its descendants and ancestors
  - nodes representing a basic region and its complement (along with the redrawing of the inner nodes of D) if the removal does not cause any nodes representing counterpart regions to disagree with regard to shading, whether they are missing or the containment of links of a constant sequence.

  To verify this rule, we check to see that DELTAS only contains:
  - `REMOVE CONSTANT LINK` deltas where for each removed constant link c there is a `REMOVE CONSTANT LINK` : $c; t$ delta for every leaf node with tag t containing c (we are sure that the entire constant sequence is being removed).
  - `REMOVE SHADE` deltas
  - `REMOVE CURVE` deltas. Note that only a `REMOVE CURVE` delta is included in DELTAS when a curve is removed in such a way that all nodes representing counterpart regions agree with regard to shading, whether they are missing, or the containment of links of a constant sequence. If other changes were made while removing the curve then other deltas would be included in DELTAS.
- Introduction of a new curve – D' is the result of adding a new curve to D which divides all of the leaf nodes into two new leaf nodes, in such a way that D' is well-formed, the other labels of D are left undisturbed, and all counterparts agree with respect to shading, whether they are missing, and the containment of links of a constant sequence. To verify this rule,

we check to see that DELTAS only contains one `ADD CURVE` delta. Note that only a `ADD CURVE` delta is included in DELTAS when a curve is added in such a way that all nodes representing counterpart regions agree with regard to shading, whether they are missing, or the containment of links of a constant sequence. If other changes were made while adding the curve then other deltas would be included in DELTAS.

- Inconsistency – This rule requires that there is some collection of nodes in the support that are all shaded and contain all the links of some constant sequence. To verify this rule we check to see that for some constant sequence c in D that every leaf node in D containing c is shaded.
- Adding shaded regions – D' is the result of adding a new leaf node which represents the counterpart of a missing leaf node in D provided that this new node is shaded (and appropriately redrawing the inner nodes of D). To verify this rule, we check to see that DELTAS only contains two deltas one each of the form `ADD REGION` : t and `ADD SHADE` : t with the same tag t. Note that if a node representing a basic region were being added then DELTAS would contain an `ADD CURVE` delta.
- Removing shaded regions – D' is the result of removing a shaded leaf node of D (and appropriately redrawing the inner nodes of D). To verify this rule, we check to see that DELTAS only contains two deltas one each of the form `REMOVE REGION` : t and `REMOVE SHADE` : t with the same tag t. As in the last case, if a node representing a basic region was being removed then DELTAS would contain a `REMOVE CURVE` delta.

Definition 6 *Given a formula φ of EVOF, a DAG D of $\mathcal{DAG}$, and the set* DELTAS *resulting from the application of Algorithm 1 to the empty DAG and d, we say that φ supports D, $\varphi \rhd D$, if every collection of `ADD CONSTANT LINK` deltas all containing the same constant in* DELTAS *is supported by φ, and each `ADD CURVE` and `ADD SHADE` delta in* DELTAS *is supported by φ. The notion of supports is defined as follows.*

- *We say that the collection of `ADD CONSTANT LINK` deltas, $d_1, \ldots d_n$, are supported by φ, $\varphi \rhd$`ADD CONSTANT LINK` : $c; t_1, \ldots,$ `ADD CONSTANT LINK` : $c; t_n$, if:*
  - *φ is unquantified, in the form $\psi(c)$, and all the leaf nodes in D containing the term c are included in $t_1, \ldots, t_n$ and $region_V(\psi(c))$ is a subregion of the union of the regions represented by the nodes $t_1, \ldots, t_n$.*
  - *φ is of the form $\exists x\ \psi(x)$, and all the leaf nodes in D containing the special term x_k are included in $t_1, \ldots, t_n$ and $region_V(\psi(x))$ is a subregion of the union of the regions represented by the nodes $t_1, \ldots, t_n$.*
- *We say that $\varphi \rhd$ `ADD CURVE` : l if the predicate l appears in φ.*
- *We say that $\varphi \rhd$ `ADD SHADE` : t if φ is of the form $\mathsf{N}x\ \psi(x)$ and the leaf node with the tag t represents a subregion of $region_V(\psi(x))$.*

Note that no formula of EVOF supports any `ADD REGION`, `REMOVE CURVE`, `REMOVE REGION`, `REMOVE SHADE`, or `REMOVE CONSTANT LINK` deltas.

Proposition 6 *The DAG produced by Algorithm 3 from the formula of MFOL is supported by that formula.*

Proof sketch for Proposition 6:

Given a formula φ of MFOL, and the DAG D produced by Algorithm 3, we assume that D is not supported by the formula to show a contradiction. We take DELTAS to be the collection of DAG deltas resulting from the application of Algorithm 1 to the empty DAG and D. We begin by observing that DELTAS can not contain any `REMOVE CURVE`, `REMOVE REGION`, `REMOVE SHADE`, `REMOVE CONSTANT LINK`, or `ADD REGION` deltas, since it was applied to the empty DAG and D. (`ADD REGION` deltas can only occur when the initial DAG contains missing leaf nodes.) So there must be one of the following curve deltas in DELTAS which prevents D from being supported by φ:

- `ADD CURVE` : l and then it must be the case that the predicate l doesn't appear in φ. However, for this delta to be in DELTAS D must contain a node representing the basic region with the label l, and for this to occur φ must contain l, contradiction.
- `ADD SHADE` : t and then it must be the case that either φ doesn't have a conjunct the form $\mathsf{N}x\ \psi(x)$ or for all its conjuncts of that form the leaf node with the tag t do not represent a subregion of $region_V(\psi(x))$. However for the node t to be shaded in D φ must have a conjunct of the form $\mathsf{N}x\ \theta(x)$ and the leaf node with the tag t must represent a subregion of $region_V(\theta(x))$, contradiction.
- `ADD CONSTANT LINK` : $c; t$ then it must be the case that either φ doesn't have a conjunct of the form $\psi(c)$ or or it does but for all of them the node t does not represent a subregion of $region_V(\psi(c))$. But since the the constant sequence appears in D, φ must have a conjunct of the form $\theta(c)$ with the node t representing a subregion of $region_V(\theta(c))$, contradiction.
- `ADD CONSTANT LINK` : $x_k; t$ then it must be the case that either φ doesn't have a conjunct of the form $\exists x\ \theta(x)$ or it does but for all of them the node t does not represent a subregion of $region_V(\theta(x))$. But since the the special term sequence appears in D, φ must have a conjunct of the form $\exists x\ \theta(x)$ with the node t representing a subregion of $region_V(\theta(x))$, contradiction.

Proof sketch for Proposition 4:

This proof involves showing that Algorithm 4 only allows valid applications of the Observe rule and that any valid application of the Observe rule is allowed by Algorithm 4. We will begin by showing the first part of this result, that Algorithm 4 only allows valid applications of the Observe rule.

Our goal will be to show that if for some diagram V of $\mathcal{EV}$ and formula of MFOL φ, that if $V \models\!\approx^{+} \varphi$ then Algorithm 4 correctly verifies the application of the rule. We begin by noting that both the definition of observation (Definition 5) and Algorithm 4 begin by working with the conjuncts of a formula of MFOL converted into EVCNF. Also, that in both cases if any of the conjuncts are not in EVOF, then nothing can be observed to hold or fail on the basis of the diagram. Furthermore, we know from Proposition 6 that the DAG D constructed from V in the first stages of Algorithm 4 (using Algorithm 3) is supported by the formula used to generate it. So we can proceed by showing the result for all formulas of EVCNF not containing any quantifiers and with predicates all containing the same term and also for quantified formulas of EVOF (since we are now only looking at groups of conjuncts containing the same term in EVCNF we will consider arbitrary conjunctions separately):

- φ is an unquantified formula of the form $\psi_1(t) \wedge \ldots \wedge \psi_n(t)$ with each $\psi_k(t)$ consisting of a possibly negated single predicate or a disjunction of possibly negated single predicates. Then we know that all of the links of the t constant sequence appear in subregions of $region_V(\psi_1(t) \wedge \ldots \wedge \psi_n(t))$. Algorithm 3 will then produce a DAG D' with a curve for each predicate in $\psi_1(t) \wedge \ldots \wedge \psi_n(t)$ and with constant symbols t in each node representing subregions of the region $region_V(\psi_1(t) \wedge \ldots \wedge \psi_n(t))$. We now consider all the possible DAG deltas resulting from the application of Algorithm 1 on the DAG generated from D and D'.
  - All `REMOVE CURVE`, `ADD REGION`, and `REMOVE SHADE` deltas are permitted in Algorithm 4.
  - `REMOVE REGION` However, as noted, it is not possible for this delta to occur.
  - `ADD CURVE` Since $D \models\!\approx^{+} \varphi$ we know that there aren't any predicates in φ which do not correspond to curves in D, and thus that it is not possible for this delta to occur.
  - `ADD CONSTANT LINK` : $c; t$ However, since in D' there is the constant sequence, c, any `ADD CONSTANT LINK` : $c; t$ deltas are permitted.
  - `ADD SHADE` : t But, since D' does not contain any shading this can not occur.
  - `REMOVE CONSTANT LINK` : $c; t$ Here there are two cases, either a link is being removed or an entire chain is being removed. We will consider the removal of a chain first. Since there is the constant sequence, c, in D', this delta would cause the rule to fail. The existence of this delta means that there is some region in D which has a link of the constant sequence which does not contain a link in D'. This means that information was *added* to D' which is not supported by the information carried by D, namely that there is less uncertainty regarding the location of the constant. Doing this is analogous to removing a disjunct of a disjunction. But if this delta was detected, it would also mean that $D \models\!\approx^{+} \varphi$ would not be the case, thus this can not be possible. In the second case (removing an entire sequence),

there will be a `REMOVE CONSTANT LINK` for each of those constant links and then these deltas are permitted.

- φ is of the form $\mathsf{N}x\ \psi(x)$. Then we know that $region_V(\psi(x))$ is shaded or missing in D. Algorithm 3 will then produce a DAG D' with a curve for each predicate in $\psi(x)$ and with each of the nodes representing subregions of $region_V(\psi(x))$ shaded. We now consider all the possible DAG deltas resulting from the application of Algorithm 1 on the DAG generated from D and D' (ignoring any deltas covered by the same argument as in the last case).
  - `ADD CONSTANT LINK` : $c; t$ But, since D' does not contain any constant sequences this delta can not occur.
  - `ADD SHADE` : t. Here there are two possibilities, first that the shaded region was missing in D and in this case the delta is permitted. However if the region was not missing in D then this delta would cause the rule to fail. But this would mean that information (that certain subregions are shaded), which was not contained in D, was added to D'. But if this delta was detected, it would also mean that $D \models\!\!\approx^{+} \varphi$ would not be the case, thus this can not be possible.
  - `REMOVE CONSTANT LINK` : $c; t$ However, since D' does not contain any constant sequences, this is not possible.
- φ is of the form $\exists x\ \psi(x)$. The argument for this case is analogous to that used for the first case except we allow for the swapping of constant symbols for special terms in the DAG.

Now we need to consider arbitrary conjunctions, in other words when a conjunct $(\psi_1(t_1) \wedge \ldots \wedge \psi_n(t_m))$ in EVCNF is observable from D that Algorithm 4 approves of the observation of each of those conjuncts on the basis of D. In the previous cases we considered very simple DAGS consisting of a number of curves with either shaded regions or one constant sequence. When considering formulas with various conjuncts, the DAG generated by Algorithm 3 will contain more shading and constant sequence information. Thus the main question that needs to be resolved is whether this additional information changes the outcome of Algorithm 4 when applied to each of these conjuncts alone. As established above, we only need to worry about `ADD CONSTANT LINK`, `ADD SHADE`, and `REMOVE CONSTANT LINK` deltas. Using the same argument for `ADD CONSTANT LINK`, `REMOVE CONSTANT LINK` as in the first case above and the argument for `ADD SHADE` as in the second case above this can be established.

We will now show that any valid application of the Observe rule is allowed by Algorithm 4. We assume that there is a valid application of the rule which isn't allowed by Algorithm 4 to show a contradiction. In other words, that there is some MFOL formula φ and Euler/Venn diagram V such that $V \models\!\!\approx^{+} \varphi$ but that failure is returned by Algorithm 4 when run using V and φ. Here we will take the DAG D to be the translation of V. We recall that Algorithm 4 begins by generating a DAG D' which is supported by φ using Algorithm 3. It then uses Algorithm 1 to compute the set DELTAS between D and D'. For the algorithm to fail, one of the following must be the case:

- `ADD CURVE` : l is in DELTAS, but in this case there is a curve in D' which isn't in D, which means that a predicate appears in φ which doesn't appear in V. However if this were the case then φ couldn't be observed, contradiction.
- `ADD CONSTANT LINK` : $c; t$ (or `ADD CONSTANT LINK` : $x_k; t$) is in DELTAS and there is no node t containing a c (or x_k) for which there are no `ADD CONSTANT LINK` : $c; t$ (or `ADD CONSTANT LINK` : $x_k; t$) deltas (i.e., an entire new constant sequence has been added). For this to occur there must be a conjunct in the EVCNF version of φ which consists of either $\psi(c)$ (or $\exists x\ \psi(x)$). However recall that the constant c (or x_k) doesn't appear in V. Since the constant c (or x_k) doesn't appear in V then no formula in EVCNF containing a conjunct of the form $\psi(c)$ (or $\exists x\ \psi(x)$) can be observed from it, contradiction.
- `ADD SHADE` : t is in DELTAS and the node with the tag t is not missing in D. Then there must be a conjunct in the EVCNF version of φ which is in the form $\mathsf{N}x\psi(x)$ with the node t a subregion of $region_V(\psi(x))$. But for this to be observed from V that region must be shaded or missing in V, contradiction.
- `REMOVE CONSTANT LINK` : $c; t$ (or `REMOVE CONSTANT LINK` : $x_k; t$) is in DELTAS but there is some leaf node t' containing c (or x_k) for which there is no `REMOVE CONSTANT LINK` : $c; t'$ (or `REMOVE CONSTANT LINK` : $x_k; t'$) delta(i.e., that the entire constant sequence was not removed). For this to occur there must be a conjunct in the EVCNF version of φ which consists of either $\psi(c)$ (or $\exists x\ \psi(x)$) but that the constant appears in a leaf node which represents a region whose counterpart representing node in D' doesn't have that constant. However, if this is the case than the formula couldn't be observed from V, contradiction.
- `REMOVE REGION` : t is in DELTAS. This is not possible as the canonical DAG D' generated from φ does not have any missing leaf nodes.
- Note that it is not possible for a `REMOVE CURVE`, `ADD REGION`, or `REMOVE SHADE` delta to cause the algorithm to fail.

Biographies

Dr. Nik Swoboda recently began a position as a Ramón y Cajal Research Professor at the Universidad Politécnica de Madrid after spending a year as an invited researcher in the Media Integration Science Laboratories of the ATR research institute in Kyoto, Japan. He received his undergraduate degree from Haverford College, and then completed a MS and Ph.D. in Computer Science at Indiana University. His main research interest is in diagrammatic reasoning and the nature of non-sentential representation systems. He also collaborates in cognitive science research studying the use of graphical representations in communication. In his spare time he enjoys visiting beautiful gardens and temples, taking day hikes, traveling, and eating tasty Japanese and Spanish food. He dreams of the day when he can forsake computers and spend his days sailing and fishing, but in the meantime, he utilizes his imagination to enjoy the world of sailing in sea adventure novels.

Dr. Gerard Allwein is currently at the Naval Research Laboratory in Washingon, D.C., USA. His undergraduate is from Purdue University and holds graduate degrees from Indiana University. Upon graduation, he served as the Visual Inference Laboratory's Assistant Director under Jon Barwise. Dr. Allwein works primarily in mathematical aspects of logic and information. His current interests are diagrammatic reasoning, algebraic logic, and information hiding. Ariel and Tinkerbell, two Siamese house weasels, would like to add that Dr. Allwein is one of the world's biggest gits.