Electronic Notes in Theoretical Computer Science 72 No. 3 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume72.html 15 pages

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

Nik Swoboda [1,2]

Media Information Science Labs
Advanced Telecommunications Research Institute
Seika, Kyoto, Japan

Gerard Allwein [3]

Department of Computer Science
Indiana University
Bloomington, IN, USA

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. It has been shown that a special collection of directed acyclic graphs (DAGs) can be used interchangeably with Euler/Venn diagrams in reasoning processes [4]. 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. In the work reported 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.

1 Overview

It has been shown that a special system of DAGs can be used to *capture the essential properties* of an extended version of Hammer's Venn reasoning system consisting of Euler/Venn diagrams [4]. This DAG system includes a

[1] 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".
[2] Email: nswoboda@atr.co.jp
[3] Email: gtall@cs.indiana.edu

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.

Once we 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. We can then propose algorithms for the analysis of these manipulations to determine their validity as applications of rules of inference. Furthermore, since these data-structure manipulations can be closely modeled in our mathematical theory, we can also prove the validity of these proof verification algorithms. In addition to this, heterogeneous rules involving DAGs and formulas of FOL can also be verified using DAGs. By extracting from the formula all information which can be expressed in an Euler/Venn diagram, a DAG can be constructed and once again DAG manipulations can be used to verify the reasoning.

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 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 7.

2 Background and motivation

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 [4]. Subsequently, theoretical issues involved in the design of the FOL and Euler/Venn logical system have been presented [5,6]. 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.

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.

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. Techniques similar to those discussed here will be used in the verification of the FOL and Euler/Venn portions of the *Openproof* system. We also believe these techniques can provide valuable insight into the characterization of diagram manipulations that could then be applied to proof verification strategies used by other diagrammatic proof systems systems.

3 A brief review of Hammer's Venn system

Euler and Venn diagram notions such as region, basic region, etc. will be used following Hammer's definitions [3]; here we will provide a very brief summary of his terminology. Hammer's Venn system consists of diagrams containing closed curves, shading, constant symbols and lines to connect constant symbols. Curves are used to represent sets. In Venn diagrams, every possible intersection of the curves of the diagram is required to exist in the diagram. An Euler/Venn diagram is a Venn diagram in which Euler-like features, such as set containment, can be expressed. In other words, Hammer's syntactic restriction requiring that every possible curve intersection exist in the diagram is slightly relaxed in the case of Euler/Venn diagrams. An example diagram of this system can be seen in Fig. 1. A *region* of a diagram is any 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*. Each basic region has a unique *label*. A *minimal region* is any region which is not crossed by any of the lines of that diagram (i.e., any region that can not be thought of as

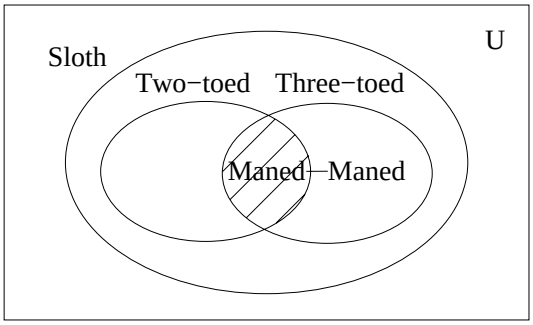

Fig. 1. An Euler/Venn diagram

the union of other regions). Two regions in two different diagrams are said to be *counterparts* if they are both interpreted as representing the same set. Shaded and missing regions in a diagram are used to express the fact that the set denoted by that region is empty. Chains of constant sequences are used to show that the object represented by that constant is a member of one of the sets denoted by the regions containing the links. The collection of well formed Euler/Venn diagrams will be referred to as $\mathcal{EV}$.

Thus, 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 Manned sloths are either Three-toed and not Two-toed or both Three-toed and Two-toed from the placement of the Manned constant symbol links. We will now proceed by presenting the notion of a tag for a region and the homogeneous rules of inference for the Euler/Venn system.

3.1 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 overlapping regions tagged with τ_1 and τ_2 the tag for the intersection of those regions will be $\tau_1 \cup \tau_2$. We then see that the complete tags correspond exactly to all the *potential* minimal regions of the diagram. Here we say "potential" due to the fact that there are complete tags which might correspond to missing minimal regions.

3.2 Euler/Venn rules of inference

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

(i) 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.

(ii) Extending a constant sequence – V' is the result of adding a new c link to a constant sequence of V in a minimal region not already containing a link of that sequence.

(iii) 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 or containment of links of a constant sequence.

(iv) Introduction of a new curve – V' is the result of adding a new curve to V 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 containment of links of a constant sequence.

(v) 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.

(vi) Adding shaded regions – V' is the result of adding a new minimal (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.

(vii) Removing shaded regions – V' is the result of removing a shaded minimal 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 into curves, and the remaining curves should be spaced out to remove points of unintended intersection.

Unification – V' can be inferred from diagrams V_1 and V_2 if it is the case that:

(i) The set of labels of V' is the union of the labels of V_1 and V_2.

(ii) If there is a region in either V_1 or V_2 that is shaded or missing than its counterpart region in V' must be either shaded or missing. Also if there is a region in V' that is shaded or missing than the counterpart region in either V_1 or V_2 must be either shaded or missing.

(iii) If there is a region in either V_1 or V_2 that contains the constant sequence c, than the counterpart region in V' must contain the constant sequence c. Also if there is a region in V' that contains the constant sequence c, then the counterpart region in either V_1 or V_2 must also contain the constant sequence.

4 Euler/Venn diagrams and DAGs

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

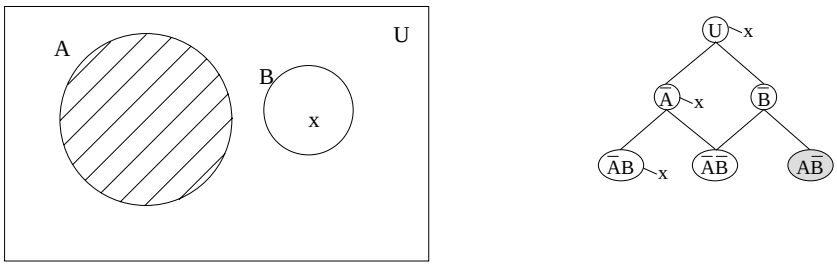

Fig. 2. A sample Euler/Venn diagram and its corresponding DAG

be taken to represent the cover relation between regions of the diagram.[4] Each DAG will have one node representing its domain (the collection of all the objects that the diagram could explicitly represent), and other nodes each uniquely representing each of its corresponding diagram's 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 the union of curve labels and their complements. With the exception being the root node labeled U, each node is labeled with the tag which corresponds to the region it represents. For example, the node labeled $\overline{A}\,\overline{B}C$ is taken to represent the intersection of the complements of the basic regions A and B, and the basic region C. Thus, the DAG's root node represents the domain of the diagram, the diagram's rectangle, and the leaf nodes of the DAG represent the minimal regions of the diagram. The collection of well formed DAGs will be referred to as $\mathcal{DAG}$.

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.

5 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

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

delta will denote some meaningful transformation to the 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 5.1 Given any curve label l, constant symbol c, and complete tag t all relating to some diagram $V \in \mathcal{EV}$, 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 simple 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 with the same 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. In the case of removing a curve, we use the Erasure rule of inference to remove that curve from the DAG. 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* DELTAS *of DAG deltas describing the differences between D and D'.*

(i) *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.*

(ii) *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 region 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 region in D is shaded we add* `REMOVE SHADE` : t *to* DELTAS *and*

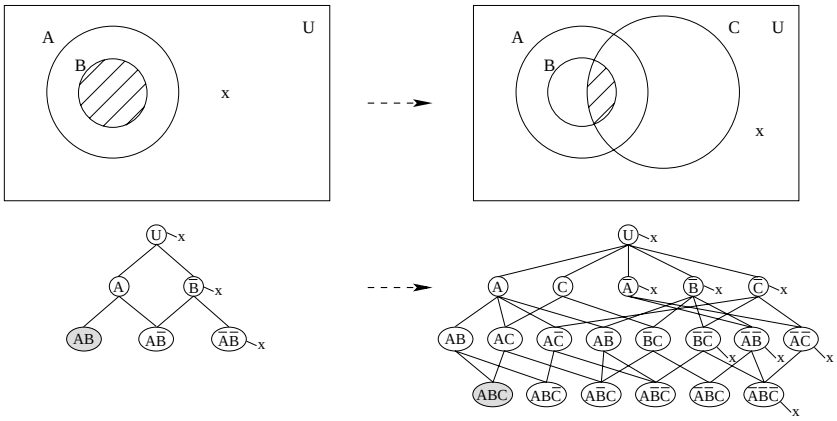

Fig. 3. An example of computing DAG deltas

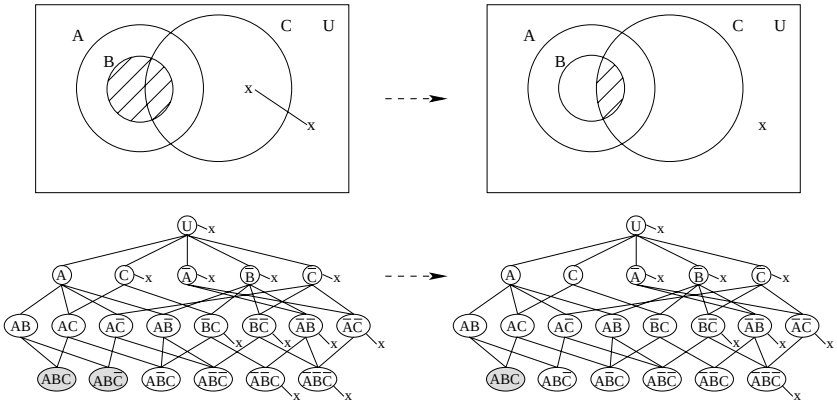

Fig. 4. The result of adding curve C to the left diagram

if it contains links of constant sequences we add `REMOVE CONSTANT LINK`$: c; t$ *deltas to* DELTAS *for each constant link* c.

(iii) *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.
  (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. 3, we first need to add a curve to the support diagram resulting in the situation in Fig. 4. Thus the collection DELTAS for Fig. 3 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. 3 there is no node labeled B since the basic region B is the same as the minimal region AB which is included in the DAG.

Lemma 5.2 *All logically significant differences between any two DAGs in $\mathcal{DAG}$ can be captured with a set of DAG deltas.*

Theorem 5.3 *Algorithm 1 for computing DAG deltas captures all logically significant differences between the two DAGs.*

6 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 without difficulty. 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 CONSTANT LINK` delta whose region is shaded in the resulting DAG. More detailed algorithms for all of the Euler/Venn homogeneous rules of inference 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 a homogeneous rule of inference(checking of the Unification rule will be given in the next section):*

- *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 region 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` $: t$ *and 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.*

6.1 The Unification rule

In the case of the Unification rule there are two support diagrams and one resulting diagram. This rule is used to combine the information contained in two different diagrams into one diagram. An example of the use of the Unification rule can be found in Fig. 5. To check this rule it is first verified that the set of basic regions in the resulting diagram is the union of the basic regions in the two supporting diagrams. If this isn't the case then the rule fails. Then, as above, the curves that are in the resulting diagram that aren't in either of the support DAGs are added, and any curves not in the resulting diagram are removed. Next we find the deltas between the first support diagram and the result and the second support diagram and the result. Then we sort through the deltas searching for a delta that is not allowed. Deltas that are allowed are deltas that are based upon information in the other support diagram. For example if region t in the second support has a constant which also appears in the result in t, then the fact that it is missing from the first support can be ignored. Also, if in one support, region t' is shaded and in the other region t' is missing then the result can have either a missing or shaded region r'. An example of this process, using Euler/Venn diagrams in place of DAGs, can be found in Fig. 5. The top two diagrams are the support diagrams and the second row of diagrams are the result of adding Pencil Urchin curve to the first support and then the Eel curve to the second support. We then compare the contents of all of the minimal regions. On close examination, it can be seen that any discrepancy between the first or the second support and the result is justified on the basis of the other support. For example, though the region with the tag $\overline{\text{Sea-Urchin}}$ Pencil-Urchin $\overline{\text{Eel}}$ is not missing in the first support and is missing in the result, it is missing in the second support so this modification is justified.

Algorithm 3 *Given $D, D', D'' \in \mathcal{DAG}$, we do the following to check the application of the Unification rule of inference.*

(i) *If there is some curve in D'' not in either of D or D', or some curve in D or D' that is not in D'', then the rule fails.*

(ii) *Using Algorithm 1 we compute the set* DELTAS *between D and D'' and the set* DELTAS$'$ *between D' and D''.*

(iii) *While using D' as "the other support" we check that for each delta $d \in$* DELTAS *that if d is:*

- *`ADD REGION` : t, then the node labeled t is shaded in the other support and in D''.*
- *`REMOVE REGION` : t, then the node labeled t is missing in the other support.*
- *`ADD SHADE` : t then the node labeled t is is shaded in the other support.*
- *`REMOVE SHADE` : t, then the node labeled t is missing in the other support and in D''.*

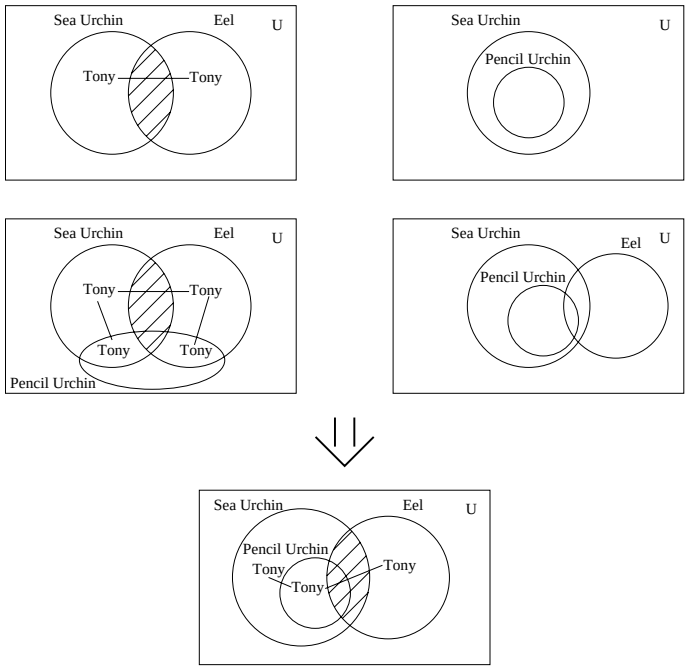

Fig. 5. The checking of the Unification rule

- *ADD CONSTANT LINK* : $c; t$, *then the node labeled* t *in the other support has the constant link* c.
- *REMOVE CONSTANT LINK* : $c; t$, *then the link* c *is missing in the node labeled* t *in the other support and there is some other minimal region in* D *and* D' *containing a link of* c *(i.e., that the constant sequence isn't being removed, and that if it isn't being removed that that constant appears in the other diagram).*
- *ADD CURVE* : l *then a curve labeled* l *appears in the other support.*
- *otherwise the use of the* Unification *rule fails (it always fails when encountering* *REMOVE CURVE* *deltas).*

(iv) *We repeat the last step to check each* $d' \in \text{DELTAS}'$ *but this time we use* D *as the other support.*

Theorem 6.1 *Algorithm 2 and Algorithm 3 correctly check the use of all the Euler/Venn homogeneous rules of inference.*

7 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 symmetric, 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 [6], 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 $\neg\exists x\ (A(x) \wedge B(x))$.

7.1 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.

Let $\mathcal{L}$ be some finite 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. 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 7.1 Euler/Venn Observational Formulas (**EVOF**)

(i) Basic formulas: For every predicate P in $\mathcal{L}$, and term t in Term, $P(t)$ is in **EVOF**.

(ii) Negations, Conjunctions, and Disjunctions: For every $\varphi_1(t), \ldots, \varphi_n(t)$ in **EVOF**, the following are also in **EVOF**:
- $\neg\varphi_1(t)$
- $(\varphi_1(t) \wedge \ldots \wedge \varphi_n(t))$
- $(\varphi_1(t) \vee \ldots \vee \varphi_n(t))$

(iii) Quantifiers: For every unquantified $\varphi(x)$ in **EVOF**, the following are also in **EVOF**:
- N$x\ \varphi(x)$ (read as "There is no x such that $\varphi(x)$.")
- $\exists x\ \varphi(x)$

Definition 7.2 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:

(i) For each basic region r labeled P in V $region_V(P(t)) = r$.

(ii) If $region_V(\varphi_1(t)) = r_1, \ldots, region_V(\varphi_n(t)) = r_n$ then:
- $region_V(\neg\varphi_1(t)) = \overline{r_1}$. [5]
- $region_V((\varphi_1(t) \wedge \ldots \wedge \varphi_n(t))) = r_1 \cap \ldots \cap r_n$
- $region_V((\varphi_1(t) \vee \ldots \vee \varphi_n(t))) = r_1 \cup \ldots \cup r_n$
- $region_V(\mathsf{N}x\ \varphi(x)) = region_V(\exists x\ \varphi(x)) = region_V(\varphi(x))$

Definition 7.3 The relations of *strong observation*, $V \models\!\!\!\sim^{+} \varphi(t)$ and $V \models\!\!\!\sim^{-} \varphi(t)$, 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:
  - $\boldsymbol{V \models\!\!\!\sim^{+} \varphi(t)}$ if the term symbol t appears in $region_V(\varphi(t))$. [6]
  - $\boldsymbol{V \models\!\!\!\sim^{-} \varphi(t)}$ if the term symbol t appears in the complement of $region_V(\varphi(t))$.
- For quantified formulas:
  - $\boldsymbol{V \models\!\!\!\sim^{+} \mathsf{N}x\ \psi(x)}$ if the region $region_V(\psi(x))$ is shaded or missing.
  - $\boldsymbol{V \models\!\!\!\sim^{-} \mathsf{N}x\ \psi(x)}$ if $V \models\!\!\!\sim^{+} \exists x\ \psi(x)$.
  - $\boldsymbol{V \models\!\!\!\sim^{+} \exists x\ \psi(x)}$ if some term symbol t appears in $region_V(\psi(x))$.
  - $\boldsymbol{V \models\!\!\!\sim^{-} \exists x\ \psi(x)}$ if $V \models\!\!\!\sim^{+} \mathsf{N}x\ \psi(x)$.

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

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

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

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

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

$V \not\approx^{?} \varphi$ will be written if neither $V \not\approx^{+} \varphi$ nor $V \not\approx^{-} \varphi$.

7.2 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, $V \not\approx^{+} \varphi$, as the negative part can be addressed analogously.

Algorithm 4 *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.*

(i) *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.*

(ii) *Convert φ into φ' in EVCNF. If any of the conjuncts of φ' is not in EVOF then the construction fails.*

(iii) *For each conjunct 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 minimal region of D that is a subregion of $region_D(\theta_1(t) \vee \ldots \theta_n(t))$.*
- $\mathsf{N}x\ \theta(x)$ *- We shade every minimal region of D that is a subregion of $region_D(\theta(x))$.*
- $\exists x\ \theta(x)$ *- We make a new constant sequence in D placing the term t in every minimal region of D that is a subregion of $region_D(\theta_1(t_1) \vee \ldots \theta_n(t_n))$.*

So now to verify the observation of a formula of MFOL from an Euler/Venn diagram we first use Algorithm 4 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 are examined to insure that no inappropriate changes have occurred. The swapping of shaded regions for missing regions, and the weakening of information is permitted. However, any changes which introduce new information are not permitted.

Algorithm 5 *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;*

(i) *Construct the DAG D' which is supported by φ using Algorithm 4. If the construction fails then the rule fails.*

(ii) *Use Algorithm 1 to compute the set* DELTAS *between D and D'.*

(iii) *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` : $c;t$ *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 region* 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 region was missing in* D*.*
- *otherwise the use of the* Observe *rule fails. (Note that is is not possible for a* `REMOVE REGION` *delta to occur, since Algorithm 4 does not produce DAGs with missing regions.)*

Theorem 7.5 *The verification technique given in Algorithm 5, correctly verifies the* Observe *rule presented in Definition 7.4.*

7.3 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).

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] Hammer, E. M., "Logic and Visual Information," CSLI and FOLLI, Stanford, 1995.

[4] 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.

[5] Swoboda, N. and G. Allwein, *A case study of the design and implementation of heterogeneous reasoning systems*, in: L. Magnani and N. J. Nersessian, editors, *Logical and Computational Aspects of Model-Based Reasoning*, Kluwer Academic, Dordrecht, 2002 .

[6] Swoboda, N. and G. Allwein, *Modeling heterogeneous systems*, in: M. Hegarty, B. Meyer and N. H. Narayanan, editors, *Diagrammatic Representation and Inference*, number 2317 in Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, 2002 pp. 131–145.