Diagrammatic Logics

Aim

To make proofs more accessible than symbolic logics

A (formal) proof is a sequence of diagrams where each diagram is an axiom or derived, using an inference rule, from diagrams written down earlier in the proof.
Diagrammatic Logics

Aim
To make proofs more accessible than symbolic logics

A (formal) proof is a sequence of diagrams where each diagram is an axiom or derived, using an inference rule, from diagrams written down earlier in the proof.

Objectives

1. Design diagrams for representing logical information. Lots done here...
2. Design sound and complete inference systems. ...and here.
3. But what about accessibility? more later!
Foundational and Pioneering Research
Diagrammatic Logics: Venn and Euler Family

Venn-I and Venn-II

2. Venn diagrams with extra syntax.

Venn Diagrams

![Diagram](image-url)
Diagrammatic Logics: Venn and Euler Family

Venn-I and Venn-II

2. Venn diagrams with extra syntax.

Venn Diagrams

NOT Venn Diagrams – these are Euler Diagrams
Diagrammatic Logics: Venn and Euler Family

Venn-I Diagrams

A

B

A

B

A

B

C
Diagrammatic Logics: Venn and Euler Family

Venn-I Diagrams

Venn-II Diagrams
Diagrammatic Logics: Venn and Euler Family

Venn-I Diagrams

Expressiveness

\( (\forall x \neg (A(x) \land B(x))) \land \exists x ((A(x) \land \neg B(x)) \lor (B(x) \land \neg A(x))) \)

Monadic First-Order Logic (MFOL)
Diagrammatic Logics: Venn and Euler Family

Inference in Venn-I

\[
\begin{align*}
A \cup B &\neq \emptyset \\
A \cap B &\neq \emptyset
\end{align*}
\]

\[
\begin{align*}
A &\neq \emptyset \\
B &\neq \emptyset
\end{align*}
\]
Diagrammatic Logics: Venn and Euler Family

Inference in Venn-I

\[ A \cap B \quad \Rightarrow \quad \bot \quad \Downarrow \quad \text{spreading} \quad \otimes \]

\[ A \quad \Rightarrow \quad \bot \quad \Downarrow \quad \text{spreading} \quad \otimes \]

\[ A \cup B \quad \Rightarrow \quad \bot \quad \Downarrow \quad \text{spreading} \quad \otimes \]

\[ A \quad \Rightarrow \quad \bot \quad \Downarrow \quad \text{spreading} \quad \otimes \]

\[ A \quad \Rightarrow \quad \bot \quad \Downarrow \quad \text{spreading} \quad \otimes \]
Key Results for Venn-I and Venn-II

1. Shin devised sound and complete inference systems.
2. Completeness proof uses normal forms.
3. Expressiveness of Venn-II: MFOL.
Diagrammatic Logics: Venn and Euler Family

Other Logics: Venn-i, Choudhury and Chakraborty (2012)

c \in A \cup B \land A \cap B \neq \emptyset

c \notin U - (A \cup B) \land A \cap B \neq \emptyset
Diagrammatic Logics: Venn and Euler Family

Other Logics: Venn-i, Choudhury and Chakraborty (2012)

c ∈ A ∪ B ∧ A ∩ B ≠ ∅

Key Points for Venn-i

1. Inference rules generalize Shin’s work.
2. Expressiveness of Venn-i: MFOL.
3. *Only* logic in this family with syntax for *absence* of individuals.
4. Similar to Euler/Venn logic, Swoboda and Allwein (2005).
Diagrammatic Logics: Venn and Euler Family

Other Logics: Spider Diagrams, Gil et al. (1999)

\[ \exists x \exists y \ x \in A \cup B \land y \in A \cap B \land x \neq y \]

\[ \exists x \exists y \ x \in A \land y \in B \land x \neq y \]
Diagrammatic Logics: Venn and Euler Family

Other Logics: Spider Diagrams, Gil et al. (1999)

Key Points for Spider Diagrams

1. Build on Euler diagrams.
2. Inference rules build on Shin’s Venn-II logic.
4. Theorem provers implemented, Urbas et al. (2012).

g.e.stapleton@brighton.ac.uk  www.ontologyengineering.org
Diagrammatic Logics: Venn and Euler Family

Other Logics

2. Euler diagrams with logical connectives, Stapleton and Masthoff (2007).
3. Euler diagrams for syllogistic reasoning, Mineshima et al. (2013).
5. Spider diagrams of order, Delaney et al. (2008).
9. and others... mostly monadic first-order.
Summary of the State-of-the-Art

1. Many formal diagrammatic logics devised.
2. Sound and, sometimes, complete inference systems.
3. Expressiveness typically monadic first-order.
4. Inference rule design targeted at getting completeness.
Diagrammatic Logics

Summary of the State-of-the-Art

1. Many formal diagrammatic logics devised.
2. Sound and, sometimes, complete inference systems.
3. Expressiveness typically monadic first-order.
4. Inference rule design targeted at getting completeness.

1. How is completeness proved?
2. How does this inform inference rule design?
3. Why is this problematic?
4. A new approach to inference rule design.
Example for Shading

A \cap B
Diagrammatic Logics: Spider Diagrams

Example for Shading

![Spider Diagram Example]

Informal Syntax and Semantics

1. Curves represent sets; curve give rise to zones, formally a pair \((in, out)\).
2. Spiders place lower bounds on set cardinality.
3. Shading places upper bounds on set cardinality.
4. Logical connectives, \(\land\) and \(\lor\), for complex expressions.

See Howse et al. (2005) for formalization.
The Axiom of Choice allows you to select one element from each set in a collection and have it executed as an example to the others.

My math teacher was a big believer in proof by intimidation.

Credit: XKCD (used under CC licence)
Diagrammatic Reasoning

### Aim

To make proofs more accessible than symbolic logics

A (formal) **proof** is a sequence of diagrams where each diagram is an axiom or derived, using an *inference rule*, from diagrams written down earlier in the proof.

### Objectives

1. Design diagrams for representing logical information. **Lots done here**...
2. Design sound and complete inference systems. **Focus been on soundness and completeness**.
3. But what about accessibility?
Diagrammatic Logics: Spider Diagrams

Example

\[ d_1 \]

\[ A \quad \quad \quad \quad B \]

\[ d'_1 \]

Rules

1. Delete shading.
2. Delete spider from non-shaded zones.
Diagrammatic Logics: Spider Diagrams

Example

![Diagram](image)

Rules

1. Delete shading.
2. Delete spider from non-shaded zones.

Mini-Completeness Result

If $d_1$ and $d'_1$ contain the same zones, all spiders are in single zones and $d_1 \models d'_1$ then $d_1 \vdash d'_1$ using delete shading and delete spiders.
Example

\begin{align*}
\text{If all unitary parts of } D &= d_1 \lor d_2 \lor \ldots \lor d_n \text{ and } D' = d'_1 \lor d'_2 \lor \ldots \lor d'_n \\
\text{contain the same zones, all spiders are in single zones and } d_i &\models d'_i \text{ then } D \models D' \text{ using delete shading and delete spiders.}
\end{align*}
Example

```
D = d₁ ∨ d₂ ∨ ... ∨ dₙ
D' = d'₁ ∨ d'₂ ∨ ... ∨ d'ₙ
```

If all unitary parts of $D = d_1 \lor d_2 \lor ... \lor d_n$ and $D' = d'_1 \lor d'_2 \lor ... \lor d'_n$ contain the same zones, all spiders are in single zones and $d_i \models d'_i$ then $D \models D'$ using delete shading and delete spiders.

Question

Given arbitrary $D$ and $D'$, can we design inference rules that allow us to use above result to prove completeness?

YES!
Equalizing zones

1. add curves (make label sets the same)
2. add zones (make zone sets the same).

Single Zone Spiders

1. split spiders

\[ A \cong \begin{array}{c} A \quad B \end{array} \quad \lor \quad \begin{array}{c} A \quad B \end{array} \]
Diagrammatic Logics: Spider Diagrams

Equalizing zones

1. add curves (make label sets the same)
2. add zones (make zone sets the same).

Single Zone Spiders

1. split spiders

Disjunctions of Unitary Diagrams: Remove $\land$

1. combining

![Diagrams](attachment:diagrams.png)

Contact:
g.e.stapleton@brighton.ac.uk www.ontologyengineering.org
Completeness Proof Strategy

$D \rightarrow D_L \rightarrow D_Z \rightarrow D_S \rightarrow D_\lor$

$D' \rightarrow D'_L \rightarrow D'_Z \rightarrow D'_S \rightarrow D'_\lor$

- add curves
- add zones
- reduce spiders
- remove $\land$
Example Proof

If $A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$. 
Example Proof

\[
\begin{align*}
&B \quad A \quad \bigcirc \quad \bigcirc \\
&d_1 \\
\wedge \\
&C \quad B \quad \bigcirc \quad \bigcirc \\
&d_2 \\
\Rightarrow \\
&C \quad A \quad \bigcirc \quad \bigcirc \\
&d
\end{align*}
\]
Example Proof

\[ \begin{align*}
\text{Diagram } d_1 &\quad \land \quad \text{Diagram } d_2 \\
\text{Diagram } d_1 &\quad \equiv \quad \text{Diagram } d_2
\end{align*} \]
Example Proof

\[ \begin{align*}
B \quad A & \quad C \quad B \\
\land & \quad \land & \quad = \\
\text{d}_1 & \quad \text{d}_2 & \\
A \quad B & \quad C \quad B \\
\land & \quad \land &
\end{align*} \]
Example Proof

\[
\begin{align*}
\text{Diagram 1:} & \quad B \in A \\
\text{Diagram 2:} & \quad C \in B \\
\text{Diagram 3:} & \quad C \in A
\end{align*}
\]
Example Proof

$$d_1 \land \vdash d_2$$

$$d_1 \land \vdash d_2$$

$$d_1 \land \vdash d_2$$

$$d_1 \land \vdash d_2$$
Example Proof

Not the most accessible proof!
Diagrammatic Logics: Summary

Proof Writing

1. Inference rule design geared toward completeness.
2. Inference rule style leads to unnatural proofs.
3. Not just spider diagrams, other diagrammatic logics have rules designed for obtaining completeness.

Rules Based on Observation

**Question:** Can inference rules be designed so that the help us write more accessible proofs?

We argue rule design should focus on information observable from diagrams, for accessibility.
Observing and Copying Principles

Observation Principles

**Curves** If we can *observe* subset or disjointness information given by the curves in one diagram then we should be able to *copy* that information to another diagram.
Observing and Copying Principles

Observation Principles

**Curves** If we can *observe* subset or disjointness information given by the curves in one diagram then we should be able to *copy* that information to another diagram.

**Spiders** If we can *observe* a lower bound on a set’s cardinality given by the spiders in one diagram then we should be able to *copy* that information to another diagram.
Observing and Copying Principles

Observation Principles

Curves If we can observe subset or disjointness information given by the curves in one diagram then we should be able to copy that information to another diagram.

Spiders If we can observe a lower bound on a set’s cardinality given by the spiders in one diagram then we should be able to copy that information to another diagram.

Shading If we can observe an upper bound on a set’s cardinality given by the shading in one diagram then we should be able to copy that information to another diagram.
Observing and Copying Principles

Observation Principles

Curves  If we can observe subset or disjointness information given by the curves in one diagram then we should be able to copy that information to another diagram.

Spiders  If we can observe a lower bound on a set’s cardinality given by the spiders in one diagram then we should be able to copy that information to another diagram.

Shading  If we can observe an upper bound on a set’s cardinality given by the shading in one diagram then we should be able to copy that information to another diagram.

Syntax

When do syntactically different regions represent the same set?
The shaded zone in $d_1$ is ($\{B\}, \{A, D\}$) and represents the set $B - (A \cup D)$. 
Comparing Syntax Across Unitary Diagrams

Abstract Syntax

The shaded zone in $d_1$ is ($\{B\}, \{A, D\}$) and represents the set $B - (A \cup D)$.

$$\{B\} - (A \cup D)$$

The set $B - (A \cup D)$ is the same as

$$(B \cap C) - (A \cup D) \cup (B - (A \cup D \cup C))$$

Thus, ($\{B\}, \{A, D\}$) $\equiv_c$ $\{(\{B, C\}, \{A, D\}), (\{B\}, \{A, D, C\})\}$
Comparing Syntax Across Unitary Diagrams

**Corresponding Regions**

\[ z = (in, out) \] represents the same set as \((in \cup \{X\}, out)\) ‘union’ \((in, out \cup \{X\})\).

Can syntactically derive when two regions represent the same set or sub/super sets of each other: corresponding (sub/super) regions.

**Theorem**

Corresponding regions represent the same set. Corresponding sub-regions (resp. super-regions) are in a subset-superset relationship (resp. superset-subset relationship).
New Inference Rules

Approach

Given $d_1 \land d_2$, define rules that alter $d_1$ using information observable from $d_2$ without changing the informational content of the conjunction.

Rules

1. Copy a curve.
2. Copy a spider.
3. Copy shading.
4. Add a region.
5. Add zones to a spider’s habitat (the region in which the spider is placed).

See Stapleton et al. (2013c) for details.
Inference Rule: Copy a Curve

Example

Copy $D$ into $d_1$, giving $d'_1$:

\[
\begin{array}{cccc}
A & B & C \\
\cap & & \\
\end{array}
\quad \quad
\begin{array}{cccc}
A & B \\
D & E \\
\cap & & \\
\end{array}
\quad \quad
\begin{array}{cccc}
A & B \\
D & C \\
\cap & & \\
\end{array}
\quad \quad
\begin{array}{cccc}
A & B \\
D & E \\
\cap & & \\
\end{array}
\]
Inference Rule: Copy a Curve

Example

Copy $D$ into $d_1$, giving $d'_1$:

In $d_1$, $\{B, C\}, \{A\}$ represents a superset of $D$, given $d_2$, using the notion of corresponding super-regions. Copy $D$ completely inside $\{B, C\}, \{A\}$, giving $d'_1$. 

g.e.stapleton@brighton.ac.uk
This is now a proof.
Inference Rule: Copy a Spider

Example

Copy a spider into the region $\{(\{A\}, \{B, C\}), (\{A, C\}, \{B\})\}$, giving $d'_1$:

$$d'_1$$

$$d'_2$$

$$d_2$$

$$d_1$$

Corresponding Regions

$\{(\{A\}, \{B, C\}), (\{A, C\}, \{B\})\} \equiv_c \{(\{A\}, \{B, D, E\})\}$ in $d_2$.

From $d_2$: $|A - (B \cup D \cup E)| \geq 1$.

Using corresponding regions: $|(A - (B \cup C)) \cup ((A \cap C) - B)| \geq 1$.

Copy this information to $d_1$. 

g.e.stapleton@brighton.ac.uk
Inference Rule: Copy Shading

Example

Copy shading into the region containing all zones inside $B$, giving $d'_1$:

Corresponding Regions

The region comprising all zones inside $B$ in $d_1$ corresponds to the similar region in $d_2$.

Both diagrams: $|B| \geq 1$.

From $d_2$: $|B| = 1$.

Copy this information to $d_1$.
Impact on Proofs

- We can now use observable information when applying inference rules.
- Still have completeness, as a conservative extension to existing logic.
- Proofs never need to get longer.
- Proofs may get shorter.
- Theorem proving support could be useful: Speedith, Urbas et al. (2012).
Real World Applications
Example

\[ A \cap B \subseteq \bot \]
\[ \forall R. A \subseteq B \]
Key Points for Concept Diagrams

1. Designed for ontology engineering.
2. Formal syntax and semantics, Stapleton et al. (2013a).
4. Expressiveness: DSOL[=], Stapleton et al. (2013a).
Concept Diagrams: Semantic Sensor Networks

In Description Logic:

\[
\begin{align*}
Object & \sqsubseteq Entity, \ Event \sqsubseteq Entity, \ Quality \sqsubseteq Entity, \ Abstract \sqsubseteq Entity, \ Object \cap Event \sqsubseteq \bot, \\
Object \cap Quality & \sqsubseteq \bot, \ Object \cap Abstract \sqsubseteq \bot, \ Event \cap Quality \sqsubseteq \bot, \ Event \cap Abstract \sqsubseteq \bot, \\
Quality \cap Abstract & \sqsubseteq \bot, \ Sensor \equiv Stimulus, \ Sensor \sqsubseteq Event, \\
PhysicalObject \cap SocialObject & \sqsubseteq \bot, \ PhysicalObject \sqsubseteq Object, \ Sensor \sqsubseteq PhysicalObject, \\
System & \sqsubseteq PhysicalObject, \ Device \sqsubseteq System, \ SensingDevice \sqsubseteq Device \cap Sensor, \\
Description \cap InformationObject & \sqsubseteq \bot, \ Description \cap Situation \sqsubseteq \bot, \\
InformationObject \cap Situation & \sqsubseteq \bot, \ Description \sqsubseteq SocialObject, \ Method \sqsubseteq Description, \\
Process & \sqsubseteq Method, \ Sensing \sqsubseteq Process, \ SensorOutput \sqsubseteq InformationObject, \\
Observation & \sqsubseteq Situation, \ Property \sqsubseteq Quality, \ MeasurementCapability \sqsubseteq Property, \\
\forall detects.Sensor & \sqsubseteq Stimulus, \ \forall hasMeasurementCapability.Sensor \sqsubseteq MeasurementCapability, \\
\forall observes.Sensor & \sqsubseteq Property, \exists implements.Sensor \sqsubseteq Sensing, \\
\forall observedProperty.Observation & \sqsubseteq Property, \ \forall observedBy.Observation \sqsubseteq Sensor, \\
\forall sensingMethodUsed.Observation & \sqsubseteq Sensing, \\
\forall observationResult.Observation & \sqsubseteq SensorOutput, \\
\exists observedProperty.Observation & \sqsubseteq Property, \ \exists observedBy.Observation \sqsubseteq Sensor, \\
\exists sensingMethodUsed.Observation & \sqsubseteq Sensing, \exists includesEvent.Observation \sqsubseteq Stimulus.
\end{align*}
\]
Conclusion

CONCLUSION: Bubble-flippin-tastic!
Conclusions

Diagrammatic Logics

- Overview of existing state-of-the-art.
- Identified need for better designed inference rules.
- Demonstrated a new approach to inference rule design using observation.
- Evaluation needed:
  - do proofs become easier to read/understand?
  - are proofs easier to write?
  - how often do proofs get shorter? (standard corpus of examples)
- Scope for improving inference rules in many diagrammatic logics.
Concluding Remarks

Ontology Engineering

1. Concept diagrams.
2. Need patterns for capturing common constraints.
3. Tool support is vital: sketching, layout, reasoning.

What does the future hold?

Tool support, and research challenges!
Concluding Remarks

Example: Ontology Creation

Need diagram creation tools:

Credits


g.e.stapleton@brighton.ac.uk  www.ontologyengineering.org
Example: Ontology Creation

Need diagram creation tools:

Credits

Concluding Remarks

Example: Ontology Creation

Need diagram creation tools:

Next Steps:

Extend SketchSet to concept diagrams; Convert abstract syntax to OWL or DL; integrate with ontology development tools.
Concluding Remarks

Example: Ontology Creation

Need OWL or DL ontology visualization tools:

g.e.stapleton@brighton.ac.uk
www.ontologyengineering.org
Concluding Remarks

Example: Ontology Creation

Need OWL or DL ontology visualization tools:

Credits

iCircles code largely written by Jean Flower (Stapleton et al. (2012)), extended by Jim Burton (and others) to integrate with Protégé.

g.e.stapleton@brighton.ac.uk
Concluding Remarks

Vision for Software Support

1. Intelligent sketching and traditional diagram editing software.
2. Automated conversion between concept diagrams and OWL/DL.
3. OWL or DL to concept diagrams: sophisticated diagram layout algorithms.
4. Support for modular, distributed ontology development.
Time for Questions

Credit: Flickr user Marcus Ramberg (used under CC licence)


