Heterogeneous Proofs: Spider Diagrams meet Higher-Order Provers

Matej Urbas and Mateja Jamnik

University of Cambridge
{Matej.Urbas,Mateja.Jamnik}@cl.cam.ac.uk

Abstract. We present an interactive heterogeneous theorem proving framework, which performs formal reasoning by arbitrarily mixing diagrammatic and sentential proof steps.
We use Isabelle to enable formal reasoning with either traditional sentences or spider diagrams. We provide a mechanisation of the theory of abstract spider diagrams and establish a formal link between diagrammatic concepts and the existing theories in Isabelle/HOL.

1 Introduction

Diagrams are often employed as illustrations in "pen and paper" reasoning. In the past, they frequently formed essential parts of proofs. Eventually, with advent of proof theory, their role became almost exclusively that of a visual help. Still, the intuitive nature of diagrams motivated the design of formal diagrammatic reasoning systems – for example, spider diagrams [6] and constraint diagrams [3]. Consequently, some purely diagrammatic theorem provers have been developed, DIAMOND [8], Edith [10] and Dr.Doodle [13] are some examples.

Heterogeneous reasoning was the next step in the development of diagrammatic reasoning systems. It merged the diagrammatic and sentential modes of reasoning and allowed proof steps to be applied to either diagrammatic, sentential or mixed formulae. In the paper *Reasoning with Sentences and Diagrams* [5], Hammer laid the formal foundations for such heterogeneous reasoning systems.

Later, Barwise [2], Barker-Plummer [1] and Shin [9] investigated heterogeneous reasoning software. The result was a framework called Openproof [1], which uses diagrammatic representation as an input method. The diagrammatic part of the framework is not formalised within the logic of the sentential reasoner. Therefore, the diagrammatic and sentential components remain logically separated.

Our goal is to enable formal interactive heterogeneous reasoning in a general purpose theorem prover. We investigate three aspects of interactive heterogeneous reasoning: *a)* the direction of proofs (e.g., from a diagrammatic assumption to a sentential conclusion and vice versa), *b)* expression of statements that contain mixed sentential and diagrammatic terms, and *c)* mixed application of diagrammatic, sentential, and heterogeneous inference steps.

Our key motivation is to provide different points of view on formulae and to enable reasoning about diagrams sententially or vice versa. We believe that

heterogeneous reasoning will not only serve as a pedagogical tool for introduction to logic, it may also improve intuitiveness and readability of formulae (and proofs) in specific domains of verification – analogous to other domain specific languages. Another motivation for heterogeneous reasoning is the ability to augment diagrams with sentential reasoning wherever diagrams fall short.

In contrast to the approach of Openproof our aim is not to keep the two reasoning systems separated, but to integrate them using heterogeneous representation and reasoning. In addition, we want to formalise diagrams and some of their inference rules in the logic of an LCF-style [4] higher-order theorem prover. With this we aim to enable certified proof reconstruction of heterogeneous proofs.

In order to build a heterogeneous reasoning framework, we first chose an existing diagrammatic reasoning language called *spider diagrams* (see Section 2). The second part is the sentential reasoner, for which we chose Isabelle [12].

We formalised the theory of spider diagrams in Isabelle/HOL (see Section 3.1). This enabled sentential reasoning about diagrams and also simplified translation from spider diagrams to sentences (see Section 3.2). Translation from sentences to diagrams, proof automation, and proof reconstruction, however, is still work in progress. Diagrammatic reasoning will be done in Speedith, our own external reasoner, which is currently in development (see Section 4).

2 The Diagrammatic Language

We have picked the language of spider diagrams [7] as the diagrammatic part of our heterogeneous reasoner because it has a formally defined syntax and semantics. Spider diagrams are equivalent to first-order monadic logic and are equipped with a number of purely diagrammatic inference rules, which have been shown to be sound[1].

Spider diagrams consist of the following basic elements (see Figure 1):

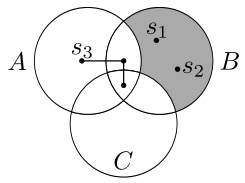

Fig. 1: A spider diagram featuring all the basic diagrammatic language elements.

Contours represent named sets. They are drawn as labelled circles (e.g., circles A, B and C in Figure 1).

Zones are also outlined areas and denote specific subsets of contours and their complements (Figure 1 contains 8 zones).

Spiders are single existentially quantified elements. One spider is a finite collection of dots that are connected with lines. The dots are called *feet*, which indicate the zones in which the spider may live.

Shaded zones indicate that a zone is a subset of its spiders (i.e., the set this zone represents may contain only spiders with a foot in it).

Contours and zones are both outlined shapes representing sets. Contours are labelled with alphabetical letters. Zones are not labelled and represent intersections and complements of contours.

[1] For more detail see *Spider Diagrams* [7] by Howse et al, and *The expressiveness of spider diagrams augmented with constants* [11] by Stapleton et al.

Zones are defined as ordered pairs, say $Z = (\Gamma, \Delta)$. The first element of the pair, Γ, is a set of contours which contain the zone. The second element, here Δ, is a set of contours which **do not** contain the zone (the zone lies entirely outside of these). The set described by the zone (Γ, Δ) is defined as

$$\texttt{set\_of}(Z) = \bigcap_{A_i \in \Gamma} \texttt{set\_of}(A_i) \setminus \bigcup_{B_i \in \Delta} \texttt{set\_of}(B_i), \tag{1}$$

where $\texttt{set\_of}(A_i)$ is the set represented by countour A_i. For example, the spider diagram in Figure 1 contains 8 zones (note that zone $(\{\}, \{A, B, C\})$ lies outside all contours). However, not all zones have to be drawn. They may be omitted if they play no role in the statement.

Spiders represent single elements. Spiders are dots which may optionally be connected with a line. Dots are the *feet* of the spider and define its *habitat*. As an example, Figure 1 contains three spiders: spiders s_1 and s_2 reside in zone $(\{B\}, \{A, C\})$, spider s_3 resides in a *region* of three zones. Regions are collections of zones, with corresponding sets defined as follows:

$$\texttt{set\_of}(R) = \bigcup_{Z_i \in R} \texttt{set\_of}(Z_i) \tag{2}$$

The following is an illustration of the semantics of the diagram in Figure 1:

$$\begin{gathered}\exists s_1\, s_2\, s_3.\ \mathit{distinct}(s_1, s_2, s_3) \wedge (s_1 \in B \setminus A \cup C) \wedge (s_2 \in B \setminus A \cup C) \ \wedge \\ (s_3 \in (A \setminus C) \cup (A \cap B \cap C)) \wedge (B \setminus A \cup C) \subseteq \{s_1, s_2\}\end{gathered} \tag{3}$$

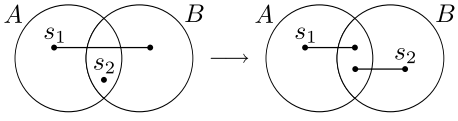

Fig. 2: A diagrammatic statement in the language of spider diagrams.

A *compound spider diagram* is a diagram that consists of spider diagrams, which are called *unitary spider diagrams*, coupled with logical connectives. Figure 2 is an example of a compound spider diagram. Formula 4 illustrates the semantics of the diagram in Figure 2:

$$\begin{gathered}\exists s_1\, s_2.\ \mathit{distinct}(s_1, s_2) \wedge (s_1 \in A \cup B \setminus A \cap B) \wedge (s_2 \in A \cap B) \\ \rightarrow \\ \exists s_1\, s_2.\ (s_1 \in A) \wedge (s_2 \in B)\end{gathered} \tag{4}$$

Note that spider names are local to unitary spider diagrams, whereas contour names are global.

Figure 3 shows a purely diagrammatic proof of the example in Figure 2. Note that the proof involves applications of three sound diagrammatic inference rules (from [7]): split spiders (removes lines connecting feet of a spider and creates a case-split for each foot), add feet (puts a new dot into a zone and connects it to an existing spider in a foreign region), and idempotency. Our goal is to enable mixing of these and other diagrammatic inference rules with sentential ones.

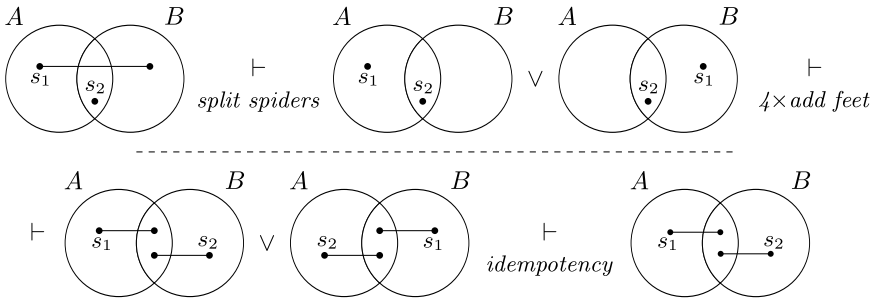

Fig. 3: A purely diagrammatic proof of the example in Figure 2.

3 Sentential Reasoner

Our first step was to provide a formalisation of the theory of spider diagrams within Isabelle/HOL (files available from `http://gitorious.net/speedith`). This not only makes the translation between the two representations easier, but also allows for direct proof reconstruction within Isabelle.

3.1 Formalisation of Diagrams in Isabelle/HOL

We formalise the basic elements of spider diagrams as follows:

Contours are identifiers of type `contour = nat` (natural numbers).

Zones are sets of contours (sets of natural numbers). Internally, zones are of the following type: `zone = contour set` (or equivalently `zone = nat set`).

Regions are sets of zones: `region = zone set` (or equivalently `region = (nat set) set`).

Spiders are identifiers of type `spider = nat`.

The interpretation of each of the above diagrammatic elements is provided by their corresponding *mapping functions*. These functions take the above identifiers and return sets (or elements – for spiders) that correspond to them. Figure 4 shows the map function for zones:

```
fun zmap :: "('e, 'a) SD_scheme ⇒ zone ⇒ 'e set" where
  "zmap_d cs = (⋂c ∈ cs. cmap_d c) - (⋃c ∈ (-cs). cmap_d c)"
```

Fig. 4: The definition of the zone map which maps a zone to its set.

We also provide proofs for relevant lemmas of the theory of abstract spider diagrams (from [7]), e.g.: disjointness of zones, additivity of the region map over unions, intersections and complements. In addition to these, we have also formalised the diagrammatic inference rules mentioned above (i.e., split spiders, add feet and idempotency). Figure 5 shows the formalised *add feet* rule.

```
lemma add_feet: "⟦ smap s ∈ rmap r; r ⊆ r' ⟧ ⟹ smap s ∈ rmap r'"
```

Fig. 5: The sentential equivalent of the *add feet* diagrammatic inference rule.

3.2 Translation

The lemma in Figure 6 is the sentential translation of the diagrammatic statement in Figure 2.

```
lemma: "(∃s s'. s ≠ s' ∧ smap s' ∈ rmap {{0}, {1}} ∧ smap s ∈ rmap {{ 0, 1 }}) ⟶
(∃s s'. s ≠ s' ∧ smap s ∈ rmap {{0}, {0, 1}} ∧ smap s' ∈ rmap {{1}, {0, 1}})"
```

Fig. 6: A sentential translation of the example in Figure 2.

Translation from diagrams to sentences currently generates n existentially quantified first-order variables, a conjunction of $\frac{n(n-1)}{2}$ inequalities and n set-inclusion predicates, where n is the number of spiders. Using the higher-order quantification provided in Isabelle/HOL, we can existentially quantify over a single set of spiders. With a single predicate, say `distinct`, we can also remove the inequalities and make translation to diagrams easier. We aim to translate as many MFOL formulae to diagrams as possible.

Heterogeneous proof automation, proof reconstruction, and translation of sentences to diagrammatic representation is work in progress.

4 Heterogeneous Integration

Ultimately, we want to enable formal graphical reasoning as is depicted in Figure 7. Sentential expressions are drawn as diagrams if the translation is feasible. More importantly, the external diagrammatic reasoner can be invoked in an interactive mode within the proof body like any other tactic in Isabelle. These tactics will invoke our diagrammatic reasoner, which in turn will return a proof trace for proof reconstruction.

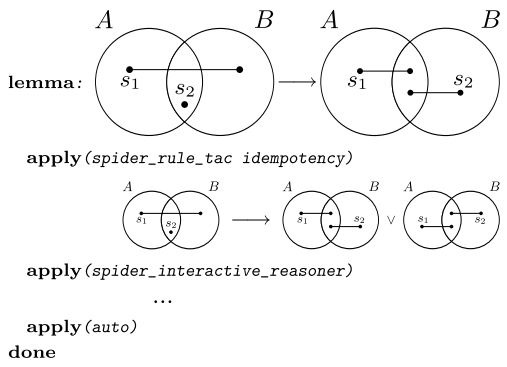

Fig. 7: A heterogeneous proof outline of the example in Figure 2.

We also want to enable visual and interactive diagram construction and application of diagrammatic inference steps. For this purpose we will use Speedith (sources available from `http://gitorious.net/speedith`) both as a standalone reasoning tool as well as a visual add-on to Isabelle's graphical user interfaces.

The user will be able to invoke the diagrammatic reasoner at any time during the proof, with an option to do so in an interactive or fully automated mode. Additionally, the currently active statement (lemma or proof obligation) will be automatically visualised as a diagram in an embedded window of the GUI.

Discussion. We outline a work-in-progress of an integration of a diagrammatic language with diagrammatic inference rules into a sentential theorem prover. This enables formal heterogeneous reasoning with mixed diagrams and sentences.

We provide a formalisation of spider diagrams with translation to sentential formulae. Heterogeneous proof automation, and a translation from first-order monadic formulae to spider diagrams is still work in progress.

The goal of this project is to provide a proof of concept heterogeneous reasoner – to show that heterogeneous reasoning is feasible. Ultimately, we plan to extend the heterogeneous framework to other domains with new diagrammatic systems (e.g.: constraint diagrams, UML, diagram chasing etc.).

In summary, we believe that heterogeneous reasoning can improve the ease of working with specific domains of verification in general purpose theorem provers.

Acknowledgments. We would like to thank Thomas Tuerk and Lawrence Paulson for invaluable input.

References

1. D. Barker-Plummer, J. Etchemendy, A. Liu, M. Murray, and N. Swoboda. Openproof - A Flexible Framework for Heterogeneous Reasoning. *LNCS*, 5223:347, 2008.
2. J. Barwise and J. Etchemendy. A Computational Architecture for Heterogeneous Reasoning. In *TARK*, pages 1–11. Morgan Kaufmann, 1998.
3. J. Gil, J. Howse, and S. Kent. Towards a Formalization of Constraint Diagrams. In *IEEE Symposia on Human-Centric Computing Languages and Environments*, page 72, 2001.
4. M. J. Gordon, A. J. Milner, and C. P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*, volume 78 of *LNCS*. Springer Berlin / Heidelberg, 1979.
5. E. Hammer. Reasoning with Sentences and Diagrams. *NDJFL*, 35(1):73–87, 1994.
6. J. Howse, F. Molina, J. Taylor, S. Kent, and J. Gil. Spider Diagrams: A Diagrammatic Reasoning System. *JVLC*, 12(3):299–324, 2001.
7. J. Howse, G. Stapleton, and J. Taylor. Spider Diagrams. *LMS JCM*, 8:145–194, 2005.
8. M. Jamnik, A. Bundy, and I. Green. On Automating Diagrammatic Proofs of Arithmetic Arguments. *JOLLI*, 8(3):297–321, 1999.
9. S.-J. Shin. Heterogeneous Reasoning and its Logic. *BSL*, 10(1):86–106, 2004.
10. G. Stapleton, J. Masthoff, J. Flower, A. Fish, and J. Southern. Automated Theorem Proving in Euler Diagram Systems. *JAR*, 39(4):431–470, 2007.
11. G. Stapleton, J. Taylor, S. Thompson, and J. Howse. The expressiveness of spider diagrams augmented with constants. *JVLC*, 20(1):30, 2009.
12. M. Wenzel, L. C. Paulson, and T. Nipkow. The Isabelle Framework. In *TPHOLs*, pages 33–38, 2008.
13. D. Winterstein, A. Bundy, and C. A. Gurr. Dr.Doodle: A Diagrammatic Theorem Prover. In *IJCAR*, pages 331–335, 2004.