DAGSTUHL
REPORTS

Volume 7, Issue 8, August 2017

Scalable Set Visualizations (Dagstuhl Seminar 17332)
Yifan Hu, Luana Micallef, Martin Nöllenburg, and Peter Rodgers 1

Computational Counting (Dagstuhl Seminar 17341)
Ivona Bezáková, Leslie Ann Goldberg, and Mark R. Jerrum 23

SLEBOK: The Software Language Engineering Body of Knowledge (Dagstuhl Seminar 17342)
Benoît Combemale, Ralf Lämmel, and Eric Van Wyk 45

Machine Learning and Formal Methods (Dagstuhl Seminar 17351)
Sanjit A. Seshia, Xiaojin (Jerry) Zhu, Andreas Krause and Susmit Jha 55

Analysis and Synthesis of Floating-point Programs (Dagstuhl Seminar 17352)
Eva Darulova, Alastair F. Donaldson, Zvonimir Rakamarić, and Cindy Rubio-González 74

ISSN 2192-5283

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at
http://www.dagstuhl.de/dagpub/2192-5283

Publication date
March, 2018

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at `http://dnb.d-nb.de`.

Digital Object Identifier: 10.4230/DagRep.7.8.i

Aims and Scope
The periodical *Dagstuhl Reports* documents the program and the results of Dagstuhl Seminars and Dagstuhl Perspectives Workshops.
In principal, for each Dagstuhl Seminar or Dagstuhl Perspectives Workshop a report is published that contains the following:
- an executive summary of the seminar program and the fundamental results,
- an overview of the talks given during the seminar (summarized as talk abstracts), and
- summaries from working groups (if applicable).

This basic framework can be extended by suitable contributions that are related to the program of the seminar, e. g. summaries from panel discussions or open problem sessions.

Report from Dagstuhl Seminar 17332

Scalable Set Visualizations

Edited by

Yifan Hu[1], Luana Micallef[2], Martin Nöllenburg[3], and Peter Rodgers[4]

1 **Yahoo! Research – New York, US,** yifanhu@yahoo-inc.com
2 **Aalto University, FI,** luana.micallef@aalto.fi
3 **TU Wien, AT,** noellenburg@ac.tuwien.ac.at
4 **University of Kent – Canterbury, GB,** p.j.rodgers@kent.ac.uk

Abstract

This report documents the program and outcomes of Dagstuhl Seminar 17332 "Scalable Set Visualizations", which took place August 14–18, 2017. The interdisciplinary seminar brought together 26 researchers from different areas in computer science and beyond such as information visualization, human-computer interaction, graph drawing, algorithms, machine learning, geography, and life sciences. During the seminar we had five invited overview talks on different aspects of set visualizations as well as a few ad-hoc presentations of ongoing work. The abstracts of these talks are contained in this report. Furthermore, we formed five working groups, each of them discussing intensively about a selected open research problem that was proposed by the seminar participants in an open problem session. The second part of this report contains summaries of the groups' findings.

Seminar August 13–18, 2017 – http://www.dagstuhl.de/17332
1998 ACM Subject Classification Human-centered computing → Visualization, Theory of computation → Design and analysis of algorithms
Keywords and phrases algorithms, information visualization, scalability, set visualization, visual analytics
Digital Object Identifier 10.4230/DagRep.7.8.1
Edited in cooperation with Tamara Mchedlidze

1 Executive Summary

Yifan Hu
Luana Micallef
Martin Nöllenburg
Peter Rodgers

Sets are a fundamental way of organizing information. Visualizing set-based data is crucial in gaining understanding of it as the human perceptual system is an analytic system of enormous power. The number of different set visualization methods has increased rapidly in recent years, and they vary widely in the visual metaphors used and set-related tasks that they support. Large volume set-based data can now be found in diverse application areas such as social networks, biosciences and security analysis. At present, set visualization methods lack the facility to provide analysts with the visual tools needed to successfully interpret large-scale set data as the scalability of existing metaphors and methods is limited.

This seminar provided a forum for set visualization researchers and application users to discuss how this challenge could be addressed.

Existing set visualizations can be grouped into several families of techniques, including traditional Euler and Venn diagrams, but also node-link diagrams, map- and overlay-based representations, or matrix-based visualizations. Inevitably, the approaches taken to drawing these visualizations are diverse, for example node-link diagrams require graph drawing methods, whereas overlay techniques use algorithms from computational geometry. However, they are similar in a number of aspects. One aspect is the underlying set theory. For instance, theoretical results into the drawability of many of these set visualization techniques for different data characteristics is possible (as already done in example Venn and Euler diagram research). Another common aspect is that the visualizations are typically focused on an end-user, so perceptual, cognitive and evaluation considerations are an important concern.

A particularly pressing issue in set visualization is that of scaling representations. The number of data items can be large and many methods aggregate individual items. Yet, even using aggregation, the limit of the most scalable of these methods is considered to be in the region of 100 sets [1]. Typical application areas that make use of sets include, e.g., social networks, biosciences and security analysis. In these applications, there may be many millions of data items in thousands of sets. Other applications have high-dimensional data, where each item is associated with a large number of variables, which poses different scalability challenges for set visualizations.

A distinct feature of set visualization is that visualizations must support set-related, element-related, and attribute-related analysis tasks [1] that involve, e.g., visually evaluating containment relations, cardinalities, unions, intersections, or set differences. For example, bioscience microarray experiments classify large numbers of genes and multiple visualization tools have been developed to visualize this data. However, current efforts can only visualize small sections of the information at once [2]. Similar scalability challenges for set visualizations appear in many other applications as well. Hence, developing effective visualization methods for large set-based data would greatly facilitate analysis of such data in a number of important application areas.

Seminar Goals

The goal of this seminar was to bring together researchers with different backgrounds but a shared interest in set visualization. It involved computer scientists with expertise, e.g., in visualization, algorithms, and human-computer interaction, but also users of set visualizations from domains outside computer science. Despite the large number of set visualization techniques, for which there is often a considerable practical and theoretical understanding of their capabilities, there has only been limited success in scaling these methods. Thus the intended focus of this seminar was to discuss and study specific research challenges for scalable set visualizations concerning fundamental theory, algorithms, evaluation, applications, and users. We started with a few overview talks on the state of the art in set visualization, but then focused on small hands-on working groups during most of the seminar week. We aimed to accelerate the efforts to improve scalability of set visualizations by addressing open questions proposed by the seminar attendees, in order to produce concrete research outcomes, including new set visualization software and peer-reviewed research publications.

Seminar Program

1. On the first two days of the seminar we enjoyed five invited overview lectures on different aspects of set visualizations. The topics and speakers were chosen as to create a joint understanding of the state of the art of set visualization techniques, evaluations and applications. Silvia Miksch gave a systematic overview of set visualization techniques, grouped by types of visual representations and tasks, with a special focus on set visual analytics. Martin Krzywinski reported about his experiences on using visual analogies for showing set-based data in the area of genomics. Sara Fabrikant took a cartographer's view on visualizing sets and explained how successful cartographic maps work as information displays by taking not only the design but also the context and the user into account. Stephen Kobourov explained how large graph-based set data can be represented using a familiar map metaphor by showing several interesting data sets and their map representations. Finally, John Howse presented how set visualizations can be used as diagrammatic reasoning systems in logic.
2. In the open problem session on the first day of the seminar we collected a list of 13 open research problems that were contributed by the seminar participants. In a preference voting we determined the five topics that raised the most interest among the participants and formed small working groups around them. During the following days the groups worked by themselves, except for a few plenary reporting sessions, formalizing and solving their respective theoretical and practical challenges. Below is a list of the working group topics; more detailed group reports are found in Section 4.
   a. **Mapifying the genome:** Can the axis of the entire genome be mapped on a 2-dimensional space based on gene function rather than a 1-dimensional line based on gene position?
   b. **Area-proportional Euler diagrams with ellipses:** Can the use of ellipses extend the size of data that can be drawn with area-proportional Euler diagrams?
   c. **Spatially informative set visualizations:** Can we improve spatial overlay-based set visualizations when allowing some limited displacement of the given set positions?
   d. **Set visualization using the metro map metaphor:** How and under which conditions can the metro map metaphor be used to visualize set systems?
   e. **Visual analytics of sets/set-typed data and time: challenges and opportunities:** What are the main research challenges and opportunities in the context of set visualizations that change over time and how can these be structured?
3. We had a flexible working schedule with a short plenary session every morning to accommodate group reports and impromptu presentations by participants. In two of those Wouter Meulemans and Nan Cao shared recent results of theirs related to set visualization.
4. During the week we encouraged participants to come up with suggestions for further strengthening this growing community of set visualization researchers. In a plenary session on Friday we collected and structured these ideas and made started planning future events related to set visualizations, see Section 1.

Future Plans

During the entire seminar, participants actively discussed ways how to disseminate, proliferate and promote scalable set visualization research in diverse specific areas, such as: set theory and diagrammatic reasoning; algorithms and graph theory; information visualization and visual analytics; evaluation, users and application areas. This led to the concretization of the following future milestones, each of which is being coordinated by volunteered seminar participants:

- **Diagrams Workshop in 2018** on Set Visualization and Reasoning (SetVR)
  The workshop aims at promoting set visualization to the Diagrams community, of which well-renowned mathematicians and logicians are members, thus proliferating relevant set theory and diagrammatic reasoning research;
- **IEEE VIS Workshop in 2019** on Set Visualization and Analytics (SetVA)
  The workshop aims at promoting set visualization to the Information Visuaization and Visual Analytics communities, at the premier forum for advances in information and scientific visualization, with the aim to generate new visualization and analytic techniques to handle large set-typed data;
- **Dagstuhl seminar in 2019** on Set Visualization and Analytics (SetVA) over Time and Space
  This seminar has revealed, for the first time, the need for visualization and analytic techniques for the set-typed data that has an element of time and/or space; thus a follow-up Dagstuhl seminar will be organized to discuss this topic, once again among researchers with diverse set visualization backgrounds;
- **Set Visualization Workshop in 2020** in the Computational Geometry Week or collocated with Graph Drawing
  This workshop aims at disseminating set visualization to a more algorithmic and computational geometry research community, to ensure the production of effective, yet efficient and scalable set visualization algorithms;
- **Set Visualization browser**, like http://setviz.net
  This browser will collect and disseminate available set visualization techniques, making them easily accessible through various categorizations, such as the type of data analysis tasks or application areas they target;
- **Set Visualization book**
  The book would serve as a guide for researchers who are new to set visualization and as a review of the current state-of-the-art of set-typed data in the various related domains.

We decided to have an annual set visualization workshop that each year focuses on one of (i) diagrammatic reasoning and logic, (ii) information visualization and visual analytics, and (iii) computational geometry and graph drawing, at premier venues of the respective research communities, to generate further research interest in all of these three diverse areas that are all important for scalable set visualizations.

Evaluation

According to the Dagstuhl survey conducted after the seminar, as well as informal feedback to the organizers, the seminar was highly appreciated. Particularly the small group size, group composition, and the seminar structure focusing on hands-on working groups was very well received. The seminar's goals to identify and initiate collaboration on new research challenges was very successful (also in comparison to other Dagstuhl seminars) as the participants rated the seminar highly for inspiring new research directions, joint projects and joint publications. We are looking forward to seeing the first scientific outcomes of the seminar in the near future and to continuing the efforts to support the growth of the set visualization community.

Acknowledgments

Schloss Dagstuhl was the perfect place for hosting a seminar like this. The unique scientific atmosphere and the historic building provided not only all the room we needed for our program and the working groups, but also plenty of opportunities for continued discussions and socializing outside the official program. On behalf of all participants the organizers want to express their deep gratitude to the entire Dagstuhl staff for their outstanding support and service accompanying this seminar. We further thank Tamara Mchedlidze for helping us collecting the contributions and preparing this report.

References

1 Bilal Alsallakh, Luana Micallef, Wolfgang Aigner, Helwig Hauser, Silvia Miksch, and Peter Rodgers. The State-of-the-Art of Set Visualization. *Computer Graphics Forum*, 35(1):234–260, 2015.
2 Sebastian Behrens and Hans A Kestler. Using VennMaster to evaluate and analyse shRNA data. *Ulmer Informatik-Berichte*, page 8, 2013.

2 Contents

Executive Summary
Yifan Hu, Luana Micallef, Martin Nöllenburg, and Peter Rodgers 1

Overview of Talks

SetVA: Visual Analytics of Sets and Set-Typed Data: Challenges and Opportunities
Silvia Miksch 7

Visual analogies and explanations
Martin Krzywinski 7

The Visualization of Sets: A Cartographer's View
Sara Irina Fabrikant 8

Set Visualization with Maps
Stephen G. Kobourov 8

Diagrams for Logic and Reasoning
John Howse 9

The Painters Problem
Wouter Meulemans 9

UnTangle Map: Visual Analysis of Probabilistic Multi-Label Data
Nan Cao 10

Working Groups

Mapifying the Genome
Radu Jianu, Martin Krzywinski, Luana Micallef, and Hsiang-Yun Wu 11

Euler Diagrams drawn with Ellipses Area Proportionally (EDEAP)
Fadi Dib, Peter Rodgers, Michael Wybrow 14

Spatially Informative Set Visualization
Thom Castermans, Mereke van Garderen, Wouter Meulemans, Martin Nöllenburg, and Xiaoru Yuan 15

Set Visualization Using the Metro Map Metaphor
Robert Baker, Nan Cao, Yifan Hu, Michael Kaufmann, Stephen Kobourov, Tamara Mchedlidze, Sergey Pupyrev, Alexander Wolff 18

Visual Analytics of Sets/Set-Typed Data and Time: Challenges and Opportunities
Daniel Archambault, Kerstin Bunte, Sara Irina Fabrikant, John Howse, Andreas Kerren, and Silvia Miksch 20

Participants 22

3 Overview of Talks

3.1 SetVA: Visual Analytics of Sets and Set-Typed Data: Challenges and Opportunities

Silvia Miksch (TU Wien, AT)

Joint work of Bilal Alsallakh, Luana Micallef, Wolfgang Aigner, Helwig Hauser, Silvia Miksch, Peter Rodgers
Main reference Bilal Alsallakh, Luana Micallef, Wolfgang Aigner, Helwig Hauser, Silvia Miksch, Peter J. Rodgers: "The State-of-the-Art of Set Visualization", Comput. Graph. Forum, Vol. 35(1), pp. 234–260, 2016.
URL http://dx.doi.org/10.1111/cgf.12722

Sets comprise a generic data model that has been used in a variety of data analysis approaches. Such approaches involve analyzing and visualizing set relations between multiple sets defined over the same collection of elements. However, visualization / visual analytics of sets is a non-trivial problem due to the large number of possible relations between them. We provide a systematic overview of state-of-the-art techniques. We classify these techniques into six main categories according to the visual representations they use and the tasks they support. We compare the categories to provide guidance for choosing an appropriate technique for a given problem. Finally, we identify challenges and opportunities in this area and propose possible research directions. The most important challenge – in my point of view – is Visual Analytics of set systems over time, called "SetVA over Time". Further resources on set visualization are available at http://www.setviz.net.

3.2 Visual analogies and explanations

Martin Krzywinski (BC Cancer Research Center, Vancouver, CA)

"The great tragedy of science–the slaying of a beautiful hypothesis by an ugly fact." wrote Huxley, in a statement that is as much about how science works as about the irrepressible optimism required to practice it. But even greater is the tragedy of obfuscating facts with impenetrable figures and demoting their natural beauty by florid visuals. The issue isn't one of pure aesthetics-lack of clarity, precision and conciseness in science communication slows our efforts to move forward. In the field of disease research, this has fateful impact on lives.

Through a series of examples from the field of genomics, I show examples of how visual analogies can fail and succeed. I point out that we must not only get the foundations right – choice of color, shape and encoding – but also carefully attend to organizing flow and continuity. The latter become particularly important for complex multi-panel figures.

Many of the kinds of data generated in genomics can be considered as sets. One example for immediate concern is the overlap between specific genomic mutations and cancer types. I argue that for large data sets, such as the recently reported inventory of mutations in 10,000 sequenced tumor genomes [1], there is value in using simple and familiar forms and resist the urge to create new and complex encodings when a series of standard ones will suffice.

References

1 A. Zehir et al. *Mutational landscape of metastatic cancer revealed from prospective clinical sequencing of 10,000 patients.* Nature Medicine 23, 703–713 (2017).

3.3 The Visualization of Sets: A Cartographer's View

Sara Irina Fabrikant (Universität Zürich, CH)

Joint work of Members of the UZH Geographic Information Visualization Group (GIVA) and students. See URL below.
Main reference Fabrikant, S.I. (2016). Evidence-based Design. Invited talk at The Cartographic Summit. The Future of Mapping. Hosted by the International Cartographic Association (ICA) and Esri Inc., Redlands (CA), Feb. 9, 2016.
http://videoembed.esri.com/iframe/4937/000000/width/960/0/00:00:00
URL http://www.geo.uzh.ch/en/units/giva/research.html

A key research question for the cognitive cartographer is "How do (Geo)graphic Information Displays work?" Better yet: how do we need to design GID's to support effective, efficient, and affective inference and decision making, and ultimately spatial behavior? It turns out that display DESIGN is not the most important factor. In an increasingly mobile information society, the use CONTEXT of the display is equally important. This, aside from the key factor: human USER. The design of cognitively supportive (geo)graphic information displays must include/consider the decision maker (i.e., individual and group differences including their spatial abilities, perception/cognitive capacities, background, training, etc.) as well as the use context of the display (i.e., uncertainty, time pressure, other users, environmental factors, etc.).

References

1 Kübler, I., Richter, K.-F., Fabrikant, S.I. *Visualization of risk and uncertainty to communicate avalanche hazard: An Empirical Study.*. Proceedings, 28th International Cartographic Conference, International Cartographic Association, Jul. 2-7, 2017, Washington, D.C. USA. 2017.

2 Griffin, A.L., White, T. , Fish, C., Tomio, B., Huang, H., Robbi Sluter, C., Meza Bravo, J.V., Fabrikant, S.I., Bleisch, S., Yamada, M., Picanço, Jr., P., Jr. *Visualization of risk and uncertainty to communicate avalanche hazard.* International Journal of Cartography, DOI: 10.1080/23729333.2017.1315988.

3.4 Set Visualization with Maps

Stephen G. Kobourov (University of Arizona – Tucson, US)

Joint work of Yifan Hu, Sergey Pupyrev
Main reference Yifan Hu, Emden R. Gansner, Stephen G. Kobourov: "Visualizing Graphs and Clusters as Maps", IEEE Computer Graphics and Applications, Vol. 30(6), pp. 54–66, 2010.
URL http://dx.doi.org/10.1109/MCG.2010.101

Visualization techniques help us understand and analyze complicated datasets and allows us to perceive relationships, patterns, and trends. While statistical techniques may determine correlations among the data, visualization helps us frame what questions to ask. Many interesting data sets can be visualized as graphs: the vertices in the graph are the objects of interest (for example, researchers) and a link between two vertices in the graph indicates a relationship (for example, research collaboration). Graph visualization aims to present such data in an effective and aesthetically appealing way. We describe ways to visualize datasets with the help of conceptual maps as a representation metaphor. While graphs often require considerable effort to comprehend, a map representation is more intuitive, as most people are familiar with maps and ways to interact with them via zooming and panning. We consider map representations of several interesting datasets: from movies on Netflix, to books on music on Last.fm, to maps of computer science.

3.5 Diagrams for Logic and Reasoning

John Howse (University of Brighton, GB)

Traditional diagrammatic representations of set-based systems include Venn diagrams and Euler Diagrams. Euler diagrams are well-matched to meaning and contain free rides where some derivations are readable from the diagram. Spider diagrams extend Euler diagrams to include the representation of elements. Sound and complete spider diagram reasoning systems have been developed, with expressiveness equivalent to first order monadic logic. More expressive notations have been developed including constraint diagrams, for software system specification, and concept diagrams, for ontology representation. These notations include the representation of relations. Finally, multi-diagrams can express information that would become cluttered when represented in a single diagram. An open question is when is it appropriate to represent information with multi-diagrams rather than single diagrams.

3.6 The Painters Problem

Wouter Meulemans (TU Eindhoven, NL)

Joint work of Arthur van Goethem, Irina Kostitsyna, Marc J. van Kreveld, Wouter Meulemans, Max Sondag and Jules Wulms
Main reference Arthur van Goethem, Irina Kostitsyna, Marc J. van Kreveld, Wouter Meulemans, Max Sondag, Jules Wulms: "The Painter's Problem: covering a grid with colored connected polygons", CoRR, Vol. abs/1709.00001, 2017.
URL http://arxiv.org/abs/1709.00001

Small multiples are a powerful visualization technique exploiting juxtaposition for concurrently displaying data under different conditions. Set visualization is characterized often by its containers that link together the lements belong to the same set. The typical Euler diagram is well known and intuitive, but suffers from drawback such as color perception of blended semi-transparent colored overlays.

With the eventual goal of a set-visualization technique that uses a disjoint representation and combines well with small multiples, we study the following problem:

Given a grid of square cells (see Figure 1, left), where each is either "red", "blue", "purple" (both red and blue) or "white" (neither), can we partition each purple cell into red and blue pieces, such that the union of all red cells and red pieces is connected, and likewise for blue? (refer to Figure 1, right)

We provide a characterization of instances that admit such a partition, allowing for an efficient algorithmic test. Moreover, if a partition is possible, we bound the maximal number of pieces needed for a single purple cell to five. If there are no white cells present, we can even improve this bound to two. That is, we can then partition each purple cell into a single red and a single blue piece, such that the result meets the connectivity constraint.

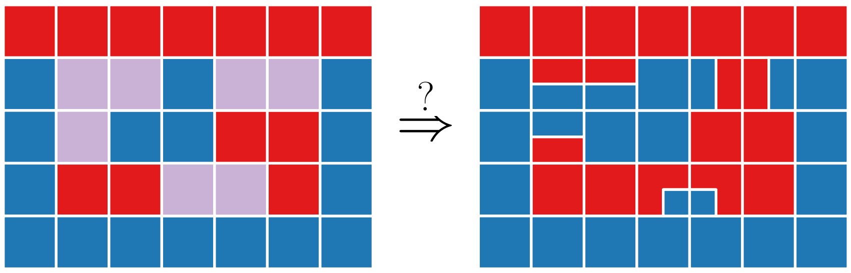

Figure 1 Illustration of the solution

3.7 UnTangle Map: Visual Analysis of Probabilistic Multi-Label Data

Nan Cao (Tongji University, CN)

Data with multiple probabilistic labels are common in many situations. For example, a movie may be associated with multiple genres with different levels of confidence. Despite their ubiquity, the problem of visualizing probabilistic labels has not been adequately addressed. Existing approaches often either discard the probabilistic information, or map the data to a low-dimensional subspace where their associations with original labels are obscured. In this talk, we introduce a novel visual technique, UnTangle Map, for visualizing probabilistic multi-labels. In our proposed visualization, data items are placed inside a web of connected triangles, with labels assigned to the triangle vertices such that nearby labels are more relevant to each other. The positions of the data items are determined based on the probabilistic associations between items and labels. UnTangle Map provides both (a) an automatic label placement algorithm, and (b) adaptive interactions that allow users to control the label positioning for different information needs. Our work makes a unique contribution by providing an effective way to investigate the relationship between data items and their probabilistic labels, as well as the relationships among labels. Our user study suggests that the visualization effectively helps users discover emergent patterns and compare the nuances of probabilistic information in the data labels.

4 Working Groups

4.1 Mapifying the Genome

Radu Jianu (City, University of London, GB), Martin Krzywinski (BC Cancer Research Centre – Vancouver, CA), Luana Micallef (Aalto University, FI), and Hsiang-Yun Wu (TU Wien, AT)

The human genome is a 1-dimensional structure of approximately 3 billion bases arranged across 23 pairs of chromosomes. Two challenges arise when attempting to visualize whole-genome data using the common and traditional genomic position axis, as in Figure 2.A. These issues are due to the fact that the data often annotate the small parts of the genome that code for proteins (exons), whose position on the genome is largely a byproduct of random shuffling during evolution and does not directly relate to the function of the genes (thus the reason why the position of the genes in a human genome is different from that of a mouse, as shown in Figure 2.B).

First, only about 2-3% codes for proteins, so the regions in which data appear are very sparse. This imposes a limit on the visibility of high-resolution elements in figures and visualizations, without smart down-sampling many of the data bins across regions of interest are smaller than a pixel or beyond visual acuity. Second, adjacency of genes along the genomic position axis does not generally relate to similar function or pathway. For example, the two genes that are involved the metabolism of trytophan into melatonin are TPH, which is on chromosome 11 and ASMT, which is on chromosome X. The position of these genes is very difficult to guess on whole-genome plots and any correlation in values is essentially impossible to follow because the gene's positions are so far apart. In some cases the use of a position axis is helpful, such as whole-genome displays that show copy number variation, often shown in a format similar to Figure 2.C, which can quickly show large-scale structural changes (e.g. loss or gain of a chromosome arm) but cannot communicate the functional consequences of these changes.

We propose to address these two issues by remapping the axis of the genome from one based on position to one based on function. This would have the effect of creating a fixed coordinate system in which parts would be associated with function (e.g. cellular membrane, cell cycle, apoptosis, etc), thereby satisfying the Gestalt grouping principle of proximity, which states that elements that are positioned close to one another are perceived as related semantically. We argue for the need for a reordering that can be used in 1-dimensional data encodings (e.g. along a line) and 2-dimensional (similar to a map). This kind of remapping is commonly applied to gene-centric visualizations of the presence of mutations (or other structural or functional disturbance) (Figure 3), in which tens of genes are grouped by function. Our proposal extends this to the full genome.

Our approach is as follows. We would first define a similarity metric between genes based on their functional similarity, as defined in the Gene Ontology [2] or another resource. Multiple criteria for similarity can be used and possibly tuned to specialize applications, in which some groupings are more relevant than others. Second, we would apply clustering and layout algorithms (e.g., spring embedder, stochastic neighborhood embedding) to derive a 2-dimensional layout of the genes. Our expectation would be that genes that are close in this layout are functionally similar. This layout would be reshaped to fit into a rectangular area suitable as a canvas on which data can be drawn, in a similar way as already done for

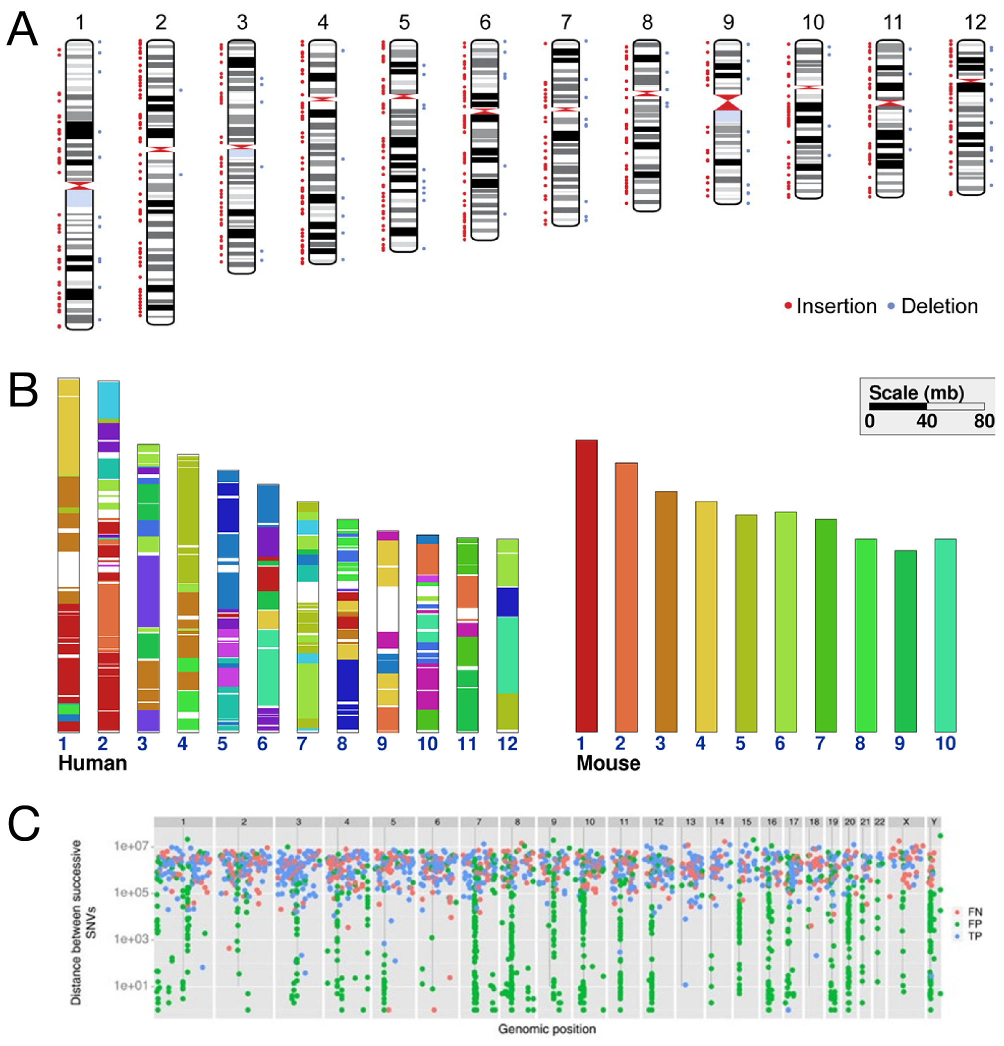

Figure 2 The human genome is 3 billion bases long. The longest chromosome (chr1) is about 250 million bases. Drawing data at single-base resolution is essentially impossible – the smallest visually discernable element in a figure covers about 250,000 bases. (A) The exact position of features such as mutations (insertions, deletions, SNPs) is impossible to assess. We cannot say which genes are affected by these mutions and, more importantly, what cellular function may be impacted by these changes. Figures like this help in understand the overall numbers and proportions of these changes but do not communicate the consequences of these changes. Adapted from Figure 3 in [6]. (B) The position of genes on the genome is driven by shuffling of genetic material within and among chromosomes during evolution. Shown here are regions of sequence similarity between human and mouse genomes. Source: [4]–Figure 2 (C) Features in data across an entire genome are very difficult to assess. Figures like this can be helpful to assess very large structural changes – when large contiguous parts of the genome are affected (e.g. deletion of a large region, or a chromosomal arm). Source: [1].

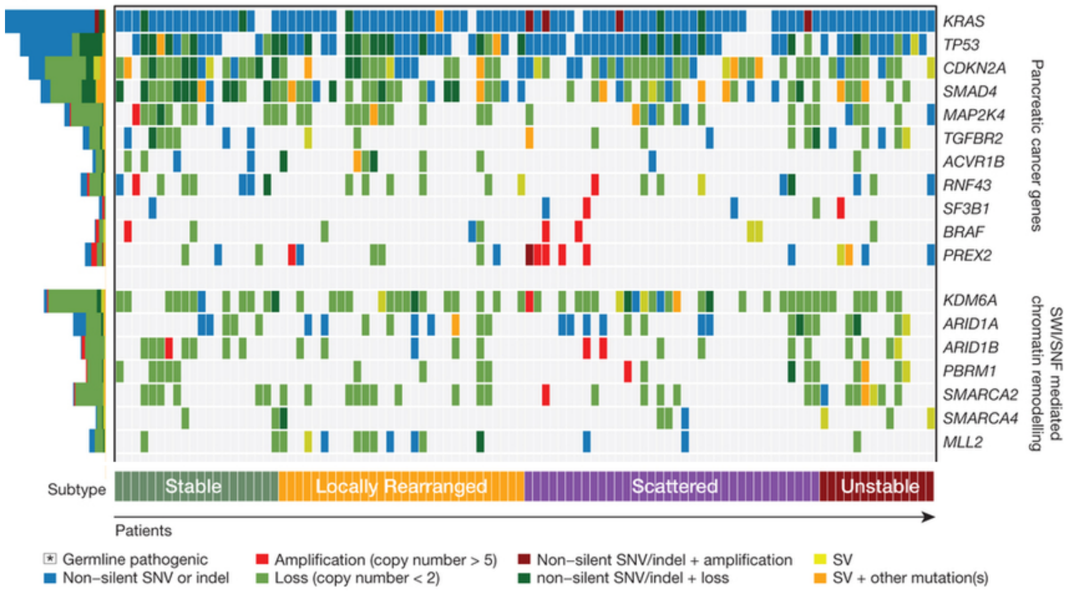

Figure 3 The idea of ordering genes by function is commonly applied when displaying information for small sets of genes that are recurrently mutated in disease. Here genes are grouped as "pancreatic cancer genes", "hromatin remodelling", "DNA damage report", "axon guidance" and "known oncogenes". Our proposal would extend this concept across the entire genome with gene order either globally fixed or based on specific applications. Source: [5].

example for graphs [3]. Third, using this 2-dimensional arrangement, we would derive a 1-dimensional order by choosing a path through the genes in 2-d (space filling, TSP, etc). This step would create an axis that would unfold the 2-d arrangement for linearized display.

This approach can be extended to generate a mapping between every base in the genome and a functionally correlated 2-dimensional position. Instead of laying out genes, their neighboring regulatory elements and inter-genetic regions would also be placed around the gene's positions so that data that might relate to the gene's activity would be drawn next to the gene. Regions of the genome very distant from genes (e.g. > 1 Mb away) that could not be unambiguously mapped to a function could be relegated into a separate part of the display. The output of this proposal would be coordinate transform file. Initially, it would remap intervals in the genome that correspond to genes to a point (or region) in a 2-dimensional rectangle and a 1-dimensional line. Subsequently, the file would include more parts of the genome, such as the regulatory regions and inter-genetic regions.

References

1 Tyler S Alioto, Ivo Buchhalter, Sophia Derdak, Barbara Hutter, Matthew D Eldridge, Eivind Hovig, Lawrence E Heisler, Timothy A Beck, Jared T Simpson, Laurie Tonon, et al. A comprehensive assessment of somatic mutation detection in cancer using whole-genome sequencing. *Nature communications*, 6:10001, 2015.

2 Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25, 2000.

3 Emden R Gansner, Yifan Hu, and Stephen Kobourov. Gmap: Visualizing graphs and clusters as maps. In *Visualization Symposium (PacificVis), 2010 IEEE Pacific*, pages 201–208. IEEE, 2010.

4 Amit U Sinha and Jaroslaw Meller. Cinteny: flexible analysis and visualization of synteny and genome rearrangements in multiple organisms. *BMC bioinformatics*, 8(1):82, 2007.

5 Nicola Waddell, Marina Pajic, Ann-Marie Patch, David K Chang, Karin S Kassahn, Peter Bailey, Amber L Johns, David Miller, Katia Nones, Kelly Quek, et al. Whole genomes redefine the mutational landscape of pancreatic cancer. *Nature*, 518(7540):495, 2015.

6 Jinchuan Xing, Yuhua Zhang, Kyudong Han, Abdel Halim Salem, Shurjo K. Sen, Chad D. Huff, Qiong Zhou, Ewen F. Kirkness, Samuel Levy, Mark A. Batzer, and Lynn B. Jorde. Mobile elements create structural variation: Analysis of a complete human genome. *Genome Research*, 19(9):1516–1526, 2009.

4.2 Euler Diagrams drawn with Ellipses Area Proportionally (EDEAP)

Fadi Dib (Gulf University for Science&Technology – Mishreff, KW), Peter Rodgers (University of Kent – Canterbury, GB), Michael Wybrow (Monash University – Caulfield, AU)

This working group addressed the research question "Can we draw general area-proportional Euler diagrams with ellipses?". The goal was to increase the number of sets that can be drawn using Ellipses, from the current state of the art, which is limited to 3 [1]. We also aim to improve on the performance of techniques that use circles [3]. The target was to develop a web based software system that can accurately layout a larger number of diagrams compared to these previous techniques. This would enable those wishing to visualize such diagrams to access a more effective tool.

The motivation behind this work is the demand for area proportional diagrams to illustrate data in areas such as medicine, biosciences and numerous other disciplines. The goal is to ensure that the areas of overlapping regions are directly proportional to the cardinality in the data. The output of current tools is often inaccurate, in particular the circles used in current tools are known to be inaccurate for three sets forming Venn-3 [2]. Such a failure at a small set size is multiplied as the number of sets increase. Fortunately, it is known that far more accurate diagrams can be drawn when ellipses are used in place of circles, accuracy rates well over 90% can be achieved for Venn-3 diagrams [1].

In order to achieve the goals, a JavaScript software system was started during the seminar, running a basic hill climber, and limited number of criteria merged into a weighted sum to form a fitness function. This result from the seminar demonstrated promise that, with refinement, the tool could meet the required goals.

Since the seminar, the group has regularly met remotely to refine the system. Much effort has gone into improving the area measurements, the calculation of fitness components, the testing framework and speed of optimization. However, before release (and write up as a research paper) the following activities need to be completed:

- Identification and Implementation of additional criteria. Local optima occur when ellipses that should intersect are entirely contained in on another, or are entirely separate. This leads to situations where there are no incremental moves to ensure the ellipses overlap as required.
- Testing of various optimizers. The current hill climber is viewed as overly simplistic and a more effective optimizing algorithm will be implemented, perhaps based on simulated annealing.

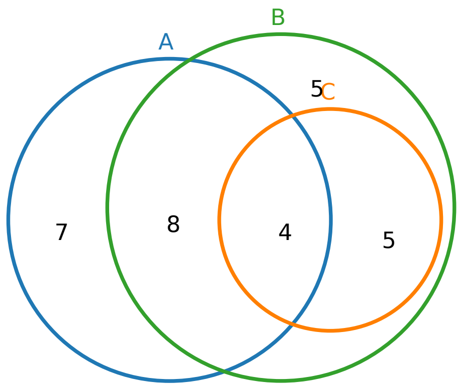

Figure 4 Output from an early version of the EDEAP software.

- The testing framework needs improvement. The weights for the criteria must be set (in line with both objective data and subjective view of what a good diagram might look like). Of particular note is that the result must be compared against the current state-of-the-art, the software of [3].

References

1 Luana Micallef and Peter Rodgers. eulerape: Drawing area-proportional 3-venn diagrams using ellipses. *PLOS ONE*, 9(7):1–18, 07 2014. URL: http://www.eulerdiagrams.org/eulerAPE, `doi:10.1371/journal.pone.0101717`.
2 P. Rodgers, G. Stapleton, J. Flower, and J. Howse. Drawing area-proportional euler diagrams representing up to three sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(1):1–1, Jan 2014. `doi:10.1109/TVCG.2013.104`.
3 L. Wilkinson. Exact and approximate area-proportional circular venn and euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 18(2):321–331, Feb 2012. `doi:10.1109/TVCG.2011.56`.

4.3 Spatially Informative Set Visualization

Thom Castermans (TU Eindhoven, NL), Mereke van Garderen (Universität Konstanz, DE), Wouter Meulemans (TU Eindhoven, NL), Martin Nöllenburg (TU Wien, AT), and Xiaoru Yuan (Peking University, CN)

Set visualization for elements with (geo)spatial locations is often dominated by the elements being shown in exact spatial locations. We set out to investigate methods where we do not keep elements at their exact location, but rather let them move away from it in order to further clarify the set visualization. However, elements are not allowed to move arbitrarily: rather, they must remain near their spatial location, to allow high-level spatial patterns to remain visible. Since spatial accuracy is not perfect, but not completely lost either, we refer to this as *spatially informative set visualization.*

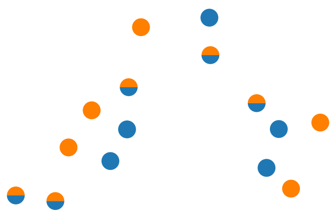
Figure 5 Point set of 15 points, each specifying one or two colors.

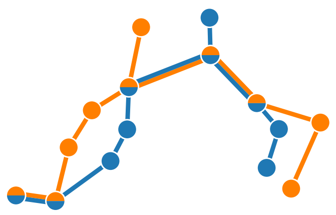
Figure 6 The shortest plane support for the point set in Figure 5.

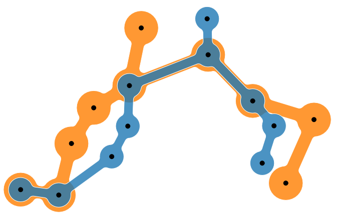
Figure 7 A Kelp-style rendering of the support shown in Figure 6.

Definitions. The first problem we faced was to define what it means to "clarify" a set visualization. We formalize set visualization using the existing concept of a *support graph*: a graph G on the elements is called a support of a set system, if each set induces a connected subgraph in G. Using such a support, we can now define what a good set visualization is, by setting requirements on the support graph. In particular, we set the following goals for a good support G: (a) G should be plane, that is, have straight-line edges without intersections; (b) G should be short, that is, the sum of edge lengths is small.

We would like to study how much the support improves if we are allowed to move points. However, little is known about finding good supports even with fixed elements. We thus first defined and studied the problem below (refer to Figures 5 and 6 for an illustration). Here we use color to indicate set memberships, as is common in research involving hypergraph supports.

SHORTESTPLANESUPPORT: *Consider a set P of points in $\mathbb{R}^2$ and a set C of colors. A function $\chi\colon P \to \mathcal{P}(C)\setminus\{\emptyset\}$ maps each point in P to a nonempty subset of the colors in C. Find a plane graph G with vertex set P such that G has minimal total length and G is a support: for each color $c \in C$, the subgraph induced by $\{p \in P \mid c \in \chi(p)\}$ is connected.*

We use PLANESUPPORT to refer to the variant where the length of G is not considered and we just want to decide whether such a graph G exists at all. We call a point p a k-colored point if $|\chi(p)| = k$. We observe that plane supports lead to Kelp-style rendering [5, 7] where sets intersect only at common members; see also Figure 7. A short support graph minimizes ink, following Tufte's principles [8].

A sample of related work. Planar supports without fixed elements [4] or short supports with fixed elements [1, 6] have been studied, but not their combination. The case where each element belong exactly to one set is also studied in a Steiner setting [2], where additional points may be placed. If all points are 1-colored, PLANESUPPORT reduces to finding nonintersecting spanning trees. For $|C| = 2$, this was already solved by Bereg et al. [3]. To the best of our knowledge, the problem is still open for $|C| > 2$.

Results. We studied the above mentioned problems and arrived at a number of interesting results. First, we observe that there always exists a plane support for a given point set, as long as there is at least one point which is mapped to all $|C|$ colors by χ.

▶ **Observation 1.** PLANESUPPORT *is always* TRUE*, if there exists at least one $|C|$-colored point.*

However, as soon as we insist on having a short length, the problem turns out to be NP-hard, even under some seemingly simplifying restrictions.

▶ **Theorem 2.** ShortestPlaneSupport *is NP-hard, even if one or more of the following conditions hold: (a) $|C| = 2$, (b) either there is no $|C|$-colored point or all* 1*-colored points are of the same color; (c) the resulting support graph G must be a tree.*

The last restriction in the theorem above insists that our resulting support graph G should be a tree. If we look more closely at this condition, this implies that the $|C|$-colored points must induce a connected subtree of G. In other words, if $|C| = 2$, there is a backbone connecting all 2-colored points; the remaining 1-colored points are connected to this backbone via single-colored trees. One may wonder whether starting with the minimum spanning tree as such a backbone can lead to an approximation algorithm. Unfortunately, we must answer this negatively, even if we add the requirement that G must be a tree.

▶ **Lemma 3.** *Assume $|C| = 2$. There exists a family $\mathcal{F}$ of point sets such that, for each point set $P \in \mathcal{F}$, the Euclidean minimum spanning tree of its* 2*-colored points is not a subgraph of any plane support tree whose length is within a constant factor of the shortest plane support tree of P.*

Finally, we developed an ILP that solves ShortestPlaneSupport. It can be customized to allow a number of crossings, weigh crossings into the optimization, and/or require that G is a tree. The ILP also allows points to have a set of candidate positions that can be selected, such that we can start developing spatially informative set visualizations.

Ongoing research. We are aiming to use our ILP to investigate how various conditions affect the length of the support graph, such as the difference between forcing G to be a tree or not, allowing a number of intersections, etc. Also, we plan to use it to compare how well (existing) heuristic and approximation algorithms work. Finally, we aim to use it to investigate the effect of allowing elements to move from their original position. Care needs to be taken to measure structural differences, rather that shortening effects obtained simply by moving points closer to one another.

References

1 H. A. Akitaya, M. Löffler, and C. D. Tóth. Multi-colored spanning graphs. In Y. Hu, M. Nöllenburg (eds), *International Symposium on Graph Drawing and Network Visualization*, LNCS 9801, pp. 81–93, 2016.

2 S. Bereg, K. Fleszar, P. Kindermann, S. Pupyrev, J. Spoerhase, and A. Wolff. Colored Non-Crossing Euclidean Steiner Forest. In K. Elbassioni, K. Makino (eds), *International Symposium on Algorithms and Computation*, LNCS 9472, pp. 429–441, 2015.

3 S. Bereg, M. Jiang, B. Yang, and B. Zhu. On the red/blue spanning tree problem. *Theoretical Computer Science*, 412(23):2459–2467, 2011.

4 K. Buchin, M. van Kreveld, H. Meijer, B. Speckmann, and K. Verbeek. On Planar Supports for Hypergraphs. *Journal of Graph Algorithms and Applications*, 14(4):533–549, 2011.

5 K. Dinkla, M. van Kreveld, B. Speckmann, and M. A. Westenberg. Kelp Diagrams: Point Set Membership Visualization. *Computer Graphics Forum*, 31(3pt1):875—-884, 2012.

6 F. Hurtado, M. Korman, M. van Kreveld, M. Löffler, V. Sacristán, A. Shioura, R. I. Silveira, B. Speckmann, T. Tokuyama. Colored Spanning Graphs for Set Visualization. *Computational Geometry: Theory and Applications*, to appear, 2017.

7 W. Meulemans, N. H. Riche, B. Speckmann, B. Alper, and T. Dwyer. KelpFusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1846–1858, 2013.

8 E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press (Cheshire, CT), 1983.

4.4 Set Visualization Using the Metro Map Metaphor

Robert Baker (University of Kent, UK), Nan Cao (Tongji University, CN), Yifan Hu (Yahoo! Research, US), Michael Kaufmann (University of Tübingen, DE), Stephen Kobourov (University of Arizona, DE), Tamara Mchedlidze (Karlsruhe Institute of Technology, DE), Sergey Pupyrev (Facebook, US), Alexander Wolff (Universität Würzburg, DE)

We consider a visualization style for hypergraphs that is inspired by schematic metro maps. Such maps are common for urban citizens, who all know that the stations traversed by the same colored curve belong to the same metro line. This intuitive understanding of grouping have been employed to visualize other abstract data forming hypergraphs. For example, Foo [3] turns personal memories into a metro map, Nesbitt [4] and Stott et al. [10] use the metro map metaphor to visualize relationships between PhD theses and items of a business plan, Sandvad et al. [7] for building Web-based guided tour systems, and Seskovec [8] uses it for visualizing historical events. One of the most popular applications is the visualization of the movies and movie genres by the creators of the website Vodkaster.

We formalize the problem of constructing such a visualization for given hypergraph as follows. Let $H = (V, \mathcal{E})$ be a hypergraph with vertex set V and edge set $\mathcal{E} \subseteq 2^V$. A *metro-map drawing* of H is a graphical representation where each node in V is depicted by a point in the plane and each hyperedge $e \in \mathcal{E}$ by an open continuous curve that passes through the points corresponding to the vertices in e. In case two hyperedges contain the same vertex, their curves both pass through the point representing this vertex and may either *touch* or *cross* at this point. We call the latter situation a *vertex crossing*, and we call a crossing of hyperedge curves that is not a vertex crossing an *edge crossing*. A metro-map drawing of a hypergraph is called *monotone* if all hyperedge curves are monotone with respect to the x-axis.

Some Simple Observations

Since both vertex and edge crossing may impair the readability of the metro-map drawing of a hypergraph, we want to characterize the hypergraphs that can be represented without, or with a few, vertex and edge crossings. We observe that each hypergraph with at most four hyperedges can be represented without vertex and edge crossings. We call a hypergraph $H = (V, \mathcal{E})$ *k-vertex-complete* for some $k \leq |\mathcal{E}|$ if any subset $E \subset \mathcal{E}$ of at most k hyperedges has a distinct vertex in common, that is, there is an injective function $f \colon \mathcal{E} \to V$ such that $f(E) \in \bigcap_{e \in E} e \neq \emptyset$. We call $|\mathcal{E}|$-vertex-complete hypergraphs simply *vertex-complete*. We observe that a 2-vertex-complete hypergraph with five hyperedges does not have a metro-map drawing without vertex and edge crossings. This follows simply from the fact that K_5 is not planar. Next, we consider drawings with vertex crossings but without edge crossings. We can show that every vertex-complete hypergraph admits a metro-map drawing without edge crossing. The idea behind the proof is to exploit the vertices to realize the intersections among the hyperedge curves. For an example, see Figure 8.

A Heuristic

For practical applications, we propose a heuristic that constructs a metro-map drawing of a given hypergraph. The heuristic consists of four steps.

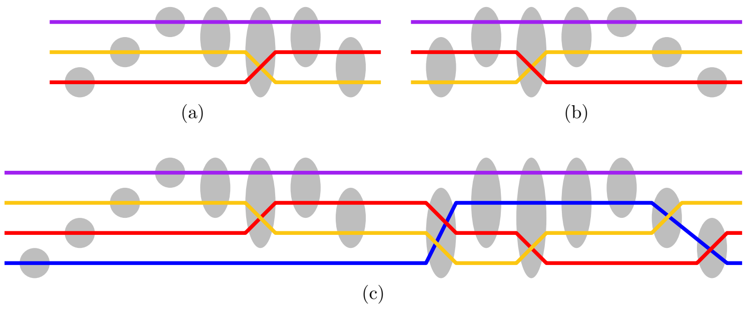

Figure 8 A metro-map drawing is a visualization of a hypergraph where the metro lines represent hyperedges and the stations represent hypervertices. (a) A metro-map drawing of a vertex-complete hypergraph with three hyperedges (violet, red, yellow), (b) a mirrored copy of (a), and (c) a metro-map drawing of a vertex-complete hypergraph with four hyperedges. The drawing is constructed recursively by routing the new (blue) hyperedge through a vertex that is only contained the new hyperedge and then through the concatenation of (a) and (b). The new hyperedge does not contain any vertex of (a), and it does contain every vertex of (b).

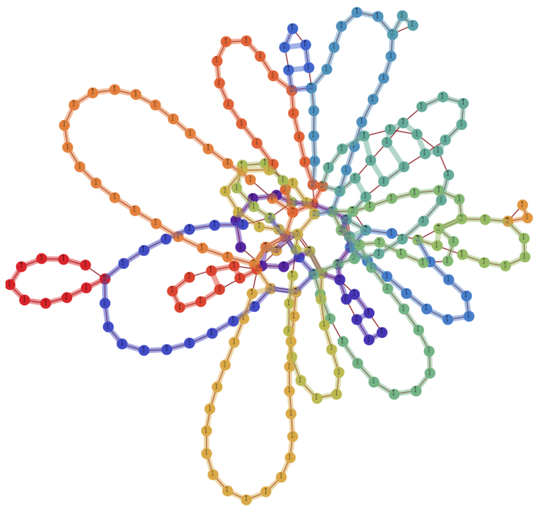

Figure 9 A preliminary drawing of the support graph of a dataset

In the first step, we simplify the hypergraph, by ignoring all vertices that belong to a single hyperedge, and contracting all vertices that belong to the same set of hyperedges.

In the second step, we construct a so-called support graph. A *support* of a hypergraph $H = (V, \mathcal{E})$ is a graph $G = (V, E)$ with the property that, for each hyperedge e of H, the graph $G[e]$ induced by e is connected. A support is *path-based* if $G[e]$ is Hamiltonian. It is NP-complete to compute a path-based support with the minimum number of edges [2]. We propose a heuristic algorithm for finding a path-based support for a given hypergraph.

In the third step, we lay out the support graph in the plane. In doing so, we try to ensure that all vertices belonging to the same hyperedges lie close by and that the paths representing the hyperedges have simple shapes. A preliminary drawing at this step is shown in Figure 9.

Finally in the fourth step, we feed the above drawing into a mixed-integer program (or some other existing algorithm) that generates a metro map layout [5].

The above steps can be implemented in many possible ways. The performance of our approach needs to be compared with existing similar approaches [1, 6, 9] experimentally.

References

1 Basak Alper, Nathalie Henry Riche, Gonzalo Ramos, and Mary Czerwinski. Design study of LineSets, a novel set visualization technique. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2259–2267, 2011. doi:10.1109/TVCG.2011.186.

2 Ulrik Brandes, Sabine Cornelsen, Barbara Pampel, and Arnaud Sallaberry. Path-based supports for hypergraphs. *J. Discrete Algorithms*, 14:248–261, 2012. doi:10.1016/j.jda.2011.12.009.

3 Brian Foo. The memory underground. http://memoryunderground.com.

4 Keith V. Nesbitt. Getting to more abstract places using the metro map metaphor. In *Proc. 8th Int. Conf. Inform. Vis. (IV'04)*, pages 488–493. IEEE, 2004. doi:10.1109/IV.2004.1320189.

5 Martin Nöllenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Trans. Vis. Comput. Graph.*, 17(5):626–641, 2011. doi:10.1109/TVCG.2010.81.

6 Francesco Paduano and Angus Graeme Forbes. Extended LineSets: a visualization technique for the interactive inspection of biological pathways. *BMC Proceedings*, 9(Suppl. 6)(S4):13 pages, 2015. doi:10.1186/1753-6561-9-S6-S4.

7 Elmer Sandvad, Kaj Grønbæk, Lennert Sloth, and Jørgen Lindskov Knudsen. A metro map metaphor for guided tours on the Web: the Webvise guided tour system. In Vincent Y. Shen, Nobuo Saito, Michael R. Lyu, and Mary Ellen Zurko, editors, *Proc. 10th Int. World Wide Web Conf. (WWW'01)*, pages 326–333. ACM, 2001. doi:10.1145/371920.372079.

8 Jure Seskovec. SNAP metromaps. http://metromaps.stanford.edu.

9 Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Trains of thought: Generating information maps. In *Proc. 21st Int. Conf. World Wide Web (WWW'12)*, pages 899–908. ACM, 2012. doi:10.1145/2187836.2187957.

10 Jonathan M. Stott, Peter Rodgers, Remo Aslak Burkhard, Michael Meier, and Matthias Thomas Jelle Smis. Automatic layout of project plans using a metro map metaphor. In *Proc. 9th Int. Conf. Inform. Vis. (IV'05)*, pages 203–206. IEEE, 2005. doi:10.1109/IV.2005.26.

4.5 Visual Analytics of Sets/Set-Typed Data and Time: Challenges and Opportunities

Daniel Archambault (Swansea University, GB), Kerstin Bunte (University of Groningen, NL), Sara Irina Fabrikant (Universität Zürich, CH), John Howse (University of Brighton, GB), Andreas Kerren (Linnaeus University – Växjö, SE), and Silvia Miksch (TU Wien, AT)

This breakout group has been composed of six people in total (see Figure 10) and represented the interdisciplinary characteristics of the overall Dagstuhl seminar consisting of experts in geographic information systems/geovisual analytics, visualization/visual analytics, formal diagrammatic reasoning, visual modelling of logic-based systems, machine learning, and data discovery in databases.

The aim of this group was to discuss, elaborate, and structure possible challenges and opportunities of set visualizations that change over time. As a first step, we discussed the communalities and specifics of sets, set-typed data, and time, to arrive at a common understanding of the various terminologies used in various cognate fields (see Figure 11). We collected state-of-the-art articles and research papers about set visualizations, spatio-temporal visualizations, visualization task taxonomies and classifications, etc. We examined various

Figure 10 (front line) Andreas Kerren, Silvia Miksch, John Howse, Kerstin Bunte; (back line) Sara Fabrikant, Daniel Archambault

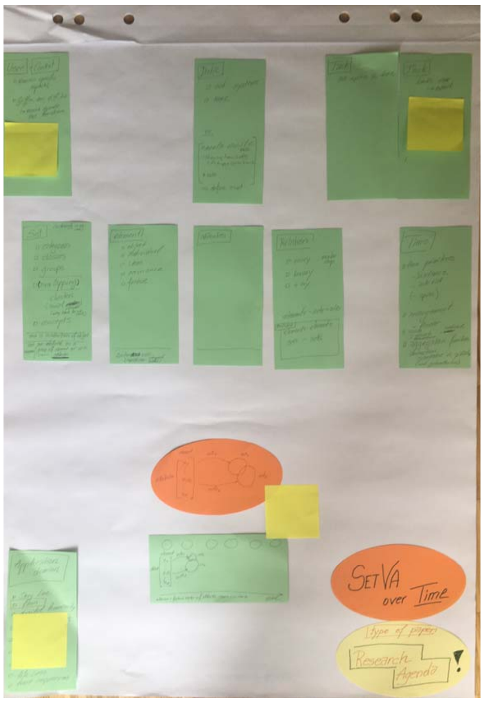

Figure 11 Structuring the various terminologies used in the different – however still computer science-related – communities.

application domains, ranging from social networks to the digital humanities, and to various kinds of flows. Finally, we defined twelve current challenges and future opportunities, which the group aims to elaborate in more detail in the next steps.

Participants

- Daniel Archambault
Swansea University, GB
- Robert Baker
University of Kent – Canterbury, GB
- Kerstin Bunte
University of Groningen, NL
- Nan Cao
Tongji University – Shanghai, CN
- Thom Castermans
TU Eindhoven, NL
- Fadi Dib
Gulf University for Science&Technology – Mishreff, KW
- Sara Irina Fabrikant
Universität Zürich, CH
- John Howse
University of Brighton, GB
- Yifan Hu
Yahoo! Research – New York, US
- Radu Jianu
City, University of London, GB
- Michael Kaufmann
Universität Tübingen, DE
- Andreas Kerren
Linnaeus University – Växjö, SE
- Stephen G. Kobourov
University of Arizona – Tucson, US
- Martin Krzywinski
BC Cancer Research Centre – Vancouver, CA
- Tamara Mchedlidze
KIT – Karlsruher Institut für Technologie, DE
- Wouter Meulemans
TU Eindhoven, NL
- Luana Micallef
Aalto University, FI
- Silvia Miksch
TU Wien, AT
- Martin Nöllenburg
TU Wien, AT
- Sergey Pupyrev
Facebook – Menlo Park, US
- Peter Rodgers
University of Kent – Canterbury, GB
- Mereke van Garderen
Universität Konstanz, DE
- Alexander Wolff
Universität Würzburg, DE
- Hsiang-Yun Wu
TU Wien, AT
- Michael Wybrow
Monash University – Caulfield, AU
- Xiaoru Yuan
Peking University, CN

Report from Dagstuhl Seminar 17341

Computational Counting

Edited by

Ivona Bezáková[1], Leslie Ann Goldberg[2], and Mark R. Jerrum[3]

1 **Rochester Institute of Technology, US,** `ib@cs.rit.edu`
2 **University of Oxford, GB,** `leslie.goldberg@cs.ox.ac.uk`
3 **Queen Mary University of London, GB,** `m.jerrum@qmul.ac.uk`

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 17341 "Computational Counting". The seminar was held from 20th to 25th August 2017, at Schloss Dagstuhl – Leibnitz Center for Informatics. A total of 43 researchers from all over the world, with interests and expertise in different aspects of computational counting, actively participated in the meeting.

Seminar August 20–25, 2017 – http://www.dagstuhl.de/17341
1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity, G.2.1 Combinatorics
Keywords and phrases approximation algorithms, computational complexity, counting problems, partition functions, phase transitions
Digital Object Identifier 10.4230/DagRep.7.8.23
Edited in cooperation with Jacob Focke

1 Executive Summary

Ivona Bezáková
Leslie Ann Goldberg
Mark Jerrum

Computational counting problems arise in practical applications in many fields such as statistical physics, information theory and machine learning. In such a problem, the goal is to compute or to estimate a weighted sum. Some typical computational counting problems include evaluating a probability, the expectation of a random variable, a partition function, or an integral.

The study of the computational complexity of counting problems requires a coherent set of techniques which are different in flavour from those employed in other algorithmic branches of computer science. Relevant techniques include the analysis of Markov chains, the analysis of correlation decay, parameterised algorithms and complexity, and dichotomy techniques for constructing detailed classifications.

Most computational problems are intractable when considered from the perspective of classical complexity, so it is important to find ways to cope with intractability. These include approximation, randomisation, as well as viewing computational counting through the lens of parameterised complexity, where the goal is to find algorithms that are efficient when some key parameter is "small". Intractability thresholds often relate to "phase transitions" as these key parameters vary. Great progress has been made in recent years towards understanding the complexity of approximate counting, based largely on a connection with these phase transitions.

Specific themes identified for consideration at the meeting included:

- *Exact counting*, including classifications, quasi-polynomial and/or moderately exponential algorithms for intractable problems, and parameterised algorithms; also complexity-theoretic limitations to obtaining exact solutions.
- *Approximate counting* including Markov Chain Monte Carlo (*MCMC*) algorithms, and algorithms based on decay of correlations; also complexity-theoretic limitations to obtaining approximate solutions.
- The interplay between *phase transitions* and computational tractability.
- *Constraint satisfaction* problems and the more general *Holant* framework. The partition functions of many models in statistical physics are included within this setting.

In the event, the talks ranged more widely than this list suggests.

Although the topic of Computational Counting has been explored at various meetings, including at Dagstuhl, for a number of years, it continues to retain its freshness. New approaches are found, new insights are gained, and new connections drawn with other areas both inside and outside computer science. Among the new directions that have emerged since the previous Dagstuhl Seminar in this series are the following.

- Results from quantum information theory applied to the apparently unrelated task of classifying the complexity of Holant problems. (Refer to the talk by Miriam Backens.)
- A new paradigm for designing polynomial-time algorithms for approximating partition functions with complex parameters. This is based on Taylor expansion in a zero-free region of the parameter space combined with an ingenious approach to enumerating small substructures. (Refer to talks by Alexander Barvinok, Jingcheng Liu, Viresh Patel and Guus Regts.)
- Emerging connections between the Lovász Local Lemma – specifically the Shearer condition and the Moser-Tardos algorithmic version – and sampling and approximate counting. (Refer to talks by Andreas Galanis and Heng Guo.)

2 Table of Contents

Executive Summary
Ivona Bezáková, Leslie Ann Goldberg, and Mark Jerrum 23

Overview of Talks

Holant problems and quantum information theory
Miriam Backens 27

TotP-completeness and a connection to the approximability of #SAT
Eleni Bakali 27

Weighted counting of 0-1 vectors in a subspace
Alexander Barvinok 27

Spatial Mixing and non-local Markov chains
Antonio Blanca 28

A fully polynomial randomized approximation scheme for Kostka numbers of two-row diagrams
Cornelius Brand 29

Finite duality, connectivity of homomorphisms, spatial mixing and Gibbs measures
Andrei A. Bulatov 29

Dyadic Tilings and Phase Transitions
Sarah Cannon 30

Information-theoretic thresholds
Amin Coja-Oghlan 30

The complexity of graph motif parameters
Radu Curticapean 31

A Fixed-Parameter Perspective on #BIS
Holger Dell 31

Matchings and the switch chain
Martin Dyer 32

Improved bounds for sampling colorings of sparse random graphs
Charis Efthymiou 32

The Complexity of Counting Surjective Homomorphisms and Compactions
Jacob Focke 33

Inapproximability of the independent set polynomial below the Shearer threshold
Andreas Galanis 33

Uniqueness of Gibbs Measures for Continuous Hardcore Models
David Gamarnik 34

A threshold result for loose Hamiltonicity in random regular uniform hypergraphs
Catherine Greenhill 34

Uniform Sampling through the Lovász Local Lemma
Heng Guo 35

Counting graph homomorphisms modulo 2
Andreas Göbel 35

Fine-grained reductions from approximate counting to decision
John Lapinskas . . . 36

The Ising Partition Function: Zeros and Deterministic Approximation
Jingcheng Liu . . . 37

Approximately counting small witnesses when there aren't too many
Kitty Meeks . . . 37

Mixing Times of Markov Chains for Self-Organizing Lists and Biased Permutations
Sarah Miracle . . . 38

Zero-free regions and approximation algorithms for graph polynomials
Viresh Patel . . . 38

Extremal bounds on partition functions via observables
Will Perkins . . . 39

Slow Convergence via Free Energy for the Ising Model
Dana Randall . . . 39

On a conjecture of Sokal concerning roots of the independence polynomial
Guus Regts . . . 40

Functional Clones and Expressibility of Partition Functions
David Richerby . . . 40

Counting Restricted Homomorphisms via Möbius Inversion over Matroid Lattices
Marc Roth . . . 40

Exact recovery in the Ising blockmodel
Piyush Srivastava . . . 41

Mixing time of random walks on dynamic graphs
Alexandre Stauffer . . . 41

Structure Learning of H-colorings
Daniel Štefankovič . . . 42

Approximating partition functions of bounded-degree Boolean counting Constraint Satisfaction Problems
Kuan Yang . . . 42

What can be sampled locally?
Yitong Yin . . . 43

Participants . . . 44

3 Overview of Talks

3.1 Holant problems and quantum information theory

Miriam Backens (University of Bristol, GB)

Holant problems are a family of counting problems on graphs, parameterised by sets of complex-valued functions of Boolean inputs. While the Holant framework was originally inspired by ideas from quantum computation, this connection has not been exploited before to analyse their complexity. We show how to apply ideas from quantum information theory to the analysis of Holant problems, using the bijection that exists between n-ary complex-valued functions of Boolean inputs and states of qubits (i.e. quantum bits), which are described by vectors in $\mathbb{C}^{2^n}$. Many interesting families of functions in the Holant framework correspond to families of quantum states that are of independent interest in quantum information theory. We sketch the role these ideas played in the derivation of two new Holant dichotomies, which make use of methods and knowledge from the theory of quantum entanglement. One of the new results, the full dichotomy for Holant^c, solves a long-standing open problem.

3.2 TotP-completeness and a connection to the approximability of #SAT

Eleni Bakali (National Technical University of Athens, GR)

Joint work of Eleni Bakali, Aggeliki Chalki, Aris Pagourtzis, Petros Pantavos and Stathis Zachos

TotP is a counting complexity class, containing all self-reducible problems in #P, with decision version in P. We study completeness under parsimonious reductions. One such TotP-complete problem is the #Clustered-Monotone-SAT, which is a version of #SAT restricted to formulas with the property that their set of satisfying assignments is connected via a tree structure. Finally we show that the general #SAT problem is reduced to #Clustered-Monotone-SAT under approximation preserving reductions.

3.3 Weighted counting of 0-1 vectors in a subspace

Alexander Barvinok (University of Michigan - Ann Arbor, US)

Joint work of Alexander Barvinok, Guus Regts

Given n complex numbers $w_1, \dots, w_n$, we define the weight of a set $X \subset \{0,1\}^n$ by

$$w(X) = \sum_{\substack{x \in X \\ x=(x_1,\dots,x_n)}} w_1^{x_1} \cdots w_n^{x_n} = \sum_{\substack{x \in X \\ x=(x_1,\dots,x_n)}} \prod_{j: x_j \neq 0} w_j.$$

We prove that if $X \subset \{0,1\}^n$ is the set of solutions to a system of homogeneous linear equations with integer coefficients such that each equation contains at most $r \geq 2$ variables and each variable is contained in at most $c \geq 1$ equations then $w(X) \neq 0$ whenever

$$|w_j| \leq \frac{0.43}{r\sqrt{c}} \quad \text{for} \quad j = 1, \ldots, n.$$

Consequently, $w(X)$ can be approximated within a relative error $0 < \epsilon < 1$ in $(rc)^{O(\ln n - \ln \epsilon)}$ time provided

$$|w_j| \leq \frac{0.42}{r\sqrt{c}} \quad \text{for} \quad j = 1, \ldots, n.$$

We apply the result to approximately count perfect matchings that are close in the Hamming distance to a given perfect matching in a given hypergraph. Namely, let $H = (V, E)$ be a hypergraph with set V of vertices, set $E \subset 2^V$ of edges, with degrees of the vertices not exceeding $r \geq 2$ and the cardinality of every edge not exceeding $c \geq 1$. Given a perfect matching M_0 in H, we consider the sum

$$\sum_M w^{\operatorname{dist}(M, M_0)},$$

where the sum is taken over all perfect matchings M in H and $\operatorname{dist}(M, M_0)$ is the number of edges of H where the matchings differ. We prove that if

$$|w| \leq \frac{0.42}{r\sqrt{c}}$$

then the sum can be approximated within a relative error $0 < \epsilon < 1$ in $(rc)^{O(\ln |E| - \ln \epsilon)}$ time.

3.4 Spatial Mixing and non-local Markov chains

Antonio Blanca (Georgia Institute of Technology - Atlanta, US)

Joint work of Antonio Blanca, Pietro Caputo, Alistair Sinclair, Eric Vigoda

In this talk, we consider the effects that the strong spatial mixing condition (SSM) has on the rate of convergence to equilibrium distribution of *non-local* Markov chains for spin systems on the integer lattice graph $\mathbb{Z}^d$. We show that SSM implies that the relaxation time (i.e., the inverse spectral gap) of general block dynamics is $O(r)$, where r is the number of blocks. We also proved that SSM implies an $O(1)$ bound for the relaxation time of the Swendsen-Wang dynamics in d-dimensional cubes of $\mathbb{Z}^d$. For *monotone* spin systems, we establish that the mixing time of systematic scan dynamics is $O(\log n(\log \log n)^2)$, provided that SSM holds. Our proofs use a variety of techniques for the analysis of Markov chains including coupling, functional analysis and linear algebra.

3.5 A fully polynomial randomized approximation scheme for Kostka numbers of two-row diagrams

Cornelius Brand (Universität des Saarlandes, DE)

Joint work of Markus Bläser, Cornelius Brand, Sandip Sinha

Kostka numbers are a central quantity in the representation theory of the symmetric group, where they appear as multiplicities in the isotypic decomposition of permutation modules. They also arise as a special case of other representation-theoretic coefficients, such as Littlewood-Richardson and Kronecker coefficients. Computing these coefficients is of high practical relevance for the Geometric Complexity Theory program, and the scarcity of results in this area make the special case of Kostka numbers a very natural place to start. Young's rule gives them a combinatorial interpretation: Kostka numbers count certain arrangements of boxes, filled with a given contingent of numbers. Narayanan [1] proved them #P-complete, even if the Kostka numbers come from two-rowed box arrangements. Although their algorithmic aspects have been thoroughly studied (see Rassart's thesis [2] and the article by Barvinok and Fomin [3]), all known algorithmic results are exact and thus, by Narayanan's result, inherently inefficient. Narayanan posed the complexity of efficiently approximating these numbers as an open problem. We make a first step towards resolving the open problem: We exhibit an FPRAS for the aforementioned subclass of Kostka numbers arising from two-rowed arrangements.

References

1 Hariharan Narayanan. *On the complexity of computing Kostka numbers and Littlewood-Richardson coefficients*. Journal of Algebraic Combinatorics, 2006

2 Etienne Rassart. *Geometric approaches to computing Kostka numbers and Littlewood-Richardson coefficients*. PhD Thesis, 2004

3 Alexander Barvinok and Sergey Fomin. *Sparse Interpolation of Symmetric Polynomials*. Advances in Applied Mathematics, 1997

3.6 Finite duality, connectivity of homomorphisms, spatial mixing and Gibbs measures

Andrei A. Bulatov (Simon Fraser University - Burnaby, CA)

Joint work of Andrei Bulatov, Victor Dalmau, Raimundo Briceno

We consider the properties and objects associated with a finite relational structure H and listed in the title:

- structure H has finite duality
- for any G the set of homomorphisms hom(G,H) is connected, that is, any homomorphism can be transformed to any other homomorphism through a sequence of homomorphisms such that any two consequent ones only differ in bounded number of points
- for any G the set hom(G,H) satisfies the strong spatial mixing property defined here in terms of extendibility of partial homomorphisms
- for any G there exist parameters (activities) such that hom(G,H) has a unique Gibbs measure. We show that for any H these properties are closely related.

3.7 Dyadic Tilings and Phase Transitions

Sarah Cannon (Georgia Institute of Technology - Atlanta, US)

Joint work of Sarah Cannon, Sarah Miracle, David Levin, Dana Randall, Alexandre Stauffer

We consider the weighted edge-flip Markov chain for dyadic tilings, show it exhibits a phase transition, and give rigorous results about its behavior at the phase transition. A dyadic tiling of size n is a tiling of the unit square by n non-overlapping dyadic rectangles, each of area $1/n$, where a dyadic rectangle is any rectangle that can be written in the form $[a2^{-s}, (a+1)2^{-s}] \times [b2^{-t}, (b+1)2^{-t}]$ for $a, b, s, t \in \mathbb{Z}_{\geq 0}$. The edge-flip Markov chain selects a random edge of the tiling and replaces it with its perpendicular bisector if doing so yields a valid dyadic tiling. We consider a weighted version of this Markov chain where, given a parameter $\lambda > 0$, we would like to generate each dyadic tiling σ with probability proportional to $\lambda^{|\sigma|}$, where $|\sigma|$ is the total edge length. We give results for both the mixing time (t_{mix}) and relaxation time (t_{rel}) of this chain. Specifically, we show there is a phase transition: when $\lambda < 1$, the edge-flipping chain has $t_{mix} = O(n^2 \log n)$ and $t_{rel} = O(n \log n)$, while when $\lambda > 1$, both t_{mix} and t_{rel} are at least $exp(\Omega(n^2))$. At the critical point $\lambda = 1$, we show that $t_{rel} = O(n^{4.09})$, which implies that $t_{mix} = O(n^{5.09})$, the first such polynomial bounds. We complement this by showing that both t_{mix} and t_{rel} are at least $\Omega(n^{1.38})$, improving upon the previously best lower bound of $\Omega(n \log n)$ coming from the diameter of the chain. For relaxation time, this means the behavior at the critical point $\lambda = 1$ is provably different than on either side of the critical point, providing support for a well-known statistical physics conjecture.

3.8 Information-theoretic thresholds

Amin Coja-Oghlan (Goethe-Universität - Frankfurt am Main, DE)

Joint work of Charilaos Efthymiou, Nor Jaafari, Tobias Kapetanopoulos, Mihyun Kang, Florent Krzakala, Amin Coja-Oghlan, Will Perkins, Lenka Zdeborová

This talk deals with the problem of inferring a given signal from a noisy observation. The question is: for what signal-to-noisy ratio is a non-trivial recovery of the signal information-theoretically possible? The first result concerns low-density generator matrix codes, for which we obtain a formula both for the information-theoretic threshold and for the mutual information. The second result is about the stochastic block model, where we obtain the information-theoretic threshold in the disassortative case with a general number of colors. Up to the information-theoretic threshold the stochastic block model and the Erdos-Renyi model are mutually contiguous. The proofs are based on some general results on discrete probability distributions, which may be of independent interest.

3.9 The complexity of graph motif parameters

Radu Curticapean (Hungarian Academy of Sciences - Budapest, HU)

Joint work of Radu Curticapean, Holger Dell, Dániel Marx

We consider the problem of counting small patterns in large graphs. Many such problems can be expressed as linear combinations of induced subgraph numbers: For fixed graphs H, let Ind_H be the function that maps input graphs G to the number of induced subgraphs of G that are isomorphic to H. Now consider the function space obtained by taking finite linear combinations of these Ind_H functions. We call this the space of "graph motif parameters".

The set $\{Ind_H|H\}$ forms a basis for this space. But it is known that so does the set $\{Sub_H|H\}$ of functions that count (not necessarily induced) subgraphs, as well as the set $\{Hom_H|H\}$ of functions that count homomorphisms from fixed H to input graphs G. In particular, the numbers of induced subgraphs, subgraphs, or homomorphisms from fixed H can be expressed as graph motif parameters.

We are interested in the computational complexity of evaluating fixed graph motif parameters f on input graphs. To this end, we define the complexity $C(f)$ as the infimum over all c such that the evaluation of f has an $O(n^c)$ time algorithm. This number is finite for every f, as we can always evaluate f by brute-force.

Our main result is that for any graph motif parameter f, the complexity $C(f)$ is precisely the maximum $C(Hom_F)$ over all F that have a non-zero coefficient when expressing f in the Hom-basis. From previous results, we have a relatively good grip on the complexity of homomorphisms from small patterns. Using this, we gained a more comprehensive understanding of the complexity of other classes of graph motif parameters, with a focus on counting subgraphs, where we obtain much simpler proofs of existing dichotomies, together with tight lower bounds under the Exponential Time Hypothesis that previously seemed out of reach.

3.10 A Fixed-Parameter Perspective on #BIS

Holger Dell (Universität des Saarlandes, DE)

Joint work of Radu Curticapean, Holger Dell, Fedor Fomin, Leslie Ann Goldberg, John Lapinskas

The problem of (approximately) counting the independent sets of a bipartite graph (#BIS) is the canonical approximate counting problem that is complete in the intermediate complexity class $\#\mathrm{RH}\Pi_1$. It is believed that #BIS does not have an efficient approximation algorithm but also that it is not NP-hard. We study the robustness of the intermediate complexity of #BIS by considering variants of the problem parameterised by the size of the independent set. We exhaustively map the complexity landscape for three problems, with respect to exact computation and approximation and with respect to conventional and parameterised complexity. The three problems are counting independent sets of a given size, counting independent sets with a given number of vertices in one vertex class and counting maximum independent sets amongst those with a given number of vertices in one vertex class. Among other things, we show that all of these problems are NP-hard to approximate within any polynomial ratio. (This is surprising because the corresponding problems without the size parameter are complete in #RHΠ_1, and hence are not believed to be NP-hard.) We also show that the first problem is #W[1]-hard to solve exactly but admits an FPTRAS, whereas the other two are W[1]-hard to approximate even within any polynomial ratio. Finally, we show that, when restricted to graphs of bounded degree, all three problems have efficient exact fixed-parameter algorithms.

3.11 Matchings and the switch chain

Martin Dyer (University of Leeds, GB)

Joint work of Martin Dyer, Heiko Müller, Mark Jerrum

The switch chain is a simple Markov chain for generating a random perfect matching in a graph. This chain was studied in a 2016 paper for certain classes of bipartite graphs. We will describe extensions of that work to classes of nonbipartite graphs.

3.12 Improved bounds for sampling colorings of sparse random graphs

Charis Efthymiou (Goethe-Universität - Frankfurt a. M., DE)

Joint work of Charilaos Efthymiou, Tom Hayes, Daniel Štefankovič, Eric Vigoda

We study the mixing properties of the single-site Markov chain known as the Glauber dynamics for sampling k-colorings of a sparse random graph $G(n, d/n)$ for constant d. The best known rapid mixing results for general graphs are in terms of the maximum degree Δ of the input graph G and hold when $k > 11\Delta/6$ for all G. Improved results hold when $k > \alpha\Delta$ for graphs with girth ≥ 5 and Δ sufficiently large where $\alpha \approx 1.7632\ldots$ is the root of $\alpha = \exp(1/\alpha)$; further improvements on the constant α hold with stronger girth and maximum degree assumptions.

For sparse random graphs the maximum degree is a function of n and the goal is to obtain results in terms of the expected degree d. The following rapid mixing results for $G(n, d/n)$ hold with high probability over the choice of the random graph for sufficiently large constant d. Mossel and Sly [1] proved rapid mixing for constant k, and Efthymiou [2] improved this to k linear in d. The condition was improved to $k > 3d$ by Yin and Zhang [3] using non-MCMC methods.

Here we prove rapid mixing when $k > \alpha d$ where $\alpha \approx 1.7632\ldots$ is the same constant as above. Moreover we obtain $O(n^3)$ mixing time of the Glauber dynamics, while in previous rapid mixing results the exponent was an increasing function in d. As in previous results for random graphs our proof analyzes an appropriately defined block dynamics to "hide" high-degree vertices. One new aspect in our improved approach is utilizing so-called local uniformity properties for the analysis of block dynamics. To analyze the "burn-in" phase we prove a concentration inequality for the number of disagreements propagating in large blocks.

References

1 Elchanan Mossel and Allan Sly. *Gibbs rapidly samples colorings of $G(n, d/n)$*. Probability Theory and Related Fields, 2009

2 Charilaos Efthymiou. *MCMC sampling colourings and independent sets of $G(n, d/n)$ near uniqueness threshold*. SODA, 2014
3 Yitong Yin and Chihao Zhang. *Sampling in Potts Model on Sparse Random Graphs*. APPROX/RANDOM, 2016

3.13 The Complexity of Counting Surjective Homomorphisms and Compactions

Jacob Focke (University of Oxford, GB)

Joint work of Jacob Focke, Leslie Ann Goldberg, Stanislav Živný

A homomorphism from a graph G to a graph H is a function from the vertices of G to the vertices of H that preserves edges. A homomorphism is *surjective* if it uses all of the vertices of H and it is a *compaction* if it uses all of the vertices of H and all of the non-loop edges of H. Hell and Nešetřil gave a complete characterisation of the complexity of deciding whether there is a homomorphism from an input graph G to a fixed graph H. A complete characterisation is not known for surjective homomorphisms or for compactions, though there are many interesting results. Dyer and Greenhill gave a complete characterisation of the complexity of counting homomorphisms from an input graph G to a fixed graph H. In this paper, we give a complete characterisation of the complexity of counting surjective homomorphisms from an input graph G to a fixed graph H and we also give a complete characterisation of the complexity of counting compactions from an input graph G to a fixed graph H.

3.14 Inapproximability of the independent set polynomial below the Shearer threshold

Andreas Galanis (University of Oxford, GB)

Joint work of Andreas Galanis, Leslie Ann Goldberg, Daniel Štefankovič

For a graph G, let $p_G(\lambda) = \sum \lambda^{|I|}$ where the sum ranges over all the independent sets I of G. For which values of λ can we approximate $p_G(\lambda)$ on graphs G of max degree D?

For $\lambda > 0$, breakthrough results of Weitz and Sly established a computational transition from easy to hard at the tree uniqueness threshold from statistical physics, given by $\lambda_c(D) = (D-1)^{(D-1)}/(D-2)^D$.

For $\lambda < 0$, the evaluation of the independent set polynomial is connected to the conditions of the Lovasz Local Lemma. Shearer identified the threshold $\lambda^*(D) = (D-1)^{(D-1)}/D^D$ as the maximum value q such that every family of events with failure probability at most q and whose dependency graph has max degree D has nonempty intersection. Very recently, Patel and Regts, and Harvey et al. have independently designed FPTASes for approximating the value of the independent set polynomial whenever $0 > \lambda > -\lambda^*(D)$.

Our main result establishes for the first time a computational transition at the Shearer threshold. We show that for all $D \geq 3$, for all $\lambda < -\lambda^*(D)$, it is NP-hard to approximate the independent set polynomial on graphs of maximum degree D, even within an exponential factor. In fact, we now have the following picture for evaluating the independent set polynomial on graphs G of max degree D.

- For $-\lambda^*(D) < \lambda < \lambda_c(D)$, the problem admits an FPTAS.
- For $\lambda < -\lambda^*(D)$ or $\lambda > \lambda_c(D)$, the problem is NP-hard to approximate.

3.15 Uniqueness of Gibbs Measures for Continuous Hardcore Models

David Gamarnik (MIT - Cambridge, US)

Joint work of David Gamarnik, Kavita Ramanan

We formulate a continuous version of the well known discrete hardcore (or independent set) model on a locally finite graph, parameterized by the so-called activity parameter $\lambda > 0$. In this version, the state or "spin value" x_u of any node u of the graph lies in the interval $[0, 1]$, the hardcore constraint $x_u + x_v \leq 1$ is satisfied for every edge (u, v) of the graph, and the space of feasible configurations is given by a convex polytope. When the graph is a regular tree, we show that there is a unique Gibbs measure associated to each activity parameter $\lambda > 0$. Our result shows that, in contrast to the standard discrete hardcore model, the continuous hardcore model does not exhibit a phase transition on the infinite regular tree. We also consider a family of continuous models that interpolate between the discrete and continuous hardcore models on a regular tree when $\lambda = 1$ and show that each member of the family has a unique Gibbs measure, even when the discrete model does not. In each case, the proof entails the analysis of an associated Hamiltonian dynamical system that describes a certain limit of the marginal distribution at a node. Furthermore, given any sequence of regular graphs with fixed degree and girth diverging to infinity, we apply our results to compute the asymptotic limit of suitably normalized volumes of the corresponding sequence of convex polytopes of feasible configurations. In particular, this yields an approximation for the partition function of the continuous hard core model on a regular graph with large girth in the case $\lambda = 1$.

3.16 A threshold result for loose Hamiltonicity in random regular uniform hypergraphs

Catherine Greenhill (UNSW Sydney, AU)

Joint work of Catherine Greenhill, Daniel Altman, Mikhail Isaev, Reshma Ramadurai

A hypergraph is s-uniform if every edge contains s vertices. Robinson and Wormald [1, 2] proved that for any constant $r \geq 3$, a uniformly random r-regular graph on n vertices is Hamiltonian with probability which tends to 1 as $n \to \infty$. We extend this result to loose Hamilton cycles in uniformly random s-uniform r-regular hypergraphs, finding the degree threshold (as a function of s) which guarantees existence.

References

1 R.W. Robinson and N.C. Wormald. *Almost all cubic graphs are hamiltonian.* Random Structures and Algorithms, 1992

2 R.W. Robinson and N.C. Wormald. *Almost all regular graphs are hamiltonian.* Random Structures and Algorithms, 1994

3.17 Uniform Sampling through the Lovász Local Lemma

Heng Guo (Queen Mary University of London, GB)

Joint work of Heng Guo, Mark Jerrum, Jingcheng Liu

We propose a new algorithmic framework, called "partial rejection sampling", to draw samples exactly from a product distribution, conditioned on none of a number of bad events occurring. Our framework builds new connections between the variable framework of the Lovász Local Lemma and some classical sampling algorithms such as the "cycle-popping" algorithm for rooted spanning trees by Wilson. Among other applications, we discover new algorithms to sample satisfying assignments of k-CNF formulas with bounded variable occurrences.

3.18 Counting graph homomorphisms modulo 2

Andreas Göbel (Hasso-Plattner-Institut - Potsdam, DE)

Joint work of Andreas Göbel, Leslie Goldberg, David Richerby

The complexity of modular counting was introduced by Papadimitriou and Zachos and it has been pioneered by Valiant who famously introduced a problem for which counting modulo 7 is easy where counting modulo 2, is intractable. A characteristic feature of modular counting is that cancellations make wider classes of instances tractable than is the case for exact (non-modular) counting.

A homomorphism from a graph G to a graph H is a function from $V(G)$ to $V(H)$ that preserves edges. Many combinatorial structures that arise in mathematics and in computer science can be represented naturally as graph homomorphisms and as weighted sums of graph homomorphisms. Modular counting provides a rich setting in which to study the structure of homomorphism problems.

In this talk we will discuss the complexity of counting graph homomorphisms modulo 2. Faben and Jerrum are the first to study the complexity of this problem. They show that for a specific family of target graphs the problem of counting homomorphisms to a graph modulo 2 is computationally easy and conjecture that for every other target graph the problem is computationally hard. They also prove their conjecture when H is a tree. Our results build upon the work of Faben and Jerrum. We first prove their conjecture for the class of cactus graphs, which are connected graphs in which every edge belongs to at most one cycle. We then show that for all graphs that contain no 4-cycles the problem of counting graph homomorphisms modulo 2 is either easy to compute or hard to compute, and that there are no target graphs H for which the problem has intermediate complexity.

3.19 Fine-grained reductions from approximate counting to decision

John Lapinskas (University of Oxford, GB)

License Creative Commons BY 3.0 Unported license
© John Lapinskas
Joint work of Holger Dell, John Lapinskas
Main reference Holger Dell and John Lapinskas. Fine-grained reductions from approximate counting to decision. Preprint.
URL https://arxiv.org/abs/1707.04609

The main problems in fine-grained complexity are CNF-SAT, the Orthogonal Vectors problem, 3SUM, and the Negative-Weight Triangle problem (which is closely related to All-Pairs Shortest Path). In this paper, we consider the approximate counting version of each problem; thus instead of simply deciding whether a witness exists, we attempt to (multiplicatively) approximate the number of witnesses. In each case, we provide a fine-grained reduction from the approximate counting form to the usual decision form. For example, if there is an $O(c^n)$-time algorithm that solves k-SAT for all k, then we prove there is an $O((c + o(1))^n)$-time algorithm to approximately count the satisfying assignments. Similarly, we get that the exponential time hypothesis (ETH) is equivalent to an approximate counting version. This mirrors a result of Sipser [2] and Stockmeyer [3], who proved such a result in the classical polynomial-time setting, and a similar result due to Müller [1] in the FPT setting.

Our algorithm for polynomial-time problems applies in a general setting in which we approximately count edges of a bipartite graph to which we have limited access. In particular, this means it can be applied to problem variants in which significant improvements over the conjectured running time bounds are already known. For example, the Orthogonal Vectors problem over $\mathrm{GF}(m)^d$ for constant m can be solved in time $n \cdot \mathrm{poly}(d)$ by a result of Williams and Yu [5]; our result implies that we can approximately count the number of orthogonal pairs with essentially the same running time. Moreover, our overhead is only polylogarithmic, so it can be applied to subpolynomial improvements such as the $n^3/\exp\big(\Theta(\sqrt{\log n})\big)$ time algorithm for the Negative-Weight Triangle problem due to Williams [4].

References

1 Moritz Müller. Randomized Approximations of Parameterized Counting Problems. *Parameterized and Exact Computation: Second International Workshop (IWPEC 2006)*, 50–59.
2 Michael Sipser. A Complexity Theoretic Approach to Randomness. *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing* (STOC 1983), 330–335.
3 Larry Stockmeyer. On Approximation Algorithms for #P. *SIAM Journal on Computing* **14.4**, 849–861, 1985.
4 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing (STOC 2014)*, 664–673.
5 Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, 1867–1877.

3.20 The Ising Partition Function: Zeros and Deterministic Approximation

Jingcheng Liu (University of California - Berkeley, US)

Joint work of Jingcheng Liu, Alistair Sinclair, Piyush Srivastava

We study the problem of approximating the partition function of the ferromagnetic Ising model in graphs and hypergraphs. Our first result is a *deterministic* approximation scheme (an FPTAS) for the partition function in bounded degree graphs that is valid over the entire range of parameters β (the interaction) and λ (the external field), except for the case $|\lambda| = 1$ (the "zero-field" case). A *randomized* algorithm (FPRAS) for all graphs, and all β, λ, has long been known. Unlike most other deterministic approximation algorithms for problems in statistical physics and counting, our algorithm does not rely on the "decay of correlations" property. Rather, we exploit and extend machinery developed recently by Barvinok, and Patel and Regts, based on the location of the complex zeros of the partition function, which can be seen as an algorithmic realization of the classical Lee-Yang approach to phase transitions. Our approach extends to the more general setting of the Ising model on hypergraphs of bounded degree and edge size, where no previous algorithms (even randomized) were known for a wide range of parameters. In order to achieve this extension, we establish a tight version of the Lee-Yang theorem for the Ising model on hypergraphs, improving a classical result of Suzuki and Fisher.

3.21 Approximately counting small witnesses when there aren't too many

Kitty Meeks (University of Glasgow, GB)

Suppose we want to know how many witnesses of size k exist in a universe of size n. If we can guarantee that the total number of witnesses is sufficiently large, we can apply well-known random sampling techniques in an obvious way to obtain a good estimate of the number of witnesses. Here, we consider the other extreme: if we know that the number of witnesses is very small (so in particular an efficient counting algorithm could in fact examine all witnesses), can we obtain an efficient algorithm to (approximately) count the number of witnesses? We answer this question in the affirmative for a large family of problems, building on previous work on enumerating small witnesses [1] to show that, so long as there is an fpt-algorithm which solves corresponding decision problem correctly with probability greater than 1/2, we can obtain an FPTRAS for the counting problem. We also consider a more challenging version of the problem in which we wish to count all minimal witnesses having size at most k, and discuss how to solve this in one specific example.

References

1 Kitty Meeks. *Randomised Enumeration of Small Witnesses Using a Decision Oracle*. IPEC, 2016

3.22 Mixing Times of Markov Chains for Self-Organizing Lists and Biased Permutations

Sarah Miracle (University of St. Thomas - St. Paul, US)

Joint work of Prateek Bhakta, Sarah Miracle, Dana Randall, Amanda Streib

In this talk I discuss a biased version of the nearest-neighbor transposition Markov chain on the set of permutations where neighboring elements i and j are placed in order (i, j) with probability $p_{i,j}$. The goal is to identify the class of parameter sets $\mathbf{P} = \{p_{i,j}\}$ for which this Markov chain is rapidly mixing.

In joint work with Prateek Bhakta, Dana Randall and Amanda Streib, we use a reduction from biased permutations to Asymmetric Simple Exclusion Processes (ASEPs) to show that the chain may be slow in the most general setting, even if $\mathbf{P}$ is positively biased (i.e., $1 \geq p_{i,j} \geq 1/2$ for all $i < j$). We then prove the chain is rapidly mixing for two new classes. The first is "Choose Your Weapon," where we are given $r_1, \ldots, r_{n-1}$ with $1 \geq r_i \geq 1/2$ and $p_{i,j} = r_i$ for all $i < j$. In the second class "League Hierarchies," the probabilities are based on binary trees.

Finally, in joint work with Amanda Streib, we consider distributions arising from k-class particle processes, where the elements are divided into k classes and the probability of exchanging neighboring elements depends on the particular classes the elements are in. We further require that k is a constant, and all probabilities between elements in different classes are bounded away from 1/2. These particle processes arise in the context of self-organizing lists and our result also applies beyond permutations to the setting where all particles in a class are indistinguishable. Additionally we show that a broader class of distributions based on trees is also rapidly mixing.

3.23 Zero-free regions and approximation algorithms for graph polynomials

Viresh Patel (University of Amsterdam, NL)

Joint work of Viresh Patel, Guus Regts

In this talk, I discuss a new way of constructing deterministic polynomial-time approximation algorithms for computing complex-valued evaluations of a large class of graph polynomials on bounded degree graphs. In particular, our approach works for the Tutte polynomial and independence polynomial, as well as partition functions of complex-valued spin and edge-coloring models. Our work builds on a recent line of work initiated by Barvinok, which provides a new algorithmic approach besides the existing Markov chain Monte Carlo method and the correlation decay method for these types of problems.

3.24 Extremal bounds on partition functions via observables

Will Perkins (University of Birmingham, GB)

Joint work of Ewan Davies, Matthew Jenssen, Barnaby Roberts, Guillem Perarnau, Will Perkins

We present a new method for proving extremal bounds on the partition function of statistical physics models on various classes of bounded degree graphs. As an example, consider the hard core model of a random independent set from a graph G, with $\Pr(I) = \lambda^{|I|}/Z_G(\lambda)$, where $Z_G(\lambda)$ is the partition function of the model. Let $\overline{\alpha}_G(\lambda) = \lambda \cdot (\log Z_G(\lambda))'/|V(G)|$ be the occupancy fraction, or the expected fraction of vertices appearing in the random independent set. Using a linear programming relaxation, we show that for any d-regular graph G and any $\lambda > 0$, $\overline{\alpha}_G(\lambda) \leq \overline{\alpha}_{K_{d,d}}(\lambda)$. This strengthens results of Kahn, Zhao, Galvin, and Tetali on the partition function of the hard core model on regular graphs, as the bound on the occupancy fraction can be integrated to obtain the corresponding bound on the log partition function. We describe a general framework for this method in terms of local weak convergence of Gibbs measures on sparse graphs and describe further applications to matchings and to independent sets in regular graphs with girth constraints.

3.25 Slow Convergence via Free Energy for the Ising Model

Dana Randall (Georgia Institute of Technology - Atlanta, US)

Joint work of David Gillman, Dana Randall

We consider mixing times of Glauber dynamics for the ferromagnetic Ising model on an $n \times n$ square lattice region. If we fix the spins on the top and bottom sides of the square to be $+$ and the left and right sides to be $-$, a standard Peierls argument shows that below some critical temperature t_c, any local Markov chain M requires time exponential in n to mix. The argument works by showing that each state in the cut has small energy (and therefore probability). We consider a family of "balanced mixed boundary conditions" formed by rotating the $+$ sides and $-$ sides pn lattice steps clockwise around the square, $0 \leq p < 1$. The Peierls argument can be extended to this family; however, one finds that the critical temperature $t_c(p)$ approaches 0 as p approaches $1/2$, and the argument breaks down when $p = 1/2$. Here we show that there is a universal temperature T below which $\mathcal{M}$ will be slow for any balanced mixed boundary (defined as above). The novelty of our argument is showing that there is an exponentially small cut indicated by the *free energy*; we show that the free energy is smaller for a certain cut set due to the energy term when p is close to 0 or 1 and due to the entropy term when p is close to $1/2$, with all other cases interpolating between these two extremes.

3.26 On a conjecture of Sokal concerning roots of the independence polynomial

Guus Regts (University of Amsterdam, NL)

Joint work of Han Peters, Guus Regts

Sokal conjectured about 16 years ago that there exists a region D_Δ in the complex plane that contains the interval $[0, \lambda_c(\Delta))$, where $\lambda_c(\Delta) := \frac{(\Delta-1)^{\Delta-1}}{(\Delta-2)^\Delta}$ and $\Delta \geq 3$, such that for any graph G of maximum degree at most Δ the independence polynomial of G does not vanish on D_Δ. In joint work with Han Peters we have settled this conjecture using complex dynamics. In this talk I will explain the connection between the location of zeros of the independence polynomial and complex dynamical systems and give some ideas of our proof of the conjecture. I will also explain how, by work of Patel and myself, building on a line of work initiated by Barvinok, this gives a new FPTAS for approximating evaluations of the independence polynomial on D_Δ. I will end with stating some open problems that arise naturally from our work.

3.27 Functional Clones and Expressibility of Partition Functions

David Richerby (University of Oxford, GB)

Joint work of Andrei Bulatov, Leslie Ann Goldberg, Mark Jerrum, David Richerby, Stanislav Živný

The complexity of counting CSPs depends very much on reductions between constraint languages, and whether functions in one constraint language can be expressed in terms of those in another. Functional clones are sets of functions closed under the appropriate notions of expressibility for CSPs. We give a tour of some relevant clones, including those related to the Ising model and matchgates, and the relationships between them.

3.28 Counting Restricted Homomorphisms via Möbius Inversion over Matroid Lattices

Marc Roth (Universität des Saarlandes, DE)

We present a framework for the complexity classification of parameterized counting problems that can be formulated as the summation over the numbers of homomorphisms from small pattern graphs $H_1, \ldots, H_l$ to a big host graph G with the restriction that the coefficients correspond to evaluations of the Möbius function over the lattice of a graphic matroid. This generalizes the idea of Curticapean, Dell and Marx [1] who used a result of Lovász stating that the number of subgraph embeddings from a graph H to a graph G can be expressed as such a sum over the lattice of partitions of H. In the first step we introduce what we call graphically restricted homomorphisms that, inter alia, generalize subgraph embeddings as well as locally injective homomorphisms. We provide a complete parameterized complexity dichotomy for counting such homomorphisms, that is, we identify classes of patterns for which the problem is fixed-parameter tractable (FPT), including an algorithm, and prove that all other pattern classes lead to #W[1]-hard problems. The main ingredients of the proof are the complexity classification of linear combinations of homomorphisms due to Curticapean, Dell and Marx [1] as well as a corollary of Rota's NBC Theorem which states that the sign of the Möbius function over a geometric lattice only depends on the rank of its arguments.

References

1 Radu Curticapean, Holger Dell and Dániel Marx. *Homomorphisms Are a Good Basis for Counting Small Subgraphs.* STOC, 2017

3.29 Exact recovery in the Ising blockmodel

Piyush Srivastava (TIFR Mumbai, IN)

Joint work of Quentin Berthet, Philippe Rigollet, Piyush Srivastava

We introduce the Ising blockmodel, a graphical model for community detection based on the classical Ising model from statistical mechanics. We establish tight bounds on the sample complexity for both information theoretic and algorithmically efficient recovery of the planted communities. We also show that the sample complexity exhibits a phase transition in the parameter space of the model.

3.30 Mixing time of random walks on dynamic graphs

Alexandre Stauffer (University of Bath, GB)

Joint work of Yuval Peres, Alexandre Stauffer, Jeff Steif

In this talk I discuss the problem of random walks on dynamic graphs (that is, on graphs that changes at the same time that the walker moves). We initially focus on the model of random walk on dynamical percolation. In this model, the edges of a graph G are either open or closed, and refresh their status at rate μ. At the same time a random walker moves on G at rate 1 but only along edges which are open. The regime of interest here is when μ goes to zero as the number of vertices of G goes to infinity, since this creates long-range dependencies on the model. When G is the d-dimensional torus of side length n, we prove that in the subcritical regime, the mixing times is of order n^2/μ. We also obtain results concerning mean squared displacement and hitting times. We present new results in this model and discuss some more general questions regarding random walks on dynamic graphs.

3.31 Structure Learning of H-colorings

Daniel Štefankovič (University of Rochester, US)

Joint work of Antonio Blanca, Zongchen Chen, Daniel Štefankovič, Eric Vigoda

We study the structure learning problem for graph homomorphisms, commonly referred to as H-colorings, including the weighted case which corresponds to spin systems with hard constraints. The learning problem is as follows: for a fixed (and known) constraint graph H with q colors and an unknown graph $G = (V, E)$ with n vertices, given uniformly random H-colorings of G, how many samples are required to learn the edges of the unknown graph G? We give a characterization of H for which the problem is identifiable for every G, i.e., we can learn G with an infinite number of samples. We focus particular attention on the case of proper vertex q-colorings of graphs of maximum degree d where intriguing connections to statistical physics phase transitions appear. We prove that when $q > d$ the problem is identifiable and we can learn G in $\mathrm{poly}(d, q) \times O(n^2 \log n)$ time. In contrast for soft-constraint systems, such as the Ising model, the best possible running time is exponential in d. When $q \leq d$ we prove that the problem is not identifiable, and we cannot hope to learn G. When $q < d - \sqrt{d} + \Theta(1)$ we prove that even learning an equivalent graph (any graph with the same set of H-colorings) is computationally hard – sample complexity is exponential in n in the worst-case. For the q-colorings problem, the threshold for efficient learning seems to be connected to the uniqueness/non-uniqueness phase transition at $q = d$. We explore this connection for general H-colorings and prove that under a well-known condition in statistical physics, known as Dobrushin uniqueness condition, we can learn G in $\mathrm{poly}(d, q) \times O(n^2 \log n)$ time.

3.32 Approximating partition functions of bounded-degree Boolean counting Constraint Satisfaction Problems

Kuan Yang (University of Oxford, GB)

Joint work of Andreas Galanis, Leslie Ann Goldberg, Kuan Yang

We study the complexity of approximate counting Constraint Satisfaction Problems (#CSPs) in a bounded degree setting. Specifically, given a Boolean constraint language Γ and a degree bound Δ, we study the complexity of $\#\mathrm{CSP}_\Delta(\Gamma)$, which is the problem of counting satisfying assignments to CSP instances with constraints from Γ and whose variables can appear at most Δ times. Our main result shows that: (i) if every function in Γ is affine, then $\#\mathrm{CSP}_\Delta(\Gamma)$ is in FP for all Δ, (ii) otherwise, if every function in Γ is in a class called IM_2, then for all sufficiently large Δ, $\#\mathrm{CSP}_\Delta(\Gamma)$ is equivalent under approximation-preserving (AP) reductions to the counting problem #BIS (the problem of counting independent sets in bipartite graphs) (iii) otherwise, for all sufficiently large Δ, it is NP-hard to approximate the number of satisfying assignments of an instance of $\#\mathrm{CSP}_\Delta(\Gamma)$, even within an exponential factor. Our result extends previous results, which apply only in the so-called "conservative" case.

3.33 What can be sampled locally?

Yitong Yin (Nanjing University, CN)

Joint work of Weiming Feng, Yuxin Sun, Yitong Yin

The local computation of Linial [1] and Naor and Stockmeyer [2] asks whether a locally definable distributed computing problem can be solved locally: for a given local CSP whether a CSP solution can be constructed by a distributed algorithm using local information. In this paper, we consider the problem of sampling a uniform CSP solution by distributed algorithms, and ask whether a locally definable joint distribution can be sampled from locally. More broadly, we consider sampling from Gibbs distributions induced by weighted local CSPs in the LOCAL model. We give two Markov chain based distributed algorithms which we believe to represent two fundamental approaches for sampling from Gibbs distributions via distributed algorithms. The first algorithm generically parallelizes the single-site sequential Markov chain by iteratively updating a random independent set of variables in parallel, and achieves an $O(\Delta \log n)$ time upper bound in the LOCAL model, where Δ is the maximum degree, when the Dobrushin's condition for the Gibbs distribution is satisfied. The second algorithm is a novel parallel Markov chain which proposes to update all variables simultaneously yet still guarantees to converge correctly with no bias. It surprisingly parallelizes an intrinsically sequential process: stabilizing to a joint distribution with massive local dependencies, and may achieve an optimal $O(\log n)$ time upper bound independent of the maximum degree Δ under a stronger mixing condition. In addition, we show that almost all nontrivial Gibbs distributions require $\Omega(\log n)$ rounds to sample in the LOCAL model. We also show a strong $\Omega(diam)$ lower bound for sampling independent set in graphs with maximum degree $\Delta \geq 6$. This lower bound holds even when every node is aware of the graph. This gives a strong separation between sampling and constructing locally checkable labelings.

References

1 Nathan Linial. *Distributive graph algorithms Global – solutions from local data*. FOCS, 1987

2 Moni Naor and Larry Stockmeyer. *What can be computed locally?*. STOC, 1993

Participants

- Miriam Backens
University of Bristol, GB
- Eleni Bakali
National Technical University of Athens, GR
- Alexander Barvinok
University of Michigan – Ann Arbor, US
- Ivona Bezáková
Rochester Institute of Technology, US
- Markus Bläser
Universität des Saarlandes, DE
- Antonio Blanca
Georgia Institute of Technology – Atlanta, US
- Cornelius Brand
Universität des Saarlandes, DE
- Andrei A. Bulatov
Simon Fraser University – Burnaby, CA
- Sarah Cannon
Georgia Institute of Technology – Atlanta, US
- Amin Coja-Oghlan
Goethe-Universität – Frankfurt am Main, DE
- Radu Curticapean
Hungarian Academy of Sciences – Budapest, HU
- Holger Dell
Universität des Saarlandes, DE
- Martin Dyer
University of Leeds, GB
- Charis Efthymiou
Goethe-Universität – Frankfurt a. Main, DE
- Jacob Focke
University of Oxford, GB
- Andreas Galanis
University of Oxford, GB
- David Gamarnik
MIT – Camridge, US
- Andreas Göbel
Hasso-Plattner-Institut – Potsdam, DE
- Leslie Ann Goldberg
University of Oxford, GB
- Catherine Greenhill
UNSW Sydney, AU
- Heng Guo
Queen Mary University of London, GB
- Thomas Hayes
University of New Mexico – Albuquerque, US
- Miki Hermann
Ecole Polytechnique – Palaiseau, FR
- Thore Husfeldt
IT University of Copenhagen, DK
- Mark R. Jerrum
Queen Mary University of London, GB
- John Lapinskas
University of Oxford, GB
- Jingcheng Liu
University of California – Berkeley, US
- Kitty Meeks
University of Glasgow, GB
- Sarah Miracle
University of St. Thomas – St. Paul, US
- Viresh Patel
University of Amsterdam, NL
- Will Perkins
University of Birmingham, GB
- Dana Randall
Georgia Institute of Technology – Atlanta, US
- Guus Regts
University of Amsterdam, NL
- David Richerby
University of Oxford, GB
- Marc Roth
Universität des Saarlandes, DE
- Piyush Srivastava
TIFR Mumbai, IN
- Alexandre Stauffer
University of Bath, GB
- Daniel Stefankovic
University of Rochester, US
- Mingji Xia
Chinese Academy of Sciences – Beijing, CN
- Kuan Yang
University of Oxford, GB
- Yitong Yin
Nanjing University, CN
- Chihao Zhang
The Chinese University of Hong Kong, HK
- Stanislav Zivny
University of Oxford, GB

Report from Dagstuhl Seminar 17342

SLEBOK: The Software Language Engineering Body of Knowledge

Edited by

Benoît Combemale[1], Ralf Lämmel[2], and Eric Van Wyk[3]

1 **IRISA – Rennes, FR,** `benoit.combemale@irisa.fr`
2 **Universität Koblenz-Landau, DE,** `laemmel@uni-koblenz.de`
3 **University of Minnesota – Minneapolis, US,** `evw@umn.edu`

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 17342 "SLEBOK: The Software Language Engineering Body of Knowledge". **Software Language Engineering** (SLE) has emerged as a scientific field, with a strong motivation to connect and integrate different research disciplines such as compiler construction, reverse engineering, software transformation, model-driven engineering, and ontologies. This seminar supported further integration of said communities with the clear objective of assembling a **Body of Knowledge** on SLE (SLEBoK). The BoK features artifacts, definitions, methods, techniques, best practices, open challenges, case studies, teaching material, and other components that will afterwards help students, researchers, teachers, and practitioners to learn from, to better leverage, to better contribute to, and to better disseminate the intellectual contributions and practical tools and techniques coming from the SLE field.

Seminar August 20–25, 2017 – http://www.dagstuhl.de/17342
1998 ACM Subject Classification D2.0 Software Engineering: General – Standards, D2.1 Software Engineering: Requirements/Specifications – Languages, D2.3 Software Engineering: Coding Tools and Techniques – Standards
Keywords and phrases body of knowledge, language design and implementation, metaprogramming, software languages
Digital Object Identifier 10.4230/DagRep.7.8.45
Edited in cooperation with Manuel Leduc

1 Executive Summary

Benoît Combemale
Ralf Lämmel
Eric Van Wyk

Overview and Motivation

Over the last 10 years, the field of **Software Language Engineering** (SLE) has emerged based on a strong motivation to connect and integrate different research disciplines such as compiler construction, reverse engineering, software transformation, model-driven engineering, and ontologies. This seminar strives for directly promoting the further integration of said communities with the clear objective of assembling a **Body of Knowledge** on SLE (SLEBoK). The BoK features artefacts, definitions, methods, techniques, best practices, open challenges, case studies, teaching material, and other components that will afterwards help students, researchers, teachers, and practitioners to learn from, to better leverage, to better contribute to, and to better disseminate the intellectual contributions and practical tools and techniques coming from the SLE field.

The following questions and issues provided the guiding principles for the seminar. The first two categories reflect on the result of the past decade and the last category looks forward to the next decade; these categories has been addressed by the seminar attendees in breakout groups.

- **Conceptual model of the SLE field**: What is a comprehensive and objective (validated) classification of SLE approaches? What are appropriate dimensions of such a classification? How to otherwise ontologically organize software language engineering, e.g., in terms of application areas, relationships to other software engineering areas, and fundamental SLE concepts?
- **Curriculum contributions by the SLE field**: What is the suite of formal notions and engineering methods, that one could want to see introduced in a computer science curriculum so that SLE is properly represented? What is a reference curriculum for SLE? What is an appropriate combination of timeless foundations and relevant (current) applications and technologies? How to contribute to or otherwise support a computer science curriculum?
- **Open SLE challenges**: What are the open challenges in SLE and how to lay out a larger research agenda that the community can refer to in the next 10 years? How to connect to important developments such as AI and IoT? How to measure the relevance of the research priorities proposed?

With the SLE field approximately 10 years old, there is a strong support by the community to analyse the situation and to move to the next level of maturity. This Dagstuhl seminar provided the ideal format for such a critical analysis and further development of SLE's foundation. As a result of the work on the above three pillars, the seminar attendees initiated the SLEBOK: **https://github.com/slebok/slebok**.

2 Table of Contents

Executive Summary
Benoît Combemale, Ralf Lämmel, and Eric Van Wyk 45

Working Groups

Reuse and modularity in specifications of software languages
Peter D. Mosses 48

Attribute Grammar working group
Anthony Sloane 48

Software language Survey
Eugene Syriani 49

Practical Guide to Parsing
Jurgen Vinju 50

Curriculum WG
Ralf Lämmel 51

Opinion Pieces

On the need for a SLEBoK
Benoit Combemale 52

Why SLE
Friedrich Steimann 53

Participants 54

3 Working Groups

3.1 Reuse and modularity in specifications of software languages

Peter D. Mosses (TU Delft, NL)

Precise and formal specifications of software languages can be used not only as definitions and documentation, but also for generation of language processing tools. For example, a context-free grammar can define the syntax of a programming language, and can be used to generate a parser for the specified language. Similarly, interpreters or compilers can be generated from semantic specifications.

However, giving and ensuring the correctness of a complete specification of a software language generally requires a significant effort. The required effort can be greatly reduced by reusing (parts of) previous language specifications. For example, one language can embed another one, or languages can extend a base language. In addition, languages evolve, and their specifications need to co-evolve; the new version might reuse parts of the specifications of the old version.

The goal of this working group is to define, distinguish and illustrate the approaches to reuse found in practice. To this end, we investigate examples of reuse, analyse how it is achieved and extract general patterns and practices. In particular, we consider the role of modular structure in connection with reuse.

The working group discussions during the seminar focused on clarifying the relevant concepts and terminology. Oscar Nierstrasz produced a MindMap that reflected the outcome of the discussions. Many of the working group participants contributed references to frameworks and tools in a collaborative WriteMe document during the seminar. The moderator drafted an outline of a more structured document (based on a previously proposed classification of reuse scenarios) at the end of the seminar, and solicited details of examples of reuse in software language specification.

After the seminar, participants of the working group contributed brief (1-page) descriptions of examples of software language specification reuse to the WriteMe document, including references to publications and websites. These examples have subsequently been grouped according to a revised (but still tentative) classification of reuse scenarios. They exploit a wide range of language specification frameworks, including Ecore+ALE, JastAdd, Kiama, Melange, MPS, Neverlang, Object Algebras, Rascal, Silver, and SugarJ. Analysis and discussion of these examples should provide a basis for achieving the stated goal of the working group.

3.2 Attribute Grammar working group

Anthony Sloane (Macquarie University – Sydney, AU)

Attribute grammars were extensively studied from their introduction by Knuth in the late 1960s through a "golden age" particularly in the 1980s and early 1990s. In this period, attribute grammars were applied successfully to problems ranging from programming language semantics to natural language processing. Many different grammar analyses and evaluation strategies were developed.

Since the golden age, interest in attribute grammars in computer science has waned somewhat, yet research progress has continued. Notable since that period are a stronger focus on higher-order attributes, addition of cross-tree references and the use of implicit forwarding for language extension. On-demand evaluation of attributes has taken a prominent position in modern attribute grammar systems. Current attribute grammar research tends to be less about novel notations or evaluation approaches and more about applications.

The aim of this working group is to make recent developments in attribute grammar research accessible to the field as a whole. Within the context of the SLEBOK we aim to produce materials that give an accurate picture of the key attribute grammar results and systems particularly since the golden age. During the seminar we focused on discussions to identify broad themes. Preliminary SLEBOK contributions were developed for attribute grammar terminology and a reader article on the main attribute grammar literature was begun. On-going work will finalise these contributions and work toward a more comprehensive survey article.

3.3 Software language Survey

Eugene Syriani (Uniersity of Montral, CA)

Introduction

During the Dagstuhl seminar 17342 on a software language engineering body of knowledge (SLEBOK), our working group has focused on answering the question: "What is a software language?". After informally discussing within our group and other participants, we could not sense a common agreement. Therefore, we decided that a formal survey needs to be conducted to better answer this question. During the seminar, we conducted a pilot survey to better formulate hypothesis and lead to a formal survey that will eventually be distributed to the SLE community. In the following, we report on the survey conducted during the SLEBOK seminar at Dagstuhl.

Research method

We prepared an online survey using Google Forms. All 25 participants of the seminar participated in the survey. However, the members of our working group were excluded from the final results. Therefore, the total number of participants was 21. The survey is divided into four sections. The first section identified the participant with only his email. The following set of questions served to collect background information on each participant, e.g., expertise related to SLE, community affiliation, and seniority status. The third section presents 64 candidate languages and asks the participant to decide whether it is a software language. The possible answers are yes, no, I don't know, and could be either. The only information available to the participant is either the name of a language, when it is well-known (e.g., XMI, SQL, English), or a one line description of its purpose (e.g., Student course feedback form). These questions were randomized. The final section had one open question for participants to leave any comment. The language questions spanned various categories, such as: API, Encoding formats, Forms and UI, human-oriented languages, metalanguages, modeling languages, programming languages, storage formats, technical domain-specific languages (DSL), etc. We noted what was the most dominant expected answer and the expected agreement rate (50% to 100%). We relied on two metrics. First, we used the raw answer to compute the agreement among participants. We ignore "I don't know" and could be either counted as positive. Second, we assigned a score per answer on a scale of 0 to 2 to compute the deviation from the answers are working group agreed upon.

Results

Overall, there are 76% agreement among participants, 75% agreement per language, and 71% agreement per category. This is considered acceptable according to Cohen's Kappa. We found there was very good consistency among languages within the same category in most cases. This survey identified clearly the following as software languages: domain-specific, programming, description, meta-, and modeling languages. It also identified clearly the following as not being software languages: artificial human, natural, and physics languages. Some language categories could be either but needed more context information to decide: API, constrained strings, Forms, UI, and spreadsheets. There was however no consensus for storage formats, encodings, structured text mechanical, and ontology languages. They depend on the background of the participant. We therefore conclude that a language may be considered a software language depending on the context in which it is used.

3.4 Practical Guide to Parsing

Jurgen Vinju (CWI, NL)

While the books on the topic of parsing cover mostly the theory of parsing algorithms and not the pragmatics of applying these algorithms, at the same time the manuals of particular parsing technologies cover only basic usage patterns and configuration options of the given tools. In between these books and manual extremes lies a knowledge and skill gap:

- which (types of) parsing tool fits the language and language processing task best?
- what design decisions must a language engineer consider when designing a parser?
- how to balance trade-offs between quality and getting-it-done?
- how to balance accuracy and correctness with efficiency and practicality?

The SLE community-at-large represents a considerably body of knowledge on obtaining parsers for software languages; together we have built parsing technologies, used them, evaluated them, improved them and applied them in a big number of projects. Therefore we seem to be in a unique position to fill this perceived gap in parsing literature. It is also noted that Wikipedia is neither complete nor up-to-date in this topic, and we envision contributing to Wikipedia where possible as a side-effect.

The goal of the break-out group at the seminar was to kick-off a concerted effort in creating "A Practical Guide to Parsing" as one of the documents in the wider SLEBOK initiative. The goal of "A Practical Guide to Parsing" is to enable newcomers and experts to get an overview of their own tasks, to enable them to make well-founded design decisions and to build better parsers for the job at hand. To start this process the break-out group brainstormed about the following topics:

- a short list of "personas"; models of people who would use the Guide. The personas serve to guide the writing team into producing actionable descriptions with practical value to the audience.
- a list of relevant parsing terms
- a list of quality dimensions for parsing
- a list of trade-offs between two or more of said dimensions with short descriptions

The results of these brainstorms sessions were captured in a raw document as notes. The notes will be used later to structure the document and the work. Later at the SPLASH conference be completed this brainstorm with a different group of members from the community. The current plan is to rework the large set of notes into an overall structure for the Guide and to assign writing tasks to several volunteering members of the community. It is noted that anybody who contributed will be listed as a co-author of the Guide, where author names are annotated with their specific role in the writing process. The roles are yet to be decided.

3.5 Curriculum WG

Ralf Lämmel (University of Koblenz-Landau)

Summary

Even during the preparation of the Dagstuhl seminar 17342 on the software language engineering body of knowledge (SLEBOK), it was clear that the community needs to take an inventory of the curriculum situation of the SLE field. At the seminar, we formed a WG on the SLE curriculum which focused on analyzing existing courses with SLE relevance on the grounds of semi-structured interviews. The interviewing and coding as well as interpretation or recommendation work is still ongoing, but we summarize the interviewing structure here and hint at some findings.

Assumptions

Programming languages and compiler construction are losing their prominent position in CS/SE/MDE curricula. SLE may combine core aspects from these areas and connect them further with a software engineering orientation and thus may fit well into modern curricula. For instance, SLE has many more traditional and modern use cases other than just a classic compiler. Analyzing actual SLE courses is helping in understanding the SLE body of knowledge, as SLE courses should be linear projections of a de-facto BOK. There is much variation to be expected across different courses because SLE is a developing area and due to preferences of teachers in terms of languages and tools as well as locally imposed constraints.

Interview

We identified 15 courses among the participants of the SLEBOK Dagstuhl. We already performed 7 interviews. The interview structure is based on questions regarding learning objectives, course topics, technological spaces, technologies, languages, actual versus "ideal" title, course workload and relevance of lectures, labs, homework, exam, etc., history and evolution of the course, similarity of the course to other SLE courses, dependencies of this course on others and vice versa, the status of the course to be a choice subject or mandatory, and the use of teaching material. Each interview takes about 10 minutes for the prepared questions. We would typically take another 10 minutes for discussion, also prepared by the general question "What questions had you expected us to ask?".

Preliminary findings

Courses with reliance on model-driven engineering are relatively common. Alternatively, mainstream programming setups or specialized (more or less academic) metaprogramming systems are leveraged in some cases. The full spectrum from Bachelor courses with a larger number of participants and relatively standardized assignments and exams up to research-oriented Master courses with a smaller number of participants and personalized project work exist. The typical course appeals to a software engineering direction and it may be driven by another major theme in software engineering (other than compiler construction, domain-specific languages, or SLE broadly), e.g., software construction or software evolution. In fact, a strong link to compiler construction is not common. Most of the identified courses are already established for 5 years or more. There is enough overlap across all the encountered courses so that some exchange of artifacts (e.g., homework assignments, project topics, and source code) should be worthwhile. The bag of learning objectives and course topics is so diverse and incomparable that some coding would be useful to facilitate better understanding the relationships between the different courses.

4 Opinion Pieces

4.1 On the need for a SLEBoK

Benoit Combemale (IRISA – Rennes, FR)

To address the complexity of modern software-intensive systems, the software engineering community is starting a technology revolution in software development, and the shape of this revolution is becoming more and more clear. Little, domain-specific, software languages (aka. Domain Specific Languages, DSLs) are increasingly being developed to continuously capitalize the domain expertise of various stakeholders, and then used as formalisations of the domains to define relationships among them and support the required integration activities.

This new language-oriented development paradigm is emerging in various guises (e.g., metamodelling, model transformation, generative programming, compilers, etc.), and in various shapes (from API or fluent API, to internal or external DSLs). All these approaches belong to the emerging field of Software Language Engineering (SLE).

The next generation of engineers will have to be fluent in many of these approaches to build the languages needed to implement large-scale and complex software-intensive systems. Software engineers for technical domains, or domain experts for business domains should be able to directly leverage their own experience to improve the efficiency and the quality of the produced software-intensive systems, as well as to support the integration of the various concerns.

While SLE is becoming an everyday reality for software engineers, it is time to ensure that the next generation of engineers has the training and knowledge necessary to synergistically apply all SLE approaches. An SLEBoK is key for collecting and disseminating this knowledge in education, in industry and in academia, and should have a huge impact in the future.

4.2 Why SLE

Friedrich Steimann (Fernuniversität in Hagen, DE)

Paraphrasing James Noble on why programming languages matter [1], I say:
1. Today, software is the most important infrastructure for everything.
2. Software is totally dependent on software languages. Ergo:
3. Software languages are the most important pieces of infrastructure for writing software, and thus the most important meta-infrastructure for everything!

This hijacking of James's thesis I justify with the equation

$$\textit{software languages} = \textit{programming languages} + x$$

which makes my thesis a generalization of James's. However, what is x? A good answer will be needed in order not to be subsumed (or looked down at) by the PL community, which is certainly a role model in terms of scientific standards. In particular, x must be non-negligible, even if viewed from outside the SLE community.

Software Language Engineering adds the engineering perspective to the field, which means we must be able to design software languages that we can guarantee to have certain desired properties. Failing such a guarantee means that someone can be held liable, and will face consequences. If we do not take it that serious, SLE will never be engineering.

References

1 James Noble, "Why Programming Languages Matter", comment to IFIP WG2.16 Programming Language Design mailing list, 17 October 2015 (list accessible at https://lists.csail.mit.edu/mailman/listinfo/pldesign); also paraphrased in Andrew Black's talk "Why Programming Languages Matter" given at SPLASH 2015 and 2016 (which brought it to my attention).

Participants

- Mathieu Acher
University of Rennes, FR
- Anya Helene Bagge
University of Bergen, NO
- Walter Cazzola
University of Milan, IT
- Andrei Chis
feenk – Wabern, CH
- Benoit Combemale
IRISA – Rennes, FR
- Thomas Degueule
CWI – Amsterdam, NL
- Sebastian Erdweg
TU Delft, NL
- Johannes Härtel
Universität Koblenz-Landau, DE
- Görel Hedin
Lund University, SE
- Marcel Heinz
Universität Koblenz-Landau, DE
- Ralf Lämmel
Universität Koblenz-Landau, DE
- Manuel Leduc
IRISA – Rennes, FR
- Tanja Mayerhofer
TU Wien, AT
- Peter D. Mosses
TU Delft, NL
- Gunter Mussbacher
McGill University – Montreal, CA
- Oscar M. Nierstrasz
Universität Bern, CH
- Anthony Sloane
Macquarie University – Sydney, AU
- Friedrich Steimann
Fernuniversität in Hagen, DE
- Eugene Syriani
University of Montréal, CA
- Tijs van der Storm
CWI – Amsterdam, NL
- Eric Van Wyk
University of Minnesota – Minneapolis, US
- Hans Vangheluwe
University of Antwerp, BE
- Jurgen J. Vinju
CWI – Amsterdam, NL
- Markus Völter
Völter Ingenieurbüro – Stuttgart, DE
- Vadim Zaytsev
RainCode – Brussels, BE

Report from Dagstuhl Seminar 17351

Machine Learning and Formal Methods

Edited by

Sanjit A. Seshia[1], Xiaojin (Jerry) Zhu[2], Andreas Krause[3], and Susmit Jha[4]

1 **University of California, Berkeley** sseshia@eecs.berkeley.edu
2 **University of Wisconsin, Madison** jerryzhu@wisc.edu
3 **ETH Zürich** krausea@ethz.ch
3 **SRI International** susmit.jha@sri.com

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 17351 "Machine Learning and Formal Methods". The seminar brought together practitioners and reseachers in machine learning and related areas (such as robotics) with those working in formal methods and related areas (such as programming languages and control theory). The meeting highlighted the connections between the two disciplines, and created new links between the two research communities.

Seminar August 28–September 01, 2017 – http://www.dagstuhl.de/17351
1998 ACM Subject Classification B.3.3 TBD Performance Analysis and Design Aids, B.5.1 TBD Design, C.1.2 TBD Multiple Data Stream Architectures (Multiprocessors), D.1.3 TBD Concurrent Programming
Keywords and phrases Formal Methods, Machine Learning
Digital Object Identifier 10.4230/DagRep.7.8.55

1 Executive Summary

Sanjit A. Seshia (University of California, Berkeley,
Xiaojin (Jerry) Zhu (University of Wisconsin, Madison)
Andreas Krause (ETH Zürich)
Susmit Jha (SRI International)

The seminar was successful in bringing the following two communities together:
- The community that works on machine learning (ML), both on theoretical topics and on applications to areas such as robotics and cyber-physical systems, and
- The community that works on formal methods (FM), both on computational proof techniques and on applications to formal verification and program/controller synthesis.

Both communities have long and vibrant histories, with associated conferences and journals. However, they have rarely intersected. The machine learning community has traditionally focused on *inductive* learning from data, with the data set considered as partial (potentially noisy) observations of some phenomenon. The formal methods community has traditionally emphasized automated *deduction*, e.g., using theorem proving or model checking, as a core reasoning method, with a heavy emphasis placed on formal models and proofs of correctness using those models. However, recent ideas and methods have appeared that demonstrate new connections between the two disciplines, which suggested that the time is ripe for a meeting to promote cross-fertilization between the areas at a deep technical level. This seminar has been a significant step forward to bring the two communities together.

More concretely, the Seminar and the interaction it facilitated has brought three kinds of benefits. First, formal methods can benefit from a more effective use of machine learning techniques particularly in the context of automated synthesis. Similarly, the increasing use of machine learning in applications that require a high level of assurance points to the need for integration with formal methods. However, the potential synergies between the two areas go beyond a simple application of the techniques in one area to the other area. Importantly, there is new fundamental science to be explored in the intersection of machine learning and formal methods, related to the the confluence of inductive and deductive reasoning, and which can inform a range of new industrially-relevant applications as well.

The seminar had about 40 participants from both the FM and ML communities. The organizers took several steps to foster discussion and cross-pollination of ideas between the two communities, including the following:

- The seminar began with a day of tutorials: a half-day tutorial on Machine Learning for Formal Methods participants, and a half-day tutorial on Formal Methods for a Machine Learning audience. These tutorials helped to establish a common vocabulary to discuss ideas, problems and solutions.
- Sessions were organized based on themes that emerged in discussions before the seminar and during the first day. The list of session topics is as follows:
  1. Probabilistic Programming
  2. Teaching and Oracle-Guided Synthesis
  3. Safe Learning-Based Control
  4. Probabilistic Program Analysis
  5. Adversarial Analysis and Repair for Machine Learning
  6. Inductive Synthesis and Learning
  7. Machine Learning for Theorem Proving and Optimization
  8. Explainable and Interpretable Machine Learning
  9. Deep Learning and Verification/Synthesis

  In organizing these sessions, the organizers tried to combine speakers from both ML and FM areas to foster discussion and comparison of approaches.
- Seating arrangements at meals were organized so that (a) each table had an approximately equal number of participants from both communities, and (b) the seating was randomly changed from meal to meal.
- A joint session was organized with the concurrent seminar on analysis and synthesis of floating-point programs. This session had a panel discussion on floating-point issues in machine learning programs.

After the seminar, we have heard positive feedback from multiple participants. One told us that he started a new research project as a direct result of the seminar. A group of participants are planning to continue the interaction via joint workshops at major venues of both communities such as CAV, PLDI, ICML, NIPS, etc.

2 Table of Contents

Executive Summary
Sanjit A. Seshia, Xiaojin (Jerry) Zhu, Andreas Krause and Susmit Jha 55

Overview of Talks

Reachability and regularity for Markov chains
S Akshay 59

A Non-monotonic Theory of Oracle-guided Inductive Synthesis
Dalal Alrajeh 59

Semantic Mapping and Mission Planning in Robotics
Nikolay A. Atanasov 61

Safe learning of regions of attractions
Felix Berkenkamp 61

Polynomial Inference of Equational Theories
Ben Caulfield 61

Constraint Learning and Dynamic Probabilistic Programming
Luc De Raedt 62

Logical Clustering for Time-Series Data
Jyotirmoy Deshmukh 62

Query Learning of Regular Languages and Richer Formalisms
Dana Fisman 63

Learning Invariants using Decision Trees and Implication Counterexamples
Pranav Garg 63

Explaining AI Decisions Using Sparse Boolean Formula
Susmit Jha 63

Formal Guarantees on the Robustness of a Classifier against Adversarial Manipulation
Matthias Hein 64

Syntax-Guided Synthesis
Rajeev Alur 64

Formal Inductive Synthesis/Oracle-Guided Inductive Synthesis
Sanjit A. Seshia 65

Combining Logical and Probabilistic Reasoning in Program Analysis
Mayur Naik 65

Data-driven Program Synthesis From Examples
Nagarajan Natarajan 66

Leveraging computational models of human cognition for educational interventions
Anna Rafferty 66

Planning for Cars that Coordinate with People
Dorsa Sadigh 67

Deduction and Induction – A Match Made in Heaven
Stephan Schulz 67

Neural Program Synthesis
Rishabh Singh 68

Learning Continuous Semantic Representations of Symbolic Expressions
Charles Sutton 68

Some ML Tasks in Theorem Proving
Josef Urban 68

PSI: Exact Inference for Probabilistic Programs
Martin Vechev 69

Learning from RNNs
Eran Yahav 69

Synthesis with Abstract Examples
Eran Yahav 69

Smooth Imitation Learning
Yisong Yue 70

Preference-Based Teaching
Sandra Zilles 70

Falsification of Cyber-Physical Systems with Machine Learning Components
Sanjit A. Seshia 70

Breakout Groups 71

Participants 73

3 Overview of Talks

3.1 Reachability and regularity for Markov chains

S Akshay (IIT, Bombay – Mumbai, India)

The dynamic behavior of a Markov chain – a basic model for probabilistic systems – can be described using sequences of probability distributions starting from an initial set of distributions. Our goal is to study the patterns and trajectories formed by such sequences during the evolution of a Markov chain over time.

A basic reachability problem which lies at the heart of this question is: does there exist an integer n such that the probability to reach in n steps from a given state to another in a Markov chain is exactly some pre-specified value r. Surprisingly, it turns out that this problem is already as hard as the Skolem Problem: a number-theoretic decision problem whose decidability has been open for decades. We look at some approximate variants as well as restrictions which allow us to reason about the trajectories in an automata-theoretic framework. If time permits, we will also draw links between these problems and classical problems such as program termination, as well as explore links to POMDP setting.

3.2 A Non-monotonic Theory of Oracle-guided Inductive Synthesis

Dalal Alrajeh (Imperial College London, UK, dalal.alrajeh@ic.ac.uk)

Specifications provide significant aid in the formal analysis of software supporting tasks such as consistency management, system verification, program synthesis, program repair and software maintenance. However writing such specifications is difficult and time-consuming. Several approaches have been proposed for automatically generating complete specifications from abstract descriptions (such as UML diagrams and user requirements) which mainly differ in their method of computation, e.g., refinement operators [7], patterns library [6]. Recent years have seen the emergence of techniques based on inductive learning, for instance [4]. The input to these techniques are samples classified as either positive or negative examples. The aim is to learn a specification that is consistent with all positive examples and inconsistent with all the negative ones. However, the quality of a learnt specification (and its proximity to the target specification) heavily depends on the quality of the samples provided to the learner. Oracle-guided inductive synthesis (OGIS) is a class of approaches that restrict the set of samples for learning to those directly relevant to some target specification [5]. The learner can query an oracle (e.g., a user or verifier) for examples. The oracle in return responds with positive or negative example that is intended to guide the learner's search for candidate specifications. The oracle is also tasked with determining whether the learner has found the correct specification. Each time a negative (resp. positive) example that is consistent (resp. inconsistent) with the current candidate specification is given, the learner proceeds in one of the following directions: (1) the candidate specification is discarded and a new one is synthesized from the set of examples accumulated thus far; or (2) an additional specification is synthesized from the new example and added to the previous ones; we say in the second case a specification has been refined. Both directions may lead to learning a correct specification (consistent with all positive examples but none of the negatives). We argue that the former, from a practical perspective, requires users to abandon any development activities they may have started based on the earlier specification. From a conceptual view, it violates the principle of elaboration tolerance [2, 8]. The applicability of the latter, however, depends on the kind of inference allowed by the learner. Typical OGIS instances that follow this direction assume monotonicity of the synthesis procedure. By this, it is guaranteed that candidate specifications generated from new examples are consistent with those of previous iterations. However, this guarantee does not hold in the case of non-monotonic learners. In this work, we conduct a formal investigation into properties of oracleguided inductive synthesis for specification refinement by examining the impact of using two types of learners: monotonic and non-monotonic [3]. In monotonic learning, inferences cannot be invalidated simply by adding new expressions to a specification. Non-monotonic learning, on the other hand, allows inferences to be made provisionally which can be retracted as new information becomes available and thus specifications are extended. Our investigation seeks to answer the following questions: Does the quality of a specification improve with a non-monotonic learner? What are the termination guarantees with nonmonotonic learner? In cases where termination is guaranteed, does the use of a non-monotonic learner improve the speed of termination? In answering these, we assume an oracle that defines a fixed ordering over the examples generated and seek to understand the influence of the following factors: (i) the types of queries submitted to an oracle (e.g., correctness and whether they provide both positive and negative examples or positive witness only); and (ii) the resources available to the learner: finite versus infinite memory. We direct our analysis to a particular instance of OGIS for synthesizing target specifications in Linear Temporal Logic (LTL) [9]. The quality of a specification is measured in terms of: the size of the formulas, and the size of the language defined by the formulas. For monotonic learning, we have developed a monotonic learner that can compute properties in tight Signal Temporal Logic (a flavor of LTL over continuous signals) from positive examples only. As for non-monotonic learning, we consider the approach described in [1] which provides a transformation function from a subclass of LTL to a non-monotonic logic and vice-versa.

References

1 Alrajeh, D., Ray, O., Russo, A., Uchitel, S.: Using abduction and induction for operational requirements elaboration. J. Applied Logic 7(3), 275–288 (2009)
2 Baral, C., Zhao, J.: Non-monotonic temporal logics for goal specification. In: Proceedings of IJCAI07,. pp. 236–242 (2007)
3 Frankish, K.: Non-monotonic inference. In: Barber, A. (ed.) Encyclopedia of Language and Linguistics. Elsevier (2005)
4 Gehr, T., Dimitrov, D., Vechev, M.: Learning commutativity specifications. In: Proceedings of CAV15. pp. 307–323 (2015)
5 Jha, S., Seshia, S.: A theory of formal synthesis via inductive learning. Acta Informatica (2017)
6 van Lamsweerde, A.: Requirements Engineering – From System Goals to UML Models to Software Specifications. Wiley (2009)
7 Li, F.L., Horkoff, J., Borgida, A., Guizzardi, G., Liu, L., Mylopoulos, J.: From stakeholder requirements to formal specifications through refinement. In: Proceedings of REFSQ15 (2015)
8 McCarthy, J.: The artificial intelligence debate: False starts, real foundations. chap. Mathematical Logic in Artificial Intelligence, pp. 297–311. MIT Press (1988)
9 Pnueli, A.: The temporal logic of programs. In: Proceedings of SFCS77. pp. 46–57 (1977)

3.3 Semantic Mapping and Mission Planning in Robotics

Nikolay A. Atanasov (University of California at San Diego, US)

Recent years have seen impressive progress in robot perception leading to real-time super-human object recognition. Surprisingly, however, most existing approaches to simultaneous localization and mapping (SLAM) in robotics rely on low-level geometric features and do not take advantage of semantic information provided by object recognition methods. In this talk, I will address the semantic SLAM problem which utilizes object identity information for loop closure and construction of meaningful maps. A major contribution of our approach is in proving that the complexity of incorporating probabilistic data association is equivalent to computing the permanent of a suitable matrix. The resulting probabilistic semantic maps allow us to specify complex robot missions as temporal logic constraints over objects in the environment.

3.4 Safe learning of regions of attractions

Felix Berkenkamp (ETH Zürich, CH)

Reinforcement learning is a powerful paradigm for learning optimal policies from experimental data. However, to find optimal policies, most reinforcement learning algorithms explore all possible actions, which may be harmful for real-world systems. As a consequence, learning algorithms are rarely applied on safety-critical systems in the real world. In this paper, we present a learning algorithm that explicitly considers safety in terms of stability guarantees. Specifically, we extend control theoretic results on Lyapunov stability verification and show how to use statistical models of the dynamics to obtain high-performance control policies with provable stability certificates. Moreover, under additional regularity assumptions in terms of a Gaussian process prior, we prove that one can effectively and safely collect data in order to learn about the dynamics and thus both improve control performance and expand the safe region of the state space. In our experiments, we show how the resulting algorithm can safely optimize a neural network policy on a simulated inverted pendulum, without the pendulum ever falling down.

3.5 Polynomial Inference of Equational Theories

Ben Caulfield (University of California – Berkeley, US)

Equational logic is a formalism used to describe infinite sets of equations between terms (theories) using finite sets of equations (presentations). Both functional programs and logic programs can be naturally represented as equational logic presentations. Therefore, learning presentations in equational logic can be seen as a form of program synthesis. In general, it is undecidable whether terms are equivalent via a finite presentation. So learning equational presentations is generally intractable.

We investigate the exact learning of a restricted class of theories known as non-collapsing shallow theories, which can be presented by equations where variables only appear at depth one. The learning algorithms use examples and queries of equations between ground terms, meaning there are no variables in the equations. It is shown that these theories cannot be learned in the limit from only positive examples. A polynomial time algorithm is given which creates a hypothesis presentation consistent with positive and negative examples which will learn in the limit a presentation for the target theory. Finally, an algorithm is given which learns a presentation in polynomial time from a minimally adequate teacher. It is shown that the learned presentations are canonical with respect to an ordering on terms and the presentation is at most polynomially larger than the minimal presentation of the same theory.

3.6 Constraint Learning and Dynamic Probabilistic Programming

Luc De Raedt (KU Leuven, BE)

The talk focussed on two issues. The first was the synthesis of a set of constraints that hold in tabular data (say a set of excel tables). The second was concerned with the use of hybrid probabilistic programs in dynamic domains and its applications in simple robotics and planning settings.

More details on our probabilistic programming language Problog can be found at https://dtai.cs.kuleuven.be/problog/.

3.7 Logical Clustering for Time-Series Data

Jyotirmoy Deshmukh (USC – Los Angeles, US)

Techniques for unsupervised learning for signals (time-series data) typically use distance metrics on the signal space to perform clustering. Clusters obtained in this fashion group qualitatively similar signal shapes, but may not respect logical properties of signals and may not be easy to interpret and explain. We propose a new paradigm of logical clustering, where we use parametric temporal logic formulas as a feature extraction mechanism, and then use off-the-shelf machine learning tools to automatically cluster similar traces with respect to the "projection" of the traces in the parameter-space. We demonstrate how this technique produces interpretable formulas that are amenable to analysis and understanding using a few representative examples.

3.8 Query Learning of Regular Languages and Richer Formalisms

Dana Fisman (Ben Gurion University – Beer Sheva, IL)

We review results on Angluin style query learning of automata. In particular for the concept classes of regular languages (via succinct representations such as alternating finite automata), regular omega-languages, regular omega-tree-languages, weighted languages, I/O languages, and languages over infinite alphabets.

3.9 Learning Invariants using Decision Trees and Implication Counterexamples

Pranav Garg (Amazon – Bangalore, IN)

We introduce ICE, a robust learning paradigm for synthesizing invariants, that learns using positive, negative and implication counter-examples, and show that it admits honest teachers and strongly convergent mechanisms for invariant synthesis. We propose the first learning algorithms in this model with implication counter-examples that are based on machine learning techniques. In particular, we extend classical decision-tree learning algorithms in machine learning to handle implication samples, building new scalable ways to construct small decision trees using statistical measures. We also develop a decision-tree learning algorithm in this model that is guaranteed to converge to the right concept (invariant) if one exists. We implement the learners and an appropriate teacher, and show that the resulting invariant synthesis is efficient and convergent for a large suite of programs.

3.10 Explaining AI Decisions Using Sparse Boolean Formula

Susmit Jha (SRI International – Menlo Park, CA, susmit.jha@sri.com)

This talk describes the problem of learning Boolean formulae from examples obtained by actively querying an oracle that can label examples as positive or negative. This problem has received attention in machine learning as well as formal methods, and it has been shown to have exponential worst-case complexity in the general case as well as for many restrictions. In this talk, we focus on learning sparse Boolean formulae which depend on only a small (but unknown) subset of the overall vocabulary of atomic propositions. We propose an efficient algorithm to learn these sparse Boolean formulae with a given confidence. This assumption of sparsity is motivated by the problem of mining explanations for decisions made by artificially intelligent algorithms, where the explanation of individual decisions may depend on a small but unknown subset of all the inputs to the algorithm. We demonstrate the use of proposed learning algorithm to automatically generate explanations of these decisions. These explanations will make intelligent systems more understandable and accountable to human users, facilitate easier audits and provide diagnostic information in the case of failure. The proposed approach treats the AI algorithm as a black-box oracle, and is therefore, broadly applicable and agnostic to the specific AI algorithm. We illustrate the practical effectiveness of our approach on a set of diverse case-studies.

3.11 Formal Guarantees on the Robustness of a Classifier against Adversarial Manipulation

Matthias Hein (Universität des Saarlandes, DE)

Recent work has shown that state-of-the-art classifiers are quite brittle, in the sense that a small adversarial change of an originally with high confidence correctly classified input leads to a wrong classification again with high confidence. This raises concerns that such classifiers are vulnerable to attacks and calls into question their usage in safety-critical systems. We show in this talk formal guarantees on the robustness of a classifier by giving instance-specific lower bounds on the norm of the input manipulation required to change the classifier decision. Based on this analysis we propose the Cross-Lipschitz regularization functional. We show that using this form of regularization in kernel methods resp. neural networks improves the robustness of the classifier without any loss in prediction performance.

3.12 Syntax-Guided Synthesis

Rajeev Alur (University of Pennsylvania – Philadelphia, US)

The classical synthesis problem is to find a program or a system that meets a correctness specification given as a logical formula. Recent work on synthesis and optimization illustrates many potential benefits of allowing the user to supplement the logical specification with a syntactic template that constrains the space of allowed implementations. The formulation of the syntax-guided synthesis problem (SyGuS) is aimed at standardizing the core computational problem common to these proposals in a logical framework. The input to the SyGuS problem consists of a background theory, a semantic correctness specification for the desired program given by a logical formula, and a syntactic set of candidate implementations given by a grammar. The computational problem then is to find an implementation from the set of candidate expressions so that it satisfies the specification in the given theory.

In this tutorial we first describe how a wide range of problems such as automatic synthesis of loop invariants, program optimization, program repair to defend against timing-based attacks, and learning programs from examples can be formalized as SyGuS instances. We then describe the counterexample-guided-inductive-synthesis (CEGIS) strategy for solving the SyGuS problem. Finally we discuss our efforts over the past three years on defining the standardized interchange format built on top of SMT-LIB, repository of benchmarks from diverse applications, organization of the annual competition, SyGuS-COMP, of solvers, and experimental evaluation of solution strategies.

3.13 Formal Inductive Synthesis/Oracle-Guided Inductive Synthesis

Sanjit A. Seshia (UC Berkeley – Berkeley, CA, sseshia@berkeley.edu)

Joint work of Susmit Jha, Sanjit A. Seshia
Main reference Susmit Jha, Sanjit A. Seshia: "A theory of formal synthesis via inductive learning", Acta Inf., Vol. 54(7), pp. 693–726, 2017.
URL http://dx.doi.org/10.1007/s00236-017-0294-5

This talk presents a theoretical framework for synthesis from examples so as to satisfy a formal specification, termed as formal inductive synthesis. We discuss how formal inductive synthesis differs from traditional machine learning. We then describe oracle-guided inductive synthesis (OGIS), a framework that captures a family of synthesizers that operate by iteratively querying an oracle. An instance of OGIS that has had much practical impact is counterexample-guided inductive synthesis (CEGIS). We present a theoretical characterization of CEGIS for both finite and infinite program classes.

Details are given in [1, 2].

References

1 Susmit Jha, Sanjit A. Seshia. A Theory of Formal Synthesis via Inductive Learning. CoRR abs/1505.03953 (2015)

2 Susmit Jha, Sanjit A. Seshia. A theory of formal synthesis via inductive learning. Acta Inf. 54(7): 693-726 (2017)

3.14 Combining Logical and Probabilistic Reasoning in Program Analysis

Mayur Naik (University of Pennsylvania – Philadelphia, US)

Existing program analyses are expressed in the form of logical rules that are handcrafted by experts. While this logic-based approach has many benefits, however, it cannot handle uncertainty and lacks the ability to learn and adapt. This in turn hinders the accuracy, scalability, and usability of program analysis tools in practice.

We present a methodology and framework to incorporate probabilistic reasoning into existing program analyses that are based on logical reasoning. The framework comprises a front-end, which automatically integrates probabilities into a logical analysis by synthesizing a system of weighted constraints, and a back-end, which is a learning and inference engine for such constraints. We demonstrate how this approach advances three important applications of program analysis: automated verification, interactive verification, and bug detection. We will describe new algorithmic techniques to solve very large instances of weighted constraints that arise not only in our domain but also in other domains such as Big Data analytics and statistical AI.

3.15 Data-driven Program Synthesis From Examples

Nagarajan Natarajan (Microsoft Research India – Bangalore, IN)

A typical data analyst is often faced with tasks involving data formatting, and in fact studies say that more than 80% of the time is spent wrangling the data, before performing scientific analysis. Automatically synthesizing code pieces that can perform such tasks is a boon. There is a long line of work in the Programming Languages community on program synthesis from examples, and more recently, Machine Learning community has gained interest in the problem space. In this talk, I will outline some of the challenges faced when formulating a machine learning problem in this setting, and present our data-driven solution for program synthesis that builds on the successful PROSE framework. Results on benchmark datasets involving a fairly sophisticated text grammar show that, in most cases, with just one input-output example, the system can surface the "right" program at the top. I will also give a short demo of the deployed solution.

3.16 Leveraging computational models of human cognition for educational interventions

Anna Rafferty (Carleton College – Northfield, US)

Educational technologies offer the opportunity to provide personalized guidance or instruction to individual learners. Often, this is achieved by tracking what a learner knows based on her behavior in the system. This tracking can be challenging, however, especially in cases where learners' behaviors cannot easily be evaluated. For example, learners make a large number of choices in games or interactive simulations that should provide information about understanding, but it may not be possible to automatically mark each choice as correct or incorrect or easily relate the choices to knowledge in a domain. My approach is to leverage computational modeling and machine learning tools to develop general frameworks that can be applied to individual educational technologies. We have developed a Bayesian inverse planning framework to make fine-grained inferences about understanding based on learners' sequences of actions, demonstrating how a formal model of human behavior can lead to methods that are applicable across different educational domains. We have applied this framework to assess learners' understanding based on actions in game-based experiments in the lab, choices in a microbiology game played by middle schoolers, and step-by-step solutions to algebraic equations. Because the framework includes a generative model of human behavior, we have applied a variation of it to design more diagnostic assessments, both for algebra and for game-based experiments.

References

1 Rafferty, A. N., LaMar, M. M. and Griffiths, T. L. (2015), Inferring Learners' Knowledge From Their Actions. Cogn Sci, 39: 584–618. doi:10.1111/cogs.12157

3.17 Planning for Cars that Coordinate with People

Dorsa Sadigh (Stanford University, US)

As human-robot systems make their ways into our every day life, safety has become a core concern of the learning algorithms used by such systems. Examples include semi-autonomous vehicles such as automobiles and aircraft. The robustness of controllers in such systems relies on the accuracy of models of human behavior. In this talk, we propose a systematic methodology for analyzing the robustness of learning-based control of human-robot systems. We focus on the setting where human models are learned from data, with humans modeled as approximately rational agents optimizing their reward functions. In this setting, we provide a novel optimization-driven approach to find small deviations in learned human behavior that lead to violation of desired (safety) objectives.

3.18 Deduction and Induction – A Match Made in Heaven

Stephan Schulz (Duale Hochschule Baden-Württemberg – Stuttgart, DE)

First-order theorem provers search for proofs of a conjecture in an infinite and highly branching search space. This search critically depends on good heuristics. Unfortunately, designing good heuristics for the different choice points and classes of problems has proved to be very hard. Indeed, even classification of proof problems into classes with similar search behaviour is a largely open research question.

One way to address the difficulty of controlling search is to use inductive approaches, i.e. to try to find good heuristics by observing and generalizing examples of successful and failing proof searches. Here we discuss the three major choice points for superposition-based provers, and how good heuristics can be found via inductive processes.

We distinguish two different learning paradigms. In the first case, only the performance of different heuristics on a test set is used as input for the inductive process. Two examples for this are the automatic generation of automatic modes for theorem provers, and the improvement of clause evaluation heuristics via parameter optimization or genetic algorithms. The second paradigm not only considers the performance of a heuristic, but tries to find new heuristics by analyzing proofs, or, more generally, a graph of the proof search.

We give some results on established work, and discuss some preliminary progress and open questions from ongoing work.

3.19 Neural Program Synthesis

Rishabh Singh (Microsoft Research – Redmond, US)

The key to attaining general artificial intelligence is to develop architectures that are capable of learning complex algorithmic behaviors modeled as programs. The ability to learn programs allows these architectures to learn to compose high-level abstractions with complex control-flow, which can lead to many potential benefits: i) enable neural architectures to perform more complex tasks, ii) learn interpretable representations (programs which can be analyzed, debugged, or modified), and iii) better generalization to new inputs (like algorithms). In this talk, I will present some of our recent work in developing neural architectures for learning complex regular-expression based data transformation programs from input-output examples, and will also briefly discuss some other applications such as program repair and optimization that can benefit from learning neural program representations.

3.20 Learning Continuous Semantic Representations of Symbolic Expressions

Charles Sutton (University of Edinburgh, GB)

Combining abstract, symbolic reasoning with continuous neural reasoning is a grand challenge of representation learning. As a step in this direction, we propose a new architecture, called neural equivalence networks, for the problem of learning continuous semantic representations of algebraic and logical expressions. These networks are trained to represent semantic equivalence, even of expressions that are syntactically very different. The challenge is that semantic representations must be computed in a syntax-directed manner, because semantics is compositional, but at the same time, small changes in syntax can lead to very large changes in semantics, which can be difficult for continuous neural architectures. We perform an exhaustive evaluation on the task of checking equivalence on a highly diverse class of symbolic algebraic and boolean expression types, showing that our model significantly outperforms existing architectures.

3.21 Some ML Tasks in Theorem Proving

Josef Urban(Czech Technical University – Prague, CZ)

The talk will show several examples of how machine learning is today used in automated and interactive theorem proving tasks. I will also briefly mention the various theorem proving fields, recent developments in them, and the collaboration between interactive and automated provers.

3.22 PSI: Exact Inference for Probabilistic Programs

Martin Vechev (ETH Zürich, CH)

This talk introduces probabilistic programming and discusses exact inference with the PSI solver (psisolver.org) based on symbolic reasoning. PSI supports higher order functions, nested inference, discrete, continuous and mixed distributions and comes with its own symbolic integration engine. The talk also discusses various open problems in the area.

3.23 Learning from RNNs

Eran Yahav (Technion – Haifa, IL)

We address the problem of extracting an automaton from a given recurrent neural network (RNN). We present a novel algorithm that uses exact learning and abstract interpretation to perform efficient extraction of a minimal automaton describing the internal states of the given RNN. We use Angluin's L* algorithm as a learner and the given RNN as an oracle, employing abstract interpretation of the RNN for answering equivalence queries. Our technique allows automaton-extraction from the RNN while avoiding state explosion, even when very fine differentiation is required between RNN states.

3.24 Synthesis with Abstract Examples

Eran Yahav (Technion – Haifa, IL)

Interactive program synthesizers enable a user to communicate his/her intent via input-output examples. Unfortunately, such synthesizers only guarantee that the synthesized program is correct on the provided examples. A user that wishes to guarantee correctness for all possible inputs has to manually inspect the synthesized program, an error-prone and challenging task. We present a novel synthesis framework that communicates only through (abstract) examples and guarantees that the synthesized program is correct on all inputs. The main idea is to use abstract examples—a new form of examples that represent a potentially unbounded set of concrete examples. An abstract example captures how part of the input space is mapped to corresponding outputs by the synthesized program. Our framework uses a generalization algorithm to compute abstract examples which are then presented to the user. The user can accept an abstract example, or provide a counterexample in which case the synthesizer will explore a different program. When the user accepts a set of abstract examples that covers the entire input space, the synthesis process is completed. We have implemented our approach and we experimentally show that our synthesizer communicates with the user effectively by presenting on average 3 abstract examples until the user rejects false candidate programs. Further, we show that a synthesizer that prunes the program space based on the abstract examples reduces the overall number of required concrete examples in up to 96% of the cases.

3.25 Smooth Imitation Learning

Yisong Yue (California Institute of Technology – Pasadena, US)

We study the problem of smooth imitation learning for online sequence prediction, where the goal is to train a policy that can smoothly imitate demonstrated behavior in a dynamic and continuous environment in response to online, sequential context input. We present an approach that combines smooth model-based controllers with black-box machine learning approaches in order to obtain flexible function classes that are regularized to ensure smooth dynamics.

References

1 Learning Online Smooth Predictors for Real-time Camera Planning using Recurrent Decision Trees. Jianhui Chen, Hoang M. Le, Peter Carr, Yisong Yue, James J. Little. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2016

3.26 Preference-Based Teaching

Sandra Zilles (University of Regina, CA)

The classical teaching dimension model, used to describe the sample complexity of learning from helpful teachers, assumes that the learner simply produces any hypothesis consistent with the data provided by the teacher. Assuming instead that the learner expects data to be chosen in a helpful way improves the sample complexity. One model in this context is Preference-Based Teaching, in which the learner uses a preference relation over concepts and always hypothesizes a most preferred concept consistent with the data. In the case of finite concept classes, the corresponding sample complexity appears to be related to the VC-dimension. Some open problems and potential connections to Formal Methods are discussed.

3.27 Falsification of Cyber-Physical Systems with Machine Learning Components

Sanjit A. Seshia (UC Berkeley – Berkeley, CA, sseshia@berkeley.edu)

Joint work of Tommaso Dreossi, Alexandre Donze, Sanjit A. Seshia
Main reference Tommaso Dreossi, Alexandre Donzé, Sanjit A. Seshia: "Compositional Falsification of Cyber-Physical Systems with Machine Learning Components", in Proc. of the NASA Formal Methods - 9th International Symposium, NFM 2017, Moffett Field, CA, USA, May 16-18, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10227, pp. 357–372, 2017.
URL http://dx.doi.org/10.1007/978-3-319-57288-8_26

Cyber-physical systems (CPS), such as automotive systems, are starting to include sophisticated machine learning (ML) components. Their correctness, therefore, depends on properties of the inner ML modules. While learning algorithms aim to generalize from examples, they are only as good as the examples provided, and recent efforts have shown that they can produce inconsistent output under small adversarial perturbations. This raises the question: can the output from learning components can lead to a failure of the entire CPS? In this work, we address this question by formulating it as a problem of falsifying signal temporal logic (STL) specifications for CPS with ML components. We propose a compositional falsification framework where a temporal logic falsifier and a machine learning analyzer cooperate with the aim of finding falsifying executions of the considered model. The efficacy of the proposed technique is shown on an automatic emergency braking system model with a perception component based on deep neural networks.

Details of this contribution are given in [2], and a broader perspective of applying formal methods to the goal of verified artificial intelligence is discussed in [1].

References

1 Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry: Towards Verified Artificial Intelligence. CoRR abs/1606.08514 (2016)

2 Tommaso Dreossi, Alexandre Donze, and Sanjit A. Seshia: Compositional Falsification of Cyber-Physical Systems with Machine Learning Components. NFM 2017, pages 357-372.

4 Breakout Groups

On the final two days of the seminar, multiple breakout sessions were organized on the following topics:

1. Machine Teaching and Oracle-Guided Synthesis
2. Solvers in Formal Methods and Machine Learning
3. Formal Guarantees for Machine Learning Systems

We present here a very brief summary of each of the breakout topics.

Machine Teaching and Oracle-Guided Synthesis

In principle, all of the work in machine learning and inductive synthesis (synthesis from examples) can be explained using an oracle-guided framework. In particular, there is a broad spectrum varying from passive learning on the one side to machine teaching on the other, with active learning, query-based learning, and counterexample-guided inductive synthesis in between. The breakout sessions on this topic discussed a common terminology to connect the formal methods and machine learning communities, and the role the framework of oracle-guided inductive synthesis can play in connecting the ideas in the two communities. The formal methods community can provide interesting sources of problems for theoretical work on machine teaching and query-based learning. Likewise, the machine learning community can provide theory and techniques to prove results about the power of oracle-guided synthesis.

Solvers in Formal Methods and Machine Learning

In Formal Methods, solvers based on SAT, SMT, BDDs, etc. play an important role. The discussions in these breakout sessions discussed how machine learning is already playing an important role in the development of these solvers as well as interactive theorem provers, such as, e.g., in portfolio SAT solving techniques. Further, the role of solvers to improve machine learning was also discussed. For example, verification tools could be used to generate more training data, such as the work on falsification of CPS with machine learning components discussed earlier in the seminar. The importance of choosing good representations shows up in the work of both communities. On the frontier of solver research, the topic of model counting and volume estimation is of high relevance to both communities.

Formal Guarantees for Machine Learning

The different forms of guarantees for machine learning systems were discussed, as well as the role of formal specifications. Topics ranging from Boolean versus quantitative specifications, worst-case versus asymptotic versus probabilistic guarantees, and which guarantees permit compositional reasoning. Formal reasoning can be applied to learning algorithms, learned models, and training data. Interpretability is an important objective. The issue of specifying objectives using logical formulas or cost functions was also discussed.

Participants

- S. Akshay
Indian Institute of Technology – Mumbai, IN
- Dalal Alrajeh
Imperial College London, GB
- Rajeev Alur
University of Pennsylvania – Philadelphia, US
- Nikolay A. Atanasov
University of California at San Diego, US
- Ammar Ben Khadra
TU Kaiserslautern, DE
- Felix Berkenkamp
ETH Zürich, CH
- Ben Caulfield
University of California – Berkeley, US
- Swarat Chaudhuri
Rice University – Houston, US
- Luc De Raedt
KU Leuven, BE
- Jyotirmoy Deshmukh
USC – Los Angeles, US
- Dana Fisman
Ben Gurion University – Beer Sheva, IL
- Pranav Garg
Amazon – Bangalore, IN
- Matthias Hein
Universität des Saarlandes, DE
- Holger Hermanns
Universität des Saarlandes, DE
- Susmit Jha
SRI – Menlo Park, US
- Kristian Kersting
TU Darmstadt, DE
- Andreas Krause
ETH Zürich, CH
- Sasa Misailovic
University of Illinois – Urbana-Champaign, US
- Mayur Naik
University of Pennsylvania – Philadelphia, US
- Nagarajan Natarajan
Microsoft Research India – Bangalore, IN
- Anna Rafferty
Carleton College – Northfield, US
- Dorsa Sadigh
Stanford University, US
- Stephan Schulz
Duale Hochschule Baden-Württemberg – Stuttgart, DE
- Sanjit A. Seshia
University of California – Berkeley, US
- Rishabh Singh
Microsoft Research – Redmond, US
- Armando Solar-Lezama
MIT – Cambridge, US
- Charles Sutton
University of Edinburgh, GB
- Josef Urban
Czech Technical University – Prague, CZ
- Martin Vechev
ETH Zürich, CH
- Eran Yahav
Technion – Haifa, IL
- Yisong Yue
California Institute of Technology – Pasadena, US
- Xiaojin Zhu
University of Wisconsin – Madison, US
- Sandra Zilles
University of Regina, CA

Report from Dagstuhl Seminar 17352

Analysis and Synthesis of Floating-point Programs

Edited by

Eva Darulova[1], Alastair F. Donaldson[2], Zvonimir Rakamarić[3], and Cindy Rubio-González[4]

1 **MPI-SWS – Saarbrücken, DE,** eva@mpi-sws.org
2 **Imperial College London, GB,** alastair.donaldson@imperial.ac.uk
3 **University of Utah – Salt Lake City, US,** zvonimir@cs.utah.edu
4 **University of California – Davis, US,** crubio@ucdavis.edu

Abstract

This report summarises the presentations, discussion sessions and panel that took place during the Dagstuhl seminar on “Analysis and Synthesis of Floating-point Programs” that took place during August 27 – 30, 2017. We hope that the report will provide a useful resource for researchers today who are interested in understanding the state-of-the-art and open problems related to analysing and synthesising floating-point programs, and as a historical resource helping to clarify the status of this field in 2017.

Seminar August 27–30, 2017 – http://www.dagstuhl.de/17352
1998 ACM Subject Classification Design and analysis of algorithms, Approximation algorithms analysis, Numeric approximation algorithms, Logic, Automated reasoning
Keywords and phrases energy-efficient computing, floating-point arithmetic, precision allocation, program optimization, rigorous compilation
Digital Object Identifier 10.4230/DagRep.7.8.74

1 Executive Summary

Eva Darulova
Alastair F. Donaldson
Zvonimir Rakamarić
Cindy Rubio-González

This report documents the program and the outcomes of Dagstuhl Seminar 17352 “Analysis and Synthesis of Floating-point Programs”.

Floating-point numbers provide a finite approximation of real numbers that attempts to strike a fine balance between range, precision, and efficiency of performing computations. Nowadays, performing floating-point computations is supported on a wide range of computing platforms, and are employed in many widely-used and important software systems, such as high-performance computing simulations, banking, stock exchange, self-driving cars, and machine learning.

However, writing correct, and yet high-performance and energy-efficient, floating-point code is challenging. For example, floating-point operations are often non-associative (contrary to their real mathematical equivalents), which creates problems when an ordering of operations is modified by either a compiler or due to nondeterministic interleavings of concurrent executions. Furthermore, the underlying floating-point hardware is often heterogeneous, hence different results may be computed across different platforms or even across components of the same heterogeneous platform. Given the underlying complexity associated with writing floating-point code, it is not surprising that there have been numerous software bugs attributed to incorrectly implemented floating-point computations.

Research related to floating-point computation spans a multitude of areas of computer science, ranging from hardware design and architecture, all the way to high-performance computing, machine learning, and software analysis and verification. The objective of this seminar was thus to bring together researchers from several of these areas, which have either traditionally been considered as non-overlapping, or which have arguably enjoyed insufficient interaction despite a clear overlap of interests. The goal in mind here was to provide opportunities to brainstorm new theoretical advances and practical techniques and tools for making floating-point computations performant and correct, and to help foster long term collaborations.

The seminar involved brief presentations from most participants, interspersed with a lot of informal technical discussion, in addition to four breakout sessions based on common themes that arose during informal discussion. In addition, a joint panel session was held between this seminar and the concurrently running "Machine Learning and Formal Methods" seminar. This report presents the collection of abstracts associated with the participant presentations, notes summarising each discussion session, and a transcript of the panel session. We hope that the report will provide a useful resource for researchers today who are interested in understanding the state-of-the-art and open problems related to analysing and synthesising floating-point programs, and as a historical resource helping to clarify the status of this field in 2017.

2 Table of Contents

Executive Summary
Eva Darulova, Alastair F. Donaldson, Zvonimir Rakamarić, and Cindy Rubio-González 74

Overview of Talks

Algorithm – Architecture Codesign
George A. Constantinides 78

Salsa: An Automatic Tool to Improve the Numerical Accuracy of Programs
Nasrine Damouche 78

Algorithms for Efficient Reproducible Floating Point Summation and BLAS
James W. Demmel 79

A Comprehensive Study of Real-World Numerical Bug Characteristics
Anthony Di Franco 80

Testing Compilers for a Language With Vague Floating-Point Semantics
Alastair F. Donaldson 80

Floating-Point Cadence
Theo Drane 81

Floating-point result-variability: sources and handling
Ganesh L. Gopalakrishnan 82

Hierarchical Search in Floating-Point Precision Tuning
Hui Guo 82

Auto-tuning Floating-Point Precision
Jeffrey K. Hollingsworth 83

Floating Point Computations in the Multicore and Manycore Era
Miriam Leeser 83

JFS: Solving floating point constraints with coverage guided fuzzing
Daniel Liew 84

Autotuning for Portable Performance for Specialized Computational Kernels
Piotr Luszczek 84

Interval Enclosures of Upper Bounds of Roundoff Errors using Semidefinite Programming
Victor Magron 85

Verification for floating-point, floating-point for verification
David Monniaux 85

Alive-FP: Automated Verification of Floating Point Optimizations in LLVM
Santosh Nagarakatte 86

Debugging Large-Scale Numerical Programs with Herbgrind
Pavel Panchekha 86

Utah Floating-Point Toolset
Zvonimir Rakamarić 87

Condition Number and Interval Computations
Nathalie Revol 87

Dynamic Analysis for Floating-Point Precision Tuning
Cindy Rubio-González 88

FPBench: Toward Standard Floating Point Benchmarks
Zachary Tatlock 88

Impacts of non-determinism on numerical reproducibility and debugging at the exascale
Michela Taufer 89

An Abstract Interpretation Framework for the Round-Off Error Analysis of Floating-Point
Laura Titolo 89

Stabilizing Numeric Programs against Platform Uncertainties
Thomas Wahl 90

Working groups

Notes from Breakout Session: “Analysis Tools for Floating-Point Software”
Pavel Panchekha 90

Notes from Breakout Session: “Specifications for Programs Computing Over Real Number Approximations”
Cindy Rubio-González 92

Notes from Breakout Session: “Compilers: IEEE Compliance and Fast Math Requirements”
Daniel Schemmel 93

Notes from Breakout Session: “Reproducibility in Floating-Point Computation”
Thomas Wahl 94

Panel discussions

Notes from Joint Panel Discussion between the “Analysis and Synthesis of Floating-Point Programs” and “Machine Learning and Formal Methods” Seminars
Alastair F. Donaldson 95

Participants 101

3 Overview of Talks

3.1 Algorithm – Architecture Codesign

George A. Constantinides (Imperial College London, GB)

Joint work of George Constantinides, Peter Cheung, Xitong Gao, Wayne Luk
URL http://cas.ee.ic.ac.uk/people/gac1

This talk takes a look at the changing face of computer architecture and the implications for numerical compute. I argue that the drive in computer architecture in recent years has been to give more silicon (and more energy) back to computation, and I note that the rise of FPGA-based computation results in many possibilities for non-standard, customised arithmetic. I review work from my group in the early 2000s, which first considered the question of precision tuning for such customisation, for LTI systems implemented in fixed-point arithmetic. I then move to look at some more recent work from my group which automatically refactors code to explore the Pareto tradeoff between accuracy, area, and performance of algorithms specified in floating-point and implemented in FPGA hardware. Finally, I pose some automation challenges to the community – success in these challenges is likely to require the combined efforts of architects, computer arithmetic researchers, numerical analysts and programming language researchers.

3.2 Salsa: An Automatic Tool to Improve the Numerical Accuracy of Programs

Nasrine Damouche (University of Perpignan, FR)

Joint work of Nasrine Damouche, Matthieu Martel
Main reference Nasrine Damouche, Matthieu Martel, "Salsa: An Automatic Tool to Improve the Numerical Accuracy of Programs", Automated Formal Methods (AFM), 2017.
URL http://fm.csl.sri.com/AFM17/AFM17_paper_4.pdf

This talk describes Salsa, an automatic tool to improve the accuracy of the floating-point computations done in numerical codes. Based on static analysis methods by abstract interpretation, our tool takes as input an original program, applies to it a set of transformation rules and then generates a transformed program which is more accurate than the initial one. The original and the transformed programs are written in the same imperative language. This talk is a concise description of former work on the techniques implemented in Salsa, extended with a presentation of the main software architecture, the inputs and outputs of the tool as well as experimental results obtained by applying our tool on a set of sample programs coming from embedded systems and numerical analysis.

3.3 Algorithms for Efficient Reproducible Floating Point Summation and BLAS

James W. Demmel (University of California – Berkeley, US)

Joint work of James W. Demmel, Peter Ahrens, Hong Diep Nguyen, Greg Henry, Peter Tang, Xiaoye Li, Jason Riedy, Mark Gates
Main reference J. Demmel, I. Dumitriu, O. Holtz, P. Koev, "Accurate and Efficient Expression Evaluation and Linear Algebra," Acta Numerica, v. 17, 2008.
URL https://tinyurl.com/y8yxq94n

We briefly present 3 topics: (1) Reproducibility. We define reproducibility to mean getting bitwise identical results from multiple runs of the same program, perhaps with different hardware resources or other changes that should ideally not change the answer. Many users depend on reproducibility for debugging or correctness. However, dynamic scheduling of parallel computing resources, combined with non-associativity of floating point addition, makes attaining reproducibility a challenge even for simple operations like summing a vector of numbers, or more complicated operations like the Basic Linear Algebra Subprograms (BLAS). We describe two versions of an algorithm to compute a reproducible sum of floating point numbers, independent of the order of summation. The first version depends only on a subset of the IEEE Floating Point Standard 754-2008, and the second version uses a possible new instruction in the proposed Standard 754-2018. The algorithm is communication-optimal, in the sense that it does just one pass over the data in the sequential case, or one reduction operation in the parallel case, requiring a "reproducible accumulator" represented by just 6 floating point words (more can be used if higher precision is desired), enabling standard tiling techniques used to optimize the BLAS. The arithmetic cost with a 6-word reproducible accumulator is 7n floating point additions to sum n words using version 1, or 3n using version 2, and (in IEEE double precision) the final error bound can be up to 10^8 times smaller than the error bound for conventional summation on ill-conditioned inputs. We also describe ongoing efforts to incorporate this in a future BLAS Standard. We seek feedback on this proposed new instruction, which is also intended to accelerate many existing algorithms for double-double, compensated summation, etc. For details see [1]. (2) New BLAS Standard. The BLAS standards committee is meeting again to consider a new version, motivated by industrial and academic demand for new precisions, batched BLAS, and reproducible BLAS. This is also an opportunity to improve floating point exception handling in the BLAS, which currently propagates exceptions, e.g. NaNs, inconsistently. We seek feedback on the draft standard, available at [2]. (3) Automatic generation of accurate floating point formulas. This was a topic at Dagstuhl 05391, 25-30 Sept, 2005, when Olga Holtz presented joint work on the decidability of whether an expression guaranteeing high relative accuracy exists for a given algebraic expression (in the 1 + delta model, so real numbers). We give an example of this (the Motzkin polynomial) and point out more recent (prize-winning) results in [3].

Topic (1) is joint work with Peter Ahrens (former UCB undergrad, now a grad student at MIT) and Hong Diep Nguyen (former UCB postdoc, now at a startup) [1].

Topic (2) is joint work with Greg Henry (Intel), Peter Tang (Intel), Xiaoye Li (LBNL), Jason Riedy (GaTech) and Mark Gates (U Tenn) [2].

Topic (3) is joint work with Olga Holtz (UCB), Ioana Dumitriu (U Wash), and Plamen Koev (former UCB grad student, now at SJSU) [3].

References

1 J. Demmel, P. Ahrens, H.-D. Nguyen. *Efficient Reproducible Floating Point Summation and BLAS.* UC Berkeley EECS Tech Report UCB/EECS-2016-121, June 2016.

2 J. Demmel, M. Gates, G. Henry, X. S. Li, J. Riedy, P. T. P. Tang. *A proposal for a Next-Generation BLAS.* Writable document (at time of writing) for submission of comments: `https://tinyurl.com/y8yxq94n`, August 2017.

3 J. Demmel, I. Dumitriu, O. Holtz, P. Koev. *Accurate and Efficient Expression Evaluation and Linear Algebra.* Acta Numerica, v. 17, 2008.

3.4 A Comprehensive Study of Real-World Numerical Bug Characteristics

Anthony Di Franco (University of California – Davis, US)

Joint work of Anthony Di Franco, Hui Guo, Cindy Rubio-González
Main reference Anthony Di Franco, Hui Guo, Cindy Rubio-González: "A comprehensive study of real-world numerical bug characteristics", in Proc. of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017, pp. 509–519, IEEE Computer Society, 2017.
URL http://dx.doi.org/10.1109/ASE.2017.8115662

Numerical software is used in a wide variety of applications including safety-critical systems, which have stringent correctness requirements, and whose failures have catastrophic consequences that endanger human life. Numerical bugs are known to be particularly difficult to diagnose and fix, largely due to the use of approximate representations of numbers such as floating point. Understanding the characteristics of numerical bugs is the first step to combat them more effectively. In this paper, we present the first comprehensive study of real-world numerical bugs. Specifically, we identify and carefully examine 269 numerical bugs from five widely-used numerical software libraries: NumPy, SciPy, LAPACK, GNU Scientific Library, and Elemental. We propose a categorization of numerical bugs, and discuss their frequency, symptoms and fixes. Our study opens new directions in the areas of program analysis, testing, and automated program repair of numerical software, and provides a collection of real-world numerical bugs.

3.5 Testing Compilers for a Language With Vague Floating-Point Semantics

Alastair F. Donaldson (Imperial College London, GB)

Joint work of Alastair F. Donaldson, Hugues Evrard, Andrei Lascu, Paul Thomson
Main reference Alastair F. Donaldson, Hugues Evrard, Andrei Lascu, Paul Thomson: "Automated testing of graphics shader compilers", PACMPL, Vol. 1(OOPSLA), pp. 93:1–93:29, 2017.
URL http://dx.doi.org/10.1145/3133917

The OpenGL shading language (GLSL) deliberately has vague semantics when it comes to floating-point arithmetic, stating (see p. 90 of the OpenGL Shading Language 4.50 specification):

"Without any [precision] qualifiers, implementations are permitted to perform such optimizations that effectively modify the order or number of operations used to evaluate an expression, even if those optimizations may produce slightly different results relative to unoptimized code."

As a result, it is hard to test compilers for GLSL: there is no predefined image that a shader should render for a given input, and differential testing across shader compilers for different platforms is not straightforward because one can legitimately expect differences in rendering results.

We have been investigating an alternative testing approach using *metamorphic testing*, inspired by recent work on compiler validation using *equivalence modulo inputs* testing. We take an initial shader and make from it a family of *variant* shaders by applying semantics-preserving transformations, or more precisely transformations that would have no impact on semantics if floating-point operations were replaced with real number operations. We then render, on a single platform, all the images generated by compiling and executing each shader in the family. While we do expect to see pixel-level differences between rendered images, since our transformations may have impacted on floating-point computation, if the initial shader is numerically stable these differences should not be large. Using an image comparison metric, such as the chi-squared distance between image histograms, we can automatically flag up variant shaders that give rise to large rendering differences. To such *deviant* shaders we then apply a form of delta-debugging whereby we iteratively reverse the semantics-preserving transformations that were initially applied, converging on a shader that differs minimally from the original shader such that the source-level difference – which should be semantics-preserving – causes a large difference in rendering. Inspection of this difference then sheds light on whether there is a compiler bug, or whether the original shader is in fact numerically unstable.

We have implemented this method in a tool, GLFuzz, which we have successfully applied to a number of commercial shader compilers, finding a large number of defects that we track via a repository (see https://github.com/mc-imperial/shader-compiler-bugs/), with highlights from the work summarized via a series of online posts (see https://medium.com/@afd_icl/689d15ce922b).

3.6 Floating-Point Cadence

Theo Drane (Cadence Design Systems – Bracknell, GB)

Main reference Theo Drane, "Lossy Polynomial Datapath Synthesis", PhD thesis, Imperial College London, 2014.
URL https://spiral.imperial.ac.uk/handle/10044/1/15566

Cadence's logic synthesis and high level synthesis divisions, Genus and Stratus respectively, both come with floating-point IP offerings. Our IP development is done in tandem with our synthesis tools and thus provide a unique opportunity for co-optimisation. Our verification draws upon other Cadence divisions in equivalence and model checking, Conformal and Jasper divisions respectively; as well as reaching out into the use of academic theorem provers. This talk will cover the offerings, challenges and potential future avenues of research and development in this area.

3.7 Floating-point result-variability: sources and handling

Ganesh L. Gopalakrishnan (University of Utah – Salt Lake City, US)

Joint work of Geof Sawaya, Michael Bentley, Ian Briggs, Ganesh Gopalakrishnan, Dong Ahn
Main reference Geoffrey Sawaya, Michael Bentley, Ian Briggs, Ganesh Gopalakrishnan, Dong H. Ahn: "FLiT: Cross-platform floating-point result-consistency tester and workload", in Proc. of the 2017 IEEE International Symposium on Workload Characterization, IISWC 2017, Seattle, WA, USA, October 1-3, 2017, pp. 229–238, IEEE Computer Society, 2017.
URL http://dx.doi.org/10.1109/IISWC.2017.8167780

Understanding the extent to which computational results can change across platforms, compilers, and compiler flags can go a long way toward supporting reproducible experiments. In this work, we offer the first automated testing aid called FLiT (Floating-point Litmus Tester) that can show how much these results can vary for any user-given collection of computational kernels. Our approach is to take a collection of these kernels, disperse them across a collection of compute nodes (each with a different architecture), have them compiled and run, and bring the results to a central SQL database for deeper analysis.

Properly conducting these activities requires a careful selection (or design) of these kernels, input generation methods for them, and the ability to interpret the results in meaningful ways. The results in this paper are meant to inform two different communities: (a) those interested in seeking higher performance by considering "IEEE unsafe" optimizations, but then want to understand how much result variability to expect, and (b) those interested in standardizing compiler flags and their meanings, so that one may safely port code across generations of compilers and architectures.

By releasing FLiT, we have also opened up the possibility of all HPC developers using it as a common resource as well as contributing back interesting test kernels as well as best practices, thus extending the floating-point result-consistency workload we contribute. This is the first such workload and result-consistency tester underlying floating-point reproducibility of which we are aware.

3.8 Hierarchical Search in Floating-Point Precision Tuning

Hui Guo (University of California – Davis, US)

Joint work of Hui Guo, Cindy Rubio-González

Floating-point types are notorious for their intricate representation. The effective use of mixed precision, i.e., using various precisions in different computations, is critical to achieve a good balance between accuracy and performance. Unfortunately, reasoning about the impact of different precision selections is a difficult task even for numerical experts. Techniques have been proposed to systematically search over floating-point variables and/or program instructions to find a profitable precision configuration. These techniques are characterized by their "black-box" nature, and face scalability limitations mainly due to the large search space.

In this talk, we present a semantic-based search algorithm that hierarchically guides precision tuning to improve performance given an accuracy constraint. The novelty of our algorithm lies in leveraging semantic information to reduce the search space and find more profitable precision configurations. Specifically, we perform dependence analysis and edge profiling to create a weighted dependence graph that presents a network of floating-point variables. We then formulate hierarchy construction on the network as a community detection problem, and present a hierarchical search algorithm that iteratively lowers precision with regard to communities. Our evaluation on real world floating-point programs shows that hierarchical search algorithm is in general more efficient than the state of the art in finding profitable configurations.

3.9 Auto-tuning Floating-Point Precision

Jeffrey K. Hollingsworth (University of Maryland – College Park, US)

Joint work of Mike Lam, Jeffrey K. Hollingsworth
Main reference Mike O. Lam, Jeffrey K. Hollingsworth, "Fine-grained floating-point precision analysis," International Journal of High Performance Computing Applications, SAGE Publications, 2016.
URL http://doi.org/10.1177/1094342016652462

In this talk I will describe the CRAFT tool for automatically exploring the required floating point precision for specific programs. The tool operates on compiled binary programs. Using different analysis modules, it can explore reduced precision (single vs. double) as well as variable precisions (specific numbers of bits of precision for each operation). I will also present results of using the CRAFT tool for different benchmark programs.

3.10 Floating Point Computations in the Multicore and Manycore Era

Miriam Leeser (Northeastern University – Boston, US)

Joint work of Miriam Leeser, Thomas Wahl, Mahsa Bayati, Yijia Gu
Main reference Yijia Gu, Thomas Wahl, Mahsa Bayati, Miriam Leeser: "Behavioral Non-portability in Scientific Numeric Computing", in Proc. of the Euro-Par 2015: Parallel Processing - 21st International Conference on Parallel and Distributed Computing, Vienna, Austria, August 24-28, 2015, Proceedings, Lecture Notes in Computer Science, Vol. 9233, pp. 558–569, Springer, 2015.
URL https://doi.org/10.1007/978-3-662-48096-0_43

The results of numerical computations with floating-point numbers depend on the execution platform. One reason is that, even for similar floating point hardware, compilers have significant freedom in deciding how to evaluate a floating-point expression, as such evaluation is not standardized. Differences can become particularly large across (heterogeneous) parallel architectures. This may be surprising to a programmer who conflates the portability promised by programming standards such as OpenCL with reproducibility.

In this talk, I present experiments, conducted on a variety of platforms including CPUs and GPUs, that showcase the differences that can occur even for randomly selected inputs. I present a study of the same OpenCL code run on different architectures and analyze the sources of differences, and what tools are needed to aid the programmer. The code is taken from popular high performance computing benchmark suites. I also introduced challenges to the research community. A major one is being able to handle programs that run on large parallel machines while providing useful information to the user regarding the importance of these differences.

3.11 JFS: Solving floating point constraints with coverage guided fuzzing

Daniel Liew (Imperial College London, GB)

Joint work of Daniel Liew, Cristian Cadar, Alastair Donaldson
Main reference Daniel Liew, Daniel Schemmel, Cristian Cadar, Alastair F. Donaldson, Rafael Zähl, Klaus Wehrle: "Floating-point symbolic execution: a case study in n-version programming", in Proc. of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017, pp. 601–612, IEEE Computer Society, 2017.
URL http://dx.doi.org/10.1109/ASE.2017.8115670

In this talk I will present the work-in-progress design and implementation of a constraint solver called JIT Fuzzing Solver (JFS). JFS is designed to investigate the use of "coverage guided fuzzing" as an incomplete tactic for solving SMT queries that use any combination of the Core, BitVector, and FloatingPoint theories defined by SMT-LIB.

JFS takes as input an SMT query, from which it generates a C++ program with a sequence of "if" branches, each corresponding to an assert from the query. The program is constructed such that finding an input to the program that traverses all the "true" edges of the branches is equivalent to finding a satisfying assignment to all the free variables of the SMT query. To find such an input, a high-performance coverage-guided fuzzer is used. We conclude with a brief discussion of initial results.

This work was motivated by a project on extending symbolic execution with support for floating-point reasoning; the "main reference" is a paper on that project.

3.12 Autotuning for Portable Performance for Specialized Computational Kernels

Piotr Luszczek (University of Tennessee – Knoxville, US)

Joint work of Piotr Luszczek, Jakub Kurzak, Mark Gates, Mike (Yaohung) Tsai, Matthew Bachstein, Jack Dongarra
Main reference Mark Gates, Jakub Kurzak, Piotr Luszczek, Yu Pei, Jack J. Dongarra: "Autotuning Batch Cholesky Factorization in CUDA with Interleaved Layout of Matrices", in Proc. of the 2017 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2017, Orlando / Buena Vista, FL, USA, May 29 - June 2, 2017, pp. 1408–1417, IEEE Computer Society, 2017.
URL http://dx.doi.org/10.1109/IPDPSW.2017.18

High performance Exascale libraries for numerical algorithms, data analytics, and statistical inference require specialized implementations of computational kernels that progressively become harder to create due to the increasingly divergent designs of extreme-scale hardware platforms. We present our ongoing efforts in automated performance engineering that encompasses code autotuning within a comprehensive framework of integrated tools that include domain specific languages, testing/deployment infrastructure, and analysis modules. These components assist in continuous development, deployment, and improvement of these essential scientific kernels.

3.13 Interval Enclosures of Upper Bounds of Roundoff Errors using Semidefinite Programming

Victor Magron (VERIMAG – Grenoble, FR)

Joint work of Victor Magron, George Constantinides, Alastair F. Donaldson
Main reference Victor Magron, George A. Constantinides, Alastair F. Donaldson: "Certified Roundoff Error Bounds Using Semidefinite Programming", ACM Trans. Math. Softw., Vol. 43(4), pp. 34:1–34:31, 2017.
URL http://dx.doi.org/10.1145/3015465

Roundoff errors cannot be avoided when implementing numerical programs with finite precision. The ability to reason about rounding is especially important if one wants to explore a range of potential representations, for instance in the world of FPGAs. This problem becomes challenging when the program does not employ solely linear operations as non-linearities are inherent to many computational problems in real-world applications.

We present two frameworks which can be combined to provide interval enclosures of upper bounds for absolute roundoff errors.

The framework for upper bounds is based on optimization techniques employing semidefinite programming (SDP) and sparse sums of squares certificates, which can be formally checked inside the Coq theorem prover.

The framework for lower bounds is based on a new hierarchy of convergent robust SDP approximations for certain classes of polynomial optimization problems. Each problem in this hierarchy can be exactly solved via SDP.

Our tool covers a wide range of nonlinear programs, including polynomials and transcendental operations as well as conditional statements. We illustrate the efficiency and precision of this tool on non-trivial programs coming from biology, optimization and space control.

3.14 Verification for floating-point, floating-point for verification

David Monniaux (Université Grenoble Alpes – Saint-Martin-d'Hères, FR)

Joint work of David Monniaux, Antoine Miné, Jérôme Feret
Main reference Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, Xavier Rival: "A static analyzer for large safety-critical software", in Proc. of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation 2003, San Diego, California, USA, June 9-11, 2003, pp. 196–207, ACM, 2003.
URL http://dx.doi.org/10.1145/781131.781153

Verification for floating-point: Verifying the safety of floating-point programs poses specific challenges. The usual assumptions about arithmetic types (addition and multiplication form a commutative ring, etc.) are now false, making it difficult to apply standard abstractions (e.g. zones, octagons, polyhedra). In the Astrée system, we applied interval analysis as well as relaxation of floating-point into reals so as to use the standard abstractions. We also developed specific abstract domains for filters.

Floating-point for verification: It is tempting to use floating-point inside decision procedures and other algorithms inside analysis tools, instead of exact precision arithmetic, which can be much slower. But how can we be sure of the results? We present here methods for e.g. using an off-the-shelf implementation of linear programming inside a sound tool.

3.15 Alive-FP: Automated Verification of Floating Point Optimizations in LLVM

Santosh Nagarakatte (Rutgers University – Piscataway, US)

Joint work of David Menendez, Santosh Nagarakatte, Aarti Gupta
Main reference David Menendez, Santosh Nagarakatte, Aarti Gupta: "Alive-FP: Automated Verification of Floating Point Based Peephole Optimizations in LLVM", in Proc. of the Static Analysis - 23rd International Symposium, SAS 2016, Edinburgh, UK, September 8-10, 2016, Proceedings, Lecture Notes in Computer Science, Vol. 9837, pp. 317–337, Springer, 2016.
URL https://doi.org/10.1007/978-3-662-53413-7_16

Peephole optimizations optimize and canonicalize code to enable other optimizations but are error-prone. This talk presents Alive-FP, an automated verification framework for floating point based peephole optimizations in LLVM. Alive-FP handles a class of floating point optimizations and fast-math optimizations involving signed zeros, not-a-number, and infinities, which do not result in loss of accuracy. This talk will describe multiple encodings for various floating point operations to account for the various kinds of undefined behavior and under-specification in the LLVM's language reference manual. We have discovered seven wrong optimizations in LLVM using Alive-FP.

3.16 Debugging Large-Scale Numerical Programs with Herbgrind

Pavel Panchekha (University of Washington – Seattle, US)

Joint work of Pavel Panchekha, Alex Sanchez-Stern, Zachary Tatlock, Sorin Lerner
URL http://herbgrind.ucsd.edu

It is hard to find and fix numerical errors in large numerical programs because it is difficult to detect the error, determine which instructions are responsible for the error, and gather context describing the computation that lead to the error. Herbgrind is a dynamic binary analysis tool to address these issues. Herbgrind uses higher-precision shadow values to compute a ground truth, correct value for every floating-point value in the program, thus making it possible to detect that program outputs are erroneous. Herbgrind then uses the local error heuristic to identify program instructions that introduce numerical errors and then propagates information about those instructions to affected program values using a taint analysis. Finally, to gather context describing the computations that lead to instructions with high local error, Herbgrind uses anti-unification to summarized expression trees into an abstract form that describes the computation. Herbgrind has been used on large numerical programs, and has proven valuable for identifying errors and providing the information necessary to fix them.

3.17 Utah Floating-Point Toolset

Zvonimir Rakamarić (University of Utah – Salt Lake City, US)

Joint work of Zvonimir Rakamarić, Ganesh Gopalakrishnan, Alexey Solovyev, Wei-Fan Chiang, Ian Briggs, Mark Baranowski, Dietrich Geisler, Charles Jacobsen
Main reference Alexey Solovyev, Charles Jacobsen, Zvonimir Rakamarić, Ganesh Gopalakrishnan: "Rigorous Estimation of Floating-Point Round-off Errors with Symbolic Taylor Expansions", in Proc. of the FM 2015: Formal Methods - 20th International Symposium, Oslo, Norway, June 24-26, 2015, Proceedings, Lecture Notes in Computer Science, Vol. 9109, pp. 532–550, Springer, 2015.
URL https://doi.org/10.1007/978-3-319-19249-9_33

Virtually all real-valued computations are carried out using floating-point data types and operations. With the current emphasis of system development often being on computational efficiency, developers as well as compilers are increasingly attempting to optimize floating-point routines. Reasoning about the correctness of these optimizations is complicated, and requires error analysis procedures with different characteristics and trade-offs. In my talk, I present both dynamic and rigorous static analyses we developed for estimating errors of floating-point routines. Finally, I describe how we extended our rigorous static analysis into a procedure for mixed-precision tuning of floating-point routines.

3.18 Condition Number and Interval Computations

Nathalie Revol (ENS – Lyon, FR)

Main reference Nathalie Revol, "Influence of the Condition Number on Interval Computations: Illustration on Some Examples," in honour of Vladik Kreinovich' 65th birthday, Springer Festschrift, El Paso, United States, 2017.
URL https://hal.inria.fr/hal-01588713

The condition number is a quantity that is well-known in "classical" numerical analysis, that is, where numerical computations are performed using floating-point numbers. This quantity appears much less frequently in interval numerical analysis, that is, where the computations are performed on intervals.

In this talk, three small examples are used to illustrate experimentally the impact of the condition number on interval computations. As expected, problems with a larger condition number are more difficult to solve: this means either that the solution is not very accurate (for moderate condition numbers) or that the method fails to solve the problem, even inaccurately (for larger condition numbers). Different strategies to counteract the impact of the condition number are discussed and, with success, experimented: use of a higher precision, iterative refinement, bisection of the input.

3.19 Dynamic Analysis for Floating-Point Precision Tuning

Cindy Rubio-González (University of California – Davis, US)

License Creative Commons BY 3.0 Unported license
© Cindy Rubio-González
Joint work of Cindy Rubio-González, Cuong Nguyen, Hong Diep Nguyen, James Demmel, William Kahan, Koushik Sen, David H Bailey, Iancu, Costin; Hough, David
Main reference Cindy Rubio-González, Cuong Nguyen, Hong Diep Nguyen, James Demmel, William Kahan, Koushik Sen, David H. Bailey, Costin Iancu, David Hough: "Precimonious: tuning assistant for floating-point precision", in Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC'13, Denver, CO, USA - November 17 - 21, 2013, pp. 27:1–27:12, ACM, 2013.
URL http://dx.doi.org/10.1145/2503210.2503296

Given the variety of numerical errors that can occur, floating-point programs are difficult to write, test and debug. One common practice employed by developers without an advanced background in numerical analysis is using the highest available precision. While more robust, this can degrade program performance significantly. In this paper we present Precimonious, a dynamic program analysis tool to assist developers in tuning the precision of floating-point programs. Precimonious performs a search on the types of the floating-point program variables trying to lower their precision subject to accuracy constraints and performance goals. Our tool recommends a type instantiation that uses lower precision while producing an accurate enough answer without causing exceptions. We evaluate Precimonious on several widely used functions from the GNU Scientific Library, two NAS Parallel Benchmarks, and three other numerical programs. For most of the programs analyzed, Precimonious reduces precision, which results in performance improvements as high as 41%.

3.20 FPBench: Toward Standard Floating Point Benchmarks

Zachary Tatlock (University of Washington – Seattle, US)

License Creative Commons BY 3.0 Unported license
© Zachary Tatlock
Joint work of Zachary Tatlock, Nasrine Damouche, Matthieu Martel, Pavel Panchekha, Chen Qiu, Alexander Sanchez-Stern
Main reference Nasrine Damouche, Matthieu Martel, Pavel Panchekha, Chen Qiu, Alexander Sanchez-Stern, Zachary Tatlock: "Toward a Standard Benchmark Format and Suite for Floating-Point Analysis", in Proc. of the Numerical Software Verification - 9th International Workshop, NSV 2016, Toronto, ON, Canada, July 17-18, 2016, [collocated with CAV 2016], Revised Selected Papers, Lecture Notes in Computer Science, Vol. 10152, pp. 63–77, 2016.
URL http://dx.doi.org/10.1007/978-3-319-54292-8_6

FPBench is a standard format and common set of benchmarks for floating-point accuracy tests. The goal of FPBench is to enable direct comparisons between competing tools, facilitate the composition of complementary tools, and lower the barrier to entry for new teams working on numerical tools. FPBench collects benchmarks from published papers in a standard format and with standard accuracy measures and metadata. As a single repository for benchmarks, FPBench can be used to guide the development of new tools, evaluate completed tools, or compare existing tools on identical inputs, all while avoiding duplication and the manual effort and inevitable errors of translating between input formats. Please see http://fpbench.org for more.

3.21 Impacts of non-determinism on numerical reproducibility and debugging at the exascale

Michela Taufer (University of Delaware – Newark, US)

Joint work of Dylan Chapp, Travis Johnston, Michela Taufer
Main reference Dylan Chapp, Travis Johnston, Michela Taufer: "On the Need for Reproducible Numerical Accuracy through Intelligent Runtime Selection of Reduction Algorithms at the Extreme Scale", in Proc. of the 2015 IEEE International Conference on Cluster Computing, CLUSTER 2015, Chicago, IL, USA, September 8-11, 2015, pp. 166–175, IEEE Computer Society, 2015.
URL http://dx.doi.org/10.1109/CLUSTER.2015.34

In message-passing applications, one notable technique is the use of non-blocking point-to-point communication, which permits communication and computation to be overlapped, leading to an increase in scalability. The price paid however, is the loss of determinism in applications' executions. Non-determinism in high performance scientific applications has severe detrimental impacts for both numerical reproducibility and debugging. As scientific simulations are migrated to extreme-scale platforms, the increase in platform concurrency and the attendant increase in non-determinism is likely to exacerbate both of these problems (i.e., numerical reproducibility and debugging). In this talk, we address the challenges of non-determinism's impact on numerical reproducibility and on debugging. Specifically, we present empirical studies focusing on floating-point error accumulation over global reductions where enforcing any reduction order is expensive or impossible. We also discuss techniques for record and replay to reduce out-of-order message rate in non-deterministic execution.

3.22 An Abstract Interpretation Framework for the Round-Off Error Analysis of Floating-Point

Laura Titolo (National Institute of Aerospace – Hampton, US)

Joint work of Marco A. Feliu, Aaron Dutle, Laura Titolo, Cesar A. Muñoz
Main reference Mariano M. Moscato, Laura Titolo, Aaron Dutle, César A. Muñoz: "Automatic Estimation of Verified Floating-Point Round-Off Errors via Static Analysis", in Proc. of the Computer Safety, Reliability, and Security - 36th International Conference, SAFECOMP 2017, Trento, Italy, September 13-15, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10488, pp. 213–229, Springer, 2017.
URL http://doi.org/10.1007/978-3-319-66266-4_14

In this work, an abstract interpretation framework for the round-off error analysis of floating-point programs is presented. This framework defines a parametric abstract semantics of the floating-point round-off error of functional programs. Given a functional floating-point program, the abstract semantics computes, for each execution path of the program, an error expression representing a sound over-approximations of the accumulated floating-point round-off error that may occur in that execution path. In addition, a boolean expression representing the input values that satisfy the path conditions of the given path is also computed. This boolean expression characterizes the input values leading to the computed error approximation. A widening operator is defined to ensure the convergence of recursive programs. The proposed framework provides the infrastructure to correctly handle the so-called unstable tests, which occur when the round-off errors of conditional floating-point expressions alter the execution path of the ideal program on infinite precision real arithmetic.

3.23 Stabilizing Numeric Programs against Platform Uncertainties

Thomas Wahl (Northeastern University – Boston, US)

Joint work of Mahsa Bayati, Yijia Gu, Miriam Leeser, Thomas Wahl
Main reference Yijia Gu, Thomas Wahl: "Stabilizing Floating-Point Programs Using Provenance Analysis", in Proc. of the Verification, Model Checking, and Abstract Interpretation - 18th International Conference, VMCAI 2017, Paris, France, January 15-17, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10145, pp. 228–245, Springer, 2017.
URL https://doi.org/10.1007/978-3-319-52234-0_13

Floating-point arithmetic (FPA) is a loosely standardized approximation of real arithmetic available on many computers today. The use of approximation incurs commonly underestimated risks for the reliability of numeric software, including reproducibility issues caused by the relatively large degree of freedom for FPA implementers offered by the IEEE 754 standard. If left untreated, such problems can seriously interfere with program portability.

In this talk I demonstrate how, using information on the provenance of platform dependencies, reproducibility violations can be repaired with low impact on program efficiency, resulting in stabilized program execution. I illustrate the use of these techniques on decision-making and purely numeric programs.

This is direct joint work with my student Yijia Gu, and is based on longer collaboration with Miriam Leeser and her students, particularly Mahsa Bayati.

4 Working groups

4.1 Notes from Breakout Session: "Analysis Tools for Floating-Point Software"

Pavel Panchekha (University of Washington – Seattle, US)

The discussion on tools focused on two central issues: the benefits and challenges of running a tools competition (heavily reliant on Zvonimir's experience with SVComp) and then on incremental steps that can be taken toward those benefits. The key conclusions were that a tools competition provides visibility for participants, including a full list of entrants; a deadline by which entrants must support a common format, which force participants to implement that functionality; ease of entry by new groups; and the opportunity to write up an analysis on the entrants, their strengths and weaknesses, and the benchmarks covered. This final benefit seemed particularly beneficial to industry users, as Sam attested. To achieve some of these benefits incrementally, a series of four steps were chosen: first, a central list of floating point research tools; second, support for common input and output formats; third, regular composition of floating-point tools; and fourth, regular competitions to compare tools in the same space.

The discussion started with Zvonimir's experience with the SVComp competition, where software verification tools compete to find assertion violations on a set of tens of thousands of benchmarks formatted as C code. The competition works by having a deadline by which entrants must submit an analyzer. Each analyzer is then run on each benchmark without any interactions, and the results are tallied, scored, and the results made available two weeks later or so. The whole evaluation is fully-scripted, though other competitions are structured differently (with interaction and a limited benchmark set). The results of the competition are then discussed at a dedicated sessions at TACAS, which also provides 50 pages of conference proceedings to publish the competition analysis (written by the competition organizer) as well as write-ups by winners. A key challenge in organizing the competition is in resolving disagreements between tools and within the community. For example, a dedicated witness and proof format was eventually invented to make it possible to resolve disagreements between tools (since not every benchmark program is labeled with correct answers), and the weighting of results has been changed over time as the emphases of the community have shifted (mistakes, for example, are now more heavily penalized than earlier). Within the community, managing different perspectives, like the difference between bug-finding (where false positives are penalized but false negatives may be acceptable) and verification (where the emphasis is reversed) has been a challenge throughout the life of the competition.

This experience informed a discussion of the dangers of a tools competition. The chief among them is the possibility of over-fitting to the benchmarks. As the community's goals change, some benchmarks may become irrelevant, and their presence in the competition distorts the competitions goals. Benchmarks are also drawn from many sources, and this leads to disagreements in the community; for example, the SVComp benchmarks range from five to fifty-thousand lines of C – disagreements over the relative importance of programs of different sizes can be a problem. A committee is also needed to handle disputes, plus organizational effort put in yearly and money to set up the competition and reward winners. In SVComp's case, this is done by a single professor, who secured a European grant for the work; over the long term, this may be necessary. Especially valuable in SVComp's case has been the support of a conference. Publishing proceedings and tool reports is an important way to reward competition entrants, and so extremely important in organizing one. (A conference such as TACAS allowing proceedings space is especially good, but in a pinch it seemed a workshop proceedings could also work. A workshop proceedings could also allow the competition to move from conference to conference, exposing the competition to new audiences.)

Some miscellaneous thoughts on the competition asked whether the competition should also involve submitting new benchmarks (whether that would be fair, and also whether or not it would lead to good benchmark submissions); and how to set the objective of the competition: highest accuracy, tightest error bounds, assertion verification, or something else? There would also need to be consideration for tools that verify specifications and tools that help users explore the design space. Since a persistent issue is accuracy improvement by increasing the domain of accurate input values, evaluation by that metric would also be valuable. With as diverse a community as floating-point is, a large suite of benchmarks and several independent competitions seemed necessary.

With the idea of a tools competition having been deeply explored, the discussion turned to composing tools. Though competitions would force tool developers to adopt common formats, composing tools would be the most valuable benefit of that change. The most valuable compositions seem to be refactoring (as in Salsa or Herbie) together with precision tuning (as in Precimonious or FPTuner); tuning and refactoring with verification (as in Rosa or FPTaylor); and input generation together with tools (like Precimonious or Herbgrind) that rely on user-supplied inputs. These compositions would yield more complicated tools with more complicated trade-offs. It would become important to show the Pareto frontier (between, for example, accuracy and speed), and to evaluate pairs of tools together.

Finally, to make the discussion even more concrete, Pavel spoke for three strategies the FPBench project had been considering. The first was a list of floating-point research tools – a website could be maintained, with information about each tool and links to project web pages, source code, and papers. Second would be an auto-evaluation capability, to download supported tools and run them on the benchmarks automatically. Third would be a series of workshop papers on the benefits and uses combining two tools together. The universal consensus was that a list of tools would be most immediately valuable; Theo described this approach as "Visibility First". The other two suggestions could follow on from that work. The list of tools could also indicate support for FPBench, which could be a form of leverage to encourage tools to support the common format. Overall, the order would be to focus on listing tools, then encouraging a common input format, then composing tools, and only then comparing them.

Additional suggestions for FPBench included an exact real evaluation mode (using, for example, the computable reals) and support for simple assertion tests (such as sign or convergence tests). For full accuracy specification, a much richer assertion language would be needed, which FPBench did not develop due to a lack of examples, but simple tests could be easily described and then verified.

4.2 Notes from Breakout Session: "Specifications for Programs Computing Over Real Number Approximations"

Cindy Rubio-González (University of California – Davis, US)

We are interested in specifications for programs computing over reals.

What is the notion of accuracy? According to the machine learning panel, and the discussions throughout the seminar, there is a mismatch between worst-case error case and programmer's expectations. Programmers want programs to be approximately correct most of the time.

What does it mean to be approximately correct? What do programmers mean by 'most of the time'? We really need to take the application into account.

What do we mean by specifications? We also need to be more specific about the kinds of specifications we are interested in. What level of specifications do we refer to? Specifications at the user level, or at the component level? Furthermore, tools work at different levels of abstraction? Are we interested in hardware specifications, library specifications, application specifications? Do we need DSLs for expressing specifications?

What do 'most of the time' mean? For many, most of the time may mean 'for most inputs'. If we consider classification in machine learning, positive/negative results can be massively wrong. On the other hand, a large different in a number may be OK for applications where accuracy is not as important. 'Most of the time' is definitely relative, and heavily dependent on the application under study.

'Most of the time' may also require for us to be aware of special cases. If we assume all inputs are independent, then we can sampling over reals/floats. However, users need to be educated in this respect. For dynamic tools, test inputs are already samples, and a main goal is for user inputs not to break the tool.

What is a good specification for verification? We really need an expert to provide the specifications. For example, the control theory part of embedded systems would provide specifications, but do they? There is just no general way to gather specifications because these depend on the application. Specifications are domain specific.

But, what does it mean to be most of the time correct? For machine learning the specification is "approximately correct". But "most of the time" is not part of the equation. We are not able to prove "all of the time", and we don't know what "most of the time" means. Perhaps what we want to find is under what conditions an algorithm is stable. That is, under what inputs is an algorithm stable? Thus, we should perhaps focus on the measure of robustness and the discovery of preconditions.

How about probabilistic assertions? Can we use these to describe the input space? We want the probability distribution. When is the environment hostile?

How about benchmarks of probabilistic assertions? Do they include floating point? Are they large enough? Are there any other floating-point issues these do not consider? For example, floating-point roundoff errors are not considered in the context of probabilistic programming.

Are there simpler specifications we can find? Perhaps we can synthesize simpler optimizations such as loop optimization with respect to precision required, or assertions to insert in the code.

In general, there is a need for better examples. There are many different application domains facing different problems. An example is radiotherapy. The main block to verification is the lack of formal semantics.

Scientists just "know" what is a correct/expected answer.

What can we do in computation that provides self certification? An example is calculating the residual for a certain computation. But, to what degree can we use self certification?

What to do when specifications are not available? How about testing without oracles? A very successful approach is to compare alternative ways to compute an answer, and then compare the results. Although no formal guarantees are provided, it is very useful in practice to detect unexpected behavior.

4.3 Notes from Breakout Session: "Compilers: IEEE Compliance and Fast Math Requirements"

Daniel Schemmel (RWTH Aachen, DE)

Prompted not only by problems in traditional languages, such as C, C++ or Fortran, but also by upcoming languages intended for other platforms, such as OpenCL or OpenACC, the questions of IEEE-754 compliance and specification of fast math requirements was discussed in this breakout session. Current state of the art is very compiler-dependent behavior, whose guarantees (where they exist at all) are not trusted very far.

Ideally, a user would be able to always rely on strong guarantees such as: specified error bounds will be met, the control flow of the program remains unchanged and/or FP optimizations will never introduce undefined behavior to an otherwise correct program. Any of these strong guarantees would however require strong whole-program analyses, which are usually infeasible. Additionally, many library operations are not just out of scope for the compiler, but may only exist as compiled code or even be written directly in assembly.

The consensus in this breakout session was that floating point optimizations should be compiled as a well-specified set of contracts, in which the developer explicitly agrees to certain restrictions in order to increase performance. Examples for such contracts could be a guarantee to never use certain kinds of values such as NaNs or Infinities (compare e.g. GCC's `-ffinite-math-only`) or the explicit permission to ignore association, which may improve vectorization at the cost of precision. In general, IEEE-754 non-compliant behavior should never be the default.

While modern C compilers are already going in a similar direction, there are big weaknesses w.r.t. IEEE rounding modes. For example, GCC does not even claim to be IEEE-754 compliant in its default mode, as it always assumes round to nearest, tie to even. The reasoning behind this is that it often inhibits even basic operations like constant folding, if the rounding mode is not known statically. During this session, the participants agreed that it would be useful to be able to statically define rounding modes per instruction or per block. This would enable static analysis while still being useful in most cases. Two additional rounding modes were suggested: First, a "native" rounding mode would be useful in the case where the developer does not really care, and maybe only a single (possibly non-standard) rounding mode is available on the target platform. This "native" rounding mode would allow maximum optimization. Second, a "dynamic" rounding mode could enable a dynamic setting, as is currently mandated by changing the floating point environment programmatically (cf. `#pragma STDC FENV ACCESS ON`). By statically expressing a desire for a specific rounding mode, it becomes more natural for compilers to either comply with the request, or to deny compilation, which would elegantly deal with the current problem of compilers that silently ignore it.

Finally, the question of how to enforce floating point contracts came up. To check and/or enforce that the contracts which the programmer agreed to are being kept, it would be useful to dynamically check for violations, which would then allow easy debugging. To deal with that, an extension to UBSan, which already includes some small FP checks, was suggested.

4.4 Notes from Breakout Session: "Reproducibility in Floating-Point Computation"

Thomas Wahl (Northeastern University – Boston, US)

Sources of non-reproducibility

- Not necessarily related to FP: exists way beyond FP.
- Test question: is integer addition associative (and hence reproducible, irrespective of evaluation order)? No, due to "one-sided overflows"

Dangers of non-reproducibility

- Do people even notice that there is non-reproducibility? Unlike classical crashes, you won't run into such issues by chance: as long as you play only with your production machine, they will not emerge.
- Nondeterminism in debugging: print instructions, assertions can change memory layout of your program and prevent/enable use of FMA.

- More generally known as “Heisenbugs”: they disappear only because you try to find them. Agreed that we need reproducibility beyond just the debugging phase.
- Non-reproducibility can also be a valuable indicator for problems in the code: if results depend on computation platform, they are bound to be numerically unstable, irrespective of the accuracy question.

Enforcing reproducibility

- Easy if you ignore the other numeric quality criteria (performance, accuracy), but probably these cannot/should not be separated.
- Even just enforcing reproducibility alone raises questions: you can determinize your code e.g. towards “strict evaluation” (supported by many compilers), but such arbitrary choice is likely not the best for accuracy and performance. “We don’ know what to reproduce.”
- Better/more realistic definition than bit-precise reproducibility: preserve control flow; allow exception: ignore differences in bit-patterns for NAN. Need a specification language of reproducibility: what am I willing to accept.
- Could specify that I allow optimizations like constant folding to ignore differences in L/R association and just fold `X * 3.5 * 4.2` to `X * 14.7`.
- Separating sources of non-reproducibility: architecture differences from compiler decisions: maybe it makes sense to determinize compiler decisions for a given fixed architecture.

Future of reproducibility

- (Exponentially) increasing diversity of architectures and compilers will make it harder, rather than easier.
- Domain-specific applications (matrix operations, HPC) will always produce highly specialized/customized architectures/compilers that have very little expectations of (and need for) reproducibility.
- Important to realize/accept that reproducibility is certainly not the top concern everywhere (we’d be happy if it got sufficient attention in SOME areas).
- For other domains (controllers, decision-making programs) we can afford more expensive techniques to worry about R.
- Similar like in verification: it is expensive, so rather than verifying everything, we focus on safety-critical applications.

We did not discuss much the relationship with other numeric quality criteria (performance, accuracy), other than observing that we cannot really look at these aspects in isolation.

5 Panel discussions

5.1 Notes from Joint Panel Discussion between the “Analysis and Synthesis of Floating-Point Programs” and “Machine Learning and Formal Methods” Seminars

Alastair F. Donaldson (Imperial College London, GB)

The following notes aim to capture a panel discussion held between concurrently held Dagstuhl seminars on “Analysis and Synthesis of Floating-Point Programs” (17352), and “Machine Learning and Formal Methods” (17351)

The panelists were:

- Eva Darulova (MPI-SWS – Saarbrücken) – representing the Floating-Point seminar
- Miriam Leeser (Northeastern University) – representing the Floating-Point seminar
- Charles Sutton (University of Edinburgh) – representing the Machine Learning and Formal Methods seminar
- Sasa Misailovic (University of Illinois – Urbana Champaign) - representing the Machine Learning and Formal Methods seminar

The panellists gave a brief introduction, which was followed by a 40-minute discussion.

Alastair Donaldson took notes on the questions and answers during this discussion, which are reproduced below. These are merely notes summarising broadly the points that were made and questions asked - they by no means represent verbatim what was said by the various participants.

Alastair Donaldson To what extent is reproducibility of floating-point computations important in machine learning?

Charles Sutton

- Of course floating-point reproducibility would be nice in principle
- However, there are so many sources of approximation in the domain already, so that this is one of many problems
- Given that there are a long list of things that introduce approximation, it would be useful to know when round-off error is moving up this list of problems – techniques to help with that would be useful.

Miriam Leeser Given that the "Machine Learning and Formal Methods" seminar has clear links with verification, I'd be interested in a perspective on numerical issues in the context of verification.

Sasa Misailovic My view, which may not be representative of the entire formal methods community, is that properties are divided into two categories: (1) properties related to safety of computation (no crashes, no out-of-bounds accesses, that mass and gravity do not become negative, etc.), and (2) accuracy of the result, which can sometimes be verified, but can sometimes be let go or estimated, depending on the application domain.

Charles Sutton You can put the arrow in "Machine Learning and Formal Methods" in either direction, so that another approach is to apply machine learning as a tool in formal methods.

Ganesh Gopalakrishan A colleague is going to give a talk in Salt Lake City's public library to members of the public on the topic of the role of computer algorithms in making decisions that might "decide their fate". I'd like a perspective on what machine learning practitioners can say to reassure the general public that certain levels of guarantees, from formal to pragmatic to social, can be provided?

Charles Sutton

- People are starting to think more and more about this, but with no final answers.
- On a social level, if a machine learning method is making sentencing recommendations to judges, how can you argue that it is unbiased? There are several approaches that could be taken – not yet clear which way is best.
- There is also the point of view that the method may not be biased, but rather that the learned model has been trained on biased data.

- In cyber-physical systems the guarantees required are different – in a deep learning model trained in one situation, might a small perturbation cause very different behaviour?
- People are well aware that these issues exist, but it is not clear what to do about it.
- Also there is a pedagogical need to train our students regarding to what we should not do with machine learning and why – a case in point being a paper on arxiv that upset people by trying to predict criminal records based on photos of faces.

Miriam Leeser What do you verify in the context of machine learning applications if you have no ground truth? We have a similar issue in reasoning about software that uses real number approximations.

Sasa Misailovic From a compiler's perspective you can treat the input program as a gold standard, with the rule that whatever transformations you apply you should get something equivalent. It's not clear what the analogue of this would be in machine learning.

Susmit Jha Floating-point round-off can insert biased noise into a machine learning model. Would machine learning experts prefer floating-point arithmetic that doesn't have a deterministic semantics, so that when you do some kind of operation that involves rounding the direction of round-off is random? To the floating-point guys: do you believe non-deterministic semantics to be limiting from a hardware point of view, or can you provide some probabilistic guarantees? To machine learning guys: do you agree that it would be useful to have unbiased rounding?

Jim Demmel A certain civil engineering firm wanted a reproducible parallel linear system solver because their customer was required by lawyers to have reproducibility. Will that come up in e.g. autonomous driving?

Charles Sutton This comes up more in the context of "explainability" rather than "reproducibility" – there is a need to be able to give an explanation for why, though not clear what form that should be.

George Constantinides I'm aware of work in machine learning on using very low precision arithmetic. Usually papers say that they can recover all or almost all of performance by incorporating this into training. Question is: if I could provide you with a 100 fold speedup but far less rigorous error guarantees would you take it? [Laughter in the room.] Secondly, if you could incorporate this in the training, what kind of spec would you like me to provide you with?

Charles Sutton For the first question: all of these methods are intensive to train. Part of reason for impressive DeepMind papers is they have 1000 times more GPUs than anyone else – this gives them a very fast development cycle. If you give me something 100x faster you've made my development cycle much faster. Can you clarify what you mean regarding a spec?

George Constantinides Training relies on properties of the network that is being trained, so what is it that you'd want?

Charles Sutton I'm maybe not best person to ask, but e.g. that noise is unbiased. Might not be a requirement, but seems natural.

Thomas Wahl Coming back to explainability: presumably one aspect of this is providing a level of confidence that I'm aware of noise, and can therefore say that confidence is not high. In floating-point we are good at knowing that a result is likely unstable. My understanding is that there are lots of examples where learners report a wrong decision with high confidence. Is that a problem for you guys?

Charles Sutton It's something we care about. Many learning algorithms we use can report a confidence, but are just learning confidence from data so can be over-confident about wrong answers.

Thomas Wahl This reminds me of a situation in verification where we generate a proof script that can be independently checked by another tool, increasing confidence substantially. Is there an analogue in machine learning?

Charles Sutton That reminds me of "ensemble" methods in machine learning. One thing we know about machine learning is that if we average together different learners we will be better, but it's still just a learner – there is no free lunch.

Eva Darulova are there cases where you have instabilities in training or classification, such that debugging tools would help you get a handle on what is going wrong?

Charles Sutton Could be – often with a deep network I don't even know and blissfully pretend they are not – bursting my bubble might make me sad! [Laughter in the room]. There are cases where we know things are bad – certain linear algebra operations for example. If working with probabilities over structured objects, we may know these things are affected by numerical precision so we do things with logarithms.

Sasa Misailovic My question to you guys [the floating-point seminar participants]: is it possible to check whether an algorithm would converge to a solution after doing some kind of perturbation, e.g. a gross numerical approximation?

George Constantinides What we want is a ranking function argument, and there is some work on introducing round-off error into ranking function arguments.

Miriam Leeser A question for verification people who use machine learning: how worried are you about issues with machine learning being unstable, etc.?

Armando Solar-Lezama Generally we don't use the output of a machine learning component as part of proof. We use it to guide the search process, e.g. searching for a proof or witness or program. Machine learning can be very useful in guiding that search, but ultimately if it is wrong it doesn't matter – it might just slow things down – but you're going to check that the artifact you're looking for is correct independent of machine learning.

[Did not catch speaker] In specification learning you might get false positives as a result of inaccuracy in machine learning, but there are false positives in program analysis anyway.

[Did not catch speaker] I'd add a bit: I'm concerned by this. The solvers we use in the verification community often don't terminate, but if they do we have high confidence in their answers. When we use e.g. belief propagation this don't terminate sometimes; we don't know why – we inject our own hacks and retry. We're not using these for things for which we need proofs – but we might in the future and then this would be a concern.

[Did not catch speaker] In tools for dynamical systems, proofs can come from semi-definite programming. We have experienced proofs that are very numerically sensitive – if you change a digit in an insignificant place, the proof no longer holds. You need to be very careful about such things. If you use machine learning to do invariant learning this problem may also persist.

Martin Vechev What's your take on being able to see a program that [something about neural automata – did not catch the full question]?

Charles Sutton

- You're alluding to two different areas.
- Neural Turing machines – let me unpack that. I wouldn't say they are particularly interpretable. There are a lot of people doing research on interpretability in machine learning, so it's a hot topic.
- You're asking whether it is real or a fad?
- Seems like it would be real – hard to see people trusting deep learning in e.g. precision medicine unless for an individual prediction you could give some evidence.
- You could argue there are cases where you don't care – perhaps with self-driving cars you're happy if they work and satisfy stability properties – but don't want your car to explain everything that it's doing. [Laughter in the room.]

Susmit Jha I've seen work in fixed-point where I can give accuracy bound [for an algorithm] and it can figure out what width to use [to meet this bound in an efficient manner]. Are there similar things in floating-point where I can tell you the expected accuracy and you can figure out most efficient floating-point types?

George Constantinides The question implies that you now have a specification for accuracy requirement – what would such a specification look like?

Susmit Jha As a an extreme I might say that output error should be bounded by actual evaluation with infinite precision. I'd like a compiler that can take my program, compile it into floating-point operations, such that it consumes less energy but meets the spec.

George Constantinides A lot of people in our seminar working on that.

Miriam Leeser But we almost never get specs from users.

Jerry Zhu We typically specify this with an evaluation stack or test set. We can at least say: "on this data you should maintain accuracy to some degree".

Cindy Rubio-González Would you care if it turned out that on other data sets the guarantee would not hold? I.e., we make this work for your training set, but there is no proof it will work in general.

Jerry Zhu Our basic assumption is that things are distributed in an even manner; might be parts of the world where things don't work so well. I would then need to redefine my loss function.

George Constantinides The benefit right now for machine learning applications is to talk about expectation of some error over a data set or some space, rather than worst case error, which is what most tools do.

Susmit Jha In terms of hardware accelerators: is it possible to have an accelerator tune its precision to meet needs of application?

Miriam Leeser there are many accelerators. GPUs are fixed. FPGAs provide the freedom to define what you want. The ability is there, but tool flows are not sophisticated. I don't know about the new tensor processor, but they've picked something and fixed it.

Susmit Jha What are the options?

Miriam Leeser On an FPGA you can do anything, which makes it so hard to write good tools! It depends on what you're targeting.

Sam Elliott Going back to GPUs: if it's needed, the next generation will have it – so GPU vendors can be influenced.

Alastair Donaldson [To Susmit Jha] Did you mean tuning precision in real time?

Susmit Jha Yes – is it possible to reconfigure the FPGA on the fly so that I can do something in this precision then that precision with very short latency?

Miriam Leeser Switching time between designs can be milliseconds, but design time can be days.

Jim Demmel Some problems are well conditioned and can be solved at various precisions, others are not. Are the problems important in machine learning well conditioned?

Charles Sutton By and large yes.

Zvonimir Rakamarić I'm curious what people are doing in this area. For me one interesting question is: once you put a neural network in a safety critical system, how do people analyse this? Are there already capable verification approaches?

Sanjit Seshia There are verification approaches for cases where decisions made by a system are based on outputs from a neural network. Some approaches treat this component as a black box. There is work within the verification community on trying to analyse neural networks in isolation; some of that scales less well than networks used in industrial applications, so there is a gap. I don't think any of it is doing a detailed analysis at the floating-point level.

Martin Vechev There are 3-4 papers I'm aware of but they usually work with small feed forward networks.

Eva Darulova There was a recent paper on CAV related to this, in which verification was over reals while the implementation was over floats.

Martin Vechev [Asked a question about the use of fp16, which I did not catch]

Charles Sutton People must have done experiments on this, as lots of people use fp16 – people have results for particular applications where they use extreme low precision.

George Constantinides I've seen cases where severe roundoff error is akin to a regularizer and can avoid over-fitting.

Miriam Leeser There must be an optimization point there which you can aim for.

Participants

- Erika Abraham
RWTH Aachen, DE
- George A. Constantinides
Imperial College London, GB
- Nasrine Damouche
University of Perpignan, FR
- Eva Darulova
MPI-SWS – Saarbrücken, DE
- James W. Demmel
University of California – Berkeley, US
- Anthony Di Franco
University of California – Davis, US
- Alastair F. Donaldson
Imperial College London, GB
- Theo Drane
Cadence Design Systems – Bracknell, GB
- Sam Elliott
Imagination Technologies – Kings Langley, GB
- Ganesh L. Gopalakrishnan
University of Utah – Salt Lake City, US
- Hui Guo
University of California – Davis, US
- Jeffrey K. Hollingsworth
University of Maryland – College Park, US
- Miriam Leeser
Northeastern University – Boston, US
- Daniel Liew
Imperial College London, GB
- Piotr Luszczek
University of Tennessee – Knoxville, US
- Victor Magron
VERIMAG – Grenoble, FR
- Matthieu Martel
University of Perpignan, FR
- Guillaume Melquiond
INRIA – Gif-sur-Yvette, FR
- David Monniaux
Université Grenoble Alpes – Saint-Martin-d’Hères, FR
- Magnus Myreen
Chalmers University of Technology – Göteborg, SE
- Santosh Nagarakatte
Rutgers University – Piscataway, US
- Pavel Panchekha
University of Washington – Seattle, US
- Sylvie Putot
Ecole Polytechnique – Palaiseau, FR
- Zvonimir Rakamarić
University of Utah – Salt Lake City, US
- Nathalie Revol
ENS – Lyon, FR
- Cindy Rubio-González
University of California – Davis, US
- Daniel Schemmel
RWTH Aachen, DE
- Oscar Soria Dustmann
RWTH Aachen, DE
- Zachary Tatlock
University of Washington – Seattle, US
- Michela Taufer
University of Delaware – Newark, US
- Laura Titolo
National Institute of Aerospace – Hampton, US
- Thomas Wahl
Northeastern University – Boston, US