Adaptive Disk Storage for Interaction Graphs

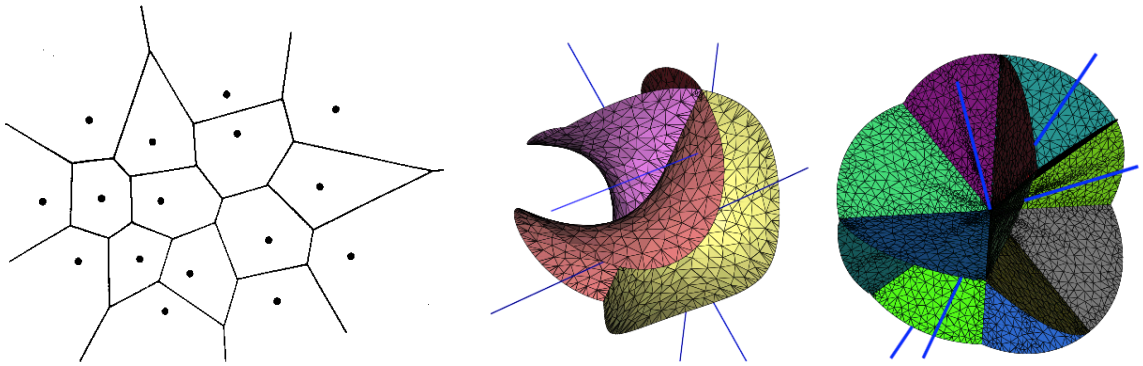
Graph databases are becoming increasingly popular for applications in the world-wide web, social networks, and telecommunications. An interaction graph is an append-only graph, where new edges and vertices are added as time progresses. A good example of an interaction graph is Twitter, where users represent vertices, and edges are mentions of other users

In this project, we are exploring ways to adapt the disk storage for interaction graphs by monitoring the query workload, and changing the layout based on historical trends. The goal is to minimize I/O costs, while respecting the storage constraints. The work will include mathematically formulating a cost model, designing an algorithm for adaptation, and a lot of hacking in C++.

Professor: Robert Soulé

UROP -- The Farthest-line Voronoi diagram in 3D

The Voronoi diagram is a powerful geometric object with numerous applications, ranging from computer graphics, robotics and CAD to diverse areas in science and engineering. Given a set of point sites in the plane, the classic Voronoi diagram is a partitioning of the plane into regions, one for each site, such that the Voronoi region of a site s is the locus of points closer to s than to any other site. For an example, see the figure on the left – remember also the Voronoi game in our 10 year anniversary. The diagram provides nearest-neighbor information in compact form.



In this project we deal with lines in 3 dimensions (see the figure on the center and right) and the farthest Voronoi diagram, where you want to know the site farthest away instead of closest. We aim to build the 3D farthest-line Voronoi diagram in Maple, and to visualize it in Maya. We need to compute the intersection of two quadric surfaces in 3D but we already have software from INRIA to perform this task. We plan to implement an algorithm and build the diagram based on this software. The summer project will focus on visualizing the diagram, where a big room for creativity is available for the student. Note that the structural complexity of the nearest-neighbor diagram is a long-standing open problem. We focus on the simpler farthest version.

Skills required:

- Courses: Programming Fundamentals II, Algorithms & Data Structures, Discrete Mathematics
- Interest in geometry, visualization, and no fear to face difficulties.
- Any knowledge of Maple, Matlab, C++ would be helpful but not necessary at all. The student will get familiar with such tools.
-

Benefits:

- Working towards a problem, on which many researchers are currently scratching their heads.
- Studying a fundamental geometric data structure
- Gaining the experience of scientific programming.
- Collaborating with a research group abroad (the Technion)

Contact: Prof. Evanthia Papadopoulou, SI-209, evanthia.papadopoulou@usi.ch
Mentor: Elena Khramtcova, SI-216, elena.khramtcova@usi.ch

Developing a web based tool for visualizing Dynamic Semantic Networks

Professors: Prof. Mehdi Jazayeri
Assistants: Saman Kamran – saman.kamran@usi.ch
Startup Project: COD (cod.inf.usi.ch)

Abstract:

The Semantic Web is a vision that has sparked a wide-ranging enthusiasm for a new generation of the Web. The central idea of the Semantic Web vision is to make the web more understandable to computer programs so that people can make more use of this gigantic asset. One of the most useful tools for visualizing large semantic data is Semantic Networks. A semantic network, or frame network, is a network that represents semantic relations between concepts. This is often used as a form of knowledge representation. It is a directed or undirected graph consisting of vertices, which represent concepts, and edges, which represent semantic relations between them. The challenging part of Semantic networks visualization is to optimize the User Experience (UX) for exploring and exploiting information from a large network.

In this project you will design and implement a customized online tool for representing semantic network, which is updating dynamically on the web. This tool will be used as a part of a larger platform (COD) that is customized for exchanging knowledge and ideas. Some specifications of the platform are:

Filtering: The main network is very large. Therefore the visualization of partial networks should be considered based on filtering defined parameters.

Usability: During the design phase a usability study is needed to enable users to access the needed information easily and with an optimized interaction in the large network.

Dynamicity: The Semantic Network visualization will be dynamic and change based on the knowledge base updates. So the platform should be able to read the knowledge base (from json files) and update the modified sections dynamically with a reasonable performance.

At the start you will receive full specifications of the project. He will also supervise the project constantly. This project enhances your Web development skills and generally gives you a broader view on the main steps of developing an industrial web application, which might be extensible for your future work.

Breaking Symmetries in SMT Solvers

Prof. Natasha Sharygina and Dr. Antti Hyvärinen

Satisfiability Module Theories (SMT) solvers are widely used as a tool for computing combinatorial subproblems emerging from a multitude of applications, such as software verification and AI planning.

One of the recent directions in SMT solver research is based on an observation that in many applications the problem contains large amounts of symmetries. The symmetries result in the solver covering unnecessarily redundant search space, significantly increasing the solving time. The goal of this UROP project is to study how symmetries can be broken using new heuristic methods.

The work, including experimentation and implementation, will be carried out as part of the ongoing development effort of the OpenSMT solver from the Verification group at USI.

We are looking for a motivated student who wants to improve his/her knowledge on software verification and constraint-based search. This project will give the student an excellent overview of a quickly developing field while being sufficiently approachable. Prior knowledge in SMT modeling is not required, though is a plus.

The aim of this project is to integrate symmetry breaking to OpenSMT. The student will be coached while:

1. Getting familiar with the internals of SMT solvers
2. Implementing existing and developing new algorithms for symmetry detection and breaking. For this task preliminary experience with C/C++ is required.
3. Designing and running a set of experiments.

Developing a smart versioning system

Prof. Natasha Sharygina and Grigory Fedyukovich

During its lifetime any program evolves. A developer can fix a bug, apply optimizations, or add a new functionality to the existent code. After such changes the developer should be sure that the new code is correct, i.e., it does not break any old functionality which is supposed to be preserved in the new version. In order to do it efficiently, a tool called eVolCheck [1] has been developed by researchers of Formal Verification and Security group at USI. eVolCheck formally verifies each version of a program and reuses some efforts (namely, function summaries) between consequent runs. For example, if just one function is changed in the program, it may be enough to re-check only the new code of this function, and do not touch the (preserved) rest of the program.

In the GUI of eVolCheck integrated with Eclipse IDE [2], the process is visualized as follows. First, the syntactic difference between the current version and the previous one is shown. Depending on the amount of the modified code, the user decides to run the upgrade checking tool. Then the Eclipse IDE internally runs eVolCheck, it configures the tool automatically and keeps the settings in a subsidiary storage, so the user does not need to adjust the environment for every check. After eVolCheck completes its tasks, the positive/negative result is returned to the user. In the former case, the change impact (namely, the number of re-checked summaries) is displayed - it represents the semantic influence of the change on the whole program. In the latter case, if a bug is found, the Eclipse IDE shows the trace to the counter-example.

Software versioning is the way to distinguish between the program versions [3]. Software versioning systems (e.g., SVN [4] or Git [5]), integrated to different IDEs, are widely used by software developers in industry and academia. But since it is always up to the user, whether to make a new revision, the existent versioning systems cannot give guarantees that the correspondent program version is correct. We propose to extend one of the existent software versioning systems to support program verification by eVolCheck. We will call it a “smart versioning system”.

The proposed smart versioning system should work as follows. If a current version of the program is proven correct, it might be a good justification to create a new revision in the versioning system. Otherwise,

if at least one assertion in the version is violated (i.e., a counter-example is given), the user should analyze the counter-example, then fix it, and finally run eVolCheck once again. So the revisions in the versioning system will become trustworthy. In our proposed smart versioning system, every new revision will be confirmed safe. We propose the student to implement such technology on the top of the existent tools.

We are sure the proposed technology can be easily implemented because its main ingredients already exist: the Eclipse IDE with available source code and APIs, eVolCheck, and eVolCheck plugin to Eclipse. The student should investigate how to use software versioning systems, and how these systems are being handled by Eclipse IDE. We believe such an integration will make easier to apply formal verification in most software development projects, which are currently unfortunately still not able to use this powerful technology. And this will ultimately increase the quality of software being developed.

There are two main benefits for a student who is going to implement the proposed technology. First, the student will become familiar with formal verification and state-of-the-art verification systems. It will give the young researcher a taste of what kind of research is being done in the modern Computer Science. No theoretical background in Formal Verification is required. Second, the student will earn an important experience in software development. Eclipse IDE is known in developers community, and experience in working with its source code will indicate the student is familiar with modern trends in software development. For this, initial experience in Java programming language is required.

References:

[1] G. Fedyukovich, O. Sery, and N. Sharygina. eVolCheck: Incremental Upgrade Checker for C. In *TACAS'13*
[2] http://eclipse.org/
[3] http://en.wikipedia.org/wiki/Software_versioning
[4] http://tortoisesvn.net/
[5] http://git-scm.com/

Developing your own SMT-based Model Checker

Prof. Natasha Sharygina and Grigory Fedyukovich

Model Checking [1] is a well-known scientific approach to check safety of a program. It is a fully automatic approach to decide whether a program is safe with respect to a given assertion or to provide a witness of the bug. Assertion is a logical expression over program variables in a particular program location. The set of well-known model checkers for software [2,3,4,5,6] includes FunFrog [5] and eVolCheck [6], developed at Formal Verification and Security group at USI.

In a nutshell, FunFrog delegates the problem of checking safety to an external decision procedure (more concretely, a SAT solver) called PeRIPLO [7]. SAT solving allows precise reasoning over the program variables and arithmetic operations, but requires an expensive encoding the program into a bit-blasted form. As a result, the SAT formulas produced even from the simplest C programs become large and incomprehensible. Alternatively, there exist SAT modulo theories (SMT) solvers that can deal with reals and integers without bit-blasting. One of the most known SMT solvers, Z3 [8], provides an understandable interactive framework available online.

We propose to extend FunFrog to support SMT encoding. The student will be given an access to the sources of the model checker, and will be taught the algorithms used in the SAT/SMT encoding. Then the student will be asked to implement new algorithms and evaluate the resulting tool. We believe, the support of SMT solving will drastically improve the performance of FunFrog for certain benchmarks. In such a case, the results of the student's work will be used as a basis for a publication in a highly ranked scientific conference in Model Checking. Thus, if the student is planning to start a research career, the experience with SAT/SMT techniques will be useful.

While this project does not require any theoretical background in Formal Methods, we will appreciate if the student already has some background knowledge in Logics and Theory of Computation. Initial experience in C++ is required.

References:

[1] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999

[2] E. Clarke, D. Kroening, and F. Lerda. A tool for checking ANSI-C programs. In *Tools and Alg. for Con. and Anal. of Sys. (TACAS '04)*, LNCS, pages 168–176, 2004.
[3] E. Clarke, D. Kroening, N. Sharygina, and K. Yorav. SATABS: SAT-based Predicate Abstraction for ANSI-C. In *Tools and Alg. for Con. and Anal. of Sys. (TACAS '05)*, LNCS, pages 570–574, 2005.
[4] D. Beyer and M. E. Keremoglu. CPAchecker: A Tool for Configurable Software Verification. In *Computer Aided Verification (CAV '11)*, LNCS, pages 184–190, 2011.
[5] O. Sery, G. Fedyukovich, and N. Sharygina. FunFrog: Bounded Model Checking with Interpolation-based Function Summarization, In *ATVA'12*
[6] G. Fedyukovich, O. Sery, and N. Sharygina. eVolCheck: Incremental Upgrade Checker for C. In *TACAS'13*
[7] http://verify.inf.unisi.ch/periplo.html
[8] http://rise4fun.com/z3

Computational Biology and Drug Design

Vittorio Limongelli

website: http://inf.usi.ch/faculty/limongelli

e-mail: vittoriolimongelli@gmail.com

Computer simulations have increased over the years their role in studying biologically relevant phenomena (1-6). In this project the student will perceive how one can use atomistic simulations to describe the binding of ligands to protein and design new drugs. We will focus on one or more members of the G-protein coupled receptor family, GPCRs, which is a large protein family involved in many diseases and targeted by approximately 40% of marketed drugs. Despite their biological relevance, the ligand binding mechanism to these receptors is poorly understood owing to the limited structural data available. However, this information is of paramount relevance to guide experiments and drug design strategies. Combining computational techniques such as homology modeling, molecular docking and molecular dynamics simulations we will investigate the binding mechanism of some known drugs to their molecular target. This understanding would allow designing more potent and selective compounds.

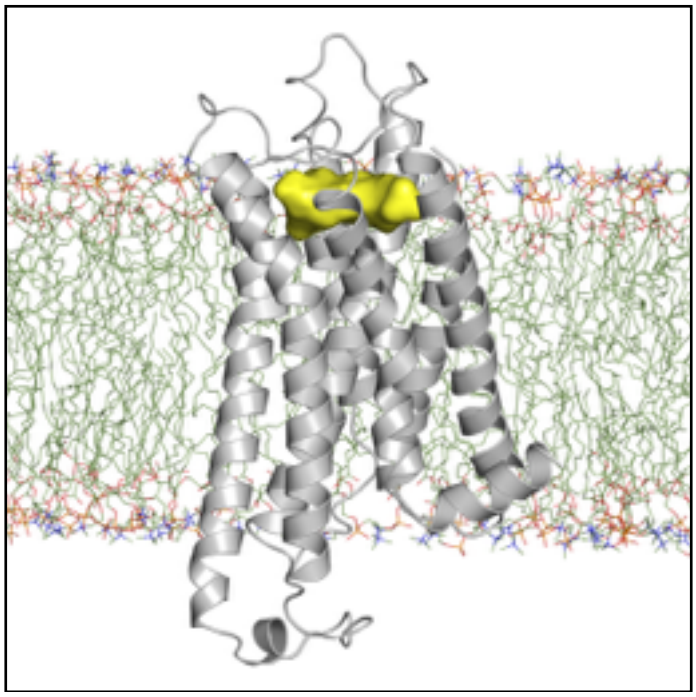


Figure. Tridimensional structure of a GPCR embedded in phospholipid bilayer. Protein is represented as grey cartoon, whilst lipids as licorice. The ligand binding site is highlighted as yellow surface.

References

1. Troussicot et al. *J. Am. Chem. Soc.* **2015** *(ahead of print)*
2. D'Amore et al. *J. Med. Chem.* **2014**, *57*, 937-954
3. Limongelli, Bonomi M, Parrinello M. *Proc. Natl. Acad. Sci. USA* **2013**, *110*, 6358-6363
4. Limongelli et al. *Angew. Chem. Int. Ed. Eng.* **2013**, *125*, 2325-2329
5. Limongelli et al. *Proc. Natl. Acad. Sci. USA* **2012**, *109*, 1467-1472
6. Limongelli et al. *Proc. Natl. Acad. Sci. USA* **2010**, *107*, 5411-5416

UROP Project Proposal – SHARING21.A

Sharing Physical Objects Using Mobile Augmented Reality

Searching for objects in the real world is a common problem that is time consuming and may induce anxiety. This is particularly often the case with shared household objects such as basement keys or DIY tools, which may be used by several family members, or even neighbors.

Within the SNF-funded SHARING21 project, we are looking into new ways of supporting sharing both digital information and physical objects. The UROP project SHARING21.A is about designing, developing, and evaluating an Augmented Reality (AR) system that supports shared access to physical household items. The envisioned approach is to overlay available information about missing items at the place they are usually located. For example, a garage key might usually hang on a peg close to the front door. If it is not there, watching the spot through an AR overlay puts available information in its place, e.g., the last mobile phone that was seen close to the key (or a picture of the phone's owner). Similar visualization could be used for other items that have a dedicated spot in the house, such as door access cards, power drills, or umbrellas, or maybe even books on shelves.

This UROP project provides a prime opportunity for a student to work on an exciting research project involving mobile technologies. We are looking for a student with good iOS/Android development skills, or a strong willingness to learn.

UROP Project Proposal – SHARING21.B

Co-Located AR-based Display Sharing

Today's mobile devices such as smartphones, tablets, and laptops support the collaborative work of co-located people, e.g., across a table surface. One drawback of such shared workspaces is the lack of privacy – which is particularly important when such collaboration is being conducted in public places such as cafes.

Within the SNF-funded SHARING21 project, we are looking into new ways of supporting sharing both digital information and physical objects. The UROP project SHARING21.B looks into designing, developing, and evaluating an Augmented Reality (AR) system that supports shared interaction with digital content. Two or more AR devices (e.g., glasses or – in a first iteration – a smartphone or tablet) are synchronized to view the same digital content, such as a movie or the content of a file. Simple gestures should allow collaborators to interact with this content.

This UROP project provides a prime opportunity for a student to work on an exciting research project involving mobile technologies. We are looking for a student with good iOS/Android development skills, or a strong willingness to learn. Input should be acquired using a Microsoft Kinect, hence experience with this platform or a strong willingness to learn is essential.