Prof. Dr. Jasmin Blanchette
Dr. Martin Desharnais-Schäfer
Dr. Michael Kirsten
Elisabeth Lempa

Ludwig-Maximilians-Universität München
Institut für Informatik
Discussion on 07.01.2026
Homework due on 14.01.2026 at 16:00

Possible solution for Exercise Sheet 11 in Scientific and Technical English for Computer Scientists

The exercise sheets consist of in-class exercises and homework. The in-class exercises take place during the second half of the lecture time slots. The homework, which is optional and ungraded, can be submitted via the "Homework" section in Moodle. The homework is subject to peer review.

Unless indicated otherwise, generative artificial intelligence assistants such as ChatGPT may be used, as long as you acknowledge how you use them as specified by the Institute's policy on plagiarism.[1] However, you may not use such tools to generate peer reviews for you. In addition, we strongly recommend that you do not use them to generate entire solutions, since that would defeat the purpose of the exercises.

In-class exercise 11-1 ***Missing Titles*** Suggest titles for five papers based on their abstracts, given below. The titles may be catchy, informative, or double-barreled.

a) A two-party cryptographic protocol for evaluating any binary gate is presented. It is more efficient than previous two-party computations, and can even perform single-party (i.e. satisfiability) proofs more efficiently than known techniques. As in all earlier multiparty computations and satisfiability protocols, commitments are a fundamental building block. Each party in our approach encodes a single input bit as 2 bit commitments. These are then combined to form 5 bit commitments, which are permuted, and can then be opened to reveal the output of the gate.

POSSIBLE SOLUTION:

An Efficient Two-Party Cryptographic Protocol for Computing Binary Gates

(Abstract from Bert den Boer, "More Efficient Match-Making and Satisfiability—The Five Card Trick," *EUROCRYPT 1989*, volume 434 of LNCS, pp. 208–217, Springer, 1989.)

[1] `https://www.medien.ifi.lmu.de/lehre/Plagiate-IfI.pdf`

b) As a vast number of ingredients exist in the culinary world, there are countless food ingredient pairings, but only a small number of pairings have been adopted by chefs and studied by food researchers. In this work, we propose *KitcheNette* which is a model that predicts food ingredient pairing scores and recommends optimal ingredient pairings. KitcheNette employs Siamese neural networks and is trained on our annotated dataset containing 300K scores of pairings generated from numerous ingredients in food recipes. As the results demonstrate, our model not only outperforms other baseline models, but also can recommend complementary food pairings and discover novel ingredient pairings.

POSSIBLE SOLUTION:

KitcheNette: A Learning Model for Recommending Food Ingredient Pairings

(Abstract from Donghyeon Park, Keonwoo Kim, Yonggyu Park, Jungwoon Shin, and Jaewoo Kang, "KitcheNette: Predicting and Ranking Food Ingredient Pairings Using Siamese Neural Network," *IJCAI 2019*, pp. 5930–5936, ijcai.org, 2019.)

c) We develop and explain the design decisions of a framework for the formal specification and analysis of interactions in generalized distributed systems. Our approach is suitable to reason about various types of agents and supports the modeling of synchronous interactions between any finite number of agents. Our proposal provides a common ground for existing modeling techniques for cryptographic protocols and security ceremonies, by generalizing and unifying them in a single formalism. We discuss the specification of security properties in our framework and demonstrate on a short voting ceremony a technique to model our ceremonies in the Tamarin prover.

POSSIBLE SOLUTION:

A Framework for Formalizing Cryptographic Protocols and Security Ceremonies

(Abstract from Barbara Fila and Sasa Radomirović, "Nothing Is Out-of-Band: Formal Modeling of Ceremonies," *CSF 2024*, pp. 464–478, IEEE, 2024.)

d) The Lean mathematical library Mathlib features extensive use of the typeclass pattern for organising mathematical structures, based on Lean's mechanism of instance parameters. Related mechanisms for typeclasses are available in other provers including Agda, Coq and Isabelle with varying degrees of adoption. This paper analyses representative examples of design patterns involving instance parameters in the finalized Lean 3 version of Mathlib, focussing on complications arising at scale and how the Mathlib community deals with them.

POSSIBLE SOLUTION:

Typeclass Design Patterns in Lean's Mathlib

(Abstract from Anne Baanen, "Use and Abuse of Instance Parameters in the Lean Mathematical Library," *Journal of Automated Reasoning* 69(1), article 1, 2025.)

e) We study transformational program logics for correctness and incorrectness that we extend to explicitly handle both termination and nontermination. We show that the logics are abstract interpretations of the right image transformer for a natural relational semantics covering both finite and infinite executions. This understanding of logics as abstractions of a semantics facilitates their comparisons through their respective abstractions of the semantics (rather that the much more difficult comparison through their formal proof systems). More importantly, the formalization provides a calculational method for constructively designing the sound and complete formal proof system by abstraction of the semantics. As an example, we extend Hoare logic to cover all possible behaviors of nondeterministic programs and design a new precondition (in)correctness logic.

POSSIBLE SOLUTION:

Understanding Program Logics as Abstract Interpretations

(Abstract from Patrick Cousot, "Calculational Design of [In]correctness Transformational Program Logics by Abstract Interpretation," *POPL 2024*, article 7, 2024.)

In-class exercise 11-2 ***Skip Lists*** Read the Wikipedia page for the skip list data structure.[2]

Pretend you are the inventor of skip lists. Suggest a title for a paper introducing skip lists and write an abstract.

[2]`https://en.wikipedia.org/wiki/Skip_list`

POSSIBLE SOLUTION:

Don’t Skip This One: A New Data Structure with Fast Search and Insert

Until today, programmers who desired search in logarithm time stored their list-shaped data in a sorted array, and those who wanted fast, dynamic insertion turned to the linked list instead. We introduce an alternative: The skip list combines the strengths of static arrays and linked lists in one data structure. By maintaining a hierarchy of sparse subsequences, skip lists allow elements to be searched for in logarithmic time. Storing data in linked lists makes insertion possible. We give two variants that differ in the selection of skipped elements, a probabilistic one and a deterministic one. Thus, skip lists give the same asymptotic expected time bounds as balanced trees while being simpler, faster, and more space-efficient.