שם הקורס: Program Synthesis and Automated Reasoning
ניתוח תכניות וסינתזה של תוכנה

מספר הקורס: 236347

סמסטר: אביב תשפ"ג

מרצה:	שחר יצחקי
שעות הרצאה:	ב׳ 14:30 - 16:30
שעת תרגול:	לפי בקשה
דרישות קדם:	תורת הקומפילציה (צמוד)
אתר הקורס:	https://moodle.technion.ac.il/course/view.php?id=5865

תאור הקורס

הקורס ילמד בשני פרקים עיקריים:

- שיטות וכלים בתחומי הוכחה פורמלית של תכונות של תכניות מחשב וסמנטיקה פורמלית של שפות תכנות. הנושאים יכללו: תחשיב למבדא, טיפוסים, פולימורפיזם, Type theory, הצרנת סמנטיקה בלוגיקה מסדר ראשון, Satisfiability modulo theory.
- סינתזת תוכנה — ייצור אוטומטי של קוד על-ידי מהדר. הנושאים יכללו: Syntax-guided synthesis, תכנות עפ״י דוגמאות, Counterexample guided inductive synthesis, Type directed synthesis.

בקורס ייעשה שימוש בכלים הבאים: Coq, Z3, Sketch, Synquid

The course consists of two major divisions:

- Automated Reasoning: tools and techniques for formal reasoning about computer programs and programming language semantics. Topics include: Lambda calculus, type checking, polymorphism, Type Theory, formalizing programs in first-order logic, Satisfiability Modulo Theory.
- Program Synthesis: automatic generation of program code using by a synthesizing compiler. Topics include: Syntax-guided synthesis, programming by example, counterexample-guided inductive synthesis, type-directed and deductive synthesis.

Throughout the course, we will learn and use the following software tools: Coq, Z3, Sketch, and Synquid.

דרישות הקורס

השתתפות בפתרון תרגילי כיתה (לפחות 75%, ללא ציון)
תרגילי בית (40% מהציון הסופי)
פרויקט מסכם (תכנותי — שימוש בכלים שנלמדו; 60% מהציון הסופי)

רשימת ספרות

Formal Reasoning About Programs / Adam Chlipala
http://adam.chlipala.net/frap/

Software Foundations / Benjamin C. Pierce *et al.*
Volumes 1, 2 https://softwarefoundations.cis.upenn.edu

תוצרי למידה

בסיום הקורס, הסטודנטים —

1. יבינו את הבניה של הוכחות בלוגיקה פורמלית מתוך עקרונות מתמטיים יסודיים, ואת ההבדלים בין הוכחות פורמליות לבין הוכחות "רגילות" עם דף ועט.
2. יכירו את המושגים לוגיקה קונסטרוקטיבית ו-Martin-Löf Type Theory, בדגש על ההבדלים בינה לבין לוגיקה קלאסית ועל המתאם הוכחות-תכניות.
3. יפתחו מיומנות הוכחת טענות באמצעות Coq Proof Assistant, מערכת להצרנה מכנית של הוכחות המבוססת על Dependent Types.
4. ילמדו על הצרנה של תכונות המתייחסות לתכניות מחשב, באמצעות שני סוגים מקובלים סמנטיקה פורמלית: סמנטיקה ביצועית וסמנטיקה אקסיומטית; וכן יוכיחו טענות מסוג זה הן בעזרת Coq והן בהתבסס על הוכחה אוטומטית תוך שימוש בכלי SMT.
5. ידעו לסווג את הגישות השונות לסינתזה של תוכנה, ויכירו דוגמאות של אלגוריתמים המיישמים כל אחת מן הגישות הללו.
6. יתנסו בסינתזה של תכניות בעזרת של כלי סינתזה: Sketch ו-Synquid.
7. יפתחו בעצמם כלי סינתזה של תוכנה בהתבסס על אלגוריתמים ואופטימיזציות שנלמדו, במסגרת פרויקט מסכם.

By the end of the course, students will —

1. Understand the mathematical construction of formal proofs from first principles, and how they differ from pen-and-paper proofs.
2. Be familiar with the terms Constructive Logic and Martin-Löf Type Theory; esp. how they differ from classical logic(s), and the proofs-as-programs correspondence.
3. Develop skills for proving logical propositions using the Coq Proof Assistant.
4. Learn to formalize properties of computer programs using Operational and Axiomatic Semantics, and prove such properties using both manually (in Coq) and automatically (using SMT solvers).
5. Know the taxonomy of software synthesis techniques and representative algorithms.
6. Experience program synthesis using Sketch and Synquid.
7. Design and code a synthesis tool from the grounds up, based on knowledge acquired in class.

פירוט תוכן הקורס לפי שבועות

Week	Title	Tech.
1	Overview of Automated Reasoning and Synthesis	
2	Pure Lambda Calculus	
3	Simply-typed Lambda Calculus + Polymorphism	
4	Theory of Dependent Types	Coq
5	Type Theory and the Curry-Howard Correspondence	
6	Reasoning About Programs: Transition Systems	
7	Hoare Logic	
8	Satisfiability Modulo Theory (SMT) and its application to verification	Z3
9	Synthesis: Programming By Example, Observational Equivalence	Sketch
10	Syntax-Guided Synthesis and Counterexample Guided Inductive Synthesis	
12	Deductive and Type-driven Synthesis	Synquid
13	Proof-guided Deductive Synthesis	SuSLik