CS771 : Foundations of Formal Methods 2021

Lecture 0: Introduction and logistics

Instructor: S. Akshay

IIT Bombay, India

26-07-2021

CS771: Foundations of Verification and Automated Reasoning (FVAR)

Also called Foundations of Formal Methods

This course in a nutshell

1. How to reason

CS771: Foundations of Verification and Automated Reasoning (FVAR)

Also called Foundations of Formal Methods

This course in a nutshell

1. How to reason
   - about human thought

CS771: Foundations of Verification and Automated Reasoning (FVAR)

Also called Foundations of Formal Methods

This course in a nutshell

1. How to reason
   - about human thought – isn't that philosophy??

CS771: Foundations of Verification and Automated Reasoning (FVAR)

Also called Foundations of Formal Methods

This course in a nutshell

1. How to reason
   - about human thought – isn't that philosophy??
   - about computing objects: systems or machines or programs

CS771: Foundations of Verification and Automated Reasoning (FVAR)

Also called Foundations of Formal Methods

This course in a nutshell

1. How to reason
   - about human thought – isn't that philosophy??
   - about computing objects: systems or machines or programs
2. How to automate this reasoning

CS771: Foundations of Verification and Automated Reasoning (FVAR)

This course in a nutshell

1. How to reason
   - about human thought – isn't that philosophy??
   - about computing objects: systems or machines or programs
2. How to automate this reasoning

OMG!: Machines reasoning about machines? Is this AI?!

CS771: Foundations of Verification and Automated Reasoning (FVAR)

This course in a nutshell

1. How to reason
   - about human thought – isn't that philosophy??
   - about computing objects: systems or machines or programs
2. How to automate this reasoning

OMG!: Machines reasoning about machines? Is this AI?!

- Not quite: but as we will see, we can use these techniques to reason about AI too!

CS771: Foundations of Verification and Automated Reasoning (FVAR)

This course in a nutshell

1. How to reason
    - about human thought – isn't that philosophy??
    - about computing objects: systems or machines or programs
2. How to automate this reasoning

What we need are

1. languages to formalize reasoning: symbolic logic
2. mathematical models of the computing objects: abstractions, automata
3. algorithms and tools to automate this whole process!

What this course is not about and what it is about

The idea of this course is not to...

- ... cover a specific topic in depth
- ... exhaustively cover all logics, automata, models

The idea of this course is to...

- ... provide a breadth-first view of this entire field of formal methods
- ... look at some logics, techniques in detail
- ... prepare students to do advanced courses and R&D/MTP in this area!

Who uses Formal Methods?

Widely used in industry

- Hardware and software verification
- Cyber-physical systems: Avionics, Automobiles, space!
- Hot area: explainability in AI/ML!

Including Amazon, Intel, IBM, Microsoft Research, Google, Samsung, TCS, NASA, Mathworks ...

Widely used in Academia too

- FM groups in all major universities around the world
- An interplay of mathematics and computer science, logic and coding, theory and practice.

For a more colorful video, see
`https://www.youtube.com/watch?v=AM_gwEKjnGY`

Course structure

Topics to be covered in this course:

- Module 1: Logics to formalize reasoning
- Module 2: The problem of satisfiability
- Module 3: Program verification
- Module 4: Automata-based model checking of systems
- Extra Module: Advanced topics on automata and computability

Course structure

Topics to be covered in this course:

- Module 1: Logics to formalize reasoning
- Module 2: The problem of satisfiability
- Module 3: Program verification
- Module 4: Automata-based model checking of systems
- Extra Module: Advanced topics on automata and computability

There are advanced courses in each of these topics! CS433, CS615, CS713, CS735, CS738, CS739 But this course is the ring that binds them all.... :-)


*

Course logistics

Mode of instruction

- Combination of: Detailed slides + live lectures (recorded) + some video voiceovers, $\sim$2-3 hrs per week
- Tutorials/Doubt clearance sessions: $\sim$0.5-1 hr per week
- Weekly reading & writing assignments, self study: 3+ hrs per week, compulsory!
- Office hours for further doubts

Course logistics

Mode of instruction

- Combination of: Detailed slides + live lectures (recorded) + some video voiceovers, ∼2-3 hrs per week
- Tutorials/Doubt clearance sessions: ∼0.5-1 hr per week
- Weekly reading & writing assignments, self study: 3+ hrs per week, compulsory!
- Office hours for further doubts

Evaluation Disclaimer: Tentative!

Course logistics

Mode of instruction

- Combination of: Detailed slides + live lectures (recorded) + some video voiceovers, ~2-3 hrs per week
- Tutorials/Doubt clearance sessions: ~0.5-1 hr per week
- Weekly reading & writing assignments, self study: 3+ hrs per week, compulsory!
- Office hours for further doubts

Evaluation Disclaimer: Tentative!

- Moodle progress quizzes : 10%
- Home works and submittable assignments : 20%
- Paper presentation (in lieu of midsem): 25%
- Final exam / Presentation + viva: 35%

Course logistics

Mode of instruction

- Combination of: Detailed slides + live lectures (recorded) + some video voiceovers, $\sim$2-3 hrs per week
- Tutorials/Doubt clearance sessions: $\sim$0.5-1 hr per week
- Weekly reading & writing assignments, self study: 3+ hrs per week, compulsory!
- Office hours for further doubts

Evaluation Disclaimer: Tentative!

- Moodle progress quizzes : 10%
- Home works and submittable assignments : 20%
- Paper presentation (in lieu of midsem): 25%
- Final exam / Presentation + viva: 35%
- Other: Class participation, interactions : 0-10%

Module 1: Logic, a language for reasoning

1. Men are mortal
2. Socrates is a man

Socrates is mortal

Module 1: Logic, a language for reasoning

1. Men are mortal
2. Socrates is a man

Socrates is mortal

Intuitive Rule/Pattern:

1. α are β
2. γ is an α

γ is β

Module 1: Logic, a language for reasoning

1. Men are mortal
2. Socrates is a man

Socrates is mortal

Intuitive Rule/Pattern:

1. α are β
2. γ is an α

γ is β

1. A barber shaves all those who don't shave themselves
2. The barber needs a shave

Who shaves the barber?

Module 1: Logic, a language for reasoning

1. Men are mortal
2. Socrates is a man

Socrates is mortal

Intuitive Rule/Pattern:

1. α are β
2. γ is an α

γ is β

1. A barber shaves all those who don't shave themselves
2. The barber needs a shave

Who shaves the barber?

A language zoo for reasoning: Symbolic logic

- Propositional logic
- Predicate logic
- Temporal logic
- Modal logic

Module 2: Satisfiability

Boolean Propositional Satisfiability

- Is a sentence written in the propositional logic satisfiable?
- That is, is there an assignment that evaluates it to true?

Module 2: Satisfiability

Boolean Propositional Satisfiability

- Is a sentence written in the propositional logic satisfiable?
- That is, is there an assignment that evaluates it to true?
- A central problem in Algorithms, complexity - NP-complete!

Module 2: Satisfiability

Boolean Propositional Satisfiability

- Is a sentence written in the propositional logic satisfiable?
- That is, is there an assignment that evaluates it to true?
- A central problem in Algorithms, complexity - NP-complete!
- Applications paramount – A whole book of problems that can be reduced to it.

Module 2: Satisfiability

Boolean Propositional Satisfiability

- Is a sentence written in the propositional logic satisfiable?
- That is, is there an assignment that evaluates it to true?
- A central problem in Algorithms, complexity - NP-complete!
- Applications paramount – A whole book of problems that can be reduced to it.
- Our formal reasoning, via logic, will lead to building efficient algorithms that solve many instances of this problem easily!

Module 2: Satisfiability

Boolean Propositional Satisfiability

- Is a sentence written in the propositional logic satisfiable?
- That is, is there an assignment that evaluates it to true?
- A central problem in Algorithms, complexity - NP-complete!
- Applications paramount – A whole book of problems that can be reduced to it.
- Our formal reasoning, via logic, will lead to building efficient algorithms that solve many instances of this problem easily!

Theory behind powerful solvers

- SAT solvers
- SMT solvers
- Pseudo-Boolean solvers?

Module 3: Application to reasoning about programs

```
int foo(int n) {
    int k, j;
    k = 0; j = 1;
    while (k != n) {
        k = k + 1;
        j = 2*j;
    }
    return(j);
```

```
list * bar(list * i) {
    list *j, *k;
    j = NULL;
    while (i != NULL) {
        k = i→next;
        i→next = j;
        j = i;
        i = k;
    }
    return(i)
```

- If "foo" is called with $n > 0$, does it always return 2^n?
- If "bar" is called with i pointing to an acyclic list, does i always point to an acyclic list when "bar" returns?

Module 3: Application to reasoning about programs

```
int foo(int n) {
    int k, j;
    k = 0; j = 1;
    while (k != n) {
        k = k + 1;
        j = 2*j;
    }
    return(j);
```

```
list * bar(list * i) {
    list *j, *k;
    j = NULL;
    while (i != NULL) {
        k = i→next;
        i→next = j;
        j = i;
        i = k;
    }
    return(i)
```

- If "foo" is called with $n > 0$, does it always return 2^n?
- If "bar" is called with i pointing to an acyclic list, does i always point to an acyclic list when "bar" returns?

- In general, we can't answer this (why?), but then what?

Module 3: Application to reasoning about programs

```
int foo(int n) {
    int k, j;
    k = 0; j = 1;
    while (k != n) {
        k = k + 1;
        j = 2*j;
    }
    return(j);
```

```
list * bar(list * i) {
    list *j, *k;
    j = NULL;
    while (i != NULL) {
        k = i→next;
        i→next = j;
        j = i;
        i = k;
    }
    return(i)
```

- If "foo" is called with $n > 0$, does it always return 2^n?
- If "bar" is called with i pointing to an acyclic list, does i always point to an acyclic list when "bar" returns?

- In general, we can't answer this (why?), but then what?
- The "art and science" of proving programs (in)correct

Module 3: Application to reasoning about programs

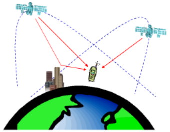
```
int foo(int n) {
    int k, j;
    k = 0; j = 1;
    while (k != n) {
        k = k + 1;
        j = 2*j;
    }
    return(j);
```

```
list * bar(list * i) {
    list *j, *k;
    j = NULL;
    while (i != NULL) {
        k = i→next;
        i→next = j;
        j = i;
        i = k;
    }
    return(i)
```

- If “foo” is called with $n > 0$, does it always return 2^n?
- If “bar” is called with i pointing to an acyclic list, does i always point to an acyclic list when “bar” returns?

- In general, we can't answer this (why?), but then what?
- The “art and science” of proving programs (in)correct – using logic and reasoning!

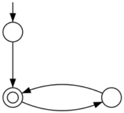
Module 4: Application to Model checking



System $\xrightarrow{\text{satisfy?}}$ specification

System Model $\xrightarrow{\text{model-checking}}$ Spec Model

$\models ?$



ϕ

Overlaps and links to other courses

- Unsurprisingly, there are overlaps with several courses

Undergrad courses

- Discrete structures (CS207) - reasoning ideas same, same but different!

Overlaps and links to other courses

- Unsurprisingly, there are overlaps with several courses

Undergrad courses

- Discrete structures (CS207) - reasoning ideas same, same but different!
- Logic (CS 228)- first part may be similar, but our focus will cover less but deeper with formal methods view. Not a prereq!

Overlaps and links to other courses

- Unsurprisingly, there are overlaps with several courses

Undergrad courses

- Discrete structures (CS207) - reasoning ideas same, same but different!
- Logic (CS 228)- first part may be similar, but our focus will cover less but deeper with formal methods view. Not a prereq!
- Automata theory (CS 310): last part of course will use basic automata

Overlaps and links to other courses

- Unsurprisingly, there are overlaps with several courses

Undergrad courses

- Discrete structures (CS207) - reasoning ideas same, same but different!
- Logic (CS 228)- first part may be similar, but our focus will cover less but deeper with formal methods view. Not a prereq!
- Automata theory (CS 310): last part of course will use basic automata

Other linked courses

- CS 433: Automated reasoning: advanced course more practical
- CS 713: Specialized course on Automata & logic connections
- CS 738/739: Model checking, cyber-physical systems
- CS 433: Artificial intelligence (before the NN revolution!?)
- CS 620/xxx: Formal methods and machine learning
- CS 766: Verification of concurrent programs

Links to other domains

Programs and Systems are everywhere and everyone wants to make sure they work!

Links to other domains

Programs and Systems are everywhere and everyone wants to make sure they work!

- Machine learning and AI
  - Techniques to verify and explain, check robustness of DNNs.
  - Formal reasoning and logic started AI.
  - Probabilistic logics
  - Prob systems: Markov chains, Markov decision processes, POMDPs

Links to other domains

Programs and Systems are everywhere and everyone wants to make sure they work!

- Machine learning and AI
  - Techniques to verify and explain, check robustness of DNNs.
  - Formal reasoning and logic started AI.
  - Probabilistic logics
  - Prob systems: Markov chains, Markov decision processes, POMDPs
- Programming languages
- Blockchains and other concurrent, distributed algorithms: verifying smart contracts.

Links to other domains

Programs and Systems are everywhere and everyone wants to make sure they work!

- Machine learning and AI
  - Techniques to verify and explain, check robustness of DNNs.
  - Formal reasoning and logic started AI.
  - Probabilistic logics
  - Prob systems: Markov chains, Markov decision processes, POMDPs
- Programming languages
- Blockchains and other concurrent, distributed algorithms: verifying smart contracts.
- Data bases Specifying properties and using solvers!
- Cryptography security protocols and formalizing attacks!

Links to other domains

Programs and Systems are everywhere and everyone wants to make sure they work!

- Machine learning and AI
  - Techniques to verify and explain, check robustness of DNNs.
  - Formal reasoning and logic started AI.
  - Probabilistic logics
  - Prob systems: Markov chains, Markov decision processes, POMDPs
- Programming languages
- Blockchains and other concurrent, distributed algorithms: verifying smart contracts.
- Data bases Specifying properties and using solvers!
- Cryptography security protocols and formalizing attacks!
- Cyber-physical, real-time systems, etc.

In conclusion

CS771: Foundations of Verification and Automated Reasoning

- Mathematical Reasoning about programs and systems
- Formalizing using Logic
- Building efficient algorithms when possible
- Showing undecidability when impossible
- A gateway to the fascinating area of formal methods in CS!

In conclusion

CS771: Foundations of Verification and Automated Reasoning

- Mathematical Reasoning about programs and systems
- Formalizing using Logic
- Building efficient algorithms when possible
- Showing undecidability when impossible
- A gateway to the fascinating area of formal methods in CS!

- Emphasis on home work, reading assigments, presentations
- Honor code will be rigorously imposed
- Visit the course webpage for all details!
  `https://www.cse.iitb.ac.in/~akshayss/courses/cs771`
  - Next class coming Thursday, at 3.30pm on teams.
  - Attend first few classes... to get a taste!
  - If you can't make it due to slot clash, but are interested, mail me!