Formal Verification

Targets (if we use a set of logic formulas to express the properties of a system)
- Verify formal specification self-consistency: well-formedness+satisfiability
- Compare different formal specifications: verify that a system specification satisfies some properties (requirements)+verify relation between 2 specifications (requirements and design)

Challenges
- Undecidable questions: no algorithm can answer the question ALWAYS correctly in finite time and using finite memory, but it can answer the question correctly in SOME INSTANCES
  - Give an answer anyway (not useful)
  - Semi-decision procedures (yes,no,I don't know)
- Decidable but too complex (unfeasible) questions
  - Abstractions (disregard aspects) to reduce complexity
  - Approximate/non-exhaustive modelling/analysis (change behaviour/disregard aspects)
  - Correctness by construction (automatic code generation)

Verification of properties Specified in logic language consisting of Model (Intepretation)+Deductive System

Approaches

Model Checking: verify validity of f given M
- **Inputs**
  - Model (interpretation=state transition model for example CFG) M
  - Property (formula) f
- **Output**
  - True (f is valid: security property satisfied)
  - False (f is not valid: security property not satisfied)+counterexample (attack)
  - Keep running forever/Tell that resources exhausted so nothing can be said
- **Example applications**
  - Propositional Logic
  - Temporal Logic
- **Techniques for model checking dynamic systems**
  - **State exploration**
    - **Output**
      - Violation of the property found (counterexample) -> property is false
      - Violation of the property not found
        - If states/runs explored exhaustively (finite state models) -> property is true
        - If states/runs not explored exhaustively -> property is probably true (since we explored a portion of the complete space!) but we cannot be certain about it
    - Explicit
    - Symbolic
    - **Limits**
      - States/transitions must be finite (need abstractions to reduce infinite state system to finite)
      - Complexity grows exponentially for concurrent systems (state explosion)
  - Reachability analysis (simpler state exploration) -> generate all states and verify that P holds in every one -> can be more efficient since as soon as P is false you can end the exploration

Theorem Proving
- **Inputs**
  - Theory T (axioms+inference rules: requires generation of theory from interpretation)
  - Property (formula) f
  - Human assistance
- **Output**
  - Yes (T is a theorem: it is valid for all the interpretations) + proof of validity
  - We cannot know if T is a theorem or not
- **Example applications**
  - Temporal Logic (verify dynamic system properties). T is composed of specific axioms and rules generated and added to a state transition system + the axioms and rules of the Temporal Logic
- **Augmented theory properties**
  - Soundness -> each theorem of the original theories are still theorems in the interpretation K of the augmented theory
  - Completeness -> each theorem that the interpretation K of the augmented theory creates, are theorems of the original theories
  - Soundness without completeness: f is a theorem for my interpretation K but may not hold for the original theories
  - Soundness+completeness: if f is a theorem for my interpretation K, then it is also a theorem for the original theories
- **Techniques**
  - Interactive Theorem Provers -> Proof Assistants
  - Automated Theorem Provers
  - For systems using a propositional logic and Horn clauses, Logic programming can be used to solve the theorem automatically and in linear time