Verification by Model Checking

- safety-critical systems
- commercially critical system

Formal verification of correctness of software systems

Formal verification techniques can be thought of as comprising three parts:

- a *framework for modelling systems*, typically a description language of some sort;
- a *specification language* for describing the properties to be verified;
- a *verification method* to establish whether the description of a system satisfies the specification.

Approaches to verification can be classified according to the following criteria:

Proof-based vs. model-based. In a proof-based approach, the system description is a set of formulas Γ (in a suitable logic) and the specification is another formula ϕ. The verification method consists of trying to find a proof that $\Gamma \vdash \phi$. This typically requires guidance and expertise from the user.

In a model-based approach, the system is represented by a model $\mathcal{M}$ for an appropriate logic. The specification is again represented by a formula ϕ and the verification method consists of computing whether a model $\mathcal{M}$ satisfies ϕ (written $\mathcal{M} \models \phi$). This computation is usually automatic for finite models.

In Chapters 1 and 2, we could see that logical proof systems are often sound and complete, meaning that $\Gamma \vdash \phi$ (provability) holds if, and only if, $\Gamma \models \phi$ (semantic entailment) holds, where the latter is defined as follows: for all models $\mathcal{M}$, if for all $\psi \in \Gamma$ we have $\mathcal{M} \models \psi$, then $\mathcal{M} \models \phi$. Thus, we see that the model-based approach is potentially simpler than the proof-based approach, for it is based on a single model $\mathcal{M}$ rather than a possibly infinite class of them.

Degree of automation. Approaches differ on how automatic the method is; the extremes are fully automatic and fully manual. Many of the computer-assisted techniques are somewhere in the middle.

Full- vs. property-verification. The specification may describe a single property of the system, or it may describe its full behaviour. The latter is typically expensive to verify.

Intended domain of application, which may be hardware or software; sequential or concurrent; reactive or terminating; etc. A reactive system is one which reacts to its environment and is not meant to terminate (e.g., operating systems, embedded systems and computer hardware).

Pre- vs. post-development. Verification is of greater advantage if introduced early in the course of system development, because errors caught earlier in the production cycle are less costly to rectify. (It is alleged that Intel lost millions of dollars by releasing their Pentium chip with the FDIV error.)