**Formal Methods**

* This chapter introduces formal specification languages and mathematically based automated formal methods for proving properties of specifications and programs .
* It examines past and present formal specification and proof technologies used in assurance .

**Formal Verification Techniques**

* Formal verification techniques rely on descriptions of properties/requirements, descriptions of systems, and verification techniques to show system descriptions meet requirements .
* Inductive verification techniques are typically more general and involve generating formulas to show system specifications meet property requirements, often using a theorem prover .
* Model checking techniques establish how well a system specification meets a set of properties using state transition systems and temporal logic, often used after development but before product release .

**Formal Specification**

* A formal specification is a specification written in a formal language with restricted syntax and well-defined semantics based on established mathematical concepts .
* Formal specifications use languages with precise semantics to avoid ambiguity and allow for proofs of properties, supporting precise descriptions of system behavior without implementation details .
* SPECIAL is a first-order logic-based language developed at SRI International as a stand-alone specification language that is well suited for writing functional specifications .

**Early Formal Verification Techniques**

* Early formal methods attempted to mechanize the entire development process, providing lessons for further research .
* The Enhanced Hierarchical Development Methodology (EHDM) focused on proofs of design, aiming to mechanize and formalize the entire development process through successive refinement of specifications .
* The Gypsy Verification Environment (GVE) focused on implementation proofs, attempting to prove a correspondence between specifications and their implementation .

**Current Verification Systems**

* The Prototype Verification System (PVS) provides mechanically checked specifications and readable proofs and is a system for writing specifications and constructing proofs .
* The Symbolic Model Verifier (SMV) is based on Control Tree Logic (CTL) and adds eight temporal connectives to those of the predicate calculus .
* The Naval Research Laboratory (NRL) Protocol Analyzer (NPA) is a special-purpose verification system used to verify cryptographic protocols .

**Functional Programming Languages**

* Functional programming is a declarative style of developing computer programs using mathematical expressions that are evaluated, instead of using statements that can change the state .
* Functional programming languages are well-defined and well-typed lending to analysis that is comparatively simple compared to nonfunctional programming languages .
* OCaml, Haskell, and Rust are three well known functional programming languages .

**Formally Verified Products**

* Most formally verified products are security kernels and recent advances in scalability of formal verification methods made formally verifying a product is becoming more feasible .
* The seL4 microkernel is an open source operating system kernel that has been formally verified against its own specification, including the ability to enforce security properties .
* The Security-Oriented Analysis of Application Programs (SOAAP) analysis tool is based on a number of annotations related to compartmentalization of execution .

**Summary**

* Formal verification begins with a specification stated in an appropriate language and has a precise syntax and well-defined semantics based on mathematical principles .
* The system design and the properties to be verified are described in formal specifications, and the specification of the design is proven to meet the specification of the properties .
* The proof may use general, inductive techniques or be tied to a specific model, in which case model checking techniques are appropriate .

**Research Issues**

* One area of research is defining and selecting security properties amenable to formal analysis that applies to realistic situations with precision .
* Developing architectures that lend themselves to formal verification is a deep area .
* Ongoing work includes verification of cryptographic protocols and verification of code on active networks .

**Further Reading**

* McLean [1301] provides a terse but enlightening review of 20 years of formal methods .
* Snow [1775, 1776] discusses the future of assurance, including the role of formal methods .
* Wing [2010] discusses the relationship of formal methods and security .