

**NAME: HUMNA HASHMI**

**ID: BSE-21S-028**

**SEMESTER: 5TH**

**COURSE NAME: FORMAL METHODS**

**COURSE CODE:**

**SECTION: B**

**TEACHER NAME: MISS SAIMA SIPPY**

**DATE: 29-03-2023**

Q1. Write about Tool Support for Specification Language. Give examples as well.

In order to create, verify, and analyse formal specifications, a variety of tools are available for specification languages. These tools are made to make the specification process more effective while ensuring the precision and consistency of requirements. Specification language tools include some of the following:

1. For drafting and updating formal specifications, specification editors offer a user-friendly interface. They might have tools that make it simpler for users to write accurate specifications, like syntax highlighting, autocomplete, and code snippets.

Examples are TLA+ Toolbox, Z Notation Tool, Promela Editor

1. Checkers: In order to find errors and inconsistencies, specification checkers study formal specifications. They may perform type checking, syntax checking, or other sorts of validation to ensure the specification is well-formed.

Examples are include Alloy Analyzer, FDR4, Z Lint, and SPARK Examiner.

1. Generators: Specification generators create new specifications based on user input or pre-existing templates. They can be used to generate code, test cases, or other kinds of artefacts based on a formal specification.

Examples are JML, QuickCheck, and TestEra

1. Simulators: By simulating the behaviour of the system under examination, specification simulators allow users to test their specifications. They might provide a graphical interface for viewing the simulation and interacting with the system.

SPIN, UPPAAL, and Simulink are a few examples.

1. Model-checkers: Model-checkers of formal specifications check the accuracy of the specifications by carefully reviewing all possible system states. They could be used to assess liveliness, safety, or other requirements.

Examples include NuSMV, PRISM, SpinJa, and LTSA.

Q2: What are types of Formal Specifications? Describe each in detail.

Formal specifications are used to represent the requirements and properties of a system in a formal language. They are mathematically precise descriptions of a computer programme, piece of hardware, or business function. There are many different types of formal specifications, each with unique benefits and drawbacks. Below are some of the most often used formal specification types:

1. Algebraic specifications are a kind of formal specification in which the behavior of a system is described using mathematical equations. It has a foundation in mathematical theories of algebra and logic and is well suited to describing systems with intricate data structures and operations. The behaviour of programming languages and libraries is frequently specified via algebraic requirements in software engineering.
2. Temporal logic requirements are used to describe how a system will function over time. By utilising formal logic to explain the connections between events and states of a system, they can specify systems with temporal restrictions, such as real-time systems, concurrent systems, and communication protocols.
3. Model-Based Specifications: Model-based specifications provide a collection of mathematical models that may be used to simulate and evaluate the behaviour of a system. Because they are well adapted to modelling large systems with several interdependent components, they are commonly applied to aerospace, defence, and other safety-critical systems.
4. 4. Process algebra is a type of formal specification that describes how concurrent and distributed systems behave. Given that it is built on the idea of process calculi, it is appropriate for modelling the behaviour of communication protocols and distributed systems.
5. Z Notation specifications use mathematical notation to describe a system's behaviour. Z Notation is a formal specification language. It is commonly used in software engineering to describe systems with complex data structures and processes and can be used to characterize the behaviour of real-time systems and databases.
6. 6. State-Based Specifications: State-based specifications describe a system's behaviour in terms of the states and transitions it makes. They are effective for representing systems with few states, such as control systems and communication protocols.