1	**MODULE CODE:**	**CIA 2326**
2	**MODULE TITLE:**	**Formal Aspects of Computer Science**
3	**SCHOOL(S) INVOLVED IN DELIVERY:**	Computing and Engineering
4	**NAME OF COURSE(S)):**	BSc(Hons) Software Development BSc(Hons) Computing Science BA(Hons) Business Computing with Software Development MEng Software Engineering
5	**MODULE LEADER:**	TL McCluskey
6	**LOCATION:**	Queensgate Campus
7	**MODULE TYPE:**	Core
8	**CREDIT RATING:**	20 credits
9	**LEVEL:**	I
10	**LEARNING METHODS:**	**Lectures:** 24 hrs **Tutorials/Practicals:** 24 hrs **Directed Unsupervised Learning:** 152 hrs
11	**PRE_REQUISITE(S):**	None
12	**RECOMMENDED PRIOR STUDY:**	None
13	**CO-REQUISITE(S):**	None
14	**PROFESSIONAL BODY REQUIREMENTS:**	N/A
15	**GRADED OR NON-GRADED:**	Graded
16	**BARRED COMBINATIONS:**	None

17 SYNOPSIS:

This module will introduce students to some of the concepts, tools and techniques that provide a foundational basis for past and future developments in computing. It will introduce theory and reasoning in propositional and predicate logic, algebra, types, formal languages and formal systems. Illustration of the application of the formalisms will be made within areas such as compilers, data and knowledge representation, web meta-languages, formal specification, rigorous software development and artificial intelligence. To assist in the theoretical presentation, a high level language or tool will be used (for example, Prolog, Haskell, or the B-Tool).

18 OUTLINE SYLLABUS

Review of propositional logic, introduction to first-order predicate logic.
Introduction to proof and semantics: inference systems for propositional and first-order predicate logic; automated reasoning; logic interpretation and properties – ambiguity, inconsistency and incompleteness; applications in database, artificial intelligence and the semantic web.
Formal Languages: grammars, lexical analysis and parsing; properties of grammars and parsing algorithms; applications in compilers and meta-languages.
Introduction to algebraic specification: simple axiomatic theories, homogenous and heterogeneous algebras, abstract data types + examples; applications in formal specification and formal methods (algebraic and model-based specifications).

The theoretical notions above will be sustained with the use of a declarative language (eg Prolog, Haskell, Gofer) and / or a proof tool, which will be taught and used in practical sessions.

19 LEARNING OUTCOMES:

19.1 Knowledge and understanding outcomes

On completion of this module, the student will be able to:
19.1.1 Demonstrate an understanding of first-order logic and algebra, and how they relate to areas within computer science and software engineering.
19.1.2 Demonstrate an understanding of the principles of formal languages and formal systems.
19.1.3 Understand the need and benefits of rigour and formality within computing, and the role of formalisms (specifically algebra and logic).

19.2 Ability outcomes

On completion of this module, the student will be able to:
19.2.1 Translate statements into the propositional calculus or predicate calculus.
19.2.2 Undertake proofs framed within the propositional or predicate calculus, both by hand and using a proof tool.
19.2.3 Use a declarative language to specify and animate software specifications.
19.2.4 Manipulate terms and expressions within the algebraic specification of abstract data types.
19.2.5 Be able to apply simple parsing techniques.

20 ASSESSMENT STRATEGY:

20.2 Summative assessment

First component of the assessment: one assignment (weighted 40%) assessing outcomes 19.1.1 and 19.2.1-19.2.3. This will involve a practical investigation into FOL or algebra, with the students having to program in the declarative language used for teaching.

Second component of the assessment: a 2 hour examination (weighted 60%) assessing all outcomes.

21 LEARNING STRATEGY:

Lectures will be used to deliver both the essential mathematics and the theoretical and technical material relating to the specification methods and their use.

Throughout, tutorials and practicals will place emphasis upon computing-oriented examples. For the first half of the module, a declarative language such as Prolog will be learned and used to underpin the theoretical material on logic, data and algebra. For the second half, the lecture material will be supported by short tutorial exercises with some use of practical tools where appropriate.

22 INDICATIVE REFERENCES:

Fenton, N and Hill, G A	Systems Construction and Analysis – A Mathematical Framework. McGraw-Hill, 1992.
Scheider, S	The B-Method. Palgrave, 2001
Turner, J and McCluskey, T L	The construction of Formal Specifications: An Introduction to the Model-Based and Algebraic Approaches. McGraw-Hill, 1995.