

THE PHILOSOPHY OF COMPUTER SCIENCE

The Philosophy of Computer Science (**PCS**) is concerned with philosophical issues that arise from reflection upon the nature and practice of the academic discipline of Computer Science. Below we indicate a few of the central questions.

- I. How is a programming language determined? What role does a semantic definition play? Does it have to be a formal abstract specification?
- II. What sense is to be made of the notion that a programming language has an ontology? What is the role of such an ontology? How is it linked to the type structure of the language?
- III. What does it mean to say that a program is correct? What role do specifications play in correctness? How does the nature and use of theorem checkers and verifiers inform the debate? What are formal methods? What is the difference between a formal method and informal one?
- IV. Is there a distinctive form of reasoning that might be called *computational reasoning*? How, if at all, does it differ from mathematical reasoning?
- V. What kinds of things are digital objects?
- VI. What is abstraction in computer science? How is it related to abstraction in mathematics?
- VII. Does the Church-Turing thesis apply to physical machines? Does it make sense to say that the universe computes?

Among others, papers that address issues that concern the methodology of the discipline, the status and nature of its claims to knowledge, the nature of its artefacts, the nature and form of computational reasoning and the philosophical basis of computational modelling are welcome.

Track chair: Raymond Turner

<http://cswww.essex.ac.uk/staff/turnr/>

Information about PCS: <http://plato.stanford.edu/entries/computer-science/>