--- Project Summary ---

The continuing evolution of computing hardware has led to enormously complex architectures with execution models that integrate advanced memory technologies and hardware models. However, these advanced architectures break assumptions that programmers have relied on, causing new safety bugs and security vulnerabilities. We target multi-processor systems and concurrent architectures, thus support the development of concurrent programs.

It is well known that reasoning about concurrency is notoriously difficult -- incorrect synchronisation can lead to many dangerous safety and security vulnerabilities, ranging from "out-of-bounds writes" and "use-after-free" errors to "improper synchronisation and race conditions". Architecture-based attacks (e.g., Spectre) show the urgency of addressing these important problems today.

Even when low-level programs are well synchronised, the design of the underlying concurrent algorithms can themselves be vulnerable. Well-understood safety conditions such as linearizability do not guarantee security, and current approaches to addressing this issue lead to overly synchronised implementations (degrading performance). This introduces a tension between the goals of the hardware designers (who aim to maximise performance), and end users (who require trustworthy software). In the middle are developers, who are tasked with producing software that balances this tension. COVERT provides mechanisms for provably correct reusable abstractions that maximise flexibility in program design, allowing fine-tuning of both safety and security guarantees based on the architecture.

Our vision is to provide (a) reusable models, tools and techniques that enable the verification of safety and security properties over a range of advanced architectures, and (b) a verified set of concurrency abstractions that guarantee both safety and security. This effort involves two key strands of work.

- In the first, we bring together different facets of modern systems and provide a rigorous foundation covering a range of architectural features such as out-of-order execution and speculative execution, and memory features such as weak memory and NVM. Additionally, we extend existing reasoning techniques above our newly established foundation by developing threat models that cast the intricate behaviours allowed by advanced architectures through the lens of a malicious attacker.
- Our second strand involves the practical realisation of these techniques via verified litmus tests, verified concurrency libraries and associated verification tools. We will build novel techniques and tools that strengthen traditional correctness criteria (linearizability, opacity etc) to provide architecture-aware correctness of both safety and security. To ensure a high degree of reliability, our theory and case studies will be mechanised within the Isabelle proof assistant.