Bugs in compilers can have severe consequences. Apart from traditional methods like testing, one of the ways of keeping compilers correct that gained traction only in recent years is *translation validation*, a technique ensuring the semantic correctness of optimizations in compilers. *Alive2* is an open-source translation validation framework for LLVM that is currently widely used by LLVM developers. In order to make any static analysis tool usable, the frequency of false alarms must be kept to a minimum. Alive2 was designed to have zero false alarms and has been very successful in this endeavor except in the case of certain loops. Our aim in this thesis is to analyze Alive2's loop algorithms in an attempt to find the cause of these false alarms. This was motivated by personal communication with authors of Alive2 who presented the false alarm issue in loops as one of the more challenging and pressing issues in Alive2. We were successful in pinpointing the cause of false alarms and even providing a fix for the issue. Our solution is now a part of the Alive2 framework. Furthermore, we have identified other potential issues in Alive2 which we discuss in the thesis as well.