Management Summary

Generating Formally Verified Communication Protocol Implementations

Introduction and Background

Evolving Communication Protocols Reliably

During the launch of a mission a critical moment occurs when communication is established for the first time. The details of the communication between ground control and a satellite are governed by the communication protocol. Such a protocol is traditionally specified in human-readable form. Engineering teams define the communication protocol early in the requirements definition phase. As the development progresses, the protocols usually evolve as well. In rare cases protocols might even change after the launch. Whenever the protocol specification changes, the documentation, code in the ground segment and space segment must be changed accordingly. Keeping all parts consistent is challenging, especially when different teams are responsible for different subsystems or instruments.

ESA addressed this need for consistency in the face of constant change with the ASN1SCC project. ASN1SCC generates documentation and code from a single source of truth: A machine-readable specification of the communication protocol. This specification is written in the ASN.1 format. ASN1SCC generates documentation and code from this specification fully automatically, which can be integrated in other documentation and software. The ASN1SCC generator can be deployed in Continuous Integration (CI) systems to generate all required artifacts automatically from the single source of truth. This ensures that all software and documentation use exactly the same protocol.

Prior to this project, ASN1SCC could generate code for the C and Ada programming languages. These programming languages are often used in embedded applications, e.g. in the space segment. Software for the ground segment is typically written in other programming languages, for example Java or Scala. Integrating ASN1SCC-generated code in those applications was not easily possible. This project's objectives were to **extend the ASN1SCC generator with a Scala code generation backend,** and to **apply formal software verification methods to the generated Scala code** to guarantee that it is defect-free**.**

The reliability of the generated communication code is obviously of critical importance, as satellites are solely remote-controlled, and because the network protocol implementation is the first point of contact for attackers. To increase the robustness, people try to use extensive manual or automated testing. This might be useful to find the most obvious bugs, but as the well-known saying **"Software testing proves the existence of bugs, not their absence."** describes, it is impossible to produce robust and defect-free software using tests alone. Even if the generated Scala code behaves as expected

in thousands of tests, this does not mean that it is free of defects. The absence of evidence is not evidence of absence. For this reason, we used the formal verification framework Stainless to provide much stronger guarantees about the generated code. We prove that the generated code behaves according to its specification and that it does not contain bugs for the most common classes.

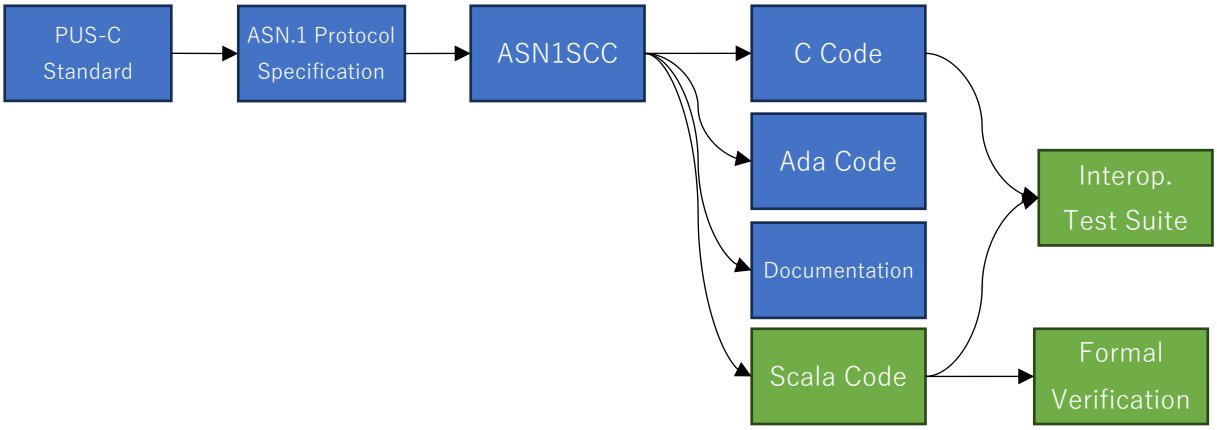

Figure 1: A protocol standard like PUS-C is formally described in the ASN.1 specification language. This allows computer programs like ASN1SCC to work with the protocol standard. Using such a specification, ASN1SCC automatically generates matching code in C or Ada. We extended the ASN1SCC generator with the ability to generate Scala code in this project. Additionally, we formally verified the generated Scala code with Stainless, and created a test suite to automatically test interoperability of the generated C and Scala code. The newly developed parts are depicted as green boxes.

Methodology and Approach

A new Scala Code Generation Backend and the Verification of its Output

We extended the ASN1SCC compiler by adapting the existing C backend. In addition to creating a new backend, the ASN1SCC software architecture had to be changed as well because of the different nature of Scala and C/Ada code. The generated ASN1SCC code relies on fixed library code (Figure 2). We ported this library code from C to Scala by hand.

We developed a new interoperability test suite to ensure that the generated Scala code produces bitwise identical results to the code generated by the existing C backend (Figure 3). The test suite automatically compares Scala- and C-generated packets defined by the PUS-C standard. This uncovered problems in both the newly developed Scala backend and the existing C backend, which were both fixed.

We formally verified the hand-written library code using the Stainless framework. Stainless proved that all 5'113 properties of the code are guaranteed to hold: for example, that divisors are not zero, certain arguments are non-negative, and that all loops terminate. We also proved that the encoding and decoding are inverses of each other, i.e. no information is lost through the encoding or decoding process.

We changed the automatically generated Scala code, for it to be formally verifiable by Stainless. We extended Stainless to support the generated code constructs. We tested this capability by verifying the generated code for all packets defined in the PUS-C standard. It took Stainless several hours to verify the resulting 327'321 properties on three dual-socket servers.

The performance of the generated Scala code and the generated C code are surprisingly similar. A detailed analysis shows this is possible, because the generated Scala code does not perform any heap allocation, and because the Scala runtime (JVM) employs more inlining optimizations.

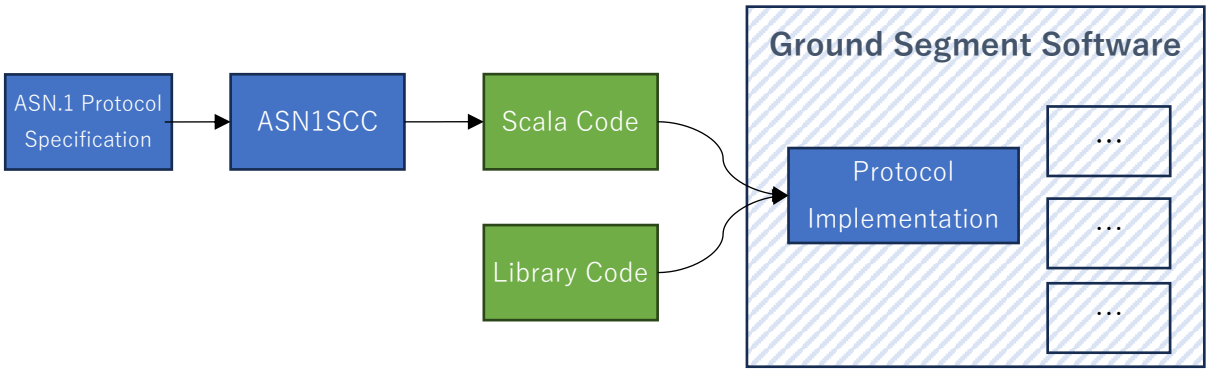

Figure 2: The automatically generated Scala Code is combined with hand-written Library Code, and together forms a protocol implementation, which is used in the ground segment software to create and interpret communication packets of a satellite.

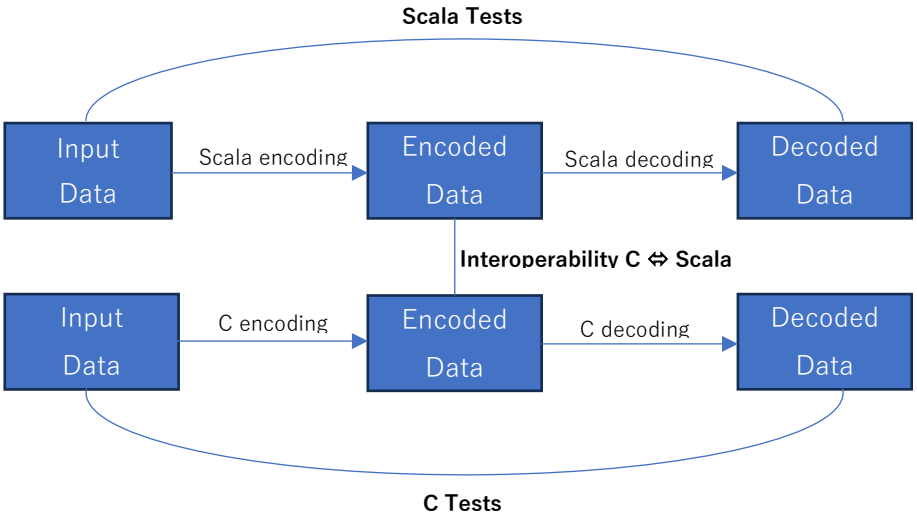

Figure 3: The newly developed interoperability test suite ensures that the packets generated by the C and Scala backends are bit-wise identical. The test suite generated packets for all packets defined in the PUS-C standard, with both the C and the Scala backend. Each packet is compared with the same packet generated by the other backend. In addition, the tests verify that the decoded data is identical to the input data.

The Scala Backend, Bugfixes and a Publication

The ability to produce Scala code was introduced in ASN1SCC release 4.5.1.2 on March 8th 2024. This version includes bugfixes for all discovered bugs in the C backend. Since its initial release, the Scala backend has already found use by early adopters.

The generated Scala code underwent manual and automated testing, and was formally verified. It is correct, reliable and performant.

Mario Bucev, Simon Felix, Samuel Chassot, Viktor Kunčak, and Filip Schramka submitted the publication “Automatic Generation of Formally Verifiable ASN.1 Serializers and Deserializers” to the 26th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2025), which will take place January 2025.