Software has become an integral and integrated part of everyday life, accompanied by a near constant stream of updates to various devices all around us. Unfortunately, software updates are a frequent source of issues [**zhangUnderstandingDetectingSoftware2021**, **Gray1986WhyDC**], often making it into production environments before the necessary edge cases are triggered. We believe that techniques from formal verification can be applied here to ensure update compatibility, which is ultimately what my project aims to do. Unlike some previous work [**ajmaniModularSoftwareUpgrades**
reitblattAbstractionsNetworkUpdate2012], I aim to prove compatibility of software updates without imposing restrictions about how the update itself is performed.

The most nebulous part of this goal is how to define "compatible", which I break down into two major categories; data compatibility and operational compatibility. Data compatibility aims to show that that the new version can correctly interpret persistent state left behind by the pervious version while operational compatibility reasons about behavior of invoking the same function or feature in both versions of the software. This project is currently working on data compatibility.

There are numerous data description languages, often paired with a particular encoding format. Notable examples of this include ASN1 [**ASN1EncodingRules2021**], protobuf [**LanguageGuideProto**], CBOR [**birkholzConciseDataDefinition2019**, **bormannConciseBinaryObject2020**] and even JSON with JSONschema [**wrightJSONSchemaMedia2022**, **brayJavaScriptObjectNotation2017**]. All of these formats have seen prior formalization in proof-assistants or proof oriented programming languages [**habibFindingDataCompatibility2021**, **ramananandroSecureParsingSerializing2025**, **yeVerifiedProtocolBuffer2019**, **niASN1ProvablyCorrect2023**], although none of these works consider the common real-world case of an update to the schema used to pass messages between nodes of a distributed system or store data while the software isn't running. Some tools already exist which can report on if two versions of a schema file are compatible, although these tools operate using ad-hoc notions of compatibility without formalizing a specification or verifying the implementation.

Using F*, I plan to verify a compatibility checker for one or more of these libraries before expanding the project to consider a wider range of data formats. It should also be possible to include some analysis of the source code operating on the state to prove more invasive updates, or even the proofs of verified pieces of software, although that is a goal for a longer timescale.