Formalizing Actuarial Mathematics in Proof Assistants

Yosuke ITO

Actuarial Mathematics

Actuarial mathematics is a field of applied mathematics mainly used to determine the prices of insurance products and to evaluate the liabilities of companies associated with insurance contracts. It is closely related to calculus, probability theory, and financial mathematics. In life insurance mathematics, evaluating the present value of insurance products is particularly important. For example, the premium of a life insurance product is determined by solving the equation representing the *equivalence principle*, which requires that the expected present value of the premiums be equal to the expected present value of the benefits. Since there are many types of life insurance products, we use the internationally standard actuarial notation[1] to compute their present values efficiently.

Formalization

My motivation originates from pure curiosity: is it really possible to develop actuarial mathematics rigorously? I often see some logically imperfect arguments in actuarial mathematics, as well as approximations introduced to mitigate the burden of calculating premiums and reserves. In that sense, formalizing actuarial mathematics without losing its general usefulness seems more challenging than formalizing pure mathematics.

In 2021, I experimentally formalized the basic part of life insurance mathematics in Rocq, which was the first study to formalize actuarial mathematics in proof assistants. The Rocq code and related documents are available in my GitHub repository.[2] However, this package requires the survival probability of the insured to be C^1-smooth.[3] This is a serious limitation because it excludes even a linearly decreasing function, which is not differentiable at the ultimate age.

[1] `https://www.casact.org/sites/default/files/database/proceed_proceed49_49123.pdf`

[2] `https://github.com/Yosuke-Ito-345/Actuary`

[3] At that time, the only available option was to use the Coquelicot library, in which the basic parts of real analysis had been formalized in Rocq. The requirement of C^1-smoothness comes from this library.

Recently, I have been working on formalizing actuarial mathematics mainly in Isabelle, where a wide range of mathematical analysis (including Lebesgue integration) and probability theory has been formalized. Although this work is still in progress, the current formalization is available in the Archive of Formal Proofs [1]. The existing entry "Actuarial Mathematics" roughly covers the theory of interest and survival models. In ongoing development, I am formalizing the equivalence principle mentioned above. In addition to Isabelle, I am partially involved in the development of MathComp-Analysis, an analysis library for Rocq.[4] This library is actively developed and already includes Lebesgue integration and the basic parts of probability theory.

These formalized libraries have the potential for future industrial use including error detections of actuarial documents and programming code. We often use actuarial modeling software to evaluate the risks of insurance products, but fully verifying it would involve great difficulties, such as formally specifying the behavior of such software. The formalized actuarial mathematics could help to reduce the burden of creating the specification document and proving the correctness of its source code.

Usage of AI

Recently we have seen rapid development in AI, which could transform how we research and formalize mathematics. As an actuary and a user of proof assistants, I hope AI could help us in some of the following ways.

Auto-formalization As in many other formalization projects, it takes considerable effort and time to formalize actuarial mathematics. If AI can assist in writing formal proofs, the progress of this research could be significantly accelerated. I have tried Aristotle API[5] and feel the tremendous potential to auto-formalize our pen-and-paper proofs.

Formal Semantic Translation Implicit parameters are often omitted in actuarial notation, as is common in many branches of mathematics. For example, the symbol v, representing the discount factor $1/(1+i)$, omits the interest rate i it depends on. However, these parameters must eventually be specified before we can verify actuarial documents with proof assistants. AI may be well suited to such translations from real-world texts into a formal language.

References

[1] Yosuke Ito. Actuarial mathematics. *Archive of Formal Proofs*, 2024. `https://isa-afp.org/entries/Actuarial_Mathematics.html`, Formal proof development.

[4] `https://github.com/math-comp/analysis`

[5] `https://aristotle.harmonic.fun/`