Software Development for the Working Actuary

David Raymond Christiansen
drc@itu.dk

The Problem

- *Solvency II* imposes new and complicated computational demands on insurers
- Software systems with forms-based interaction are not sufficiently flexible
- Current techniques do not support the full generality of actuarial models

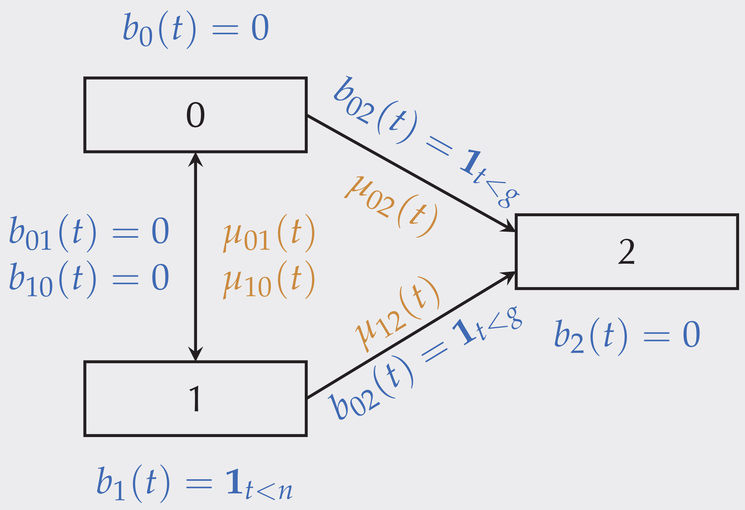
Who are actuaries?

- Trained mathematicians
- Some programming background (usually in R)

Current Practice

1. Actuarial Models

Actuaries use continuous-time Markov models. Payment streams $b_j(t)$ are attached to states, single payments $b_{jk}(t)$ are attached to transitions. Transition intensities $\mu_{jk}(t)$ represent chance of state change.



This product pays $1 per year in case of disability until some expiration n and a $1 lump-sum benefit upon death prior to g.

2. Calculating Reserves

Using payment and risk information from the model, derive Thiele's differential equations:

$$\frac{d}{dt}V_j(t) = \left(r(t) + \sum_{k;k\neq j}\mu_{jk}(t)\right)V_j(t) - \sum_{k;k\neq j}\mu_{jk}(t)V_k(t) - b_j(t) - \sum_{k;k\neq j}b_{jk}(t)\mu_{jk}(t)$$

(In practice, most companies use a standard reference work of products)

3. Programmers Write Code

Professional software developers convert analytic solutions into software. Consequences:

- Long turnaround time
- Repeated effort for similar products
- Models must be simplified

Work supported by the Danish Advanced Technology Foundation (*Højteknologifonden*) (017-2010-3). AML was designed in collaboration with Peter Sestoft from IT University of Copenhagen and Henning Niss, Niels Kokholm, Klaus Grue, and Kristján Sigtryggsson from Edlund A/S.

Our Vision: the Actulus Modeling Language

The **Actulus Modeling Language** (AML) seeks to empower actuaries to write their own analyses and run them efficiently. The language supports actuarial concepts directly.

Actuaries Write the Code

Actuaries design model in AML → **AML type system finds errors** → **Numerical ODE solver computes results**

Properties of AML

Functional taking advantage of actuaries' strong mathematical background

Domain-specific supporting actuarial concepts directly

Total enabling analyses and preventing bugs

Dependent types provide very strong correctness guarantees

Products and Models in AML

AML separates state models, risk models, and products. This allows modular construction of actuarial models.

```
statemodel Disability(p : Person) where
  states = active
           disabled
           dead
  transitions = active -> disabled
                active -> dead
                disabled -> active
                disabled -> dead
```

```
riskmodel DisabilityRisk(p : Person)
          : Disability(p) where
  intensities =
    active   -> dead by gm(p)
    disabled -> dead by gm(p)
    active   -> disabled by ...
    disabled -> active by ...
```

```
product Policy(p : Person, n : TimePoint, g : TimePoint) : Disability(p) where
  obligations = at t pay $1 per year provided(disabled and t < n)
                at t pay $1 when(not dead -> dead) provided(t < g)
```

Static Types in AML

Static types catch errors early:

- Wrong state model
- Missing transition intensities
- Adding currency to time

Dependent Types

Dependent types allow library writers to give very strong guarantees:

- A product for Cathy is used with a statistical model for Cathy, not one for Joe
- Tracking units: dollars per year vs. dollars

Goals

- Fast turnaround — no professional programmers needed
- Use one technical artifact for different calculations and for administration, across technical platforms
- Catch errors early
- Readable notation

Ongoing Work

- Specification and formalization of the type system
- Implement AML and put it in front of real users.

IT University of Copenhagen