Formalizing a simple loan agreement in mcrl2

Joe Watt

September 13, 2022

This is a work in progress.

Contents

1 Intro and shortcomings of DFA formalization

TODO:
Write a better intro and provide more background on [Flood and Goodenough, 2021].

In [Flood and Goodenough, 2021], the authors claim that many financial contracts are inherently computational in nature. They argue that the computational structure of many such contracts can be formalized via deterministic finite automata (DFAs), with states representing various situations. Transitions between these states then correspond to events triggering a change in these situations. This is demonstrated using a simple loan agreement [Flood and Goodenough, 2021, Table 1], which they formalize directly as a DFA.

While this approach is a nice proof of concept, there are various shortcomings with such a formalism, one of them being that encoding such a DFA manually is a laborious process. A simplified visual representation of the automaton corresponding to this simple contract [Flood and Goodenough, 2021, Fig. 1] already contains more than 20 states and 40 transitions. This would explode in complexity if the contract were more complex, like say if it contained more repayment stages instead of just two.

Upon closer inspection of the states in the DFA, one may notice that this high complexity is a result of a verbose duplication of states along the main “happy path". Note here that when we say "duplication", we do not mean that there are states which play exactly the same role from the point of view of bisimilarity or language acceptance as in the sense of the Myhill-Nerode technique of DFA minimization. What we mean instead is that there are similarly named states whose subgraphs have similar structure. These include the 2 main repayment stages along this path, given by the "Payment... accruing" states. Each of the subgraphs rooted at these nodes has a similar structure, with borrower default events giving rise to transitions to the unhappy paths. These in turn also have similar structure, with similar "Payment... accelerating" and "Crisis..." states.

The DFA formalization in [Flood and Goodenough, 2021] also has some subtle concurrency bugs in that the DFA does not allow for some possible interleaving of events. To see this, consider the following scenario. On May 31, 2015, the day before payment 1 is due, the borrower defaults on his representations and warranties. On the next day, when payment 1 becomes due, the borrower diligently repays that payment. One day later, on June 2, 2015, the lender notifies the borrower of his earlier default, which does not get cured after another 2 days. Now all outstanding payments become accelerated, and the borrower pays off the remaining amount of \$525 in time, causing the contract to terminate. This scenario, when formalized as a word over the event alphabet, is unfortunately not accepted by the automaton.

Perhaps for the sake of simplicity or otherwise, there is an implicit assumption being made that once an event of default occurs, the borrower will immediately be notified by the lender, so that no other events like payments can occur in between. A more realistic encoding of this simplified contract should allow for such interleaving of events, as the real world is inherently concurrent in nature. However, this would further increase the complexity of the DFA and thus the tedium involved in its manual encoding. This suggests that for practical purposes, one should not use a DFA directly, but rather, a formalism that can conveniently accommodate the concurrency that is present in the real world. Better still if it could then be compiled automatically into an automaton of some kind and automated analysis performed on it.

In search for such a formalism, we have surveyed various approaches for representing legal contracts, using the simplified contract in [Flood and Goodenough, 2021, Fig 1.] as an example. In particular, one of the formalisms we explored is the `mcrl2` toolset [Groote and Mousavi, 2014, Bunte et al., 2019]. This toolset provides a high-level modeling language based on a *process algebra*, a formalism ...

TODO:
Describe process algebras

Here, we discuss our formalization and show how it generates a DFA that better handles the concurrent interleaving of events. We also demonstrate how we can use the `mcrl2` toolset to perform automated analysis with the contract.

References

[Bunte et al., 2019] Bunte, O., Groote, J. F., Keiren, J. J. A., Laveaux, M., Neele, T., de Vink, E. P., Wesselink, W., Wijs, A., and Willemse, T. A. C. (2019). The mcrl2 toolset for analysing concurrent systems. In Vojnar, T. and Zhang, L., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 21–39, Cham. Springer International Publishing.

[Flood and Goodenough, 2021] Flood, M. D. and Goodenough, O. R. (2021). Contract as automaton: representing a simple financial agreement in computational form. *Artificial Intelligence and Law*.

[Groote and Mousavi, 2014] Groote, J. F. and Mousavi, M. R. (2014). *Modeling and Analysis of Communicating Systems*. The MIT Press.