Formalizing a simple loan agreement in L4 and B

Joe Watt

December 10, 2022

This is a work in progress.
Most of this was originally written based on the work done in mcrl2 but is now being rewritten for L4 and its connection to the modelling done in B.

Contents

1 Intro and shortcomings of DFA formalization

1.1 Intro

TODO
Write a better intro and provide more background on [2].

In [2], 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 [2, Table 1], which they formalize directly as a DFA.

1.2 Shortcomings

While this approach is a nice proof of concept, there are various shortcomings with such a formalism, perhaps the most apparent being that the manual encoding of a contract as a DFA is a laborious process. A simplified, arguably inaccurate, visual representation of the automaton [2, Fig. 1] corresponding to this simple contract already contains more than 20 states and 40 transitions.

One of the main causes of this is the concurrent interleaving of real world events. By this we mean that many real world events can occur in any order and perhaps for the sake of simplicity, this has not been accounted for in the DFA formalization. As an example of such a scenario, consider the following. 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 sequence of events, when viewed as a word over the event alphabet, is unfortunately not accepted by the automaton.

Closer inspection of the DFA suggests that there is an implicit assumption being made that once an event of default occurs, the borrower will be notified by the lender soon after, with no other events like payments occurring in between. More formally, it is assumed that the default and notification events occur *within the same atomic step*. Splitting this up into separate events would increase the number of states in the DFA, especially if we account for concurrency.

We argue that another weakness of the approach is that aside from complexity, it is not natural to think of a contract in terms of a DFA and encode it as such.

TODO
Other weaknesses to mention:
Representation of time and deadlines. Not handled well by the DFA, but is an important notion that exists in L4. These are translated to B.
No explicit notion of deontics, which is important to humans because that's what people are used to reasoning about and normative language shows up everywhere in legal drafting. In L4, we have a notion of deontics, though not quite the same as deontic logic. We have an operational interpretation that is made precise by the translation to B.
The focus here is that a friendly-to-use language should provide appropriate constructs and abstractions corresponding to concepts found in the application domain. For the legal domain, these include the notions of rules, actors, events, deadlines and deontics. Another reason for modelling these explicitly is to be able to achieve a nice correspondence (more technically an isomorphism) with legal texts that are drafted in natural language.

2 The L4 approach

2.1 Intro to L4 as a language

TODO
Talk about how L4 is designed to address these issues.
Firstly it comes with a concurrent model of computation (this is made precise by the translation to B) which allows for arbitrary interleaving of events.
It is also friendly to use, containing ontological concepts from the legal domain that people are used to thinking about and working with.
Discuss the precise roles of these concepts and how the execution of a contract can be viewed abstractly as a process that arises from the interaction between them.
Mention that the precise operational interpretation of rules in L4 is tightly linked to the translation in B and this was heavily inspired by [1, 7].
We view a rule as a wrapper around an event, a deontic (ie permission, obligation or prohibition) and an actor. Our treatment of deontics accounts for the notion of *compensability*, and is based on [4].
Note that the difference is that instead of reasoning about deontics using deontic logic, we provide an *operational* interpretation for them.

2.2 Modelling the loan agreement in L4

3 Modelling contracts in B and how it relates to L4

In search for such a suitable formalism for encoding contracts, we have surveyed various approaches, using the simplified contract in [2, Fig 1.] as an example. One of these which we explored is the B language and the ProB toolset [5, 6].

In this section, we begin by providing some background on B before we discuss our formalization of the simple loan agreement and along the way, we explain how we addressed the aforementioned shortcomings. In particular, we show how we can generate an arguably more accurate DFA that accounts for the concurrent interleaving of real world events. Thereafter, we demonstrate how we can use ProB to help us visualize and reason about the contract.

3.1 Background on B

Techniques originating from the field of formal methods have been devised to automatically analyze the behavior of computer systems. These often rely on first modelling these systems as *labelled transition systems* (LTS), which can be seen as generalizations of nondeterministic finite automata). As with finite automata, LTSes can be seen as directed graphs, with nodes representing states that the system can be in, and labelled edges denoting transitions that change the state of a system. Where they differ from finite automata is that they are allowed to have an infinite (possibly uncountable) number of states and transitions between them. They are also not required to come with a notion of initial and final states.

While DFAs and LTSes in general are formalisms that are well suited for computers to analyze and reason about, they are cumbersome for humans to use to encode systems. This is especially so for systems like the simple loan agreement of [2] which involve concurrency, in that there are many possible ways in which events can be interleaved. Modelling concurrent systems like this directly as a transition system is often impractical as the interleaving of events gives rise to numerous states and transitions. Moreover, as previously discussed, they leave implicit many of the domain specific concepts that people directly reason about.

The toolset we use, ProB, allows us to model and analyze such concurrent systems. As with others like it, it comes equipped with a more sophisticated formalism that allow us to more conveniently specify these transition systems.

3.2 Translating from L4 to B

TODO
Use the loan agreement as an example here.
Include sections showing off the visualization that B provides, as well as simulating and analyzing the contract.

3.3 Visualizing and simulating the contract

3.4 Analyzing via model checking

TODO
Here we should mention that in [3], the author highlights an issue with using temporal logic to reason about deontics like obligations. We sidestep that issue due to the way we define rules as wrappers around events, and deontics, so that the violation of a rule corresponds directly to whether that event occured or not.
For instance, a rule of the form "Party P must do A by D" is an obligation that is violated if P does not perform A by the deadline D. In this way, we can reason about the violation of obligations by using temporal logic to talk about whether an event occurred or not.

4 Related work

5 Limitations and future work

References

[1] Arenas, A. E., Aziz, B., Bicarregui, J., and Wilson, M. D. An event-b approach to data sharing agreements. In *Integrated Formal Methods* (Berlin, Heidelberg, 2010), D. Méry and S. Merz, Eds., Springer Berlin Heidelberg, pp. 28–42.

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

[3] Governatori, G. Thou shalt is not you will. *CoRR abs/1404.1685* (2014).

[4] Hashmi, M., Governatori, G., and Wynn, M. Normative requirements for regulatory compliance: An abstract formal framework. *Information Systems Frontiers 18* (05 2015).

[5] Leuschel, M., and Butler, M. ProB: A model checker for B. In *FME 2003: Formal Methods* (2003), K. Araki, S. Gnesi, and D. Mandrioli, Eds., LNCS 2805, Springer-Verlag, pp. 855–874.

[6] Leuschel, M., and Butler, M. J. ProB: an automated analysis toolset for the B method. *STTT 10*, 2 (2008), 185–203.

[7] Parvizimosaed, A., Roveri, M., Rasti, A., Amyot, D., Logrippo, L., and Mylopoulos, J. Model-checking legal contracts with symboleopc. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems* (New York, NY, USA, 2022), MODELS '22, Association for Computing Machinery, p. 278–288.