System Development with Formal Specification from the Empirical Modeling perspective

Xiaofeng Li
0750478

Abstract

System development, or broadly speaking, software engineering, would introduces risks and bugs as the increase of complexity. Formal specification is researched and developed for several years to constrain the states and transitions. This paper explores the system development with formal specification from the Empirical Modeling perspective, which is a novel approach to computer-based modeling focusing on observables, dependency, and agency. A specific example, train barrier controller system, is developed for reference to illustrate the ideas here.

1 Introduction

The theme of this paper is how formal specification works with the system development. The work of this paper also broadens the scope by questioning how it is useful for different relatives, how it is interpreted from empirical modeling (EM) perspective, what is the difference in the traditional way and EM way, and what can EM do more. A train barrier controller system is developed for reference.

Formal specification is used to constrain the states and transition of the system. This paper would focus on a form of this kind methodology, the B-method. B-method is a formal approach to the specification and development of computer software systems, even a formal verification of such systems. Abstract machine notation (AMN) is introduced to illustrate the specification of a (part of) system. Several central ideas of B-method are about formal specification, refinement, and implementation. In this paper, all the formal specification would be expressed in AMN.

The structure of this paper would be like a process of software development. The specification of the system would be discussed first and illustrated in empirical modeling perspective. Then a further discussion about the specification in EM perspective would be covered including refinement and implementation. A little verification of the system would also be explored in the paper. At last, a future view for specification in EM would be demonstrated.

2 Formal specifications to the train controller system

2.1 What is the train controller system?

A simple description about a train barrier controller system would like this: there is a barrier for stopping the car drivers, a light for showing the road condition. If the train is approaching, the light should show red and put the barrier down. If the train is leaving or no trains are coming, the barrier should be up and the light is green. If the barrier were up when the train is coming, there would be an accident. But if the barrier is down in a safe condition, this case can also be accepted.

Even for this description, the answer to the sub-title question varies from different viewers. Let us try to answer the question from following relatives:

1. If you were a car driver, the train barrier controller would be the red or green light and up or down barrier.
2. If you were the system controller, who clicks the buttons to control the barrier up or down and to let the light show green or red, buttons are what you care a lot.
3. If you were the system designer, the barrier controller would include following things: the specification about what the system should do properly, the simulation for the system to verify whether it works rightly, the code you write to meet the demand and etc.
4. If you were a viewer from the sky, you can have a whole view about what is happening right now, e.g. where is the train, what is the barrier and light condition, whether a disaster would happen or not.
5. And etc.

From these observations, the train barrier controller means a lot. Therefore how to interpret these images would be difficult. Let us see from the formal way first.

2.2 A formal approach to train controller system

Traditional computation is based on the automata theory. In this way, the train controller system can be divided into barrier and light automata. Each one has two states. For barrier, the approaching would lead up state to the down state while if the train is leaving, then the barrier would up from down condition. So does the light. This kind of automata illustrates the transition from one state to the other. But it is too abstract and hard to be understood. Few people would build up the picture of train controller from several circles.

Formal specification can also constrain the state and transition for the system. The example of specification using B-method is like following:

Barrier AMN	Light AMN	Pseudo Train AMN
MACHINE Barrier	MACHINE Light	MACHINE Train
SEES Light	SEES Barrier	SEES Light, Barrier
SETS	SETS LIGHTCONDTION =	SETS POSITION
BARRIERCONDTION =	{red, green}	VARIABLES
{up, down}	VARIABLES lightcondition	trainheadposition
VARIABLES	INVARIANT lightcondition :	INVARIANT
barriercondition	LIGHTCONDITION	trainheadposition :
INVARIANT	INITIALISATION	POSITION
barriercondition :	lightcondtion := red	INITIALISATION
BARRIERCONDITION	OPERATIONS	trainposition := 0
INITIALISATION	turngreen <-	OPERATIONS
barriercondtion := down	PRE barriercondtion =	trainmove <-
OPERATIONS	down && lightcondition =	PRE there is no crash
barrierup <-	red && trainisleaving	THEN trainpositionupdate
PRE barriercondtion =	THEN lightcondition :=	END
down && lightcondition =	green	END
red && trainisleaving	END;	
THEN barriercondition :=	turnred <-	
up	PRE barriercondition = up	
END;	&& lightcondition = green	
barrierdown <-	&& trainisapproaching	
PRE barriercondition = up	THEN lightcondition := red	
&& lightcondition = green	END	
&& trainisapproaching		
THEN barriercondition :=		
down		
END		

In this way, we can get a clearer picture about the train controller system. The abstract machine is quite like the class idea in objective oriented programming. So we may find three objectives are vital for our system. And we can also find it is more friendly for codes. Several variables, invariants and weakest preconditions are defined to constrain the transition of states and to verify the system behavior. Furthermore, we can find some relationship from one machine and others, which is important for the system structure building. From the formal specification, it is easy to know what the system should do and what may cause some errors.

3 Empirical modeling to the train controller system

Empirical modeling offers a new way for interpreting problems in a computer-based human computing. The observables by definitive script in EM are a kind of reflection or expression of the observation. The relationship between different objectives is illustrated by the idea called dependency. And the agent can be treated as the events, which lead the transition of states. Now let us analyze the train controller system based on observables, dependency and agency.

3.1 The observables, dependency and agency to the train controller system

From our former different observations, the main observables in the system would be the barrier, the light, the position of the train, and the buttons. These different observables are the expressions of different viewers and understandings of the system. The condition of the barrier and light highly depends on the click of the button. And the events of clicking of the button depend on the position of the train. The behavior like clicking the button can be regarded as an agent to change the state of the system. Based on the tools of EM, a developer can draw the picture virtually to simulate the situation, which is a useful way for them to check their thinking.

3.2 The development in the EM

Now let us have a discussion on how to develop this system with formal specification in EM. Firstly, we can assist with some tools like B-tool to write some specifications about the system. These specifications could also be supplied to the user of the system to make sure what the system should do. For each abstract machine, the developer could use the EM tool to draw some pictures for simulation. Furthermore, the operations of the abstract machines can be regarded as the agents in EM, e.g. buttons in the train controller system. The invariant of variables and weakest condition in every operation are the dependency in EM, which should be related in the observables or the agents.

A general process for development in EM should be like following: set down the specification, simulate each AMN, find observables from invariant of variables and weakest condition of operations, and link the operations with some agents.

4 Further discussions on formal specification from EM

Former mentioned process of development could help us develop some systems. But what would we do when we find some more features or demands for the system? Several questions with reference to the model would be proposed like following:

1. If I want to change the color of the light to show the road condition or change the position of the barrier, would these influence our system?
2. If a new train were imported into our system, would it have an impact on the time of a down barrier?
3. If a railway with reverse direction of train was introduced into the system, how is it interpreted into our system?

These questions are raised from the real world. Some questions would be related with the specification in our system, and some don’t. Cases like question 1 would not affect the system specification. In EM, some redefinition could be imported into the definition script to express these changes. And the existing tools can help us to “view” the changes and make more improvement to meet these demands. For Question 2, we may change the definition of the observables and the relative dependency. But the formal specification would not change a lot. We would more interested in dealing with question 3 in formal specification and EM.

4.1 Refinement and implementation in EM

In formal specification, refinement is vital in the process of generating the code from specification. The refinement could include data refinement and operation refinement. A refinement machine would focus on the refinement of the system rather than the existing invariant on variables or the weakest condition on existing operations.

In EM, we can broaden the scope of refinement. The refinement of the system could not only for code generation (to EM code), but also for importing new features to existing components. This second level of refinement is different from the redefinition. A redefinition just changes the existing property. New observables would be imported into the system by writing a new definition script. But the refinement would make the formal specification more related to the definitive script, would import some objectives based on existing observables, would keep and link the dependency of the system components. The idea is a little like the concept of inheritance in objective oriented programming language. A bus class is the son of the car class, which has the whole property of the car class but also has its own characteristic.

In our train controller system, a reverse direction train could be imported based on the existing definitive script. But some more features could be modified to meet the demand, e.g. the length of the train would be different with the existing one, the moving direction is reverse, etc. Correspondently, the dependency of the system should be changed. For example, the barrier and light condition depends on two trains’ positions. Moreover the agents would be different. New buttons would be imported to control or view the situations. Refinement influence in AMN code too. The position of the train could be expressed by the observables in EM. The move operations are based on the observables of the railway.

In conclusion, the refinement in EM would be relate with two aspects:

1. The refinement for the formal specification in EM is used to translate the ideas of specification more related to EM definition script.
2. The refinement for the system in EM is used to import more observables into the system based on the existing ones, and let the modification to these new observables to be available.

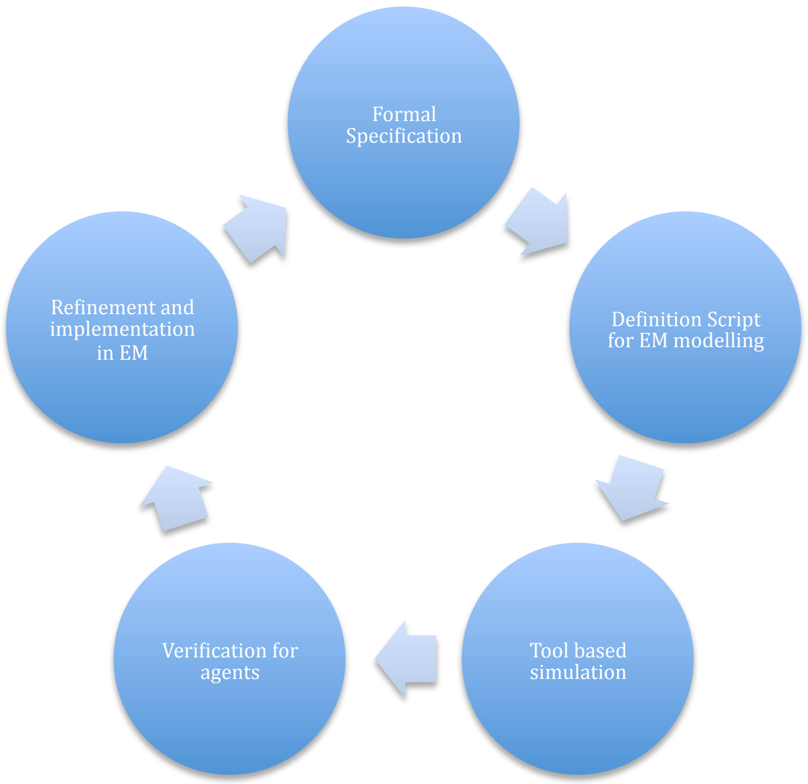
4.2 Verification to the system in EM

Traditional computation theory develops a lot of methodologies for verification of the system. W. Thomas proposes several ideas about automata and reactive systems. Following part would discuss how to do the verification in EM with reference to the train controller system.

In a well-designed and developed system in EM, the verification is based on the observables, dependency and agency. Taking the advantage of EM and EM tools, the users of the system would generate their personal understanding about the system and to compare these with the images in their minds. For a car driver, he or she could observe the case that when the barrier is up, the light is green so that he or she can drive cross the road. When the barrier is down and the light is red, the driver should stop and wait for the train to leave. If there is no accident, the system is safe, although the cost of verification is huge. For a system controller, even though the picture of the simulation was not given, he or she should also get some experience about the system by click buttons several times. Furthermore a robust system should warn the user what to do. After a period of time using the buttons and view the instructions, the system controller should know, even without the simulation, when the system show the train is approaching, the barrier down button should be clicked to avoid the crash. Moreover, for system developers, when they test the system, the relative specification should be applied for them to ensure the system behavior. So the verification to the system in EM is based on the experience of the user and their demands for the system.

5 Future view for specification in EM

The abstract definitive machine (ADM) is quite like the abstract machine in B-method. An abstract definitive machine illustrates the entity by the combination of definitions and actions. More advanced and complete development with formal specification calls some tools for the relation from one to another. The definitions of the ADM should be constrained by the invariant of AMN. The actions of the ADM should be constrained by the weakest condition of AMN. An approach for achieving this goal is to transfer the B-tool into EM definition, which is software for B-method formal specification. It works well for the refinement we discussed here in first level, but the second level of refinement is hard. EM modeler can transfer the mechanism of inheritance into LSD in similarity with some objective oriented programming languages. Basically, the LSD still calls tools for development. If these tools are settled down, the development of the EM model should be like following graph:


Formal Specification
Definition Script for EM modelling
Tool based simulation
Verification for agents
Refinement and implementation in EM

6 Conclusion

The Empirical Modeling offers a novel way for system development with formal specification. The observables, dependency and agency could interpret the specifications of system using AMN well. Refinement in EM further broadens the scope of code generation to inheriting of existing observables, linking dependency with new observables and permitting the agents to verify the system from experience. More tools are required to support the development cycle for system.

References

M.Beynon, J. Rungrattanaubol and J. Sinclair. *Formal specification from an observation-oriented perspective.*
Steve Schneider. *The b-method an introduction*
W M Beynon, S B Russ. *The Interpretation of States: a New Foundation for Computation?*
http://www2.warwick.ac.uk/fac/sci/dcs/research/em/applications/concurrenteng/
http://www2.warwick.ac.uk/fac/sci/dcs/research/em/applications/softwaredevelopment