Applying SOFL to Constructing a Smart Traffic Light Specification

Wahyu Eko Sulistiono[1(✉)] and Shaoying Liu[2]

[1] Graduate School of Computer and Information Sciences, Hosei University, Tokyo, Japan
sulistiono.eko.wahyu.7x@stu.hosei.ac.jp
[2] Department of Computer and Information Sciences, Hosei University, Tokyo, Japan
sliu@hosei.ac.jp

Abstract. Smart Traffic Light (STL) is a system for controlling traffic lights based on patterns of traffic loads in related intersection. Since this is a safety-critical system, we need to construct an accurate specification to build a firm foundation for implementation of the system. In this paper, we describe how the SOFL formal engineering method is applied to construct a Smart Traffic Light specification through the three-step modeling approach of SOFL that helps us manage the complexity and difficulty of constructing a formal specification.

Keywords: Formal specifications · SOFL · Smart traffic light

1 Introduction

Traffic lights have been used widely to control the traffic in junctions in order to avoid collision and congestion. Collision could be avoided by applying safe signal timing to traffic lights, such as minimum yellow signal duration, while congestion could be managed by determining signal timing that corresponds to traffic loads of the junctions.

Currently, many traffic lights are based on fixed signal timing, which works well if the number of vehicles flowing at each direction does not vary significantly throughout the day. However, in many junctions, such as ones near business district or school, the traffic loads at some directions may change significantly at certain hours. To achieve smooth-flowing traffic, these junctions require traffic lights that could change their signal timing. In this paper, we specify a system called Smart Traffic Light (STL) that uses patterns of traffic loads for determining optimal signal timing.

Since STL is safety-critical system and formal methods are considered capable of delivering accurate specification of such system, we use SOFL, one of the formal methods, to construct this system specification. Compared to other formal methods, SOFL offers benefit: SOFL employs an evolutionary approach in constructing a specification, which helps developers deal with difficulty in creating formal specification. It combines waterfall method and transformations to allow developers start from informal specification and progress in steps into formal specification [1–4].

S. Liu and Z. Duan (Eds.): SOFL+MSVL 2013, LNCS 8332, pp. 166–174, 2014.
DOI: 10.1007/978-3-319-04915-1_12,

In this paper we describe how SOFL are employed to construct formal specification of STL. In particular, we describe the development through SOFL three-step modeling. First, we define the system using natural language in informal specification. Second, we define formally all but pre and post condition using SOFL language in semi-formal specification. And finally, we formalize all parts of the system in formal specification.

The remainder of this paper is organized as follows. Section 2 briefly describes SOFL formal engineering method. Section 2 describes the smart traffic light system we devise for this paper. Section 3 discusses how we construct a specification for smart traffic light using SOFL. Section 4 describes our experience and lesson learned from this project. Section 5 discusses related works. Finally, in Sect. 6 we conclude the paper and provide several suggestions for future research.

2 Smart Traffic Light

Current traffic lights generally set to fixed signal timing. This setting, however, could lead to less optimal traffic control when traffic load varies significantly. In order to achieve smooth-flowing traffic, a traffic light needs to adjust its signal timing responding to traffic condition of its junction. In many places, this traffic condition varies forming a pattern. For example, at certain hour and day, the traffic load at direction toward business district increases significantly while at other directions it remains the same. This pattern could be used as consideration when determining signal timing for traffic lights as a way for improving traffic flow.

Smart traffic light (STL) in this paper is a system designed to process historical traffic data to reduce waiting time for vehicles through the traffic roads. In other words, STL uses feedback mechanism to improve its control function. The change in traffic condition will result changes in its control behavior. For example, if one side of the road is getting longer in waiting then the traffic light for that side of the road will have longer green signal.

This system relies on the ability of the elements of the system to communicate to each other in order to coordinate in optimal traffic light control [5]. The system itself comprises of a traffic management center, vehicles, and traffic light controllers, as can be seen in Fig. 1. A traffic management center will function to manage data gathered from others. For example, it will compute green signal duration for each junction. Vehicles are having capability of sending information regarding traffic condition they experience. Last, traffic light controllers are going to use information from traffic management center in controlling their lights.

2.1 Vehicle

In this system, vehicles must transmit traffic data they experience to Traffic Management Center (TMC). To support this, every vehicle is equipped with sensors and wireless communication. The sensor will detect when the vehicle is in queue behind a red light. The vehicle also receives information regarding identity of traffic light where the vehicle in queue for. When the vehicle runs pass through the traffic light,

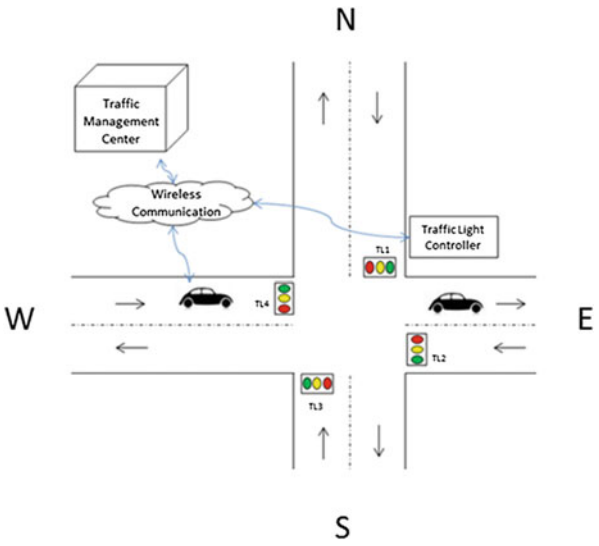

Fig. 1. Smart traffic light

also the sensor will detect it. The process inside vehicle will compute the time taken for queuing and send this information to TMC by wireless communication.

2.2 Traffic Management Center (TMC)

TMC will collect data concerning waiting time to pass a traffic light from all vehicles. This is done by storing data sent by vehicles. Every week, TMC will calculate the average waiting time for every traffic light in same hour for weekday and weekend. For example, TMC will calculate average waiting time for a traffic light in 10.00 in the morning for weekday by averaging all waiting time happened from 10.00 to 11.00 from Monday to Friday. Similarly, TMC will calculate this for weekend, by averaging data happened in Saturday and Sunday. The differentiation happened because the traffic pattern between weekday and weekend is usually different.

By knowing the average waiting time, we can further compute green light timing for every traffic light. We will determine the green time of a traffic light to be proportional to the average waiting time for that traffic light. At the end, TMC will provide this green light timing for all traffic light controllers.

2.3 Traffic Light Controller (TLC)

In each intersection, there are four traffic lights, which are divided to two pairs of traffic lights with the same operation. In Fig. 1 TL1 operation will be the same as TL3 operation, while TL2 operation will be the same as TL4 operation. The operation of the traffic lights will depend on the information about green light timing from TMC. The operation of these traffic lights will be directed by a traffic light controller. One controller is for each junction.

The TLC will fetch information regarding green timing for each pair of traffic lights. Then, this controller will calculate yellow and red light timing. After that, it will turn on traffic lights based on that timing continuously. The same timing will be used for one hour only. After one hour, new timing can be computed and used to run the traffic lights for the next one hour.

The traffic lights in one junction are controlled under one controller to make sure that timing can be run precisely. For example, at the time one pair of traffic lights gets green, the other pair of traffic lights should already be red. If traffic light is controlled independently, then it would need timing synchronization that may be difficult to implement.

In this case study, the system only considers one junction at a time without considering the effect of traffic in other junctions. Furthermore, it is assumed in the current case study, to simplify our system, the junctions do not have traffic light control for pedestrian. Moreover, there is no dedicated turn-right signal.

3 SOFL Specification of Smart Traffic Light

Following SOFL three-step modeling approach, constructing a specification consists of three steps, namely informal specification, semi-formal specification, and finally formal specification. In this section we describes each of these steps for the development of STL specification. In this development we use SOFL Tool, which provides text editor and CDFD diagram editor to support the creation of complete specifications using SOFL language.

3.1 Informal Specification

In informal specification step, we describe requirement of the system using natural language and we structure the requirement of STL into three sections: functions, data resources, and constraints, as shown in the Fig. 2. In this specification, function section describes functionality of STL. We express functionality of vehicle, TMC, and TLC in function 1.1, 1.2, and 1.3 respectively. In each of these functions, we further describe detailed functionality of related element of STL. In data resources section we list two data items: we assume there are 100 junctions for this system and each junction consists of four traffic lights. In constraints section, we describe three constraints. At the end of each constraint, there is a notation (F.1.3), which means this constraint is applied to function 1.3. Note that this reference notation is optional.

3.2 Semi-formal Specification

In the semi-formal specification, we have three tasks. First, we group related functions, data resources, and constraints in modules. Second, we declare any data type needed in this module. Finally, we write pre and post condition for every process in the module using natural language. As an example, Fig. 3 shows semi-formal specification of module Control_Traffic_Light_decom, which represents Traffic Light Controller element of STL.

1 Functions
- 1.1 For every vehicle, count and send waiting time information each time it passes a traffic light
  - 1.1.1 Start time counter when arriving at the queue behind a red light
  - 1.1.2 Send waiting time information to TMC (Traffic Management Center) after passing the traffic light
- 1.2 For TMC (Traffic Management Center), compute green signal duration for every traffic light
  - 1.2.1 Calculate the average waiting time at every traffic light for each hour every day from data collected in a week
  - 1.2.2 Calculate the average waiting time at every traffic light for each hour from the data above
  - 1.2.3 Calculate the maximum waiting time for a road at every junction based on traffic light waiting time on that road
  - 1.2.4 Calculate the green signal duration for every road for every hour based on road's waiting time in a junction
- 1.3 For TLC (Traffic Light Controller), control traffic lights according signal timing from TMC
  - 1.3.1 Set signal timing
  - 1.3.2 Set signal

2 Data Resources
- 2.1 There are 100 junction.
- 2.2 Each junction controls 4 traffic lights

3 Constraints
- 3.1 Traffic signal for one direction is the same as it for the opposite direction. (F1.3)
- 3.2 Traffic lights display signals following this sequence: green, yellow, red (F1.3)
- 3.3 Yellow and green signal in one road must be accompanied by red signal at the other road of the same junction (F1.3)

Fig. 2. Informal specification of smart traffic light

```
module Control_Traffic_Light_decom/SYSTEM_TMS

type
GreenDuration = SYSTEM_TMS.GreenDuration;
SignalTiming = composed of
                roadNS_green : nat0
                roadNS_yellow : nat0
                roadNS_red : nat0
                roadWE_red : nat0
                roadWE_green : nat0
                roadWE_yellow : nat0
        end;

var
ext #green_duration_set : set of GreenDuration;
ext #signal_timing : SignalTiming;
ext #junction_no : nat0;

inv
- Traffic signal for one direction is the same as traffic signal for the opposite
  direction
- Traffic lights display signals following this sequence: green, yellow, red
- Yellow and green signal in one road must be accompanied by red signal at the other
  road of the same junction

process Set_Signal_Timing (hourly_trigger : sign, hour: nat0)
        ext rd green_duration_set
                rd junction_no
                wr signal_timing
        pre green_duration_set must contain exactly one tuple for each junction in
          one hour
        post calculate duration of every signal in road A and road B in this junction
end_process;

...

end_module
```

Fig. 3. Semi-formal specification of module Control_Traffic_Light

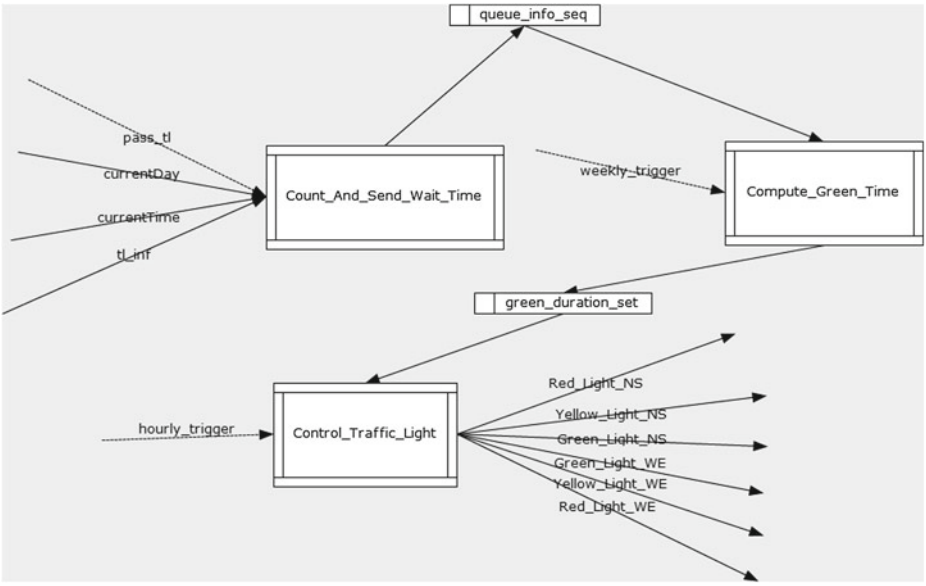

Fig. 4. CDFD of top module

3.3 Formal Specification

In formal specification, from semi-formal specification we develop CDFD diagrams and their module specifications. CDFD will reveal the architecture of the system. It shows the relation between processes. Figure 4 shows CDFD of the top module of STL.

In formal specification, all processes are specified in SOFL language. If a process cannot be specified using pre and post condition, then it may be too complex and therefore need to be decomposed into other modules. In our top module, all processes have to be decomposed into other modules. This decomposition continues until simple processes are achieved. As an example, module Set_Signal_WE does not need decomposition and it can be specified entirely using SOFL language. The CDFD of this module can be seen in Fig. 5 and its module specification can be seen in Fig. 6.

As we complete this formal specification, it can serve as a foundation for implementation of the system.

4 Experience and Lesson

Compared to other formal methods, SOFL has some features that help us reduce the difficulty of creating formal specifications. One of them is CDFD that help us visualize the design architecture. This increases the readability of the specification, which is crucial especially in large specification. Using CDFD we can trace more easily the

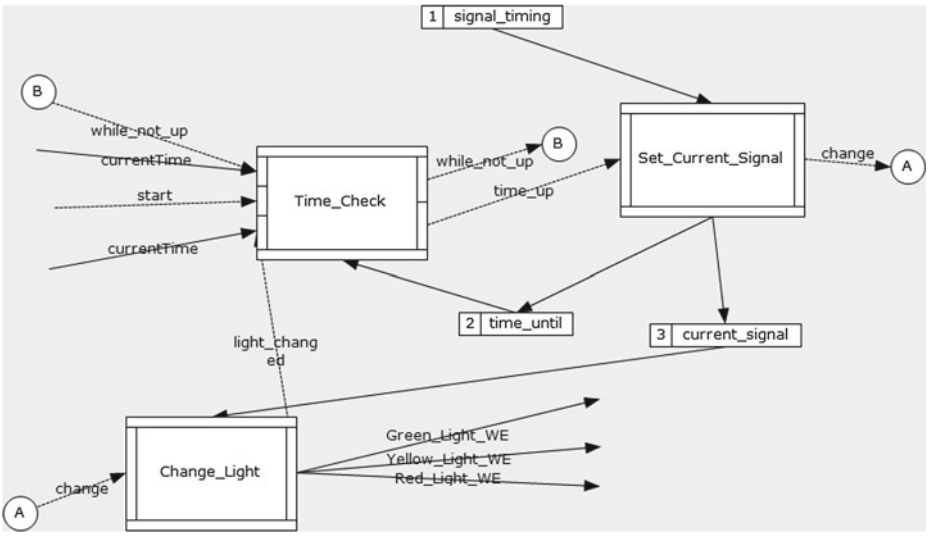

Fig. 5. CDFD of module Set_Signal_WE_decom

```
module Set_Signal_WE_decom/Control_Traffic_Light_decom

type
SignalTiming = Control_Traffic_Light_decom.SignalTiming;
Time = SYSTEM_TMS.Time;
Signal = SYSTEM_TMS.Signal;

var
ext #signal_timing : SignalTiming;
ext time_until : Time;
ext current_signal : Signal;


process Change_Light(change: sign)Green_Light_WE: boolean, Yellow_Light_WE: boolean,
                Red_Light_WE: boolean, light_changed: sign
ext rd current_signal : Signal
pre true
post
   current_signal = <Green> and Green_Light_WE = true and Yellow_Light_WE = false and
   Red_Light_WE = false and bound(light_changed)
or
   current_signal = <Yellow> and Green_Light_WE = false and Yellow_Light_WE = true and
   Red_Light_WE = false and bound(light_changed)
or
   current_signal = <Red> and Green_Light_WE = false and Yellow_Light_WE = false and
   Red_Light_WE = true and bound(light_changed)
end_process

...

end_module
```

Fig. 6. Specification of module Set_Signal_WE_decom

relationships among processes and also the decomposition from top level module to the lowest ones and trace the transformation flow.

In general, constructing a formal specification faces high barrier. The culprit is the difficulties in writing and reading mathematical expression both for developer and user. SOFL method has ameliorated this matter using three-step modeling approach.

It helps developer and user construct the specification in gradual formality. As a result, developer and user can proceed more smoothly in developing the specification. From informal specification which is easily constructed because of the use of natural language and in turn facilitating communication between developer and user, the process proceeds to next steps that introduce formality in part where they have deeper understanding of the system to be built. Finally, they arrived at formal specification. There all are defined formally when they have understood all very well.

In traditional method, natural language is used for writing specification. With that, developers will find it difficult to ensure that there is no inconsistency between parts in the specification. Not only because of the possible ambiguity of natural language, but also lack of tool to think more thoroughly. One important point of using SOFL is that it forces developers to clarify thinking. It helps them think in precise what is going to be built. For example, when the developer defines data type in semiformal specification, they clarify what kind of data will flow between processes or stored in data store. Furthermore, when they writes pre and post condition of processes, they think in precise what kind of input expected and the effect of the transformation to the output of the processes. The consistency among items on the specification is checked. As a result, it is less likely there is wrong data type, because it has already been thought against all related processes.

5 Related Work

SOFL has been used successfully in various case studies. In [3] SOFL is applied to construct a specification for railway crossing controller, a safety-critical and real-time system that produce signal of coming trains and control crossing gates. In [6] insulin pump system, a safety-critical embedded system for controlling insulin injection, also has been constructed using SOFL. In [7] SOFL is also used to carry out case study of auto-cruise control for the purpose of hazard analysis. In [8] SOFL has been tried on non-safety critical system, i.e. university information system. In [9] the specification of automatic automobile driving simulation system is also constructed using SOFL. Finally, [2] employs ATM (Automated Teller Machine) as a case study for evaluating effectiveness of the framework for developing dependable software systems using the SOFL formal engineering method.

Although SOFL has been applied in various case studies, none has constructed a specification for traffic light system. Thus, this paper is the first SOFL case study on traffic light system. Compared to those case studies, this system shares some same characteristic, such as real-time and safety-critical, which are good reasons to employ formal methods like SOFL. Moreover, with more potential features, this system becomes a complex system, which needs such method even more.

6 Conclusion and Future Work

In this paper, we have presented the development of a smart traffic light specification using SOFL formal engineering method. This method helps us deal with the complexity of the system while maintaining preciseness of the specifications. In this case

study, the system can be specified through combination of CDFD and module specification.

In the future, we will add more features to STL. We will consider more collaborative work among elements of the system. For example, we are going to coordinate multiple junctions to provide better traffic control. Furthermore, we will develop inspection methods to verify the specification.

Acknowledgement. This work has been conducted as a part of "Research Initiative on Advanced Software Engineering in 2012" supported by Software Reliability Enhancement Center (SEC), Information Technology Promotion Agency Japan (IPA).

References

1. Liu, S.: Formal engineering for industrial software development – an introduction to the SOFL specification language and method. In: Davies, J., Schulte, W., Barnett, M. (eds.) ICFEM 2004. LNCS, vol. 3308, pp. 7–8. Springer, Heidelberg (2004)
2. Liu, S.: A framework for developing dependable software systems using the SOFL formal engineering method. In: 2010 International Conference on Intelligent Computing and Integrated Systems (ICISS), pp. 561–567 (2010)
3. Liu, S., Asuka, M., Komaya, K., Nakamura, Y.: Applying SOFL to specify a railway crossing controller for industry. In: Proceedings of the 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques 1998, pp. 16–27 (1998)
4. Liu, S.: Formal Engineering for Industrial Software Development. Springer, Heidelberg (2004)
5. Wang, Q., Hu, J., Wang, Y., Zhang, Y.: A simulation study for the communication subsystem of wireless traffic information service system. In: 7th International Conference on Information, Communications and Signal Processing 2009, ICICS 2009, pp. 1–6 (2009)
6. Wang, J., Liu, S., Qi, Y., Hou, D.: Developing an insulin pump system using the SOFL method. In: 14th Asia-Pacific Software Engineering Conference 2007, APSEC 2007, pp. 334–341 (2007)
7. Abdullah, A.B., Liu, S.: Hazard analysis for safety-critical systems using SOFL. In: 2013 IEEE Symposium on Computational Intelligence for Engineering Solutions, Singapore (2013)
8. Liu, S., Shibata, M., Sato, R.: Applying SOFL to develop a university information system. In: Proceedings of Sixth Asia Pacific Software Engineering Conference 1999, APSEC 1999, pp. 404–411 (1999)
9. Mat, A., Liu, S.: Applying SOFL to construct the formal specification of an automatic automobile driving simulation system. In: International Conference on Software Technology and Engineering, vol. 3308, pp. 42–48. World Scientific Publishing, Chennai (2009)