Modeling Agent-Based Traffic Simulation Properties in Alloy

F. Araujo, J. Valente, R. Z. Wenkstern
Multi-Agent and Visualization Lab
University of Texas at Dallas
Richardson, TX, USA
{frederico.araujo, juniavalente, rymw}@utdallas.edu

Keywords: multi-agent systems, traffic simulation, intelligent transportation systems (ITS), formal specification, Alloy.

Abstract

The advances in Intelligent Transportation Systems (ITS) call for a new generation of traffic simulation models that support connectivity and collaboration among simulated vehicles and traffic infrastructure. In this paper we introduce MATISSE, a complex, large scale agent-based framework for the modeling and simulation of ITS and discuss how Alloy, a modeling language based on set theory and first order logic, was used to specify, verify, and analyze MATISSE’s traffic models.

1. INTRODUCTION

For the past twenty years, Intelligent Transportation Systems (ITS) have been considered as possible solutions for the traffic safety and congestion problems. ITS are defined as “the application of advanced sensor, computer, electronics, and communication technologies and management strategies in an integrated manner to increase the safety and efficiency of the surface transportation system” [19]. The work presented in this paper is based on a novel, multilayered integrated ITS for safety improvement and congestion reduction. The ITS infrastructure is the result of discussions conducted by a group of researchers at the University of Texas at Dallas [3, 25]. Traffic is viewed as a bottom-up phenomenon that is the consequence of individual decisions at the micro-level, and traffic management as a top-down activity that is the result of decisions taken at the macro-level. Both macro and micro-levels consist of multi-agent based infrastructures where autonomous entities continuously communicate and interact with each other. Even though some of the proposed ITS components have already been implemented, the overall infrastructure is still in its conceptual stage.

Given the critical role of interactions among ITS components and their independent decision making capabilities, the use of simulation techniques to test traffic scenarios under nominal and extreme conditions was necessary. MATISSE (Multi-Agent based Traffic Safety Simulation systEm) is an agent-based “tailor made” simulation framework that was designed to provide a platform for the execution of such scenarios. The design of this large-scale, distributed, multi-agent based simulation framework revealed the need for the definition of additional entities, adding a layer of complexity to the problem. Before embarking on the full-scale development of MATISSE, the specification and validation of the simulation framework’s properties proved to be necessary.

Alloy is a modeling language based on set theory and first order logic that has been used in both industry and academia to validate a wide variety of systems [7, 10, 16]. The language has a simple and concise syntax and comes with a powerful, integrated tool for compiling and analyzing models. The purpose of this paper is to present a formalization of the MATISSE model in Alloy, and discuss how the model’s core properties are verified using Alloy’s Analyzer. In particular, we discuss an approach to produce execution traces from the specification. These traces serve two purposes: they allow for a thorough analysis and evaluation of the traffic model; and demonstrate the suitability of MATISSE for the simulation of ITS scenarios.

In the following section we give an overview of related works. In Section 3 we briefly present the proposed ITS and MATISSE’s high level architecture. In Sections 4 and 5 we discuss how Alloy was used to specify, verify, and analyze MATISSE’s model. Finally, in Section 6 we share the lessons learned from this experience.

2. TRAFFIC SIMULATION

There are two major approaches to simulate traffic scenarios. Macroscopic models [1, 17] describe traffic as a physical flow of fluid and make use of mathematical equations relating macroscopic quantities (e.g., traffic density, flow rate and average velocity). These models assume rational driving behavior and fairly consistent traffic streams and thus are unfit to model real traffic operations.

In contrast, microscopic models consider the characteristics of individual traffic elements (e.g., vehicles, traffic lights, traffic signals, driver behavior) and their interactions. Typical microscopic models are based on analytical techniques (e.g., queuing analysis, shock-wave analysis) [14] and assume traffic elements with predefined behavioral models. This is a limitation since realistic traffic simulation scenarios call for the modeling of unexpected behavior and unforeseen environmental conditions. The multi-agent paradigm alleviates this
limitation by providing means to address non-deterministic behavior in non-deterministic, unpredictable environments.

Over the last decade, a large number of agent-based traffic simulation systems have been proposed. Some focus on specific small scale traffic problems (e.g., driver behavioral modeling, tactical driving, evacuation management, intersection management) [11, 22, 24] while others attempt to tackle complex large scale traffic scenarios [2, 6, 12]. In this section we restrict our discussion to those that best compare to MATISSE, namely MatSim [2], and Transims [6].

MatSim [2] is an agent-based framework for modeling transport demand. MatSim represents individual travelers as agents endowed with predefined plans. These agents follow a utility based strategy to determine their optimal daily plan. Interactions among agents are implicitly encoded into the agent’s utility function. In its current version, traveler agents cannot directly interact with other agents. In addition, agents are not capable of perceiving their environment dynamically. They act upon global environmental knowledge seeded at initialization time.

Similarly, Transims [6] is a large-scale microscopic simulation system for transportation planning and congestion evaluation. In Transims travelers are modeled as agents which can walk, drive cars, or use buses. Traveler agents can decide which plan to select depending on their current state but they cannot dynamically perceive their environment. It is also unclear whether they can interact with other agents. Transims’ environment is static and fully observable, thus reducing its capabilities to model complex and realistic scenarios.

Our work enhances the conventional urban traffic simulation by proposing a multi-agent based framework that simulates macro and micro-level traffic entities and their interactions within and across levels. The unique characteristics of MATISSE are: 1) The simulation environment is open, i.e., non-deterministic, dynamic, inaccessible and continuous [23]. The environment has mechanisms that allow the simulation of event propagation. 2) The agents are not given global environmental knowledge to act upon. They dynamically perceive their surroundings through various senses (e.g., vision, hearing, smell). 3) At run-time, the user can change the properties of the simulated agents (e.g., driver “awake” to driver “asleep”, disable agent sensors) and the environment (e.g., change the laws that govern the environment) without interrupting the simulation. To the best of our knowledge, no other existing framework offers this feature.

A recent system called JaSim [12] was developed along the same premises as MATISSE. Even though it shares the same environment structure and similar agent perception mechanisms, it lacks the advanced simulation features of event propagation and dynamic property modification discussed above.

3. OVERVIEW OF ITS AND MATISSE

In this section, we briefly present the main components of the proposed ITS and discuss MATISSE’s architecture. More detailed discussions on these topics can be found in [3, 25, 26].

3.1. Elements of a novel ITS

The proposed ITS aims at enforcing communication, interaction, and collaboration between various types of elements defined at various levels of abstraction1.

The infrastructure is based upon two underlying concepts:

- In order to manage a large environment efficiently, it is necessary to partition the space into smaller defined regions called traffic area;
- Each traffic area is assigned a tower. A tower is required to: 1) autonomously manage environmental information about its traffic area; 2) be aware of the traffic elements (e.g., vehicles, traffic devices) located in its defined area; 3) be able to interact with local traffic elements to inform them about changes in their surroundings; 4) be able to communicate with other towers to inform them of external events.

In order to manage traffic information efficiently, traffic towers are organized as a hierarchy (see Figure 1). This structure is particularly important for the case when towers need a higher level of knowledge to properly manage their traffic areas. For example, if congestion is caused by an accident in an area, and the micro-level information is insufficient for the tower to determine the best exit route for its local vehicles, it will communicate with a higher level traffic tower to obtain a broader image of the traffic.

The micro-level entities are classified in two categories: Mobile Context-Aware Intelligent (CAI) vehicles [4]. These are vehicles equipped with devices that allow them to 1) monitor the driver’s behavior in order to prevent possible accidents; 2) communicate with other vehicles and traffic devices; and 3) interact with the traffic tower infrastructure to obtain traffic information and guidance in real time.

Stationary Context-Aware Intelligent (CAI) traffic devices. These include traffic lights, traffic collection devices, and relay units. They serve the purpose of improving safety and traffic flow on roads and highways by providing information about the physical traffic infrastructure and congestion condition. Traffic lights are equipped with adaptive systems that allow them to 1) interact with the traffic tower infrastructure to obtain traffic information in real time, 2) communicate

1In the remainder of this paper we will use the word “micro-level” element to refer to an entity that has very limited knowledge of the state of the world. In contrast, a “macro-level” element refers to one that is aware of a larger portion of the world.
with vehicles for intersection coordination, and 3) communicate with other traffic light controllers to improve traffic flow when necessary. Traffic collection devices are used on highways (e.g., toll units) to collect information about traffic, and communicate the information to the traffic management system for further analysis (e.g., identification of a drunk driver on the highway). Relay units are used to pass on information between the various communicating entities when the physical distance is too great.

### 3.2. MATISSE Architecture

MATISSE is a “tailor made” multi-agent based simulation platform designed to specify and execute simulation models for the above-mentioned ITS. We define an agent as a software entity which [21]: 1) is driven by a set of tendencies in the form of individual objectives; 2) can communicate, collaborate, coordinate and negotiate with other agents; 3) possesses resources of its own; 4) executes in an environment that is partially perceived; 5) possesses skills and can offer services. A virtual agent is an application specific agent that represents a real world concept (e.g., vehicle, traffic device).

MATISSE defines virtual agents for each micro- and macro-level element used in the ITS. Vehicle agents simulate the behavior of human drivers, have individual goals (e.g., arriving at some destination in a reasonably short time), influence other agents (e.g., turning signals and changing lanes), and are governed by environmental norms and constraints (e.g., speed limits and traffic signals). Traffic light and traffic collection agents are aware of and influence nearby vehicles, are able to perceive and adapt to changing conditions, and collaboratively work to achieve certain objectives. Finally, Traffic Tower agents autonomously manage and control their traffic area, including the vehicles and traffic devices they enclose.

In addition to these virtual agents, and for software design purposes, it is necessary to introduce two design related concepts: a cell is a repository that encompasses all information related to a traffic area. A cell controller is a special purpose agent whose main role is to consistently provide virtual agents located within its cell with a correct perception of their surroundings. This is a complex and critical role in any realistic simulation. More information on this topic can be found in [20]. It is important to note that a cell controller does not correspond to a real world concept since real perception is achieved through physical sensors.

#### High Level Architecture

As shown in Figure 2, MATISSE’s high level architecture includes three main components: the Agent-Environment System (AES) creates simulation instances; the Data Management System (DMS) stores and processes information collected from the AES; and the Visualization Framework receives information from the DMS and creates 2D or 3D images of the simulation.

![MATISSE High Level Architecture](image)

**Figure 2.** MATISSE high level architecture

#### Matisse’s Virtual Agent Platforms

The four types of agents identified by MATISSE are naturally managed by four distinct agent platforms within the Agent-Environment System (AES) component. The Virtual Vehicle Platform manages mobile agents that represent vehicles. Vehicle-agents are created by the Vehicle-Agent Management Component, and vehicle-agents communicate with each other through the Vehicle-Vehicle Message Transport Service. The Virtual Traffic Device Platform manages stationary agents that represent traffic lights, relays and information collection devices. The Traffic-Device-Agent Management Component creates and manages traffic-device-agents within the simulation while Device-Device Message Trans-
port Service handles communication between these stationary traffic-agents. The Virtual Tower Platform creates and manages the hierarchical infrastructure of traffic-tower-agents. Finally the Simulated Environment Platform creates and manages cell controllers. The Environment Agent Management Component creates cell controllers, assigns them to a cell, and maintains the cell controller hierarchy for the simulation.

4. SPECIFYING MATISSE IN ALLOY

Due to the scale and complexity of the simulation architecture, from a software engineering perspective, we found it necessary to formally specify and validate various simulation properties before starting the implementation of MATISSE. In this section we briefly introduce the Alloy language [15] and present a specification of the simulation properties of MATISSE in Alloy.

4.1. Overview of Alloy

In the past two decades, several formalisms have been proposed for multi-agent systems (e.g., temporal logic, multi-modal logic). These formalisms are generally abstract and not related to concrete computational models [9]. Other approaches have used traditional formal languages such as Z and CSP [5, 18]. While providing an accessible notation, these formalisms lack the diagrammatic representation and tool support necessary to effectively analyze models.

Alloy is a specification language based on set theory and first-order relational logic [15]. The language has a simple and concise syntax that can represent complex structural properties and behavior. It comes with an Analyzer, a powerful, integrated tool for compiling and analyzing models. The Analyzer supports two types of automatic analysis: 1) the search for an instance that satisfies all the constraints and relations specified in a model; 2) the identification of a counterexample that violates the assertions specified in a model. Both analysis are performed within a user defined scope that bounds the cardinality of entity sets in instances of the model. Outputs can be graphically depicted using the visualizer and evaluated using the command-line evaluator.

Alloy has been used in both industry and academia [7, 10, 16]. Jackson and Vaziri [16] have proposed an approach to verify Java methods in Alloy. At IBM, a subset of Alloy has been used to develop a technique for efficient checking of data structure invariants [10]. Alloy was also used in [8] to test and find bugs in Galileo, a dynamic fault tree analysis tool used at NASA [7].

4.2. Specification of MATISSE Static Properties

The static properties of a model describe the conceptual entities and their relationships. In Alloy, these are specified through the signature declaration.

For example, module TrafficSimulationEntity, in Figure 3, specifies vehicle-agents, traffic-light-agents, and tower-agents. It also specifies VirtualEnvironment, TrafficArea, Cell and CellController. The one keyword constrains the model to one virtual environment. Simulation events (both external and internal) are specified by Event.

module TrafficSimulationEntity
abstract sig VirtualAgent()
sig Vehicle extends VirtualAgent()
sig TrafficLight extends VirtualAgent()
sig Tower extends VirtualAgent()
sig one VirtualEnvironment{}
sig TrafficArea{}
sig Cell{}
sig CellController{}
sig Event{}

Figure 3. Traffic Simulation Entities

Figure 4 shows a partial specification of MATISSE’s model. module TrafficSimulation makes use of the elements defined in module TrafficSimulationEntity to specify the relations and constrains of the model. An example of a relation, in sig Simulation, is guide that corresponds to the relationship between tower-agents and vehicle-agents. The aggregation of module TrafficSimulationEntity and TrafficSimulation makes up the complete MATISSE simulation model.

module TrafficSimulation
open TrafficSimulationEntity
sig Simulation{
  dividedIntoArea: VirtualEnvironment one → TrafficArea,
  control: Tower one → TrafficLight,
  guide: Vehicle one → Vehicle,
  manage: Tower one → TrafficArea,
  containTower: TrafficArea one → one Tower,
  containVehicle: TrafficArea one → Vehicle,
  towerCollaborate: Tower → Tower,
  ·
  visionVehicle: CellController one → Vehicle,
  ·
  knows: Vehicle → Event,
  requestVicinity: Vehicle → CellController,
  grantVicinity: CellController → Vehicle → Vehicle
  sendEvent: Vehicle → Event → Tower,
  notifyEvent: Tower → Event → Vehicle,
  propagateEvent: Tower → Event → Tower
}

Figure 4. Partial Specification of MATISSE’s Model

Alloy enables the precise specification of static properties such as “each virtual traffic area is assigned a tower-agent”. Using relation multiplicities, containsTower:
TrafficArea one → one Tower specifies a one-to-one relation between traffic area and tower elements. Further, the constraint containTower = ¬manage ensures that each tower-agent is assigned to a unique traffic area, and that each area is uniquely associated to its tower-agent.

4.3. Specification of MATISSE Dynamic Properties

In Alloy, operations are specified through predicates, which relate valid instances of Simulation through a change in its composition. For instance, the predicate requestCCVicinity adds the relation between a vehicle and its cell controller to \( s \) in order to produce \( s' \), in which \( s \) and \( s' \) denote the before and after states of Simulation.

\[
\text{pred requestCCVicinity[vc: Vehicle→CellController, s, s':Simulation]} |
\text{vc in [Simulation→Vehicle]} |
\text{s' requestVicinity = s requestVicinity + vc}
\]

Thus far, the presented specification produces unrelated instances of the MATISSE simulation model. This is not sufficient for modeling simulation scenarios where a sequence of operations relating different instances of Simulation is required. As such, we extend our model with execution traces to allow the ordered execution of operations. To produce execution traces, we specify a linear ordering over Simulation elements (see Figure 5).

![Figure 5. (a) Unrelated instances of the model (b) Execution trace of the model](image)

This is achieved by importing the library module util/ordering. This module includes functions first, next, and last. As depicted by Figure 5 (b), first returns the first element \( S1 \), \( s1.next \) returns \( S2 \) and \( s2.next \) returns \( S3 \), and last returns the last element \( S3 \).

The following fragments of MATISSE’s specification illustrate the new constraints added to the model to enable execution traces. The pred init defines the initial conditions (i.e., the initial composition) and pred inv defines invariants (i.e., properties that never change during an execution trace) of Simulation. Any adjacent Simulation in the ordering is related by fact traces. For instance, if a vehicle requests its vicinity in \( s \), then the vehicle’s vicinity will be granted in \( s' \) through operation grantVehicleVicinity.

```alloy
module TrafficSimulation
open util/ordering[Simulation] as t

pred init[s:Simulation] { ... }

pred inv[s, s':Simulation] {
  s' dividedIntoArea = s dividedIntoArea
  s' containTower = s containTower
  ... }

fact traces {
  init[first]
  all s:Simulation - last | let s' = s next {
    inv[s, s']
  ...

  (#s knows ≠ 0 and #s requestVicinity ≠ 0 and #s grantVicinity ≠ 0) ⇒ {
    s' knows = s knows
    s' requestVicinity = s requestVicinity

    let v = (s requestVicinity) CellController | {
      (v in (s requestVicinity) CellController) and
      (v not in Vehicle (CellController (s grantVicinity)))
      ⇒ {
        let x = (cellControllers[v Simulation] Vehicle)
        & (CellController vicinityVV[v Simulation]) &
        grantVehicleVicinity[flip23[x], s, s']
      }
    }
    else s' grantVicinity = s grantVicinity
  }
  ...
}
```

With this specification, it is possible to analyze the static and dynamic properties of MATISSE. In addition, a number of ITS traffic scenarios involving collaboration, information dissemination, and event propagation can be planned and designed to validate MATISSE’s traffic model.

5. ANALYZING MATISSE’S PROPERTIES

In this section, we show how the above-discussed Alloy models can be analyzed to ensure the consistency of the specification and satisfaction of MATISSE’s traffic model properties. Hereafter, we refer to the consistency checking of the Alloy specification as verification, and reserve the term validation to the activity of ensuring that the specified model copes with the intended high level requirements of MATISSE.

5.1. MATISSE’s Properties Verification

For the purposes of verification, the Alloy Analyzer is used to find instances that violate the assertions specified in the model. In particular, we show how static and dynamic properties of the MATISSE simulation model can be verified using Alloy, thus ensuring the necessary rigor to analyze the specification against design flaws.

The assertion VehicleSendEventOnlyToTowerGuiding states that for all instances of the Simulation a vehicle can send an event (through relation sendEvent) only to the tower guiding it. No counterexample is found for this static property following the constraining fact (sendEvent.Tower).(Vehicle.sendEvent) in `guide specified in Simulation`.

```alloy
assert VehicleSendEventOnlyToTowerGuiding{
  all s: Simulation{
    let v = (s sendEvent Tower) Event |
    (v ≠ none) implies
    Event (v (s sendEvent)) = (s guide) v
  }
}
```
The assertion RequestIsGranted is an example of verification of a dynamic property of the model, in which we ensure consistency between adjacent instances of Simulation. It states that if a vehicle makes a request for vicinity in s, then the request must be granted in s’. No counterexample is found for this property.

```
assert RequestIsGranted(
  all s:Simulation-last, s':s-next |
  let v = (s/requestVicinity) CellController |
  (v - Vehicle (CellController (s/grantVicinity)))
  none ⇒
  v in Vehicle (CellController (s'/grantVicinity))
)
```

5.2. Traffic Scenario Description

The scenario depicted in Figure 6 demonstrates the suitability of MATISSE for safety improvement and congestion reduction. This ITS scenario consists of vehicles driving on a one-way road. An event (e.g., an accident, an obstruction on the road, or any other abnormal condition) has occurred in Traffic Area A0, and vehicle V0 perceives the event within its field of vision (shown as a green cone). Under this scenario, V0 takes the following steps: 1) Informs all vehicles located in its close vicinity 2 (shown as a circle) about the perceived event via vehicle-to-vehicle interactions. The notified vehicles are able to take the necessary actions to avoid a major accident. 2) Informs traffic tower T0 about the perceived event via vehicle-to-infrastructure interactions.

After deliberation and based on the event characteristics, T0 alerts the vehicles located in A0 under its control (i.e., V1 to V4) about the event (to enhance safety). T0 also determines the potential impact of this event on to neighboring traffic areas and informs the adjacent traffic tower T1 of the event. T1 deliberates, and informs all vehicles located within Traffic Area A1 of the event and guides them in their choice of the best alternate route to follow (to avoid congestion). All vehicles in the traffic area make use of the broader traffic information to improve the overall safety condition and avoid traffic congestion.

5.3. Traffic Scenario Validation

The execution traces generated from the Alloy models validate the specified MATISSE’s properties (i.e., virtual agent perception, agent-to-agent interaction, and event propagation) with respect to the above-described traffic scenario. Additionally, these traces highlight how safety improvement and congestion reduction goals are achieved in MATISSE.

The execution of the model produces the execution trace consisting of the following sets of elements:

---

2 A vehicle’s vicinity represents the vehicles that are positioned within the vehicle’s range of communication, determined by the available communication technology [27].
there邻里塔楼，由事件（例如，Tower1）和传递 Event 到其中。这在 propagateEvent[Event] 关系中体现了。接收到此信息后，Tower1 存储 Event 信息到其知识库中，以关系 towerKnowsViaTower 表示。

在 Simulation3 中，Vehicle0 传递了 Event 到 Vehicle1 和 Vehicle2，其后将该信息存储到各自的知识库中，关系 knowsViaVehicle 表示。接收 Event 信息后，所有 Tower0 区域的车辆存储事件信息到其知识库中，由 knowsViaTower 关系表示。同时，Tower1 也通过 notifyEvent[Event] 关系将其事件信息传递给所有本地车辆。

在 Simulation4 中，所有 Tower1 区域的车辆（Vehicle5, Vehicle6, Vehicle7, Vehicle8）存储事件信息到其知识库中，关系 knowsViaTower 表示。

6. LESSONS LEARNED

通过在 Alloy 中指定和执行 MATISSE 的模型，我们学习到以下要点：

Abstraction. 通过抽象掉实现细节，Alloy 能够让我们聚焦在系统设计的最重要方面，并探索设计的多种可能。例如，它允许我们重新审视当模拟代理间的交互（例如，车辆与交通塔台之间的协作，事件在交通区域间的传播）时分配给代理的各种责任。

Process. 我们以较小模型开始设计活动，随后在不编写单行代码的情况下逐步增加细节。每一步中，我们都会模拟、检查并分析系统行为的不同状态，从而验证其高层属性。

Model Execution. 包括 trace 执行的步骤，是一个非常有价值的工具，它使我们能够识别模型中的概念性不一致。逐步的场景执行能够让我们对交通模型的多种状态进行分析和验证，从而验证其高层属性。

虽然我们遇到的困难是由于时间因素的困难。例如，没有内置机制允许我们指定和验证实时属性，例如通信时间的约束。尽管 Alloy 包含了一些基本的时间约束，但它在指定更复杂的时序属性方面不如传统的时间逻辑。此外，在指定行为约束以确保相邻 trace 执行的前一个和后一个的交互时，我们需要注意不要过度约束模型。在这点上，根据我们的观察，我们相信将 Alloy 结合 Statecharts [13] 可以克服这些限制，允许我们为模型指定和验证实时属性，而其提供的行为表达更加自然。最终，尽管 Alloy 显然非常简洁，但用于指定涉及通信和合作的代理间的复杂交互的能力对于 Alloy 来说非常重要。

基于这些观察，我们认为一种结合 Alloy 和 Statecharts [13] 的方法可以克服这些限制，允许为模型指定和验证实时属性，而其提供的行为表达更加自然。最终，尽管 Alloy 显然非常简洁，但用于指定涉及通信和合作的代理间的复杂交互的能力对于 Alloy 来说非常重要。

基于这些观察，我们认为一种结合 Alloy 和 Statecharts [13] 的方法可以克服这些限制，允许为模型指定和验证实时属性，而其提供的行为表达更加自然。最终，尽管 Alloy 显然非常简洁，但用于指定涉及通信和合作的代理间的复杂交互的能力对于 Alloy 来说非常重要。

Based on these observations, we believe that an approach combining Alloy and Statecharts [13] can alleviate these limitations, allowing for the incorporation of real-time constraints while contributing to a more natural representation of behavioral properties. Furthermore, this extended formalism could help with the modeling and analysis of concurrent behavior.
7. CONCLUSIONS

MATISSE is a multi-agent based simulation platform designed to specify and execute traffic simulations for a new generation of ITS. MATISSE’s unique features include its open, decentralized and distributed environment; the ability of agents to perceive their surroundings in simulated real time; and the ability to execute micro- and macro-level ITS scenarios within the same framework.

In this paper we discussed how Alloy was used to specify and analyze the static and dynamic properties of such a complex simulation system, and shared the lessons learned from this experience. Future work includes determining how to integrate Alloy models with state-charts to incorporate real-time constraints and obtain a more natural representation of behavioral properties, as well as to evaluate the scalability of the Alloy specification for more complex interaction patterns.

Acknowledgments. This project is partially supported by Rockwell Collins under the grant number 5-25143.

REFERENCES