

**COMPUTATIONAL MODELING AND FORMAL
ANALYSIS TECHNIQUES IN INTERDISCIPLINARY
STUDIES OF COMPLEX SYSTEMS**

by

Mona Vajihollahi

B.Sc., Sharif University of Technology, 2001

M.Sc., Simon Fraser University, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

© Mona Vajihollahi 2009
SIMON FRASER UNIVERSITY
Summer 2009

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author.

APPROVAL

Name: Mona Vajihollahi
Degree: Doctor of Philosophy
Title of thesis: Computational Modeling and Formal Analysis Techniques in Interdisciplinary Studies of Complex Systems

Examining Committee: Dr. Binay Bhattacharya
Chair

Dr. Uwe Glässer, Senior Supervisor

Dr. Patricia Brantingham, Supervisor

Dr. Vahid Dabbaghian, Supervisor

Dr. Arthur Kirkpatrick,
Associate Professor of Computing Science
SFU Examiner

Dr. Warren Hare, External Examiner,
Assistant Professor of Mathematics
University of British Columbia Okanagan

Date Approved:

July 23, 2009

The logo for Simon Fraser University, consisting of the letters 'SFU' in a white, bold, sans-serif font on a black rectangular background.

SIMON FRASER UNIVERSITY
LIBRARY

Declaration of Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the "Institutional Repository" link of the SFU Library website <www.lib.sfu.ca> at: <<http://ir.lib.sfu.ca/handle/1892/112>>) and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada

Abstract

Computational methods offer new problem solving and analysis techniques that play a key role in advancing the boundaries of many research disciplines. Particularly for social systems, the precision and rigor offered by mathematical models facilitate establishing a clear understanding of the underlying complex system. Computational models also allow for dynamic testing and computer-assisted experiments that may be impossible to carry out in the real world.

Modeling something as complex and diverse as a social system is a highly iterative and potentially open-ended process, which calls for software development techniques that address its specific needs. We present an integrated methodological framework and tool environment for design, validation, and simulation of models of complex social systems. We illustrate consistency and applicability of the framework through novel applications in two different interdisciplinary contexts: Computational Criminology and Identity Management Systems (IMS).

The Computational Criminology project, called Mastermind, aims at developing computational models of criminal behavior to facilitate systematic experimental studies of a wide range of criminal activities in urban environments. The Mastermind model, developed in close collaboration with criminologists, focuses on spatial and temporal aspects of different forms of crime. Pushing beyond conventional empirical research, it provides a solid basis for engaging the use of computational thinking and social simulations in crime analysis research and practice.

The IMS project, called Identity Management Architecture, aims at consolidating diverse multidisciplinary views on identity management in a systematic fashion. We propose a firm semantic foundation that facilitates a rigorous study of IMS and provides improved accuracy in reasoning about their key properties. The proposed framework is built upon essential

*To my Mom,
and to my anchor, my Dad.
To Madar.
And to the light of these days...*

*“Knowing is not enough; we must apply.
Willing is not enough; we must do.”*

— GOETHE

Acknowledgments

I would like to express my gratitude to all my colleagues, my friends, and my family, who touched my life in one way or another during the last 5 years, and helped me be the person I am today.

My special thanks go to:

Dr. Uwe Glasser for giving me the opportunity to venture this new field of research,

Dr. Patricia Brantingham for being a true mentor and for encouraging me in every step of this journey,

Dr. Warren Hare and Dr. Vahid Dabbaghian, the directors of the MoCSSy program, for their endless support for me personally, and for creating an environment that nurtured exciting collaborations and close friendships,

Dr. Carrie Matteson for her incredible professional insight and for being amazingly kind,

Dr. Diane Finegood for giving me the opportunity to explore my strengths,

My friends and colleagues,

Piper for his work and help on the Mastermind project,

Laurence for patiently proof-reading pages of my writing, and

Roosbeh, Mike, Dylan and Flavia for sharing parts of my graduate life.

Contents

Approval	ii
Abstract	iii
Dedication	v
Quotation	vi
Acknowledgments	vii
Contents	viii
List of Tables	xii
List of Figures	xiii
List of Acronyms	xiv
1 Introduction	1
1.1 Modeling Framework for Social Systems	2
1.2 Case Studies	3
1.3 Significance of Our Research	4
1.4 Thesis Structure	6
2 Background and Motivation	7
2.1 Formal Modeling Using Abstract State Machines: A Case Study	8
2.2 Abstract State Machine Method	9
2.3 Distributed Abstract State Machines	10

2.3.1	Partially Ordered Runs	11
2.3.2	Reactivity and Time	12
2.4	ASM Ground Models	13
2.5	ASM Refinement Method	14
2.6	Summary	15
3	Modeling of Complex Social Systems	16
3.1	Complex Systems Modeling	16
3.2	Social Simulation	18
3.3	Multi-Agent Systems (MAS)	19
3.3.1	Agency and Autonomy	20
3.3.2	Architecture	21
3.3.3	Society of Agents	22
3.4	Formal Approaches to Agent-Based Systems	24
3.5	Multi-Agent Based Simulation (MABS)	27
3.6	Agent-Based Social Simulation (ABSS)	28
3.7	Summary	29
4	Vision and Methodology	31
4.1	Framework Architecture	32
4.2	Formal Modeling Technique	35
4.2.1	Rapid Prototyping with CoreASM	38
4.2.2	Interactive Design with Control State ASMs	38
4.3	Summary	41
5	Mastermind: Modeling Crime Patterns	42
5.1	Background	42
5.1.1	Computational Criminology	42
5.1.2	Related Work	45
5.1.3	Navigation	46
5.1.4	Learning and Memory	48
5.2	Project Overview	50
5.3	Mastermind Formal Model	52
5.3.1	Entity Classification: Linking Agent-Based Systems to ASM	53

5.3.2	Agent Architecture	54
5.3.3	Urban Landscape Model	56
5.3.4	Person Agent	59
5.3.5	Navigation	61
5.3.6	Target Selection	68
5.3.7	Experiments	70
5.4	Lessons Learned	75
5.5	Summary and Future Work	78
5.5.1	Genius: A Decision Support System for Counterterrorism Planning and Response	79
5.5.2	Modeling Physical Activity and Chronic Diseases in Urban Environments	79
6	Identity Management Architecture	81
6.1	Background	83
6.1.1	Basic Definitions	83
6.1.2	Identity (Entity) Resolution	84
6.1.3	Identity Management Systems	85
6.1.4	Identity Theft and Identity Fraud	87
6.1.5	Privacy and Trust	88
6.1.6	Advanced Research on Identity Management	89
6.2	The Formal Model	90
6.2.1	Basics	90
6.2.2	Mapping Partial Identities to Identities	93
6.2.3	Evolution of Partial Identities	95
6.3	Applications	97
6.3.1	Identity Theft	98
6.3.2	Other Applications	102
6.4	Summary and Future Work	103
7	Conclusions	104
7.1	Summary of Contributions	104
7.2	Future Research Directions	107

Appendices	110
A Dijkstra's Shortest Path Algorithm	110
B Identity Management Glossary	112
Reference List	115

List of Tables

5.1	Entity classification and taxonomy through different layers	53
5.2	Mean crime density calculated from simulations on 8×8 and 16×16 grids . . .	72
5.3	Average number of crimes calculated from simulations on a 12×12 grid using basic target selection	73
5.4	Average number of crimes calculated from simulations on a 12×12 grid con- sidering familiarity in target selection	73
5.5	Coverage of activity space and crime occurrence space for different mobility styles	74
5.6	Analyzing the similarity of crime occurrence spaces created by different mo- bility styles	75

List of Figures

2.1	A partially ordered set of moves for two agents where arrows represent the ordering	12
4.1	Different phases in modeling complex social systems	33
4.2	Model construction: The phase transition from the conceptual model to the mathematical model	34
4.3	A sample CoreASM program in Eclipse: CoreASM offers minimal encoding and maintains executability of abstract models	37
4.4	Mastermind visualization plugin complements model execution with familiar visualizations for domain experts	39
4.5	CSDe: A Control State Diagram editor plugin for the Eclipse development environment	39
5.1	Mastermind system architecture	51
5.2	Java & CoreASM implementations of Mastermind	52
5.3	The architecture of a person agent	55
5.4	Geographic environment	58
5.5	Control state diagram of the program of a DASM person (offender) agent . .	59
5.6	Control state diagram of the Space Evolution Module (SEM)	62
5.7	Control state diagram of the Target Selection Module (TSM)	69
5.8	Deterministic vs. tear-drop navigation on a 16×16 grid	71
6.1	Major contexts in defining identity and identity management	86
6.2	Mapping partial identities to identities within the same context	93
6.3	Mapping partial identities to identities across contexts	95
6.4	Conceptual process model of identity theft domain	98

List of Acronyms

ABSS	Agent-Based Social Simulation
ADM	Agent Decision Module
ASM	Abstract State Machine
BDI	Belief-Desire-Intention
BPEL or BPEL4WS	Business Process Execution Language for Web Services
CA	Cellular Automata
CBR	Case-Based Reasoning
CSD	Control State Diagram
CSP	Communicating Sequential Processes
DASM	Distributed Abstract State Machine
GIS	Geographic Information System
ICURS	Institute for Canadian Urban Research Studies
IMS	Identity Management Systems
IMA	Identity Management Architecture
KIF	Knowledge Interchange Format
MABS	Multi-Agent Based Simulation
MAS	Multi-Agent Systems
PA	Physical Activity
R & D	Research & Development
SDL	Specification and Description Language
SEM	Space Evolution Module
STL	Software Technology Lab
TSM	Target Selection Module

Chapter 1

Introduction

We are living in a period of intellectual revolution that is powered to a great extent by advances in information technology. Computational methods play a key role in advancing the boundaries of a wide range of research disciplines in science and engineering. In many research disciplines, such as biology or chemistry, the use of computational techniques has become well-established and valuable in advancing the boundaries of those fields. Mathematical and computational models are known to facilitate developing a better understanding of complex systems of interacting elements by providing precision and rigor in defining such systems and enforcing a focus on their essential aspects [17]. Beyond seeing computers as tools, *computational thinking* [196] has become a way of solving problems and designing systems in a variety of disciplines in the physical and social sciences.

More specifically for the social sciences, applying computational techniques helps in overcoming some of the core limitations of studying social phenomena. Social scientists have always been limited by the inextricability of the subject of their research from its environment. Thus, it is difficult to study different factors influencing a phenomenon in isolation. Systematic study of such systems using computational models facilitates dealing with their inherent high complexity and high dynamics. Computer models of social systems simulate dynamic aspects of individual behavior to study characteristic properties and dominant behavioral patterns of societies as a basis for reasoning about real-world phenomena. Using computer simulations, researchers can perform experiments that are difficult, if not impossible, in real life. Furthermore, computational models can be used as decision support tools for policy makers and practitioners in order to explore and analyze different ‘what-if’ scenarios.

1.1 Modeling Framework for Social Systems

Arguably, the nature of modeling a system as complex and diverse as a social system is a highly iterative and potentially open-ended process. Just as the rapid increases in computer power and complexity of problems led to the *software crisis* of the 60s and 70s [69], the lack of structure and rigor in computational modeling of such complex systems could become detrimental. This calls for existing software development techniques to be adapted or new ones to be created to address the specific needs of such interdisciplinary settings. Our approach builds on the successful application of agile formalization techniques [95] in different stages of design and development of computer-based systems, and the importance of the precision and rigor offered by mathematical models in establishing a clear and consistent understanding of a complex system. Building on our experience in applying the abstract state machine (ASM) method [30, 22] to the study of key aspects of Web services architectures [186, 86], here we consider a much broader context—one outside the usual comfort zone of computer science.

Our goal is to develop a framework that facilitates a smooth transition between different phases of the iterative process of modeling and simulation of complex systems in interdisciplinary settings. Such a framework brings together three important aspects of the iterative process of model building. First, the use of formalism and the process of formal modeling enforces *structure* and *logical thinking*, as well as *precision* and *rigor* in critical analysis of even the basic assumptions about the system. Second, utilizing the freedom of abstraction in the ASM method, we aim at building *simple* models that focus on key characteristics of the systems with details and complexity added in a systematic fashion through proper refinements. Third, the framework explicitly accounts for different steps of *validation* through analytical and mathematical reasoning, and also facilitates rapid prototyping.

Building a computational model of a complex social system requires overcoming two major obstacles: (1) capturing the complexity of the domain in a systematic fashion while ensuring the computational model reflects the understanding of domain experts, and (2) transforming real-life events, which are not usually thought of in a discrete, mathematical manner, into a mathematical model. To address these issues, we distinguish three essential phases in the process of modeling behavioral aspects of complex social systems, namely conceptual modeling, mathematical modeling, and computational modeling, with

several critical phase transitions and validation feedback loops. Furthermore, in an interdisciplinary research context, we recognize the need for clear and effective collaboration among multidisciplinary team members and the importance of developing a common conceptual view in order to succeed in the underlying goal of scientific discovery.¹ We propose a novel methodological framework that addresses these practical needs by offering a systematic development method and supporting tools for interactive design, rapid prototyping and computer simulation.

1.2 Case Studies

The proposed framework has been developed and tested in the context of two different case studies, namely an interdisciplinary R&D project in Computational Criminology, called *Mastermind* [45, 46, 47, 48, 113], and a project on Identity Management, called Identity Management Architecture [114, 115].

The Mastermind project, jointly managed by the Institute for Canadian Urban Research Studies (ICURS) and the Software Technology Lab at Simon Fraser University, is a step forward in novel research directions in Criminology that aims at applying computational methods to overcome limitations of conventional statistical methods [42, 144]. The goal is to develop computational models of criminal activity patterns in urban environments, with a special focus on spatiotemporal characteristics of crime, potentially involving multiple offenders and multiple targets. Mastermind builds on top of widely-used approaches to modeling human behavior and societies, and adopts an agent-based view for modeling criminal behavior. Besides training and experiments, we aim at developing intelligent decision support systems and advanced analysis tools for systematic reasoning about possible scenarios. Such tools will not only be used by criminology researchers to analyze crime patterns, but also can be used by modern policing agencies and city planners to assist with prediction of criminal activities and effective urban planning by taking the geography of crime into account. Developing a robust and scalable model also enables us to apply Mastermind in other domains such as counterterrorism (e.g., developing real-time decision support systems for emergency response).

¹The words multidisciplinary and interdisciplinary are often used interchangeably to refer to a field of study that crosses traditional boundaries between academic disciplines. In this thesis, we use the word multidisciplinary to emphasize the existence of different views from different disciplines that are not necessarily integrated.

The Identity Management Architecture (IMA) project focuses on developing a firm unifying semantic foundation for a systematic study of identity management and improved accuracy in reasoning about key properties in identity management system design. Like many complex systems of today, identity management systems are influenced by a combination of social and technical aspects. They play a crucial role in many application contexts, including e-government, e-commerce, business intelligence, crime investigation, and homeland security. The variety of approaches to and techniques for identity management, while they address some of the challenges, have introduced new problems especially concerning interoperability and privacy. As such, any attempt to consolidate such diverse multidisciplinary views and approaches to identity management in a systematic fashion requires a precise and rigorous unifying semantic framework. Our proposed framework is built upon essential concepts of identity management and serves as a starting point for bringing together different approaches in a coherent and consistent manner.

Furthermore, we have applied the proposed modeling approach and supporting tools in other application domains, including computational modeling and formal analysis of public safety and security regulations, specifically focusing on civil aviation security [111, 112].

1.3 Significance of Our Research

The work presented in this thesis aims at

- Developing an integrated methodological framework and tool environment for design, construction, validation, development and simulation of models of complex social systems. The framework
 - is based on universal mathematical notions, common abstraction principles, and integrated modeling and design methodologies of the ASM method. This application of ASMs to social systems is original and unprecedented.
 - specifically addresses the issues of building and validating a model, which are often neglected in widely-used agent-based approaches to modeling social systems.
- Illustrating consistency, applicability and scalability of the proposed framework through novel applications in two, inherently different, interdisciplinary domains:
 1. Computational Criminology: The Mastermind Project

- (a) Mastermind model of criminal activity in urban environments is developed in close collaboration with criminologists and shows the robustness of our approach in dealing with challenges of truly interdisciplinary research.
 - (b) Mastermind combines the ASM method with the established view of agent-based modeling of social systems and provides a precise semantic foundation—something multi-agent system modeling is lacking.
 - (c) Mastermind delivers the following results:
 - The main theoretical result is the abstract behavior model of *person* agents interacting with their objective and subjective environments (formally defined as the *geographic environment*).
 - The main practical result is the Mastermind system architecture which serves as a platform for the construction and experimental validation of discrete event simulation models.
 - The simulations, done at different levels of abstraction, illustrate the advantages and huge potential of applying computational modeling as a decision support tool for researchers (e.g., to test theories) and policy makers (e.g., to explore different scenarios and intervention strategies).
 - (d) Mastermind is also scalable.
 - It is not only applicable to a broad range of crimes but also to other environments such as airports, ports, shopping centers, and subway stations.
 - It offers a reliable platform for systematic expansion and further application in other domains (e.g., counterterrorism, chronic disease modeling).
2. Identity Management Systems: The Identity Management Architecture (IMA) Project
- (a) The IMA project highlights the potentials of our approach in dealing with challenges of socio-technical systems, e.g., Identity Management Systems (IMS).
 - (b) It provides a novel semantic framework for
 - unifying a bewildering list of notions in identity management.
 - analyzing and reasoning about basic concepts and key properties of IMS.
 - (c) It also exemplifies the practicality and value of such a formal framework: the model is applied to study semantic aspects of identity theft, clarifying the basic underlying definitions.

1.4 Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 describes the background and motivation behind this work. Chapter 3 provides an overview of the literature on computational modeling and simulation of complex social systems. Chapter 4 presents our proposed modeling framework and supporting tools. Chapter 5 describes in detail the key elements of the Mastermind project on computational modeling of criminal activity in urban environments. Chapter 6 provides an extensive review of cutting-edge identity management literature and presents our work on Identity Management Architecture. Chapter 7 concludes the thesis and outlines the ongoing and future directions of the research.

Chapter 2

Background and Motivation

For years, the benefits and shortcomings of using formal techniques in the software development process have been discussed by researchers of the Software Engineering field [120, 34, 136, 15]. In certain areas, the use of formal methods is now well established, including safety-critical systems [33], the requirements specification phase of software development [74], and semantic modeling of protocols, languages, and architectures for sequential, parallel, or distributed systems.

In recent years, the value of using *lightweight* formal methods in different phases of software development has been widely recognized in the literature [75, 87, 103]. Various lightweight formal methods, which emphasize partial formalization and focused application [132], exist that address the pragmatic needs of software development. Well-known lightweight formal approaches include the Alloy object modeling notation [131] (which is based on Z [180]), B [5], and *abstract state machines* (ASMs) [30]. Being state-based and machine-based, these languages share a common conceptual foundation. They are widely used in both academia and industry for the design and analysis of hardware and software systems [24].

Abstract state machines are known for their versatility in computational and semantic modeling of algorithms, architectures, languages, protocols and virtually all kinds of sequential, parallel and distributed systems [30]. We build on the success of our previous work, on semantic modeling and validation of Web services architectures using ASMs [86, 186], and aim at applying formal modeling techniques in a much broader application context: complex social systems. The precision and rigor offered by mathematical models helps establish a clear and consistent understanding of a complex system by revealing potential

inconsistencies, vagueness, or loose ends in the assumptions made about the system under study. A mathematical model can be formally analyzed either manually or using automated tools such as model checkers. Furthermore, mathematical models can be made executable for analyzing the system through simulation.

In this chapter, we first briefly outline the successful application of ASMs in our previous work on Web services, and then review the fundamental concepts of the ASM method. The ASM paradigm is known for its versatility in modeling a wide array of systems, and its flexibility in blending different specification styles. Furthermore, its simple language and operational character facilitate communication with domain experts and rapid prototyping, which are key in modeling modeling complex social systems. In the next chapter, we focus on modeling social systems and review some the most-widely used modeling approaches in the literature.

2.1 Formal Modeling Using Abstract State Machines: A Case Study

Our previous work on formalization of the Business Process Execution Language for Web Services (BPEL4WS, or BPEL for short) [8] is an example of formal semantic modeling of languages and architectures for distributed systems. BPEL is an XML based specification language for automated business processes, proposed by OASIS (WSBPEL-TC 2004) as an industrial standard for the e-business world. We developed an abstract operational model of the language based on the ASM [30] formalism. That is, we abstractly modeled dynamic properties of the key language constructs through the construction of a *BPEL abstract machine*. The goal of our work was to provide a precise and well defined semantic framework for establishing the key language attributes. To this end, the BPEL abstract machine forms a comprehensive and robust formalization closely reflecting the view of the informal language definition. As a result of building the ASM model, we discovered a number of deficiencies in the language definition [86]. A comprehensive list of deficiencies is provided in [81, 186].

Beyond reasoning about the language design and checking consistency and validity of semantic properties, our BPEL abstract machine also served as a platform for experimental validation through simulation and testing [186]. Experimental validation of high-level design specifications facilitates design exploration and helps eliminate deficiencies prior to

coding. BPEL is going through different phases of standardization at OASIS, and the dynamic nature of standardization, being an ongoing and potentially open-ended activity, calls for flexibility and robustness of the formalization approach. To this end, we believe that the ASM formalism and abstraction principles offer a sensible compromise between mathematical elegance and practical relevance—already proven useful for practical purposes in other standardization contexts [108].

2.2 Abstract State Machine Method

The *abstract state machine* (ASM) modeling paradigm [30] offers a universal mathematical framework for computational modeling of system requirements in abstract functional and operational terms. The ASM design and analysis method offers a unified “*precise yet simple conceptual framework*” that systematically integrates major software lifecycle activities with principal techniques for modeling and analysis of complex systems [22].

Declarative, functional and operational description styles can be blended as desired in order to model dynamic system properties at a level of abstraction that is considered most appropriate, i.e., providing the right degree of detail and precision. By viewing the behavior of a system under study as the evolution of abstract states, computational behavior is represented by the executions, or *runs*, of an ASM. The underlying abstraction principles also directly support concurrent and reactive behavior as well as timing aspects.

Abstract state machines have been used for modeling various kinds of sequential, parallel and distributed systems [30]. Applications of ASMs have been studied extensively by researchers in academia and industry with the intention to bridge the gap between formal and empirical approaches [84]. Widely recognized applications include semantic foundations of popular system design languages, like SDL¹ [108, 79, 130], VHDL [27, 26] and SystemC [153]; programming languages, like Prolog [18, 19], JAVA [183, 29] and C# [25]; embedded control systems [28]; wireless networks [109]; communication architectures [110]; and Web services [86]. For a comprehensive list of references to ASM applications, we refer the reader to the ASM Research Center at www.asmcntr.org.

The ASM method is based on three fundamental concepts: (1) *abstract state machines* provide the underlying rigorous mathematical foundation, (2) *ASM ground models* provide

¹The ASM model of SDL is part of ITU’s SDL standard [130]

precise abstract blueprints of the system formulated in domain-specific terms, and (3) the *ASM refinement method* allows for linking different stages of design and development in a coherent and consistent way [30, 22]. Abstract state machine specifications can be directly executed using several available execution engines. This facilitates using simulations to validate ground models against the requirements. Furthermore, within the ASM conceptual framework, ground models are seamlessly linked to the executable code via ASM refinements. Refinements provide systematic documentation of linking abstract ground models to executable code and contain explicit descriptions of the software structure and of the major design decisions [22]. In the following, we review these concepts in more detail.

2.3 Distributed Abstract State Machines

The asynchronous computation model of *distributed abstract state machine* (DASM)

defines concurrent and reactive behavior as observable in distributed computations performed by autonomously operating computational agents, in terms of *partially ordered runs*.

A DASM M is defined over a given vocabulary V by its program P_M and a non-empty set I_M of initial states. V consists of a finite collection of symbols denoting mathematical objects and their relation in the formal representation of M , where we distinguish *domain symbols*, *function symbols* and *predicate symbols*. Symbols that have a fixed interpretation regardless of the state of M are called *static*; those that may have different interpretations in different states of M are called *dynamic*. A state S of M results from a valid interpretation of all the symbols in V and constitutes a variant of a first-order structure. Intuitively, states can be viewed as a variant of partial *many-sorted structures*, one in which relations are formally represented as Boolean-valued functions.

Concurrent control threads in an execution of P_M are modeled by a dynamic set AGENT of computational *agents*. This set may change dynamically over runs of M , as required, to deal with varying computational resources. Agents of M interact with one another, and possibly also with the operational environment of M , by reading and writing shared locations of a global machine state. The underlying semantic model regulates such interactions so that potential conflicts are resolved according to the definition of partially ordered runs.

P_M consists of a statically defined collection of agent programs P_{M_1}, \dots, P_{M_k} , $k \geq 1$, each of which defines the behavior of a certain *type* of agent in terms of state transition rules.

The canonical rule consists of a basic update instruction of the form

$$f(t_1, t_2, \dots, t_n) := t_0,$$

where f is an n -ary dynamic function symbol and the t_i 's ($0 \leq i \leq n$) are terms. Intuitively, a dynamic function can be seen as a *function table* where each row associates a sequence of argument values with a function value. An update instruction specifies a pointwise function update, i.e., an operation that replaces an existing function value by a new value to be associated with the given function arguments. Complex rules are inductively defined by a number of simple rule constructors allowing the composition of rules in various ways.

A computation of an individual agent of M , executing program P_{M_j} , is modeled by a finite or infinite sequence of state transitions of the form

$$S_0 \xrightarrow{\Delta_{S_0}(P_{M_j})} S_1 \xrightarrow{\Delta_{S_1}(P_{M_j})} S_2 \xrightarrow{\Delta_{S_2}(P_{M_j})} \dots,$$

such that S_{i+1} is obtained from S_i , for $i \geq 0$, by firing $\Delta_{S_i}(P_{M_j})$ on S_i , where $\Delta_{S_i}(P_{M_j})$ denotes a finite set of updates computed by evaluating P_{M_j} over S_i . Firing an update set means that all the updates in this set are fired simultaneously in one atomic step. The result of firing an update set is defined if and only if the set does not contain conflicting updates (attempting to assign different values to the same location).

2.3.1 Partially Ordered Runs

A DASM M performs a computation step whenever one of its agents performs a computation step. In general, one or more agents may participate in the same computation step of M . A single computation step of an individual agent is called a *move*. In this model, moves are atomic. Naturally, conflicting moves must be ordered so that they do not occur in the same step of M .

A partially ordered run ρ of M is given by a triple (Λ, A, σ) satisfying the following four conditions (adopted from [118, Section 6.5]):

1. Λ is a partially ordered set of moves, where each move has only finitely many predecessors. Figure 2.1 presents one such partially ordered set of moves where each m_i represents a move.
2. A is a function on Λ associating agents to moves such that the moves of any single agent of M are linearly ordered. In Figure 2.1, m_1, m_2, m_4 , and m_6 belong to agent a_1 while m_3 and m_5 belong to agent a_2 .

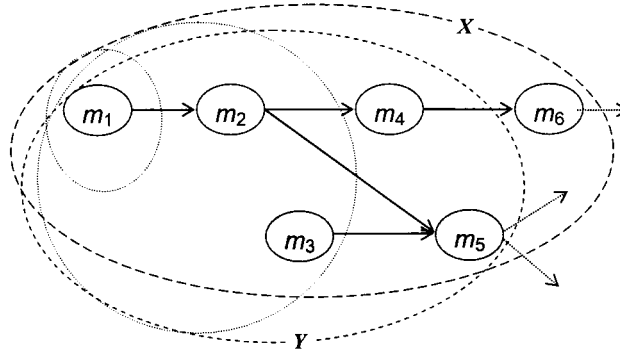


Figure 2.1: A partially ordered set of moves for two agents where arrows represent the ordering

3. σ assigns a state of M to each initial segment X of Λ , where $\sigma(X)$ is the result of performing all moves in X . An initial segment of Λ is a substructure Y of Λ such that if $y \in Y$ and $x < y$ in Λ then $x \in Y$. In Figure 2.1, dashed circles specify initial segments of Λ .
4. *Coherence condition:* If x is a maximal element in a finite initial segment X of Λ and $Y = X - \{x\}$, then $A(x)$ is an agent in $\sigma(Y)$ and $\sigma(X)$ is obtained from $\sigma(Y)$ by firing $A(x)$ at $\sigma(Y)$. In Figure 3-1, m_6 is the maximal element of X and $Y = X - \{m_6\}$.

A partially ordered run defines a class of admissible runs of M rather than a particular run. In general, it may require more than one (even infinitely many) partially ordered runs to capture all admissible runs of M . From the coherence condition it follows that all *linearizations* of the same finite initial segment of a run of M have the same final state.

2.3.2 Reactivity and Time

A DASM M models interactions with a given operational environment, the part of the external world with which M interacts, through actions and events as observable at external interfaces, formally represented by externally controlled functions. Of particular interest are *monitored functions*. Such functions change their values dynamically over runs of M , although they cannot be updated internally by agents of M . A typical example is the abstract representation of global system time.

In a given state S of M , the global time (as measured by some external clock) is given by a monitored nullary function *now*, taking values in a linearly ordered domain TIME. Values of *now* increase monotonically over runs of M . Additionally, ∞ represents a distinguished value of TIME, such that $t < \infty$ for all $t \in \text{TIME} \setminus \{\infty\}$. Finite time intervals are given as elements of a linearly ordered domain DURATION.

2.4 ASM Ground Models

A ground model is a *system blueprint* that captures the key functional requirements of a system in a precise and reliable form, genuinely reflecting an intuitive understanding of the system under study. The role of a ground model is to “ground the design in reality” by providing an *understandable* and *checkable* characterization of the system that is inspectable by both domain experts and system designers [20].

As discussed in [30], the concept of a ground model is inevitably present in every system design, but often not in an explicit form. The ASM ground model technique makes this concept explicit by addressing the most important properties of ground models, namely *understandability* and *checkability*.

Abstract state machines build on universal mathematical notions that “accurately support the way domain experts use high-level process-oriented descriptions and software practitioners use pseudo-code” [20, p. 151]. Furthermore, ASM models allow calibrating the degree of precision in order to focus on key domain concepts. As such, ASM ground models directly address the *communication problem* among the stakeholders by providing *understandable* blueprints of the system [20].

Checkability implies applying reasoning and experimentation to establish completeness and correctness of a ground model; i.e., to ensure that it conveys the intentions of the domain experts. Since ASM ground models are formulated in application-domain terms, they can be manually inspected by domain experts. The mathematical foundation also facilitates mathematical checking for consistency. Furthermore, the operational character of ASMs (in terms of ASM ‘runs’) allows for performing experiments by simulating the ground models [20, 22]. Simulations are used to validate ground models against the requirements (“user scenarios”). They also provide another means to communicate with domain experts. Several tools exist that facilitate checking of ASM models, including ASM execution engines (e.g., ASM Workbench [66], XASM [10], AsmL [151], and CoreASM [12]), model-checkers

(e.g., [67]), and theorem provers (e.g., [170]).

In addition, given the mathematical nature of ASMs, the ASM ground model method provides a way to build *consistent* and *unambiguous, simple and concise, abstract and complete* blueprints of a system [20].

2.5 ASM Refinement Method

Refinement is a general methodological principle which is present wherever a complex system or problem is described piecemeal, decomposing it into constituent parts which are detailed in steps to become manageable. [21]

The ASM refinement method provides a framework to systematically link models at successive stages of system development cycle, supporting a coherent system view through different levels of abstraction. The abstract blueprint of the system (ground model) can be linked to the executable code through a chain of stepwise refinements that reflects design decision at each stage.

Abstract state machine refinements are not necessarily syntax-directed; i.e., e.g., the syntax may change from one stage to the next. Furthermore, successive refinements provide a systematic documentation of the design decisions and code development which facilitates capturing changes in the requirements. Clear separation of different levels of abstractions allows for localizing the “right” level of abstraction at which the changes have to be performed, and systematically transferring the changes to the lower levels. The notion of *conservative refinement*, where changes are captured in a purely incremental fashion, further simplifies validation, verification, and reuse of system components [23].

The ASM refinement scheme is mathematically defined as follows. In order to refine an ASM M to an ASM M^* , the designer has the freedom to define the following major constructs [21]:

- a notion of *refined state*,
- a notion of *states of interest* and the *correspondence* between states of interest in M and M^* through the refinement,
- a notion of *abstract computation segments* in M (τ_1, \dots, τ_m , where τ_i is a single M -step) and corresponding refined computation segments in M^* ($\theta_1, \dots, \theta_n$, where θ_i is a single M^* -step), which in given runs lead from

corresponding states of interest to (usually the next) corresponding states of interest (the resulting refinements are called (m, n) -refinements),

- a notion of *locations of interest* in M and of *corresponding locations* in M^* ,
- a notion of *equivalence* (denoted by the symbol ' \equiv ') of the data in the locations of interest; these local data equivalences usually accumulate to a notion of equivalence of corresponding states of interest.

Once the notions of corresponding states and their equivalence are fixed, one can define that M^* is a *correct refinement* of M if and only if every (potentially infinite) refined run (in M^*) simulates an (potentially infinite) abstract run with equivalent corresponding states (in M).

The flexibility in defining states of interest, their correspondence, and also the notion of data equivalence, offers the ability to *change the signature* (or syntax) in a refinement. Furthermore, *changes in the control* are achieved using flexible definition of corresponding computations. As such, the ASM refinement method naturally “integrates declarative and operational techniques and classical modularization concepts” [23]. They support architectural and component design, reflecting modular techniques for accurately crossing system design levels. They provide a systematic and controllable transformation of ground models to code together with precise documentation that can be used for inspection, reuse and maintenance [22, 23].

2.6 Summary

The ASM paradigm is known for its wide use in both academia and industry and its versatility in design and analysis of virtually any type of sequential, parallel, and distributed computer systems. Building on this body of work, we contend that the ASM paradigm can be further applied for modeling systems beyond the conventional context of hardware and software systems. Several features of the ASM method, including its pseudo-code style syntax, its precise semantics and its operational character, can be effectively utilized in modeling complex social systems. In the next chapter, we provide a review of some the most-widely used approaches in the literature of social systems modeling and explore the limitations of existing approaches.

Chapter 3

Modeling of Complex Social Systems

The main focus of this work is on computational modeling and simulation of complex social systems using agent based methods. In the following we briefly introduce key concepts in complex systems modeling, mainly focusing on modeling social systems. Specifically, we study multi-agent systems as one of the most popular approaches currently used in modeling and simulation of complex social systems.

3.1 Complex Systems Modeling

There is no precise definition of complex systems. However, there is a general agreement on the properties of a complex system [17]: a complex system is a system comprising a number of interacting elements whose individual actions and interactions lead to emergent global dynamics [17, 165]. This definition applies to a wide range of systems in different scientific disciplines from biology and ecology to physics and economics.

Mathematical models have been widely used in studying such complex systems. They facilitate studying a system by providing a “representation of the essential aspects” [11] and precisely identifying the relevant questions about the system [17]. A variety of mathematical approaches, such as differential equations, statistical models, game theory, cellular automata, and agent-based systems, have been used to describe complex systems [17, 165]. Regardless of the particular approach, there are fundamental principles that are crucial in

modeling and simulation of complex systems.

One key issue is the *complexity* of models. Although systems under study are inherently complex, it is strongly argued that models of complex systems must be kept as simple as possible [17, 168]. This, in turn, requires finding the right level of abstraction and detail that is required to address modeling objectives [145].

Another important issue to address is *validation*. Having a clear and structured validation process, in order to ensure the model accurately captures the system under study, is essential. Specifically for complex social systems, finding the right validation approach has proven very challenging, as it largely depends on the purpose of the model and the level of abstraction [145, 104]. For instance, although it is desirable to compare the output of a model with empirical data, for many social systems such comparisons are not easy to carry out and do not lead to a clear answer. Instead, abstract models of social systems can be used as part of the process of developing theories. As such, they are subject to the criteria normally applied in evaluating theories. That is, models should be based on plausible micro-level behaviors and should produce expected macro level patterns that are interpretable by domain experts. [104]. In order to guard against alternative explanations, the fit between the model and the theory can also be evaluated through *sensitivity analysis*; i.e., evaluating the changes in the macro-level behavior when parameters of the model are systematically changed. The goal is to ensure that any changes in the behavior can be reasonably interpreted by domain experts [104].

The literature on modeling complex social systems has repeatedly discouraged the use of models for prediction, and instead has emphasized the advantages of modeling in developing a better understanding of a complex system [145, 107, 182]. The goal is not to provide a “silver bullet” or a single final answer to a question; rather, computational models offer new ways of analyzing a system through running different “what-if” scenarios, studying the dependencies between different variables, and running experiments in a trial and error fashion [145, 104]. On the other hand, the structure that is imposed by the mathematical modeling process helps clarifying basic assumptions and problems in the understanding of a system [145]. Gilbert [104] emphasizes the role of models in specifying a *research question* and compares the modeling process to stripping away the layers of an onion, from a “general area of investigation, through a particular topic, to a question that could be answered”. Hence, it can be argued that one of the biggest advantages of modeling complex systems is offering a process for identifying those key building blocks of the system whose interaction

leads to the overall complex behavior [165].

3.2 Social Simulation

Using computer simulations to conduct virtual experiments or to analyze “what-if” scenarios is now commonly practiced in social sciences. *Social simulations* and modeling of social systems serve many advantages. They allow capturing intricate dynamics of social systems, developing interdisciplinary perspectives on those systems, and facilitate reasoning about them [105]. In addition, they provide a sandbox for testing different hypotheses, integrating different theories and building new ones [72]. Social simulation models also provide ways of dealing with complicated interdependencies among a large number of components and bring together multidisciplinary perspectives to analyze policy problems. As such, reasoning about the target system is facilitated by analyzing different scenarios (simulation runs) before making irreversible policy decisions [107, 105].

The history of social simulation starts with differential equations and goes through stochastic processes, game theory, cellular automata and, finally, distributed artificial intelligence and multi-agent systems [106]. The *agent based modeling* paradigm, which is popular for describing self-organizing systems and processes that lack central coordination, has become the focus of latest developments in computational modeling and simulation of social phenomena [13, 107, 152].

Why Agent-Based Modeling?

For many years, cellular automata [171] have been used for modeling social systems [107]. A cellular automaton (CA) is formed by a collection of cells on a grid. Each cell has a finite number of *states*, usually represented by different colors. The state of a cell evolves in discrete time steps according to a set of rules which define the new state based on the cell’s current state and the state of its *neighbors*. Cellular automata can operate on different types of grids, e.g., a one-dimensional line or a d -dimensional lattice. Similarly, it is possible to specify the neighborhood in different ways, e.g., a square neighborhood or a diamond-shaped one [171].

Agent-based models can be seen as a natural progression of CA models. *Agents*, like cells, operate in an *environment* and there are rules governing their interactions. However, the behavior of an agent is usually much more complex than that of a cell. Agents are

autonomous self-controlled units that define their own behavior based on perceptions of the environment, including other agents. Rules governing agents' behavior range from simple conditional statements to sophisticated machine learning algorithms [89]. The bottom-up nature of agent-based system and the fact that each single agent controls its own behavior make agent-based modeling a powerful tool for analyzing different "what-if" scenarios in complex social systems [78]. Agent-based modeling opens new ways of understanding a complex system through linking the micro-level behavior of individual agents to intricate macro-level dynamics of the system as a whole. This feature gives agent-based modeling and simulation an edge over more traditional modeling approaches [105, 104].

The literature on agent-based modeling and social simulation is, unfortunately, overloaded with terminology and, in some cases, redundant concepts. In order to better characterize different areas of research that closely relate to our work, we use the classification provided by Davidsson in [63]. Research on computational aspects of agent-based modeling is broadly classified under the umbrella of Multi-Agent Systems (MAS). The application of this paradigm in building simulation models of different sorts is studied under the sub-field of Multi-Agent Based Simulation (MABS). The specific application of MABS for simulating social systems is identified as Agent-Based Social Simulation (ABSS). We use this classification to organize the review of key concepts in agent-based modeling of social systems.

3.3 Multi-Agent Systems (MAS)

A Multi-Agent System (MAS) [184, 198, 89] is a collection of autonomous agents that interact with each other, and also with other objects that exist in a given environment. In [89], Ferber identifies the followings basic elements of a MAS: an *environment*, a set of passive *objects* situated in the environment, a set of active *agents* which can manipulate the objects through *operations*, an assembly of *relations* among agents and objects, and the *laws of the universe*. Normally, there is no global control in a MAS; data is decentralized and computation is asynchronous. Moreover, a single agent has a limited viewpoint; i.e., it does not have complete information or capabilities to solve the problem on its own [184].

Multi-agent systems have gained widespread popularity as a modeling paradigm for distributed systems, open dynamic environments, and systems that can naturally be regarded as a society of interacting agents [105]. However, despite being widely used in different

application contexts, there is still a bewildering list of notions that are not consistently defined within the paradigm. For instance, there is no universally accepted definition of agents [97]. Likewise, the distinction between the notions of *agency* (an agent) and *autonomy* (an autonomous/intelligent agent) is not clearly defined [146].

In the following, we review some of the key aspects of multi-agent systems.

3.3.1 Agency and Autonomy

The literature on agent based systems offers a variety of definitions for the notion of agent. Russel and Norvig even argue against a clear-cut definition of an agent and point out that “the notion of agent is meant to be a tool for analyzing systems, not an absolute characterization that divides the world into agents and non-agents” [167]. Here, we outline some of the well-known definitions of the field. Later, we also discuss the need for formal approaches to agent-based modeling.

- Wooldridge [197] defines an agent as “a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives”. This definition of agent reads as a hardware or (more usually) software-based computer system with the following properties:
 - *autonomy*: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
 - *social ability*: agents interact with other agents (and possibly humans) via some kind of agent-communication language.
 - *reactivity*: agents perceive their environment, which may be the physical world, a user via a graphical user interface, or a collection of other agents, and respond in a timely fashion to changes that occur in it.
 - *pro-activeness*: agents do not simply act in response to their environment; they are able to exhibit goal-directed behavior by taking the initiative.
- Ferber [89] describes an agent as a “physical or virtual entity which is capable of acting in an environment, which can communicate, which is driven by a set of tendencies, which possesses resources of its own, which is capable of perceiving its environment, which has only a partial representation of this environment, which possesses skill, . . .”.

- Franklin and Graesser [97] describe the *essence of agency* and offer the following, in an attempt to provide a more formal definition: “An autonomous agent is a system situated within a part of an environment that senses environment, and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future.”

Based on the key aspects addressed in the definitions outlined above, agents can be classified in various categories. For instance, according to the type of the environment they operate in [197] (e.g., static vs. dynamic, or discrete vs. continuous), according to the individual agent reasoning mechanism (e.g., cognitive, reactive, or hybrid) [184, 89, 198], or according to their architecture (e.g., layered, reactive, belief-desire-intention, etc.) [198]. In the next sections we explore some of these aspects in more detail.

3.3.2 Architecture

An agent architecture dictates how the functional complexity of the agent is organized. In [197], four well-known types of agent architecture are described, namely (1) Logic based, (2) Belief-Desire-Intention (BDI), (3) Reactive, and (4) Layered.

Logic based architectures follow the traditional approach to building artificial intelligent systems, known as symbolic AI [124]. It suggests that intelligent behavior can be generated in a system by providing a symbolic representation of the environment, identifying the desired behavior, and syntactically manipulating the symbolic representation. Symbolic representations are logical formulae, and the syntactic manipulation corresponds to logical deduction, or theorem proving; the agents can thus be seen as theorem provers. This approach offers the great advantage of having a simple, elegant logical semantics for the system. However, logic-based approaches also have some disadvantages. In particular, the inherent computational complexity of theorem proving leads to efficiency problems especially in time-constrained environments. Moreover, there are still many unresolved problems regarding representation of complex, dynamic, physical environments and reasoning about them [197].

The BDI architecture has its roots in the philosophical understanding of human practical reasoning developed by Bratman [49]. Practical reasoning is defined in terms of deciding what goals we want to achieve (*deliberation*), and how we are going to achieve those goals (*means-ends reasoning*). Therefore, the basic components of a BDI architecture are data

structures representing *beliefs*, *desires*, and *intentions* of an agent; *beliefs* about the environment give rise to *desires* which in turn give rise to *intentions*, and these intentions play a central role in determining the actions by way of *deliberation* and *reasoning*. The strength of this architecture is in being intuitive, and in providing a clear functional decomposition, which indicates what sorts of subsystems might be required to build an agent [102]. The main difficulty, however, is properly linking the design with an efficient implementation [197]. Due to its very nature, the BDI architecture is most widely used in modeling human decision-making behavior [102].

Reactive architectures are specifically designed for agents who only react to their environment and do not reason about it. Although there are several advantages to reactive approaches, including simplicity, traceability, and robustness, the inherent problem with purely reactive architectures is that they are local in nature and have only a short term view; i.e., no learning or adaptation takes place. Hence, we consider them not an appropriate choice for modeling the behavior of individuals.

Layered architectures address the need for capturing both reactive and proactive behaviors of an agent by decomposing the systems into separate subsystems, each dealing with one type of behavior. This idea leads naturally to a class of architectures in which the various subsystems are arranged into a hierarchy of interacting layers. Typically, there are at least two layers to deal with: reactive and proactive behaviors. However, in principle, there is no reason why there should not be many more layers (for instance for capturing social behavior). Layered architectures have gained wide-spread popularity mainly because layering represents a natural decomposition of functionality [184], but such a clear-cut distinction of behavior is not always easy to identify. Therefore, the main problem with layered architectures is the lack of conceptual and semantic clarity in defining each layer. Another issue is that layering creates an overhead due to the complexity of interactions between layers [197].

3.3.3 Society of Agents

A multi-agent system, in essence, forms a *society of interacting agents*. In such a setting, some fundamental concepts such as the organization of agents, their interaction, means of communication, coordination, and planning are introduced and discussed in the MAS literature.

Agent Interaction

Interactions between agents occurs “when two or more agents are brought into a dynamic relationship through a set of reciprocal actions. Interaction is the consequence of the plural aspect of multi-agent societies, bringing in a dimension which goes beyond the individual” [89].

Ferber [89] introduces three types of interactions: *indifference*, *cooperation* and *antagonism*. This classification is made based on agents’ goals, their resources, and their set of skills. For instance, cooperation offers a number of advantages such as accomplishing tasks impossible that cannot be realized by a single agent given its resources and skills.

Organization

In a multi-agent system organization defines *how* the agents interact with each other. In other words, it dictates the type of the agent society; e.g., predefined vs. emergent, or static vs. dynamic.

Sycara [184] defines organization as “a framework for agent interactions through the definition of roles, behavior expectations and authority relations”. *Dynamic organization* of agents and the issue of *adaptivity* (so that the organization can adapt to changes) are identified as two crucial problems. Henceforth, depending on the goal of a MAS and the type of interactions needed between agents, different organizational structures can be used. Examples of such structures include a hierarchical model of authority, or, alternatively, a market model where the agents compete for resources and task [184].

Communication

Communication is the *means* by which agents interact with each other. The basic components of communication are sender-receiver links, mediums of communication, and the intention to communicate. Communication can be point-to-point or broadcast [89].

Wooldridge [198] provides a survey of different agent-based communication languages that have been developed. Examples include the Knowledge Interchange Format (KIF) [101] language which is based on first-order logic and is used to express message content, and KQML [198] and FIPA [90] languages which are both message-based and are used to define a common format for messages sent among agents.

Task Allocation and Collaboration

Task allocation refers to the process of breaking a complex task into sub-tasks (i.e., task breaking) and allocating those tasks to different agents (i.e., task allocation) to be performed concurrently or sequentially and to complete the original task more effectively and efficiently. Task allocation is a topic of active research for which various sophisticated protocols exist in the MAS literature. However, this use of multi-agent systems for improving the performance and efficiency of a system is out of the scope of this work and is not further explored here.

Coordination

In order to have cooperation between agents, certain *supplementary tasks* are required. These tasks aim to satisfy the common goals rather than individual ones [89, 198]. This concept is known as coordination in the MAS literature and is closely related to some other MAS concepts such as planning, conflict resolution, arbitration, and negotiation [184], which are all topics of active research in MAS.

3.4 Formal Approaches to Agent-Based Systems

Although the field of agent-based modeling encompasses decades of research and experience, its success is often hindered due to the lack of a formal platform. d’Inverno et. al [70] contend that there is a dire need for formalism in agent-based systems:

There is a lot of formal theory in the area but it is often not obvious what such theories should represent and what role the theory is intended to play. Theories of agents are often abstract and obtuse and not related to concrete computational models.

While a common terminology exists, the agreement seems, at best, syntactic and the semantics differs considerably from one model to another [73]. The theoretical aspects are often not directly and easily translated to practical concepts, which impedes their applicability. Consequently, a mature methodology that guides the process of specification, verification and implementation cannot be achieved [126].

In [188], Wagner explains the need for formalism in very simple terms:

We should not attempt to define what is an agent in general. This is not necessary [...], as there is no definition of what is a number in mathematics, but only definitions of specific kinds of numbers [...], such as natural or rational numbers. [...] While we can certainly not find a generic definition of the agent, we should look what are the important cases of agent types to be captured by precise mathematical definitions. Such a conceptualization can only be successful if it is based on a sufficiently rich collection of practical experience.

While some work has been done on formal approaches to multi-agent systems, e.g., [92, 146, 126], there is still a need for a robust formal framework that deals with methodological aspects and software engineering techniques. This can be achieved either by constructing new techniques for reasoning about and specifying multi-agent systems or by properly adopting existing formalism.

Such fundamental issues regarding formalism in multi-agent systems were discussed in [70] by a panel consisting of prominent researchers of the field. They categorized the existing formal approaches to multi-agent systems into three groups. Here, we use the same classification and provide brief overviews of some of the more popular approaches under each group.

- *Adopting well-known formal specification languages from traditional software engineering:* An example is the formal agent framework proposed by Luck and Inverno [146], which uses the Z specification language [180] to precisely define common concepts in agency and autonomy. They describe a three-tiered hierarchy composed of *objects*, *agents*, and *autonomous agents*. Objects are entities with *attributes* and *behavior*. Agents are viewed as objects with explicit *goals*, and autonomous agents are agents with *motivations* that, in turn, give rise to *goals*. In [71], the original model is extended by addressing methodological issues of agent systems specification, agent development and agent deployment. The concepts of inter-agent relationships, social behavior and agent plans are also formalized. A criticism of their work relates to the limitations of Z for modeling interactions between agents [91]. To address this problem, the authors propose using other formalisms such as Communicating Sequential Processes (CSP) [127] in combination with Z. Additionally, in a critique of this work, Wagner [188] raises questions about the practical relevance of the proposed approach based on Z.

Wagner argues that the use of Z introduces many limitations in defining several aspects of an agent's behavior, including goals, the state of the external environment, and perception mechanism. Furthermore, high-level functionalities regarding information and knowledge processing are neglected, due to the limitations imposed by Z.

Hilaire et al. [126] also present a formal approach to MAS based on a composition of Object-Z [178] and statecharts [122] formalisms. One of the main goals of this work is to offer a formalization that fits in with prototyping and simulation oriented processes, bridging the gap between theory and practice. Object-Z is used to specify the transformational aspects of the system and statecharts is used to specify the reactive aspects. Despite being practice-oriented, the approach faces some implementation limitations including executability of Object-Z specifications.

- *Using executable (temporal) logics, where specifications can be directly executed:* Executability is an important feature of these approaches; however, it usually introduces limitations on the expressive power of the underlying logic. The framework proposed by Wooldridge and Fisher [92] fits into this category. They outline a formal approach for specification, verification, and rapid prototyping of multi-agent systems. Agent specifications are developed in a temporal logic and are made executable by using the *Concurrent METAEM* platform. One critique of these approaches relates to the difficulty of understanding temporal logic specifications for non-experts, which makes validation of the models more challenging.
- *Using modal logic for defining relationships between various mental states (e.g., belief, desire, intention):* These approaches mainly aim at *defining* complex mental state of the agents without much concern for their respective computational models. Some work has been done in this direction by combining existing logics dealing with different aspects of agency.¹ However, one of the most challenging problems of combining logics is determining the expressive power of the combination. Also, depending on the combination technique that is used the resulting logic may have different properties. For instance, it may maintain only the common properties of both logics (called *fusion*), or may be a completely symmetric combination of both logics (called *full-fibring*) [70].

¹For a more detailed description and a complete list of references, we refer to [70].

3.5 Multi-Agent Based Simulation (MABS)

Multi-Agent Based Simulation (MABS) uses the MAS paradigm and builds on top of some of the existing simulation paradigms, such as parallel and distributed discrete event simulation, object oriented simulation, and dynamic micro simulation [63], to model a plethora of systems— those ranging from simple entities to groups of complex entities, from simple to complex interactions, from static to open dynamic environments. The flexibility of the paradigm in dealing with different notions of *individuals*, different types of behavior (e.g., reactive and cognitive), and different levels of abstraction (e.g., group or individual) facilitates its use in a variety of scientific domains, including biology, physics, ecology, and economics [73].

Nonetheless, there is a lack of well-established frameworks, methodologies or software engineering techniques for MABS. This is partly due to the lack of a well-defined semantic model of agents, as discussed in 3.4. Most existing agent-based simulation models use weak notions of agents and define agents only at a conceptual level rather than a concrete level. While the *concept* of agent is used in designing simulation models, there usually is a semantic disconnect between the agent as it is designed, and the one which is eventually implemented (and used in simulation experiments). As a result, the success of MABS is hindered [73]:

The semantics associated [with the core multi-agent concepts] differ considerably from one model to another, or from one implementation to another [...] This fuzziness, at the computational level, about what an agent really is can be found in all the other levels required for the design of a simulation.

This lack of a coherent view to agents leads to inconsistencies between design and implementation and intensifies the notorious validation problem. As [73] argues,

there is absolutely no guarantee that what is being designed and implemented corresponds to what has been desired and modeled by the thematicians [domain experts] [...] Computational agents as they are defined in MAS are simply not used in today's MABS.

Furthermore, in the major methodological frameworks of MABS (e.g., [106, 93]), the task of transforming the initial domain model into a computational model is assumed to be a natural one. There is not much attention paid to *building* the model, although such a transformation is not trivial [73].

To address this problem, [73] presents a design process for MABS as a *role-playing game* and identifies three main roles; *thematicians*: domain experts, *modelers*: software engineering people, and *computer scientists*: programmers.² The role of the thematicians is to provide relevant information about the target system by converting their knowledge of the system into a *domain model* which contains *real agents*³ and their respective behaviors taken from observations, theories and assumptions. The modeler translates the domain model into a more formal *design model*. At this level, the real agents are refined into *conceptual agents* defined based on the principles of multi-agent systems (e.g., behavioral model, interaction, environment, etc.). This step is considered the most difficult one, since the design model has to remove ambiguities and inconsistencies in the domain model, clearly specifying the conceptual agents. In the last step, the computer scientist translates the design model into an *operational model* on top of which a *computational system* is built. The operational model defines *computational agents* that are implementations of conceptual agents.

The proposed design process also includes *participatory design of simulations* through role-playing games performed by the experts and non-experts. However, the approach does not address the lack of a clear operational semantics for modeling multi-agent systems, which still remains as an impediment in transforming conceptual agents to computational ones.

In [175], the authors identify similar problems by citing “the number of commercial agent-based applications is not large, for the lack of mature, off the shelf, methodologies for agent based application development. One would like the advantages of an organized development process such as re-usability, testing and maintenance to be applied to agent-based systems as well.”

Introducing formal methods in the realm of agent-oriented analysis and design can serve as a solution to many of the problems that MABS is facing.

3.6 Agent-Based Social Simulation (ABSS)

In the recent years, agent-based models have gained wide-spread popularity for modeling social phenomena [104]. As described in 3.2, the history of social simulations has evolved

²Although the terminology used in this work does not match the widely-accepted terminology in computer science, the presented model points out fundamental problems in designing MABS models.

³Real agents represent agents that can be observed and analyzed in the target system [73].

into agent-based simulation models, and there are known advantages in using agent-based modeling, especially, to capture non-linear dynamics of social systems [106].

In [107], Gilbert and Troitzsch point out that although social simulations can be used both for *exploration* and *prediction*, the potential of simulation models to assist in *discovery* and *formalization* is the main reason behind social scientists' increasing interest in simulation. In this context, the *algorithmic* nature of agent-based models offers more flexibility in specifying complex social systems, in comparison with other modeling techniques such as system dynamics [94], or microsimulation [121].

Similarly, given the high complexity of social systems, Srbljinovic and Skunca [182] point out that the primary purpose of agent-based modeling in social sciences should not be prediction, as it is nearly impossible to attain a satisfactory level of accuracy. Thus, they advocate the use of explanatory models rather than predictive ones, which would “provide us with means of performing simulation-enhanced thought experiments aimed at improving our intuition and understanding about the modeled phenomenon”. As such, simulation models are found useful in developing new theories and formalizing existing ones.

Along the same lines, Drogoul and Ferber [72] point out the advantages of using ABSS in: (1) *testing hypotheses* about emergence of social structures from individual behavior (by experimenting at the micro level and deriving patterns at macro-level), (2) *building theories* that contribute to sociological development, and (3) *integrating different theories* from different disciplines into a general framework.

Seror [173] highlights the advantages of using mathematical models as basis for simulations. Mathematical models provide rigorous specifications that are accessible to scientific criticism and replicable by other researchers. They also facilitate the development of rigorous frameworks based on specifications, and allow for formal or informal reasoning about the model that provides insight into the behavior of the target system. de Marchi [64] also discusses the virtues of having a unified framework for combining formal models, statistics, and computation in the study of social sciences.

3.7 Summary

Several approaches from mathematical modeling, statistics, and simulation, spanning stochastic processes, game theory, cellular automata and multi-agent systems have been used for modeling complex social systems [105]. In particular, the agent-based modeling paradigm

has become the focus of latest developments in computational modeling and simulation of social phenomena [13, 152]. Multi-Agent Based Simulation (MABS) uses the MAS paradigm and builds on top of some of the existing simulation paradigms, such as parallel and distributed discrete event simulation, object oriented simulation, and dynamic micro simulation [63], to model a plethora of systems. The paradigm has been used in a variety of scientific domains, including biology, physics, ecology, and economics [73]. For social systems, MABS models are used for testing hypotheses about social behavior, integrating different theories from different disciplines, and developing new theories [72]. The specific application of MABS for modeling and simulation of social systems is characterized as Agent-Based Social Simulation (ABSS) [63].

Nonetheless, the lack of well-established frameworks, methodologies and software engineering techniques for agent-based simulation has hindered its success [73]. This is partly due to the lack of a well-defined semantic model of agents [70] that leads to inconsistencies between design and implementation. It also makes validation more difficult since the connection between what is desired by domain experts and what is being implemented is very hard to establish [73].

Furthermore, in the major methodological frameworks of MABS and ABSS, including [93] and [106], the task of transforming the initial domain model into a computational model is assumed to be a natural one. There is not much attention paid to *building* the model, although such a transformation is not trivial. This issue is addressed in some existing frameworks, such as [73], by proposing a participatory design of simulations through role-playing games performed by the experts and non-experts. However, the problem of the lack of a clear operational semantics for modeling MAS is not addressed and still remains as an impediment in transforming conceptual agents to computational ones.

In the next chapter, we propose a methodological framework and supporting tool environment for computational modeling of complex social systems, specifically focusing on the shortcomings of existing agent-based approaches.

Chapter 4

Vision and Methodology

Computational modeling of social systems, due to their inherent complexity and diverse multidisciplinary aspects, is an iterative and potentially open-ended process that involves frequent changes and adjustments. Lack of rigorous and structured approaches for modeling could lead to a complex web of problems similar to the ones in early days of Software Engineering. The variety of software development processes and methodologies available today have been developed over the last few decades in response to the *software crisis* that originated in the 60s and 70s. Existing methodologies aim at providing systematic approaches to deal with the complexity of software development, including poor specifications, size, validation, and maintenance. Despite this variety, the focus of available methods is on the end result; i.e., the *software product*., which can not be directly applied in modeling and simulation of social systems.

In developing a methodology for computational modeling and simulation of social systems, it is important to note that such models serve two main purposes: (1) they offer sandboxes and decision support tools for scientists, policy makers, and practitioners to explore different ideas or what-if scenarios in order to develop a better understanding of a system, and explore the potential implications of different intervention strategies; (2) the process of building a computational model enforces logical thinking which leads to critical analysis of even the basic assumptions about a system. Such mathematical rigor opens new ways of thinking about a system, and leads to identifying and resolving hidden assumptions, ambiguities, loose-ends, and possible errors in the understanding of a system.

Any proposed methodological framework for computational modeling and simulation of

complex social systems must support well-known best practices in complex systems modeling; models should be as *simple* as possible, the *validation* approach and its purpose must be clearly defined, and the modeling process must be *well-structured* and assumptions must be *well-documented* in order to uncover potential problems and support decision making (see 3.1). It is argued that the biggest advantage of modeling social systems is in developing a deeper understanding of the system under study, testing different hypotheses, evaluating the interplay of different variables involved, and eventually providing a better framework for thinking about alternative solutions. This requires a close collaboration between domain experts, modelers and users of the models such as practitioners and policy makers.

Therefore, like any software development project, the *communication problem* presents a major challenge [95, 15]. In interdisciplinary research, this is intensified because of the inherent differences of disciplines involved. In order to effectively communicate, a shared conceptual understanding of the system under study must be developed. Therefore, it is important to find the right level of abstraction for communicating the essence of computational models in order to allow for critical inspection, validation and modification by domain experts. Furthermore, contrary to most software projects where the focus is on producing a final product, computational modeling projects are more concerned with testing theories and generating new ideas. These characteristics shift the focus of the development cycle to the design or prototyping phase instead of implementation [164].

We build on the successful application of *agile formal methods* [95] in software development, and develop a framework that is specifically tailored for the above-mentioned requirements. Agile methods are well suited to our purpose, particularly because of their emphasis on individuals and their interactions, and the idea of iterative development [95].

4.1 Framework Architecture

In the process of collaborative modeling of behavioral aspects of complex social systems, we distinguish three essential phases, namely *conceptual modeling*, *mathematical modeling*, and *computational modeling*, with several critical phase transitions and feedback loops as illustrated in Figure 4.1. Starting from a conceptual model that reflects the characteristic properties of the phenomena under study in a direct and intuitive way, as perceived by application domain experts, a discrete mathematical model is derived in several steps. The first phase transition focuses on model construction, gradually formalizing core properties

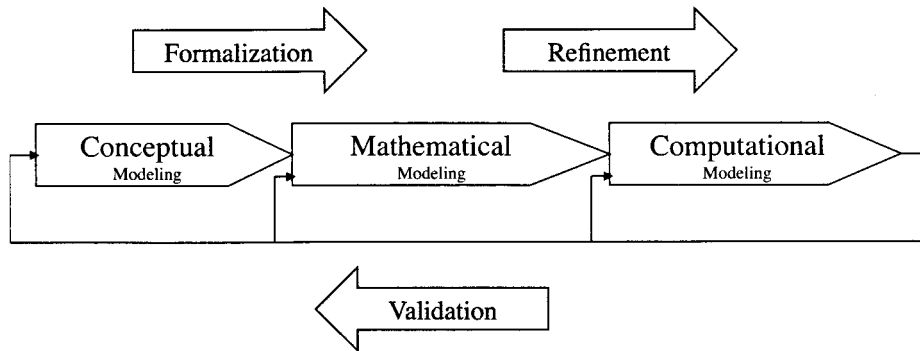


Figure 4.1: Different phases in modeling complex social systems

of the system under study in abstract mathematical and/or computational terms. Different mathematical modeling techniques may be used in this phase to capture key elements of the system.¹ The resulting model is then transformed into an initial computational model that is executable in principle; that is, any aspects that have been left abstract provisionally should be filled in as the result of subsequent refinement steps. Ideally, any such refinement would be restricted to just adding details as required for running experiments, both to help establishing the validity of the formal representation of the conceptual model and for further experimental studies. In reality, however, modeling is a highly iterative and non-linear process with feedback loops within and also across the various phases, potentially affecting the design of the model in its entirety.

The role of the mathematical model is to assist in formalizing the conceptual view of the target domain, so as to provide an exact description of the characteristic properties as a reliable basis for deriving a computational model. Marking the transition from an informal (or semi-formal) to a formal description, the mathematical model serves three main purposes:

1. it provides a precise and rigorous blueprint of the system under study and formalizes key system attributes, allowing systematic analysis and reasoning about those attributes. In turn, such formal analysis serves as feedback to the initial transformation step, which is typically the most challenging one;
2. the process of mathematical modeling enforces logical and structured thinking about

¹This includes well-established mathematical modeling techniques e.g., queuing theory, cellular automata, game theory, graph theory, etc.

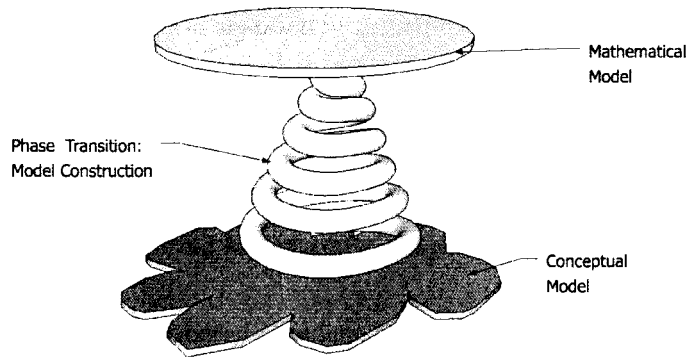


Figure 4.2: Model construction: The phase transition from the conceptual model to the mathematical model

the key aspects of the system. Specifically, for social systems, it unveils hidden assumptions, loose-ends, and gaps in the conventional understanding of such systems;

3. it serves as ‘semantic middleware’ for bridging the gap between the conceptual and the computational view. It provides a platform to ensure that key attributes are properly established and well understood prior to actually building the computer model.

Building the mathematical model involves several iterations in order to transform the informal descriptions of the concept into precise, rigorous and clearly defined ones, as shown in Figure 4.2. The process involves finding the right level of abstraction to represent the conceptual elements in simple yet meaningful terms, and clarifying the scope and boundaries of the model, which lead to a better understanding of the system itself. Depending on the choice and representation of the mathematical model, its transformation to a computational model can be less problematic, whereas the validation of the outcome of the computational phase usually poses another difficult problem.

While many modeling techniques used in computational modeling of complex social systems have a firm mathematical foundation (e.g., graph theory), some of the most widely used techniques, such as multi-agent systems (MAS), lack formal semantics and the required precision and rigor (see 3.5). This leads to a lack of coherence and consistency in mapping the abstract intuitive understanding of the conceptual model to its computational representation.

The approach we propose here emphasizes a smooth and seamless transition between the three phases of modeling and accommodates the highly iterative process of modeling and

validation. We build on common abstraction principles of applied computational logic and discrete mathematics and use the abstract state machine (ASM) method as the underlying mathematical paradigm [30, 22]. Abstract state machines are known for their versatility; ASMs capture the principal models of computation and specification in the literature, including classical models of computation such as automata. In particular, distributed ASMs provide a natural fit for capturing key concepts of agent-based modeling in a formal framework.

The popularity of the agent-based paradigm in modeling complex social system is mainly due to its ability to deal with *individual* agents and its flexibility in defining those agents. However, such flexibility has also led to a bewildering range of definitions and architectures for agent based systems, where there is often no clear connection between conceptual agents and their computational counterparts. Our approach aims at bridging this gap by adopting concept-oriented agents (defined using common agent-based methods) and mapping them to more computation-oriented agents (defined in ASMs). We take advantage of the power of ASMs in dealing with semantic aspects at desired levels of abstraction to handle the complexity of different aspects of agency and autonomy in a structured way. It is important to emphasize that we do not offer ASMs as the solution to all complex issues in the MAS field (e.g., cognition). Instead we propose the ASM method as a systematic approach to identify and separate concerns through proper abstractions, to gain clarity and precision, and finally to sharpen the blur between conceptual and computational views with mathematical rigor.

Furthermore, the role of ASMs in our methodology goes beyond the linkage with MAS modeling. Many complex systems of today are influenced by a combination of social and technical aspects, for which no established modeling methodology exist. Our proposed framework and the precise, yet simple semantic ground provided by ASMs plays a key role in identifying and analyzing essential properties of such systems and addressing the open problems. Chapter 6 provides a comprehensive description of one such case study on identity management architecture.

4.2 Formal Modeling Technique

A central question in computing science is how to precisely define the notion of *algorithm*. Traditionally, Turing machines have been used in the study of the theory of computation as a formal model of algorithms [177]. For semantic purposes, however, this model is utterly

inappropriate due to its fixed level of abstraction.

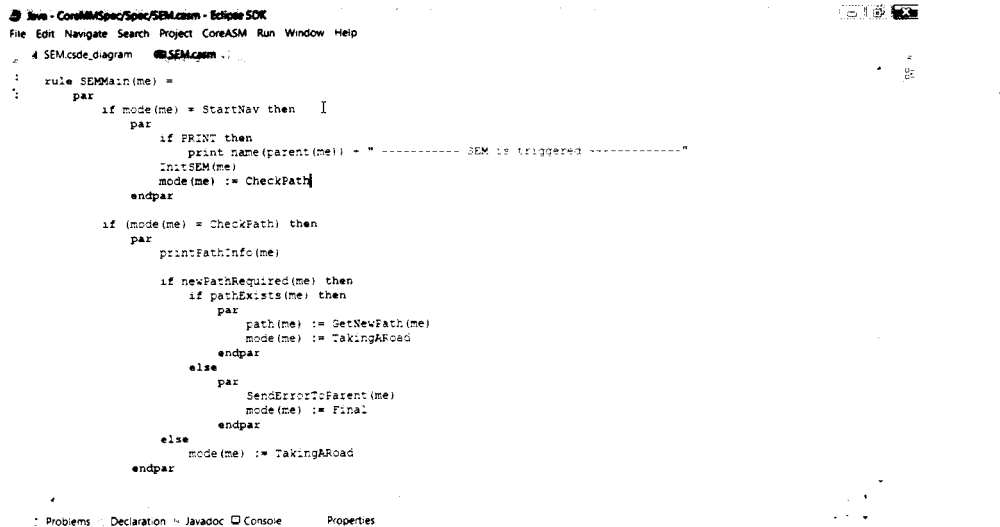
The origin of abstract state machines was the idea to devise a generalized machine model so that any algorithm, regardless of its level of abstraction, can be modeled at its natural level of abstraction. That is, every computation step of the algorithm essentially has a direct counterpart (usually a single step) performed by the machine model. Theoretical foundations show that both the notion of sequential algorithm and of parallel algorithm are captured respectively by the models of *sequential ASM* [119] and *parallel ASM* [16] in the aforementioned sense. For distributed algorithms (including concurrent and reactive systems), the *distributed ASM* framework provides a generalization of the two other models that is characterized by its asynchronous computation model with multiple computational agents operating concurrently.²

An ASM ground model (see 2.2) serves as a precise and unambiguous foundation for establishing the characteristic dynamic properties of a system under study in abstract functional and operational terms with a suitable degree of detail that does not compromise conceivable refinements [23]. A ground model can be inspected by analytical means (verification) and empirical techniques (simulation) using machine assistance as appropriate. Focusing on semantic rather than on syntactic aspects, the very nature of ASM ground models facilitates the task of critically checking the consistency, completeness and validity of the resulting behavioral description.

Abstract executable specifications offer many advantages in model-based systems engineering and serve as a tool for design exploration and experimental validation through simulation and testing [83]. Pertinent to our purpose, they greatly facilitate validating a ground model by executing different scenarios and comparing the resulting behavior with the behavior *expected* by the domain experts. In many cases, observation of system behavior can lead to the discovery of new concepts or elements in the underlying system that may have been previously neglected.

Abstract state machine models can be executed using any of the existing advanced executable ASM languages, including CoreASM [83], Asmeta [100], AsmL [151], the ASM Workbench [66], XASM [9], and AsmGofer [172]. However, among all available tools, only CoreASM comes with a run-time system supporting the execution of distributed ASM models, which are essential in our work. Furthermore, the design of CoreASM is novel and the

²As such, it closely matches the basic concepts of MABS (see 5.3.1).



```

Eclipse - CoreASMSpecSpec/SEM.casem - Eclipse SDK
File Edit Navigate Search Project CoreASM Run Window Help
SEM.casem diagram SEM.casem .i...
rule SEMMain(me) =
  par
    if mode(me) = StartNav then
      par
        if PRINT then
          print name(parent(me)) = "----- SEM is triggered -----"
          InitSEM(me)
          mode(me) := CheckPath
        endpar
      endpar
    if (mode(me) = CheckPath) then
      par
        printPathInfo(me)
        if newPathRequired(me) then
          if pathExists(me) then
            par
              path(me) := GetNewPath(me)
              mode(me) := TakingARoad
            endpar
          else
            par
              SendErrorToParent(me)
              mode(me) := Final
            endpar
          else
            mode(me) := TakingARoad
          endpar
        endpar
      endpar
  endpar
  Problems Declaration Javadoc Console Properties

```

Figure 4.3: A sample CoreASM program in Eclipse: CoreASM offers minimal encoding and maintains executability of abstract models

underlying design principles are unprecedented among other existing languages; it is designed for systematic language extensions using an advanced plugin architecture. CoreASM is also the only language that builds on untyped language concepts which is key to the underlying theoretical model of ASMs. Besides facilitating experimental validation of ASM models, CoreASM also provides support for model checking [147]. Through model checking the correctness of a system with respect to all of its possible behaviors can be formally verified [85].

In the following we review key features of the CoreASM tool environment which are essential in our framework for computational modeling of complex social system. Specifically, the Control State Diagram editor (CSDe) tool has been developed during the course of the Mastermind project (further discussed in Chapter 5) to address the needs of interdisciplinary research projects on computational modeling and simulation of social systems.

4.2.1 Rapid Prototyping with CoreASM

CoreASM is a novel executable ASM language³ which is well suited to *exploring* a problem space in early stages of designing a system through rapid prototyping of ASM system models. By minimizing the need for encoding in mapping the problem space to a formal model, the language allows writing highly abstract and concise specifications, starting with mathematically-oriented, abstract and untyped models, gradually refining them down to more concrete versions with a degree of detail and precision as needed.

CoreASM maintains executability of even fairly abstract and incomplete models, which is a great asset in improving communication with domain experts in order to reach a common ‘computational’ view of the system. The principle of minimality, in combination with robustness of the underlying mathematical framework, improves modifiability of the design while effectively supporting the highly iterative nature of specification and design [83].

CoreASM also offers a high level of flexibility in design through a well defined plugin architecture [82]. Specifically, it supports writing simple plugins that address the specific needs of a domain, such as visualizing the results of an experiment or even defining domain-specific languages. In other words, plugins allow for encapsulating the mathematical artifacts of a computational model into a comprehensible and familiar format for the domain experts. This greatly facilitates communication with domain experts and analysis of the results for validation purposes. The use of this feature in our case study is explained in more detail in Chapter 5. Figures 4.3 and 4.4 provide snapshots of a sample CoreASM code and a CoreASM visualization plugin.

4.2.2 Interactive Design with Control State ASMs

One of the fundamental principles of our approach is the direct involvement of non-computing experts in the design and development process. Arbitrary design choices made by computing experts not intimately familiar with the social system under study are potentially dangerous and can lead to fatal design flaws due to misconceptions or oversights. However, it is usually difficult for non-computing team members to understand the development process and especially the formal representation of a system. Hence, it is necessary to make development as transparent as possible, for instance, by using visual representation means, such as *ASM control state diagrams (CSD)*, also called *Control State ASMs*.

³See www.coreasm.org for how to obtain the tool environment and documentation.



Figure 4.4: Mastermind visualization plugin complements model execution with familiar visualizations for domain experts

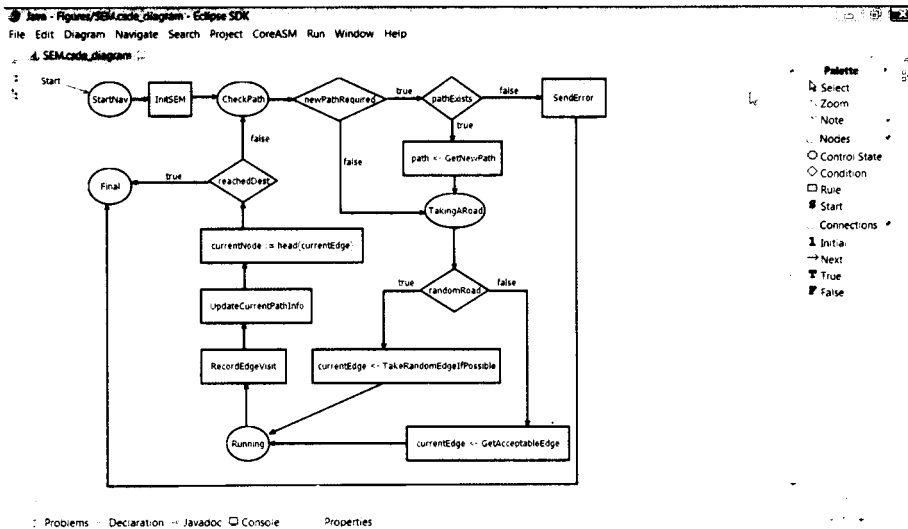


Figure 4.5: CSDe: A Control State Diagram editor plugin for the Eclipse development environment, with automatic translation to CoreASM code

Control State ASMs are a very practical class of abstract state machines. They can be represented as a directed graph, where nodes consist of control states⁴ interspersed optionally with conditions and/or rules. Conditions direct the flow of execution; rules denote actions taken as part of state transitions [85]. As such, they are viewed as “a normal form for UML activity diagrams [96] and allow the designer to define machines which below the main control structure of finite state machines provide synchronous parallelism and the possibility to manipulate data structures” [30, p. 44]. The expressive flexibility of this type of ASM is demonstrated by their capacity for representing many classical automata such as various extensions of finite state machines [30]. Control state ASMs, by definition, can be depicted graphically, and offer a sound foundation for visual modeling.

Despite similarity to the more complicated UML activity diagrams, ASM CSDs do not require any special training to understand. Their simplicity allows the interdisciplinary reader to focus on the content of the description rather than the formalism. The accessibility and ease of use of CSDs make them an integral part of our framework and modeling process. In our experience, domain experts are able to understand a CSD, and even suggest changes to it, regardless of their technical background. As such, CSDs act as both a means of clarifying communication between development partners and of enabling conceptual validation.

The Control State Diagram editor (CSDe) is a software tool for creating and modifying Control State ASMs. The tool was developed during the course of the Mastermind project (see Chapter 5) to facilitate the involvement of non-computing domain experts in the modeling and design process. The editor allows not only the construction and editing of CSDs through a graphical interface, but also automatic translation of the diagrams into CoreASM code. Since a CSD may not include the initial system state or other information required to run as program, it is possible that the CoreASM file generated is not directly executable. However, the resulting code acts as a foundation for further development of the structure under consideration. This automated translation from a diagram into code improves the ease of transition from high-level design towards subsequent stages of development [85]. Figure 4.5 shows a control state ASM created using CSDe within the context of the Mastermind project.

⁴Control states are similar to internal states of Finite State Machines and are used to describe different system modes [30].

4.3 Summary

We propose a methodological framework and supporting tool environment for computational modeling and software development of complex social system in interdisciplinary research contexts. We utilize the ASM formalism and CoreASM tool suite to facilitate interactive design and validation of such models. One fundamental characteristic of our framework is the focus on the model building phase of the development, emphasizing the importance of the cooperative process of transforming an abstract concept into a computational artifact. The ASM method also facilitates rigorous and seamless transitions between different phases of modeling ensuring the coherence and consistency of the final models. The proposed framework has been developed and tested in close collaboration with non-computing researchers, as will be seen in Chapter ???. It has been successfully applied in different application contexts, including the Mastermind project and the Identity Management Architecture project, as further discussed in the next two chapters.

Chapter 5

Mastermind: Modeling Crime Patterns

Mastermind is an interdisciplinary R&D project in Computational Criminology, jointly managed by the Institute for Canadian Urban Research Studies (ICURS) and the Software Technology Lab (STL) at Simon Fraser University (SFU). The project aims at developing computational models of criminal activity patterns in urban environments, with a special focus on spatiotemporal characteristics of crime, potentially involving multiple offenders and multiple targets.

5.1 Background

Here, we outline the basics of Computational Criminology and explore the existing approaches to modeling crime. We also study the existing literature on some of the key elements required for building models of criminal activity, such as navigation, learning, and the environment.

5.1.1 Computational Criminology

Crimes are complex multi-faceted events. They are comprised of at least four necessary dimensions: the law, the offender, the target and the place [36]. For several decades, criminologists have contended that there is definite patterning in the temporal and spatial characteristics of physical crime [32]. In particular, Environmental Criminology focuses on

studying crime in the context of people's movements in the course of everyday routines. Three major theories of the field, namely, Crime Pattern theory [39], Routine Activity theory [88], and Rational Choice theory [61] contend that crime locations are not random, but rather are determined through a combination of habitual movement and activity patterns, each of which is at least partly determined by the perceptions of the physical and social environment. Offenders choose good opportunities over bad risks through rational decisions made based on cues from the environment. Within this setting, one can identify the importance of several elements in analysis of criminal events, including offenders' movement, routine activity patterns, perception of the underlying environment, and different environmental cues influencing offenders' decision making.

Research in crime analysis strongly supports this theoretical theme for a broad range of crimes [32, 199, 166], but the methods used are mostly statistical and empirical in nature and rely entirely on direct extrapolations from past data. Furthermore, the conventional research methods in Criminology, like in other social sciences, face the problem of lack of control in running experiments. Novel research directions [40, 117, 144], thus, suggest a fundamentally different approach. Due to the intricate and highly dynamic nature of the underlying sociological systems, empirical deduction is not sufficient any more; mathematical and computational models are needed for reasoning about most likely scenarios [42].

Computational Criminology aims at pushing the existing limits in study of crime through interdisciplinary work with mathematics and computing science. Computational models allow for running experiments in simplified artificial situations where abstraction is used conveniently and systematically to analyze the influence of different elements under study. This facilitates dealing with the highly complex and dynamic nature of most social phenomena. Computer models can serve as a practical instrument for studying crime patterns and for reasoning about likely scenarios, facilitating the understanding of and experimenting with crime patterns.

Related modeling and simulation work in criminology [128, 40] confirm the value of pursuing computational methods to predict patterns in crime. Given the predictive nature of most crime analyses, especially those working from what is broadly construed the Environmental Criminology perspective, the blending of criminology, computing science and mathematics is a natural fit [51, 57, 117]. The application of computing science and modeling techniques in different branches of criminology have gained momentum in recent years. This includes developing simulation models of criminal justice systems [7], hotspot analysis

[137, 143, 200], offender profiling [14], and database and information systems management [31, 80, 125].

Environmental Criminology

Criminology has a rich history of interest in the spatial distribution of crime and criminal events. The sub-field of Environmental Criminology stems, in part, from this early ecological tradition [36]. Key concepts include the routine nature of many of our daily activities [88] and the structured way in which we become aware of, and interact with, our environment [37, 38]. These concepts are used in the analysis of violent and property offenses ranging from serial homicides to robbery and burglary [166, 199, 194]. Environmental criminologists contend that criminal events can be understood in the context of people's movements in the course of their everyday lives; i.e., offenders commit offenses near places where they spend most of their time, and victims are victimized near places where they spend most of their time. This line of theory and supporting research argues that the location of crimes is determined through a decision process shaped by perceptions of environment based on which good criminal opportunities are separated from bad risks. This implies there are a set of patterns/rules that govern the workings of a social system. One that is composed of criminals, victims and targets, interacting with each other. The movement of each individual in a given urban environment is influenced by the environment's underlying land use, street networks and transportation patterns, as well as high activity nodes like shopping centers and entertainment districts.

During the course of their everyday lives, most people are tied to at least three main classes of activity nodes: home, work and recreation [88]. They travel between these nodes using familiar pathways; the more often an area is visited, the more knowledge they will gain regarding the immediate surrounds for both the nodes and the pathways connecting them [39]. A person's general knowledge of the environment forms his/her *awareness space*. Within this *known* environment, a person develops a more specialized understanding of the places he/she frequently visits, which forms the *activity space* [36]. The activity space is the playground for criminals to commit crime; i.e., motivated offenders observe opportunities, or targets, within their activity space and potentially act upon them [39].

The following example further explains these ideas. If a person starts his or her day at home, and then travels to work (or school), for example, he or she will typically take the most direct and most easily navigated route. Along the way, this person will take notice of a

range of phenomena in his/her activity space. Even if the person does not immediately react to a specific phenomenon (e.g., stopping to purchase a coffee, or steal from a car), he/she will often remember such sites and visit them at a later time. Environmental criminologists view the learning and decision-making process for crimes to be much the same as those for non-criminal activities. However, different people have different levels of criminal propensity (i.e., the inclination to take advantage of observed criminal opportunities). People who are more criminally predisposed will respond to observed crime opportunities more frequently than people with lower crime potentials.

The process of choosing a target involves target templating [38]; i.e., potential criminal opportunities are compared to the offender's crime template in order to assess the value of potential rewards against the risks or the amount of energy required for successful execution of the act [58, 61, 163]. As an example, we consider a typical crime such as burglary, otherwise known as break and enter ("B & E"). For the offender to be aware of a potential target, the site is usually located within his/her activity space. For instance, a burglar may notice a residential building while traveling from home to work everyday and identify it as a *good* target that *fits* his/her crime template. Criminology research [162, 163, 77] suggests that several variables such as property value, obvious entry opportunities and lack of occupants form *cues* [76] based on which a decision to commit crime is made by an offender.

5.1.2 Related Work

In examining the theoretical foundation of the Mastermind project, we should mention the research done at the Virginia Institute for Justice Information Systems (VIJIS). This group at the University of Virginia applies statistical methods to the analysis of both physical and cyber crime. In an outline of a design for a multi-agent simulation system [117] they consider central aspects of criminological analysis, including spatial mobility, rational choice and routine activities theories and unveil important issues to consider when building such a system and the elements that it requires. Three hypotheses of criminal activity are examined in detail: spatiality, rational choice and routine behavior. Their justification for the value of a multi-agent system (MAS) is worthy of consideration. We have also found many of the points raised to be important in the development of Mastermind. In [200], criminal incidents are viewed and analyzed as spatial choice processes. Xue and Brown [200] propose two models for criminal site selection, which they obtain by modifying a traditional

discrete choice model. In a comparison based on real crime data both spatial choice models outperform the hot spot model they are compared to in their predictions of future spatial choices for crimes.

In terms of experimental focus, the Mathematical and Simulation Modeling of Crime project, called UC MASC [35], being undertaken at the University of California shares several important characteristics with Mastermind. The value of agent-based modeling in understanding offender behavior is highlighted in their work, and statistical and mathematical methods are employed to determine the movement of agents within the simulation environment. They also mention the use of geographic information systems for the comparison to real-world results and verifying the accuracy of the model. Their work is more focused on large-scale patterns: they base their methods on the mechanics of swarm behavior, while the Mastermind project looks more carefully at the decisions and planning made at the individual level. However, it is important to heed the warning of the UC MASC project: the results of a simulation are only as good as the veracity of the elements used to build it [43].

Another project with related subject matter is the SimDrug project [155], which looks at the trade and use of heroin in Melbourne, Australia, during the drug “drought” of 2000. It differs from Mastermind because it is more concerned with inter-agent activity, and also because it takes place in a hypothetical environment. An interesting contrast is emphasis on the value of complexity. By including many sources of real-world data, the SimDrug project hopes to give the generated results a better grounding in reality. On the other hand, the systematic use of abstraction in Mastermind encourages simplicity in modeling. However, it is also important to recognize the point that SimDrug presents: computing is an ideal tool for the analysis of data-rich simulations.

5.1.3 Navigation

The problem of *navigation*, also called *path finding* or *way finding*, is a complicated one. In simple terms, it can be viewed as moving an entity from source S to destination D by identifying the different paths that can be taken, evaluating those paths under given circumstances, and finally choosing the most suitable path. One such suitable path from a source to a destination is the *shortest path*, or the path with the *minimum cost*. In the following, we review some of the existing algorithms for navigation.

Shortest Path Problem

The problem of finding the shortest path in static graphs/networks is well-studied after decades of research and experimentation. The shortest path problem is also one of the most fundamental issues in network optimization and robot path planning.

In [60], the shortest-path problem is formalized as follows.

- A weighted, directed graph $G = (V, E)$ is given, where $V = \{v_1, \dots, v_n\}$ is the set of vertices, $E \subseteq V \times V$ is the set of edges on the graph.
- A weight function $w : E \rightarrow R$ maps every edge in the graph to a real-valued weight.
- The weight of a path $p = \langle v_0, v_1, \dots, v_k \rangle$, $v_i \in V$ is defined as the sum of the weights of its constituent edges;

$$w(p) = \sum_{i=1}^k w(v_{i-1}, v_i) \quad (5.1)$$

- A path p between vertex u and vertex v is denoted by $u \xrightarrow{p} v$, where $u, v \in V$.
- The weight of the shortest path from u to v , where $u, v \in V$, is defined by:

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\} & \text{if there is a path from } u \text{ to } v. \\ \infty & \text{otherwise.} \end{cases} \quad (5.2)$$

- The shortest path from vertex u to vertex v is defined as any path p such that weight $w(p) = \delta(u, v)$.

The most famous shortest path algorithm is the one proposed by Dijkstra in 1959, known as the *Dijkstra's shortest path algorithm* [68]. This greedy algorithm guarantees to find the optimal shortest path in a given graph with non-negative edges in time $O(n^2)$. A brief description of the algorithm is provided in Appendix A. There are several implementations of this basic algorithm that use heap and queue data structures to achieve time and space efficiency and improve the performance to $O(n \log(n))$. Most other shortest path algorithms are variations of this generic algorithm. A good description of the classical shortest path algorithms and their implementation appears in [99].

Instead of exploring all possible paths, the A* search algorithm uses heuristics to first search more promising paths. This avoids exploring unfruitful directions and leads to a faster

search. The A* algorithm is most widely used in gaming and Artificial Intelligence (AI). The construction of the heuristic function involves some overhead and should be weighed against its benefits. Furthermore, this algorithm is not guaranteed to find the optimal path, although optimality can be achieved under certain imposed conditions (on the heuristic function).

Several surveys and experimental evaluations have been carried out to compare the performance of different shortest path algorithms, classical and new. In [56], an exhaustive study is performed on 17 shortest path algorithms including the Dijkstra's algorithm and its different implementations. A number of simulated networks with varying degrees of complexity are used in the experiments. The results show that there is not one best algorithm for all problems; however, for graphs with non-negative weights, Dijkstra's algorithm outperforms the rest. Zhan and Noon [154] have performed a similar study on road networks which shows that Dijkstra-based algorithms outperform other algorithms. Primarily based on the above studies, Zhan [202] identifies three fastest algorithms for real road networks, two of which are Dijkstra-based algorithms.

5.1.4 Learning and Memory

In modeling the behavior of a person, or a potential offender, one important factor is to build a model of a specific behavior that is as close as possible to reality. For instance, a navigation algorithm that always provides the most efficient path is not ideal in our application context; alternatively, an ideal solution would provide the most *natural* or the most *intuitive* solution. Thus, the complexity lies in defining what is most natural, and validating if a solution is natural.

One behavioral element that needs to be considered in this respect is *learning*. As discussed in Section 5.1.1, a person's past experience has a substantial impact on his/her behavioral patterns. This topic is extensively studied in the field of *case-based reasoning*, which aims at incorporating learning into problem solving. In this section, we briefly review this topic and explore its application in our project.

Case-Based Reasoning

The case-based reasoning (CBR) paradigm relies on remembering previous situations, called *cases*, and applying this knowledge from past experiences to new problems [6]. It has

been studied both from the Cognitive Science perspective, to model human reasoning and learning, and the Artificial Intelligence perspective, to develop relevant computer reasoning technologies. The CBR paradigm assumes a certain regularity about the world that allows for adapting previous solutions to solve routine or novel problems [140]. Thus, *learning* plays a central role in CBR. A case-based reasoner learns from its previous successes as well as failures by incorporating those into its reasoning. As a result, it becomes more *competent* and more *efficient* over time, as it encounters and solves more cases [140, 139].

A case-based reasoner can be either *interpretive* or *problem solving*. Interpretive reasoners use prior cases for classifying or characterizing new problems, while problem solvers use previous cases to suggest solutions to a given problem based on past experiences [139].

All CBR approaches share a common four-step reasoning process: (1) *retrieve* a case, from the library of past cases, that matches the current problem, (2) *reuse* the retrieved case to solve the current problem, (3) *revise* and *adapt* the solution if necessary, and (4) *retain* the final solution as a new case in the case library [4]. Each of these tasks is broken down into sub-tasks and a variety of approaches has been proposed to implement each sub-task.

Case-based reasoning has a wide array of applications in different fields including medicine, law, automation and robotics. Commercial applications of CBR include diagnosis systems for retrieving past cases with similar symptoms, help desk systems used in the customer service area, and decision support systems. For a comprehensive review of applications of CBR, we refer to [193].

With respect to our problem domain, adopting case-based reasoning in Mastermind could offer an effective way to incorporate learning in our model. Rather than re-solving problems from scratch every time, we can develop a more natural and intuitive model of a person's behavior by learning from past experiences and using them in solving new problems.

Hybrid Systems

In CBR, past experiences, in the form of *cases*, are utilized in order to solve a problem by *remembering* instead of solving it from scratch using *rules* [6]. Rule-based approaches, on the other hand, explicitly solve the problem from scratch using *generalized rules* [149].

Rules and cases have complementary strengths. Due to their interchangeable nature, rules and cases can be easily integrated in order to produce an effective reasoning framework, overcoming the disadvantages of each reasoning method [158]. In [116], an example of such a hybrid system is provided where a case-based reasoner is used to improve the accuracy

and efficiency of a rule-based reasoner (RBR) in a direction that the system could not have achieved with the rules alone.

The architecture of a hybrid system may be *rule dominant*, *case dominant*, or *balanced*, depending on which component has the primary role in the decision making process [158]. Numerous successful examples of such hybrid systems applied in different domains can be found in the literature (e.g., [158, 116, 148]).

5.2 Project Overview

The goal of the Mastermind project is to capture the complexity and diversity of criminal behavior in a robust and systematic way. A variety of software development methods were applied and constantly reviewed with respect to their usability, expressiveness and effectiveness, the result of which has led to the development of the modeling framework presented in Chapter 4.

Our focus is on the concepts of Environmental Criminology, which argue that in spite of their complexity, criminal events can be understood in the context of people's movements in the course of everyday routines [36, 88]. Therefore, we place possible offenders in an environment that they can navigate. Through movement within this environment, they develop mental maps that correspond to the concepts of *awareness space* (the places a person knows) and *activity space* (the places a person regularly visits) [36, 41]. In the course of their daily routine activities, agents move from one location to another, and may visit potential targets along the way [88]. In its core, Mastermind captures what is suggested by Crime Pattern theory; i.e., crime occurs when a motivated individual encounters a suitable target [41].

The main building block of Mastermind is a robust abstract state machine (ASM) ground model (see 2.4) developed through several iterations. To this end, we applied simple graphical notation for communicating the design, using CSDe (see 4.2), and utilized abstract executable models in early stages of design, using CoreASM (see 4.2). Furthermore, the ground model is refined into more concrete models with specific details systematically added, an example of which is the simulation model of Mastermind implemented in Java. This version provides a responsive user interface and a simulation environment based on real-world Geographical Information System (GIS) data. We also refined the CoreASM executable ground

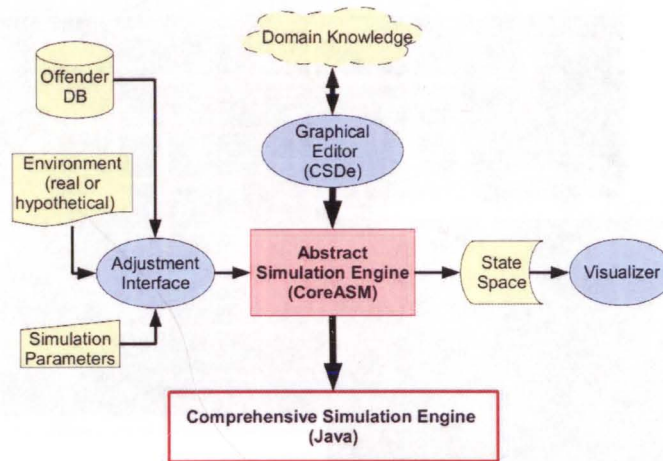


Figure 5.1: Mastermind system architecture

model to create more controlled experiments, which allow for a structured analysis of theories in a hypothetical world. Both Java version and CoreASM version provide visualization features which are a priority for criminology publications.

Figure 5.1 shows the core architectural components of the Mastermind system. We define a single interface, called the *adjustment interface*, for dealing with the inputs to the model, including different representation of the environment, various simulation parameters, and characteristics of offenders captured in their profile. The system allows the environment to be either hypothetical and defined manually, or to be imported from standard Geographic Information System (GIS) databases. At this stage, offenders' profiles are set up manually, but the architecture allows for direct connections to offender databases in order to automatically import information about known offenders (available in crime information warehouses) into the simulation. This feature conceptually opens up different ways of using the system by bringing together offenders' information and their dynamic spatiotemporal behavior patterns in a single framework

Figure 5.2 shows snapshots of both implementations of Mastermind, illustrating agents' movement between activity nodes, the formation of their activity spaces and the effects on crime hotspots. The CoreASM model is meant to study concepts at a higher level of abstraction, using a simple grid structure. In contrast, the Java version runs on the real road network of downtown Vancouver, including Stanley Park, and captures a finer degree

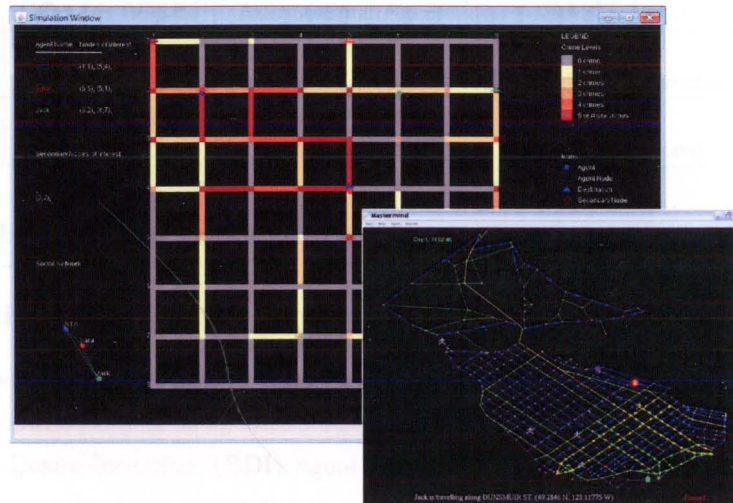


Figure 5.2: Java & CoreASM implementations of Mastermind. The Java version (front) aims at using computational power to simulate the dynamic interaction of a variety of factors, including a street network based on real world data. The CoreASM version (back) is more abstract, focusing exclusively on specific elements.

of detail and complexity.

The results of our work on the Mastermind project have been well-received both by the researchers in academia and law enforcement officials. Building on this success, the project is now continuing in several different directions as discussed in Section 5.5.

5.3 Mastermind Formal Model

As discussed in 4.1, the agent-based modeling paradigm plays a vital role in our modeling framework, bringing together predominant views in the world of social systems with the formal ASM view. The formal model we present here is based on a MAS view of a social system. Accordingly, one needs to make a distinction between various entities that constitute the underlying social urban environment. For instance, it is important to distinguish criminals, police officers and regular people, who are independent alive entities, from traffic lights and buses, which are lifeless entities with behavior, and Automated Teller Machines (ATMs), buildings and streets, which are entities with specific properties but no behavior.

Social System	MAS Model	ASM Model
Offender, Victim	Autonomous Agent	DASM Agent
Car, ATM	Active Object (Attributes, Behavior)	Object (Static/dynamic functions)
Cash, Drugs	Passive Object (Attributes)	Object (Static functions)

Table 5.1: Entity classification and taxonomy through different layers

5.3.1 Entity Classification: Linking Agent-Based Systems to ASM

We propose a generic classification of entities into three different categories¹: *passive objects*, *active objects* and *autonomous agents*. The core of this classification is based on the essentials of the Belief-Desire-Intention (BDI) agent architecture [50, 197] and the framework for agency and autonomy [146] (see 3.3).

A *passive object* is an entity that only comprises a set of attributes reflecting its characteristic features. For instance, an ATM machine is a passive object that has attributes such as location and bank name. An *active object* is an entity with attributes and an associated behavior defined as observable changes in the state of the object. However, this behavior is induced by the environment and not controlled by the object itself. A traffic light is an active object that has a set of attributes (such as location) and a predefined but changing behavior (such as being red, yellow, or green). An *autonomous agent* is an entity that, in addition to attributes and behavior, has *rules*, *motivations*, and a *memory*. The behavior of an autonomous agent is generated by a set of rules triggered by the agent itself to change its internal state or the state of its environment. Consequently, an autonomous agent is responsible for generating all of its behaviors. Motivations are incentives or goals that direct the behavior, and memory is nothing but a collection of facts representing the agent's knowledge of the environment.² Hence, a criminal offender is an autonomous agent with attributes (e.g., a personal profile), behavior (e.g., commuting), motivations (e.g., greed), and a memory (e.g., knowledge about targets).

¹Despite being generic, this categorization is not intended to be universal; it is only meant to capture the dynamics of our target system.

²This is analogous to a BDI architecture, whereby memory represents the beliefs, motivations represent the desires, and the rules represent the deliberative and means-end reasoning phase of the BDI agents.

We map each MAS entity onto a mathematical object in the ASM model as follows. We distinguish different types of MAS entities by defining different domains in the ASM model. A passive object is modeled as an element of the passive domain for which only static functions are defined representing the static attributes. Active objects are modeled, as elements of the active domain for which both static and dynamic functions are defined. Finally, autonomous agents are modeled as DASM agents (see 2.3) where the program of each agent characterizes the rules governing its behavior, and its memory and motivations are abstractly represented by functions. Table 5.1 illustrates the entity mapping through the three different layers.

5.3.2 Agent Architecture

The central component of our model is an autonomously acting entity, called a *person agent*, which represents an individual living in an urban environment and commuting between activity nodes, such as home, work, and recreation locations. Here we mainly focus on the criminal behavior of a potential offender, but, in general, person agents *navigate* within the environment and may assume different roles such as offender, victim, or guardian (e.g., police officers), depending on which they exhibit different behaviors.

The agent architecture presented here provides a robust yet flexible design to capture different aspects of an individual's behavior. Intuitively, it is based on a Belief-Desire-Intention (BDI) model providing a structural decomposition of the behavior into different logical components as illustrated in Figure 5.3. Each component captures certain aspects of the overall behavior following the classical Divide and Conquer approach.³ The proposed architecture has gone through several iterations and has been tested in various scenarios. It has proven to be scalable and robust, as well as flexible for future extensions and for accommodating other application contexts (see 5.5) for a detailed discussion).

An agent's personal attributes and preferences are represented by the *profile*. The profile is a repository of all the factors that are specific to an individual agent and have an impact on the behavior under study. These factors include agents' skills, activity nodes, or demographic factors such as age and sex.

³In our Divide and Conquer approach to model criminal behaviour of intelligent agents we emphasize a clear separation of concerns. We effectively break a complex computational problem into a number of individual sub-problems that can be addressed separately. Each sub-problem is defined as a module focusing on a specific aspect of the behavior, such as navigation or target selection. The solutions to the sub-problems are then combined to provide a solution to the original problem.

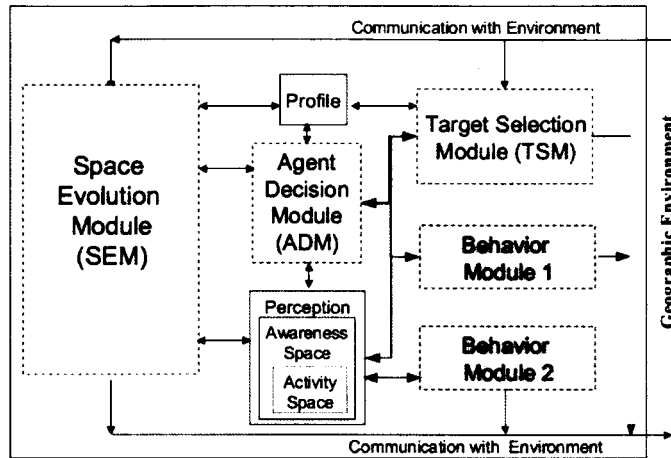


Figure 5.3: The architecture of a person agent

To model the urban environment, we follow extant theories of environment in behavioral sciences and divide the environment into two broad categories: the *objective* and the *subjective* environment [138]. The objective environment is external to an agent and encompasses the totality of all things jointly forming the physical reality. The subjective environment, on the other hand, refers to *perceptions*; i.e., a filtered view of the objective environment as an agent perceives it. The perception is modeled as a memory-like repository that is constantly updated as the agent moves in the environment and commits crime. An agent's perception is further divided into two sub-categories [179]. The part of the perception that an agent is aware of through current events, past experiences and interaction with other agents forms the *awareness space* of the agent. The *activity space* is the part of the awareness space that the agent has visited more frequently over a recent period of time. The agent typically has very detailed information about this part of the environment.

A person's navigation behavior is modeled by the *Space Evolution Module (SEM)*. It provides a navigation algorithm to move the agent from an origin to a destination considering the particular preferences of the agent. These preferences reflect the importance of different factors in navigation for different types of agent. For instance, teenagers have different priorities in finding their paths compared to working adults. The SEM is also responsible for recording the paths frequently used by agents which in turn leads to formation of their

activity spaces and awareness spaces.

The *Agent Decision Module* (ADM) captures the decision making process that sets the goals of the agent. This includes basic decisions such as selecting the next destination based on the personal preferences of an agent. In other words, the ADM decides on ‘what to do’, and then relegates the decision to other modules on ‘how to do it’. While the ADM may reflect a very simple decision making behavior, it is designed as an interface for incorporating complex intelligent decision making behaviors using existing AI methods in our model.

The criminal behavior of offenders is captured by the *Target Selection Module* (TSM). This module works in parallel with the SEM to monitor potential targets on the familiar pathways, and to select attractive targets. Targets are selected based on agent-specific selection criteria and also an agent’s propensity to commit crime. The TSM carves out the *crime occurrence space* of an agent at the micro-level, which leads to formation of *crime patterns* at the macro-level.

We would like to emphasize the flexibility of this architecture for adding additional behaviors to the model. For instance, a module can be added to model social interactions between agents that lead to formation of *social networks*. Several factors such as common spatio-temporal aspects or common criminal goals may be considered in the evolution of social networks and captured by the module. Similarly, it is possible to enrich the behavior of different types of agents; e.g., victims can take a more active role by adding a behavior module to model their interaction with the environment.

5.3.3 Urban Landscape Model

We abstractly model the physical environment as representing some urban landscape with an *attributed directed graph*. This model potentially includes everything from road and rail traffic networks to walkways and hiking trails. In principle, it may also extend to the layout of public spaces such as shopping malls, underground stations, and even airports and seaports. In the following, we concentrate on street networks, although the same modeling approach applies to virtually any type of urban traffic and transportation system. We gradually define the physical environment model in several steps as follows.

Let $H = (V, E)$ be a directed connected graph representing the detailed street network of some urban area as specified by a city map or, more adequately, by a Geographic Information System (GIS). Let $V = \{v_1, \dots, v_n\}$ be the set of vertices representing the intersections and other distinguished points of interest located on or next to a road, such as highway exit and

entry points, gas stations, recreational facilities and shopping centers. Further, let $E \subseteq V \times V$ be the set of directed edges representing the identifiable road segments; unidirectional road segments are represented by a single edge and bidirectional ones by a pair of oppositely directed edges connecting the same two vertices.⁴

For labeling the edges and vertices of H , let Θ_e and Θ_v denote two disjoint sets of labels, called *edge attributes* and *vertex attributes* respectively. Θ_e splits into two disjoint subsets: Θ_e^{stat} and Θ_e^{dyn} . Θ_e^{stat} consists of edge attributes that are statically defined such as distance, road type and speed limit. Θ_e^{dyn} consists of those attributes that may change dynamically depending on various factors; e.g., weather phenomena can affect road conditions, time of the day affects traffic conditions, and special conditions may exist on a road, like blockages or closures due to construction work.

In contrast, vertex attributes specify information on locations and characteristic features, such as geographic coordinates and highway exit numbers, as well as other, more specific information related to points of interest.

Next, we define the *geographic environment* as an attributed directed graph $G_{GeoEnv} = (H, \psi)$ by associating a non-empty set of attributes with each of the vertices and edges of H . We therefore introduce a labeling scheme $\psi = (\psi_v, \psi_e)$, with $\psi_e = (\psi_e^{stat}, \psi_e^{dyn})$ consisting of three finite mappings as follows:

1. $\psi_v : V \rightarrow 2^{\Theta_v}$ assigns a finite set of vertex attributes to each vertex in V .
2. $\psi_e^{stat} : E \rightarrow 2^{\Theta_e^{stat}}$ assigns a finite set of static edge attributes to each edge in E .
3. $\psi_e^{dyn} : E \rightarrow 2^{\Theta_e^{dyn}}$ assigns a finite set of dynamic edge attributes to each edge in E .

Figure 5.4 illustrates the representation of the geographic environment for a simple example consisting of two interconnected points of interest.

G_{GeoEnv} represents the objective urban environment—the physical reality—and serves as the basis for defining an agent’s subjective perception of this environment (see 5.3.2). We model perception by introducing an additional labeling on top of G_{GeoEnv} . The fact that, in general, each agent perceives the geographic environment differently implies that each agent also sees different agent-specific attributes associated with certain edges and vertices of G_{GeoEnv} .

⁴Refining the granularity, one may also represent the individual lanes of a given street network in exactly the same way.

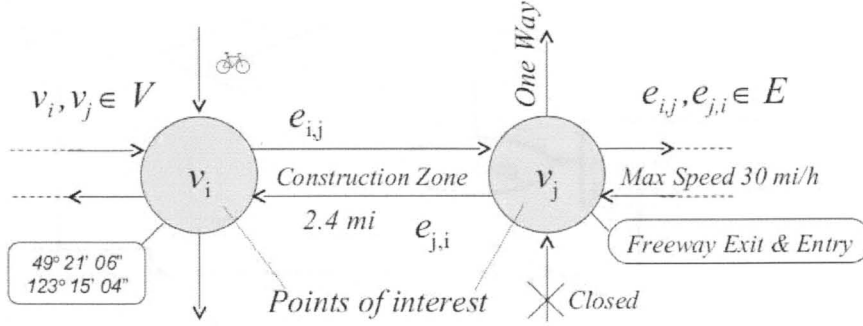


Figure 5.4: Geographic environment

Let Λ_v and Λ_e denote two additional sets of labels for vertices and edges respectively. The urban environment, integrating both the objective environment and the subjective environment for each of the agents, is defined as an attributed directed graph $G_{Env} = (G_{GeoEnv}, \lambda)$ where $\lambda = (\lambda_v, \lambda_e)$ abstractly represents the agent specific labeling of vertices and edges by means of two injective mappings as follows:

- $\lambda_v : AGENT \times V \rightarrow 2^{\Lambda_v}$, for each agent in $AGENT$ and each vertex in V , yields a non-empty set of vertex attributes, and
- $\lambda_e : AGENT \times E \rightarrow 2^{\Lambda_e}$, for each agent in $AGENT$ and each edge in E , yields a non-empty set of edge attributes.

G_{Env} can be seen as a attributed directed graph with *colored attributes*. Each color refers to the specific perception of an individual agent. Λ_v , for instance, specifies the frequency of visits to a location, intensity of activity as well as the agent’s subjective interest in this location. Λ_e , for instance, specifies the frequency of using a road, and intensity of activity.

Finally, the awareness space and activity space of each agent in any given system state is computed from the abstract representation of the urban environment by means of operations on G_{Env} that extract a subgraph with edges and nodes with an associated intensity above a certain threshold. Targets are located on edges or vertices, hence the opportunity space for a certain type of crime is the set of edges and vertices on which the respective type of target is located. Likewise, the crime occurrence space of an agent for a certain type of crime is a subset of the intersection of the opportunity space and the activity space of an agent.

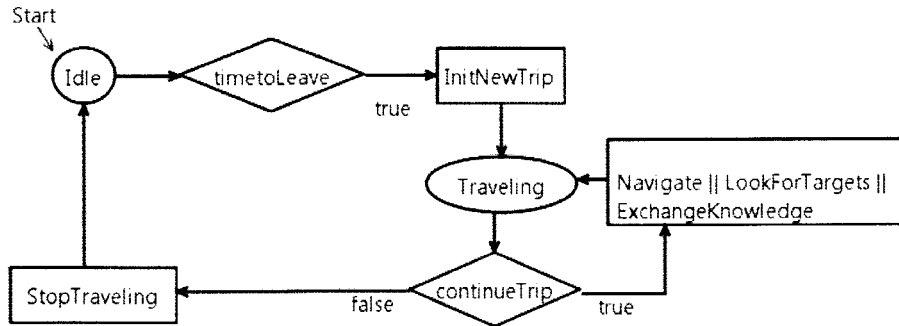


Figure 5.5: Control state diagram of the program of a DASM person (offender) agent

5.3.4 Person Agent

We formally describe the behavior of an offender by further refining the person agent (see 5.3.2) and defining the interaction between its different behavioral components in terms of an ASM rule. The control state diagram of Figure 5.5 defines the program of a DASM person (offender) agent. The control states distinguish different *modes* of operation. At every step, the person agent is in a certain control state which defines its behavior. Transitions between control states occur when the conditions stated by the respective *guards* hold. *Update rules*, shown in rectangular boxes, define the ASM rules that update the underlying data structure.⁵

A person agent is in the *idle* mode until it decides to start a new trip. This decision, which can be triggered by the ADM or any external event, is abstractly modeled with the *timeToLeave* predicate. When it is time to leave, a new trip to a destination must be started. This requires adopting the new destination and setting other preferences for navigation which are encapsulated in the *InitNewTrip* rule. The mode then switches to *Traveling*. At any step during traveling, the agent may decide to interrupt the trip or change the destination captured by checking the *continueTrip* predicate. While the predicate is *true* the agent will continue navigating the environment. At the same time, the agent may also look for targets to commit crime or exchange knowledge with other agents in its group or social network. When the trip is ended, the mode changes back to *Idle* where the agent rests until the next trip.

The following CoreASM code shows how the control state ASM of the person agent is

⁵The || symbol is used to indicate ASM rules running in parallel.

directly translated into CoreASM code. This example also exemplifies the freedom of abstraction in the ASM method; abstract functions and rules are freely refined at different levels and details are added as needed. For instance, here we refine the `Navigate` and `LookForTargets` rules by using the respective SEM and TSM modules. However, the `ExchangeKnowledge` rule is intentionally left abstract, and may further refined in later modeling iterations.

```
rule PersonMain =
  par
    if mode(self) = Idle then
      if timeToLeave then
        par
          InitNewTrip
          mode(self) := Traveling
        endpar
      if mode(self) = Traveling then
        if continueTrip then
          par
            Navigate
            LookForTargets
            ExchangeKnowledge
          endpar
        else
          par
            StopTravelling
            mode(self) := Idle
          endpar
        endpar
      endpar
    endpar
```

```
rule Navigate = SEMMain(spaceM(self))
```

```
// The Navigate rule is refined by passing the execution to the SEM module.
```

```
rule LookForTarget = TSMMain(targetM(self))
```

```
// The LookForTarget rule is refined by passing the execution to the TSM module.
```

```
rule ExchangeKnowledge = skip
```

```
// The ExchangeKnowledge rule is left abstract.
```

Similarly, abstract functions and predicates are refined at different levels and using different approaches. For instance, the *continueTrip* predicate is defined to be true if an error occurs in navigation (e.g., there is no path to a chosen destination), or if the agent has reached its destination (which is determined by the SEM agent), or if the trip has taken too long and the agent changes its mind. While the first two options are concretely defined in the SEM module, the *tripTookTooLong* predicate is modeled using a probabilistic distribution.

```
rule continueTrip =
```

```
  return r in
```

```
    r := not (errorOccured or destReached or tripTookTooLong)
```

5.3.5 Navigation

An agent's navigation behavior is a centerpiece in the context of the Mastermind project. It is important to have a robust and flexible model of navigation behavior that reflects natural and intuitive choices a person makes while moving in an urban landscape. In the following, we explain the role of the Space Evolution Module (SEM) in more detail and describe our proposed path finding algorithm.

Space Evolution Module: ASM Model

The main responsibility of the SEM is to model how a person agent navigates the urban environment G_{Env} during the course of his or her daily routine activities. Intuitively, the SEM moves a person agent in discrete steps from his or her current position on the graph—a vertex or an edge as identified by functions *currentNode* and *currentEdge*—to the destination. It also keeps track of the places visited by the agent leading to the evolution of agent's

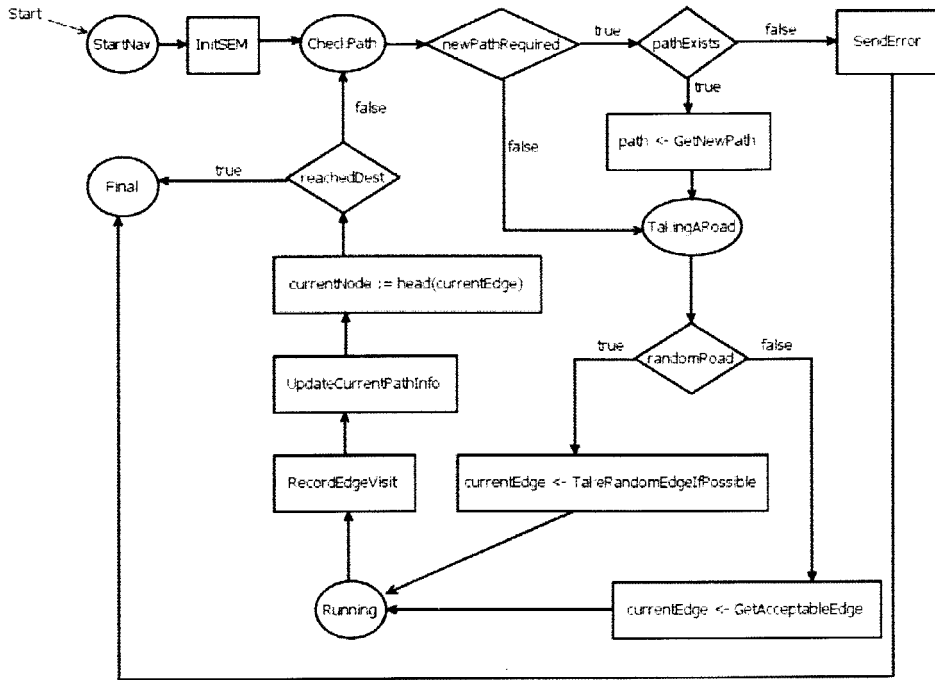


Figure 5.6: Control state diagram of the Space Evolution Module (SEM)

activity space, thus affecting the attribute values of G_{Env} . Such attributes are accessed and manipulated through operations on the graph structure.

The SEM model presented here has gone through several iterations in order to capture different variations of agent navigation behavior in a flexible and robust manner, and these will be outlined below. Abstractly speaking, given an origin and a destination, the SEM finds a *potential* path that reflects the specific preferences of the agent. It then moves the agent on this path, traversing one road segment (edge) at the time. However, at anytime, due to a variety of reasons the agent may divert from this path, e.g., deciding abruptly to take a random road, or being forced to take an alternate road due to the traffic. At that point, the SEM is required to re-route the agent toward the destination by finding a new path. The trip ends when the agent reaches the destination.

We formalize the operation of the SEM in terms of a control state ASM, as illustrated in Figure 5.6. The operation of the SEM starts in the *StartNav* mode, where the SEM is initialized and the mode switches to *CheckPath*. In this mode, the SEM first checks if a

new path is required (*newPathRequired*), i.e., a new trip is started, or the previously found path is no longer appropriate for any reason. In that case, a new path to the destination is chosen. It is important to note that, at this point, we abstract away from the details of the specific navigation algorithm used for path finding and add the details incrementally in further refinement steps.⁶ When a path to the destination is found, the SEM mode changes to *TakingARoad* where a road is chosen to be traversed by the agent. Here, the roads on the suggested path are examined on the fly and several possibilities are considered: (1) the agent may decide to take a random road, or (2) the SEM attempts to take the next road on the existing suggested path to the destination. However, this road may be *not acceptable* due to local conditions such as traffic or construction. Thus, (a) the next road on the path is taken if the local conditions are acceptable, or there is no other alternative; (b) otherwise, an alternate road is chosen instead. We leave the details on how to choose an alternate road abstract: it could be refined to simply choose a random road, or a more complex approach may be used. Once the road is chosen, the mode switches to *Running* where the agent is essentially moving on the selected road. If the agent reaches the destination, the mode changes to *Final*. Otherwise, the mode switches back to *CheckPath* to continue the agent's travel to the destination.

In order to exemplify how the abstract ASM model of Figure 5.6 is refined, we present the refinement of the *GetAcceptableEdge* rule as formalized below. The first step is to take the next edge available on the existing path and check its local conditions. If the conditions are acceptable then no further search is required; otherwise, an acceptable alternative must be found. This operation can be refined into a simple random choice among all the outgoing edges, or a more complex one as presented here. When finding a viable alternative, it is important to take into account that the current unacceptable edge may be the only way to the destination. Therefore, if the search for an alternative leads to selecting the same edge again, the edge must be taken regardless of bad conditions, such as heavy traffic.

⁶This path, called *suggestedPath*, is calculated using the information that the agent has about its environment. This may include using a map, or referring to the memory, if the destination is previously known.

GetAcceptableEdge(me) \equiv

```

return  $res$  in
  let  $e = \text{GetNextEdgeOnPath}(me)$  in
    if  $\text{acceptable}(e, me)$  then
       $res := e$ 
    else
       $res := \text{FindAcceptableAlternative}(e, me)$ 

```

FindAcceptableAlternative(e, me) \equiv

```

return  $r$  in
  if  $\text{attemptedBefore}(e, me)$  then
     $r := e$ 
  else
    par
       $r := \text{TakeRandomEdgeIfPossible}(me)$  // Same edge is used if no alternative exists
       $\text{attemptedBefore}(e, me) := true$ 
    endpar

```

Path Finding Algorithm

The algorithm used for path finding by the agents needs to capture natural and intuitive choices a person makes while moving in an urban landscape. The path taken might not be a globally optimal and best one, but a more natural and good-enough one. In collaboration with the domain experts we have identified key factors that are known to influence human path planning. These factors include *global* (and typically static) elements such as distance, and *local* (and typically dynamic) factors such as traffic. These factors work as proxies through which a person agent perceives the environment.

During the course of the Mastermind project, different models of path finding have been developed and validated through experiments and discussions with the domain experts. Here we briefly describe different phases in the evolution of the path finding algorithm into its current version.

First Iteration: Exploration & Learning

In the first version, a sophisticated algorithm was designed to model path finding as a combination of *exploration* and *learning*.

The exploration algorithm provides a *model-based reasoning* mechanism, and uses a number of *influence factors* which play an integrated role in influencing the path selection process. They include distance, road type, number of intervening stops, angle toward the destination, traffic, road condition and familiarity. On the other hand, learning is modeled by developing a *case-based reasoner* which facilitates using agent's past experience in path finding. As such, in this version of the algorithm the `GetNewPath` rule (see Figure 5.6) was refined as follows:

```

rule GetNewPath(me)  $\equiv$ 
  seq
    suggestedPath(me) := GetPathMemory(me, currentNode(me), destNode(me))
    if suggestedPath(me) = undef then
      suggestedPath(me) := GetPathExplorer(me, currentNode(me), destNode(me))
  endseq

```

The agent would first try to find a path using its past experiences that are stored in memory. If the destination is new and there is no related experience in the memory to help with path finding, path finding is done by exploring the environment.

Generally, a person agent has some global information about the environment. The exploration algorithm uses this global information to perform *global planning*. On the other hand, local information discovered on the fly is also considered to perform *local re-planning*. This allows the agent to examine changes in highly dynamic factors such as traffic and road condition and revise its decision accordingly.

Technically, the algorithm is a variation of the Dijkstra's shortest path algorithm [68]. Assume a person agent wants to move from source S to destination D . The preference of an edge $e = (S, B)$ for the agent me that chooses the path is defined as:

$$edgePref(me, e, D) \equiv$$

$$globalWeight(me) * globalEdgePref(me, e, D) +$$

$$localWeight(me) * localEdgePref(me, e, D)$$

where

- $globalEdgePref(me, e, D)$ yields the preference of taking a 'best' path from S to D via

B considering agent's global knowledge of the environment (global planning). A best path is computed using Dijkstra's shortest path algorithm.

- $localEdgePref(me, e, D)$ yields the preference of choosing e considering local information on the edge (local re-planning).

The effect of local re-planning and global planning in the overall preference of an edge is controlled by the weights assigned to each type of preference, namely $globalWeight$ and $localWeight$.

Path finding using the memory, on the other hand, follows a hierarchical problem-solving technique. First, the person tries to find a complete path in the memory that takes him/her from the source to the destination; if no such path exists, the person relies on memory to get a partial path that takes him/her *close enough* to the destination and then uses a map (exploration) to find the rest of the path. Failure to find any such path in the memory would prompt the explorer to take over. This process is formally defined as follows:

```

rule  $GetPath_{Memory}(me, s, d) \equiv$ 
  let  $pathCBB \leftarrow GetPath_{CBB}(me, s, d)$  in
    if not ( $complete(pathCBB)$ ) then
      // The path is not to the final destination, but a close enough one
      let  $pathEXP \leftarrow GetPath_{Explorer}(me, tail(pathCBB), d)$ 
        return  $Concat(pathCBB, pathEXP)$ 
    else
      return  $pathCBB$ 
where
   $complete(p) \equiv tail(p) = d$ 

```

This approach was refined and fully implemented in AsmL [176]. The same approach was implemented and validated by running experiments using the Java version of Mastermind. Simulation results closely followed the behavior expected by established theories; however, the simulation model was too intimidating for criminologists to be used as a tool in experimentation. The complexity of the algorithm and the interdependence of the underlying factors hindered their confidence in the model. In other words, the algorithm was seen as a black box that produced results not suitable for peer-review and precise analysis.

The Java version of Mastermind and respective validations marked the end of one major

iteration of the modeling process. Running simulation experiments and our continued interaction with our criminologist partners shed light on a new set of open research questions. Despite the rich functionality offered by this version of Mastermind, the complexity of the model hindered its use for answering some fundamental questions about different types of navigation. Furthermore, it became clear that developing a deeper understanding of the exploration behavior is more crucial as the first step in studying underlying crime patterns. Hence, the subsequent iteration of the path finding algorithm focused on these issues, as explained below.

Second Iteration: Mobility Styles

Our experience in the first iteration indicated the need for more systematic ways of modeling path finding behavior which are more tailored towards crime analysis needs. As such, we focused on simplified path finding algorithms by separating different concerns. Instead of integrating various complex aspects of navigation, these algorithms may only deal with one aspect of path finding behavior to analyze their respective impact on crime patterns. In this phase, we developed experiments using the CoreASM version of Mastermind. The CoreASM program code is easier for a non-specialist to read, and it is well-suited for designing controlled experiments.

We identified three categories of path finding approaches, called *mobility styles*: (1) *pre-determined* where an agent always uses the same path (usually the optimal one) between two nodes without any divergence, (2) *random walk* where edges are selected completely randomly, and (3) *mixed* (called *tear-drop*⁷) where an agent may divert from its pre-determined path by choosing a random road, but will continue on another pre-determined path from there.

We then focus on different path finding factors, such as distance, travel time, and type of road, separately. The algorithm used for finding a path that optimizes each of these factors is based on Dijkstra's shortest path algorithm [68]. Factors such as angle to the destination can also be incorporated into path finding using the A*-algorithm [174].

It is important to point out the robustness of the SEM model in accommodating this new approach. Modeling different mobility styles did not require any changes to the SEM model of Figure 5.6. Instead, each style can be modeled by properly refining the abstract

⁷The name refers to a pre-determined path with tear-drop perturbations

function *randomRoad*, which determines whether or not to take a random road instead of following the suggested path. For instance, modeling a completely random walk requires refining this function such that it always returns *true*, whereas in the tear-drop style the result is determined using a probability distribution.

Despite the simplicity of this approach, the results of our experiments using CoreASM have led to interesting observations and proven useful for satisfying criminology queries, which are discussed in more detail in Section 5.3.7.

5.3.6 Target Selection

The behavior of an offender in a physical environment is influenced by a number of social, spatial and temporal factors. In Section 5.3.5, we presented an ASM ground model focusing on movement patterns of a person, modeled as an agent, in an urban environment. While the activity pattern of an agent is a major element in crime analysis, offence occurrence behavior, called *target selection* behavior, is of equal importance in understanding crime patterns. Here, we briefly describe how this behavior is defined in terms of an ASM ground model. The model presented here is a refinement of the *Target Selection Module (TSM)* introduced in 5.3.2.

One of the challenges of building target selection into Mastermind is abstraction of the crime process such that it is applicable to a wide range of crime types. For our purposes, we are interested in crime that takes place in the course of daily activities. We model what is suggested by Crime Pattern theory; i.e., crime occurs when a motivated individual encounters a suitable target [41]. This can either be the entire offence if an agent acts immediately upon noticing a target, or simply the moment of awareness that leads to a more complex plan.

Figure 5.7 shows the high level control state ASM of the TSM, where the key aspects of the behaviour are captured while technical implementation details are left abstract.

The target selection process starts in the *Idle* mode. Agents come across opportunities, and potentially commit crime, in the course of daily routine activities. Thus, the target selection procedure becomes active when the agent is travelling between nodes. As such, the *moving* predicate captures the interplay between the target selection and the navigation processes. If this predicate is true, the mode changes to *CurrentAction*. In this mode, if there is a *triggeringEvent* to persuade the agent to commit a crime, the agent will search for a suitable target, and the mode changes to *SearchCompleted*. The *triggeringEvent* predicate

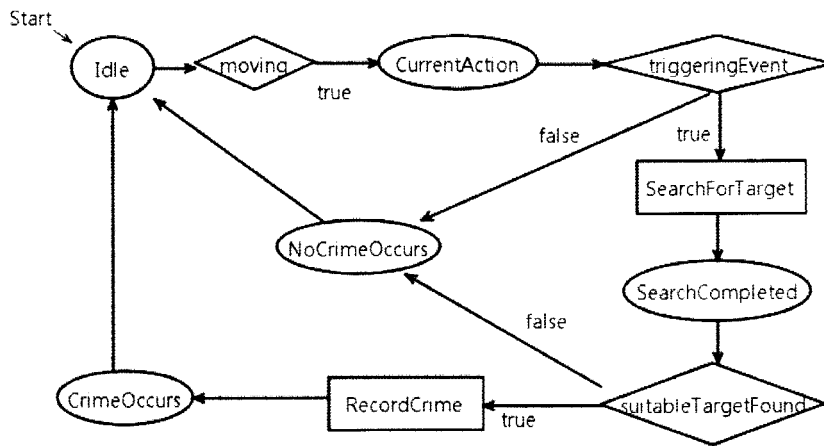


Figure 5.7: Control state diagram of the Target Selection Module (TSM)

abstractly captures the complex process that triggers criminal behavior in an agent. A number of factors are important in determining the result of this process, including the agent's propensity to commit crime or the existence of guardians (such as police officers). In the *SearchCompleted* mode, the agent checks if a suitable target is found (*suitableTargetFound*), as a result of which a crime has occurred.⁸ If no suitable target is found, the mode changes to *NoCrimeOccurs*. Once one round of target selection process is complete the mode changes back to *Idle*.

It is important to emphasize one of the key advantages of using ASMs in modeling the target selection behavior: separation of design and implementation. For instance, the *triggeringEvent* predicate is an abstract representation of a potentially complex process that can be implemented in different ways; in the CoreASM version of Mastermind, this process is refined using probabilistic methods, while in the Java version Fuzzy Logic [201] is used to determine the viability of a potential target. Classical logic could also be used to implement this component. However, at a higher level of abstraction these divergent decision-making technologies are equivalent from a practical point of view. Each of them is able to take the information given to them and make an evaluation about the present situation. Hence, different technical approaches can be attempted as solutions without requiring the design

⁸The crime may not be committed immediately. Instead, the identification of a suitable target may lead to a more complex plan to commit crime later.

to change.

The TSM can also be extended to consider the impact of social interactions on crime. Options for criminal targets are not only developed through direct personal experience, but also by knowledge gained through social interaction [65]. This aspect is not explored in the current version of Mastermind and is subject of future work.

5.3.7 Experiments

We have run a number of experiments using both versions of Mastermind. The Java version of Mastermind was used to study differences in criminal behavior of agents with different profiles (e.g. goal-oriented, wanderer, etc.). The simulation results are reported in [46]. The scale of these experiments has not been large enough to declare the results as definitive. But they definitely provided food for thought, led to a better understanding of the simulation needs, and spurred on the configuration of following experiments using CoreASM. The Java version is currently used as the basis of a comprehensive tool environment for counterterrorism planning and response (see 5.5).

On the other hand, the CoreASM version has proven to be extremely useful in the iterative process of exploration, validation, and establishing the right level of abstraction for the model. In CoreASM simulations we focused on different key elements of the model separately, and studied the influence of these elements on the overall behavior. In a series of experiments, we focused on three different styles of movement in the environment: (1) a pre-determined, deterministic (completely non-random) style using Dijkstra's algorithm; (2) a random walk; and (3) a variation of Dijkstra with tear-drop perturbations.⁹ The hypothesis was that these different styles of movement would produce different patterns of crime.

A grid structure was used for the road network in order to simplify the analysis. Experiments were run on 8×8 , 12×12 , and 16×16 grids. We also explored the impact of new urban developments by connecting some 8×8 grids by a bridge to a companion 8×8 grid. In some experiments, the companion grid contained an activity node. Finally, the agents could have different levels of motivation (e.g., high or low) for committing crime.

In the following we briefly present some of the more interesting results. A more comprehensive analysis of the results can be found in our publication [46, 44].

⁹In our experiments, tear-drop was modeled by having a 50% chance of choosing a random road instead of following the pre-determined path. For random walk, the likelihood of choosing a random road was increased to 95%. This ensures that the agent reaches its destination eventually, albeit in a longer time.

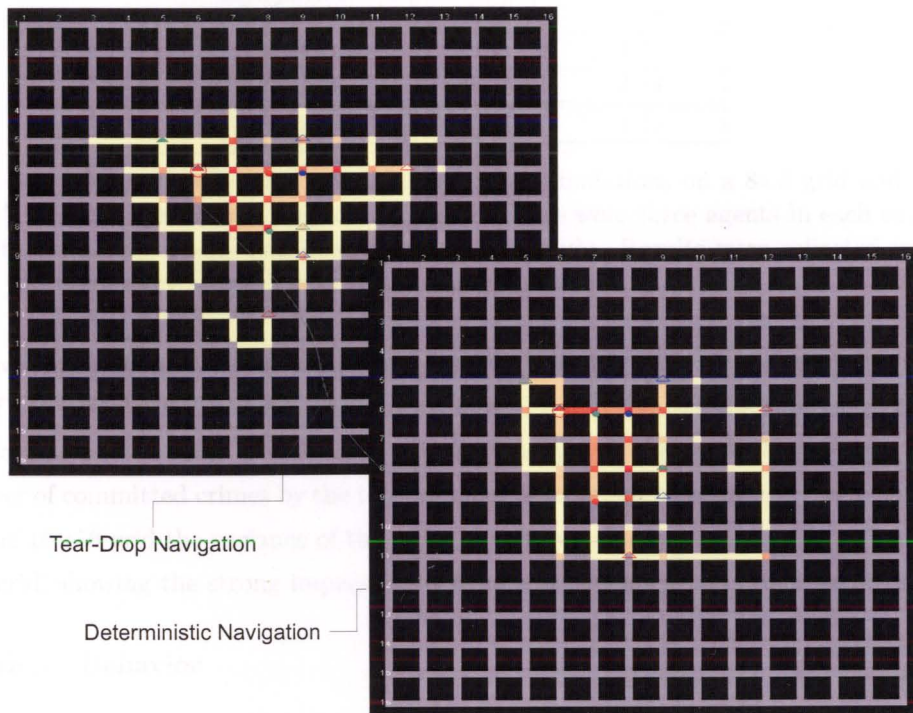


Figure 5.8: On the 16×16 grid, the tear-drop navigation (left) produces distinctly different paths in comparison with the deterministic navigation (right).

Edge Effect

In the first set of experiments, we analyzed the behavior of agents on a 8×8 grid. Contrary to expectations, we found that different mobility styles produced very similar crime patterns on the 8×8 grid. These patterns were not sensitive to changes to activity nodes or motivation levels. As such, the results did not support the hypothesis that mobility styles impact crime patterns. However, when the 8×8 grid was expanded to a 16×16 one, with the same set of variables, differences in the behavior became evident. The directed paths of the deterministic agents were distinctly different from the cloud-like explorations of the agents using tear-drop navigation (see Figure 5.8).

After further analysis, it became clear that the anomaly on the 8×8 grid can be explained with what is known in the Criminology literature as the *edge effect*. The limited movement opportunities, imposed by the size of the grid, create a side-effect that overshadows the role of mobility styles. This observation is important because it re-affirms the key role

Mean Crime Density		
	8×8 Grid	16×16 Grid
Deterministic	6.71	5.98
Tear-Drop	5.35	4.48
Random	5.50	3.94
Variance	0.56	1.11

Table 5.2: Mean crime density as calculated from simulations on a 8×8 grid and a 16×16 grid. The same agents were used on each grid. There were three agents in each simulation, with three activity nodes and one shared activity node. Results were collected from 2000 runs. Motivation levels could vary in different simulations.

of environmental features in crime analysis. On the 16×16 grid, agents (with the same set of parameters) had more options to explore. Table 5.2 shows the summary of different simulations on 8×8 and 16×16 grids. Crime density was calculated by dividing the total number of committed crimes by the total number of locations in which they were committed. On the 16×16 grid the variance of the crime density was almost 2 times the variance on the 8×8 grid, showing the strong impact of *environment boundaries* on crime patterns.

Wanderer Behavior

An interesting question to investigate is whether or not wandering more in the environment would increase crime activity. The hypothesis is that agents who spend more time moving through the environment are more criminally active. In other words, since the random walk and tear-drop navigation types force the agents to move in a less optimal manner (from a time-efficiency perspective), they should be able to find more opportunities to commit crime.

This question led to an in-depth analysis of the results and the design of new experiments. In this series of experiments, we used 10 different agents with different activity nodes. The behavior of each agent was simulated using all three different mobility styles. Each experiment simulated 30 days with an average of 2 trips made per day and consisted of maximum 30,000 simulation steps.

In the first set of experiments, a very simple target selection process was simulated; targets are uniformly distributed and selected once encountered by the agents. The results showed that the total number of crimes committed by wanderer agents (the ones with some degree of random behavior) is more than the deterministic agents on average; e.g., on a

	Average Number of Crimes
Deterministic	29
Tear-Drop	50
Random	79.2

Table 5.3: Average number of crimes as calculated from simulations on a 12×12 grid using basic target selection. There were ten agents in each simulation. Results were collected from 30,000 simulation steps.

	Average Number of Crimes
Deterministic	12.96
Tear-Drop	13.39
Random	13.44

Table 5.4: Average number of crimes as calculated from simulations on a 12×12 grid considering familiarity in target selection. There were ten agents in each simulation. Results were collected from 30,000 simulation steps.

12×12 grid, the tear-drop navigation produces 1.72 times more crimes than the deterministic style, and the random walk produces 2.7 times more crimes, as shown in Table 5.3.

The above-mentioned target selection mechanism can be refined by further emphasizing the impact of *familiarity* with the environment on target selection. In the next set of experiments, we refined the target selection process to take into account the *intensity* of the location of the crime. The intensity of a location reflects the level of activity (visits) at that location. Hence, agents are more likely to commit crimes at locations with higher intensity; i.e., locations that they have visited more frequently. This scheme introduces more complexity and changes the crime patterns dramatically. The results, as shown in Table 5.4 revealed a much smoother impact for different mobility styles. On average, the number of crimes committed by random walkers and tear-drop path finders were only 3.7% and 3.3% more than deterministic path finders, respectively. This huge difference from the previous experiment points out an important factor in analyzing criminal behavior: although wanderers come across more opportunities for committing crime, in a given time frame, they develop less familiarity with their neighborhood and hence their tendency for committing crime will be lower. Although, this observation does not refute the hypothesis that wanderers commit more crimes, it raises questions about the real impact of random behavior on the level of criminal activity.

We further compared the spatial aspect of crime patterns generated by different mobility

	Deterministic	Tear-Drop	Random Walk
Crime Occurrence Space (COS) (% of the grid)	19.58%	26.11%	27.15 %
Activity Space (AS) (% of the grid)	33.04%	59.03%	91.37 %
Crime Occurrence Space / Activity Space	59.28%	44.24%	29.72%

Table 5.5: Coverage of activity space and crime occurrence space for different mobility styles considering familiarity in target selection. The results were collected from simulations of ten agents on a 12×12 grid.

styles against each other. First, we prune the crime occurrence space¹⁰ of each agent by looking at locations with more than 1% chance of having a crime committed. As shown in Table 5.5, the results confirm the expectation that wanderers have a bigger activity space; i.e., they visit more locations during their daily routine activities. For instance, a random walker on a 12×12 grid (on average) covers around 91% of the grid. However, this coverage is not as strong when we look at crime occurrence space; e.g., for the same random walker crime locations only cover 27% of the grid. Hence, while around 59% of the activity space of a deterministic path finder is covered by crime locations, for the random walker this number is only 29%, hinting that the criminal activity of the random walker is not as widely spread as his/her navigation.

In order to further compare the patterns created by each style, we looked at the intersection of crime occurrence spaces; i.e., for each set of experiments, we specifically looked at those crime locations that were shared by each pair of styles. The results, as shown in Table 5.6, do not show a strong similarity between patterns, confirming our hypothesis that different mobility styles do create different crime patterns.

Micro vs. Macro Analysis

We would like to emphasize the power of the Mastermind simulation model in breaking down the complexity of the system and providing means of analysis both at a micro-level and a macro-level. The design of the model and the simulation environment allows for analyzing the behavior of one single agent under different circumstances, e.g., different mobility styles, or changes in the urban environment. At a macro-level, it is possible to

¹⁰Crime occurrence space of an agent is a set of locations where the agent has committed a crime.

	Number of shared crime locations of A & B / Number of crime locations of B $\ COS(A) \cap COS(B)\ /\ COS(B)\ $ * *COS: Crime Occurrence Space
A = Deterministic B = Tear-Drop	59.1%
A = Tear-Drop B = Random Walk	49.02%
A = Deterministic B = Random Walk	35.73%

Table 5.6: Analyzing the similarity of crime occurrence spaces created by different mobility styles. The results were collected from simulations of ten agents on a 12×12 grid.

aggregate these results in several different ways. For instance, one can overlay all crime occurrence spaces created by different agents to define the overall crime pattern. However, our experiments showed that in this form of analysis understanding the impact of different variables become too complex; e.g., different agents have different activity nodes whose location strongly impact the overall pattern of the crime and can overshadow the impact of other important factors such as mobility style when studying aggregate patterns. In our experiments, we explored different ways of linking the macro-level behavior to macro-level patterns. A proper analysis approach must be selected based on the requirements of the question at hand. However, it is clear that the simple and systematic break-down of the behavior provided by our model and supported by the simulation environment facilitates dealing with the complexity and allows for systematic exploration of macro-level behavior in a manageable manner.

5.4 Lessons Learned

Mathematical modeling of criminal activity in the form of discrete event models that define the cooperative behavior of multiple autonomous agents in abstract operational and functional terms has proven to be a sensible choice in the cross-disciplinary research context of the Mastermind project. The task of reasoning about and analyzing complex crime activity patterns and their representation in computational form requires an amalgamation of expertise from criminology and computer science. Therefore, developing a coherent and consistent common view, one that has a virtually identical meaning in both worlds, is crucial

in overcoming fatal misconceptions, especially in the transitions between modeling phases (see 4.1).

A typical challenge in the early formalization phases is lack of a thorough understanding of the functional requirements, which can lead to vague descriptions and fuzzy architectural concepts. Real-life events are not usually thought of in a discrete, mathematical manner that would easily transform into something computable. Striving for more clarity and regularity, any encoding should not only be minimal but also be as direct and intuitive as possible, so as to not add extra weight to the overall problem scope. In light of such practical needs, the relatively simple semantic foundation of the ASM formalism contributes a great deal to the ease of using this approach as a precise analytical means for communicating ideas and for reasoning about design decisions. Viewing states as first-order structures, or simply as abstract *mathematical data structures* being manipulated by autonomously operating computational agents, indeed greatly simplifies the formalization task. CoreASM facilitates experimentation and supports design for change by providing an untyped language and a minimal instruction set for describing state transition behavior, combined with flexible extensibility and refinement techniques. Finally, the ability to freely choose and mix common modeling styles, e.g., declarative, functional and operational, depending on what appears most suitable in a given application context, is invaluable.

An important part of the modeling exercise is identifying the right level of granularity required for modeling behavior at the micro-level and investigating the impact on the macro-level behavior patterns. To facilitate this process, we have used CoreASM to run experiments in very early stages of design. Through these experiments, we have been able to identify key elements which affect the macro-level patterns of behavior, but are often left unnoticed at the micro-level. For instance, the specific method used by agents for finding a path to the destination (e.g., completely deterministic vs. random walk) is generally expected to have a huge impact on macro-level crime patterns. But the experiments have shown that the impact of the *boundaries* (or restrictions) imposed by the environment, such as the size of the road network, could be even stronger than individual path finding preferences. Such results re-affirm the benefits of computational models in developing and testing theories of crime.

The Java version of Mastermind is capable of simultaneously processing the daily activities of multiple agents whose activity spaces and selected targets are displayed on-screen. With its recognizable landscape and dynamic agents, this simulation model turned out

to be particularly effective at illustrating concepts in Environmental Criminology to non-criminologists [41]. As such, it provides the core of a full-fledged decision support tool to be used by police officers and crime analysts. As discussed in 5.2 the architectural design of Mastermind allows for integration with offender databases as well as enhanced 3D visualization features based on Geographic Information Systems (GIS). This direction is currently followed by using Google Earth [2] visualization features and is further discussed in the next section on the future work.

The complexity of the Java version and the fact that it is considered as a black box by domain experts introduces limitations on its academic usage. Even though the behavior of the agents *appeared* to follow established theory, domain experts could not be confident that the program semantics followed their understanding of the phenomena. Furthermore, if the behavior of the program is not clearly explained, the produced results are not useful in an academic publication. The CoreASM program code is easier for a non-specialist to read, and it is well-suited for designing *controlled* experiments.¹¹ The results of our experiments using CoreASM have been more focused and useful for satisfying criminology research queries. Furthermore, taking advantage of the highly flexible plugin architecture offered by CoreASM we were able to rapidly develop the Mastermind Plugin to address the specific needs of criminologists, especially with respect to visualizing the results. In other words, the Mastermind Plugin encapsulates the mathematical structure of the ASM model in a comprehensible and familiar format for domain experts. This greatly facilitates communication with domain experts and analysis of the results for validation purposes.

Mastermind has been central to the creation of our framework for computational modeling of complex social systems. It has shown the necessity of a robust and extensible, yet flexible, design that can be easily re-applied in new experiments. It has also illustrated the importance of reducing the communication barriers between team members from different disciplines, facilitating a smooth transition between different modeling phases and validation of intermediate models in each iteration.

¹¹The ASM formalism offers much of the freedom that comes with using pseudocode as a design language—just that pseudocode usually does not have a precise (unambiguous) meaning and thus is not executable. For a direct comparison of CoreASM with other specification & design languages, we refer to [134].

5.5 Summary and Future Work

We have adopted a novel approach to computational modeling of crime patterns and theories in crime analysis and prevention. Pattern and Routine Activities theories suggest that through a combination of decisions and innate understanding of environmental cues, likely offenders are able to separate good criminal opportunities from bad risks. The nature of this process is highly structured and allows researchers to anticipate likely concentrations of daily activities and a variety of criminal offences. We model spatial and temporal aspects of crime in urban environments as a foundation for systematic development of simulation models. With the proposed formal model, it is possible to better understand crime patterns, and also to test crime pattern and prevention theory. Our model is designed in such a way that it is scalable and applicable not only to a broad range of crimes, but also to other environments such as airports, ports, shopping centers, and subway stations.

Our main theoretical result is the abstract behavior model of person agents (based on the *agent architecture*) interacting with their objective and subjective environments which jointly form the *geographic environment*. Our main practical result is the Mastermind system architecture which serves as a platform for the construction and experimental validation of discrete event simulation models.

Mathematical and computational modeling of crime serves multiple purposes. It has a direct value in a range of criminal justice applications. For law enforcement purposes, crime prevention interventions can be modeled and analyzed prior to physical implementation. For proactive policing, modeling of crime makes it feasible to build scenarios in crime analysis and prevention and provides a basis for experimental research, allowing experiments that can often not easily be done in a real-world setting. Models such as Mastermind would provide program planners and analysts with another tool to predict likely activity spaces for both typical street crimes and those requiring more focused and long-term investigations, such as those involving serial or persistent (chronic) offenders.

The Mastermind framework provides a scalable, reliable, and extensible platform providing a firm ground for applications beyond crime analysis and prevention, as further described below.

5.5.1 Genius: A Decision Support System for Counterterrorism Planning and Response

In order to better position public safety communities against potential threats, it is of utmost importance to set priorities, identify existing gaps and develop proper approaches to address those. Aiming at improving response capabilities and better organizing counterterrorism activities, national security agencies in Canada have identified different priority areas [1]. One major area is *risk assessment and priority setting*, which focuses on developing advanced tools and techniques for establishing a reliable understanding of threats, and providing consolidated risk assessment and rating of threat scenarios. A well-defined risk assessment approach leads to a systematic analysis of capability gaps and provides guidelines for setting investment priorities such that the most critical gaps are addressed [1].

Computational and mathematical methods have an enormous potential for serving practical needs of counterterrorism initiatives; they offer new approaches and tools for assessing the risks of various scenarios involving terrorist attacks and analyzing the impact of potential responses. The Genius¹² project aims at expanding the Mastermind framework to be used in the analysis of terrorism and counterterrorism. It is built on top of the Mastermind platform and provides an enhanced set of visual components using the Google Earth platform and a 3D virtual environment. It also extends the Mastermind model to capture a more general agent behavior in the environment (e.g., crowd behavior) and to address different forms of threats (e.g., simultaneous occurrence of multiple threats).

The goal is to provide a decision support system for terrorist threat response planning and risk assessment. The model uses spatio-temporal features of the environment and (potentially real-time) threat indicators for risk analysis and real-time situation analysis. The proposed system can be applied in different applications contexts including critical infrastructure protection, dangerous hazard emergency response, and special event security planning, such as the 2010 Winter Olympics in Vancouver.

5.5.2 Modeling Physical Activity and Chronic Diseases in Urban Environments

The Mastermind framework is also being used in a completely different application domain for modeling the impact of objective and perceived factors of the surrounding environment

¹²In Roman Mythology, a Genius is a tutelary deity or guardian spirit of a person or place

on people's physical activity levels.

Physical activity (PA) shows benefits in many aspects of our health, such as reducing the incidence of and mortality from type 2 diabetes and even some forms of cancer. A growing body of research suggests that the built environment plays an important role in, shaping people's behavior, as it can encourage or discourage healthy eating and taking part in physical activity. The built environment is measured by both perceived and objective environmental elements. However, whether perceived or objective measures of the built environment are better predictors of physical activity behaviour is unknown.

The Mastermind-Obesity-PURE Study (MOPS) project aims at applying computational modeling to better understand key factors that impact decision making regarding PA and determine the relationship between perceived and objective environmental measures. Such models would serve as experiment testbeds for identifying how PA is affected by changes to the structure of an environment and to human perception of the environment. They also serve as valuable decision support tools for planners and policy makers by facilitating the analysis of intervention policies.

The project brings together three main components: (1) Mastermind framework on modeling routine activity patterns in urban environment; (2) A comprehensive objective environmental data set collected from 2 different neighborhoods in Vancouver (low income and high income); (3) Individuals' data collected by the PURE study [185], where 358 adults from the two communities were assessed for socio-demographics, physical activity levels, environmental perceptions, and anthropometry.

The modeling exercise has already proven useful in unveiling some hidden assumptions and limitations of the previous studies in the field. The structure and rigor inherent in the formal modeling process enforces logical thinking and precise analysis of the most basic concepts and key assumptions about the underlying system. For instance, the initial phase of the project has clarified a critical assumption made by the majority of studies in the built environment literature; i.e., overall PA levels directly correspond with PA in home neighborhood. Based on that assumption, correlations have been made between the environmental features of a person's home neighborhood and their overall PA levels in a much wider environment. We identify this as a major limitation of those studies. Interestingly, some of the most recent studies in the literature have also started raising the same question, critiquing the assumptions made by conventional approaches [150].

Chapter 6

Identity Management Architecture

The research presented in this chapter was funded in part by the Ministry of Labour & Citizens' Services of British Columbia, Canada.

Identity and *Identity Management* are two key concepts that have been addressed by researchers from different disciplines with different methods in various application domains. Across disciplines, there is a consensus on the vital role of identity management in many strategic applications, including investigation contexts, services provided by governments (e-government), e-commerce, business intelligence, and homeland security; although, the focus in each context is different. For instance, social scientists are mainly interested in the theoretical discussion on 'what is identity?' and 'what constitutes identity?' [161], while in the *digital world* context, the main focus is on 'digital identity' and its required elements [53]. One of the most challenging issues related to identity management is *privacy*. Privacy related requirements impose important restrictions on the design and effectiveness of any identity management system, and hence are of utmost importance [55]. At a more technical level, there are a number of outstanding issues, including resolution techniques [156, 135, 189], centralized, distributed or federated design of systems [195], and differences between *identification* and *anonymous authentication* [53].

Although there have been substantial efforts to address the challenges in each of the above mentioned areas, the reality is that there is no common agreement or understanding on even the basic concepts, such as what constitutes identity or what is identification. In other words, in the absence of a common starting point for research on identity, different assumptions have been used by different researchers in solving specific problems. The lack of such a unifying view has several negative implications including, but not limited to:

- interoperability problems (especially within government or surveillance contexts);
- privacy related issues (e.g. What should be protected? What is ‘personal data’? How to maintain privacy in interchange data format in health applications);
- issues related to reconstructing identities for investigation or profiling purposes (e.g. legitimate vs. illegitimate profiling);
- difficulty of bringing together research results (from different areas) on this topic.

In recent years, the need for a well-designed Identity Management System (IMS) has widely been recognized by different groups working on identity management around the globe. Namely, there have been several initiatives in Europe to address this issue. The study “Identity Management Systems (IMS): Identification and Comparison” [129] provides an overview of the open issues in the field. Future of Identity in Information Society (FIDIS) [98] and Privacy and Identity Management for Europe (PRIME) [159] are two research initiatives by the European Union toward advanced research on identity management. Challenges include establishing proper associations between entities (civilians or institutions) and their identities, matching of identities, detecting fake identities, etc. In the aftermath of September 11th 2001, even more attention has been directed to this area in order to provide governments and intelligence agencies with better intelligence and better tools for identifying and responding to possible threats.

Following the principles of mathematical modeling of complex systems as presented in Chapter 4, and building on our experience with semantic modeling of behavioral aspects of complex distributed systems, such as semantic foundations of Web service architectures [86], in this chapter we focus on developing a firm unifying semantic foundation for a systematic study of identity management and improved accuracy in reasoning about key properties in the IMS design. Like many complex systems of today, identity management systems are influenced by a combination of social and technical aspects. Given the diversity and variety of the concepts and disciplines involved, we argue that mathematical rigor and precision is essential in consolidating existing approaches and harmonizing the sharing and integration of information.

We propose a novel conceptual model of identity along with a simple, but universally applicable, mathematical framework, based on the abstract state machine (ASM) method [30], for establishing a precise semantic foundation for the most fundamental elements of

identity management. This model serves as a starting point for bringing together different approaches to identity management in a more systematic fashion. Through an extensive review of the literature, we also identify key requirements of any IMS (such as privacy, user-control, and minimality), and focus on the practical relevance of developing a distributed, approach to identity management (as opposed to a centralized one). Finally, we illustrate the practicality of our approach by applying the model to a rigorous definition of identity theft.

Section 6.1 presents an extensive review of the identity management literature, summarizing common identity management concepts and terminology. Section 6.2 introduces the formal semantic framework, and Section 6.3 mainly focuses on the application of the model in identity theft. Section 6.4 presents future directions and concludes the chapter.

6.1 Background

Identity, identity management and related issues are studied in different application contexts and across disciplines. Given the variety of disciplines and application contexts involved in the study of identity management, here we explore the existing literature in search of basic definitions and a common terminology. We also review some related technical and non-technical issues such as identity resolution, identity theft, and privacy and trust.

6.1.1 Basic Definitions

The issues surrounding identity, identity matching, and identity resolution are being discussed and studied in different application contexts and across different disciplines. As noted by Camp [53], “the word ‘identity’ refers to an increasing range of possible nouns—from a set of personality-defining traits to a hashed computer password.” A glossary of the terminology used in digital identity systems is provided in [53], which is the result of a cooperative attempt to develop a common understanding of the terminology across several disciplines [54]. This glossary was developed to address the overload of identity terminology and “the lack of conceptual clarity” [53]. Hence, we consider it as a main reference point in designing our model of identity, although some of the terms are still loosely defined and, in some cases, vague. The terms that are introduced and defined in the glossary include: *attribute*, *identifier*, *personal identifier*, *anonym*, *pseudonym*, *identity*, *identification*, *authentication*, *identity authentication*, *attribute authentication* and *authorization*. The complete definition

of these terms is provided in Appendix B.

In [123], Harper provides a slightly different taxonomy of the elements involved in an identity system. *Identifiers* are introduced as the building blocks of identification, and special attention is dedicated to the classification of identifiers into the following groups; *something-you-are*, *something-you-are-assigned*, *something-you-know*, and *something-you-have*. Appendix B provides a closer look at this view.

Most of the work in the domains of digital identity, identity management, and identity matching, however, use a wide variety of terms not necessarily defined by any of the above mentioned sources. For instance, in [59] the authors introduce the notion of “partial identities” which are subsets of a person’s identity and uniquely identify the person. They also used the word “pseudonym” to refer to identifiers of subjects. Although these concepts proved to be useful in modeling an identity management system, they are not introduced or addressed in either [123] or [53].

6.1.2 Identity (Entity) Resolution

The problem of *matching and relating* identities fits under the broader Entity Resolution problem, which is a well-known issue in databases and data mining. Specific to identities, in [135] a matching technique known as identity resolution is described. This approach was originally proposed to address identity matching problems in Las Vegas casinos and is “designed to assemble i identity records from j data sources into k constructed, persistent identities” [135]. It uses a deterministic technique based on expert rules in combination with a probabilistic component to determine *generic* values for identifiers. Generic values, such as the phone number of a travel agency, are so widely used that they can not be relied upon in identity matching. In addition to identity matching, relationships between identities are detected which leads to useful alerts about potentially dangerous relationships between the identities in the system. Identity resolution is sold as an off-the-shelf product by IBM [3] and has been used in several application domains including gaming, retail, national security, and disaster response.

The Artificial Intelligence Lab at the University of Arizona focuses on developing algorithms that automatically detect false identities to assist police and intelligence investigations. Wang et al. [189] propose a record comparison method to address the problem of identifying deceptive criminal identities. The algorithm builds on a taxonomy of identity deception developed from a case study of real criminal deception patterns [190]. In the same

study, criminal deception patterns are categorized into four groups: name deception, residency deception, Date/Place of Birth (DPB) deception, and ID deception. The algorithm focuses on these fields, and the overall disagreement between two records is then defined as the summation of the disagreements between their constructing fields. A supervised training process is used to determine an appropriate disagreement threshold for matching. Hence, there is a need for training data. In [191], a probabilistic Naïve Bayes model is proposed to address the same problem. The model uses the same four features for matching identities, but a semi-supervised learning method is used that reduces the effort required for labeling training data.

Phiri and Agbinya propose a system for management of digital identities using techniques of information fusion [156]. Instead of relying on a single credential, such as PIN number or password, for authentication, they suggest a *multimode credential authentication* involving a combination of a number of credentials. In this approach, a set of credentials is presented by the user and the result of the information fusion process determines the outcome of authentication. This technique, however, requires training data to fine-tune the underlying neural network which performs the information fusion process.

6.1.3 Identity Management Systems

With the growth of the Internet and its wide variety of applications (such as e-government and e-business) in the recent years, the problem of managing several identities has attracted much attention. Several attempts have been made recently to characterize the requirements of such identity management systems (IMS) which addresses the need of both the users and the managers of the identity.

In [129], a study is performed on systems providing user-controlled management of own identities. The study is built on four pillars, (1) basic requirements for IMS, (2) usage scenarios, (3) evaluating identity management applications, and (4) survey of experts' expectations. The definition and requirements of identity are analyzed in three different contexts: social context, legal context and technical context, as shown in Figure 6.1. The figure also shows how our work on Identity Management Architecture fits into these contexts.

For the purpose of usage scenarios, different application domains such as e-Government, e-Commerce, and e-Health are considered. After evaluating a comprehensive list of existing products and prototypes for identity management, the report concludes that “none of the evaluated products meets all elaborated criteria”. Most notable, “are significant deficiencies

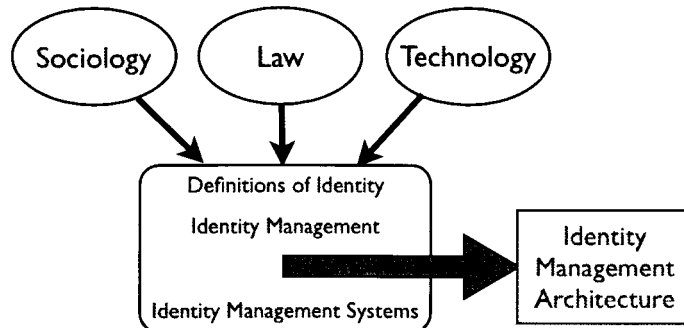


Figure 6.1: Major contexts in defining identity and identity management [129]. Our proposed identity management architecture builds on basic definitions of identity and concepts of identity management to provide a firm semantic foundation for identity management systems.

regarding privacy, security and liability”. Consequently, experts have consistently scored privacy protection, security, and usability as the essential functions of IMS.

The authors of [59] present an approach for developing IMS with respect to multilateral security. They emphasize the role of IMS in realizing privacy laws and its importance for establishing user confidence in e-Commerce applications. They also highlight the lack of *universal or standardized approaches* for IMS. Their proposed system is built on top of SSONET, an architecture for multilateral security, and defines the requirements of an identity manager (including creation and management of pseudonyms, certification and managing credentials, user-control, and privacy). The requirements of the identity manager are developed through analysis of different usage scenarios; however, issues regarding complexity and confidentiality of the identity manager remain open. Furthermore, the proposed identity manager relies on the secure functionalities provided by the SSONET framework.

In [62], the results of the Roadmap for Advanced Research in Privacy and Identity Management (RAPID) project are presented. Identity management is regarded as “definitions and lifecycle management for digital identities and profiles, as well as environments for exchanging and validating such information”. Similar to [59], the notion of *nym*s and partial identities are used in this paper. A nym provides a identity for an individual to operate in a specific environment, such as a user name to be used in a chat room. The paper also

highlights three basic requirements of an IMS: (1) reliability and dependability, (2) controlled information disclosure, and (3) mobility support. The paper explores the obstacles and open problems in developing a system that supports Multiple and Dependable Digital Identity (MDDI) in order to direct future research in this area.

In [169], the issue of rapid changes in the behavior of Internet users and the consequent challenges of dealing with new (and sometimes conflicting) requirements are discussed. The paper analyzes some of these new trends and the respective needs of the end users. One of the important needs of end users is having “multiple identities that are also streamlined and portable”. Portability of identities is a valuable feature, especially for users of reputation-based services such as eBay or Craig’s list. Another issue is preserving privacy in an environment where highly personalized information is revealed in many networking websites. Another problem is dealing with “dispatching” or killing an identity, which is an important part of the digital identity lifecycle. The authors also point to the interesting phenomenon of “generational shift” and how the behavior and needs of the younger generation are different from the previous generation. This shift introduces new challenges for the design of identity management systems, since users are becoming less cautious or concerned about revealing their personal information.

6.1.4 Identity Theft and Identity Fraud

Broadly speaking, identity theft or identity fraud is defined as the misuse of another person’s identity information. With a rapid growth in recent years, it has attracted much attention by the law enforcement agencies in the US and Canada. According to a report by a bi-national working group to the Minister of Public Safety and Emergency Preparedness Canada and the Attorney General of the United States [160], “during a one-year period in 2002-2003, total losses to individuals and businesses related to identity theft in the United States were estimated at approximately US\$53 billion. In Canada the losses for 2002 were estimated at approximately CAN\$2.5 billion”. According to a 2007 report, identity theft loss in the US declined to US\$49.3 billion in 2006, due to an increased vigilance among consumers and businesses [133]. Unfortunately, despite all the attention and public concern, there is little agreement among researchers, practitioners and law enforcement officials on a clear definition of identity theft and identity fraud.

Sproule and Archer address this problem by systematically developing a conceptual process model of the problem domain by consulting different stakeholders [181]. The process

model classifies activities related to identity theft and fraud into different groups which contribute to each phase of the process. The process starts with *collection of personal information* or *creation of a fictitious identity/manipulation of one's own id*. The collected information is then used to *develop a false identity* or is directly used in committing *crimes, enabled by a false identity*.

In [192], a contextual framework for identity theft is presented. The framework identifies the major stakeholders in identity theft domain, the interactions between them and how the information flows. The stakeholders include (1) identity owners, (2) identity issuers, (3) identity checkers, (4) identity protectors, and (5) identity thieves. The paper also classifies the activities important in combating identity theft and identifies the open problems of the field. Such a framework facilitates harmonizing and integrating research and development in the identity theft problem domain, but it does not provide a precise definition of the fundamental concepts of the domain such as identity and identity theft. For instance, the term identity is loosely used in place of what seems to be identity information (or personal information).

6.1.5 Privacy and Trust

The issues of safety, privacy and trust in the digital world, and more particularly on the Internet, are strongly linked to identity problems and the lack of a native *identity layer* [52]. As pointed out by the Information and Privacy Commissioner of Ontario (Canada), “the existing identity infrastructure of the Internet is no longer sustainable. The level of fraudulent activity online has grown exponentially over the years and is now threatening to cripple e-commerce. Something must be done now before consumer confidence and trust in online activities are so diminished to lead to its demise” [55]. One proposed solution is designing a unifying identity metasystem that provides interoperability among underlying identity systems by offering a reliable way to establish who is connecting to what [52]. A set of principles, called “Laws of Identity” [52], are developed through an open consensus process among experts and stakeholders to capture the pragmatic requirements of such a system.¹ These seven laws of identity are

1. User control and consent – identifying information must be only revealed with users' consent;

¹These requirements have proven useful based on practical experience.

2. Minimal disclosure for a constrained use – disclosing the least amount of identifying information and limiting the use of information;
3. Justifiable parties – disclosure of information is limited to justified parties;
4. Directed identity – supporting both “omni-directional” and “unidirectional” identifiers;
5. Pluralism of operators and technologies – allowing for multiple underlying identity technologies;
6. Human integration – protecting against attacks through an unambiguous human-machine communication mechanism;
7. Consistent experience across contexts – guaranteeing a consistent experience while allowing multiple technologies.

It is argued in [55] that such an identity metasystem (based on the seven laws of identity) contributes significantly to improving security and privacy in the online world.

As the issue of privacy is closely related to identification, any attempt of integrating privacy protection into a system must address identity related issues. This is addressed by a joint multi-national project, called Privacy Incorporated Software Agent (PISA) [157]. The project aims at identifying possible threats to the privacy of individuals resulting from the use of agent technology, and demonstrating ways of applying Privacy-Enhancing Technologies (PET) to agent technology in order to reduce the impact of these threats. Even the most basic concepts such as personal data, identifiable data subjects, and identification are essential in defining privacy and identifying privacy threats. These issues and other results of the PISA project are addressed in [187].

6.1.6 Advanced Research on Identity Management

The Liberty Alliance project [142] aims at establishing open standards, guidelines, and best practices for identity management, mainly focusing on identity-based Web services. The project promotes a *federated identity management* approach which focuses on building *trust relationships* between businesses and the ability to *federate* isolated accounts of users among well-defined *trust circles* [141]. The goal is to develop open standards that address interoperability, management of privacy, and identity theft prevention.

Within the European Union, there have been several multidisciplinary initiatives to study digital identities, their management, and related problems. The Future of Identity in Information Society (FIDIS) project [98] aims at integrating research efforts across different European nations focusing on challenging problems such as: (1) supporting identity and identification, (2) interoperability of identity and identification concepts, (3) Identity theft, privacy and security, and (4) profiling and forensic implications. FIDIS defines seven research branches, each of which focuses on one important aspect of identity, such as privacy, interoperability, mobility, etc. One of the main research branches of the project focuses on exploring the definition of identity and identification. This research branch, called Identity of Identity, aims at developing an inventory of definitions in the identity domain and their respective use cases, presenting the existing models of identity management, as well as developing an overview of future directions of such models.

The Privacy and Identity Management for Europe (PRIME) project [159] addresses the lack of identity infrastructure for the Internet, identifies essential requirements such as security and privacy, and aims at defining the right balance of such requirements in emerging identity management architectures. Similar to [52] and [55], PRIME takes a *user-centric* approach to identity management. A high-level architecture for identity management is proposed and is accompanied by the PRIME toolbox and middleware to support and implement user-centric privacy and identity management functionalities.

6.2 The Formal Model

This section presents a precise yet abstract semantic model of identity management concepts, called $\text{IMA}_{\mathcal{AM}}$: *Identity Management Architecture abstract model*. Building on the existing literature, we identify key concepts in the domain of identity management and formalize their intuitive meaning in abstract functional and operational terms. We start with the most basic definitions.

6.2.1 Basics

In principle, the term *entity* as used in the following may either refer to an individual or an organization existing in the real world. However, the focus of this work is on individuals rather than organizations.

Attribute A characteristic associated with an entity, such as name, date of birth, height, fingerprints, iris scan, genetic code, etc. [54, 53].

Identity We define identity as an abstract (mental) picture of an entity, such that, for the identity management concepts considered here, an entity's identity is logically equivalent to the physical presence of this particular entity. In this view, any distinction between an entity and its associated identity is irrelevant.

Partial Identity Any subset of properties (read attributes) associated with users (read entity), such as name, age, credit card number, or employee number, that the user (entity) uses for interacting with other parties [62].²

Identifier An identifier identifies a distinct person, place or thing within the context of a specific namespace [54, 53]. There are two types of identifiers:³

- *Personal Identifier*: A persistent identifier associated with an individual human consisting of one or more attributes that are difficult or impossible to change, such as date of birth, height and genetic code.
- *Pseudonym*: An identifier associated with non-persistent attributes or sets of transactions, with no permanent or personal identifier.

Context A context refers to a specific application domain or circumstance in which a partial identity is defined and has a meaning.⁴ Henceforth, we associate with each partial identity the specific context in which this partial identity is defined.

In the formal model, we regard an *identity* as the abstract representation of the corresponding *entity*. All the *attributes* of the entity then help in defining its identity. Therefore, the main building blocks of our model are as follows:

²In reality, one commonly uses a combination of characteristics in order to distinguish an entity from other entities, so that it becomes identifiable based on a certain set of attributes; however, it seems virtually impossible to find any such set that is generally suitable as a placeholder for an entity's identity in an absolute sense as assumed here.

³For the purpose of the abstract model, we do not distinguish between personal identifiers and pseudonyms.

⁴Several contexts may come together under the umbrella of a *domain*. For instance, several contexts exist within the health domain, including hospital records, health insurance records, etc.

```

universe IDENTITY // abstract representation of all individuals
universe ATTRIBUTE
universe CONTEXT
universe PARTIAL_IDENTITY
universe IDENTIFIER  $\equiv \mathcal{P}(\text{ATTRIBUTE})$ 
//  $\mathcal{P}$  denotes the power set, i.e., the set of all subsets of a given set.

```

```

attributeSet : IDENTITY  $\rightarrow \mathcal{P}(\text{ATTRIBUTE})$ 
// The set of all attributes that belong to an identity

```

A number of identifying functions exist that map an identifier to its respective partial identity within a context. These functions play the role of keys in databases.

$$g : \text{IDENTIFIER} \times \text{CONTEXT} \rightarrow \text{PARTIAL_IDENTITY}$$

A partial identity is used to *represent* an identity within a distinguished context. In other words, a partial identity is the ‘realization’ of its respective identity within the context. Hence, the ideal is to always have a one-to-one relation between partial identities and real identities. In fact, an assumption often made by IMS designers is that a new partial identity is created only if the respective identity is not already represented in the context by another partial identity. However, this assumption is frequently violated in real world applications, which is the cause of many of the core problems in identity management.

It is important to emphasize that from a context point of view the ‘real’ (actual) association between partial identities and identities typically remains hidden; that is, there is no feasible way of checking or validating that the association is a correct one in an absolute sense. For our modeling purposes, however, we assume to have this absolute knowledge, although this possibility serves for illustrative purposes only. We therefore introduce an *oracle* that provides that hidden information. The oracle is defined as the relation⁵ O and is meant to provide the real identity(ies) behind a partial identity. In each state of the system, the oracle maps a partial identity defined within a context to the identity that is represented by that partial identity.

$$O \subseteq \text{CONTEXT} \times \text{PARTIAL_IDENTITY} \times \text{IDENTITY}$$

⁵It is important to note that the oracle is not necessarily a single-valued function.

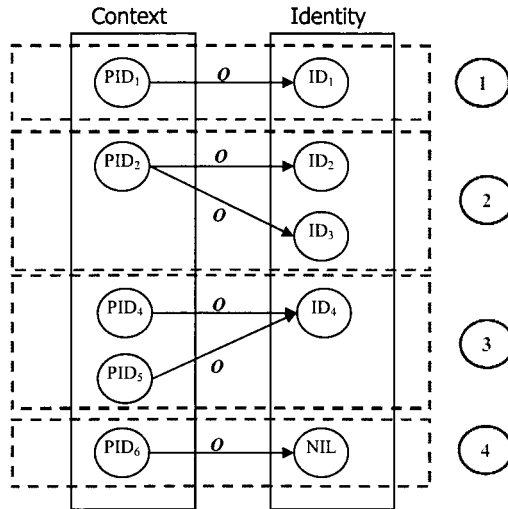


Figure 6.2: Mapping partial identities to identities within the same context

As noted above, in any given context, the ideal situation is to have a one-to-one mapping between partial identities and real identities. We thus define the following integrity constraints for the oracle. Of course, these constraints refer to the ideal situation and may not (and often do not) hold in reality.

O is meant to be a single-valued mathematical function. (6.1)

That is, within one context any single partial identity cannot represent more than one individual identity.

O is meant to be injective. (6.2)

The second constraint means that two partial identities existing in the same context can not be mapped to the same identity.

6.2.2 Mapping Partial Identities to Identities

Figure 6.2 illustrates four basically different cases of mapping partial identities to identities, some of which are potentially problematic. Note that in any given state, the oracle represents

these mappings. In other words, for each context, the oracle has an understanding similar to the one shown in the diagram. Here, we describe each case in more detail.

1. **Valid Mapping:** In this case, a given partial identity PID_1 is uniquely mapped to an existing identity ID_1 . In other words, the oracle knows that ID_1 is behind the partial identity PID_1 .
2. **O is not a single-valued function:** PID_2 represents both ID_2 and ID_3 . This happens when sufficient information about entities does not exist in a context. Hence, one partial identity is created which maps to two actual identities. This is obviously NOT a desired situation and violates the integrity constraint 6.1.
3. **O is not injective:** In this case, one identity is represented by several partial identities. This violates the injectivity constraint 6.2.
4. **Fake (fictitious) Identity:** This case captures situations where there is no real identity behind a partial identity.

Cases 2, 3, and 4 capture different *undesired* situations with respect to partial identities within one context. These cases can be categorized as *logical* inconsistencies and are explained using the mathematical representation. When analyzing partial identities across contexts, one can identify a second category of inconsistencies, called *semantic* inconsistencies. In this case, within one context a mathematically valid mapping exists between a partial identity and its respective identities (i.e., case 1 in Figure 6.2 applies); however, an inconsistency can be detected when other contexts are included into the analysis. Figure 6.3 shows two different cases of mapping of two partial identities across different contexts. The first case shows a valid mapping across contexts: PID_1 and PID'_1 are mapped to the same identity (ID_1) and thus need to be semantically consistent; i.e. there is no inconsistency in their constituting attributes. In the second case, however, there is inconsistency between PID_1 and PID'_1 in the sense that respective attributes have different values. Semantic inconsistencies within one context, such as having outdated records in one context, contributes to this type of inconsistencies across contexts.

One of the goals in identity management is to identify such cases, deal with them and minimize their occurrence. This is further discussed in the next section.

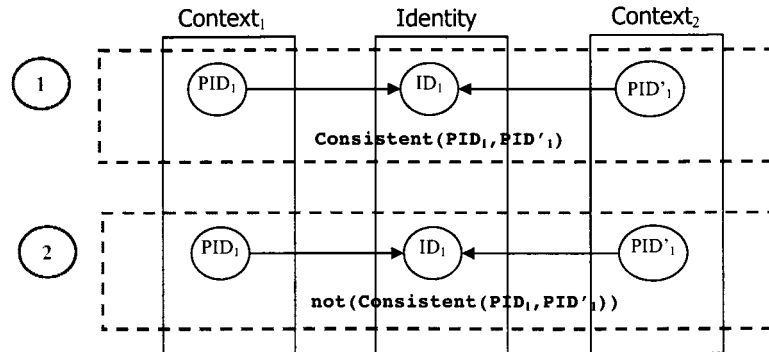


Figure 6.3: Mapping partial identities to identities across contexts

6.2.3 Evolution of Partial Identities

In general, the set of partial identities defined within a context changes dynamically over time. New partial identities are introduced, existing ones are discarded or merged to form new partial identities, or one partial identity is split into two. These changes define the *lifecycle* of partial identities within a given context.

Identity Resolution

We consider identity resolution as the *internal* process of changing the state of a context with respect to its partial identities. These changes are meant to improve the overall quality of the mapping between partial identities and identities, and, as a result, target the problematic cases discussed in Section 6.2.2. Several techniques have been proposed to identify and resolve these problems [156, 135, 189]. We provide a high-level view of identity resolution, abstracting from the operational details of how and when it is performed, and focusing on the semantic aspects.

Formally speaking, identity resolution is required when a violation of one of the integrity constraints (6.1) or (6.2) is detected. From the perspective of an outside observer, the state of a context changes after performing an identity resolution operation. Here, we split the identity resolution process into two sub-tasks, each dealing with one type of integrity constraint.

$$\text{IdentityResolution} \equiv \text{FunctionResolve} \wedge \text{InjectivityResolve}$$

At the highest level of abstraction, we describe the effect of each operation in terms of pre-conditions and post-conditions that hold before and after performing each task respectively.⁶

The function resolution operation is triggered when a new piece of information is obtained that reveals that one single partial identity is referring to two or more identities.⁷ In our model, this new information is abstractly represented by the function *newInfo*. Assuming $idSet(c, p_1, newInfo)$ refers to the set of identities that, based on *newInfo*, are all represented by a single partial identity p_1 in the context c , the `FunctionResolve` operation is specified as follows.

pre $\exists id_1, id_2 \in idSet(c, p_1, newInfo)$
 $O(c, p_1, id_1) \wedge O(c, p_1, id_2) \wedge id_1 \neq id_2$

FunctionResolve($c : \text{CONTEXT}, p_1 : \text{PARTIAL_IDENTITY}, newInfo : \text{INFO}$)

post $\forall id_1, id_2 \in idSet(c, p_1, newInfo), \forall p \in \text{PARTIAL_IDENTITY}$
 $O(c, p, id_1) \wedge O(c, p, id_2) \Leftrightarrow id_1 = id_2$

The injectivity resolution operation is triggered when a set of partial identities ($pSet$) is detected, all elements of which refer to the same identity id_1 . Therefore, the `InjectivityResolve` operation is specified as follows.

pre $\exists id_1 \in \text{IDENTITY}, \forall p \in pSet \ O(c, p, id_1)$

InjectivityResolve($c : \text{CONTEXT}, pSet : \mathcal{P}(\text{PARTIAL_IDENTITY})$)

post $\forall p_1, p_2 \in \text{PARTIAL_IDENTITY}$
 $O(c, p_1, id_1) \wedge O(c, p_2, id_1) \Leftrightarrow p_1 = p_2$

It is worth mentioning that the existing literature does not provide a precise high-level definition of identity resolution. Instead, identity resolution is commonly defined in a bottom-up fashion by describing the underlying algorithms that perform the operation. This is a potential source of inconsistencies, especially when interoperability is a priority. To address this problem, we illustrate how one can properly specify each of the above mentioned

⁶We define here a more general case with n identities/partial identities.

⁷In practice, different heuristic approaches and AI-based techniques are used to extract this information [156, 135, 189].

operations in a systematic fashion, such that they can be further refined into appropriate algorithms and implementations. While simple and easy to understand, the formalization used here is precise and grounded in mathematical logic based on the abstract state machine (ASM) [30] formalism and abstraction principles (see Chapter 2).

```

FunctionResolve( $c : \text{CONTEXT}, p_1 : \text{PARTIAL\_IDENTITY}, \text{newInfo} : \text{INFO}$ )  $\equiv$ 
  // Resolved by splitting  $p_1$  into different partial ids
  Delete( $c, p_1$ )
  SplitIntoNewPartialIDs( $c, p_1, \text{newInfo}$ )

```

```

InjectvtyResolve( $c : \text{CONTEXT}, pSet : \mathcal{P}(\text{PARTIAL\_IDENTITY})$ )  $\equiv$ 
  // Resolved by merging partial identities into one
  forall  $pid$  in  $pSet$  do [in parallel] Delete( $c, pid$ )
  MergeIntoANewPartialID( $c, pSet$ )

```

The delete, split, and merge operations are purposefully left abstract at this level. Details can be added as needed in systematic refinement steps.

As a result of the resolution process, one or more partial identities are deleted from the context, and one or more new partial identities are created using the information from the old ones. Here, we introduce a new concept: the new partial identities may come with two additional attributes, specifying the *confidence* in the newly created partial identity and the *history* of its evolution (allowing to undo the operation if necessary) respectively. The confidence value is a standardized numerical value expressed as a real number in the interval $[0, 1]$.

6.3 Applications

In this section, we explore potential applications for which the semantic framework of the Identity Management Architecture could be used and illustrate how it would be helpful. An important usage of the framework is for achieving interoperability across different domains, which is impossible without a coherent and consistent semantic foundation. Here, we can make an analogy to the programming languages domain and the importance of having a well defined semantic for a given programming language. Otherwise, different implementations of the same language may result in different interpretations of some concepts of the language,

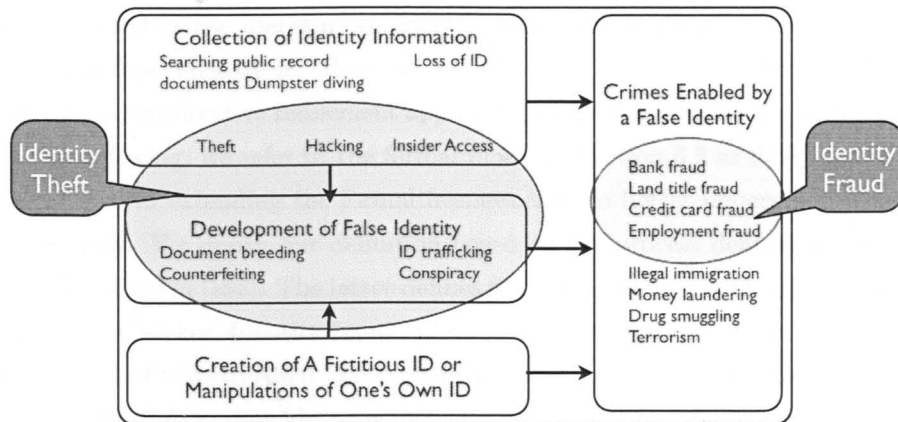


Figure 6.4: Conceptual process model of identity theft domain presented in [181]

leading to inconsistent behavior of the same program in different implementations. The framework can also be applied in contexts such as identity theft, privacy, and criminal investigations, as further discussed below.

6.3.1 Identity Theft

Section 6.1.4 provides a brief overview of the literature on identity theft. The authors of [181] developed a conceptual model of identity theft and identity fraud by identifying related common activities, as shown in Figure 6.4.

As a result, the following two definitions are provided [181]:

Identity theft: *the unauthorized collection, possession, transfer, replication or other manipulation of another person's personal information for the purpose of committing fraud or other crimes that involve the use of a false identity.*

Identity fraud: *the gaining of money, goods, services, other benefits, or the avoidance of obligations, through the use of a false identity.*

A conceptual model clearly delineating identity theft and identity fraud is a valuable contribution and a good starting point for integrating research and practices in the field. However, we believe that such fundamental concepts need to be more rigorously defined. For instance, both definitions use the term *false identity*, which is only loosely defined and lacks a precise semantics. In fact, the fundamental question of 'what constitutes an identity'

is mostly neglected in the literature on identity theft. Consequently, it is difficult to infer what exactly a false identity is. Here, we use the basic model of Section 6.2 and further extend it using a conservative refinement approach (see 2.5) to capture the notion of identity theft. In the following, we refer to the formal model of Section 6.2 as $IMA_{\mathcal{AM}core}$.

Our first goal in extending the formal framework is to better understand what exactly is a false identity. We derive our definition based on the process defined in [181] and the framework described in [192]. The latter defines five roles involved in identity theft: Identity Owner, Identity Checker, Identity Issuer, Identity Protector, and Identity Thief (see 6.1.4).

In our model of identity theft, we introduce a new, more abstract role, called the *Presenter of Identity Information*. In principle, this entity is either the owner of the presented information or a third party. If the third party is *not authorized* to use the presented information, he/she is considered a thief. Hence, the *presented information* is a key element in defining a presenter. As explained in 6.2, in the $IMA_{\mathcal{AM}core}$ identities are defined as abstract representations of entities. Here, we use this abstraction and define a presenter p as a tuple consisting of an identity id_p (the real person) and a set of attributes being presented $presAtrbts_p$ (the presented information).

$$\forall p \in \text{PRESENTER}, p \equiv (id_p, presAtrbts_p)$$

$$\text{where } id_p \in \text{IDENTITY} \wedge presAtrbts_p \in \mathcal{P}(\text{ATTRIBUTE})$$

We also formalize the notion of *identification* using the definition provided in [53]. In the process of identification, a physical entity that presents a set of attributes⁸ is *matched* to an *existing* identity.⁹

$$identification : \text{PRESENTER} \rightarrow \text{IDENTITY}$$

We can now precisely define the difference between a presenter's identity and the identification of a presenter, which is a source of ambiguity. The identity (id_p) is an integral part of the presenter; it is defined statically and can not be detached from the presenter. Referring back to the $IMA_{\mathcal{AM}core}$, the $attributeSet(id_p)$ function provides the set of attributes associated with id_p . This set may change over time as the attributes of an identity change, whereas the identity itself does not change. Identification of a presenter, on the

⁸We are not concerned with the authentication of the attribute set and assume the attributes are already authenticated.

⁹Note that if matching results in several identities, a logical inconsistency exists (see case 2 in Figure 6.2) which has to be resolved separately. Hence, we restrict here to one identity only.

other hand, is a dynamic process which is based on the *presented attributes* of the presenter ($presAtrbts_p$), not necessarily its *real attributes*—the ones that legitimately belong to the presenter ($attributeSet(id_p)$).

We abstractly define the process during which the presented attributes are mapped to an identity. A presenter p presents a set of attributes $presAtrbts_p$ to an *issuer* (or *checker*) of partial identities. This is where identification happens. In the IMA_{AM} core, *partial identities* are used within a given context as representations of identities. As such, the identification process logically splits into two consecutive steps: (1) mapping the attribute set to a partial identity in a given context, and (2) implicit association between the partial identity and its respective identity. The following schema summarizes the identification process.

$$p \xrightarrow{\text{presents}} presAtrbts_p \xrightarrow{\text{matches}} pId \xrightarrow{\text{represents}} id \quad (6.3)$$

We now explicitly define what is referred to as *false identity* in the literature. A false identity is assumed by a presenter when presenting attributes for identification that do not *belong* to the presenter's identity, as stated below.

$$falseIdentity(p) \Leftrightarrow presAtrbts_p \notin \mathcal{P}(attributeSet(id_p)) \quad (6.4)$$

As shown in Figure 6.4, the conceptual process model of [181] identifies two different categories of activities that happen before a false identity is developed: (1) collection of personal information, and (2) creation of fictitious identity or manipulation of one's own identity. However, only the activities in the first category are considered as identity theft. We want to address this issue in further detail using our formal model. In the following, we use the terms id , pId , and $matches$ from Schema 6.3.

In the first case, in a given context c , the collected information $presAtrbts_p$ forms a **false identity that matches a real identity** id ; i.e.,

$$\begin{aligned} &\exists pId \in \text{PARTIAL_IDENTITY}, id \in \text{IDENTITY} \\ &matches(c, presAtrbts_p) = pId \wedge O(c, pId, id) \end{aligned}$$

It is important to point out that, in this case, it is *implicitly* assumed that the collection of personal information is not *authorized* by the owner of the identity information. We later discuss the implications of this assumption and the importance of making it explicit.

A **fictitious identity** is a false identity which is not based on a real person [181]. This

happens when the result of the identification process yields ' $id = undef$ '; more precisely, for a presenter p in a given context c ,

$\exists pId \in \text{PARTIAL_IDENTITY}$

$\text{matches}(c, \text{presAtrbts}_p) = pId \wedge O(c, pId, undef) \neq O$ is the oracle

Therefore, it is safe to assume that identity theft does not occur in the case of mapping to a fictitious identity. However, if a false identity is created by manipulation of one's own identity attributes, it may still match a real identity (similar to the first case). Hence, we argue that identity theft can technically occur in the latter case.

In the following, we use our formal model to highlight some of the loose ends in the existing definition, arguing that there is a need for more precision and rigor in defining identity theft. In fact, we look at the problem from a different perspective and reorganize different cases as follows.¹⁰

1. The presented information does not belong to the presenter and is mapped to another person's identity; i.e., $id_p \neq id$ and $id \neq undef$.
 - (a) The presenter (id_p) has proper authorization from the owner of the identity information (id); i.e., $\text{isAuthorized}(id_p, id, now)$ holds. It is important to point out the need for *real-time* evaluation of the authorization, which emphasizes the dynamic nature of the process.¹¹
 - (b) The presenter (id_p) does NOT have proper authorization from the owner (id); i.e., **identity theft** occurs.
2. The presented information lead to a fictitious identity; i.e., $id = undef$. Hence, identity theft does not occur.
3. A person, by mistake, presents his/her own attributes incorrectly. Since these attributes are not owned by that person, according to the Definition 6.4, a false identity is assumed.

¹⁰It is important to point out that these cases capture the notion of identity theft, whereas, according to [181], identity fraud occurs when the actual crime takes place.

¹¹Other factors, such as the specific context where identification occurs, should also be considered in authorization. For simplicity we use this broader definition of authorization.

- (a) Such a false identity may be fictitious, as described above (i.e., $id = undef$). In that case, identity theft does not occur.
- (b) If the false identity still maps to the right person (i.e., ' $id = id_p$ '), for instance due to the flexibility of the mapping algorithm, identity theft does not take place either.
- (c) However, if the false identity is NOT fictitious and does NOT map to the right person (i.e., $id_p \neq id$ and $id \neq undef$), **identity theft** has taken place, by definition.

It is interesting to note the different insight that the formal definition provides. Firstly, the notion of authorization introduces further complexity of real-time evaluation, context of authorization, and validation that deserve clarification in the definition of identity theft. Secondly, as far as identity theft is concerned, it is not important whether the presented information is a manipulated/modified version of one's own attributes, or is completely stolen. If the presented attributes are mapped to an identity which is different from the identity of the presenter, an unauthorized use of the personal information of the person behind that identity has happened; hence, there is a case of identity theft. This important observation has not been clearly addressed before. In the conceptual model of [181], it is implicitly assumed that "manipulation of one's own identity" is not malicious and is mostly used for preserving privacy. As a result, it is not included in the activities that contribute to identity theft. Part of the problem may have been caused by the vague definition of *one's identity*, which allows for using terms like 'multiple identities' for one person, or 'manipulation of one's own identity' without differentiating between identity and the attributes presented for identification.

Having a unified semantic framework facilitates integration across contexts and allows for distributed analysis approaches for fraud detection. Within such a framework, one can properly define and identify different cases of identity theft, and develop proper safeguards against the misuse of identity.

6.3.2 Other Applications

In an investigation context, the identity management problem is viewed from a different perspective. The police, or the investigators, collect bits and pieces of information in order to derive the identity of a criminal offender. In the terminology of our framework, they try

to collect enough *attributes* to reconstruct a *partial identity* which can be mapped to an *identity*. This is a highly dynamic process, since every new piece of information may change the constructed partial identity(ies) and their respective mappings. Crime investigation techniques must deploy inference methods that oversee causal relationships between events and partial identities, time and location of crime events, and crime signatures. The high dynamics calls for systematic approaches to the design of computer-assisted investigation tools. As such, we contend that our semantic framework can be the first building block in that direction.

6.4 Summary and Future Work

Identity management is a challenging problem in many strategic application domains, especially in security-related and Web-based contexts. Identity management systems are complex in nature and involve different social, legal and technical aspects. Although a wide variety of solutions has been offered to address some of the key issues of identity management, there is still no common agreement on the definition of the most fundamental concepts, such as what constitutes an identity.

Addressing the lack of such a unifying view, we adopt a formal modeling approach and propose a precise semantic model based on common notions and structures of computational logic using the ASM method. This model provides a well defined framework, called *Identity Management Architecture*, for analyzing and reasoning about identity management concepts and requirements on Identity management systems. To exemplify the practicality of the framework in the systematic study of identity management, we applied the model to semantic aspects of identity theft, trying to clarify the underlying definitions and basic concepts.

This work can be further extended to study other critical requirements of IMS, such as privacy. Having an abstract architectural view facilitates analyzing the impact of using different privacy preserving techniques, such as anonymization, especially when multiple parties (contexts) are involved. The framework can also be used to study the semantics of some of the existing standards for identity management, e.g. the concept of federated identity management.

Chapter 7

Conclusions

7.1 Summary of Contributions

The research presented in this thesis has produced four major contributions. First, we have developed a methodological framework for computational modeling and software development of complex social systems in interdisciplinary research contexts. Second, we have adopted and further developed a supporting tool environment specifically addressing the needs and requirements of modeling complex social systems. Third, the framework has been successfully applied and tested in the context of the Mastermind project on modeling crime patterns and theories in crime analysis and crime prevention, a key aspect in Computational Criminology. Finally, we have shown the value of mathematical modeling and rigorous analysis of highly complex socio-technical systems, such as identity management systems, by applying a semantic model of identity management in systematic study of identity theft. We contend that these examples affirm consistency, applicability and scalability of our approach.

Our work has addressed the lack of well-established methodologies for design, construction, validation and simulation of models of social systems. The inherent complexity of such systems and the high interdependence and diversity of the variables involved make the modeling process highly iterative and potentially open-ended. Given the ever-growing use of social systems modeling, we have emphasized the importance of developing proper modeling methodologies in order to avoid a predicament similar to the software crisis of the 60s and 70s. In particular, we have considered agent-based modeling approaches which, due to their flexibility in defining individual agent behavior, have gained wide-spread popularity

for modeling social phenomena in the recent years. We have pointed out that, despite their popularity, the shortcomings of existing agent-based approaches in addressing methodological questions, e.g., how to build a model or how to validate a model, have raised concerns about their reliability, limiting their success and applicability.

We have utilized the ASM formalism and CoreASM tool suite for interactive design and validation of agent-based models of complex social phenomena. Although unconventional, the application of the ASM paradigm to social systems has proven to be a promising approach. It addresses the fundamental needs and requirements of the iterative process of modeling, namely:

1. *Simplicity* of models is managed by systematic use of the freedom of abstraction provided by the ASM paradigm;
2. *Validation* is integrated into all different phases of the process (i.e., conceptual, mathematical and experimental);
3. The inherent *precision* and *rigor* offered by the underlying mathematical foundation enforces structure, logical thinking and critical analysis of the fundamental assumptions and key aspects of the systems under study.

The ASM paradigm nicely integrates with the established view of multi-agent modeling of social systems and provides a precise semantic foundation—something multi-agent system modeling is lacking. One fundamental characteristic of our framework is its focus on the *model building* phase of the development, emphasizing the importance of the cooperative *process* of transforming an abstract concept into a computational artifact. The ASM method also facilitates seamless transitions between different phases of modeling, ensuring the coherence and consistency of the final models.

Application

The proposed framework addresses the needs of interdisciplinary R&D projects and has been developed in close collaboration with non-computing researchers, in particular criminologists [45, 46, 47, 48, 113]. It has been successfully applied in the context of the Mastermind project on modeling crime patterns and theories in crime analysis and prevention. Mathematical and computational modeling of crime serve multiple purposes in various contexts, including

law enforcement, intelligence-led policing, and proactive crime reduction and prevention. For intelligence-led policing, our model makes it possible to simulate likely activity spaces for serial offenders for use in apprehension or precautions. For proactive policing, modeling of crime allows analyzing different scenarios in crime analysis and prevention and provides a basis for experimental research; simulation models make it possible to run experiments that can often not easily be done in a real-world setting. Furthermore, the Mastermind model is designed in such a way that it is scalable and not only applicable to a broad range of crimes but also to other environments such as airports, ports, shopping centers, and subway stations.

The framework has also been applied in the context of identity management systems, which are influenced by various social, legal and technical aspects. Despite a wide range of research and development efforts in the domain, different aspects of identity management are mostly studied separately, and there is no common agreement even on the most fundamental concepts of the field. The lack of a unifying framework for bringing together different aspects of identity management has caused new problems especially concerning interoperability, privacy and identity fraud. We have adopted the proposed formal modeling approach to build a precise semantic model capturing essential concepts of identity management. Our proposed Identity Management Architecture facilitates a systematic study of identity management and provides improved accuracy in reasoning about key properties of identity management system design. The successful application of the model to the study of semantic aspects of identity theft has further affirmed its value by unveiling some of the loose ends in the common existing understanding of identity theft and fraud.

These case studies have provided proof of the applicability of the framework in different contexts. Building on mathematical abstractions, the CoreASM tool suite facilitates communication with domain experts, rapid prototyping and systematic construction of computer simulations. Furthermore, the inherent rigor of the process provides a new means for understanding the underlying systems, challenges the existing assumptions by eliciting ambiguities, and creates new questions about the underlying systems. The ongoing research and development on the framework and its application is another proof of its reliability and scalability, as further discussed below.

7.2 Future Research Directions

Beyond crime analysis and prevention, one may apply the Mastermind approach in counterterrorism, specifically, in event planning and emergency response, or in public safety for improving security measures or protecting public spaces. Mastermind is now being used as a basis for developing a decision support tool for counterterrorism planning and developing response strategies. The tool provides large-scale simulation models of crowd movement in urban environments, and allows for analyzing different scenarios and the impact of different types of potential threats. It features an enhanced 3D visualization system using the Google Earth platform. The goal is to apply the tool for analyzing threats and developing response strategies in safety and security planning of large events such as 2010 Winter Olympics in Vancouver.

In a different context, the Mastermind model and the proposed framework is being used to introduce computational modeling for studying the impact of environmental factors on chronic diseases such as obesity. Mastermind offers a new computational approach, not explored in the literature before, to modeling physical activity levels of people in their built environment. The modeling exercise, within the proposed framework, enforces structured logical thinking about the most basic concepts and demands precision and rigor in analyzing key assumptions about the system under study. So far, it has helped unveiling some hidden assumptions and limitations of the conventional studies in the field. Simulating the behavior of individual agents in urban environments is invaluable in answering crucial questions for policy makers in health care planning; e.g., what happens to physical activity levels if certain changes take place in the environment, or what features of the environment can be changed to have a desired impact on physical activity levels.

Exploring new directions and applying the framework in new contexts will help further identify the requirements that need to be addressed by the methodology. This leads to the development of new tools and best practices to support the modeling process. For instance, we have already identified the need for a *state explorer* tool for the CoreASM tool suite. Being able to closely monitor changes of the state would provide a better understanding of the working of the model and facilitate debugging and validation processes. Providing better visualization tools that closely match the needs of the domain experts is also essential in this type of research. In particular, it is an integral requirement for making the models applicable for practitioners and policy planners. As such, the next steps of development

include defining well-structured interfaces to existing visualization tools, e.g., Google Earth and available game engines, and providing standard visualization plugins for basic needs.

Final Remarks

Computational and mathematical modeling offer new problem solving and analysis techniques that play a key role in advancing the boundaries of many research disciplines. For the social sciences, in particular, the precision and rigor offered by mathematical models facilitate establishing a clear and consistent understanding of the underlying complex social systems. At the same time, computational models allow for dynamic testing and computer-assisted experiments which may be impossible to carry out in the real world. As such, computational models of complex social systems serve as decision support and exploration tools. They facilitate analyzing the interplay between different variables of the system, exploring trade-offs, and developing alternate solutions by running experiments in a virtual world. The research presented here further supports the efforts in applying computational modeling techniques to the study of social phenomena by providing a systematic framework for modeling and simulation, and showing the value of such rigorous approaches in novel application contexts.

Appendices

Appendix A

Dijkstra's Shortest Path Algorithm

Dijkstra's shortest path algorithm guarantees to find the shortest path between a source vertex s and every vertex v in a given graph in time $O(n^2)$ [68].

The algorithm is defined on a weighted, directed graph $G = (V, E)$, where V is the set of vertices, $E \subseteq V \times V$ is the set of edges on the graph, and a weight function $w : E \rightarrow R+$ maps every edge in the graph to a non-negative real-valued weight. For each vertex v the cost d_v denotes the distance of the shortest path between s and v . A path p between vertex s and vertex v is denoted by $s \xrightarrow{p} v$.

The algorithm works as follows:

1. Initialize d_s to 0. For every other vertex v in V initialize d_v to *infinity*, ' ∞ ', representing the fact that no path to those vertices is known yet.
2. Initialize the set of *unvisited vertices* Q to all the vertices in the graph except s ; i.e., $Q = V - \{s\}$.
3. While Q is not empty ($Q \neq \emptyset$), do the following:
 - (a) Find vertex u with minimum distance from source s .
 - (b) Remove u from Q .
 - (c) For each outgoing edge (u, v) from u , update the distance as follows:
if $d_v > d_u + w(u, v)$ then

- Update the distance of v from s : $d_v = d_u + w(u, v)$.
- Add u to the shortest path of v ; i.e., add u to $s \xrightarrow{P} v$.

When the algorithm finishes, d_v will be the cost of the shortest path from s to v , or *infinity*, if no such path exists.

Appendix B

Identity Management Glossary

Sources: [54, 53, 123]

- **Anonym (as in anonymous):** An authenticated attribute that is not linked to an identifier. An identifier associated with no personal identifier, but only with a single-use attestation of an attribute. An anonymous identifier ascertains an attribute, once. An anonymous attribute used more than once becomes a pseudonym [54, 53].
- **Attribute:** A characteristic associated with an entity, such as an individual. Examples of persistent attributes include height, eye color and date of birth. Examples of temporary attributes include address, employer and organizational role. A Social Security Number is an example of a long-lived attribute. Some biometrics are persistent (e.g. fingerprints), some change over time or can be changed (e.g. hair color) [54, 53].
- **Attribute Authentication:** Proving an association between an entity and an attribute. Confirming some one's age is an example. This is usually a two-step process, where the association between an entity and an identifier is established, and then a link between identifier and attribute is established [54, 53].
- **Authentication:** Proving an association between an identifier or attribute, and the relevant entity. For example, an automobile is identified by its license plate, and that is authenticated as legitimate by the database of cars that are not being sought for enforcement purposes [54, 53].
- **Authentication vs. Identification:** Authentication accepts a possibility of error; reasonable risk of misidentification or mis-authorization [123].

- **Authorization:** A decision to allow a particular action based on an identifier or attribute. Examples include the ability of a person to make claims on lines of credit, the right of an emergency vehicle to pass through a red light or a certification of a radiation-hardened device to be attached to a satellite [54, 53].
- **Identification:**
 - In [54, 53]: Association of a personal identifier with an individual presenting certain attributes. For example, accepting the association between a physical person and claimed name, or determining the association with a medical record and a patient using physical attributes.
 - In [123]: Identification is “having enough assurance of who a person is to proceed with a transaction. What constitutes “enough” is very contextual. It turns on the risks of misidentification”
- **Identifier:**
 - In [54, 53]: An identifier identifies a distinct person, place or thing within the context of a specific namespace. For example, an automobile, a bank account and person each have identifiers. The automobile has a license plate and the bank account has a number. The person may be associated with either the auto or the account through additional information, such as a certificate of ownership, or a social security number. One identity can have multiple identifiers: a car has a permanent serial number and a temporary license plate. Each identifier is meaningful only in a specific context, or namespace, and can reasonably be thought of as having a $\langle \text{thing identified, identifier, namespace} \rangle$ set.
 - In [123]: Identifiers are building blocks of identification. These are facts that distinguish people and entities from one another (same as *characteristics* or *attributes* used for sorting or categorizing entities). Identifiers are classified as follows:
 1. *Something-you-are*: inherent characteristics (mostly) attached to physical body, e.g. DNA. These are known as *biometrics* and are further categorized into *physiological* and *behavioral*.
 2. *Something-you-are-assigned*: socially defined titles such as names and addresses. These identifiers are not unique and are subject to change.

3. *Something-you-know*: some distinct knowledge such as password, or mother's maiden name. Known as *epistemic* identification, the *knowledge* of a person is compared to what he/she is supposed to know given her alleged identity (Fact-Checking).
4. *Something-you-have*: possessing some distinct item such as identity card (also called 'token'). Tokens are physical objects that help identifying their bearer (driver's license, access cards, etc.).

Identifiers are of different *qualities*. Some are unique (DNA), some are fixed (mother's maiden name), and some are transient (wearing a specific shirt). Quality of identifiers are defined along the vectors of fixity, distinctiveness, and permanence which determines how they should be used. Most importantly the quality of an identifier depends on how useful it is over *time*.

- **Identity**: The set of permanent or long-lived temporal attributes associated with an entity [54, 53].
- **Identity Authentication**: Proving an association between an entity and an identifier. For example, the association of a person with a credit or educational record [54, 53].
- **Multifactor/multi-identifier Identification**: Using a combination of identifiers to achieve higher quality [123].
- **Personal identifier**: Persistent identifiers associated with individual human and attributes that are difficult or impossible to change, such as human date of birth, height and genetic code [54, 53].
- **Pseudonym**: An identifier associated with attributes or sets of transactions, but with no permanent or personal identifier. [54, 53]

Bibliography

- [1] CRTI—Call for Proposals: Guidebook for Fiscal Year 2008-2009, June 2008.
- [2] Google earth 5.0, Last visited in June 2009. <http://earth.google.com/>.
- [3] IBM Identity Resolution Software, Last visited in June 2009. <http://www-01.ibm.com/software/data/db2/eas/identity>.
- [4] A. Aamodt and E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications*, (1):39–52, 1994.
- [5] J. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.
- [6] A. Agnar and P. Enric. Case-based Reasoning : Foundational Issues, Methodological Variations, and System Approaches, 1994.
- [7] A. Alimadad, P. Brantingham, P. Brantingham, V. Dabbaghian-Abdoly, R. Ferguson, E. Ferguson, A. H. Ghaseminejad, C. Giles, J. Li, N. Pollard, A. R. Rutherford, and A. van der Waall. Simulation of the Criminal Justice System for Policy Analysis: The Complementary Strengths of Discrete Event Models and System Dynamics Models. In L. Liu and J. Eck, editors, *Artificial Crime Analysis Systems: Using Computer Simulations and Geographic Information Systems*. Information Science Reference, 2008.
- [8] T. Andrews et al. Business process execution language for web services version 1.1, May 2003. Last visited Feb. 2005, <http://ifr.sap.com/bpel4ws/>.
- [9] M. Anlauff. XASM – An Extensible, Component-Based Abstract State Machines Language. In Y. Gurevich and P. Kutter and M. Odersky and L. Thiele, editor, *Abstract State Machines: Theory and Applications*, volume 1912 of *LNCS*, pages 69–90. Springer-Verlag, 2000.
- [10] M. Anlauff and P. Kutter. *eXtensible Abstract State Machines*. XASM open source project: <http://www.xasm.org>.
- [11] R. Aris. *Mathematical Modelling Techniques*. New York : Dover, 1994.
- [12] S. T. L. at Simon Fraser University. *The CoreASM Project*. <http://www.coreasm.org>.
- [13] M. Batty. *Cities and Complexity: Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals*. The MIT Press, Sept. 2005.
- [14] K. Baumgartner, S. Ferrari, and C. Salfati. Bayesian Network Modeling of Offender Behavior for Criminal Profiling. In *44th IEEE Conference on Decision and Control and European Control Conference CDC-ECC'05*, pages 2702–2709, 2005.
- [15] D. M. Berry. Formal Methods: The Very Idea—Some thoughts about why they work when they work. *Science of Computer Programming*, 42(1):11–27, 2002.

- [16] A. Blass and Y. Gurevich. Abstract State Machines Capture Parallel Algorithms. *ACM Transactions on Computational Logic*, 4(4):578–651, 2003.
- [17] N. Boccarda. *Modeling Complex Systems*. Springer, 2004.
- [18] E. Börger. A Logical Operational Semantics for Full Prolog. Part I: Selection Core and Control. In E. Börger, H. Kleine Büning, M. M. Richter, and W. Schönfeld, editors, *CSL'89. 3rd Workshop on Computer Science Logic*, volume 440 of *LNCS*, pages 36–64. Springer, 1990.
- [19] E. Börger. A Logical Operational Semantics of Full Prolog. Part II: Built-in Predicates for Database Manipulation. In B. Rován, editor, *Mathematical Foundations of Computer Science*, volume 452 of *LNCS*, pages 1–14. Springer, 1990.
- [20] E. Börger. The ASM Ground Model Method as a Foundation of Requirements Engineering. In N. Dershowitz, editor, *Verification: Theory and Practice*, volume 2772 of *LNCS*, pages 145–160. Springer-Verlag, 2003.
- [21] E. Börger. The ASM Refinement Method. *Formal Aspects of Computing*, 15:237–257, 2003.
- [22] E. Börger. The ASM Method for System Design and Analysis. A Tutorial Introduction. In *Frontiers of Combining Systems*, volume 3717/2005 of *Lecture Notes in Computer Science*, pages 264–283. Springer Berlin / Heidelberg, 2005.
- [23] E. Börger. Construction and Analysis of Ground Models and their Refinements as a Foundation for Validating Computer Based Systems. *Formal Aspects of Computing*, 19(2):225–241, 2007.
- [24] E. Börger, M. Butler, J. P. Bowen, and P. Boca, editors. *Proceedings of Abstract State Machines, B and Z First International Conference, ABZ 2008*. Lecture Notes in Computer Science. Springer, 2008.
- [25] E. Börger, N. G. Fruja, V. Gervasi, and R. F. Stärk. A High-level Modular Definition of the Semantics of C#. *Theoretical Computer Science*, 336(2/3):235–284, May 2005.
- [26] E. Börger, U. Glässer, and W. Müller. The Semantics of Behavioral VHDL'93 Descriptions. In *EURO-DAC'94. European Design Automation Conference with EURO-VHDL'94*, pages 500–505, Los Alamitos, California, 1994. IEEE CS Press.
- [27] E. Börger, U. Glässer, and W. Müller. Formal Definition of an Abstract VHDL'93 Simulator by EA-Machines. In C. Delgado Kloos and P. T. Breuer, editors, *Formal Semantics for VHDL*, pages 107–139. Kluwer Academic Publishers, 1995.
- [28] E. Börger, E. Riccobene, and J. Schmid. Capturing Requirements by Abstract State Machines: The Light Control Case Study. *Journal of Universal Computer Science*, 6(7):597–620, 2000.
- [29] E. Börger and W. Schulte. A Practical Method for Specification and Analysis of Exception Handling: A Java/JVM Case Study. *IEEE Transactions on Software Engineering*, 26(10):872–887, October 2000.
- [30] E. Börger and R. Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer-Verlag, 2003.
- [31] P. Borne, B. Fayech, S. Hammadi, and S. Maouche. Decision Support System for Urban Transportation Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 33:67–77, 2003.

- [32] A. E. Bottoms and P. Wale. Environmental Criminology. In M. Maguire, R. Morgan, and R. Reiner, editors, *The Oxford Handbook of Criminology (3rd ed.)*, pages 620–656. Oxford University Press, 2002.
- [33] J. Bowen and V. Stavridou. Safety-critical Systems, Formal Methods and Standards. *Software Engineering Journal*, 8(4):189–209, July 1993.
- [34] J. P. Bowen and M. G. Hinchey. Seven More Myths of Formal Methods. *IEEE Software*, 12(4):34–41, 1995.
- [35] P. J. Brantingham, A. Bertozzi, G. Tita, and L. Chayes. UC Mathematical and Simulation Modeling of Crime Project, 2006. Last visited in June 2009, <http://paleo.sscnet.ucla.edu/ucmasc.htm>.
- [36] P. J. Brantingham and P. L. Brantingham. *Patterns in Crime*. New York: Macmillan Publishing Company, 1984.
- [37] P. J. Brantingham and P. L. Brantingham. Introduction: The Dimensions of Crime. In P. J. Brantingham and P. L. Brantingham, editors, *Environmental Criminology*, pages 7–26. Waveland Press, 1991.
- [38] P. J. Brantingham and P. L. Brantingham. Environment, Routine and Situation: Toward a Pattern Theory of Crime. *Routine Activity and Rational Choice: Advances in Criminological Theory*, pages 259–294, 1993.
- [39] P. J. Brantingham and P. L. Brantingham. Nodes, Paths and Edges: Considerations on the Complexity of Crime and the Physical Environment. *Journal of Environmental Psychology*, pages 3–28, 1993.
- [40] P. J. Brantingham and P. L. Brantingham. Computer Simulation as a Tool for Environmental Criminologists. *Security Journal*, pages 21–30, 2004.
- [41] P. J. Brantingham and P. L. Brantingham. The Rules of Crime Pattern Theory. *Environmental Criminology and Crime Analysis*, 2008.
- [42] P. J. Brantingham, P. L. Brantingham, and U. Glässer. Computer Simulation as a Research Tool in Criminology and Criminal Justice. *Criminal Justice Matters*, (58), February 2005.
- [43] P. J. Brantingham and G. Tita. Offender Mobility and Crime Pattern Formation from First Principles. In L. Liu and J. Eck, editors, *Artificial Crime Analysis Systems: Using Computer Simulations and Geographic Information Systems*. Information Science Reference, 2008.
- [44] P. L. Brantingham, U. Glässer, P. Jackson, and M. Vajihollahi. Modeling Criminal Activity in Urban Landscapes. Technical Report SFU-CMPT-TR-2008-13, Simon Fraser University, Aug 2008.
- [45] P. L. Brantingham, U. Glässer, P. Jackson, and M. Vajihollahi. Modeling Criminal Activity in Urban Landscapes. In N. Memon, J. Farley, D. Hicks, and T. Rosenorn, editors, *Mathematical Methods in Counterterrorism*. SpringerWienNewYork, 2009.
- [46] P. L. Brantingham, U. Glässer, B. Kinney, P. Jackson, and M. Vajihollahi. Mastermind: Computational Modeling and Simulation of Spatiotemporal Aspects of Crime in Urban Environments. In L. Liu and J. Eck, editors, *Artificial Crime Analysis Systems: Using Computer Simulations and Geographic Information Systems*. Information Science Reference, 2008.

- [47] P. L. Brantingham, U. Glässer, B. Kinney, K. Singh, and M. Vajihollahi. A Computational Model for Simulating Spatial Aspects of Crime in Urban Environments. In M. Jamshidi, editor, *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, pages 3667–74, October 2005.
- [48] P. L. Brantingham, U. Glässer, B. Kinney, K. Singh, and M. Vajihollahi. Modeling Urban Crime Patterns: Viewing Multi-Agent Systems as Abstract State Machines. In D. Beauquier and et al., editors, *Proceedings of the 12th International Workshop on Abstract State Machines (ASM'05)*, March 2005.
- [49] M. E. Bratman. *Intention, Plans, and Practical Reason*. CSLI Publications, 1987.
- [50] M. E. Bratman, D. Israel, and M. E. Pollack. Plans and Resource-Bounded Practical Reasoning. *Computational Intelligence*, 4:349–355, 1988.
- [51] D. Brown. The Regional Crime Analysis Program (ReCAP): A Framework for Mining Data to Catch Criminals. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 3, pages 2848–2853 vol.3, 1998.
- [52] K. Cameron. The Laws of Identity, December 2005. <http://www.identityblog.com/?p=354>.
- [53] J. L. Camp. Digital identity. *Technology and Society Magazine, IEEE*, 23:34–41, 2004.
- [54] L. J. Camp and et al. Identity in Digital Government: A Research Report of the Digital Government Civic Scenario Workshop, 2003. Research Report.
- [55] A. Cavoukian. 7 Laws of Identity: The Case for Privacy-Embedded Laws of Identity in the Digital Age, 2006.
- [56] B. V. Cherkassky, A. V. Goldberg, and T. Radzik. Shortest Paths Algorithms: Theory and Experimental Evaluation. *Math. Program*, 73(2):129–174, 1996.
- [57] R. Clarke. Crime Proofing of Products: The Idea and the Substance. In *Technology and Society, 2003. Crime Prevention, Security and Design. ISTAS/CPTED 2003. Proceedings. 2003 International Symposium on*, pages 1–6, 2003.
- [58] R. V. Clarke. Situational Crime Prevention: Its Theoretical Basis and Practical Scope. In M. Tonry and N. Morris, editors, *Crime and Justice: An Annual Review of Research*, pages 225–256, 1983.
- [59] S. Clauß and M. Köhntopp. Identity Management and its Support of Multilateral Security. *Comput. Networks*, 37(2):205–219, 2001.
- [60] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction To Algorithms, Second Edition*. MIT Press, September 2001.
- [61] D. Cornish and R. V. Clarke. Introduction. *The Reasoning Criminal: Rational Choice Perspectives on Offending*, pages 1–16, 1986.
- [62] E. Damiani, S. D. C. di Vimercati, and P. Samarati. Managing Multiple and Dependable Identities. *Internet Computing, IEEE*, 7:29–37, 2003.
- [63] P. Davidsson. Agent Based Social Simulation: A Computer Science View. *Journal of Artificial Societies and Social Simulation*, 5, January 2002.
- [64] S. de Marchi. *Computational and Mathematical Modeling in the Social Sciences*. Cambridge University Press: Cambridge, 2005.

- [65] S. H. Decker and R. T. Wright. *Burglars on the Job: Streetlife and Residential Break-ins*. Northeastern University Press, 1994.
- [66] G. Del Castillo. Towards Comprehensive Tool Support for Abstract State Machines. In D. Hutter, W. Stephan, P. Traverso, and M. Ullmann, editors, *Applied Formal Methods — FM-Trends 98*, volume 1641 of *LNCS*, pages 311–325. Springer-Verlag, 1999.
- [67] G. Del Castillo and K. Winter. Model Checking Support for the ASM High-Level Language. In S. Graf and M. Schwartzbach, editors, *Proceedings of the 6th International Conference TACAS 2000*, volume 1785 of *LNCS*, pages 331–346. Springer-Verlag, 2000.
- [68] E. W. Dijkstra. A Note On Two Problems In Connection With Graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [69] E. W. Dijkstra. The Humble Programmer. *Communications of the ACM*, 15:859–866, 1972.
- [70] M. d’Inverno, M. Fisher, A. Lomuscio, M. Luck, M. de Rijke, M. Ryan, and M. Wooldridge. Formalisms For Multi-Agent Systems. In *First UK Workshop on Foundations of Multi-Agent Systems*, 1996.
- [71] M. d’Inverno and M. Luck. Formal Agent Development: Framework to System. In J. L. Rash, C. Rouff, W. Truszkowski, D. F. Gordon, and M. G. Hinchey, editors, *Formal Approaches to Agent-Based Systems, First International Workshop, FAABS 2000 Greenbelt, MD, USA, April 5-7, 2000, Revised Papers*, volume 1871 of *Lecture Notes in Computer Science*. Springer, 2001.
- [72] A. Drogoul and J. Ferber. Multi-Agent Simulation as a Tool for Studying Emergent Processes in Societies. In N. Gilbert and J. Doran, editors, *Simulating Society: The Computer Simulation of Social Phenomena*, chapter 6, pages 127–142. UCL Press, London, 1994.
- [73] A. Drogoul, D. Vanbergue, and T. Meurisse. Multi-Agent Based Simulation: Where Are the Agents? *Journal of Artificial Societies and Social Simulation*, 6, July 2002.
- [74] S. Easterbrook, R. Lutz, R. Covington, J. Kelly, Y. Ampo, and D. Hamilton. Experiences using lightweight formal methods for requirements modeling. *IEEE Transactions on Software Engineering*, 24(1):4–14, 1998.
- [75] S. Easterbrook, R. Lutz, R. Covington, J. Kelly, Y. Ampo, and D. Hamilton. Experiences using Lightweight Formal Methods for Requirements Modeling. *IEEE Transactions on Software Engineering*, 24(1):4–14, Jan 1998.
- [76] J. E. Eck and D. Weisburd. Crime Places in Crime Theory. In J. E. Eck and D. Weisburd, editors, *Crime and Place*. Criminal Justice Press, 1995.
- [77] P. Ekblom and N. Tilley. Going Equipped: Criminology, Situational Crime Prevention and the Resourceful Offender. *British Journal of Criminology*, pages 376–398, 2000.
- [78] J. M. Epstein and R. Axtell. *Growing Artificial Societies: Social Science from the Bottom Up*. Cambridge: MIT Press, 1996.
- [79] R. Eschbach, U. Glässer, R. Gotzhein, M. von Löwis, and A. Prinz. Formal Definition of SDL-2000: Compiling and Running SDL Specifications as ASM Models. *Journal of Universal Computer Science*, 7(11):1024–1049, 2001.
- [80] L. Fang, K. Hipel, D. Kilgour, and X. Peng. A Decision Support System for Interactive Decision Making - Part II: Analysis and Output Interpretation. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 33:56–66, 2003.

- [81] R. Farahbod. Extending and Refining an Abstract Operational Semantics of the Web Services Architecture for the Business Process Execution Language. Master's thesis, Simon Fraser University, Burnaby, Canada, July 2004.
- [82] R. Farahbod, V. Gervasi, and U. Glässer. CoreASM: An Extensible ASM Execution Engine. *Fundamenta Informaticae*, pages 71–103, 2007.
- [83] R. Farahbod, V. Gervasi, U. Glässer, and M. Memon. Design Exploration and Experimental Validation of Abstract Requirements. In *Proceedings of the 12th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'06)*. Essener Informatik Beiträge, June 2006.
- [84] R. Farahbod and U. Glässer. Semantic Blueprints of Discrete Dynamic Systems: Challenges and Needs in Computational Modeling of Complex Behavior. In *New Trends in Parallel and Distributed Computing, Proc. 6th Intl. Heinz Nixdorf Symposium, Jan. 2006*, pages 81–95. Heinz Nixdorf Institute, 2006.
- [85] R. Farahbod, U. Glässer, P. Jackson, and M. Vajihollahi. High Level Analysis, Design and Validation of Distributed Mobile Systems with CoreASM. In *Proceedings of 3rd International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2008)*, pages 797–814. Springer, October 2008.
- [86] R. Farahbod, U. Glässer, and M. Vajihollahi. An Abstract Machine Architecture for Web Service Based Business Process Management. *International Journal of Business Process Integration and Management*, 1:279–291, 2007.
- [87] M. S. Feather. Rapid Application of Lightweight Formal Methods for Consistency Analyses. *IEEE Trans. Software Engineering*, 24(11):949–959, 1998.
- [88] M. Felson. Routine Activities and Crime Prevention in the Developing Metropolis. *Criminology*, pages 911–931, 1987.
- [89] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison Wesley, February 1999.
- [90] FIPA. The foundation for intelligent physical agents, Last visited in June 2009. <http://www.fipa.org>.
- [91] M. Fisher. If Z is the Answer, What Could the Question Possibly Be? In J. P. Müller, M. J. Wooldridge, and N. R. Jennings, editors, *Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III*, volume 1193, pages 65–66. Springer-Verlag: Heidelberg, Germany, 12–13 1997.
- [92] M. Fisher and M. Wooldridge. Towards Formal Methods for Agent-Based Systems. In D. Duke and A. Evans, editors, *Proceedings of the Northern Formal Methods Workshop, Electronic Workshops in Computing*. Springer Verlag, 1997.
- [93] P. A. Fishwick. Computer Simulation: Growth Through Extension. *Transactions of Society of Computer Simulation*, 14(1):13–23, 1997.
- [94] J. W. Forrester. System Dynamics and the Lessons of 35 Years. In K. B. D. Greene, editor, *Systems-Based Approach to Policymaking*. Kluwer Academic Publishers, 1993.
- [95] M. Fowler. The New Methodology. April 2003. <http://martinfowler.com/articles/newMethodology.html>.

- [96] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.)*. Addison-Wesley, 2004.
- [97] S. Franklin and A. Graesser. Is it an Agent, or Just a Program? A Taxonomy for Autonomous Agents. In *ECAI '96: Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, pages 21–35, London, UK, 1997. Springer-Verlag.
- [98] Future of Identity in the Information Society – FIDIS. Website, Last visited in January 2008. <http://www.fidis.net>.
- [99] G. Gallo and S. Pallottino. Shortest Paths Algorithms. *Annals of Operations Research*, 13:3–79, 1988.
- [100] A. Gargantini, E. Riccobene, and P. Scandurra. A Metamodel-based Simulator for ASMs. In *Proc. of the 14th Intl. Abstract State Machines Workshop*, June 2007.
- [101] M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report logic-92-1, Computer Science Department, Stanford University, 1992.
- [102] M. P. Georgeff, B. Pell, M. E. Pollack, M. Tambe, and M. Wooldridge. The Belief-Desire-Intention Model of Agency. In *ATAL '98: Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, pages 1–10, London, UK, 1999. Springer-Verlag.
- [103] V. Gervasi and B. Nuseibeh. Lightweight Validation of Natural Language Requirements: A Case Study. In *ICRE '00: Proceedings of the 4th International Conference on Requirements Engineering (ICRE'00)*, page 140, Washington, DC, USA, 2000. IEEE Computer Society.
- [104] N. Gilbert. *Agent-Based Models*. SAGE Inc., 2007.
- [105] N. Gilbert and J. Doran. Simulating Societies: An Introduction. In N. Gilbert and J. Doran, editors, *Simulating Society: The Computer Simulation of Social Phenomena*, chapter 1, pages 1–19. UCL Press, London, 1994.
- [106] N. Gilbert and K. G. Troitzsch. *Simulation for the Social Scientist*. Open University Press, 1999.
- [107] N. Gilbert and K. G. Troitzsch. *Simulation for the Social Scientist*. Open University Press; 2nd edition, 2005.
- [108] U. Glässer, R. Gotzhein, and A. Prinz. The Formal Semantics of SDL-2000: Status and Perspectives. *Computer Networks*, 42(3):343–358, 2003.
- [109] U. Glässer and Q.-P. Gu. Formal Description and Analysis of a Distributed Location Service for Mobile Ad Hoc Networks. *Theoretical Comp. Sci.*, 336:285–309, May 2005.
- [110] U. Glässer, Y. Gurevich, and M. Veanes. Abstract Communication Model for Distributed Systems. *IEEE Trans. on Soft. Eng.*, 30(7):458–472, July 2004.
- [111] U. Glässer, S. Rastkar, and M. Vajihollahi. Computational Modeling and Experimental Validation of Aviation Security Procedures. In S. Mehrotra, D. D. Zeng, H. Chen, B. M. Thuraisingham, and F.-Y. Wang, editors, *Intelligence and Security Informatics, IEEE International Conference on Intelligence and Security Informatics, ISI 2006, San Diego, CA, USA, May 23-24, 2006, Proceedings*, volume 3975 of *Lecture Notes in Computer Science*, pages 420–431. Springer, 2006.

- [112] U. Glässer, S. Rastkar, and M. Vajihollahi. Modeling and Validation of Aviation Security. In H. Chen and C. Yang, editors, *Intelligence and Security Informatics: Techniques and Applications*, volume 135 of *Studies in Computational Intelligence*, pages 337–355. Springer, 2008.
- [113] U. Glässer and M. Vajihollahi. Computational Modeling of Criminal Activity. In *Proceedings of the European Conference on Intelligence and Security Informatics (EuroISI'08)*. Springer, 2008.
- [114] U. Glässer and M. Vajihollahi. Identity Management Architecture. In *Proceedings of IEEE International Conference on Intelligence and Security Informatics, ISI 2008*, pages 137–144. IEEE, 2008.
- [115] U. Glässer and M. Vajihollahi. Identity Management Architecture. *Annals of Information Systems Special Volume on Security Informatics*, 2009.
- [116] A. R. Golding and P. S. Rosenbloom. Improving Rule-Based Systems Through Case-Based Reasoning. In *National Conference on Artificial Intelligence*, pages 22–27, 1991.
- [117] L. Gunderson and D. Brown. Using a Multi-Agent Model to Predict Both Physical and Cyber Criminal Activity. *IEEE International Conference on Systems, Man, and Cybernetics*, 4:2338–2343, 2000.
- [118] Y. Gurevich. Evolving Algebras 1993: Lipari Guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1995.
- [119] Y. Gurevich. Sequential Abstract State Machines Capture Sequential Algorithms. *ACM Transactions on Computational Logic*, 1(1):77–111, July 2000.
- [120] A. Hall. Seven Myths of Formal Methods. *IEEE Software*, 7(5):11–19, 1990.
- [121] A. Harding. Dynamic microsimulation models: Problems and prospects. 1990. Discussion Paper No. 48.
- [122] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231–274, 1987.
- [123] J. Harper. *Identity Crisis: How Identification Is Overused and Misunderstood*. Cato Institute, 2006.
- [124] J. Haugeland. *Artificial Intelligence: The Very Idea*. Cambridge, Mass.: MIT Press, 1985.
- [125] J. Hendrick, T. Howell, D. London, E. Luehrs, M. Saliba, D. Brown, J. Dalton, F. Prats, and B. Johnstone. Webcat: the Design and Implementation of the Web-based Crime Analysis Toolkit. In *Proceedings of the 2004 IEEE Systems and Information Engineering Design Symposium*, pages 95–103, 2004.
- [126] V. Hilaire, A. Koukam, P. Gruer, and J.-P. Müller. Formal Specification and Prototyping of Multi-agent Systems. In A. Omicini, R. Tolksdorf, and F. Zambonelli, editors, *ESAW*, volume 1972 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2000.
- [127] C. A. R. Hoare. Communicating Sequential Processes. *Communications of ACM*, 21(8):666–677, 1978.
- [128] P. Hooimeijer and A. Oskamp. Advances in the Microsimulation of Demographic Behaviour. In L. J. G. v. Wissen and P. A. Dykstra, editors, *Population Issues: An Interdisciplinary Focus (The Plenum Series on Demographic Methods and Population Analysis)*, pages 229–263. Plenum Press, 1999.

- [129] Independent Centre for Privacy Protection Schleswig-Holstein, Germany and Studio Genghini & Associati, Italy. Identity Management Systems (IMS): Identification and Comparison Study, 2003.
- [130] ITU-T Recommendation Z.100 Annex F (11/00). *SDL Formal Semantics Definition*. International Telecommunication Union, 2001.
- [131] D. Jackson. Alloy: A Lightweight Object Modelling Notation. *ACM Trans. Software Engineering. Methodol.*, 11(2):256–290, 2002.
- [132] D. Jackson and J. Wing. Lightweight Formal Methods. *IEEE Computer*, (29):22–23, April 1996.
- [133] Javelin Strategy and Research. 2007 Identity Fraud Survey Report, February 2007.
- [134] O. Jensen, R. Koteng, K. Monge, and A. Prinz. Abstraction using ASM Tools. In A. Prinz, editor, *Proceedings of the 14th International ASM Workshop (ASM'07)*, 2007.
- [135] J. Jonas. Threat and Fraud Intelligence, Las Vegas Style. *Security & Privacy Magazine, IEEE*, 4:28–34, 2006.
- [136] C. B. Jones, D. Jackson, and J. Wing. Formal Methods Light. *IEEE Computer*, 29(4):20–22, 1996.
- [137] K. Kianmehr and R. Alhajj. Crime Hot-Spots Prediction Using Support Vector Machine. In *IEEE International Conference on Computer Systems and Applications*, pages 952–959, 2006.
- [138] K. Koffka. *Principles of Gestalt Psychology*. Harcourt, 1967.
- [139] J. Kolodner and D. Leake. A Tutorial Introduction to Case-Based Reasoning. pages 31–65, 1996.
- [140] D. B. Leake. *Case-Based Reasoning: Experiences, Lessons and Future Directions*. MIT Press, 1996.
- [141] Liberty Alliance. Liberty Alliance Identity Federation Framework (ID-FF) 1.2 Specifications, December 2007.
- [142] Liberty Alliance Project. Website, Last visited in January 2008. <http://www.projectliberty.org>.
- [143] H. Liu and D. Brown. Spatial-temporal Event Prediction: A New Model. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 3, pages 2933–2937 vol.3, 1998.
- [144] L. Liu and J. Eck, editors. *Artificial Crime Analysis Systems: Using Computer Simulations and Geographic Information Systems*. Information Science Reference, January 2008.
- [145] J. C. Lowery. Introduction to Simulation in Health Care. In *Proceedings of the 1996 Winter Simulation Conference*, pages 78–84, 1996.
- [146] M. Luck and M. d’Inverno. A Formal Framework for Agency and Autonomy. In V. Lesser and L. Gasser, editors, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 254–260, San Francisco, CA, USA, 1995. AAAI Press.
- [147] G. Z. Ma. Model Checking Support for CoreASM: Model Checking Distributed Abstract State Machines Using Spin. Master’s thesis, Simon Fraser University, Canada, May 2007.

- [148] C. Marling, M. Sqalli, E. Rissland, H. Munoz-Avila, and D. Aha. Case-based Reasoning Integrations. *AI Mag.*, 23(1):69–86, 2002.
- [149] M. B. Mary Lou Maher and D. M. Zhang. *Case-Based Reasoning in Design*. Lawrence Erlbaum Associates, 1995.
- [150] Y. L. Michael and I. H. Yen. Invited Commentary: Built Environment and Obesity Among Older Adults—Can Neighborhood-level Policy Interventions Make a Difference? *Am. J. Epidemiol.*, 169(4):409–412, 2009.
- [151] Microsoft FSE Group. *The Abstract State Machine Language*, 2003. <http://research.microsoft.com/fse/asml/>.
- [152] J. H. Miller and S. E. Page. *Complex Adaptive Systems: An Introduction to Computational Models of Social Life*. Princeton University Press, March 2007.
- [153] W. Müller, J. Ruf, and W. Rosenstiel. An ASM Based SystemC Simulation Semantics. In W. Müller et al., editors, *SystemC - Methodologies and Applications*. Kluwer Academic Publishers, June 2003.
- [154] C. E. Noon and F. B. Zhan. Shortest Path Algorithms: An Evaluation Using Real Road Networks. *Transportation Science*, 1996.
- [155] P. Perez, A. Dray, A. Ritter, P. Dietze, T. Moore, and L. Mazerolle. Simdrug: A Multi-agent System Tackling the Complexity of Illicit Drug Markets in Australia. In N. Gilbert, N. Ferrand, D. Batten, and P. Perez, editors, *Complex Science for a Complex World. Exploring human ecosystems with agents*, pages 193—223. Canberra Australia: ANU E Press, 2003.
- [156] J. Phiri and J. Agbinya. Modelling and Information Fusion in Digital Identity Management Systems. In *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on*, pages 181–187, 2006.
- [157] PISA – Privacy Incorporated Software Agent. Information Security, Privacy and Trust, Last visited in February 2008. http://iit-iti.nrc-cnrc.gc.ca/projects-projets/pisa_e.html.
- [158] J. Prentzas and I. Hatzilygeroudis. Integrations of Rule-Based and Case-Based Reasoning. In *The International Conference on Computer, Communication and Control Technologies (CCCT-03)*, volume IV, pages 81–85, 2003.
- [159] PRIME – Privacy and Identity Management for Europe. Website, Last visited in January 2008. <http://www.prime-project.eu>.
- [160] Public Safety and Emergency Preparedness Canada. Report on Identity Theft, October 2004.
- [161] C. D. Raab. Perspectives on ‘Personal Identity’. *BT Technology Journal*, 23:15–24, 2005.
- [162] G. F. Rengert. Burglary in Philadelphia: A Critique of an Opportunity Structure Model. In P. J. Brantingham and P. L. Brantingham, editors, *Environmental Criminology*, pages 189–201. Waveland Press Inc., 1991.
- [163] G. F. Rengert. The Journey to Crime: Conceptual Foundations and Policy Implications. In D. J. Evans, N. R. Fyfe, and D. T. Herbert, editors, *Crime, Policing and Place: Essays in Environmental Criminology*, pages 109–117. Routledge, 1992.

- [164] C. D. Rickett, S. Choi, C. E. Rasmussen, and M. J. Sottile. Rapid Prototyping Frameworks for Developing Scientific Applications: A Case Study. *The Journal of Supercomputing*, 36:123–134, May 2006.
- [165] L. Rocha. Complex Systems Modeling: using Metaphors from Nature in Simulation and Scientific Models, 2003. Los Alamos National Laboratory, <http://informatics.indiana.edu/rocha/complex/csm.html>.
- [166] D. K. Rossmo. *Geographic Profiling*. CRC Press, 2000.
- [167] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
- [168] J. D. Salt. Keynote Address: Simulation Should be Easy and Fun. In *Proceedings of the 1993 Winter Simulation Conference*, pages 1–5. New York: Association of Computing Machinery, 1993.
- [169] C. Satchell, G. Shanks, S. Howard, and J. Murphy. Beyond Security: Implications for the Future of Federated Digital Identity Management Systems. In *OZCHI '06: Proceedings of the 20th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artefacts and environments*, pages 313–316, New York, NY, USA, 2006. ACM.
- [170] G. Schellhorn and W. Ahrendt. Reasoning about Abstract State Machines: The WAM Case Study. *Journal of Universal Computer Science*, 3(4):377–413, 1997.
- [171] J. L. Schiff. *Cellular Automata: A Discrete View of the World*. Wiley & Sons, Inc., December 2007.
- [172] J. Schmid. *AsmGofer*, Last visited in July 2008. <http://www.tydo.de/Doktorarbeit/AsmGofer/>.
- [173] A. Seror. Simulation of Complex Organizational Processes: a review of methods and their epistemological foundations. In N. Gilbert and J. Doran, editors, *Simulating Societies*, chapter 2. UCL Press, 1994.
- [174] S. C. Shapiro. *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, 1992.
- [175] O. Shehory and A. Sturm. Evaluation of Modeling Techniques for Agent-Based Systems. In *Proceedings of the Fifth International Conference on Autonomous agents*, pages 624–631. ACM Press, 2001.
- [176] K. Singh. An Abstract Mathematical Framework for Semantic Modeling and Simulation of Urban Crime Patterns. Master's thesis, Simon Fraser University, Canada, September 2005.
- [177] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.
- [178] G. Smith. *The Object-Z Specification Language*. Kluwer Academic Publishers, 2000.
- [179] J. Sonnenfeld. Geography, Perception and the Behavioral Environment. In P. W. English and R. C. Mayfield, editors, *Man, Space and the Environment*, pages 244–251. Oxford University Press, New York, 1972.
- [180] J. M. Spivey. *The Z Notation: a reference manual*. Prentice Hall International Series in Computer Science, 2 edition, 1992.
- [181] S. Sproule and N. Archer. Defining Identity Theft. In *Management of eBusiness, 2007. WCM eB 2007. Eighth World Congress on the*, pages 20–31, 2007.

- [182] A. Srbljinovic and O. Skunca. Agent Based Modelling and Simulation of Social Processes. *Interdisciplinary Description of Complex Systems*, 1:1–8, 2003.
- [183] R. Stärk, J. Schmid, and E. Börger. *Java and the Java Virtual Machine: Definition, Verification, Validation*. Springer-Verlag, 2001.
- [184] K. P. Sycara. Multiagent Systems. *AI Magazine*, 19(2):79–92, 1998.
- [185] K. Teo, C. K. Chow, M. Vaz, S. Rangarajan, and S. Yusuf. The Prospective Urban Rural Epidemiology (PURE) Study: Examining the Impact of Societal Influences on Chronic Noncommunicable Diseases in Low-, Middle-, and High-income Countries. *American Heart Journal*, pages 1–7, Jul 2009.
- [186] M. Vajihollahi. High Level Specification and Validation of the Business Process Execution Language for Web Services. Master’s thesis, Simon Fraser University, Burnaby, Canada, April 2004.
- [187] G. van Blarkom, J. Borking, J. Giezen, R. Coolen, and P. Verhaar. *Handbook of Privacy and Privacy-Enhancing Technologies – The Case of Intelligent Software Agents*. College bescherming persoonsgegevens, 2003.
- [188] G. Wagner. Practical Theory and Theory-Based Practice. In J. P. Müller, M. J. Wooldridge, and N. R. Jennings, editors, *Proceedings of the ECAI’96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III*, volume 1193, pages 67–69. Springer-Verlag: Heidelberg, Germany, 12–13 1997.
- [189] G. Wang, H. Chen, J. Xu, and H. Atabakhsh. Automatically Detecting Criminal Identity Deception: An Adaptive Detection Algorithm. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 36:988– 999, 2006.
- [190] G. A. Wang, H. Atabakhsh, T. Petersen, and H. Chen. Discovering Identity Problems: A Case Study. In *LNCS: Intelligence and Security Informatics*. Springer Berlin / Heidelberg, 2005.
- [191] G. A. Wang, H. Chen, and H. Atabakhsh. A Probabilistic Model for Approximate Identity Matching. In J. A. B. Fortes and A. Macintosh, editors, *Proceedings of the 7th Annual International Conference on Digital Government Research, DG.O 2006, San Diego, California, USA, May 21-24, 2006*, pages 462–463. Digital Government Research Center, 2006.
- [192] W. Wang, Y. Yuan, and N. Archer. A Contextual Framework for Combating Identity Theft. *Security & Privacy Magazine, IEEE*, 4:30–38, 2006.
- [193] I. Watson and F. Marir. Case-based Reasoning: A Review. *The Knowledge Engineering Review*, (4):355–381, 1994.
- [194] P. Wiles and A. Costello. The ‘Road to Nowhere’: The Evidence for Travelling Criminals. *London: Policing and Reducing Crime Unit, Home Office Research, Development and Statistics Directorate*. , (Home Office Research Study No. 207), 2000.
- [195] P. J. Windley. *Digital Identity*, chapter Federating Identity, pages 118–142. O’Reilly, 2005.
- [196] J. M. Wing. Computational Thinking. *Communications of the ACM*, 4(3):33–35, 2006.
- [197] M. Wooldridge. Intelligent Agents: The Key Concepts. In M. Luck, V. Marik, O. Stepankova, and R. Trappl, editors, *Multi-Agent Systems and Applications II*, pages 3–43. Springer, 2001.
- [198] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley and Sons Ltd, 2002.

- [199] R. T. Wright and S. H. Decker. *Armed Robbers in Action: Stickups and Street Culture*. Northeastern University Press, 1997.
- [200] Y. Xue and D. Brown. A Decision Model for Spatial Site Selection by Criminals: A Foundation for Law Enforcement Decision Support. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 33:78–85, 2003.
- [201] L. Zadeh. Fuzzy Sets. *Information and Control*, (3):338–353, 1965.
- [202] F. B. Zhan. Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures. *Journal of Geographic Information and Decision Analysis*, 1(1):69–82, February 1997.