



# **Using Spatiotemporal Patterns to Qualitatively Represent and Manage Dynamic Situations of Interest: A Cognitive and Integrative Approach**

**Thèse**

**Foued Barouni**

**Doctorat en Informatique**

Philosophiæ Doctor (Ph.D.)

Québec, Canada

© Foued Barouni, 2016

# **Using Spatiotemporal Patterns to Qualitatively Represent and Manage Dynamic Situations of Interest: A Cognitive and Integrative Approach**

**Thèse**

**Foued Barouni**

Sous la direction de :

Bernard Moulin, directeur de recherche

# Résumé

Les situations spatio-temporelles dynamiques sont des situations qui évoluent dans l'espace et dans le temps. L'être humain peut identifier des configurations de situations dans son environnement et les utilise pour prendre des décisions. Ces configurations de situations peuvent aussi être appelées « situations d'intérêt » ou encore « patrons spatio-temporels ». En informatique, les situations sont obtenues par des systèmes d'acquisition de données souvent présents dans diverses industries grâce aux récents développements technologiques et qui génèrent des bases de données de plus en plus volumineuses. On relève un problème important dans la littérature lié au fait que les formalismes de représentation utilisés sont souvent incapables de représenter des phénomènes spatiotemporels dynamiques et complexes qui reflètent la réalité. De plus, ils ne prennent pas en considération l'appréhension cognitive (modèle mental) que l'humain peut avoir de son environnement. Ces facteurs rendent difficile la mise en œuvre de tels modèles par des agents logiciels.

Dans cette thèse, nous proposons un nouveau modèle de représentation des situations d'intérêt s'appuyant sur la notion des patrons spatiotemporels. Notre approche utilise les graphes conceptuels pour offrir un aspect qualitatif au modèle de représentation. Le modèle se base sur les notions d'événement et d'état pour représenter des phénomènes spatiotemporels dynamiques. Il intègre la notion de contexte pour permettre aux agents logiciels de raisonner avec les instances de patrons détectés. Nous proposons aussi un outil de génération automatisée des relations qualitatives de proximité spatiale en utilisant un classificateur flou. Finalement, nous proposons une plateforme de gestion des patrons spatiotemporels pour faciliter l'intégration de notre

modèle dans des applications industrielles réelles. Ainsi, les contributions principales de notre travail sont :

- Un formalisme de représentation qualitative des situations spatiotemporelles dynamiques en utilisant des graphes conceptuels.
- Une approche cognitive pour la définition des patrons spatio-temporels basée sur l'intégration de l'information contextuelle.
- Un outil de génération automatique des relations spatiales qualitatives de proximité basé sur les classificateurs neuronaux flous.
- Une plateforme de gestion et de détection des patrons spatiotemporels basée sur l'extension d'un moteur de traitement des événements complexes (Complex Event Processing).



# Abstract

Dynamic spatiotemporal situations are situations that evolve in space and time. They are part of humans' daily life. One can be interested in a configuration of situations occurred in the environment and can use it to make decisions. In the literature, such configurations are referred to as "situations of interests" or "spatiotemporal patterns". In Computer Science, dynamic situations are generated by large scale data acquisition systems which are deployed everywhere thanks to recent technological advances. Spatiotemporal pattern representation is a research subject which gained a lot of attraction from two main research areas. In spatiotemporal analysis, various works extended query languages to represent patterns and to query them from voluminous databases. In Artificial Intelligence, predicate-based models represent spatiotemporal patterns and detect their instances using rule-based mechanisms. Both approaches suffer several shortcomings. For example, they do not allow for representing dynamic and complex spatiotemporal phenomena due to their limited expressiveness. Furthermore, they do not take into account the human's mental model of the environment in their representation formalisms. This limits the potential of building agent-based solutions to reason about these patterns.

In this thesis, we propose a novel approach to represent situations of interest using the concept of spatiotemporal patterns. We use Conceptual Graphs to offer a qualitative representation model of these patterns. Our model is based on the concepts of spatiotemporal events and states to represent dynamic spatiotemporal phenomena. It also incorporates contextual information in order to facilitate building the knowledge

base of software agents. Besides, we propose an intelligent proximity tool based on a neuro-fuzzy classifier to support qualitative spatial relations in the pattern model. Finally, we propose a framework to manage spatiotemporal patterns in order to facilitate the integration of our pattern representation model to existing applications in the industry.

The main contributions of this thesis are as follows:

- A qualitative approach to model dynamic spatiotemporal situations of interest using Conceptual Graphs.
- A cognitive approach to represent spatiotemporal patterns by integrating contextual information.
- An automated tool to generate qualitative spatial proximity relations based on a neuro-fuzzy classifier.
- A platform for detection and management of spatiotemporal patterns using an extension of a Complex Event Processing engine.

# Contents

<b>Résumé .....</b>	<b>iii</b>
<b>Abstract.....</b>	<b>v</b>
<b>Contents .....</b>	<b>vii</b>
<b>List of Tables .....</b>	<b>xi</b>
<b>List of Figures.....</b>	<b>xii</b>
<b>Acronyms .....</b>	<b>xvii</b>
<b>Foreword.....</b>	<b>xx</b>
<b>Chapter 1 General Introduction .....</b>	<b>1</b>
Introduction .....	1
1.1    Problems and Research Issues .....	4
1.2    Objectives .....	7
1.3    Research methodology.....	8
1.4    Contributions .....	9
1.5    Organization of the Thesis.....	10
<b>Chapter 2 Qualitative Spatiotemporal Situation of Interest: A State of the Art .....</b>	<b>14</b>
Introduction .....	14
2.1    Large Scale Monitoring and Data Acquisition Systems .....	15
2.1.1    Big Data .....	15
2.1.2    Pervasive and Ubiquitous Computing.....	16
2.1.3    Examples of Large Scale Monitoring Systems in the Industry .....	17
2.1.4    Synthesis .....	18
2.2    Situation Awareness .....	20
2.2.1    Definition of Situation Awareness .....	20
2.2.2    Situation Management .....	23
2.2.3    Synthesis .....	24
2.3    Classical Approaches in Artificial Intelligence for Situation Modeling.....	25
2.3.1    Situation Calculus .....	25

2.3.2	Event Calculus .....	26
2.3.3	Synthesis .....	27
2.4	Situations of Interest Modeling Using Patterns .....	28
2.4.1	Predicate-Based Approaches.....	28
2.4.2	Query-Based Approaches .....	31
2.4.3	Synthesis .....	35
2.5	Complex Event Processing .....	36
2.5.1	Synthesis .....	39
2.6	Semantic-Based Approaches for Situation Modeling.....	40
2.6.1	Synthesis .....	45
2.7	Qualitative Spatiotemporal representation and reasoning .....	45
2.7.1	Qualitative Spatial Representation and Reasoning .....	46
2.7.2	Qualitative Temporal Representation and Reasoning.....	49
2.7.3	Synthesis .....	51
2.8	Discussion.....	51
2.8.1	Definition of Situation of Interest .....	52
2.8.2	Integration of Situations of Interest in Large Scale Systems .....	55
2.9	Conclusion .....	55
<b>Chapter 3 Qualitative Representation of Dynamic Spatiotemporal Patterns.....</b>		<b>58</b>
Introduction.....		58
3.1	Definition of Dynamic Spatiotemporal Patterns.....	59
3.1.1	From Situations of Interest to Spatiotemporal Patterns .....	59
3.1.2	Conceptual Graphs for Spatiotemporal Pattern Representation.....	62
3.2	A Dynamic Spatiotemporal Environment.....	65
3.2.1	Spatial Objects .....	66
3.2.2	Spatial Relations .....	67
3.2.3	Dynamic Spatiotemporal Situations.....	67
3.2.4	Temporal Relations.....	69
3.2.5	State.....	70
3.2.6	Event .....	72
3.2.7	Summary on the Spatiotemporal Situation Representation.....	74
3.3	Dynamic Spatiotemporal Pattern .....	75
3.3.1	Qualified Spatiotemporal Situation.....	76

3.3.2	Simple Pattern .....	77
3.3.3	Complex Pattern.....	79
3.3.4	Timer Pattern .....	81
3.3.5	Repetitive Pattern.....	83
3.3.6	Conclusion .....	85
3.4	Contextual Information.....	85
3.4.1	The role of context in knowledge representation.....	85
3.4.2	How to Define Contexts in Patterns? .....	86
3.4.3	Our Definition of Context in Patterns .....	87
3.4.4	Conclusion .....	89
3.5	Case Study: Outage Management Systems .....	90
3.5.1	Background.....	90
3.5.2	Proposed Approach.....	92
3.5.3	System Architecture.....	94
3.5.4	Pattern Specifications.....	94
3.6	Discussion.....	98
3.7	Conclusion .....	100
<b>Chapter 4 Using a Neuro-Fuzzy Classifier to Automatically Generate Spatial Proximity Quantifiers.....</b>		<b>101</b>
	Introduction .....	101
4.1	Qualitative Spatial Proximity.....	103
4.1.1	Distance-Based Approaches .....	104
4.1.2	Context-Based Approaches.....	107
4.1.3	Discussion .....	110
4.2	The Neurofuzzy Classifier Structure .....	112
4.3	Implementation Details.....	116
4.3.1	Training the Neurofuzzy Classifier.....	116
4.3.2	Results.....	120
4.4	A Qualitative Proximity Tool: Architecture and Implementation.....	123
4.5	Conclusion .....	125
<b>Chapter 5 A Framework for Managing Qualitative Spatiotemporal Patterns.....</b>		<b>127</b>
	Introduction .....	127

5.1	Motivations .....	128
5.2	Pattern Abstraction Module .....	130
5.3	Data Processing and Pattern Detection Module .....	132
5.3.1	Data Processing .....	136
5.3.2	Automated Pattern Conversion from Conceptual Graphs to EPL .....	138
5.4	A Spatial Extension of the CEP Engine .....	145
5.5	Software Architecture and Implementation .....	147
5.6	Case Study .....	150
5.6.1	Remote Fiber Test System .....	150
5.6.2	Background and Challenges .....	152
5.6.3	Proposed Solution .....	154
5.6.4	Spatial Environment .....	155
5.6.5	Pattern Specification .....	157
5.6.6	Using Contextual Information .....	160
5.7	Conclusion .....	165
<b>Chapter 6 Conclusion .....</b>		<b>166</b>
6.1	Synthesis .....	166
6.2	Contributions .....	167
6.3	Limits and Drawbacks .....	170
6.4	Future Work .....	171
<b>Bibliography .....</b>		<b>173</b>
<b>Appendix .....</b>		<b>183</b>

# List of Tables

Table 2-1: A summary of different approaches for situation of interest modeling.....53

Table 4-1: Fuzzy distance as proposed by [Brennan and Martin, 2006] ..... 105

Table 4-2: NFC features (inputs) ..... 117

Table 4-3: NFC Classes (outputs)..... 117

Table 4-4: NFC data set training preparation..... 119

Table 5-1: An example of simple pattern conversion algorithm..... 140

Table 5-2: Complex pattern conversion algorithm ..... 143

# List of Figures

Figure 1.1: IEC recommendations .....	4
Figure 1.2: Existing solutions to manage situations of interest .....	5
Figure 1.3: Organization of the thesis .....	12
Figure 2.1: A Typical Big Data Architecture.....	16
Figure 2.2: A difference between data and information [Endsley et al, 2012] .....	19
Figure 2.3: The general architecture of SAIL [Baader et al, 2009] .....	21
Figure 2.4: A semantic graph representation of a situation [Anagnostopoulos et al, 2006] .....	22
Figure 2.5: A situation represented using Description Logic [Anagnostopoulos et al, 2006] .....	23
Figure 2.6: An example of Situation Management approach [Jakobson et al, 2007] ..	24
Figure 2.7: Event Calculus Principle of Operation [Shanahan, 1999].....	27
Figure 2.8: A framework for pattern detection as defined by [Holzmann, 2007].....	29
Figure 2.9: An example of a spatiotemporal pattern [Gerhke et al, 2005].....	29
Figure 2.10: An example of a rule of association defined by [Lattner et al, 2006] ....	30
Figure 2.11: Different operations on pattern supported by [Lattner et al, 2006] .....	31
Figure 2.12: A general event definition according to [Etzion and Zolotorvesky, 2010] .....	37
Figure 2.13: A pattern expressing an excessive energy usage situation [Hasan et al, 2012] .....	37
Figure 2.14: A typical CEP architecture [Helmer et al, 2011].....	38
Figure 2.15: An EPL statement example .....	39
Figure 2.16: A spatiotemporal model of situations [Haddad and Moulin, 2010] .....	42
Figure 2.17: An example of a state [Haddad and Moulin, 2010].....	43
Figure 2.18: An example of an event [Haddad and Moulin, 2010] .....	44
Figure 2.19: An example of a process [Haddad and Moulin, 2010] .....	44
Figure 2.20: An example of spatial relations types [Holzmann and Ferscha, 2010] ..	47
Figure 2.21: RCC-8 topological relations from [Renz, 2002] .....	49



Figure 2.22: Qualitative temporal intervals relations [Allen, 1983] .....	51
Figure 3.1: A dynamic STP model.....	59
Figure 3.2: Representation of situations of interest using patterns (adapted from [Russell and Norvig, 1995] .....	60
Figure 3.3: Agent decision model for way finding problem. Adapted from [Freksa, 2007] .....	61
Figure 3.4: A Conceptual Graph example using linear and graph representations.....	64
Figure 3.5: Our conceptual model for spatiotemporal patterns. Adapted from [Haddad and Moulin, 2010] .....	65
Figure 3.6: An example of concept lattice including context specialization .....	66
Figure 3.7: An example of a topological relation between a Sensor and a Building using a Conceptual Graphs representation.....	67
Figure 3.8: Static and dynamic situations according to Desclés.....	68
Figure 3.9: A change from one state to another triggered by the occurrence of an event .....	68
Figure 3.10: Illustration of time intervals according to [Moulin, 1997] .....	69
Figure 3.11: An example of temporal relation using Conceptual Graphs representation .....	70
Figure 3.12: A state describing a normal fiber state. The state is defined in a time interval between July 17th at 10:00 AM and July 24th at 05:33 PM.....	71
Figure 3.13: A state describing a degraded fiber state. The state is defined at distance 23Km of a fiber link between Quebec and Montreal.....	71
Figure 3.14: A state represented using CGs' linear notation .....	71
Figure 3.15: An event representation using temporal relations to link with the state before the occurrence and after the occurrence.....	74
Figure 3.16: An example of a qualified situation.....	76
Figure 3.17: Structure of a simple pattern.....	78
Figure 3.18: An example of a simple pattern.....	79
Figure 3.19: A simple pattern example using the negation operator .....	79
Figure 3.20: Structure of a complex pattern.....	80

Figure 3.21: A complex pattern example where the communication_error_pattern is related with the temporal relation after to the emergency_pattern .....	81
Figure 3.22: An example of a timer pattern .....	82
Figure 3.23: Structure of the timer pattern.....	83
Figure 3.24: Structure of the repetitive pattern .....	84
Figure 3.25: An example of a repetitive pattern .....	84
Figure 3.26: Our approach to model support contextual information in our spatiotemporal pattern formalism .....	88
Figure 3.27: A pattern definition using contextual information .....	89
Figure 3.28: Different components of an Outage Management System.....	92
Figure 3.29: An outage map in the State of New York [conEdison, 2013] .....	92
Figure 3.30: Our agent's reasoning model .....	94
Figure 3.31: A system architecture involving the outage management system and the interaction with the goal-based agent.....	95
Figure 3.32: An example of simple pattern with linear notation .....	95
Figure 3.33: An example of rule used for crew assignment .....	96
Figure 3.34: Output of the action AssignFreeCrew in the Prolog+CG console. ....	97
Figure 3.35: A complex pattern representation.....	97
Figure 4.1: A simple pattern example using a qualitative spatial relation .....	102
Figure 4.2: An example of possible proximity relations between two objects [Schultz et al, 2007].....	104
Figure 4.3: A Java-based proximity platform developed by [Schultz et al, 2007], where fuzzy quantifiers are used to query spatial objects.....	106
Figure 4.4: A conceptual framework proposed by Brennan and Martin to compute impact area from contextual information and geographic distance. ....	108
Figure 4.5: An example of difference between influence areas [Kettani and Moulin, 1999] and impact area [Brennan and Martin, 2012] .....	108
Figure 4.6: A general architecture of the neuro-fuzzy system for proximity modeling proposed by [Yao and Thill, 2007] .....	110
Figure 4.7: An overview of the proposed approach.....	112
Figure 4.8: Partition of the feature space [Sun and Jang, 1993] .....	113

Figure 4.9: A neurofuzzy classifier, adapted from [Cetişli and Barkana, 2010] .....	115
Figure 4.10: Fuzzy rules used to prepare the data set .....	118
Figure 4.11: NFC training Performance for 3 features and 4 classes .....	121
Figure 4.12: The distance feature as defined by user to train the NFC.....	121
Figure 4.13: The distance feature after the NFC training .....	121
Figure 4.14: The road traffic feature as defined by user to train the NFC.....	122
Figure 4.15: The road traffic feature after the NFC training.....	122
Figure 4.16: User's familiarity with area feature as defined by user to train the NFC .....	122
Figure 4.17: User's familiarity with area feature after the NFC training.....	123
Figure 4.18: Architecture of our qualitative proximity tool.....	124
Figure 5.1: Overview of the different components of the proposed framework.....	129
Figure 5.2: Architecture of the pattern specification module .....	131
Figure 5.3: A concept type lattice using the Amine Platform.....	131
Figure 5.4: A simple pattern represented using the Amine Platform.....	132
Figure 5.5: A typical CEP architecture [Helmer et al, 2010].....	135
Figure 5.6: An example of sensor definition.....	137
Figure 5.7: A sample of events generated by a Fault Current Indicator sensor .....	137
Figure 5.8: An example of a class for event definition in ESPER.....	137
Figure 5.9: Description of the pattern detection module.....	138
Figure 5.10: Examples of simple patterns in CG and EPL formats .....	141
Figure 5.11: An output example of the complex pattern conversion algorithm .....	144
Figure 5.12: Architecture and software packages used in our framework.....	148
Figure 5.13: The Pattern Management Tool .....	149
Figure 5.14: OTDR Trace Information [Foa, 2015] .....	151
Figure 5.15: The general architecture of an RFTS.....	151
Figure 5.16: The Crew Manager's reasoning model .....	155
Figure 5.17: A metro network example in Quebec City. Fiber optic routes are represented by green lines and the RTU location is represented by a Yellow Square. .....	156
Figure 5.18: A specialization example of the concept type Location .....	156

Figure 5.19: An alarm report about an event generated by the “RTU Paris” as described by [NTest. 2014] RFTS .....	157
Figure 5.20: A Conceptual Graph representation of the event reported by “RTU Paris” .....	158
Figure 5.21: An example of a simple pattern where the event occurrence changes the state of a fiber from degraded to broken .....	159
Figure 5.22: An example of a complex pattern linking two simple patterns .....	160
Figure 5.23: A concept type lattice representing contextual information.....	161
Figure 5.24: A simple pattern example with spatial contextual information.....	162
Figure 5.25: A simple pattern example with semantic contextual information .....	163

# Acronyms

<b>AI</b>	Artificial Intelligence
<b>ANFIS</b>	Adaptive Neuro-Fuzzy Inference System
<b>B-DAD</b>	Big Data Analytics and Decisions
<b>BTS</b>	Base Transceiver Stations
<b>CDMA</b>	Code Division Multiple Access
<b>CDR</b>	Call Detail Records
<b>CEP</b>	Complex Event Processing
<b>CG</b>	Conceptual Graphs
<b>CIS</b>	Customer Information System
<b>CPC</b>	Context Propositional Content
<b>CPPC</b>	Complex Pattern Propositional Content
<b>CRM</b>	Customer Relation Management
<b>CTL</b>	Concept Type Lattice
<b>DBMS</b>	Database Management System
<b>DMS</b>	Distribution Management System
<b>DNP</b>	Distributed Network Protocol
<b>DSPM</b>	Data Stream Processing Models
<b>DSTP</b>	Dynamic Spatiotemporal Patterns
<b>EC</b>	Event Calculus
<b>ECA</b>	Event Condition Action
<b>EDA</b>	Event Driven Architecture
<b>ELE</b>	ETALIS Language for Events
<b>EMS</b>	Energy Management System
<b>EPC</b>	Event Propositional Content
<b>EPL</b>	Event Pattern Language
<b>ESA</b>	Event Spatial Attribute
<b>ETS</b>	Event Time Stamp
<b>FOR</b>	Fiber Optic Route
<b>GIS</b>	Geographic Information System
<b>GSM</b>	Global System for Mobile Communications
<b>IEC</b>	International Electro-technical Commission
<b>IE</b>	Inference Engine
<b>IVR</b>	Interactive Voice Response
<b>KPI</b>	Key Performance Indicators
<b>NFC</b>	Neurofuzzy Classifier
<b>NoSQL</b>	Not Only SQL
<b>OGC</b>	Open Geospatial Consortium

<b>OMS</b>	Outage Management System
<b>OSS</b>	Operations Support System
<b>OTDR</b>	Optical Time Domain Reflectometer
<b>QSR</b>	Qualitative Spatiotemporal Reasoning
<b>RCC</b>	Region Connection Calculus
<b>RDMS</b>	Relational Database Management Systems
<b>RFTS</b>	Remote Fiber Test System
<b>RPC</b>	Repetitive Pattern Count
<b>RPPC</b>	Repetitive Pattern Propositional Content
<b>RTU</b>	Remote Terminal Units
<b>SA</b>	Situation Awareness
<b>SAIDI</b>	System Average Interruption Duration Index
<b>SAIL</b>	Situation Awareness by Inference and Logic
<b>SC</b>	Situation Calculus
<b>SCADA</b>	Supervisory Control and Data Acquisition
<b>SFTP</b>	Secure File Transfer Protocol
<b>SLA</b>	Service Level Agreement
<b>SM</b>	Situation Management
<b>SPC</b>	State Propositional Content
<b>SPPC</b>	Simple Pattern Propositional Content
<b>SQL</b>	Standard Query Language
<b>SSA</b>	State Spatial Attribute
<b>STI</b>	State Time Interval
<b>STP</b>	Spatiotemporal Pattern
<b>STPS</b>	Spatiotemporal Pattern System
<b>STS</b>	Qualified Spatiotemporal Situation
<b>TPC</b>	Timer Pattern Content
<b>TPPC</b>	Timer Pattern Propositional Content

*To my parents.*

# Foreword

This thesis becomes a reality with the kind support and help of many people. I would like to extend my sincere thanks to all of them.

I'm grateful to the members of my thesis jury: Prof. Jules Desharnais, Prof. Christophe Claramunt, Prof. Luc Lamontagne, Prof. Sehl Mellouli and Prof. Thierry Badard for accepting to review this work, for their time and their interest.

I'm deeply indebted to my supervisor Prof. Bernard Moulin for his extended enthusiasm and continuous support throughout my Ph.D. studies. He gave me the opportunity to join his research group and he continuously encouraged me to go further in my work. I owe him a great debt of gratitude for his tremendous patience and for his valuable guidance.

I would like to thank all my colleagues and friends for their precious advices and constant encouragement.

My parents and sisters receive my love and gratitude. My hard-working parents have sacrificed their lives for my sisters and myself and they raised us with great values. My sisters have been my best friends all my life and I want to recognize their contribution to this achievement.

I saved the last word of acknowledgement for my dear wife Mouna, who has been with me all these years. Without her faithful patience and her unconditional encouragement especially during the critical moments, this work would not have been done.



# Chapter 1

## General Introduction

### Introduction

Recent advances in Computer Science (hardware and software) and in telecommunication systems led to the emergence of a new reality: a flood of data is being generated every day. According to IBM, 2.5 Exabytes (1 Exabyte=1 billion Gigabyte) were generated every day in 2012. This reality affects a large number of domains such as finance (e-trading, commercial transactions, credit cards), health (remote monitoring, patient management) and the social area (social networks, news, media streaming), to name a few. In this thesis, we focus on a new generation of data acquisition systems widely present in industries such as power utilities and telecommunications. In these areas, massive amounts of data are generated by a new generation of devices enhanced by powerful processing capabilities. These devices are deployed everywhere and they collect numerous observations about phenomena occurring in the world to report them in various data formats to centralized enterprise servers. They generate heterogeneous and dynamic data<sup>1</sup> which is also geo-referenced. The phenomena that we consider here have spatial and temporal characteristics. We will call them ‘spatio-temporal situations’.

---

<sup>1</sup> Data is said to be dynamic when it changes in space and time.

In recent years, many research teams attempted to cope with this new reality by offering a set of tools, frameworks and commercial products to manage such huge amounts of data. Managing this flood of data is about: 1) managing the way data is stored; 2) managing the interoperability between heterogeneous data formats; 3) offering the end-user efficient tools to take advantage of this flood of data. These efforts led to the emergence of new paradigms such as *Big Data*, *Internet of Things*, *Sensor Web*, *Everyware*, *Ambient Intelligence* and *Pervasive Computing*. One of the main objectives of these domains is to offer the end-user the possibility to make decisions related to dynamic spatiotemporal situations occurring in the world. This objective becomes crucial in the current context of competition between different companies which offer critical services such as power services, cellular operators, TV broadcasting services and internet providers, to name a few. Different key performance indicators (*KPIs*) are driving these domains such as SLA (*Service Level Agreement*) in telecommunications and SAIDI (*System Average Interruption Duration Index*) in power utilities. Existing solutions in these domains attempt to offer decision tools to help end-users visualizing interesting data. To this end, the end-user should be able to express “*what he wants to observe*” in the world so that existing tools need to query data collected from different sources and display the requested information.

Actually, in a large number of application areas, human operators need to identify *typical configurations (patterns)* in their operational environments based on the exploitation of observations provided by distributed devices and correlated with additional information which may be obtained from different kinds of static data sources such as Geographic Information Systems (GIS) and other enterprise applications. The human operator’s goal is to analyze (‘reason about’) these configurations in order to make appropriate decisions to manage/control the situations when a human intervention is needed. We will call these typical configurations *dynamic spatiotemporal situations of interest* because they are dynamic situations that are characterized by spatial and temporal properties and are particularly interesting for decision makers.

The notion of *situation of interest* plays a key role in humans' decision processes. We suggest that it needs to be carefully addressed by existing tools in a manner that suits human operators' cognitive expectations. Modeling a situation of interest as such is not a trivial task. It requires complex cognitive capabilities to model the semantics of dynamic objects that are involved in the situation and their changes. The proposal of an adequate representation model of situations of interest is one of the challenging aspects of this thesis.

This thesis takes place in the context of research dealing with the modeling and management of dynamic spatiotemporal situations applied to large scale acquisition and monitoring systems. These systems are widely deployed at different levels across organizations and generate data in heterogeneous structures. They are usually built up in monolithic and specialized application systems. The example depicted below (Figure 1.1) is adapted from the recommendations of the IEC (*International Electrotechnical Commission*) group under the *Smart Grid Initiative for the Power Utility Industry*. This group tries to define how software solutions can be deployed in large utilities, how they can interact (if possible) and how they can be implemented in organizations. A multilayer architecture is proposed starting at the *Market Level* where a *SCADA* (*Supervisory Control and Data Acquisition*) system and an *EMS* (*Energy Management System*) can be involved. The *Enterprise Level* comes after where a *GIS*, an *Asset Management System* and a *CIS* (*Customer Information System*) are deployed. The *Operation Level* makes the link between the Enterprise Level and the Field Level. A *SCADA*, an *EMS*, a *DMS* (*Distribution Management System*), an *OMS* (*Outage Management System*) are the main components used at this level for operation management purposes. Finally, the *Field Level* is specified to manage raw data retrieved from all devices and sensors deployed in the field such as *data concentrators*, *voltage regulators*, *breakers* and others which are commonly used in the power industry.

Similar hierarchical architectures can be found in other domains such as *security surveillance*, *weather monitoring* and *telecommunications*, to name a few. These

architectures have in common some aspects of volume, variety, heterogeneity, uncertainty, and distribution of data. Most of the proposed architectures seek to reduce the so-called ‘*semantic gap*’ between the big amount of data they generate and the end-user’s requirements to support the decision making process, which mainly aims at offering a certain level of quality and a continuous service (power, security, information, communication) to customers.

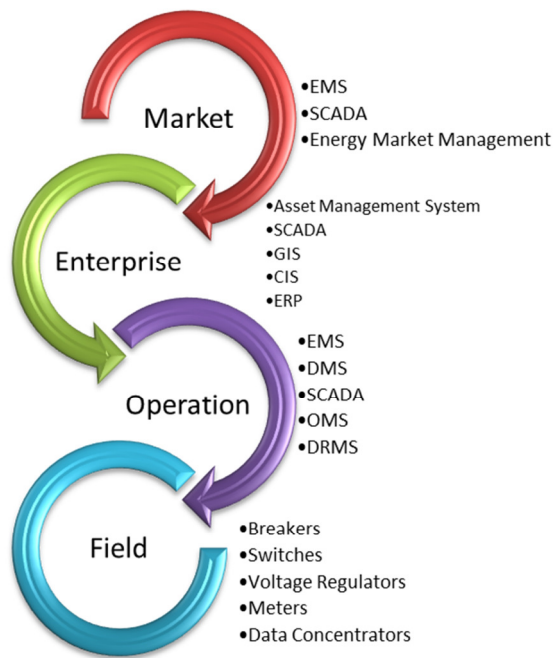


Figure 1.1: IEC recommendations

## 1.1 Problems and Research Issues

As we mentioned earlier, the main characteristics of large scale monitoring systems are variety, uncertainty, and heterogeneity of data. Figure 1.2 depicts an example about how data is processed from various sources and displayed to an end-user through a set of solutions and tools. Basically, data sources (sensors for example) generate observations in a raw data format which can be either stored as such or presented to the end-user in a legible form after some elementary processing. Such data may be displayed in various ways (alarms, views, tables, charts, graphs) or may

take the form of reports and files, or even be modeled as digital numbers or even by more sophisticated data structures capturing events and states. For decision making purposes, a user may need to retrieve data from different acquisition systems and to process it. For example, a user may be interested by some “situations” in relation to weather conditions in a particular location, at a certain time. Therefore, s/he needs to access a sensor web system for weather monitoring to get data about weather conditions (wind speed, wind direction etc...). Then, s/he may need to collect data from other systems such as a GIS and eventually establish relationships between the retrieved data to finally draw conclusions in the form of useful information presented for example on a map. In other words, the user needs to carry out a cognitive effort to interpret correlated data in terms of situations that he recognizes in the operational environment. Here, we define a *situation of interest* a set of correlated data which represents a specific configuration of the real world which is interesting for the end-user. In this thesis we consider that the user (or the operator) builds his/her own interpretation of the data obtained from acquisition systems, according to his cognitive (or ‘mental’) interpretation of the operational environment.

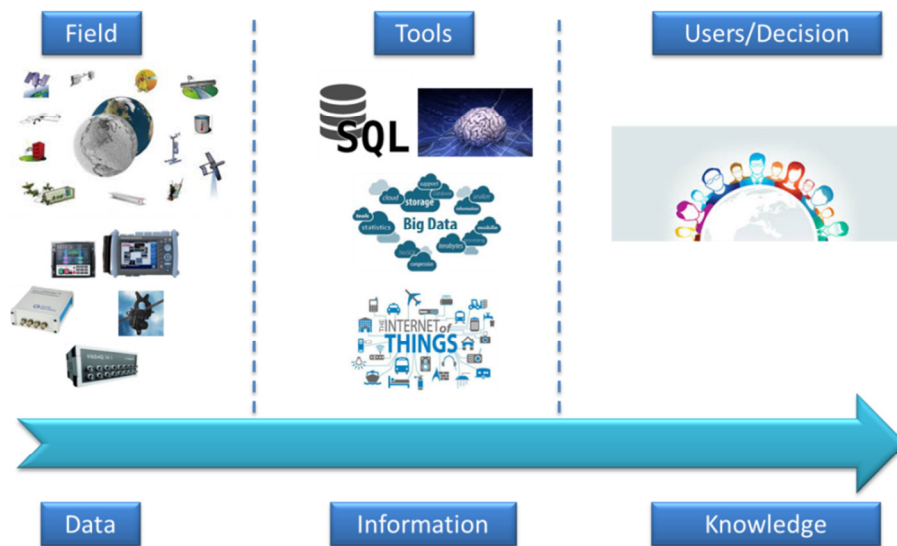


Figure 1.2: Existing solutions to manage situations of interest

Decision makers are considered as expert of their domains of expertise. They usually have multidisciplinary capabilities and they are application domain oriented. They need to identify situations of interest in order to make decisions in a competitive context, act on mission critical environments, and last but not least, comply with regulated markets. To this end, decision makers need to carry out the following main actions:

- Collect data from various data sources (online data streams, offline databases, statistical data, etc.)
- Identify dynamic spatiotemporal phenomena of interest
- Correlate these phenomena and identify situations of interest while taking advantage of their expertise.

Despite the large number of available sophisticated software solutions, decision makers carry out the above steps and make a cognitive effort because of the currently existing ‘semantic gap’ between what decision makers are looking for and what the proposed approaches are offering. Several approaches have been proposed to represent situations of interest, using various notions such as *spatiotemporal patterns*, *spatiotemporal phenomena* and *spatiotemporal situations*. A variety of formalisms and tools have been proposed to help querying information from large size databases or collecting data on the fly from data sources and to process them according to predefined knowledge structures. However, most of these approaches do not explicitly consider the way end-users reason about such situations of interest. In this thesis we suggest that there is a need to provide a clear definition of a situation of interest *that takes into account the user’s mental model* in order to reduce the existing semantic gap between the way data is presented by currently available approaches/software and what is a situation of interest from a user’s perspective. We argue that a qualitative and application independent representation formalism is needed to describe possible configurations of a situation of interest and to represent complex spatiotemporal phenomena. Current approaches in the literature offer limited capabilities to cope with the aforementioned issues.

## 1.2 Objectives

The objective of this thesis is twofold:

*To propose a knowledge engineering approach to represent and manage qualitative spatiotemporal situations of interest:* this approach will offer a methodology to build a cognitive approach to represent and manage situations of interest in large scale data acquisition systems. This objective can be achieved through the following sub-objectives:

- *To propose a model that represents dynamic spatiotemporal situations of interest in a qualitative manner.* The formalism should allow for expressing situations of interest using a representation language close to Natural Language. The formalism should also consider the user's mental perception of the environment and allow for reasoning about situations of interest using intelligent software systems.
- *To propose an approach which supports qualitative representation and reasoning with spatial relations:* this approach will be used to help users building a mental model of the spatial environment and to integrate qualitative spatial relations in the definition of situations of interest. Particular emphasis should be placed on spatial proximity relations.

*To propose an approach and a set of tools to manage instances of situations of interest occurring in a dynamic spatiotemporal environment.* To validate our approach, we propose two application domains: a Remote Fiber Optic Monitoring System which is used in the telecommunication industry and an Outage Management System which is used in the public power industry. We will develop our approach and the set of tools to help users to manage different tasks aiming at maintaining a

certain level of quality of service and to react in an efficient way to critical situations that may occur in their operational environments.

### 1.3 Research methodology

To achieve the objectives of this thesis, we adopted a four-steps approach 1) conducting a literature review of research works on spatiotemporal situation of interest modeling and related subjects; 2) proposing a knowledge engineering approach to represent and manage spatiotemporal situations of interest; 3) developing a set of tools to detect instances of spatiotemporal situation of interest in dynamic environments.; 4) Validating the proposed approach using two case studies.

The state of the art is the first step in our research methodology where we carried out an in-depth analysis of existing approaches on modeling dynamic situations of interest. In the literature, researchers usually refer to situations of interest using the term *spatiotemporal patterns*. For the rest of this thesis, we refer to dynamic situations of interest as dynamic spatiotemporal patterns (DSTP). Our literature review on DSTP addresses several perspectives. *The representation perspective* deals with the issue of how DSTP are represented in terms of syntax and semantics. *The cognitive perspective* deals with the way DSTPs are used in knowledge-based approaches. *The application perspective* attempts to review different research areas and application domains involving the representation of DSTP and reasoning with them. In particular, we will discuss domains such as big data, ubiquitous computing, ambient intelligence, location/context aware systems, pervasive computing, Artificial Intelligence and Complex Event Processing.

Reviewing current approaches to qualitatively represent and manage situations of interest is a mandatory step which will help identifying their current advantages, limits and shortcomings. Hence, as a next step of our research work, we will propose



a novel approach to qualitatively represent situations of interest by considering several requirements such as offering a qualitative representation and the possibility of using such an approach using a cognitive perspective. This will imply selecting a suitable representation formalism and providing a clear definition of a dynamic situation of interest. Both spatial and temporal dimensions need to be supported as well as other fundamental notions such as events and states.

The development of software tools to support the proposed approach is the third step of the thesis. We will develop a framework to manage instance of situations of interest which can be detected in a dynamic spatiotemporal environment. To this end, a hybrid architecture based on the integration of our formal definition of situations of interest to a pattern detection framework will be proposed. The proposed framework is integrated with a Complex Event Processing engine which is used for pattern detection.

The validation of the proposed approach is the fourth step of the thesis in which two scenarios from telecommunication and power distribution industries are used to illustrate the application of the main aspects of our research work and to emphasize their usefulness and applicability.

## 1.4 Contributions

**As a first contribution**, this thesis presents a novel knowledge engineering approach to qualitatively represent and manage dynamic spatiotemporal situations of interest. While most works on spatiotemporal patterns focus on extending database query languages, our approach tackles the issue of DSTP modeling from a cognitive perspective. To define a DSTP we take into account a cognitive representation of the operational environment and propose a knowledge engineering methodology to build the DSTP model. To this end, our approach uses the rich semantic capabilities of Sowa's Conceptual Graphs formalism and its equivalence to First Order Logic. This

will open the possibility to couple our software tools to be used with agent-based systems in order to enable them to manipulate patterns and to reason about them. Furthermore, we introduce the notion of contextual knowledge in our formalism to enhance and refine the definition of pattern. Last but not least, since this thesis deals with dynamic environments, our pattern formalism represents dynamic spatiotemporal phenomena and allows for the explicit representation of states, events and temporal relations between them.

To integrate the proposed DSTP model to spatial environments and to support qualitative spatial reasoning capabilities, **the second main contribution** of this thesis is a qualitative technique to represent and to reason about spatial relations. The proposed approach uses a Neurofuzzy classifier to integrate contextual knowledge in the qualification of spatial proximity. Using the proposed technique, qualitative spatial quantifiers are closer to the human's mental model of the spatial environment and are used to represent spatial proximity relations in our DSTP model.

**The third main contribution** of this thesis is a set of tools and algorithms to manage spatiotemporal pattern instances from a stream of data generated by large scale applications. Actually, in our first contribution we propose a qualitative DSTP model but it does not address the issue of detecting DSTP instances in the context of real applications. Hence, we developed a hybrid framework integrated with an existing *Complex Event Processing* engine and we created a set of algorithms to implement our semantic DSTP model and to manage the detection of DSTP instances. These detected instances of patterns can be used to build knowledge bases that can be exploited by software agents.

## 1.5 Organization of the Thesis

This thesis is organized as follows (Figure 1.3). After introducing different research issues in the current chapter, **Chapter 2** presents a state of the art of different works

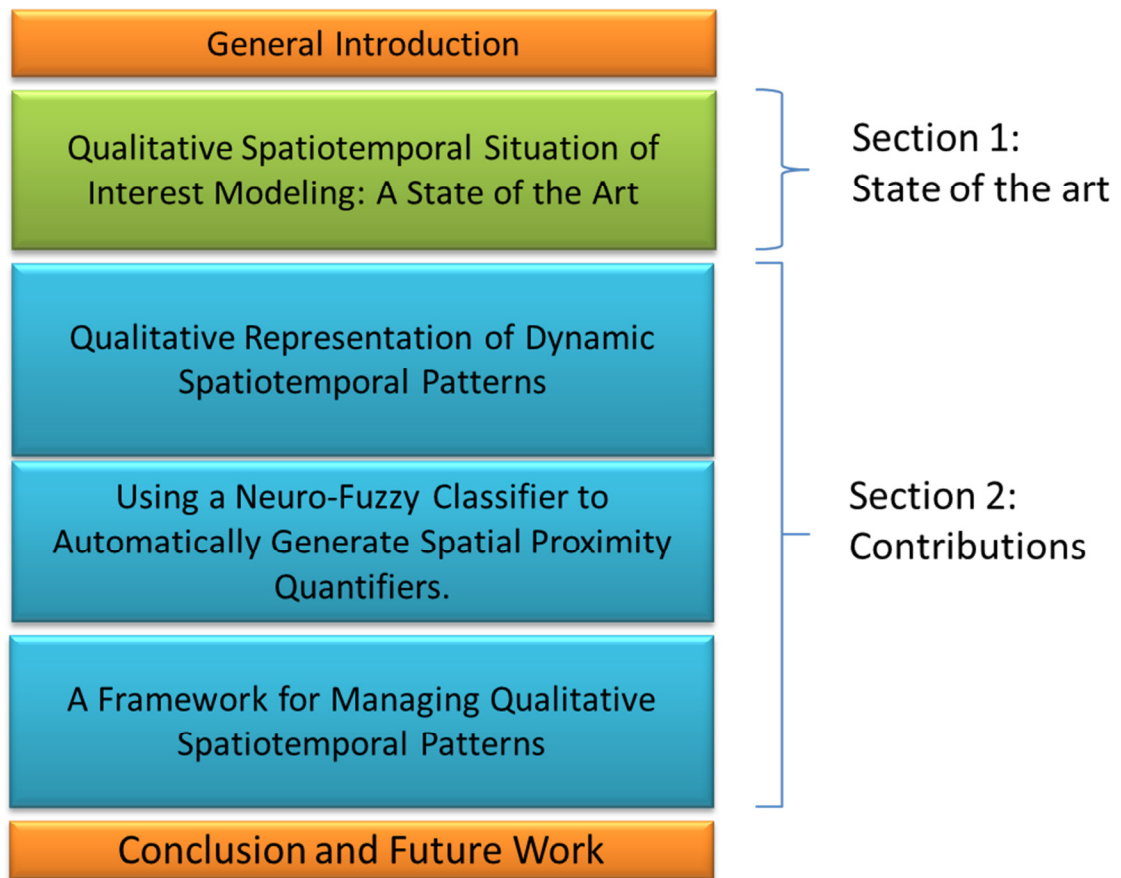
on modeling situations of interest. In this chapter we present the context of large scale monitoring and data acquisition systems and discuss the limits of current approaches with respect to the objectives of this thesis.

*Chapter 3* presents our first contribution and proposes a novel formalism to qualitatively represent spatiotemporal patterns using Conceptual Graphs. These spatiotemporal patterns represent dynamic spatiotemporal situations while adopting a cognitive perspective and taking into account contextual information. The chapter includes a short case study to illustrate some examples of the proposed formalism and to raise some difficult issues that will be solved in the next chapters.

Spatial relations are a key element in the spatiotemporal formalism that we propose in Chapter 3. In particular, spatial proximity is used in several applications to qualify the proximity between spatial references. In *Chapter 4* we propose an automated approach based on a neurofuzzy classifier to generate qualitative spatial proximity relations. These relations are integrated in a GIS and used in our pattern formalism.

Some challenges related to pattern detection and management discussed in Chapter 3, are now addressed in *Chapter 5* which presents the framework that we developed to manage spatiotemporal patterns. The framework integrates a Complex Event Processing engine and the qualitative spatial proximity tool that we presented in Chapter 4, as well as our qualitative spatiotemporal pattern model described in Chapter 3. A case study illustrates different aspects of our solution.

Finally, Chapter 6 concludes the thesis by summarizing its main contributions and by outlining some limits that will be explored in future research work.



*Figure 1.3: Organization of the thesis*

# **Part I**

## **State of the Art**

## Chapter 2

# Qualitative Spatiotemporal Situation of Interest: A State of the Art

### Introduction

Several research areas such as Psychology, Spatiotemporal Reasoning and Artificial Intelligence (AI) investigated situation modeling. In AI, several approaches and formalisms were proposed for situation modeling such as *Situation Calculus*, *Situation Awareness*, *Event Calculus* and *Spatiotemporal Patterns*, to name a few. In this chapter, a state of the art of spatiotemporal situation modeling is presented.

The rest of this chapter is organized as follows. Section 2.1 presents a brief overview of large scale monitoring and data acquisition systems. Section 2.2 introduces the concept of Situation Awareness which is a widely used approach for situation modeling. Section 2.3 surveys some Artificial Intelligence approaches for situation modeling such as Situation and Event Calculi. In Section 2.4, we present an overview of works based on patterns to represent situations of interest. Section 2.5 introduces Complex Event Processing and discusses how this technology is used to manage and detect situations of interest. In Section 2.6 we review some approaches which address the semantic aspects of situation modeling. Section 2.7 surveys classical qualitative spatiotemporal representation and reasoning techniques. Section 2.8 discusses the

limits of these approaches and positions them in the context of this thesis. Finally, Section 2.9 concludes the chapter.

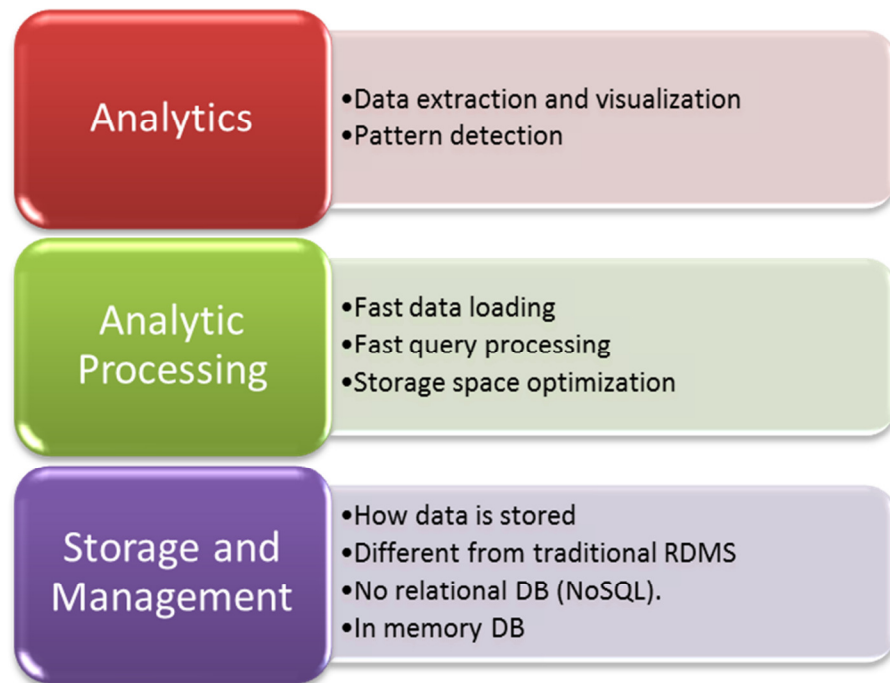
## **2.1 Large Scale Monitoring and Data Acquisition Systems**

This thesis is defined in the context of a new generation of data acquisition and monitoring systems which are addressed in the literature by several research areas. Without being exhaustive, we propose in this subsection a brief overview of these concepts.

### **2.1.1 Big Data**

“Big Data” is a term that applies to systems where datasets are constantly increasing until they become too large to be managed using traditional database management systems [Elgendy and Elragal, 2014]. Consequently, advanced analytics techniques are used on big datasets in order to deal with challenges related to dataset management such as capture, storage, search and visualization. According to [Elgendy and Elragal, 2014] three main features characterize big data: volume, variety and velocity (namely the three V’s). The size of the dataset corresponds to the volume. The variety of the dataset is related to the heterogeneity aspect of data. The velocity of datasets is the way data is changing and how often it is generated. These characteristics led to the definition of new requirements for developing tools capable of fast access and efficient analysis of these datasets while these capabilities are not supported by traditional database management systems. For example, the Big Data Analytics and Decisions (B-DAD) [Elgendy and Elragal, 2014] is a framework that establishes a link between Big Data Analytics tools and Decision making systems. A typical (B-DAD) architecture is depicted in Figure 2.1. A *Storage and Management Module* is used to deal with data storage. It can be stored in typical hard disks or hosted in memory (called in-memory DBs structure) in order to optimize the read/write access speed. Other aspects may be addressed in this module such as how the data repository is structured. It can be different from traditional relational

databases and it can use the so-called *Not Only SQL* approach (NoSQL) which separates data management and data storage for better performances. The *Analytic Processing Module* addresses issues such as fast data loading, fast data processing and storage space optimization. Finally, the *Analytics Module* addresses data extraction and visualization issues and pattern detection mechanisms.



*Figure 2.1: A Typical Big Data Architecture*

It is worth to mention that Big Data is still considered as an emerging set of concepts and ideas which make this paradigm continuously evolving. This explains why no clear and unique definition can be found yet and that some of the existing definitions overlap with other domains such as Pervasive Computing.

### **2.1.2 Pervasive and Ubiquitous Computing**

Pervasive Computing is a paradigm that emerged in the first decade of the 21<sup>st</sup> century. It is based on the idea of providing access to applications everywhere, anytime, by means of any device and through natural interactions so that users may



not even be aware that they are using computational devices [Cascado et al, 2011]. Ambient Intelligence and Ubiquitous Computing are other approaches related to Pervasive Computing. The development and implementation of pervasive applications emerged in various domains such as Internet technologies, mobile and distributed computing, computer hardware, wireless communication networks, sensor networks and intelligent systems, to name a few [Obaidat et al, 2011]. The research community on pervasive computing identified several challenges related to three main issues [Saha and Mukherjee, 2003]: *scalability*, *heterogeneity*, and *context-awareness*. A system is *scalable* when it allows for adding new devices without the need to design a new application and allows for deploying a large number of such devices. A system is *heterogeneous* when it allows for using a variety of services and different types of devices, networks, systems and environments. *Context Awareness* is defined as the use of information to characterize the situation of an entity, be it a person, a place or an object [Dey, 2001]. One of the most popular categories of context-aware systems is *location aware systems*. We emphasize here that the concept of context is characterized by different dimensions such as time, space and user's preferences. According to [Cascado et al, 2011] user's preferences change according to his context. Therefore, context aware-systems must adapt to the user when needed; and provide the following three characteristics: 1) present information and services to a user; 2) automatically execute services for a user; 3) tag the context related to the provided information for later use [Dey, 2001].

### **2.1.3 Examples of Large Scale Monitoring Systems in the Industry**

Several domains are concerned with challenges addressed by Big Data and Pervasive Computing systems. They share a common set of characteristics such as very large datasets and the need to offer decision making capabilities to end-users.

In the telecommunication industry, Fiber Quality Monitoring Systems use laser-based sensors to scan fiber optic links, detect critical events such as fiber degradations and fiber breaks and locate these events using a GIS. They help users to reduce network

downtime and respect service level agreements. Operations Support Systems (OSS) are widely used in telephone networks to support operation functions such as network configuration and monitoring, service commissioning and asset management.

In the Power Utilities industry, Outage Management Systems use smart sensors which are deployed throughout the distribution network. These sensors are mounted on power cables and they report an outage event to centralized servers using cellphone communication channels and industrial communication protocols. The reported events are also located using a GIS which models the distribution network. The main function of an OMS is to track outage events and to update customers with the real time outage status in their respective regions. Other large scale monitoring systems are used in the Power industry such as Energy Management System (EMS) to monitor, measure, control and optimize electric generation and transmission networks and Distribution Management System (DMS) to optimize and control electric distribution networks.

#### **2.1.4 Synthesis**

Various architectures and approaches are proposed by Big Data, Pervasive Computing and other similar paradigms to manage large scale systems and their datasets. These architectures share some common aspects such as variety, heterogeneity and distribution of data sources. Since these systems are used by humans on a daily basis, they try to reduce the semantic gap between the data they provide and the end-user's requirements to support his decision making process. Hence, the more supportive of end-users' decision making process a system is, the more it is useful and efficient. For example, [Endsley et al, 2012] claimed that a user-centric design should be considered rather than a data-centric design for the new generation of computer software. This will help turning raw data into relevant information (Figure 2.2). This information could be visual (alarms, views, tables, charts, graphs) or could take the form of reports and files. It can also be categorized as events, states, digital numbers, or more generally, as dynamic spatiotemporal

phenomena. For decision making purposes, a user may need to collect this information from various systems and reason about it. For example, a user may be interested by some “situations” about traffic conditions in a given city, at a certain time. Therefore, he needs to access cameras providing information about traffic conditions. Then, he may need to get information from other systems such as a GIS, and eventually, establish relationships between all these retrieved information to draw conclusions. In other words, users need to carry out cognitive processes to transform heterogeneous data into recognized situations of interest according to the way they perceive their environment. A *situation of interest* (also called a pattern<sup>2</sup>) can be thought of as the result of the correlation of a sequence of information to identify a specific configuration of the real world. Human beings have mental representations of the world and carry out their own interpretations of the information collected from system outputs, according to the way they perceive the surrounding environment.

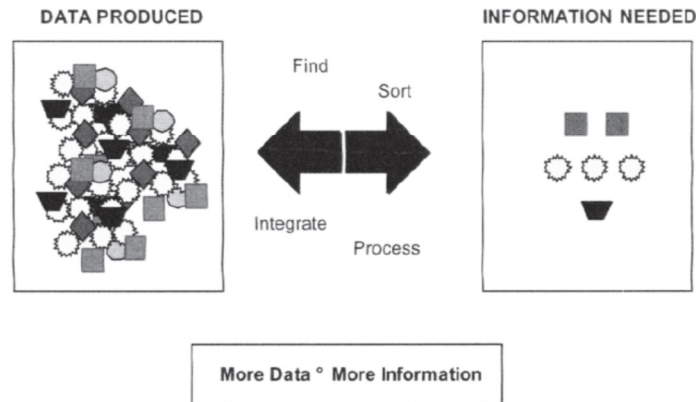


Figure 2.2: A difference between data and information [Endsley et al, 2012]

The rest of this chapter reviews different approaches proposed to model spatiotemporal situations of interest. Our research on this topic led us to conclude that works on situations of interest can be subdivided into several sub-families. There are approaches based on situation awareness and situation management. There are classical approaches in Artificial Intelligence that use logic to reason about situations.

<sup>2</sup> Several works in the literature use the word “pattern” to refer to “situations of interest”. We will review these works in the next sections of this chapter.

There are also approaches that propose to represent situations of interest as spatiotemporal patterns such as SQL-based approaches and event-driven approaches. Finally, there are approaches which address the semantic aspects of different situations' components such as events, states and spatiotemporal phenomena in general. To align this review with the context of this thesis, we will try to evaluate how these approaches deal with the following issues:

- How situations of interest are represented and how close these representation formalisms are to Natural Language?
- How dynamic and complex situations are represented?
- How situations of interest are detected and managed?
- How situations of interest are used from a cognitive perspective?
- How the proposed approaches can be integrated in real computing solutions currently existing in the industry?

## **2.2 Situation Awareness**

A quick Google search on spatiotemporal situations yields a multitude of references to situation awareness and situation management. In this section, we present a brief overview of these concepts.

### **2.2.1 Definition of Situation Awareness**

A formal definition of Situation Awareness (SA) is given by [Endsley, 1988]: “SA is the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future”. Originally, SA has been carefully studied in the military domain. Then, the notion and the related techniques have been generalized to other application domains where a user needs to be aware of what is happening around him and which information is important for him in order to intervene appropriately. In other words, “being situation aware” means to have the information which is the most important to make decisions. For example, a car driver must be aware of the relative distances of

other cars surrounding his car in the traffic, the car speed and the fuel level; but the user does not need to know what the car's engine characteristics are. According to [Endsley et al, 2012], SA can be defined through several main stages: 1) *Perception of the elements* in the environment using different means (visual, auditory), sensors (in general) and a combination of these means; 2) *Comprehension of the current situation*, which consists in integrating many pieces of the perceived data to build information and in defining a priority of each piece of information according to its importance, reliability and meaning; 3) *Projection of future status*. At this stage, a user can take advantage of his comprehension of the current situation (stage 2) to anticipate (predict) what will happen in the future and to choose a course of actions. Several SA systems have been proposed to support the management of a large number of data sources such as sensors, textual information and databases [Baader et al, 2009] [Fischer et al, 2014]. These systems have been applied to different domains. For example [Baader et al, 2009] developed a system called SAIL (Situation Awareness by Inference and Logic) applied to the military domain which uses surveillance data and a formal definition of situations based on events. The authors used a Prolog-like syntax to define rules implementing automated reasoning capabilities. The high level system architecture of SAIL is depicted in Figure 2.3.



Figure 2.3: The general architecture of SAIL [Baader et al, 2009]

Anagnostopoulos and colleagues [2006] proposed a semantic definition of situations based on interrelated concepts and using contextual information. Contextual information describes spatial context, temporal context, artifact context (which represents the context of the user's computational entity (operating system, PDA, etc.) and personal context which is the role that a user may play in a given situation. A semantic graph representation of a situation is illustrated in Figure 2.4 and a formal representation using Description Logic is given in Figure 2.5.

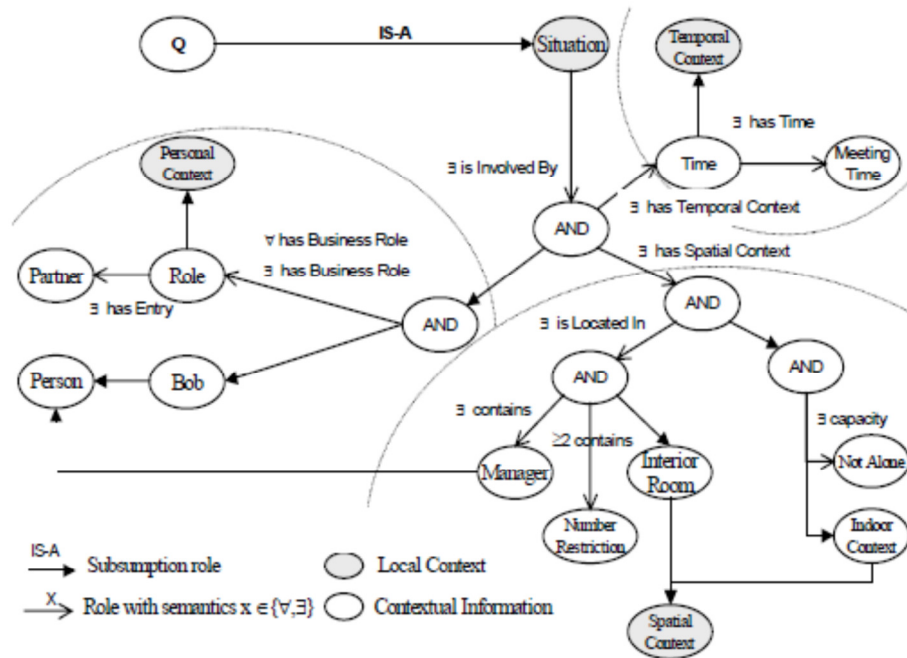


Figure 2.4: A semantic graph representation of a situation [Anagnostopoulos et al, 2006]

Situation Awareness is a multidisciplinary domain where researchers contributed to different aspects such as data collection, user interface, situation modeling and spatiotemporal analysis. Creating a synergy between all these contributions became a challenge and led to the emergence of the Situation Management paradigm.

$Q \subseteq \text{Situation} \sqcap (\exists \text{ is Involved By. (Bob} \sqcap$ $\exists \text{ has Time. Meeting Hour} \sqcap$ $\exists \text{ is Located In. (Interior Room} \sqcap \exists \text{ contains. Manager)} \sqcap$ $\exists \text{ has Business Role. Partner} \sqcap$ $\forall \text{ has Business Role. Business Partner}))$
$\text{Formal Meeting} \subseteq \text{Meeting} \sqcap (\exists \text{ is Involved By. (Partner} \sqcap$ $\exists \text{ has Time. Meeting Hour} \sqcap \exists \text{ is Located In.}$ $\quad (\text{Meeting Room} \sqcap$ $\quad \exists \text{ contains. Manager} \sqcap$ $\quad \exists \text{ contains. Business Partner)}) \sqcap$ $\exists \text{ has Business Role. Partner} \sqcap$ $\forall \text{ has Business Role. Business Partner}))$

Figure 2.5: A situation represented using Description Logic [Anagnostopoulos et al, 2006]

## 2.2.2 Situation Management

Situation Management (SM for short) is a research area that aims at finding a synergy between different contributions in situation representation and reasoning approaches. SM is “a framework of concepts, models and enabling technologies for recognizing, reasoning about, acting on, and predicting situations that are happening or might happen in dynamic systems during a pre-defined operational time.” [Jakobson et al, 2007]. Managing situations is a goal-directed process which involves different aspects [Jakobson et al, 2007].

- Sensing and information collection;
- Perceiving and recognizing situations;
- Analyzing past situations and predicting future situations;
- Reasoning, planning and implementing actions to reach desired goals.

Figure 2.6 depicts a general process loop of situation management. The core of SM is the situation model which is said to be *investigative* if the situation model aims at recognizing existing situations while considering situations that occurred in the past. The situation model is said to be *predictive* when it aims at predicting future

situations. A real situation occurring in the world is sensed and perceived in the form of *events*. The problem solving module uses the situation model's output to propose an action plan. In this model we note that the transition between real situations and events through sensing is confusing and limits the characterization of a situation to the use of events only.

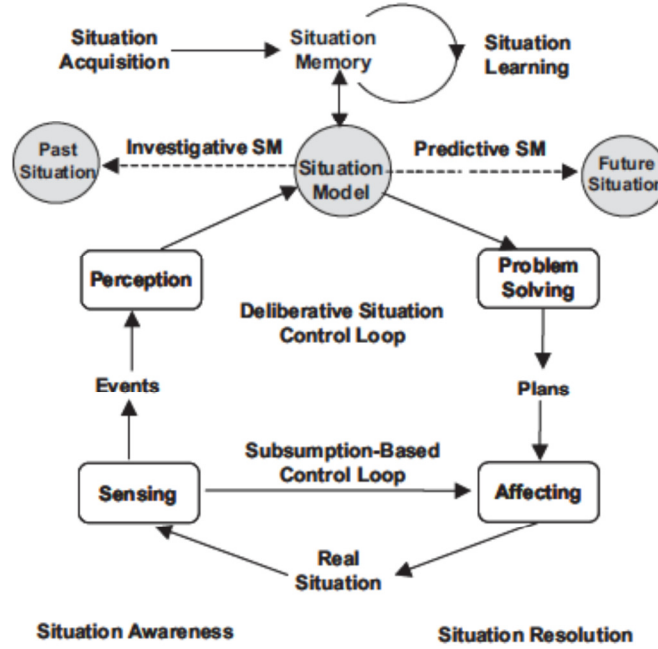


Figure 2.6: An example of Situation Management approach [Jakobson et al, 2007]

### 2.2.3 Synthesis

Situation Management is based on three core processes which are Situation Modelling, Situation Recognition and Situational Reasoning. Both with Situation Awareness, they integrate approaches from different disciplines such as Artificial Intelligence, Situation Awareness, Information Fusion, Multi-Agent Systems and Sensor Networks. The main contribution of these approaches is the organization of data in a layered architecture to adequately present it to end-users. However, one of the major shortcomings of SA is the absence of a clear definition of a situation of interest. For example, the situation model (one of the three core aspects of SM) is



essentially based on a classical definition of states and events and possible relations between them. In Situation Management, reasoning about situations is supported using classical Artificial Intelligence techniques such as Event Calculus and Situation Calculus. These classical techniques reduce the semantic capabilities of Situation Awareness and limit their use in cognitive-based systems as we will show in the next subsection.

## **2.3 Classical Approaches in Artificial Intelligence for Situation Modeling**

Spatiotemporal reasoning has been addressed in AI through a variety of approaches. In the context of spatiotemporal situation modeling, the logical approach has gained an increasing attention and tried to offer different formalisms in order to model situations. In this subsection, we briefly present the most commonly used approaches such as Situation Calculus and Event Calculus.

### **2.3.1 Situation Calculus**

Situation Calculus (SC) is the name of a particular way of modeling the notion of change using First-Order Logic. It conceives the world as consisting of a sequence of situations, each of which is a snapshot of the state of the world. Situations are generated from previous situations by actions [McCarthy and Hayes, 1969]. A given relation or property that can change over time can be handled adding an extra situation argument in the corresponding predicate. If the position of an agent is represented by the predicate  $At(Agent, location)$  and a situation is represented by the constant  $S$ , then the location of the agent in the corresponding situation is denoted  $At(Agent, location, S)$ .

To represent how the world changes from one situation to the next one, Situation Calculus uses the function  $Result(action, situation)$  to denote the situation that results from performing an action in some initial situation. Situation Calculus is commonly

used to perform planning tasks. Given a set of possible actions, an inference engine can be used to find a sequence of actions that achieves a desired effect [Guesgen and Marsland, 2010].

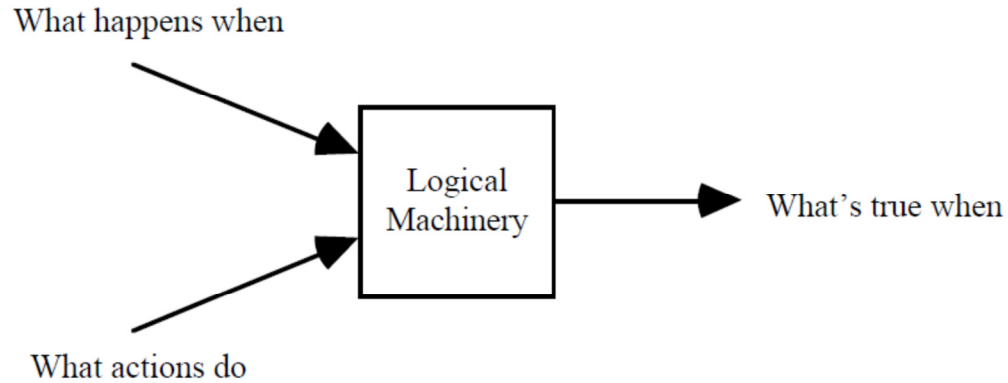
Although situation calculus has been used to represent situations and the change over time, it has several shortcomings. According to [Russell and Norvig, 1995] Situation Calculus defines situations as instantaneous points in time which are not very useful for describing changes that occur continuously over time. Moreover, Situation Calculus is more suitable when only one action occurs at a time. This approach does not offer an explicit time and space representation [Guesgen and Marsland, 2010]. To overcome these limitations, the Event Calculus has been proposed as an improved version of Situation Calculus.

### **2.3.2 Event Calculus**

The event calculus has been introduced by Kowalski and Sergot in 1986. They proposed a logic programming formalism to represent events and their effects and applied this formalism especially to database applications [Kowalski & Sergot, 1986]. A simplified version of Event Calculus was presented later by [Kowalski, 1992] which initiated the emergence of a number of Event Calculus dialects since then. Like Situation Calculus, Event Calculus is a formalism for reasoning about action and change. Actions in Event Calculus are simply called *events* and they are defined to reason about changes. In other words, the Event Calculus is a logical mechanism that infers what's true given “what happens when”, and “what actions do”. The “what happens” part is a narrative of events. The “what actions do” part describes the effect of actions [Shanahan, 1999]. Figure 2.7 describes the Event Calculus principle of operation.

Other versions of Event Calculus have been proposed such as the Basic Event Calculus (BEC) [Shanahan, 1997] and the Event Calculus (EC) proposed by [Miller

and Shanahan, 1999]. They have been applied to a variety of problems including natural language processing and vision.



*Figure 2.7: Event Calculus Principle of Operation [Shanahan, 1999]*

### 2.3.3 Synthesis

One of the major shortcomings of Situation Calculus is the frame problem [Russell and Norvig, 1995]. The research community attempted to propose several solutions. Some of these efforts led to the introduction of Event Calculus which was widely used to represent and to reason about situations of interest. In the spatiotemporal research community, researchers attempted to provide an implicit representation of situations of interest and defined some reasoning mechanisms using First Order Logic. Nevertheless, Event Calculus becomes ineffective when considering complex spatiotemporal phenomena occurring in the real world. Typical examples are dynamic situations which are not clearly represented using a combination of states and events [Haddad, 2009]. The majority of the proposed works seem to be outdated and not easily integrated in modern applications. Usually, authors tend to use simple assumptions to simplify logical rules and the proposed formalisms need to be used by well-trained users.

## 2.4 Situations of Interest Modeling Using Patterns

The notion of *pattern* is usually related to the Data Mining research domain. However, patterns have been used by a limited number of researchers to represent and reason about situations of interest. These works can be categorized in two different families: predicate-based approaches and query-based approaches.

### 2.4.1 Predicate-Based Approaches

In Section 2.3, we presented some foundations of Event and Situation Calculi. These approaches do not necessarily focus on the formal definition of spatiotemporal situation of interest. However, several works in the literature used predicates to build patterns representing situations of interest. For example, Holzmann [Holzmann, 2007] proposed a qualitative model applied to dynamic environments for recognition and detection of situations involving spatiotemporal objects defined in a specific environment and using different contexts. His model uses a combination of spatial and temporal relationships and a set of logical rules.

Figure 2.8 shows the main modules of Holzmann's framework. Spatiotemporal patterns are defined by a "human expert" as a set of rules. The "Rule Engine" gets the qualitative relationships from spatial sensors stored in the relationship repository and performs pattern matching operations according to logical rules stored in the rule base. The detected patterns are used to select actions that the system can carry out.

Gerhke [Gerhke et al, 2005] proposed an explicit representation of traffic scenes using spatiotemporal data (called the "background knowledge") and situation patterns. Situation patterns are defined by a combination of spatial and temporal predicates. An example of a dynamic situation describing two approaching vehicles with a risk of collision is depicted in Figure 2.10. The "Holds" operator specifies the validity of a predicate  $p$  during a time interval  $i$ .

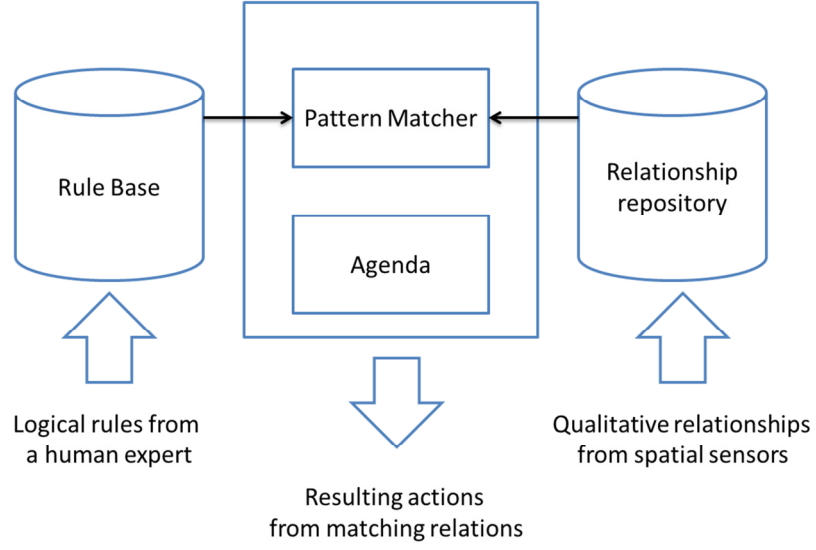


Figure 2.8: A framework for pattern detection as defined by [Holzmann, 2007]

$$\begin{aligned}
 & HOLDS(\text{CollisionCourse}(v_1, v_2), i) \Leftarrow \exists dist : HOLDS(\text{DistanceTrend}(v_1, v_2, \text{DecreasingDistance}), i) \wedge \\
 & HOLDS(\text{RelativeDirection}(v_1, v_2, \text{OppositeDirection}), i) \wedge HOLDS(\text{RelativeLane}(v_1, v_2, \text{SameLane}), i) \wedge \\
 & HOLDS(\text{TimeDistance}(v_1, v_2, dist), i) \wedge dist \leq \text{MediumDistance}
 \end{aligned}$$

Figure 2.9: An example of a spatiotemporal pattern [Gerhke et al, 2005]

Hadji's approach [Hadji et al, 2005] is different from formerly presented works. Hadji defines a spatiotemporal pattern in the form of a sequence of spatial predicates and temporal constraints. Moreover, the authors proposed spatiotemporal algorithms to evaluate and classify spatiotemporal queries in order to reduce the execution time in large-size databases. Lattner [Lattner et al, 2006] proposed a framework for the prediction of situations of interest and behaviors of mobile robots moving on a soccer field. The proposed system is based on a set of algorithms to detect patterns from a dynamic environment. Patterns are defined by a sequence of spatial or conceptual predicates linked by temporal relations. Figure 2.10 illustrates an example of a

spatiotemporal pattern defined by Lattner. The pattern "p77" describes a scene where a robot controls the ball first and where a second robot is free and located in front of the first robot. By combining these predicates with temporal relationships (in this example *younger*, *younger & contemporary*) it was possible to deduce using a knowledge base (called the set of "rules of association" by Lattner) that the first robot will pass the ball to the second robot with a probability of 63%.

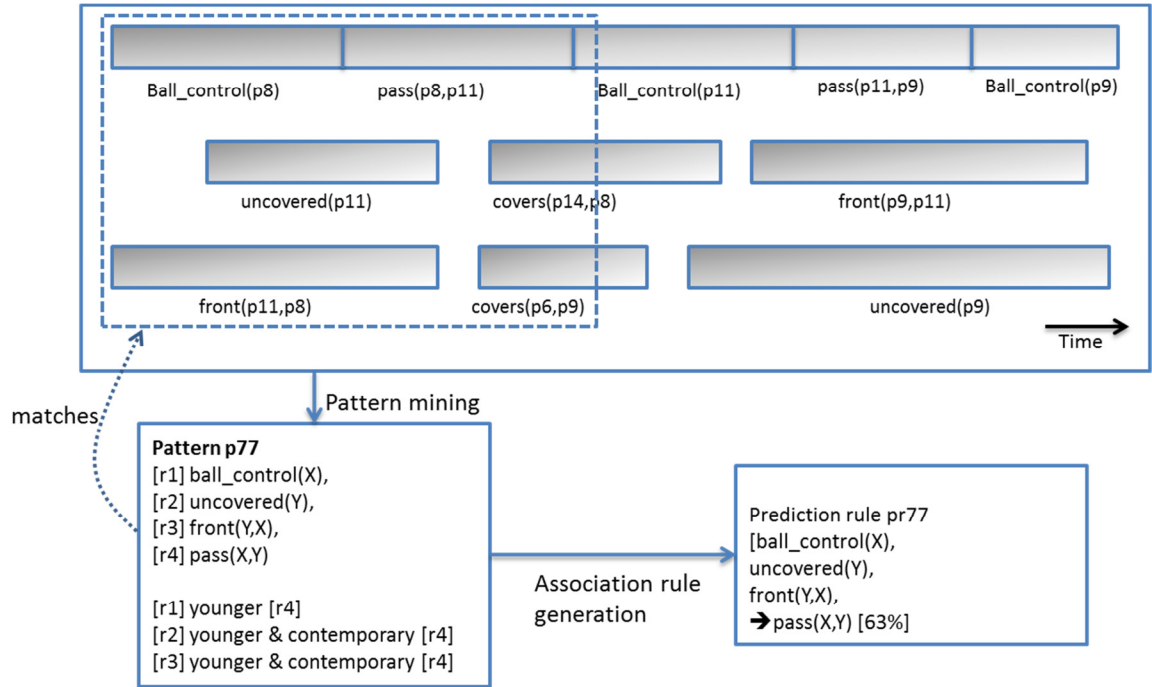


Figure 2.10: An example of a rule of association defined by [Lattner et al, 2006]

The authors also proposed a hierarchy of patterns (see Figure 2.11) and defined a set of specialization and generalization operations. The specialization of a pattern is possible either by adding a new predicate, by adding a new temporal relationship, or by specializing a predicate. The generalization of a pattern is carried out by eliminating a predicate, by removing a temporal relationship or finally by generalizing a predicate.

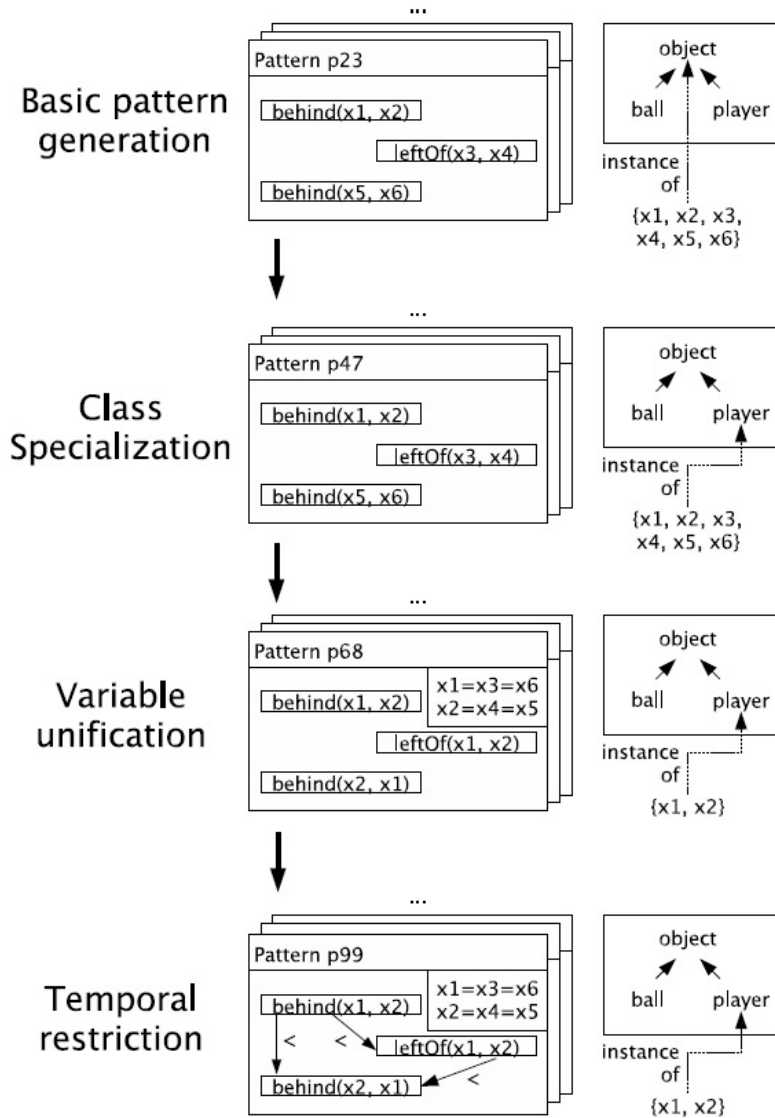


Figure 2.11: Different operations on pattern supported by [Lattner et al, 2006]

## 2.4.2 Query-Based Approaches

The concept of spatiotemporal patterns (STP) has been used to model situations of interest using extensions of query languages running on spatiotemporal databases. An in-depth review can be found in [Erwig, 2004] who categorized works on STPs in two areas. The first area aims at proposing a set of representation tools to query spatiotemporal patterns. Generally speaking, most of these works propose an enhanced version of existing query languages such as SQL. The second area aims at

using data mining and knowledge discovery techniques to find relevant patterns in large datasets. Most of the works in this category propose algorithms to find “group patterns” using statistical techniques such as [Sakr and Guting, 2014]. For example, [Bermingham and Lee, 2014] proposed a statistical approach to extract spatiotemporal meta-data by combining a set of photos collected from a social network (i.e. Flickr). In this subsection, we focus on the first category since we are interested in spatiotemporal situations representation and reasoning.

Existing query-based approaches tend to offer extensions of query languages in order to find changes in objects in spatial or spatiotemporal datasets and relationships between objects. This was applied to different areas such as meteorology, medicine and geophysics. Change phenomena in human-related activities were also considered such as movement of terrorists and criminals, traffic management and military operations [Erwig, 2004]. In order to specify spatiotemporal patterns, the extended query languages used temporal and spatial formalisms such as temporal logic and topological relations. However, these works were limited in terms of expressiveness and they offered languages too complex to be used by users who are not familiar with such query languages [Erwig, 2004]. Erwig proposed to overcome such a limitation by the introduction of spatiotemporal predicates which “*describe precisely the ‘developments of objects’ and their relationships in a simple way*”. According to Erwig, a predicate  $P$  is a function that maps a pair of spatiotemporal objects to a Boolean value:  $P: a \times b \rightarrow bool$  for  $a, b \in \{Point, Region\}$ . A query that finds the pattern “ships leaving the oil spill” can be formulated as follows:

```
SELECT sname
FROM Ships, Pollutions
WHERE pname='Spill' AND Pos Leaves Reg
```

Where *Leaves* is a predicate that describes a “leaving development” of a specific object and can be defined as:  $Leaves := Inside \text{ meet } Disjoint$ . Notice that *Leaves* is a



predicate that takes the value true for two objects  $O_1$  and  $O_2$  if for some time,  $O_1$  was inside  $O_2$ , then  $O_1$  touched  $O_2$ 's border and finally  $O_1$  disjointed from  $O_2$ .

Another way to process spatiotemporal patterns using query languages was proposed by [Sakr and Guting, 2010]. The authors used a set of general and powerful classes of predicates called “lifted predicates” to enhance the pattern query language to support moving objects. Moving objects are defined using the *moving type constructor* which defines the moving counterpart of any static component such as point, region or line. Lifted predicates are a time-dependent version of static predicates. Instead of returning a Boolean value like standard predicates do, lifted predicates return a *moving(bool)*. For example, the static predicate *Quebec inside Canada* returns a Boolean type and the lifted predicate *Bus\_803 inside Quebec* returns a *moving(bool)* type. The query language uses Allen’s operators to specify relationships between time intervals. Hence, the spatiotemporal predicates proposed by the authors are defined as a set of time-dependent predicates that are fulfilled in a certain temporal sequencing order. The situation “the snow storms that could increase their area over  $\frac{1}{4}$  square km during the first traversed 5 km” can be represented by the following pattern:

```
SELECT *
FROM snowstorms
WHERE pattern (
[distancetraversed(rough_center(storm) <= 5000.0 as pred1, area(storm) > 25000.0 as
pred2], [stconstraint (“pred1”, “pred2”, meanwhile)])
```

Where *distancetraversed* and *rough\_center* are lifted predicates representing respectively “the distance that the moving point traversed since the start of its definition time” and “the aggregation of the moving region into a moving point that represents its center of gravity”.

To optimize the computation time of their query language, [Sakr and Guting, 2010] proposed to integrate their spatiotemporal pattern queries in a query optimizer. The proposed solution has been developed and made publicly available as a SECONDO plugin [Secondo web site, 2013] and an extended version of their work was proposed in [Sakr and Guting, 2014] to support “*group patterns*”.

Another work addressed the issue of spatiotemporal representation using query languages. With the recent advances in ubiquitous computing technologies, Vieira et al [2010] investigated human motions generated by Call Detail Records (CDR) of cell phone networks. Typically, any phone call received by a human is logged in the form of a CDR which includes information about originating and destination phone numbers, time and date when the call started and which towers were used to make this call. Such a variety of information is inherently spatiotemporal and may be of interest for many studies such as human’s mobility behaviors, cellular network performance and so on. The CDR records are stored in very large relational databases. The authors noticed that searching for STPs in existing commercial databases is computationally expensive and requires a large number of queries. They proposed a Spatiotemporal Pattern System (STPS) to query patterns in CDR databases. The STPS is designed to “*express mobility pattern queries using a regular expression like language*”. It uses spatial and temporal predicates in the pattern definition. Spatial predicates are used to locate Base Transceiver Stations (BTS) and their covering areas and to establish topological relationships between them. Three forms of temporal predicates are defined in the STPS. 1) Time interval like for example “between  $t_1$  and  $t_2$ ” where  $t_1 \leq t_2$ . 2) Time snapshot like for example “at 5:00 PM”. 3) Relative temporal relations like for example “1 hour after user left his home”.

The expression: “*find all mobile users that on Saturdays first start in an arbitrary area different from Neighborhood-2 in the morning, then immediately go to Airport, then pass along the Stadium-1 between 6pm and 8pm, then go in the Neighborhood-1*”

*neighborhood between 8pm and 10pm, and finally return to their first area*”, can be represented by the following query:

$$Q := (\langle @x, t_{from} = 8 \text{ am} : t_{t0} = 3 \text{ pm} \rangle. \langle \text{Airport}, t_r = 1 \text{ min} \rangle. \langle \text{Stadium} - 1, t_{from} = 6 \text{ pm} : t_{t0} = 8 \text{ pm} \rangle. \langle \text{Neighborhood} - 1, t_{from} = 8 \text{ pm} : t_{t0} = 10 \text{ pm} \rangle. \langle @x \rangle, C = \{ @x! = \text{Neighborhood} - 2, \forall t_i, t_j \in S, \text{Date}(t_i) \wedge \text{Day}(t_i) = \text{"Saturday"} \})$$

Notice that formulating such a query needs expertise in query languages that the average user does not possess.

### 2.4.3 Synthesis

Spatiotemporal situations of interest have been represented by a limited number of pattern-based approaches in the literature. Instead of dealing with pattern representation, predicate-based approaches rather addressed pattern matching issues by proposing different algorithms for this purpose. Patterns are usually built using temporal relations between predicates. They are stored in pattern bases and pattern matching algorithms are used to detect pattern instances from incoming data. Although reasoning capabilities are enabled using such approaches, they remain limited since the predicate language makes it difficult to represent dynamic and complex spatiotemporal situations. The integration of such works in large-scale systems seems to be quite challenging and weakly addressed by these works.

Query-based approaches proposed several extensions to the SQL language to query patterns from spatiotemporal databases. Proposed query languages are usually associated with specific platforms because they require some specific implementations to optimize the search time (see the SECONDO platform for example). In the context of Big Data and the new generation of large scale monitoring systems, this becomes a real challenge. Furthermore, using SQL languages and their special extensions requires well-trained users and reduces the

semantic expressiveness capabilities of these patterns. Finally, Query-based approaches consider several aspects of situations of interest such as moving objects and temporal relations. However, dynamic situations of interest may involve other components such as events and states which are not clearly represented by these languages. In the next subsection, we explore another alternative to overcome some limitations related to query-based approaches which is Complex Event Processing.

## 2.5 Complex Event Processing

The limitations of spatiotemporal query approaches and the increasing number of distributed applications which need to process data generated by any kind of sensors, led to the emergence of Complex Event Processing as a novel approach to manage event streams [Cugola and Margara, 2012]. CEP systems consider the stream of data as notifications of events happening in the external world. The main objective of a CEP system is to detect the occurrence of complex events also called *patterns*. As a consequence of this detection, responding actions can be taken [Anicic et al, 2010]. Generally, events are instantaneous and happen at one specific point of time. Complex events or patterns are the result of correlations between sets of events using temporal and/or causal relations as well as aggregations. Originally, CEP has been used in the financial industry to predict phenomena such as market development and exchange rate trends. The notion of event has been defined in its different aspects (general, semantic, etc.) by the CEP research community. A general definition of an event is given by [Luckham, 2002] as “something that happens”. [Etzion and Zolotorvesky, 2010] proposed a general representation of the concept of event depicted in Figure 2.12. Events which have the same meaning and which are defined with the same set of event attributes can be considered as instances of a given *event type*. Event attributes can be divided into three main sections: Header, Payload and Event Relations. The *Header* contains the meta-information about an event. The *Payload* contains specific information about the event occurrence itself. The *Event Relations*, which is optional, allows for establishing relationships between several events.

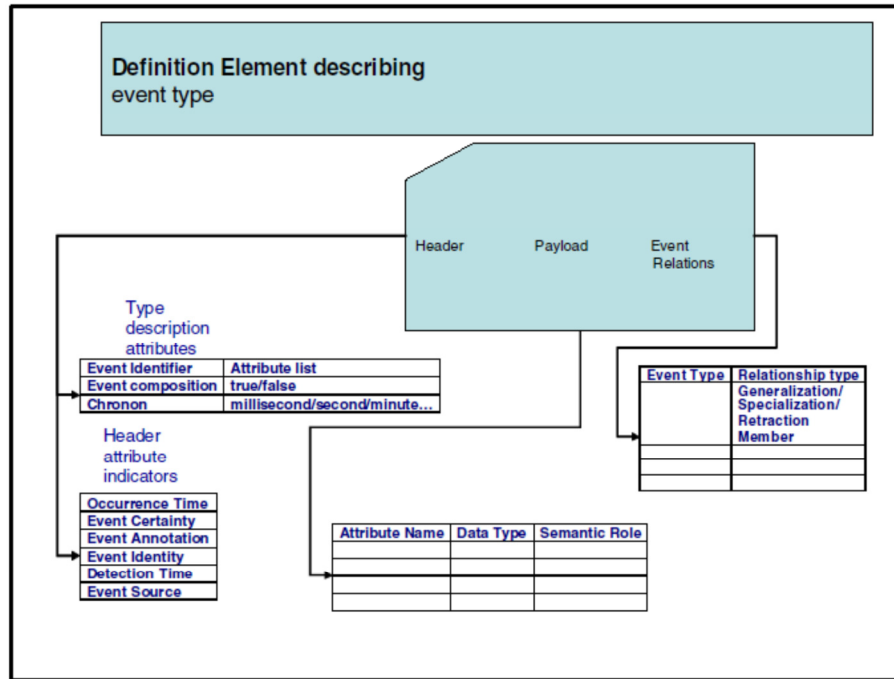


Figure 2.12: A general event definition according to [Etzion and Zolotorvesky, 2010]

Patterns are generated by the result of event correlation and can be expressed using SQL-like languages. For example, the Event Pattern Language (EPL) [Esper, 2015] is a pattern language proposed by the ESPER CEP Framework. A pattern describing a situation of excessive energy usage in a floor using power meter inputs can be expressed using EPL statements as described in Figure 2.13.

```

INSERT INTO ExcessiveEnergyUsageByFloor
SELECT a.floor as floor
FROM PATTERN [(a=FloorEmptySensor -> every
               b=DeviceEnergyUsageSensor(a.floor=b.floor))]
.WIN:TIME(30 min)
GROUP BY a.floor
HAVING SUM(b.usage) > GetAcceptableThreshold(a.floor)

```

Figure 2.13: A pattern expressing an excessive energy usage situation [Hasan et al, 2012]

Several CEP frameworks have been proposed by the Computer Science community. They allow for event representation and management and they provide pattern detection capabilities. A detailed review on these frameworks can be found in [Cugola and Margara, 2012]. A typical architecture of a CEP framework is depicted in Figure 2.14 where events are generated by *Event Producers* (e.g. sensors, terminal units or social network feeds) and managed through the *Event Channel*. The *Event Processing Engine* uses the *Event Condition Action* (ECA) principle to trigger the (Event) *Pattern Matching Engine*, to verify patterns' conditions (Condition) and to direct detected patterns through the *Event Channel* to *Event Consumers* (Actions). *Event Development Tools* and *Event Management Tools* are also available with CEP frameworks.

One of the key features of CEP is the support of contextual information. *Contexts* are used to change the way an event (or a pattern) is processed [Etzion and Zolotorvesky, 2010]. Contexts are used in CEP to partition a stream of events using different context types such as time and space. Contextual information can be either implicit or explicit. When they are explicit, contexts are clearly expressed in event pattern languages and they are used by the Event Processing Engine to evaluate patterns' conditions.

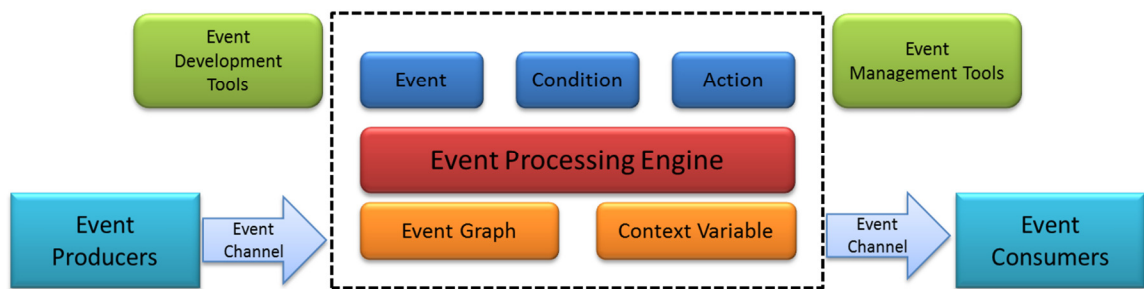


Figure 2.14: A typical CEP architecture [Helmer et al, 2011]

The following EPL statement (Figure 2.15) illustrates how contexts are defined in the ESPER Event Pattern Language [Esper, 2015]. The statement refers to the context

“segmented by customer” created to detect a money withdrawal which is bigger than a given amount, followed by a second withdrawal that occurred within a time interval of minutes of the first withdrawal. Both events are related to the same customer. The customer plays the role of the contextual information in this example.

```
context SegmentedByCustomer
select * from pattern [ every a=BankTxn(amount > x) -> b=BankTxn(amount
> x) where timer:within(y minutes)]
```

*Figure 2.15: An EPL statement example*

### **2.5.1 Synthesis**

Complex Event Processing is an emerging research area for event-based pattern representation, detection and management. Existing CEP approaches deal mainly with the syntactical processing of raw data, constructive event database views and stream management [Teymourian and Paschke, 2010]. Patterns are expressed using a correlation of events and can be detected by taking into account contextual information. The CEP approach in the context of situation management is more efficient than classical relational database management systems (RDMS) since CEP allows for detecting patterns on the fly whereas patterns are retrieved from databases offline, which may make the pattern detection in RDMS both resource and time consuming. CEP can be considered as a natural evolution to overcome RDMS limitations in terms of performance and expressiveness. CEP systems are part of Event Driven Systems where events and contextual information play a key role in the pattern definition and detection. However, most of CEP systems use a SQL-like language for pattern representation which limits the expressiveness of situations of interest and does not facilitate their integration with agent-based software [Teymourian and Paschke, 2010]. The ETALIS project [Anicic et al, 2010] is one of the rare CEP frameworks using a non SQL-like language for pattern representation. ETALIS uses a declarative logic language for pattern representation and reasoning

and proposes a set of operators (mostly based on Allen’s interval logic) to build “complex event descriptions”. It implements the ETALIS Language for Events (ELE) and EP-SPARQL for event pattern representation. ETALIS uses a rule-based language with a clear syntax and declarative formal semantics. The logic-based approach used in ETALIS is expressive enough and allows for using contextual information which can be expressed as PROLOG rules and facts. However, the ELE language and EP-SPARQL are limited to few temporal operators (such as sequence [Zhou et al, 2012]) and they lack other semantic operators (such as spatial operators). Furthermore, the notion of pattern is still limited to events. The tool does not allow for expressing dynamic and complex situations of interest which may involve, besides events, other components such as states and spatial objects, to name a few.

## 2.6 Semantic-Based Approaches for Situation Modeling

Researchers still do not agree on a unique semantic representation of a situation of interest in a dynamic environment. Several related aspects are discussed such as change representation, the ontological aspect of various components involved in a situation of interest, possible relationships between these components and mechanisms to reason about situations of interest. To address these aspects, a particular research group defined situations of interest as spatiotemporal phenomena. For example, [Galton, 2004] discussed the incorporation of time in GIS to represent spatiotemporal phenomena. He started from a well-known approach in this domain which relies on *time-indexed snapshots*. Using a set of snapshots related by temporal relations, one can represent an animated sequence of some features of interest. [Galton, 2004] discussed some limitations of such an approach since, in real life, spatiotemporal information does not come bundled in complete snapshots. Indeed, spatiotemporal phenomena can be related to a set of observations of different events occurring at different points of time; and about spatial objects situated in different locations. The author defined a change as a relationship between a set of events  $e \in E$  and spatial objects  $c \in C$ . He proposed to represent the effect of an event as follows:



$$effect(e) = \{\langle c_1, p_1, p'_1 \rangle, \langle c_2, p_2, p'_2 \rangle, \langle c_3, p_3, p'_3 \rangle, \dots, \langle c_n, p_n, p'_n \rangle\}$$

Where  $c_1, c_2, \dots, c_n$  are the “participants” (objects) in  $e$  and the effect of  $e$  on  $c_i$  is to change its position from  $p_i$  to  $p'_i$ . Further details on the event definition are proposed by the author who also discussed some challenges related to the representation of complex phenomena (called *multi-aspect phenomena*) such as storms, floods, a protest march, etc.

In other works, the concept of situation of interest is represented through the combination of events and spatial objects. [Cole and Hornsby, 2005] presented an approach to model events and sequences of events in a dynamic geospatial domain. Events are referred to as *occurents* and they are modeled using an UML diagram. A sequence of events is defined to specify movements which capture the dynamic experience of an entity as it travels through space. The authors used temporal logic to manipulate temporal relations between events and implicitly represent situations of interest. A sequence of events can be defined by a set of events ordered by their occurrence times. This approach limits the definition of dynamic situations of interest to events only, whereas other objects such as spatial objects, states and processes should be considered to represent complex spatiotemporal phenomena.

Other definitions of spatiotemporal situations are offered by [Tan et al, 2009] , [Lin et al, 2009] and [Aigong, 2009] who proposed a set of 8 situations types and [Devaraju and Kauppinen, 2012] who proposed a semantic definition of events and a rule engine to reason about them.

These works partially address the notion of situation of interest via the definition of events, states or processes. However, they do not consider possible relationships between these different entities to represent complex and dynamic situations. Finally, these works do not seem to be easy to integrate in software agent systems.

To overcome the above limitations, Haddad and Moulin proposed in [Haddad and Moulin, 2010] a novel definition of a spatiotemporal situation taking advantage of Desclés's work. According to this French linguist [Desclés, 2005], a spatiotemporal situation can be either static (state) or dynamic (event or process). A dynamic situation allows for the explicit representation of change using temporal relations. Haddad and Moulin used the Conceptual Graph formalism [Sowa, 1984] to represent different types of situations. Using Conceptual Graphs (CG) makes their formalism closer to natural language and ready to be used by software agents. The various concepts introduced in their model are based on the ontological work introduced by [Grenon and Smith, 2004] where entities populating the world are either called *endurant* (static) or *occurrent* (dynamic). Haddad and Moulin proposed the conceptual model illustrated in Figure 2.16.

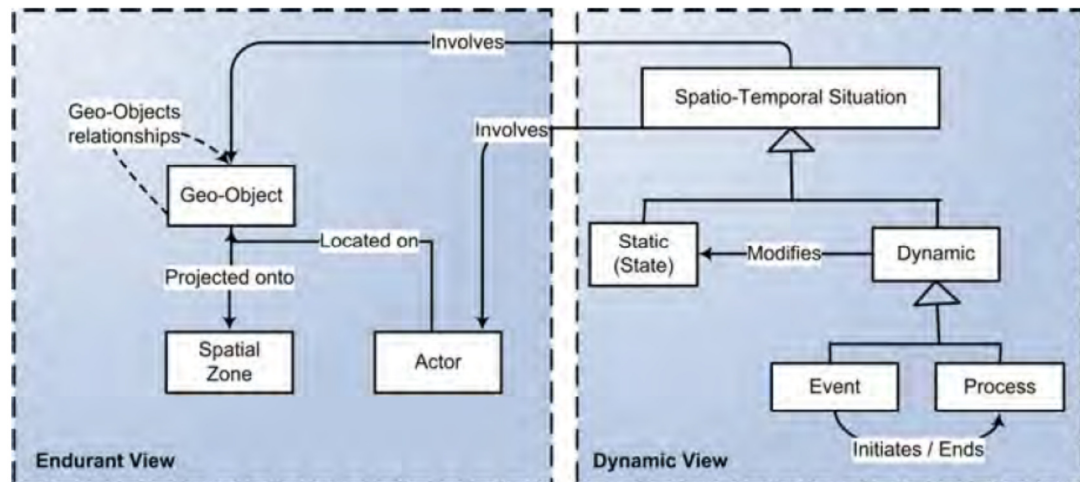


Figure 2.16: A spatiotemporal model of situations [Haddad and Moulin, 2010]

The endurant view defines static entities called *geo-objects* such as (buildings, trees, mountains, etc.) and *actors* (moving objects such as people, cars, etc...). Given the spatial position of geo-objects, their relations can be described using qualitative spatial relations such as topological relations, proximity relations and superposition relations. The dynamic view defines spatiotemporal situations such as states, events and processes inspired by Desclés' definition. A *state* is a finite configuration of

some aspect of the world in a limited region of space that remains stable for a certain period of time.

Figure 2.17 illustrates a CG example of a state describing that Dany was sick during a time interval and that Dany was in two different locations (Paris and Quebec). An event as defined by the authors expresses a temporal occurrence that appears in a “static background”. An event may or may not change an aspect of the world from one state to another. This change can be represented using temporal relations as illustrated by Figure 2.18 where *Before Situation* and *After Situation* are temporal relations used to relate the states describing the current spatial location of the Actor “Hedi”. The event *Spatial\_Zone\_Entry\_Event* describes the entrance of the actor in a specific spatial zone.

Attribution_State: st1 BT: September 23 2004; ET: January 20 2005; TS: Date
[PERSON: Dany]->(ATT)->[Sick]
{<23-09-2004, 11-12-2004, Québec>, <12-12-2004, 20-01-2005, Paris>}

Figure 2.17: An example of a state [Haddad and Moulin, 2010]

Finally, a process expresses a change initiated by an event that marks the beginning of the process, and may have an end-event and a resulting state. An example of a process is illustrated in Figure 2.19 where the temporal relation *During Situation* is used to link a process to the current state of the world which holds during the process.

The authors used the proposed spatiotemporal situation formalism to support “What-if” reasoning and its particular application to the planning of course of actions (COAs). They implemented several scenarios from the aerial search and rescue domain (SAR) on a multi-agent geo-simulation platform called MAGS-COA [Haddad, 2009].

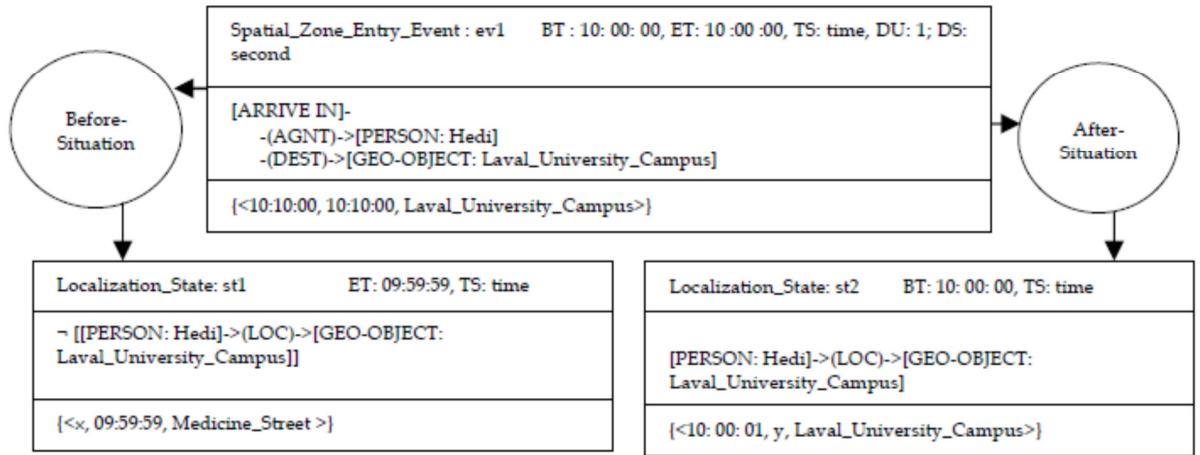


Figure 2.18: An example of an event [Haddad and Moulin, 2010]

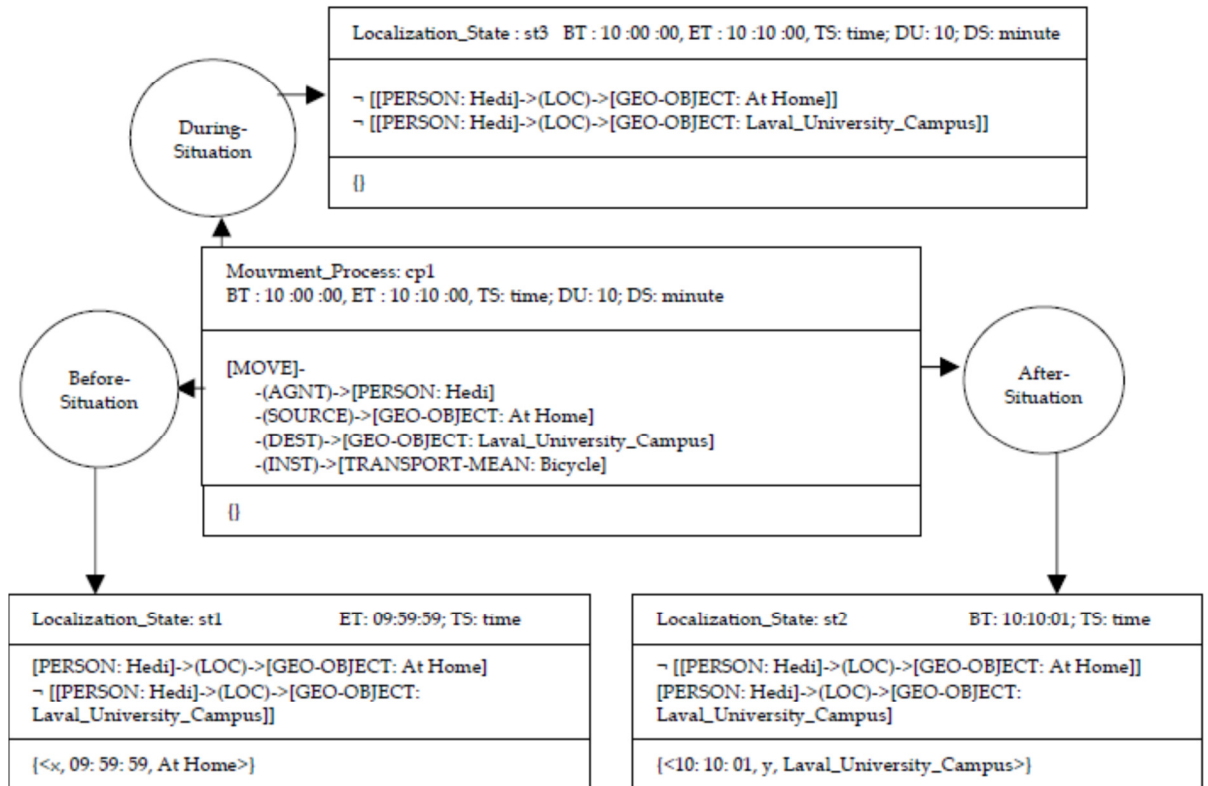


Figure 2.19: An example of a process [Haddad and Moulin, 2010]

### 2.6.1 Synthesis

The approach proposed by Haddad and Moulin seems to be the most advanced one to represent situations of interest. While other works limit the definition of phenomena to events only, Haddad and Moulin's approach is very promising thanks to the strength of the representation formalism used to represent dynamic situations (i.e. Conceptual Graphs and Desclés' definitions). However, the authors did not provide a formal definition of situation of interest in their approach. Unlike Complex Event Processing, the framework proposed by Haddad and Moulin does not support contextual information and does not address the issue of detecting situation instances from a stream of data. This does not make this approach directly suitable for Big Data and Pervasive Computing applications discussed in Section 2.1.

## 2.7 Qualitative Spatiotemporal representation and reasoning

Temporal and spatial relations are widely used in most of the works which deal with different components of spatiotemporal situations of interest. They take advantage of recent advances in the spatiotemporal research domain to provide a set of tools and formalisms that help representing and reasoning about space and time in a qualitative manner. Actually, qualitative modeling is one of the key elements of knowledge representation. Forbus [Forbus, 2007] defines qualitative modeling as follows: *“qualitative modeling concerns representation and reasoning about continuous aspects of entities and systems in a symbolic, human-like manner”*. In other words, qualitative modeling makes a bridge between quantitative information (raw data) and the way a user considers this data in his mental model. People do not need to know about complex equations and mathematical theory to achieve everyday tasks by handling the common sense world of quantities, motion, time and space. A software agent may need to make decisions in a similar way as humans. Therefore, agents have to apply qualitative modeling and reasoning as abstraction means and to transform quantitative information into entities and symbols. In this subsection, we briefly

introduce some interesting aspects of qualitative spatial and temporal modeling which are major research subjects in Qualitative Modeling.

### **2.7.1 Qualitative Spatial Representation and Reasoning**

Spatial knowledge can be represented using either a quantitative approach or a qualitative approach [Renz, 2002]. A quantitative spatial representation deals with numerical (classical) spatial information. Software agents which share a global coordinate system need to consider their local coordinate systems to reason about spatial objects. The exact position and properties of a spatial object must be known, which makes reasoning about inexact information difficult [Renz, 2002]. A qualitative approach allows for representing and reasoning about spatial objects without using exact numerical values. It rather uses a human-like vocabulary such as “*far*”, “*close*”, “*inside*” as possible spatial relations between spatial objects. This approach led to the definition of the Qualitative Spatial Reasoning (QSR). QSR is a subfield of knowledge representation and symbolic reasoning which deals with qualitative knowledge representation of the spatial domain; and reasoning using finite qualitative relations [Wolter and Wallgrun, 2013]. The term “qualitative” is used because the aim of QSR is to model human common-sense understanding of space. QSR has been mainly used in GIS-based applications. It has been also applied to several application domains such as Sensor Networks, Sensor Web, Path Planning and Robot Navigation, to name a few. According to [Cohn and Renz, 2007] most of QSR approaches deal with two aspects of space: spatial knowledge representation and qualitative spatial reasoning. In such applications, users need to query spatial data in a qualitative manner and to have an abstract view of numerical (quantitative) data to reason about spatiotemporal situations. In the following paragraphs, we review different spatial representation approaches and their related reasoning mechanisms.

In a spatial knowledge representation, there are different ways to represent two spatial entities and to specify spatial relations between these entities using topological, distance or other relation types. In [Cohn and Renz, 2007] the authors proposed an

overview of different approaches related to qualitative spatial representation which is a major topic in spatial representation. It deals with the study of different spatial relations types that can be defined to link spatial entities. Usually, spatial relations are binary relations and they are applied to a given spatial domain. Hence, a spatial relation can take the form  $R=\{(a,b) \mid a,b \in D\}$  where  $D$  is the infinite spatial domain. Using algebraic operators such as union and intersection allows for the definition of an algebra of spatial relations between spatial entities. Different works on spatial relations tried to restrict relations to one particular aspect of space such as topology, shape and orientation. This resulted in the definition of four types of spatial relations as described in Figure 2.20: orientation (e.g. front of, right of), direction (e.g. right direction, left direction), distance (e.g. near, far), and topology (e.g. disjoint, equal, overlaps) [Holzmann and Ferscha, 2010].

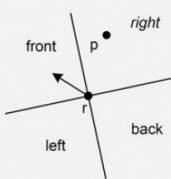
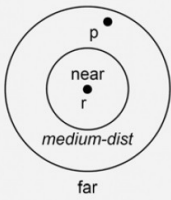
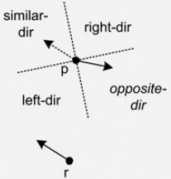
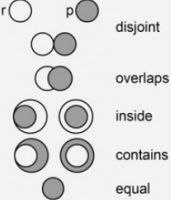
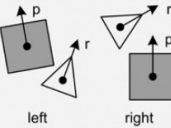
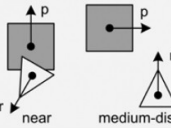
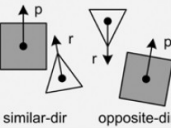
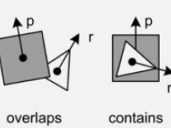
	Orientation	Distance	Direction	Topology
Definition				
Examples				

Figure 2.20: An example of spatial relations types [Holzmann and Ferscha, 2010]

*Direction relations* describe the direction of one object relatively to another one using three basic concepts: the primary object, the reference object and the frame of reference. Therefore, using a binary relation is not sufficient to represent direction/orientation relations. The STAR algebra is one of the approaches proposed to define direction relations [Renz and Mitra, 2004].

*Distance relations* describe the proximity of objects. The notion of distance is one of the most fundamental primitives used in computational and common sense reasoning [Bera and Claramunt, 2003]. A distance relation can be absolute or relative. Absolute distance can be obtained by dividing a real line between two spatial objects into different sectors such as “close”, “very close” and so on. *Relative distance* can be obtained by comparing the distance to a relative spatial reference which may be a spatial entity for example. This gives relations such as “closer than”, “farther than” or “equidistant” [Renz, 2002]. Usually, a distance relation is metric-based; but it can also be defined using absolute and relative measurements. Recent works tend to consider other aspects of proximity in the definition of distance relations. These aspects are related to contextual information. For example, the distance between Quebec City and Montreal (which is about 250 Km) can be qualified as *far* according to a pedestrian and *close* according to a car driver. Hence, the transportation mean is a contextual information which can be combined with metric information to qualify the distance relation. Other researchers noticed that spatial proximity relations are vague. They used Fuzzy Logic techniques to better manage the vagueness of relations and to provide a smooth mapping from metric data to natural language quantifiers [Guesgen and Albrecht, 2000], [Brennan and Martin, 2006] and [Yao and Thill, 2007]. Using Fuzzy logic, it is also possible to use *if-then* rules to reason about spatial properties and to derive new distance relations. Another way to define distance relations without using distance metrics is the work of [Bera and Claramunt, 2003] who proposed a topology-based measure of proximity.

The topological distinction between spatial entities is inherently qualitative and makes topological relations a widely used concept in human spatial reasoning according to [Renz, 2002]. It also makes the qualitative approach more popular than the quantitative approach in dealing with this type of relations. Examples of topological relations are “A is inside B”, “A overlaps B” and “A touches B” which are invariant regarding to any transformation occurring in the surrounding space [Renz, 2002]. One of the well-known approaches in topological relations in the Region Connection Calculus (RCC) which is a fully axiomatized first-order logic for



topological relations representation and reasoning [Cohn et al, 1997]. RCC is based on a single primitive relation between spatial regions which is the “connected” relation  $C$ . The topological interpretation of the relation  $C(a, b)$ , where  $a$  and  $b$  are spatial regions, is that  $a$  and  $b$  are connected if and only if their topological closures share a common point [Renz, 2002]. Other relations can be derived using the definition of  $C(a, b)$ . RCC-8 is a particular implementation of RCC which defined 8 primitive relations depicted in Figure 2.21.  $DC(x, y)$  is the Disconnected relation,  $EC(x, y)$  is the Externally Connected relation,  $TPP(x, y)$  is the Tangential Proper Part relation,  $PO(x, y)$  is the Partially Overlaps relation,  $EQ(x, y)$  is the Equals relation,  $NTPP(x, y)$  is the Non-Tangential Proper Part relation.  $TPP^{-1}(x, y)$  and  $NTPP^{-1}(x, y)$  are respectively the converse of the non-symmetrical relations  $TPP(x, y)$  and  $NTPP(x, y)$ . RCC-8 is one of the most popular approaches for topological relations in the literature.

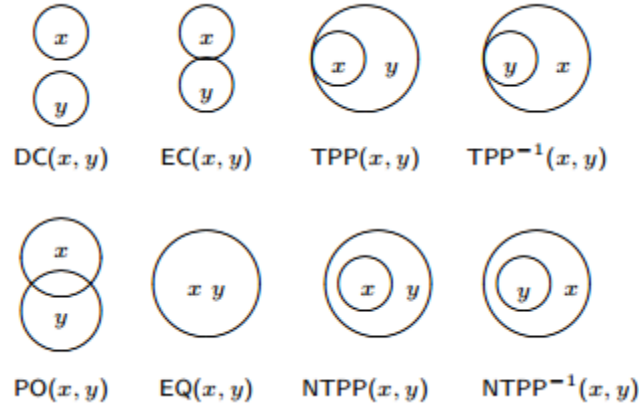


Figure 2.21: RCC-8 topological relations from [Renz, 2002]

## 2.7.2 Qualitative Temporal Representation and Reasoning

The notion of time can vary from one domain to another. In Philosophy and in Natural Language Processing, time can be considered as absolute or relative. In signal processing and mathematics, time can be discrete or continuous. In Event Processing Systems time can be defined as detection time and occurrence time. Detection time is

the time when an event is recorded by a system (as for example a database management system). Occurrence time is the time when an event occurred in the real world. In temporal databases, transaction time and valid time are defined. Transaction time is the time where information is recorded whereas valid time is the time when a phenomenon occurred [Haddad, 2009]. Other research communities addressed the notion of time from different perspectives. In Artificial Intelligence, [Chittaro and Montanari, 2000] divided temporal reasoning in two main subfields: reasoning about actions and change and reasoning about temporal constraints.

*Reasoning about actions and change* focuses on the evolution of the world as the result of the occurrence of actions and events. Most of reasoning approaches are interested in the future prediction or the past interpretation of phenomena pursuant to the occurrence of events. This kind of reasoning is also called *temporal projection* and can be divided into two categories: forward projection and backward projection. Forward projection tries to deal with the following proposition: “if  $X$  is true at time  $t_1$ , then  $Y$  is true at time  $t_2$ , where  $t_1 \leq t_2$ ”. Backward projection tries to find an explanation about what happened in the past (post-diction) and deals with the following proposition: “if  $X$  is true at time  $t_2$ , then  $Y$  is true at time  $t_1$ , where  $t_1 \leq t_2$ ”. The Situation Calculus and the Event Calculus are among the well-known formalisms developed to deal with reasoning about actions and change. In these approaches possible relations between temporal entities such as points and intervals are implemented using the temporal logic proposed by [Allen, 1983] who proposed the well-known formalism for temporal representation and reasoning about time intervals. He defined a set of 6 temporal relations that are *before*, *meets*, *overlaps*, *starts*, *during*, *finishes*, and their inverses and the basic relation equal which gives thirteen relations in total (See Figure 2.22).

Other time interval or point formalisms have been proposed in the literature such as Vilain’s and Kautz point algebra and Van Beek’s Continuous end point Algebra. A detailed review of these works can be found in [Chittaro and Montanari, 2000].

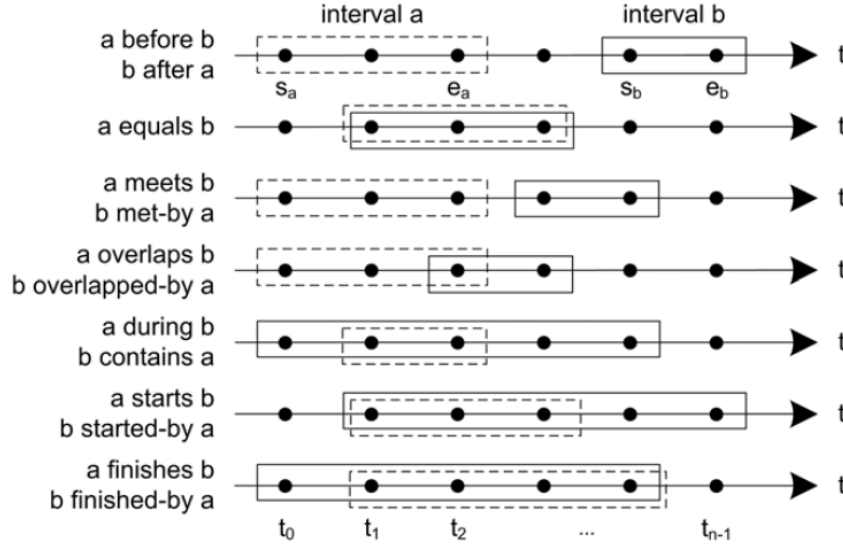


Figure 2.22: Qualitative temporal intervals relations [Allen, 1983]

### 2.7.3 Synthesis

Spatiotemporal data plays a key role in spatiotemporal situation modeling. It helps users to better understand the temporal and spatial aspects of situations and to associate them with the surrounding environment. Our short survey on spatiotemporal analysis shows recent advances in this research area to offer a large variety of tools and approaches for qualitative representation and reasoning. In the spatial domain, RCC-8 is one of the most known and popular approaches for topological relations whereas Fuzzy Logic has been widely used for proximity relations. In the temporal domain, Allen's temporal logic is the approach mostly used in the literature.

## 2.8 Discussion

We presented in this chapter a review of the main approaches, models and concepts that have been proposed to model spatiotemporal situations. We discussed the relevance of these approaches in the context of our thesis. Here we present a synthesis of these approaches in Table 2-1 where we outline the main features supported by each approach and its shortcomings. In the following sub-sections, we review these

works from three different viewpoints. First, we discuss the way these works define spatiotemporal situations of interest. Then, we review how situations of interest are represented by these approaches and which reasoning capabilities have been proposed, if any. Finally, we address the issue of the integration of these formalisms in real applications and how situations of interest are managed and detected when they occur in the real world. It is worth mentioning that the above approaches use temporal and spatial relations in their situation modeling formalisms. They leverage the recent advances in Qualitative Spatiotemporal Reasoning techniques that we briefly presented in Section 2.7.

### **2.8.1 Definition of Situation of Interest**

The definition of situation of interest varies from one research area to another. In Artificial Intelligence, situations are represented and manipulated using Situation Calculus and Event Calculus. In predicate-based approaches, patterns are used to represent and to reason about situations as proposed by [Gerhke et al, 2005] or [Lattner et al, 2006]. These patterns are expressed using events related by temporal and spatial relations. Patterns are also used by query-based approaches to query situations of interest from spatiotemporal databases.

A similar approach was proposed by the Complex Event Processing research community where event-based patterns are used to represent situations. Most of these works consider events in the definition of situations of interest. [Haddad and Moulin, 2010] proposed one of the approaches where situations are defined considering other spatiotemporal phenomena such as states and processes. The authors proposed an explicit definition of change by using temporal relations between states and events to represent dynamic phenomena and used an extension of Conceptual Graphs [Sowa, 1984] to represent these situations in a qualitative manner. However, their formalism does not provide a clear definition of patterns since they only use situations to represent a state, an event or a process and are not able to represent relationships between them.

<b>Approaches</b>	<b>Features</b>	<b>Shortcomings</b>
Classical approaches (Artificial Intelligence)	<ul style="list-style-type: none"> <li>-Event based</li> <li>-Temporal relations support</li> <li>-Most used approach in spatiotemporal analysis</li> </ul>	<ul style="list-style-type: none"> <li>-Frame problem</li> <li>-User needs familiarity with FOL</li> <li>-Limited expressiveness</li> <li>-Do not allow representation of dynamic situations</li> </ul>
Semantic based approaches	<ul style="list-style-type: none"> <li>-In-depth semantic definition of spatiotemporal phenomena</li> <li>-Qualitative representation, close to Natural Language</li> <li>-Explicit representation of change</li> </ul>	<ul style="list-style-type: none"> <li>-Implicit representation of situation of interest</li> <li>-No integration with real complex systems</li> <li>-Need complex algorithms to detect situations</li> <li>-Contextual information is not supported</li> </ul>
Situation Awareness/Management	<ul style="list-style-type: none"> <li>-Multi-layers architectures to manage the flow of data from sensors.</li> <li>-Scalability</li> <li>-Heterogeneity</li> </ul>	<ul style="list-style-type: none"> <li>-No explicit representation of situations</li> <li>-FOL based</li> <li>-Contextual information is limited to spatial information</li> </ul>
Predicates and query-based approaches	<ul style="list-style-type: none"> <li>-Pattern detection from large databases</li> <li>-Need complex algorithms for pattern matching</li> </ul>	<ul style="list-style-type: none"> <li>-Time and resource consuming</li> <li>-Not suitable to build knowledge bases for intelligent systems</li> </ul>
Complex Event Processing	<ul style="list-style-type: none"> <li>-Integration in complex systems</li> <li>-Complex Event Processing engine for online pattern detection</li> </ul>	<ul style="list-style-type: none"> <li>-User needs familiarity with SQL based languages</li> <li>-Contextual information is used for filtering purposes.</li> <li>-Dynamic situations are not represented</li> </ul>

*Table 2-1: A summary of different approaches for situation of interest modeling*

Another important aspect in the definition of situations of interest is contextual information. Human beings and software agents need to use situations of interest for decision making purposes. Therefore, the definition of situations of interest shall consider the human mental model which is context-based [Freksa et al, 2007]. Contextual information shall be considered when defining situations of interest. We already mentioned that theories such as Situation Management, Situation Awareness and Complex Event Processing attempt to integrate the notion of context into the definition of situations of interest. However, contextual information in these approaches is usually limited to spatial information (in Situation Awareness and CEP) and temporal information (in CEP). It is also used for filtering purposes in CEP.

Providing a novel definition of situations of interest from a cognitive perspective is a key step in this thesis. Several aspects shall be considered such as dynamic situation representation and support of contextual information. This is one of the main challenges of our thesis.

The representation formalism of situations of interest shall be carefully selected to allow for expressing complex phenomena detected in the real world and for building knowledge bases to be used either by humans or by software agents. Our literature review showed that in predicate-based approaches and, generally in Artificial Intelligence, First Order Logic has been often used to represent situations of interest. This representation formalism is quite limited when it comes to deal with complex spatiotemporal situations. In the domains of spatiotemporal databases and in Complex Event Processing, extended versions of SQL languages are usually proposed [Sakr and Guting, 2010], but the resulting queries are too complex to be manipulated by human operators. They require people with advanced knowledge of these SQL-based languages and they are not suitable to easily build knowledge bases that can be used by software agents. [Haddad and Moulin, 2010] used Conceptual Graphs (CG) to represent and to reason about spatiotemporal situations. This

approach seems promising thanks to the advantages of the CG representation language and will be explored in the next chapters of this thesis.

### **2.8.2 Integration of Situations of Interest in Large Scale Systems**

Another objective of this thesis is to propose a representation model of situations of interest which can be used in large scale monitoring and data acquisition systems. The proposed situation of interest formalism shall be easily integrated in such systems. Complex Event Processing is one of the promising approaches that help meeting such a requirement. Indeed, patterns representing situations of interest need to be detected on the fly from event streams which are continuously generated by distributed devices. Other works that deal with the semantic aspect of situations of interest such as [Haddad and Moulin, 2010] and [Gehrke et al, 2005], proposed some approaches to detect instances of these situations from simulated data (such as the MAGS-COA system in [Haddad, 2009]) or from real data. Moreover, theories such as *Situation Management and Situation Awareness* attempted to propose different architectures to manage the flow of data obtained from heterogeneous devices. These works tend to use algorithms closely related to the application domain in order to detect instances of situations. This makes the use of the proposed approaches in other application domains quite challenging. Hence, the Complex Event Processing technology remains one the best current options to manage patterns and to integrate their definitions in real applications.

## **2.9 Conclusion**

In this chapter, we presented a short survey of different concepts and techniques related to spatiotemporal situations modeling. First, we introduced general concepts related to a new generation of large scale monitoring data acquisition systems where these situations can be identified. Then, we surveyed different works on spatiotemporal situation modeling. We also proposed a brief overview of qualitative spatiotemporal representation and reasoning techniques which are widely used in

situation modeling approaches. Finally, we discussed the relevance the surveyed approaches and their limits to achieve the objectives of this thesis. The next chapters of this thesis will present the different contributions that we propose to overcome the aforementioned limits.



# Part II

## Contributions

### Part II presentation

One of the main objectives of this thesis is to propose a framework to qualitatively represent dynamic situations of interest and to facilitate reasoning about them. In Chapter 2 we discussed how dynamic situations of interest are usually referred to as patterns. In Chapter 3 we propose a formalism to qualitatively represent patterns using an extension of Sowa's Conceptual Graphs and we provide some examples of such patterns using a case study.

Spatial proximity is a key element of patterns that can be identified in several domains such as telecommunications and power distribution. We propose in Chapter 4 an approach to qualitatively represent spatial proximity. We developed a software that combines numerical distance and contextual information and uses a NeuroFuzzy classifier to deduce spatial proximity quantifiers. The system input is a training dataset and a set of inference rules provided by human operators based on their experience about a specific application domain. The system output is a set of membership functions of spatial proximity quantifiers.

Chapter 5 integrates our pattern model presented in Chapter 3 and our qualitative proximity tool presented in Chapter 4 in a pattern management framework. A case study from the power industry is presented to illustrate different aspects of this work.

# **Chapter 3**

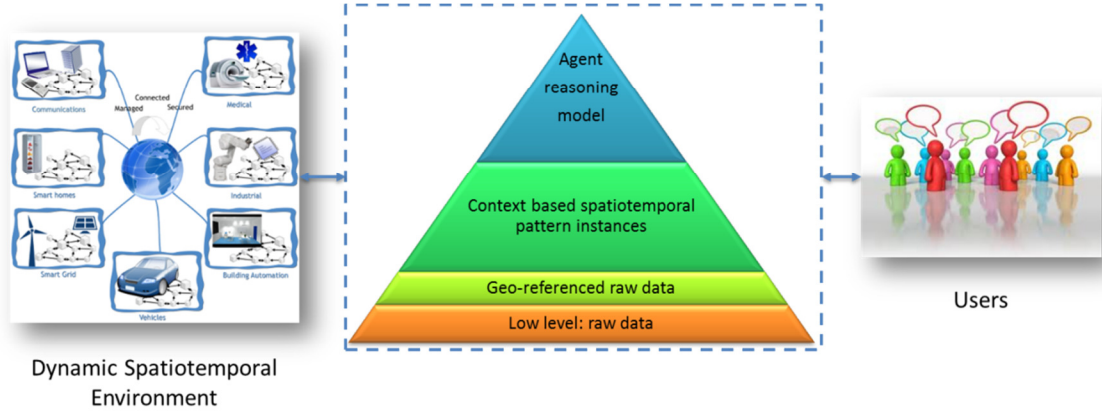
## **Qualitative Representation of Dynamic Spatiotemporal Patterns**

### **Introduction**

In this chapter, we propose a framework for the qualitative representation of dynamic spatiotemporal patterns (DSTP for short), which corresponds to the first objective of this thesis. As illustrated in Figure 3.1, our formalism aims at transforming raw data generated by different sensors geographically distributed in the real world in a form that agrees with the user's mental model more closely. To this end, the proposed representation formalism uses spatiotemporal patterns applied to geo-referenced raw data as well as contextual information to represent situations of interest. The detected instances of these patterns will populate a knowledge base which can be used by humans to make decisions or by software agents to support the human decision making process.

This chapter is organized as follows. Section 3.1 provides a definition of spatiotemporal patterns in the qualitative spatiotemporal reasoning domain. Section 3.2 introduces the Conceptual Graph approach and explains our motivation to use it to represent DSTPs. Sections 3.3 and 3.4 are the core of this chapter since they introduce the formal definition of the dynamic spatiotemporal environment and the formal definition of DSTPs as well as the contextual information in the pattern

definition. Section 3.5 presents a case study in the power distribution domain and some application scenarios to illustrate the proposed approach. In Section 3.6 we discuss some challenges related to our DSTP formalism and Section 3.7 concludes this chapter.



*Figure 3.1: A dynamic STP model*

## 3.1 Definition of Dynamic Spatiotemporal Patterns

In Chapter 1 we briefly discussed how situations of interest can be referred to as spatiotemporal patterns. In this section, we show how the notion of pattern can be used to represent situations of interest as perceived by human beings. We also provide a formal definition of spatiotemporal patterns which will be used in the rest of this thesis.

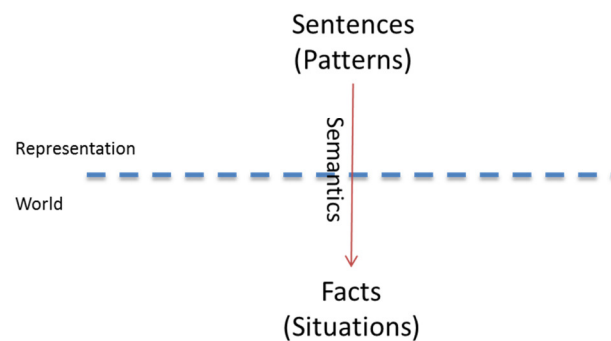
### 3.1.1 From Situations of Interest to Spatiotemporal Patterns

Human beings observe the world and find relevant links between phenomena and interpret the entailed observations according to the way they see the world [Political Psychology, 2003]. In fact, this is a cognitive process which is driven by the person's interest in certain configurations appearing in the real world in relation to these phenomena. Such 'configurations' take a certain meaning for a person according to her mental representation of the environment or of the real world. We will refer to

this representation as *the person's mental model of the environment*. These configurations will be called *situations of interest* in this thesis. We define a situation of interest as *a set of correlated information (phenomena in our case) which represents a specific configuration of the real world*.

We also have the objective to enable software agents to reason about situations of interest in order to support humans in their decision making process, Hence, we need a knowledge representation formalism to represent situations of interest in a computer tractable form.

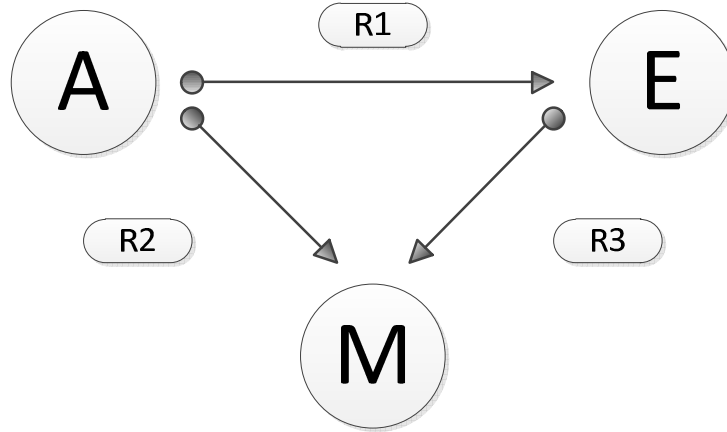
For example, let us recall that renowned authors in Artificial Intelligence [Russell and Norvig 1995] have described how knowledge representation formalisms are used to translate humans' perception of the world ('facts' or 'situations' in the context of this thesis) to statements or sentences that can be stored within software agents' knowledge bases (Figure 3.2).



*Figure 3.2: Representation of situations of interest using patterns (adapted from [Russell and Norvig, 1995])*

Let us also recall that the spatiotemporal research community represents a situation of interest (corresponding to a fact in Figure 3.2) causing the notion of spatiotemporal *pattern* (corresponding to a sentence in Figure 3.2).

Several researchers attempted to model operational environments in a way that agrees with humans' mental models. For example, [Freksa et al, 2007] proposed a cognitive approach to model humans' mental representation of the spatial environment using spatial contextual information and the potential interactions between agents and the environment. Freksa defined three main components: 1) an *agent* A which is associated with a set of goals; 2) a *spatial environment* E where agent A evolves and about which it maintains a mental representation; 3) a *map* M which is used by the agent to enrich his level of spatial knowledge of the environment E. Figure 3.3 depicts a simplified version of Freksa's model with the aforementioned three elements and the relationships holding between them. Relation R1 establishes a correspondence between the environment E and the mental representation of agent A. Relation R2 establishes a correspondence between the agent's mental representation and the map M. The environment E may use the map M through relation R3.



*Figure 3.3: Agent decision model for way finding problem. Adapted from [Freksa, 2007]*

Freksa's approach will be an inspiration for our spatiotemporal pattern definition in the next section. The use of Freksa's model is motivated by the fact that this model considers contextual information which plays a key role in human cognition. In Freksa's model, contextual information is only related to the spatial information which is provided by the map M. Our pattern formalism will extend this model to consider other kinds of contextual information such as the temporal and semantic

contexts. Although several works proposed algorithms and frameworks to match and recognize patterns, we have found a limited number of pattern definitions in the spatiotemporal research area. The notion of pattern is more related to the discovery of noteworthy structures that may exist in a large dataset: these works can be mainly associated with data mining techniques.

One of the pattern definitions found in the spatiotemporal analysis domain is provided by [Imfeld, 2000] which states that: “*Regular structures in space and time, in particular, repeating structures, are often called patterns. Patterns that describe changes in space and time are referred to as spatiotemporal patterns*”. Since we are interested in dynamic spatiotemporal situations of interest, we can extend Imfeld’s definition as follows:

**Definition 1:** “Regular structures in space and time, in particular, repeating structures, are often called patterns. Patterns that describe one or a set of dynamic spatiotemporal situations are referred to as *dynamic spatiotemporal patterns*.”

We use the notion of *dynamic spatiotemporal situation* which has a broader scope than the notion of *change* used by [Imfeld, 2000]. A dynamic spatiotemporal situation is a critical component of spatiotemporal patterns (STP for short) and it will be defined in the next sections.

### **3.1.2 Conceptual Graphs for Spatiotemporal Pattern Representation**

One of the goals of this chapter is to propose a representation formalism of spatiotemporal patterns in a way that they can be manipulated by software agents. These patterns are used to structure the *knowledge* required by agents and humans for reasoning purposes. Conceptual Graphs (CGs for short) are one of the representation

schemes widely used for knowledge representation. According to Sowa [Sowa, 1984]:

*Conceptual graphs (CGs) are a system of logic based on the existential graphs of Charles Sanders Peirce and the semantic networks of Artificial Intelligence. They express meaning in a form that is logically precise, humanly readable, and computationally tractable. With their direct mapping to language, conceptual graphs serve as an intermediate language for translating computer-oriented formalisms to and from natural languages. With their graphic representation, they serve as a readable, but formal design and specification language. CGs have been implemented in a variety of projects for information retrieval, database design, expert systems, and natural language processing.*

The CG formalism will be used in our pattern representation approach. This choice is motivated by several reasons. First, the CG formalism is a powerful knowledge representation formalism which offers an easy mapping from and to natural language. CGs are easily understood by knowledge experts who can quickly learn the formalism and easily use it to express spatiotemporal patterns. Second, CGs are equivalent to first order logic and offer manipulation and reasoning capabilities based on several operations such as generalization, specialization and join of conceptual graphs, to name a few, by taking advantage of their graph-oriented structure. They also enable the specification of *if-then* rules to build a rule-based system which can be integrated in agents reasoning models. Third, CGs have been successfully used in various applications such as natural language processing, and in particular, in qualitative spatial reasoning and qualitative simulation. For example, Moulin [Moulin, 1997] presented a temporal extension of Allen's formalism and proposed a temporal situation formalism. Mekni [Mekni, 2010] proposed a semantic abstraction of virtual geographic environments for multiagent geosimulation. [Arioua et al, 2014] used CGs in a query-answering system to access multiple data sources defined over a common ontology. In health care systems, [Kamsu-Foguem et al, 2014] used CGs to

represent clinical practice guidelines and protocols and to help users visualize different steps of the knowledge reasoning process. To the best of our knowledge, there is no existing work in the literature which uses CGs to represent spatiotemporal patterns. Using this representation formalism is one the original aspects of this thesis. According to Sowa [Sowa, 1984] a Conceptual Graph (CG) is a semantic network composed of concepts and conceptual relations. *Concepts* are representations of objects of the application domain. A concept is characterized by two elements: a type which represents the set of all the occurrences of a given class (i.e., human, animal etc.) and a referent which represents a given occurrence of the class that is associated with the concept (i.e., John, Mary, etc.). A CG can be represented using a graph notation and/or a linear notation. Using the linear notation, a concept is specified between square brackets: [TYPE-NAME: referent]. The concept types are classified within a type lattice whose root is the *universal concept*, denoted *T*. The type lattice supports operations on concepts such as generalization and specialization as well as the determination of the minimal common generalization and of the maximal common specialization of two concepts. A *conceptual relation* links two concepts or more. When it links two concepts, a conceptual relation is binary. Using CG linear notation, a binary conceptual relation is represented between brackets and is associated with concepts by means of arrows. The following example illustrates the linear representation and the graph representation of the sentence “A *severe storm* in *Quebec City*”

[Event: Storm]-  
 -(CHRC)→[Severe]  
 -(loc)→[City: Quebec City]

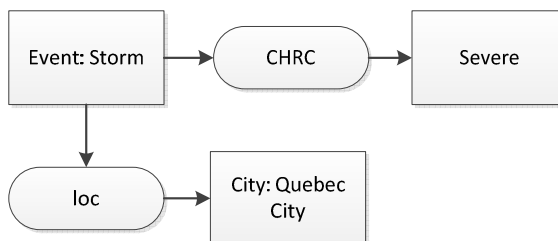


Figure 3.4: A Conceptual Graph example using linear and graph representations



## 3.2 A Dynamic Spatiotemporal Environment

Spatiotemporal patterns are defined in a dynamic spatiotemporal environment where geo-referenced data is reported by different types of devices geographically distributed. Several works in the literature characterized dynamic environments. For example [Grenon and Smith, 2004] proposed to characterize a dynamic environment using two different layers. A static layer defines static objects of the environment such as geographic objects. A dynamic layer defines dynamic spatiotemporal situations called by Galton “occurents”. A similar approach was proposed by Haddad and Moulin [2010] who introduced a static view and a dynamic view (Figure 2.16) in their spatiotemporal situation formalism. In this thesis, for the characterization of our dynamic spatiotemporal environment we propose to extend this approach to include spatiotemporal patterns. Our conceptual model is depicted in Figure 3.5, where we characterize a spatiotemporal pattern by several components appearing in the endurant and the dynamic views. Each component can be represented by a knowledge structure called a *concept*. Since patterns are represented using conceptual graphs, they can be organized in a *Concept Type Lattice* [Sowa, 1984]. In Figure 3.6 we present a concept type lattice that organizes concepts such as situation, context and spatial object and their specialized concept types. These concepts will be defined throughout the next subsections.

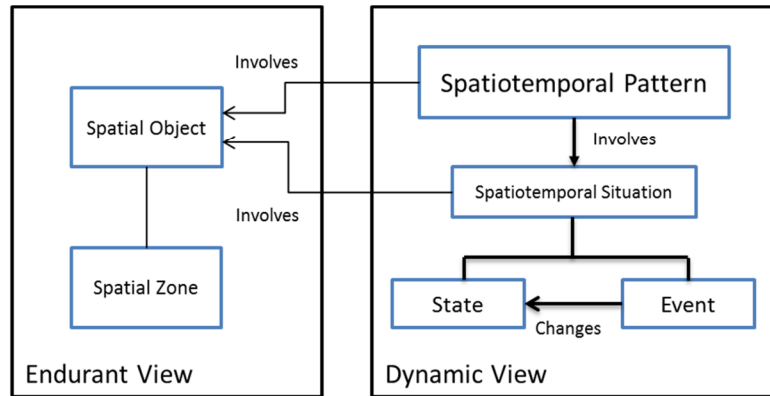


Figure 3.5: Our conceptual model for spatiotemporal patterns. Adapted from [Haddad and Moulin, 2010]

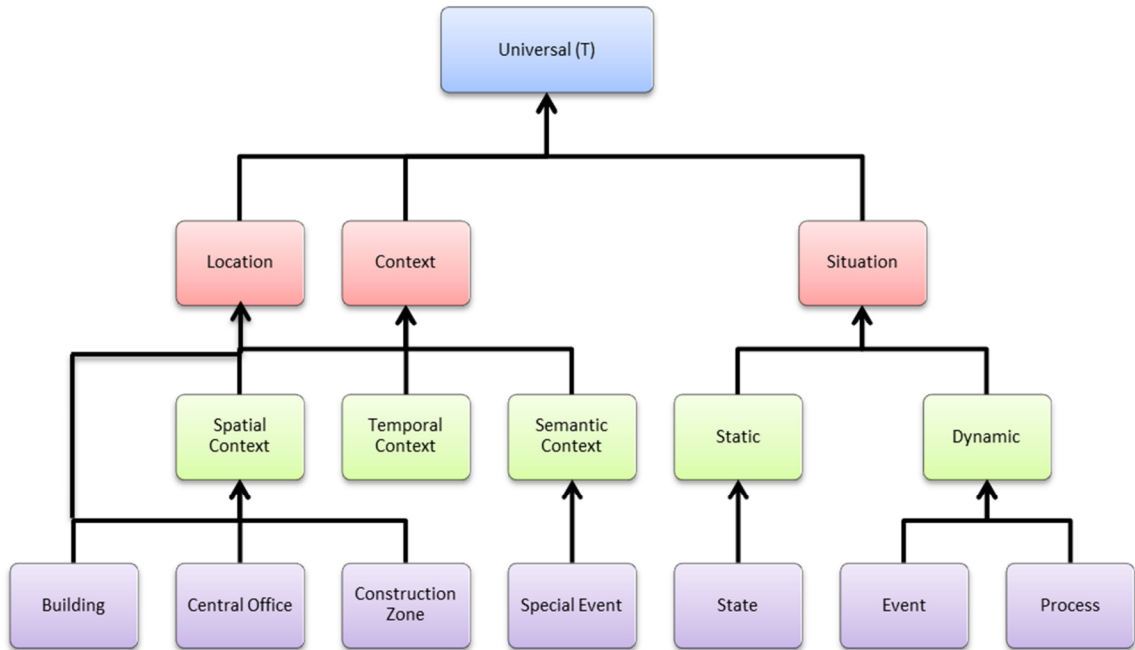


Figure 3.6: An example of concept lattice including context specialization

Let us mention that a Concept Type Lattice is used instead of a hierarchical tree structure because although trees are considered to be the simplest hierarchies (where each type except the top has only one immediate super-type), they cannot represent a type which has two supertypes. For example, the type *Building* has *Location* and *Spatial Context* as super-types. A lattice allows types to have multiple immediate super-types [Sowa, 1984].

### 3.2.1 Spatial Objects

*Spatial objects* are defined as spatial entities which are part of spatial regions. *Spatial regions* are part of the “container view” called *Space* which is the entire spatial universe according to [Grenon and Smith, 2004]. A spatial object has its own borders that distinguish it from other objects in the environment. *Borders* can be concrete for objects such as buildings, roads and mountains or abstract for objects such as cities and countries. Spatial objects belong to the enduring view. They capture individual

things extended in space that can be identified and described by properties and relations. Spatial objects can be classified into *object types* based on shared properties and relations [Kuhn and Ballatore, 2015].

### 3.2.2 Spatial Relations

*Spatial relations* are used to describe the relative spatial position of entities defined by spatial attributes. They are usually binary relations [Cohn and Renz, 2007]. Generally, spatial relations are categorized into three families: directional/orientation relations, distance relations and topological relations. An overview on these spatial relations was already presented in Chapter 2. Figure 3.7 depicts the spatial relation *inside* which links two spatial objects (a Sensor and a Building) using the CG linear notation.



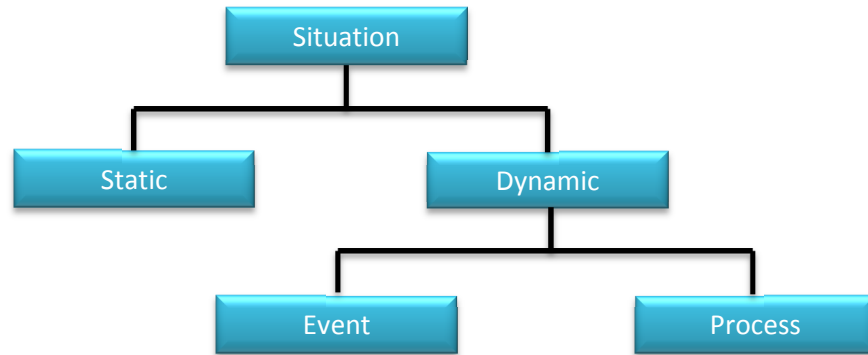
Figure 3.7: An example of a topological relation between a Sensor and a Building using a Conceptual Graphs representation

### 3.2.3 Dynamic Spatiotemporal Situations

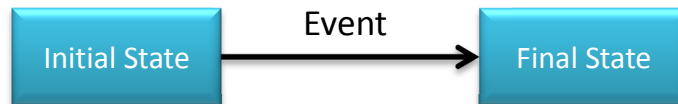
Spatiotemporal situations are part of the dynamic view as described by Figure 3.5. A *situation* is a finite configuration where objects that are part of this situation, have properties and are related to each other [Sowa 1984]. When a situation evolves in time and is defined with regards to spatial attributes and entities from the environment, it is called a *spatial-temporal situation*. According to [Desclés, 1985], situations can be categorized in two types: static situations and dynamic situations (Figure 3.8). *Static situations* remain stable during a given time interval and no change on the subject described by the situation is observed. *Dynamic situations* define changes in time and space references. A change is a transition from one state

to another (Figure 3.9). Dynamic situations can be either categorized as *events* or *processes*. To conclude, a spatiotemporal situation is based on two main aspects:

- It can be static or dynamic. When static, it is named a state. When it is dynamic, it can be either an event or a process.
- Changes from one state to another are triggered by the occurrence of events or of processes.



*Figure 3.8: Static and dynamic situations according to Desclés*



*Figure 3.9: A change from one state to another triggered by the occurrence of an event*

Our spatiotemporal pattern model will be used to represent situations of interest defined in the context of large data acquisition systems existing in the telecommunication and the power distribution industries. In these applications, dynamic situations can be represented using states and events. Consequently, only states and events will be considered in the definition of spatiotemporal situations in this thesis, although the model proposed by [Haddad, 2009] included the concept of process to represent complex geographic phenomena. However, the formalism that

we propose will allow for extending this definition to include the concept of process and to represent more complex situations if needed in future applications. This may be useful in other domains such as weather monitoring (for example, a thunderstorm can be represented as a process) and multi-agent geo-simulation.

### 3.2.4 Temporal Relations

Examples of temporal relations are *A* occurred After *B*, *A* Meets *B* and so on. The 13 qualitative temporal interval relations that have been introduced by Allen in his interval algebra [Allen, 1983] are used for this purpose, with the difference that we are not concerned with a continuous time and we deal only with discrete time. Moulin proposed an extended version of Allen's logic using CG representation formalism. He considered “*Before*” and “*During*” as two primitive relations [Moulin, 1997].

Considering two time intervals *A* and *B*, *BT* and *ET* respectively correspond to *Begin Time* and *End Time* for a given time interval. The temporal relation *Before*(*A*, *B*) holds if we have the following conditions:  $BT(A) < ET(A)$ ;  $BT(B) < ET(B)$  and  $BT(A) < BT(B)$ .

Considering the operation *DB* which stands for distance between the  $BT(A)$  and  $BT(B)$ , and *DE* the distance between the  $ET(A)$  and  $ET(B)$  the relation *During*(*A*, *B*, *DB*, *DE*) holds under the following constraints:  $BT(A) < ET(A)$ ;  $BT(B) < ET(B)$ ;  $BT(A) \geq BT(B)$ ;  $ET(A) \leq ET(B)$ ;  $BT(A) - BT(B) = DB$  and  $ET(B) - ET(A) = DE$ . Figure 3.10 illustrates an example of the *During* relation with all the corresponding parameters.



Figure 3.10: Illustration of time intervals according to [Moulin, 1997]

Temporal relations can be used to link two situations together or one situation and a temporal reference. In Conceptual Graphs, they can be represented using conceptual

relations. Figure 3.11 depicts an example of the temporal relation *Before* used to link the event Storm and the temporal reference 12:00 AM.

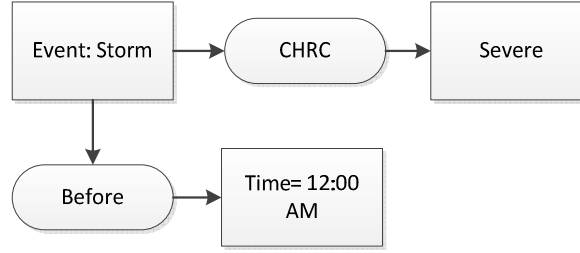


Figure 3.11: An example of temporal relation using Conceptual Graphs representation

### 3.2.5 State

A state is part of the dynamic view. It is defined, according to [Desclés, 1994], as a “static situation describing the stability and the non-change”. We use the state representation and the *during* temporal relation proposed by Haddad [Haddad, 2009] who extended the temporal situation formalism proposed by Moulin [Moulin, 1997] to support the spatial attributes of a spatiotemporal situation. A state is a spatiotemporal situation with the following elements:

- A pair (*situation type*, *situation referent*) where *situation type* takes the value of a state type from the state type hierarchy and *situation referent* is a reference to the state instance. The type hierarchy is a knowledge structure based on the CG formalism used to represent concept types and relation types.
- *State Propositional Content* (SPC) which is a knowledge structure describing the state. This structure is non-temporal, which means that it does not contain any time reference.
- *State Time Interval* (STI), which is a knowledge structure that allows for representing the temporal information associated with the state.
- *State Spatial Attribute* (SSA), which is a knowledge structure that represents the spatial information associated with the state.

Haddad and Moulin [2010] extended the graphical representation of CGs to allow for a compact representation of these four elements. Figure 3.12 depicts an example of a situation identified by #FiberState describing the state of a fiber optic link identified by #QuebecMontreal during the period between July 17th and 24th 2013. The fiber status is normal. Note that it is possible to have a state without the “ET” attribute (ET= end time) which means that the state has not changed yet. Figure 3.13 depicts an example of a degraded state of the fiber optic at distance 23 Km, without end time. Figure 3.14 illustrates the CG linear notation of the state given in Figure 3.12.

State : #FiberState; BT:July 17th 2013 10:00AM; ET July 24th 2013 05:33 PM
[Fiber :#QuebecMontreal]->(status)->[Normal]

*Figure 3.12: A state describing a normal fiber state. The state is defined in a time interval between July 17th at 10:00 AM and July 24th at 05:33 PM.*

State : #fiberState; BT:July 17th 2013 10:00AM
[Fiber :#QuebecMontreal]-(status)->[Degradation]
[Distance]-(Attr)->[Real=23]

*Figure 3.13: A state describing a degraded fiber state. The state is defined at distance 23Km of a fiber link between Quebec and Montreal*

[State: #fiberState]-
-(BT)->[Date=July 17th 2013 10:00AM]
-(ET)->[Date=July 24th 2013 05:33 PM]
-(SPC)->[Proposition:[Fiber: #QuebecMontreal]->(status)->[Normal]]

*Figure 3.14: A state represented using CGs' linear notation*

### 3.2.6 Event

Events are also part of the dynamic view. They can be defined according to two dimensions: 1) *The semantic dimension*, which aims at studying the meaning of events and at defining its significance in the geospatial world and 2) *The representation dimension* which aims at defining the programming entity used to represent the event object [Etzion and Zolotorvesky, 2010]. In the semantic dimension, Haddad [Haddad, 2009] adopted Desclés’s event definition as a “*temporal occurrence that appears in a static background, which may or not change a state in the world*”.

Other researchers proposed event definitions. Luckham [Luckham, 2002] defined an event by “*something that happens in reality*”. [Cole and Hornsby, 2005] stated that: “*events capture happenings or activities in a domain that require intervention, for example, a system-generated notification that evidence of a significant concurrent has been detected in the database*”. Galton [Galton, 2004] defined two types of events: punctual events and durative events. Punctual events mark the onset or termination of states of affairs (for example, an object is starting to move). The OGC [2015] proposed a similar definition which states that an event is an action that occurs at an instant or over an interval of time. In our model, we consider an event as a happening and we only consider punctual events as proposed by [Cole and Hornsby, 2005]. All these models used a similar event structure. The event is defined by an *event type*, an *event identifier*, a *temporal attribute* (either time or time interval) and other attributes that may vary from one approach to another. For example, if the event is defined in a geospatial domain, the spatial attribute is specified in the event definition. Usually, an event type is defined to allow event instantiation.

In the Complex Event Processing research community, Etzion [Etzion and Zolotorvesky, 2010] proposed a general structure for an event object containing three main structures: a *header* which contains the meta-data about the event, a *payload*



which contains specific information about the event occurrence and finally an *event relation*, which is optional and handles semantic relationships between events.

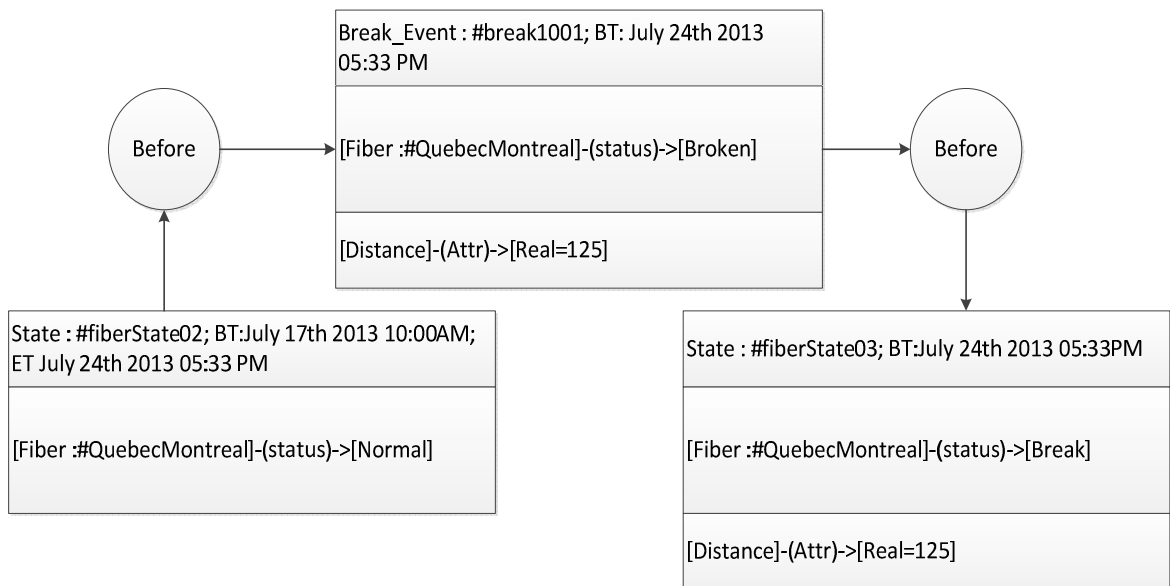
Since an event means something that happens and results in a change from one state to another, several works attempted to consider the *notion of change* in the event model. For example, Cole and Hornsby [Cole and Hornsby, 2005] proposed a model where all events (called occurrences) have a spatial attribute, a time stamp and a spatial attribute named “expected next” that allows for representing what is the next spatial zone reached by a mobile object.

Using temporal relations Haddad [Haddad, 2009] proposed a more generic approach where an event is a spatiotemporal situation linking two states. We use this approach in our model because it explicitly represents change using temporal relations between an event and the initial and final states. This is an important issue in large scale data acquisition systems where a user needs to know the state of the system before and after the event’s occurrence.

We define an event as a spatiotemporal situation with the following elements:

- A pair (*event type*, *event referent*), where *event type* takes the value of an event type from the event type hierarchy and *event referent* is a reference to the event instance.
- *Event Propositional Content (EPC)*, which is a knowledge structure describing the event. This structure is non-temporal, which means that it does not contain any time reference.
- *Event Time Stamp (ETS)*, which is a knowledge structure that allows for representing the temporal information associated with the event.
- *Event Spatial Attribute (ESA)*, which is a knowledge structure that represents the spatial information associated with the event.

We use temporal relations to link the event occurrence with the state holding before it, as well as with the state which holds after it. Moulin’s formalism [Moulin, 1997] introduced in Section 3.2.4 is used to represent temporal relations. Figure 3.15 graphically illustrates an instance of an event of type *break* that occurred on a fiber optic link between *Quebec City* and *Montreal*. The break occurred at distance *125 km* at *05:33 PM* and changed the state of the fiber optic from “*normal*” to “*broken*”. Note that the state which comes after the break event has a begin time only; whereas the state that comes before the break event has a time interval delimited by BT (Begin Time) and ET (End Time). In addition, spatial information about the event may be present in the resulting state to provide further information about the position where the break occurred (at distance 125 km).



*Figure 3.15: An event representation using temporal relations to link with the state before the occurrence and after the occurrence*

### 3.2.7 Summary on the Spatiotemporal Situation Representation

The definition of spatiotemporal situation (state or event) is mainly adapted from Haddad’s model [Haddad, 2009]. This choice was motivated by several reasons. First, the spatiotemporal situation definition is compatible with most of the event and state definitions in the literature. It integrates spatial and temporal attributes, which

makes it agree with the spatiotemporal environment defined in Freska’s model (introduced in Section 3.1). Second, the spatiotemporal situation definition considers both states and events and allows for representing dynamic situations whereas other works in the literature restrict their definitions to events. Third, the proposed model uses the conceptual graph formalism, which makes it expressive and human readable. This was one of the requirements for our spatiotemporal pattern representation. States and events can be related by temporal relations. Spatial relations can be used in situation definitions to relate the events’ spatial attributes. Now that we presented our representation of a dynamic spatiotemporal environment and the main concepts related to it, we present in the next section our spatiotemporal pattern definition and we propose some pattern types.

### 3.3 Dynamic Spatiotemporal Pattern

A spatiotemporal situation is defined as a finite configuration involving states and events, spatial and temporal relations. Different configurations representing these situations may generate different pattern types. Some existing works recommend using predefined pattern types. Among them, the OGC initiative [OGC, 2015] submitted a work proposal with a classification of pattern types. Although this work is not in its final stage, we will use it in our pattern classification. Four types of patterns are defined: simple pattern, repetitive pattern, complex pattern and timer pattern.

Using our definition of spatiotemporal situations in the previous sections, we introduce here the concept of *qualified spatiotemporal situation* which is a key element of our definition of spatiotemporal patterns. Then, different pattern type definitions will be proposed.

### 3.3.1 Qualified Spatiotemporal Situation

When an agent reasons about a situation, it often needs to “locate” it relatively to the environment. To this end, it can use the situation’s spatial and temporal attributes. Such a cognitive activity is also called *qualification* [Galton, 2004]. Agents need to qualify situations with regards to a temporal or spatial reference because a situation may be interpreted in different ways. For example, a cyclone observed by a city habitant can appear as a storm for another village habitant because both habitants who are qualifying such an event are located in two different spatial regions and observe the cyclone with two different time references [Galton, 2004]. Hence, we introduce the concept of a *qualified spatiotemporal situation* (*qualified STS* for short) in the following definition:

**Definition 2:** A *qualified STS* is a situation which has a *qualifying relation* (temporal and/or spatial) that *qualifies its temporal and/or spatial attributes relatively to the environment*.

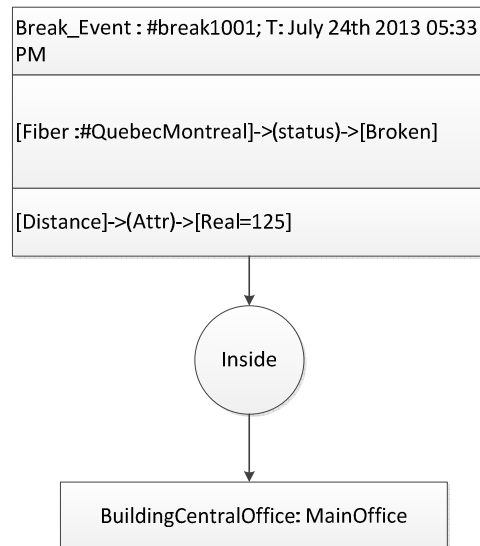


Figure 3.16: An example of a qualified situation

Figure 3.16 illustrates an example of a qualified STS describing the following configuration: a “Break Event” event type occurred at distance “125 km” on the spatial object “Fiber #QuebecMontreal” and *inside* the spatial object building of type “central office”.

### 3.3.2 Simple Pattern

A *simple pattern* is a qualified spatiotemporal situation. It is a finite configuration that can represent change in space and time and can be perceived by software agents or humans when recognizing phenomena that occurred in the environment. Therefore, we consider a qualified STS as the simplest way to define a spatiotemporal pattern.

**Definition 3:** *A simple pattern is a spatiotemporal situation that represents a change in space and time and that can be qualified using spatial and/or temporal relations relatively to the environment.*

This definition restricts the use of STS to only events since states do not represent changes in space and time. Indeed, an event may represent change when using temporal relations relating the event to a state before and to another state after its occurrence.

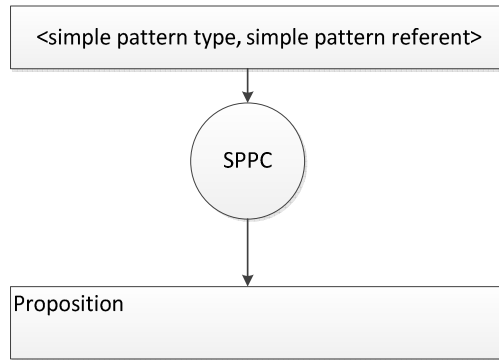
From the above definition, a simple pattern can take one of the following configurations:

- An event which triggers a change from one state to another. This is the simplest pattern configuration.
- An event which triggers a change from one state to another and whose its temporal attribute is qualified by a temporal reference.
- An event which triggers a change from one state to another and which has a spatial attribute qualified in relation to a spatial object.
- An event which triggers a change from one state to another and which has a spatial attribute qualified in relation to a spatial object, and which has a temporal attribute qualified in relation to a temporal reference.

- A state cannot be a spatiotemporal pattern since it does not represent a change.

A simple pattern is represented by a knowledge structure with the following elements:

- A pair (*simple pattern type, simple pattern referent*) where *simple pattern referent* is a reference to the simple pattern instance.
- A *Simple Pattern Propositional Content (SPPC)* which is a knowledge structure describing the simple pattern. This is basically the qualified spatiotemporal situation.



*Figure 3.17: Structure of a simple pattern*

Figure 3.17\_presents the structure of a simple pattern. Figure 3.18 illustrates an emergency pattern example where a fiber state changes to break after the occurrence of a break event at a certain distance. Figure 3.19 illustrates another example where the non-occurrence of an event (expressed by the negation operator “ $\neg$ ”) is considered as a communication error pattern and may require the verification of the corresponding sensor. Such a type of pattern can be very useful in sensor-based monitoring systems. A sensor can be in a healthy status but may not be able to report observations due to a communication failure. At the occurrence of such a pattern, a field technician may investigate the communication link to repair it.

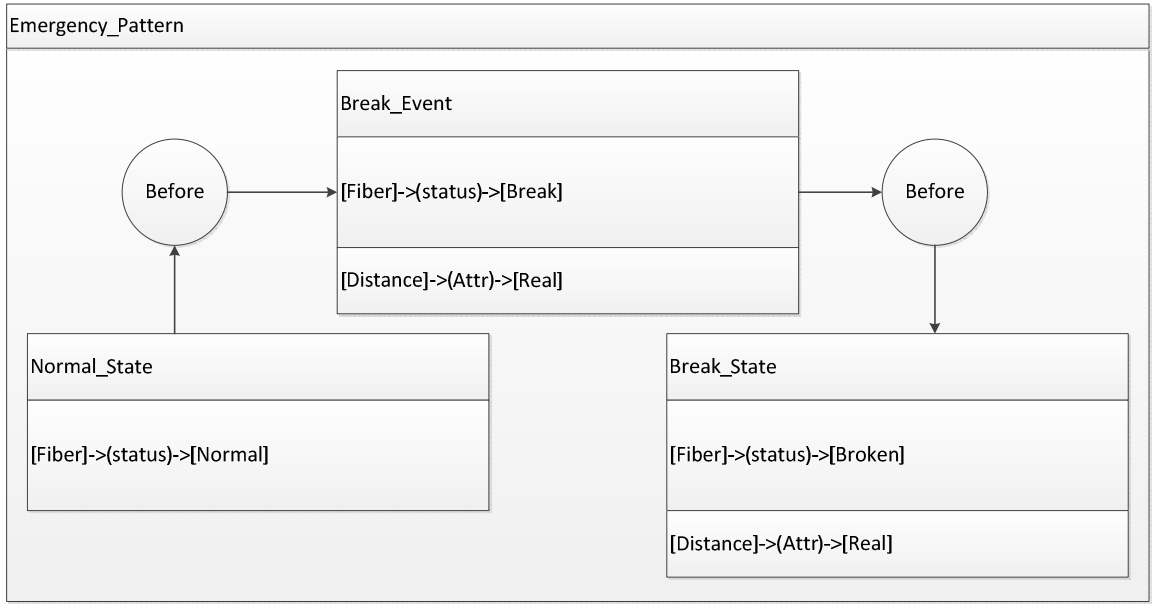


Figure 3.18: An example of a simple pattern

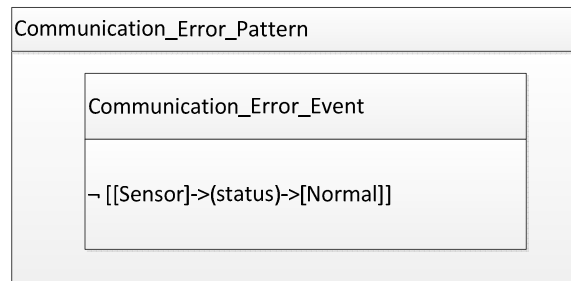


Figure 3.19: A simple pattern example using the negation operator

### 3.3.3 Complex Pattern

A software agent can be programmed to monitor or observe dynamic phenomena. During its observation activity the agent may detect interesting configurations that usually involve several STSs. A single STS can be interpreted and qualified to produce a simple pattern. However, such an interpretation may change when one STS is related to one or even several other STSs. When several STSs are related using spatial or temporal relations, they define what we call a *Complex Pattern*.

**Definition 4:** A complex pattern is a set of simple patterns related by spatial and/or temporal relations.

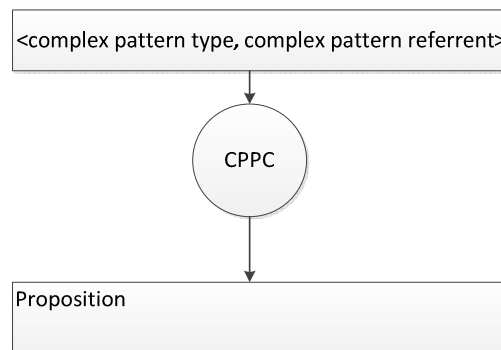
Two possible configurations can be present in a complex pattern:

Case 1: When a spatial relation links two simple patterns, the relation is established between the event spatial attributes of each simple pattern.

Case 2: When a temporal relation links two simple patterns, the relation is established between the event temporal attributes of each simple pattern.

In our representation model, a complex pattern is a knowledge structure which has the following elements:

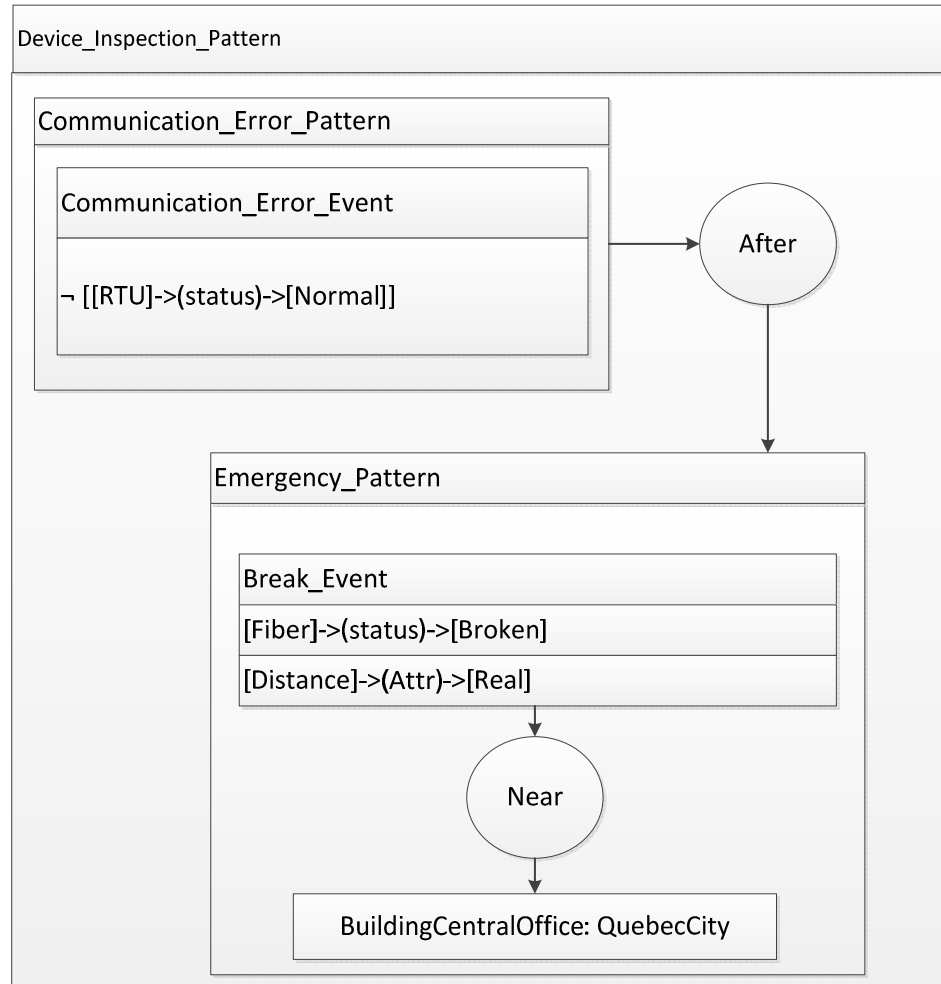
- A pair (*complex pattern type*, *complex pattern referent*) where *complex pattern referent* is a reference to the complex pattern instance.
- A *Complex Pattern Propositional Content (CPPC)* which is a knowledge structure describing the complex pattern. The content of this knowledge structure is basically a set of simple patterns related by temporal and/or spatial relations. It is worth noticing that temporal and spatial relations are binary. However, one simple pattern can be related to  $n$  simple patterns by  $n$  spatial/temporal relations.



*Figure 3.20: Structure of a complex pattern*



Figure 3.20 illustrates the structure of a complex pattern and Figure 3.21 illustrates a complex pattern example where the temporal relation *after* is used to link the simple pattern *Communication Error* to the simple pattern *Emergency*.



*Figure 3.21: A complex pattern example where the communication\_error\_pattern is related with the temporal relation after to the emergency\_pattern*

### 3.3.4 Timer Pattern

Some patterns can be qualified relatively to temporal information. Temporal information can be:

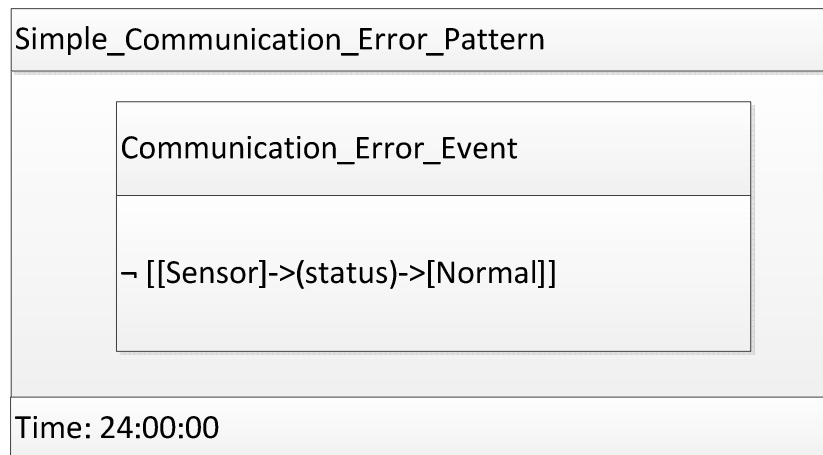
- A position in time (or temporal reference). For example Monday, April 16<sup>th</sup> 2015 10:05:03 AM;
- A time interval.

**Definition 5:** *A timer pattern is a pattern qualified with regards to a temporal information provided by a system clock.*

In our representation model, a timer pattern is a knowledge structure which has the following elements:

- A pair (*pattern type*, *pattern referent*) where *pattern referent* is a reference to the pattern instance. The pattern type can be simple or complex.
- A *Timer Pattern Propositional Content (TPPC)* which is a knowledge structure describing the pattern.
- A *Timer Pattern Content (TPC)* which is a knowledge structure describing the temporal information. A temporal relation is used to link the PPC to the TPC.

Figure 3.22 illustrates an example of a timer pattern describing a communication error on a sensor occurring after 24 hours and Figure 3.23 presents the general structure of the timer pattern.



*Figure 3.22: An example of a timer pattern*

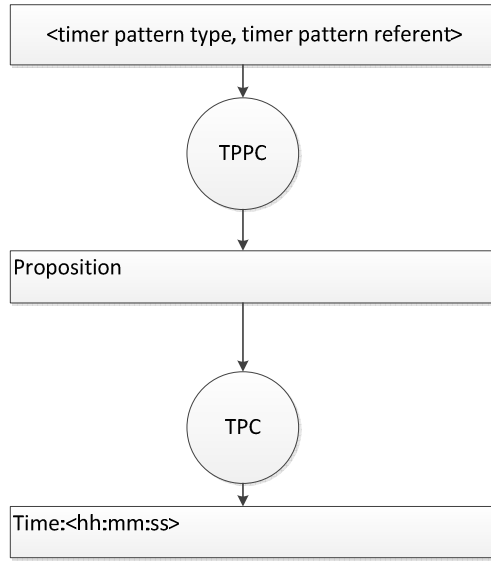


Figure 3.23: Structure of the timer pattern

### 3.3.5 Repetitive Pattern

In some applications, the repetitive pattern can be used to count the matches for a simple or a complex pattern instances. For example, a System Engineer may be interested to detect when a sensor reported a connection failure 5 times so that he can send someone to inspect the communication link between the sensor and the monitoring system.

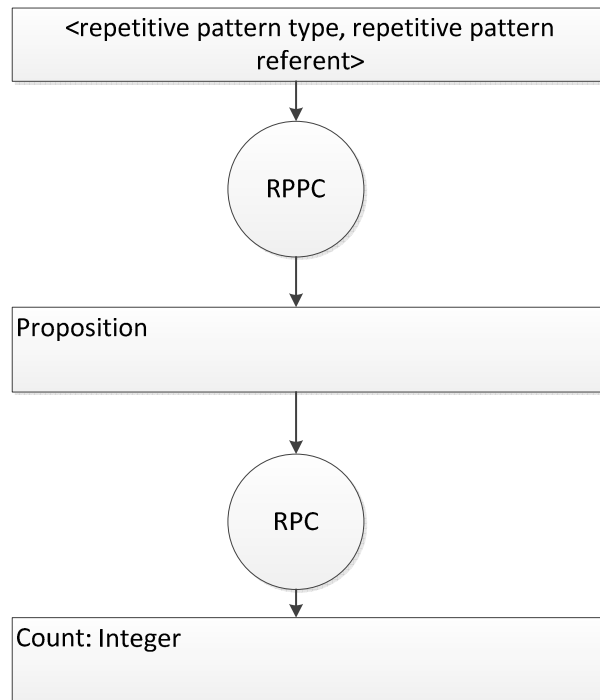
**Definition 6:** A repetitive pattern is the repetition of a sub-pattern for a specific number of times.

In our representation model, a repetitive pattern is a knowledge structure which has the following elements:

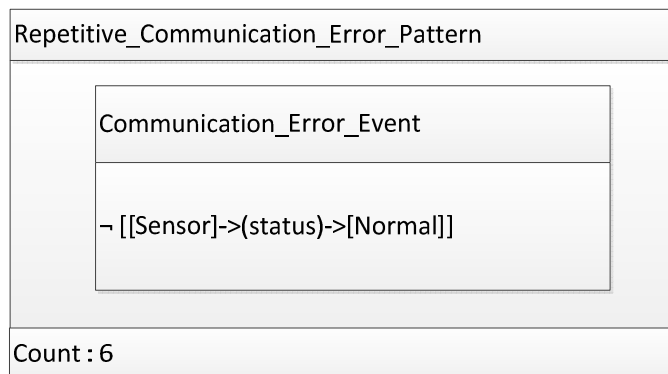
- A pair (*repetitive pattern type, repetitive pattern referent*) where *repetitive pattern referent* is a reference to the repetitive pattern instance.
- A *Repetitive Pattern Propositional Content (RPPC)* which is a knowledge structure describing the pattern which will be repeated.

- A Repetitive Pattern Count (RPC), which is the number of repetition of the sub pattern.

Figure 3.24 presents the structure of the repetitive pattern and Figure 3.25 illustrates an example of a repetitive pattern describing a communication error on a sensor occurring 6 times.



*Figure 3.24: Structure of the repetitive pattern*



*Figure 3.25: An example of a repetitive pattern*

### **3.3.6 Conclusion**

The core elements of our pattern formalism are events and states which allow for representing dynamic situations. We proposed in this section a number of pattern types that may be used in real time data acquisition systems. The representation formalism allows for the definition of other patterns types suitable to specific application domains. Thanks to CGs, our patterns can serve as knowledge base for software agents. As discussed in Section 3.1, the agents may have a mental model of the environment and may have their own interpretation of situations of interest according to contexts.

## **3.4 Contextual Information**

In this section, we discuss how contextual information can be integrated in our spatiotemporal pattern definition to enhance agent's reasoning capabilities. Our literature review in Chapter 2 showed that contextual information is not supported by most of spatiotemporal pattern models. In this section, we adapt Frekas's model introduced in Section 3.1 to include contextual information in our pattern formalism. Let us first discuss the role of contexts in pattern definitions. Then, we study how context can be defined in our pattern formalism and we propose a formal definition.

### **3.4.1 The role of context in knowledge representation**

Generally speaking, a given entity can have properties which have different meanings when this entity is evaluated in some contexts. In other words, context is defined as additional information in the perceptual/cognition level which allows for “perceiving or recognizing something that is actually not there” [Toussaint, 1978]. In the knowledge representation community the relationship between context and knowledge is a debated issue. [Bastien, 1998] states that “context cannot be separated from the knowledge it organizes, the triggering role context plays and the field of validity it defines”. According to Toussaint, the use of context tackles three main

problems in pattern recognition: disambiguation, error correction and filling the gap. Filling the gap means providing more information which is missing to reason about the related pattern. McCarthy [McCarthy, 1993] states that “formulas are not just true or false; they are true or false in some contexts”. Patterns which play the role of formulas here can be also true or false in some contexts. They are interpreted in some contexts and may assign different meanings about the situations of interest that occur in the world.

### 3.4.2 How to Define Contexts in Patterns?

Now that we have discussed the role of context in knowledge representation, let us try to find a suitable context definition for our pattern formalism. In Artificial Intelligence, people agree that no clear and final definition of context can be provided. It is difficult to find a context description common to most disciplines [Bazire and Brezillon, 2005]. Pervasive Computing is one of the domains where the concept of context is very popular. For example, in location-aware applications, contextual information is related to spatial information and used to help software agents track objects positions and adapt their goals [Chen and Kotz, 2000]. Even in context/location aware applications, a clear definition of context is missing and tends to be more focused on the spatial aspect and does not consider other dimensions (temporal, semantic, etc..).

In Complex Event Processing, context is defined relatively to the sole notion of event. For example, context according to [Sharon and Etzion, 2008] “specifies the relevance of the events participating in pattern detection”. The authors define three types of contexts: *temporal context*, *spatial-based context* and *semantic-based context*. [Adi et al, 2003] states that context partitions the space of events according to several dimensions; *temporal*, *spatial*, *state-based* and *semantic*. These two definitions are adopted by the *Complex Event Processing* research community. They tend to partition the so-called event cloud into multiple instances and do neither consider other pattern components nor the cognitive aspect of the application. They use context for event filtering purposes. In contrast, our pattern definition does not

consider only events but also involves other components such as states and spatial objects.

Brezillon [Brezillon, 1999] worked on the context definition for more than 25 years and he concluded that since context does not have a clear definition, it is better to design a knowledge model that takes into account contextual information and helps in making decisions. A similar conclusion was drawn by [Freksa et al, 2007] who proposed an approach to model context from a cognitive perspective using a modular architecture.

In our model, we adopt Freksa's approach, where the context is defined from a cognitive perspective in order to provide additional information that helps agents to interpret spatiotemporal patterns. We use Freksa's model as an inspiration to define the relationship between contexts and spatiotemporal patterns.

### **3.4.3 Our Definition of Context in Patterns**

Figure 3.26 depicts an overview of the approach that we propose. The relation R1 between the Agent (A) and Environment (E) defines the way A perceives situations of interest using our pattern model. These patterns can be interpreted according to contextual information defined by the relation R2 between the agent A and the context base C. In our model, the context base C contains three types of contextual information: M (map) for the spatial information as defined in Freksa's model, S (semantic) and T (temporal).

Formally, a context is a knowledge structure expressed using an extended form of Conceptual Graphs having the following elements:

- A pair (*context type*, *context referent*) where *context type* is a concept type that takes values depending on the application domain.
- *Context Propositional Content (CPC)* which is a knowledge structure describing the contextual information according to the context type.

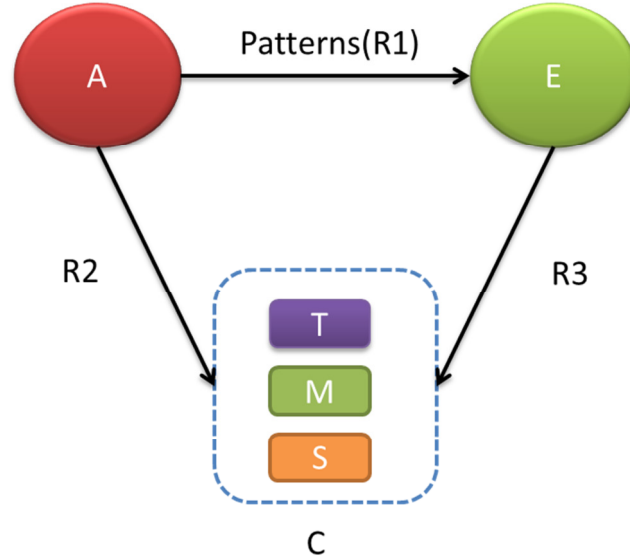


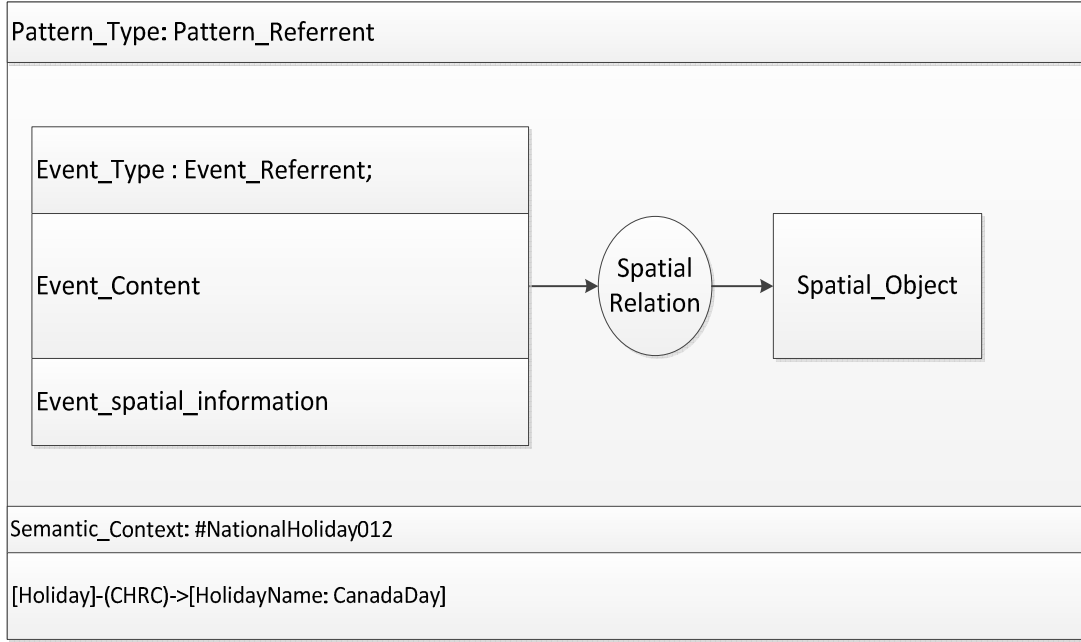
Figure 3.26: Our approach to model support contextual information in our spatiotemporal pattern formalism

The most used context types in the Pervasive Computing and Event Processing domains [Etzion and Zolotorvesky, 2010] are *semantic context*, *temporal context* and *spatial context*.

Semantic context is a non-temporal and non-spatial information that can be added to characterize the related pattern. “National Holiday” or “Vacations” are examples of semantic contexts that can be used to characterize patterns. Temporal context is a temporal information structure that can be a time interval or a single time reference. Spatial context is a spatial information structure. *Context referent* is the identifier of the instance of the context type in our definition.

Figure 3.27 depicts an example of a spatiotemporal pattern with a semantic context of type National Holiday. In some applications, a pattern that occurs during a national holiday may have a special meaning and require a special attention from decision makers.





*Figure 3.27: A pattern definition using contextual information*

### 3.4.4 Conclusion

We introduced the contextual information in our pattern definition by extending Freksa's model. Our definition goes beyond spatial context and includes temporal and semantic contexts. Moreover, the contextual information is represented using a conceptual graph.

Introducing the contextual information in the spatiotemporal situation model is an original aspect of the model proposed in this chapter. Since in the literature classical pattern definitions use query based languages, they are limited in terms of expressiveness and do not make use of contextual information. Other semantic-based models such as [Tan et al, 2009], [Lin et al, 2009] and [Aigong, 2009] do not support contextual information either. In Complex Event Processing, contexts are used for filtering purposes whereas our model uses contextual information to extend the interpretation of situations of interest.

In the next section, we present a case study to illustrate some examples of patterns that may be detected to represent situations of interest in the domain of Power Distribution.

### **3.5 Case Study: Outage Management Systems**

Since this thesis is defined in the context of large scale monitoring and data acquisition systems, we apply our pattern model to the Power Distribution domain and in particular to Outage Management Systems (OMS). We define qualitative spatiotemporal patterns to detect interesting dynamic spatiotemporal situations and to help agents making decision about crew assignment and outage status in a distribution network. This case study will provide some examples of the use of spatiotemporal patterns by agents in their decision making process. Consequently, some assumptions are proposed for simplification purposes. We assume that pattern instances are already populated in a given knowledge base and are accessible by agents. Here, we address neither issues related to data collection from the event cloud, nor the automated agent interaction with the environment and pattern detection aspects.

#### **3.5.1 Background**

An increasing pressure from customers (residential, commercial and industrial) and media is placed over electric distribution companies to supply reliable information about outage in their distribution network. An OMS is a computer system used by operators of electric distribution systems to assist in restoration of power [Chakravarty and Wickramasekara, 2014]. Using OMS mainly aims at reducing the outage time in the distribution or transmission network for power utilities and at keeping customers updated about the real time status of an outage in the network. To this end, an OMS offers a variety of capabilities. It needs to predict the location of electric devices such as switches and breakers that opened during a failure. It also needs to identify faulty electrical devices such as the transformer or the cable that

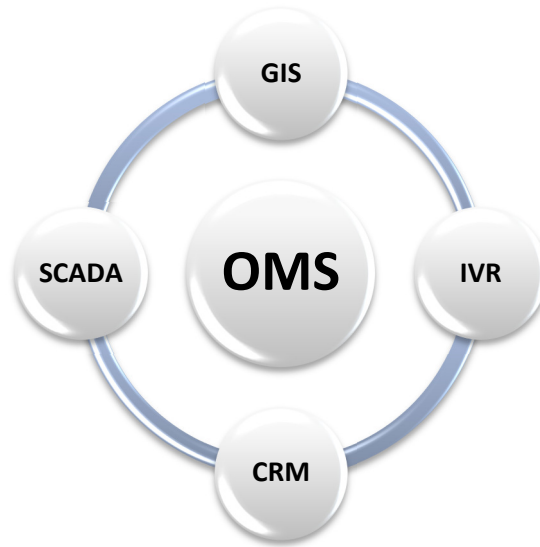
caused the outage. Furthermore, it can help prioritize restoration efforts based on several criteria such as the location of emergency facilities, crew availability, crew expertise, and the outage size.

An OMS is usually designed to integrate information from several software applications. A GIS is used for spatial network modeling and to locate the outage when the related event is reported. An outage event (or “incident”) can be reported in different ways. The first source of outage information is the Supervisory Control and Data Acquisition System (SCADA) which is a system designed to monitor the status of electrical devices in real-time, stores acquired information in historian<sup>3</sup> databases and reports the information when a fault occurs. The second source of outage information is the Interactive Voice Response (IVR) which is a software application that answers customer’s phone calls, routes information, compiles information and recalls customers as programmed. Besides outage information, the OMS is connected to a Customer Relation Management (CRM) system. A CRM is a software application that manages customers’ information and profiles. It manages billing information and stores customers’ requirements and preferences. These systems are used by the OMS to achieve two main goals: keeping customers informed about the status of an outage and dispatching the crews according to several criteria. Figure 3.28 depicts an overview of a standard OMS architecture.

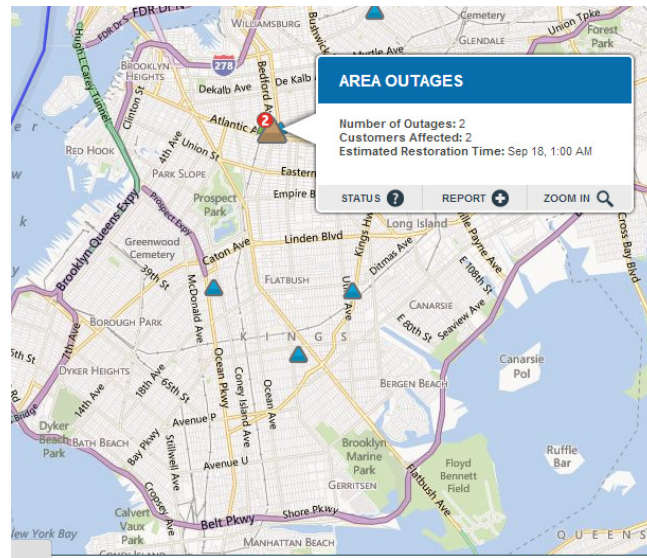
The main function of an OMS is to help managing crews and to update customers with the real time outage status in their respective regions. Figure 3.29 depicts an example of an outage map where an outage is associated with a particular area. The number of outages can be grouped by subareas. The number of customers affected and the estimated restoration time is displayed.

---

<sup>3</sup> A historian is the term used in the electrical power industry referring to a software application that logs time-based process data and records trends in a database for future reference.



*Figure 3.28: Different components of an Outage Management System*



*Figure 3.29: An outage map in the State of New York [conEdison, 2013]*

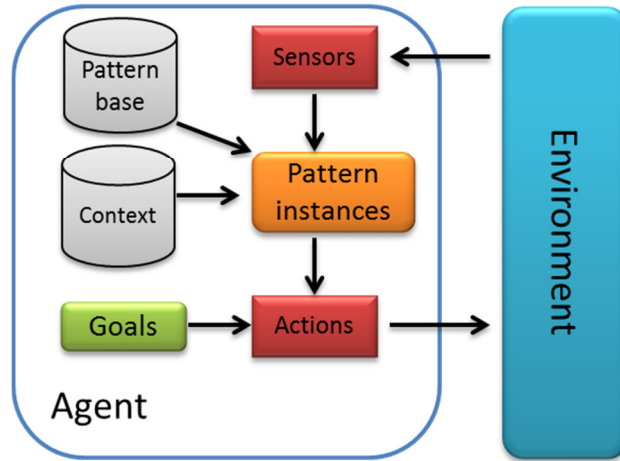
### 3.5.2 Proposed Approach

We propose to develop a software agent that correlates information collected from several data sources and establishes relationship between them to identify interesting situations. Data sources which belong to the environment may be static or dynamic. Static data sources provide information such as states and geographic data. Dynamic

data sources (such as sensors) provide information about outage events. The agent has its own contextual information and may correlate it with data provided by OMS sources in order to interpret interesting situations. Based on the result of this interpretation, an agent can suggest solutions for crew assignment and updates the outage status.

Software agents' reasoning models can be classified in four categories according to [Russel and Norvig, 1995]: *simple reflex* agents, agents *that keep track of the world*, *goal-based* and *utility-based* agents. In our approach, we adopt a goal-based agent architecture. An agent maintains a list of goal specifications stored in the *Goal Specification* module. The agent's reasoning module uses contextual information, retrieves information from different data sources (mainly sensors) and uses its mental representation of the environment to detect instances of patterns. Actions will be taken according to the agent's goals which are updating the outage status and assigning crews to the new/existing outages according to crews' availability and to the occurrence of outages in the same outage area. Figure 3.30 illustrates an example of our agent's reasoning model adapted from [Russel and Norvig, 1995].

In order to implement our spatiotemporal pattern model, we use the *Amine Platform* [Kabbaj, 2006] to define CGs, to build a knowledge base and to define the reasoning mechanisms used by agents. Amine is a Java based multi-layer platform designed for the development of intelligent and multi-agents systems [Kabbaj, 2006]. CG structures and operations in Amine can be called using Java APIs. The Amine platform provides rule-based functions expressed in the PROLOG+CG language which is an extension of PROLOG using the Java object oriented language and Conceptual Graphs.



*Figure 3.30: Our agent's reasoning model*

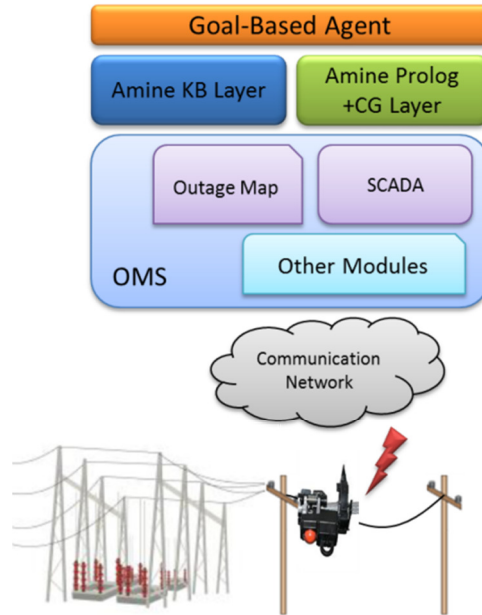
### 3.5.3 System Architecture

The global system architecture that we propose is depicted in Figure 3.31 where outage sensors are deployed throughout the distribution network. They are mounted on power cables and report events by sending a message using an industrial communication protocol such as DNP (Distributed Network Protocol) [IEEE, 2010] or Modbus [Cena et al, 2014] to a centralized SCADA using a wireless cell communication (GSM or CDMA). The retrieved messages are located using a GIS which is also used to model the distribution network. The Amine knowledge base (KB layer) is used to store the pattern specifications and the detected pattern instances. Amine's Prolog +CG layer is used by agents to identify relevant patterns and to update the environment with action results.

### 3.5.4 Pattern Specifications

We propose two types of patterns: simple patterns and complex patterns. Simple patterns are used to represent situations where only one event occurs. Complex patterns are used for more complex situations where there are temporal and spatial relations holding between situations. A simple pattern is expressed using Amine's linear CG notation and is illustrated in Figure 3.32. It represents the following situation:

*“An event type *OutageEvent* occurs near an area (here defined as *Outage Area*), the resulting state of the cable is *Broken* and the crew status is *free*”.*



*Figure 3.31: A system architecture involving the outage management system and the interaction with the goal-based agent*

```
[Normal_Outage_Pattern]-
-Pattern_Description->[Proposition : [[OutageEvent]-
    -Before->[[CableState]-status->[Broken]]]
    -near->[Outage_Area]]
-Context_Description->[Proposition : [ElectricCrew]-attr->[CrewStatus="Free"]]
```

*Figure 3.32: An example of simple pattern with linear notation*

Thanks to temporal relations between events and states, it becomes possible to represent the change that occurred on a specific cable after the occurrence of the outage event. Such a feature is very important in this kind of systems to enhance the users' decision making capabilities. In some cases, an outage can be reported but it

may not necessarily be related to a cable break. Hence, a user may need to investigate other elements of the network. When an agent detects an instance of such a pattern, it performs two actions: first, it assigns a crew to the corresponding area to fix the problem. Second, it updates the outage map using the outage information. Both actions can be achieved using Prolog+CG programs:

1. *Assign\_Free\_Crew:-SimplePattern\_With\_ContextInfo*  
*(\_Simple\_Pattern,Environment)* which assigns the crew to the outage event,
2. *UpdateOutageMap(\_Simple\_Pattern,Environment)* which updates the outage map with the information on the related event information.

Figure 3.34 shows an example of the Prolog+CG console output when the action AssignFreeCrew is performed. Note that the agent can change the crew status and retrieve it using CG rules as illustrated by Figure 3.33. Therefore, the next time an outage event occurs, the agent's contextual information will indicate that the crew belonging to this specific area is already assigned and that there is a need to find another alternative. Additional details about the agent's reasoning model and the environment configuration in PROLOG+CG is given in the Appendix section.

```
If Normal_Break Pattern and [Crew ="Team2"]-attr->[CrewStatus = "Free"]
Then
[Crew ="Team2"]-attr->[CrewStatus = "Assigned"]
```

*Figure 3.33: An example of rule used for crew assignment*

In some cases a user may be interested in situations which involve more than one event. For example, the complex pattern illustrated in Figure 3.35 represents the following situation:

*“An event type OutageEvent occurs near an area (here defined as \_Area). Before that, a communication error event occurred on a sensor in the same area”.*



```

Prolog+CG Console - Untitled
Console Font Consult Debugger
*****
PROLOG+CG, Version 7.0
(c) 2004-2010, Dr. Adil KABBAJ
INSEA, MOROCCO
Amine Group
*****
loadOntology("C:\DevPhd\knowledgeBase\OMSKB.xml").
yes
?-
?- AssignFreeCrew.
Normal_Break pattern detected
Crew AssignedIn Region SainteFoy
New Team2 status is Assigned
yes
?-

```

Figure 3.34: Output of the action *AssignFreeCrew* in the *Prolog+CG* console.

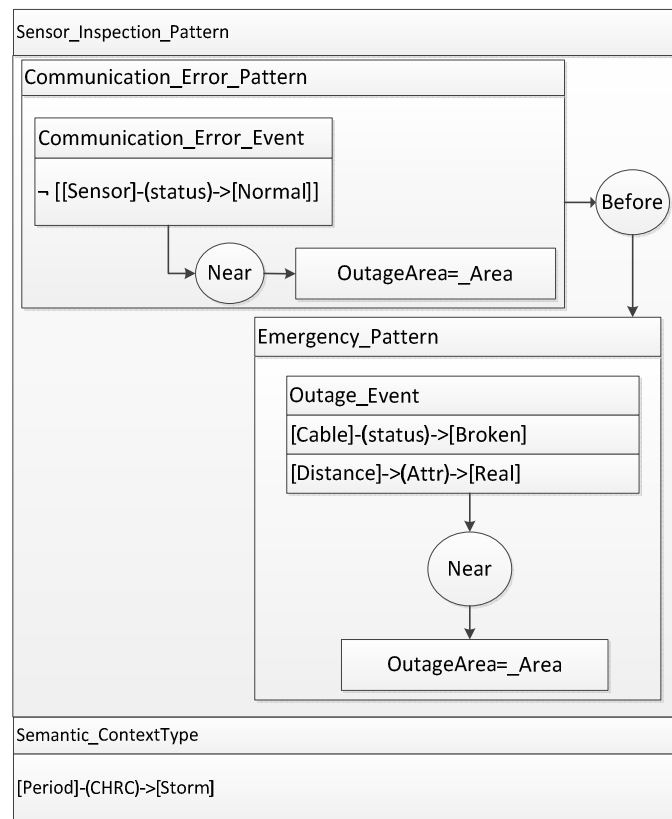


Figure 3.35: A complex pattern representation.

This is a typical use case in monitoring systems. A sensor can be out of service and the system detects an outage event in the same area, but reported by another sensor. Such a situation can occur during a storm where sensors are in service but some communication links have failed. Hence, the storm can be used as contextual information in the pattern definition. When such a pattern is detected, the crew should repair the outage and inspect the sensor and fix the issue. Note the presence of the negation operator in the structure of the communication error event which means that the non-occurrence of an event can be considered as an event by a user.

### **3.6 Discussion**

This case study shows how spatiotemporal situations play a key role in an OMS where software agents need to detect patterns using information collected from different data sources. An agent can also use its own contextual information to make decisions giving spatiotemporal situations, resulting in different interpretations from one agent to another. Using the CG formalism enhances agents' reasoning capabilities about spatiotemporal patterns. We proposed a cognitive approach which integrates contextual information to better interpret spatiotemporal patterns. Although agent's reasoning capabilities are enhanced using our spatiotemporal formalism, the proposed approach is far from offering a complete solution to manage patterns. There are several issues that still need to be addressed.

First, the proposed approach uses qualitative spatial and temporal relations. Mapping temporal relations from quantitative data to qualitative quantifier is supported thanks to Moulin's model [Moulin, 1997]. However, our formalism does not allow for a smooth mapping from quantitative spatial data to qualitative spatial relations. In the next chapter, we address this issue by proposing an intelligent spatial proximity tool to qualify spatial proximity relations using a neurofuzzy classifier and contextual information. This particular attention to proximity relations is motivated by the application domains used in this thesis where proximity is very often used in pattern definitions.

Second, we assumed in our case study that pattern instances are stored in a specific knowledge structure. This is not the case in practice where events, states and spatiotemporal situations in general can be represented in a heterogeneous way, with different semantics and structures and can be generated by distributed data sources. Although Conceptual Graphs provide an approach that helps matching pattern instances to pattern definitions, they remain limited to manage and to process a large dataset. For example, if  $n$  events are generated by  $i$  different data sources (where  $n$  and  $i \in \mathbb{N}$ ) how does Amine platform handle these events? Which events will be selected to detect a specific pattern instance? Which knowledge structure will be used to carry these events? What will happen to the selected events?

Our formalism for spatiotemporal situations is based on states and events but it does not integrate the concept of process to represent more complex situations. This may be useful in some specific domains such as weather monitoring (for example, a thunderstorm can be represented as a process) and multi-agent geo-simulation. The integration of the notion of process in our pattern definition may raise some challenges related to pattern detection which could not be addressed by existing technologies. For this reason, we propose to address this issue in future works.

Finally, most of Conceptual Graph tools and in particular the Amine platform, do not offer temporal and spatial operators. Furthermore, these CG tools cannot be easily integrated in existing monitoring applications such as SCADA, OMS and Sensors Web systems. The issue of integrating our pattern model in existing monitoring systems is quite challenging and will be addressed in Chapter 5 of this thesis.

It is worth mentioning that the main content of this chapter has been published in the *International Conference of Conceptual Structures* held in Iasi, Romania in 2014 [Barouni and Moulin, 2014a]. Using Conceptual Graphs in the context of Big Data applications is a new topic in the Conceptual Structure research community and may give the opportunity to explore several interesting research aspects in the future.

### 3.7 Conclusion

In this chapter, we proposed our model for spatiotemporal pattern representation and reasoning. The proposed formalism uses a qualitative approach to represent patterns instead of using query languages such as most pattern definitions currently proposed in the literature. Moreover, it allows for representing dynamic spatiotemporal phenomena thanks to the extension of Haddad's model [Haddad, 2009]. Another original aspect of our representation formalism is that it integrates contextual information. Using contextual information allows users and software agents to have their own interpretation of situations of interest according to their mental models of the environment. Other works in the literature as reviewed in Chapter 2 do not use context to enrich patterns definitions and to offer different interpretation of situations of interest. Finally, we proposed a qualitative representation model extending the Conceptual Graph formalism and we implemented some examples using the Amine platform. Using Conceptual Graphs allows making the representation of qualitative spatiotemporal situations of interest close to Natural Language and hence allows users to more easily understand and represent spatiotemporal patterns.

Moreover, some challenges related to qualitative spatial relations and pattern detection should be addressed to facilitate the integration of our pattern model to real applications in the industry. These issues will be addressed in the next chapters of the thesis.

# **Chapter 4**

## **Using a Neuro-Fuzzy Classifier to Automatically Generate Spatial Proximity Quantifiers**

### **Introduction**

In Chapter 3, we proposed a spatiotemporal pattern formalism using Conceptual Graphs. A spatiotemporal pattern involves spatial and temporal relations between its various components. These relations are expressed in a qualitative manner. In particular, we give a special attention to spatial proximity relations which are widely used in real-time monitoring and data acquisition systems for decision making purposes. Figure 4.1 illustrates an example where the qualitative spatial relation “near” qualifies the distance between an outage event and a distribution cabinet.

The notion of proximity is one of the fundamental concepts in daily human cognition studied by researchers over the last decades. Several authors proposed tools to reason about proximity and solutions which can be automated and integrated in a GIS. The goal is to reduce the semantic gap between quantitative data in GIS (metric distance) and qualitative data (proximity) as used by humans [Cohn and Renz, 2007]. Such works used advanced qualitative techniques (such as fuzzy sets and fuzzy logic) as well as conceptual notions such as *influence and impact areas*. Other empirical works have also been conducted. However, spatial distance is not the only factor that influences human reasoning about spatial proximity. Actually, proximity relations have two characteristics: they are context *dependent and uncertain*. For example, the means of transportation used to travel from Paris to London may change the traveler's perception of distance (context-dependence). When a person parks a car, she does not need to know the exact distance of the empty space between two cars (uncertainty). A suitable model of spatial proximity should consider both characteristics in order to be closer to the human apprehension of proximity.

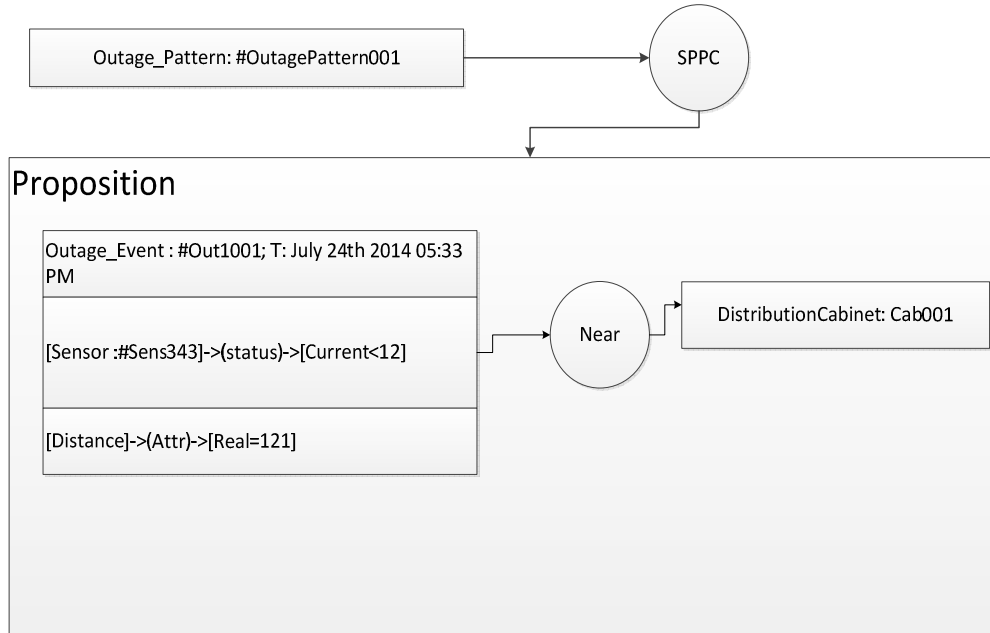


Figure 4.1: A simple pattern example using a qualitative spatial relation

In this chapter, we propose a novel approach to represent and reason about spatial proximity. The approach is based on contextual information and uses a neuro-fuzzy

classifier to handle the uncertainty aspect of spatial proximity. Neuro-fuzzy systems [Negnevitsky, 2011] are a combination of neural networks and fuzzy systems and incorporate the advantages of both techniques. While Fuzzy systems are focused on knowledge representation, they do not allow for the estimation of membership functions. Conversely, neuronal networks use powerful learning techniques, but they are not able to explain how results are obtained [Gliwa and Byrski, 2013]. Neuro-fuzzy systems benefit from both techniques by using training data to generate membership functions and by using fuzzy rules to represent expert knowledge [Borkar et al, 2013]. Moreover, contextual information is collected from a knowledge structure. The complete solution that we propose is integrated in a GIS, enhancing it with proximity reasoning capabilities.

This chapter is organized as follows. Section 4.1 gives an overview of some research works on qualitative spatial proximity and outlines their limitations. Section 4.2 presents an overview of the neuro-fuzzy classifier used in our approach. Section 4.3 presents the experiments that we carried out to validate our approach and the results. Section 4.4 presents the architecture of our qualitative proximity tool. Section 4.5 concludes this chapter and discusses some research outlooks.

## **4.1 Qualitative Spatial Proximity**

Spatial proximity reasoning is a research area which has been addressed by the qualitative spatial reasoning community, adopting different perspectives such as geography, cognitive science and linguistics [Yao and Thill, 2007]. A large number of prior works used Fuzzy Logic and qualitative techniques to deal with spatial proximity because it has inherent fuzziness [Robinson, 1990]. While reasoning with proximity, human beings may also consider metric distances and other parameters referred to in the literature as contextual information. In the following sub-sections, we present an overview of some works which used uncertainty techniques and contextual information or a combination of them.

### 4.1.1 Distance-Based Approaches

[Guesgen, 2002] used fuzzy sets and associated each set with a qualitative spatial relation. The idea behind Guesgen’s approach is to interpret qualitative proximity relations between spatial objects as restrictions of spatial linguistic variables such as *near* and *far*. Each linguistic variable is associated with a fuzzy set. The proximity relation is therefore represented by a membership degree of each of these fuzzy sets using a membership function. For example, the expression “*the object A is near the object B*” may be interpreted by a *near membership value* which is associated with each *near* relation between object A and surrounding objects such as B, C and D. Generally, the nearest an object is to A, the highest its membership value is (Figure 4.2).

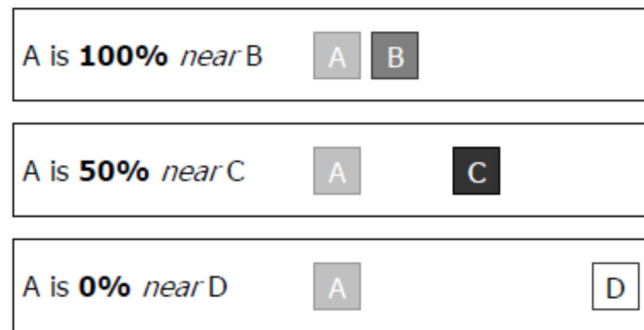


Figure 4.2: An example of possible proximity relations between two objects [Schultz et al, 2007]

[Schultz et al, 2007] implemented Guesgen’s formalism using the Euclidian distance between two objects. A Java-based software helps a user to define a “nearness” factor which will be used to specify other proximity relations. Then, the user can query spatial objects using proximity relations. Figure 4.3 shows a query example searching for spatial objects of type Cables (blue lines) moderately near spatial objects of type Buildings (transparent red boxes).

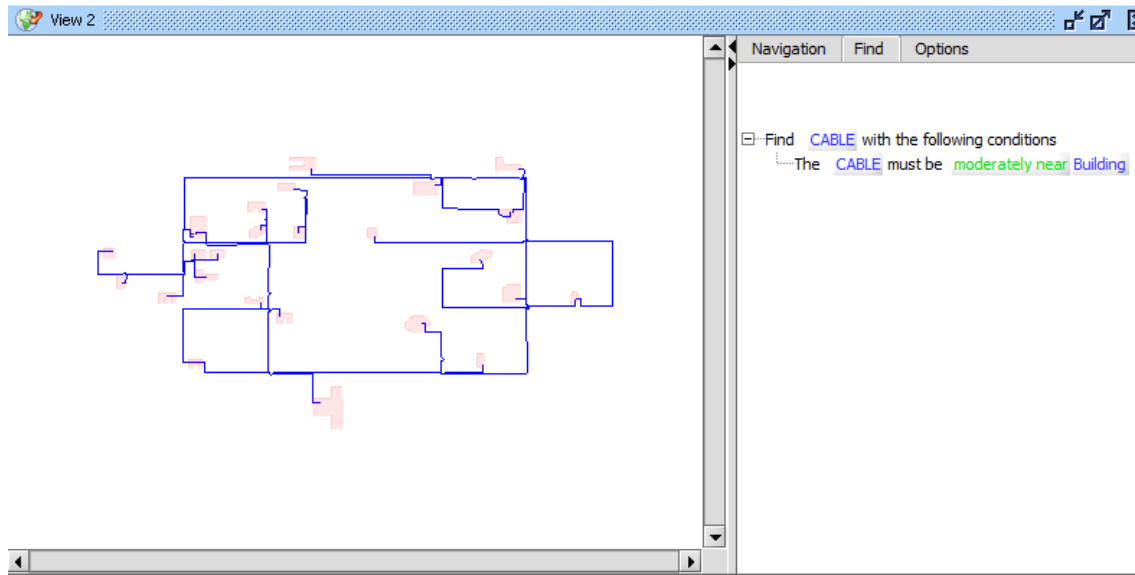


[Brennan and Martin, 2006] stated that most fuzzy-based proximity formalisms proposed in the literature suffer from an important shortcoming: membership functions are not clearly defined. To overcome this limitation, they used Gahegan's approach [Gahegan, 1995] who proposed a method to identify spatial proximity using three factors: the absolute distance, the relative distance between two spatial objects and the combination of both. An absolute distance may be a spatial relation such as *very close*, *close* and *far*. A relative distance may be a spatial relation such as *closest* or *farthest*. The combination of both absolute and relative distances was defined by Gahegan to reason about spatial relations using a fuzzy union operator.

Since Gahegan did not use experimental data to validate his approach, [Brennan and Martin, 2006] proposed an approach to evaluate membership functions and showed how they can be combined using fuzzy logic operators. To this end, they used the absolute distance membership function  $\mu_{abs}(A, B)$  proposed by Gahegan. This function is presented in Table 4-1 where  $A$  and  $B$  are spatial objects,  $Dist(A, B)$  is the absolute distance between  $A$  and  $B$ .  $Max$  is the maximum distance between all the places in the data set and it is used to normalize the value of  $Dist(A, B)$ . The authors used the function  $\mu_{rel}(A, B)$  proposed in [Worboys, 1996] to compute the relative distance membership function, where  $reldis(A, B)$  is the relative distance between  $A$  and  $B$  which is calculated using the distance between  $A$  and  $B$ ; and divided by the mean of the distances between  $A$  and every object in the data set (see Table 4-1).

Absolute Distance Metrics	$\mu_{abs}(A, B) = 1 - \frac{Dist(A, B)}{Max}$
Relative Distance Metrics	$\mu_{rel}(A, B) = \frac{1}{(reldis(A, B) + 1)}$
Fuzzy Union	$\mu_{comb\_u}(A, B) = MAX(\mu_{abs}(A, B), \mu_{rel}(A, B))$
Fuzzy Intersection	$\mu_{comb\_i}(A, B) = MIN(\mu_{abs}(A, B), \mu_{rel}(A, B))$

*Table 4-1: Fuzzy distance as proposed by [Brennan and Martin, 2006]*



*Figure 4.3: A Java-based proximity platform developed by [Schultz et al, 2007], where fuzzy quantifiers are used to query spatial objects*

The results of the experiments conducted by [Brennan and Martin, 2006] demonstrated that the absolute distance and relative distance membership functions can be used separately and generate linear distributions. However, using the *union* operator to combine these metrics gives clustered distributions and may not be relevant for proximity reasoning whereas fuzzy intersection gives better results. The authors proposed to use the fuzzy union and they implemented their approach in a GIS. However, they neither expressed the meaning of fuzzy intersection in terms of spatial proximity, nor justified the use of fuzzy logic since fuzzy rules were not used. Moreover, the way such an approach can be applied to qualitative spatial reasoning was not clearly discussed.

In [Worboys et al, 2004] the authors addressed the issue of possible relationships between the geometric notions of proximity and direction. They discussed human conceptual models of an environmental space, and possible combinations of proximity and directional relations. They also addressed the issue of granularity of vague space relations. To this end, Worboys and colleagues conducted an empirical

study involving a number of observations of people having the same background and belonging to the same spatial domain (students on campus). Despite the small size of the dataset used in these experiments, some interesting findings were drawn. First, there exists a context-dependent relationship between the cognitive aspect of nearness (nearness as perceived by humans) and geometric distances. Second, combining direction (i.e. leftness) and proximity (i.e. nearness) enhanced the efficiency of qualitative spatial reasoning.

#### 4.1.2 Context-Based Approaches

Other works on spatial proximity started from the observation that proximity is context-dependent. For example, [Brennan and Martin, 2012] proposed a conceptual framework to qualitatively represent spatial proximity and to enhance the capacity of spatial reasoning systems using contextual information. They considered contextual information as a key element in any model of spatial proximity. For example, a degree of proximity to an object may vary if the object is meant to be seen or reached. To reason about spatial proximity, Brennan and Martin introduced the notion of *impact area* which is a generalization of the influence area concept introduced by [Kettani and Moulin, 1999]. An *influence area* is a portion of space surrounding an object: it has an interior and exterior border such that the borders of the influence area and the border of the object have the same shape [Kettani and Moulin, 1999]. Euclidean geometry has been used by Kettani and Moulin to compute the width of an influence area. Brennan and Martin proposed a more generic approach motivated by the fact that spatial proximity is more than a metric measure. Proximity is rather context dependent. Furthermore, other spatial relations such as topological and directional relations have some unified views within the research community. Hence, they introduced the impact area concept which involves contextual information to qualify spatial proximity. The impact area of an object takes into account both the nature of the object and its surrounding environment. Some examples in [Brennan and Martin, 2012] demonstrate how an impact area is more generic than an influence area, and how this notion uses contextual information in proximity analysis. As described in Figure 4.4, contextual information is defined as any “information

collated by an expert who is expected to incorporate all relevant factors into the impact area”. Figure 4.5 illustrates the difference between an influence area and an impact area for two couples of objects. Objects *A1* and *B1* are water tanks and objects *A2* and *B2* are radio towers. The distance between *A1* and *B1* is equal to the distance between *A2* and *B2*. The influence area of water tank is equal to the influence area of radio towers because all objects have the same shape and size and because they are located at the same distance from each another. If we consider the functionality of the towers (range of frequency) and the surrounding Cliff, the impact areas will be different.

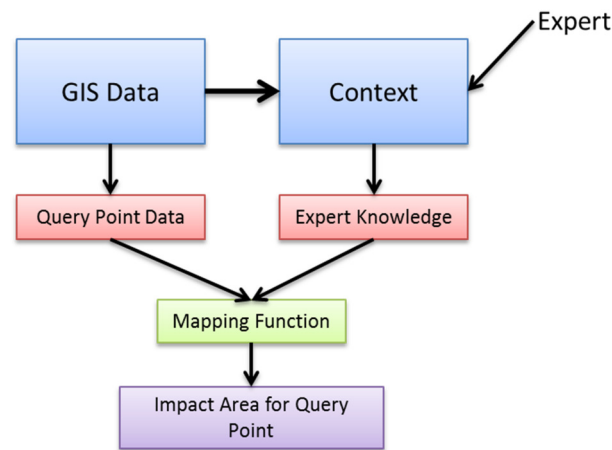


Figure 4.4: A conceptual framework proposed by Brennan and Martin to compute impact area from contextual information and geographic distance.

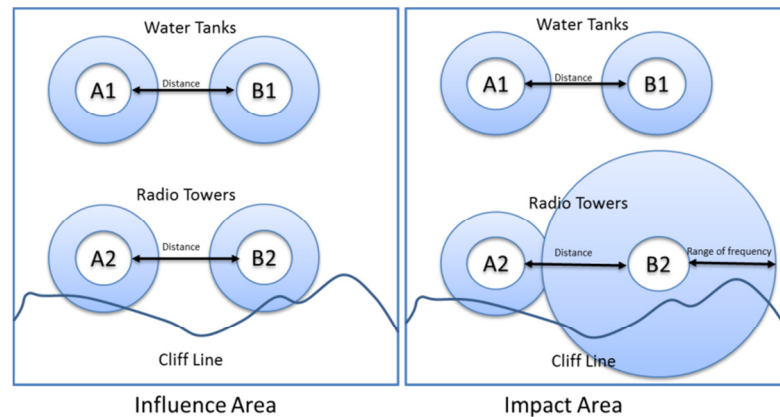
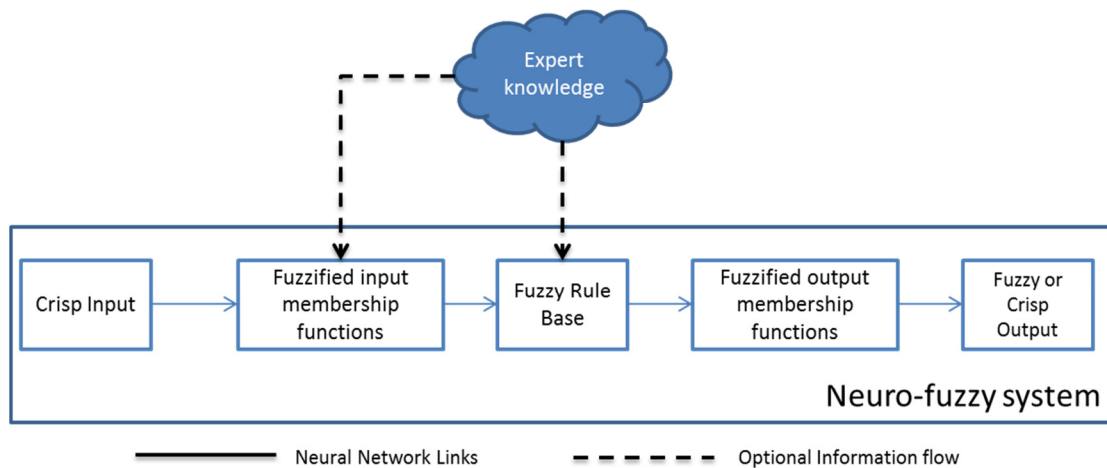


Figure 4.5: An example of difference between influence areas [Kettani and Moulin, 1999] and impact area [Brennan and Martin, 2012]

According to [Yao and Thill, 2007] proximity relations have two main characteristics. The first one is that a proximity relation is context dependent. These authors classified context factors as ‘objective’ and ‘subjective’. *Subjective context factors* may have different values according to the involved person. Examples include the navigator’s familiarity with the area, his time and his budget. *Objective context factors* have values which do not depend on the person who perceives the distance like the type of activity (run, walk), object reachability and transportation mode, to name a few. The second characteristic of proximity is the *uncertainty of distance measures*. If a person wants to park her car, she does not need to know the exact distance between her car and the other cars around. The authors proposed a review of existing approaches based on fuzzy logic to handle the uncertainty aspect of proximity measures and they noticed that most of these works preset the form of membership functions. To overcome this limitation, Yao and Thill proposed a novel approach based on neuro-fuzzy techniques which allows for reasoning with spatial proximity by considering contextual information and by handling its uncertainty aspect. Neuro-fuzzy systems, take advantage of both techniques by using training data instead of preset membership functions and by using fuzzy rules to represent expert knowledge. The general architecture of the neuro-fuzzy system for proximity modeling proposed by Yao and Thill is depicted in Figure 4.6. The contextual factors used in Yao and Thill’s approach (objective and subjective) take the form of crisp inputs. They are fuzzyfied using membership functions which can be preset by an expert. Fuzzyfied outputs are generated as consequences of applying all fuzzy rules. The final output (either fuzzy or crisp) is calculated through defuzzification using the weighted average of fuzzyfied outputs. Yao and Thill used ANFIS (Adaptive NeuroFuzzy Inference System) to implement and validate their approach. ANFIS is a Takagi-Sugeno fuzzy inference system [Nauck and Kruse, 1999]. It uses a back propagation algorithm to train the fuzzy neural network. This algorithm computes the error between the training data and the neural network output and uses the error to adjust the rules’ weights. The experimental results demonstrated that a neuro-fuzzy approach gives higher prediction accuracy when training data and testing data are compared. Finally, the proposed approach allowed for overcoming the problem of

using preset membership functions reported by the authors and by [Brennan and Martin, 2006].



*Figure 4.6: A general architecture of the neuro-fuzzy system for proximity modeling proposed by [Yao and Thill, 2007]*

### 4.1.3 Discussion

Several metric-based approaches used advanced qualitative techniques to represent and reason about spatial proximity. For example, using fuzzy sets allows for a smooth mapping from quantitative to qualitative distances using fuzzy quantifiers close to natural language, thanks to the use of so-called linguistic variables [Zadeh, 1965]. Later, it has been shown that metric distance is not necessarily the only factor that influences the human cognitive apprehension of spatial proximity.

A suitable model of spatial proximity should acknowledge other factors such as contextual information as claimed by [Brennan and Martin, 2012]. However, their work suffers from two main shortcomings: first it only allows for reasoning about nearness. There is no clear definition of other proximity relations such as “*far*” and “*close to*”. Therefore, it is difficult to relate their definition of impact area with different proximity relations used by humans and by GIS solutions. Second, the definition of impact area seems to be particularly domain-specific. Examples

provided by the authors do not clearly explain how and from where contextual information is obtained and how contextual information can influence the spatial proximity. Therefore, implementing this work in a generic GIS is quite challenging. [Yao and Thill, 2007] proposed an innovative solution to reason about proximity by using contextual information to handle the cognitive aspect and by using neuro-fuzzy techniques to handle the qualitative aspect of spatial proximity. Although Yao and Thill's approach is quite suitable for qualitative spatial reasoning, it suffers from several drawbacks. The neuro-fuzzy system used for the implementation of their approach for their experiments is ANFIS which is an approximation system [Nauck and Kruse, 1999]. ANFIS' output is a crisp value, which does not help in "classifying" the spatial proximity. Usually, given a number of context factors, a user/agent tries to answer the following question: what is the proximity relation between object *A* and object *B*? Possible answers are *very near*, *near*, *far* or *very far*. *We believe that qualitative proximity is a classification problem rather than an approximation problem.* Furthermore, Yao and Thill proposed a general architecture to implement their solution and conducted experiments to prove its relevance. However, their solution was not integrated in a GIS.

In the next sections, we propose a new framework to reason about qualitative proximity with the following features: first, a neuro-fuzzy classifier (NFC) is used instead of ANFIS in order to handle the uncertainty aspect of proximity. Second, we integrate the proposed solution in a GIS to enhance its qualitative proximity reasoning capabilities. Figure 4.7 illustrates the general architecture of our approach. A user specifies contextual information and fuzzy rules to generate training data. This dataset is used by the NFC to train the fuzzy inference system and to generate qualitative proximity quantifiers that will be used in the qualitative proximity tool. An overview of the NFC structure used in our approach is presented in the following section.

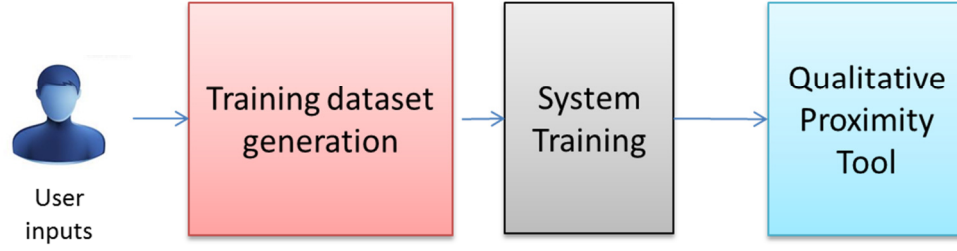


Figure 4.7: An overview of the proposed approach

## 4.2 The Neurofuzzy Classifier Structure

The structure of a neuro-fuzzy system is similar to a multilayer neural network. In general, a neuro-fuzzy system has one input layer, one output layer, and three hidden layers [Negnevitsky, 2011]. In neuro-fuzzy classification systems, the feature space is partitioned into multiple fuzzy subspaces which are managed by fuzzy rules. Rules are represented by a network structure and their parameters (weights) are optimized using learning techniques.

A fuzzy classification rule  $R_i$ ; establishes the relation between the input feature spaces and classes (output). It is defined as follows:

$$R_i: \text{if } x_{c1} \text{ is } S_{i1} \text{ and } \dots \dots x_{cj} \text{ is } S_{ij} \text{ and } x_{cn} \text{ is } S_{in} \text{ then output class is } C_k$$

Where  $x_{cj}$  is the  $j^{\text{th}}$  input variable of the  $c^{\text{th}}$  sample;  $S_{ij}$  denotes the fuzzy set of the  $j^{\text{th}}$  feature in the  $i^{\text{th}}$  rule; and  $C_k$  represents the  $k^{\text{th}}$  label of class.  $S_{ij}$  is associated with a suitable membership function [Sun and Jang 1993]. Using fuzzy rules allows for splitting the feature space into multiple fuzzy subspaces. These rules can be represented by a neural network. Figure 4.8 depicts an example of a space partition of two inputs  $x_1$  and  $x_2$ . Each input (feature) has three fuzzy sets described by linguistic variables. Hence, for this example there are nine fuzzy rules overall.



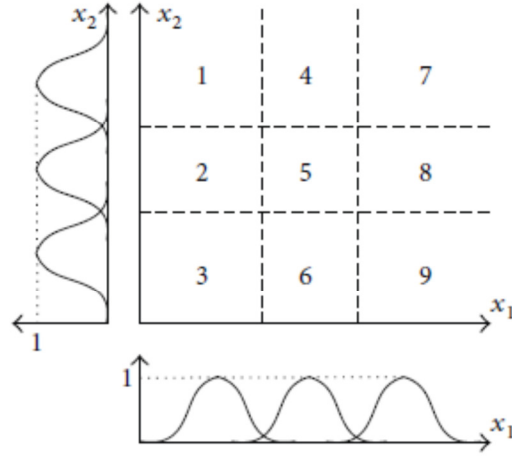


Figure 4.8: Partition of the feature space [Sun and Jang, 1993]

A neuro-fuzzy classifier is a multilayer network with six layers. The first layer is the input layer (also called “features”). The last layer is the output layer (also called “classes”). The other layers are defined as follows.

**Membership layer:** each input is identified using fuzzy sets and each fuzzy set is associated with a linguistic variable. Fuzzy sets are represented by membership functions. According to [Cetişli and Barkana, 2010] bell-shaped functions are the most used functions in neuro-fuzzy classifiers since these functions have fewer parameters and smoother partial derivatives. Such a function is used for this layer and is defined as follows:

$$\mu_{ij}(x_{cj}) = \exp\left(-\frac{(x_{cj} - c_{ij})^2}{2\sigma_{ij}^2}\right) \quad [\text{Cetişli and Barkana, 2010}]$$

Where  $\mu_{ij}(x_{cj})$  is the membership grade of  $i^{\text{th}}$  rule and  $j^{\text{th}}$  feature;  $x_{cj}$  represents the  $c^{\text{th}}$  sample and  $j^{\text{th}}$  feature;  $c_{ij}$  and  $\sigma_{ij}$  are the center and the width of bell shaped function, respectively. The membership functions of input variables  $x_1$  and  $x_2$  of Figure 4.8 are examples of bell shaped functions.

**Fuzzification layer:** Each node in this layer is a fuzzy rule. The antecedent of the fuzzy rule is a fuzzy set. The output is a singleton membership function. The fuzzy rule premises become weights for the rule neurons of this layer [Gliwa and Birsky, 2013]. The conclusion of a rule is a connection from the rule neuron to the next layer. Each node in this layer has an activation function which corresponds to the degree of fulfillment of the fuzzy rule for the  $x_{cj}$  sample. The activation function  $a_{ic}$  of a fuzzy rule is defined as follows:

$$a_{ic} = \prod_{j=1}^n \mu_{ij}(x_{cj}) \text{ [Gliwa and Birsky, 2013]}$$

Where  $n$  is the total number of features.

**Defuzzification layer:** In this layer, according to its weight each rule affects each class. The more a rule influences a class, the bigger is the weight between that rule output and the specific class. Otherwise, the class's weights are small. The weighted output for a given sample  $x$  that belongs to a class  $k$  is computed as follows:

$$\beta_{ck} = \sum_{i=1}^Z a_{ic} w_{ik} \text{ [Do and Chen, 2013]}$$

Where  $w_{ik}$  denotes the degree of belonging to the  $k^{\text{th}}$  class that is controlled by the  $i^{\text{th}}$  rule and  $Z$  represents the number of rules [Do and Chen, 2013].

**Normalization layer:** Depending on the defuzzification method, the weighted sum may generate a value greater than 1. Therefore, this sum should be normalized using the following formula:

$$N_{ck} = \frac{\beta_{ck}}{\sum_{h=1}^K \beta_{ch}} \text{ [Do and Chen, 2013]}$$

Where  $N_{ck}$  is the normalized value of the  $c^{\text{th}}$  sample that belongs to the  $k^{\text{th}}$  class and  $K$  is the number of classes.

Finally, the output class  $C_c$  is the maximum of normalized values given by the normalization layer:

$$C_c = \max_{k=1,2,\dots,K} \{N_{ck}\} \text{ [Do and Chen, 2013]}$$

Figure 4.9 depicts a neurofuzzy classifier network and the different layers mentioned above. This network has two input features namely  $X_1$  and  $X_2$  and three output classes namely  $C_1$ ,  $C_2$  and  $C_3$ .

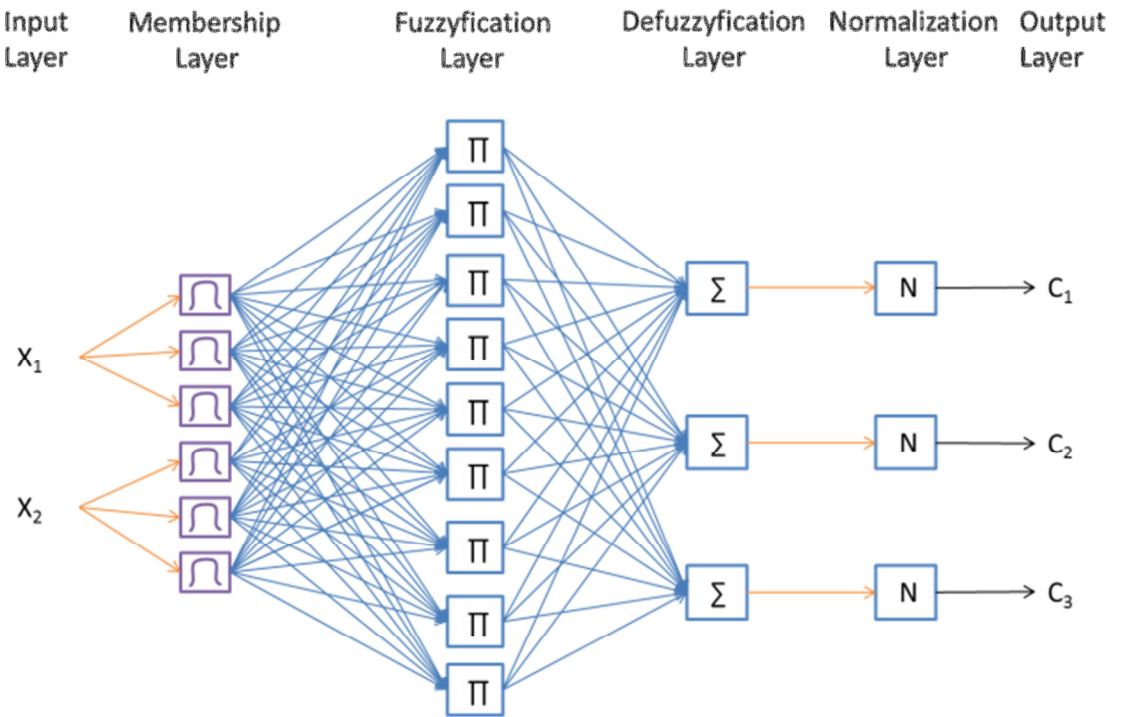


Figure 4.9: A neurofuzzy classifier, adapted from [Cetiřli and Barkana, 2010]

## 4.3 Implementation Details

After briefly presenting the structure of a neuro-fuzzy classifier in the previous section, we present our approach to build a qualitative spatial proximity representation and the associated reasoning tool. The creation of this tool was achieved in several steps that are detailed in the next subsections.

### 4.3.1 Training the Neurofuzzy Classifier

One of the advantages of using a neuro-fuzzy classifier is that membership functions and rules can be learned from data sets. If  $S_{Z \times n}$  and  $C_{Z \times n}$  are the sigma and the center values of the bell membership function; and  $W_{Z \times K}$  is the weight matrix of connections from the fuzzification layer to the defuzzification layer; then  $O = \{S_{Z \times n}, C_{Z \times n}, W_{Z \times K}\}$  is the set of parameters that will be optimized by the learning algorithm.

Several training algorithms have been proposed for neuro-fuzzy classifiers [Do and Chen, 2013]. However, the *scale conjugate gradient* (SCG) is one of the most efficient algorithms with less errors and high efficiency. This algorithm was enhanced and implemented in a software package by [Cetiřli and Barkana, 2010]. We use this algorithm to train our data set. A detailed overview of the training algorithm can be found in [Cetiřli and Barkana, 2010].

We adapted a Matlab implementation of the NFC which has been developed by [Cetiřli and Barkana, 2010]. This program first trains the NFC; then it generates the fuzzy inference system parameters that are used by the qualitative proximity reasoning engine. In this section, we present how data sets were prepared to train the NFC and we present the obtained results.

We use Yao and Thill's definition of contextual information to select the NFC inputs. The *contextual information is objective* when variables are selected independently of the person who perceives the distance. The *contextual information is subjective* when variables are dependent on the person who perceives the distance. We use Euclidian distance as the objective contextual information. The transportation mean is one of the most popular contextual information used by humans to qualify proximity. Since a dataset was generated from user's experience about Quebec City, we realized that cars are the main transportation mean in this city. Therefore, we decided to use another subjective contextual information instead. In our approach we used road traffic and user's familiarity with the area as subjective contextual information. Four classes are used for the proximity output (close, medium, far and very far). It is worth mentioning that Euclidian distance is used for simplification purposes. For other experiments, a distance between two geographic features can be computed using a path planning software or using various GIS functions available either in web-based or desktop applications. Table 4-2 and Table 4-3 detail the chosen NFC inputs and outputs.

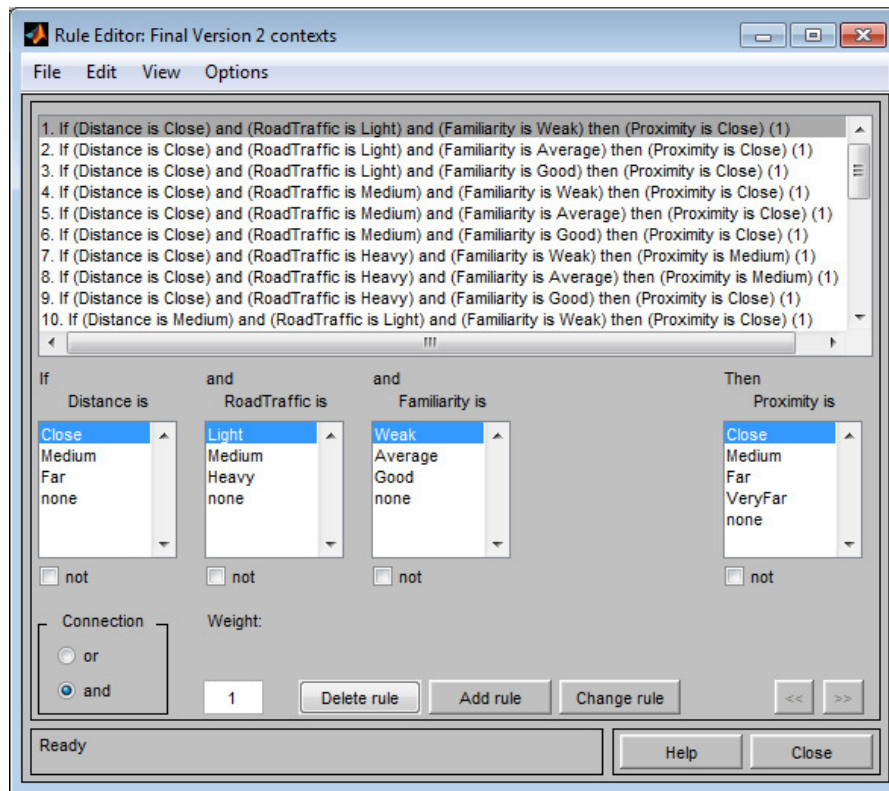
Variable name	Fuzzy sets	Range
Euclidian distance	Close, medium, far	0-500 KM
Traffic	Light, medium, heavy	0-1
User familiarity with the site	Week, average, good	0-1

*Table 4-2: NFC features (inputs)*

Classes	Value
Close	1
Medium	2
Far	3
Very Far	4

*Table 4-3: NFC Classes (outputs)*

To train the NFC, the data set shall be separated into two sub-sets. The first set is used to train the NFC and the second one is used to test the generated NFC parameters. We conducted our experiments with the help of a surveyed population based in Quebec City. Hence, a data set with 300 samples has been prepared using locations in Quebec City. 150 samples are used to train the NFC and 150 samples are used to test the trained NFC. To generate the different classes (outputs) in the sample data set, the user defined a fuzzy inference system (FIS) with 13 rules taking advantage of his experience and familiarity with the spatial environment. An example of these rules is displayed in Figure 4.10.



*Figure 4.10: Fuzzy rules used to prepare the data set*

The data set has been created using a specific area which is delimited using a square. All distances are computed between geographic objects that belong to this particular area. Therefore, the training results will be valid only for this specific area. If the selected area changes, the NFC training must be repeated for the new area. Each

feature is associated with three fuzzy sets which are represented by a bell shaped membership function. The FIS uses a Sugeno-type system [Nauck and Kruse, 1999] to calculate the weighted sum of the fuzzy rules output for each sample vector. The FIS gives a crisp output with a float number which is not suitable to make a decision about which class the proximity belongs to. Therefore, we apply a “ceil function<sup>4</sup>” to extract from the float number the related class. An algorithm to automate data generation is described by the pseudo-code in Table 4-4.

---

**Algorithm 1:** NFC dataset preparation

---

Generate Data Set (V, O)

**Variables:**

V: vector of inputs

O: output of the FIS

1. Get all the geographic objects inside the training area.
  2. Compute the Euclidian distance between the objects.
  3. For each distance between two geographic objects,
  4.           Assign two contextual information
  5.           Save in the Sample Data Set
  6. End For
  7. Load Sample Data
  8. For each vector feature from the  $V = \begin{bmatrix} x_{c1} \\ x_{c2} \\ x_{c3} \end{bmatrix}$  in the data set,
  9.           Calculate the output of the fuzzy rules.
  10.          Calculate the FIS output of the specific vector V
  11.          The output class = Ceil the FIS output.
  12. End For
- 

*Table 4-4: NFC data set training preparation*

---

<sup>4</sup> Ceil is a function defined in Matlab which is used to round an element to the nearest integer greater than or equal to that element

### 4.3.2 Results

Now that our data set is ready, we use it to train the NFC and to collect the outputs. The root mean square error (RMSE) is used to evaluate the error between the NFC output and the testing output. It converges after 55 epochs<sup>5</sup> and remains stable at 0.229963 (Figure 4.11). This means that 77% of the testing inputs have been successfully classified by the NFC. Note that this score is presented here for illustration purposes and that it varies from one data set to another. In other experiments, higher scores were obtained (around 90%). A user may use other data sets and run additional iterations until she reaches a satisfying score. The classifier generated a new set of features with new fuzzy sets and new membership functions. For example, the distance feature in the data set used to train the algorithm had three fuzzy sets: short, medium and far. But, the new distance feature has four fuzzy sets named as follows: *close*, *average*, *far* and *very far*. Indeed, these names have been chosen by an expert who intuitively associates the values of linguistic variables with different features proposed by the NFC.

The same logic applies to other features (road traffic and user's familiarity with the region). Figures 4.12, 4.13, 4.14, 4.15, 4.16 and 4.17 illustrate the fuzzy sets of each input feature as proposed by the user to generate the data set and the fuzzy sets of each input feature as classified by the NFC. These outputs are only valid for the geographic area in which the data set has been defined. If the area changes, these membership functions are no longer valid and the system should be trained again.

---

<sup>5</sup> The term epoch is used by the NFC research community to express the term iteration.



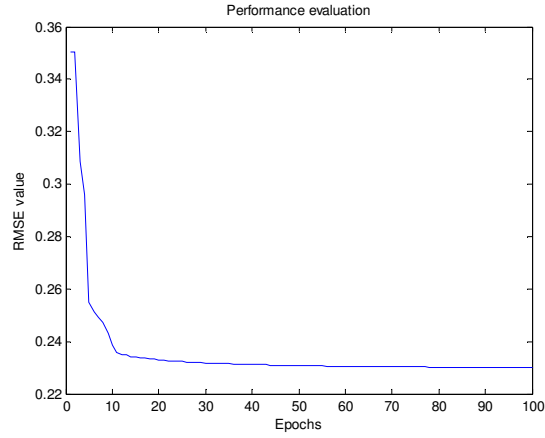


Figure 4.11: NFC training Performance for 3 features and 4 classes

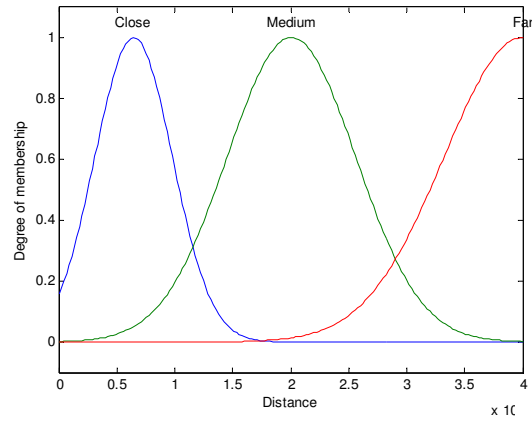


Figure 4.12: The distance feature as defined by user to train the NFC

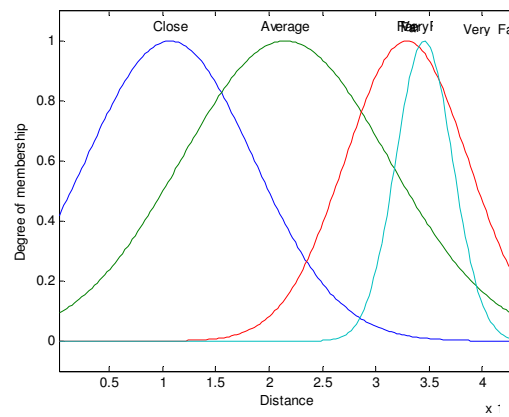


Figure 4.13: The distance feature after the NFC training

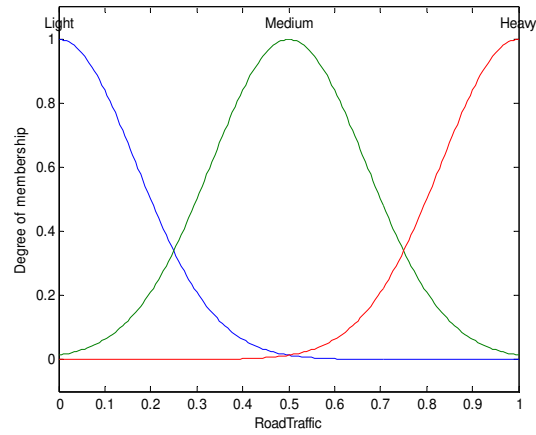


Figure 4.14: The road traffic feature as defined by user to train the NFC

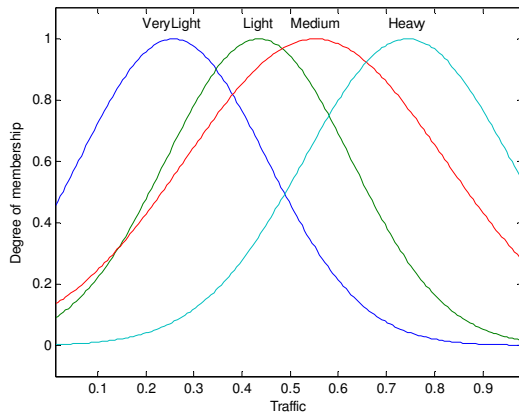


Figure 4.15: The road traffic feature after the NFC training

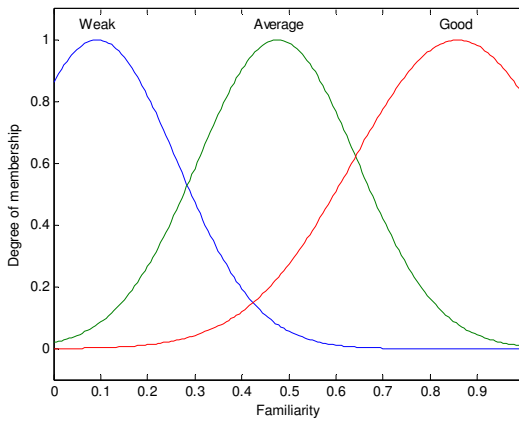
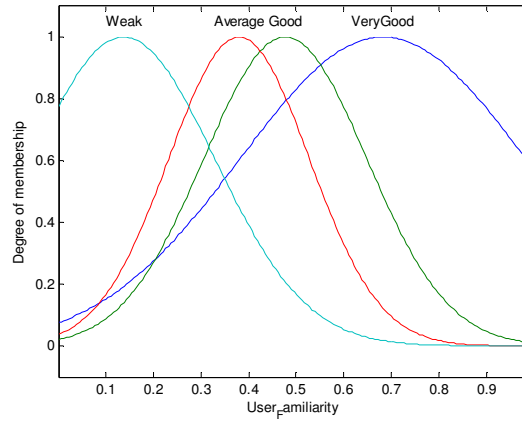


Figure 4.16: User's familiarity with area feature as defined by user to train the NFC



*Figure 4.17: User's familiarity with area feature after the NFC training*

[Yao and Thill, 2007] used ANFIS to generate spatial proximity quantifiers from a dataset. As we mentioned in Section 4.1.3, ANFIS is an approximation function. We used instead a neuro-fuzzy classifier to generate our spatial proximity quantifiers. Furthermore, [Yao and Thill, 200] did not integrate their solution in a GIS. Let us also mention that several works compared ANFIS and NFC and proved that ANFIS is not suitable for classification problems [Cetişli and Barkana, 2010]. For this reason, comparing our results to Yao and Thill's results [Yao and Thill, 2007] becomes non relevant since we are not comparing two classification approaches. In the next section, we present the qualitative proximity software that we developed to integrate our neuro-fuzzy classifier solution in a GIS using a Fuzzy Inference System (FIS).

## **4.4 A Qualitative Proximity Tool: Architecture and Implementation**

The main goal of using NFC is to generate membership functions and fuzzy rules for the fuzzy inference system based on the data sets and initial fuzzy rules specified by an expert. We developed a software tool to integrate the NFC output in a qualitative proximity reasoning tool which has been integrated in a GIS.

The general architecture of our tool is depicted in Figure 4.18. The NFC is a Matlab-based program which uses the data set to generate fuzzy membership functions and fuzzy rules. These outputs are used by a fuzzy inference system (FIS) which is managed by the *jFuzzyLogic* module. *jFuzzyLogic* is an open source fuzzy logic library implementing industry standards to simplify the development of fuzzy systems. It is a java package which uses FIS files to implement fuzzy rules. The *Java Topology Suite* (JTS) is a software package which is used to handle the spatial calculus of geographic features which are stored in the database (using *PostGIS*). The system output of our framework is a qualitative spatial relation represented by a linguistic variable which can be used by our pattern definition presented in Chapter 3.

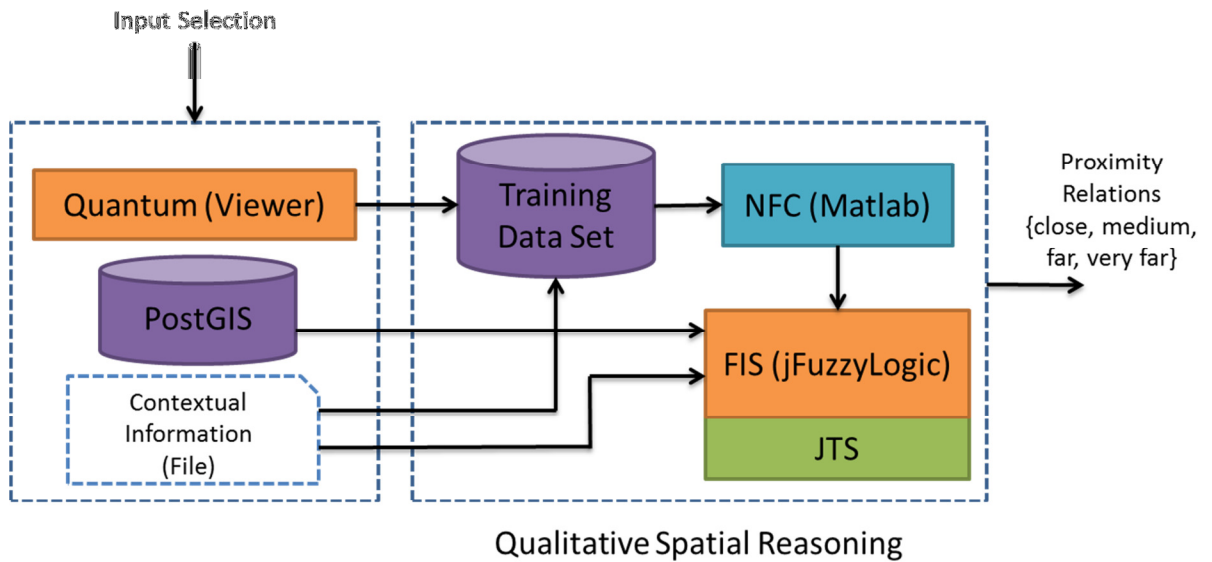


Figure 4.18: Architecture of our qualitative proximity tool

The qualitative proximity tool is designed to help users qualify spatial proximity using various factors and not only spatial distance. A user can train the tool using her experience and by specifying proximity parameters in a data set. If training outputs are satisfying, a user can use the generated membership functions to build an inference engine which determines the proximity.

Our proximity software allows for determining proximity in an automated way. This makes our approach original when it is compared to Yao and Thill's work [Yao and Thill, 2007] who limited their approach to a set of experiments based on Matlab and did not integrate their approach in a GIS.

## 4.5 Conclusion

Qualitative reasoning about spatial proximity is not limited to metric distances. Other factors can influence the user's perception of the proximity relation between two locations. A human being does not need to know the exact metric values of distances in his daily's life, which makes proximity relations uncertain and fuzzy. In this chapter, we considered these aspects in designing a solution for qualitative spatial proximity reasoning tool. Such a solution is integrated in a GIS and can be used in various application domains. The main contribution of our work is twofold. First, while [Yao and Thill, 2007] used ANFIS, we used a neuro-fuzzy classifier to train our system since determining proximity relations can be thought of as a classification problem. Second, we developed a software tool that integrates our solution in an existing GIS. This offers to users an automated tool that enables them to determine spatial proximity whereas [Yao and Thill, 2007] only applied their approach to a limited number of experiments.

Some improvements of our solution remain to be made. For example, it would be interesting to enhance the human machine interaction model so that a user could easily pick up locations and select contextual information and get the proximity relation. Moreover, the contextual information is currently specified from a simplified database. In the future, it would be interesting to get such information from a knowledge base. The next chapter will detail how this tool can be used in a spatiotemporal pattern detection framework to qualify the relative distance between pattern components. Finally, it is worth mentioning that an early version of this chapter has been published in the ISPRS Conference held in Toronto in 2014 [Barouni and Moulin, 2014b]. An extended version of this paper appeared in a special

issue on *Advances in Geospatial Statistical Modeling, Analysis and Data Mining* of the Canadian Institute of Geomatics [Barouni and Moulin, 2015a].

## **Chapter 5**

# **A Framework for Managing Qualitative Spatiotemporal Patterns**

### **Introduction**

In this chapter, we present a framework that we developed to manage qualitative spatiotemporal patterns. This framework is designed for large scale monitoring systems. Actually, a significant effort has been made in recent years to propose a new generation of data acquisition systems with reliable and efficient communication capabilities. These systems generate a huge amount of real-time data in various formats. End-users are very interested in finding data configurations based on their expertise and attempt to leverage the large amounts of data generated by acquisition systems. Several software tools have been proposed to help users achieve such goals. However, the available commercial solutions are mostly based on relational databases and use SQL queries to implement such functionalities. These systems do not allow for real-time detection of situations of interest (also called “patterns” in this domain) due to the weak expressiveness of SQL queries to represent situations of interest. We present a novel approach which extends Complex Event Processing to the real-time detection of situations of interest based on events, states and spatial objects. We leverage the rich semantics of the qualitative pattern representation model that we presented in Chapter 3 as well as the powerful proximity analysis tool introduced in Chapter 4 to present a complete solution to qualitatively represent patterns and to detect their instances from the event cloud under severe time constraints. Thanks to

our approach, a user will be able to react to detected patterns instead of trying to identify (or “to mine”) patterns in databases as it is proposed in currently available approaches.

This chapter is organized as follows. Section 5.1 presents the motivations of this work as well as an overview of the proposed approach. Section 5.2 presents the pattern abstraction layer of the proposed solution. Section 5.3 presents the data processing and pattern detection layer and a set of algorithms for pattern conversion. Section 5.4 presents a spatial extension of the CEP engine using fuzzy quantifiers. Section 5.5 presents a pattern management tool and the software architecture of our framework. Section 5.6 presents a case study that illustrates the different characteristics of the proposed framework. Section 5.7 concludes the chapter.

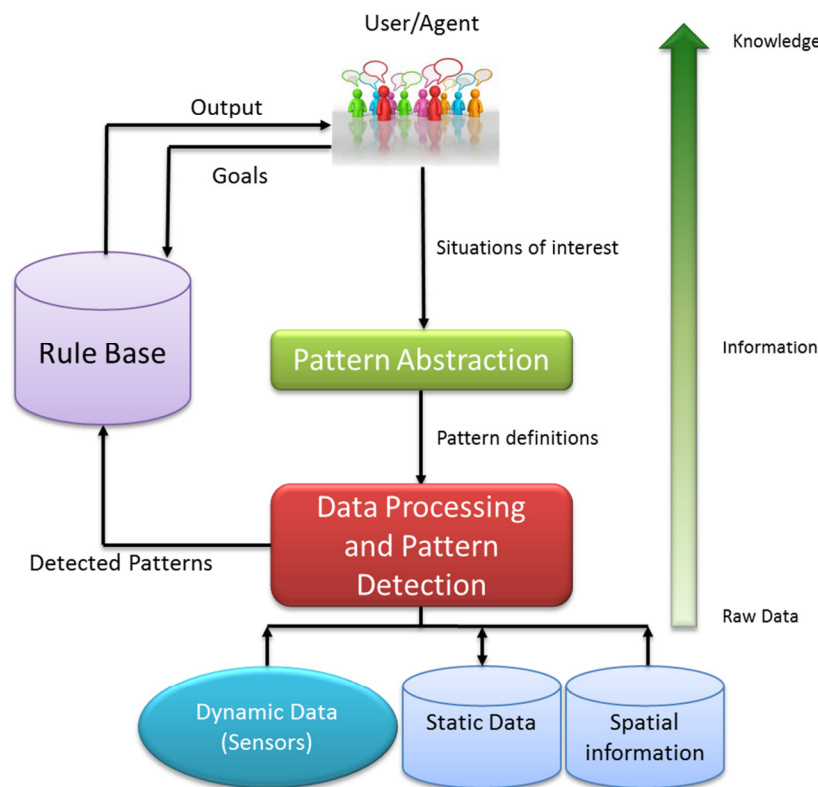
## 5.1 Motivations

The spatiotemporal pattern (STP) definition introduced in Chapter 3 allows for representing patterns using extended Conceptual Graphs. This knowledge representation provides a smooth mapping from natural language to computer processing and allows for building knowledge bases which can enable agents to reason about the detected pattern instances.

One of the goals of this thesis is to apply the STP definitions in the domain of large size acquisition systems. These systems have two types of data sources (also called views according to [Haddad, 2009]): static data sources and dynamic data sources. *Static data* can be available through GIS and knowledge bases describing the different states of the system’s components. *Dynamic data* is collected from geographically distributed sources (such as sensors) which generate data in various formats in real time. Generally, the sampling rate in acquisition systems is very high and the reaction to data change must meet severe timing constraints [Hong et al, 2013]. A human being or a software agent leverages their field experience in a



specific domain to define situations of interest that occur in the environment in which they operate. Using our STP formalism, such situations can be expressed as qualitative patterns and can use contextual information. First Order Logic can be used to reason about these patterns. Nevertheless, the detection of pattern instances from large acquisition systems has not been addressed in the former chapters of this thesis. A short case study has been proposed in Chapter 3 and raised some practical questions about how pattern instances can be detected from data provided by the event cloud. In this chapter, we address these questions and present a framework for spatiotemporal pattern representation, detection and reasoning which can be used by human operators as well as software agents to make decisions. An overview of the proposed approach is depicted in Figure 5.1 which emphasizes the global workflow of our framework.



*Figure 5.1: Overview of the different components of the proposed framework*

The proposed approach is automated and was developed in two main stages when we designed and implemented our pattern management framework. First, the semantic representation of STPs is handled by the *Pattern Abstraction Module*. These patterns are represented using our extended CG formalism (see Chapter 3) to facilitate the qualitative representation of different pattern components. Second, since this formalism does not allow for the detection of pattern instances from large data acquisition systems, this detection is handled by the *Pattern Detection and Data Processing Module* which facilitates the integration of our framework to current acquisition systems in different domains. Finally, reasoning about patterns is another aspect addressed by the proposed framework. Thanks to their expression in CGs, pattern instances can be stored in a knowledge base. They can be processed by software agents for decision making purposes. The next sections of this chapter detail each stage, present the corresponding implemented software modules and provide additional justifications for our technological choices.

## 5.2 Pattern Abstraction Module

The *Pattern Abstraction Module* is defined to handle the qualitative representation of spatiotemporal patterns. We use the formalism that we introduced in Chapter 3 to represent STPs. A user can represent patterns using our extended CG formalism which offers powerful mapping capabilities from natural language to computer language and vice versa. The pattern specifications are edited using the Amine Platform [Kabbaj, 2006] and stored in a knowledge base. The user defines a concept/relation lattice suitable to her application domain. Figure 5.2 illustrates the architecture of our pattern abstraction module.

Pattern specifications are input by the user using the *Amine GUI* which allows for adding new concepts in the concept type lattice. Several elements are involved in the spatiotemporal pattern definition such as events, states, geographic objects, spatial relations and temporal relations. These elements can be represented in a knowledge base using an abstraction level which is handled by the Concept Type Lattice (CTL).



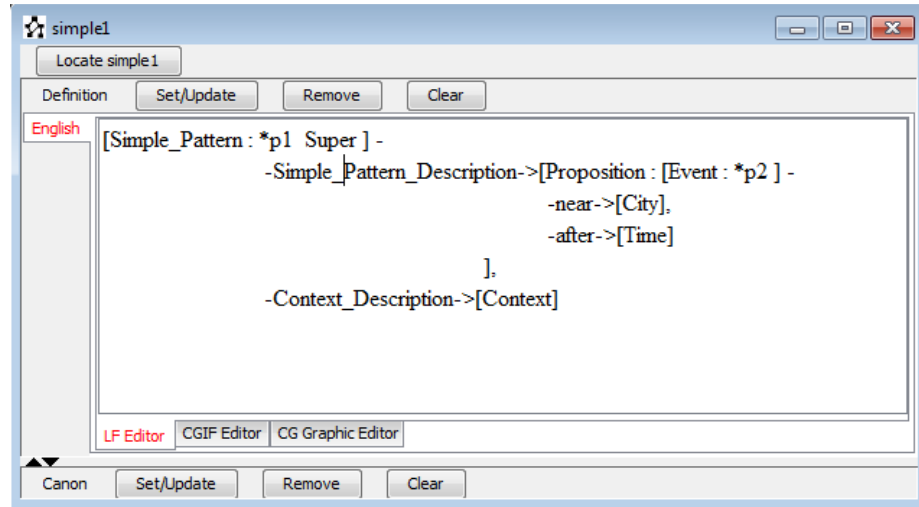


Figure 5.4: A simple pattern represented using the Amine Platform

Now that situations of interest are defined using patterns, we need to address the issue of pattern detection from large applications and introduce our *Data Processing and Pattern Detection Module* in the next section.

### 5.3 Data Processing and Pattern Detection Module

One of the challenges of large scale systems is the management (querying, storage and display) of large volumes of data in a short time period: this is a practical issue that falls in the field of big data management which has been gaining an increasing popularity in recent years. Such systems usually collect events generated by a large number of sensors deployed in the field. Ideally, these events should be correlated and linked to other components from the environment to generate pattern instances. In Chapter 3, we addressed the issue of pattern representation using qualitative techniques and the issue of representing dynamic spatiotemporal phenomena. However, the issue of detecting pattern instances from real world data was not tackled. In this section, we introduce Complex Event Processing as an effective way to manage the stream of events. We propose to explore this technology and see how it can be used in conjunction with our pattern processing module to detect pattern instances.

Pattern detection in real-time acquisition systems is an issue that has been addressed in the literature by several research communities. Some of these works are based on *database management systems* (DBMS) in which data is stored and indexed before processing. When a user needs to find a pattern instance, he needs to query the database. This cannot comply with several use cases in monitoring applications where patterns should be detected as soon as sensors report observations. Furthermore, the concept of event is not clearly defined in DBMS and the pattern detection mechanism is usually complex, time and labor consuming. The notion of *Event Driven Architecture* (EDA) has been introduced to overcome such limitations in DBMS. EDA is a software architecture which manages the creation, processing and monitoring of events. After several years of work, the research community proposed a set of EDA approaches to overcome the limitations of classical DBMS in terms of pattern detection and management. According to [Cugola and Margara, 2012] these solutions can be classified in two families: *Data Stream Processing Models* (DSPM), and *Complex Event Processing Models* (CEP). DSPMs are a special version of databases. They are designed to deal with transient data which is continuously updated, while traditional databases are more suitable to deal with persistent data. Therefore, while a DBMS returns a complete answer to a query than runs just once, a DSPM runs continuous queries and returns updated results when new data arrives from various sources. Although DBMSs are different from DSPMs, both kinds of systems process the incoming data through a sequence of transformations based on SQL operators such as aggregation and join.

*Complex Event Processing* (CEP) has been introduced in recent years to overcome the limitations of DSPM and DBMS. CEP systems correlate simple events temporally and/or spatially using advanced temporal and spatial operators. This kind of correlations allows for inferring more meaningful “complex” events [Helmer et al, 2010]. CEP systems increase the expressive power of their query language to consider complex patterns of events that involve the occurrence of multiple related events [Cugola and Margara, 2012].

Basically, the main difference between Complex Event Processing and relational databases relies on the way events are defined and processed. *Relational databases* and the *Standard Query Language* (SQL) are designed to manage static data. They can support complex queries. They are optimized for disk access on data servers where data are stored. A query is used to retrieve data from a database. If a system needs some data 10 times per second it must trigger the same query 10 times per second. This does not scale up well to hundreds or thousands of queries per second! [Esper, 2015]. Other solutions have been proposed to overcome these performance limitations such as using *database triggers* which can be activated in response to database update events. However, database triggers tend to be slow and often cannot easily perform complex condition checking and perform application functionalities to appropriately react to an incoming event [Esper, 2015]. As an alternative, a CEP can be thought of as a ‘database turned upside-down’. The CEP engine allows applications to store queries, channel the data through predefined streams, detect patterns and store results in databases. Therefore, the CEP engine gives an answer in real-time when pattern conditions are satisfied. The execution model is thus continuous rather than being only reactive to query submission.

A typical CEP architecture is depicted in Figure 5.5. Events are generated by so-called *event producers* (input). Events are time-stamped and have some attributes which are part of the event definition. Event producers vary from one application domain to another. A given CEP system can deal with several event producers which can be heterogeneous (i.e. specified in different ways). The CEP engine uses temporal and logical operators to correlate events and then sends them to event consumers (output). *Event consumers* can be humans, software agents, other intelligent applications or storage systems. They receive events communicated by the event processing engine and react to these events. The reaction might include the generation of new events, so that consumers can be event producers as well.

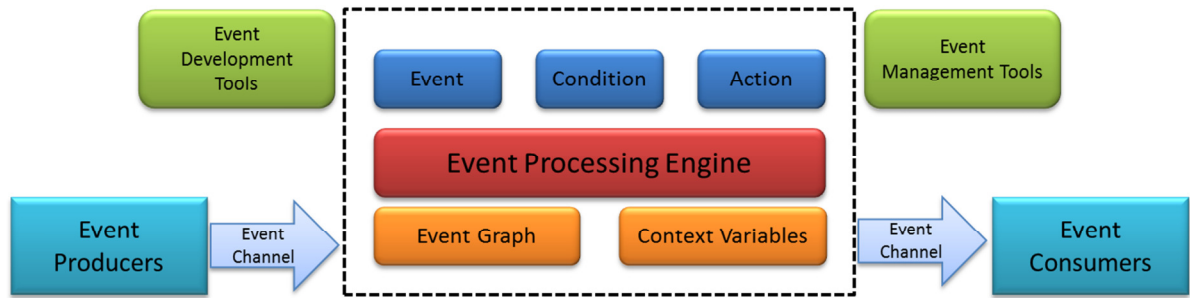


Figure 5.5: A typical CEP architecture [Helmer et al, 2010]

The *Event Processing Engine* receives events from different *event channels*. It is based on the *Event Condition Action* paradigm (ECA) which consists in triggering actions when appropriate conditions are satisfied. Possible actions can be “sending the events to the event consumer” or “generating a set of new events”. The processing of events can be carried out on separate events or on a combination of events using temporal, spatial, causal operators and other user-defined functions. The correlation of events can be expressed using a specific language that is used to generate the so-called patterns. In a CEP, *Event Development Tools* are used to define events, to define rules (patterns), to program user-defined functions for event correlation and to set parameters for the overall system. *Event Management Tools* are used to fine-tune the system, to collect system outputs and to monitor the system’s performance.

An interesting review of CEP frameworks can be found in [Cugola and Margara, 2012]. For our framework, we selected the ESPER software which is a lightweight CEP system offering an *Event Pattern Language* (EPL for short) to represent patterns using an SQL-like language. EPL queries are created and stored in the pattern base and publish results to so called *listeners* when events are received by the engine or when *timer events* occur and match the criteria specified in the query.

*Events can be consumed* according to different *consumption policies* that can be configured in the CEP engine. For example, an event is consumed when a specific pattern is detected and then it is removed from the event stream. But, if certain events

are present in several pattern definitions, it is possible to keep them in the event stream so that all related patterns will be detected. The suitable *consumption strategy* can vary from one application to another.

### 5.3.1 Data Processing

In our framework, we implemented the data processing module in three steps.

1. **Step 1: Connect to data sources.** Data sources are usually sensors which report dynamic variations. The connection to these sensors can be point to point or through an aggregator (i.e. a data concentrator). The data format varies from one domain to another. For example, in the power industry, events are reported using industrial protocols. In the telecommunication industry, people use real time database systems. Each sensor is defined by a set of attributes. Figure 5.6 depicts a Fault Current Indicator Sensor used in outage management systems as introduced in Chapter 3. Note the presence of spatial attributes in the sensor characteristics that allow for locating these sensors in a spatial database.
2. **Step 2: Define the event types** that can be generated by each sensor type in the CEP. Now that the CEP is connected to data sources, it is important to extract relevant data from the incoming events. For each sensor type, we need to select a specific number of event types and to define their corresponding event classes. Figure 5.7 depicts an example of events generated by a Fault Current Indicator Sensor used in the power distribution industry to detect outages and to measure the status of power lines. These events are collected by a data acquisition system through a cellular link using an industrial communication protocol. Figure 5.8 depicts the class defining an event in ESPER which will be used by the engine to collect events from the data acquisition system. This structure can vary from one event type to another and its attributes can be different from one application to another.



ID : b195d772-0ad7-45a7-beb3-d6b116ade734  
 EnumSmartSensorTypeID : 1  
 Name : Sensor1  
 SerialNumber : 1523  
 FirmwareRevision : 52  
 CustomerNumber : 5869  
 Latitude : 46.781828  
 Longitude : -71.27829  
 IsAcknowledgeRequired : False  
 EnumPhaseID : 1  
 EnumStatusFlags : 1  
 LastSequenceUsed : 2  
 IsDeviceCalibrated : True  
 OverCurrentThreshold :  
 ResetCurrentThreshold :  
 RowState: Unchanged

Close

Figure 5.6: An example of sensor definition

Signal	Δ	Current Value	Previous Value	Scale	Offset	Reported	Last Update Time
BatteryVoltage	▼	0	0	0.001	0	✓	23/04/2015
ChargeCircuitryEnabled	▼	0	0	1	0	✓	23/04/2015
DeviceTemperature	▼	36	0	0.01	0	✓	23/04/2015
Fault	▼	3	0	1	0	✓	23/04/2015
MomentaryOutage24hCount	▼	0	0	1	0	✓	23/04/2015
NominalCurrent	▼	0	0	1	0	✓	23/04/2015
OverCurrent	▼	0	0	1	0	✓	23/04/2015
PeakCurrent	▼	14	0	1	0	✓	23/04/2015
Power	▼	0	0	1	0	✓	23/04/2015
SampledCurrent	▼	0	0	1	0	✓	23/04/2015

Figure 5.7: A sample of events generated by a Fault Current Indicator sensor

<<Java Class>>

**PeakCurrent**

com.espertech.esper.example.qos\_sla.eventbean

▲ eventName: String  
 ▲ eventID: int  
 ▲ location: String  
 ▲ time: SimpleDateFormat

● PeakCurrent(String,int,String,SimpleDateFormat)  
 ● getEventName():String  
 ● getTime():SimpleDateFormat  
 ● getEventID():int  
 ● setPosition(double):void

Figure 5.8: An example of a class for event definition in ESPER

3. **Step 3: Configure the ESPER engine.** The CEP requires a specific set of parameters to be loaded at runtime. For example, our CEP runtime system loads the event definitions, a set of user-defined functions, the data source connection parameters and the pattern listeners. For each pattern type, a pattern listener is defined to process the detected patterns and perform actions. We will detail the notion of pattern listener in a forthcoming section.

Since ESPER uses the *Event Pattern Language* (EPL) as a language to represent patterns, we developed a set of algorithms to convert abstract pattern definitions from our Conceptual Graph formalism to EPL. These algorithms are presented in the next section.

### 5.3.2 Automated Pattern Conversion from Conceptual Graphs to EPL

Pattern specifications are represented using the *Pattern Abstraction Module*. They need to be represented in the EPL language to allow the ESPER engine to query data from data sources (either static or dynamic) and to detect pattern instances. The detected pattern instances are stored in text files. Figure 5.9 gives a description of our Pattern detection and data processing module.

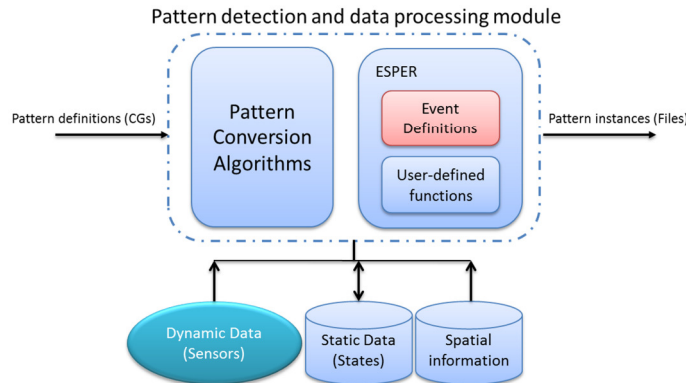


Figure 5.9: Description of the pattern detection module

We used the Java programming language and the Eclipse development tool to implement the algorithms which convert the simple and complex patterns introduced in Chapter 3 to an enriched version of Event Pattern Language (EPL). Since EPL is SQL-based, the proposed algorithms convert pattern definitions from Conceptual Graphs to enriched SQL-like statements.

### 5.3.2.1 Simple Pattern Conversion Algorithm

Our pattern conversion algorithm shall consider two aspects. First, it needs to find the event in the simple pattern and its relations to spatial and/or temporal attributes. Second, if the event is related to states, the algorithm needs to find the temporal relations between the event and the states. We also use the ESPER user-defined functions to represent custom temporal relations between events and states as well as other relation types such as spatial relations. User-defined functions are custom functions which are implemented using the Java programming language. They can be used in the EPL statement and can be called by the ESPER runtime engine. Table 5-1 summarizes the pseudo code of our conversion algorithm for simple patterns.

---

**Algorithm 1:** Simple Pattern Conversion SimplePatternFinder(KB: input)

---

**Variables**

RelationType ///Can be temporal or spatial relation.

SpatialObject///An abstraction object from the environment with spatial attributes.

TemporalReference. // A reference on the time axis. Can be a semantic reference or a numerical data.

WhereClause. //The where clause of the EPL statement

States //List of states related to a given Event

KB //The Knowledge Base //

ListPatterns //List of simple patterns

i: counter. Number of generated EPL statements.

**Output**

EPLStatement: Array of Strings. //Simple Patterns in EPL

---

Begin

1. Initialize all the variables ().
  2. Load all the concept type lattices
  3. ListPatterns  $\leftarrow$  Get all the simple patterns from the KB
  4. For each pattern in ListPatterns
-

---

```

5.    Event ← Get the Concept type Event
6.    Relations ← Get_the_outcome6_relations_(Event)
7.    For each relation in Relations
8.        If the relation type is spatial relation Then
9.            SpatialObject ← Get the spatial object type
10.           WhereClause ← Relationtype(Event, SpatialObject)
11.        End If
12.        If the relation type is temporal relation Then
13.            TemporalReference ← Get the temporal attribute
14.            WhereClause ← Relationtype(Event, TemporalReference)
15.        End If
16.    End For
17.    States ← GetStates(Event)
18.    TempRel ← GetTemporalRelations(Event)
19.    EPLStatement[i] ← CreateEPLStatmt(Event, States, WhereClause, TempRel)
20.    i=i+1
21. End For
End

```

---

*Table 5-1: An example of simple pattern conversion algorithm*

Basically, the above algorithm allows for filling a generic EPL statement with information extracted from the CG representation of an STP to build the enriched EPL pattern representation that we propose. For example, one of the generic EPL statements for a simple pattern can take the form illustrated by the code below. This simple pattern represents a qualified event with a temporal reference using a temporal relation. The second statement gives an example where an event is also related to a state using the temporal relation “beforeEvent”. This allows for representing dynamic situations as introduced in our pattern representation model (Chapter 3).

---

<sup>6</sup> An outcome relation is a relation which originates from a concept and relates it to another one.

```

{
_sqlstatement="Select "+eventType+" where " + eventType+ ".time is "+
temporalRelation +" "+temporalReference ;
}

{
_sqlstatement="Select * where " + eventType+ ".time is "+ temporalRelation
+" "+temporalReference" and " beforeEvent(eventType,stateType);
}

```

Figure 5.10 illustrates some output examples of the proposed algorithm. The algorithm starts by loading all pattern definitions expressed in conceptual graphs from the knowledge base populated using the pattern abstraction module. Then, it displays each simple pattern expression in a CG linear form followed by the enriched EPL statement (using the EPL language). Note the presence of some user-defined functions *near* and *far from* in the EPL statement. These user-defined functions will be discussed in detail in a forthcoming subsection.

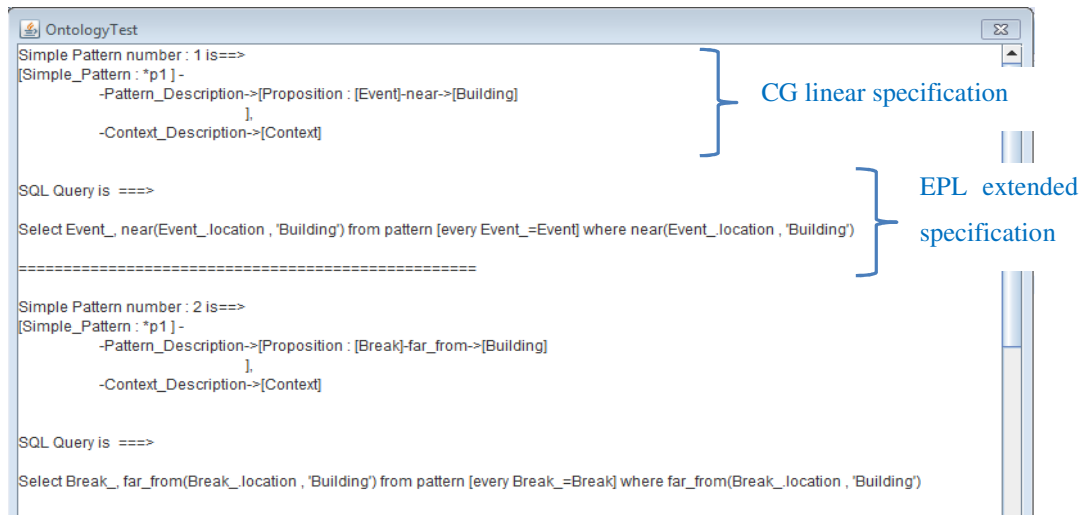


Figure 5.10: Examples of simple patterns in CG and EPL formats

### 5.3.2.2 Complex Pattern Conversion Algorithm

Now that we have a simple pattern algorithm that allows for mapping simple pattern definitions (in our CG formalism) to our enriched EPL statements, we propose in this subsection another algorithm to map complex patterns abstract definitions (expressed in CG) to enriched EPL statements. In Chapter 3, a complex pattern is defined as a set of simple patterns related by spatial and/or temporal relations. The basic idea of the proposed algorithm is to identify in the CG specification all simple patterns included in the complex pattern structure and to find relations between these simple patterns. Table 5-2 describes the pseudo code of our conversion algorithm for complex patterns.

---

**Algorithm 2:** Complex Pattern conversion

---

**Variables**

RelationType ///Can be temporal or spatial relation.

SpatialObject///An abstraction object from the environment with spatial attributes.

TemporalReference. // A reference on the time axis. Can be a semantic reference and numerical data.

WhereClause. //The where clause of the EPL statement

SimplePatterns //List of simple patterns in a complex pattern

KB //The Knowledge Base //

ListPatterns //List of complex patterns

i: counter. Number of generated EPL statements.

**Output**

EPLStatement: Array of String. //Complex Patterns in EPL

---

**Begin**

1. Initialize all the variables ().
  2. Load all the concept type lattices
  3. ListPatterns  $\leftarrow$  Get all the complex patterns from the KB
  4. For each pattern in ListPatterns
  5.     SimplePatterns  $\leftarrow$  Get all the simple patterns(ListPatterns)
  6.         For each s\_pattern in SimplePatterns
  7.             Event $\leftarrow$ Get the Concept type Event(s\_pattern)
  8.             Relations $\leftarrow$ Get all the spatial or temporal relations related to Event.
  9.                 For each s\_relation in Relations
  10.                     If the s\_relation is Spatial\_relation Then
-

---

11.	SpatialObject	←	Get the spatial object type
12.	WhereClause	←	Relationtype(Event, SpatialObject)
13.	End If		
14.	If the relation type is Temporal_relation	Then	
15.	TemporalReference	←	Get the temporal attribute
16.	WhereClause	←	Relationtype(Event,
	TemporalReference)		
17.	End If		
18.	End For		
19.	End For		
20.	EPLStatement[i]	←	CreateEPLStatement (SimplePatterns, WhereClause)
21.	i=i+1		
22.	End For		
	End		

---

*Table 5-2: Complex pattern conversion algorithm*

Figure 5.11 displays the output of the proposed algorithm for a given example of a complex pattern. The algorithm starts by loading from the knowledge base all the pattern definitions expressed using our extended CG formalism. Then, it searches for all the relations between simple patterns. Finally, it calls the simple pattern conversion algorithm which was presented in the previous subsection.

The program output depicted in Figure 5.11 shows the difference in terms of expressiveness between a pattern expressed using our CG and the resulting EPL representation. The CG formalism allows for representing patterns in a more human - readable form whereas the EPL enriched statement is more efficient to represent these patterns in a format interpretable by a CEP engine. If a complex pattern has a complex data structure; the corresponding EPL statement can become really complex; hence the practical interest of our automated conversion algorithm

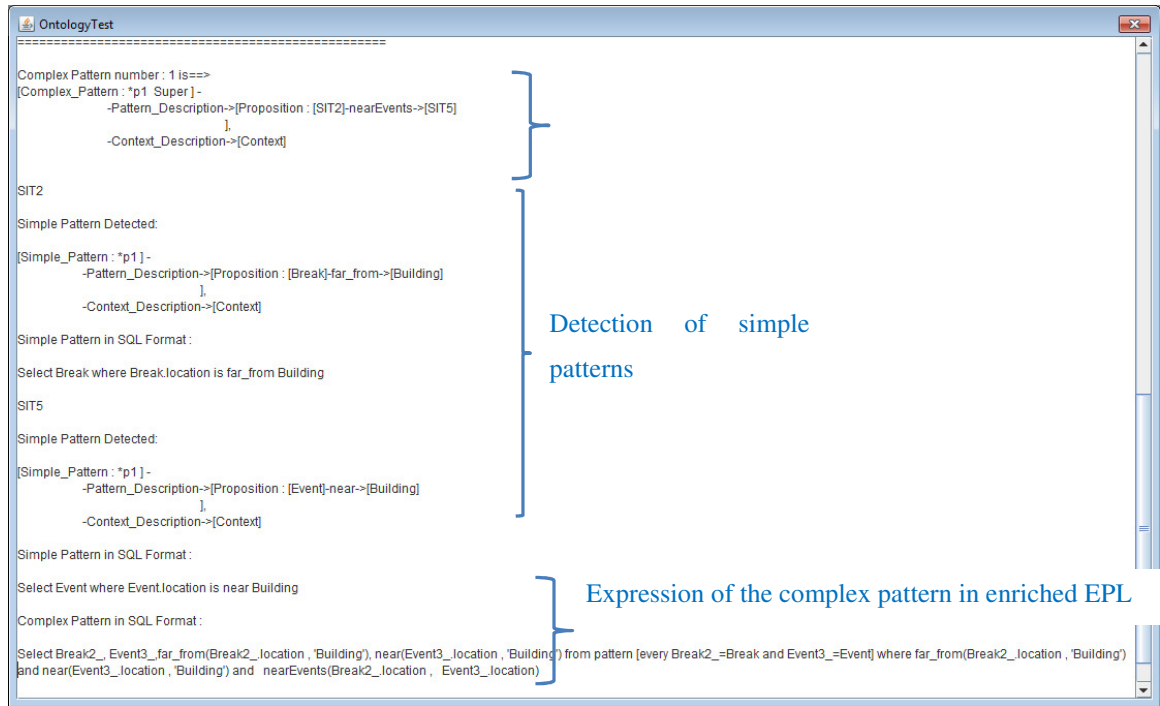


Figure 5.11: An output example of the complex pattern conversion algorithm

In the Pattern Detection and Data Processing Module illustrated in Figure 5.9, the *User defined functions module* is part of the ESPER engine. *User defined functions* implement special functions that can be called in *pattern definitions*. In our framework, these functions are used for two purposes:

1. A *State Manager Module* has been specified to access the static data source and to retrieve or update the status of a component of the environment which is involved in a spatiotemporal pattern definition.
2. In order to support qualitative spatial proximity relations, we implemented a spatial extension of the ESPER framework using our qualitative spatial tool presented in Chapter 4. We detail this extension in the next section.



## 5.4 A Spatial Extension of the CEP Engine

Since the *Pattern Detection Module* deals with qualitative spatiotemporal patterns, qualitative spatial relations used in pattern definitions need to be supported. The CEP framework needs to be enhanced to support these qualitative spatial relations. We propose in this section a spatial extension of a CEP engine to support spatial proximity relations since they are the most used in practical applications. Our software aims at generating automatically spatial proximity quantifiers using user defined functions.

CEP systems have been originally designed and applied to the financial and economic industries in order to predict market development and exchange rate trends. CEP patterns emerge from relationships between event's attributes such as *time* and *cause* (causal relation between events) and *aggregation* (significance of an event's activity with respect to other events). Processing geospatial events requires the extension of CEP to include location attributes [Resch et al, 2010]. Spatial relations are used to combine geo-referenced events and to create spatiotemporal patterns.

In the literature, several works proposed spatial extensions to CEP engines. Among them, we note SpatialRules [SpatialRules, 2013] which is a CEP engine for geospatial data and is compliant with OGC geospatial specifications [OGC, 2015]. Event processing is performed through rules that offer geospatial and temporal operators. Geospatial operators are mainly topology or distance based. *GCEP* [GCEP, 2014] is an extension of the *ESPER* CEP engine that allows for using OGC geospatial functions in the rules used to filter events. The *GCEP* engine offers 12 topological functions for *ESPER* for Java that conforms to Open Geospatial Consortium standards. *ruleCore CEP Server* [ruleCore, 2013] is a CEP engine used for real-time detection of complex event patterns. The system is scalable and can be used to implement event driven architecture solutions: *ruleCore* enables defining rules using location information. Location data can be collected from GPS or other sensors and

can be natively processed by *ruleCore*. The engine allows for stream creation based on events coming from specific geographic areas.

Although the above solutions support the processing of spatiotemporal events, they do not deal with uncertainty and vagueness and therefore they are not close enough to users' qualitative reasoning which is essentially based on natural language. Taking advantages of Zadeh's linguistic variables [Zadeh, 1965], we leverage our approach presented in Chapter 4 to develop an extension of the ESPER CEP engine by introducing fuzzy spatial relations between events to allow for the definition of qualitative spatiotemporal patterns. To this end, we defined and implemented fuzzy proximity relations between two events using our qualitative proximity reasoning framework. A Neuro-Fuzzy Classifier is used to train the system using data sets provided by users. Spatial relations can be automatically deduced from the qualitative proximity tool. To integrate this tool to ESPER we implemented a number of user defined functions. Examples of fuzzy spatial relations used in our framework are "very near", "near", "far from" and "very far from". Here is an example of a pattern expressed in EPL and using a fuzzy spatial relation.

*Select \*from Pattern [Every Break] Where near(Break.location, Building) and  
Before(Break, Normal)*

Where *near(Break.location, Building)* is a user-defined function expressing a qualitative spatial proximity relation between the event spatial attribute of type *Break* and a spatial object of type *Building*. This function takes two spatial objects as parameters and returns a Boolean value.

The automated generation of spatial relations can be summarized in the following three steps, taking advantage of the software modules that we presented in Chapter 4:

1. **Step 1:** A user provides a training data set to the NFC according to his experience.

2. **Step 2:** The NFC generates the fuzzy quantifiers and their related membership functions.
3. **Step 3:** User-defined functions are used in ESPER to deduce spatial relations using a Fuzzy Inference System (FIS).

Now that we have introduced all the elements of our pattern management framework, we present in the next section the software architecture and the implementation details. Then, we propose a case study to illustrate some application examples of this framework.

## 5.5 Software Architecture and Implementation

The software architecture of our framework is depicted in Figure 5.12. In order to train the NFC (see the *NFC* block in the figure), a dataset is provided by the user and can be loaded from an Excel spreadsheet. The membership functions corresponding to the spatial proximity quantifiers (subjective and objective) are generated by the NFC which is implemented in Matlab. JFuzzyLogic is a JAVA package that we used to implement a fuzzy inference system for qualitative spatial reasoning (Chapter 4). The output of the fuzzy inference system is a fuzzy spatial relation which can be used by the Amine platform in the pattern representation formalism and by the ESPER engine for pattern matching operations. The Amine platform is also used to implement reasoning mechanisms thanks to the *Prolog+CG module* (*Agents* block in the figure). Pattern instances are stored in the pattern repository as well as the contextual information. Software agents interact with these modules as part of their reasoning model. The *Pattern Detection and Data Processing* module (black lined block in the figure) loads the system configuration and initializes pattern listeners. A *pattern listener* is a computer program which subscribes to a pattern definition and takes actions on the detection of a pattern instance. In our framework, a specific pattern listener is associated with each pattern definition loaded from the knowledge base: it creates a new instance of the detected pattern and matches its different

components with detected data. The detected pattern instance is saved in a text file for future use.

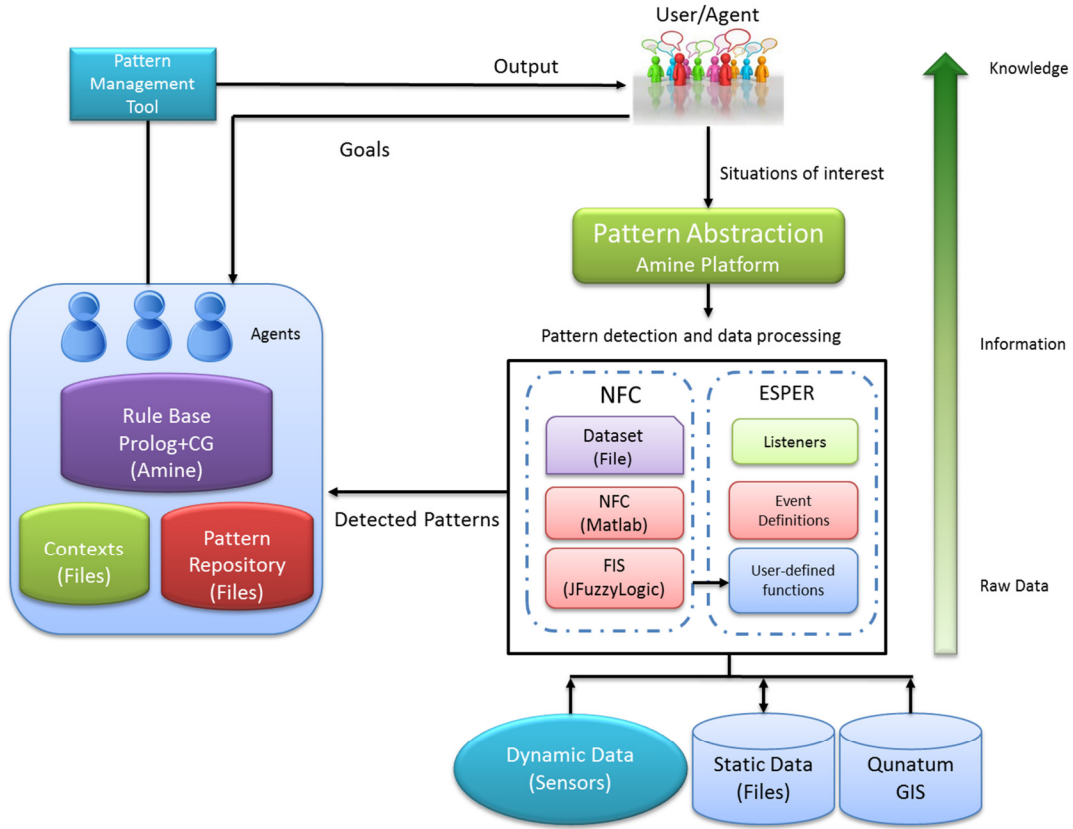


Figure 5.12: Architecture and software packages used in our framework

We developed a *pattern management tool* (blue block in the figure) to facilitate the user's interaction with our framework. A user can visualise and manage patterns. The main window of the tool is depicted in Figure 5.13. A user can load the knowledge base (KB) in Amine KB format (XML file) (1 in Figure 5.13). An algorithm fetches all the patterns in the KB and lists them in a table (2 in Figure 5.13). To view a pattern, the user can select one pattern entry and use the *View Selected Pattern* button (3 in Figure 5.13). He can also view the pattern in EPL format by selecting the *Convert Selected Pattern* button (4 in Figure 5.13). Using the *Start Event Generator* button (5 in Figure 5.13), a user can generate a number of different types of events. Each event is time-stamped and geo-referenced. Finally, the user can visualize the generated events and the detected patterns instances from the Pattern Management

Tool window (7 in Figure 5.13). A user can load a Quantum GIS Lisboa project file in order to configure the spatial environment (6 in Figure 5.13). He can also select a dataset from an Excel spreadsheet in order to train the fuzzy inference system. Finally, the user can view the resulting spatial proximity relations and their associated membership functions.

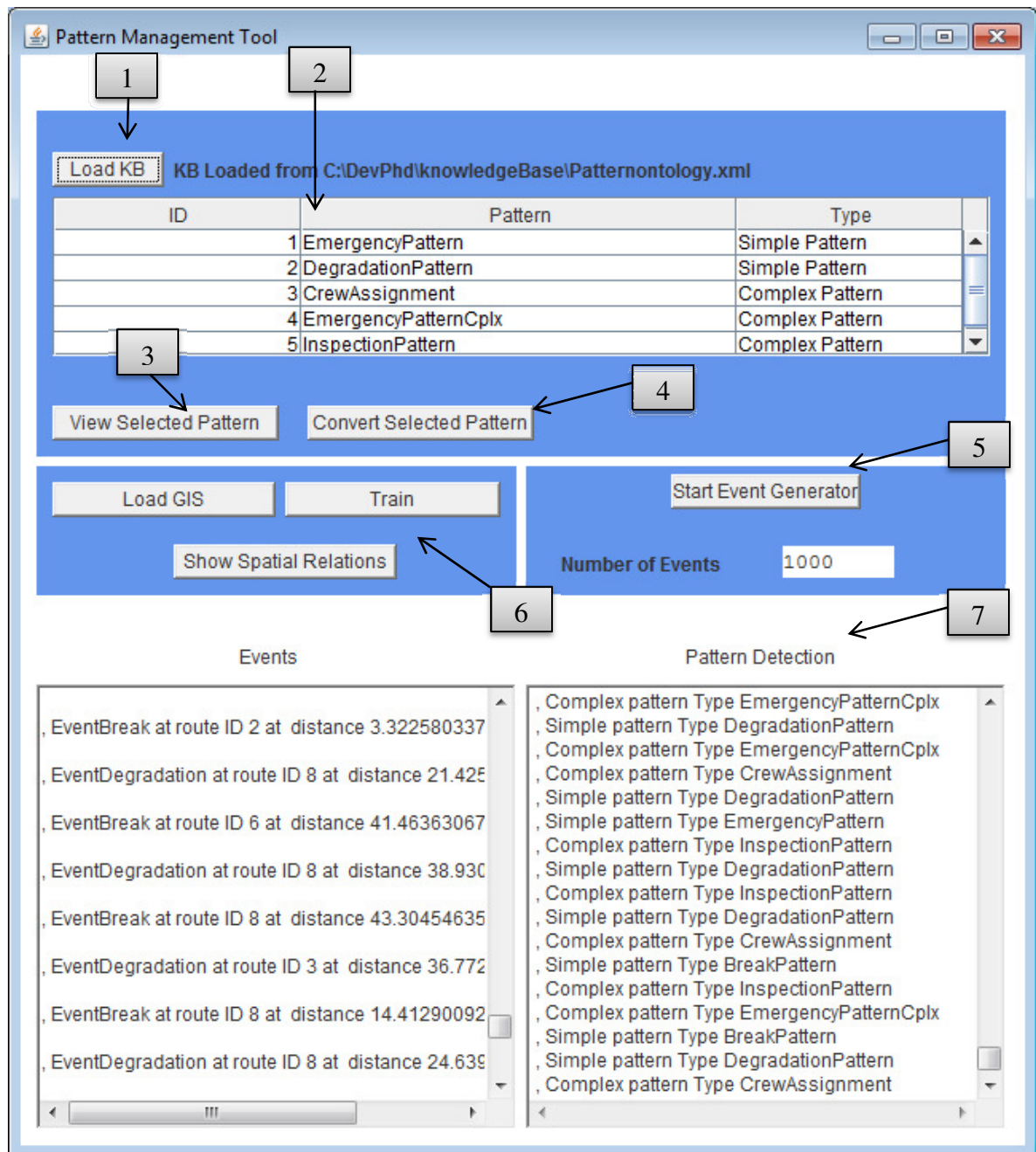


Figure 5.13: The Pattern Management Tool

After presenting how patterns are managed by our framework, we present in the next section a case study from the telecommunication industry to provide some examples of using pattern instances and contextual information in agents' decision models. We also show some pattern configurations that may be defined by users to identify typical situations of interest in their application domain.

## 5.6 Case Study

This case study is motivated by a typical application in the telecommunication domain. In particular, we address the issue of asset management and monitoring. Typically, A *Remote Fiber Test System* (RFTS) is a real time acquisition system designed to monitor fiber optic networks [Ponchon and Champavere, 2011]. It is widely used in the Telecommunication industry to optimize the *Service Level Agreement* between service providers and consumers. It can also be used in the Oil & Gas industry to monitor pipelines. In that case Optic Fibers can be deployed throughout gas pipelines and the occurrence of a fiber fault event means that the related pipeline is damaged.

### 5.6.1 Remote Fiber Test System

The RFTS is based on the *Optical Time Domain Reflectometer* (OTDR) technology [Ponchon and Champavere, 2011]. OTDRs are commonly used to characterize the loss and length of fibers as they go from initial manufacturing through to cabling, warehousing while wound on a drum, installation and then splicing. OTDRs are also commonly used for fault finding on installed systems. Figure 5.14 presents an example of an OTDR acquisition and the different information that can be deduced about a fiber optic link.

The RFTS is a distributed system where *Remote Terminal Units* (RTU) are geographically deployed and continuously run to monitor the fiber status using OTDRs. When a fiber fault occurs, it is detected by an RTU and reported to a central

server which maps the fault in a GIS. Figure 5.15 depicts the general structure of a remote fiber test system.

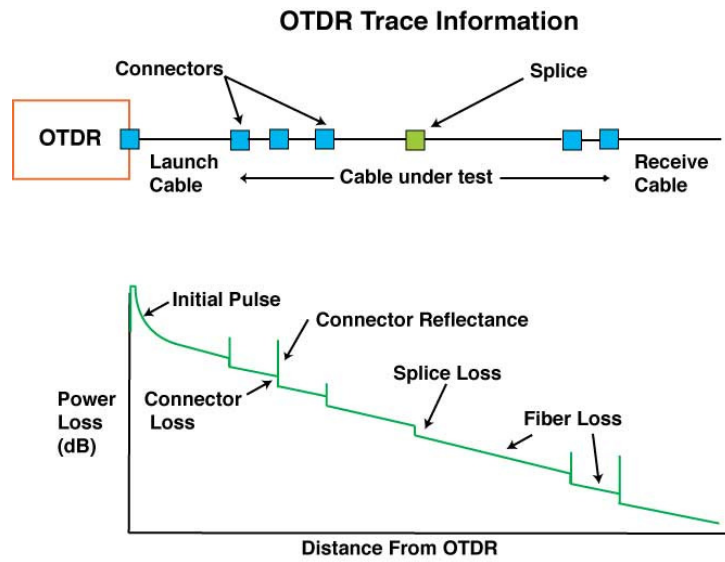


Figure 5.14: OTDR Trace Information [Foa, 2015]

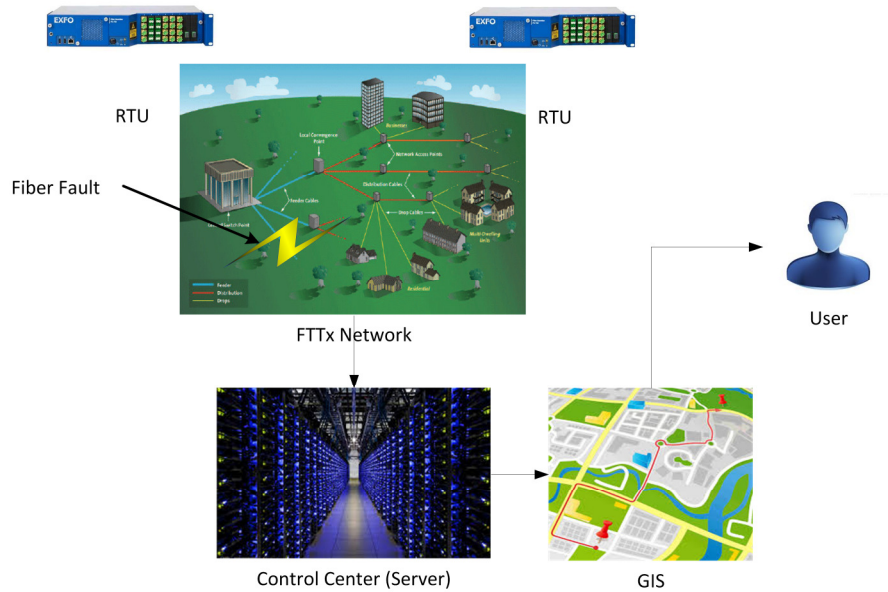


Figure 5.15: The general architecture of an RFTS

A user needs to evaluate the distance between the fault location and several locations such as: crew location, location of critical sites where Service Level Agreements (SLA) must be fully respected (such as bank, broadcasting channel building and security cameras) and asset locations (splice point, transmission equipment and so on). Hence, the notion of proximity is very important in such applications and plays a key role in users' decisions.

### 5.6.2 Background and Challenges

Basically, an OTDR is designed to calculate the total attenuation of a fiber link. Attenuation can be affected by the total length of the fiber or by the presence of splices and connectors. Mainly, the OTDR reports two types of events: *fiber break events* and *fiber degradation events*. The fiber break event occurs when the fiber optic link is cut or when a splice is damaged, or when a connector is disconnected. This type of event can also occur when there are maintenance operations performed on the link (known reasons, they are usually planned) or when there are damages on the fiber caused by unplanned actions such as construction activities in the area. Current commercial OTDRs are designed to support up to 128 optical ports. They are usually installed at the central office or deployed at the outside plant. When a new fiber is commissioned, the OTDR sends a laser signal on the fiber and saves the data acquisition to become a trace reference<sup>7</sup>. The trace reference allows for characterizing the fiber and keeps the information about the total fiber length, the number of splices/connectors and their positions on the fiber; and the total fiber loss. The OTDR starts to monitor the fiber according to a test cycle. This test cycle depends on the number of fibers under test and the sampling period for each fiber.

When a fiber optic event is triggered by an OTDR, the RTU generates the event data and sends an XML file using a secure connection (SFTP protocol for example) to the main server. This file is parsed by a *Fiber Optic Network modeling* software to get

---

<sup>7</sup> In fiber optic testing, a data acquisition is called « reference trace » when it is used by the system as a basis to compare to new data acquisitions. Any variation relatively to the reference and which exceeds a user-defined threshold will generate an alarm.



the fiber ID and the event distance and to locate the fault on the map. The event data is also used to generate alarms and to notify end-users using different notification channels such as display screens, emails and text messages.

As we mentioned earlier, SLA is one of the key performance indicators defined to evaluate the quality of service provided to customers in the telecommunication world. To optimize SLA, asset management teams use monitoring systems to guaranty the reliability of their transmission systems. In case of a fiber failure, the asset management team should promptly react to locate the fault and assign a crew to repair the fiber. The monitoring system can also store data in a historical database and make it available to experts for further analysis in order to setup preventive maintenance mechanisms.

Since the main goal of using an RFTS is to help telecommunication operators guarantee Service Level Agreements, it can be involved in the following tasks:

- Help to organize crew assignment to various work orders in the field and prioritize the work orders according to their degree of emergency/criticality.
- Make network events data available to experts to conduct in depth analysis and help in predefining patterns;
- Display the information about network failures in a fast and efficient way using GIS mapping.

There are several systems in the industry which offer efficient solutions for data acquisition and network modeling using advanced spatial modeling tools such as [Exfo, 2014], [JDSU, 2014] and [NTest, 2014], to name a few. However, they suffer from several drawbacks. For example, events are usually displayed separately to the end-user. There is no correlation between these events and they are not linked to previous fiber states. GIS systems are offered with an enriched semantic description of the telecommunication assets as well as the surrounding objects such as buildings and roads. However, the spatial information is only used for mapping purposes.

Events are located on the map and they are not qualified with regards to the surrounding objects. Hence, this increases the user's workload to analyze the data provided by various data sources.

We propose to use our framework to overcome the above mentioned limitations. Basically, our spatiotemporal pattern framework allows for managing the stream of events generated by OTDRs and for making the necessary correlations between different components of the environment to detect pattern instances. It also provides a rule-based engine to implement some business rules defined by the user in order to efficiently manage the crew assignment process.

### **5.6.3 Proposed Solution**

We proposed an architecture to implement the agent reasoning module which is illustrated by Figure 5.16. We define a *Crew Manager agent* which is responsible of detecting urgent situations related to the fiber optic network and which manages crew dispatching operations. The Crew Manager is a goal-based agent which uses pattern instances detected by the *Pattern Detection and Data Processing Module* and contextual information to make decisions and propose action plans to the human operator. The *inference engine* (IE) is a computer program that takes pattern instances from the pattern detection module and logically manipulates symbolic patterns (expressed in CG) using First Order Predicate Calculus. We use the Amine Prolog+CG language which is an object oriented and CG-based extension of Prolog. It supports CG operations and allows for loading the pattern instances from the knowledge base and uses rule-based functions. Human operators can use the IE answers to make decisions and intervene on the environment. Notice that the Crew Manager is also defined using a Prolog+CG program.

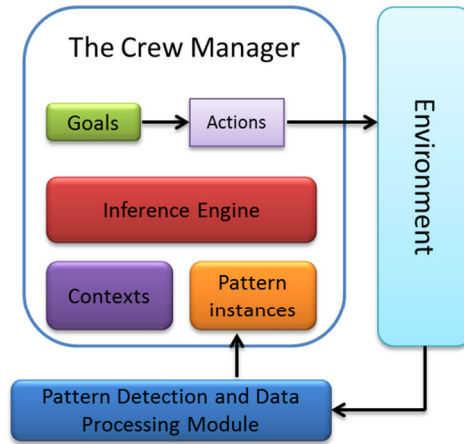


Figure 5.16: The Crew Manager's reasoning model

### 5.6.4 Spatial Environment

For illustration purposes, we use a fictitious fiber optic network in the city of Quebec. Three RTUs are deployed in the field and are used to monitor a number of fiber optic routes. Each fiber optic route is connected to one RTU. The fiber optic route (FOR) is a set of cable spans. Each cable span length is 4km and contains up to 144 fiber optic links. Therefore, a splice is necessary to connect all the cable spans and build the FOR. Each splice is represented by a splice point on the map. FORs can be aerial or underground. Two types of networks can be monitored. Urban networks concern cities with high density of population (also called *Metro networks*). Rural networks are networks which are deployed outside the city and which serve low densities of populations. It is also used to interconnect cities. The GIS also provides other information about the spatial environment such as Central Offices buildings where the crews are located, other buildings that are provisioned with fiber links and which need a special attention from the network engineers as they are considered in the SLA. A partial representation of a fiber optic network in the City of Quebec is depicted by Figure 5.17. A semantic abstraction of the spatial environment was manually created using Conceptual Graphs. Automating the abstraction process is out of the scope of this thesis. Several works exist in the literature to automate the

abstraction of a spatial environment such as [Mekni, 2010]. We use the Concept Type Lattice to represent spatial concepts according to their level of generality. Figure 5.18 illustrates a specialization example of the concept type Location.

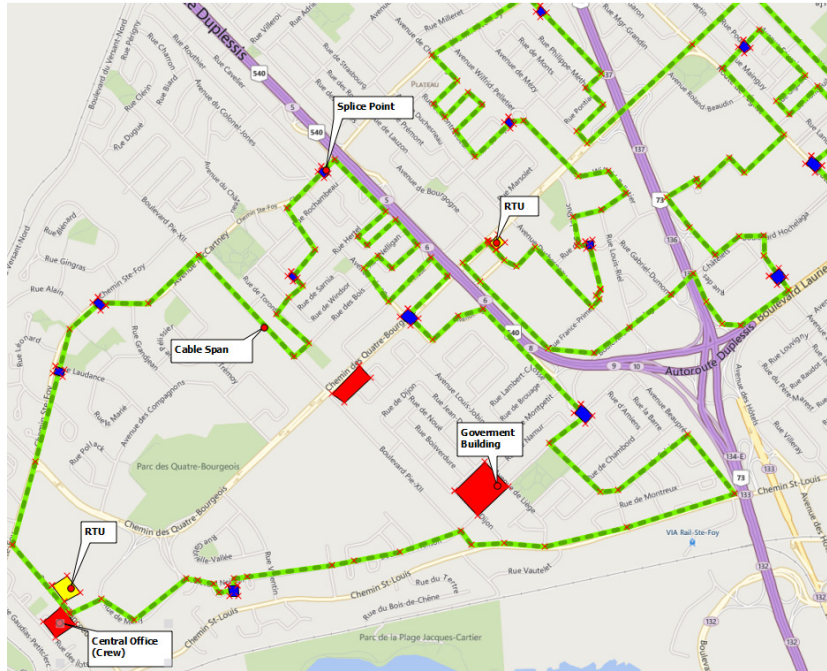


Figure 5.17: A metro network example in Quebec City. Fiber optic routes are represented by green lines and the RTU location is represented by a Yellow Square.

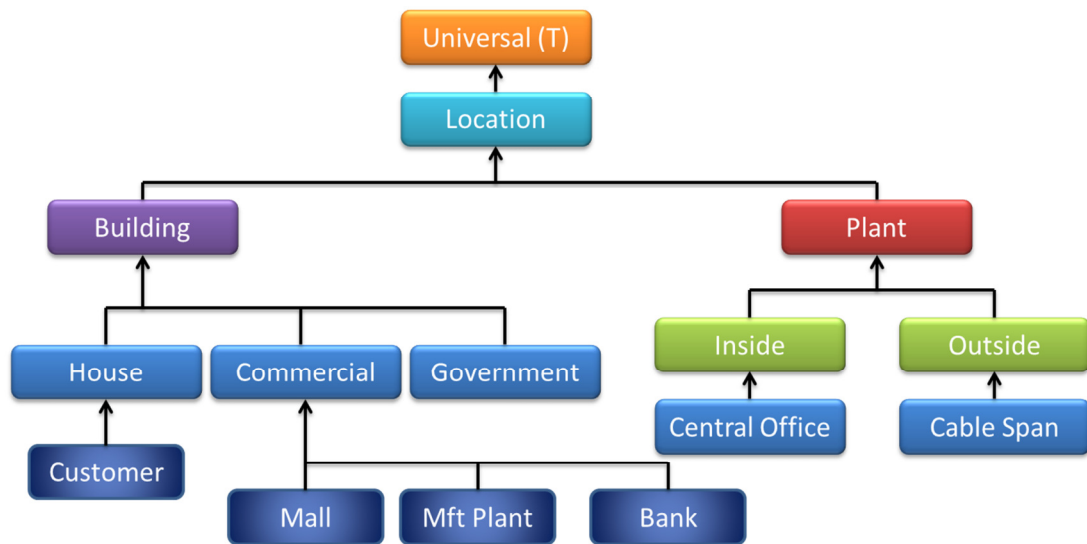


Figure 5.18: A specialization example of the concept type Location

### 5.6.5 Pattern Specification

In this section, we present some examples of simple and complex patterns that can be used in the RFTS application. We also present how contextual information can be integrated into pattern definitions and how it can be used to extend the pattern meaning. Current commercially available RFTSs detect events and display them to the end-user with additional information about the spatial environment. For example, the alarm report illustrated in Figure 5.19 shows some information about an event reported by the “RTU Paris” as described by one RFTS system available in the telecommunication market. Note the presence of two distances: a physical distance (“sheath distance”) and an optical distance. Physical distance is the distance of the FOR including cable spans, splices and slack loops. Optical distance is the distance measured by the OTDR. The optical distance is always greater than the physical distance and may vary slightly from one acquisition to another depending on the laser pulse. This makes the optical distance imprecise. Moreover, the spatial information is provided in this event by locating it relatively to the surrounding spatial references.

---

**Alarm Report**

---

Fiber Circuit Alarm for XZ-23H  
Possible Break Detected

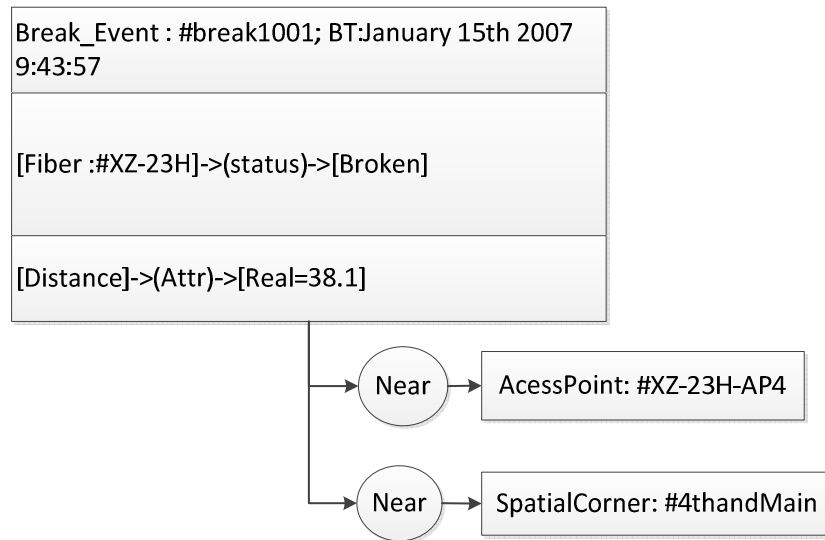
---

Fiber Circuit: XZ-23H  
Severity: Critical  
Date: 15 January 2007  
Time: 09:43:57 EST  
Affected Domains: Northern, Spring, HSBC  
Probe: Paris RTU  
Specific Problem: Bad Fiber Scan Analysis  
Probable Cause: Possible Break Detected  
Optical Distance: 40.2km  
Sheath Distance: 38.1km  
0.4km after “Corner of 4th and Main”  
0.2km before “access point XZ-23H-AP4”  
Latitude: 34 23' 43"  
Longitude: -105 43' 01"

*Figure 5.19: An alarm report about an event generated by the “RTU Paris” as described by [NTest. 2014] RFTS*

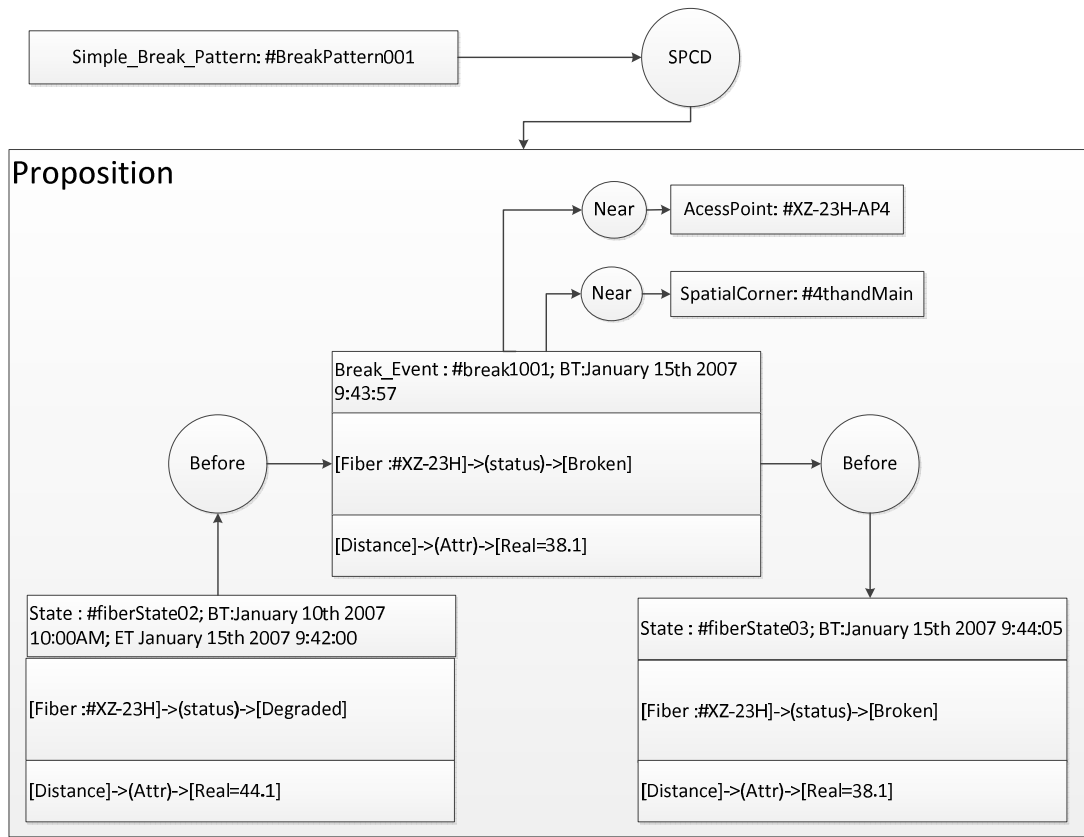
The alarm generated by “RTU Paris” can be represented using the graph depicted in Figure 5.20. We use the spatial relation “near” to qualify the distance between the

break location and the two spatial references “access point XZ-23H-AP4” and “Corner of 4<sup>th</sup> and Main”.



*Figure 5.20: A Conceptual Graph representation of the event reported by “RTU Paris”*

Using our pattern representation model, a user may configure a situation of interest related to this alarm. Figure 5.21 represents a simple pattern where the event type Break\_Event occurs on a fiber and changes its state from “degraded” to “broken”. This may help the user to conclude that the break occurred on the fiber due to a degradation problem. This is a typical use case in the fiber optic industry. As another example inspired from a real customer case in a telecommunication company that we visited in Bolivia, outside plant engineers noticed that a degradation event can occur on a fiber changing its state to “degraded” for a certain period of time. A break event can be followed changing the fiber state to “broken”. Further inspection after the detection of such a pattern revealed that tropical spiders gnaw these fibers and caused these damages.



*Figure 5.21: An example of a simple pattern where the event occurrence changes the state of a fiber from degraded to broken*

Another way to represent the example of fiber optic networks in Bolivia is to correlate simple patterns. Such a correlation generates a complex pattern and can be used to inform the user that a fiber degradation event was followed by another degradation event that occurred near the same location. Such a situation of interest can be expressed by a complex pattern illustrated in Figure 5.22. Note that we used a spatial relation to correlate simple patterns. The detection of the Fiber Inspection Pattern will trigger a fiber inspection maintenance task and a crew will be assigned to it.

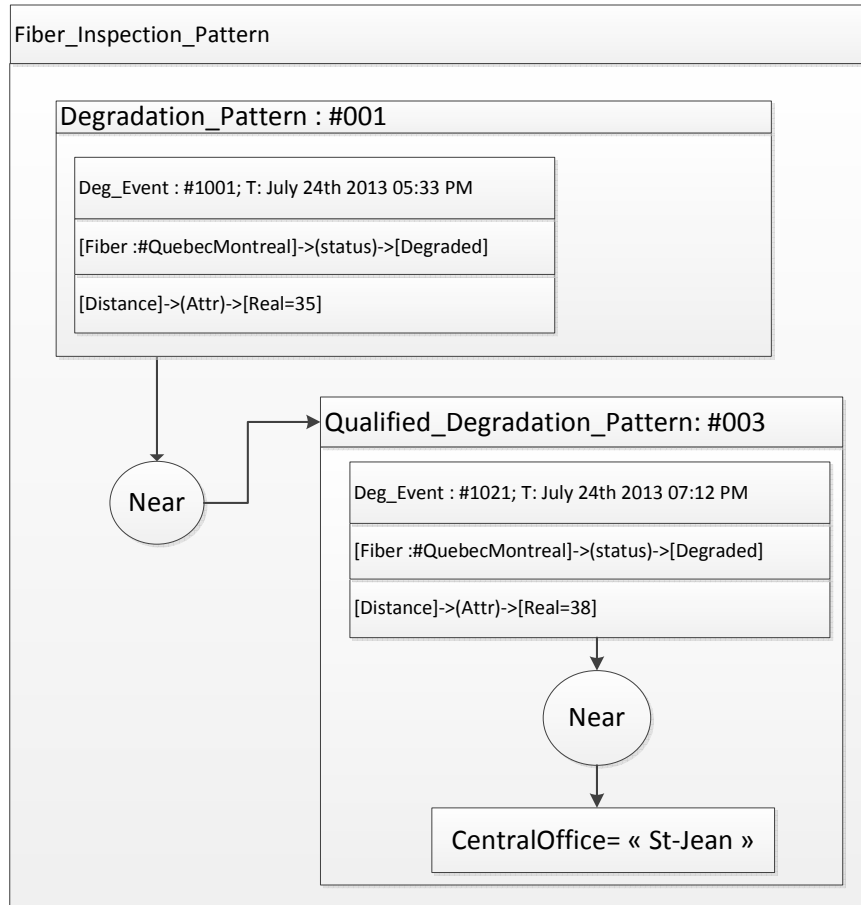


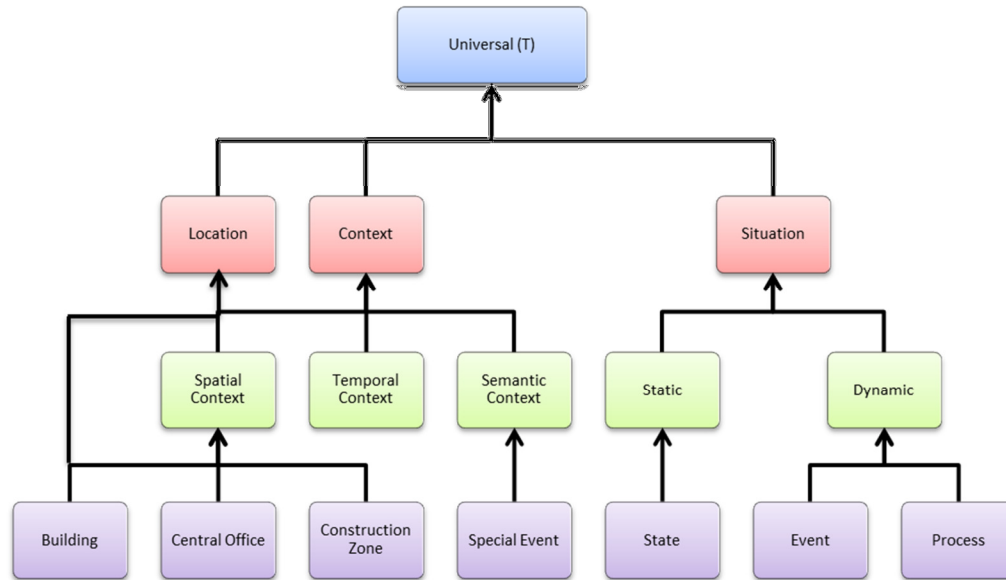
Figure 5.22: An example of a complex pattern linking two simple patterns

### 5.6.6 Using Contextual Information

Context is defined as additional information in the perceptual/cognition level which allows “perceiving or recognizing something that is actually not there” [Toussaint, 1978]. In our approach, contextual information is related to a specific spatiotemporal pattern. A different contextual information may change a pattern meaning. Therefore, contextual information is represented in the same knowledge structure as the spatiotemporal pattern. Contextual information is defined at two levels: at concept type lattice level where it can be used in the pattern definition and at the reasoning level where context instances can be linked to pattern instances which are detected by the Pattern Detection Module (see Figure 5.9). For our case study, the contextual information is represented by a Concept Type Lattice where three main types of



contexts are defined: spatial context, temporal context and semantic context. Each context type can be specialized in different sub-types (Figure 5.23).

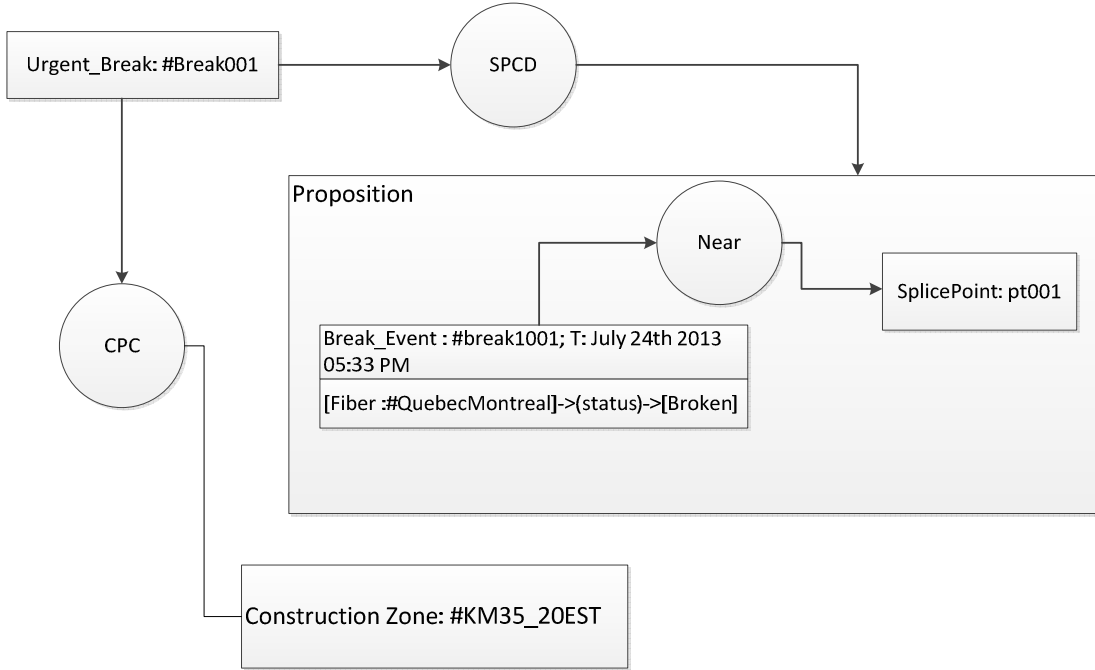


*Figure 5.23: A concept type lattice representing contextual information*

Typically, a fiber break event may be caused by construction works. During their activities, workers may damage the underground fibers since they do not have any knowledge about their presence. Therefore, a “construction zone” can be defined as a spatial contextual information and can be associated with a simple pattern describing a fiber break or degradation event. Hence, other events reported by the system in the same construction zone will get a similar meaning and the management team will coordinate with construction workers to better manage their activities and prevent such damages in the future (Figure 5.24).

The second example uses semantic contextual information to define emergency patterns. A break event can occur on a fiber which serves a given region. An amphitheater which hosts popular events (music shows, sport competitions) belongs to this region. If the break event occurs during an event hosted by the building, the situation is critical. The problem should be fixed to guarantee a continuous

transmission from/to the amphitheater. Figure 5.25 illustrates how a semantic context can be used in a simple pattern.



*Figure 5.24: A simple pattern example with spatial contextual information*

The characteristics of a crew can be defined in the knowledge base using CGs. Each crew can be located in a central office; it can belong to an area and it can have a state. A crew can be “Free” if it is located in the central office. It can be “Assigned” when it is working to repair a fiber optical link. Both Crew characteristics can be expressed in Prolog+CG as follows:

```

cg( [Crew = "Team1"] -loc-> [Area = "Beauport"] ).
cg( [Crew = "Team2"] -loc-> [Area = "SainteFoy"] ).

cg( [Crew = "Team1"] -attr-> [CrewStatus = "Free"] ).
cg( [Crew = "Team2"] -attr-> [CrewStatus = "Assigned"] ).

```

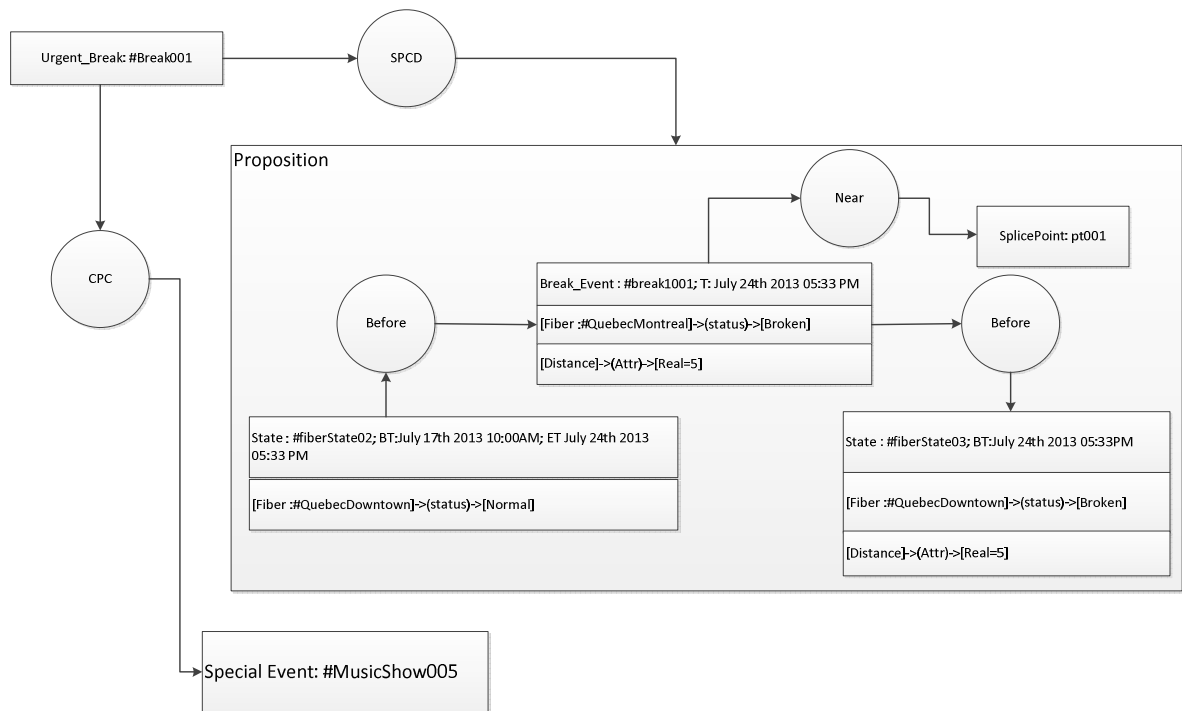


Figure 5.25: A simple pattern example with semantic contextual information

Each Remote Test Unit (RTU) used to test the fibers can be represented in the knowledge base using the following statement:

```
cg( [RTU = "sensor001"] -
    -loc-> [Area = "Beauport"],
    -attr-> [SerialNumber = "001" ]).
```

The Crew Manager's goals are defined using simple and complex patterns. For example the following complex pattern describes a situation of interest where a fiber break changed the status from broken to true and where the contextual information indicates that a crew is assigned to the related break area.

//Complex pattern to detect the recovery of a break configuration that occurred in a specific //area.

ComplexPatternWithContextInfo(\_CPattern,Environment,x)

```
:-cg([Complex_Pattern=_CPattern]-
Pattern_Description>[Proposition:[Event=[Break=_brk1]-
-EPC->[Proposition: [RTU=_sens1]-attr->[FiberState=Broken]]
-ESA->[Area=_Area]]
-before->[State=[NormalState]-SPC->[Proposition:[RTU]-attr-
->[FiberState]]]
-before->[Event=[Repair]-
-EPC->[Proposition: [RTU=_sens1]-attr->[FiberState=Rapaired]]
-ESA->[Area=_Area]]]),
//Contextual Information
cg([Complex_Pattern = _CPattern]-
-Context_Description->[Proposition : [Crew=x]-attr->[CrewStatus="Assigned"]]).
```

The relation *EPC* stands for *Event Propositional Content*. The relation *ESA* stands for *Event Spatial Attribute*. The relation *SPC* stands for *State Propositional Content*. These relations are part of the event definition as introduced in Chapter 3. The detection of an instance of this complex pattern means that a broken fiber has been repaired by the assigned crew according to the observation of the RTU (*\_sens1* in the above example).

When a fiber break is repaired, a Crew Manager agent can also revoke the crew and update its status to *Free* using the following Prolog+CG instruction:

```
RevokeCrew:-ComplexPatternWithContextInfo(_CPattern,Environment,x),
write(_CPattern), writeln(" pattern detected"),
```

```

write("Crew revoked from :"), writeln(x),
UpdateCrewStatus(x,"Free"),
getCrewStatus(_CPattern,x,_status),
write("New "),write(x),write(" "), write("status is "), writeln(_status).

```

A human operator may have his own interpretation of spatiotemporal situations of interest in a certain context. We used the above examples to show the integration of contextual information in the pattern definition. Using Prolog+CG a user can easily specify her own business rules by combining patterns instances and contextual information.

## 5.7 Conclusion

In this chapter, we presented a framework to manage qualitative spatiotemporal patterns. This framework takes advantage of the qualitative spatiotemporal formalism introduced in Chapter 3 and the qualitative spatial relations model introduced in Chapter 4. The framework allows for detecting patterns from large datasets found in real-time applications. It also helps (software and human) agents to reason about the detected patterns since they are expressed using Conceptual Graphs. A case study from the telecommunications industry has been presented to illustrate the usefulness of the framework. Three papers based on this chapter have been published in three conferences covering various domains such as Spatial Analysis [Barouni and Moulin, 2012] and Smart Grids [Barouni and Moulin, 2014c] and [Barouni and Moulin, 2015b]. This emphasizes the multidisciplinary aspect of our work. Another research paper is under preparation and will be submitted to the Big Data Applications journal by July 2015. The next chapter discusses the limits related to our approach and concludes this thesis.

# **Chapter 6**

## **Conclusion**

The main objective of this thesis was to propose a novel approach for qualitative representation and management of dynamic spatiotemporal situations of interest. In this chapter, we conclude the thesis by summing up the goals that have been achieved with respect to our initial objectives. We also present an overview of some future research topics that may represent prospective opportunities of our approach.

### **6.1 Synthesis**

In this thesis, we proposed a novel approach to manage dynamic spatiotemporal situations of interest. Managing situations of interest is made possible thanks to the qualitative representation of their definitions, to the detection of their instances from a stream of data and to offering reasoning capabilities close to the user's mental model. Our approach is defined in the context of Big Data applications and large scale data acquisition and monitoring systems routinely used in the industry. This objective has been achieved throughout the following stages.

First, we proposed a novel approach to qualitatively represent dynamic spatiotemporal situations of interest. We based our approach on the notion of spatiotemporal pattern which is widely present in the spatiotemporal analysis domain. We used Conceptual Graphs to qualitatively represent pattern definitions and we integrated the concept of contextual information in the pattern model in order to allow agents to reason about these patterns.

Second, we proposed a novel approach to automate reasoning about spatial proximity. Our model is based on contextual information which can be either objective such as spatial distance or subjective such as traffic conditions or the user's familiarity with the city. We integrated a neuro-fuzzy classifier to train a dataset input by an expert and to generate fuzzy quantifiers that are used by a Fuzzy Inference System to qualify spatial proximity relations. As a concrete result of this approach, we integrated our qualitative proximity tool in a GIS and we made qualitative spatial relations available to our qualitative pattern model.

Third, we proposed a framework to manage spatiotemporal patterns occurring in a dynamic environment. This framework is based on a hybrid architecture where Conceptual Graphs are used to: 1) qualitatively represent pattern definitions; 2) build the knowledge base of software agents. We developed an extension to a Complex Event Processing engine to detect pattern instances from a stream of data generated by distributed sensors and to enable agents' qualitative spatial reasoning capabilities. Detected pattern instances can be populated in knowledge bases and integrated in agents' reasoning models.

We proposed two case studies to illustrate different aspects related to our framework. In the first case study, agents use pattern definitions to make decision about different outage configurations which are generated by an Outage Management System. In the second case study, agents use detected patterns to assign crews to intervene over a fiber optic network.

## **6.2 Contributions**

The first contribution of this thesis is a cognitive approach which allows for qualitative representation and management of situations of interest using spatiotemporal patterns. Modeling situations of interest has been addressed by several

research communities such as data mining, Complex Event Processing or Artificial Intelligence. For example [Fisher et al, 2014] proposed a mathematical model to recognize situations of interest in surveillance applications and [Baader et al, 2009] proposed a query-based approach to retrieve situations of interest from situation awareness systems. There are also works in the spatiotemporal analysis domain which tend to use query-based approaches to find patterns that represent situations of interest such as the work of [Sakr and Guting, 2014]. These works are limited with respect to their capability to represent complex and dynamic phenomena and do not provide a relevant approach to enable agents to reason about them. We also use the concept of patterns to represent situations of interest. However, our approach is different since we use Conceptual Graphs as a powerful representation formalism close to Natural Language. Furthermore, our pattern definition goes beyond event-based patterns as it is used in CEP approaches since it supports the representation of dynamic phenomena and integrates contextual information. The resulting model provides a cognitive flavor that attempts to better reflect human's mental model about the environment in the agent's reasoning model.

The second contribution of this thesis is a qualitative proximity tool based on contextual information. We integrate contextual information in a qualitative proximity model and we automate the generation of spatial proximity quantifiers. While other works such as [Yao and Thill, 2007] used an approximation function (i.e. ANFIS) and did not integrate their approach in a GIS, our work uses a neuro-fuzzy classifier and a Fuzzy Inference System to qualify proximity relations. Another original aspect of this contribution is that it has been used to extend a Complex Event Processing Engine to support qualitative spatial relations in pattern definitions.

The third contribution of this thesis is a qualitative pattern management framework based on a hybrid architecture. We integrate our pattern definition of dynamic situations of interest in a CEP engine that we extended in order to facilitate the use of our pattern model in large scale monitoring applications. There are several original aspects in this contribution. First, it enhances the semantic capabilities of the CEP



engine by using Conceptual Graphs to represent patterns instead of using SQL-based languages. Second, we use a cognitive approach to integrate contextual information in the pattern definition. While current CEP solutions use context to manage the stream of events [Etzion and Zolotorvesky, 2010], contextual information in our model is used by agents to create their own interpretation of detected patterns. Third, this contribution is part of recent efforts existing in the research community to address a new generation of *semantic CEP* or *knowledge-based CEP* [Teymourian and Paschke, 2010] where knowledge bases are used to enrich event patterns such as works proposed by [Zhou et al, 2012],[Anicic et al, 2010] and [Teymourian, 2014]. Our approach is different from the aforementioned works for different reasons. We use Conceptual Graphs to represent patterns whereas [Zhou et al, 2012] use query-based approaches, [Anicic et al, 2010] use a predicate-based language to represent event patterns and offers a limited number of temporal operators. [Teymourian, 2014] and [Anicic et al, 2010] used so called event patterns where events are the main component of patterns. Our approach provides an adequate expressiveness to describe the semantics of events, states, temporal and spatial concepts to extend the semantic capabilities of spatiotemporal patterns and to represent real complex phenomena. Our framework provides an abstraction layer higher than spatiotemporal data in raw format. Users no longer need to be familiar with advanced computer languages and can focus on contents rather than on data format as discussed by [Kuhn and Ballatore, 2015].

The knowledge base CEP proposed by [Teymourian, 2014] queries ontologies to extend event semantics during the pattern matching process. This affects the pattern engine capabilities, since an external data source (the ontology) must be accessed during the event processing. We propose a different approach which enhances the semantics of patterns *a priori* (i.e. before processing). This approach reduces the processing time drastically since the pattern detection engine preloads pattern definitions converted from Conceptual Graphs to an enriched version of EPL.

## 6.3 Limits and Drawbacks

Our approach to manage situations of interest can be enhanced in different ways. A temporary limit concerns the proposed pattern definition to represent situations of interest. Indeed, our pattern model is based on the concepts of states and events to represent dynamic situations. The notion of process is not yet integrated in our model. This limits the expressiveness capabilities of patterns to represent certain complex phenomena that may be identified in applications such as video surveillance and weather monitoring, to name a few. Supporting the notion of process in the pattern definition will be a significant contribution to the CEP community where, to the best of our knowledge, this notion is not clearly defined yet.

While spatial relations are defined in our pattern model, our framework only supports fuzzy spatial proximity. The other qualitative spatial relations such as topology are supported using classical spatial tools currently available in commercial solutions. Using fuzzy qualitative topological and directional relations needs to be addressed in the near future in order to better reflect the human's mental model of the spatial environment. The same limit applies to temporal relations. Our pattern model uses the temporal model proposed by [Moulin, 1997] and may be extended in the future to support fuzzy temporal relations.

Owing to time constraints, the pattern management framework presented in Chapter 5 does not fully integrate all pattern types proposed in Chapter 3. We only developed algorithms to detect simple and complex pattern types from a stream of data. Moreover, the agent's reasoning module is called from an external module and it is not completely integrated in the pattern management framework. The integration of both modules will create new research opportunities which will be detailed in the next section.

Finally, although our approach enables agent's reasoning about patterns, we did not explore in detail advanced reasoning mechanisms. For example, we did not address the aspect of deriving new patterns from the knowledge base or the implementation of complex transactional actions.

## 6.4 Future Work

Besides improvements to overcome the limits of our approach which are mentioned in Section 6.3, we mention here several opportunities that can be explored in future works.

A possible research opportunity is to implement *pattern verification mechanisms to discover patterns* that will be never detected from large systems due to a lack of consistency [Anicic et al, 2010]. Doing so, a user could be notified when some defined patterns are not detected after a certain period of time. Other patterns can be suggested to the user in this case. The concept of fuzzy patterns can be a good option where pattern components are defined with a certain degree of uncertainty.

There are also issues related to the implementation of advanced reasoning mechanisms such as causality relations between patterns, reasoning with course of actions [Haddad, 2009] and automating the execution of actions generated by agent's deduction modules (such as in Event Condition Action architectures). In current *Event Condition Action systems*, events are used to trigger the evaluation of a condition and to execute an action. Patterns (which are based on a set of events) may be used to trigger the evaluation of a condition and execute actions.

Another research topic that may be explored as an application of our approach concerns Smart Cities. *Smart Cities* is a research topic which has emerged in recent years as a result of the increasing number of applications and connected devices generating temporal and spatial data in the urban environment. Cities can be "smart"

when there are tools and solutions that integrate and synthesize data in a way that improves efficiency and sustainability [Batty et al, 2012]. In this context, our approach can offer a good option to address several issues such as traffic management, smart grid (energy efficiency and demand response) and human mobility (modeling trajectory patterns followed by humans during their daily activity). Through the use of real-time systems and sensors, data are collected from citizens and objects and then processed in real-time [Wikipedia, 2015]. Our pattern model could be used to define patterns to help governments and cities make decisions in several areas.

In *Demand Response applications, Advanced Metering Infrastructures* (AMI) are deployed in customer facilities and they provide real time data about power consumption. Some countries such as Italy and Sweden have 100% of houses using this emerging technology.

Demand response applications can be characterized by a dynamic environment due to the variation of residential power consumption in a specific city according to several patterns such as touristic activities during hot seasons or people migration. Several works attempted to identify *peak load patterns* such as [Simmhan et al, 2011]. ]. However, these works are based on data mining techniques and SQL-based patterns. In such applications, our pattern model would offer enough expressiveness to represent complex situations in such dynamic environment with spatial and temporal dimensions. It could also be used to detect patterns in real time using a CEP.

## Bibliography

- Adi, A., Biger, A., Botzer, D., Etzion, O., Sommer, Z., (2003). *Context awareness in Amit*. Proceedings of the Autonomic Computing Workshop, pp.160-166
- Aigong, X. and Lakmal, A. H. (2009). *Conceptual Framework For Spatio-Temporal Process Model*. Proceedings of International Cartographic Conference, Santiago, Chile.
- Allen, J.F. (1983). *Maintaining Knowledge about Temporal Intervals*. ACM Common, Vol. 26 (11), 1983, pp. 832–843
- Anagnostopoulos, C.B., Ntarladimas, Y. and Hadjiefthymiades, S. (2006). *Situation Awareness: Dealing with Vague Context*, Pervasive Services, in ACS/IEEE International Conference, pp.131-140
- Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N. and Studer, R. (2010). *ETALIS: Rule-Based Reasoning in Event Processing*, Chapter in Reasoning in Event-based Distributed Systems, Series in Studies in Computational Intelligence, Sven Helmer, Alex Poulovassilis and Fatos Xhafa editors. Springer.
- Arioua, A., Tamani, N. and Croitoru, M. (2014). *On conceptual graphs and explanation of query answering under inconsistency*. In Graph-based representation and reasoning, edited by N. Hernandez, R. Jaesche, and M. Croitoru, Proceedings of the International Conference of Conceptual Structures (ICCS), July 27-30, 2014, Iași, Romania, LNCS Vol. 8577, pp 51-64
- Baader, F., Bauer, A., Baumgartner, P., Cregan, A., Gabaldon, A., Ji, K., Lee, K., Rajaratnam, D. and Schwitler, R. (2009). *A Novel Architecture for Situation Awareness Systems*, Proceedings of the 18th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, July 06-10, 2009, Oslo, Norway, pp.77-92
- Batty, M., Axhausen, K.W., Giannotti, F., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G., Portugali, Y. (2012). *Smart cities of the future*. European Journal of Physics Special Topics. Vol. 214(1), pp 481-518

- Barouni, F. and Moulin, B. (2012). *An Extended Complex Event Processing Engine to Qualitatively Determine Spatiotemporal Patterns*. In Proceedings of Global Geospatial Conference, June 2012, Quebec City, Canada
- Barouni, F. and Moulin, B. (2014a). *A Framework for Qualitative Representation and Reasoning about Spatiotemporal Patterns*. In Graph-based representation and reasoning, edited by N. Hernandez, R. Jaesche, and M. Croitoru, Proceedings of the International Conference of Conceptual Structures (ICCS), July 27-30, 2014, Iași, Romania, LNCS Vol. 8577, pp 79-91
- Barouni, F. and Moulin, B. (2014b). *An Intelligent Spatial Proximity System Using Neurofuzzy Classifiers and Contextual Information*. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XL-2, pp.107-114
- Barouni, F. and Moulin, B. (2014c). *Using Complex Event Processing to Manage Patterns in Distribution Networks*. CIRED Workshop, June 11-12, 2014, Rome, Italy
- Barouni, F. and Moulin, B. (2015a). *An Intelligent Spatial Proximity System Using Neurofuzzy Classifiers and Contextual Information*. Geomatica Special Issue, Advances in Geospatial Statistical Modeling, Analysis and Data Mining. Vol. 69(2), to Appear in September 2015
- Barouni, F. and Moulin, B. (2015b). *Spatiotemporal Pattern Detection in Outage Management Systems*. Power and Energy Automation Conference, March 10-12, 2015, Spokane, Washington, USA
- Bastien, C. (1998). *Contexte et situation*. Dictionnaire des Sciences Cognitives. Paris: PUF.
- Bazire, M. and Brézillon, P. (2005). *Understanding Context Before Using It*. Modeling and Using Context. Lecture notes in computer science. Springer-Verlag, Vol. 3554, pp. 29-40
- Bera, R. and Claramunt, C. (2003). *Topology-based proximities in spatial systems*. Journal of Geographical Systems, Springer-Verlag, Vol. 5, pp. 353-379
- Bermingham, L. and Lee, I. (2014). *Spatio-temporal Sequential Pattern Mining for Tourism Sciences*. Procedia Computer Science, Vol. 29, pp. 379-389, ISSN 1877-0509,
- Borkar, P. and Malik, L.G., (2013). *Acoustic signal based traffic density state estimation using adaptive Neuro-Fuzzy classifier*. IEEE International Conference on Fuzzy Systems, pp.1-8

- Brennan, J. and Martin, E. (2006). *Membership functions for spatial proximity*. In Advances in Artificial Intelligence, edited by Sattar, A., Kang, B.H., Lecture Notes in Artificial Intelligence, Springer Verlag, Berlin, Vol. 4304, pp. 942–949
- Brennan, J. and Martin, E. (2012). *Spatial proximity is more than just a distance measure*. International Journal of Human Computer Studies. Vol. 70(1), pp. 88–106
- Brezillon, P. (1999). *Context in Artificial Intelligence: II, Key elements of context*. Computing and informatics, Vol. 18(5)
- Chakravarty, P., Wickramasekara, M.G. (2014). *A better GIS leads to a better DMS*. Power Systems Conference (PSC), Clemson University, pp.1-5
- Cascado, D., Sevillano, J. L., Fernandez-Luque, L., Johan-Grøttum, K., Vognild, L. K. and Burkow, T. M. (2011). *Standards and Implementation of Pervasive Computing Applications, in Pervasive Computing and Networking*, edited by M. S. Obaidat, M. Denko and I. Woungang, John Wiley & Sons, Ltd, Chichester, UK. doi: 10.1002/9781119970422.ch9
- Cena, G., Bertolotti, I.C., Tingting Hu, Valenzano, A., (2014). *Design, verification, and performance of a MODBUS-CAN adaptation layer*. 10th IEEE Workshop on Factory Communication Systems (WFCS), pp.1-10
- Cetişli, B. and Barkana, A. (2010). *Speeding up the scaled conjugate gradient algorithm and its application in neuro-fuzzy classifier training*. Soft Computing. Vol. 14(4), pp. 365-378.
- Chen, G. and Kotz, D. (2000). *A survey of context-aware mobile computing*. Research. Technical report. Dartmouth College Hanover, NH, USA
- Chittaro, L. and Montanari, A. (2000). *Temporal representation and reasoning in artificial intelligence: issues and approaches*. Annals of Mathematics and Artificial Intelligence. Vol. 28, pp. 47–106.
- Cohn, A. and Renz, J. (2007). *Qualitative Spatial Reasoning*. Handbook of Knowledge Representation, edited by Van Harmelen, F., Lifschitz, Frank V. and Porter, B., Elsevier
- Cohn, A.G, Bennett, B., Gooday, J. and Gotts, N.M. (1997). *Qualitative Spatial Representation and Reasoning with the Region Connection Calculus*. GeoInformatica, 1997, Vol. 1(3), pp-275
- Cole, S. and Hornsby, K. (2005). *Modeling Noteworthy events in a geospatial domain*. Lecture Notes in Computer Science. Vol. 3799, pp. 77-89

- ConEdison. (2013). <https://www.comed.com/layouts/comedsp/OutageMap.aspx> [Accessed in 2013].
- Cugola, G. and Margara, A. (2012). *Processing flows of information: From data stream to complex event processing*. ACM Computer Surveys. Vol. 44(3), Article 15
- Desclés, J.-P. (1985). *Représentation des connaissances: Archétypes cognitifs, schèmes conceptuels et schémas grammaticaux*. Actes Sémiotiques, Paris, EHESS, Documents VII, pp. 69-70
- Desclés, J.-P. (1994). *Quelques Concepts Relatifs au Temps et à l'Aspect pour l'Analyse des Textes*. Stanislaw et al. (Eds.) : Études Cognitives: Sémantique des Catégories d'aspect et de Temps, Vol. 1, pp. 57-88
- Devaraju, A. and Kauppinen, T. (2012). *Sensors tell more than they sense: Modeling and reasoning about sensor observations for understanding weather events*. Special Issue on Semantic Sensor Networks, International Journal of Sensors, Wireless Communications and Control
- Dey, A.K. (2001), *Understanding and using context*. Personal and Ubiquitous Computing. Vol. 5(1), pp. 4-7
- Do, Q. H. and Chen, J. F. (2013). *A Neuro-Fuzzy Approach in the Classification of Students' Academic Performance*, Computational Intelligence and Neuroscience. Vol. 2013 Article ID 179097
- Elgendy, N. and Elragal, A. (2014). *Big Data Analytics: A Literature Review Paper*. Advances in Data Mining. Applications and Theoretical Aspects Lecture Notes in Computer Science Vol. 8557, pp. 214-227
- Endsley, M. R. (1988). *Design and evaluation for situation awareness enhancement*. In Proceedings of the Human Factors Society 32<sup>nd</sup> Annual Meeting Santa Monica, CA: Human Factors Society, pp. 97-101
- Endsley, M. R., Bolte, B., and Jones, D. G. (2012). *Designing for situation awareness: An approach to human-centered design*. London: Taylor & Francis.
- Erwig, M. (2004). *Toward spatiotemporal patterns*. In: Caluwa, R, Tré, G, Boudogua, G eds. (2004) *Toward spatiotemporal patterns, spatio-temporal databases*. Springer-Verlag, New York, pp. 29-54
- Esper. (2015). <http://esper.codehaus.org/> [accessed in 2015]
- Etzion, O. and Zolotorvesky, N. (2010). *Spatial Perspectives in Event Processing*. In From active data management to event-based systems and more, Edited by Kai



Sachs, Ilia Petrov, and Pablo Guerrero. Springer-Verlag, Berlin, Heidelberg, pp. 85-107

Exfo. (2014). <http://exfo.com> [accessed in 2014]

Fischer, Y., Reisch, A. and Beyerer, J. (2014). *Modeling and recognizing situations of interest in surveillance applications*. IEEE International Inter-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), pp. 209-215

Foa. (2015). *The Fiber Optic Association*. Thefoa.org, 2015, [www.thefoa.org](http://www.thefoa.org). [Accessed in 2015].

Forbus, K. (2007). *Qualitative modeling*. In F. Harmelen, V. Lifschitz, & B. Porter (Eds.), *Handbook of knowledge representation*. Elsevier, New York, pp. 361-394

Freksa, C., Klippel, A. and Winter, S. (2007). *A cognitive perspective on spatial context*. In *Spatial cognition: Specialization and Integration*. Edited by Anthony G. Cohn and Christian Freksa and Bernhard Nebel. Vol. 05491, ISSN 1862-4405

Gahegan, M. (1995). *Proximity Operators for Qualitative Spatial Reasoning*. In *COSIT '95 Proceedings: Spatial Information Theory: A Theoretical Basis for GIS*, Edited by A. U. Frank and W. Kuhn. Berlin, Germany: Springer-Verlag, pp. 31-44

Galton, A. (2004). *Fields and Objects in Space, Time, and Space-time*. *Spatial Cognition and Computation*, Vol. 4, pp. 39-68

Gcep. (2014). [www.surna.org](http://www.surna.org) [Accessed in 2014]

Gehrke, J., Lattner, A. and Herzog, O. (2005). *Qualitative Mapping of Sensory Data for Intelligent Vehicles*. Workshop on Agents in Real-Time and Dynamic Environments at the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), U. Visser, G. Lakemeyer, G. Vachtesevanos and M. Veloso, Eds., Edinburgh, UK, pp. 51-60

Gliwa, B. and Byrski, A. (2013). *Hybrid Neuro-Fuzzy Classifier Based on Nefclass Model*. *Computer Science*, Vol. 12, ISSN 2300-7036.

Grenon, P. and Smith, B. (2004). *SNAP and SPAN: Towards Dynamic Spatial Ontology*. *Spatial Cognition and Computation*, Vol. 4 (1), pp. 69-103

Guesgen, H.W. (2002). *Reasoning About Distance Based on Fuzzy Sets*. *Applied Intelligence*, Vol. 17, pp. 265-70

- Guesgen, H.W. and Albrecht J. (2000), *Imprecise reasoning in geographic information systems*. Fuzzy Sets and Systems (Special Issue on Uncertainty Management in Spatial Data and GIS), Vol. 113, pp. 121-131
- Guesgen, H.W. and Marsland, S. (2010). *Spatio-temporal reasoning and context awareness*. Handbook of Ambient Intelligence and Smart Environments, pp. 609-634
- Haddad, H. (2009). *Une approche pour supporter l'analyse qualitative des suites d'actions dans un environnement géographique virtuel et dynamique*. L'analyse « What-if » comme exemple, PhD Thesis, Laval University, Quebec, Canada.
- Haddad, H. and Moulin, B. (2010). *A Framework to Support Qualitative Reasoning about COAs in a Dynamic Spatial Environment*. Journal of Experimental & Theoretical Artificial Intelligence, Taylor & Francis, Vol. 22(4), pp. 341-380
- Hadji, M., Kollios, G. and Bakalov, P. (2005). *Complex Spatio-temporal pattern Queries*. In Proceedings of the 31st international conference on Very large data bases (VLDB '05). Trondheim, Norway, VLDB Endowment pp. 877-888.
- Hasan, S., O'Riain, S. and Curry, E. (2012). *Approximate semantic matching of heterogeneous events*. In Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems (DEBS '12). ACM, New York, NY, USA, pp. 252-263.
- Helmer, S. Poulouvassilis, A. and Xhafa, F. (2011). *Reasoning in Event-Based Distributed Systems*. Studies in Computational Intelligence. Springer, Vol. 347, pp. 99-124
- Holzmann, C. (2007). *Rule based reasoning about qualitative spatiotemporal relations*. In Proceedings of the 5th international workshop on Middleware for pervasive and ad-hoc computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference (MPAC '07). ACM, New York, NY, USA, pp. 49-54
- Holzmann, C. and Ferscha, A. (2010). *A framework for utilizing qualitative spatial relations between networked embedded systems*. Journal of Pervasive and Mobile Computing. Vol. 6(3), pp. 362-381
- Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B., and Koldehofe, B. (2013). *Opportunistic spatio-temporal event processing for mobile situation awareness*. In Proceedings of the 7th ACM international conference on Distributed event-based systems (DEBS '13). ACM, New York, NY, USA, pp. 195-206.
- IEEE. (2010). *IEEE Standard for Electric Power Systems Communications -- Distributed Network Protocol (DNP3)*," IEEE Std 1815-2010 , pp.1-775

- Imfeld, S. (2000). *Time, Points and Space - Towards a Better Analysis of Wildlife Data in GIS*. Dissertation, University of Zurich
- Jakobson, G., Buford, J. and Lewis, L. (2007). *Situation Management: Basic Concepts and Approaches*. In Information Fusion and Geographic Information Systems, 2007, pp. 18-33.
- JDSU. (2014). <http://jdsu.com> [accessed in 2014]
- Kabbaj, A. (2006). *Development of intelligent systems and multi-agents systems with Amine platform*. In Henrik Schärfe, Pascal Hitzler, and Peter Øhrstrøm editors, ICCS'06: Proceedings of the 14th International Conference on Conceptual Structures, Lecture Notes in Computer Science, Springer Verlag, Vol. 4068, pp. 286-299.
- Kamsu-Foguem, B., Tchuenté-Foguem, G. and Foguem, C. (2014). *Using conceptual graphs for clinical guidelines representation and knowledge visualization*. Information Systems Frontiers Vol. 16(4), pp. 571-589
- Kettani, D. and Moulin, B., (1999). *A spatial model based on the notion of spatial conceptual map and of object's influence areas*. In Spatial Information Theory Cognitive and Computational Foundations of Geographic Information Science, edited by Mark, D.M., Freksa, C., LCNS, Springer Verlag, Berlin, Vol. 1661, pp. 401-416.
- Kowalski, R. (1992). *Database updates in the event calculus*. Journal of Logic Programming. Vol. 12(162), pp. 121-46
- Kowalski, R. and Sergot, M. (1986). *A Logic-Based Calculus of Events*. New Generation Computing, Vol. 4, pp. 67-95
- Kuhn, W. and Ballatore, A. (2015). *Designing a Language for Spatial Computing*. In AGILE 2015, Lecture Notes in Geoinformation and Cartography
- Lattner, A., Miene, A., Visser, U. and Herzog, O. (2006). *Sequential Pattern Mining for Situation and Behavior Prediction in simulated Robotic Soccer*. Lecture notes in Computer Science. Springer. Vol. 4020. pp. 118-129
- Lin, L., Gong, H., Li, L. and Wang, L. (2009). *Semantic event representation and recognition using syntactic attribute graph grammar*. Pattern Recognition Letters, Vol. 30(2), pp. 180-186, ISSN 0167-8655
- Lind, J. (2003). *Patterns in agent-oriented software engineering*. Lecture Notes in Computer Science, Vol. 2585, pp. 47-58.

- Luckham, D.C. (2002). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley.
- McCarthy, J., and Hayes, P. (1969). *Some philosophical problems from the standpoint of artificial intelligence*. Machine Intelligence 4, Edinburgh U. Press.
- McCarthy, J. (1993). *Notes on formalizing context*. In Proceedings of IJCAI'93 Conference, Chambéry, France.
- Mekni, M. (2010). *Automated Generation of Geometrically-Precise and Semantically-Informed Virtual Geographic Environments Populated with Spatially-Reasoning Agents*. Ph.D. Thesis, Laval University, Quebec, Canada.
- Miller, R. and Shanahan, M. (1999). *The event-calculus in classical logic - alternative axiomatizations*. Electronic Transactions on Artificial Intelligence, Vol. 3(1), pp. 77-105
- Moulin, B. (1997). *Temporal contexts for discourse representation: An extension of the conceptual graph approach*. Applied Intelligence, Vol. 7, pp. 225-227
- Nauck, D. and Kruse, R. (1999). *Neuro-fuzzy systems for approximation functions*. Fuzzy sets and systems, Vol. 101, pp. 261-271.
- Negnevitsky, M. (2011). *Artificial Intelligence: A Guide to Intelligent Systems*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, ISBN 0201711591
- NTest. (2014). <http://www.ntestinc.com/> [Accessed in 2014]
- Obaidat M.S. and Woungang. I. (2011). *Pervasive Computing Systems*. Pervasive Computing and Networking (eds M. S. Obaidat, M. Denko and I. Woungang), John Wiley & Sons, Ltd, Chichester, UK.
- OGC. (2015). *Open Geospatial Consortium*. [www.opengeospatial.org](http://www.opengeospatial.org) [Accessed in 2015]
- Political Psychology. (2003). *Special issue on Neuroscientific contributions to political psychology*, *Political Psychology*, Vol. 24(4)
- Ponchon, J., Champavere, A., (2011). *PON test systems - From theory to field deployments*. Optical Fiber Communication Conference and Exposition (OFC/NFOEC), and the National Fiber Optic Engineers Conference, pp.1-3, 6-10
- Renz, J. (2002). *Qualitative Spatial Reasoning with Topological Information*. Vol. 2293, Springer: Berlin, Germany

- Renz, J. and Mitra, D. (2004). *Qualitative direction calculi with arbitrary granularity*. In: Zhang, C., Guesgen, H.W., Yeap, W.-K. eds. (2004) PRICAI 2004: Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zealand, Lecture Notes in Computer Science, Vol. 3157, pp. 65-74, Springer, Berlin
- Resch, B., Lippautz, M. and Mittlboeck, M. (2010). *Pervasive Monitoring - A Standardized Sensor Web Approach for Intelligent Sensing Infrastructures*. Sensors. Special Issue "Intelligent Sensors 2010", pp. 11440-11467.
- Robinson, V. B. (1990). *Interactive Machine Acquisition of a Fuzzy Spatial Relation*. Computers and Geosciences, Vol. 16(6), pp. 857-72.
- Rulecore. (2013). [www.rulecore.com](http://www.rulecore.com) [accessed in 2013]
- Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, ISBN 0-13-604259-7
- Saha, D. and Mukherjee, A. (2003) *Pervasive computing: a paradigm for the 21st century*. IEEE Computer Vol. 36, pp. 25-31
- Sakr, M. and Guting, R. (2010). *Spatiotemporal Pattern Queries*. Geoinformatica, Vol. 15(3), pp. 497-540
- Sakr, M. and Guting, R. (2014). *Group Spatiotemporal Pattern Queries*. Geoinformatica, Vol. 18(4), pp. 669-746
- Schultz, C.P.L., Guesgen, H.W. and Amor, R. (2007). *A System for Querying With Qualitative Distances in Networks*. Fuzzy Systems Conference. FUZZ-IEEE, London, United Kingdom, pp. 1-6
- SECONDO web site. (2013). <http://dna.fernuni-hagen.de/Secondo.html/> [Accessed in 2013]
- Shanahan, M. (1997). *Solving the frame problem: A mathematical investigation of the common sense law of inertia*. MIT Press, ISBN 0-262-19384-1
- Shanahan, M. (1999). *The Event Calculus Explained*. In Artificial Intelligence Today edited by Wooldridge, M., & Veloso, M., Lecture Notes in Artificial Intelligence. Vol. 1600, pp. 409-430.
- Sharon, G. and Etzion, O. (2008). *Event Processing Networks - model and implementation*. IBM System Journal, Vol. 47(2), pp. 321-334
- Sowa, J.F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Massachusetts.

- SpatialRules. (2013). [www.objectfx.com](http://www.objectfx.com) [accessed November 2013]
- Sun, C.-T. and Jang, J.-S. (1993). *A neuro-fuzzy classifier and its applications*. Fuzzy Systems, Vol. 1, pp. 94-98.
- Tan, Y., Vuran, M. C. and Goddard, S. (2009). *Spatio-temporal event model for Cyber-Physical Systems*. In Proc. of 29th IEEE International Conference on Distributed Computing Systems Workshops, pp. 44-50
- Teymourian, K. and Paschke, A. (2010). *Enabling knowledge-based complex event processing*. In Proceedings of the 2010 edbt/icdt workshops, New York, NY, USA, 2010, pp 1-7.
- Teymourian, K. (2014). *A Framework for Knowledge-Based Complex Event Processing*, Ph.D. Thesis, University of Berlin, Germany.
- Toussaint, G. T. (1978). *The use of context in pattern recognition*. Pattern Recognition. Vol. 10, pp. 189-204.
- Vieira, M.R., Frías-Martínez, E., Bakalov, P., Frías-Martínez, V. and Tsotras, V.J. (2010). *Querying Spatio-temporal Patterns in Mobile Phone-Call Databases*. Eleventh International Conference on Mobile Data Management (MDM), May 2010, pp. 239-248
- Wolter, D. and Wallgrün, J. O. (2013). *Qualitative Spatial Reasoning for Applications: New Challenges and the SparQL Toolbox*. In Geographic Information Systems: Concepts, Methodologies, Tools, and Applications, Hershey, PA: Information Science Reference, pp. 1639-1664.
- Worboys, M. F. (1996). *Metrics and topologies for geographic space*. In Advances in Geographic Information Systems Research II: Proceedings of the International Symposium on Spatial Data Handling, Delft, Netherlands, pp. 7A.1-7A.11
- Worboys, M.F., Duckham, M. and Kulik, L. (2004). *Commonsense notions of proximity and direction in environmental space*. Spatial Cognition and Computation. Vol. 4(4), pp. 285-312
- Yao, X. and Thill, J.-C. (2007). *Neurofuzzy Modeling of Context-Contingent Proximity Relations*. Geographical Analysis, Vol. 39, pp. 169-194.
- Zadeh, L (1965). *Fuzzy sets*. Information and Control, Vol. 8, pp. 338-353
- Zhou, Q., Simmhan, Y. and Prasanna, V. (2012). *Incorporating semantic knowledge into dynamic data processing for smart power grids*. In International Semantic Web Conference, Lecture Notes in Computer Science, Vol. 7650 (2012), pp. 257-273

## Appendix

The Crew Dispatcher Agent structure is defined in three main sections: knowledge, goals and actions. The knowledge section contains the contextual information related to the crew status. The following code snippet gives an example of an agent instantiation with PROLOG+CG language and the agent's knowledge structure.

```
//We define Bob as a CrewDispatcher Agent
CrewDispatcher( Bob).

//Agent's knowledge.

//Contextual information.
cg([Pattern] -Context_Description->[Proposition : [Crew="Team2"]-attr->[CrewStatus="Free"]]).
cg([Pattern] -Context_Description->[Proposition : [Crew="Team1"]-attr->[CrewStatus="Assigned"]]).
```

Agent's goals are defined using spatiotemporal patterns. The following code snippet gives an example of a pattern definition.

```
//Complex pattern to detect the recovery of an outage configuration happened on a specific area.
ComplexPatternWithContextInfo(_CPattern,Environment,x)
:-cg([Complex_Pattern = _CPattern] -Pattern_Description->[Proposition : [Event=[Outage=_out1]-
-EPC->[Proposition: [OutageSensor=_sens1]-attr->[OutageState=_st1]]
-ESA->[Area=_Area]]
-before->[State=[NormalState]-SPC->[Proposition:[OutageSensor]-attr->[OutageState]]]
-before->[Event=[Outage=_out2]-
-EPC->[Proposition: [OutageSensor=_sens2]-attr->[OutageState=_st1]]
-ESA->[Area=_Area]]]),
cg([Complex_Pattern = _CPattern] -Context_Description->[Proposition : [Crew=x]-attr->[CrewStatus="Assigned"]]).
```

The environment is configured through a separate PROLOG+CG instance. It contains static information such as crews and sensor locations and dynamic information such as events generated by sensors. The following code snippet illustrates an example of the environment's configuration.

```

//Configuration of the environment.

cg( [Crew = "Team1"] -loc-> [Area = "Beauport"] ).
cg( [Crew = "Team2"] -loc-> [Area = "SainteFoy"] ).

cg( [Crew = "Team1"] -attr-> [CrewStatus = "Free"] ).
cg( [Crew = "Team2"] -attr-> [CrewStatus = "Free"] ).

cg( [OutageSensor = "sensor001"] -
    -loc-> [Area = "Limoilou"],
    -attr-> [SerialNumber = "001" ] ).

cg( [OutageSensor = "sensor002"] -
    -loc-> [Area = "Limoilou"],
    -attr-> [SerialNumber = "002" ] ).

cg( [OutageSensor = "sensor003"] -
    -loc-> [Area = "Limoilou"],
    -attr-> [SerialNumber = "003" ] ).

cg( [OutageSensor = "sensor004"] -
    -loc-> [Area = "Beauport"],
    -attr-> [SerialNumber = "004" ] ).

```