Behaviour Recognition on Noisy Data-streams Constrained by Complex Prior Knowledge

Romain Rincé

▶ **To cite this version:**

Romain Rincé. Behaviour Recognition on Noisy Data-streams Constrained by Complex Prior Knowledge. Modeling and Simulation. Université de nantes, 2018. English. NNT : . tel-01983311

HAL Id: tel-01983311
https://theses.hal.science/tel-01983311

Submitted on 16 Jan 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE NANTES

ONERA
THE FRENCH AEROSPACE LAB

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE NANTES
COMUE UNIVERSITÉ BRETAGNE LOIRE
ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par
Romain RINCÉ

Behaviour Recognition on Noisy Data-streams Constrained by Complex Prior Knowledge

Thèse présentée et soutenue à Palaiseau le 7 Novembre 2018
Unité de recherche : LS2N/Duke et ONERA/MIDL
Thèse N° :

Rapporteurs avant soutenance :

Thomas Schiex	Directeur de recherche à l'INRIA MIAT
Audine Subias	Maître de conférences au LAAS-CNRS

Composition du Jury :

Présidente :	Céline Rouveirol	Professeur des universités au LIPN
Examinateur :	Benoit Delahaye	Maître de conférences au LS2N
Membre invité :	Jean-Lou Farges	Ingénieur à l'ONERA
Dir. de thèse :	Philippe Leray	Professeur des universités au LS2N
Enc. de thèse :	Romain Kervarc	Ingénieur à l'ONERA

Thèse

en vue d'obtenir le grade de

Docteur de l'Université de Nantes
pour la spécialité informatique

préparée à
l'Office National d'Études et de Recherches Aérospatiales

au titre de l'école doctorale
Mathématiques et Sciences de Technologies
de l'Information et de la Communication

présentée et soutenue publiquement par

Romain Rincé

le 7 Novembre 2018

Behaviour Recognition on Noisy Data-streams Constrained by Complex Prior Knowledge

Jury

Céline Rouveirol	Professeur des universités	Présidente du Jury
Thomas Schiex	Directeur de recherches	Rapporteur
Audine Subias	Maître de conférences	Rapporteuse
Benoit Delahaye	Maître de conférences	Examinateur
Jean-Lou Farges	Ingénieur ONERA	Membre invité
Philippe Leray	Professeur des universités	Directeur de thèse
Romain Kervarc	Docteur ingénieur	Encadrant ONERA

Il vaut mieux mobiliser son intelligence sur des conneries
que mobiliser sa connerie sur des choses intelligentes.

Proverbe Shadok

Remerciements

Bien que cette thèse soit l'aboutissement d'un travail personnel effectué durant trois années, sa réalisation n'aurait pu être possible sans l'aide technique et scientifique de nombreuses personnes ainsi que le soutien moral de mon entourage durant toute cette période.

Si le bon déroulement d'une thèse est influencée par de nombreux facteurs, il est indéniable que les encadrants y jouent un rôle prépondérant. À ce titre, je ne peux donc que remercier mon directeur de thèse, Philippe Leray*, pour sa disponibilité et sa réactivité à toutes mes questions ainsi que mon encadrant à l'ONERA, Romain* Kervarc*, pour son écoute, sa compréhension et les corrections d'articles suintantes de rouge. Conjointement, je souhaite les remercier pour la confiance et la grande liberté qu'ils m'ont données tout au long de mon travail. Merci de m'avoir donné l'opportunité d'initier une thèse et de m'avoir accompagné jusqu'à sa conclusion.*

Je souhaite aussi remercier les membres de mon comité de suivi de thèse : Céline Roueirol*, qui fut aussi la présidente de mon jury, et Karim* Tabia*. Bien que je n'ai eu la chance d'échanger avec eux qu'épisodiquement, il·elle·s ont été, à chaque occasion, force d'encouragement et de conseil. Le dernier chapitre de cette thèse n'aurait sans doute pas existé sans eux.*

Je tiens à exprimer ma gratitude aux autres membres de mon jury de soutenance : Jean-Lou Farges*, qui a accepté d'en faire partie et qui a suivi la plupart de mes travaux, Benoit* Delahaye*, qui a par ailleurs pris le temps de relire et commenter mon manuscrit, et plus particulièrement Thomas* Schiex *et Audine* Subias *qui ont accepté·e·s d'être les rapporteur·euse·s de ma thèse. Je ne peux que saluer le temps qu'il·elle·s ont déployé pour lire ma thèse et y apporter une quantité phénoménale de commentaires.*

Je remercie sincèrement les membres de l'unité MIDL de m'avoir intégré à leur équipe. En particulier, je souhaite remercier Michaël Lienhardt *qui, bien qu'arrivé tardivement, m'a permis d'échanger sur le sujet des chroniques. Son arrivée fut pour moi une réelle bouffé d'air dans un département tel que le DTIS qui manquait cruellement de personnes à même de discuter de mon sujet de recherche. Je tiens à exprimer toute ma gratitude à Stéphanie* Lala *d'avoir pris le temps de relire et commenter ma thèse, mais plus généralement pour la gentillesse et la*

grande bienveillance qu'elle fît montre à mon égard. Bien que ne faisant pas directement partie de l'unité MIDL, je remercie infiniment notre secrétaire de département, Florence Marie*, qui, au-delà d'être toujours de bonne humeur et compréhensive, est d'une efficacité redoutable pour résoudre tout problème administratif qui vient régulièrement ponctuer la vie de doctorant. Je pense pouvoir affirmer sans hésiter que l'ensemble des doctorant·e·s du DTIS lui sont reconnaissant·e·s.*

Mes remerciements ne seraient être complets sans évoquer les doctorant·e·s et stagiaires de l'ONERA qui furent en quelque sorte une seconde famille. Des moments aussi simples que les soirées jeux, les escape games, les pauses-cafés de lassitude des vendredis après-midi, ou les discussions interminables entre deux étages à refaire le monde ont été indispensables au maintien de ma bonne santé mentale. Je m'excuse auprès de tous ceux que je ne citerai pas directement, mais je leur prie de croire que j'ai apprécié tous les moments passés ensemble, qu'ils furent courts ou non.

À cet égard, je souhaite porter une mention spéciale à quelques-uns d'entre-eux. À Émilien Flayac*, qui bien que passant son temps à «* chercher la merde *», est régulièrement venu m'aider à combler mes lacunes en mathématiques. À Nathan* Michel *(prononcez Mitchell), qui trouvait le temps, parfois, de venir à l'ONERA entre deux anniversaires de son frère. À mes deux voisins de bureau, Ali* Hebbal *et Capitaine* Pelamatti*, pour avoir supporté les cris provenant du bureau jouxtant le leur, mais aussi pour les nombreuses discussions, sérieuses ou «*trollesques *», venues divertir mes journées à l'ONERA. Et à Elinirina* Robinson*, qui derrière une attitude discrète, fait preuve d'un esprit brillant et caustique que j'ai eu le plaisir de découvrir.*

Pour conclure ces remerciements, je voudrais adresser ce paragraphe aux personnes qui, bien que n'ayant nul rapport avec le sujet technique de cette thèse, auraient par leur absence entrainé la fin prématurée de celle-ci.

Je souhaite donc remercier en premier lieu Vincent Chabridon *qui, pendant ces trois années, fut un co-bureau exceptionnel et un soutien indéfectible. Les moments à échanger sur la sociologie, l'histoire, ou la politique ont été salvateur contre la colère, la frustration et la morosité qu'ont pu amener ce travail. Et bien que nos prises de becs furent épiques, elles ne sont qu'à la hauteur de l'affection que je lui porte.*

Je souhaite aussi remercier Georgia Effgen Santos *qui, paradoxalement à la distance de plus de 8 000 km qui nous séparait, a été l'une des personnes les plus présentes durant ces années. Elle m'a soutenu tout au long de cette thèse et m'a permis de supporter mon quotidien dans une ville que j'ai toujours haïe. Il est indéniable que sans elle cette thèse n'aurait abouti à*

rien d'autre qu'une dépression. Je lui suis extrêmement reconnaissant, et bien que ne pouvant sans doute jamais compenser ce qu'elle m'a offert, je tiens à ce qu'elle sache qu'elle sera toujours accueillie à bras ouverts par moi et ma famille.

Enfin, il me paraît naturel de terminer en remerciant mes deux parents, Anabel et Manuel, qui me supportent, dans tous les sens du terme, depuis bien plus longtemps que ces quelques années de thèse. Je ne voudrais pas que ces simples mots concluant mes remerciements soient dévaluées de par l'incroyable banalité qu'ils peuvent revêtir car, si je ne devais citer que deux personnes à remercier, ce ne pourrait être qu'eux. Je leur dois tout et je les aime.

Contents

Introduction **13**

1 Complex event processing in the nutshell **17**
1.1 Main goals of Complex Event Processing 18
1.1.1 Semantics 18
1.1.2 Interesting properties of CEP systems 20
1.2 A brief state of the art on complex event processing 21
1.2.1 Event calculus 21
1.2.2 SASE 23
1.2.3 TESLA 27
1.2.4 ONERA Chronicles 29
1.3 Drone model 33
1.3.1 Communication loss between three agents during a drone flight 34
1.3.2 Problem model: telecommand and radio losses 35

2 Complex Event Processing under uncertainty **41**
2.1 Understanding uncertainty on Complex Event Processing 42
2.1.1 Data Uncertainty 42
2.1.2 Pattern uncertainty 44
2.1.3 Understanding of goals and concepts of Complex Event Processing under uncertainty 46
2.2 Markov Logic Approaches 48
2.2.1 Event calculus using Markov Logic Networks 48
2.2.2 Interval approaches 51
2.2.3 Ad-hoc approaches 53
2.2.4 Summary on Markov Logic Networks approaches 54
2.3 Bayesian network approaches 56

2.4 Automaton-based methods . . . 60
2.4.1 SASE Approaches . . . 61
2.4.2 Automaton-based representations from Albanese et al. and its extensions . . . 64
2.4.3 Automaton-based representation from Fazzinga et al. and its extensions . . . 66
2.5 Other approaches . . . 68
2.5.1 ProbLog method . . . 69
2.5.2 HMM . . . 69
2.5.3 Petri-net approaches . . . 71
2.5.4 Grammar approaches . . . 72
2.6 Conclusion . . . 75

3 Background **77**
3.1 Markov Logic Networks . . . 78
3.1.1 Markov network . . . 78
3.1.2 Markov Logic . . . 79
3.1.3 Inference . . . 81
3.1.3.1 MAP inference . . . 81
3.1.3.2 Conditional inference . . . 83
3.2 ProbLog . . . 86
3.3 Non-homogeneous Markov models . . . 90
3.3.1 Context . . . 90
3.3.2 Construction of the NHM . . . 91

4 Chronicle representation with uncertainty using Markov logic networks **93**
4.1 Chronicles expressed with logical rules . . . 94
4.1.1 Chronicle semantic in Markov logic . . . 95
4.1.2 Data stream structure and LLEs dependencies . . . 96
4.1.3 Chronicle rules . . . 97
4.1.4 Operator rules . . . 98
4.1.5 Conclusion . . . 101
4.2 Specific logical structures leading to intractable problems for WalkSAT 102
4.2.1 A first example of unexpected behaviour with MC-SAT inference 102
4.2.2 WalkSAT efficiency analysis on specific problems . . . 104
4.3 Consequences of WalkSAT inference on the chronicle model for MLNs . 115
4.4 Conclusion . . . 117

5 Exact computation of uncertainty for chronicles using ProbLog 119
5.1 Difference in semantic interpretation with Markov logic networks . . . 120
5.2 A chronicle representation with ProbLog . . . 123
5.2.1 Event uncertainty . . . 123
5.2.2 Chronicle representation . . . 124
5.2.3 Conclusion . . . 126
5.3 Instantiation of models under uncertainty using the ProbLog chronicle model . . . 126
5.3.1 The hurricane model . . . 127
5.3.2 The drone model . . . 130
5.4 Conclusion . . . 132

6 Chronicle inference on complex systems using Markov chains and automata 133
6.1 Uncertainty estimation using Markov chains . . . 134
6.1.1 Initial approach . . . 134
6.1.2 Application on a toy example . . . 135
6.1.3 Discussion . . . 138
6.2 Expressing high-level constraints using chronicles . . . 139
6.2.1 Evidence behaviours expressed as sub-streams . . . 140
6.2.2 Graph representations of constraint sub-streams . . . 142
6.2.3 Sampling using constraint graphs . . . 144
6.2.4 Finding the set of constraint sub-graphs for a chronicle . . . 145
6.2.5 Discussion . . . 147
6.3 Speeding up inference computation . . . 148
6.3.1 Precisions about the chronicle representation . . . 148
6.3.2 Model-checking approach for chronicle inference . . . 149
6.3.3 Fast computation of the automaton of a chronicle evidence . . . 151
6.4 Probability Estimation of a Behaviour of the Drone Model . . . 151
6.4.1 Drone model to NHM . . . 151
6.4.2 Results and Analysis . . . 153
6.5 Conclusion . . . 156

Conclusion 159

A Definition of chronicle with the Markov logic semantic 163
A.1 Binary operators . . . 164
A.2 Unary operators . . . 164

B Drone model in ProbLog **167**
B.1 Temporal part 167
B.2 Subsystems part: Exemple from the RPS system 168
B.3 Uncertainty part 174

Bibliography **177**

Acronyms **190**

List of Tables **191**

List of Figures **193**

Introduction

INCE the rise of the *Information Age*, the number of communication exchange in the world increased continuously, and exploded with massive computerisation and the expansion of Internet. Data is flourishing through every aspect of our society. More than 200 000 SMS are sent every second, administration procedure are replaced by HTML forms, fast-trading is supplanting any trader, more than 100 000 planes take off every day, data centers use more than 5% of the worldwide consumption. Data is everywhere, but data is chaotic. Produced by billions of agents continuously, understanding and extracting sense from this data is a necessity for efficiency and security reasons. For instance, cyberattacks are a serious concern for enterprises or nations, but are not easily detected and require detection of complex patterns in a huge amount of data.

Analysis of data is a wide domain that covers many approaches and concepts. Neural networks, for instance, are notably efficient to identify relations over data and produce deductions from it, but lack of techniques explaining the decisions made by it. In contrast, Complex Event Processing (CEP) is a domain that try to identify behaviours in data using logical models describing them. Most of the time, data is organised temporally and behaviours are described with specific organisation of small observable activities named events. This domain for data analysis is well covered, but, usually, does not consider that data gathered on system might be erroneous, missing or corrupted. Unfortunately, erroneous data might be a considerable trouble for identification of behaviours by missing their apparition on data, or detecting false behaviours. Both situations might be undesirable or dangerous. For instance, not detecting an intrusion on the computer system of a company might results in stolen information. Similarly, detecting a behaviour that does not exist might activates counteracting measures with damageable consequences. For instance, detecting erroneously missile strikes during a warfare situation may lead to unjustified reprisal measures.

This thesis takes place in this context where data is not totally reliable but it is still necessary to analysis it to identify a specific behaviour. Furthermore, this thesis aims to extend a specific representation for CEP, named *chronicles* [Pie14], to be robust to uncertainty. Literature on data uncertainty in CEP may focus on two different aspects. The first one considers the uncertainty on the values from data, while the second aspect consider the existence of information in data. For instance, supposing a radar detects that a car passes at high speed, first approach would consider uncertainty on the value of the car speed, while the second approach would be interested to determine if a car passed effectively. These two aspects, which are usually named respectively attributes and events uncertainty, might be correlated, but not necessarily. In this thesis, we focused on event uncertainty since a main aspect of the chronicle representation focuses on the relative positions of events between them, meaning that existence of these events is an important requirement for the identification of behaviours with chronicles.

For this work, we aimed to produce an extension for the chronicle system allowing uncertainty representation and estimation of recognition regarding data provided with three properties as goals:

Scalability The extension should be able to be applied on realistic scenarii that might potentially have numerous events provided on the data with realistic chronicles to recognise. Optimally, uncertainty computation should be performed online.

Precision of computation The extension should assert that the chronicle recognition probability is correct regarding a model, or at least being able to choose the precision of the evaluation.

High-level prior information Usually, information provided in the data (events) are relatively low level information easy enough to recognise and capture. This information necessarily impacts the probability computation of the chronicle recognition. Nonetheless, it would be useful to be able to define, before inference, rules that should be asserted. For instance, it might be possible to know that a known behaviour occurred during the production of the data, but without necessarily knowing when it happened precisely.

First part of Chapter 1 is a brief introduction to CEP, its particularities, and its goals and presents at the same time the common vocabulary used by the community. The second part presents some CEP representations selected for their extensions to uncertainty excepts for the chronicle model that we aim to extend. Finally, the main case study of a drone model, used in this thesis, is introduced.

Chapter 2 introduces the main concept of uncertainty in CEP and presents the different aspects and general representations of uncertainty that appear in the literature. State of the art is then detailed regarding different types of methods such as Markov Logic Network (MLN), Bayesian networks or automaton-based.

Chapter 3 presents the different tools and methods that have been used for different approaches of uncertainty representations in this work.

Chapter 4 introduces our first experimentations of a chronicle representation robust to uncertainty based on MLNs. In the first part, chronicle representation with the Markov logic semantic is expressed as well as the uncertainty representation. The second part discusses the results inconsistencies obtained with inference and investigates on the possible explanations leading to these results. In particular, a set of experiments is conducted on the algorithm WalkSAT, which is the core algorithm used for inference with MLNs, to evaluate its efficiency on specific problems related to our chronicle representation.

Chapter 5 presents a second approach solving data uncertainty for chronicles. Since MLNs use a method based on stochastic local search to provide satisfiable solution of First-Order Logic (FOL) problems, this approach is based on exhaustive search for SAT problems. Such approach requires using specific representations, like Binary Decision Diagrams (BDDs), to solve a problem. Transformation from a logical problem to this kind of representation is performed with ProbLog. Last part of this chapter presents the application of this representation to the drone model.

Finally, Chapter 6 presents, in the first part, an original ad-hoc method based on Markov Chain Monte-Carlo (MCMC) with non-homogenous Markov models (NHMs) solving scalability issues that may appear with ProbLog. This method separates the uncertainty model from the computation of a chronicle recognition probability using data as constraints on the Markov model representing the system. The second part details how to express high-level constraints with this method using chronicle as prior knowledge thanks to transformations of chronicles into sub-graphs of constraints. Third part presents extensions for this approach for speeding up computation and, notably a model-checking approach for chronicle inference. Finally, last part details the application of this method to the drone model and discusses the issues remaining with such method.

CHAPTER 1

Complex event processing in the nutshell

Contents

1.1 Main goals of Complex Event Processing 18
1.1.1 Semantics 18
1.1.2 Interesting properties of CEP systems 20
1.2 A brief state of the art on complex event processing 21
1.2.1 Event calculus 21
1.2.2 SASE 23
1.2.3 TESLA 27
1.2.4 ONERA Chronicles 29
1.3 Drone model 33
1.3.1 Communication loss between three agents during a drone flight 34
1.3.2 Problem model: telecommand and radio losses 35

HIS chapter presents an overview on CEP. In the first two sections, we review the main goals and concepts of CEP without considering uncertainty yet. We present a small state of the art focusing on the approaches that

propose an extension to some form of uncertainty. It should not be seen as an exhaustive overview of existing formalisms, but as brief insight of CEP through specific formalisms that have been extended for uncertainty. The last section presents a case study of CEP that has been previously used in [Pie14] and that will be used later in this thesis.

In Section 1.1, we introduce the goals of CEP and interesting transversal properties. We also propose the semantics we chose in the scope of this thesis. Section 1.2 presents some existing works and semantics used by the CEP community. Section 1.3 presents the case study of a communication loss during a Unmanned Aerial Vehicle (UAV) flight.

1.1 Main goals of Complex Event Processing

CEP consists in analysing information provided in a partial or linear time ordering to recognise complex behaviours out of their trace. This information might be provided by different sources and is usually represented on *streams* of data. Data is generally annotated with occurrence times, consequently each piece of data associated to its time annotation is called an *event*. The main idea behind CEP is that events on data streams provide relative low levels of information, but they might hide more important information when combined in specific patterns, which can be automatically recognised from the data stream through computation. To distinguish these different degrees of complexity on information, events from the data stream may be referred to Low Level Events (LLEs) or Simple Derived Events (SDE), while recombined information make patterns that are called complex events (CEs) or High Level Events (HLEs). Unfortunately, among the community, lots of different terms exist that may easily be ambiguous. For instance, terms like *events*, *behaviours*, or *activities* may refer to both LLE and CE, which is literally due to a lack of terms to distinguish real activities and their equivalent computerized counterparts, both being referred to as *events*.

1.1.1 Semantics

For the sake of clarity, let us precise the semantics that will be used along this thesis. To this end, semantics, we will define it in parallel with a small illustration of a hurricane alarm (cf. Figure 1.1). Along with this model, a graphical representation of the semantic structure is provided on Figure 1.2. The hurricane alarm is a simple system made of an alarm that is supposed to ring when a hurricane appears. The formation of the hurricane or the alarm ringing are both states of the physical world during a

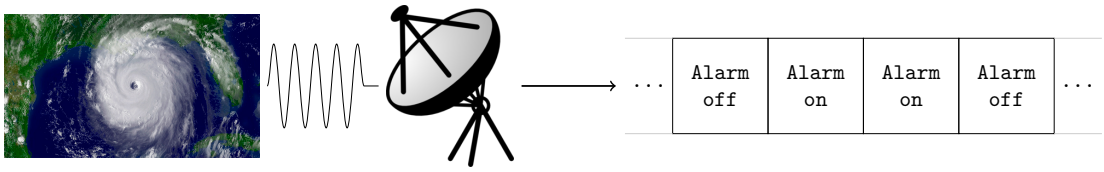


Figure 1.1: A simple hurricane alarm scenario producing a data-stream.

period of time, which we call *simple activities*. These simple activities might be combined or ordered together to form more complex activities that we call *behaviours*. For instance, in our model, if during a period of time the alarm was not triggered during a hurricane, it might be described as a *behaviour* as it is composed of simple activities. Both *simple activities* and *behaviours* might be referred as *activities* since they describe states of the world even if they might be at different scales.

The purpose of CEP is to represent and automatically process these concepts so all activities have their computerized version. Consequently, *simple events* (or *LLEs*) are the computerized representation of *simple activities* and *complex events* (or *CEs*) the computerized representation of *behaviours*. As for activities, both *complex events* and *simple events* might be referred to as *events*. Simple events are generally the pieces of data provided in the data stream. In our illustrative example, the data stream might be the recorded logs indicating when the alarm is either ringing or in stand-by and considered as a simple event. A complex event would be a combination of LLEs that try to model a behaviour using a specific language that describes the potential relations between events. For instance, with our model, previously described behaviour of alarm failure using an invented language might be represented as CE with `fail_alarm:` `hurricane and no alarm, between` t `and` t'. Generally, these CEsare reused to define new CEs and model more complex behaviours. For instance, `alarm_failure_lasts_3:fail_alarm lasts 3 min` models a behaviour where the alarm did not work for three consecutive minutes using the previously defined `fail_alarm`.

Both LLEs and CEs might embed attributes, that are values associated to the events. Conceptually, attributes are any kind of connected information observed at the time of an event. For instance, attributes for an event `alarm`, when an alarm triggers, might be numerical values like the wind speed at this moment, or non-numerical values like the ID of the alarm. These attributes might be used to define the CE of interest. Nonetheless, some approaches do not consider events with attributes and make only use of 0-arity events.

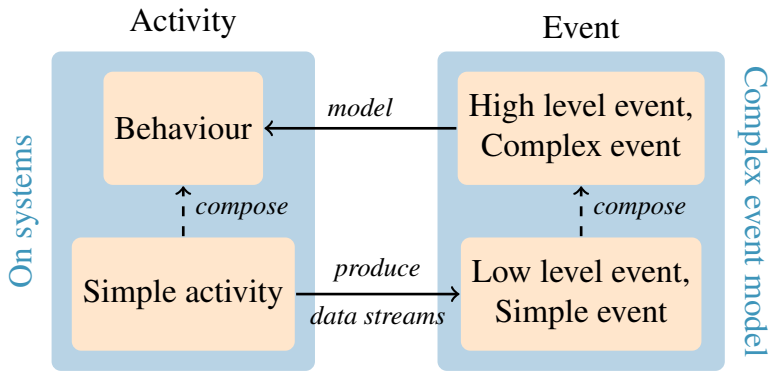


Figure 1.2: Structural representation of our semantic for activities and events.

1.1.2 Interesting properties of CEP systems

The main goal of CEP is to provide a formalism to express CEs and methods to recognise them from data streams. Relative to this main goal, the community often refers to different concepts of interest which may be seen as sub-goals:

Time representation Time might be seen differently along the community and, consequently, represented according to these points of view. In general, time is linear and may be discrete or continuous, events are represented as time points or intervals, but it might happen that some models do not describe time as a value on a clock and just uses a total or partial order between the events on streams.

Expressiveness It refers to the formalism used to describe CEs and the variety of possible combinations of behaviours to be extracted from data streams that it captures. Consequently, with sometimes very ad-hoc constructs, it may be hard to say if a formalism is more expressive than another and any choice will be indubitably subjective. Nonetheless, some constructs are very commonly found and it is possible to assess formalisms relatively to a standard basis of main operations such as that proposed by Alevizos et al. in [Ale+17] to evaluate :

- *Sequence:* Two events following each other in a given time order.
- *Disjunction:* Either of two events occurring, regardless of temporal relations.
- *Iteration:* An event occurring N times in sequence, where $N \geq 0$.
- *Absence:* Absence of event occurrence.
- *Selection:* Select those events whose attributes satisfy a set of predicates/-relations, temporal or otherwise.

- *Projection:* Return an event whose attribute values are a possibly transformed subset of the attribute values of its sub-events.
- *Windowing:* Evaluate the conditions of an event pattern within a specified time window.

In this case, the choice of operations to define expressiveness is clearly oriented towards time representation and attributes management which are two main goals generally pursued in CEP. However, sometimes only one of the two goals is sought, in which case it is possible to choose a set of operations focusing more on the relevant aspect.

Efficiency Efficiency usually refers to computation speed. It is an important goal as many models try to analyse data online or in real-time. Even when analysis is not supposed to be performed online, CEP goal would still be to process large amount of data in an acceptable amount of time, which is basically the same. In general, the speed of a model is expressed in LLEs parsed per second, rather than just the computation time.

Multiplicity Most CEP formalisms do not just focus on answering whether a specific behaviour exists in a stream, but will also memorise all occurrences of it. Multiplicity may be important when a CE is dependent on multiple intermediate CEs. On this objective, every recognition should be memorised, even if restrictions may be specified to help computation. For instance, windowing, that introduces a notion of decay on the events which removes the oldest events that might be irrelevant, is commonly used with this purpose.

Historisation Historisation is somewhat related to multiplicity as it is the ability to track all the events that appears in a recognition of a CE. As multiple CE occurrences may appear on the data, they are not composed of the same LLEs. Historisation keeps tracks on which LLE leads to a CE occurrence.

1.2 A brief state of the art on complex event processing

1.2.1 Event calculus

Event Calculus (EC), introduced by Kowalski and Sergot [KS89], is a framework for reasoning about time and events. Semantic of EC is a logical representation based on

Predicate	Meaning
$\texttt{happensAt}(E, T)$	Event E is occurring at time T
$\texttt{initially}(F = V)$	The value of fluent F is V at time 0
$\texttt{holdsAt}(F = V, T)$	The value of fluent F is V at time T
$\texttt{initiatedAt}(F = V, T)$	At time T a period of time for which $F = V$ is initiated
$\texttt{terminatedAt}(F = V, T)$	At time T a period of time for which $F = V$ is terminated

Table 1.1: Main predicates of the EC. [Art+12a]

Horn clauses, which use a set of predefined predicates. These predicates may define, for instance, the begining or the end of a CE recognition (cf. Table 1.1).

Specific changes and improvements have been done since [KS89] ([Art+10; ASP12; CM96; CD97; Far+05]), but foundations of EC are usually composed of *simple events*, *fluents*, *time points* and a set of specific *predicates*. *Simple events* are axioms that capture any basic information at a specific moment and corresponding to a simple activity from the system under observation. Each simple event may be attached to a set of attributes carrying information related to this event. *Fluents* are properties derived from logical rules and observed events. Each fluent can take different values and, contrarily to events, possesses an inertia principle which set a specific value to a fluent, over time until anything modifies it. They may be seen as a property of the system or the agents. For instance, in the hurricane alarm scenario, suppose the alarm is able to provide the speed of the wind, a possible fluent might be the intensity of the hurricane regarding the Saffir-Simpson scale[1]. The fluent keeps the same value along time, corresponding to the category of the hurricane detected. The value of this fluent is only modified if wind intensity changes. Note that wind intensity might be measured only periodically, but the fluent value remains until new information modifies it. *Time points* are just real values used to indicate time of actions or value modifications. *Predicates* are the core of EC as they describe the relations between simple events, fluents and time. Different types of predicates exist among the literature which serve specific purposes like continuous change. A good idea of this predicate diversity might be seen through the work of Shanahan and Miller [Sha99; MS02; MS99]. Table 1.1 presents some of the most common predicates of the EC. The predicates are associated together in a Horn clause which describes a new behaviour (defined in a

[1]The Saffir-Simpson scale classifies hurricane regarding their intensity in five categories. Winds for a category 1 hurricane go at 43-49m/s, while winds for a category 5 go above 70m/s

new fluent).

As a matter of example from [Art+12b], equation 1.1 describes the rule for the fluent *reducing_passenger_satisfaction* that indicates that the satisfaction of a passenger using public transportation starts to decrease when the temperature is high inside the bus (event) and the bus is not punctual (fluent). In this rule, *punctuality* is a previously defined fluent.

$$\begin{aligned} &\texttt{initiatedAt}(reducing_passenger_satisfaction(Id; VehicleType) = \texttt{true}, T) \leftarrow \\ &\texttt{happensAt}(temperature_change(Id, VehicleType, very_warm), T), \\ &\texttt{holdsAt}(punctuality(Id, VehicleType) = non_punctual); T) \end{aligned} \tag{1.1}$$

Finding a behaviour within the temporal data will consist in solving a logical problem and identify when a fluent modelling this behaviour takes a specific value. These problems are usually solved using classic approaches like Prolog.

1.2.2 SASE

Gyllstrom et al. proposed their CEP system called SASE [Gyl+06]. SASE is a query-based approach, the semantics of which is similar to SQL. A query is generally structured as follows:

```
[FROM <stream name>]
EVENT <event pattern>
[WHERE <qualification>]
[WITHIN <window>]
[RETURN <return event pattern>]
```

where:

- `FROM` clause provides the name of the stream parsed
- `EVENT` clause describes the structure of the CE pattern.
- `WHERE` clause enforces value-based constraints on the captured events (e.g. imposing ID identity between two elements).
- `WITHIN` clauses defines a sliding window that guarantees that the pattern defined in the `EVENT` field is recognised in a specified time interval.
- `RETURN` clause defined the “shape" of the CE recognised. Generally it is just a listing of attributes of interest.

As a matter of example, the query (Q1), from [Gyl+06], provides a good insight of the model.

```
Q1:
EVENT  SEQ(SHELF_READING} x, !(COUNTER_READING} y),
       EXIT_READING} z)
WHERE  x.TagId = y.TagId ^ x.TagId= z.TagId
WITHIN 12 hours
RETURN x.TagId, x.ProductName, z.ArealId,
       _retrieveLocation(z.ArealId)
```

This query is supposed to capture a shoplifting behaviour regarding the actions performed on objects identified with RFID tags. It describes that an item is taken from a shelf and taken out of the store without being checked[2].

Construction of CEs may be defined using sequential time pattern, negation or assertion on values. Negation describes that an event does not occur. If events are defined before and after the negation in the pattern, the range of the negation is limited between the recognition of these events. For instance, in the query Q1, `!COUNTER_READING y` describes that this event should not be observed between an event `SHELF_READING` and an `EXIT_READING` event, but may be detected outside this range.

SASE+ [DIG07] is an extension of the model proposed by Diao, Immerman, and Gyllstrom that allows pattern description to use Kleene closure [HM00]. Formally, it is possible to specify a CE with an iterative description. For instance, in the `EVENT` clause, the semantic `Event_type+ e[]` describes that the event `e` might be recognised an unspecified number of time but at least once. Furthermore, it is possible to specify constraints on these events: constraints may be defined either on a specific event or as relations between successive events. For instance, the query below:

```
Q2:
PATTERN SEQ(STOCK+ a[ ])
WHERE skip_till_next_match(a[ ]!)
{ [symbol] ^
a[1].price = 10 ^
a[i].price > a[i-1].price ^
```

[2] `READING` term refers to the RFID sensors supposed to read RFID tag from the articles.

```
a[a.LEN].price = 20 }
WITHIN 1 hour
HAVING avg(a[].volume) > a[1].volume
RETURN a[1].symbol, a[].price
```

captures a list of products where the first one should cost 10, the last 20, and, overall, an item should cost more than the previous one in the sequence. The first predicate `[symbol]` asserts that the list of items share the same symbol. Finally, it is even possible to define aggregation functions like `sum` and `avg` to describe a general constraint. The clause `skip_till_next_match` describes an event selection strategy. Event selection strategies define how events should be selected with the Kleene closure depending on the constraints and, by correlation, how to terminate the Kleene closure. Four strategies exist (cf. Figure 1.3):

- *Strict contiguity* enforces two selected events to be strictly continuous in the input stream. If iterative conditions are not fulfilled, recognition of the CE abort. For instance, in Figure 1.3a, *abc* sequence is only recognised when all events are continuous.

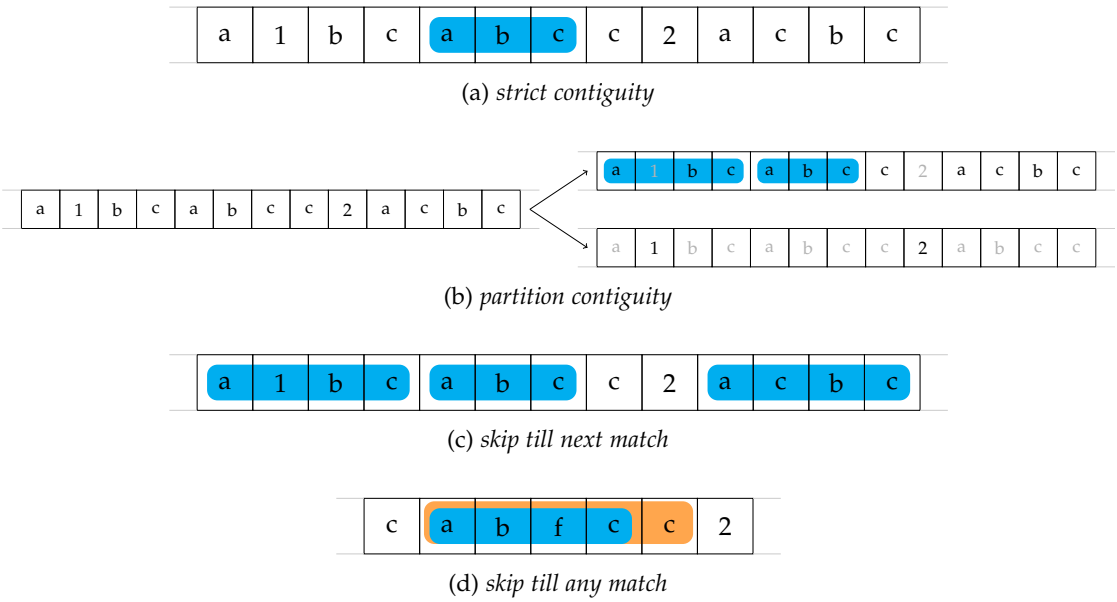
- *Partition contiguity* is a relaxation of strict contiguity where it is needed only into the partition[3] defined by specific constraints. For instance, `[symbol]` creates an event partition with equivalent symbols and iterative conditions should be fulfilled contiguously inside it. For instance, in Figure 1.3b, the data-stream is separated into two partition: one for the letters and one for the numbers. Then sequence *abc* is recognised if they are continuous in the partition.

- *Skip till next match* releases contiguity in a partition and skips events not fulfilling the conditions defined. As this strategy might potential ignores all the skipped events, it continues until reaching a termination criterion. For instance, in Figure 1.3c, recognitions may skip events not fulfilling the conditions.

- *Skip till any match* computes the transitive closure over the relevant events. Such conditions allow the recognitions of all possible matching sequences independently of the order that they are appearing in the data-stream. For instance, Figure 1.3d, sequence *abc* is recognised two times, even if the blue one is a valid recognition[4]. With a *skip till next match* condition, the orange sequence would

[3] A partition is a subset of the input stream usually defined by specific constraints like selecting values in a defined range.

[4] Figure 1.3d presents *Skip till any match* on a smaller example, since a stream like Figure 1.3a produces 16 recognitions, which is difficult to represent clearly.



(a) *strict contiguity*

(b) *partition contiguity*

(c) *skip till next match*

(d) *skip till any match*

Figure 1.3: Graphical examples of SASE strategies.

have not been recognised since the blue sequence reached a termination criterion. Another example might be to find the longest sequence of increasing values for a data-stream $< 1, 2, 3, 1, 5 >$. *Skip till next match* would return $< 1, 2, 3 >$, while *Skip till any match* would return $< 1, 2, 3, 5 >$.

Termination criteria are an important concept with Kleene closure since it may define whether a recognition is valid or not, especially for strategies like *skip till next match* or *skip till any match*. In SASE+, three termination criteria are available:

- **Condition on the last event selected** is a termination criterion specified on the query. For instance in query 2, `a[a.LEN].price = 20` defines a termination criterion on the price value. If this condition is verified, the query produces a recognition.

- **Next component of the pattern** applied only on selection strategies *skip till next match* and *skip till any match*. When an event is valid for the recognition and fulfils all the condition, it is a sufficient termination criterion. Nonetheless, recognition termination of a CE does not imply the recognition process to end. Formally, if a future event is valid with all the previous events, it would produce a new recognition.

- **Window constraints** is the simplest form of termination as it forces Kleene closure to terminate if the window constraint specified in the `WITHIN` clause is reached. If all the constraints have not been fulfilled, this termination criterion does not produce a recognition.

Automata-based representation The recognition process for SASE and SASE+ is based on linear Non-deterministic Finite Automata (NFA). Each query is defined as a NFA to represent the different states of recognitions. Having a non-deterministic property allows the model to ignore valid events with the aim of finding other potential recognitions[5]. The NFA is combined to a match buffer to keep track of the potential matches. The choice of state transition or buffering matching is highly correlated to the event selection strategy. We do not go into further details here, but the reader may refer to [DIG07] for a full account.

1.2.3 TESLA

Trio-based Event Specification LAnguage (TESLA) [CM10] is a CEP approach proposed by Cugola and Margara which, like SASE, uses a semantic close to SQL to define CEs. As stated in its name, TESLA is constructed above TRIO, which is a first order language augmented with temporal operators.

TRIO [GMM90; MMG92] semantics is closely related to temporal logic, as we may find similar operators from TESLA that are equivalent to LTL operators [Pnu77], like *Until* and *AlwF* that correspond respectively to $\mathcal{U}$ and $\Diamond$ in LTL. One of the important aspect is that it allows operators with a specific value for time instants or intervals. For instance, TRIO specifies operators like $\text{Lasts}(A,t)$ that defines that a term A holds from the current time until t, or $\text{Next}(A,t)$ that refers to the closest instant t in the future where A is true.

CEs in TESLA are described using the following semantic:

$$\begin{array}{ll} define & CE(Att_1 : Type_1, \ldots, Att_n : Type_n) \\ from & Pattern \\ where & Att_1 = f_1, \ldots, Att_n = f_n \\ consuming & e_1, \ldots, e_n \end{array} \tag{1.2}$$

where

[5] Especially in *skip till next match* and *skip till any match* strategies.

- The *define* field provides name and attributes of the CE
- The *from* field indicates the LLEs used and conditions that should apply to validate a recognition,
- The *where* field describes how the attributes of the CE are defined regarding events and event attributes used in the *from* field,
- The *consuming* field that define if an event used for recognition remains valid for future recognitions of the same rule.

TESLA allows different kind of patterns:

- *Event composition* allows describing CEs using multiple sequential sub-events. The sequence is described using a *within* operator. Three operators are defined for event composition: *each-within*, *first-within* and *last-within*. For instance, the following rule :

$$\begin{array}{ll} define & Fire(Val) \\ from & Smoke(Area = \$x)\ and \\ & each\ Temp(Val > 45\ and\ Area = \$x) \\ & within\ 5min\ from\ Smoke \\ where & Val = Temp.Val \end{array} \tag{1.3}$$

  describes a CE *Fire* that is recognised if a temperature above 45° is detected within five minutes before detecting smokes. Choice of operator impacts the selection policy for the recognition. Operator *each-within* recognise the CE with every *Temp* event above 45° for each *Smoke* event in the range. Operators *first-within* and *last-within* only recognise the CE with respectively the first and last *Temp* event above 45° in the range for each *Smoke* event. It is possible to use, in a single pattern, multiple *within* operator, which may be relative to different events and times.

- *Parametrisation* allows defining rules between attributes of the events used for recognition in the field *where*. In rule 1.3, parametrisation is used to specify that the *Area*, where smoke and a high temperature are detected, is the same.
- *Negation* allows defining a pattern where an event should not appear into a specified interval. This interval may be declared either using two events or using an

event and a time. For instance, rule 1.3 may be modified into rule 1.4, which specifies that it should not rain between the smoke event and the high temperature event.

$$\begin{array}{ll} define & Fire(Val) \\ from & Smoke(Area = \$x)\ and \\ & each\ Temp(Val > 45\ and\ Area = \$x)\ within\ 5min\ from\ Smoke\ and \\ & not\ Rain(Area = \$x)\ between\ Temp\ and\ Smoke \\ where & Val = Temp.Val \end{array} \tag{1.4}$$

- *Event consumption* defines that an event occurrence may only be used only once for recognitions of a given CE. If an event occurrence may appear in two recognitions of the same CE, only the first recognition performed is valid.
- *Hierarchies of events* allows defining CEs using CEs that are already defined, except if one of the events creates a circular dependence. Unfortunately, it is not specified clearly in [CM10] what time is associated to the recognition of a CE which makes the behaviour of *within* operators unclear when using CEs in the pattern definition. For instance, given a CE c used in another CE c', if c is recognised between time t and t', it is not defined if the operator *within* is valid if t is in the range of the operator, if t' is in the range of the operator, or if both should be in the range of the operator.
- *Iterations*, or *Kleene closure*, are theoretically possible to define using multiple rules definitions of the same CE. Even if possible, this possibility is not embedded directly in the semantics and needs some tweaks.

Automata-based representation The recognition process uses an automaton model similar to SASE[6]. Each rule is transformed into an automaton where constraints on events selections are linked to the edges of the automaton. Multiple instance of an automaton might be created in parallel to recognise every occurrence. The full system is embedded in a CEP middleware named T-Rex [CM12].

1.2.4 ONERA Chronicles

The chronicle system was initially proposed by Ornato and Carle [OC94a; OC94b]. The authors aimed at extending the intention recognition domain to be able to recognize a

[6]Note that SASE is based on NFAs while TESLA uses deterministic automata

behaviour in data that may contain unrelated activities with the behaviour. Chronicle Recognition System (CRS) is based on a temporal language describing the behaviour to identify using typed LLE. The semantic is composed of the following operators to describe CEs named chronicles :

- Sequence, conjunction, disjunction of events.
- Non-occurrence of a chronicle within the scope of recognition of a second chronicle.
- A delay operator.
- A cut operator aiming at reducing the potential number of combinatorial recognitions.
- An indexation operator that identifies simple events related to each chronicle recognition.

Time is discrete and represented as a counter and recognition of chronicle may be re-injected into the data stream to be potentially used as an event for more complex chronicles.

This model is implemented using duplicable automata. Each automaton might be duplicated each time a LLE is recognised and takes place in a CE recognition, which is also how a non-deterministic component is introduced in the recognition model. Consequently, each automaton represents an occurrence of the CE it models. This method is quite similar to the one used by SASE with T-Rex middleware [CM12].

This representation of chronicles changed with the work of Bertrand, Carle, and Choppy [BCC07; BCC08; Ber09; CBC09], who introduced a Petri-net-based representation for chronicle recognition. The main formalism has been refined multiple times since [OC94a] (e.g. in [Ber09] and [CCK11]), but we focus on the latest formalism proposed by Piel [Pie14].

The formalism proposed in [Pie14] introduced many notable changes compared to previous works:

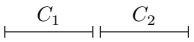
- It is now possible to specify attributes associated to events. There is no particular restriction on the number, the type or the value of attributes. They may be modified during the analysis, renamed or associated to define the value of CEs attributes. For instance, given two events A and B that have respectively an

attribute x and y, a CE C might be defined with an attribute z defined by the attributes x and y such as $A(x) \vee B(y) \rightarrow C(z = x \vee y)$.

- Existing operators are extended with Allen's interval algebra [All83] and other delay operators.
- Time does not need to be a counter any more and may be represented continuously, with a function "Look-ahead" providing the next instant where computation is needed.

Operators definition We present briefly the existing different operators as we will refer to them quite often along this thesis. We just detail the temporal constraints used to compose chronicles together, but it should be kept in mind that conditions on attribute values may be defined together with the operators. By convention, recognition of a chronicle C_i is written r_i. Recognition r_i happens in a interval of time $[T_{min}(r_i), T_{max}(r_i)]$ where $T_{min}(r_i)$ and $T_{max}(r_i)$ are respectively the first and last instants of the event recognised as r_i.

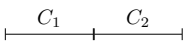
- A **sequence $\mathbf{C_1}$ $\mathbf{C_2}$** is recognised if a recognition r_1 happens before a recognition r_2. The temporal constraint to be asserted is $T_{max}(r_1) < T_{min}(r_2)$.



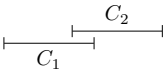
- A **disjunction $\mathbf{C_1}||\mathbf{C_2}$** is recognised either if a recognition r_1 or if a recognition r_2 happens.
- A **conjunction $\mathbf{C_1}$&$\mathbf{C_2}$** is recognised if both recognitions r_1 and r_2 happen (without any condition on their time positioning, contrarily to the sequence).

- An **absence** $(\mathbf{C_1}) - [\mathbf{C_2}]$ is recognised if a recognition r_1 happens but r_2 does not happens during the interval of time of r_1. The temporal constraint to be asserted is
  $\forall r_2, T_{min}(r_1) > T_{min}(r_2) \vee T_{max}(r_1) < T_{max}(r_2)$.
  It is possible to define variants of absence with either inclusive or exclusive intervals:

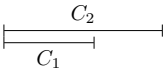
  - temporal constraint for $(\mathbf{C_1})-]\mathbf{C_2}[$ asserts that $\forall r_2, T_{min}(r_1) \geq T_{min}(r_2) \vee T_{max}(r_1) \leq T_{max}(r_2)$;
  - temporal constraint for $(\mathbf{C_1}) - [\mathbf{C_2}[$ asserts that $\forall r_2, T_{min}(r_1) > T_{min}(r_2) \vee T_{max}(r_1) \leq T_{max}(r_2)$;
  - temporal constraint for $(\mathbf{C_1})-]\mathbf{C_2}]$ asserts that $\forall r_2, T_{min}(r_1) \geq T_{min}(r_2) \vee T_{max}(r_1) < T_{max}(r_2)$.
- A chronicle $\mathbf{C_1}$ **meets** $\mathbf{C_2}$ is recognised if a recognition r_1 finishes at the exact moment that a recognition r_2 starts. The temporal constraint to be asserted is
  $T_{max}(r_1) = T_{min}(r_2)$.

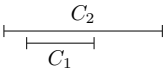
- A chronicle $\mathbf{C_1}$ **overlaps** $\mathbf{C_2}$ is recognised if a recognition r_2 starts after the start of a recognition r_1, but finishes after r_1 finishes. The temporal constraint to be asserted is
  $T_{min}(r_1) < T_{min}(r_2) < T_{max}(r_1) < T_{max}(r_2)$.

- A chronicle $\mathbf{C_1}$ **starts** $\mathbf{C_2}$ is recognised if a recognition r_2 starts at the same time as a recognition r_1, but finishes after it. The temporal constraint to be asserted is
  $T_{min}(r_1) = T_{min}(r_2) < T_{max}(r_1) < T_{max}(r_2)$.

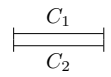
  

- A chronicle $\mathbf{C_1}$ **during** $\mathbf{C_2}$ is recognised if a recognition r_2 starts before a recognition r_1 and finishes after it. The temporal constraint to be asserted is
  $T_{min}(r_1) > T_{min}(r_2) \wedge T_{max}(r_1) < T_{max}(r_2)$.

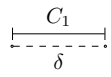
  

- A chronicle $\mathbf{C_1}$ **finishes** $\mathbf{C_2}$ is recognised if a recognition r_2 starts before a recognition r_1 starts, but both finish at the same time. The temporal constraint to be asserted is
  $T_{min}(r_1) > T_{min}(r_2) \wedge T_{max}(r_1) = T_{max}(r_2)$.

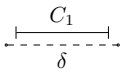
- A chronicle **C_1 equals C_2** is recognised if both recognitions r_1 and r_2 happen over the very same time interval. The temporal constraint to be asserted is $T_{min}(r_1) = T_{min}(r_2) \wedge T_{max}(r_1) = T_{max}(r_2)$.



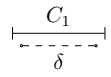
- A chronicle **C_1 lasts δ** is recognised if a recognition r_1 happens and lasts exactly δ. The temporal constraint to be asserted is $T_{max}(r_1) - T_{min}(r_1) = \delta$.



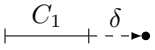
- A chronicle **C_1 at most δ** is recognised if a recognition r_1 happens and lasts at most δ. The temporal constraint to be asserted is $T_{max}(r_1) - T_{min}(r_1) < \delta$.



- A chronicle **C_1 at least δ** is recognised if a recognition r_1 happens and lasts at leasts δ. The temporal constraint to be asserted is $T_{max}(r_1) - T_{min}(r_1) > \delta$.



- A chronicle **C_1 then δ** is a chronicle delayed from a duration δ after the end of a recognition r_1. Recognitions of such chronicle are associated not to an interval but to a time point t where $t = T_{max}(r_1) + \delta$.



For the sake of conciseness, we do not detail further, especially on the attribute management formalism which include naming and transformation functions and predicate definitions on attributes. For more details, the reader should refer to [Pie14]. This formalism has been implemented into a C++ library named Chronicle Recognition Library (CRL)[7] that will be used later in this thesis.

1.3 Drone model

Along this thesis, we have applied our approaches on a specific context of communication loss during an Unmanned Aircraft (UA) flight between a UAV, its pilot on the ground and the Air Traffic Control (ATC). In section 1.3.1, we describe the detailed context and the problematic due to uncertainty. In section 1.3.2, we present how this case study has been modelled and describe the different variations that may be used.

[7] https://github.com/ChronicleRecognitionLibrary/crl

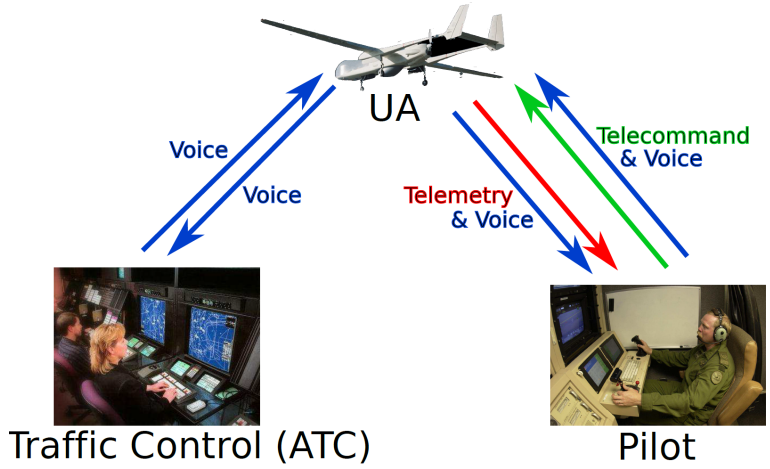


Figure 1.4: Schematic representation of the drone loss communication study case

1.3.1 Communication loss between three agents during a drone flight

The growing interest and developments on UAV usage raise many problems of safety. In this thesis, we used a case study of a potentially dangerous situation that may happen when communication with a drone is lost.

Our case study, graphically represented in Figure 1.4, considers three agent: the UA or drone controlled remotely by the pilot via the Remote Pilot Station (RPS) using the telecommand (TC). The UA may send various information (speed, topography, anti-collision system, etc.) to the remote station through telemetry (TM). RPS and UA may be described as one entity named Unmanned Aircraft System (UAS)[8]. Furthermore, the pilot may occasionally need to contact the ATC established through radio communication exchange (Voice).

Due to different unpredictable factors, like a system failure or specific topology, communication between the agents may be lost. It may be any of the three (TC, TM, voice), or possibly all at the same time. In such situations a full protocol defines precisely the behaviour that each agent should follow[9]. Without possible communication, the agents have to deduce, regarding their situation and their partial individual knowledge, in which state the unreachable agent is.

[8] *'an unmanned aircraft and associated elements [...] required for the pilot in command to operate safely and efficiently in the national airspace system.'* [Hou12].

[9] Described in section 1.3.2.

In a previous work [Pie14], CEP was used on this case-study to identify unplanned risky situations. For instance, when an agent detects a communication loss and triggers back-up protocol while another remains in its original state leading to potential conflicting decisions between these two agents. LLE were schematically the actions done by each agent on the system or their change of state. However, this first approach used no uncertainty representation so recognition of a chronicle requires to know the events produced by all agents, which is contradictory with the current study case that focus on loss communication. Indeed, such approach would require an omniscient observer on system which is not achievable in practice.

On the other hand, introducing uncertainty into this CEP model with probabilities allows each agent to have its own "*local probabilistic observer*" that estimates the current state of the others when communication is lost. This would allow each agent to detect dangerous behaviour with partial or contradictory information.

1.3.2 Problem model: telecommand and radio losses

We can now present the protocol that should be followed by the agents in case the telecommand or the radio is lost. We describe these scenarii separately but, as we explained earler, nothing prevent they append together. Then, we present more precise examples of dangerous situations where two agents have different understanding of the system state.

Protocol followed by each agent regarding the different situation is described in the state diagram in Figure 1.5. The diagram is separated between the three agents: UA, RPS, and ATC. Each agent is decomposed into sub-systems. A sub-system represents a functionality controlled by the agent or a specific knowledge it may possess about the overall system situation. For instance `RPS Voice` sub-system describes if the RPS is aware of a potential radio communication loss. Each sub-system is a set of states that have to be followed in a specific order regarding the system evolution. Arrows between states describes this order and necessary conditions to trigger a state change and may eventually associate to a specific action to be performed by the agent. This information is labelled on arrows into three different parts: *event*, *condition* and *action* and written `event [condition]/action`.

Telecommand loss In this part, we describe the case of telecommand loss, meaning the pilot is not able to send order to the drone. Without losses, each agent is in `Nominal` state and the drone follows a predefined flight plan known by both RPS and ATC.

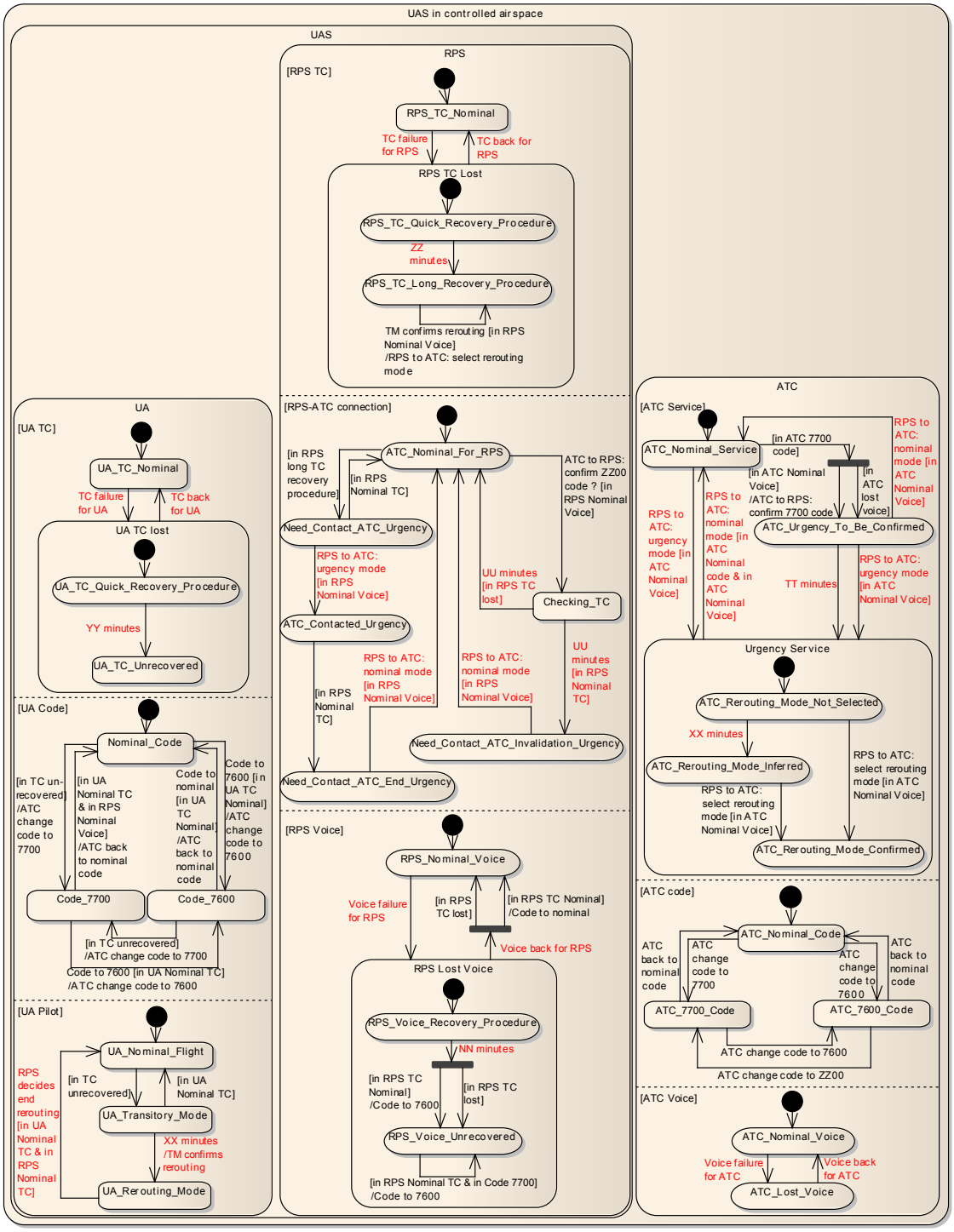


Figure 1.5: State diagram of telecommand and radio loss.

If the UA detects a telecommand loss, sub-system `UA TC` goes to state `UA TC Quick Recovery Procedure` where it will try quick attempts to recover the signal. If it fails to retrieve it, the drone switch to state `UA TC Unrecovered`. In this state, the drone should emit, with its transponder[10], a code 7700 that indicates an emergency. This code might be detected by the ATC. At the same time, the UA changes its state in the sub-system `UA Pilot` in `UA Pilot Transitory Mode`. After a fixed delay, it passes in `UA Pilot Rerouting Mode` where it follows a predefined emergency flight plane. This flight plan might be changed again if the telecommand signal is back and RPS and ATC decided to follow the original plan.

In case of telecommand loss, RPS may detect it and change his state in the TC sub-system from `RPS TC Nominal` to `RPS TC Quick Recovery Procedure`. In this state, the pilot tries quick measures to recover the telecommand. After a delay and without being able to recover the signal, the RPS change his state to the `RPS TC Long Recovery Procedure`. At this moment, the pilot should try to contact the ATC to warn him about the situation. In the subsystem `RPS-ATC connection`, state changes to `Need Contact ATC Urgency`. If, before ATC is contacted, TC signal comes back, RPS can come back to his nominal state. When ATC is acknowledged of the situation that the initial flight plan may have been changed automatically to the emergency one by the drone, RPS switches to `ATC Contacted Urgency` state and stays in it until the end of the flight or if the telecommand signal comes back. This exchange requires the voice connection to be working. In this last scenario, RPS change his state to `Need Contact ATC End Urgency` and has to re-contact the ATC to take a decision about the flight plan to follow. ATC and RPS may decide to reselect the original flight plan or keep the emergency one. After the communication, both agents may return to their `Nominal` states. It may happen that the ATC detects an emergency before the RPS. In this situation, they may contact the RPS to obtain information. When they are contacted, they should check if control over the drone has effectively been lost (`Checking TC` state). If telecommand is lost, protocol restart from nominal as described previously. In contrary, if telecommand is working, the RPS has to contact a second time the ATC to confirm the flight plan and invalidate the emergency (in state `Need Contact ATC Invalidation Urgency`).

For the ATC, two situations may arise accordingly to the behaviour of the other agents. In the first case, the pilot contacts it to warn it about the loss of the telecommand changing the state of the sub-system `ATC Service` to `ATC Rerouting Mode Not`

[10] A transponder is an automated transceiver in an aircraft that emits a coded identifying signal in response to an interrogating received signal.

`Selected`. RPS may indicates immediately that rerouting has been selected by the drone changing the ATC state to `ATC Rerouting Mode Confirmed`. If the ATC has no information about the rerouting, it will infer it after a delay and go in state `ATC Rerouting Mode Inferred`. From this state, a contact with the RPS may confirm or deny the rerouting, but the contact should come from the RPS. The second situation arises when the ATC detected a 7700 code emitted by the drone, but the RPS did not try to notify it. In this situation, ATC should contact the RPS. Contacted or not ATC switches to `ATC Urgency To Be Confirmed`. After a delay or if the RPS reached him to confirm the emergency, state change to `ATC Rerouting Mode Not Selected` and protocol follows the same scheme as in the first situation.

Radio signal loss It may happen that the radio communication fails. Reasons might be the same as for the telecommand loss since the radio signal goes through the UA. RPS and ATC agents have both a sub-system `Voice` describing their current understanding of the radio signal state. There is no special procedure to follow in case of radio loss except that the RPS has to ask the drone to emit 7600 code. This warns the ATC of the communication loss. Nonetheless, losing the radio signal has a huge impact on the protocol described previously. Indeed, each time RPS and ATC have to communicate, it requires voice communication to be operational. If both telecommand and radio are lost, some steps of the protocol are not reachable any more and may lead to contradictory beliefs regarding a situation. Let us illustrate this by a simple example. Every agent starts in nominal states, but, due to technical issues telecommand is suddenly lost. During the quick checks performed by the RPS, radio breaks down too. When the RPS reaches the `Long recovery Procedure`, he should contact the ATC (`Need Contact ATC Urgency` state), which is impossible due to the radio signal loss. In the meantime, UA detects the telecommand loss and emit a 7700 code. ATC detects the emergency and may conclude, after a while, to a rerouting. But during this time, RPS retrieved the telecommand before the drone entered in rerouting mode. As situation went back to normal, for the RPS point of view, he may return to the `ATC Nominal For RPS` state. ATC receives the nominal code, but its protocol does not permit it to go back to nominal state, and he should still assume the rerouting while the RPS follows the initial flight plan.

We presented in this section the drone model that we used in this thesis as a support for our experimentations on uncertainty integration with chronicles. We will use it in

Chapters 5 and 6 with slightly different representations to adapt it to the uncertainty model that we use. Each alternate version is presented in its corresponding part.

CHAPTER 2

Complex Event Processing under uncertainty

Contents

2.1 Understanding uncertainty on Complex Event Processing 42
2.1.1 Data Uncertainty 42
2.1.2 Pattern uncertainty 44
2.1.3 Understanding of goals and concepts of Complex Event Processing under uncertainty 46
2.2 Markov Logic Approaches 48
2.2.1 Event calculus using Markov Logic Networks 48
2.2.2 Interval approaches 51
2.2.3 Ad-hoc approaches 53
2.2.4 Summary on Markov Logic Networks approaches 54
2.3 Bayesian network approaches 56
2.4 Automaton-based methods 60
2.4.1 SASE Approaches 61
2.4.2 Automaton-based representations from Albanese et al. and its extensions 64

2.4.3 Automaton-based representation from Fazzinga et al. and its extensions . . . 66
2.5 Other approaches . . . 68
2.5.1 ProbLog method . . . 69
2.5.2 HMM . . . 69
2.5.3 Petri-net approaches . . . 71
2.5.4 Grammar approaches . . . 72
2.6 Conclusion . . . 75

HIS chapter presents an overview on CEP under uncertainty. We first discuss the various possible understandings of uncertainty among the works from the CEP domain or related to it. We then present the state-of-the-art of CEP under uncertainty regarding the different approaches used like MLNs, Bayesian networks and automaton-based methods.

Section 2.1 provides an analysis of uncertainty on CEP and more specifically, we tried to express how it is understood along the community. This same section puts into perspective the differences between goals and perspectives of CEP with and without uncertainty incorporated on the models. State-of-the-art starts in Section 2.2 with a review of methods based on MLN, followed by the Bayesian networks approaches in Section 2.3, and automaton-based methods in Section 2.4. Finally, Section 2.5 presents some approaches based on less common techniques or representations.

2.1 Understanding uncertainty on Complex Event Processing

Before presenting a state of the art on CEP under uncertainty, we need to discuss shortly the actual meaning of "uncertainty" in this context. Indeed, the same word "uncertainty" appears to have different meanings among the existing works. Even if it might be difficult to draw a precise boundary along the different interpretations of uncertainty, two main categories may be described: uncertainty on data and uncertainty on patterns.

2.1.1 Data Uncertainty

Data uncertainty commonly refers to noisy or corrupted data streams. In this case, different kinds of uncertainties may be distinguished :

- *Event uncertainty*. For many systems, information is gathered through sensors, which are not necessarily always reliable and might produce inconsistent data. Two possible scenarii could be described.

  1. The first possibility is that a real simple activity goes undetected and consequently the corresponding LLE does not appear in the data stream. Missing an LLE obviously has a huge impact on the recognition process since all possible CEs depending on it will not be recognised.
  2. The second possibility is that LLEs have been produced that are inconsistent with reality. By inconsistent, we mean that an LLE literally appears on the data stream, but does not match any real simple activity, which may be interpreted as a noise. Consequences are quite similar to those of a missing LLE, as it is possible to recognise CEs while the correct behaviour did not occur. But in general, missing an LLE will produce false negatives during the recognition process, while an inconsistent LLE will produce false positives.
  3. A third possibility may be considered as event uncertainty when a simple activity is registered as LLE of an inconsistent type. Formally, this possibility might be seen as the outcome of the composition of the two previous ones, where a simple activity goes unrecognised, but an inconsistent LLE is produced at the same time.

- *Uncertainty on attribute values.* Even if LLEs might be correctly detected, it is possible to have uncertainty on their attribute values. For the same reasons as for event uncertainty, sensors that capture this kind of information are not immune to faulty detections. According to the type of attributes embedded on events, uncertainty may be managed differently. For instance, if events have a finite discrete number of attributes, each set of possible values might be considered as a distinct LLE and managed with the same strategies than event uncertainty. But, on many cases, attribute values are continuous and might not be easily discretised, thus cannot be formally considered as variant of the event uncertainty issue and must be handled differently.

  Dealing with event and attribute value uncertainties in the same method is not common and existing approaches tend to focus on either of them. We will see that works combining these two kinds of uncertainty exist, but usually require a lot of assumptions on the effect of attributes on events. For instance [Cug+15] considers independent event recognitions and attribute values.

- *Time uncertainty.* Even if time uncertainty might be seen as a specific case of attribute value uncertainty, consequences might be quite different. Time uncertainty usually refers to imprecisions of the detection time of a LLE. In general, it means that if an LLE is recognised at a given time t, it might, in fact, have happened at a close, yet distinct, time $t \pm \delta$. This kind of uncertainty may induce shifts between the real sequence of activities and the order of their corresponding events. Consequently, it may have a huge impact on recognition as in many CEP frameworks, the order of events in data streams is an important feature. This particular case of uncertainty is not often addressed, but some rare works exist.

2.1.2 Pattern uncertainty

Pattern uncertainty is conceptually easy to visualise, but we will see that it may sometime refer to counter-intuitive representations closer to data uncertainty. It commonly describes a lack of knowledge needed to describe a behaviour and, in its most common approach, the idea is to embed probabilities into the CEP representation. For instance, many methods represent uncertain CEs as probabilistic automata, which would intuitively mean that changing from a state to another is dependent not just on conditions, but on random variables too. In another point of view, it might be understood as a representation of a lack of knowledge for the behaviour design of the CE. The interest in this is straightforward as, in many situations, it may be possible to know that a complex behaviour has emerged, but not necessary which sequence or pattern of activities actually lead to it. Probabilities on the pattern describes a sort of confidence on the scheme the complex behaviours follow. For this reason, many works that focus on pattern uncertainty proposed methods to learn automatically the transition probabilities.

Unfortunately, many approaches tend to confuse data and pattern uncertainty often trying to embed data uncertainty into a probabilistic model of CE. This seems quite counter-intuitive, since there is no reason for the CE representation to be an efficient model to capture data uncertainty and a potential probabilistic dependence between events or time. As a matter of example (Figure 2.1), suppose we want to recognise a CE that is a sequence of a LLE a followed by a LLE b, written ab. This CE is known to be approximate and many works would usually try to learn its probability[1]. For

[1]It could induce complex computation of dependencies with conditional probability defined between events inside the CE, but, at the end of inference, the goal is almost always to estimate the marginal probability of the CE. Consequently, in our example, we focus on the marginal probability, since its sufficient to show that a CE structure is not completely relevant to capture pattern uncertainty.

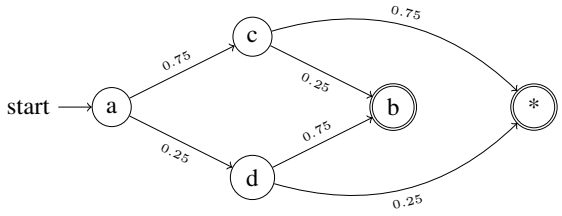


Figure 2.1: Example of conditional dependencies between LLEs on a data stream. The star node represents a state where b will not appear.

this example, we suppose there exists a pattern matching perfectly the behaviour of interest and modeled by the sequence adb. Statistically, we will suppose that after a LLE a it may only appears a LLE c or a LLE d with respectively 75% and 25% chance (cf. Figure 2.1). And furthermore, a LLE b has 25% to appear in the future of a LLE c and 75% after a LLE d. Without the *true* pattern, we should suppose that the CE ab has 50% chance to be valid when recognised in a data stream because contextual information is not considered. We can see in this case that the CE probability, that should express the lack of knowledge in the pattern design, is in fact a problem of data uncertainty where contextual information around the CE is ignored. It would be possible to argue that the purpose of using uncertainty on CE ab is exactly capturing this lack of context, but events c and d are known and are, nonetheless, assumed to produce no effect on the CE recognition. This pattern uncertainty would certainly be more consistent as a structural uncertainty problem, but few works approach it that way. This confusion tends to appear regularly, especially on works from the visual data analysis community. This is probably due to the fact that, in this field, data-streams and patterns are quite fuzzy or difficult to define. Nonetheless, there are works trying to combine pattern and data uncertainty with independence assumption [Cug+15] or without it [MM07].

Temporal impact on pattern uncertainty may be addressed differently. Some works where event order is not a major concern (like in [CM12; Cug+15]) ignores completely the impact of time for the CE probability estimation, which means that probability is calculated independently of the number of events in the stream and time between events. On the other side, works that integrate time in their representation often describe it as conditional dependencies between two successive LLEs in the CE definition, which may ignore the impact of time between two events[2].

[2]One exception is the work of Molinaro et al. that introduce a notion of decay, but this decay is a fixed value which seems hard to estimate.

2.1.3 Understanding of goals and concepts of Complex Event Processing under uncertainty

In section 1.1, we provided an overview of the main goal of CEP and its core concepts. It seems necessary to address the fact that these points of interest can not have the same exact meaning when considering uncertainty.

For instance, it is not always possible to provide separately CEP and uncertainty formalisms since they might, quite potentially, be interconnected and each method will have to consider which uncertainty representations[3] are best suited for each problem. Furthermore, similar questions arise about methods to extract CEs from a data stream: do they provide recognition with probability higher than a confidence threshold? Do they provide all the possible recognitions, even with low confidence? Do they summarise recognition confidence under one probability? There is no single answer to these questions and approaches may consider one representation or another regarding what they aim at expressing.

These differences between common CEP and CEP under uncertainty impact the usual sub-goals (presented in Section 1.1). We review them one more time, but now considering uncertainty:

Expressiveness It immediately appears that expressiveness would now refer to how uncertainty may be represented in the model and more precisely which kind of dependency may be defined. For example, data and pattern uncertainties lead obviously to different degree of expressiveness.

Incidentally, it is necessary to look over the type of inference that should be performed since approaches may focus on different inference goals. For instance, most works tend to compute marginal probability of a given CE $x(t)$ regarding a set E of observed LLEs:

$$P(x(t)|E) \tag{2.1}$$

where t might be as well at time point or a time interval for the recognition of x. Other research, in contrast, focuses on maximum a posteriori (MAP) inference and, in general, on a sequence of states θ maximising the probability of the CE.

$$\arg\max_{\theta} P(\theta|x(t), E) \tag{2.2}$$

[3] cf. Sections 2.1.1 and 2.1.2.

Furthermore, some approaches may find interest in considering hard constraint rules as observations. These rules might describe different pieces of information, like mandatory relations between events in the system or a CE considered as prior knowledge. Obviously, this type of possibility is highly correlated with the uncertainty representation used. For instance, MLN approaches may easily represent these constraints while automata methods are not well-suited for this kind of computation.

Efficiency Even if the number of events analysed per second remains an important characteristic to measure efficiency, it might not be the only criterion anymore. As approaches have to produce some sort of confidence value, it is necessary to measure the efficiency of methods. Most works would evaluate this efficiency applying their approaches on known datasets and comparing them using an F-measure score. Furthermore, various approaches may use techniques that use approximate or exact inference, which intuitively cannot provide the same accuracy. For instance, [Ska+15b] or [MD11] look for all possible worlds regarding a given data stream, which provides exact inference according to the model, whilst [Ska+11] uses an MLN approach, based on the classical MC-SAT algorithm, that does not ensure exact estimation according to the model.

It is also noteworthy that some methods ignore potential contextual information (especially when evaluating pattern uncertainty), therefore computing the confidence on CE recognitions might be fuzzy due to a lack of expressiveness or excessive independence assumptions.

Multiplicity Multiplicity, which consists in recognising and memorising all possible recognitions, raises difficult questions. For instance, should recognitions with a low probability be counted as interesting or stored for future use? Memorising everything might be intractable, but putting a threshold may miss necessary analysis information for more advanced CEs.

Furthermore, considering, in a uncertain data stream, a method that tries to estimate the probability of a CE, which is designed using two consecutive intermediate CE, it may be possible to estimate probabilities of each one independently, but nothing ensures that it is possible to multiply their two probabilities since the sub-CEs may not be independent. Without uncertainty, multiplicity mainly consists in memorising all identified patterns. With uncertainty, it is necessary to understand possible dependencies to be sure that the computation just "makes sense" and to consider all possible versions of a stream.

For pattern uncertainty, multiplicity is likely to be less problematic, as the data stream is supposed safe, which avoids combinatory explosions or at least keeps it under some control.

Historisation Historisation faces issues similar to those for multiplicity, since it implies storing all possible LLEs with all associate probabilities leading to a recognition.

Time representation Time representation may have a huge impact on the uncertainty representation. Indeed, time may be underlying relations or dependencies that should be considered when analysing a data stream during the recognition process. Uncertainty regarding these time-dependent relationships might be represented in many ways. It is possible to suppose marginal probabilities independent of time or probability dependencies using specific probability distribution. For instance, some works ([Mol+14]) use automaton to recognise a pattern, but probability computation is modified regarding a delay function defined between two states of the automaton.

2.2 Markov Logic Approaches

MLNs introduced by Richardson and Domingos [RD06] are usually presented as Probabilistic Graphical Models (PGMs), but are in fact closer to a framework based on logic to construct Markov random fields. Briefly, MLNs are defined using a set of logical clauses and a weights associated to each clause that represents the confidence on the formula to be asserted: the higher the weight, the higher the confidence. Some formulae are hard constraints, meaning that they cannot be broken, and are defined as formulae with infinite weights.

As MLNs associate the expressiveness of FOL with a probabilistic model, they have been attracting increasing intention for the last ten years among the CEP community, where an important research area focuses on trying to add uncertainty to their models. We present MLNs in further detail in Section 3.1 and we now focus on the existing CEP approaches using MLN.

2.2.1 Event calculus using Markov Logic Networks

Skarlatidis et al. [Ska+11] proposed an adaptation of Event Calculus using MLN to soften detection, when pattern detection of HLEs is not certainly defined. The authors try to detect activities from the dataset of the CAVIAR project.

The CAVIAR project is a dataset of surveillance videos annotated with the activities of people or objects that appear on it. This information is just made of activity tags: *active*, *inactive*, *walking*, and *running*, with an ID identifying the person or object. These tags may be seen as LLEs. More complex activities are annotated and correspond to interactions like *meeting*, *moving*, *fighting*, and *droping an object*.

The goal of the authors is to deduce these behaviours using LLEs, but as the pattern to recognise a CE is not well defined, the idea is to use MLNs to let recognition be more tolerant by softening associated logical formulae. The authors used a reduce version of Event Calculus named Deterministic Event Calculus which uses less predicates to define the rules of the system than in the initial version of Event Calculus (*happens*, *holdsAt*, *initiates*, and *terminates*) and where time is considered as discrete.

MLNs do not use the close world assumption so circumscription has to be instantiated. Briefly, circumscription consists in eliminating predicates that are not entailed by the formulae. For instance, given a set of variables $x, y, z \in V$ and a set of formulae $F = \{a(x,y) \wedge b(y,z) \implies c(x,z)\}$, if $c(x,z)$ is true, circumscription ensures that it exists a $b(y,z)$ and $a(x,y)$ that are true too. Without circumscription, this is not guaranteed. Consequently, rules for deducing HLEs should be written with the inverse of every formula. For instance, the inverse of F is $F' = \{\exists y\, c(x,z) \implies a(x,y) \wedge b(y,z)\}$. Unfortunately, with MLNs, existential quantifiers applied on the right side of a clause force the model to create a number of clauses relative to the size of the set V and the number of variables under the existential quantifier, which may impact the inference performances. For instance, if $V = \{1,2,3\}$, formula F' would be rewritten:

$$F' = \begin{cases} c(x,z) & \implies a(x,1) \vee a(x,2) \vee a(x,3) \\ c(x,z) & \implies a(x,1) \vee a(x,2) \vee b(3,z) \\ c(x,z) & \implies a(x,1) \vee b(2,z) \vee a(x,3) \\ c(x,z) & \implies a(x,1) \vee b(2,z) \vee b(3,z) \\ & \vdots \\ c(x,z) & \implies b(1,z) \vee b(2,z) \vee b(3,z) \end{cases} \tag{2.3}$$

In this case, the existential quantifier makes the number of clauses to grow with a factor $2^{|V|}$. This will be discussed in detail in chapter 4. To avoid this problem, the authors rewrite formulae to be domain-dependant to reduce the number of parameters per formula. For instance, Formula 2.4 which describes the conditions to initiate the fluent *meet* with a non-domain-dependent formulation, is rewritten into formula 2.5

to constraint the *initiatedAt* predicate to the fluent *meet* in the clause head to avoid the F variable and use directly $meet(ID_1, ID_2)$, which is more restrictive.

$$
\begin{aligned}
&initiatedAt(F, T) \leftrightarrow \\
&\quad \exists ID_1,\ ID_2\ (F = meet(ID_1, ID_2) \wedge \\
&\quad happens(active(ID_1),\ T) \wedge \\
&\quad \neg happens(running(ID_2),\ T) \wedge \\
&\quad close(ID_1,\ ID_2,\ 25,\ T))
\end{aligned}
\tag{2.4}
$$

$$
\begin{aligned}
&initiatedAt(meet(ID_1, ID_2), T) \leftrightarrow \\
&\quad happens(active(ID_1),\ T) \wedge \\
&\quad \neg happens(running(ID_2),\ T) \wedge \\
&\quad close(ID_1,\ ID_2,\ 25,\ T))
\end{aligned}
\tag{2.5}
$$

The authors study the impact on performance of formulae that may not be broken (hard constrained) and formulae that may be broken (soft constrained) to evaluate how prior knowledge may influence inference. Two sets of formulae are defined: one describes complex activities (LLEs imply HLEs) and the other describes the circumscription. The authors choose to soften one either at a time or both and compare the results to a fully logical model named EC-Crisp on a noisy data stream. This work might be considered as one of the few that analyse the performance of pattern uncertainty approaches when embedded into a CEP representation.

Further work from these authors [Ska+15b] provides more results on this case study where the formulae are separated into two new sets: one representing the recognition of a fluent and the other representing the inertia of this fluent along time. Interestingly, softening the second set will be equivalent as putting a decay on the recognition: after a certain amount of time, and without detection of LLE, every fluent tends to be uncertain with a 50-50 chance. By applying this on the dataset modified with different levels of noise, they show that it is possible this way to tweak the impact of data uncertainty by making the model to be more tolerant about false observations.

However, these two approaches still have a lot of limitations, principally due to MLN behaviour. Defining the weight for the formulae is a difficult task as the confidence associated to a weight is relatively dependent of all the others weight of the problem [DL09; Jai11; Ska+15b]. It is still possible to learn the weights with supervised learning, but this technique might be difficult to set up on specific cases, especially when knowledge comes principally from specialists and might cause performance issues. Furthermore, the model presented is quite simple as it just us one level of deduction between LLEs and HLEs and seems to be domain-dependent for every case, but dimensionality of the problem might not always be reduced so easily.

2.2.2 Interval approaches

Interval-based approaches with MLNs applied on CEP do not usually come from the CEP community but from the video analysis domain. Data analysed by this research community is generally noisy and research aims at improving the reliability of behaviour recognition. This noise on data is usually produced by the inherent blurriness of image support and compression, but it might also come from preprocessing techniques used to extract objects or persons from images. Indeed, these techniques are not fully reliable and may produce inconsistent data. Consequently, MLNs are used to represent this uncertainty and reduce the impact of errors from preprocessed LLEs by considering underlying activity dependencies along time. These dependencies are represented in many works by combining events together with Allen's interval algebra [All83].

Morariu and Davis [MD11] aim at detecting activities in a video of basketball match. Their logical model is constructed from *observations*, which consist of *events* or *properties*. Basically, *observations* are evidences on the system, e.g. `obs_in_air`(i) meaning the ball has been detected in the air on the video during the instant of time i. *Properties* are states of the game that may represent rules of basketball, e.g. `can_dribble`(i) indicating whether a player is allowed to dribble or not according to the rules. *Events* are complex actions that are induced by the observations and should verify the properties (i.e. the rules). Allen's algebra is used to define relations between events or property; however the *sequence* operator is not used in the model[4] to avoid long-term temporal relations, thus reducing drastically the size of the Markov network. Furthermore, the authors do not use classical MC-SAT algorithm[5] [PD06] on MLN for inference but a branch-and-bound algorithm [MD09]. MLNs are just used to construct the ground Markov network. The main difference between these two algorithms is that MC-SAT computes approximate marginals while the branch-and-bound algorithm produces exact inference. This algorithm is, in fact, closer to other techniques for dealing with uncertainty like d-DNNF [Dar01; Dar04] or Sentential Decision Diagram (SDD) [Dar11] used in ProbLog [RKT07; Dri+15]. It provides fast and strong results on small-dimensional problems, but higher-dimensional problems may face a computational bottleneck. For instance, with long-term temporal relations it would be reasonable to think that the inference time would have taken an exponential factor on computation.

[4] In the article, the *sequence* operator is referred as *after* or *before* operators
[5] cf. Section 3.1

Top Level	Mid Level	Low Level
Make Tea	FillKettle	GraspKettle, CarryKettle, TurnonFaucet, FillWater, TurnoffFaucet, CarryKettle, ReleaseKettle
	GetIngredients	GoToCupboard, GetCupFromCupboard, GetTeaboxFromCupboard
	PrepareTeabag	GraspTeabox, OpenTeabox, PutBagIntoCup
	BoilWater	TurnOnKettle, TurnOffKettle
	PourHotWater	GraspKettle, PourWaterIntoCup, ReleaseKettle

Table 2.1: Hierarchy of event from [Son+13].

The model proposes certainly more complex relationship between observations, activities and behaviours compared to [Ska+11], but, still, most of the rules of the model are just composed of two levels of deductions from observations to behaviours.

Song et al. [Son+13] proposed a method to help task recognitions and tracking of these tasks using MLN with different sources of information: one from visual data and another from language. They specifically focused on the impact of a hierarchical representation of events to recognised complex tasks or helping to fill the gap of undetected or erroneous LLEs. This hierarchy is composed of three levels of events where each event from a level may be combined with temporal constraints defined with Allen's algebra. An example of this hierarchical structure for the activity *MakeTea* is provided in Table 2.1. While visual data might only detect LLEs, the language source, which is the subject's utterances, might detect mid-level or high-level events making possible to confirm, discover or invalidate LLEs regarding the results of the visual data. Still, the problem dimension remains quite small with approximately 100 LLEs and three levels of deduction.

Recent works from Gayathri, Easwarakumar, and Elias [GEE17] provide an application for activity recognition in smart homes. One major difference from previous works is that the input LLEs for MLN are not temporal data points but temporal intervals. A preprocessing phase identifies time intervals of interest, which will be the input LLEs. Due to this fixed interval segmentation, it is possible to construct relations

between events using Allen's algebra, but they will be represented as hard constraints and may be, in fact, preprocessed. Furthermore, rules or structure of the model are not provided so it is quite difficult to estimate the complexity of the problem solved with the MLN.

Snidaro, Visentini, and Bryan [SVB15] proposed to use MLNs and Allen's algebra to fuse data with different level of confidence and applied on maritime awareness. They identified complex activities like a vessel docking or two vessels meeting in the sea. Time is not represented directly in the model so time properties are described more conceptually. For instance, the predicate `overlaps(V1,V2)` indicate that the period of time where the vessels `V1, V2` appeared in sensors overlapped, but no precise time-point is provided. Similarly, space relations are defined without precise position but using a predicate `proximity(Vx,Vy)`. Even if experimentations show good estimations of CE, the model remains quite small with only three levels of deductions and only five vessels involved. Therefore, it is not possible to say if the approach could support a larger model.

2.2.3 Ad-hoc approaches

Numerous approaches exist that use neither standard previous works on CEP nor Allen's algebra, but remain related to complex events recognition.

Sadilek [SK10; SK12; Sad12] applies MLN on a *capture-the-flag* game with the purpose of recognising behaviours of the players and actions between them by using only the GPS positions of each player and the rules of the game. GPS sensors have a precision of a few meters around a player, meaning their true position is uncertain. Furthermore, data from GPS are collected only every second, which means that a short action from a player might easily be missed. Time has an impact on the system but every action is assumed to be only dependant from the previous instant in time. This construction is similar to a Hidden Markov Model (HMM) or a dynamic Bayesian network (DBN). The full model is only made with three levels of deduction but with a local temporal dependency from previous instant. One specificity of this work is that it uses learning to discover new activities related to the ones already described. For instance, the authors describe the *capture* action that consist of capturing an adversary by touching them during the game. The rule for capturing is soft-constrained and is

described as follows:

$$\begin{aligned} \forall a_1, a_2, t : \quad & \text{enemies}(a_1, a_2) \wedge \text{onHomeTer}(a_1, t) \\ & \wedge \text{onEnemyTer}(a_2, t) \wedge \text{isFree}(a_2, t) \\ & \wedge \text{samePlace}(a_1, a_2, t) \implies \text{capturing}(a_1, a_2, t) \end{aligned} \tag{2.6}$$

meaning that if two adversaries are close enough, the player who is not on his territory is captured. As the rule is softened, it is not necessarily true, and an action of capturing might succeed, fail, or not happen[6]. So the authors want to learn new formulae describing each scenario. To do so, they use a structure learning method to construct automatically the CE representing failed captures using sets of failed and succeeded attempts. These news rules are supposed to be soft-constrained and weights may be learned through MLN techniques. This is one of the rare works that are interested in structural learning for CEP under uncertainty.

Wu and Aghajan [WA10; WA11] do not propose to recognise actions, but objects in rooms, regarding the behaviour of the persons in the room. This is not CEP *stricto sensu*, but it is closely related as it may be seen as a reversed process from activities to clues. For instance, if a person is sitting, it may imply that there is at least one chair or sofa. Furthermore, close objects influence classification of an object: if a TV is close from another object, this one might more likely be a chair than a microwave. These kinds of rule induce a sort of loopy relationship between objects, but no time relations exist in this model.

Most of the other applications based on MLNs remain quite similar to the presented works. MLNs are mainly used to fuse data from multiple sources [BTF07], to reduce uncertainty, or to define a model that does not capture completely correctly the relationships between LLEs [HNS11; HNS10; LDL17; TD08].

2.2.4 Summary on Markov Logic Networks approaches

In this part, we want to summarise which aspects of MLNs might be beneficial or not in order to deal with uncertainty in CEP. In addition, we want to discuss some aspects that were only partially, if at all covered by the literature.

[6] This would mean that the rule used is not sufficiently precise. Which is obviously the case for the equation 2.6.

It appears obvious that the main interest of MLNs comes from their syntax inherited from FOL providing an easy way to model an existing CEP representation. Even without prior knowledge of the CEP domain, other approaches, ad-hoc or interval-based, might make good uses of MLNs, since recognising an activity may generally be described using FOL as a set of clauses.

Furthermore, as MLNs are in fact a procedure to construct a random Markov field, they come with all the techniques existing in the field and, more specifically, techniques associated to weight learning. Modelling a system subject to uncertainty might be sometimes difficult as it may be hard to quantify this uncertainty, even with experts. Consequently, having techniques that learn the weights of the model is really useful in these situations.

In return, it appears from the literature that the expensive calculation time needed to perform inference leads to minimizing the size of the model. Consequently, existing models from CEP may not be translated into MLNs straightforwardly, as it might first seem.

Another negative point is the weight representation, which is often considered as confusing since it does not match trivially relative probabilistic confidence of the associated clauses.

Finally, we want to address some points that are almost never presented despite the interest they should have for the CEP community.

Performance in the different articles mainly considers precision, recall, or F-measure, when inference time is almost never discussed (Except for [MD11; TD08]), whereas it should be of interest in domains related to uncertainty in CEP. For instance, an important part of the literature is from the video analysis domain which is usually interested in online analysis, which requires computation to be performed in a matter of seconds at its longest. This lack of information on inference time is even harder to understand as many of these publications raise the issue of exponential growth in computation times for models like MLN, but usually almost no detail is provided about their computation time.

Correlated to computation time, scalability to more complex models is never discussed either. This might be explained by the number of studies focusing on a specific application rather than on solutions to make these methods robust on higher dimensions. Even works based on existing CEP methods, like [Ska+15b] do not discuss scalability, although this property may be highly correlated to the number of events

considered (the length of the data stream) or the complexity of the HLEs which is correlated to the number of rules used.

2.3 Bayesian network approaches

In this section, we discuss existing works focusing on Bayesian structures to capture or represent uncertainty in CEP.

Wasserkrug et al. [WGE05; Was+08; Was+12] describes a framework divided into two main parts: a selection part that filters the data flow according to a set of rules describing the CE and a Bayesian network computing the uncertainty. Formally, the language used to represent these rules does not matter, as long as it can be used to filter the data stream. In the data structure, LLEs are represented as tuples representing the attribute values for each LLE. LLEs might be certain or not. If an LLE is certain, it is represented by only one tuple, otherwise it is represented by a various number of tuples, where each tuple is a possible outcome for this LLE. Each of these alternative tuples is associated with a marginal probability. The probability sum of all element in a tuple should be smaller than or equal to one. If the sum is smaller than one, its complementary to one represents the probability that the event does not occur at all. Computation of uncertainty is performed separately from the selection by a Bayesian network, where probabilistic relations are automatically learned regarding the events selected and constructed at the previous step.

Performances are highly correlated to the selection process as the method looks for all possible outcomes that the rules entail, called worlds. The processing time would be linearly dependent on the number of worlds and the number of world might be exponentially dependent on the number of events (and attributes) and the rules. To avoid this problem, the authors proposed a sampling method that can be performed given a Bayesian network previously defined, allowing the method to bypass its construction.

Another approach using Bayesian networks was provided by Cugola and Margara [Cug+15]. Their work is based on a specific language designed for CEP: TESLA [CM10]. They extend it to a new model able to deal with uncertainty: CEP2U. CEP2U is supposed to deal with the two kinds of uncertainty that we previously mentioned: data and pattern uncertainty. More specifically, CEP2U deals principally with attributes of events, which is not so common without considering finite and discrete

sets of attribute. Consequently, the specific order of events in time is less important in this model.

The semantics of this model looks like a SQL language for rule definitions and is directly inherited from the TESLA model. For instance, consider the following rule:

```
define  TVS_Malfun()
from    Oxygen(km=$a) and
last Temp($a-10 < km < $a+10 and value>30)
within 5 min from Oxygen
```

which represents a malfunctioning of a Tunnel Ventilation System (TVS) that induces decreasing concentration of oxygen and higher temperatures. `Oxygen` and `Temp` are two LLEs detected by sensors. The `Oxygen` event-type detects an abnormally low concentration of oxygen around a specific place in the tunnel given by the attribute `km`. The `Temp` event-type measures the temperature and stores it in the attribute `value` at a specific position in the tunnel provided by the attribute `km`. The `TVS_Malfun` is event-type recognised when the last detection of the temperature indicates that it was higher than 30 degrees and a low level of oxygen was detected closely, with these two events appearing together in a five-minute window of time.

Uncertainty on events is represented using a marginal probability, while attribute uncertainty is represented with a specific probability distribution. For instance

```
Temp@10 %0.9 (km=<16.2, U(-1, 1)>, value=<24.5, N(0,1)>)
```

describes the LLE associated to a temperature measurement where `Temp@10` represents the type of LLE followed by its ID and `%0.9` the probability of confidence on this event. The event is associated with two attributes `km` and `value`. Attributes might have a certain degree of erroneousness, so each attribute is provided with the actual measurement (24.5 for the temperature) and a probability distribution quantifying the possible variations around the measurement (standard normal distribution around 24.5 for the temperature).

Probabilities are then computed for every case in the rule implying an uncertainty value from an attribute. Each constraint is supposed independent and the probabilities are then multiplied to compute the overall probability. For instance, for the previously defined rule:

$$P_{Malfun} = P_{detect} \cdot P(-10 < X_{Temp.km} - X_{Oxygen.km} < 10) \cdot P(X_{temp.value} > 30) \qquad (2.7)$$

where P_{detect} is the probability of correct detection, which is here 0.9.

Rule uncertainty is computed separately using a Bayesian network which serves to extend the relations between LLEs and the composite CE to include external factors that might influence the observations. For instance, a TVS malfunction induces a rising temperature and a lower oxygen concentration. But these two parameters might be induced by a traffic jam which may impact the probabilities of a TVS malfunction.

The structure of the Bayesian network is automatically extracted from the rules and, without tuning, does not modify the probability distribution from the initial model. To influence them, it has to be tuned by domain experts.

At the end, the model will produce two probabilities: one from the events and one from the rules. By design, authors assume that these probabilities are independent and may be merged by multiplication as they represent uncertainty from distinct elements. But concretely, it seems quite difficult to tell where the limit is between rules and events uncertainty, especially since probabilities from the events are computed using conditional properties such as `temp.value > 30`. Consequently, asserting that these probabilities are independent seems to be a rather strong assumption.

An interesting use of Bayesian networks is proposed in [MM07] by Muncaster and Ma. The authors defined a specific structure of DBN called hierarchical DBNs, which define relations between HLE and LLE. Figure 2.2 presents the representation of the model. $X_t^{(d)}$ variables represent the states of the system, where t is the time and d the complexity level. Consequently, X_t^{HL} might be seen as CEs and X_t^{LL} as LLEs. X_t^{PH} represent *phases*, which might be seen as sub-events that should be triggered before an LLE and defined using a Coxian phase-distribution regarding X_t^{LL}. Formally, the LLE might pass through a number of phases n^{PH} and terminates when it reaches the final phase. Probability distribution is defined as follows:

$$\Pr(E_t^{PH} = 1 | X_t^{PH} = i) = \begin{cases} 1 & i = n^{PH} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

$$\Pr(X_t^{PH} = j | X_{t-1}^{PH} = i, X_t^{LL} = k) = \begin{cases} 1 - p_{i \to i+1}^{PH,k} & j = i, i < n^{PH} \\ p_{i \to i+1}^{PH,k} & j = i+1, i < n^{PH} \\ \pi_j^{PH,k} & i = n^{PH} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

where, in context k, $p_{i \to i+1}^{PH,k}$ is a free parameter representing the probability of advancing to the next phase from phase i and $\pi_j^{PH,k}$ is the probability of starting the sequence

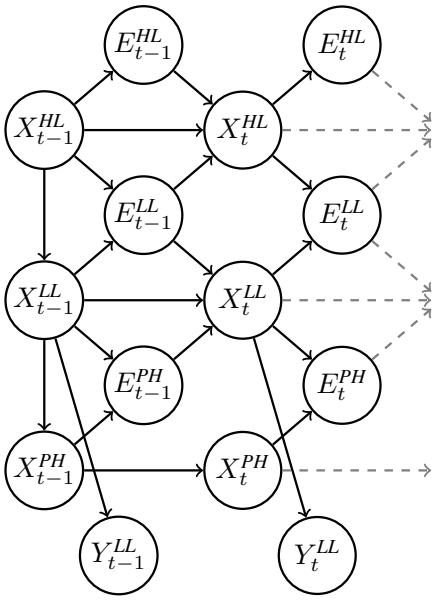


Figure 2.2: Model HDBN from [MM07]

in phase j.

X^{PH}_t may be seen as lower level events that might not be directly observed. Furthermore, X^{PH}_t may be seen as a representation of sequential time constraints as they might represent successive steps to assert before a change of state from a X^{LL}_t may be triggered. $E^{(d)}_t$ are binary variables representing activation constraints from bottom to top of the levels of states, which may be seen as rules for deducting HLEs, as these variables embed the constraints allowing a CE to be triggered according to the state of lower events. Finally, the Y^{LL}_t variables are vectors representing observations from the video, assumed to be drawn from a Gaussian process.

This method proposes an interesting way to deal with uncertainty, especially since the observable variables are not directly the lowest "events" on the system and may represent sequential constraints that are needed to trigger the observed LLEs. A similar approach exists in [MHZ10; Man09] using relational DBNs. This method is quite similar to the work of Muncaster and Ma, but dependencies between different levels of events are not just linear but defined as a tree structure where the inner nodes correspond to FOL expressions. This provides a broad expressiveness to the model.

Wang and Ji propose an approach for video analysis that uses DBN to introduce contextual information to help recognising events [WJ12; WJ14]. It is not stricto sensu CEP, but it possible to see some correspondences. Indeed, some of the contextual

information used is the surrounding objects that are detected on a frame at a specific moment and might interact with agents on the frame. Furthermore, events from the previous frame influence the present recognition of events. These dependencies are learned with supervised MAP.

The main difference with CEP is that, even if these events might be seen as complex in a sense that the concept is not completely clear for a computer like "*getting into a vehicle*", they are not a composition of sub-events, but are rather directly provided by usual techniques from video analysis. In a way, this approach is similar to the work of Biswas, Thrun, and Fujimura [BTF07] using MLNs to combine multiple sources of information.

The work from Wang, Gao, and Chen [WGC17] differs a bit from other approaches, since the authors do not aim at dealing with uncertainty but at anticipating a potential future behaviour. In their approach (cf. Figure 2.3), different event sources emit LLEs on potentially multiple data streams. These streams are parsed using Basic Event Processing Agents (BEPA) that recognise CE occurrences. BEPA are linked to Event Channels (EC) that might be used by other BEPAs. Finally, all the information of interest is sent to the Probabilistic Event Processing Agent (PEPA). The skeleton is, basically, a database of statistical dependences between recognitions. From then, the idea is to infer the probability of each CE type of interest from a Bayesian network. This Bayesian network, both parameters and structure, is learned and evolves according to data provided continuously.

Even if the Bayesian network structure is learned using a reduced greedy search approach which may impact the probability estimation accuracy, it is a really interesting approach since it does not depend of the CEP representation used and is constantly learning from data streams to identify correlation from preprocessed stream using BEPA.

2.4 Automaton-based methods

We present in this section different methods that rely on automaton structures to represent and reason under uncertainty. Most of these approaches are either extensions of the SASE+ model presented in section 1.2.2 or the works of Albanese et al.

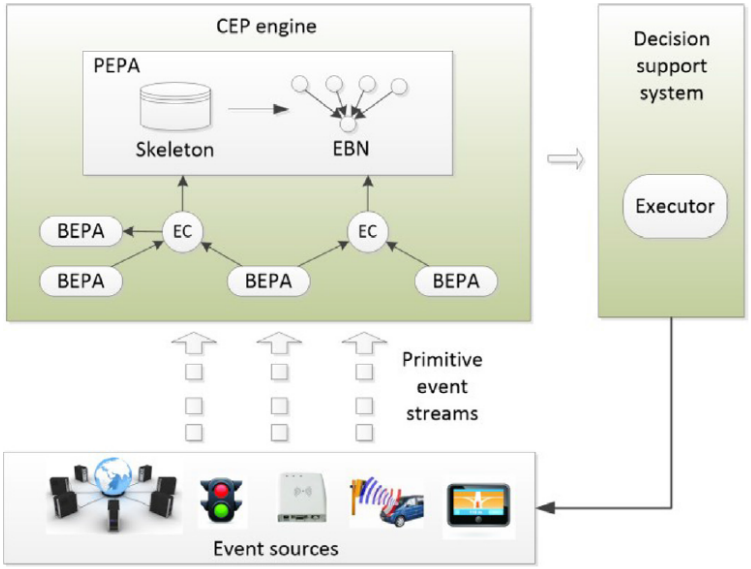


Figure 2.3: Architecture system proposed in [WGC17]

2.4.1 SASE Approaches

Shen, Kawashima, and Kitagawa [SKK08] propose to use the SASE model, and more specifically the NFA structure to compute recognition uncertainty. The authors suppose that LLEs are provided on a discrete time stream as a set of alternatives at each instant. Each alternative has its own probability and is independent to each other.

Using the query in its NFA form and the stream, they compute an Active Instance Graph (AIG) that records the set of actives states. The AIG is an acyclic directed graph that represents all possible combinations between all potential events. An example of these different representations is provided in Figure 2.4. The NFA represents a sequence of LLEs with a Kleene closure on the B-type event. Formally, it would be described as the clause SEQ(A a, B+ b[], C c) with the SASE syntax. Incoming LLEs are provided through the probabilistic event stream producing the AIG. As may be seen, the alternate event $b_{1,12}$ does not appear on the AIG as it does not match with the recognition process represented by the NFA, which should start with an A-type event. When the full AIG is constructed, each termination state and its anterior states are potential recognitions of the CE. More precisely, a path from 0-state to a termination state is exactly one potential recognition. Computation of the CE probability is performed as follows: each node in the graph is associated with its marginal probability, the probability of a path from a termination node to a 0-node is the product

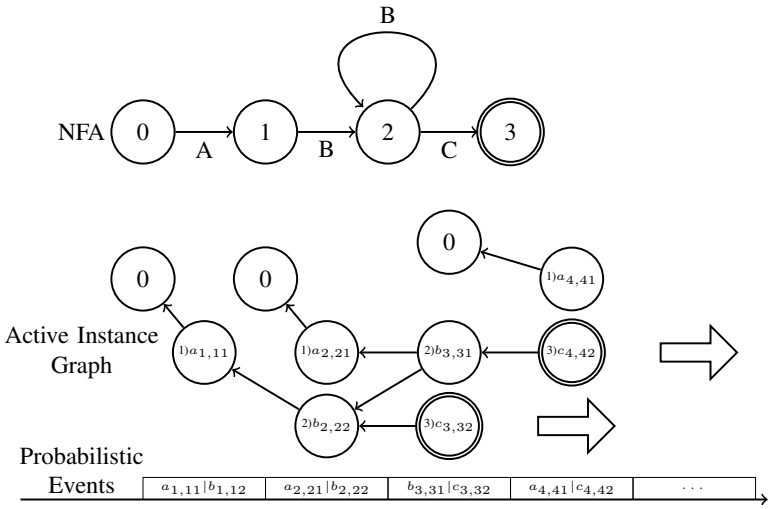


Figure 2.4: NFA to AIG [SKK08]

of the nodes along the path, and the probability of the CE is the sum of all paths. This is the main idea for the computation, but the tree structure avoids an expensive calculation of all paths. For instance, on the $b_{3,31}$ node, if the probability of streams $(a_{1,11}, b_{2,22})$ and $(a_{2,21})$ is computed, the cumulative probability of all the streams on the $b_{3,31}$ node is the product of its marginal probability by the sum of the probabilities of each stream. Formally, computation might be performed linearly on the number of nodes and with more subtle optimisation might be performed using constraints validation or threshold to reduce the AIG size. Threshold optimisation has been used by Kawashima, Kitagawa, and Li in [KKL10] and filter optimisation and variation of the AIG structure have been used by Chuanfei et al. in [Chu+10].

Even if this model is really interesting, mainly for its speed [SKK08], it assumes two strong hypotheses of independence between events. The first hypothesis is the independence assumption between alternatives at a given time, which supposes consequently that two events can not be produced simultaneously. The second hypothesis is the independence assumption between events from an instant to another that may be unrealistic for many applications that need to represent this time dependency.

Still based on the SASE model, Zhang, Diao, and Immerman [ZDI10] proposed an approach to deal with imprecise timestamps. Concretely, LLEs are certain, but the given time is not, meaning the real time of the corresponding activity is somewhere within a specific interval around the observed time. As time is supposed to be discrete, each possible instant for a LLE has its probability defined by a probability mass

function, which is uniform by default. The authors' goal is to provide a probability confidence on each *match signature* of a query. A *match signature* is defined as the unique sequence of LLE in a match. It simply means that a CE may be recognised on many different timestamps, but composed with the same sequence of LLEs at different instants. So the authors want to compute the confidence in the recognition of each match signature, and not all recognitions.

Two frameworks are proposed for the computation of CE confidences. The first framework is point-based: it considers each possible time point for an LLE independently and try to reconstruct the recognition pattern based on these points. This method is costly when the number of time points to consider is high. Formally, if LLEs are detected at time t and the uncertainty interval for these events is $[t-\delta, t+\delta]$, in the worst case complexity is $O(\delta^l)$ with l the pattern length. The second framework is an event-based approach that looks for possible lower and upper bounds for each LLE associated to a match signature. These bounds allow the computation to prune efficiently the solution and seems more scalable on variations on the size δ of the uncertainty interval.

Unfortunately, this approach still has some drawbacks since it does not seem highly scalable on the pattern length. Authors experimentations showed the approach to be intractable for pattern length above 6, especially for the event selection strategy *skip till any match*. Furthermore, it is impossible with this method to use negation or Kleene closure. The Kleene closure for imprecise timestamps has been addressed in more recent works [ZDI14], but the computation cost might be easily exponential and does not seem tractable. Nonetheless, Zhang, Diao, and Immerman proposed an approach to the problem of imprecise timestamps, which is a subject that is rarely addressed along the other works.

Wang, Cao, and Zhang [WCZ13] extend the approach proposed in [SKK08]. First of all, the authors provide new operators to construct the CE pattern like `ANY`, $\neg$, `AND`, `FIRST`, and `LAST`. The authors allow execution of parallel streams. The model supports the hierarchical definition of CEs and parallel computation. Finally, the framework removes the independence assumption between events in time and assumes that events may follow the Markov property. The probability between LLEs between two instants is provided by a conditional probability table. This method solves many problems that were present in the work from Shen, Kawashima, and Kitagawa and still performs fast computation (thousand of events per second). Unfortunately, results may be difficult to interpret. Experimentation are set using a meta parameter λ that measures the rate

of events appearing in the pattern. Formally, for a given event in the set of primitive events $e \in E$, $\lambda_e = \frac{\text{count}(e)}{\sum_{e' \in E} \text{count}(e')}$, where $\text{count}(e)$ counts the apparition of e in the stream. The parameter λ is calculated as the sum of λ_{e^*} where e^* is an event appearing in a recognition. This parameter has a huge impact on scalability since the method seems intractable above $\lambda > 0.25$. But, it seems difficult to estimate the values that this parameter might take in a real application. Furthermore, if the robustness to large window size is demonstrated, no experimentation is performed on the impact of the pattern length, whereas it was shown to have a big impact in [SKK08]. Nonetheless, it is a promising approach with apparently great performance, which suppresses part of the independence assumptions by using the Markov property.

2.4.2 Automaton-based representations from Albanese et al. and its extensions

In 2007, Albanese et al. proposed an approach based on probabilistic automata [Alb+07]. In this model, LLEs are not uncertain and CEs are represented by an automaton with probabilistic transitions. Even if it may be seen like a Markov chain, in fact, in these automata, time is not formally represented and an automaton just represents all possible chains of processes allowed for explaining a CE. Consequently, probabilities represent chances of going from a state to another, not at each instant, but in general. For instance, the authors describe a CE for a regular operation on an ATM: after inserting the card, two states are possible: *withdrawing cash* or *inserting checks*. The first option has a 0.7 probability and the second a 0.3 probability. Time elapsed between the moment the card is inserted and one of the two options does not matter and may be as well seconds or minutes.

This model was later extended in [Alb+11] and time representation were added. Formally, a transition between two states takes a specific duration. This duration is supposed to be always identical and does not have any influence except to restrict durations of CEs. The main goal of the paper is not to find probabilities of the activity, which was already the objective in [Alb+07], but to identify unexplained activities. Formally, the idea is to find activities not described by any automaton. To do so, the algorithm looks for all possible executions and computes their probability regarding the probabilistic automaton associated to the corresponding CE. Each recognition (called *occurrence*) may potentially be inconsistent with another occurrence, especially if the

same CE is detected during two overlapping periods of time[7]. Combinations of occurrences or their absence are seen as possible worlds and a probability of an occurrence is defined as the sum of the probabilities of world containing it. A sequence which has no explanation along the different worlds is supposed to be unexplained and extracted. This method may perform marginal or MAP inferences, but the representation of time is implicit and may not have constraints relative to time intervals.

In 2014, Molinaro et al. proposed PADUA[8] [Mol+14]. They introduced *probabilistic penalty graphs (PPG)*, which extend the stochastic activity definition of [Alb+11]. These PPGs handle noise during the recognition process using a noise degradation function ρ. In [Alb+11], the probability of a activity was the product of the change of states probabilities. These probabilities were represented using a $\delta(e)$ function that provides the change of state probability on a specific edge e of the automaton describing the activity. In [Mol+14], each edge of the automaton has an additional value $\rho(e)$ providing the decay in case of unexplained events. This decay is introduced in a score function associated to the recognition:

$$score(L^*, I^*) = \prod_{j \in [1, m-1]} \delta(l_{i_j}.action, l_{i_{j+1}}.action) \cdot \rho(l_{i_j}.action, l_{i_{j+1}}.action)^z \tag{2.10}$$

where $L^* = \langle l_1, \ldots, l_n \rangle$ a subsequence of LLEs that match a recognition, $I^* = \langle i_1, \ldots, i_m \rangle$ the indices of LLEs taking part in the recognition, with m the length of the recognition without irrelevant LLEs, and $z = i_{j+1} - i_j - 1$ is the number of irrelevant LLE between two LLEs taking part to the recognition. For instance, if an activity pattern is defined as the PPG in figure 2.5, knowing the sub-stream $L^* = \langle$PreFirewallAccess, PostFirewallAccess, X, MobileAppServerAccess, X, X, PostFirewallAccess$\rangle$ with X representing noisy events with, consequently, $I^* = \langle 1, 2, 4, 7 \rangle$, the score for this recognition is computed as follows: $score(L^*, I^*) = 1 \cdot 0.1 \cdot 0.2^1 \cdot 0.5 \cdot 0.9^2 \cdot 0.2 = 0.00162$. The authors define furthermore an *unexplained situation* that represents a partial recognition of a behaviour. In term of automata, it means that an event equivalent to a start state occurred on a stream but a terminal node was never reached. These unexplained situations may be recognised if the length is maximal and their scores are higher than a specified threshold τ. The authors propose, furthermore, different algorithms for fast and distributed computation.

[7] The authors consider it as an impossible situation, since they suppose that only one specific type of action might be performed at a given time by a single subject.

[8] Parallel Architecture to Detect Unexplained Activities

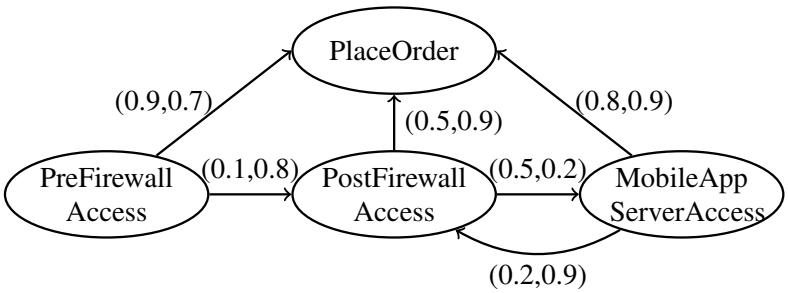


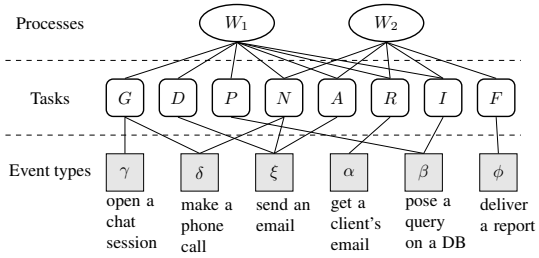
Figure 2.5: Example PPG definition of an activity from [Mol+14]. Edges are labelled with their δ and ρ values: $(\delta(e), \rho(e))$.

2.4.3 Automaton-based representation from Fazzinga et al. and its extensions

Some approaches that aims at dealing with uncertainty in data stream may be found in the Process mining community. In this perspective, an interesting work from this domain is provided by Fazzinga et al. [Faz+18b]. In this work, the authors suppose that the data stream is produced by distinct processes. These processes might be seen as workflows of tasks to achieve a specific objective. A simple example of that might be the refund process of a defective object from customer service point of view. This process follows a succession of tasks like getting information from the client. But in this scenario, the authors suppose that the events produced do not carry sufficient information to know which task they are associated to. For instance, in the refund procedure, the LLE detected might be the action of sending an email, action that might appears in other tasks like warning the managers of an issue with a customer. Figure 2.6 shows an example of relations between different tasks, processes and events. Note that this example provide no information as of the order of tasks: an edge between a process and a task indicates that the task may appear in the workflow[9] of the related process, and an edge between a event and a task indicates that the task may produce an instance of the connected event types.

This problem might be seen as a pattern uncertainty problem since the correlation between LLEs and tasks is fuzzy. In [Faz+18b], LLEs are provided with a probability distribution function p^a that provide the a-priori probability that a LLE e is generated by a task a. Furthermore, a process is a workflow of tasks described with two sequential rules. $X \Longrightarrow_\tau Y$ (resp. $X \Longrightarrow_\tau \neg Y$) describes that an instance (resp. no instance) of task Y should appear in an interval of time τ after each instance of task X. In this

[9]The workflow is not represented here but it is consider during the inference.



Activity label	Activity description
A	Send an alert to managers
D	Dispatch a contract proposal
F	Fix the issue
G	Get more information form the client
I	Retrieve data for the IS
N	Notify the request's outcome
P	Define a service package
R	Receive a request

Figure 2.6: An example of relationship between processes, tasks, and events. ([Faz+18a])

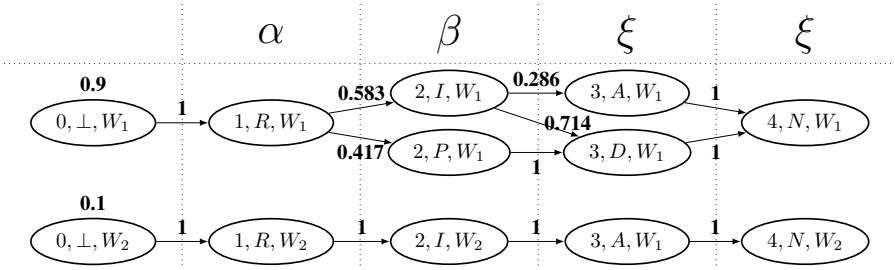


Figure 2.7: A CAS-graph example for a data stream $\varphi = \alpha\beta\xi\xi$. ([Faz+18a])

system, only LLEs may be observed, while tasks and processes might not. It is possible to consider that tasks and processes are equivalent to some CEs, but where only the relation between tasks and LLEs is imprecise.

When the program receives a data stream $\varphi = \alpha\beta\xi\xi$, it tries to deduce the probability that processes ended or the most likely explanation for the data stream. In [Faz+18b], calculation is performed using a MCMC approach that generates valid interpretations of the data stream using the probability distribution functions from the LLEs and asserting, at the same time, order constraints enforced by the processes. In [Faz+18a], authors proposed an approach similar to that of [SKK08] and [Alb+07] by constructing a *conditioned activity sequence graph* (CAS-graph) that represents the possible explanation of a data stream. Calculation of probabilities and suppression of inconsistent explanations is performed using a forward-backward algorithm that, first, tries to expend all possible solutions iteratively regarding the constraints and, second, removes branches that did not reach the end of the data stream and equilibrates the probabilities. Figure 2.7 is a example of CAS-graph based on the relationship represented in Fig. 2.6 where the two processes W_1 and W_2 have the following restriction: both processes should start with a task R and conclude with a task N. Process W_1 is designed with two more constraints : $P \Longrightarrow D$ and $D \Longrightarrow \neg P$.

In [Faz+18a], further extensions are proposed. The authors extend their model so that tasks might be represented by a succession of LLEs and not just only one per task. In fact, the order of LLEs in a task is modelled by an automaton that provides constraints during the construction of the CAS-graph, but the principle remains quite the same. The second extension introduces a notion of erroneousness in the model and allows the program to produce a graph with inconsistency. The construction algorithm explicitly allows a certain number of mistakes per path in the graph. These mistakes may be almost any kind of violation like rules broken in a process, LLEs generated from a task that should not, etc. Each erroneous path is assigned with a loss coefficient dependant of the number of violations observed along the path. Even if this extension is interesting, the loss coefficient is arbitrary and provides no distinction between the different types of violations.

The method proposed by Fazzinga et al. is certainly an interesting and complete approach since it proposes a fast method (few milliseconds) to compute MAP inference and deduce which process was in progress on a specific data stream, but it remains circumscribed to particular problems due to its structure. For instance, the intermediate level represented by the tasks seems to bring more restrictions than the case where processes would be defined directly from the LLEs. It surely offers a certain level of abstraction but it degrades the expressiveness of the model[10]. Finally, it remains difficult to grasp how expressive the model might be, since it is mainly focused on sequences of events.

2.5 Other approaches

Many other approaches exist and, in this section, we present some of them briefly, as they may use different tools to deal with uncertainty. But in some way they may be related to previously presented works. Indeed, even if these approaches are based on ProbLog [Kim+11], HMMs, Petri-nets, or grammars, their representation may easily seen as equivalent to MLNs, Bayesian networks, or automaton-based methods with a few modifications. As for the previous presented approaches, the main difference comes from the understanding of uncertainty and what problem is tackled.

[10] Authors showed it possible to represent tasks as automata on LLEs, so why not directly represent the processes as automata too

2.5.1 ProbLog method

Skarlatidis et al. proposed to model uncertainty problems using ProbLog[11] [Ska+15a]. The CE representation is based on Event Calculus and ProbLog is used to compute uncertainty on the LLEs and propagate it along all possible recognitions of CEs. The EC model is represented into the ProbLog formalism, mainly based on Horn's clauses, allowing computation of the theoretical exact probability for each CE. The authors compared resilience to noises on the data-stream between their probability version of EC and a non-probabilistic one, which imposed to use a threshold on the results provided by the probabilistic version. Results showed the probabilistic approach in order to be more robust than a non-probabilistic approach. This method is relatively similar to the MLN approach proposed in [MD11], even if ProbLog and MLN seem, intuitively, different tools to manipulate uncertainty. Indeed, both approaches rely on techniques from the SAT community to find solutions of a logical problem and then use these results to compute the probability of a CE.

2.5.2 HMM

Ré et al. proposed an approached based on HMM sampling and particle filter named Lahar [Ré+08]. As in many other works, authors focus on RFID deployment and try to fill uncertainty produced by inaccurate sensors. In the context of the article (cf. Figure 2.8), RFID sensors locate positions of employees inside a building and Lahar detects their activities like making a coffee, mostly using their positions along time.

On their approach, CEP is used over a *probabilistic event database*. A probabilistic event database is composed of streams where a stream is a set of events with the same type and key. For instance, a stream based on the event InRoom(*person*,*room*,T)[12] lists all the positions of a person *person*. Each event is associated with a probability regarding its attributes values and principally, on their use case, localisation. A sequence of event on a stream might be independent or Markovian[13], but events between streams are necessarily considered independent. The probabilistic event database is produced using a particle filter.

The particle filter operates with the collected sensor data as input. Given a time and a person's position, the algorithm tries to predict the next position by sampling regarding the distance between the previous position and the new one observed by

[11] cf. Section 3.2 for details on ProbLog.
[12] The event key is underlined.
[13] The choice to consider independence between events depends if the Lahar system is used online or not.

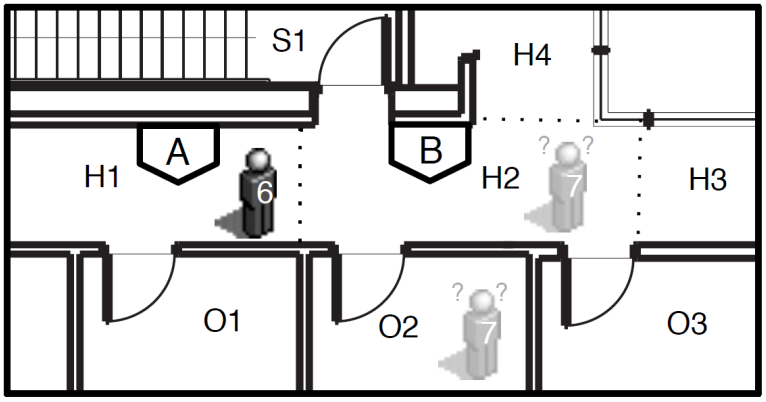


Figure 2.8: Schematic representation of the study case in [Ré+08]. RFID reading has detected the person at instant 6 in the H1 zone, but its position has been lost due to imprecise reading at time 7.

the captor. Each possible position is then weighted regarding the number of samples (or particles) at the corresponding position. This defines a set of possible places along with their probabilities for the next time, which will be reused to sample the following instant. In the case of independent streams, particle generation only depends on the input value of the sensors. When the process terminates, the probabilistic event database is complete.

CE recognition is performed on the newly constituted database using Cugoya's semantics [Dem+06] to define CEs. Briefly, Cugoya's system supports sequences, Kleene's plus and may define conditions on the attributes or relations between elements in the database. For instance, the following query:

$$\begin{aligned} q_{\text{AnyCoffee}} = & \sigma_{\theta_1}(At(\underline{p}, l_1); At(\underline{p}, l_2)^+ \langle \{p\}, \theta_2 \rangle; At(\underline{p}, l_3)) \\ & \text{where } \theta_1 = \texttt{Person}(p) \texttt{ AND } \texttt{Office}(p, l_1) \texttt{ AND } \texttt{CRoom}(l_3) \\ & \text{and } \theta_2 = \texttt{Hall}(l_2) \end{aligned} \tag{2.11}$$

looks for any person that went from his office to the coffee room passing by the hall. A query is defined with sub-goals like *At(person, room)* that define a relation between attributes of events without time considerations. θ_1, θ_2 are conditions on the query. The Kleene operator is defined with a specific syntax that specifies the condition and shared variable along for each event captured by the Kleene operator. For instance, in equation 2.11, sub-goal $At(\underline{p}, l_2)^+ \langle \{p\}, \theta_2 \rangle$, which captures events where a person passes through a hall, has a shared variable p, which should always be the same between the captured events, while l_2 is free. Consequently, this operator may capture

when a person passes through different halls.

The authors proposed four different algorithms associated to different types of query, namely Regular, Extended Regular, Safe or others[14]:

- **Regular Queries** assert that sub-goals share no variable. Authors show that this type of query may be solved linearly on the size of the database using a transformation from the query to NFA and then to Markov chain.
- **Extended Regular Queries** assert that if a variable is shared between sub-goals, it is a shared among all sub-goals and is a key in each of them. The algorithm is the same as for regular queries, except that it should be performed on every possible value of the shared variable, leading the algorithm to create a Markov chain for every possibility.
- **Safe Queries** assert that queries that where not at least extended regular might be defined as a sequence of extended regular sub-queries that allows the algorithm to solve these queries in $O(|\mathcal{W}|^2)$ with $\mathcal{W}$ the probabilistic event database.
- **Other Queries** are computed using a sample algorithm that runs the query multiple times on the database and counts when it is satisfied.

Even if this approach is only based on sequence and Kleene operators to define temporal relations, it introduces some interesting ideas. Surprisingly, this work is one of the few that use particle filters to produce conditional probabilities from an event to the next one. Other approaches usually prefer to represent failure detection of sensor with prior marginal probability. Furthermore, this study provides efficient algorithms for the three main categories of queries, even if queries that do not enter these types are more or less dismissed.

2.5.3 Petri-net approaches

Some works propose using Petri nets to represent CEs or/and uncertainty. A Petri-net is defined as a tuple (P, T, F) where P is a set of *places*, T a set of *transitions*, F a set of *arcs* with $F \subset (P \times T) \cup (T \times P)$. In the example, in Figure 2.9, *places* are represented by the blue circles, the *transitions* by the black rectangle, and the *arcs* by the edges.

For instance, Albanese et al. [Alb+08] represent a CE as a Probabilistic Petri Net (PPN). A PPN is a classical Petri net but allows setting probabilities on initial places and transitions. Computation of probabilities follows a set of rules regarding the

[14] Queries that do not belong to the three previous categories.

topography of the Petri-net. When firing a transition with a constrained transition probability, the resultant probability is the product of the probability computed at the previous place times the transition probability (cf. Figures 2.9a and 2.9b). In case of concurrent transitions (cf. Figures 2.9c and 2.9d), places derived from the transition share the same probability as the previous place. And, in case of synchronization (cf. Figures 2.9e and 2.9f), the derived place probability is the product of the previous places probability.

Probability p is computed regarding a specific CE and a data-stream by following the sequence of states along the Petri-net. Then, the *relative probability* p_{rel} is computed by $p_{rel} = \frac{p}{p_{max}}$, where p_{max} is the maximum probability achievable by the Petri net regarding all possible sequences. The authors provide a method to compute this p_{max} probability.

Even if this approach is based on Petri net representations of CE, uncertainty is computed in a way quite similar to the automaton-based approaches proposed by Albanese [Alb+07; Alb+11].

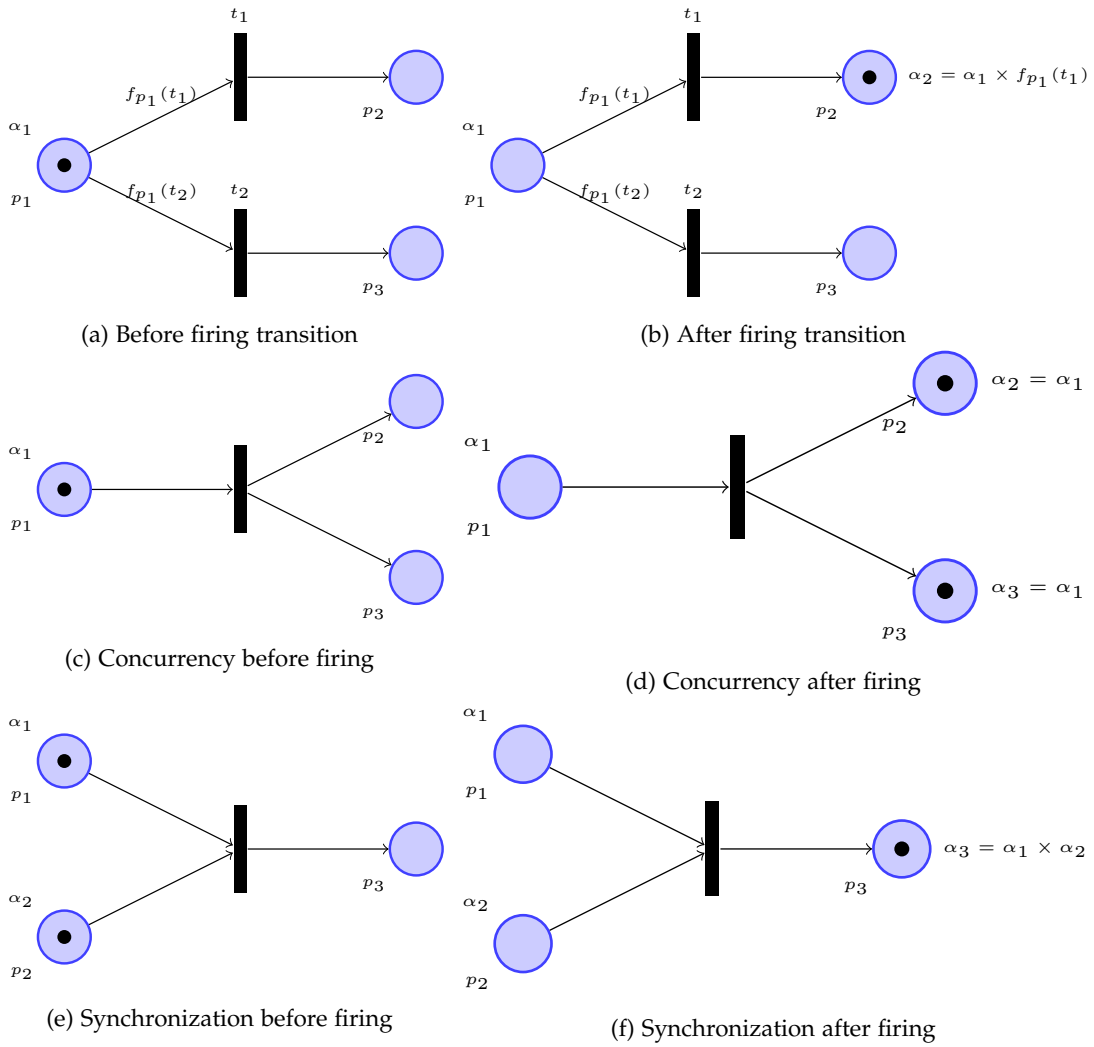
Another Petri-net-based approach is proposed by Lavee, Rudzsky, and Rivlin [LRR13] where, as in [Alb+08], CEs are represented by a Petri net, but this formalism is not really used for uncertainty computation afterwards. The structure is used to define a Bayesian recursive filter to estimate the confidence on the designed CE regarding observations provided by the data stream. The authors suppose the first-order Markov assumption and assume the model to be dynamic, since the state of the system is supposed to be dependent of its previous state in time. This approach is really similar to [MM07] but the relational structure between LLEs is initially defined by the Petri net.

2.5.4 Grammar approaches

Finally, we want to present some works based on grammar representations. One of the interesting aspects of such an approach is that CEs are represented using context-free grammars, which means it is possible to describe theoretically a CE not represented by Deterministic Finite Automaton (DFA).

Most of the works using grammars are extensions of stochastic context-free grammars (SCFGs) [Sto95], which offer a framework to transform a grammar into a HMM representation. In this formalism, each production rule from the grammar may have a probability associated:

$$A \rightarrow \lambda[p] \tag{2.12}$$



(a) Before firing transition

(b) After firing transition

(c) Concurrency before firing

(d) Concurrency after firing

(e) Synchronization before firing

(f) Synchronization after firing

Figure 2.9: PPN computation rules from [Alb+08].

The production rule in equation 2.12 as a probability $p = P(A \rightarrow \lambda)$ to be chosen. Note that this probability is in fact conditional since it may be chosen only if the non-terminal A has to be expanded. Rules in a grammar are independent so the probability for a particular complete derivation is just the product of the probabilities of the rules used in the derivation.

SCFG computation is based on the Earley-Stocke parsing algorithm [Ear70], which uses a specific notion of *state*. A state S_k^i is defined as follows:

$$i : X_k \rightarrow \lambda.Y\mu \ [\alpha, \gamma] \tag{2.13}$$

where X and Y are non-terminals, λ and μ are sub-strings, and the dot "." refers to the current position in the input stream. Indices i and k indicate respectively the marker and the starting position of X. Variables α and γ represent *prefix* and *inner* probabilities. The prefix probability α is the probability of the parsed stream up to position i and inner probability γ is the probability of the sub-stream starting at k and ending at i.

The Earley algorithm performs three steps to expend the derivation tree, called *prediction*, *scanning*, and *completion*. The *prediction* step looks for possible derivation states regarding the leftmost states in the derivation tree. Then the *scanning* step matches the predicted state with the stream to keep only relevant states. Finally, the *completion* step updates marker positions according to the scanning. During this process, new probabilities α' and γ' are calculated. For instance, during the completion, if the position marker is at j, $X_k \rightarrow \lambda.Y\mu$, it might be advanced if there is a state starting at j where $i : Y_j \rightarrow \nu.$, which is a state consuming, after scanning, the v sub-string. Marker position is updated accordingly:

$$\begin{cases} lj : X_k \rightarrow \lambda.Y\mu \ [\alpha, \gamma] \\ i : Y_j \rightarrow \nu. \ [\alpha'', \gamma''] \end{cases} \implies i : X_k \rightarrow \lambda.Y\mu \ [\alpha', \gamma'] \tag{2.14}$$

and α' and γ' are computed regarding all possible derivations:

$$\begin{aligned} \alpha' &= \textstyle\sum_\nu \alpha(i : X_k \rightarrow \lambda.Z\mu)\mathbf{R}_U(Z, Y)\gamma''(i : Y_j \rightarrow \nu.) \\ \gamma' &= \textstyle\sum_\nu \gamma(i : X_k \rightarrow \lambda.Y\mu)\mathbf{R}_U(Z, Y)\gamma''(i : Y_j \rightarrow \nu.) \end{aligned} \tag{2.15}$$

where $\mathbf{R}_U$ is a reflexive transitive closure of unit production relation between non-terminals in the grammar which are simply the probabilities to produce a specific non-terminal through a given one.

SCFG is initially used to generate and select uncertain derivations, but cannot deal with missed detections and insertion errors into data-streams. Ivanov and Bobick [IB00] modified slightly this approach to deal with these kinds of error, allowing to provide, on data-streams, LLEs together with their recognition probability. Similar approaches based on stochastic context-free grammars are proposed in [ME02; MES03]. In addition, Ryoo and Aggarwal propose in [RA06] to construct HMMs through the grammar, but the main difference with [IB00] is the possibility to represent interval dependencies. In more recent work [RA09], probability uncertainty is computed using a Constraint Satisfaction Problem (CSP) approach, which provides optimal evaluation but is NP-hard to solve.

2.6 Conclusion

In this chapter, we have provided a detailed reading of the main objectives for CEP under uncertainty and more specifically shown that it might not be realistic to propose a method that performs well on every one of them. The variety of CE and uncertainty representations, the diversity of uncertainty computations, and the range of needs for each particular domain leads to this profusion of works and approaches.

Among all approaches, MLNs remain, indubitably, the most expressive. Conceptually speaking, MLNs may be used to model every possible structure. Furthermore, since this model manages uncertainty on logical rules, it is quite easy to represent hard constraints rules, which is almost only possible with MLNs compared to other approaches. But, as we discussed in Section 2.2.4, the models proposed by the community seem to be of reduced size and only few works provide information regarding the time performance of their approach. Additionally, uncertainty weight representation seems to be cumbersome to use properly when constructing a model without learning it. Finally, logical rule semantics may lead to a counter-intuitive understanding of relations in the model. We did not discuss this too much in this chapter, but it has been addressed by Jain in [Jai11] and we provide a good example to understand this problematic later on in this thesis in Section 5.1.

Even if, at first glance, it may be counter-intuitive, the ProbLog approach proposed in [Ska+15b] is quite related to MLNs. Indeed, both approaches focus on finding solutions for a FOL problem using solvers provided by the SAT community and both have almost the same level of expressiveness, since they both rely on FOL, and may define hard rules on the model. The main difference would be the probabilistic inference, which is based on approximate methods for MLNs and exact ones for ProbLog.

In contrast with MLNs, Bayesian networks or DBNs provide a simple representation of uncertainty with clear probabilistic dependences and come with efficient techniques to perform inference on these structures. But existing works do not completely mix this representation with a well-defined CEP semantic and seem to be more a collection of ad-hoc approaches for CE representation. This observation does not apply for the works based on CEP2U [Cug+15], but even for this approach, Bayesian networks are not used along with the CE model, but only as method to provide contextual information after the recognition of a CE.

While Bayesian network approaches provide better formalism for uncertainty but lack structured CE representation, automaton-based methods tend to suit well CE models. This seems quite understandable as automata provide an intuitive representation for successions of elements or repetition of patterns[15]. Computation of uncertainty is usually based on the enumeration of possible recognitions, with eventually branch and bound techniques. In our opinion, computations performed and uncertainty representations are closely related to DBN or HMM, but they are almost never addressed that way in the different works except in the work of Demers et al. [Dem+06]. HMMs are also used in other approaches quite similar to automata like those based on Petri nets and grammars.

[15]There is no surprise in finding quite often Kleene operator in the semantics of these models.

CHAPTER 3

Background

Contents

3.1 Markov Logic Networks 78
3.1.1 Markov network 78
3.1.2 Markov Logic 79
3.1.3 Inference 81
3.2 ProbLog 86
3.3 Non-homogeneous Markov models 90
3.3.1 Context 90
3.3.2 Construction of the NHM 91

IN the previous chapter, we introduced the state of the art for CEP under uncertainty and showed the diversity among the various approaches deployed to this end. For this thesis, based on our review, we tried several approaches and tools and studied their performance. As these tools come from the state of the art, they are not exactly part of this thesis, but it remains necessary to introduce them properly so as to understand the impact they might have on the models that we proposed.

Section 3.1 presents the MLN formalism and the Markov logic semantic that is used in Chapter 4. This introduction focuses on inference and specific details about its

sampling method based on WalkSAT. Section 3.2 introduces the fundamental aspects of ProbLog which is used in Chapter 5. Finally, Section 3.3 presents the concept of non-homogenous Markov models (NHMs) and, in particular, its construction method as presented in [PRB11].

3.1 Markov Logic Networks

As presented in section 2.2, plenty of studies used MLNs to integrate uncertainty into CEP formalisms or related domains. MLNs [RD06] aim at offering an easy method to combine probabilities and logic. Indeed, logic is an easy and understandable way to represent a problem or a system, but is limited when trying to express erroneous behaviour or uncertainty and lack of methods to produce evaluations of this uncertainty. Indeed, it is easy with logic to define that two things are equal or different, but difficult to express a degree of similarity. In contrast, Probabilistic Graphical Models (PGMs) offer well-established methods to deal with uncertainty, but it might be difficult to design complex relationships between elements. Usually, PGMs mostly capture correlation, but not necessarily causality, which might be easily described by logic. MLNs aim at combining these two domains with the purposes of benefiting of the expressiveness of both FOL and the probabilistic representation of PGMs. For short, the formalism of MLNs proposes a method to transform a set of formulae into a Markov network. We now present this formalism together with the existing inference techniques that have been used during this thesis.

3.1.1 Markov network

A Markov network or Markov random field (MRF) is a model representing a joint distribution of variables $X = (X_1, \ldots, X_n) \in \mathcal{X}$ [Kin80]. It is represented as an undirected graph G and a set of potential functions ϕ_k. Each variable is represented by a node in the graph and each clique between connected nodes is associated to a potential function. The joint distribution among variables in X is given by:

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}) \tag{3.1}$$

where $x_{\{k\}}$ is the combined state of the variables of the k^{th} clique and Z is the *partition function*, which is a summation over all possible values of X, expressed as

$Z = \sum_{x \in X} \prod_k \phi_k(x_{\{k\}})$ and assert normalisation of equation 3.1 by summation to one of all values of X.

The joint distribution might be rewritten in an exponential form using features f_k:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_k w_k f_k(x_{\{k\}})\right) \tag{3.2}$$

where w_k is a weight associated to the feature function. Features f_k are commonly seen as indicators on the value of the cliques. Exact inference in a MRF is #P-complete so, generally, approximate inference is preferred, using MCMC techniques like Gibbs sampling.

3.1.2 Markov Logic

In FOL, a first-order knowledge base is expressed as a set of formulae applied on a set of variables. A possible set of values for these variables is called a *world* and may either satisfy the knowledge base or violate it. In Markov logic, the logic semantics used to describe MLNs, a world may violate only partially a knowledge base. Formulae are associated with a confidence value indicating a trust factor of a formula. If the value is high, the formula should be less likely to be violated, while a low value indicates that a formula may be easily broken. Each formula in Markov logic is described as a FOL formula, but combined with a weight expressing the importance of the formula in the knowledge base. It is possible to represent a hard formula by setting the weight associated to an infinite value.

Definition 3.1.1. As defined in [RD06; DL09], a Markov logic network L is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$, a Markov network $M_{L,C}$ is defined as follows:

1. $M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground atom is true, and 0 otherwise.

2. $M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.

Based on definition 3.1.1, a MLN might be seen as a relational model between predicates that can be instantiated regarding a set of constants C, where constants

English	FOL	Clausal form	Weight
Friends of friends are friends	$\forall x \forall y \forall z\ \mathrm{Fr}(x,y) \wedge \mathrm{Fr}(y,z) \implies \mathrm{Fr}(x,z)$	$\neg\mathrm{Fr}(x,y) \vee \neg\mathrm{Fr}(y,z) \vee \mathrm{Fr}(x,z)$	0.7
Smoking causes cancer	$\forall x\ \mathrm{Sm}(x) \implies \mathrm{Ca}(x)$	$\neg\mathrm{Sm}(x) \vee \mathrm{Ca}(x)$	1.5
Friends have same smoking habits	$\forall x \forall y\ \mathrm{Fr}(x,y) \implies (\mathrm{Sm}(x) \Leftrightarrow \mathrm{Sm}(y))$	$\neg\mathrm{Fr}(x,y) \vee \mathrm{Sm}(x) \vee \neg\mathrm{Sm}(y)$, $\neg\mathrm{Fr}(x,y) \vee \neg\mathrm{Sm}(x) \vee \mathrm{Sm}(y)$	1.1 1.1

Table 3.1: Example of a first-order knowledge base and MLN. Fr() is short for Friends(), Sm() for Smokes(), and Ca() for Cancer() [RD06].

are truth values assigned to a set of predicates. Formally, each formula is valued, i.e. grounded, with each possible constant. The newly grounded predicates are the nodes of the network. Two nodes are connected by an edge if they appear in the same formula. Consequently, the probability distribution over possible worlds x is given by:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right) \tag{3.3}$$

where $n_i(x)$ is the number of true groundings of F_i in x. Formulae F_i are only expressed in Cunjunctive Normal Form (CNF) (or clausal form) and quantifiers are suppressed. CNF formulae are convenient for evaluation since they only need one true term to be true also: this representation allows for lazy implementation for inference [DL09]. While universal quantifiers might be directly suppressed as results of the grounding phase, existential quantifiers are suppressed through Skolemization, which modifies the formula by replacing existentially quantified terms by a disjunction of their groundings.

For instance, Table 3.1 present an example of a knowledge base expressing existing potential relations between being friends, smoking, and having cancer expressed in English, in FOL, and in CNF[1]. Each formula is associated with its own confidence weight as they might not be always true. Figure 3.1 represents the ground Markov network from the two last formulae in Table 3.1 with constants "Anne" and "Bob". Each clique on the graph is the graphical representation of a grounded formula in its clausal form.

[1]The reader may notice that the third formula in the table is separated into two formulae in CNF.

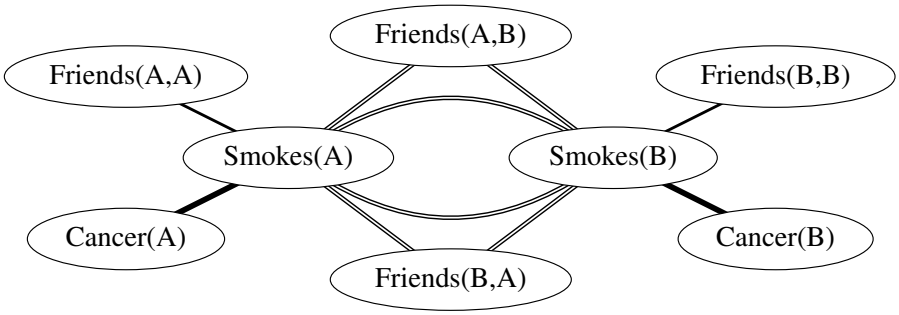


Figure 3.1: Ground Markov network obtained by applying the last two formulae in Table 3.1 to the constants Anna(A) and Bob(B). Larger edges indicates higher weights [RD06].

3.1.3 Inference

MLNs allow performing two types of inference: MAP inference, which will look for the most probable world consistent with evidences, and conditional inference of probabilities for predicates or formula over all possible worlds.

3.1.3.1 MAP inference

MAP inference aims at finding the most probable world x consistent with evidences e. Evidences are expressed as truth values of a subset of grounded predicates in the MLN L. Formally, the process is expressed as follows:

$$\begin{aligned} \arg\max_x P(e|x) &= \arg\max_x \frac{1}{Z_e} \left(\sum_i w_i n_i(e,x) \right) \\ &= \arg\max_x \sum_i w_i n_i(e,x) \end{aligned} \tag{3.4}$$

Computation for this problem is performed using weighted variants of FOL solvers. Theoretically, both exact and approximate might be used to this purpose. Richardson and Domingos suggested to use MaxWalkSAT to this task, a weight version of the well-known WalkSAT, a local-search satisfiability solver [SKC+93].

MaxWalkSAT is described as Algorithm 1. Its goal is to find the best possible assignment of all values for a knowledge base. As the algorithm does not necessary converge to the optimal solution (or reaches the given threshold), search is generally started several times to begin with different random worlds, which might lead to better solutions. This number of restarts m_t is bounded as an input of the algorithm. Moreover, since

Algorithm 1: MaxWalkSAT($KB, m_t, m_f, target, p$) [PD06]

inputs : KB, a knowledge base in CNF
m_t, maximum number of tries
m_f, maximum number of flips
$target$, target solution cost
p, probability of random step

outputs: $soln$, best variable assignment found

```
1  vars ← variables in L
2  for i ← 1 to m_t
3      soln ← a random truth assignment to vars
4      cost ← sum of weights of unsatisfied clauses in soln
5      for i ← 1 to m_f
6          if cost ≤ target
7              return soln
8          c ← a randomly chosen unsatisfied clause
9          if Uniform(0,1) < p
10             v_f ← a randomly chosen variable from c
11         else
12             for each variable v in c
13                 v_f ← v with lowest DeltaCost
14             soln ← soln with v_f flipped
15             cost ← cost + DeltaCost(v_f)
16 return soln
```

there is also no way to determine whether the solver will quickly reach a global optimal solution, each try is bounded to a number of flips m_f where a flip is a change of truth value of one ground predicate, this cap also being an input to the algorithm.

One try-iteration goes as follows. First, each predicate of the knowledge base is assigned to a random truth value. Then the solver will perform m_f flips on the knowledge base. Each flip is applied on a predicate from an unsatisfied clause. The choice of the predicate inside the clause may be done according to two strategies selected randomly with respect of an input probability p. If the first strategy is selected the predicate is randomly chosen in the clause, otherwise the solver chooses the predicate that minimizes DeltaCost(v_f), which is the function that computes the difference in the sum of the weights of the unsatisfied clauses. This function is easy to compute since the cost variation is conditioned only by the clauses connected to the modified literal.

Even if MAP inference is not used in this thesis, various methods has been proposed since then, with much better results at both precision and speed [Hur+16; Rie12].

3.1.3.2 Conditional inference

Theoretically, inference queries with MLNs might cover probability estimations of a formula F_1 conditionally to a second formula F_2 where F_1 and F_2 are not necessary in the knowledge base. Then, given a set of constants C and a MLN L:

$$\begin{aligned} P(F_1|F_2, L, C) &= P(F_1|F_2, M_{L,C}) \\ &= \frac{P(F_1 \wedge F_2, M_{L,C})}{P(F_2, M_{L,C})} \\ &= \frac{\sum_{x \in \mathcal{X}_{F_1} \cap \mathcal{X}_{F_2}} P(X=x|M_{L,C})}{\sum_{x \in \mathcal{X}_{F_2}} P(X=x|M_{L,C})} \end{aligned} \tag{3.5}$$

where $\mathcal{X}_{F_i}$ is the set of worlds where F_i holds, and $P(X = x|M_{L,C})$ is given by Equation 3.3. Generally, F_1 and F_2 are conjunctions of 0-arity predicates making the problem much simpler since it is not necessary to find logical solutions for F_2. Even if both query and evidence formulae are conjunction of 0-arity predicates, exact inference might be intractable except for a small domain. Consequently, approximate inference using MCMC estimation is preferred. Originally, in [RD06], the authors propose to use Gibbs sampling to perform inference where ground nodes are sampled in an arbitrary order given their Markov blanket. The Markov blanket of a ground atom is the set of ground atoms that appear in some grounding of a formula with it. The probability of

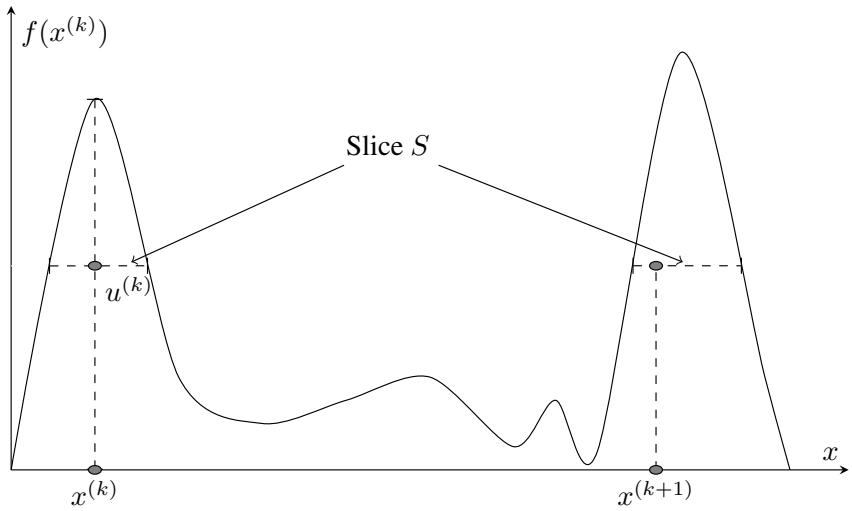


Figure 3.2: Graphical representation of a slice step during sampling.

a ground atom X_l when its Markov blanket B_l is in state b_l is:

$$P(X_l = x_l | B_l = b_l) = \frac{\exp(\sum_{f_i \in F_i} w_i f_i(X_l = x_l, B_l = b_l))}{\exp(\sum_{f_i \in F_i} w_i f_i(X_l = 0, B_l = b_l)) + \exp(\sum_{f_i \in F_i} w_i f_i(X_l = 1, B_l = b_l))} \tag{3.6}$$

where F_l is the set of ground formulas that X_l appears in, and $f_i(X_l = x_l, B_l = b_l)$ is the value (0 or 1) of the feature corresponding to the ith ground formula when $X_l = x_l$ and $B_l = b_l$.

MC-SAT The authors noted that Gibbs sampling breaks down when the knowledge base has deterministic dependencies[2] [PD06; DL09]. To solve this issue, they proposed another MCMC technique that samples called MC-SAT [PD06] based on satisfiability and simulated annealing.

Slice sampling [Nea03] aims at sampling a distribution for a variable x defined by a function $f(x)$. The sampling process is iterative and follows a few simple steps, given an $x^{(k)}$ value:

1. Sample a $u^{(k)}$ value uniformly between $\{0, f(x^{(k)})\}$.
2. Sample an $x^{(k+1)}$ value uniformly over the slice region $S = \{x : u^{(k)} < f(x)\}$ (cf. Figure 3.2).

[2] In case of hard constraints corresponding to formulae with infinite weights.

Algorithm 2: MC-SAT(L, n)

inputs : L, a set of weighted clauses $\{w_j, c_j\}$
n, number of samples
outputs: $\{x^{(1)}, \ldots, x^{(n)}\}$, set of n samples

$x^{(0)} \leftarrow$ Satisfy(hard clauses in L)
for $i \leftarrow 1$ **to** n
  $M \leftarrow \emptyset$
  for all $(w_k, c_k) \in L$ *satisfied by* $x^{(i-1)}$
    With probability $1 - e^{-w_k}$ add c_k to M
  Sample $x^{(i)} \sim \mathcal{U}_{SAT(M)}$
return $\{x^{(1)}, \ldots, x^{(n)}\}$

3. Repeat from the first step with the new value $x^{(k+1)}$ until enough samples have been produced.

Slice sampling requires $u^{(k)}$ to be sampled uniformly over the slice region S, which might be difficult to obtain.

The full MC-SAT algorithm is described in Algorithm 2. MC-SAT starts from a generated sample $x^{(0)}$ using a solving algorithm on hard clauses. Grounded predicates unaffected by hard clauses are sampled randomly within $[0, 1]$. Then, the algorithm selects a subset M of the satisfied clauses from this world $x^{(0)}$, where each clause (w_0, c_0) might be added to M with probability $(1 - e^{-w_0})$. Each clause in M will be temporarily considered as hard-constrained when a new world $x^{(1)}$ is sampled from the current world $x^{(0)}$. This process is, then, iterated until the required number of samples has been performed.

This algorithm needs to call a satisfiability solver during initialisation and sampling. The initialisation of $x^{(0)}$ is performed using WalkSat, which is required by the slice sampling approach since it produces a better overall sampling when the initial world is a good solution [Nea03]. Sampling of $x^{(i)}$ is performed using *SampleSat* which is a modification of MaxWalkSat with simulated annealing and is described below.

SampleSAT SampleSAT is proposed by Wei, Erenrich, and Selman [WES04] to sample uniformly over the space of satisfying assignments of a given FOL knowledge base. The solver uses a mixture of two strategies, selected randomly regarding a probability p, to flip a predicate. The first one is a random walk strategy using WalkSAT[3] and

[3]We did not present WalkSAT, but only its weighted version. Without weight the criterion used to determine the best predicate is the difference of satisfied clauses.

the second one is a simulated annealing [Met+53] approach to the FOL satisfiability problem.

Applied on this problem, for one iteration the simulated annealing algorithm selects a random neighbour assignment from the current state. An assignment is a neighbour for another one if it differs from only one truth value assignment on a predicate. If the cost variation ($\Delta Cost$, the difference number of satisfied clauses) decreases, this neighbour is selected. Otherwise, the algorithm chooses to move to this neighbour with a probability $e^{-\Delta Cost/Temp}$. $Temp$ is usually dynamic in this type of algorithm, but, in this particulat case, it is a fixed value[4].

Simulated annealing is known to be less efficient than WalkSAT [SKC+93; WES04], but the idea behind this hybrid method is to use WalkSAT in the first steps of solving to reach a solution and then to rely on temperature annealing to explore the cluster of satisfiable assignments. This hybrid method decreases the performance to find satisfiable assignment, but samples over the solution space much more uniformly than WalkSAT.

SampleSAT with MLN For conditional inference, the SampleSAT algorithm is slightly modified to deal with weights on MLN. Consequently, the algorithm uses MaxWalkSAT instead of WalkSAT and cost in temperature annealing is computed according to weights.

Since samples are supposed uniform, conditional probabilities for a predicate or a query might be evaluated by counting in which samples the query formula or predicates is satisfied.

In this section, we just discussed inference with MLNs, as it is necessary for this thesis, but state-of-the-art works on MLNs propose multiple algorithms for weight and structure learning, lazy inference, extensions to continuous domains, infinite domains or recursive models, etc. We will not discuss these matters, but readers may refer to [DL09] for further details.

3.2 ProbLog

In this thesis, we used MLNs to perform approximate inference to compute probability estimation of chronicles over uncertain executions of systems. Even if approximate inference is preferred to improve speed computation, MLNs unfortunately do not offer

[4]For SampleSAT, the authors manually tuned this parameter to 0.1 [WES04].

fast inference and attempts to make it more tractable bring important expressiveness restrictions [DW12]. Furthermore, approximate methods are not fully reliable and it might be difficult to estimate this reliability when solving large theoretical models without any real data to assert the results. In this situation, a tool performing exact inference might be beneficial to compare results provided with approximate methods. Moreover, it is interesting to study the limits of such approaches when applied on problems related to CEP. In this section, we present ProbLog [RKT07], the tool we used in this thesis to represent chronicles and achieve the previously specified tasks.

A ProbLog program follows a Prolog-like semantics except that every clause c_i is labelled with a probability p_i. For instance, Listing 3.1 describes appreciation of people regarding their friendship relations: if two people are friends, they should like each other with probability 1 and friends of friend might like each other with a probability 0.8.

```
1.0:: likes(X,Y):- friendof(X,Y).
0.8:: likes(X,Y):- friendof(X,Z), likes(Z,Y).
0.5:: friendof(john,mary).
0.5:: friendof(mary,pedro).
0.5:: friendof(mary,tom).
0.5:: friendof(pedro,tom).
```

3.1: ProbLog code example [RKT07].

Formally, given a ProbLog program $T = \{p_1 : c_1, \ldots, p_n : c_n\}$, the probability distribution over logic programs $L \subseteq L_T = \{c_1, \ldots, c_n\}$ is defined as:

$$P(L|T) = \prod_{c_i \in L} p_i \prod_{c_i \in L_T \setminus L} (1 - p_i) \tag{3.7}$$

A ProbLog program usually tries to compute the probability of a specified query q and, more precisely, its success probability $P(q|T)$ defined by:

$$P(q|L) = \begin{cases} 1 & \exists \theta : L \models q\theta \\ 0 & otherwise \end{cases} \tag{3.8}$$

$$P(q, L|T) = P(q|L) \cdot P(L|T) \tag{3.9}$$

$$P(q|T) = \sum_{M \subseteq L_T} P(q, M|T) \tag{3.10}$$

This corresponds to the probability that the query q has a proof θ given the distribution over logic programs.

In practice, this approach is intractable considering all logical programs needing to be computed. The common approach employs Selective Linear Definite clause resolution (SLD resolution), and BDDs. SLD resolution is used to compute proofs of a query on a logical program into a SLD-tree. Each proof is independent of others and is defined by a set of clauses that proves it. Consequently, the probability of this proof is $\prod_i p_i$, where p_i is the probability of the clause c_i that was used for the proof. Therefore, the probability of a query is the summation of the probability of its proofs. Since each proof may be seen as a conjunction of terms, a query may be seen as a disjunction of its proofs which is its Disjunctive Normal Form (DNF). Unfortunately, computing directly the DNF probability is quickly intractable, so it is preferred to transform the SLD-tree of a given query into BDDs.

A BDD is a oriented graph structure representing a logical formula where each node on the i^{th} level is labelled with the i^{th} variable in the formula[5] (given a specific order of the variables). Each node has two outgoing edges, namely *high* and *low*, which correspond respectively to a 1 and 0 assignment. The structure has only two terminal nodes, 1 and 0, representing whether the assignment[6] does or does not satisfy the formula. This structure may be easily derived from the SLD-tree. For example, given proofs for a query in a logical program represented by the formula $(\neg X_1, \neg X_2, \neg X_3) \vee (X_1, X_2) \vee (X_2, X_3)$ in DNF, a possible resulting BDD would be like the one represented in Figure 3.3. Note that, each X_i correspond to a clause $c \in T$ used to prove the query. The variable order used to construct the BDD may have a huge impact on the tree structure. Many heuristics exist to reorder automatically variables and reduce the overall structure.

Given a BDD, probability might easily be computed following algorithm 3. Interestingly, conditional probabilities of a given query, in regard with evidences, might be computed effortlessly if the BDD structure without evidence is known. For instance, in Figure 3.3, if the boolean variable X_2 was set true as evidence, during the algorithm traversal, the low child of node X_2 would be blocked and the probability of the node set to 1.

Current state of the art of ProbLog does not use BDD anymore [Fie+15; Dri+15] and relies on Deterministic Decomposable Negation Normal Form (d-DNNF)[Dar01; Dar04] or SDD[Dar11]. These structures provide specific advantages compared to BDD

[5] Nonetheless, not all variables in the formula appear necessarily in the BDD.

[6] An assignment may be seen as one path from the initial node to a terminal node in the structure.

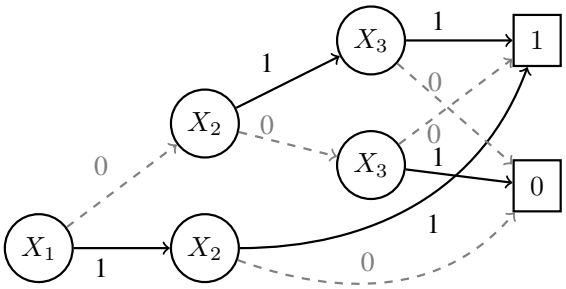


Figure 3.3: BDD example for formula $(\neg X_1, \neg X_2, \neg X_3) \vee (X_1, X_2) \vee (X_2, X_3)$

Algorithm 3: Recursive algorithm for probability computation at a given node n in the BDD [RKT07]

function *Probability(node n)*

inputs : n, a node in the BDD
outputs: The probability of the formula represented by the tree rooted in n

```
if n is the 1-terminal
    return 1
if n is the 0-terminal
    return 0
let h and l be the high and low children of n
prob(h) = Probability(h)
prob(l) = Probability(l)
let p_n the marginal probability of n ∈ T
return p_n · prob(h) + (1 − p_n) · prob(l)
```

like better guarantees on the size structure which improve overall time construction. We do not go into further details on these structures and their construction algorithm since it is not necessary in the context of this thesis and probability computation remains quite similar between BDD, d-DNNF, or SDD.

3.3 Non-homogeneous Markov models

The last necessary tool to introduce for this thesis is NHM. NHMs follow the Markov property, meaning that a modelled system changes conditionally regarding its last state, but they distinguish themselves from standard Markov chains that state change probabilities are not uniform during an execution. Consequently, NHMs might express more complex dependencies, in particular over time, but might be harder to describe. For this reason, this section focuses on a method introduced by Pachet, Roy, and Barbieri [PRB11] that defines a NHM from a standard Markov chain and a set of unary and binary constraints.

3.3.1 Context

Initially, this approach is used to generate samples of music for a particular style of music. The style is supposed to be learned as a simple Markov model M that stores the probability that a note is played after another. But, the authors desire to generate samples under a set of constraints representing specific musical properties or structures. Formally, the problem might be described as follows: given a Markov process M defines over a finite state space $A = a_1, \ldots, a_n$ and a sequence $s = s_1, \ldots, s_L$ with $s_i \in A$, S is the set of all sequences of length L generated by M with a non-zero probability:

$$P_M(s) = P_M(s_1) \cdot P_M(s_2|s_1) \cdot \cdots \cdot P_M(s_L|S_{L-1}) \tag{3.11}$$

Defining $V_1, \ldots, V_L$ with $V_i \in A$ a set of constraints variables representing states that might appears at a specific length i. States unexpressed in V_i cannot happen at length i. Regarding these constraints and the Markov process M, the set S produced by M is now constrained to a set S_C. Given these notations, the authors define a Markov process $\tilde{M}$ verifying the two following properties:

$$P_{\tilde{M}}(s) = \begin{cases} 0 \text{ if } s \notin S_C \\ P_M(s|S_C) \text{ otherwise} \end{cases} \tag{3.12}$$

These properties assert that $\tilde{M}$ generates exactly the sequence $s \in S_C$ and that sequences in S_C have the same probabilities in M and $\tilde{M}$ up to a constant factor $\alpha = P_M(s \in S_C)$ so $\forall s \in S_C,\ P_{\tilde{M}}(s) = 1/\alpha \cdot P_M(s)$. Authors show that such process might be described into a NHM represented by a set of L transitions matrices $\tilde{M}^{(i)}$.

3.3.2 Construction of the NHM

The construction of the NHM is dependent of a specific case of CSP named Binary-Sequential CSP (BSC). A BSC is a set of unary constraints $U_1, \ldots, U_L$ and binary constraints $B_1, \ldots, B_{L-1}$. Unary constraints V_i might be seen as states restrictions at given length i. Binary constraints B_j allow representation of conditional constraints by defining the possible transitions from states at time j to $j+1$. Given $a_k, a_m \in A$, if $B_j(a_k, a_m) = False$ it implies that the probability of transition in the NHM should be null $P_{\tilde{M}^{(j)}}(a_m|a_k) = 0$. Construction of the NHM $\tilde{M}$ from the Markov model is performed in two steps.

Constraints transposition First step generates L intermediate sequential matrices $Z^{(i)}$ from the initial Markov model M. M is represented as a simple matrix, but it should be defined with prior probabilities noted M_0. Construction process for the $Z^{(i)}$ is the following:

- $Z^{(0)} = M_0$
- $Z^{(i)} = M,\ \forall i = 1, \ldots, L-1$
- For each $a_k \in A$ removed regarding V_i: $Z^{(i)}_{j,k} = 0,\ \forall j = 1, \ldots, n$ (set the k-th column to zero)
- All forbidden transitions regarding the binary constraints B_i should be removed from the matrices: $Z^{(i)}_{j,k} = 0 \forall i, j, k$ such that $B_i(a_j, a_k) = False$

Normalisation The matrices $\tilde{M}^{(i)}$ from the NHM $\tilde{M}$ are then constructed from the $Z^{(i)}$ matrices by normalisations. Normalisation might be done on each matrix individually, but it would break the properties defined in Equation 3.12. Indeed, the initial probability distribution of sequences should be preserved so normalisation is

performed with backward algorithm:

$$\tilde{M}_{j,k}^{(L-1)} = \frac{Z_{j,k}^{(L-1)}}{\alpha_j^{(L-1)}}, \qquad \alpha_j^{(L-1)} = \sum_{k=1}^{n} Z_{j,k}^{(L-1)} \tag{3.13}$$

$$\tilde{M}_{j,k}^{(i)} = \frac{\alpha_k^{(i+1)} Z_{j,k}^{(i)}}{\alpha_j^{(i)}}, \qquad \alpha_j^{(i)} = \sum_{k=1}^{n} \alpha_k^{(i+1)} Z_{j,k}^{(i)} \qquad 0 < i < L-1 \tag{3.14}$$

$$\tilde{M}_{k}^{(0)} = \frac{\alpha_k^{(1)} Z_{j,k}^{(0)}}{\alpha^{(0)}}, \qquad \alpha_j^{(0)} = \sum_{k=1}^{n} \alpha_k^{(1)} Z_{k}^{(0)} \tag{3.15}$$

Note that $\alpha^{(0)}$ is the probability factor that a sequence s satisfies the constraints $\alpha^{(0)} = P_M(S_C)$.

CHAPTER 4

Chronicle representation with uncertainty using Markov logic networks

Contents

4.1 Chronicles expressed with logical rules 94
4.1.1 Chronicle semantic in Markov logic 95
4.1.2 Data stream structure and LLEs dependencies 96
4.1.3 Chronicle rules 97
4.1.4 Operator rules 98
4.1.5 Conclusion 101
4.2 Specific logical structures leading to intractable problems for Walk-SAT 102
4.2.1 A first example of unexpected behaviour with MC-SAT inference 102
4.2.2 WalkSAT efficiency analysis on specific problems 104
4.3 Consequences of WalkSAT inference on the chronicle model for MLNs 115
4.4 Conclusion 117

As shown in the state of the art in Chapter 2, MLN is one of the most popular approaches to represent uncertainty in CEP. Furthermore, this approach does not just exist in the CEP community but also appears regularly in the video analysis domain. Most of the existing works seem to produce interesting results and the expressiveness of Markov logic looks sufficient at first glance to justify this choice to represent a chronicle model with uncertainty. In this chapter, we present our representation of chronicles using MLNs and show that the logical formalism may lead to counter-intuitive representations. The first results that we obtained showed inconsistent recognitions even on small problems. We will show that these inconsistencies are caused by poor sampling during the inference produced by MC-SAT and in particular by the WalkSAT algorithm used to reach a solution. We show that performance of the algorithm is highly related to specific problem structures. All of our experimentations were conducted using *Alchemy 2.0*[1] to test our models and produce inference estimations.

In Section 4.1, we present our representation of chronicles based on Markov logic and detail a specific case that may seem quite counter-intuitive. Due to inconsistent results with our representation, we present in Section 4.2 our work to explain these inconsistencies produced by MC-SAT during the inference, providing examples and experimentations. Finally, in Section 4.1.2 we discuss the consequences of these results as to our objectives.

4.1 Chronicles expressed with logical rules

As a quick reminder, we want to represent a chronicle model in MLN where events on the data-stream might be erroneous or missing, and still be able to provide a probability for a given CE. We aim at representing a generic context-free model that may be reused independently of the domain. Furthermore, we want the model to be able to deal with hard rules and more specifically the possibility to define a CE as an observation for the inference. We detail in this section how we structured the rules of the model to fulfil these requirements.

[1] https://alchemy.cs.washington.edu/

4.1.1 Chronicle semantic in Markov logic

Markov logic, which we presented in Section 3.1, consists mainly of a set of FOL formulae associated to weights representing the confidence on each formula. Our representation of Chronicles in MLN consists consequently just in a set of logical formulae. These formulae are defined using a few predicates that we will soon detail. For reminder, in Markov logic semantic, by convention variables start with a lower case letter and constants with an upper case letter. Predicates start with an upper case letter like constants but are usually associated with a set of constants and variables between parentheses like $Predicate(var, Const)$ where var is a variable and $Const$ a constant associated to the predicate $Predicate$. To construct a formula, common operators of FOL are accepted: logical *and* ($\wedge$), logical *or* ($\vee$), implication ($\implies$), equivalence ($\iff$). Universal $\forall$ and existential $\exists$ quantifiers might be used on variables, and unquantified variables are by defaults assumed to be quantified universally. Even if we will mostly not use weight in this model, semantically speaking, a weighted formula is indicated with its weight in front of it. At the opposite, when a formula is hard[2], in Markov logic, it is indicated with a dot at the end of the formula. We kept this semantics in this thesis.

Let us present the main three predicates that we used and their meaning. They are listed in Table 4.1. Predicate $Ev(e, t)$ describes an LLE in the data stream and is defined with two parameters, which are its type and the time of detection in the data stream. In this representation, LLEs are supposed to happen at specific time points. Predicate $Ch(c, t_1, t_2)$ represents a recognition of a chronicle of type c starting at time t_1 and finishing at time t_2. Note that, contrary to the predicate Ev, the type of the chronicle c is defined inside the model, while the type e of the event is supposed to be a known type provided by the data stream, which is not supposed to be created in the model. Finally, we use two kinds of operator predicates. Binary operator predicates are of the form $OpX(c_1, c_2, t_1, t_2, t_3, t_4)$ and describe the meaning of one of the binary operators defined in Section 1.2.4, X just represents the corresponding operator used: e.g. the sequence operator is named $OpSeq$. Unary operator predicates, are of the form $OpX(c, t_1, t_2, \delta)$ with a different set of parameters, e.g. $OpAtMost$ represents the recognition of a chronicle c with its start and end times t_1 and t_2 associated to a duration parameter δ. The meaning of parameter δ obviously depends on the considered operator.

[2]Meaning it can not be violated by any world. Consequently the weight associated is supposed to be infinite.

Predicate	Meaning
$Ev(e,t)$	An LLE of type e observed at time t.
$Ch(c,t_1,t_2)$	A chronicle of type c recognised from time t_1 to t_2.
$OpX(c_1,c_2,t_1,t_2,t_3,t_4)$	A binary relational operator X between two chronicles of respective types c_1 and c_2, respectively recognised from t_1 to t_2 and from t_3 to t_4.
$OpX(c,t_1,t_2,\delta)$	A unary relational operator X defined by a specific duration δ on a chronicle of type c recognised from t_1 to t_2.

Table 4.1: List of predicates used for the MLN model

4.1.2 Data stream structure and LLEs dependencies

As we saw in Chapter 2, LLE uncertainty might be represented in different ways, but , for our model, we consider LLEs to be independent since we do not focus on specific relationships between LLEs for now and just want to perform probabilistic inference on designed CEs, even thought it would be quite easy to define at least first order Markovian dependencies with MLNs. For instance, a rule

$$w\ Ev(X,t_1) \wedge (t_2 = t_1 + 1) \implies Ev(Y,t_2) \tag{4.1}$$

defines a dependency between a X-type LLE and a Y-type LLE and means that the probability of the Y event depends on the previously observed LLE and weight w. But, since we choose a representation where LLEs are independent, they are represented by a prior probability, which is modelled by defining a rule with only one predicate that is an event type predicate. So, for instance, if the model had two LLEs of respective types A and B with different marginal probabilities, they should be described as follows:

$$\begin{aligned} w_1 \quad & Ev(A,t) \\ w_2 \quad & Ev(B,t) \end{aligned} \tag{4.2}$$

This representation is sufficient to describe a potential for LLEs, even if weights make it quite difficult to know what is the prior probability of these events. The formulae in Equation 4.2 may also be seen, in a different way, as a noisy detection in the data stream, since they may produce events from nowhere.

4.1.3 Chronicle rules

Chronicle predicates are used to define CEs in our model. By default, we suppose that a LLE produces a chronicle of the same type with recognition times both equal to the time of the detection of the event.

$$Ev(e,t) \implies Ch(e,t,t). \tag{4.3}$$

This formula is hard since there is no reason to ignore data provided by the data stream. Missing detection may be represented by the following rule

$$w \quad \neg Ev(e,t) \implies Ch(e,t,t) \tag{4.4}$$

which describes that a chronicle may be created even if no event is detected in the data stream. With these rules, we define a set of rules that constrain our model and, in particular, the parameters provided with the chronicle predicate:

$$Ch(c,t_1,t_2) \implies t_1 \leq t_2. \tag{4.5}$$

$$Ch(e,t_1,t_2) \implies t_1 = t_2. \quad | \quad \text{for } e \text{ a LLE type} \tag{4.6}$$

Equation 4.5 sets a constraint between t_1 and t_2, asserting that the beginning time of a chronicle recognition is always equal to or smaller than the ending time. Equation 4.6 is a specific constraint on time recognition for our model that ensures that chronicles produced by a LLE should have the same beginning and ending times. These rules are necessary since they assert that grounded predicates that do not satisfy these rules have a zero probability: without them, inconsistent grounded predicate would all have a default 0.5 probability.

Before presenting how operators might be defined in this approach, we want to address how a chronicle is constructed using the operators. Lets assume that operators have been defined and follow the semantic presented in Table 4.1, chronicles may be defined almost straightforwardly using these operators as constraints. Suppose, we are parsing a data stream with four type of LLEs: A, B, C, and D and we want to design a chronicle recognising a sequence of events A and B. Event chronicles are already defined with Equation 4.3 so we only need the definition of a chronicle $Ch(AB,t_1,t_2)$. Remember that the constant AB is just a name for the chronicle type and has no influence on the definition, which is provided with the following rule:

$$OpSeq(A,B,t_1,t_3,t_4,t_2) \implies Ch(AB,t_1,t_2). \tag{4.7}$$

Nevertheless, the rule is not sufficient to achieve our initial objectives since chronicles might be used as constraints for the model. Consequently, it is necessary to represent circumscription, which means that a chronicle recognition should induce the recognition of sub-events. For this purpose, a dual formula is necessary

$$\exists t_3, t_4 \; Ch(AB, t_1, t_2) \implies OpSeq(A, B, t_1, t_3, t_4, t_2). \tag{4.8}$$

The existential quantifier in Formula 4.8 is quite simple to understand, since a recognition of a chronicle AB between t_1 and t_2 does not imply that all ground predicates $OpSeq$ under these times should be satisfied. However, this formula cannot be considered as a simple formula as, with MLN, it should be transformed into a formula where all the ground predicates linked to the variables under the existential quantifiers should be added to the formula with a $\vee$ operator. For instance, if $t_3, t_4 \in \{0, 1\}$, formula 4.8 should be rewritten:

$$\begin{aligned} Ch(AB, t_1, t_2) \implies \; & OpSeq(A, B, t_1, 0, 0, t_2) \vee OpSeq(A, B, t_1, 0, 1, t_2) \\ & \vee OpSeq(A, B, t_1, 1, 0, t_2) \vee OpSeq(A, B, t_1, 1, 1, t_2). \end{aligned} \tag{4.9}$$

It immediately appears that such formulae will grow quadratically with the number of values are which t_2 and t_3 range, but they seem difficult to avoid in order to represent hard constraints with chronicle and logical circumscription.

4.1.4 Operator rules

We now address the definition of operators. Operator predicates have to encode the temporal constraints presented in Section 1.2.4. They may almost be described in the same way with a few exceptions, so we will detail one definition and provide an oversight of the other, with additional information for a few particular operators. Definition for all operators are detailled in Appendix A.

The sequence operator $OpSeq$ may be described with the following formula:

$$Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge t_2 < t_3 \iff OpSeq(c_1, c_2, t_1, t_2, t_3, t_4). \tag{4.10}$$

It is important to remember that one of our goals is to be able to set a chronicle as an evidence for the inference. To assert it, equivalences are necessary. Remember that an MLN is expressed with a set of Horn's clauses (cf. Section 3.1), thus Equation 4.10 will

be represented as a set of four clauses :

$$\begin{aligned} Ch(c_1,t_1,t_2) \wedge Ch(c_2,t_3,t_4) \wedge t_2 < t_3 &\implies OpSeq(c_1,c_2,t_1,t_2,t_3,t_4). \\ OpSeq(c_1,c_2,t_1,t_2,t_3,t_4) &\implies Ch(c_1,t_1,t_2). \\ OpSeq(c_1,c_2,t_1,t_2,t_3,t_4) &\implies Ch(c_2,t_3,t_4). \\ OpSeq(c_1,c_2,t_1,t_2,t_3,t_4) &\implies t_2 < t_3. \end{aligned} \tag{4.11}$$

It is important to remember how MLNs are constructed since it has an impact on the size of the model, on the inference, on the weight definition, and, obviously, on computation time. The definition of an unary operator, like *at most*, shows almost no difference since they may be described using a similar structure. For instance, the operator $OpAtMost$ is represented by the formula 4.12.

$$Ch(c,t_1,t_2) \wedge (t_1 + \delta < t_2) \iff OpAtMost(c,t_1,t_2,\delta). \tag{4.12}$$

We will just detail two particular operators that have a slightly different construction regarding other operators. The *absence* chronicle is an operator that relies on another operator to be defined. It might be specified in other ways but the easiest approach consists in using the *during* operator.

$$OpAbs(c_1,c_2,t_1,t_2,t_3,t_4) \iff Ch(c_1,t_1,t_4) \wedge \neg OpDur(c_1,c_2,t_1,t_2,t_3,t_4). \tag{4.13}$$

With this definition operator $OpAbs$ might be counter-intuitive since it only applies to variables t_1,t_2,t_3,t_4 so it does not mean that no chronicle of type c_2 might be found between t_1 and t_4, but that no chronicle of type c_2 appears at the times t_2,t_3. Expressing that no chronicle of type c_2 should appear between t_1 and t_2 is done when defining a specific absence chronicle. For instance, assuming a chronicle AB, that represents a sequence A to B, has already been defined, then the definition of chronicle $ABnC$, i.e. a sequence AB without any event C between A and B, should be structured as follows:

$$\begin{aligned} \exists t_3,t_4\ OpAbs(AB,C,t_1,t_3,t_4,t_2) &\implies Ch(ABnC,t_1,t_2). \\ Ch(ABnC,t_1,t_2) &\implies OpAbs(AB,C,t_1,t_3,t_4,t_2). \end{aligned} \tag{4.14}$$

Formula 4.14 defines the new type of chronicle $ABnC$ based on an absence operator. The definition is almost equivalent to formulae 4.7 and 4.8 defined for the sequence operator but existential quantifiers are set differently. Note that for the absence, existential quantifier is not used on the same formula when defining a chronicle than for other operators. The chronicle predicate might be entailed only if every ground

predicate $OpAbs$ are satisfied.

Another interesting operator to look at is the *and* operator. At first glance, defining an *and* operator should be easy as it is possible to use logic, but it is without considering that beginning and ending of a chronicle should be clearly indicated. Indeed, depending on the order of sub-chronicles, time boundaries from the chronicle based on the *and* operator might be provided by the two sub-chronicles. For instance, supposing a chronicle $AandB$, that is recognised when A and B are recognised, the beginning time of the chronicle $AandB$ might be the starting time of A or B as well, depending on their own position in the data stream. Similarly, to the *absence* operator, it is necessary to define intermediate operators to capture all possible orderings of sub-chronicles. If we consider our chronicle example $AandB$ and a recognition r_{AandB} of this chronicle, the starting time of this recognition, noted $\min(r_{AandB})$, depends on two situations and defined as follows:

$$\min(r_{AandB}) = \min(\min(r_A), \min(r_B)) \tag{4.15}$$

This observation is equally true (*mutatis mutandis*) for the ending point of the recognition $\max(r_{AandB})$:

$$\max(r_{AandB}) = \max(\max(r_A), \max(r_B)) \tag{4.16}$$

These possible situations need to be represented by four intermediate operators that are defined following formulae 4.17.

$$\begin{aligned}
OpConj_1(c_1, c_2, \underline{t_1}, t_3, t_4, \underline{t_2}) &\iff Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge (t_1 \leq t_3) \wedge (t_4 \leq t_2) \\
OpConj_2(c_1, c_2, \underline{t_3}, t_1, t_4, \underline{t_2}) &\iff Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge (t_3 < t_1) \wedge (t_4 \leq t_2) \\
OpConj_3(c_1, c_2, \underline{t_1}, t_3, t_2, \underline{t_4}) &\iff Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge (t_1 \leq t_3) \wedge (t_2 < t_4) \\
OpConj_4(c_1, c_2, \underline{t_3}, t_1, t_2, \underline{t_4}) &\iff Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge (t_3 < t_1) \wedge (t_2 < t_4)
\end{aligned} \tag{4.17}$$

The names of these operators are completely arbitrary and the value i from $OpConj_i$ is only used to distinguish between them. Note that even if these rules are simple, the order of the variable into an operator $OpConj_i$ is chosen so as to assert that the first time and last time (underlined in formula 4.17) are respectively the minimal and maximal times of recognitions for the operator. This construction makes the final $OpConj$ predicate easier to define since it asserts that the recognition time boundaries

are always at the same positions on every intermediate predicates.

$$\begin{aligned} OpConj_1(c_1, c_2, t_1, t_3, t_4, t_2) \vee \cdots \vee \quad & OpConj_4(c_1, c_2, t_1, t_3, t_4, t_2) \\ \implies \quad & OpConj(c_1, c_2, t_1, t_2) \end{aligned} \tag{4.18}$$

$$\begin{aligned} \exists t_3, t_4 \quad & OpConj(c_1, c_2, t_1, t_2) \implies \\ & OpConj_1(c_1, c_2, t_1, t_3, t_4, t_2) \vee \cdots \vee \ OpConj_4(c_1, c_2, t_1, t_3, t_4, t_2) \end{aligned} \tag{4.19}$$

Note that predicate $OpConj$ is defined directly with only two time variables. With this construction, a chronicle based on an *and* operator might be defined straightforwardly:

$$OpConj(c_1, c_2, t_1, t_2) \iff Ch(c_1_and_c_2, t_1, t_2). \tag{4.20}$$

Inconsistent inference with the model In parallel with the definition of the chronicle model, each of the operators was tested on small data streams containing four or five LLEs to see whether rules followed our expectations. Furthermore, we tried to remove some events from the data stream and provide chronicle as observation on our model to see whether it was possible to reconstruct the data stream in a top-down approach. From these small test cases, it appeared that inference with MC-SAT produces small estimation errors on fully logical problems and even sometimes produced inconsistencies or strange probabilities[3]. These inconsistencies do not require a complex program to appears and only defining a chronicle with one or two operators applied on few events is sufficient. From this point, we preferred investigate on the reasons of these inconsistencies before trying to apply our model to a full study case.

4.1.5 Conclusion

In this first section, we defined our model and specific details to design our problem with the Markov logic semantic. We presented in this section how chronicles and operators are constructed and defined. With this construction, it would normally be possible to represent uncertainty on a data stream and produce probabilities estimations with soft constraints[4] and high-level constraints. It may be not obvious, but all rules presented in this section entail both recognitions and non-existing recognitions, which is necessary if we aim at representing circumscription. In this section, we provide first insights on the relations between a formula and its representation in MLNs

[3]For instance, two LLEs that should have the same probability of apparition on the data have after inference, asymmetrical probabilities.

[4]Formally, the LLEs in the data stream.

and on the weight mechanism, but, in the next sections, we discuss these aspects in deeper detail and more specifically the consequences that they may have on inference, in order to explain the observed inconsistencies.

4.2 Specific logical structures leading to intractable problems for WalkSAT

We presented, in the first section, the model that we designed to represent chronicles. We expressed at the end of the section that inference with MLNs produced inconsistent evaluations regarding our model and the problem designed to evaluate it. After further investigation, we concluded that these inconsistencies were not due to a design fault, but to structural problems leading SampleSAT, the sampler used by MC-SAT, to perform bad sampling leading to erroneous estimates. In particular, we focus on the original algorithm WalkSAT, core of SampleSAT, and its ability to solve specific logical problems. Consequently, in this section, we detail further the experimentations that we performed to investigate the matter. These experimentations do not use the chronicle model expressed in Section 4.1, but focus on specific structures closely related to it.

4.2.1 A first example of unexpected behaviour with MC-SAT inference

As we explained in the previous section, inconsistencies appeared occasionally during the design process of the chronicle model in MLN. They did not appear consistently and might produced small variations (around 10 percent variations) on probability estimations. At first glance, these variations might be interpreted as possible errors from sampling, but these errors might appear even on problems structured with no uncertainty[5]. On such logical problems, if the inference produces a probability with a value different from 1 or 0 (or extremely close), it means the sampler reached worlds that does not satisfy the problem and return it as a valid sample. This is not unexpected since WalkSAT and its weighted variations do not guarantee to find a solution due to their local stochastic search approach. But performances of MLNs mostly rely on WalkSAT performances so it would be interesting to assess the ability of WalkSAT to reach a solution. To our knowledge, this observation is rarely seen across the literature, which usually tends to focus on computation time.

Before introducing our approach to study WalkSAT reliability, we provide a really small example of problem that produces inconsistencies to show that the issue is not

[5]All rules are hard and the data stream is completely provided.

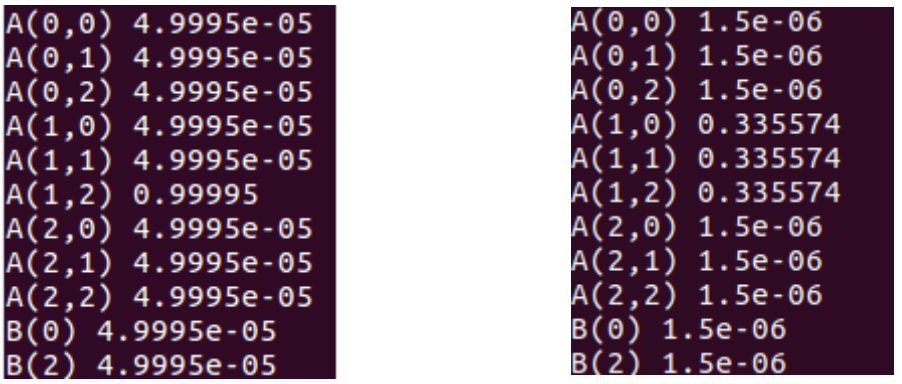
necessarily size related. Consider the following problem designed in Markov logic:

$$\begin{aligned} &9\ \neg A(t_1, t_2) \\ &9\ \neg B(t_1) \\ &A(t_1, t_2) \implies B(t_1). \\ &\exists t_2\ B(t_1) \implies A(t_1, t_2). \end{aligned} \tag{4.21}$$

where $t_1, t_2 \in \{0, 1, 2\}$. Consequently, there are nine ground predicates A and three B. The two first formulae are weighted and describe that, without any contradictory deduction, predicates A and B should not be satisfied. Since the two last rules are hard constrained, they should always been prioritised against the weighted rules. With no evidence provided, the optimal solution consists in considering every ground predicate as false since each hard constrained rules remains satisfied. In this case, probability estimation is, as intended, 0 for every predicate. Now, consider the same problem, but with the evidence that $B(1)$ is satisfied: the third rule remains satisfied for any ground predicates $A(t_1, t_2)$, but the fourth rule enforces that $A(1,0) \vee A(1,1) \vee A(1,2)$ must be satisfied[6]. This last formula might be asserted in seven configurations of truth values and violated in one configuration (when each predicate is false). Due to the first rule of Equation 4.21, assignments of value for ground predicates A should be equiprobable. Figures 4.1a and 4.1b show the results of inference with Alchemy on this problem using MC-SAT inference and belief propagation. The result of MC-SAT is unexpected since it shows that $A(1,0)$ and $A(1,1)$ are never true in any possible solution for this problem, which is obviously wrong: belief propagation, on the other side, provides consistent results where all ground predicates $A(1, t_2)$ are equiprobable. It is worth noting that results from MC-SAT can be changed in Alchemy if a different seed is provided. For instance, seed 444 gives a 0 probability to every ground predicate except $A(1,0)$.

Side notes on weight influence It may be surprising to see that belief propagation estimates ground predicates $A(1, t_2)$ with a $0.33\ldots$ probability while it should have 3 configurations on 7 satisfying each of these ground predicates, which suggests it should be approximately a 0.43 probability. Interestingly, probabilities for these predicates are highly correlated to the weight ratio between the two first and the two

[6]Remember that, with MLNs, existential quantifiers are replaced with the disjunction of all ground predicates impacted by the quantifiers. Consequently, fourth rule of Equation 4.21 might be rewritten $B(t_1) \implies A(t_1,0) \vee A(t_1,1) \vee A(t_1,2)$. As $B(1)$ is set as evidence, for $t_1 = 1$, the fourth rule might be simplified in: $A(1,0) \vee A(1,1) \vee A(1,2)$.



(a) MC-SAT inference

(b) Belief propagation inference

Figure 4.1: Inference result with Alchemy on the problem described in Equation 4.21 with $B(1)$ set as evidence. Initialised with seed 445. Left side MC-SAT inference, right side Belief propagation.

last rules in Equation 4.21. Theoretically speaking, the ratio is close to zero since weight of hard constraints is supposed to be infinite. In Alchemy, hard constraints are just weighted formulae with a weight ten times bigger than the highest weight. But, since the weight for predicates A is already quite big (usually weights are set between $[-1, 1]$), having two predicates $A(1, t_2)$ set at true is less likely since it would break more ground formulae. In this situation, the first rule, and more precisely its weight, makes ground predicates mutually exclusive with a high enough weight. With a smaller weight, probability estimates may change from 1 to 0.33. These variations are plotted in Figure 4.2. This represents a good example of weight influence on logical representations and its misleading effects since, in this case, weight have a direct influence on a specificity, like mutual exclusivity, that would be intuitively represented by a rule and not by the choice of a weight. Furthermore, it has been shown that the choice of the weight is dependent of the size of ground network.

4.2.2 WalkSAT efficiency analysis on specific problems

In the state-of-the-art, inconsistencies observed during our experimentations have, apparently, never been reported. This induces that the chronicle representation presents particular properties sufficient to produce these errors on inference, and that these properties are not present in the other approaches presented in Section 2.2. It is simple to extract few differences between our approaches and existing ones:

- Most of the approaches do not represent logical circumscription, meaning they do not have to use equivalences between a CE and its sub-events (except [Ska+11]).

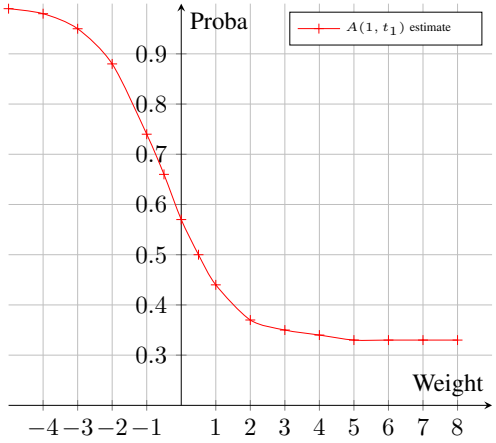


Figure 4.2: Variations of probability estimates of ground predicates $A(1, t_1)$ regarding the weigh for the first rule in Equation 4.21. For practical reasons, hard constrained rules are defined as weighted rules with weight 100.

Consequently, most problems might be usually represented as an acyclic oriented graph (oriented respectively with the implications between events). Remember that circumscription is necessary to be able to provide chronicle recognitions as evidences for the inference.

- Another main difference from the chronicle representation and previous methods is that the latter are usually domain-specific. In [Ska+11] for instance, Event calculus semantic is voluntarily reduced and adapted to reduce the overall size of the network.

- Finally, even if it is not possible to present it as a difference, all the approaches from the state-of-the-art seemed to be relatively small regarding the size of predicates, clauses, and the number of levels of deduction. By levels of deduction, we refer to the hierarchical semantic of rules. For instance, a chronicle may be defined using sub-chronicles, themselves defined by other sub-chronicles, etc. Usually, in literature, this number of hierarchical levels are rarely above four, but for being able to define a realistic chronicle using the syntax defined previously, it is almost certain that more than four levels would be necessary.

As presented in Section 3.1, the MC-SAT algorithm strongly relies on MaxWalkSAT. With MC-SAT, poor estimations result necessarily from a bad sampling which is affected by two parameters (cf. Algorithm 2): the selection of clauses at each iteration or the production of a sample $x(i)$. Since experimentations were performed on

problems with just hard constrained formulae (excepted for unary clauses representing the prior probability of LLEs without prior knowledge), all clauses are selected for sampling, meaning that the reasons of inconsistencies are necessarily due to a poor sampling produced by SampleSAT.

As previously detailed in Chapter 3, SampleSAT randomly chooses between a walk step and a simulated annealing step. Note that for a problem designed mostly with hard rules, the simulated annealing step does not have a great impact on computation. Indeed, if the cost of a random neighbour assignment is better the current one, the algorithm change to this assignment. This behaviour is mostly like WalkSAT when it selects a clause that might have a better assignment. Otherwise, if there is no better assignment, simulated annealing would most likely never change the current assignment, since such modification may happen with a probability $1 - e^{\Delta cost/Temp}$, with $\Delta cost$ the cost between the two assignment relative to the weights provided by the MLN and $Temp$ a constant[7]. Since all grounded predicates are connected to hard constrained rules, such a change will most likely have a probability extremely close to 0. For this reason, sample production mostly relies on MaxWalkSAT.

Linear chains To test MaxWalkSAT, we designed a set of experiments to evaluate if one of the difference from the literature may be the reason of the results obtained with chronicles. The first specificity from our approach might be the circumscription representation. Existential quantifiers aside, circumscription might be represented with equivalence, so the first experimentation is extremely simple and represent a chain of deduction. Given N predicates x_i, $i \in 1, \ldots, N$ the first scenario is designed as follows:

$$x_i \iff x_{i+1} \quad | \, i \in 1, \ldots, N-1 \tag{4.22}$$

and is compared with a second scenario designed only with implication:

$$x_1 \wedge (x_i \implies x_{i+1}) \quad | \, i \in 1, \ldots, N-1 \tag{4.23}$$

In scenario described by Equation 4.23, predicate x_1 is given as clauses to reduce the number of solutions. Indeed, Equation 4.22 has two solutions were all the predicates are assigned to the same value (true or false), while 4.23 has N solutions if the clause x_i is not provided and only one with it. It is necessary to reduce the number of solutions

[7]Among the literature on MLNs, the $Temp$ parameter is never addressed. We suppose that in most cases it is ignored or set to one.

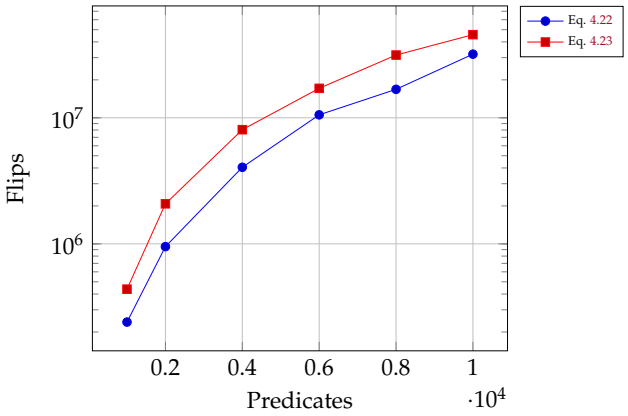


Figure 4.3: Flips necessary to find a satisfiable assignment for Equations 4.22 and 4.23 regarding the number of predicates.

since it is easier to reach a satisfiable assignment with a problem that has more solutions. Figure 4.3 shows the average number of flips necessary to reach a solution on both problems. In this case simple or equivalences seem to have no particular impact on the computation, but note that this case is considered as a difficult 2-SAT problem for WalkSAT [WS02]. Interestingly, this type of problem is considered as hard due to the implications chaining structure.

Ternary chains 2-SAT instances are not usually studied[8] in WalkSAT, but previous works ([Pre05; WS02]) refer regularly to the 3-SAT version of chained implications which might be simply described as follows:

$$x_1 \wedge x_2 \wedge (x_i \wedge x_{i+1} \implies x_{i+2}) \mid i \in 1, \ldots, N-2 \tag{4.24}$$

Results for this problem are displayed in Figure 4.4. Some preliminary context is necessary to make sense of these results. In the literature, influences of model parameters on inference for solver (local search or not) are often analysed on random instances of k-SAT problems. For instance, the most common one is the ratio α that measures the proportion of clauses against the number of predicates. It has been shown in [HS05] that this ratio may have a great influence on computation. This might not really apply here since this influence on this ratio has been shown on random problems. Indeed for a random problem, it has been shown that above 4.5, the probability that this problem is unsatisfiable is almost one [HS05]. For Equation 4.24, it is guaranteed that a solution

[8] Algorithms like DPLL are linear for 2-SAT instances, so local search algorithm are tend to be less studied for this kind of problem.

N	Eq. 4.24	Eq. 4.24 with $\Longleftrightarrow$
10	121	168
20	17010	9993
30	$2.50 \cdot 10^6$	$6.1 \cdot 10^5$
35	$2.28 \cdot 10^7$	$3.95 \cdot 10^6$
40	$3.05 \cdot 10^8$	$2.91 \cdot 10^7$

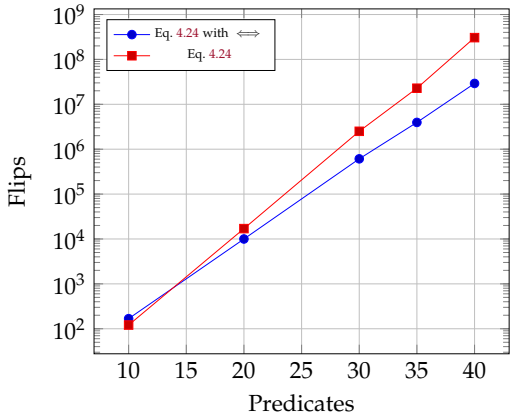


Figure 4.4: Average number of flips used to find a assignment for Equation 4.24 with simple and double implications regarding the number of predicates.

Problem Instance	WalkSAT/SKC	WalkSAT/TABU
uf200/hard	$0.85 \cdot 10^6$	$3.70 \cdot 10^6$
flat100/hard	192788	229496
par8-5-c	13345	8388
logistics.d	398277	332494
bw large.a	13505	7563
bw large.c	$9.76 \cdot 10^6$	$2.00 \cdot 10^6$

Table 4.2: Common benchmarks of 3-SAT problems tested for WalkSAT inference. Source: [HS05], Chapter 6.

exists, so the α ratio is not important here. But, similarly, the choice of the probability p that determines the choice between a random step or a local search step may modify greatly the performances and it is generally set between 0.45 and 0.6. As reference, [SAO05] applied WalkSAT on a random 3-SAT instances using different p values on a random 3-SAT instance with $N = 10^5$. The authors showed that, with a probability around 0.5, WalkSAT found a assignment in $10 \cdot N$ with α below to 4. Note that with the ternary chain from Equation 4.24, results with WalkSAT are worse since it finds an assignment in $10^7 \cdot N$ for $N = 40$. For a better perspective of WalkSAT performance, Table 4.2 presents classical non-random benchmarks used traditionally by the SAT community with the average number of flips required to find a satisfiable assignment. These benchmarks range from 75 to 3 016 predicates. These results come from [HS05] (Chapter 6) and display inference for two WalkSAT versions. The SKC version is the same as Algorithm 1 in Section 3.1 while the TABU version is almost equivalent but

prevents that successive flips happen on the same predicates. Independently of these two versions, the ternary chains take far more iterations than 3-SAT random and classic hard 3-SAT problems of same dimensions. Interestingly, some other type of 3-SAT instances have been reported to require an unusually high amount of flip to be solved as the 3-XORSAT instances [GY11].

Simplified chronicles This ternary problem is an interesting cue to explain why the chronicle representation seems to produce inconsistent results, but the ternary chains and the chronicle representation are not totally similar, so we propose a simplistic representation that is closer to the chronicle's cases. Given a set of boolean variables $V = \{x_1^1, \ldots, x_{i-j+1}^j, \ldots, x_1^L\}$ for $j \in \{1, \ldots, L\}$ and $i \in \{1, \ldots, L-j+1\}$, with L the total number of layers, and given the set of clauses $\mathcal{R} = \{c_1, \ldots, c_n\}$, the problem is designed as follows: $\mathcal{F} = \bigwedge_{c_i \in \mathcal{R}} c_i$ and

$$\begin{cases} (x_i^j \wedge x_{i+1}^j \implies x_i^{j+1}) \in \mathcal{R} \\ (x_i^{j+1} \implies x_i^j) \in \mathcal{R} \\ (x_i^{j+1} \implies x_{i+1}^j) \in \mathcal{R} \\ x_i^1 \in \mathcal{R} \end{cases} \quad \text{with} \quad \begin{cases} 0 < j \leq L; \\ 0 < i \leq L - j + 1 \end{cases} \tag{4.25}$$

where j is the height of a predicates layer, i the position of a predicate on j^{th} layer. An example is provided in Figure 4.5 with a height of 3 and a base length of 5. We sometimes use the terms *higher* predicate (respectively *lower*) and *highest* predicate (resp. *lowest*) to describe the position of a predicate regarding the layers as they are depicted in Figure 4.5. Consequently, the *highest* (resp. *lowest*) predicates are on the layer L (resp. layer 1). Equivalently, we distinguish clauses regarding their direction from a layer to another using the term *ascending* clauses (resp. *descending*) for clauses describing a implication from a lower layer to a higher layer (resp. from a higher layer to a lower layer). Note that this model might be represented without circumscription if all the *descending* clauses are removed. Formula 4.25 does not completely capture the chronicle representation but it captures the succession of deductions used to represent a CE. The simplest way to see the proximity between the two models is to consider the first layer of predicates x_i^1 as a succession of LLEs in the data-stream and the upper layer as CEs. In reality, the number of predicates in the upper layers should be higher since it should have a predicate for each combination of events on the same layer . In this case, predicates are associated to two consecutive predicates from the

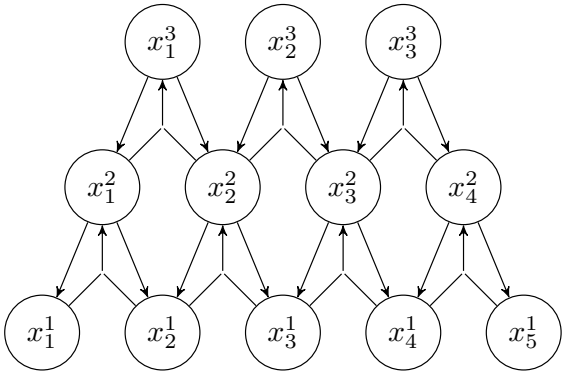
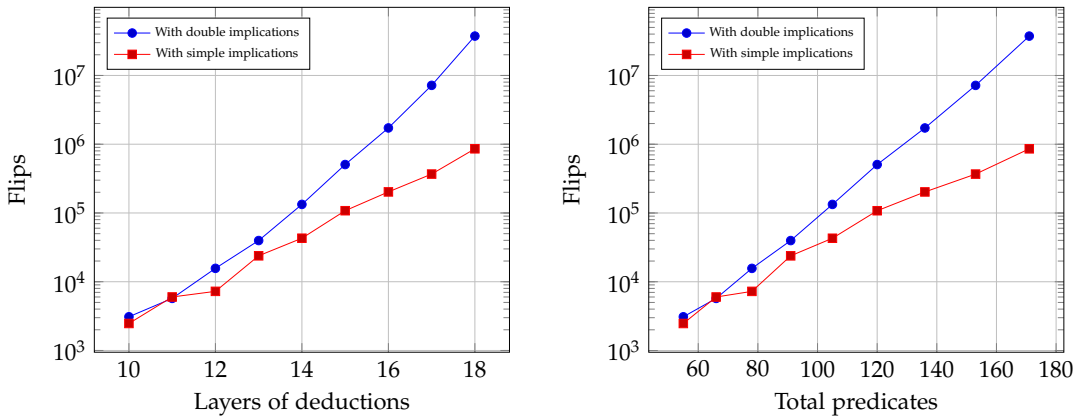


Figure 4.5: A simplified chronicle representation with a height of 3 and a base length of 5. Each arrow on the graph is clause representing an implication between the corresponding predicates.



(a) Total flips regarding the structure height.

(b) Total flips regarding the predicates total.

Figure 4.6: Flips necessary to find a satisfiable assignment for Equation 4.25 where height of the structure is equal to its base length.

layer below, which makes the structure easy to represent graphically. Note that this structure presents an equivalent chaining of implication as in Equation 4.24.

In order to show that the deduction chain length is an important factor on computability, we set an experiment based on Equation 4.25 where the height of the structure is maximal for a particular base length. Consequently, the height of the structure is always equals to its base length. This might be seen as a problem to recognise one instance of a chronicle composed of L LLEs. Results (cf. Figure 4.6) show that this kind of representation presents a behaviour similar to the ternary chains and becomes quickly intractable with a small number of predicates. It appears from Figure 4.6a that the height of the structure, and consequently the complexity of chronicle in terms of

number of operators used, have an important impact on calculability. This partially explains why models from the literature (cf Chapter 2) did not face erroneous results, since they are mostly limited in terms of deduction levels and do not usually have more than three or four. Note that inference has been tested on the same problem without circumscription. Concretely, this representation is equivalent to Equation 4.25 with some slight changes that suppress the *descending* clauses:

$$\begin{cases} (x_i^j \wedge x_{i+1}^j \implies x_i^{j+1}) \in \mathcal{R} \\ x_i^1 \in \mathcal{R} \end{cases} \quad \text{with} \quad \begin{cases} 0 < j \leq L; \\ 0 < i \leq L - j + 1 \end{cases} \tag{4.26}$$

It appears from Figure 4.5 that computations are faster without representing circumscription. This may be another explanation of the difference in results with the state-of-the-art since many approaches do not represent circumscription, but we will show that it should be carefully considered regarding the dimension of the problem and more particularly the number of predicates per clauses.

Impact of k in k-SAT instance Usually in k-SAT random problem, the higher is k, the longer the computation takes for an equivalent number of predicates [CLS15]. The chronicle representation and its simplified version, nonetheless, are not technically pure k-SAT instances since many clauses are just composed of two predicates. But it may be interesting to measure the impact of long clauses (generally created by existential quantifiers) on the computation.

Equation 4.27 is the generalised version of Equation 4.25 adapted for any k-clauses. The general structure is basically the same, except that the *ascending* clauses are composed of k predicates.

$$\begin{cases} (x_i^{j-1} \wedge \cdots \wedge x_{i+k-2}^{j-1} \implies x_i^j) \in \mathcal{R} \\ (x_i^j \implies x_i^{j-1}) \in \mathcal{R} \\ \vdots \\ (x_i^j \implies x_{i+k-2}^{j-1}) \in \mathcal{R} \end{cases} \quad \text{with} \quad \begin{cases} 1 < j \leq L; \\ 0 < i \leq (k-2) \cdot (L-j) + 1 \end{cases}$$
$$\begin{cases} x_i^1 \in \mathcal{R} \end{cases} \quad \text{with} \quad \begin{cases} 0 < i \leq (k-2) \cdot (L-1) + 1 \end{cases} \tag{4.27}$$

For instance, a 4-SAT version of this structure with three layers is presented in Figure 4.7. Performance of WalkSAT on different k-SAT instances of this model are dis-

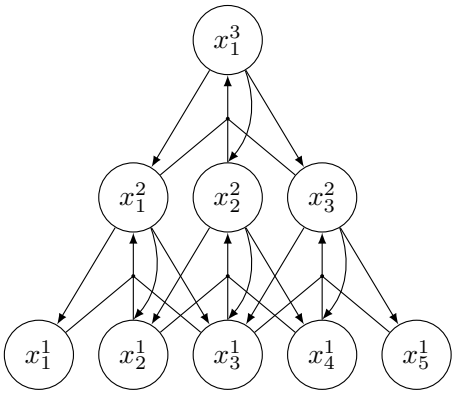
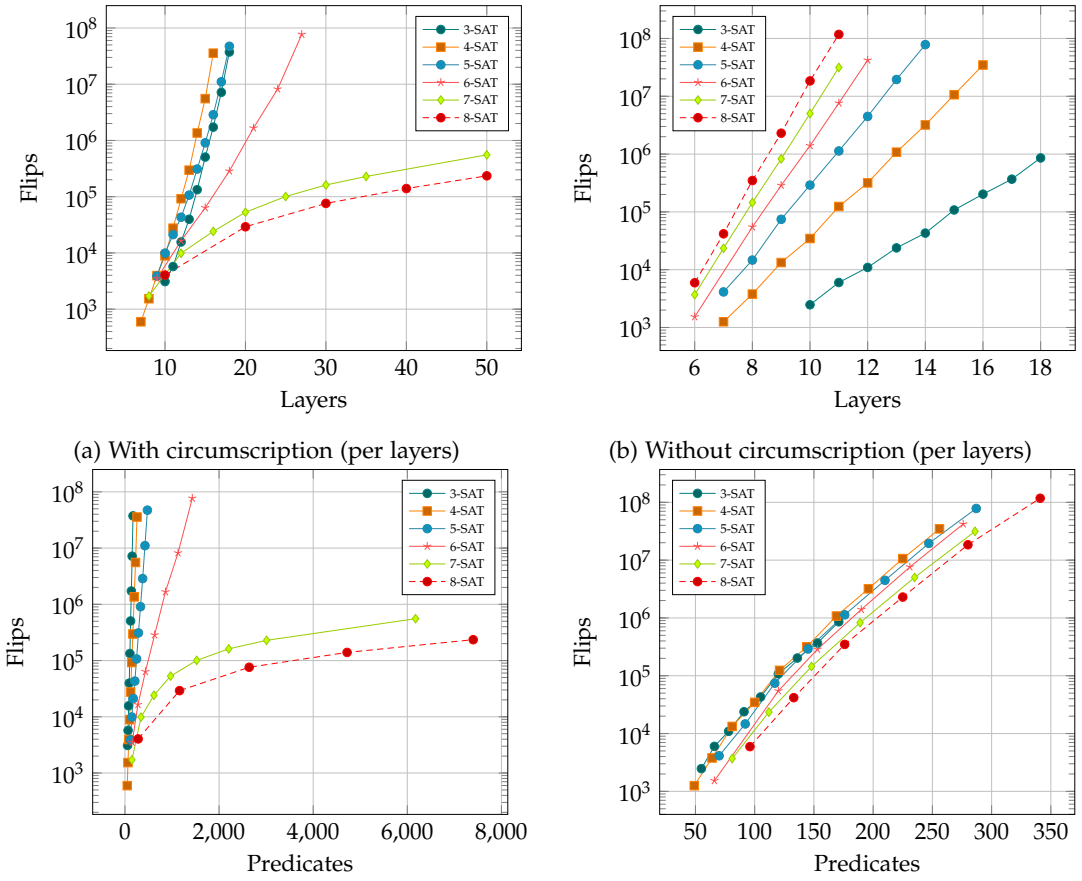


Figure 4.7: Graphical representation of Equation 4.27 with $L = 3$ and $k = 4$.

played in Figure 4.8. Experimentations have been done with and without circumscription representation. Surprisingly, it appears that when circumscription is represented, the number of iterations to solve a problem drastically decreases (at constant number of predicates) when k increases, which seem to imply that with k high enough the clauses composed of two predicates help with the computation. But, for the 4-SAT versions, both have the same performances, and for the 3-SAT versions the representation without circumscription is faster than the one with it. This is an important information since the operators, as they are defined in the chronicle representation with MLNs, are mostly composed of three predicates and time constraints. Since time constraints might be evaluated when the network is grounded, they do not count as predicates, meaning that all the binary operators are clauses with three predicates, which is, as reported in Figure 4.8a, one of the worst k-SAT instantiations for computation efficiency.

Complexity of evidences In this section, all the experimentations focused on the propagations of evidences through a logical problem while using WalkSAT, but in everyone of them the evidences were considered to be provided through "*lowest*" predicates. But, remember that, for the chronicle representation, it should be possible to provide evidences as CEs, which means the *highest* predicates (aka the more complex in term of dependencies) in the previous models. Note that, regarding Equation 4.27, if clauses $x_i^1 \in \mathcal{R}$ are replaced by the clauses x_i^L the solution of the model is the same with all the predicates to be evaluated as true. Figure 4.9 displays the amount of flips necessary in average to reach a solution when the evidence is provided with the *highest* predicates. These results should be compared to Figures 4.8a and 4.8c. It appears



(a) With circumscription (per layers)

(b) Without circumscription (per layers)

(c) With circumscription (per predicates)

(d) Without circumscription (per predicates)

Figure 4.8: Flips necessary to reach a solution regarding the value k in Equation 4.27.

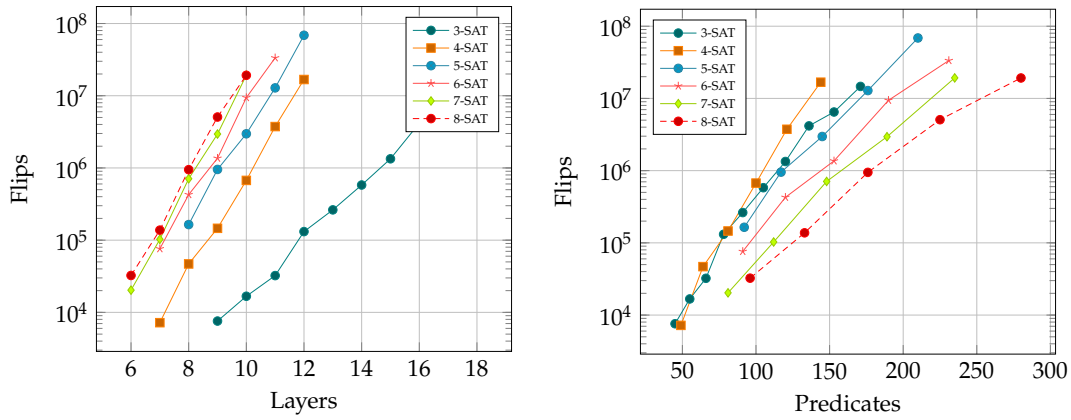


Figure 4.9: Flips necessary to reach a solution regarding the value k in Equation 4.27 when evidences is provided with x_1^L (*highest* predicate).

that this type of evidence does not benefit from the *descending* clauses, especially for structure with a high k. Obviously, it has not been tested on the model without circumscription since it the lack of descending clauses makes it impossible to propagate truth values when evidence is provided on the highest predicates.

Evidences values Another interesting parameter to evaluate is the propagation of truth value among predicates regarding the values of evidences. Indeed, until now, it has always been considered that the evidences to propagate were true values, but in fact, nothing guarantee that propagation of a false value is equivalent. For this last experiment, the evidences are just considered to be false values, which, in terms of chronicles, might be interpreted as no observation of an event at a given time. Such a model is equivalent to Equation 4.27 with unary clauses replaced by their negation:

$$\begin{cases} (x_i^j \wedge x_{i+1}^j \implies x_i^{j+1}) \in \mathcal{R} \\ \neg x_i^1 \in \mathcal{R} \end{cases} \quad \text{with} \quad \begin{cases} 0 < j \leq L; \\ 0 < i \leq L - j + 1 \end{cases} \tag{4.28}$$

The results of inference are displayed in Figure 4.10 for the 3-SAT structure. Other k-SAT structures are not presented but have a similar behaviour. It appears that propagation of truth values along the structure is not balanced between true and false evidences. It may seem unimportant, but remember that for the chronicle representation LLEs are supposed to be uncertain (cf. Section 4.1.2), so the truth values of LLEs are supposed to be unknown. Consequently, if WalkSAT is trying to find a solution, it is more likely to find an assignment with zeros for predicates on higher layers. To

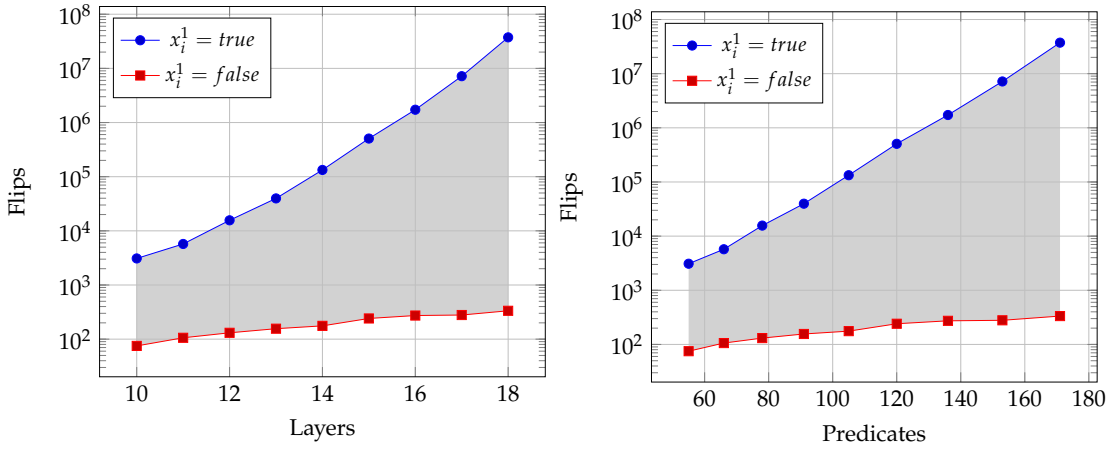


Figure 4.10: Flips necessary to reach a solution regarding the truth value of evidences.

Layers	10	11	12	13	14	15	16	17	18
Ratio	0.263	0.229	0.199	0.173	0.138	0.119	0.096	0.075	0.066

Table 4.3: Ratio of true assignments for Equation 4.29.

assert this, we modified Equation 4.25 slightly:

$$\begin{cases} (x_i^j \wedge x_{i+1}^j \implies x_i^{j+1}) \in \mathcal{R} \\ (x_i^{j+1} \implies x_i^j) \in \mathcal{R} \\ (x_i^{j+1} \implies x_{i+1}^j) \in \mathcal{R} \\ x_i^1 \iff x_1^L \in \mathcal{R} \end{cases} \quad \text{with} \quad \begin{cases} 0 < j \leq L; \\ 0 < i \leq L - j + 1 \end{cases} \tag{4.29}$$

Unary clauses have been replaced by double implications between the highest predicate and the lowest predicates of the structure. With this set up, only two assignments for the predicates are possible: all false or all true. Table 4.3 presents the ratio of true assignments for this model regarding the number of layers L where it appears clearly that truth assignment of value is unbalanced.

4.3 Consequences of WalkSAT inference on the chronicle model for MLNs

As we already discussed and showed in Chapter 3, the WalkSAT algorithm is the main core of the inference computation. Since MC-SAT is a simple Monte-Carlo approach,

its efficiency and performance rely exclusively on the quality of the sampling. In the previous section, we showed that SampleSAT, the algorithm used for producing the samples, might be reduced to WalkSAT since simulated annealing steps may not reach a worse truth assignment due to the presence of hard clauses that mostly prevents simulated annealing to reach a distant solution. This point should not be left out since the main goal of simulated annealing in SampleSAT is not to reach solutions of the logical problem[9], but to produce small random perturbations that allows WalkSAT to reach solutions with similar cost, but fundamentally distant in term of variables assignment [WES04]. Solutions are supposed to be found quasi uniformly, which is necessary for MC-SAT (cf. Algorithm 2, Section 3.1). Unfortunately, SampleSAT has been mostly tested on random 3-SAT instances, which does not assure that a specific problem might not be troublesome. Chronicle representation, as we showed, is structurally one of these troublesome problems. Consequently, if WalkSAT produces non-uniform samples, SampleSAT will not improve them. And, regarding our experiments, WalkSAT generates samples that might be strongly unbalanced regarding the overall shape of the chronicle. Nonetheless, simulated annealing steps on sample-SAT depend on two parameters when trying to reach a assignment that does not improve the cost: the cost and a fix value $Temp$. As explain in Section 3.1, such change might be performed with a probability $e^{-\Delta cost/Temp}$, so it should be theoretically possible to set the $Temp$ parameter high enough to counteract the cost of hard clauses[10], for instance by setting $Temp = max(\Delta cost)$ between every couple of assignment with a distance 1. Among the literature on MLNs, settings for SampleSAT are not given and it seems that, in *Alchemy*, $Temp$ is set to 1. Putting $Temp = max(\Delta cost)$ has not been tested so the possible effects are unknown, however applying such an approach might produce side effects, since simulated annealing would consequently always flip a predicate that is only connected to weighted clauses.

Sampling uniformity left aside, our experiments show that the inference process might be quite expensive for a structure like the chronicle representation. This is especially true for the propagation of true evidences as LLEs and CEs as well. We did not provide times since they are really dependent of the computer configuration, but, for instance, our experiments were performed on a computer with 16 Go of RAM and a 3.50GHz four core processor[11]. Computing a result taking at least 10^7 iterations is approximately done in more than 1s, while a result with 10^8 iterations is usually

[9]It has been shown that simulated annealing is inefficient for solving SAT problems alone [SKC+93].

[10]Remember that in practice, weight for hard clauses is not infinite, but defined at ahigh enough value to be largely bigger that any other weight

[11]In practice, only one core is used.

performed in more than 20s. These times might seem short, but remember that only short structures were presented that are far from a realistic application. Remember also that WalkSAT is used here as a sampler, and thus is necessarily called many times. In *Alchemy*, by default, the inference process requires 1 000 calls to MaxWalkSAT, which means that for a problem solved, in average, in 20s with MaxWalkSAT, it would require for the MC-SAT algorithm more than 5 hours of computation to provide an estimation. For example, an instance of Equation 4.25 with $L = 20$, which is 210 predicates, would be solved in more than 1 000s. Note that many optimisations were proposed to reduce the inference time like eliminations of predicates not required by a query[12] or variations of WalkSAT like TABU[13], but none of these optimisations seem efficient enough to counteract the observed effects. Indeed, exponential growth has to be considered, which means that slightly larger structures would be quickly intractable.

4.4 Conclusion

In this chapter, we presented our work to represent chronicles into the Markov logic formalism. Even if these formulae are generally simple, we showed that Markov logic semantic might be sometimes inadequate to represent specific rules or properties.

Prior results on this representation showed inconsistency produced by inference with MC-SAT. We provide a small example showing that MC-SAT might produces inconsistencies even on the simplest problem. Based on this problem, we discussed briefly the ability of MLNs to represent logic construction, and particularly how the choice of weights may impact a representation. We did not develop it further since it has already been expressed in [Jai11], where it is recommended to use Markov logic to construct Markov networks, but not as an extension of logic for probabilistic representations. Jain showed that weights have a big influence on the representation, but furthermore, that the impact of weights is highly correlated to the size of the network. Consequently, weight importance might change regarding the number of grounding predicates and clauses, meaning weights should be adjusted regarding the resultant network build from the model in Markov logic. Even for a model like the

[12]Predicate eliminations are really efficient to reduce the size of a network, but we showed that even small instance of a problem might be troublesome and, furthermore, it is difficult to remove a predicate in a CEP context since all events are connected together.

[13]TABU inference has not been fully tested here, but it seems that while it might produce slightly better results for the chronicle structure or the ternary chain, this is still not sufficient regarding the size of the problem.

chronicle representation that does not use so much weighted formulae, except for representing uncertainty for the data-stream, it would imply that a modification of the data-stream (e.g. its length) would impact the marginal probabilities of LLEs, even if weights are learned, which is not acceptable for online use.

In a second time, we present our experiment on the core algorithm for inference, i.e. WalkSAT. We showed that the overall design for the chronicle representation rises unexpected problems for inference with this algorithm and impacts directly the computation speed and the uniformity of samples produce which results in inconsistencies for MLN inference. Such problems have too many consequences for inference on the chronicle representation, which lead us to think that MLNs should not be used for our case and our goals.

Since a probabilistic approach based on sampling and local search algorithm is inefficient, it might be relevant to try other approaches from the spectrum of the SAT community based on exploratory techniques (like CDCL). For such an approach, ProbLog is an interesting method that combines logic representation with probabilities (cf. Section 3.2). In the next chapter, we present our work to represent chronicles with the ProbLog formalism.

CHAPTER 5

Exact computation of uncertainty for chronicles using ProbLog

Contents

5.1 Difference in semantic interpretation with Markov logic networks . 120
5.2 A chronicle representation with ProbLog 123
5.2.1 Event uncertainty 123
5.2.2 Chronicle representation 124
5.2.3 Conclusion 126
5.3 Instantiation of models under uncertainty using the ProbLog chronicle model 126
5.3.1 The hurricane model 127
5.3.2 The drone model 130
5.4 Conclusion 132

IN the previous chapter, we showed that MLNs were not suitable to represent chronicles due to their FOL structure. Since the main issue seems resulting from the sampling approach produced by a local search satisfiability algorithm, it may be beneficial to use exploratory approaches like BDD that store all the

possible solutions, but benefit of strong optimisation techniques provided by the SAT community. Such approaches were used in literature for CEP problems under uncertainty, but are not so common. Dimensionality may potentially be problematic for this kind of method because of the exploratory search that tends, in many applications, to face combinatory explosion of possible worlds regarding the size of logical problems. Nonetheless, they seem to provide fast computation for reasonably sized problems and exact computation regarding the defined model since every solution explaining a situation is explored and evaluated with regard to the probability provided.

In the literature, one method [MD11] is based on the Markov logic semantic to construct the network used for latter inference, but since we saw, in the previous chapter, that this formalism produces counter-intuitive representation due to weight and rules structures, we preferred to try an approach based on ProbLog used in [Ska+15b]. Indeed, even if Markov logics and ProbLog semantics look quite similar, they do not compute uncertainty the same way. Furthermore, ProbLog uncertainty representation describes directly rules probabilities without using a weight system, which is more convenient for designing the model.

In Section 5.1, we detail conceptual differences between Markov logic and ProbLog semantics, since, even if they are quite close, the interpretation for fomulae that look identical are quite distinct. Section 5.2 presents details about the chronicle representation using ProbLog and Section 5.3 applied it on two models: a toy-example representing a hurricane alarm and a representation of the drone model presented in Section 1.3.

5.1 Difference in semantic interpretation with Markov logic networks

Before presenting the chronicle model, we want to address a particularity between the interpretation of the semantic for ProbLog and Markov logic. Intuitively, one might think they are equivalent since they both represent clauses. Nevertheless, the probability associated to a clause is not exactly applied with the same meaning.

With MLNs, a weight on a clause indicates its confidence on the knowledge base and, consequently, how often it should be satisfied. This simple implication $a \implies b$ might be satisfied with three value configurations equally probable if not specified otherwise $[\{a : true, b : true\}, \{a : false, b : true\}, \{a : false, b : false\}]$. If this is the only formula in the knowledge base and it is hard, a and b are true respectively with $\frac{1}{3}$ and $\frac{2}{3}$ probability.

```
0.5::a(1).
0.5::b(1).
0.5::c(1).
a(X):-b(X),c(X).
```

```
0 A
0 B
0 C
B ^ C => A.
```

5.1: A simple example for semantic interpretation. Both codes represent the same program. Left code is Problog semantic. Right code is Markov logic semantic.

In contrast, in ProbLog, the probability attached expressed the chance that the head of the clause is proven if and only if conditions in the body are satisfied[1]. More specifically, a clause `0.7:: b(X):- a(X).` does not mean it should be satisfied in 70% of worlds, but that if $a(X)$ is proven, $b(X)$ is proven with 0.7 probability[2]. Reciprocally, if $b(X)$ is an evidence in the program, and without other formulae, $a(X)$ have to be proven too, independently of the 0.7 probability.

As a matter of example, we provide a basic program 5.1, written in ProbLog and Markov logic, that describes a implication between three predicates $a(X)$, $b(X)$ and $c(X)$. For each predicate, an unary clause is defined with a given probability. Readers may be confused with the zero weight in the Markov logic semantic, but they should remember it does not represent a probability but a weight where zero represents, in this case, a 0.5 probability. The implication formula is set as a hard constraint.

Even if these two codes may represent the same formulae, they do not represent the same program. In table 5.1, we provided the inference results given by the two codes. To understand these differences, we will now explain how these probabilities are computed for the case where $b(1)$ or $\neg b(1)$ are set as evidence. When the evidence is $b(1)$ (or `B` for MLNs), ProbLog supposes $b(1)$ to be proven, $c(1)$ might only be proven by its unary clause, while $a(1)$ might be proven both by its unary clause and by the implication. Two situations may arise. First, $a(1)$ is proven by its unary clause (without regard if it is proven by the implication) which has a 50% chance. Second possibility, $a(1)$ is not proven by the unary clause, which happens with an independent probability 0.5, but by the implication which supposes $c(1)$ to be proven, which happen with a probability 0.5. In this second case, the probability is the product of this two facts, consequently 0.25. The full probability for $a(1)$ to be proven is the summation of the probabilities of this two scenarii, consequently 0.75.

[1]This is true only if the head might be proven only by one clause. If the head might be proven by other clauses, to be proven it requires that at least one body among the clauses is proven.

[2]In this case, the formula is equivalent to $P(b(X)|a(X))$, but it would not be the case if $b(X)$ might be proven by another clause.

Evidence	ProbLog			MLN		
	$a(1)$	$b(1)$	$c(1)$	$a(1)$	$b(1)$	$c(1)$
$a(1)$	1	0.6	0.6	1	0.5	0.5
$\neg a(1)$	0	$0.\bar{3}$	$0.\bar{3}$	0	$0.\bar{3}$	$0.\bar{3}$
$b(1)$	0.75	1	0.5	$0.\bar{6}$	1	$0.\bar{3}$
$\neg b(1)$	0.5	0	0.5	0.5	0	0.5

Table 5.1: Inference results with ProbLog and MLN with different evidences.

With Markov logic, the only restriction is the implication $B, C \implies A$ to be satisfied. Since B is satisfied, the implication may be rewritten $C \implies A$. Only three assignments of truth values assert it: $[\{A : true, C : true\}, \{A : true, C : false\}, \{A : false, C : false\}]$ where A is true in two assignments which gives the $0.\bar{6}$ probability. This probability would change if unary clause weights for A and C were not the same as it would balance differently the probabilities of the three assignments.

For the case where $\neg b(1)$ is set as evidence, results are the same for both program but the underlying logic and computation remain quite different. For ProbLog, since $b(1)$ is not proven, $a(1)$ may not be proven by the implication, but just by the unary clause. With Markov logic, implication is satisfied so A and C are free from all constraints and have therefore 0.5 probabilty to be satisfied. Furthermore, readers may note that changing the weight for C in the MLN code would modify probabilities of A, while changing the probability of the unary clause of $c(1)$ would have no impact on the $a(1)$ probability.

ProbLog for chronicles The semantic used by ProbLog is quite convenient since it represents implicitly circumscription that is necessary for our approach and would be, consequently, much easier to provide in the chronicle model. Furthermore, the probability associated to rules that represents a conditional probabilistic relation between predicates from the condition and the head is independent of other rules where, at contrary, with MLNs, the importance of a weight is always relative to the other weights and formulae. But it is sill possible to represent dependencies since, in ProbLog, two rules with the same head are not independent any more. This formalism allows a great expressiveness for representing specific relations and, in particular, it allows an easier way to describe the rules previously expressed with existential quantifiers in MLNs.

5.2 A chronicle representation with ProbLog

Despite the difference presented, the chronicle representation based on ProbLog looks relatively similar from the MLN representation. In general, ProbLog semantic allows some simplifications that we present here. Note that in this chapter, we use the notation convention from ProbLog where predicates and constants start with a lower case letter, while variables start with an upper case letter. Since rules in ProbLog do not express implications, we will use the symbol $\vdash$ to separate head and body in a rule in a prolog semantic way. In the body, predicates are in conjunction and comma separated. Probability of a rule is written in front of it, separated with ::. Without probability, the rule has probability 1. For instance

$$p :: a(X, Y) \vdash b(X), c(Y) \tag{5.1}$$

represents a rule where the head $a(X)$ is deduced from the body $b(X), c(Y)$ with probability p when it is satisfied.

5.2.1 Event uncertainty

Remember that event uncertainty, and more precisely LLEs uncertainty, might be a loss of information that consequently does not appear in the data stream, or, at contrary, a noise introduced that might append randomly. This two types of uncertainty might be easily represented with ProbLog. To do so, LLEs and simple activities are separated in two predicates, respectively $ev(E, T)$ and $act(E, T)$, with E the type of the event and T the recognition time. *ev* predicates represent the information provided by the data stream and might be erroneous and *act* predicates the *hidden* states of the simple activity. Time is supposed to be discrete and, given a domain of instants $\mathcal{T}$ and a domain of LLE types $\mathcal{E}$, truth values of every predicate $ev(E, T)$ for $E \in \mathcal{E}$ and $T \in \mathcal{T}$ should be provided. Consequently, loss and noisy uncertainties might be represented as follows:

$$p_1 :: ev(E, T) \vdash act(E, T) \tag{5.2}$$

$$p_2 :: ev(E, T) \vdash \neg act(E, T) \tag{5.3}$$

$$p_3 :: act(E, T) \tag{5.4}$$

Equation 5.2 represents the loss uncertainty with p_1 the probability that an activity has been correctly recognized so its corresponding event appears in the data stream.

Reciprocally, $1 - p_1$ is the probability that activity did not produce any event in the data-stream. Equation 5.3 represents the noise uncertainty with p_2 the probability that, even without any activity, an event is recognised in the data-stream. Note that for Equation 5.2 the probability describes the normal scenario while in Equation 5.3 the probability describes the erroneous scenario. This representation is quite classic with PGMs since the predicate *ev* might be seen as an observation of a hidden state of the model represented by the predicate *act*. Equation 5.4 has been declared since the prior probability of predicates *act* might necessarily be provided. This representation assumes independence between activities, but dependencies with previous rules or more complex relations might be easily defined.

As for MLNs, transformation from a LLE to a chronicle is straightforward, nevertheless it should be defined from the predicate *atc* and not *ev* since *atc* is supposed to represent the real state of an activity.

$$ch(E, T, T) \vdash act(E, T) \tag{5.5}$$

ch predicate represents a chronicle where the first parameter is its type and the two last are the times when start and finish the recognition.

5.2.2 Chronicle representation

The chronicle model with ProbLog is quite similar with MLNs, but existential qualifiers are not necessary any more thanks to circumscription. Consequently, it is not required to pass by an intermediate predicate representing the relation between two recognitions. For instance, a chronicle *ab*, sequence of two chronicles *a* and *b*, would be represented as follows :

$$\begin{aligned} seq(T_1, T_2, T_3, T_4) &\vdash T_1 \leq T_2,\ T_2 < T_3,\ T_3 \leq T_4. \\ ch(ab, T_1, T_4) &\vdash ch(a, T_1, T_2),\ ch(b, T_3, T_4),\ seq(T_1, T_2, T_3, T_4). \end{aligned} \tag{5.6}$$

with $T_1, T_2, T_3, T_4 \in \mathcal{T}$. The *seq* predicate is in fact not necessary and might be represented directly with the definition of the chronicle, but this presentation is more comprehensible. If compared with the definition of chronicle in Chapter 1, note that it is required to define the limit constraints of chronicles like $T_1 \leq T_2$. This was specified with MLNs in a single constraining formula, but impossible to define in ProbLog so each rule should assert the times ordering of sub-chronicles.

Other relations might be defined equivalently, but note that with this representation, predicate *ch* is in fact not mandatory since its type might be directly represented

as a predicate. For instance, Equation 5.6 might be represented as well:

$$ab(T_1, T_4) \vdash a(T_1, T_2),\ b(T_3, T_4),\ seq(T_1, T_2, T_3, T_4). \tag{5.7}$$

Using the *ch* predicate has been preferred since it permits easier representation of *absence* and *and* operator that we briefly review.

For the absence, it is still necessary to pass by an intermediate predicate. So a possible definition of the absence predicate *abs* would be:

$$\begin{aligned} &during(T_1, T_2, T_3, T_4) \vdash T_1 < T_2, T_2 \leq T_3, T_3 < T_4. \\ &dur(C, T_1, T_4) \vdash ch(C, T_2, T_3), during(T_1, T_2, T_3, T_4). \\ &abs(C, T_1, T_4) \vdash \neg dur(C, T_1, T_4). \end{aligned} \tag{5.8}$$

Contrary to the other operators, *abs* predicate has an attribute C representing the type of the chronicle. As for the sequence operator *seq*, it seems more convenient for defining chronicle based on absence with this construction of *abs*. For instance, defining a chronicle *ab_not_c* that asserts that no chronicle *c* appears between a sequence of a chronicle *a* and a chronicle *b* named *ab* would be defined as follows:

$$ch(ab_not_c, T_1, T_4) \vdash ch(ab, T_1, T_4), abs(c, T_1, T_4). \tag{5.9}$$

Without using a *ch* operator, it would be necessary to create an intermediate predicate each time an absence chronicle is defined. The *ch* operator allows, to some extent, a generic specification.

Equivalently, the *and* operator is defined using the same approach and, as for MLNs, uses intermediate predicates to process the possible combination of chronicle arrangements.Equation 5.10 presents the definition of these intermediate operators.

$$\begin{aligned} and_1(T_1, T_2, T_3, T_4) &\vdash T_1 \leq T_2, T_3 \leq T_4, T_1 < T_3, T_2 < T_4. \\ and_2(T_1, T_2, T_3, T_4) &\vdash T_1 \leq T_2, T_2 \leq T_3, T_3 \leq T_4. \\ and(C_1, C_2, T_1, T_4) &\vdash ch(C_1, T_1, T_2), ch(C_2, T_3, T_4), and_1(T_1, T_2, T_3, T_4). \\ and(C_1, C_2, T_1, T_4) &\vdash ch(C_2, T_1, T_2), ch(C_1, T_3, T_4), and_1(T_1, T_2, T_3, T_4), C_1 \neq C_2. \\ and(C_1, C_2, T_1, T_4) &\vdash ch(C_1, T_2, T_3), ch(C_2, T_1, T_4), and_2(T_1, T_2, T_3, T_4). \\ and(C_1, C_2, T_1, T_4) &\vdash ch(C_2, T_2, T_3), ch(C_1, T_1, T_4), and_2(T_1, T_2, T_3, T_4), C_1 \neq C_2. \end{aligned} \tag{5.10}$$

where each and_i represents a different set of time arrangements then used to define the operator *and*. It is important that the rules used to define *and* do not cover twice

the same arrangement. Without asserting that the *and* rules are mutually exclusive, computation of probability would count the same recognition multiple time. For the same reasons, second and fourth rules for the *and* predicates assert that C_1 and C_2 are equals. Indeed, these rules are required to assert that a chronicle $C_1\&C_2$ produces the same recognition as $C_2\&C_1$, but if $C_1 = C_2$, *and* rules would capture two times each valid recognition. For instance, consider a stream $\varphi = [\langle a, 1\rangle, \langle c, 2\rangle, \langle b, 3\rangle]$ and two chronicles *ab* and *ac* that recognise respectively a sequence of LLEs *a* then *b* and a sequence of LLEs *a* then *c*. If an *and* chronicle is defined with these two chronicles $and(ab, ac, T_1, T_4)$, it should produce the same results as $and(ac, ab, T_1, T_4)$. If the chronicle is defined using the definition $and(ab, ac, T_1, T_4)$, recognition is produced by the fourth rule of predicate *and*, whereas using definition $and(ac, ab, T_1, T_4)$ would produce a recognition by means of the third rule. Now supposing a new *and* chronicle defined $and(ab, ab, T_1, T_4)$ for a stream $\varphi' = [\langle a, 1\rangle, \langle b, 2\rangle, \langle b, 3\rangle]$, only two recognition should be produced, but, without the condition $C_1 \neq C_2$, third and fourth rules would both be satisfied and would, consequently, produce all together four recognition and change the probability calculation.

5.2.3 Conclusion

In this section, we briefly presented our chronicle representation based on ProbLog. Although there are some subtleties, the chronicle representation with ProbLog is relatively close to the Markov logic one, but lighten the rules and the overall representation while providing a more comprehensible model for probabilities representation, dependencies and calculation. Since prior tests were encouraging, we applied it on two models to estimate its robustness against dimension.

5.3 Instantiation of models under uncertainty using the ProbLog chronicle model

In this section, we present two models on which we applied the chronicle representation. We will show that ProbLog, even if it performs indubitably better than the MLN representation, quickly faces combinatorial explosion. Nonetheless, these models will be valuable in the next chapter, since ProbLog provides exact computation of probabilities with respect to the model, they can be used to assert future results.

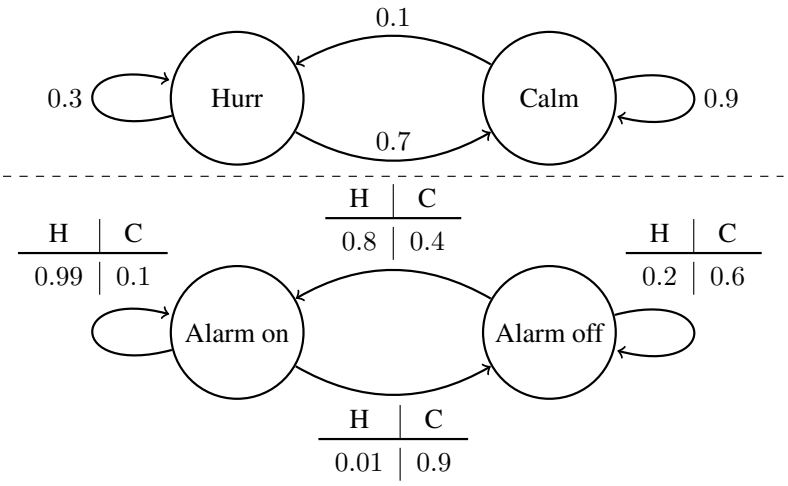


Figure 5.1: Hurricane alarm model and transition probabilities

5.3.1 The hurricane model

The first model represent a hurricane alarm. This alarm is supposed to activate when a sensor detects a hurricane. Unfortunately, the sensor is not fully reliable and may produce inconsistent observations. For instance, it may detect a hurricane during calm weather or, alternatively, not detect a real hurricane. We suppose this model to follow the Markov property:

$$\mathbb{P}\Big(X_{n+1} = j \mid X_0 = i_0, X_1 = i_1, \ldots, X_{n-1} = i_{n-1}, X_n = i\Big) = \mathbb{P}\left(X_{n+1} = j \mid X_n = i\right) \tag{5.11}$$

Figure 5.1 presents the model represented as a HMM. Probabilities of transition are displayed on edges. The full system is separated into two sub-systems representing the weather and the alarm. Top automaton represents the evolution of the weather from *Calm* to *Hurricane*. Chance of transition depends on nothing else than the current state. The bottom automaton represents the states of the alarm. Probabilities of state changes are conditioned by the current alarm and weather states. Concretely, state of the weather is the hidden variable of our model while the alarm state corresponds to the observation. So, contrarily to Equation 5.4, hidden activities are not provided with a prior probability, but are dependent of a Markov process.

To represent this model in ProbLog, we specified two predicates *state* and *trans*:

- The predicate $state(M, T, S)$ represents the state S of the given sub-system M at time T. For instance, $state(weather, 4, calm)$ indicates that the weather is calm at time 4.

- The predicate $trans(M, T, S_1, S_2)$ represents the probability transitions from a state to the other modelised by the edges in Figure 5.1. M defined the sub-system for the transition, T represents the time when the transition is fired, S_1 and S_2 are respectively the current state and the state at time $T+1$. For instance, $trans(alarm, 3, on, off)$ indicates that at times 3 the alarm was on state *on* and switched to state *off* for $T = 4$.

The change of state rule may be defined easily with these predicates.

$$state(M, T, S_2) \vdash trans(M, T', S_1, S_2), state(M, T', S_1), T' + 1 = T. \tag{5.12}$$

that simply expresses that a change of state from S_1 to S_2 for the sub-system M requires that a transition exists between these two states and that the current state is S_1. From now, each transition has to be defined with respect to the Markov chain. Transitions for the weather sub-system are represented as follows:

$$\begin{aligned} &0.1 :: trans(weather, T_1, calm, hurricane); 0.9 :: trans(weather, T_1, calm, calm). \\ &0.3 :: trans(weather, T_1, hurricane, hurricane); 0.7 :: trans(weather, T_1, hurricane, calm). \end{aligned} \tag{5.13}$$

The semantic used is specific to ProbLog and allows to declare the prior probability of predicates that should be mutually exclusive. Mutually exclusive predicates are separated by a semicolon.

Definition for the alarm sub-system is similar, but consider specific conditions like the state of the weather. The transitions for the alarm are defined as follows:

$$\begin{aligned} 0.4 :: trans(alarm, T_1, off, on); \quad & 0.6 :: trans(alarm, T_1, off, off) \\ & \vdash state(weather, T_2, calm), T_1 = T_2 + 1. \\ 0.8 :: trans(alarm, T_1, off, on); \quad & 0.2 :: trans(alarm, T_1, off, off) \\ & \vdash state(weather, T_2, hurricane), T_1 = T_2 + 1. \\ 0.1 :: trans(alarm, T_1, on, on); \quad & 0.9 :: trans(alarm, T_1, on, off) \\ & \vdash state(weather, T_2, calm), T_1 = T_2 + 1. \\ 0.99 :: trans(alarm, T_1, on, on); \quad & 0.01 :: trans(alarm, T_1, on, off) \\ & \vdash state(weather, T_2, hurricane), T_1 = T_2 + 1. \end{aligned} \tag{5.14}$$

Each rule is conditioned by the previous state of the weather. Mutual exclusivity is used in these rules too. So, for instance, the first rule describes that, from the state *off* in the alarm sub-system, a transition may be fired to the state *on* with probability 0.4 or stays in state *off* with probability 0.6 if, at the current, time weather is calm. Checking

the current state of the alarm is unnecessary since the condition is already required in Equation 5.12. Finally, the full system needs to include the starting states of the HMM:

$$\begin{aligned} &state(weather, 0, calm). \\ &state(alarm, 0, off). \end{aligned} \tag{5.15}$$

meaning that the system starts on a calm weather and with the alarm off.

Chronicle definition Chronicles may be defined easily on top of it, using the semantic provided in Section 5.2 and considering the alarm states as our LLEs and the weather states as our activity. So, as explained in the previous Section 5.2, chronicles are defined on the hidden states and not the LLEs.

Suppose we want to define the following chronicle

$$((hurricane\ hurricane) - [calm])\ equals\ ((off\ off) - [on]) \tag{5.16}$$

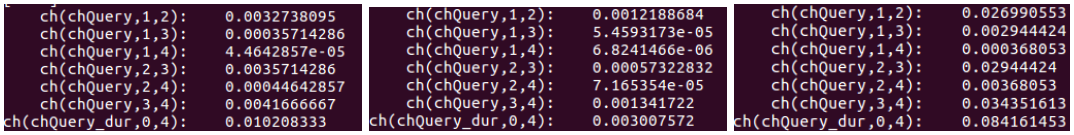
that represents a succession of time during which a hurricane was active but the alarm did not activate. This chronicle might be defined in the model with the following rules:

$$\begin{aligned} ch(seqHurricane, T_1, T_2) :- \quad & ch(hurricane, T_1, T_1), ch(hurricane, T_2, T_2) \\ & , seq(T_1, T_1, T_2, T_2). \\ ch(seqHNoCalm, T_1, T_2) :- \quad & ch(seqHurricane, T_1, T_2), abs(calm, T_1, T_2). \\ ch(seqOff, T_1, T_2) :- \quad & ch(off, T_1, T_1), ch(off, T_2, T_2) \\ & , seq(T_1, T_1, T_2, T_2). \\ ch(seqOffNoOn, T_1, T_2) :- \quad & ch(seqOff, T_1, T_2), abs(on, T_1, T_2). \\ ch(chQuery, T_1, T_2) :- \quad & ch(seqOffNoOn, T_1, T_2), ch(seqHNoCalm, T_1, T_2). \end{aligned} \tag{5.17}$$

Each rule describes the definition of an operator used in Equation 5.16. So *seqHurricane* defines the sub-chronicle $(hurricane\ hurricane)$, *seHNoCalm* describes the sub-chronicle $(hurricane\ hurricane) - [calm]$, etc. until the final definition of *chQuery* representing the full query.

For instance, given a stream $\varphi_1 = [\langle off, 0\rangle, \langle off, 1\rangle, \langle off, 2\rangle, \langle off, 3\rangle, \langle off, 4\rangle]$ and set the chronicle *chQuery* as query for the inference would return the probabilities in Figure 5.2a.

Each answer provides the probability that the query appear on a precise interval. The last line is a chronicle *chQuery_dur* computing the probability that the *chQuery* appears at least once in the data-stream. Obviously, it should not be the summation of



(a) Inference with φ_1 as evidence (b) Inference with φ_2 as evidence (c) Inference with φ_1 and a chronicle as evidences

Figure 5.2: Results for the query defined in Equation 5.16.

all the probabilities provided since the chronicle *chQuery* might be recognised more than one time per stream. Furthermore, it is possible to infer the probability even with partial data-stream. For instance probabilities inferred with the following stream $\varphi_2 = [\langle off, 0\rangle, \langle off, 4\rangle]$ are provided in Figure 5.2b.

Finally, it is possible to infer with a chronicle set as evidence. For instance, given a chronicle *hurricane during* $(off\ off)$ that expressed that at least one hurricane appears between two instants where the alarm was off and defined as evidence between time 1 and 4 the inference would provide the probability in Figure 5.2c.

Unfortunately, the computation time remains quite high and, paradoxically, providing evidences slows down computation. Without any evidence, the problem is intractable above fifteen units of time using the SDD structure[3].

5.3.2 The drone model

The second model implemented is the drone model presented in Section 1.3. The representation in ProbLog is, in fact, quite similar to its initial specification, but with small variations. The model is decomposed in autonomous sub-systems like for the hurricane model, for instance *RPS TC* or *ATC service* sub-systems. Sub-systems may influence each other in two different ways. Like for the hurricane model, state change of sub-system may be activated regarding the state of another sub-system, but it might be activated by an action produced during a change of state of another sub-system. For instance, when the RPS detects a telecommand loss, it should contact the ATC making the *ATC System* to change its current state. Furthermore, states may change when a period has elapsed. For instance, in the *RPS TC* sub-system, the state changes from *quick recovery* to *long recovery* when ZZ minutes elapsed. In our model time is discretised and should be represented in a number of instant. For short, change of state may be conditioned by state, transitions taken by sub-systems, and time.

[3]BDD and d-DNNF produce worse results.

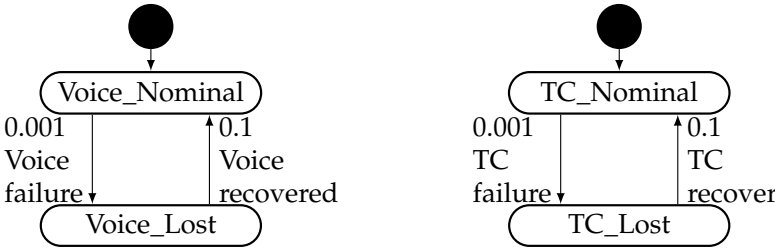


Figure 5.3: Sub-systems for the voice and telecommand failures.

This representation reveals a problem that was not initially mentioned since, at a given time, a sub-system may receive conflicting choices. For instance, the sub-system *RPS TC* might be in *quick recovery* since the necessary time to make it change to the *long recovery*, but, at the same time, telecommand might come back implying that the state should change to *nominal* as well. To be consistent, it is important to identify carefully possible conflicts and prioritised them. In our model, state changes produced by any other change in the system are prioritised over inactivity and time delays.

Finally, two small additional sub-systems (Figure 5.3) were added to the original model and represent the telecommand and radio failures. In our approach, states of these two sub-systems are hidden and changes of state may randomly appear with a given prior probability. Any change of state from one of these sub-systems is not supposed to be detected immediately by the RPS, the drone, or the ATC. Detection of a failure by a agent in this model is a random chance defined with a prior probability, but it would have been possible to model it otherwise. Obviously, if one of these sub-systems is in its *lost* state, transitions dependent from the corresponding communication cannot be fired.

This model was designed to be used in the point of view of one of the agents. The agent has a perfect knowledge of the state of its sub-systems, but may have only partial knowledge of the state of the other agents (and no information about the hidden sub-systems). This model is supposed to measure the risk of an undesired behaviour with potential partial knowledge on the global system. For instance, an interesting behaviour to detect would be to identify if two agents have a different understanding of the situation with no possibility to fix it except in case of a second failure. Such behaviour may append when radio and telecommand are lost at the same time, the RPS detects the telecommand loss and finally switches to mode *long recovery procedure*, but without the radio it cannot warn the ATC. At the same time the UA detects the telecommand loss too and sends a 7700 code which is detected by the ATC. After TT minutes, its state change to *urgency service*. But, during this time, the telecommand

came back and the RPS returned in nominal state. Consequently, even if the radio comes back, the RPS will never try to contact the ATC that will eventually consider that the drone is redirected, despite the fact that it is not and that the RPS does not know that the ATC is in urgency mode. Part of the corresponding ProbLog program is presented in Appendice B.

As well as the hurricane model, it is possible to infer the probability of a chronicle query with partial information and use instances of chronicles as observations. In the drone model, noisy information has not been represented, but it might be easily defined. Nonetheless, time computation remains the main problem since it is difficult for this model to go further than ten instants for the duration of the system.

5.4 Conclusion

In this chapter, we presented the definition of chronicles using ProbLog. We showed that, contrary to MLNs, this approach is more convenient for this task, notably with the semantic of rules that assumes closed world and circumscription. Furthermore, using directly a probability rather than a weight is more suitable and does not change the impact of rules for different sizes of model instantiation.

But, this approach is quickly intractable and not scalable on large models. Applying this approach on larger stream would require to simplify the model by, for instance, assuming independence between events that would suppress the Markov chain model.

Such representation based on ProbLog is not satisfactory since it is facing combinatorial explosion or bring too many restrictions.

Nonetheless, the dependency representation between events based on Markov chains presented here is often used in literature and, notably, among automaton-based approaches[4]. This is quite normal since automata are practical and used in many CEP representations since it is usually possible to define an automaton recognizing a given CE. Consequently, in the next chapter we explore an ad-hoc approach based on automaton and NHMs to model and produce consistent probability estimation of CEs.

[4]More complex structure may be preferred to Markov chains like DBNs ([MM07]).

CHAPTER 6

Chronicle inference on complex systems using Markov chains and automata

Contents

6.1 Uncertainty estimation using Markov chains 134
6.1.1 Initial approach 134
6.1.2 Application on a toy example 135
6.1.3 Discussion 138
6.2 Expressing high-level constraints using chronicles 139
6.2.1 Evidence behaviours expressed as sub-streams 140
6.2.2 Graph representations of constraint sub-streams 142
6.2.3 Sampling using constraint graphs 144
6.2.4 Finding the set of constraint sub-graphs for a chronicle 145
6.2.5 Discussion 147
6.3 Speeding up inference computation 148
6.3.1 Precisions about the chronicle representation 148
6.3.2 Model-checking approach for chronicle inference 149

6.3.3 Fast computation of the automaton of a chronicle evidence . . 151
6.4 Probability Estimation of a Behaviour of the Drone Model 151
6.4.1 Drone model to NHM 151
6.4.2 Results and Analysis 153
6.5 Conclusion 156

INCE previous approaches provided unsatisfactory results, in this chapter, we propose a new method to produce good estimates of a chronicle probability on an uncertain data-stream. This approach is inspired from previous works using automaton-based representation (cf. Chapter 2) and the work of Pachet, Roy, and Barbieri [PRB11] who generate music and lyrics samples with respect to a style learned with a NHM.

The common problem from the previous approaches was the non-scalability to the size of the model. MLNs are highly sensitive to the size of chronicles and ProbLog to the duration of execution making them quickly intractable or, worst, not reliable for the MLN case. Consequently, the method presented in this chapter focuses on its scalability for higher dimensional problems, and on allowing specifications of complex relations between LLEs using instances of CEs providing a high level of expressiveness for the declaration of prior knowledge.

Section 6.1 introduces our ad-hoc approach to evaluate the probability of a chronicle recognition. Section 6.2 presents the method to apply high-level constraints expressed as chronicles representing prior knowledge. Section 6.3 proposes improvement techniques to accelerate computation of our method. Finally, Section 6.4 applies our method to the drone model.

6.1 Uncertainty estimation using Markov chains

6.1.1 Initial approach

In [PRB11; Bar+12], the authors are interested in generating music partitions or lyrics using Markov chains. These Markov chains are usually learned with a specific corpus to capture its style, like jazz composition or lyrics. But such artistic productions may have many restrictions relative to the domain. For instance, lyrics usually rhyme and follows specific rhythmic which require temporal dependencies. These constraints are used to create a NHM (cf. Section 3.3) and then used to sample partitions or lyrics.

Our approach is inspired from this method where our model is represented by a Markov chain describing the (conditional) relations between the LLEs produced by the system. Accordingly, local constraints are the observations provided by the potentially erroneous data-stream. Note that the NHM is not used to compute the probability of a given data-stream as it is usually the case for Markov chains, but to find a real execution of the system explaining it.

For our case, we are not interested to find one possible execution explaining the given data-stream, but the probability of appearance of a chronicle *ch* regarding all possible explanations. An explanation is a sequence of states of the system that may produce the data-stream. Fortunately, given a Markov chain M and a data-stream φ, the resultant NHM $\tilde{M}$ asserts that sequences produced follows the probability distribution of M given the constraints. Furthermore, the probability distribution of the chronicle *ch* in this model is directly connected to the sequence probability distribution since

$$P_{\tilde{M}}(ch) = \sum_{s \in S_C} g(ch, s) \times P_{\tilde{M}}(s) \text{ with } \begin{cases} g(ch, s) = 1 \textbf{ if } s \text{ produces a recognition of } ch \\ g(ch, s) = 0 \textbf{ otherwise} \end{cases} \tag{6.1}$$

with s a sequence of states and S_C the set of all sequences of states explaining the data-stream. Consequently, it is possible to perform a simple MCMC method that samples explanatory sequences using the NHM and then parses each sample with a CE recognition system to identify if it produces a recognition of the chronicle *ch*. With enough samples, the sampled sequences would tend to probability distribution of possible sequences provided by the NHM and, consequently, the probability distribution of the chronicle recognition. In this thesis, we used CRL to describe chronicles and parse the samples. After parsing each sample, due to Equation 6.1, the ratio of recognition along the full set tends to the probability of recognition in the model represented by $\tilde{M}$. This process is summarised in Figure 6.1.

This process is relatively simple, but for convenience we provide a small example based on the hurricane model used in the previous chapter.

6.1.2 Application on a toy example

For this example, the goal is to reproduce the results obtained in the hurricane model by ProbLog in the previous chapter with the same experiments. As shown in Figure 6.1, this method requires four components: the Markov chain, the data-stream, the chronicle used as query, and the CE recognition system.

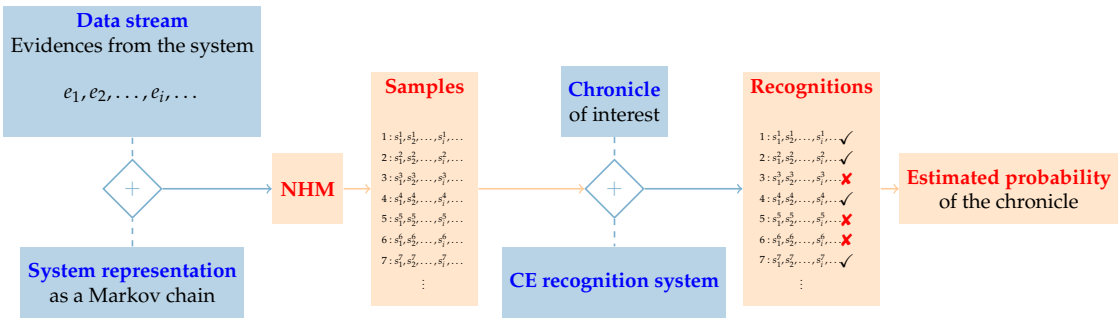


Figure 6.1: Initial sampling approach representation with the different parts required or calculated

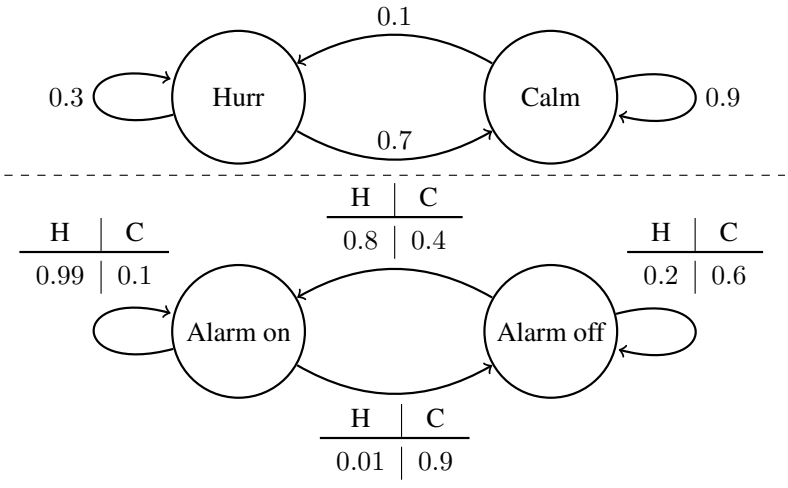


Figure 6.2: The hurricane alarm model.

C & Off	1
C & On	0
H & Off	0
H & On	0

(a) Initial

t \ $t+1$	C & Off	C & On	H & Off	H & On
C & Off	0.54	0.36	0.02	0.08
C & On	0.81	0.09	0.001	0.099
H & Off	0.42	0.28	0.06	0.24
H & On	0.63	0.07	0.003	0.297

(b) Transitions

Table 6.1: Marginal probability for the initial state and conditional probabilities from a state to another. (C:Calm weather, H:Hurricane, Off:Alarm off, On:Alarm on)

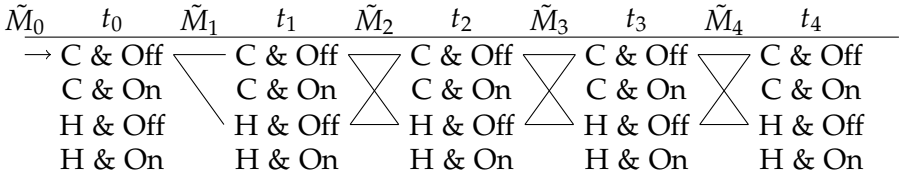


Figure 6.3: Dependency graph for the NHM $\tilde{M}$ computed for the hurricane model and the data-stream $\varphi_1 = [\langle off, 0\rangle, \langle off, 1\rangle, \langle off, 2\rangle, \langle off, 3\rangle, \langle off, 4\rangle]$. An edge between two states indicates a possible state change from t to $t+1$.

For the Markov chain, the model in Figure 6.2 is transformed into two matrices representing the probabilities for the initial state and the transition probabilities from an instant to the next one. Since the model is divided into two subsystems composed of two states each, it is transformed into one model with four states which is the Cartesian product of the subsystems' states represented in Table 6.1. Since all new states are mutually exclusive, it is just the product of the two probabilities provided by each subsystem. Remember, though, that a change of state of the alarm from t to $t+1$ is dependant of the state of the alarm at t and the state of the weather at $t+1$ and not the weather at t. For instance, the transition probability p from a state where the weather is calm and the alarm is off (C & Off) to the state where there is a hurricane with the alarm off (H & Off) is the product $p = 0.1 \times 0.2$ and not 0.1×0.6, since the probability for the alarm to change is conditioned by the new state of the weather.

For the data-stream, using the same $\varphi_1 = [\langle off, 0\rangle, \langle off, 1\rangle, \langle off, 2\rangle, \langle off, 3\rangle, \langle off, 4\rangle]$ as in the previous chapter, it is possible to generate the NHM suppressing all inconsistent states and transitions. For a stream φ_1, the corresponding graph of dependencies is presented in Figure 6.3 where each possible transition from a state to another according

to the time is represented. NHM is computed as explained in Section 3.3. Note that, even if the possible transitions from t_1 to t_2 are the same as from t_3 to t_4, their respective transition probabilities are not equal. For instance, in this case:

$$\tilde{M}_2 = \begin{pmatrix} 0.96961326 & 0 & 0.03038674 & 0 \\ 0 & 0 & 0 & 0 \\ 0.89215686 & 0 & 0.10784314 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tilde{M}_4 = \begin{pmatrix} 0.96428571 & 0 & 0.03571429 & 0 \\ 0 & 0 & 0 & 0 \\ 0.875 & 0 & 0.125 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.2}$$

With respect to the probabilities in the NHM, samples might be easily produced with a simple forward algorithm.

The last step requires to define a chronicle as query and to provide the CE recognition system to parse the samples. In this case, the chronicle is the same as in the previous chapter defined in Equation 5.16. Using the Chronicle Recognition Library[1] (CRL), each sample is parsed to detect a recognition. The ratio of samples producing a recognition approximates the probability of this chronicle to appear with φ_1. Note it is easy to specify queries asking for a certain number of recognition per stream. Indeed, CRL counts in each sample how many times the chronicle has been recognised, so it is possible to consider only samples that produce more, less, or exactly as specific number of recognitions. For instance, in this scenario, for 10 000 samples generated, CRL would detect approximately 101 samples producing at least one recognition of the chronicle in Equation 5.16 where, in average, 94 produce only one recognition, 6.5 produce 3 recognitions, and 0.5 produce 6 recognitions. This might not be done in ProbLog without defining a specific chronicle that represents n occurrences of a particular chronicle, but it may be performed with our MCMC approach with no additional cost.

6.1.3 Discussion

Speed and efficiency are mainly dependent on the number of samples and their length. Computation time is linearly dependent on the number of samples since each sample is the same length, but it is quadratic on the length of the samples due to the analysis with CRL. Time computation with the MCMC approach is around 300 seconds for 10 000 samples shorter than 60 LLEs. In Figure 6.4, computation speed for this problem is compared with Problog, MLN, and our MCMC technique. From these results, note that MLN speed is provided, but probability estimation is inconsistent; ProbLog is

[1] http://chroniclerecognitionlibrary.github.io/crl/o.html

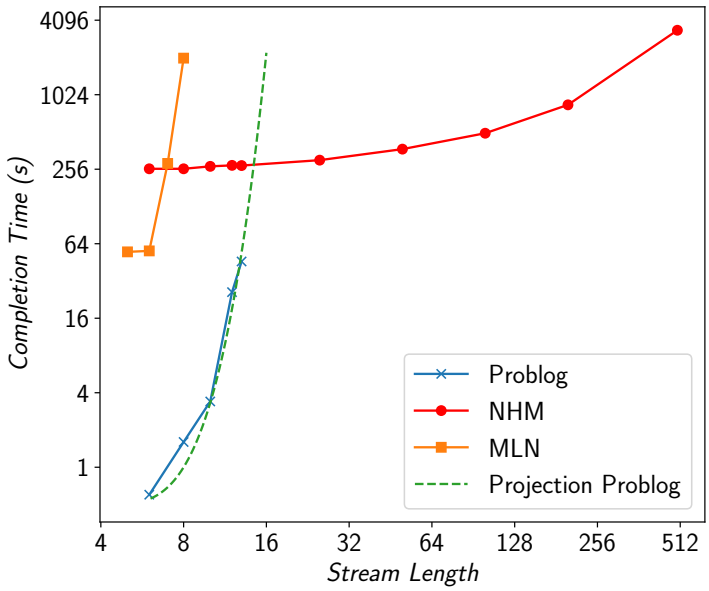


Figure 6.4: Inference time for 10 000 samples regarding stream length with ProbLog, MLN and our MCMC approach

faster than the MCMC approach but is intractable when samples length reached 16 elements[2].

Even if this approach may perform better than MLNs and ProbLog, it may still be slower than other works like Lahar [Ré+08] or PADUA [Mol+14]. Furthermore, our method does not allow defining chronicles as observations on the model, which is quite common with HMM or automaton based methods. Nonetheless, in the next section, we present an extension of our work allowing for specification of evidences as chronicles.

6.2 Expressing high-level constraints using chronicles

In the previous section, we presented a new approach to perform chronicle probability inference. But, remember that one of the main goal is to represent evidences as rules or high level expressions; in this case in the form of a chronicle. It is a complex task and few works achieved it and when a method achieved it, it usually comes with scalability issues.

[2]Problog produced, in fact, an out of memory with a stream length of 16 elements after hours of computation on 16Go of RAM.

For our method, we consider that a chronicle used as an evidence should be defined between an interval. This interval should be specified with two instants of time and not conditioned by other chronicles or events. The time interval might be the exact boundaries of the evidence chronicle or an interval defining the chronicle should appear in it[3]. We will present the different steps required to perform the inference. Figure 6.5 summarises the different steps and might be followed through the illustration.

6.2.1 Evidence behaviours expressed as sub-streams

Expressing a chronicle as a constraint over an interval is equivalent to consider as constraints all possible sub-streams Φ producing a recognition of this chronicle on this interval. A sub-stream should not be confused with the data-stream. In this case, a sub-stream is a complete explanatory succession of states of the system during an interval, while the data stream is just partial knowledge. In this section, we mostly consider the problem without specified data-stream, but it might be easily integrated to the final computation.

The set Φ represents all possible executions of the system regarding the constraints during two instants of time and one sub-stream is a possible execution of the system interval of time. Finding all the sub-streams is not our main purpose here, and may be defined as a CSP handled by a solver. To avoid combinatorial explosion of the solving, the chronicle is specified as a constraint over a restricted interval of time interval and not over the whole stream. For instance, specifying a sequence of two states as a constraint in a time interval gathers all the possible sub-streams on this interval where this chronicle should be recognised.

Considering that we have Φ, a naive approach would be to sample over each of these sub-streams $\varphi \in \Phi$, since each could be expressed with unary and binary constraints. Given the set Φ of sub-streams, the probability of a sample s on these constraints would be equal to:

$$P(s|\Phi) = \sum_{\varphi_i \in \Phi} P_M(\varphi_i|\Phi) \times P_{\tilde{M}_{\varphi_i}}(s) \tag{6.3}$$

where $P_{\tilde{M}_{\varphi_i}}(s) = P_M(s|\varphi_i)$ is the probability that the sample is generated by the model M under the constraint φ_i. Computing this probability is achieved using the previous approach based on NHM.

[3]Which is, in fact, equivalent to use an operator *During*.

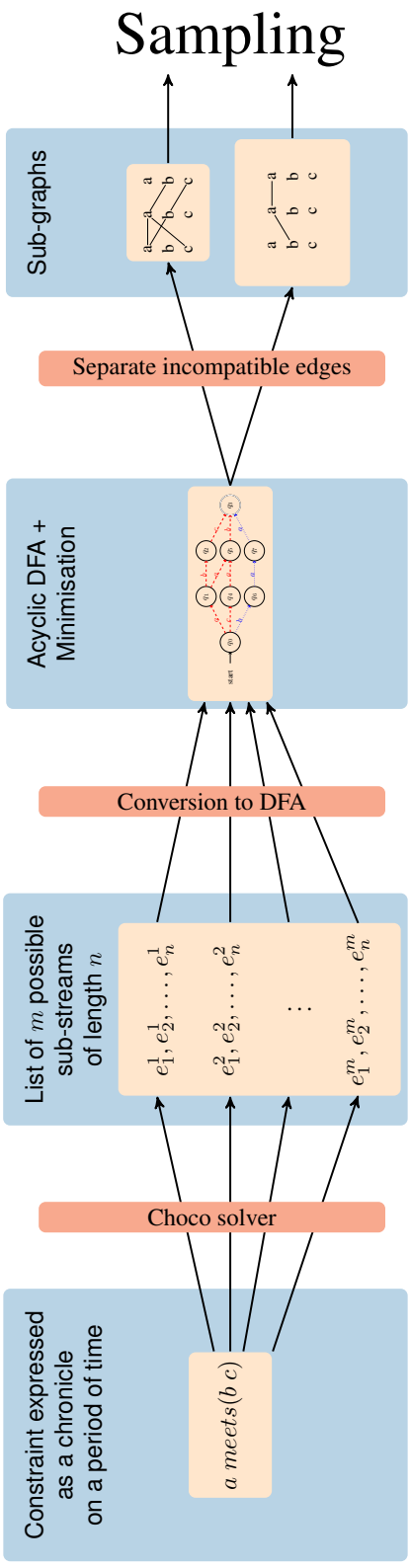


Figure 6.5: Chronicle processing as evidence for inference.

But, as no φ_i is included in another φ_j, the sequences that they might generate are independent. Consequently, if a stream s follows the constraints Φ, it can be generated by only one sub-stream $\varphi' \in \Phi$[4], meaning only one $P_{\tilde{M}_{\varphi'}}(s)$ has a non-zero probability. Furthermore, because of the independence:

$$P_M(\varphi_i|\Phi) = \frac{P_M(\varphi_i)}{P_M(\Phi)} \tag{6.4}$$

And if φ' is the sub-stream producing s, Eq. 6.3 may be rewritten:

$$P(s|\Phi) = \frac{P_M(\varphi')}{P_M(\Phi)} \times P_{\tilde{M}_{\varphi'}}(s) \tag{6.5}$$

However, this approach requires sampling over all sub-streams and their number grows exponentially with their length (depending of course on the chronicle: the more it is constraining, the less the possible outcomes), it could lead to a drastic amount of sampling even using parallel computing. A solution might be to adapt the number of samples regarding the probability of the sub-stream. For instance, if computation uses N samples, the number of samples for this sub-stream might be reduced to $|P_M(\varphi_i|\Phi) \times N|$. Unfortunately, this technique cannot be used for a fixed number of samples N if $|\Phi| > N$ and especially when, given a sub-set $\Phi' \subseteq \Phi$ with $\varphi_j \in \Phi'$, $P(\varphi_j|\Phi) < \frac{1}{N}$ but $P(\Phi'|\Phi) \not\approx 0$. In such conditions, it might still be possible to use more samples, but it still requires to compute many NHMs which would slow down computation. To counteract this, we propose an alternate representation of Φ based on graphs and automata that reduces the number of NHMs required.

6.2.2 Graph representations of constraint sub-streams

To reduce the number of necessary samples, we take advantage of the NHM structure to represent the constraint streams. Indeed, consider as an example the dependency graph for the NHM in Figure 6.3, it represents a set of binary constraints between states, but, as well, the set of possibles streams explaining some evidences. Following this idea, since a chronicle might be represented as a set of sub-streams, these sub-streams might be represented as a set of dependency graphs as well. These graphs might be considered as a constraint on the interval specified for the NHM structure. We name these graphs *constraint sub-graphs* and define the notion of acceptance:

[4]Note that the opposite is not true since many streams might be generated by the same φ'.

$$\varphi_w = [\langle a,1\rangle, \langle b,2\rangle, \langle c,3\rangle]$$
$$\varphi_x = [\langle a,1\rangle, \langle a,2\rangle, \langle b,3\rangle]$$
$$\varphi_y = [\langle c,1\rangle, \langle a,2\rangle, \langle b,3\rangle]$$

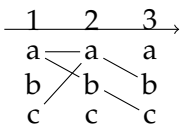


Figure 6.6: Three streams represented as an accepting graph

$$\varphi_z = [\langle b,1\rangle, \langle a,2\rangle\langle a,3\rangle]$$

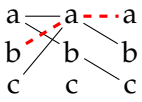


Figure 6.7: Non accepting graph for $\Phi' = \Phi \bigcup \varphi_z$

Definition 6.2.1. A constraint sub-graph is *accepting* for the constraint sub-streams if all paths are in bijection with them.

For instance, if a constraint Φ is defined from the streams $\{\varphi_w, \varphi_x, \varphi_y\} = \Phi$ in Figure 6.6 they may be represented in one sub-graph. Following the different paths from left to right reproduces Φ, consequently, this graph is accepting for Φ. But not all sets of constraint might produce only one accepting graph. For instance, if a new set of constraint streams $\Phi' = \Phi \bigcup \varphi_z$ is defined with $\varphi_z = [\langle b,1\rangle, \langle a,2\rangle\langle a,3\rangle]$ it is not possible to represent it with one graph since it would not be accepting. Indeed, the graph directly constructed from Φ' in Figure 6.7 would recognise the sub-stream $[\langle a,1\rangle\langle a,2\rangle\langle a,3\rangle]$ which is not part of the set Φ'. Nonetheless, it might be represented as a set of accepting graphs for sub-set of Φ'. Figure 6.8 represents a possible set of accepting graphs for Φ' where the left graph is accepting for Φ and the right is accepting for φ_z, so this set of graph is accepting for Φ'.

Note that the number of sets is entirely correlated to the set of sub-streams used as constraints and not correlated to the number of sub-streams. For instance, defining a set $\Phi'' = \Phi' \bigcup \varphi_\alpha \bigcup \varphi_\beta \bigcup \varphi_\gamma$ with $\varphi_\alpha = [\langle a,1\rangle, \langle a,2\rangle, \langle a,3\rangle]$, $\varphi_\beta = [\langle b,1\rangle, \langle a,2\rangle, \langle b,3\rangle]$, and $\varphi_\gamma = [\langle c,1\rangle, \langle a,2\rangle, \langle a,3\rangle]$, it is possible to represent this set in only one accepting graph (cf. Figure 6.9).

Figure 6.8: Accepting graphs for $\Phi' = \Phi \bigcup \varphi_z$

$$\varphi_\alpha = [\langle a,1\rangle, \langle a,2\rangle, \langle a,3\rangle]$$
$$\varphi_\beta = [\langle b,1\rangle, \langle a,2\rangle, \langle b,3\rangle]$$
$$\varphi_\gamma = [\langle c,1\rangle, \langle a,2\rangle, \langle a,3\rangle]$$

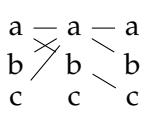


Figure 6.9: Accepting graph for $\Phi'' = \Phi' \bigcup \varphi_\alpha \bigcup \varphi_\beta \bigcup \varphi_\gamma$

6.2.3 Sampling using constraint graphs

Given a set of graphs G representing all possible constraint sub-streams Φ_G regarding a specific chronicle set as evidence, it is possible to use each graph $g \in G$, representing the set of sub-streams $\Phi_g \in \Phi_G$ as constraint to define a corresponding NHM $\tilde{M}_g$. Regarding the model M and an explanatory stream s, the probability for this stream to be produced by a graph g is

$$P(s \in \Phi_g) = P(\Phi_g|\Phi_G) = \frac{P_M(\Phi_g)}{P_M(\Phi_G)} \tag{6.6}$$

Indeed, $\Phi_g \bigcup \Phi_{g'} = \emptyset$ for any $g, g' \in G$ with $g \neq g'$. This marginal probability of a set $P_M(\Phi_g)$ is the product of the unnormalized constrained matrices of the NHM $\tilde{M}_g$, which is easily evaluated during the backward algorithm. And $P_M(\Phi_G)$, due to independence between graphs, is the sum of probabilities for all graphs:

$$P_M(\Phi_G) = \sum_{g \in G} P_M(\Phi_g) \tag{6.7}$$

Consequently, the probability $P(s|\Phi_G)$ of a stream s is similar to Equation 6.3, but considering sets of sub-streams:

$$P(s|\Phi_G) = \sum_{g \in G} P_M(\Phi_g|\Phi_G) \times P_{\tilde{M}_{\Phi_g}}(s) \tag{6.8}$$

Equivalently, only one graph may produce the stream s so $P_{\tilde{M}_{\Phi_g}}(s) = 0$ for every graph that cannot produce s. So, given the graph g_s that can produce s:

$$P(s|\Phi_G) = P_M(\Phi_{g_s}|\Phi_G) \times P_{\tilde{M}_{\Phi_{g_s}}}(s) \tag{6.9}$$

with $P_{\tilde{M}_{\Phi_{g_s}}}(s)$ evaluated with the NHM $\tilde{M}_{\Phi_{g_s}}$. From this point, sampling might be performed with different approaches. Each NHM may be used to produce the same number of samples and the results are weighted regarding $P_M(\Phi_{g_s}|\Phi_G)$ or samples are produced with respect to the distribution $P_M(\Phi_{g_s}|\Phi_G)$.

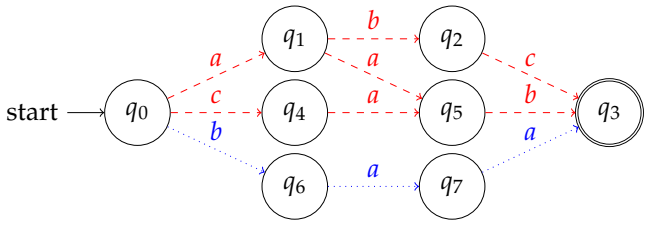


Figure 6.10: Minimal automaton $\mathcal{A}_{\Phi'}$ for the streams $\Phi' = \varphi_w, \varphi_x, \varphi_y, \varphi_z$ and resulting separation.

6.2.4 Finding the set of constraint sub-graphs for a chronicle

We did not explain how to find the set of constraint sub-graphs representing the chronicle set as evidence, which is not trivial. In Section 6.2.1, we showed that a chronicle set as evidence in a time interval might be expressed as a set Φ of sub-streams. Each sub-stream is finite and has the same length, so, given Φ, it is possible to represent them into an acyclic DFA $\mathcal{A}$. The language of this automaton is the sub-streams set it represents $\mathcal{L} = \Phi$ and its alphabet Σ is the set of states of the Markov chain M. For instance, the sub-streams set Φ' previously defined (cf. Figure 6.7) might be represented into the minimal automaton in Figure 6.10 with $\mathcal{L} = \Phi'$ and $\Sigma = a, b, c$.

The automaton representation is used to find the accepting graphs for Φ.

Definition 6.2.2. An automaton $\mathcal{A}$ is accepting for a set of sub-stream Φ if this set might be represented in only one accepting graph.

If an automaton $\mathcal{A}$ is not accepting for Φ, it should be divided into a set of automata $\mathfrak{A}$ where each $\mathcal{A}_i \in \mathfrak{A}$ is accepting for its corresponding language $\Phi_i \in \Phi$. Languages between two automata should have no intersection.

Luckily, this might be achieved quite easily since it requires to identify *incompatible streams* in the automaton

Definition 6.2.3. Two edges of the DFA are incompatible if they are at the same depth, share the same transition from Σ leading to two distinct states in $\mathcal{Q}$. Two sub-streams are incompatible if they each contain one of two incompatible edges.

If an automaton has incompatible streams, the corresponding sub-streams set is not representable in one accepting graph. Inversely, if there is no incompatible stream, the automaton might be represented into an accepting graph. Conversion is almost straightforward from an automaton to a graph since edges from the automaton are nodes of the dependency graph and are connected if consecutive edges in the automaton have corresponding labels.

For instance, in Figure 6.10, transitions between q_1, q_5 and q_4, q_5 lead to the same state and thus are not incompatible. By contrast, transitions between q_4, q_5 and q_6, q_7 are incompatible as they are at the same depth from the initial state, depend on the same symbol a, and lead to different states. In the figure, a possible division of the automaton is proposed in two automata: one with dashed red edges and the other with dotted blue edges. Note that these automata are the representation of an accepting graph set for Figure 6.8.

For the automaton division, we proposed a breadth-first division algorithm recursive on depth detailed in Algorithm 4 and that we briefly explain. Lines 1 and 2 take all the edges at the specified depth and construct a set of sets of edges where no set contains two incompatibles edges. Many approaches may be proposed for this task, but we just applied a pushing algorithm that place each edge one by one in the first set without incompatibility. If each existing set as an incompatible edge with the current one, a new set is created. Line 5 creates a new automaton using a set of edges and takes all descendant and ancestors edges accessible from the current set. This assert than no sub-stream is removed during the division. Lines 9 and 7 define the recursion and end criteria. Algorithm stops when each recursion reached the terminal node.

Algorithm 4: Split_automaton

inputs : $\mathcal{A}$, the automaton associated to the language
d, depth of a layer
outputs: sol, a set of automata with no incompatible edges

1 $edges_depth \leftarrow$ all edges at depth d
2 $edges_sets \leftarrow$ sets of edges from $edges_depth$ with no incompatible edges
3 **for** $set \in edges_sets$
4 Create new automaton $\mathcal{A}'$ with all edges from set, descendants of set and ancestor of set
5 Minimize $\mathcal{A}'$
6 **if** $d = len(\mathcal{A}')$
7 add $\mathcal{A}'$ in sol
8 **else**
9 add $Split_automaton(\mathcal{A}', d+1)$ in sol
10 **return** sol

This algorithm does not necessarily provide a minimal splitting in terms of amount of sets, but ensures that the resultant sets are indeed independent, and is sufficient for our purpose. Finding the optimal separation that minimises the number of automaton seems difficult to perform, and we did not investigate further this open question.

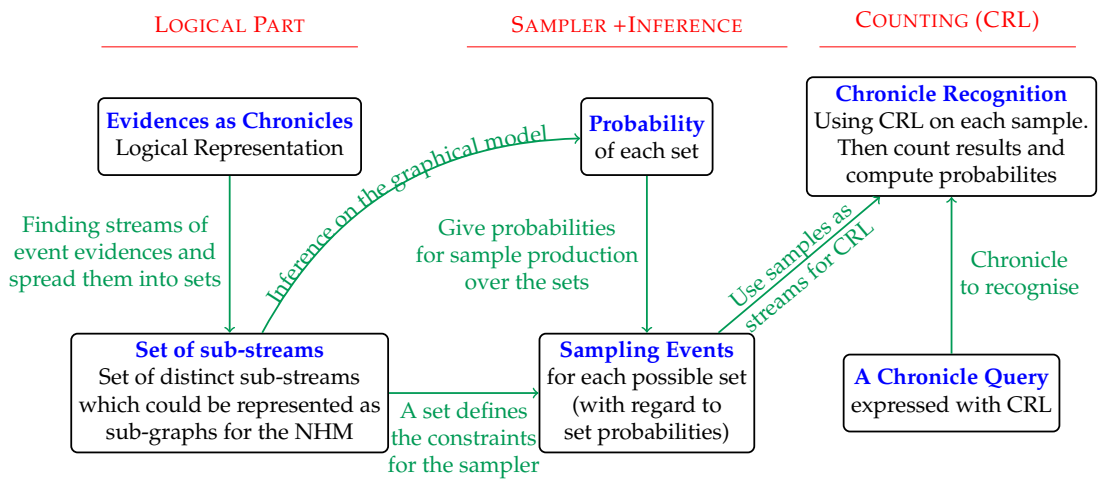


Figure 6.11: Full process representation for the MCMC approach with chronicles set as evidence.

6.2.5 Discussion

In this section, we presented an original method, summarised in Figure 6.5, to represent a chronicle set as evidence into a set of constraint sub-graphs. From the overall perspective of the sampling, represented in Figure 6.11, the full process is quite similar to the process without complex prior knowledge, and just performs prior transformations for evidence chronicles (cf. Logical part in Figure 6.11). We considered initially for the explanations that no data-streams were provided, but setting the constraints from the data-stream over the NHMs is almost straightforward since it just requires to constrain every NHM with the information from the data-stream. Since such process provide new constraints to NHMs, it may potentially lead to remove inconsistent NHMs (with zero probability), but it does not break independence between NHMs.

Compared with ProbLog (cf. Figure 6.12), inference with the MCMC approach produces expected results regarding the number of sample. Meaning that it is possible to obtain estimations with one percent error in order of minutes on models that ProbLog could not process.

Despite these results, this technique may, in some perspectives, be considered slow since it may takes several minutes and perhaps hours for long duration analysis. Furthermore, CE recognition may takes extensive times depending on specific samples and query chronicles.

In the case of evidence chronicles, two strong critiques might be made. Due to the CSP phase, which may easily produce combinatorial explosion, interval used to bound

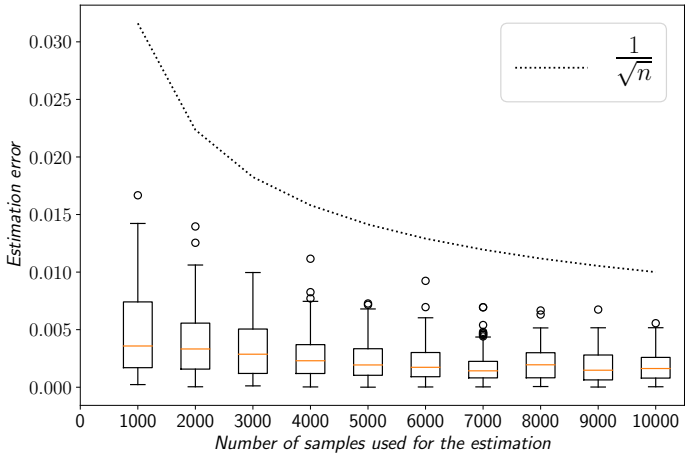


Figure 6.12: Estimation error compared to exact evaluation (based on ProbLog).

the evidence chronicle has to be particularly small (less than ten instants in general) and may easily have a long computation time, slowing down the overall method. Furthermore, the division of the resultant automaton produces uncertain results. Indeed, since the chronicles tested are restricted in size, because of the interval boundary, automata tested were always small[5] and rarely produce more than four divisions, but nothing asserts that, in case of a bigger automaton, divisions are contained and do not produce a large amount of automata. In the next section, we propose different improvements mainly to reduce the overall computation time.

6.3 Speeding up inference computation

In this section, we propose different extensions or modification of our work with aim to improve the speed of our method. They may have some restrictions, but usually speed up computation considerably.

6.3.1 Precisions about the chronicle representation

In our work, we did not consider that, despite chronicles are expressed with a context-free grammar, they may easily be represented as DFA. In fact, in the early implementation of chronicles, they were represented by *duplicate automata* [KCC10]. Duplication

[5]Small regarding their number of states, but, nonetheless, they still represent thousand of sub-streams.

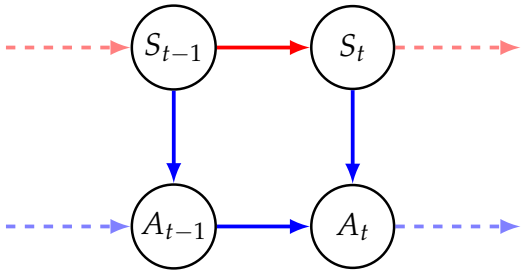


Figure 6.13: Markov chain representation for chronicle inference. In blue $P(A_t|A_{t-1}, S_t)$ provided by the automaton, in red $P(S_t|S_{t-1})$ provided by the NHM.

of automata was necessary to assert historisation of recognition (cf. Chapter 1), but, in most of the cases treated in this thesis, enumerating all possible recognitions regardless of their probability is not interesting and maybe not feasible.

During this thesis, transformation of a chronicle representation into an automaton was not studied, but we are confident that this task may be achieved and automatised regarding existing representations for chronicles like duplicate automata and Petri nets [KCC10; Pie14]. For the presented extensions, we will assume that a DFA representation exists and may be easily found. We do not think this hypothesis to be too strong, since most chronicle operators may be represented as combination of sequences. For instance, a chronicle $(a\ b)$ *overlaps* $(c\ d)$ might be rewritten as a chronicle made only with sequences $a\ c\ b\ d$, which may be directly represented as a DFA, even if a, b, c and d are as well DFAs representing chronicles.

6.3.2 Model-checking approach for chronicle inference

Supposing that, for a chronicle set as query for inference, its DFA representation is known, it is possible to speed up inference and compute the exact probability with respect to the model. Indeed, given the variable S_t that represents the state of the system at time t and A_t the state of the automaton $\mathcal{A}$ describing the chronicle, $\mathcal{A}$ might be seen as representing the conditional probability $P(A_t|A_{t-1}, S_t)$[6]. The problem might consequently be represented as a Markov chain like in Figure 6.13 where $P(S_t|S_{t-1})$ is provided by the NHM and $P(A_t|A_{t-1}, S_t)$ by the DFA. Computation is done trivially as for a Markov chain using forward propagation. The probability for the chronicle will be the probability at the end of the process that the automaton reached a terminal state.

[6]Nonetheless, in this case, given A_{t-1} and S_t fixed, $P(A_t|A_{t-1}, S_t)$ is equal to one for only one state and null for all the other states.

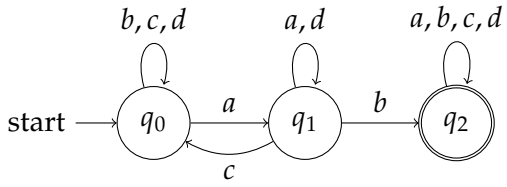


Figure 6.14: DFA for chronicle $(a\ b) - [c]$ with $\Sigma = a, b, c, d$

	q_0	q_1	q_2
a	0	1	0
b	1	0	0
c	1	0	0
d	1	0	0

(a) $P(q_0|A_{t-1}, S_t)$

	q_0	q_1	q_2
a	0	1	0
b	0	0	1
c	1	0	0
d	0	1	0

(b) $P(q_1|A_{t-1}, S_t)$

	q_0	q_1	q_2
a	0	0	1
b	0	0	1
c	0	0	1
d	0	0	1

(c) $P(q_2|A_{t-1}, S_t)$

Table 6.2: Representation of chronicle $(a\ b) - [c]$ with $\Sigma = a, b, c, d$ as conditional probabilities $P(A_t|A_{t-1}, S_t)$

As a matter of example, if we define a chronicle $(a\ b) - [c]$, which is a sequence $a\ b$ without c between them, where a, b, c are states of an imaginary system composed of four states $\Sigma = a, b, c, d$, the chronicle might be represented as the DFA in Figure 6.14. Rewritten as a conditional probability $P(A_t|A_{t-1}, S_t)$, the automaton would be represented by the three probability table in Table 6.2.

Using this approach is extremely beneficial for computation speed since inference computation would not require sampling any more. For instance, on a data-stream of length 500, this approach would perform the calculation in less than a second, while the sampling approach would take around 1 000 seconds. For scale comparison, this approach may process a data-stream of length 10^6 in 30 seconds. Note that this method has no impact on the NHM computation, with or without chronicle set as evidences so it might be used with no restriction. But, there is still difference with the sampling approach because this method cannot compute the probability that a chronicle was recognised n times. To keep track of the number of recognitions it would be necessary to design a chronicle c that recognises n times a chronicle c'. It is still possible to represent it, but the chronicle would be quite complex to write without automatic process for large n.

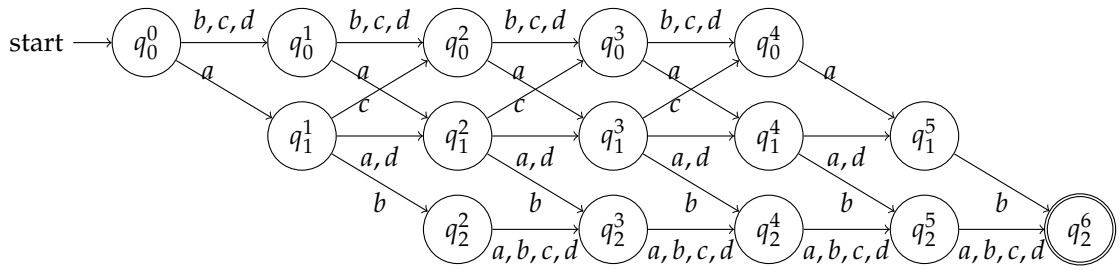


Figure 6.15: Acyclic version of DFA in Figure 6.14 with seven iterations

6.3.3 Fast computation of the automaton of a chronicle evidence

The main issue when setting chronicles as evidences is to find the corresponding acyclic automaton. With a CSP approach, computation is highly inefficient and restrains drastically the range of the interval for the chronicle. Instinctively, using the DFA seems the simplest way to do it since the representation is given. But, remember that, in order to constrain the NHM and find the independent constraint sub-graphs, it is necessary to find the acyclic DFA, computed initially by the CSP solver. In some automaton-based approaches for CEP uncertainty, like the AIG in [SKK08; WCZ13], or the tree structure extension on cas-graph from [Faz+18a], alternative executions are usually expressed in a tree structure, resulting of all possible executions of the DFA. Such representation might be expensive to compute, as for the CSP solving approach. It is, in fact, unnecessary since it is possible to construct directly the acyclic automaton $\tilde{\mathcal{A}}$ from the non-acyclic DFA $\mathcal{A}$.

Indeed, since the interval length l of the constraint is known as well as $\mathcal{A}$, $\tilde{\mathcal{A}}$ might be constructed iteratively developing $\mathcal{A}$ l times with only constraint to start from an initial state and to end in a final state. For instance, the DFA in Figure 6.14 might be developed in its acyclic version of length 7, presented in Figure 6.15.

6.4 Probability Estimation of a Behaviour of the Drone Model

6.4.1 Drone model to NHM

As for the ProbLog approach, the drone model has been tested with the Markov chains method. Some modifications have been introduced from the original model presented in Section 1.3. The new model is presented in Figure 6.16. First difference is that the UAV is not represented. For this model, it has been chosen to represent all information provided by the drone as a stochastic process. For instance, a transponder code change

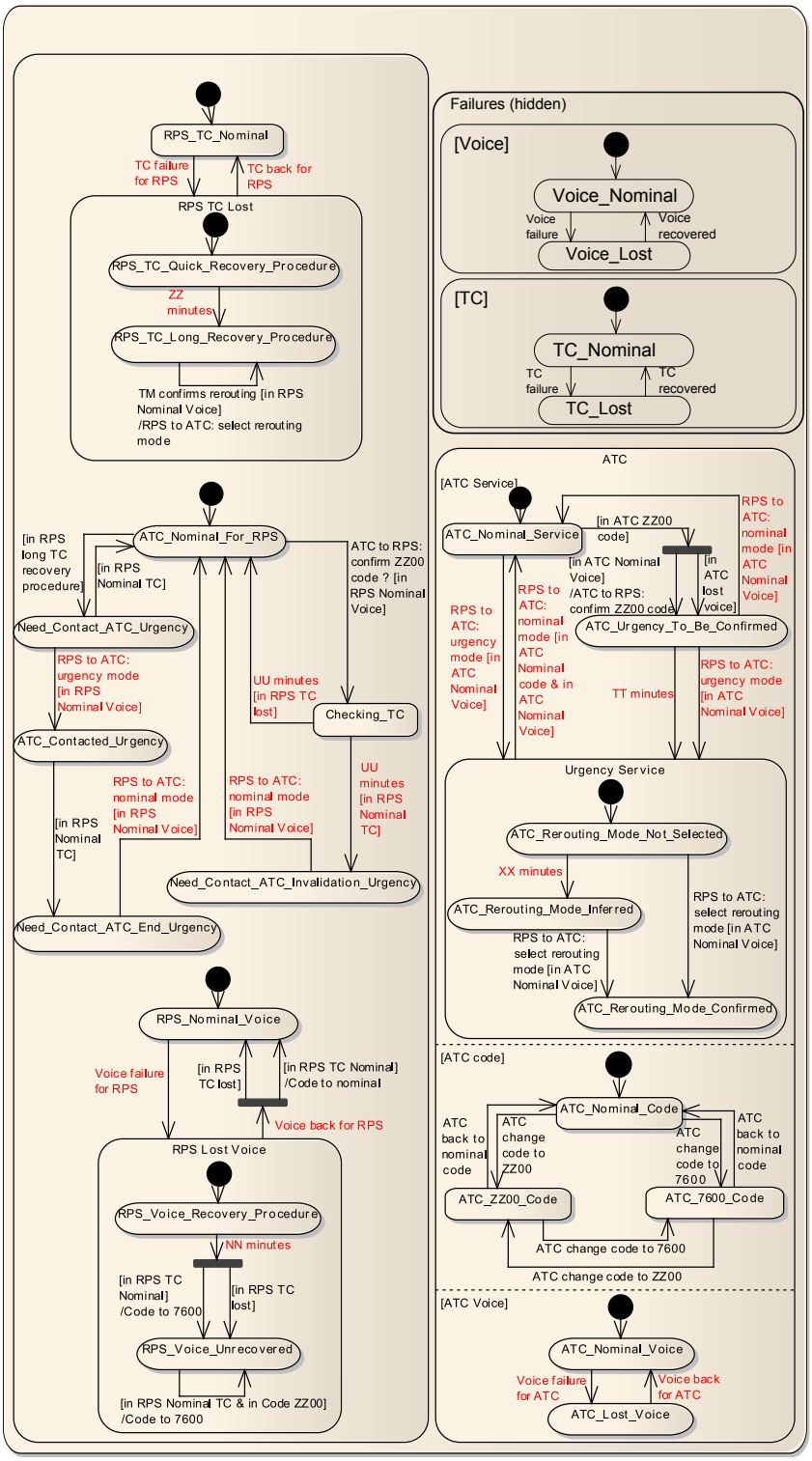


Figure 6.16: Communication lost protocol

is just defined by a probability dependent of other sub-systems, like the current TC state. This allows to reduce the size of the matrix used for computation.

As the method needs to be structured as a Markov chain, we transform our set of sub-systems into one general system where states are combinations of possibles states of all sub-systems. Initially, there were eight sub-systems that could change between two and six states each leading, after transformation, to an unique system of 6 480 states. Fortunately, not all the states are reachable so the system might be reduced down to 3360 states with their transition probabilities. All transitions are represented into a matrix M used then to build the NHM.

For the statistical model, humans are not supposed to misunderstand each other during communication, even if in real life it could be a frequent source of system failure. Furthermore, it has been chosen to not represent waiting time between two states with a finite counter but with a transition probability which may be triggered at each instant. For instance, transition between `TC_Quick_Recovery_Procedure` and `TC_Long_Recovery_Procedure` is supposed to be `ZZ minutes` delay, but on our model it is described with a chance at each time to change the current state. It would have been possible to represent it as a true delay but it would have increased drastically the size of our model, without being really interesting for this study.

6.4.2 Results and Analysis

For performance analysis, a specific test case has been designed using the new drone model and with one chronicle specified as a constraint. The study does not use the speed up approaches presented in Section 6.3 and focuses on the sampling method.

The chronicle defined as constraint is the following:

```
ATC_Rerouting_Mode_Not_Selected
ATC_Rerouting_Mode_Inferred
ATC_Rerouting_Mode_Confirmed
```

This chronicle is really simple as it describes that a sequence of three events happened during the execution: the controller detected a failure, then supposed a rerouting, which was finally confirmed by the RPS. Still, other events might be triggered during this sequence. This behaviour happened during time 20 to 27. Expressed as constraint streams, it represents 11 565 possible sub-streams for the only sub-system `ATC Service`, but as the other sub-systems might change their states in parallel, the number

of possible executions is much bigger. Using our method to define chronicles as constraints, we have to separate the possible sub-streams into different NHMs. Regarding this constraint, our algorithm splits them into ten NHMs.

No data-stream is used since the constraints it provides has no impact on the sampling time and analysis time. The number of constraints only affects the computation of the NHM, but it will be showed later that it is quite negligible regarding the other parts of the computation.

For the computation, the number of samples is set to 10^4, which gives a precision of 10^{-2}. The query used is the following chronicle:

```
( ( (RPS_TC_Long_Recovery_Procedure
    (ATC_Rerouting_Mode_Not_Selected ||
    ATC_Rerouting_Mode_Inferred ||
    ATC_Rerouting_Mode_Confirmed)
    ATC_Nominal_For_RPS)
  -[ATC_Nominal_For_RPS]) then 10)
-[ATC_Nominal_Service]
```

meaning that the RPS passed into a long recovery procedure and went back to the nominal state while the ATC believes at a moment that the emergency situation was active and never went back to nominal ten units of time after the RPS went back to nominal. This behaviour is potentially dangerous as the two agents had different beliefs on the current situation with no communication planned.

Table 6.3 presents the computation time depending on the length of samples. Times for samples generations and parsing with CRL are separated. Last column displays the results for the model checking approach presented in Section 6.3.2. With MCMC

Stream length	Sampling (s)	Analysis (s)	Full time (s)	Model checking (s)
50	56	81	137	2.4
100	115	121	236	4.8
200	233	274	507	10
500	575	1690	2265	25
1000	1075	10300	11375	52
2000	2327	100075	102402	102

Table 6.3: Computation time regarding the length of samples for the drone model (for 10^4 samples).

approach, most of the computation time comes from the analysis, but it could be

easily reduced using a validity window with CRL that would reduce drastically the computation time on long data stream. Nonetheless, results shows the model checking approach to be much faster than the sampling technique.

NHM computation time and space The time computation presented in Table 6.3 does not include the time used to calculate the NHM. The computation time for the NHM depends on two parameters: the number of states of the system and the time of the latest constraint used to build the NHM. Dependence from the first one is obvious as it directly impacts the size of the transition matrix. For instance, with the drone model, the dimension of the transition matrix is 3360×3360 so the NHM is composed of a set of matrices with the same dimension. But, in fact, the most important parameter on computation time is the instant of the latest constraint. Indeed, it is not needed to compute more transition matrices after this point as the model just requires the initial transition matrix M. For instance, suppose a stream length of 1000 but no constraint is defined after time 500, the system will change as defined by the homogeneous Markov chain base on M. So no more matrix needs to be calculated between times 501 and 1000. Consequently, the computation time is linearly dependent on the time of the last constraint.

In fact, computation time is less restrictive than the space needed to store the NHM. Supposing that calculation uses double-precision floats encoded in 64 bits, one matrix of the model would use around 86 Mio of space. Storing a NHM composed of 100 matrices will use more than 8 Gio on memory space, so it might be difficult to constraint a full data stream compound of thousands events. Hopefully, it is possible to drastically reduce the storage space using interlaced computation of the NHM and the samples. Technically, NHM is computed a first time but just some matrices at a given interval are stored. When sampling is performed, matrices between two stored ones are re-calculated just for the sampling process. For instance, if the available memory space only allows storing 100 matrices, it is still possible to perform our method with a NHM of 2500 matrices where 50 are used as memory steps of the NHM computation and the 50 others are used as buffer for re-calculated matrices during the sampling step. This comes with a cost on time computation up to a factor 2, but it remains a quite interesting compromise.

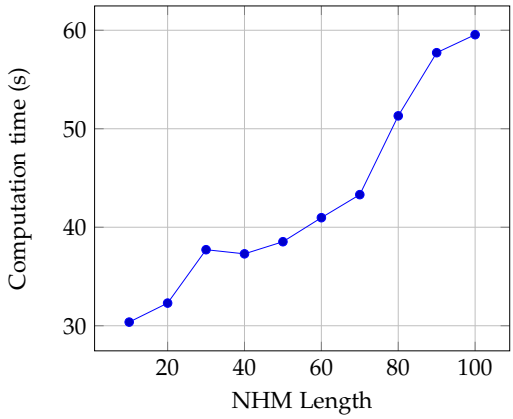


Figure 6.17: Computation time for the NHM construction regarding its length

6.5 Conclusion

Overview In this chapter we presented a new approach to structure and compute uncertainty for a CEP perspective and more precisely the data uncertainty. The objectives were to compute the probability of a chronicle defined as a query on an uncertain data-stream. Furthermore, we wanted to be able to specify complementary knowledge provided as a chronicle observed in an interval of time.

Our initial approach separates clearly the uncertainty representation from the CE recognition. The uncertainty is modelled using a Markov chain which might be restrictive, but at least permits representing complex conditional dependencies explaining an observed data-stream, while many works still consider independence between events. The data-stream is used as constraint on the Markov chain to define a NHM. The resulting NHM embedded all the possible executions of the system with regard to the initial Markov chain. The NHM is used to generate samples of explanatory executions which may be parsed using a CE recognition system to identify and count the occurrence of the CE defined as query. Note that, since the CE recognition system is completely disconnected from the uncertainty representation, it does not require to be specifically the chronicle representation and might be any other CE representation.

In a second time, we showed it was possible to specify high-level constraints on inference using chronicle specification to define the constraints. Again, the CE representation is not restricted to chronicles and might be any system that might be solved by a CSP solver regarding a specified interval of time.

We presented different methods to fasten inference by improving representation of

high-level constraints into sub-graphs or using model-checking approach to represent chronicles as a property to assert which avoids sampling. Finally, we showed our general approach to be applied on realistic scenario with a system dimension, a time of inference and an expressiveness on observations that, to our knowledge, do not appear, all together, in the current state of the art.

Discussion Before closing this chapter, we want to address some critics on our approach or point out some limitations.

The first limitation above all might obviously be the time representation. In the systems we presented, time is discretised and, more importantly, bounded on fix interval. Meaning that between two instants the duration is always the same. Obviously, it is an important restrictions since, for many systems, it would be preferable to consider time continuous or, at least, discretised with variable durations between two instants. Nonetheless, it does not seem impossible to adapt, but requires to design the NHM using a representation of time impacting the transition probabilities. This would be an interesting subject to investigate.

The MCMC approach is a simple and efficient way to produce consistent estimation of probability, but it remains too slow for online inference and, in particular, when applied on long data stream[7]. Nonetheless, we showed that most of the time computation was used by the CE recognition system and recent investigations on CRL let us think that this time might be reduced drastically. Furthermore, the model-checking approach, even if it brings restriction on the type of query used for inference, reduces drastically inference time in proportion that would make it viable for online analysis. In this perspective, it would be necessary to reduce the time necessary to compute the NHM. Even if the times presented in Figure 6.17 are high, we think that it might be due to a poor implementation, especially on constraint specification and might be optimised, even it is strongly dependent to the size of the NHM.

Indeed, in our approach, the Markov chain is always defined by composition of states of sub-systems so it is easy to define and assert independence of streams necessary for computation, but it leads to potentially huge Markov chains. We did not investigate this subject, but it is primordial for speed and space efficiency.

A final interesting point that would require investigation is the automaton division for representing and inferring with chronicles set as constraints. Indeed, performances of this algorithm are unknown (in number of resulting automata), but it may suffer

[7]For instance, with the drone model, it takes more than one day to process a stream of 2000 events.

from combinatorial explosion on specific cases. For instance, the acyclic DFA in Figure 6.15 produced a number of divisions which is definitely not linear with the size of the acyclic DFA. In this particular case, it is due to the fact that between two layers of nodes between d and $d+1$, it always exists two incompatible edges that will induce a division. At the end the number of divisions would be quadratic in the size of the automaton. It seems there is no obvious solution to counteract this effect, but it may be possible to use the constraints provided by the data-stream to reduce the number of incompatibles edges.

Conclusion

IN this thesis, we explored different solutions to perform CEP under uncertainty. More precisely, we focused on the evaluation of a CE on a data-stream when it might be erroneous. In particular, we were interested on models that allow representation and recognition of CE under uncertainty regarding three different properties. First property was the scalability of the method used for inference to the dimension of the problem since many techniques provided in the literature have an exponential growth calculation in the size of the problem. Second property was the precision of the probability inference which is usually dependent of the approach used to infer the CE probability. For instance, MCMC techniques would always provide estimations while other methods may provide exact probability with respect to the model. Finally, the last property focus on expressing prior knowledge more complex than a simple event triggering at a specific time. This knowledge would usually be a specific behaviour observed at a certain time and duration, but where no information is provided on time recognition of the sub-activities.

The first solution explored used MLNs to represent chronicles. The representation of Markov logic is principally derived from FOL, inference algorithm uses a slice sampling approach based on WalkSAT. We have shown that the logical structure of the chronicle tends to reduce drastically the algorithm efficiency due to unbalanced rate of solutions explored by WalkSAT. Consequently, the sampling approach produces inconsistent results. Furthermore, we showed that, contrary to the common belief, problem design with Markov logic might be particularly difficult due to the uncertainty representation with weights and due to misconceptions between conditional dependency and logical implication. This work led to a publication in IEA/AIE [RKL17].

In a second time, a representation of chronicles with ProbLog was considered. Its rule-based system suits well for a chronicle representation and inference, contrary

to MLNs, is based on exploratory techniques based on BDD representation[8]. Such technique allows ProbLog to compute exact probability of a CE with respect to the model. But, consequently, it comes with non-scalability to high-dimensions due to the NP-complete problem, making it not suitable for our work, even if it gives good results on small instances and easily allows expressing rules as constraints.

In the last chapter, we presented a new approach based on MCMC, but, contrary to the previous approaches, separates clearly uncertainty representation and CE recognition, which allows expressing query as chronicles, but more generally any type of CE. The method is able to produce the probability regarding the number of possible occurrences of a chronicle on a given data-stream. Furthermore, we showed that using CSP, or if the CE representation might be represented into an automaton, it was possible to express prior knowledge with observed behaviours. This work led to a publication in RuleML [RKL18]. In addition, we proposed, under small query restriction, a faster inference method, close to model-checking techniques, that might be applied for computing probability online.

Directions for future research

This work opens numerous questions and perspectives for future research.

The MLNs approach, for instance, despite it was not efficient for chronicle representation, rise questions on the impact of a structure on SAT solving for local search algorithm, and, in particular, for sampling on FOL problems. SAT-sample is an interesting approach for such purposes, but is still based on one of the oldest algorithm for local search and it may take benefit from more recent works. Recent works exist on new algorithm for MLNs, but the search space problem remains an open questions relative to complex properties such as the connectivity of a search space or the density of solutions [Bib09; HS05].

As discussed in Chapter 6, many questions remains for our Markov chain approach and its extensions. Notably, it involves specific restrictions in time representation such as discretised instant and fixed duration between to instant that is not realistic for a large amount of application. It would be interesting to find an approach that alleviate these restrictions. Furthermore, the Markov chain approach requires to express complex prior information represented by an automaton, or find all the streams for an interval consistent with the information. Automaton representation is restrictive for expressiveness or complex to produce and listing all streams is quickly intractable.

[8] Or similar representation like d-DNNFs.

Investigating new methods for representation and computation of complex prior information seems necessary. About chronicle representation, it might be necessary to prove the translation of every chronicle into automaton[9]. And further works should be initiated to avoid combination of states for the creation of the Markov model M.

On general perspectives for CEP under uncertainty, integrating attributes into CEP models for uncertainty, even without uncertainty considerations on attributes, remains an open and complex question where current propositions bring important restrictions and assumptions. But, it is obvious that such problematic will become increasingly urgent to solve. Representation of uncertainty, and in particular differentiation of data and pattern uncertainty, is an important candidate for further investigations and studies. It was already discussed in Chapter 2, but this subject leads to misconceptions and confusions between data and patterns uncertainty. For instance, it seems particularly curious that, despite the amount of research on pattern uncertainty, so few works are interested into structure uncertainty and more precisely on the ability that a representation has to embed uncertainty. Another aspect of interest would be the time representation for uncertainty. In almost every work, time has a low impact on uncertainty. For instance, automaton-based models usually does not consider that the delay between two recognitions may have consequences for probability estimations (except [Mol+14]). Same observations might be made for Markov model or Bayesian approaches, like our approach, that usually assume fix durations or time independence.

[9]Note this work exists for representation into Petri nets in [Pie14]

APPENDIX A

Definition of chronicle with the Markov logic semantic

Sequence

$$Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge t_2 < t_3 \iff OpSeq(c_1, c_2, t_1, t_2, t_3, t_4) \tag{A.1}$$

Absence

$$OpAbs(c_1, c_2, t_1, t_2, t_3, t_4) \iff Ch(c_1, t_1, t_4) \wedge \neg OpDur(c_1, c_2, t_1, t_2, t_3, t_4) \tag{A.2}$$

And

$$\begin{aligned}
OpConj_1(c_1, c_2, \underline{t_1}, t_3, t_4, \underline{t_2}) &\iff Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge (t_1 \leq t_3) \wedge (t_4 \leq t_2) \\
OpConj_2(c_1, c_2, \underline{t_3}, t_1, t_4, \underline{t_2}) &\iff Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge (t_3 < t_1) \wedge (t_4 \leq t_2) \\
OpConj_3(c_1, c_2, \underline{t_1}, t_3, t_2, \underline{t_4}) &\iff Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge (t_1 \leq t_3) \wedge (t_2 < t_4) \\
OpConj_4(c_1, c_2, \underline{t_3}, t_1, t_2, \underline{t_4}) &\iff Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge (t_3 < t_1) \wedge (t_2 < t_4)
\end{aligned} \tag{A.3}$$

$$\begin{aligned}
OpConj_1(c_1, c_2, t_1, t_3, t_4, t_2) \vee \cdots \vee \; & OpConj_4(c_1, c_2, t_1, t_3, t_4, t_2) \\
\implies \; & OpConj(c_1, c_2, t_1, t_2)
\end{aligned} \tag{A.4}$$

$$\begin{aligned}
\exists t_3, t_4 \quad & OpConj(c_1, c_2, t_1, t_2) \implies \\
& OpConj_1(c_1, c_2, t_1, t_3, t_4, t_2) \vee \cdots \vee \; OpConj_4(c_1, c_2, t_1, t_3, t_4, t_2)
\end{aligned} \tag{A.5}$$

A.1 Binary operators

During

$$Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge t_1 < t_3 \wedge t_2 < t_4 \iff OpDur(c_1, c_2, t_1, t_2, t_3, t_4) \quad \text{(A.6)}$$

Equals

$$Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_1, t_2) \iff OpEq(c_1, c_2, t_1, t_2) \quad \text{(A.7)}$$

Finishes

$$Ch(c_1, t_1, t_3) \wedge Ch(c_2, t_2, t_3) \wedge t_2 < t_1 \iff OpFini(c_1, c_2, t_1, t_2, t_3) \quad \text{(A.8)}$$

Meets

$$Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_2, t_3) \wedge t_2 < t_1 \iff OpMeets(c_1, c_2, t_1, t_2, t_3) \quad \text{(A.9)}$$

Or

$$Ch(c_1, t_1, t_2) \vee Ch(c_2, t_1, t_2) \iff OpOr(c_1, c_2, t_1, t_2) \quad \text{(A.10)}$$

Overlaps

$$Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_3, t_4) \wedge t_1 < t_3 \wedge t_3 < t_2 \wedge t_2 < t_4 \iff OpOverl(c_1, c_2, t_1, t_2, t_3, t_4) \quad \text{(A.11)}$$

Starts

$$Ch(c_1, t_1, t_2) \wedge Ch(c_2, t_1, t_3) \wedge t_2 < t_3 \iff OpStarts(c_1, c_2, t_1, t_2, t_3, t_4) \quad \text{(A.12)}$$

A.2 Unary operators

At most

$$Ch(c, t_1, t_2) \wedge (t_1 + \delta < t_2) \iff OpAtMost(c, t_1, t_2, \delta) \quad \text{(A.13)}$$

At least

$$Ch(c, t_1, t_2) \wedge (t_1 + \delta > t_2) \iff OpAtLeast(c, t_1, t_2, \delta) \quad \text{(A.14)}$$

Lasts

$$Ch(c, t_1, t_2) \wedge (t_1 + \delta = t_2) \iff OpLasts(c, t_1, t_2, \delta) \tag{A.15}$$

Then

$$Ch(c, t_1, t_2) \wedge (t_2 + \delta = t_3) \implies OpThen(c, t_2, t_3, \delta) \tag{A.16}$$

$$\exists t_1 OpThen(c, t_2, t_3, \delta) \implies Ch(c, t_1, t_2) \wedge (t_2 + \delta = t_3) \tag{A.17}$$

APPENDIX B

Drone model in ProbLog

The drone model program might be separated in three distinct parts:

- The temporal part describes the time mechanism that allows a system to change its state from an instant to another.
- The subsystems part describes the rules of interactions between systems and subsystems as described in Figure 1.5.
- The uncertainty part provides the probabilities associated to a change of state in the system.

For convenience, we will present only a small part of the program that provide a good insight of each different part. Consequently, rules presented for the subsystem and the uncertainty part focus only on the RPS system representation.

B.1 Temporal part

```
%%-------------------------------------------------------------------------------

maxTime(11).
sup(T,T2):- maxTime(T3),between(0,T3,T),T2 is T+1.
sup(T1,T2) :- sup(T1,T3),sup(T3,T2).
supeq(T1,T2) :- sup(T1,T2).
supeq(T,T).
oneDist(T,T2):- maxTime(T3),between(0,T3,T),T2 is T+1.
```

```
%%%%%%%%%%%%
% Chronicles operations
%%%%%%%%%%%%
order(T1,T2,T3,T4) :- T1 =< T2, T3 =< T4.
seq(T1,T2,T3,T4) :- order(T1,T2,T3,T4), T2 < T3.
then(T1,T2,T3) :- T2 is T1 + T3.
during(T1,T2,T3,T4) :- sup(T1,T2), sup(T3, T4).

%%%%%%%%
%State definition
%%%%%%%%
state(M,0,S) :- start(M,S).
state(M,T,S) :- oneDist(TT,T), 0 < T,trans(M,TT,S2,S),state(M,TT,S2), S\=S2.
state(M,T,S) :- oneDist(TT,T), 0 < T, state(M,TT,S), \+ trans(M,TT,S,S2).


%%%%%%%%
%Duration State
%Compute the duration a state stay active
%%%%%%%%
durationState(M,0,S,0) :- start(M,S).
durationState(M,T,S,0):-oneDist(TT,T), 0 < T ,state(M,T,S),  state(M,TT,SS).
durationState(M,T,S,D):-oneDist(TT,T), 0 < T ,state(M,T,S),  state(M,TT,S),
                        oneDist(DD,D), 0 < D,durationState(M,TT,S,DD).

%%%%%%%%
%% Transitions
%%%%%%%%
acTrans(M,T1,Sfrom,A) :- state(M,T1,Sfrom), action(M,A,T1),
                         transitionChance(M,T1,Sfrom,A).
timeNoChange(M,T1,Inter,Sfrom):-durationState(M,T1,S,D), supeq(Inter,D).
timeNoChangeNoAc(M,T1,Inter,Sfrom,A):-durationState(M,T1,S,D), supeq(Inter,D),
                                      \+acTrans(M,T1,Sfrom,A).
```

B.2 Subsystems part: Exemple from the RPS system

```
%%-------------------------------------------------------------------------
%%%%%%
%%RPS TC
```

```
%to nominal
trans(rps_tc,T1,quick_recovery_procedure,nominal) :-
                    acTrans(rps_tc,T1,quick_recovery_procedure,ac_to_nominal).
trans(rps_tc,T1,long_recovery_procedure,nominal) :-
                    acTrans(rps_tc,T1,long_recovery_procedure,ac_to_nominal).

% to quick
trans(rps_tc,T1,nominal,quick_recovery_procedure) :-
                    acTrans(rps_tc,T1,nominal,ac_to_quick).

%to long
trans(rps_tc,T1,quick_recovery_procedure,long_recovery_procedure) :-
      timeNoChangeNoAc(rps_tc,T1,2,quick_recovery_procedure,ac_to_nominal).

%lost
rps_tc_lost(T1) :-state(rps_tc,T1,quick_recovery_procedure).
rps_tc_lost(T1) :-state(rps_tc,T1,long_recovery_procedure).

%% Emited actions
action(atc_service,ac_to_rerouting,T1) :-oneDist(T2,T1),
                                    action(rps_tc,ac_to_rerouting,T2),
                                    state(rps_tc,T2,long_recovery_procedure),
                                    state(rps_voice,T2,nominal).


%%%%%%
%% RPS-voice


%lost
rps_voice_lost(T1) :-state(rps_voice,T1,recovery_procedure).
rps_voice_lost(T1) :-state(rps_voice,T1,unrecovered).

%to nominal
trans(rps_voice,T1,recovery_procedure,nominal) :-
                acTrans(rps_voice,T1,recovery_procedure,ac_to_nominal).
trans(rps_voice,T1,unrecovered,nominal) :-
                acTrans(rps_voice,T1,unrecovered,ac_to_nominal).

%to recovery
```

```
trans(rps_voice,T1,nominal,recovery_procedure) :-
                acTrans(rps_voice,T1,nominal,ac_to_recovery).

%to unrecoverd
trans(rps_voice,T1,recovery_procedure,unrecovered) :-
      timeNoChangeNoAc(rps_voice,T1,2,recovery_procedure,ac_to_nominal).

%% Emited actions
action(ua_code,ac_to_code_nominal,T1) :-oneDist(T2,T1),rps_voice_lost(T2),
                                        state(rps_tc, T1,nominal),
                                        state(rps_voice,T1,nominal).
action(ua,ac_code_7600,T1) :-oneDist(T2,T1),
                             state(rps_voice, T2,recovery_procedure),
                             state(rps_voice, T1,unrecovered),
                             state(rps_tc, T1,nominal).
action(ua,ac_code_7600,T1) :-oneDist(T2,T1),state(rps_voice, T2,unrecovered),
                             state(rps_voice, T1,unrecovered),
                             state(rps_tc, T1,nominal),
                             state(ua_code, T2,code_zz00).


%%%%%%
%% RPS-ATC


%to nominal
trans(rps_atc,T1,need_contact_urgency,nominal) :- state(rps_tc,T1,nominal),
                                    state(rps_atc,T1,need_contact_urgency).
trans(rps_atc,T1,checking_tc,nominal) :- rps_tc_lost(T1),
                                         timeNoChange(rps_atc,T1,2,checking_tc).
trans(rps_atc,T1,need_contact_invalidation,nominal) :-
      state(rps_voice,T1,nominal),
      state(rps_atc,T1,need_contact_invalidation).
trans(rps_atc,T1,need_contact_end,nominal) :-state(rps_voice,T1,nominal),
                                            state(rps_atc,T1,need_contact_end).

%to need contact urgency
trans(rps_atc,T1,nominal,need_contact_urgency) :-
      state(rps_tc,T1,long_recovery_procedure), state(rps_atc,T1,nominal).

%to contacted urgency
```

```
trans(rps_atc,T1,need_contact_urgency,contacted_urgency) :-
      state(rps_voice,T1,nominal),
      state(rps_atc,T1,need_contact_urgency).

%to checking TC
trans(rps_atc,T1,nominal,checking_tc) :- state(rps_voice,T1,nominal),
                        acTrans(rps_atc,T1,nominal,ac_to_checking_tc).

%to need contact invalidation
trans(rps_atc,T1,checking_tc,need_contact_invalidation) :-
         state(rps_tc,T1,nominal), timeNoChange(rps_atc,T1,5,checking_tc).

%to need contact end
trans(rps_atc,T1,contacted_urgency,need_contact_end) :-
              state(rps_tc,T1,nominal),state(rps_atc,T1,contacted_urgency).

%% Emited actions
action(atc_service,ac_to_urgency,T1) :-oneDist(T2,T1),
                                       state(rps_atc,T2,need_contact_urgency),
                                       state(rps_atc,T1,contacted_urgency).
action(atc_service,ac_to_nominal,T1) :-oneDist(T2,T1),
                                       state(rps_atc,T2,need_contact_end),
                                       state(rps_atc,T1,nominal).
action(atc_service,ac_to_nominal,T1) :-oneDist(T2,T1),
                                       state(rps_atc,T2,need_contact_invalidation),
                                       state(rps_atc,T1,nominal).
action(ua_pilot,ac_to_nominal,T1) :-oneDist(T2,T1),
                                    state(rps_atc,T2,need_contact_urgency),
                                    state(rps_atc,T1,nominal).


%%%%%%%%%
% Chronicles
%%%%%%%%%

%RPS_TC
ch(from_rps_tc_nominal,TT,TT) :- state(rps_tc,TT,S), TT > 0,T is TT - 1,
                                 state(rps_tc,T,nominal), S \= nominal.
ch(to_rps_tc_nominal,TT,TT) :- state(rps_tc,TT,nominal), TT > 0,T is TT - 1,
                               \+state(rps_tc,T,nominal).
ch(from_rps_tc_quick_recovery_procedure,TT,TT) :-state(rps_tc,TT,S), TT > 0,
```

```
                                                T is TT - 1,
                                                state(rps_tc,T,quick_recovery),
                                                S \= quick_recovery.
ch(to_rps_tc_quick_recovery_procedure,TT,TT) :-
                                        state(rps_tc,TT,quick_recovery_procedure),
                                        T is TT - 1,
                                        \+state(rps_tc,T,quick_recovery_procedure).
ch(from_rps_tc_long_recovery_procedure,TT,TT) :-
                state(rps_tc,TT,S),TT > 0,T is TT - 1,
                S \= long_recovery_procedure.
ch(to_rps_tc_long_recovery_procedure,TT,TT) :-
                   state(rps_tc,TT,long_recovery_procedure),
                   TT > 0,T is TT - 1,
                   \+state(rps_tc,T,long_recovery_procedure).


% %RPS_ATC
ch(from_rps_atc_nominal_for_rps,T,T) :- state(rps_atc,TT,S), TT > 0,T is TT - 1,
                                        state(rps_atc,T,nominal), S \= nominal.
ch(to_rps_atc_nominal_for_rps,T,T) :- state(rps_atc,TT,nominal), TT > 0,
                                        T is TT - 1, \+state(rps_atc,T,nominal).
ch(from_rps_need_contact_atc_urgency,T,T) :-
                           state(rps_atc,TT,S), TT > 0,
                           T is TT - 1,
                           state(rps_atc,T,need_contact_urgency),
                           S \= need_contact_urgency.

ch(to_rps_need_contact_atc_urgency,T,T):-state(rps_atc,TT,need_contact_urgency),
                                         TT > 0,
                                         T is TT - 1,
                                        \+state(rps_atc,T,need_contact_urgency).
ch(from_rps_checking_tc,T,T) :- state(rps_atc,TT,S), TT > 0,T is TT - 1,
                                state(rps_atc,T,checking_tc), S \= checking_tc.
ch(to_rps_checking_tc,T,T) :- state(rps_atc,TT,checking_tc), TT > 0,T is TT - 1,
                              \+state(rps_atc,T,checking_tc).
ch(from_rps_atc_concacted_urgency,T,T) :- state(rps_atc,TT,S), TT > 0,
                                          T is TT - 1,
                                          state(rps_atc,T,concacted_urgency),
                                          S \= concacted_urgency.
ch(to_rps_atc_concacted_urgency,T,T) :- state(rps_atc,TT,concacted_urgency),
                                        TT > 0,T is TT - 1,
```

```
                                        \+state(rps_atc,T,concacted_urgency).
ch(from_rps_need_contact_atc_invalidation_urgency,T,T) :-
            state(rps_atc,TT,S),TT > 0,T is TT - 1,
            state(rps_atc,T,need_contact_invalidation),
            S \= need_contact_invalidation.
ch(to_rps_need_contact_atc_invalidation_urgency,T,T) :-
            state(rps_atc,TT,need_contact_invalidation), TT > 0,T is TT - 1,
            \+state(rps_atc,T,need_contact_invalidation).
ch(from_rps_need_contact_atc_end_urgency,T,T) :-
            state(rps_atc,TT,S), TT > 0,
            T is TT - 1,
            state(rps_atc,T,need_contact_end), S \= need_contact_end.
ch(to_rps_need_contact_atc_end_urgency,T,T):-state(rps_atc,TT,need_contact_end),
                                              TT > 0,T is TT - 1,
                                             \+state(rps_atc,T,need_contact_end).

% %RPS_VOICE
ch(from_rps_nominal_voice,T,T) :- state(rps_voice,TT,S), TT > 0,T is TT - 1,
                                  state(rps_voice,T,nominal), S \= nominal.
ch(to_rps_nominal_voice,T,T) :- state(rps_voice,TT,nominal), TT > 0,T is TT - 1,
                                \+state(rps_voice,T,nominal).
ch(from_rps_voice_recovery_procedure,T,T) :-
          state(rps_voice,TT,S),TT > 0,T is TT - 1,
          state(rps_voice,T,recovery_procedure), S \= recovery_procedure.
ch(to_rps_voice_recovery_procedure,T,T) :-
          state(rps_voice,TT,recovery_procedure), TT > 0,T is TT - 1,
          \+state(rps_voice,T,recovery_procedure).
ch(from_rps_voice_unrecovered,T,T) :- state(rps_voice,TT,S), TT > 0,T is TT - 1,
                                      state(rps_voice,T,unrecovered),
                                      S \= unrecovered.
ch(to_rps_voice_unrecovered,T,T) :- state(rps_voice,TT,unrecovered),
                                    TT > 0,T is TT - 1,
                                    \+state(rps_voice,T,unrecovered).


%%%%%%%%%
% Proba Actions
%%%%%%%%%


% RPS_TC
```

```
0.9::rps_tc_loss_detection(T):- maxTime(T2),between(0,T2,T).
action(rps_tc,ac_to_quick,T) :- action(tc_failure,lost,T),
                                state(rps_tc,T,nominal),
                                rps_tc_loss_detection(T).
action(rps_tc,ac_to_nominal,T) :-  action(tc_failure,back,T),
                                   state(rps_tc,T,quick_recovery_procedure),
                                   rps_tc_loss_detection(T).
action(rps_tc,ac_to_nominal,T) :-  action(tc_failure,back,T),
                                   state(rps_tc,T,long_recovery_procedure),
                                   rps_tc_loss_detection(T).

%RPS_VOICE
0.9::rps_voice_loss_detection(T):- maxTime(T2),between(0,T2,T).
action(rps_voice,ac_to_recovery,T) :- action(voice_failure,lost,T),
                                      state(rps_voice,T,nominal),
                                      rps_voice_loss_detection(T).
action(rps_voice,ac_to_nominal,T) :-  action(voice_failure,back,T),
                                      state(rps_voice,T,recovery_procedure),
                                      rps_voice_loss_detection(T).
action(rps_voice,ac_to_nominal,T) :-  action(voice_failure,back,T),
                                      state(rps_voice,T,unrecovered),
                                      rps_voice_loss_detection(T).

%%%%%%%%
% Start
%%%%%%%%

start(rps_tc,nominal).
start(rps_voice,nominal).
start(rps_atc,nominal).
```

B.3 Uncertainty part

```
%%-------------------------------------------------------------------------------
0.99::transitionChance(rps_tc,T1,quick_recovery_procedure,ac_to_nominal) :-
                       maxTime(T2),between(0,T2,T1).
0.99::transitionChance(rps_tc,T1,long_recovery_procedure,ac_to_nominal) :-
                       maxTime(T2),between(0,T2,T1).
0.99::transitionChance(rps_tc,T1,nominal,ac_to_quick) :-
```

```
                        maxTime(T2),between(0,T2,T1).
0.99::transitionChance(rps_voice,T1,recovery_procedure,ac_to_nominal) :-
                        maxTime(T2),between(0,T2,T1).
0.99::transitionChance(rps_voice,T1,unrecovered,ac_to_nominal) :-
                        maxTime(T2),between(0,T2,T1).
0.99::transitionChance(rps_voice,T1,nominal,ac_to_recovery) :-
                        maxTime(T2),between(0,T2,T1).
0.99::transitionChance(rps_atc,T1,nominal,ac_to_checking_tc) :-
                        maxTime(T2),between(0,T2,T1).

transitionChance(tc_failure,T1,nominal,lost) :-maxTime(T2),between(0,T2,T1).
transitionChance(tc_failure,T1,lost,back) :-maxTime(T2),between(0,T2,T1).
transitionChance(voice_failure,T1,nominal,lost) :-maxTime(T2),between(0,T2,T1).
transitionChance(voice_failure,T1,lost,back) :-maxTime(T2),between(0,T2,T1).
```

Bibliography

[Alb+07] Massimiliano Albanese, Vincenzo Moscato, Antonio Picariello, and V. S. Subrahmanian. “Detecting Stochastically Scheduled Activities in Video”. In: *IJCAI* (2007), pp. 1802–1807 (cit. on pp. 60, 64, 67, 72).

[Alb+08] Massimiliano Albanese, Rama Chellappa, Vincenzo Moscato, Antonio Picariello, V. S. Subrahmanian, Pavan Turaga, and Octavian Udrea. “A constrained probabilistic petri net framework for human activity detection in video”. In: *IEEE Transactions on Multimedia* 10.6 (2008), pp. 982–996 (cit. on pp. 71–73).

[Alb+11] Massimiliano Albanese, Cristian Molinaro, Fabio Persia, Antonio Picariello, and V. S. Subrahmanian. “Finding "Unexplained" Activities in Video.” In: *IJCAI*. 2011, pp. 1628–1634 (cit. on pp. 64, 65, 72).

[Ale+17] Elias Alevizos, Anastasios Skarlatidis, Alexander Artikis, and Georgios Paliouras. “Probabilistic Complex Event Recognition: A Survey”. In: *ACM Computing Surveys* 50.5 (Sept. 2017), pp. 1–31 (cit. on p. 20).

[All83] James F. Allen. “Maintaining knowledge about temporal intervals”. In: *Communications of the ACM* 26.11 (1983), pp. 832–843 (cit. on pp. 31, 51).

[Art+10] Alexander Artikis, Georgios Paliouras, François Portet, and Anastasios Skarlatidis. “Logic-based representation, reasoning and machine learning for event recognition”. In: *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*. ACM, 2010, pp. 282–293 (cit. on p. 22).

[Art+12a] Alexander Artikis, Anastasios Skarlatidis, François Portet, and Georgios Paliouras. “Logic-based event recognition”. In: *The Knowledge Engineering Review* 27.04 (2012), pp. 469–506 (cit. on p. 22).

[Art+12b] Alexander Artikis, Anastasios Skarlatidis, François Portet, and Georgios Paliouras. "Logic-based event recognition". In: *The Knowledge Engineering Review* 27.04 (2012), pp. 469–506 (cit. on p. 23).

[ASP12] Alexander Artikis, Marek Sergot, and Georgios Paliouras. "Run-time composite event recognition". In: *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*. ACM, 2012, pp. 69–80 (cit. on p. 22).

[Bar+12] Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. "Markov constraints for generating lyrics with style". In: *Proceedings of the 20th European Conference on Artificial Intelligence*. IOS Press, 2012, pp. 115–120 (cit. on p. 134).

[BCC07] Olivier Bertrand, Patrice Carle, and Christine Choppy. "Chronicle modelling using automata and colored Petri nets". In: *The 18th International Workshop on Principles of Diagnosis (DX-07)*. 2007, pp. 229–234 (cit. on p. 30).

[BCC08] Olivier Bertrand, Patrice Carle, and Christine Choppy. "Towards a coloured Petri nets semantics of a chronicle language for distributed simulation processing". In: *CHINA 2008 (Concurrency metHods: Issues aNd Applications)*. Citeseer, 2008, pp. 105–119 (cit. on p. 30).

[Ber09] Olivier Bertrand. "Detection d'activites par un systeme de reconnaissance de chroniques et application au cas des simulations distribuees HLA". PhD thesis. Paris 13, 2009 (cit. on p. 30).

[Bib09] Marenglen Biba. "Integrating Logic and Probability: Algorithmic Improvements in Markov Logic Networks". PhD thesis. University of Bari, Italy, 2009 (cit. on p. 160).

[BTF07] Rahul Biswas, Sebastian Thrun, and Kikuo Fujimura. "Recognizing activities with multiple cues". In: *Human MotionUnderstanding, Modeling, Capture and Animation*. Springer, 2007, pp. 255–270 (cit. on pp. 54, 60).

[CBC09] Christine Choppy, Olivier Bertrand, and Patrice Carle. "Coloured petri nets for chronicle recognition". In: *Reliable Software TechnologiesAda-Europe 2009*. Springer, 2009, pp. 266–281 (cit. on p. 30).

[CCK11] Patrice Carle, Christine Choppy, and Romain Kervarc. "Behaviour Recognition Using Chronicles". In: *5th International Symposium on Theoretical Aspects of Software Engineering (TASE)*. IEEE, Aug. 2011, pp. 100–107 (cit. on p. 30).

[CD97] Luca Chittaro and Michel Dojat. "Using a general theory of time and change in patient monitoring: Experiment and evaluation". In: *Computers in Biology and Medicine* 27.5 (Sept. 1997), pp. 435–452 (cit. on p. 22).

[Chu+10] Xu Chuanfei, Lin Shukuan, Wang Lei, and Qiao Jianzhong. "Complex Event Detection in Probabilistic Stream". In: *12th International Asia-Pacific Web Conference (APWEB)*. IEEE, Apr. 2010, pp. 361–363 (cit. on p. 62).

[CLS15] Shaowei Cai, Chuan Luo, and Kaile Su. "Improving WalkSAT By Effective Tie-Breaking and Efficient Implementation". In: *The Computer Journal* 58.11 (Nov. 2015), pp. 2864–2875 (cit. on p. 111).

[CM10] Gianpaolo Cugola and Alessandro Margara. "TESLA: a formally defined event specification language". In: *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*. ACM, 2010, pp. 50–61 (cit. on pp. 27, 29, 56).

[CM12] Gianpaolo Cugola and Alessandro Margara. "Complex event processing with T-REX". In: *Journal of Systems and Software* 85.8 (2012), pp. 1709–1728 (cit. on pp. 29, 30, 45).

[CM96] L. Chittaro and A. Montanari. "Efficient temporal reasoning in the cached event calculus". In: *Computational Intelligence* 12.3 (Aug. 1996), pp. 359–382 (cit. on p. 22).

[Cug+15] Gianpaolo Cugola, Alessandro Margara, Matteo Matteucci, and Giordano Tamburrelli. "Introducing uncertainty in complex event processing: model, implementation, and validation". In: *Computing* 97.2 (Feb. 2015), pp. 103–144 (cit. on pp. 43, 45, 56, 76).

[Dar01] Adnan Darwiche. "Decomposable negation normal form". In: *Journal of the ACM (JACM)* 48.4 (2001), pp. 608–647 (cit. on pp. 51, 88).

[Dar04] Adnan Darwiche. "New advances in compiling CNF to decomposable negation normal form". In: *Proceedings of the 16th European Conference on Artificial Intelligence*. IOS Press, 2004, pp. 318–322 (cit. on pp. 51, 88).

[Dar11] Adnan Darwiche. "SDD: A new canonical representation of propositional knowledge bases". In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Vol. 22. 2011, p. 819 (cit. on pp. 51, 88).

[Dem+06] Alan Demers, Johannes Gehrke, Mingsheng Hong, Mirek Riedewald, and Walker White. "Towards Expressive Publish/Subscribe Systems". In: *Advances in Database Technology - EDBT 2006*. Ed. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Yannis Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Mike Hatzopoulos, Klemens Boehm, Alfons Kemper, Torsten Grust, and Christian Boehm. Vol. 3896. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 627–644 (cit. on pp. 70, 76).

[DIG07] Yanlei Diao, Neil Immerman, and Daniel Gyllstrom. "Sase+: An agile language for Kleene closure over event streams". In: *UMass Technical Report* (2007) (cit. on pp. 24, 27).

[DL09] Pedro Domingos and Daniel Lowd. "Markov Logic: An Interface Layer for Artificial Intelligence". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3.1 (2009), pp. 1–155 (cit. on pp. 50, 79, 80, 84, 86).

[Dri+15] Anton Dries, Angelika Kimmig, Wannes Meert, Joris Renkens, Guy Van den Broeck, Jonas Vlasselaer, and Luc De Raedt. "ProbLog2: Probabilistic logic programming". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 312–315 (cit. on pp. 51, 88).

[DW12] Pedro Domingos and William Austin Webb. "A Tractable First-Order Probabilistic Logic." In: *AAAI*. 2012 (cit. on p. 87).

[Ear70] Jay Earley. "An Efficient Context-Free Parsing Algorithm". In: *Communications of the ACM* (1970), p. 9 (cit. on p. 74).

[Far+05] Andrew D. H. Farrell, Marek J. Sergot, Mathias Sallé, and Claudio Bartolini. "Using the event calculus for tracking the normative state of contracts". In: *International Journal of Cooperative Information Systems* 14.02n03 (June 2005), pp. 99–129 (cit. on p. 22).

[Faz+18a] Bettina Fazzinga, Sergio Flesca, Filippo Furfaro, Elio Masciari, and Luigi Pontieri. "Efficiently interpreting traces of low level events in business process logs". In: *Information Systems* 73 (Mar. 2018), pp. 1–24 (cit. on pp. 66–68, 151).

[Faz+18b] Bettina Fazzinga, Sergio Flesca, Filippo Furfaro, and Luigi Pontieri. "Online and offline classification of traces of event logs on the basis of security risks". In: *Journal of Intelligent Information Systems* 50.1 (Feb. 2018), pp. 195–230 (cit. on pp. 66, 67).

[Fie+15] Daan Fierens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. "Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas". In: *Theory and Practice of Logic Programming* (2015), p. 54 (cit. on p. 88).

[GEE17] K.S. Gayathri, K.S. Easwarakumar, and Susan Elias. "Probabilistic ontology based activity recognition in smart homes using Markov Logic Network". In: *Knowledge-Based Systems* 121 (Apr. 2017), pp. 173–184 (cit. on p. 52).

[GMM90] Carlo Ghezzi, Dino Mandrioli, and Angelo Morzenti. "TRIO: A logic language for executable specifications of real-time systems". In: *Journal of Systems and software* 12.2 (1990), pp. 107–123 (cit. on p. 27).

[GY11] Marco Guidetti and A. P. Young. "Complexity of several constraint-satisfaction problems using the heuristic classical algorithm WalkSAT". In: *Physical Review E* 84.1 (July 2011) (cit. on p. 109).

[Gyl+06] Daniel Gyllstrom, Eugene Wu, Hee-Jin Chae, Yanlei Diao, Patrick Stahlberg, and Gordon Anderson. "SASE: Complex event processing over streams". In: *arXiv preprint cs/0612128* (2006) (cit. on pp. 23, 24).

[HM00] John E Hopcroft and Rajeev Motwani. *Introduction to Automata Theory, Languages and Computability*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000 (cit. on p. 24).

[HNS10] Rim Helaoui, Mathias Niepert, and Heiner Stuckenschmidt. "A statistical-relational activity recognition framework for ambient assisted living systems". In: *Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence (ISAmI 2010)*. Springer, 2010, pp. 247–254 (cit. on p. 54).

[HNS11] Rim Helaoui, Mathias Niepert, and Heiner Stuckenschmidt. "Recognizing interleaved and concurrent activities: A statistical-relational approach". In: *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–9 (cit. on p. 54).

[Hou12] 12th Congress House of Representatives. *FAA reauthorization and reform act of 2011*. Feb. 2012 (cit. on p. 34).

[HS05] Holger H. Hoos and Thomas G. Stützle. *Stochastic local search Foundation and application*. Vol. 156. Springer, 2005 (cit. on pp. 107, 108, 160).

[Hur+16] Barry Hurley, Barry OSullivan, David Allouche, George Katsirelos, Thomas Schiex, Matthias Zytnicki, and Simon de Givry. "Multi-language evaluation of exact solvers in graphical model discrete optimization". In: *Constraints* 21.3 (2016), pp. 413–434 (cit. on p. 83).

[IB00] Yuri A. Ivanov and Aaron F. Bobick. "Recognition of visual activities and interactions by stochastic parsing". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 852–872 (cit. on p. 75).

[Jai11] Dominik Jain. "Knowledge engineering with markov logic networks: A review". In: *Evolving Knowledge in Theory and Applications* 16 (2011) (cit. on pp. 50, 75, 117).

[KCC10] Romain Kervarc, Christine Choppy, and Patrice Carle. "Chronicles: a temporal logic framework for the study of large simulations". In: *Proc. of the 10th Meetings on Applied Scientific Computing and Tools*. 2010 (cit. on pp. 148, 149).

[Kim+11] Angelika Kimmig, Bart Demoen, Luc De Raedt, Vitor Santos Costa, and Ricardo Rocha. "On the implementation of the probabilistic logic programming language ProbLog". In: *Theory and Practice of Logic Programming* 11.2-3 (2011), pp. 235–262 (cit. on p. 68).

[Kin80] Ross Kindermann. "Markov random fields and their applications". In: *American mathematical society* (1980) (cit. on p. 78).

[KKL10] H. Kawashima, H. Kitagawa, and Xin Li. "Complex Event Processing over Uncertain Data Streams". In: *2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*. Nov. 2010, pp. 521–526 (cit. on p. 62).

[KS89] Robert Kowalski and Marek Sergot. "A logic-based calculus of events". In: *Foundations of knowledge base management*. Springer, 1989, pp. 23–55 (cit. on pp. 21, 22).

[LDL17] Fagui Liu, Dacheng Deng, and Ping Li. "Dynamic Context-Aware Event Recognition Based on Markov Logic Networks". In: *Sensors* 17.3 (Mar. 2017), p. 491 (cit. on p. 54).

[LRR13] Gal Lavee, Michael Rudzsky, and Ehud Rivlin. "Propagating Certainty in Petri Nets for Activity Recognition". In: *IEEE Transactions on Circuits and Systems for Video Technology* 23.2 (Feb. 2013), pp. 326–337 (cit. on p. 72).

[Man09] Cristina Manfredotti. "Modeling and Inference with Relational Dynamic Bayesian Networks". In: *Advances in Artificial Intelligence*. Ed. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Yong Gao, and Nathalie Japkowicz. Vol. 5549. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 287–290 (cit. on p. 59).

[MD09] Radu Marinescu and Rina Dechter. "AND/OR Branch-and-Bound search for combinatorial optimization in graphical models". In: *Artificial Intelligence* 173.16-17 (Nov. 2009), pp. 1457–1491 (cit. on p. 51).

[MD11] Vlad Morariu and Larry S. Davis. "Multi-agent event recognition in structured scenarios". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3289–3296 (cit. on pp. 47, 51, 55, 69, 120).

[ME02] Darnell Moore and Irfan Essa. "Recognizing Multitasked Activities from Video Using Stochastic Context-Free Grammar". In: 2002, pp. 770–776 (cit. on p. 75).

[MES03] D. Minnen, I. Essa, and T. Starner. "Expectation grammars: leveraging high-level expectations for activity recognition". In: vol. 2. IEEE Comput. Soc, 2003, pp. II–626–II–632 (cit. on p. 75).

[Met+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6 (June 1953), pp. 1087–1092 (cit. on p. 86).

[MHZ10] Cristina Manfredotti, Howard Hamilton, and Sandra Zilles. "Learning RDBNs for Activity Recognition". In: *NIPS Workshop on Learning and Planning from Batch Time Series Data*. 2010 (cit. on p. 59).

[MM07] Justin Muncaster and Yunqian Ma. "Activity Recognition using Dynamic Bayesian Networks with Automatic State Selection". In: IEEE, Feb. 2007, pp. 30–30 (cit. on pp. 45, 58, 59, 72, 132).

[MMG92] Angelo Morzenti, Dino Mandrioli, and Carlo Ghezzi. "A model parametric real-time logic". In: *ACM Transactions on Programming Languages and Systems* 14.4 (Oct. 1992), pp. 521–573 (cit. on p. 27).

[Mol+14] Cristian Molinaro, Vincenzo Moscato, Antonio Picariello, Andrea Pugliese, Antonino Rullo, and V. S. Subrahmanian. "PADUA: Parallel Architecture to Detect Unexplained Activities". In: *ACM Transactions on Internet Technology* (2014), p. 23 (cit. on pp. 45, 48, 65, 66, 139, 161).

[MS02] Rob Miller and Murray Shanahan. "Some alternative formulations of the event calculus". In: *Computational logic: logic programming and beyond*. Springer, 2002, pp. 452–490 (cit. on p. 22).

[MS99] Rob Miller and Murray Shanahan. "The Event Calculus in Classical Logic-Alternative Axiomatisations". In: *Electronic Transactions on Artificial Intelligence*. Vol. 3. 1999, pp. 77–105 (cit. on p. 22).

[Nea03] Radford M Neal. "Slice sampling". In: *Annals of statistics* (2003), pp. 705–741 (cit. on pp. 84, 85).

[OC94a] Manuel Ornato and Patrice Carle. "Reconnaissance d'intentions sans reconnaissance de plan". In: *2es Journées Francophones dIntelligence Artificielle Distribuée et Systèmes Multi-Agents* (1994), p. 29 (cit. on pp. 29, 30).

[OC94b] Manuel Ornato and Patrice Carle. "Une alternative à labduction pour la reconnaissance de plan". In: *Secondes Rencontres des Jeunes Chercheurs en Intelligence Artificielle* (1994), p. 29 (cit. on p. 29).

[PD06] Hoifung Poon and Pedro Domingos. "Sound and efficient inference with probabilistic and deterministic dependencies". In: *AAAI*. Vol. 6. 2006, pp. 458–463 (cit. on pp. 51, 82, 84, 197).

[Pie14] Ariane Piel. "Reconnaissance de comportements complexes par traitement en ligne de flux devenements". PhD thesis. U. Paris 13, 2014 (cit. on pp. 14, 18, 30, 33, 35, 149, 161).

[Pnu77] Amir Pnueli. "The temporal logic of programs". In: IEEE, Sept. 1977, pp. 46–57 (cit. on p. 27).

[PRB11] François Pachet, Pierre Roy, and Gabriele Barbieri. "Finite-length Markov processes with constraints". In: *22nd International Joint Conference on Artificial Intelligence*. 2011 (cit. on pp. 78, 90, 134).

[Pre05] Steven Prestwich. "Random walk with continuously smoothed variable weights". In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 2005, pp. 203–215 (cit. on p. 107).

[RA06] M.S. Ryoo and J.K. Aggarwal. "Recognition of Composite Human Activities through Context-Free Grammar Based Representation". In: vol. 2. IEEE, 2006, pp. 1709–1718 (cit. on p. 75).

[RA09] M. S. Ryoo and J. K. Aggarwal. "Semantic Representation and Recognition of Continued and Recursive Human Activities". In: *International Journal of Computer Vision* 82.1 (Apr. 2009), pp. 1–24 (cit. on p. 75).

[RD06] Matthew Richardson and Pedro Domingos. "Markov logic networks". In: *Machine Learning* 62.1-2 (2006), pp. 107–136 (cit. on pp. 48, 78–81, 83).

[Ré+08] Christopher Ré, Julie Letchner, Magdalena Balazinksa, and Dan Suciu. "Event queries on correlated probabilistic streams". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 715–728 (cit. on pp. 69, 70, 139).

[Rie12] Sebastian Riedel. "Improving the Accuracy and Eciency of MAP Inference for Markov Logic". In: (2012) (cit. on p. 83).

[RKL17] Romain Rincé, Romain Kervarc, and Philippe Leray. "On the Use of WalkSAT Based Algorithms for MLN Inference in Some Realistic Applications". In: *30th International Conference on Industrial, Engineering, Other Applications of Applied Intelligent Systems*. Arras, 2017, pp. 121–131 (cit. on p. 159).

[RKL18] Romain Rincé, Romain Kervarc, and Philippe Leray. "Complex Event Processing Under Uncertainty Using Markov Chains, Constraints, and Sampling". In: *International Joint Conference on Rules and Reasoning*. Springer, Cham, 2018, pp. 147–163 (cit. on p. 160).

[RKT07] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. "ProbLog: A Probabilistic Prolog and its Application in Link Discovery". In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (2007) (cit. on pp. 51, 87, 89, 197).

[Sad12] Adam Sadilek. "Modeling human behavior at a large scale". PhD thesis. Rochester University, 2012 (cit. on p. 53).

[SAO05] Sakari Seitz, Mikko Alava, and Pekka Orponen. "Threshold Behaviour of WalkSAT and Focused Metropolis Search on Random 3-Satisfiability". In: *Theory and Applications of Satisfiability Testing*. Ed. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Fahiem Bacchus, and Toby Walsh. Vol. 3569. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 475–481 (cit. on p. 108).

[Sha99] Murray Shanahan. "The event calculus explained". In: *Artificial intelligence today*. Springer, 1999, pp. 409–430 (cit. on p. 22).

[SK10] Adam Sadilek and Henry A. Kautz. "Recognizing Multi-Agent Activities from GPS Data." In: *AAAI*. Vol. 39. 2010, p. 109 (cit. on p. 53).

[SK12] Adam Sadilek and Henry Kautz. "Location-based reasoning about complex multi-agent behavior". In: *Journal of Artificial Intelligence Research* (2012), pp. 87–133 (cit. on p. 53).

[Ska+11] Anastasios Skarlatidis, Georgios Paliouras, George A. Vouros, and Alexander Artikis. "Probabilistic event calculus based on markov logic networks". In: *Proceedings of Rule-Based Modeling and Computing on the Semantic Web*. Vol. 7018. LNCS. Springer, 2011, pp. 155–170 (cit. on pp. 47, 48, 52, 104, 105).

[Ska+15a] Anastasios Skarlatidis, Alexander Artikis, Jason Filippou, and Georgios Paliouras. "A probabilistic logic programming event calculus". In: *Theory and Practice of Logic Programming* 15.02 (2015), pp. 213–245 (cit. on p. 69).

[Ska+15b] Anastasios Skarlatidis, Georgios Paliouras, Alexander Artikis, and George A. Vouros. "Probabilistic event calculus for event recognition". In: *ACM Transactions on Computational Logic (TOCL)* 16.2 (2015), p. 11 (cit. on pp. 47, 50, 55, 75, 120).

[SKC+93] Bart Selman, Henry Kautz, Bram Cohen, et al. "Local search strategies for satisfiability testing". In: *DIMACS Series in Discrete Mathematics*. Vol. 26. 1993, pp. 521–532 (cit. on pp. 81, 86, 116).

[SKK08] Zhitao Shen, Hideyuki Kawashima, and Hiroyuki Kitagawa. "Probabilistic event stream processing with lineage". In: *Proc. of Data Engineering Workshop*. 2008 (cit. on pp. 61–64, 67, 151).

[Son+13] Young Chol Song, Henry Kautz, James Allen, Mary Swift, Yuncheng Li, Jiebo Luo, and Ce Zhang. "A Markov Logic Framework for Recognizing Complex Events from Multimodal Data". In: *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*. ICMI '13. New York, NY, USA: ACM, 2013, pp. 141–148 (cit. on p. 52).

[Sto95] Andreas Stolcke. "An efficient probabilistic context-free parsing algorithm that computes prefix probabilities". In: *Computational linguistics* 21.2 (1995), pp. 165–201 (cit. on p. 72).

[SVB15] Lauro Snidaro, Ingrid Visentini, and Karna Bryan. "Fusing uncertain knowledge and evidence for maritime situational awareness via Markov Logic Networks". In: *Information Fusion* 21 (2015), pp. 159–172 (cit. on p. 53).

[TD08] Son D. Tran and Larry S. Davis. "Event modeling and recognition using markov logic networks". In: *Computer VisionECCV 2008*. Springer, 2008, pp. 610–623 (cit. on pp. 54, 55).

[WA10] Chen Wu and Hamid Aghajan. "Recognizing objects in smart homes based on human interaction". In: *Advanced Concepts for Intelligent Vision Systems*. Springer, 2010, pp. 131–142 (cit. on p. 54).

[WA11] Chen Wu and Hamid Aghajan. "User-Centric Environment Discovery With Camera Networks in Smart Homes". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41.2 (Mar. 2011), pp. 375–383 (cit. on p. 54).

[Was+08] Segev Wasserkrug, Avigdor Gal, Opher Etzion, and Yulia Turchin. "Complex event processing over uncertain data". In: ACM Press, 2008, p. 253 (cit. on p. 56).

[Was+12] S. Wasserkrug, A. Gal, O. Etzion, and Y. Turchin. "Efficient Processing of Uncertain Events in Rule-Based Systems". In: *IEEE Transactions on Knowledge and Data Engineering* 24.1 (Jan. 2012), pp. 45–58 (cit. on p. 56).

[WCZ13] Y.H. Wang, K. Cao, and X.M. Zhang. "Complex event processing over distributed probabilistic event streams". In: *Computers & Mathematics with Applications* 66.10 (Dec. 2013), pp. 1808–1821 (cit. on pp. 63, 151).

[WES04] Wei Wei, Jordan Erenrich, and Bart Selman. "Towards efficient sampling: Exploiting random walk strategies". In: *Proceedings of the 19th National Conference on Artificial Intelligence*. 2004, pp. 670–676 (cit. on pp. 85, 86, 116).

[WGC17] Yongheng Wang, Hui Gao, and Guidan Chen. "Predictive complex event processing based on evolving Bayesian networks". In: *Pattern Recognition Letters* (May 2017) (cit. on pp. 60, 61).

[WGE05] Segev Wasserkrug, Avigdor Gal, and Opher Etzion. "A Model for Reasoning with Uncertain Rules in Event Composition Systems". In: *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. Edinburgh, Scotland: AUAI Press, 2005, pp. 599–608 (cit. on p. 56).

[WJ12] Xiaoyang Wang and Qiang Ji. "Incorporating Contextual Knowledge to Dynamic Bayesian Networks for Event Recognition". In: *21st International Conference on Pattern Recognition*. 2012, pp. 3378–3381 (cit. on p. 59).

[WJ14] Xiaoyang Wang and Qiang Ji. "Context augmented Dynamic Bayesian Networks for event recognition". In: *Pattern Recognition Letters* 43 (July 2014), pp. 62–70 (cit. on p. 59).

[WS02] Wei Wei and Bart Selman. "Accelerating Random Walks". In: *International Conference on Principles and Practice of Constraint Programming*. Springer, 2002, pp. 216–232 (cit. on p. 107).

[ZDI10] Haopeng Zhang, Yanlei Diao, and Neil Immerman. "Recognizing patterns in streams with imprecise timestamps". In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 244–255 (cit. on pp. 62, 63).

[ZDI14] Haopeng Zhang, Yanlei Diao, and Neil Immerman. "Optimizing Expensive Queries in Complex Event Processing". In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 217–228 (cit. on p. 63).

Acronyms

AIG Active Instance Graph. 53, 54, 143

ATC Air Traffic Control. 27–29, 31, 32

BDD Binary Decision Diagram. 9, 80, 81, 111, 152, 169, 173

BSC Binary-Sequential CSP. 82

CE complex event. 12–15, 18, 19, 21–25, 35–39, 46, 48, 50–57, 59, 61–64, 86, 88, 89, 97, 127, 139, 148, 149, 151, 152

CEP Complex Event Processing. 7–9, 11–15, 17, 21, 24, 29, 34, 36, 38, 40, 42, 43, 45–48, 52, 61, 69, 70, 79, 86, 109, 124, 143, 148, 151

CNF Cunjunctive Normal Form. 72

CRL Chronicle Recognition Library. 27, 127, 130, 146, 147

CRS Chronicle Recognition System. 24

CSP Constraint Satisfaction Problem. 67, 82, 132, 140, 148, 152

d-DNNF Deterministic Decomposable Negation Normal Form. 81, 152

DBN dynamic Bayesian network. 45, 50, 51, 68, 124

DFA Deterministic Finite Automaton. 64, 137, 141–143, 149

DNF Disjunctive Normal Form. 80

EC Event Calculus. 16, 17, 61, 167

FOL First-Order Logic. 9, 40, 47, 51, 67, 70–73, 77, 78, 87, 111, 151, 152

HLE High Level Event. 12, 40–42, 48, 50

HMM Hidden Markov Model. 45, 60, 61, 64, 67, 68, 119, 121

LLE Low Level Event. 12, 13, 15, 22, 24, 29, 35–38, 40–44, 46, 48–58, 61, 64, 67, 87–89, 98

MAP maximum a posteriori. 38, 60, 73

MCMC Markov Chain Monte-Carlo. 9, 59, 71, 75, 76, 127, 130, 131, 139, 151, 171

MLN Markov Logic Network. 9, 34, 39–47, 52, 60, 61, 67, 69–71, 73, 75, 78, 86, 94, 104, 111–114, 151, 152, 167

MRF Markov random field. 70, 71

NFA Non-deterministic Finite Automaton. 21, 23, 53, 63

NHM non-homogenous Markov model. 9, 70, 81, 82, 124, 126, 127, 129, 130, 132, 134, 136, 139, 146–149

PGM Probabilistic Graphical Model. 40, 70, 116

PPN Probabilistic Petri Net. 63, 65, 169

RPS Remote Pilot Station. 28, 29, 31, 32

SCFG stochastic context-free grammar. 64, 66, 67

SDD Sentential Decision Diagram. 43, 81

SLD resolution Selective Linear Definite clause resolution. 80

TESLA Trio-based Event Specification LAnguage. 21

UA Unmanned Aircraft. 27–29, 31

UAS Unmanned Aircraft System. 28

UAV Unmanned Aerial Vehicle. 12, 27, 28

List of Tables

1.1 Main predicates of the EC. [Art+12a] 22

2.1 Hierarchy of event from [Son+13]. 52

3.1 Example of a first-order knowledge base and MLN. Fr() is short for Friends(), Sm() for Smokes(), and Ca() for Cancer() [RD06]. 80

4.1 List of predicates used for the MLN model 96
4.2 Common benchmarks of 3-SAT problems tested for WalkSAT inference. Source: [HS05], Chapter 6. 108
4.3 Ratio of true assignments for Equation 4.29. 115

5.1 Inference results with ProbLog and MLN with different evidences. . . . 122

6.1 Marginal probability for the initial state and conditional probabilities from a state to another. (C:Calm weather, H:Hurricane, Off:Alarm off, On:Alarm on) 137
6.2 Representation of chronicle $(a\ b) - [c]$ with $\Sigma = a, b, c, d$ as conditional probabilities $P(A_t|A_{t-1}, S_t)$ 150
6.3 Computation time regarding the length of samples for the drone model (for 10^4 samples). 154

List of Figures

1.1 A simple hurricane alarm scenario producing a data-stream. 19
1.2 Structural representation of our semantic for activities and events. . . . 20
1.3 Graphical examples of SASE strategies. 26
1.4 Schematic representation of the drone loss communication study case . 34
1.5 State diagram of telecommand and radio loss. 36

2.1 Example of conditional dependencies between LLEs on a data stream. The star node represents a state where b will not appear. 45
2.2 Model HDBN from [MM07] 59
2.3 Architecture system proposed in [WGC17] 61
2.4 NFA to AIG [SKK08] 62
2.5 Example PPG definition of an activity from [Mol+14]. Edges are labelled with their δ and ρ values: $(\delta(e), \rho(e))$. 66
2.6 An example of relationship between processes, tasks, and events. ([Faz+18a]) 67
2.7 A CAS-graph example for a data stream $\varphi = \alpha\beta\zeta\zeta$. ([Faz+18a]) 67
2.8 Schematic representation of the study case in [Ré+08]. RFID reading has detected the person at instant 6 in the `H1` zone, but its position has been lost due to imprecise reading at time 7. 70
2.9 PPN computation rules from [Alb+08]. 73

3.1 Ground Markov network obtained by applying the last two formulae in Table 3.1 to the constants Anna(A) and Bob(B). Larger edges indicates higher weights [RD06]. 81
3.2 Graphical representation of a slice step during sampling. 84
3.3 BDD example for formula $(\neg X_1, \neg X_2, \neg X_3) \vee (X_1, X_2) \vee (X_2, X_3)$ 89

4.1 Inference result with Alchemy on the problem described in Equation 4.21 with $B(1)$ set as evidence. Initialised with seed 445. Left side MC-SAT inference, right side Belief propagation. 104
4.2 Variations of probability estimates of ground predicates $A(1, t_1)$ regarding the weigh for the first rule in Equation 4.21. For practical reasons, hard constrained rules are defined as weighted rules with weight 100. . 105
4.3 Flips necessary to find a satisfiable assignment for Equations 4.22 and 4.23 regarding the number of predicates. 107
4.4 Average number of flips used to find a assignment for Equation 4.24 with simple and double implications regarding the number of predicates. 108
4.5 A simplified version of chronicles representation 110
4.6 Flips necessary to find a satisfiable assignment for Equation 4.25 where height of the structure is equal to its base length. 110
4.7 Graphical representation of Equation 4.27 with $L = 3$ and $k = 4$. 112
4.8 Flips necessary to reach a solution regarding the value k in Equation 4.27. 113
4.9 Flips necessary to reach a solution regarding the value k in Equation 4.27 when evidences is provided with x_1^L (*highest* predicate). 114
4.10 Flips necessary to reach a solution regarding the truth value of evidences. 115

5.1 Hurricane alarm model and transition probabilities 127
5.2 Results for the query defined in Equation 5.16. 130
5.3 Sub-systems for the voice and telecommand failures. 131

6.1 Initial sampling approach representation with the different parts required or calculated 136
6.2 The hurricane alarm model. 136
6.3 Dependency graph for the NHM $\tilde{M}$ computed for the hurricane model and the data-stream $\varphi_1 = [\langle off, 0\rangle, \langle off, 1\rangle, \langle off, 2\rangle, \langle off, 3\rangle, \langle off, 4\rangle]$. An edge between two states indicates a possible state change from t to $t + 1$. 137
6.4 Inference time for 10 000 samples regarding stream length with ProbLog, MLN and our MCMC approach 139
6.5 Chronicle processing as evidence for inference. 141
6.6 Three streams represented as an accepting graph 143
6.7 Non accepting graph for $\Phi' = \Phi \bigcup \varphi_z$ 143
6.8 Accepting graphs for $\Phi' = \Phi \bigcup \varphi_z$ 143
6.9 Accepting graph for $\Phi'' = \Phi' \bigcup \varphi_\alpha \bigcup \varphi_\beta \bigcup \varphi_\gamma$ 144

6.10 Minimal automaton $\mathcal{A}_{\Phi'}$ for the streams $\Phi' = \varphi_w, \varphi_x, \varphi_y, \varphi_z$ and resulting separation. 145
6.11 Full process representation for the MCMC approach with chronicles set as evidence. 147
6.12 Estimation error compared to exact evaluation (based on ProbLog). . . 148
6.13 Markov chain representation for chronicle inference. In blue $P(A_t|A_{t-1}, S_t)$ provided by the automaton, in red $P(S_t|S_{t-1})$ provided by the NHM. . 149
6.14 DFA for chronicle $(a\ b) - [c]$ with $\Sigma = a, b, c, d$ 150
6.15 Acyclic version of DFA in Figure 6.14 with seven iterations 151
6.16 Communication lost protocol 152
6.17 Computation time for the NHM construction regarding its length . . . 156

List of Algorithms

1 MaxWalkSAT($KB, m_t, m_f, target, p$) [PD06] 82
2 MC-SAT(L, n) 85
3 Recursive algorithm for probability computation at a given node n in the BDD [RKT07] 89
4 Split_automaton 146

Résumé Le traitement d'événements complexes (Complex Event Processing – CEP) consiste en lanalyse de flux de données afin den extraire des motifs et comportements particuliers décrits, en général, dans un formalisme logique. Dans lapproche classique, les données dun flux – ou événements – sont supposées être lobservation complète et parfaite du système produisant ces événements. Cependant, dans de nombreux cas, les moyens permettant la collecte de ces données, tels que des capteurs, ne sont pas pour autant infaillibles et peuvent manquer la détection dun événement particulier ou au contraire en produire. Dans cette thèse, nous nous sommes employé à étudier les modèles possibles de représentation de lincertain et, ainsi, offrir au CEP une robustesse vis-à-vis de cette incertitude ainsi que les outils nécessaires pour permettre la reconnaissance de comportement complexe de façon pertinente les flux dévénements en se basant sur le formalisme des chroniques. Dans cette optique, trois approches ont été considérées. La première se base sur les réseaux logiques de Markov pour représenter la structure des chroniques sous un ensemble de formules logiques adjointe dune valeur de confiance. Nous montrons que ce modèle, bien que largement appliqué dans la littérature, est inapplicable pour une application concrète au regard des dimensions dun tel problème. La seconde approche se basent sur des techniques issues de la communauté SAT pour énumérer lensemble des solutions possibles dun problème donné et ainsi produire une valeur de confiance pour la reconnaissance dune chronique exprimée, encore une fois, sous une requête logique. Finalement, nous proposons une dernière approche basée sur les chaines de Markov pour produire un ensemble déchantillons expliquant lévolution du modèle en accord avec les données observées. Ces échantillons sont ensuite analysés par en système de reconnaissance pour compter les occurrences dune chronique particulière.

Mots clés: Traitement d'événements complexes ; Gestion de l'incertitude ; Reconnaisance de comportement ; Réseaux logiques de Markov ; ProbLog ; Chaînes de Markov non-homogènes ; WalkSAT

Abstract Complex Event Processing (CEP) consists of the analysis of data-streams in order to extract particular patterns and behaviours described, in general, in a logical formalism. In the classical approach, data of a stream – or events – are supposed to be the complete and perfect observation of the system producing these events. However, in many cases, the means for collecting such data, such as sensors, are not infallible and may miss the detection of a particular event or on the contrary produce. In this thesis, we have studied the possible models of representation of uncertainty and, thus, to offer the CEP a robustness to this uncertainty as well as the necessary tools to allow the recognition of complex behaviours based on the chronicle formalism. In this perspective, three approaches have been considered. The first one is based on Markov logical networks to represent the structure of the chronicles under a set of logical formulas of a confidence value. We show that this model, although widely applied in the literature, is inapplicable for a realistic application with regard to the dimensions of such a problem. The second approach is based on techniques from the SAT community to enumerate all possible solutions of a given problem and thus to produce a confidence value for the recognition of a chronicle expressed, again, under a logical structure. Finally, we propose a last approach based on the Markov chains to produce a set of samples explaining the evolution of the model in agreement with the observed data. These samples are then analysed by a recognition system to count the occurrences of a particular chronicle.

Keywords: Complex event processing; Behaviour recognition; Markov logic networks; Non-homogeneous Markov model; ProbLog; WalkSAT

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE NANTES

Titre : Behaviour Recognition on Noisy Data-streams Constrained by Complex Prior Knowledge
Mots clés : Traitement d'événements complexes, gestion de l'incertain, modèles graphiques probabilistes, réseaux logiques de Markov, chaînes de Markov non-homogènes

Résumé : Le traitement d'événements complexes (CEP) consiste en l'analyse de flux de données pour en extraire des comportements particuliers décrits, en général, dans un formalisme logique. Dans l'approche classique, les événements d'un flux sont supposés être l'observation parfaite du système produisant ces événements. Cependant, les moyens utilisés pour la collecte de ces données ne sont pas toujours infaillibles et peuvent manquer la détection d'un événement particulier ou au contraire en produire. Dans cette thèse, nous nous sommes employé à étudier les modèles possibles de représentation de l'incertain pour offrir au CEP les outils nécessaires pour permettre la reconnaissance de comportements complexes de façon pertinente les flux incertain d'événements en se basant sur le formalisme des chroniques.

Trois approches ont été considérées. La première se base sur les réseaux logiques de Markov pour représenter des chroniques via un ensemble de formules logiques adjointe d'une valeur de confiance. La seconde approche se base sur des techniques issues de la communauté SAT pour énumérer l'ensemble des solutions possibles d'un problème donné et ainsi produire une valeur de confiance pour la reconnaissance d'une chronique exprimée, encore une fois, sous une requête logique. Finalement, nous proposons une dernière approche basée sur les chaines de Markov pour produire un ensemble d'échantillons expliquant l'évolution du modèle en accord avec les données observées. Ces échantillons sont ensuite analysés par en système de reconnaissance pour compter les occurrences d'une chronique particulière.

Title : Behaviour Recognition on Noisy Data-streams Constrained by Complex Prior Knowledge

Keywords : Complex event processing, incertitude representation, Markov logic networks, Non-homogenous Markov models, probabilistic graphical models

Abstract : Complex Event Processing (CEP) consists of the analysis of data-streams in order to extract particular patterns and behaviours described, in general, in a logical formalism. In the classical approach, data of a stream -- or events -- are supposed to be the complete and perfect observation of the system producing these events. However, in many cases, the means for collecting such data, such as sensors, are not infallible and may miss the detection of a particular event or on the contrary produce. In this thesis, we have studied the possible models of representation of uncertainty and, thus, to offer the CEP a robustness to this uncertainty as well as the necessary tools to allow the recognition of complex behaviours based on the chronicle formalism. In this perspective, three approaches have been considered.

The first one is based on Markov logical networks to represent the structure of the chronicles under a set of logical formulas of a confidence value. We show that this model, although widely applied in the literature, is inapplicable for a realistic application with regard to the dimensions of such a problem. The second approach is based on techniques from the SAT community to enumerate all possible solutions of a given problem and thus to produce a confidence value for the recognition of a chronicle expressed, again, under a logical structure. Finally, we propose a last approach based on the Markov chains to produce a set of samples explaining the evolution of the model in agreement with the observed data. These samples are then analysed by a recognition system to count the occurrences of a particular chronicle.