

Analysing and Visualising (Cyber)crime data using Structured Occurrence Nets and Natural Language Processing



Tuwailaa Alshammari

Supervisor: Prof. Maciej Koutny

School of Computing

Newcastle University

This dissertation is submitted for the degree of

Doctor of Philosophy

Saturday 1st March, 2025

To my parents, family - especially my siblings, nieces, and nephews- my supervisor, and dear friends, whose support and encouragement have been invaluable.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains less than 40,000 words including appendices, bibliography, footnotes, tables and equations and has about 50 figures.

Tuwailaa Alshammari
Saturday 1st March, 2025

Acknowledgements

First and foremost, all praise is due to Allah the Almighty for His blessings. Secondly, I would like to express my sincere gratitude to the individuals who contributed to the development of this thesis.

I am deeply thankful to my parents and family for their unwavering support, which has been a consistent source of inspiration and motivation.

I am especially grateful to my supervisor and mentor, Professor Maciej Koutny, for everything really. This work would not have been achieved without his support, guidance, and encouragement. His expertise and exceptional personality have made a significant impact on my academic journey. I feel fortunate to have had the opportunity to learn from and work with him.

Special thanks to the members of the SON group, Professor Brian Randell, Salma, and my second supervisor, Dr. Marta Pietkiewicz-Koutny. I am deeply grateful to Dr. Anirban Bhattacharyya for his constructive discussions, feedback, and suggestions. Massive thank you, Ani. My deep and sincere thanks go to my close colleagues, Mohammed and Nadiyah, for their unwavering support during my research journey; their support and discussions are valued and remembered with appreciation. Additionally, a huge thanks to my close friend and colleague, Dr. Talal Alharbi, for his support, encouragement, and guidance. My heartfelt thanks to my dear and close friend, Mohammed Allwaish, for his invaluable support. I would also like to thank my examiners, Prof. Prem Prakash Jayaraman and Dr. Tejal Shah for their expertise and valuable suggestions which improved this thesis.

Finally, I want to acknowledge everyone who contributed, in any way, to my academic journey. Your involvement has been crucial in helping me complete this thesis.

Abstract

Structured Occurrence Nets (SONs) are a Petri net-based formalism designed to represent the behaviour of complex evolving systems, capturing concurrent events and interactions between subsystems. Recently, the modelling and visualisation of crime and cybercrime investigations have gained increasing interest. In particular, SONs have proven to be versatile tools for modelling and visualising various applications, including crime and cybercrime. This thesis presents two contributions aimed at making SON-based techniques suitable for real-life applications.

The main contribution is motivated by the fact that manually developing SON models from unstructured text can be time-consuming, as it requires extensive reading, comprehension, and model construction. This thesis aims to develop a methodology for the formal representation of unstructured textual resources in English. This involves experimenting, mapping, and deriving relationships between natural and formal languages, specifically using SON for crime modelling and visualisation as an application.

The second contribution addresses the scalability of SON-based representations for cybercrime analysis. It provides a novel approach in which acyclic nets have been extended with coloured features to enable reduction of net size to help in visualisation.

While the two contributions address distinct challenges, they are unified by their use of SONs as a formalism to model complex systems. Structured occurrence nets demonstrated their adaptability in representing both crime scenarios and cybercrime activities.

Table of contents

| | |
|---|------------|
| List of figures | xv |
| List of tables | xix |
| 1 Introduction | 1 |
| 1.1 Background | 1 |
| 1.2 Aims and objectives | 3 |
| 1.3 Contributions | 4 |
| 1.4 Thesis outline | 5 |
| 1.5 List of Publications | 5 |
| 2 Structured Occurrence Nets, Visualisation, and Natural Language Processing | 7 |
| 2.1 (Cyber)crime Modelling and Visualisation | 7 |
| 2.1.1 Decision Trees | 8 |
| 2.1.2 Attack Graphs | 9 |
| 2.1.3 Crime Visualisation using Machine Learning | 10 |
| 2.2 Natural Language Processing in Crime Analysis | 11 |
| 2.2.1 Overview | 11 |
| 2.2.2 SPACY | 12 |
| 2.2.3 Crime Information Extraction | 13 |
| 2.3 Structured Occurrence Nets (SONs) | 15 |
| 2.3.1 Occurrence Nets (ONs) | 15 |
| 2.3.2 Communication Structured Occurrence Nets (CSO-nets) | 16 |

| | | |
|----------|---|-----------|
| 2.3.3 | Behavioural Structured Occurrence Nets (BSO-nets) | 17 |
| 2.3.4 | Other formal representations | 17 |
| 2.4 | Challenges and Limitations for Crime Modelling in SONS | 20 |
| 3 | Acyclic Nets and Communication Structured Acyclic Nets | 21 |
| 3.1 | Introduction | 21 |
| 3.2 | Acyclic nets | 21 |
| 3.2.1 | Conflict, Concurrency, and Causality | 25 |
| 3.2.2 | Step sequence semantics of acyclic nets | 26 |
| 3.2.3 | Well-formed acyclic nets | 29 |
| 3.3 | Communication structured acyclic nets (CSA-nets) | 30 |
| 3.3.1 | Step sequence semantics of CSA-nets | 35 |
| 3.3.2 | Well-formed CSA-nets | 37 |
| 3.4 | Conclusion | 38 |
| 4 | SONs and NLP Mapping and Integration | 39 |
| 4.1 | Introduction | 39 |
| 4.2 | Methodology | 40 |
| 4.2.1 | Terminology | 42 |
| 4.3 | Dataset | 42 |
| 4.4 | Manual modelling | 43 |
| 4.4.1 | Manual modelling experiment | 43 |
| 4.5 | Analysis and Mapping to SONS | 44 |
| 4.5.1 | Evaluation of Human Modelling | 44 |
| 4.5.2 | Rules for SON and Natural Language Mapping | 46 |
| 4.5.3 | Formalisation of the construction | 47 |
| 4.6 | Automatic extractor development | 48 |
| 4.6.1 | Automatic extraction: TEXT2SON | 49 |
| 4.7 | Discussion | 52 |
| 4.8 | Background | 56 |

| | | |
|----------|--|-----------|
| 4.8.1 | Visualisation and modelling | 56 |
| 4.8.2 | Use of NLP in formal representation | 57 |
| 4.9 | Conclusions | 58 |
| 5 | Extracting Data from Unstructured Text for Representation in SONS | 61 |
| 5.1 | Introduction | 61 |
| 5.2 | Datasets | 62 |
| 5.3 | Manual Modelling and Analysis | 63 |
| 5.3.1 | Manual modelling | 63 |
| 5.4 | Analysis and Preprocessing | 69 |
| 5.5 | Automatic Modelling Experiment | 70 |
| 5.5.1 | Entities identification and analysis | 70 |
| 5.5.2 | Verbs identification and analysis: <i>pattern-based</i> | 73 |
| 5.5.3 | Entity-verb Linking and Communication Structuring | 75 |
| 5.6 | Discussion | 81 |
| 5.7 | Conclusions | 82 |
| 6 | Coloured Acyclic Nets (CA-nets) | 85 |
| 6.1 | Introduction | 85 |
| 6.2 | Datasets | 87 |
| 6.2.1 | Dataset analysis | 87 |
| 6.2.2 | Feature selection | 88 |
| 6.3 | Model Design | 89 |
| 6.3.1 | Modelling | 89 |
| 6.3.2 | Coloured acyclic nets | 92 |
| 6.3.3 | Classifier design | 93 |
| 6.4 | Experiment | 98 |
| 6.4.1 | Evaluation and discussion | 99 |
| 6.5 | Related Work | 101 |
| 6.6 | Conclusions | 103 |

| | | |
|-------------------|---|------------|
| 7 | Concluding Remarks | 105 |
| 7.1 | Limitations | 106 |
| 7.2 | Future work | 107 |
| Appendix A | | 109 |
| A.1 | Modelling examples | 109 |
| A.1.1 | More models for STORY A | 109 |
| A.1.2 | Crime B story Models and analysis | 111 |
| A.1.3 | Crime C story Models and analysis | 116 |
| A.1.4 | STORY D models and analysis | 119 |
| A.1.5 | STORY E models and analysis | 123 |
| | References | 131 |

List of figures

| | | |
|-----|---|----|
| 2.1 | An example of a decision tree for a crime situation classification into dangerous or neutral. | 8 |
| 2.2 | An example of an attack graph from [39]. | 10 |
| 2.3 | Example depicting SPACY pipeline from [60]. | 12 |
| 2.4 | Visualisation of street crime hotspots across London, UK, extracted from online news reports (from [48]). | 14 |
| 2.5 | Simple occurrence net. | 16 |
| 2.6 | Communication structured occurrence net (CSO-net). | 16 |
| 2.7 | Behavioural structured occurrence net (BSO-net). | 17 |
| 3.1 | Acyclic net $acnet_1$ | 22 |
| 3.2 | Backward deterministic acyclic net $bdacnet_1$ | 24 |
| 3.3 | Occurrence net. | 24 |
| 3.4 | Two maximal scenarios of the acyclic net in Figure 3.1. | 25 |
| 3.5 | Acyclic net $bdacnet_1$ of Figure 3.2 with the initial marking indicated. | 29 |
| 3.6 | Non-well-formed acyclic net. | 30 |
| 3.7 | Communication structured acyclic net $csan$ | 32 |
| 3.8 | Communication structured occurrence net $cson$ | 33 |
| 4.1 | Methodology used to build the automatic extractor. | 41 |
| 4.2 | Simple occurrence net. | 41 |
| 4.3 | A short fragment of a crime story | 44 |
| 4.4 | Manual modelling for text in Figure 4.3 done by SON expert users. | 45 |

| | | |
|-----|---|----|
| 4.5 | Example of spaCy pipeline from [61]. | 49 |
| 4.6 | Including NeuralCoref 4.0 in spaCy pipeline. | 49 |
| 4.7 | Tools for extractor design (a); and extractor design (b) | 51 |
| 4.8 | The example text in Figure 4.3 after applying coreference resolution using NEURALCOREF 4.0 | 51 |
| 4.9 | Visualisation for the automated extraction of text in Figure 4.8 using the three methods. (a) METHOD1: extracting <i>root</i> verbs, (b) METHOD2: extracting <i>root verbs</i> and <i>common verbs</i> , and (c) METHOD3: extracting all verbs. . . . | 54 |
| 5.1 | STORY A representation by SON expert users (a) MODELLER1 and (b) MOD- ELLER2, the rest of modeller representations are in Appendix A. | 68 |
| 5.2 | Example of SON model analysis for STORY A. | 69 |
| 5.3 | Example of an annotation process shows text and entity labels. The entity labels in this example include tags for PERSON and WEAPON. | 71 |
| 5.4 | Format of the dataset prepared for training. | 71 |
| 5.5 | Extraction output shows how lists are structured. | 75 |
| 5.6 | Example of constructing an acyclic net. | 76 |
| 5.7 | Example of constructing buffer place q_1 between $entity^1$ and $entity^2$ <i>acyclic</i> <i>nets</i> | 77 |
| 5.8 | A potential SON model could be developed based on the results of the extraction for STORY A. The extraction approach identified more entities and verbs compared to manual identification in Figure A.2. | 80 |
| 6.1 | Genuine and fake account follower/following ratio comparison using a sam- ple from the dataset in [17]. | 90 |
| 6.2 | Two different designs: (a) with conflict between a and b ; and (b) without conflict. | 91 |
| 6.3 | Example showing different colours where only tokens of colour Twitter enables the transition. | 92 |
| 6.4 | Classifier design using CA-net. | 94 |

| | | |
|------|--|-----|
| 6.5 | Schematic reachability graph. | 98 |
| 6.6 | Tokens distribution in the CA-net. | 100 |
| A.1 | Two models by two modellers for STORY A | 110 |
| A.2 | Two models created by different SON expert users depict the events from STORY B and demonstrate close entity identification. | 112 |
| A.3 | Two models created by different SON expert users depict the events from STORY B and demonstrate close entity identification. | 113 |
| A.4 | A potential SON model could be developed based on the results of the extraction for the same story (STORY B). The extraction approach identified more entities and verbs compared to human identification in Figures A.2 and A.3. | 115 |
| A.5 | Two models created by different SON expert users depict the events from STORY C. | 117 |
| A.6 | Another two models created by different SON expert users depict the events from STORY C. | 118 |
| A.7 | A models created by different SON expert user depict the events from STORY D. | 120 |
| A.8 | A models created by different SON expert user depict the events from STORY D. | 121 |
| A.9 | Two models created by different SON expert users depict the events from STORY D. | 122 |
| A.10 | A model created by SON expert user depicts the events from STORY E. . . . | 125 |
| A.11 | A model created by SON expert user depicts the events from STORY E. . . . | 128 |
| A.12 | A model created by SON expert user depicts the events from STORY E. . . . | 129 |
| A.13 | A model created by SON expert user depicts the events from STORY E. . . . | 130 |

List of tables

| | | |
|-----|--|-----|
| 4.1 | Key terminology and their definitions used in this chapter | 42 |
| 4.2 | List of all extracted verbs by various manual modellers and methods. | 55 |
| 5.1 | Identified entities representing <i>acyclic nets</i> by manual modellers | 66 |
| 5.2 | Verbs representing events extracted manually from STORY A | 67 |
| 5.3 | NER model performance validation | 72 |
| 5.4 | NER model performance testing on different dataset | 72 |
| 5.5 | List of all patterns identified from the manual models created by SON users [POS]verb[POS]. | 74 |
| 5.6 | Entities from STORY A. | 79 |
| 5.7 | Number of acyclic nets and transitions extracted for Manual and Automatic methods | 79 |
| 6.1 | List of all account features from the dataset. Default feature values are assigned to every account upon creation | 88 |
| 6.2 | Selected account features we used in building the classifier model | 89 |
| 6.3 | The explanation of all nodes from the proposed classifier model | 95 |
| 6.4 | Performance evaluation metrics for the classifier | 100 |
| A.1 | Comparison of verb extraction methods applied to STORY A. | 109 |
| A.2 | Extracted Entities from STORY A. | 111 |
| A.3 | Comparison of verb extraction methods applied to STORY B. | 114 |
| A.4 | Extracted Entities from STORY B. | 114 |

| | | |
|------|---|-----|
| A.5 | Comparison of verb extraction methods applied to STORY C. | 116 |
| A.6 | Extracted Entities from STORY C. | 119 |
| A.7 | Comparison of verb extraction methods applied to STORY D. | 123 |
| A.8 | Extracted Entities from STORY D. | 123 |
| A.9 | Comparison of verb extraction methods applied to STORY E. | 126 |
| A.10 | Extracted Entities from STORY E. | 127 |

Chapter 1

Introduction

1.1 Background

Complex Evolving Systems (CES) are characterised by the dynamic interactions and evolution of the system's components and structure. The analysis of such systems has been studied across diverse fields, such as computer science, engineering, and biology. Formal models play an important role in understanding CES by supporting a representation approach to capturing, visualising, and analysing their behaviour.

Formal models cover a diverse range of frameworks, each with its own syntactical and semantic properties. These frameworks enable the representation of complex systems formally verified for correctness, and some representations are interpretable. In particular, Structured Occurrence Nets (SONs) represent a formal method that is distinguished by its capacity to model and analyse the occurrence of events within a system. By visually representing the synchronised interactions and causal links among events, SONs provide a powerful tool for analysing CES.

SONs are a Petri net-based formalism used to represent the behaviour of complex evolving systems that consist of interdependent subsystems which proceed concurrently and interact with each other. An extension of SONs are the Communication Structured Acyclic nets (CSA-nets), which are based on *acyclic nets*. A CSA-net joins two or more acyclic nets by employing *buffer places* to connect pairs of *transitions* from different acyclic nets. The nature

of such connections can be synchronous or asynchronous. In synchronous communication, transitions are executed simultaneously, whereas in asynchronous communication, transitions may be executed simultaneously or in sequence. Furthermore, an *occurrence net* is an acyclic net with a completely unambiguous record of all causal dependencies between involved transitions. In particular, it exhibits both backward and forward determinism. This thesis uses the terms ‘acyclic net’ and ‘occurrence net’ interchangeably.

Visualising crime can be highly beneficial for investigators, as it transforms textual information into structurally visualised models. Prior to the study reported in this thesis, SONs had proven to be a powerful tool for crime modelling and visualisation. Previous research into the JFK assassination [27] demonstrated the effectiveness of SONs as a robust framework for modelling crimes. Similarly, another study that analysed a burglary referred to as the Silverlink case [55], used SONs to model and understand the sequence of events. However, the construction of SON models in these studies was performed manually making the modelling process laborious. Building on this prior work, this thesis introduces methods for automatically extracting crime data from unstructured text. The implementation of automatic extraction can help to produce faster and more consistent models.

As a result, the main contribution of this thesis lies in the innovative approach to *identifying and extracting* SON components from unstructured textual resources, using crime stories as a case study. Unstructured text, such as crime reports, presents a challenge for modelling due to its variability and lack of uniform structure. This thesis addresses this challenge by developing a methodology to map textual elements to SONs using Natural Language Processing (NLP). NLP tools can assist in analysing text by systematically labelling words with part-of-speech and dependency tags. This, therefore, can greatly help in the extraction of meaningful data from unstructured texts. Currently, NLP tools have demonstrated more accurate extraction results having many new tools being introduced in recent years, such as spaCy, CoreNLP, and AllenNLP. This led us to investigate the possibility of integrating NLP with SON to extract crime information from unstructured text and use it for crime modelling and analysis.

The extraction process involves several steps, beginning with the identification of appropriate mappings between SON components and textual elements. This includes developing methods to map transitions and acyclic nets to the corresponding words or phrases in natural language. An important part of this process is the creation of a custom Named Entity Recognition (NER) model tailored to identify crime-related entities within the text. Additionally, methods are proposed to identify accurately the verbs, which we found are essential for modelling transitions in SONS.

The second contribution of this thesis extends acyclic nets to reduce the size of net visualisations by introducing Coloured Acyclic Nets (CA-nets). CA-net features enable the reduction of the size of net components by introducing coloured tokens. Moreover, transitions have the ability to evaluate token values, thus moving tokens to the appropriate path.

To conclude, this thesis makes two original contributions to the field of formal modelling and analysis. These contributions address the following questions:

1. Is it possible to automatically extract SON models from unstructured text?
2. Is it possible to reduce the size of acyclic nets without changing behaviour?

1.2 Aims and objectives

Aims: The aims of this thesis are: (i) to develop a methodology for identifying and extracting Structured Occurrence Net (SON) components from unstructured textual resources, using crime stories as a case study; and (ii) to extend acyclic nets with coloured features in order to reduce the size of net representations.

Objectives:

Identifying and mapping SON components from text: Develop a robust methodology to identify and map words and phrases in natural language to the corresponding acyclic nets and transitions. This approach is crucial for bridging the gap between natural language and formal representations, utilising Natural Language Processing (NLP) to identify and extract meaningful data from unstructured textual sources.

Developing methods for model extraction: Create and implement methods to accurately identify and extract SON components relying on the mapping methods from the first objective. This is crucial in order to extract acyclic nets and transitions from unstructured texts which are essential for SON modelling.

Extending acyclic nets to reduce size representation: Develop *coloured acyclic net* model which incorporates coloured attributes to reduce the size required for the representation of complex interactions.

1.3 Contributions

Methodology for identifying and mapping SON components: This thesis presents a novel approach for mapping and extracting *acyclic nets* and *transitions* from unstructured text, particularly crime stories. Based on practical experiments, mapping rules between SONs and natural language are proposed.

Entity and verb identification and extraction methods: An original key contribution is the development of a custom NER model tailored to identify crime-related entities within the text. Moreover, methods to accurately identify and extract verbs have been proposed and experimented with. These methods are crucial to building SON models.

Introduction and application of Coloured Acyclic Nets: This thesis introduces a Coloured Acyclic Net (CA-net) model which is a SON extension that incorporates additional attributes to help improve visualisation. The model's practical application is demonstrated through the design and construction of a classifier using CA-net to analyse social media accounts to determine genuine and fake accounts.

Note that I can confirm the ethical clearance for this study was obtained and approved, ensuring that all research activities adhered to the necessary ethical standards and guidelines. This approval demonstrates compliance with the principles of integrity, confidentiality, and respect for the participants involved in the research.

1.4 Thesis outline

The rest of the thesis is organised as follows:

Chapter 2: Provides background related to SONS, NLP, and crime modelling.

Chapter 3: Presents formal notations and properties for acyclic nets and CSA-nets.

Chapter 4: Provides an introduction to mapping natural language text to SONS, and an automated extraction experiment.

Chapter 5: Presents a methodology for extracting data from unstructured crime text stories in order to construct SON representations.

Chapter 6: Extends acyclic nets to CA-nets using coloured tokens.

Chapter 7: Summarises, concludes the thesis, and proposes directions for further work.

1.5 List of Publications

Parts of this thesis have been published as follows:

1. Alshammari, T. *Towards Automatic Extraction of Events for SON Modelling*. PNSE@Petri Nets (2022)
2. Alshammari, T. *Integrating NLP and Structured Occurrence Nets for Crime Modelling: A Pattern-based Approach*. PNSE@Petri Nets (2023)
3. Alshammari, T. *Extracting data from unstructured crime text to represent in Structured Occurrence Nets using Natural Language Processing*. PNSE@Petri Nets (2024)
4. Alshammari, T. *Using Coloured Acyclic Nets to Detect Fake Accounts on Social Media*. International conference on computer, software and modeling (ICCSM 2024)
5. Alahmadi, M., Alharbi, S., Alharbi, T., Almutairi, N., Alshammari, T., Bhattacharyya, A., Koutny, M., Li, B., and Randell, B. *Structured acyclic nets*. (2024). CoRR, abs/2401.07308.

Chapter 2

Structured Occurrence Nets, Visualisation, and Natural Language Processing

This chapter provides an overview of the various areas that collectively underpin this thesis. In the ongoing technological evolution where numerous projects are emerging from multidisciplinary domains, this thesis is no exception. The chapter starts with an overview of cybercrime and crime modelling and visualisation. It then explores natural language processing techniques and algorithms and their integration into crime analyses and modelling. Then it introduces an overview of SONS, their basic extensions, and covers other formal modelling frameworks. It concludes by discussing some SON modelling challenges.

2.1 (Cyber)crime Modelling and Visualisation

The rise of crime and cybercrime activities has motivated the development of advanced tools and methods to model and visualise criminal activities. Effective modelling and visualisation techniques are important to understand crime and cyber attacks nature. Different approaches have been explored to model and visualise crimes and cybercrime; for example, using

techniques from graph theory to create more detailed crime representations. Here are some approaches used for this purpose.

2.1.1 Decision Trees

Decision trees are graphical representations for classification that divide a dataset into smaller subsets and form tree-like structure. They are suitable for crime and cybercrime modelling because they can classify and predict outcomes based on different rules and attributes. In crime modelling, decision trees can be used to analyse large datasets and identify crime patterns. For example, a decision tree can assist law enforcement to discover crime patterns and predict future trends which can help to improve crime prevention plans. For instance, in [45] the decision tree is built by analysing attributes derived from user experiences, such as heartbeat rate, voice tone, and facial emotion, to classify situations as either “dangerous” or “neutral”. The situation evaluation process involves recording these attributes during different crime scenarios, training the decision tree to recognise patterns, and then show them in a decision tree as illustrated in Figure 2.1.

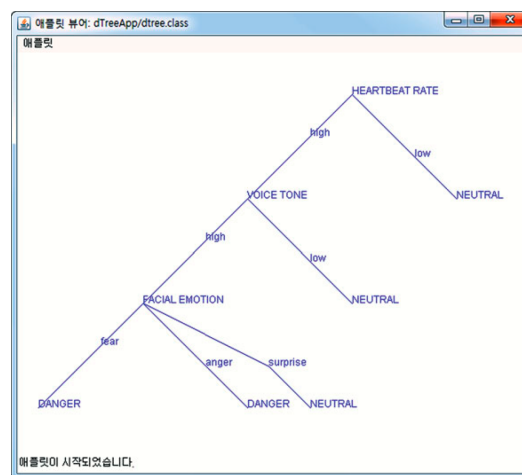


Fig. 2.1 An example of a decision tree for a crime situation classification into dangerous or neutral.

Furthermore, decision trees can be used to classify and predict suspicious activities in network traffic which helps in the early detection of cyber attacks. This approach is beneficial for computer forensics where decision trees help to identify associations among

massive data and detect crime-related evidence efficiently [69]. Moreover, enhanced decision tree algorithms have been proposed to produce accurate and faster classifications to detect suspicious criminal activities in cyberspace [58].

Decision trees can also provide methods to help comprehend crime by visualising patterns in complex datasets. For instance, using visualisation techniques such as clustering combined with decision trees, can enhance the interpretation of large datasets. This can be achieved by dividing the dataset into smaller segments which will allow for clearer visualisation of the decision making process [2]. Such visualisation can help analyse crime offences and criminal behaviour leading to identifying relations between criminal behaviour and criminal groups.

2.1.2 Attack Graphs

Attack graphs are an important approach in cybercrime modelling and visualisation. They offer a method to represent the sequence of events leading to a successful attack [35]. These graphs (as shown in Figure 2.2) can aid in cyber attack understanding by visually showing events flow that lead to successful attacks in a structured way. Thus, they are useful visual aids that illustrate potential attack paths and can be used for cyber attack perception [39].

Integrating attack graphs into cybersecurity tools enables the analysis of network vulnerabilities by generating and exploring potential attack paths. This method helps security analysts to visualise and prioritise risks, making it more effective to identify and mitigate critical security threats [47]. Furthermore, advanced interactive analysis and visualisation tools are proposed. A tool is introduced to represent attacks as nodes and actions as edges in an attack graph to enable for detailed analysis and aid cyberattack investigation [52].

Attack graphs can be extended to represent specific protocols as demonstrated in the study [62]. This method incorporates physical network topology and advanced communication protocols to model a broader range of cyber attacks. This extension is crucial for analysing vulnerabilities in diverse environments which make attack graphs a versatile tool for modern cybersecurity challenges. An empirical study [38] demonstrates the effectiveness of attack graphs in aiding the perception of cyber attacks.

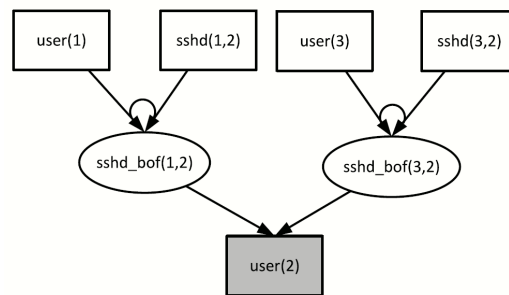


Fig. 2.2 An example of an attack graph from [39].

2.1.3 Crime Visualisation using Machine Learning

Machine Learning (ML) algorithms and models can analyse large amounts of data and identify patterns. In crime modelling, machine learning algorithms have been used to predict crime hotspots, visualise, and analyse crime trends. For example, a study on crime data utilised decision trees and random forest algorithms to classify crimes and predict future incidents based on geographical and temporal patterns [70].

In cybercrime, machine learning techniques have been helpful in detecting and preventing cyber attacks. An overview that highlights the effectiveness of various machine learning algorithms such as supervised, unsupervised, and reinforcement learning. It discusses their application in different cybersecurity contexts. The study investigates specific techniques such as decision trees, support vector machines (SVM), and neural networks to assess their roles in enhancing cybersecurity measures [20]. These algorithms enable real-time analysis and response to cyber threats, thereby strengthening the ability of organisations to protect their networks and data.

Visualisation tools that rely on machine learning models are crucial in both crime and cybercrime analysis. These tools leverage techniques such as clustering and pattern recognition to help analysts understand complex datasets. They help to uncover hidden relationships among different variables. For instance, a human-centered approach was proposed to analyse crime patterns. It integrates domain expert knowledge with machine learning to facilitate the extraction of associations from unstructured crime reports. Then visualise them through interactive clustering and bipartite knowledge graphs [51]. This approach not only enhances

the accuracy of analysis but also improves the interpretability of the results which makes it easier for law enforcement to develop effective strategies. Additionally, the use of machine learning techniques on large crime datasets allows for the detection of crime patterns and the accurate prediction of future incidents. This can be done by transforming big data into advanced visualisations, such as 3D scatter plots and polar plots, along with statistical analyses, these methods enable law enforcement to make more informed decisions [42].

2.2 Natural Language Processing in Crime Analysis

2.2.1 Overview

Natural Language Processing (NLP) techniques can be used in information extraction and in the analysis of textual data sources, such as crime reports. According to [44], NLP originated in the 1950s. The rise of computers necessitated the development of human-machine interaction to enable computers to understand human language by analysing human text and speech. NLP is a branch of artificial intelligence that can be used to facilitate this task. It involves the development of algorithms and systems for processing, understanding, and generating natural language text and speech. NLP combines aspects of computer science, linguistics, and machine learning to enable machines to interpret human language in a meaningful way. Early approaches to NLP included tasks such as machine translation and speech recognition and modern advances have led to applications such as sentiment analysis, dialogue systems, and information extraction [29].

NLP tools and methods have evolved significantly over the years, incorporating both rule-based and statistical techniques. Popular tools include Natural Language Toolkit (NLTK), Stanford CORENLP, and SPACY, just to name a few. NLTK provides a comprehensive suite of libraries for processing text, including tokenisation, stemming, and part-of-speech tagging [11, 23]. Stanford CORENLP is a text processing toolkit written in Java. It provides core linguistic analysis and annotations such as tokenisation, part of speech, and named entities. CORENLP currently supports text analysis for eight languages [41, 26]. Another notable tool is SPACY, an open-source natural language processing toolkit designed to help

developers implement natural language processing annotations and tasks. It is a statistical model known for its speed and efficiency in processing large volumes of text.

Despite its strengths, NLP faces several challenges. One of the main challenges is that human language is complex and hard for machines to understand accurately due to its ambiguity, reliance on context, and complex rules. Scalability is another concern, as processing large datasets requires significant computational resources. Additionally, while statistical methods and deep learning have improved the accuracy of NLP systems, they often require large amounts of labelled data and can be prone to biases present in the training data [13].

2.2.2 SPACY

SPACY [60] is an open-source library for Natural Language Processing in Python. It was created by *Explosion AI*, a software company co-founded by Matthew Honnibal and Ines Montani in 2015 [1]. SPACY provides pre-trained statistical models to support a variety of NLP tasks, including part-of-speech tagging, dependency parsing, and named entity recognition. The library is known for its speed and efficiency, making it a popular choice for both academic research and industrial applications.

The architecture of SPACY is built on powerful linguistic annotations which provide information about the syntactic structure of texts. It offers models with different sizes tailored to various needs, such as the lightweight models for quick prototyping and the larger models for tasks requiring higher accuracy. Additionally, it supports integration with deep learning frameworks like TensorFlow and PyTorch, allowing users to fine-tune models on custom data. This flexibility and performance make SPACY a versatile NLP tool [60].

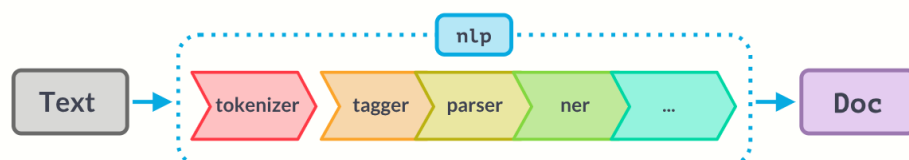


Fig. 2.3 Example depicting SPACY pipeline from [60].

The SPACY *pipeline* processes raw text through a series of steps, transforming it into a structured, analysable document. As shown in Figure 2.3, it begins with *tokenisation*, breaking the text into individual tokens. The *tagger* then assigns part-of-speech tags, and the *parser* builds a syntactic tree to understand grammatical relationships. The *named entity recogniser* (NER) identifies entities such as names of people and places. Additional components or models can be added for specific tasks. The final output is a *Doc* object containing all annotations prepared for detailed analysis.

2.2.3 Crime Information Extraction

Information extraction (IE) is the process of scanning text to identify and extract specific information relying on NLP techniques. It plays a crucial role in extracting data from various sources, especially unstructured text such as police reports, witness statements, and online news articles.

For instance, authors in [37] developed a system that extracts information from police and witness statements. The system integrates information extraction with cognitive interview techniques. It allows for detailed and accurate reporting by enabling users to describe incidents in natural language. This approach effectively captures key details, such as the individuals involved, locations, actions taken, and objects associated with the crime. These details are then used to generate comprehensive reports and follow-up questions that improve the overall crime data analysis process.

The work in [10] focused specifically on online newspaper articles reporting theft incidents. They specifically developed a method to automatically identify crime locations by classifying sentences as either crime location sentences or non-crime location sentences. By using NER and Conditional Random Fields (CRF), their study demonstrated the critical role of precise tool selection to achieve accurate information extraction from unstructured textual sources.

The CRIMEPROFILER framework introduced in [21] was designed to extract and create crime-related information from online news articles. This system not only identifies critical details (such as the names of criminals, victims, nature of the crime, and locations), but it also

uses a semi-supervised learning approach to keep crime dictionaries updated with emerging patterns. The resulting crime register demonstrates how integrating NLP techniques can aid in extracting information from unstructured text and provide insights for crime analysis.

The work in [53] introduces a system tailored to the Indonesian language for extracting crime-related information from online news articles. Their system identifies key crime entities such as crime type, victim, perpetrator, location, and time. An ontology-based classification method further categorises crime events and shows the need of creating language-specific tools for accurate information extraction. This system is valuable for helping to inform about crime from Indonesian news.

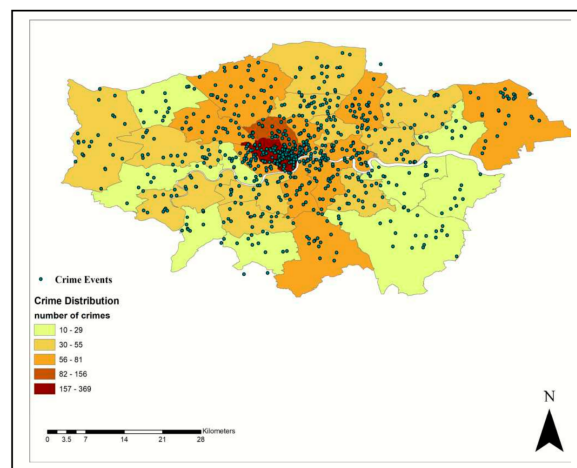


Fig. 2.4 Visualisation of street crime hotspots across London, UK, extracted from online news reports (from [48]).

Furthermore, NLP can help in spatial and temporal crime analysis by extracting relevant information from unstructured data sources. For example, researchers have developed methods to extract spatio-temporal data from news reports to detect crime hotspots and predict future criminal activities. This approach uses various NLP techniques to gather automatically data about the locations and times of crimes, as well as other relevant details. The information is then organised using a crime domain ontology that allows for better understanding of where and when crimes occur [48]. The system was tested with crime reports to identify areas in London with high crime rates and provide insights for crime prediction and prevention as illustrated in Figure 2.4. Another study utilised NLP to process

eight years of crime data from news archives. These data are transformed from unstructured data into actionable insights to predict criminal behaviour and identify criminal networks [67]. These applications demonstrate how NLP enhances the efficiency and accuracy of crime analysis and provides valuable tools for law enforcement.

Crime modelling and visualisation have been studied in the literature and reviewed in terms of classification, prediction, and visualisation. For instance, decision trees are employed to classify crime and thereby predict future crimes. Furthermore, spatial representation can show the geographic location of crimes which can provide law enforcement agencies with valuable tools for crime prevention. However, while these approaches are valuable, we aim to build models that can represent the *dynamic behaviour* of crime events by capturing how crime events occurred and identify the involved participants.

2.3 Structured Occurrence Nets (SONs)

Structured Occurrence Nets (SONs) are a subclass of Petri nets which can be used to analyse and visualise complex evolving systems. This section provides a general overview of SONs and their extensions, namely CSONs and BSONs, and overviews other formal representations, namely (*general*) *Petri Nets* and *Coloured Petri Nets*. Formal definitions of SONs and their behaviour are given in Chapter 3.

2.3.1 Occurrence Nets (ONs)

An *Occurrence Net* (ON) is a basic form of SON. Occurrence nets [4] are directed acyclic graphs that represent causality and concurrency information concerning a single execution of a distributed system. Multiple ONs representations can be used in combination to show the evolution of a system. Graphically, an ON consists of two types of nodes, *places* (conditions) represented by circles and *transitions* (events) represented by rectangles, and directed *arcs* that connect places to transitions and transitions to places. Places represent the state of the system using *tokens*, and each place holds at most one token at any time.

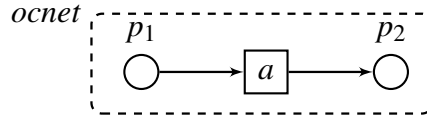


Fig. 2.5 Simple occurrence net.

2.3.2 Communication Structured Occurrence Nets (CSO-nets)

Communication Structured Occurrence Nets (CSO-nets) consist of two or more occurrence nets that are connected using *buffer places*. Buffer places are special nodes that are used to represent and model asynchronous and synchronous communication [36] by linking transitions from different occurrence nets. Figure 2.6 depicts a simple CSO-net, which comprises two occurrence nets: $ocnet_1$ and $ocnet_2$. Asynchronous communication between the two occurrence nets is represented by directed arcs as seen between the transitions c and a linked through a single buffer place q_0 . Thus, transition a will never be executed before c in any execution sequence. A synchronous communication is represented by directed arcs using, for example, a pair of buffer places q_1 and q_2 between transitions b and d . Thus, these two transitions have to be executed simultaneously.

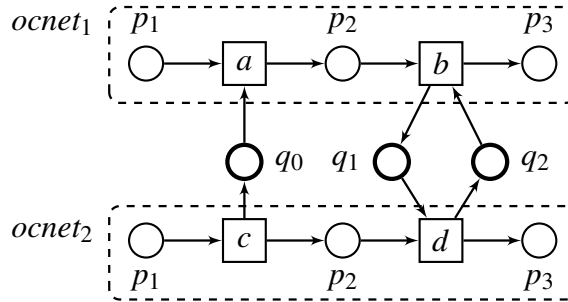


Fig. 2.6 Communication structured occurrence net (CSO-net).

Moreover, we introduce CSA-nets rather than CSO-nets because in future one might want to represent uncertainties in people's statements. Therefore, we have discussed acyclic nets as they are needed to build CSA-nets.

2.3.3 Behavioural Structured Occurrence Nets (BSO-nets)

Behavioural Structured Occurrence Nets (BSO-nets) are used to abstract evolving systems processes. A BSO-net contains two levels: lower level and upper level. The lower level shows the detailed system behaviour. The upper level shows an abstraction of the lower level by hiding details. For example, the upper level can reveal insights into the stages of a system's evolution. The arcs connecting the places across both levels capture the relationships between the two behavioural levels. Figure 2.7 provides an example of BSO-net, where the upper level provides insight as to what may occur in a particular system and its evolution, and the lower level provides insight into the detailed system behaviour. In other words, the upper level can be seen as a behavioural abstraction of the lower level.

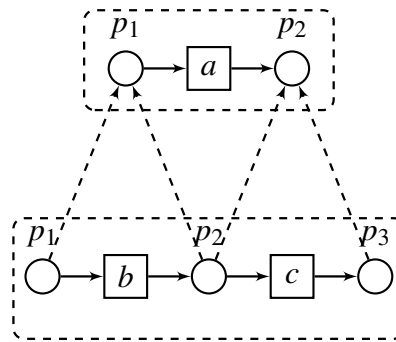


Fig. 2.7 Behavioural structured occurrence net (BSO-net).

2.3.4 Other formal representations

Petri Nets

Petri nets were invented by Carl Adam Petri in his 1962 PhD thesis. They are a mathematical and graphical modelling tool used for describing and analysing systems that exhibit concurrent, asynchronous, and distributed behaviour. Petri nets are structured as bipartite graphs consisting of *places* (represented by circles) and *transitions* (represented by bars or rectangles) connected by directed arcs. The places hold (possibly many) tokens, which resemble the state of the system, and the transitions describe the changes in the system state.

Petri nets are widely used for modelling and analysing processes in a variety of systems. For example, they are used in manufacturing processes and communication protocols, because they can depict dynamic behaviours and system states clearly [43, 18, 12].

Petri nets have been used in modelling cyber-physical systems to simulate and analyse interactions in scenarios involving cyber attacks on critical infrastructure. Their ability to model concurrent, asynchronous, and distributed systems makes them suitable for representing various complex scenarios. Work on the smart grid [17] highlights how Petri nets can effectively model interconnected cyber-physical threats and coordinated attacks. By introducing a hierarchical method, it offers a more effective alternative to traditional attack trees for handling concurrent processes. This method allows different experts to create smaller, detailed models, which are then combined into a scalable model. The approach simplifies the modelling of large systems and enhances the ability to understand and defend against sophisticated attacks.

Furthermore, Stochastic Petri Nets (SPNs) have been used to model information from technical documents. The work in [50] introduces a synergy of methods to extract action associations in technical documents. These methods involve kernel extraction, formal language modelling, SPN mapping, and event representation via SPNs. In this context, a kernel refers to a consecutive set of words based on the *Agent* \rightarrow *Action* \rightarrow *Patient* model, where these elements correspond to the subject, verb, and object of a sentence, respectively. The proposed methodology aims to identify and represent these kernels from text sentences. However, a limitation of this approach is its focus on extracting only complete kernels, meaning that it does not recognise incomplete kernels where either the agent or patient is missing or implied. This could lead to the omission of critical information from sentences where these components are not explicitly stated.

Coloured Petri Nets (CPNs)

Coloured Petri Nets (CPNs) are an extension of classical Petri nets, developed to enhance their modelling power and versatility. Introduced by Kurt Jensen in the early 1980s, CPNs incorporate data types and expressions into the basic Petri net structure to allow for more compact

and readable models. CPNs provides a high-level graphical language that is particularly well-suited for modelling and analysing complex systems where concurrency, synchronisation, and communication are critical elements. The inclusion of *colours* enables the representation of different tokens within the same place, significantly reducing the complexity and size of nets [32, 33].

CPNs have been applied to a variety of domains including workflow management, communication protocols, and manufacturing systems. They offer analysis techniques such as state space exploration, invariant analysis, and performance evaluation. One of the key strengths of CPNs is their ability to integrate the graphical representation of Petri nets with the functional programming capabilities of languages like Standard ML. An extensive discussion on the practical applications and benefits of CPNs in industrial case studies that illustrate the use of CPNs in the design, simulation, and verification of concurrent systems is presented in [34].

Cybercrime is another field where CPNs are employed. A study in [31] presented an approach for modelling and detecting cyber threats using CPNs. The focus is on creating models resembling malware behaviour to aid the detection process. The models were created to resemble malware behaviour by analysing run-time events and system calls aiding the detection process. Additionally, machine learning algorithms were employed to classify software accurately based on information collected at run-time. This approach allows for system monitoring during regular user interactions, enhancing detection capabilities. The use of coloured Petri nets techniques provides a method to detect and prevent cyber attacks targeting computer systems.

Compared to the mentioned tools, SONs have demonstrated their capability to model the behaviour of complex evolving systems comprising different subsystems. Various non-formal methods discussed in the literature can assist in the static visualisation of crime or cybercrime incidents. However, formal methods have the ability to model and visualise these incidents dynamically. Furthermore, the inherently structured nature of SONs helps bridge the gap between unstructured text and formal modelling tools. Their design which involves different acyclic nets each representing a part of a system and connected using buffer places facilitates

this process. This enables SONS to extract and construct models of these incidents with simplicity and accuracy.

2.4 Challenges and Limitations for Crime Modelling in SONS

Despite the successful modelling of accidents [40], crime [27], and cybercrime [6, 3] using SONS, there is still a lack of certain capabilities. Currently, SON models are constructed manually, which requires a modeller to read, digest, and then model cases. This process involves manually defining places, transitions, tokens, and arcs based on the system or scenario being modelled. Users typically input data and specify relationships through graphical user interfaces from textual sources, ensuring that each component accurately represents the system's behaviour and interactions. The SONCraft tool offers features for analysing and simulating SONS, but the creation process remains dependent on human expertise and interpretation, requiring significant time and effort to ensure the models are comprehensive and accurate. This manual approach is usually precise but lacks scalability and efficiency, especially when dealing with complex or large-scale systems.

Another limitation is the challenge of mapping of textual information onto the formal structures required by SONS. A SON is based on a well-defined set of places, transitions, and tokens to model the dynamic behaviour of systems accurately. Converting unstructured information from text into this formalism is complex and can be challenging. Therefore, the integration of SONS and Natural Language Processing (NLP) to identify systems and subsystem components is crucial. For example, the extraction and interpretation of relevant events from large volumes of unstructured text, such as crime reports, can assist investigators analysis a crime. NLP techniques like named entity recognition (NER) and event extraction must be advanced enough to identify specific wording representing actions and entities accurately. However, the complexity of natural language often causes challenges leading to inaccuracies in extracted data.

Chapter 3

Acyclic Nets and Communication Structured Acyclic Nets

3.1 Introduction

This chapter provides some of the definitions and notations adapted from [4]. In particular, it presents the notations and properties involving *acyclic nets* and *communication structured acyclic nets*. The latter can be described as the representations of the interactions between different acyclic nets using buffer places.

This chapter is organised as follows. Section 3.2 introduces the basic notions, semantics, and examples related to acyclic nets. Communication structured acyclic net (CSA-net) basic notions and semantics are introduced in Section 3.3. Section 3.4 concludes this chapter.

3.2 Acyclic nets

In the SON approach, an acyclic net is a bipartite graph with three main components, namely, places, transitions, and arcs. Places represent the local states of a system, transitions represent the changes in the system state, and arcs represent explicitly causality between places and transitions. Acyclicity means there is no loop (cycle) in the graph so that no token visits the same place twice. Intuitively, acyclic nets can be considered non-deterministic meaning

that the path a token takes from the initial node to a terminating node can include choices. Moreover, in *backward deterministic acyclic net*, a token can be traced backward from a terminating place to the initial place by precisely stating which transitions have occurred prior to the latest one.

Graphically (as shown in Figure 3.1), in SONs places are represented by circles, transitions by rectangles, and the flow relation by directed arcs. The flow relation links any two nodes of different kinds, meaning flow links places to transitions and transitions to places.

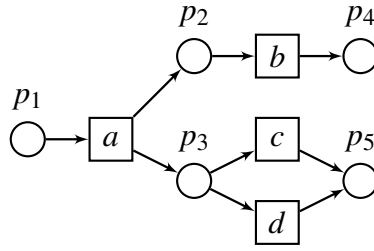


Fig. 3.1 Acyclic net $acnet_1$.

Definition 3.2.1 (acyclic net [4]) An acyclic net is a triple $acnet = (P, T, F)$, where P and T are disjoint finite sets of places (or conditions) and transitions (or events), respectively, and $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation such that:

1. P is nonempty and F is acyclic.
2. For every $t \in T$, there are $p, q \in P$ such that pFt and tFq .

Notation: The set of all acyclic nets is denoted by AN . We also use P_{acnet} , T_{acnet} and F_{acnet} to denote P , T and F , respectively. \diamond

In addition to the acyclicity of F , it is required that each transition t has at least one pre-place p (i.e., pFt) and at least one post-place q (i.e., tFq). Acyclic net can exhibit backward non-determinism (more than one arrow is incoming to a place) as well as forward non-determinism (more than one arrow is outgoing from a place).

Notation 1 (direct precedence in acyclic net) Let $acnet$ be an acyclic net. To indicate relationships between different nodes, for all $x \in P_{acnet} \cup T_{acnet}$ and $X \subseteq P_{acnet} \cup T_{acnet}$, we denote the directly preceding and directly following nodes as follows:

$$\begin{aligned} \bullet x &= \text{pre}_{acnet}(x) \stackrel{\Delta}{=} \{z \mid zF_{acnet}x\}, & \bullet X &= \text{pre}_{acnet}(X) \stackrel{\Delta}{=} \bigcup \{\bullet z \mid z \in X\} \\ x\bullet &= \text{post}_{acnet}(x) \stackrel{\Delta}{=} \{z \mid xF_{acnet}z\}, & X\bullet &= \text{post}_{acnet}(X) \stackrel{\Delta}{=} \bigcup \{z\bullet \mid z \in X\}. \end{aligned}$$

Moreover, the initial and final places are respectively given by:

$$P_{acnet}^{init} \stackrel{\Delta}{=} \{p \in P \mid \bullet p = \emptyset\} \text{ and } P_{acnet}^{fin} \stackrel{\Delta}{=} \{p \in P \mid p\bullet = \emptyset\}.$$

Note that having the notations like $\bullet x$ in addition to (more explicit) $\text{pre}_{acnet}(x)$ helps to keep some of the subsequent formulas short.

Proposition 3.2.1 ([4]) $P_{acnet} = P_{acnet}^{init} \uplus \text{post}_{acnet}(T_{acnet}) = P_{acnet}^{fin} \uplus \text{pre}_{acnet}(T_{acnet})$, for every acyclic net $acnet$.

Example 1. In Figure 3.1, $acnet_1$ is an acyclic net such that $p_3^\bullet = \text{post}_{acnet_1}(p_3) = \{c, d\}$ and $\bullet a = \text{pre}_{acnet_1}(a) = \{p_1\}$. Moreover, $P_{acnet_1}^{init} = \{p_1\}$ and $P_{acnet_1}^{fin} = \{p_4, p_5\}$. \diamond

Definition 3.2.2 (backward deterministic acyclic net) A backward deterministic acyclic net is an acyclic net such that $|\bullet p| \leq 1$, for every place p .

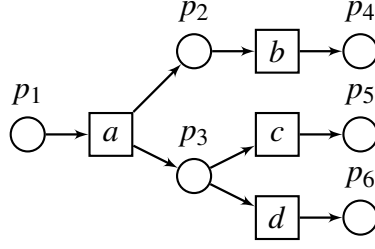
Notation: The set of all backward deterministic acyclic nets is denoted by $BDAN$. \diamond

Example 2. Figure 3.2 shows a backward deterministic acyclic net. \diamond

An acyclic net may exhibit both forward and backward nondeterminism, therefore, it can represent several different possible execution histories. This is no longer the case for the class of acyclic nets defined next, where each net represents a single execution history.

Definition 3.2.3 (occurrence net [4]) An occurrence net is an acyclic net such that $|\bullet p| \leq 1$ and $|p\bullet| \leq 1$, for every place p .

Notation: The set of all occurrence nets is denoted by ON . \diamond

Fig. 3.2 Backward deterministic acyclic net $bdacnet_1$.

In the literature, backward deterministic acyclic nets are sometimes called *nondeterministic occurrence nets* or even *occurrence nets* (in which case occurrence nets as defined above are called *deterministic occurrence nets*).

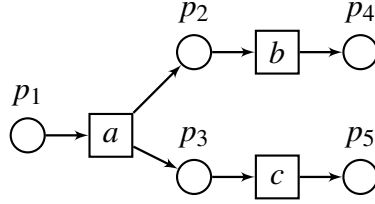


Fig. 3.3 Occurrence net.

An acyclic net may represent several different possible execution histories. The role of the next notion is to identify structurally all such execution histories which can then be inspected, simulated, and analysed.

Scenarios of an acyclic net are acyclic subnets which start at the same initial marking and are both backward and forward deterministic. As a result, each scenario represents a distinct execution history with clearly determined causal relationships. Scenarios are much more abstract than (mixed) step sequences defined later in Section 3.2.2, as one scenario will in general correspond to many step sequences. Below, for two acyclic nets, $acnet$ and $acnet'$, we denote $acnet \sqsubseteq acnet'$ if $P_{acnet}^{init} = P_{acnet'}^{init}$, $T_{acnet} \subseteq T_{acnet'}$,

$$F_{acnet} = \{(x, y) \in F_{acnet'} \mid x, y \in P_{acnet} \cup T_{acnet}\}$$

and, for every $t \in T_{acnet}$:

$$\text{pre}_{acnet}(t) = \text{pre}_{acnet'}(t) \text{ and } \text{post}_{acnet}(t) = \text{post}_{acnet'}(t).$$

Definition 3.2.4 (scenario and maximal scenario of acyclic net) A scenario of an acyclic net $acnet$ is an occurrence net $ocnet$ such that $ocnet \sqsubseteq acnet$. Moreover, $ocnet$ is maximal if there is no scenario $ocnet' \neq ocnet$ such that $ocnet \sqsubseteq ocnet' \sqsubseteq acnet$.

Notation: The set of all scenarios of $acnet$ is denoted by $\text{scenarios}(acnet)$, and the set of all maximal scenarios of $acnet$ is denoted by $\text{maxscenarios}(acnet)$. \diamond

Intuitively, scenarios represent possible executions (concurrent histories), which are both deterministic and consistent with the dependencies implied by the flow relation. Maximal scenarios are complete in the sense that they cannot be extended [4, 8].

Example 3. Figure 3.4 shows two maximal scenarios, $ocnet_1$ and $ocnet_2$, of the acyclic net in Figure 3.1. \diamond

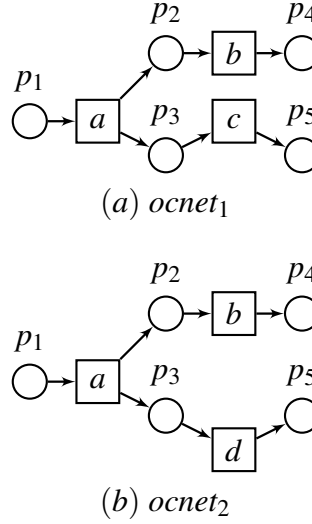


Fig. 3.4 Two maximal scenarios of the acyclic net in Figure 3.1.

3.2.1 Conflict, Concurrency, and Causality

This section introduces structural properties of acyclic nets.

Definition 3.2.5 (structural notions) Let $acnet = (P, T, F)$ be an acyclic net.

1. Two transitions $t \neq u \in T$ are in direct conflict, denoted $t \#_0 u$, if they have a common pre-place, i.e., $\bullet t \cap \bullet u \neq \emptyset$.
2. Two transitions $t \neq u \in T$ are in direct backward conflict if they have a common post-place, i.e., $t^\bullet \cap u^\bullet \neq \emptyset$.
3. Two nodes $x \neq y \in P \cup T$ are concurrent, denoted $x \text{ co } y$, if neither $x \# y$ nor $x F^+ y$ nor $y F^+ x$.
4. $\text{caused}_{acnet}(x) = \{y \in P \cup T \mid x F^+ y\}$ are the elements caused by $x \in P \cup T$. Moreover, $\text{caused}_{acnet}(X) = \bigcup_{x \in X} \text{caused}_{acnet}(x)$, for $X \subseteq P \cup T$. \diamond

That is, conflicts arise when two transitions share a common pre-place (in forward non-determinism), or a common post-place (in backward non-determinism). Note that *forward non-determinism* can only occur when there are multiple output transitions for some place p (i.e., $|p^\bullet| > 1$), and *backward non-determinism* can only occur when there are multiple input transitions for some place p (i.e., $|\bullet p| > 1$). Moreover, concurrency is a result of multiple post-places emerging from a single transition [56].

3.2.2 Step sequence semantics of acyclic nets

This section presents several acyclic net behavioural concepts, including initial marking, enabled step, step and mixed step sequence.

Definition 3.2.6 (step and marking of acyclic net) Let $acnet$ be an acyclic net.

1. $\text{steps}(acnet) \stackrel{\Delta}{=} \{U \subseteq T_{acnet} \mid U \neq \emptyset \wedge \forall t \neq u \in U : \bullet t \cap \bullet u = \emptyset\}$ are the steps.
2. $\text{markings}(acnet) \stackrel{\Delta}{=} \{M \mid M \subseteq P_{acnet}\}$ are the markings.
3. $M_{acnet}^{init} \stackrel{\Delta}{=} P_{acnet}^{init}$ is the initial marking. \diamond

Graphically, markings are indicated by black tokens placed inside the corresponding circles.

Example 4. For the acyclic net $bdacnet_1$ depicted in Figure 3.2, we have $M_{bdacnet_1}^{init} = \{p_1\}$ and $\text{steps}(bdacnet_1) = \{U \subseteq \{a, b, c, d\} \mid U \neq \emptyset \wedge (c \notin U \vee d \notin U)\}$. \diamond

Definition 3.2.7 (enabled and executed step of acyclic net) *Let M be a marking of an acyclic net $acnet$.*

1. $\text{enabled}_{acnet}(M) \triangleq \{U \in \text{steps}(acnet) \mid \bullet U \subseteq M\}$ are the steps enabled at M .
2. A step $U \in \text{enabled}_{acnet}(M)$ can be executed and yield $M' \triangleq (M \cup U^\bullet) \setminus \bullet U$. This is denoted by $M[U]_{acnet} M'$. \diamond

Enabling a step amounts to having all its input places marked. The execution of such a step adds tokens to all its output places and then removes tokens from all its input places.

Note that markings of acyclic nets are ‘safe’ by definition, i.e., a place can only ‘hold’ at most one token. That is, if $M[U]_{acnet} M'$ and $a \neq b \in U$ are such that $p \in a^\bullet \cap b^\bullet$, then p will ‘hold’ only one token in M' .

We use step sequences to capture the behaviour of acyclic nets (and later other nets).

Definition 3.2.8 ((mixed) step sequence of acyclic net) *Let M_0, M_1, \dots, M_k ($k \geq 0$) be markings and U_1, \dots, U_k be steps of an acyclic net $acnet$ such that $M_{acnet}^{init} = M_0$ and we have $M_{i-1}[U_i]_{acnet} M_i$, for every $1 \leq i \leq k$. Then*

1. $\mu = M_0 U_1 M_1 \dots M_{k-1} U_k M_k$ is a mixed step sequence from M_0 to M_k .
2. $\sigma = U_1 \dots U_k$ is a step sequence from M_0 to M_k .

The above two notions are denoted by $M_0[\mu]_{acnet} M_k$ and $M_0[\sigma]_{acnet} M_k$, respectively. Moreover, $M_0[\sigma]_{acnet}$ denotes that σ is a step sequence enabled at M_0 , and $M_0[\]_{acnet} M_k$ denotes that M_k is reachable from M_0 . \diamond

Note that if $k = 0$ then $\mu = M_0$ and the corresponding step sequence σ is the *empty* sequence denoted by λ .

The previous definition considers the start point of an execution as an arbitrary marking. The next definition will introduce various notions related to behaviour, and assumes that the starting point of system executions is the default initial marking.

Definition 3.2.9 (behaviour of acyclic net) *The following sets capture various behavioural notions related to step sequences and reachable markings of an acyclic net $acnet$.*

1. $sseq(acnet) \triangleq \{\sigma \mid M_{acnet}^{init}[\sigma]_{acnet} M\}$ step sequences.
2. $mixsseq(acnet) \triangleq \{\mu \mid M_{acnet}^{init}[\mu]_{acnet} M\}$ mixed step sequences.
3. $maxsseq(acnet) \triangleq \{\sigma \in sseq(acnet) \mid \neg \exists U : \sigma U \in sseq(acnet)\}$
maximal step sequences.
4. $maxmixsseq(acnet) \triangleq \{\mu \in mixsseq(acnet) \mid \neg \exists U, M : \mu U M \in mixsseq(acnet)\}$
maximal mixed step sequences.
5. $reachable(acnet) \triangleq \{M \mid M_{acnet}^{init}[\]_{acnet} M\}$ reachable markings.
6. $finreachable(acnet) \triangleq \{M \mid \exists \sigma \in maxsseq(acnet) : M_{acnet}^{init}[\sigma]_{acnet} M\}$
final reachable markings.
7. $fseq(acnet) \triangleq \{U_1 \dots U_k \in sseq(acnet) \mid k \geq 1 \implies |U_1| = \dots = |U_k| = 1\}$
firing sequences.

Notation: We can treat individual transitions as singleton steps; e.g., a step sequence $\{t\}\{u\}\{w, v\}\{z\}$ can be denoted by $tu\{w, v\}z$. \diamond

Example 5. The following hold for the acyclic net $bdacnet_1$ in Figure 3.5.

1. $sseq(bdacnet_1) = \{\lambda, a, ab, ac, ad, abc, acb, abd, adb, a\{b, c\}, a\{b, d\}\}$.
2. $mixsseq(bdacnet_1) = \{\{p_1\}, \{p_1\}a\{p_2, p_3\}, \{p_1\}a\{p_2, p_3\}\{b, c\}\{p_4, p_5\}, \dots\}$.

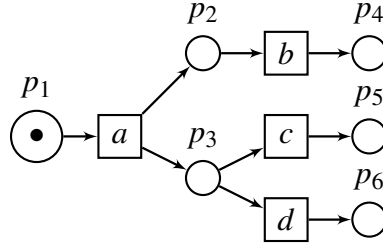


Fig. 3.5 Acyclic net $bdacnet_1$ of Figure 3.2 with the initial marking indicated.

3. $\text{maxsseq}(bdacnet_1) = \{abc, acb, a\{b, c\}, abd, adb, a\{b, d\}\}.$
4. $\text{maxmixsseq}(bdacnet_1) = \{\{p_1\}a\{p_2, p_3\}\{b, c\}\{p_4, p_5\}, \dots\}.$
5. $\text{reachable}(bdacnet_1) = \{\{p_1\}, \{p_2, p_3\}, \{p_2, p_5\}, \{p_2, p_6\}, \{p_4, p_3\}, \dots\}.$
6. $\text{finreachable}(bdacnet_1) = \{\{p_4, p_5\}, \{p_4, p_6\}\}.$
7. $\text{fseq}(bdacnet_1) = \{\lambda, a, ab, ac, ad, abc, acb, abd, adb\}.$ ◇

Note that there is a straightforward way of introducing causality in acyclic nets by looking at their step sequences. More precisely, let t and u be two transitions of an acyclic net $acnet$. Then u is a *cause* of t if, for every step sequence $\sigma U \in \text{sseq}(acnet)$ such that $t \in U$, it is the case that u occurs in σ .

3.2.3 Well-formed acyclic nets

A basic requirement of acyclic nets is well-formedness. Its essence is to ensure an unambiguous representation of causality in behaviours they represent. The definition of a well-formed acyclic net is derived from the notion of a well-formed step sequence.

Definition 3.2.10 (well-formed step sequence of acyclic net) *A step sequence $U_1 \dots U_k$ of an acyclic net is well-formed if the following hold:*

1. $t^\bullet \cap u^\bullet = \emptyset$, for every $1 \leq i \leq k$ and all $t \neq u \in U_i$.
2. $U_i^\bullet \cap U_j^\bullet = \emptyset$, for all $1 \leq i < j \leq k$. ◇

In a well-formed step sequence of an acyclic net, no place receives a token more than once. It then follows, e.g., that in such a step sequence no place is a pre-place of an executed step more than once, the order of execution of transitions does not influence the resulting marking, and each step sequence can be sequentialised to a firing sequence.

Definition 3.2.11 (well-formed acyclic net) *An acyclic net is well-formed if each transition occurs in at least one step sequence and all the step sequences are well-formed.*

Notation: The set of all well-formed acyclic nets is denoted by WFAN. ◇

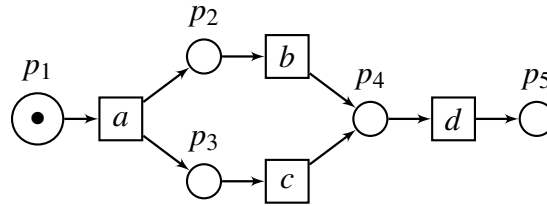


Fig. 3.6 Non-well-formed acyclic net.

Example 6. Figure 3.5 shows a well-formed acyclic net, but the acyclic net shown in Figure 3.6 is non-well-formed acyclic net since the step sequence $\{a\}\{b, c\}\{d\}$ is not well-formed. Note that in such a case it is not possible to determine which transition (b or c) caused d to fire. ◇

As the next result shows, checking that an acyclic net is well-formed can be done by looking at its firing sequences.

Proposition 3.2.2 ([4]) *An acyclic net $acnet$ is well-formed iff each transition occurs in at least one firing sequence and all the firing sequences are well-formed.*

3.3 Communication structured acyclic nets (CSA-nets)

A Communication Structured Acyclic Net (CSA-ne) is a ‘database’ consisting of a number of disjoint acyclic nets which communicate through special buffer places. These buffer places allow synchronous or asynchronous transfer of tokens. CSA-nets can exhibit backward and forward non-determinism.

Definition 3.3.1 (CSA-net) A communication structured acyclic net (or CSA-net) is a tuple $csan = (acnet_1, \dots, acnet_n, Q, W)$ ($n \geq 1$) such that:

1. $acnet_1, \dots, acnet_n$ are well-formed acyclic nets with disjoint sets of nodes (i.e., places and transitions). We also denote:

$$\begin{aligned} P_{csan} &\triangleq P_{acnet_1} \cup \dots \cup P_{acnet_n} & Q_{csan} &\triangleq Q \\ T_{csan} &\triangleq T_{acnet_1} \cup \dots \cup T_{acnet_n} & W_{csan} &\triangleq W \\ F_{csan} &\triangleq F_{acnet_1} \cup \dots \cup F_{acnet_n} & net_{csan,i} &\triangleq acnet_i \quad (\text{for } 1 \leq i \leq n). \end{aligned}$$

2. Q is a finite set of buffer places disjoint from $P_{csan} \cup T_{csan}$.
3. $W \subseteq (Q \times T_{csan}) \cup (T_{csan} \times Q)$ is a set of arcs.
4. For every buffer place q :
 - (a) There is at least one transition t such that tWq .
 - (b) If tWq and qWu then transitions t and u belong to different component acyclic nets.

Notation: The set of all CSA-nets is denoted by $CSAN$. ◇

That is, in addition to requiring the disjointness of the component acyclic nets and the buffer places, it is also required that buffer places pass tokens between different acyclic nets.

Notation 2 (direct precedence in CSA-net) Let $csan = (acnet_1, \dots, acnet_n, Q, W)$ be a CSA-net, $x \in P_{csan} \cup T_{csan} \cup Q_{csan}$, and $X \subseteq P_{csan} \cup T_{csan} \cup Q_{csan}$. Then

$$\begin{aligned} \text{pre}_{csan}(x) &\triangleq \{y \mid yF_{csan}x \vee yW_{csan}x\}, & \text{pre}_{csan}(X) &\triangleq \bigcup \{\text{pre}_{csan}(z) \mid z \in X\} \\ \text{post}_{csan}(x) &\triangleq \{y \mid xF_{csan}y \vee xW_{csan}y\}, & \text{post}_{csan}(X) &\triangleq \bigcup \{\text{post}_{csan}(z) \mid z \in X\} \end{aligned}$$

denote direct predecessors and successors of x and X , respectively. Moreover,

$$P_{csan}^{init} \triangleq P_{acnet_1}^{init} \cup \dots \cup P_{acnet_n}^{init} \quad \text{and} \quad P_{csan}^{fin} \triangleq P_{acnet_1}^{fin} \cup \dots \cup P_{acnet_n}^{fin} \cup (Q_{csan} \setminus \text{pre}_{csan}(T_{csan}))$$

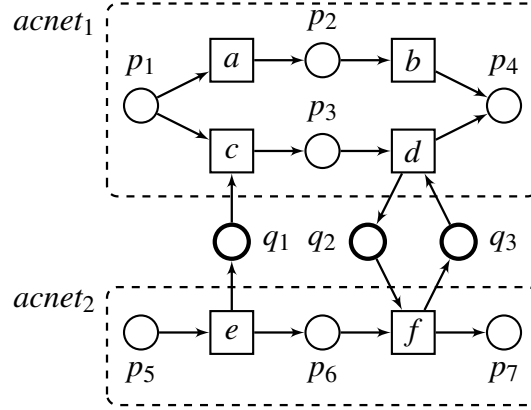


Fig. 3.7 Communication structured acyclic net *csan*.

are the initial and final places of *csan*, respectively. \diamond

In CSA-nets, the buffer places can be considered as intermediary points that transfer tokens between different component acyclic nets. By holding tokens, buffer places facilitate the flow of control between individual acyclic nets.

Proposition 3.3.1 ([4]) For every CSA-net *csan*,

$$P_{acnet} \uplus Q_{acnet} = P_{acnet}^{init} \uplus \text{post}_{csan}(T_{csan}) = P_{acnet}^{fin} \uplus \text{pre}_{acnet}(T_{acnet}).$$

Example 7. Figure 3.7 shows a CSA-net which is composed of two acyclic nets such that $\text{post}_{csan}(p_1) = \{a, c\}$, $\text{pre}_{csan}(p_4) = \{b, d\}$, and $\text{post}_{csan}(e) = \{q_1, p_6\}$. Moreover, transitions *e* and *c* are communicating *asynchronously*, so they can be executed either at the same time, or *e* then *c*. However, *c* cannot be executed before *e* in *asynchronous* communication. The relation between *e* and *c* can be captured by the presence of $q_1 \in \text{post}_{csan}(e) \cap \text{pre}_{csan}(c)$. Whereas, *d* and *f* must be executed simultaneously because they are involved in *synchronous* communication. \diamond

Note that CSA-net can exhibit backward and forward non-determinism. Additionally, they can contain synchronous cycles involving only buffer places.

Definition 3.3.2 (CSO-net) A communication structured occurrence net (or CSO-net) is $csn \in CSAN$ such that:

1. The component acyclic nets belong to ON.
2. $|\text{pre}_{cson}(q)| = 1$ and $|\text{post}_{cson}(q)| \leq 1$, for every $q \in Q_{cson}$.
3. No place in P_{cson} belongs to a cycle in the graph of $F_{cson} \cup W_{cson}$.

Notation: The set of all CSO-nets is denoted by CSON. ◇

A CSO-net provides a full record of all causal dependencies between the events involved in a single ‘causal history’. In terms of behaviour defined later, CSO-nets exhibit both backward determinism and forward determinism. CSA-nets are also considered, however, with only forward nondeterminism.

Example 8. Figure 3.8 depicts a CSO-net. ◇

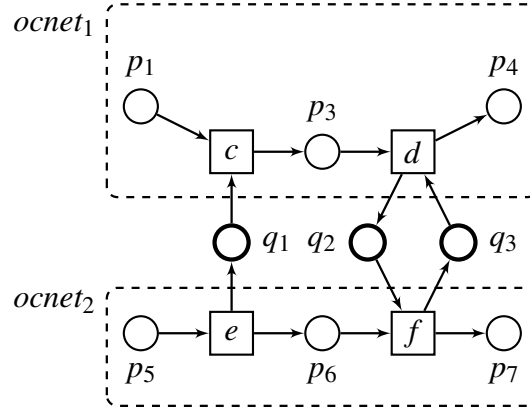


Fig. 3.8 Communication structured occurrence net *cson*.

Definition 3.3.3 (BDCSA-net) A backward deterministic communication structured acyclic net (or BDCSA-net) is $bdcsan \in CSAN$ such that:

1. The component acyclic nets belong to BDAN.
2. $|\text{pre}_{bdcsan}(q)| = 1$, for every $q \in Q_{bdcsan}$.

Notation: The set of all BDCSA-nets is denoted by BDCSAN. ◇

BDCSA-net is a CSA-net providing full and unambiguous record of backward causal dependencies. BDCSA-nets exhibit backward determinism, but forward determinism is not required.

Scenarios for CSA-nets can be defined similarly as for acyclic nets.

Below, for two CSA-nets,

$$csan = (acnet_1, \dots, acnet_n, Q, W) \text{ and } csan' = (acnet'_1, \dots, acnet'_n, Q', W'),$$

we denote $csan \sqsubseteq csan'$ if

1. $acnet_i \sqsubseteq acnet'_i$, for every $1 \leq i \leq n$.
2. $Q \subseteq Q'$.
3. $\text{pre}_{csan}(t) = \text{pre}_{csan'}(t)$ and $\text{post}_{csan}(t) = \text{post}_{csan'}(t)$, for every $t \in T_{csan}$.

Definition 3.3.4 (scenario and maximal scenario of CSA-net) *Let $csan$ be a CSA-net. A scenario of $csan$ is a CSO-net $cson$ such that $cson \sqsubseteq csan$. Moreover, $cson$ is maximal if there is no scenario $cson'$ of $csan$ such that $cson' \neq cson$ and $cson \sqsubseteq cson'$.*

Notation: The set of all scenarios of $csan$ is denoted by $\text{scenarios}(csan)$, and the set of all maximal scenarios of $csan$ is denoted by $\text{maxscenarios}(csan)$. \diamond

Intuitively, scenarios represent possible executions (concurrent histories) consistent with the dependencies imposed by the flow relation. This, in particular, means that all the transitions have been executed and places marked at some point during an execution. Maximal scenarios are complete in the sense that they cannot be extended any further.

Although scenarios represent behaviours of CSA-nets in a different way than step sequences, markings and related notions, there is a close relationship between these two approaches.

Proposition 3.3.2 ([4]) $\text{maxscenarios}(cson) = \{cson\}$, for every CSO-net $cson$.

That is, an occurrence net has only one maximal scenario (itself), which can be understood as a precise depiction of a single execution history.

Example 9. A maximal scenario for the CSA-net in Figure 3.8 is shown in Figure 3.7. \diamond

3.3.1 Step sequence semantics of CSA-nets

This section will introduce notions related to the behaviour of CSA-nets.

Definition 3.3.5 (step and marking of CSA-net) *Let $csan$ be a CSA-net.*

1. $\text{steps}(csan) \triangleq \{U \subseteq T_{csan} \mid U \neq \emptyset \wedge \forall t \neq u \in U : \text{pre}_{csan}(t) \cap \text{pre}_{csan}(u) = \emptyset\}$ are the steps.
2. $\text{markings}(csan) \triangleq \{M \mid M \subseteq P_{csan} \cup Q_{csan}\}$ are the markings.
3. $M_{csan}^{init} \triangleq P_{csan}^{init}$ is the initial marking. ◇

That is, in CSA-net steps can involve events from one or more acyclic nets, provided that $\text{pre}_{csan}(t) \cap \text{pre}_{csan}(u) = \emptyset$ for every $t \neq u$. Furthermore, markings in CSA-net can consist of both places and buffer places. The initial marking in a CSA-net is made up of the initial markings of all acyclic nets participating in the CSA-net. Note that buffer places are excluded from initial marking.

Example 10. $\text{steps}(csan_1) = \{U \subseteq \{a, b, c, d, e, f\} \mid U \neq \emptyset \wedge (a \notin U \vee c \notin U)\}$, for the CSA-net in Figure 3.7. ◇

Definition 3.3.6 (enabled and executed step of CSA-net) *Let M be a marking of a CSA-net $csan$.*

1. $\text{enabled}_{csan}(M) \triangleq \{U \in \text{steps}(csan) \mid \text{pre}_{csan}(U) \subseteq M \cup (\text{post}_{csan}(U) \cap Q)\}$ are the steps enabled at M .
2. A step $U \in \text{enabled}_{csan}(M)$ can be executed yielding a new marking

$$M' \triangleq (M \cup \text{post}_{csan}(U)) \setminus \text{pre}_{csan}(U).$$

This is denoted by $M[U]_{csan} M'$. ◇

Enabling a step in a CSA-net amounts to having all input places belonging to the component acyclic nets present/marked in a global state. Moreover, if an input buffer place is not present, then it must be an output place of a transition belonging to the step. Such a mechanism allows one to synchronise transitions coming from different component acyclic nets.

The dynamic behaviour of *csan* is captured by step sequences and mixed step sequences, as follows.

Definition 3.3.7 ((mixed) step sequence of CSA-net) *Let M_0, \dots, M_k ($k \geq 0$) be markings and U_1, \dots, U_k be steps of a CSA-net *csan* such that $M_{i-1}[U_i]_{csan} M_i$, for every $1 \leq i \leq k$.*

1. $\mu = M_0 U_1 M_1 \dots M_{k-1} U_k M_k$ is a mixed step sequence from M_0 to M_k .
2. $\sigma = U_1 \dots U_k$ is a step sequence from M_0 to M_k .

The above two notions are denoted by $M_0[\mu]_{csan} M_k$ and $M_0[\sigma]_{csan} M_k$, respectively. Moreover, $M_0[\sigma]_{csan}$ denotes that σ is a step sequence enabled M_0 , and $M_0[\]_{csan} M_k$ denotes that M_k is reachable from M_0 . \diamond

Note that if $k = 0$ then $\mu = M_0$ and the corresponding step sequence σ is the *empty* sequence denoted by λ .

Definition 3.3.8 (behaviour of CSA-net) *The following sets capture various behavioural notions related to step sequences and reachable markings of a CSA-net *csan*. This definition assumes that the starting point is the default initial marking.*

1. $sseq(csan) \triangleq \{\sigma \mid M_{csan}^{init}[\sigma]_{csan} M\}$ step sequences.
2. $mixsseq(csan) \triangleq \{\mu \mid M_{csan}^{init}[\mu]_{csan} M\}$ mixed step sequences.
3. $maxsseq(csan) \triangleq \{\sigma \in sseq(csan) \mid \neg \exists U : \sigma U \in sseq(csan)\}$ maximal step sequences.
4. $maxmixsseq(csan) \triangleq \{\mu \in mixsseq(csan) \mid \neg \exists U, M : \mu U M \in mixsseq(csan)\}$ maximal mixed step sequences.

5. $\text{reachable}(csan) \triangleq \{M \mid M_{csan}^{init} \rangle_{csan} M\}$ reachable markings.
6. $\text{finreachable}(csan) \triangleq \{M \mid \exists \sigma \in \text{maxsseq}(csan) : M_{csan}^{init}[\sigma]_{csan} M\}$ final reachable markings.

Example 11. The following hold for the CSA-net in Figure 3.7.

1. $\text{sseq}(csan_1) = \{\lambda, \{a, e\}, \{a, e\}b, \dots\}$.
2. $\text{mixsseq}(csan_1) = \{\{p_1, p_5\}, \{p_1, p_5\}\{a, e\}\{p_2, p_6, q_1\}, \dots\}$.
3. $\text{maxsseq}(csan_1) = \{ab, aeb, abe, \{a, e\}b, a\{b, e\}, ec\{d, f\}, \{e, c\}\{d, f\}\}$.
4. $\text{maxmixsseq}(csan_1) = \{\{p_1, p_5\}\{a, e\}\{p_2, p_6, q_1\}b\{p_4, p_6, q_1\}, \dots\}$.
5. $\text{reachable}(csan_1) = \{\{p_1, p_5\}, \{p_2, p_6, q_1\}, \dots\}$.
6. $\text{finreachable}(csan_1) = \{\{p_4, p_6, q_1\}, \{p_4, p_7\}\}$. \diamond

3.3.2 Well-formed CSA-nets

A fundamental requirement for CSA-net is well-formedness, which essentially guarantees a clear representation of causality in the behaviours they represent. As is for acyclic nets, the concept of a well-formed CSA-net is derived from the idea of a well-formed step sequence.

Definition 3.3.9 (well-formed step sequence of CSA-net) A step sequence $U_1 \dots U_k$ of a CSA-net $csan$ is well-formed if the following hold:

1. $\text{post}_{csan}(t) \cap \text{post}_{csan}(u) = \emptyset$, for every $1 \leq i \leq k$ and all $t \neq u \in U_i$.
2. $\text{post}_{csan}(U_i) \cap \text{post}_{csan}(U_j) = \emptyset$, for all $1 \leq i < j \leq k$. \diamond

Intuitively, in CSA-net, a well-formed step sequence means that no place or buffer place is filled by a token more than once in any given step sequence. In other words, no transition is executed and no token is consumed more than once. The order of execution of transitions does not influence the resulting marking.

Definition 3.3.10 (well-formed CSA-net) *A CSA-net is well-formed if each transition occurs in at least one step sequence and all step sequences are well-formed. The set of all well-formed CSA-nets is denoted by WFCSAN.* \diamond

By satisfying backward and forward determinism, CSO-nets enjoy some essential behavioural properties ‘for free’.

Proposition 3.3.3 ([4]) *Each CSO-net is well-formed.*

Proposition 3.3.4 ([4]) *A CSA-net is well-formed iff each transition occurs in at least one scenario, and each step sequence is a step sequence of at least one scenario.*

Proposition 3.3.5 ([4]) *Step sequences of a well-formed CSA-net do not contain multiple occurrences of transitions.*

Each step sequence $\sigma = U_1 \dots U_k$ of a well-formed CSA-net $csan$ induces a scenario $\text{scenario}_{csan}(\sigma) = \text{scenario}_{csan}(U_1 \cup \dots \cup U_k)$ such that $\sigma \in \text{maxsseq}(\text{scenario}_{csan}(\sigma))$. Thus, in a well-formed CSA-net, different step sequences could generate the same scenario. However, each scenario is generated by at least one step sequence. Additionally, two maximal step sequences could generate the same scenario only if their executed transitions are identical.

Now, the next chapter (Chapter 4) will explore methods for mapping and extracting SON components, in particular, *transition* and *acyclic nets* nodes from unstructured text. This is to facilitate the modelling process for complex systems from textual sources.

3.4 Conclusion

This chapter introduced the fundamental concepts, definitions, and examples related to acyclic nets and CSA-nets. It covered their structure, semantics, step sequences, and well-formedness. Note that CSA-nets use both asynchronous and synchronous communication between different subsystems via buffer places. These features enable CSA-nets to model and visualise complex evolving systems that exhibit synchronisation and concurrency.

Chapter 4

SONs and NLP Mapping and Integration

Data visualisation is the process of transforming data into a visual representation to make it easier for humans to comprehend and derive knowledge. By offering detailed overview of events, such as crime incidents, data visualisation technologies have the potential to assist investigators in analysing crimes. This chapter discusses how models are built manually. Logically, models produced manually may not look the same, depending on the length, complexity, and modeller's understanding of the text. Therefore, mapping and linking natural language to formal representations are crucial for automatic extraction to construct SON models. Consequently, mapping text into SON components is proposed and examined experimentally for automatic extraction.

4.1 Introduction

To identify and categorise events and the individuals or entities involved is a challenging task. Especially when dealing with events from unstructured text that includes several entity participants. Defining what constitutes an event for SON modelling is particularly challenging. For instance, a burglary is a crime event, but with SONs, we are interested in all the specific occurrences during that crime. This includes the detailed movements and actions of the criminal and any other possible entities involved. In SONs, *transitions* represent individual actions that build the causal relationship between crime events. For example, in a burglary,

a felon drove to a store, broke the door, and entered the store. Modelling this in SON will illustrate the actions (transitions) that the felon took (*drove, broke, entered*) and allow for the analysis of the causal relationships.

The challenge for automatic extraction lies in the process of identifying SON components from unstructured text, specifically how to extract *transitions* and *acyclic nets* from such text. Additionally, identifying transitions poses a significant difficulty. While it may be intuitive for a human reader to recognise events that can be defined as SON transition, this task becomes complex when dealing with structurally and linguistically varied textual sources.

This chapter presents the methodology employed to identify and map textual words using Natural Language Processing (NLP) to SON components. This methodology involves the identification, extraction, and proposal of construction methods for formal nets. The chapter is structured as follows. Section 4.2 explains the methodology. Section 4.4 discusses the analyses the manually created models. Section 4.5 analyses and proposes a mapping mechanism between natural language and formal models. Section 4.6 introduces and proposes an automatic extraction approach. Section 4.7 discusses and compares the results and shortcomings of both manual and automatic extraction. Section 4.8 provides an overview of some background and related work for SON and NLP. Finally, Section 4.9 concludes the chapter and provides an overview of future work.

4.2 Methodology

The extraction methodology we will apply is designed to gather and analyse information from text. That is by focusing on deriving insights from the construction of manual models as shown in Figure 4.1. The process begins with studying and understanding how models are built through manual effort by humans (Figure 4.1(a)). This initial phase involves close observation and study of manual modelling processes to identify the key steps and decisions involved in constructing these models (Figure 4.1(b)). Then, based on the analysis of these manually produced models, a mapping between natural language and SONs is defined as shown in Figure 4.1(c). This step is critical in bridging the gap between human-driven

modelling and automated processes, ensuring that the extracted information mimics the manual method we observed.

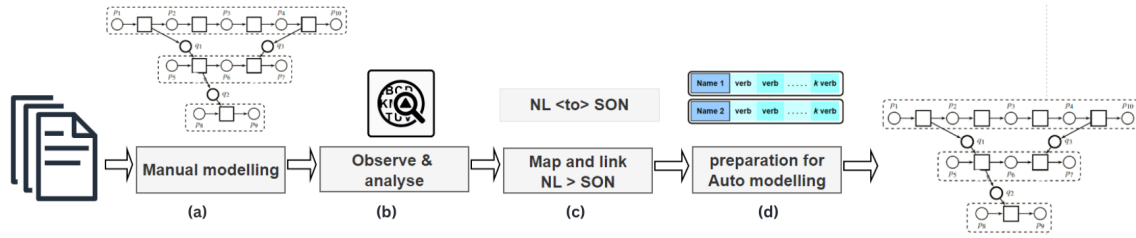


Fig. 4.1 Methodology used to build the automatic extractor.

Following these steps, we will leverage NLP techniques to extract relevant information based on the predefined *acyclic nets* and *transitions* mapping. NLP helps to automate the extraction process by preprocessing and transforming raw text data into a suitable format. By applying NLP models, labels (or tags) are assigned to words in the text by the different models preparing it for further analysis and extraction processes as shown in Figure 4.1(d).

Finally, various extraction methods are proposed. The extracted information is used to develop automatic modelling approaches. These approaches will allow us to input textual data and automatically extract the model components to construct SON models. This step will significantly enhance the efficiency and consistency of model construction. It will ensure that the models are built based on well-studied, predefined rules. This methodology not only simplifies the process of model building but also ensures that the models generated are consistent with manually derived standards.

To illustrate how text is represented in SONs, consider the phrase: “Allen saw ...”. Figure 4.2 shows a representative *occurrence net* that can be generated from this phrase. This figure helps to visualise how individual entities and events are mapped into the structure of a SON by building transitions and occurrence nets.

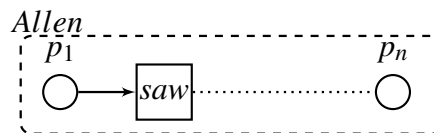


Fig. 4.2 Simple occurrence net.

4.2.1 Terminology

Table 4.1: Key terminology and their definitions used in this chapter.

| | |
|--|---|
| Tokenisation | The process of splitting words in a sentence into a series of tokens. |
| Part-of-Speech (POS) model | Assigns part of speech tags to tokens. |
| Dependency Parsing model | Links tokens (words) as they appear grammatically in the sentence and assigns them parsing tags that show their relationships, i.e., subject, object, conjunction, etc. |
| Entity | Is a proper noun that refers to a distinct real-world object or entity, such as a person, organisation, or location. |
| Named Entity Recogniser (NER) model | Is a model where entities are identified within a text and tagged with name type labels. |
| Coreferencing | Resolve pronouns and mentions to the original names they refer to. |
| ROOT VERBS label | Main verbs appear in sentences and are predicated by the PARSER as the primary word from which the sentence is parsed. |
| Occurrence Net (ON) | An <i>acyclic net</i> that provides a full and unambiguous record of all causal dependencies between its events. |
| COMMON VERBS | A list of the most frequently occurring verbs compiled from crime stories in [15]. |
| SHARED_VERBS | Similar verbs appearing in different <i>acyclic nets</i> . |
| PRE-TRAINED_MODEL | In NLP tools, a pre-trained model is a model that has already gone through the training process allowing it to have a baseline understanding of natural language. |

4.3 Dataset

The dataset used in this section focuses on identifying the most common verbs in crime stories. The stories were obtained from [15] which are provided by "The Violence Policy Center" and details incidents of homicide shootings. The dataset includes key information such as the location of the incidents, the individuals involved, the type of weapons used, and the context or circumstances surrounding each shooting. We have used this dataset to obtain the most common crime verbs which we then used to propose Method2 presented in

Section 4.6.1. This method used these verbs to identify transitions for a more accurate verb identification.

4.4 Manual modelling

Crime can be conceptualised as a complex evolving system involving various interacting participants, such as individuals, weapons, law enforcement, or other entities. Understanding and examining the behaviour within these systems is crucial to assist investigators in decision-making. Investigators typically rely on a variety of sources, including written police reports and witness statements. CSA-nets provide a distinctive method for analysing such types of crime by representing events and chain of events to uncover causal relationships between them. Also, CSA-nets can assist in better comprehension and visualisation of events. This section discusses the manual construction of *acyclic nets* and CSA-nets using identified useful information from written sources, and represent events through CSA-nets.

4.4.1 Manual modelling experiment

The traditional approach for modelling involves a manual, step-by-step process through which a modeller constructs a SON. This requires a modeller to first read the text, understand it, and then create the models in accordance with the text. This is a laborious and time consuming process. Consequently, there is an incentive to develop an automatic approach to automate these steps. Prior to work reported in this thesis, the SON modelling approach lacked the ability to extract information automatically from unstructured written sources and reports. Therefore, initially an experiment was conducted based on a short fragment of a crime story shown in Figure 4.3. The three modellers who contributed to this work are researchers with knowledge and expertise in building SON models. They are members of the SON research group at Newcastle University. The modellers were asked to extract crime data and represent them as a SON model. This is to observe the style of the human extraction and modelling processes in order to assess the produced models and therefore derive knowledge.

Figure 4.4 shows the result of the extraction and representation of information by three expert SON users.

ROSS AND SPICER HAD PLAYED A GAME OF DICE. ROSS LOST AND HE WAS UPSET,
ACCORDING TO POLICE RECORDS. THE NEXT DAY HE, WEARING BODY ARMOR,
RETURNED TO SPICER'S HOME AND FATALLY SHOT SPICER

Fig. 4.3 A short fragment of a crime story

The manual experiment aims to construct SON models by hand, and then study what information was extracted from the unstructured data. This is to build a base understanding of how model components get selected from text to build a bridge between natural language and formal representation. We now discuss the mapping we have proposed based on the analysis and observations of the manual models.

4.5 Analysis and Mapping to SONs

The analysis and mapping of formal models built from textual resources are important steps to understating the linking process. The modelling process begins with extracting information from textual sources, which is then used to build a SON manually. This section discusses the analyses and mapping back to the textual components to ensure a thorough understanding of the structuring relationships. This manual analysis and mapping are essential to identify patterns and insights that inform the development of automatic approaches. The aim is to leverage this understanding to propose automated methods for transforming text into SONs efficiently.

4.5.1 Evaluation of Human Modelling

From the manual modelling experiments, we found that users extracted the following *transitions* from the sentences: *play*, *lost*, *leaves*, *wearing*, *goes/returned*, and *shoot*. Nevertheless, not all users agreed on the exact model design and in terms of wording. For example, MODELLER1 extracted only three transitions (*play*, *lost*, and *shot*), whereas MODELLER3

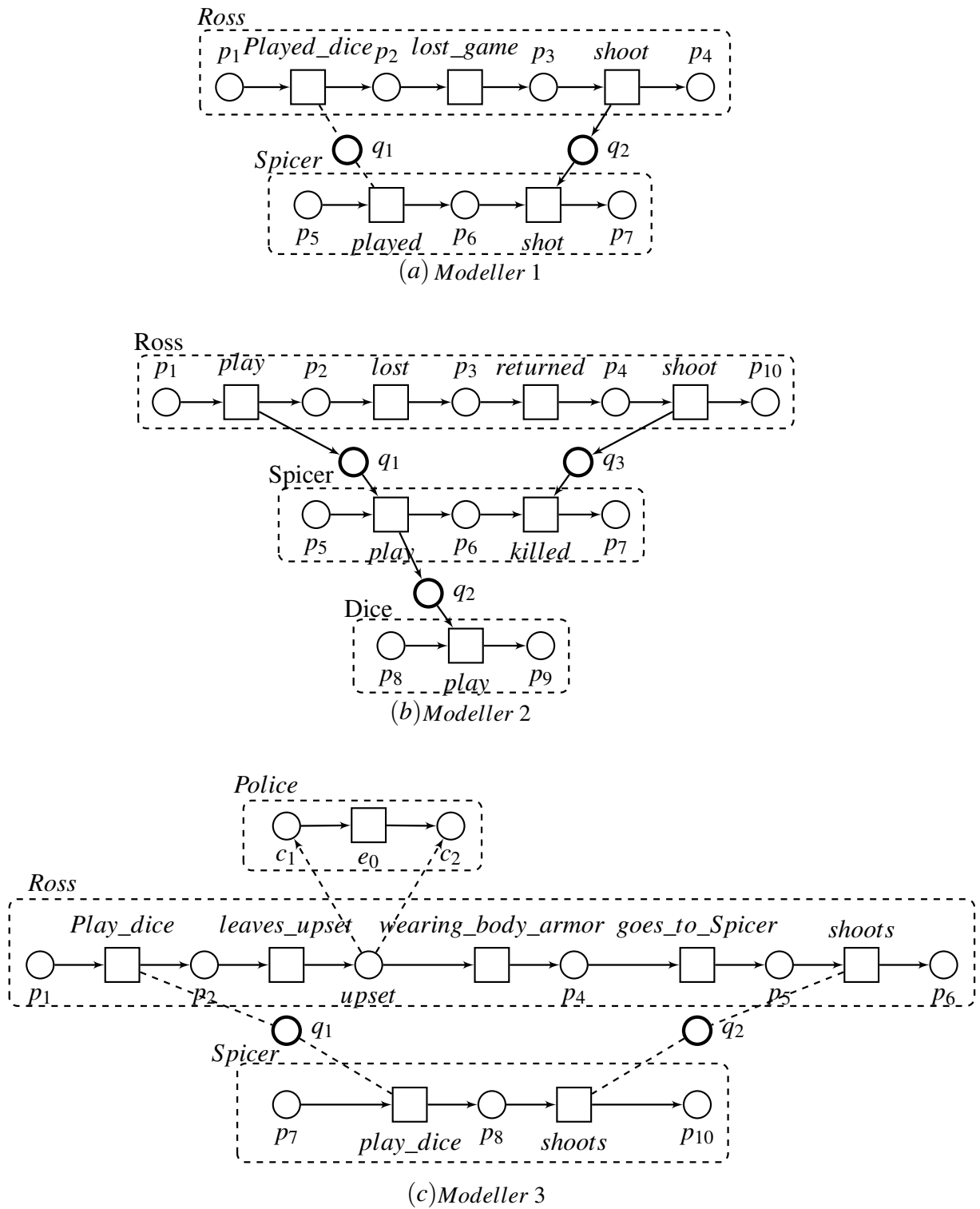


Fig. 4.4 Manual modelling for text in Figure 4.3 done by SON expert users.

extracted five transitions including transitions that were not explicitly mentioned in the sentences. More precisely, MODELLER3 added the words (*leaves* and *goes*) which are not present in the text, but can be explained by the human ability to comprehend and express events differently.

Furthermore, all modellers identified proper nouns represented by *acyclic nets*. MODELLER1 identified ROSS, and SPICER to represent in acyclic nets. MODELLER2 identified ROSS, SPICER, and DICE, and finally MODELLER3 selected ROSS, SPICER, and POLICE.

Despite minor differences in representation (e.g., the amount of information provided by different modellers), the experiment revealed semantically similar models. In comparison to other modellers, MODELLER1 extracted just enough data. MODELLER1 extracted and presented the crime in a very straightforward manner by extracting two acyclic nets and five transitions. MODELLER2, on the other hand, added one additional acyclic net, DICE, that the other two modellers did not, instead inserting DICE into the play transitions, *played_dice* and *play_dice*, by MODELLER1 and MODELLER3, respectively.

4.5.2 Rules for SON and Natural Language Mapping

From the manual experimentation analyses we observed that all modellers represented proper nouns with acyclic nets. A proper noun can be identified and tagged by NLP tools as an *entity*. Therefore, we represent the entities with acyclic nets. We also noticed that most transitions represented verbs. Therefore, the following rules have been established to guide the linking of natural language with SONs.

Rule 1: ENTITIES in text are represented by SON acyclic nets. For example, in Figure 4.4 MODELLER1 used the entity name ROSS for acyclic net in SON.

Rule 2: VERBS are represented by transitions within *acyclic nets*. For example, modellers in Figure 4.4 selected verbs *play* and *shoot* to be represented by transitions.

Rule 3: Shared VERBS by several entities will result in the formation of communication buffer places (communication links). Referring to Figure 4.4, MODELLER2 built a

buffer place linking *play* transition from ROSS acyclic nets with *play* transition in SPICER acyclic nets.

4.5.3 Formalisation of the construction

Now we provide a formal description of the steps taken by the proposed extraction method and construction procedure, for the first of the proposed event extraction methods (the remaining two are similar).

We assume that the NLP stage generated, from a given text of k sentences (written in a natural language), an *extracted* sequence:

$$ExtractedText = ExtractedS_1 \ ExtractedS_2 \ \dots \ ExtractedS_k$$

such that each $ExtractedS_i$ is a pair $(Entities_i, event_i)$, where $Entities_i$ are the entities associated with the i -th sentence, and $event_i$ is the verb of i -th sentence. Moreover, let

$$Entities = Entities_1 \cup \dots \cup Entities_k = \{ent_1, \dots, ent_n\}$$

be the set of all the entities. Then, for every entity ent , let $events(ent)$ be the sequence of events

$$events(ent) = x_1 \dots x_k$$

where $x_i = event_i$ if ent belongs to $Entities_i$, and otherwise x_i is the empty string. Intuitively, $events(ent)$ is the ordered sequence of events in which entity ent ‘participated’, and such a sequence can be used to provide a time-line for this entity. Following this observation, for each entity ent with $events(ent) = ev_1 \dots ev_l$, we construct an occurrence net $ON_{ent} =$

(P, T, F) , where:

$$\begin{aligned} P &= \{p_{(ent, ev_i)} \mid i = 1, \dots, l\} \cup \{p_{ent}\} \\ T &= \{t_{(ent, ev_1)}, \dots, t_{(ent, ev_l)}\} \\ F &= \{(p_{(ent, ev_i)}, t_{(ent, ev_i)}) \mid i = 1, \dots, l\} \cup \\ &\quad \{(t_{(ent, ev_i)}, p_{(ent, ev_{i+1})}) \mid i = 1, \dots, l-1\} \cup \{(t_{(ent, ev_l)}, p_{ent})\}. \end{aligned}$$

Finally, referring to Rule 3, for each pair $(t, t') = (t_{(ent, ev)}, t_{(ent', ev)})$ of transitions in ON_{ent} and $ON_{ent'}$, where $ent \neq ent'$, we add channel places $q = q_{(t, t')}$ and $q' = q_{(t', t)}$ together with the arcs

$$(t, q) \quad (q, t') \quad (t', q') \quad (q', t)$$

to enforce synchronisation between t and t' . One can then show that the result is a CSO-net which can be used for analysis and visualisation.

4.6 Automatic extractor development

This section outlines the tools and steps taken to develop the automatic extraction approach.

Several NLP tools provide linguistic analysis models, such as spaCy. SpaCy is an open-source library for advanced natural language processing in Python. It includes pre-trained models for various languages that support a wide array of NLP tasks, such as tokenisation, part-of-speech tagging, named entity recognition, and dependency parsing. The library is highly optimised for performance which enables for fast and accurate processing of text. That makes it a popular choice for both academic research and industrial applications [68, 9].

SpaCy works by using a pipeline where raw text data is passed through a series of processing steps. Each step in the pipeline applies a specific NLP task using appropriate models such as tokenisation or named entity recognition. Users can customise the pipeline by adding or removing models based on their specific needs [61]. For instance, a typical spaCy pipeline includes a tokeniser, a tagger, a parser, and an entity recogniser model as shown in Figure 4.5.

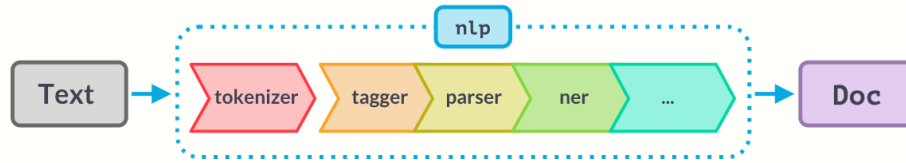


Fig. 4.5 Example of spaCy pipeline from [61].

Ambiguity in text due to pronouns presents a challenge for machines. While humans can distinguish and associate pronouns, machines require the use of coreferencing tools and models for this task. Coreference resolution [71, 63] in natural language processing involves the identification and linking of expressions in a text that refer to the same entity. For example, in the sentence “*Allen went to the store. He bought groceries.*” “*Allen*” and “*He*” refer to the same person. This process of coreferencing enables finding and linking entities across different parts of the text.

Therefore, in this experiment we employed NEURALCOREF 4.0 [46] which is a neural network-based coreference resolution library built for spaCy. NEURALCOREF 4.0 leverages advanced deep learning techniques to improve the accuracy and efficiency of coreference resolution tasks. It integrates well with spaCy and allows users to easily incorporate coreference resolution capabilities into their NLP pipelines as shown in Figure 4.6.

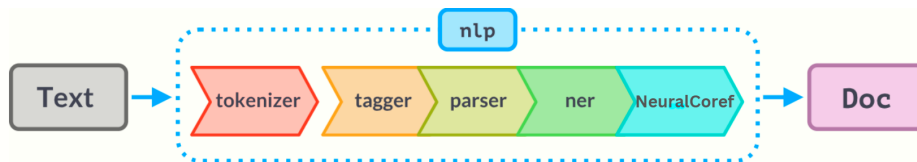


Fig. 4.6 Including NeuralCoref 4.0 in spaCy pipeline.

4.6.1 Automatic extraction: TEXT2SON

Following the observations made from the manual modelling process in Section 4.4, the objective is to develop a tool capable of extracting entities, crime events, and relationships between them. This tool will be used to build SON models for behaviour analysis and visualisation. Three extraction methods for event (transition) extraction have been applied to

evaluate and identify the most accurate method compared to human extraction presented in the previous section.

Method1: ROOT VERBS

Initially, we only considered extracting the verb that spaCy's parser nominates as the main verb of a sentence, tagged as *root* (we will call them *root verbs*). *root* tags appear once in every sentence representing the main word carrying the meaning of the sentence (Table 4.1).

Method2: ROOT VERBS and COMMON VERBS

The second method is based on the use of *root verbs* and *common verbs*. The most frequently occurring verbs in the dataset are evaluated in order to identify the common crime verbs. Approximately 570 crime stories from the website of *The Violence Policy Center* [15] were examined, leading to the compilation of list of the most frequently occurring verbs. The verbs in this list are identified as *common verbs*.

Method3: ALL VERBS

The third proposed method is to assume that all verbs present in the text represent transitions. This assumption is made to evaluate whether it is preferable to represent all verbs of a crime text or employ one of the two previously mentioned approaches. In other words, this method tests whether it is better to have a restricted set of verbs or whether it is more effective to use all verbs to construct SONs.

In order to extract data automatically, we propose to extract entities of type PEOPLE and verbs. Recall that we consider entities as representations for acyclic nets and verbs as representations of transitions within acyclic nets. We also consider that the *shared events* between different acyclic nets represent potential synchronised communications, and we formally connect them using buffer places. The process of constructing the extractor begins with setting up the appropriate environment that involves installing spaCy, Nuralcoref, and including Nuralcoref in spaCy's pipeline as shown in Figure 4.7(a). Then, the text used in the manual modelling in Figure 4.3 is passed through the pipeline for processing and tagging.

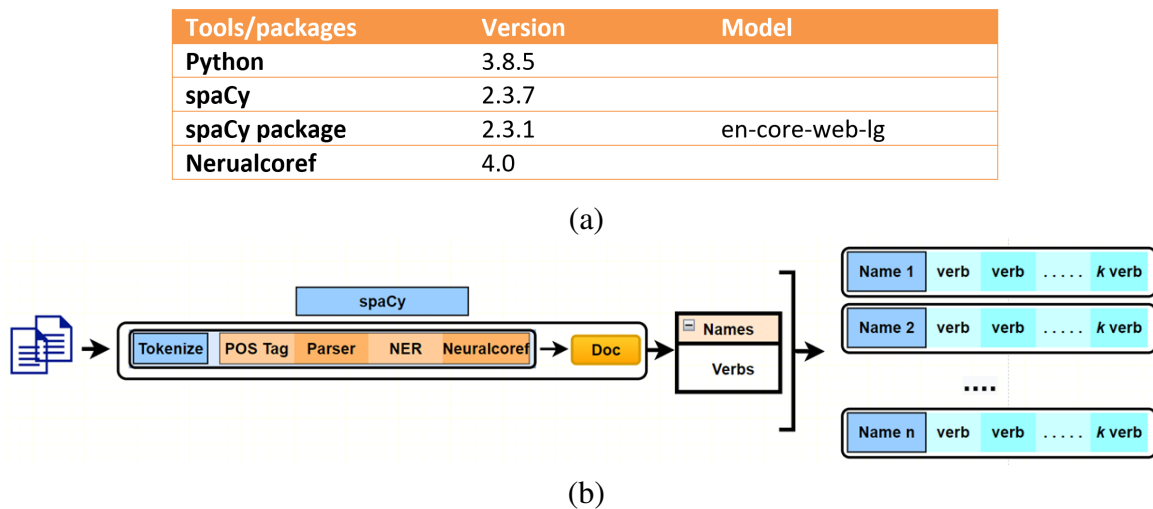


Fig. 4.7 Tools for extractor design (a); and extractor design (b)

Then, three different experiments were applied to the text to test the three different methods mentioned previously. The extractor read raw text as input from Figure 4.8, processed the text, and output lists that contain entities and related verbs as illustrated in Figure 4.7(b).

To expand more on the example depicted in Figure 4.7(b), it illustrated the steps involved in the extraction process. To begin with, text data is fed into spaCy's pipeline, where it is tokenised into individual tokens. The tokens are then passed to spaCy's tagger, which assigns anticipated POS tags depending on the pre-trained model predictions. Next, the dependency parser assigns tags indicating the relationships between words (tokens) in the sentence. The NER then assigns labels to entities, such as people, organisations, or dates, to mention a few. However, we are interested only in default NER tags, specifically the PEOPLE tag.

ROSS AND SPICER HAD PLAYED A GAME OF DICE. ROSS LOST AND ROSS WAS UPSET, ACCORDING TO POLICE RECORDS. THE NEXT DAY ROSS, WEARING BODY ARMOR, RETURNED TO SPICER'S HOME AND FATALLY SHOT SPICER

Fig. 4.8 The example text in Figure 4.3 after applying coreference resolution using NEURAL-COREF 4.0

To resolve pronouns to names, NEURALCOREF 4.0 was integrated into spaCy's pipeline to apply the resolution. Figure 4.8 shows how the crime example sentences from Figure 4.3

are modified after applying the coreferencing resolution. More precisely, we can observe the replacement of the pronoun HE with the person's name ROSS.

The extractor will then search for entities with the label PEOPLE and add them to a *names_list*. It will loop the entire text and avoid making duplicate name entries by checking names in the list before adding new names. Following this phase, it analyses every sentence by searching for the presence of people's names. These names are compared to those listed in the *names_list*. Any matching name results in the creation of a new list (*nameList_i*) with the name text inserted in the first element (*index 0*).

Subsequently, by carrying different experiments, the text is examined for verbs extraction in accordance with the three methods. In every experiment, the extractor will add the verbs associated with entities in the same sentence to the name list. That is, these verbs are grouped together in the respective *nameList_i*. This process continues until the end of the text, resulting in the creation of lists of people's names and their associated events (verbs).

4.7 Discussion

In order to evaluate the modelling approaches, we used the resulting models generated by both manual and automatic extraction. That is, we compared the resulting models from the manual modelling with the models generated by the extractor. Recall that we conducted manual extraction and modelling experiments with expert SON users. The results were similar in terms of the models explaining the case, although the forms differed. Fundamentally, these models were not semantically dissimilar, but rather varied in terms of the amount of information displayed. (Recall that Figure 4.4 illustrates the human expert modelling of the example sentences in Figure 4.3.)

All the SON models shown in Figure 4.4 convey the same narrative because all the modellers reported or modelled the semantics (meaning) of the sentences in the example sentences. However, the amount of information (transitions and acyclic nets) incorporated into the models varied among the modellers which reflects their individual understandings and perspectives. This variation, however, may indicate a lack of modelling consistency

Algorithm 1 Extraction Algorithm: method 1

```

1: Input: Text document
2: Output: Lists of entities with verbs - [entities_with_verbs file]
3: Step 1: Read document text
4: Step 2: pass text into spaCy's pipeline
5: Step 3:
6: Create entity_check [] list
7: Create allONs [[]] list
8: for Sentence in Text do
9:   for word in Sentence do
10:    if word is ent type and word  $\notin$  entity_check[] then
11:      add word to entity_check[]
12:      create new_list[] and add to allONs [[]]
13:      add word text to new_list[0]
14:    else
15:      continue
16:
17: for list_value[] in allONs [[]] do
18:   get list_value[0]
19:   for ent in Sentence do
20:    if ent equals list_value[0] then
21:      for verb in Sentence do
22:        if verb.label equals root then
23:          add verb to list value[]
24:

```

due to the volume of data presented in the experiment. Another issue that can be subject for further investigation in the future is the amount of time required for such modelling. To construct a model from text, spending time on reading, comprehending, and then modelling is bound to impact on the final result.

For the automatic extraction, three methods discussed in Section 4.6.1 are employed and compared to the human extraction described in Section 4.4. At first, we considered the METHOD1 where the main verb that spaCy's parser indicates as *root* verb is represented by a transition. Recall that the *root* tags appear only once in every sentence representing the main word carrying the meaning of the sentence which is usually a verb. Then, we applied METHOD2 by extracting *root verbs* alongside a list of *common verbs* used in criminal reporting. METHOD1 and METHOD2 were restrictive, meaning that the verbs were extracted

based on specific conditions (*root* and *common verbs*). For METHOD3, we considered extracting all verbs present in the text to study the variation of the resulting models.

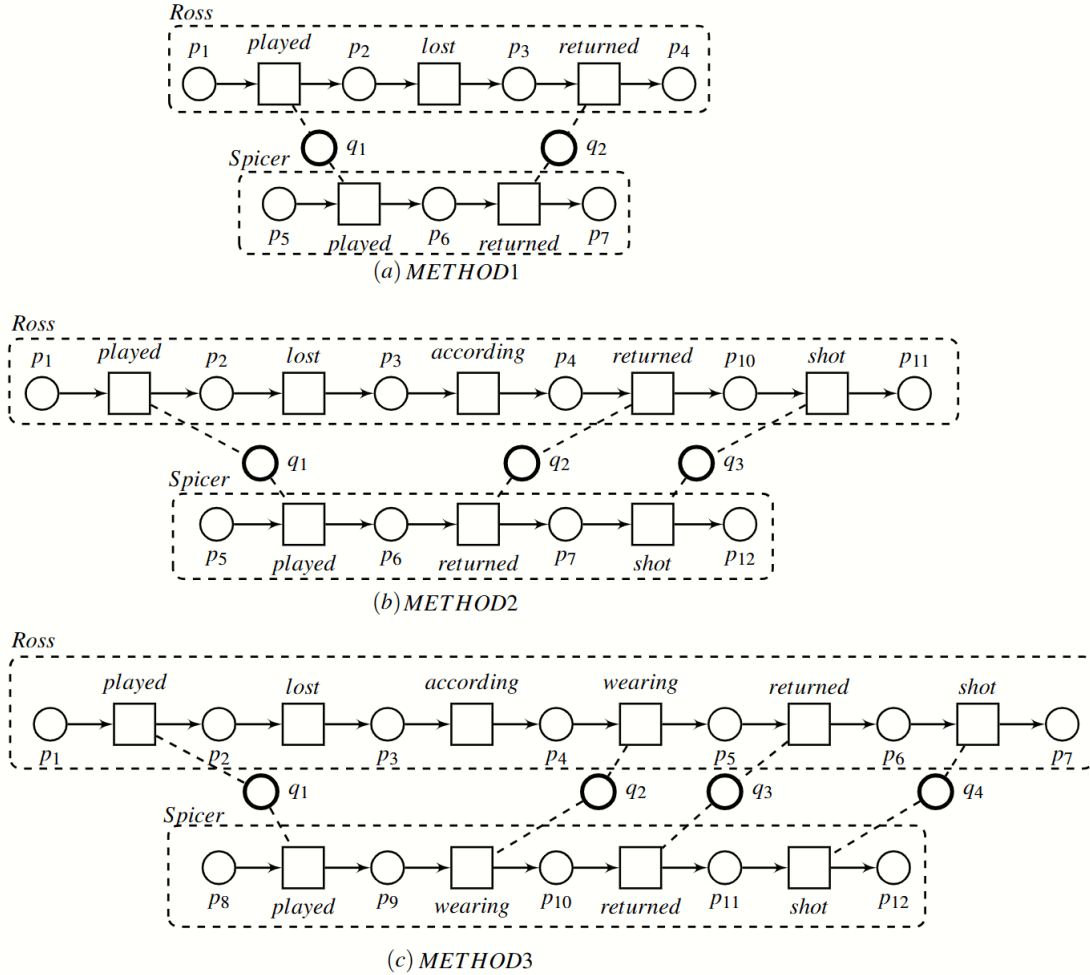


Fig. 4.9 Visualisation for the automated extraction of text in Figure 4.8 using the three methods. (a) METHOD1: extracting *root* verbs, (b) METHOD2: extracting *root verbs* and *common verbs*, and (c) METHOD3: extracting all verbs.

In METHOD1, the extractor demonstrated a lower extraction accuracy, by extracting fewer verbs and also associating verbs with unrelated entities. The assumption is that if the verb is a root verb and appears in a sentence with other entities, we can reasonably presume that they are linked. However, because each sentence can only have one root verb, this leads to neglecting of other non-root verbs. For instance, a sentence may contain more than one verb. However, the extractor will ignore all other verbs and extract only the verb labelled as *root* which results in information loss.

However, when we fed the extractor a list of *common* crime verbs, it performed notably better in terms of transition extraction. Yet, the same assumption was applied by linking the newly extracted verbs to all other entities in the sentence. As previously discussed, this is not necessarily accurate, because linking a verb that relates to a single acyclic net to other acyclic nets will lead to the establishment of an inaccurate communication link between the two entities. This can be observed clearly when METHOD3 is applied. The extractor extracted all the verbs present in the sentence and included all verbs to both entities present in the same sentence. When comparing this method to METHOD1 and METHOD2, it can be seen that more verbs are extracted and a new buffer place is established.

It is also noteworthy that there is a discrepancy in the amount of information displayed in visualised models in comparison to human modelling. Human modellers frequently augment the information representing events with additional words. A comparison between models in Figure 4.4 and Figure 4.9 revealed that human modellers tend to add more words to label transitions, such as the *played dice*, *lost game*, *leave upset*, and *goes to Spicer's* transitions. This addition provides further descriptions for the model aiding visualisation. On the other hand, the automatic extractor extracts only one word (verb) and uses the verb text to label a transition.

| Modeller1 | Modeller2 | Modeller3 | METHOD1 | METHOD2 | METHOD3 |
|-------------|-------------|-----------|----------|-----------|-----------|
| played | play | play | played | played | played |
| lost | lost | leaves | lost | lost | lost |
| shoot(shot) | return | wearing | returned | according | according |
| - | shoot(kill) | goes | - | returned | wearing |
| - | - | shoots | - | shot | returned |
| - | - | - | - | - | shot |

Table 4.2 List of all extracted verbs by various manual modellers and methods.

To facilitate comparison, Table 4.2 lists all the verbs extracted or selected for modelling by the various modellers. The manual modellers are identified as MODELLER1, MODELLER2, and MODELLER3. The remaining columns illustrate the different automatic methods. It can be observed that the majority of extraction methods extracted the verbs *played*, *lost*, and *shot*, which may be considered to express the essential shooting event. Extracting only the *root*

verb produced satisfactory results except for the omission of the shooting incident, which we think to be important. As previously stated, some sentences may contain more verbs other than the root verb. This is one of the shortcomings of METHOD1, which prompted us to experiment with two additional methods: METHOD2 and METHOD3. A comparison of the various approaches reveals that METHOD2 produced more consistent and acceptable results compared with manually produced models. METHOD2 compared to the other methods extracted the closest model structure compared to the models produced manually, as shown in Figure 4.9 and Table 4.2.

The above observations present an opportunity for further enhancement and development of the tool. For example, improving the model by incorporating word relationships such as subject and object via the use of the dependency parser. Additionally, creating and training a NER model, and introduce new labels can help in the extractions process for SON modelling.

4.8 Background

This section provides an overview of the use of visualisation, NLP, modelling, and data extraction for information, specifically in the context of crime data analysis. In particular, it will explore the advancements in crime modelling using formal methods and the extraction of crime data through various NLP techniques.

4.8.1 Visualisation and modelling

The visual representation of crime events, for instance, can be invaluable for crime analysis. A notable example is a tool for criminal investigations that employs Twitter data to provide contextual details about crime occurrences in a specific location [59]. The tool was tested as a prototype in the San Francisco region. It visually represents criminal incidents and related tweets allowing users to explore the tweets and crimes that occurred before and after an incident. It also offers information about the spatial and temporal characteristics of a crime enhancing the understanding of crime dynamics through internet-based exploration.

NLP plays a crucial role in crime modelling by automating the extraction of key information from unstructured data such as identifying named entities (e.g., suspects, locations) and classifying crime types. This extracted information is then fed for modelling, simulation and analysis of the flow of criminal activities. This integration enhances the understanding and prediction of criminal behaviour which enable law enforcement agencies to develop more effective strategies for crime prevention and investigation. NLP is used to process textual resources like criminal posts in social media to identify spatial and temporal crime patterns that can be modelled. This approach not only improves the accuracy of crime predictions but also supports the development of decision support systems for law enforcement [16, 49].

The work in [28] discusses the design of a Decision Support System (DSS) for police vehicle command and control using Petri Nets to optimise resource allocation in response to incidents. Petri nets can model concurrent and asynchronous activities and are therefore used to simulate the workflow from incident reporting to dispatch. The model integrates heuristic selection criteria into formalised procedures which improve decision efficiency and operational flexibility. The DSS dynamically adapts to different scenarios, improving police response times and resource management. This approach demonstrates the potential of Petri nets in building effective crime management systems.

4.8.2 Use of NLP in formal representation

By incorporating NLP techniques, it is possible to extract and process vast amounts of textual data from sources such as crime reports, social media, and online news. Combining Petri Nets and NLP in modelling provides a powerful approach to analysing complex system behaviours. Authors in [25] presented WoPeD (Petri net editor and simulator), which has new capabilities for combining Business Processing and NLP. WoPeD is an open-source Java application that allows the creation of business processes using workflow nets. The paper demonstrated algorithms for converting graphical process models to textual descriptions and vice versa. Nonetheless, the tool encounters a prevalent problem of semantic ambiguity.

SONs have been successful in modelling complex evolving systems such as accidents, crimes, and cybercrimes. [40] presented a model of a train accident. In this work, SONs

are applied to model a rail crash, providing a clear structure of the involved parties. The crash was caused by a train passing a red signal and colliding with a high-speed train. The created SON model divides the system into five components and illustrates their behaviours, communications, and relationships to help investigators understand and trace the cause of the accident. A previous study [27] demonstrated the potential of a modelling and analysis approach to crime incidents. The study evaluates the potential of SONs in the investigation of the assassination of John F. Kennedy. It uses SONCraft, which can assist in the modelling and visualisation of crime evidence. The evaluation focused on the potential accomplices of Oswald (the assassin), with SON assisting in the investigation. However, no accomplices were identified in this study. Moreover, [6] proposed the use of SONs to detect DNS tunnelling during a cyber attack. The author developed a unique method based on SONs for detecting DNS tunnelling and discussed how the data was pre-processed and eventually translated into SONs.

4.9 Conclusions

This chapter introduced a methodology and experiment for mapping and linking between Natural Language Processing and Structured Occurrence Nets. The methodology focuses on gathering and analysing text to build SON models. It starts with manually modelling a short passage of text. Then, it involves observing and analysing the modelling processes to identify how the modellers selected certain words. Insights from these analysis are then used to propose the mapping rules which help in identifying SON model components. The rules are used to create an automatic extraction approach from text to create SON models. This ensures that constructing models is based on manual efforts. Finally, experiments were applied using the proposed methods to extract SONs.

The algorithm is designed to at first extract people's names from unstructured text. Then, after extracting all people's names, it extracts transitions associated with them using three different approaches. We then used the manually produced models to extract and model the transitions and acyclic nets in SONs. Finally, We compared human extraction to the final

output produced by our automatic modeller. A comparison of the various approaches reveals that the METHOD2 produced more consistent and acceptable results.

Finally, the work reported in this chapter needs to be expanded to overcome some challenges that reduce extractor effectiveness. For instance, the extractor uses the default NER model with limited default labels that will not identify all crime-related entities, like weapons, for instance. Therefore, developing a new NER model or custom training a pre-trained model on a larger dataset and introducing new distinct NER labels suitable for crime extraction is necessary. Moreover, the most suited extraction method uses a pre-compiled list of words, which needs to be generalised. We will look at these challenges in the next chapter.

Chapter 5

Extracting Data from Unstructured Text for Representation in SONs

In this chapter, we extend the work from Chapter 4 by building a new customised Named Entity Recognition (NER) model tailored specifically to extract crime-related entities with increased precision. This involves fine-tuning the model to recognise and classify entities unique to crime text to enhance the accuracy of information extraction. Additionally, we propose and test a novel method for identifying verbs represented by transitions based on syntactical pattern analysis. This approach aims to improve the identification of key actions and events within crime narratives, providing a more robust method for subsequent analysis and modelling.

5.1 Introduction

This chapter builds upon the work presented in Chapter 4, which combined NLP and SONs to extract crime information. The aim is to address two shortcomings identified in the previous chapter Section 4.9: *(i)* resolving entities representative of acyclic nets, and *(ii)* proposing an enhanced method to detect verbs; recalling the important findings from Chapter 4:

Rule 1: ENTITIES in text are represented by acyclic nets.

Rule 2: VERBS are represented by transitions.

Rule 3: Shared VERBS result in the formation of communication via buffer places.

In this chapter, we will address the identified limitations by introducing a novel method for crime-related verb identification, and also develop a new effective NER model to identify crime entities. To accomplish this, work has been carried out in three phases. First, manually modelling short crime stories. Second, carefully analysing the models to identify entities and verbs. From these analyses, we will map and design relationships based on the manually produced models. The final phase focuses on designing the extractor, based on the findings from the previous phases.

The remainder of this chapter is organised as follows: Section 5.2 discusses and explains the dataset used in designing and evaluating of the NER model. Section 5.3 explains the process of building and analysing the manual models. Section 5.4 presents an analysis of the challenges in mapping manually created CSA-nets models to automatic extraction approach and discusses some of the challenges. Section 5.5 discusses the building and testing of the automatic extractor. Finally, Section 5.7 concludes the chapter and outlines the future work.

5.2 Datasets

There are two datasets used in this work, one for modelling experimentation, NER training, and validation, and the other for NER testing. The first dataset used in this chapter is the same as that used in Chapter 4, provided by [15], “The Violence Policy Center.” However, the focus here is on identifying the patterns modellers use to construct models representing crime stories in SONS. The dataset includes critical information for extraction purposes, such as incident locations, involved individuals, law enforcement, weapons used, and the context or events surrounding each shooting. The manual models are produced from from this dataset. The stories were distributed to human modellers to produce models that we will study and from which we will draw conclusions. The dataset is split into two sets to train and evaluate the new NER model, using 80% for training and 20% for validation.

The analysis of this data allows one to gain insights into the patterns and description of such incidents. A comprehensive review of the entire dataset was conducted to ensure that each shooting incident involved at least two individuals. Consequently, reported suicide cases are excluded. The aim is to identify a homicide pattern involving a shooter and a victim, and so efforts were made to ensure that each story included at least one shooter and one victim.

The second dataset is used for testing. This dataset is collected from *Fatal Encounters* in [24] which aims to create a comprehensive, searchable database of people killed during interactions with police in the U.S. since January 1, 2000. The project emphasises transparency and public access to aid research and raise awareness of police-related fatalities. The two datasets differ in terms of sentence structure and wording which is important to examine the NER model precision.

5.3 Manual Modelling and Analysis

This section describes the manual experiment and analysis of the models produced to draw conclusions. Specifically, we identify SON components (transitions and acyclic nets) from free textual resources to prepare for extraction. The initial step involved finding a method to comprehend how SON expert users identify net components. To achieve this, expert users engaged in modelling stories in SONS, which were then evaluated to observe the variations in the resulting models.

5.3.1 Manual modelling

In the manual modelling process, human modellers were provided with short homicide stories as the base texts. In this part of the research, four expert SON researchers contributed to building the manual models. The modellers' task began with a thorough reading of each story, allowing them to comprehend the narrative fully. This comprehension phase is crucial as it forms the basis for identifying significant words and phrases within the text. The modellers looked to identify keywords explaining the homicide narrative. These keywords were then systematically extracted based on their relevance to the crime story and their importance in

showing the sequence of events to represent SON components. This extraction was guided by the modellers' understanding and interpretation of the text, which ensured that the details and complexities of each story were captured accurately.

Once the components had been identified, the modellers started to construct the SON models. The modellers used the extracted elements to create acyclic nets and transition nodes describing the causal relationships of events in the story. The models ensured that all critical events were represented. This careful process resulted in complete models that reflected the narrative structure and dynamic flow of each homicide story. It provided a detailed visual representation that could be analysed further.

Example: Analysing STORY A

To show an example of manual modelling, Figure 5.1 presents manual models developed by two modellers for STORY A. These models differ in the number of acyclic nets with one model having four acyclic nets and the other models having five acyclic nets. One noticeable difference is that MODELLER2 depicted the STORE in an acyclic net, while MODELLER1 included the *store* in the event where *witnesses saw* the two men involved in the shooting entering the store (*enter_store*) transition. Furthermore, the modellers differ in the number of transitions assigned to the models; however, this is compensated by adding more information to labels on the transitions. For instance, MODELLER1 added *takes_bag_and_flees* where MODELLER2 pointed *pickup* in one transition and *flee* is represented by a second transition.

Story A:

'Brenton Rhasheem Davis, 24, allegedly shot and killed Tommy Jones III, 19, with a 9mm handgun in the parking lot of a Walgreens in Columbus, Georgia. Witnesses say they saw Davis and Jones enter the store together. When they left toward the parking lot, Jones ran from Davis who then allegedly shot Jones in the leg and back. Davis then walked up to Jones, who had collapsed, and allegedly fired more shots. Jones was shot at least four times. Nine shell casing from the 9mm handgun were found at the scene. A witness added that Davis then picked up a bag Jones had been carrying and then fled the scene. It is unknown what

was inside the bag. Jones had gone to the store to meet Davis in order to buy a 45-caliber handgun from him. The two men did not previously know each other. Davis was charged with one count of murder.’ [15]

Table 5.1 presents the entities extracted from STORY A by four different modellers to create CSA-net models, showing both common and unique entities identified by each modeller. The table shows that all modellers identified *Davis*, *Jones*, and *witnesses* as central figures in the story. *Davis* and *Jones* are the main entities explaining the story, with the *witnesses* playing an important role by observing some of the crime events. Moreover, the *bag picked up* by *Davis* after the *shooting* is recognised by most modellers, highlighting its importance as a piece of evidence in the story. These commonly extracted entities indicate a shared understanding of the key characters and elements crucial to the unfolding events in the story. However, the modellers also identified unique entities that reflect their individual perspectives on what additional details were significant. All modellers included the store as a key location where the initial encounter between *Davis* and *Jones* took place. However, they differ on what net structure was used (represented as acyclic net, or added to transition’s label). This addition provides context to the setting of the events. MODELLER3 added *shell casings* and *incident*, which adds more complexity and highlights differences in human understanding of modelling choices. The use of abstraction by MODELLER3 might necessitate the use of words such as *incident*, which is very challenging to identify using NLP tools, as the word *incident* is not mentioned in the text. MODELLER4 uniquely identified the *gun* specifying the weapon used in the shooting. This detail is crucial as it directly relates to the method of the crime and provides a specific piece of information that the other modellers did not explicitly mention. By including the *gun*, MODELLER4 emphasises the importance of the weapon in understanding the dynamics of the crime scene. The differences between the modellers highlight both common understandings and unique interpretations of the story.

Table 5.2 shows the verbs extracted by the four different modellers from the same story. Each modeller identified actions and events differently resulting in few variations in their extractions. There is consistency and variability between the produced models. MODELLER1, MODELLER2 and MODELLER4 seem to have a similar level of detail by

| Entities | | | |
|-----------|-----------|---------------|-----------|
| MODELLER1 | MODELLER2 | MODELLER3 | MODELLER4 |
| witness | Witnesses | Witnesses | Davis |
| Davis | Davis | Davis | Jones |
| Jones | Jones | Jones | Witnesses |
| Bag | Bag | Bag | Gun |
| | Store | A witness | |
| | | Shell casings | |
| | | Incident | |

Table 5.1: Identified entities representing *acyclic nets* by manual modellers .

identifying key actions like *entering*, *leaving*, *running*, *shooting*, and *picking up* the bag. In contrast, MODELLER3 included more detailed descriptions for all transitions. The main events including *entering* the store, *leaving* the store, *running*, *shooting*, *collapsing*, *picking up the bag*, and *fleeing*, are recognised by all the modellers with variations in wording. For example, all modellers captured the sequence of *shooting*, *collapsing*, and *picking up* the bag but the labelling differ.

For instance, MODELLER3 provides the most descriptions on transition labels making it easier to understand the sequence of events; however, the modeller used words that are not exactly present in the text. The modeller depended on the use of Behavioural Abstraction (BSA-net) leading to the use of modeller’s explaining either using extra words or changing the phrasal tense. Recall that BSA-nets comprise two levels, the upper level provides an abstraction view whereas the lower level captures all the details of a series of events.

In comparison to MODELLER3, other modellers focus on direct actions and consequences, using terms such as ‘*saw*’, ‘*ran_from_Davis*’, ‘*shot*’, and ‘*picked_up_bag*’ which are concise and clear. Despite the consistency in the sequences of actions across all modellers indicating agreement on the main events of the story, there are minor variations in the order and specificity of actions. However, the overall flow from entering the store to being charged with murder remains intact. Some differences reflect how each modeller interprets the story. For example, MODELLER1 use of ‘*more_shoots*’ versus MODELLER4 use of ‘*fired*’ indicates a difference in how they perceive the shooting events. These differences between modellers

highlight the variation in the manual extraction of events from stories. While the core events are consistently identified, the level of detail, tense, and labelling choices slightly vary.

| Verbs | | | | |
|-------|---------------------|-----------|---|-----------------|
| No | MODELLER1 | MODELLER2 | MODELLER3 | MODELLER4 |
| 1 | entered_store | hadGone | Goes to store to meet Jones | enter_the_store |
| 2 | left_store | saw | Goes to store to buy handgun from Davis | saw |
| 3 | runs | enter | Enters store | ran_from_Davis |
| 4 | shoots | left | Leaves store | shoot |
| 5 | collapses | ran | Davis and Jones seen to enter the store together | shot |
| 6 | walks | shot | Runs from Davis | killed |
| 7 | more_shoots | collapsed | Shoots Jones | picked_up_bag |
| 8 | takes_bag_and_flees | walked | Shot by Davis | fled |
| 9 | | fired | Collapses | |
| 10 | | killed | Drops bag | |
| 11 | | died | Bag dropped by Jones | |
| 12 | | carring | Walks up to Jones | |
| 13 | | pickedup | Shoots Jones | |
| 14 | | fled | Shot by Davis | |
| 15 | | | Picks up bag | |
| 16 | | | Bag picked up by Davis | |
| 17 | | | Flees | |
| 18 | | | Davis shoots and kills Jones | |
| 19 | | | Davis picked up a bag Jones had been carrying then fled the scene | |
| 20 | | | charged with murder | |

Table 5.2 Verbs representing events extracted manually from STORY A

From the modelling experiments, twenty models were obtained for analysis. The resulting models varied in net structure, yet they conveyed approximately the same meaning. This variation is due to several factors. For instance, individual modellers might add additional wording to transition labels, creating more abstract representations compared to those designed by other modellers. Moreover, different modellers may prioritise different

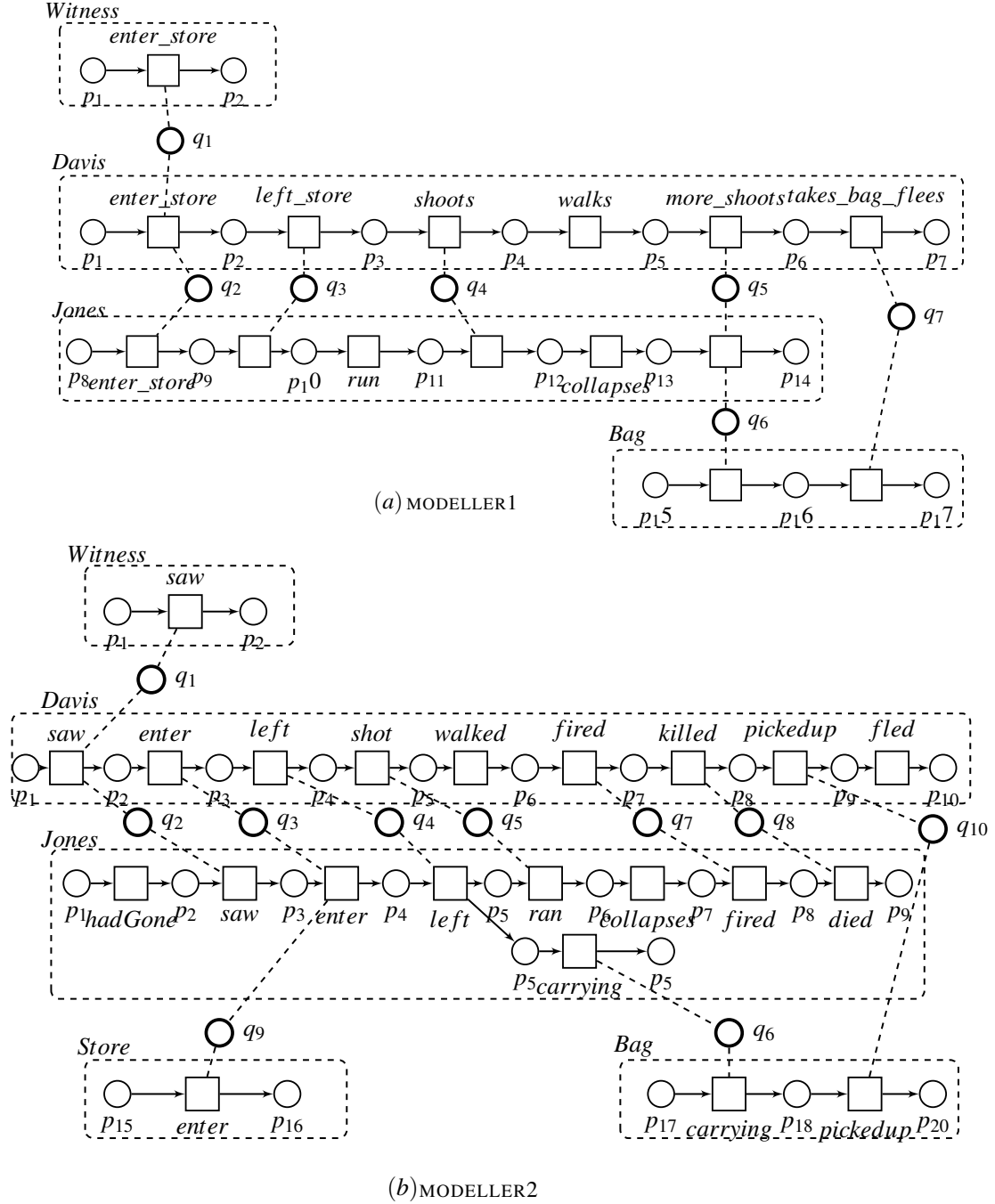


Fig. 5.1 STORY A representation by SON expert users (a) MODELLER1 and (b) MODELLER2, the rest of modeller representations are in Appendix A.

| USING LARGE MODEL NER | | Manual extraction of ANs done manually by different modellers | | Event extracted manually | |
|---------------------------------|--|---|---|---|--|
| 14 PERSON Story 5 | Time: 15 mins witness People (2) Bag Store | Model/ner ANs what/possible? PERSON | Time: 14-30 mins Witnesses People (2) Bag Store | Model/ner ANs what/possible? PERSON | Time: 6 mins People (2) Witnesses Gun |
| 1 PERSON:Brenton Rhasheem Davis | 1 PERSON:Brenton Rhasheem Davis | 1 PERSON:Brenton Rhasheem Davis | 1 PERSON:Brenton Rhasheem Davis | 1 PERSON:Brenton Rhasheem Davis | 1 PERSON:Brenton Rhasheem Davis |
| 1 DATE:24 | 1 DATE:24 | 1 DATE:24 | 1 DATE:24 | 1 DATE:24 | 1 DATE:24 |
| 1 PERSON:Tommy Jones III | 1 PERSON:Tommy Jones III | 1 PERSON:Tommy Jones III | 1 PERSON:Tommy Jones III | 1 PERSON:Tommy Jones III | 1 PERSON:Tommy Jones III |
| 1 DATE:15 | 1 DATE:15 | 1 DATE:15 | 1 DATE:15 | 1 DATE:15 | 1 DATE:15 |
| 2 QUANTITY:9mm | 2 QUANTITY:9mm | 2 QUANTITY:9mm | 2 QUANTITY:9mm | 2 QUANTITY:9mm | 2 QUANTITY:9mm |
| 1 ORG:Walgreens | 1 ORG:Walgreens | 1 ORG:Walgreens | 1 ORG:Walgreens | 1 ORG:Walgreens | 1 ORG:Walgreens |
| 1 GPE:Columbus | 1 GPE:Columbus | 1 GPE:Columbus | 1 GPE:Columbus | 1 GPE:Columbus | 1 GPE:Columbus |
| 1 GPE:Georgia | 1 GPE:Georgia | 1 GPE:Georgia | 1 GPE:Georgia | 1 GPE:Georgia | 1 GPE:Georgia |
| 1 PERSON:Davis | 1 PERSON:Davis | 1 PERSON:Davis | 1 PERSON:Davis | 1 PERSON:Davis | 1 PERSON:Davis |
| 1 PERSON:Jones | 1 PERSON:Jones | 1 PERSON:Jones | 1 PERSON:Jones | 1 PERSON:Jones | 1 PERSON:Jones |
| 1 CARDINAL:at least four | 1 CARDINAL:at least four | 1 CARDINAL:at least four | 1 CARDINAL:at least four | 1 CARDINAL:at least four | 1 CARDINAL:at least four |
| 1 CARDINAL:None | 1 CARDINAL:None | 1 CARDINAL:None | 1 CARDINAL:None | 1 CARDINAL:None | 1 CARDINAL:None |
| 1 ORG:Jones | 1 ORG:Jones | 1 ORG:Jones | 1 ORG:Jones | 1 ORG:Jones | 1 ORG:Jones |
| 1 CARDINAL:two | 1 CARDINAL:two | 1 CARDINAL:two | 1 CARDINAL:two | 1 CARDINAL:two | 1 CARDINAL:two |
| 1 CARDINAL:one | 1 CARDINAL:one | 1 CARDINAL:one | 1 CARDINAL:one | 1 CARDINAL:one | 1 CARDINAL:one |
| 26 | 26 | 26 | 26 | 26 | 26 |
| 22 PROPN | 22 PROPN | 22 PROPN | 22 PROPN | 22 PROPN | 22 PROPN |
| 19 PUNCT | 19 PUNCT | 19 PUNCT | 19 PUNCT | 19 PUNCT | 19 PUNCT |
| 11 NUM | 11 NUM | 11 NUM | 11 NUM | 11 NUM | 11 NUM |
| 12 ADV | 12 ADV | 12 ADV | 12 ADV | 12 ADV | 12 ADV |
| 22 VERB | 22 VERB | 22 VERB | 22 VERB | 22 VERB | 22 VERB |
| 9 COORD | 9 COORD | 9 COORD | 9 COORD | 9 COORD | 9 COORD |
| 18 ADP | 18 ADP | 18 ADP | 18 ADP | 18 ADP | 18 ADP |
| 18 DET | 18 DET | 18 DET | 18 DET | 18 DET | 18 DET |
| 20 NOUN | 20 NOUN | 20 NOUN | 20 NOUN | 20 NOUN | 20 NOUN |
| 7 PRON | 7 PRON | 7 PRON | 7 PRON | 7 PRON | 7 PRON |
| 10 ADJ | 10 ADJ | 10 ADJ | 10 ADJ | 10 ADJ | 10 ADJ |
| 3 ADJ | 3 ADJ | 3 ADJ | 3 ADJ | 3 ADJ | 3 ADJ |
| 1 SCOUN | 1 SCOUN | 1 SCOUN | 1 SCOUN | 1 SCOUN | 1 SCOUN |
| 1 PART | 1 PART | 1 PART | 1 PART | 1 PART | 1 PART |

Fig. 5.2 Example of SON model analysis for STORY A.

aspects of the text that can lead to variations in which parts of the story are emphasised or neglected. This is based on modellers own interpretation of what should be modelled. Appendix A shows all the list of verbs identified by all modellers in the twenty models produced manually. Moreover, the extracted entities, identified verbs, and proposed models resulting from automatic extraction are also included in Appendix A.

The analysis of produced models involved several steps. First, we examined acyclic nets by compiling and listing all the words used to label them, and additionally conducted experiments using NLP techniques to determine if these words could be identified using default NLP models. Second, we collected all the words used represented by transition within the nets (as shown in Figure 5.2).

5.4 Analysis and Preprocessing

From studying the manual models, we confirmed the previous findings that all modellers selected proper nouns which can be identified as entities to represent acyclic nets, and verbs to represent transitions. However, the modellers chose proper nouns for acyclic nets that current default NLP models cannot identify as entities. For instance, modellers used words like *Police*, *9mm handgun*, *witnesses*, *SWAT* to represent acyclic nets. Therefore, the work is organised into steps, starting with the introduction of new NER labels, followed by annotating

data using the new labels. Finally, training and validating the custom NER model using the annotated dataset.

For the presence of pronouns present in texts, we kept the same previous setup by integrating NEURALCOREF 4.0 to resolve coreferences. It is worth mentioning that applying coreference models to texts of considerable size can result in uncertain resolutions.

Additionally, transitions were analysed to suggest a novel method for identifying verbs. As mentioned in Chapter 4, we had three methods for verb identification: METHOD1 (using root verbs), METHOD2 (using root and common verbs), and METHOD3 (using all verbs). However, we aim to develop a new method that more closely mimics human verb identification. Therefore, we proposed a *pattern-based method* for verb identification.

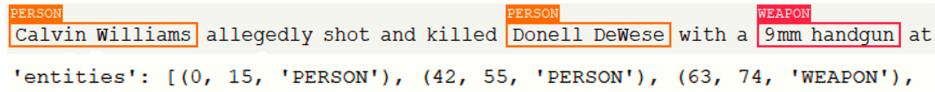
5.5 Automatic Modelling Experiment

5.5.1 Entities identification and analysis

One noteworthy observation was the frequent mentioning of entities like weapons, transportation, and family members / relatives, for instance, which were often undetected by default NER models due to the absence of specific related labels. In response to these findings and after an in-depth analysis, we introduced new entity labels to reinforce the model's accuracy and comprehensiveness. The newly added labels cover a range of entities including ['LAW_ENFOR', 'WITNESS', 'WEAPON', 'TRANSPORTATION', 'RELATIVE', 'PROFESSION', 'UNNAMEDPER']. This improvement aims to significantly advance the NER model's ability to recognise diverse crime-related entities, ensuring a more effective analysis of crime narratives.

To address this, we started an annotation process that required human intervention for identifying and labelling named entities such as people's names, weapons, and locations, in the data text according to the predefined labels. Annotators highlight entities and assign the appropriate labels. These annotations provide labelled data for training and testing NER models as shown in Figure 5.3 to enable accurate recognition and categorisation of entities.

The NER annotation is done using TagEditor [64] which is used for designing custom NER models tailored to specific needs. TagEditor provides a user-friendly interface for creating annotations allowing for the efficient tagging of entities within text. The first step in using TagEditor for NER annotation involves preparing the dataset that will be annotated. This includes collecting and organising the text that needs to be processed. Next, TagEditor is used to create an annotation project and the new labels are added to the project. Annotators can then manually tag these entities within the text using TagEditor's interface, ensuring that the annotations are consistent and accurate. Once the annotation process is complete, the annotated data is exported from TagEditor into SPACY to train a custom NER model.



```
'entities': [(0, 15, 'PERSON'), (42, 55, 'PERSON'), (63, 74, 'WEAPON'),
```

Fig. 5.3 Example of an annotation process shows text and entity labels. The entity labels in this example include tags for PERSON and WEAPON.

For this experiment we collected and annotated approximately 334 crime stories, then divided them into 80% for NER model training and 20% for model evaluation. The training involved reading all stories and tagging the phrases with one of the labels PERSON, LOC, LAW_ENFOR, WITNESS, WEAPON, TRANSPORTATION, RELATIVE, PROFESSION, and UNNAMEDPER. The training dataset consists of the raw text, the spans index, and the label tags as shown in Figure 5.4.

```
TRAIN_DATA = [
    ('raw text', {'entities': [(index_start, index_end, 'LABEL'), ...]}),
    ...
    ('raw text', {'entities': [(index_start, index_end, 'LABEL'), ...]}),
]
```

Fig. 5.4 Format of the dataset prepared for training.

The NER model was validated using the other 20% of the data from [15] that was manually annotated. The data used here is similar to the data used to train the model. We used precision,

recall, and F1-score as evaluation metrics for our results. Precision measures how many of the predicted entities are correct. Recall measures how many of the actual entities in the dataset were predicted correctly. The F1 score is the harmonic mean of precision and recall.

As shown in Table 5.3, the precision indicates that roughly 83.95% of predicted entities were correctly identified which highlights the accuracy of the model's predictions. Moreover, the recall score suggests that the model captures around 87% of all relevant entities in the test texts demonstrating its ability to retrieve entities. The F1-score 84.84% provides a balanced assessment of the model's performance reflecting good identification of relevant entities while minimising false positives and negatives. These results illustrate that the NER model has a good performance indicating its effectiveness across various NER labels.

| Data Set | Measure | Score % |
|------------|-----------|---------|
| Validation | Precision | 83.95 |
| | Recall | 87.45 |
| | F1 | 84.84 |

Table 5.3: NER model performance validation.

Then the model was tested on more than 50 stories related to police shootings obtained from [24], which was not part of the training dataset. The testing scores reveal acceptable performance as shown in Table 5.4, with a precision of 69.72% and a recall value of 77.15%. While the model effectively identifies relevant instances, the F1-score of 71.08% indicates balanced performance. In general, the model shows potential for practical applications. However, further training and fine-tuning may be necessary depending on the specific type of crime to ensure optimal performance.

| Data Set | Measure | Score % |
|----------|-----------|---------|
| Testing | Precision | 69.72 |
| | Recall | 77.15 |
| | F1 | 71.08 |

Table 5.4: NER model performance testing on different dataset.

The evaluation results show a noticeable performance difference between the validation and testing datasets. The model achieved higher scores on validation compared to testing. The drop in performance can be attributed to the nature of the datasets, as the validation dataset closely resembles the training data, sharing similar structural features that the model has been exposed to during the training process. This similarity enabled the model to achieve stronger results.

In contrast, the testing dataset comes from unseen data with different wording and sentence structure. The lower precision on the testing dataset indicates that the model is more likely to identify entities incorrectly. Furthermore, the lower recall reflects the model's difficulty in capturing all true entities. Therefore, the drop in performance shows that while the model is effective in handling data similar to its training set, it struggles when applied to new datasets with different sentence structure. This suggests that the model needs to be trained on more diverse datasets to improve its generalisability.

5.5.2 Verbs identification and analysis: *pattern-based*

The models were also analysed to find a new method to identify transitions. Identifying verbs that accurately represent SON events from crime stories presents a significant challenge. To enhance previous methods that relied on *root verbs* and a list of *common verbs*, we explored a more general approach. We adopted a *pattern-based* approach to identify verbs representing transitions from manually created models. To ensure precision, we examined the manually produced SON models created by expert users, then selected verbs that were identified by at least 50% of the modellers. Through dependency parsing, we determined the positional relationships of these verbs within sentences concerning surrounding words.

We specifically identified the type of words and their dependency relationships preceding and following specified verbs collected from manually created models. We then analysed the part-of-speech tags and syntactical tags of the words before and after the verbs. This enabled us to recognise syntactical patterns for event identification based on the human process observed in manual SON models. This approach led to the identification of *seven* patterns, outlined in Table 5.5. For instance, a verb may be preceded by a noun, proper

noun, or pronoun and followed by a noun, proper noun, or pronoun with subject and object relationship.

To elaborate mode, the *pattern approach* employs linguistic features to identify verbs representing transitions. SpaCy processes text through tokenisation and linguistic annotations, assigning each token (word) with part-of-speech (POS) tags and dependency labels. These annotations facilitate the accurate identification of verbs, nouns, and other parts of speech, as well as the determination of syntactic relationships within sentences. By analysing manually selected verbs and their associated patterns from expert users, we compiled a list of patterns. If a verb pattern matches any of the patterns in this list, it is considered a SON transition. Furthermore, the evaluation of verb extraction is unnecessary, as it relies more on pattern-rule-based techniques rather than statistical predictive identification.

Table 5.5 List of all patterns identified from the manual models created by SON users [POS]verb[POS].

| No | Part-of-Speech Relationship | Dependency Relationship |
|----|---|---|
| 1 | [noun, proper noun, pronoun][EVENT][noun, proper noun, pronoun] | [nsubj, nsubjpass] [dobj, iobj, pobj] |
| 2 | [proper noun, pronoun][EVENT][verb, adverb] | [nsubj, nsubjpass] [ccomp, conj, advcl, xcomp, advmod] |
| 3 | [verb][EVENT][verb] | [advcl] [advcl] |
| 4 | [noun, proper noun, pronoun][EVENT][verb, adposition] | [dobj, iobj, pobj] [ccomp, conj, advcl, xcomp, advmod, pcomp, root] |
| 5 | [verb, adverb][EVENT][verb, auxiliary] | [advcl, advmod, ccomp][advcl, conj, ccomp, root] |
| 6 | [pronoun][EVENT][noun] | [nsubj] [relcl] |
| 7 | [noun][EVENT][verb] | [nsubj] [advcl] |

The next section discusses the process used to link the extracted SON components. It explains how transitions are connected within acyclic nets, as well as between different acyclic nets resulting in the creation of CSA-nets.

5.5.3 Entity-verb Linking and Communication Structuring

Entity-verb linking

The NER model is used to extract entities, and by employing the proposed verb-pattern-based identification, verbs are also extracted. Subsequently, lists are generated for each acyclic net and all relevant transitions found in the same sentence are appended to each entity's list. Algorithm 2 describes the extraction process and the assignment of verbs to entities. Figure 5.5 shows the structure of the output lists (one list for each story), denoted as *ExtractedData_Lists*. Each list within the *ExtractedData_Lists* contains information regarding an entity and its corresponding verbs extracted from the text. For each entity, with each verb associated with it, three information values are provided: the verb (v), the location of the verb within the sentence (v_{sent_loc}), and the index of the verb within the text (v_{index}). Following the extraction from each story, a list can contain multiple entities along with their respective verbs extracted from the text.

$$\begin{aligned}
 \text{ExtractedData_Lists} = & \left[[entity^1, v^{1,1}, v_{sent_loc}^{1,1}, v_{index}^{1,1}, \dots, v^{1,k_1}, v_{sent_loc}^{1,k_1}, v_{index}^{1,k_1}], \right. \\
 & \vdots \\
 & \left. [entity^n, v^{n,1}, v_{sent_loc}^{n,1}, v_{index}^{n,1}, \dots, v^{n,k_n}, v_{sent_loc}^{n,k_n}, v_{index}^{n,k_n}] \right]
 \end{aligned}$$

Fig. 5.5 Extraction output shows how lists are structured.

To transform entity lists into CSA-nets, a sequential acyclic net is constructed to represent each entity list. This involves creating a line-like structure where each verb encountered in the entity list leads to the construction of a transition. The construction of acyclic net $acnet_m$ starts by creating an initial place $p^{m,0}$. Then, for every verb $v^{m,i}$ encountered in the $entity^m$ list, a corresponding transition modelling $v^{m,i}$ is constructed, followed by the insertion of a new place $p^{m,i}$, as illustrated in Figure 5.6.

Following the construction of acyclic nets, we use the verb information (i.e., v , v_{sent_loc} , and v_{index}) to construct the communication links. This will involve looping and searching for

similar verb information among the entities. If such information is found, a communication link is established.

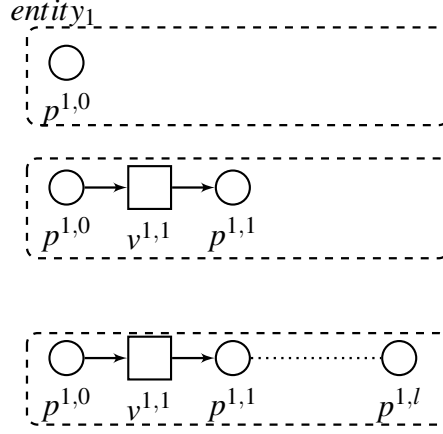


Fig. 5.6 Example of constructing an acyclic net.

Communication identification

Communication between different *acyclic nets* is possible and leads to *CSA-nets*. Such nets are constructed when transitions in different acyclic nets are connected using buffer places. To accomplish this, we have already extracted verbs along with their corresponding information, such as the specific location within the text and the sentence in which the verb is found. This extracted information enables us to establish communication by identifying identical verbs across different acyclic nets. Upon discovering a match, new buffer places are created to facilitate communication (synchronisation). For instance, if transition t modelling $v^{m,i}$ in acyclic net $acnet_m$ (representing $entity^m$) and transition w modelling $v^{l,j}$ in acyclic net $acnet_l$ (representing $entity^l$) are such that $v^{m,i} = v^{l,j}$, then we add two buffer places, b_{tw} and b_{wt} , and four arcs between transition t and w to ensure their synchronicity (the added arcs are: (t, b_{tw}) , (b_{tw}, w) , (w, b_{wt}) , and (b_{wt}, t)). To improve readability, in the diagrams we depict such an addition of places and arcs by showing just a single buffer place linked by undirected edges with t and v , as shown in Figure 5.7.

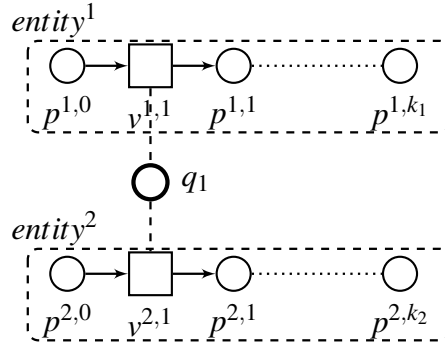


Fig. 5.7 Example of constructing buffer place q_1 between $entity^1$ and $entity^2$ acyclic nets.

Algorithm 2 Entities and Pattern-based transition extraction algorithm

```

1: Input: Text document
2: Output: Lists of entities with verbs - [ExtractedData_Lists file]
3: Read the text file, then process the text using spaCy, produce Doc
4: Initialize entLabels list with NER labels
5: Initialize empty lists names, verbs, name_info, and verb_info
6: for each sentence in doc do
7:   for each token in sentence do
8:     if token's entity label  $\in$  entLabels then
9:       if token not in names then
10:        Add the token to names list
11:     else
12:       if token is verb and in the verb pattern then
13:         if token  $\notin$  verbs then
14:           Add token, sent_number, and token index to verbs and verb_info
15: Initialize name_verb_mapping{ }
16: for each name in names do
17:   Initialize name_info with name
18:   for each sentence in doc do
19:     if name appears in sentence then
20:       for each token in sentence do
21:         if token is verb and token  $\in$  verb_info then
22:           if token matches any verb in verbs and is in syntactical relation with name
23:             then
24:               Add token, sent_number, and token index to name_info
25:   Sort name_verb_mapping by the order of appearance
26: write to FILE

```

The algorithm processes text from documents to extract entities and verbs using the seven identified patterns. It begins by reading and processing the text using spaCy's pipeline

to generate the Doc object that contains token objects. These token objects include POS, dependency, and NER labels, which are used to identify the linguistic attributes of each token. Initially, the algorithm iterates through the sentences in the document, adding recognised entities to a names list and verbs matching the seven patterns to a verbs list. Additionally, the algorithm captures metadata, which are the sentence number where each verb is found and the verb's index number. These data are appended next to each verb. This initial extraction ensures systematic capture of both entities and verbs for further processing.

After that, the algorithm iterates through each name in the names list and creates a separate list for each name, placing the name at *index₀* of its list. It then iterates through the document to look for verbs that match verbs stored in the verb_info list. If a match is found within the same sentence, the verb and its metadata are added to the corresponding name's list. Once all mappings are established, the verbs are sorted by the order of appearance in the text and written to an output file.

Story A models analysis

In evaluating different modelling approaches for Story A, we compared the models generated through both manual and automatic extraction. By analysing the resulted models we found that different manual modellers produced different SONS in terms of size and net components. For instance, the number of acyclic nets "entities" varied among the modellers, where we have 4, 5, 7, and 4, respectively (as shown in Table 5.6). All modellers selected and modelled the names of the people involved in the shooting (Davis and Jones) as well as the witnesses. Three of the modellers included the "Bag" picked by the shooter, however, they all failed to model the "gun" used which was identified only by the fourth modeller.

The automatic extractor successfully identified all people names, witnesses, and mentioned weapons. Moreover, the extractor identified the location where the incident took place "parking_lot". However, it failed to identify the "bag", this is due to the linguistic nature of the word as it is a common noun. Further training of the model could improve its ability to identify terms referring to objects, such as "bag" in this example.

| Entities | | | | |
|-----------|-----------|---------------|-----------|--------------------|
| MODELLER1 | MODELLER2 | MODELLER3 | MODELLER4 | Automatic |
| Davis | Davis | Davis | Davis | Davis |
| Jones | Jones | Jones | Jones | Jones |
| witness | Witnesses | Witnesses | Witnesses | Witnesses |
| Bag | Store | A witness | Gun | Parking_lot |
| | Bag | Bag | | A witness |
| | | Shell casings | | 09mm_handgun |
| | | incident | | 45_caliber_handgun |

Table 5.6 Entities from STORY A.

| | Manual | | | | | | | | Automatic | |
|----------|--------------|---|---|---|-------------|----|----|---|--------------|-------------|
| | Acyclic nets | | | | Transitions | | | | Acyclic nets | Transitions |
| Modeller | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | Auto | Auto |
| | 4 | 5 | 7 | 4 | 15 | 22 | 23 | 9 | 7 | 45 |

Table 5.7 Number of acyclic nets and transitions extracted for Manual and Automatic methods

Moreover, Table 5.7 illustrates that the number of transitions also varied. When examining the manually produced models, we found transition counts of 15, 22, 23, and 9 by different modellers, respectively, compared to 45 transitions identified by the automatic extractor (as shown in Figure 5.8). This higher number of transitions in the automatic extractor is attributable to the representation of verbs across different acyclic nets. Manual modelling used augmented transition labels that provided additional information within these labels, resulting in fewer number of transitions.

Furthermore, the automatic extractor consistently identified the same data demonstrating its ability to extract information in a reliable and consistent manner. Manual models representing Story A in SONS are depicted in Figures 5.1 and A.1, while the results of the automatic extraction are shown in Figure 5.8.

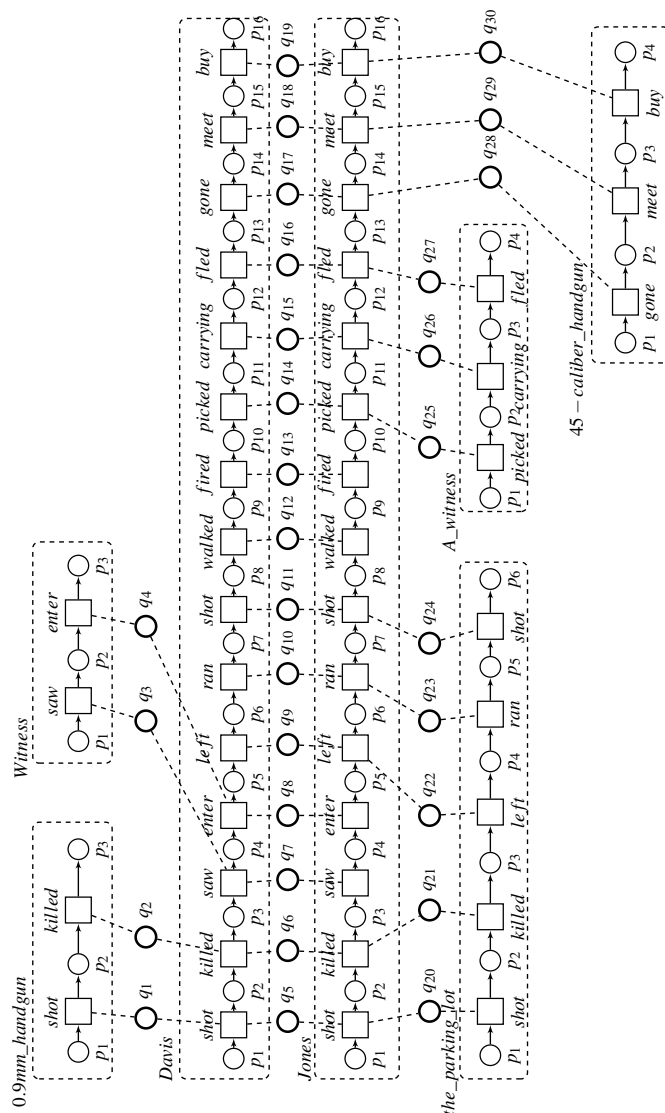


Fig. 5.8 A potential SON model could be developed based on the results of the extraction for STORY A. The extraction approach identified more entities and verbs compared to manual identification in Figure A.2.

5.6 Discussion

This section discusses the models produced by both the manual and the automatic methods. It discusses the enhancements and limitations of the two extraction methods. In analysing the produced models using manual and automatic extraction, four human modellers were compared with each other and then against automatic approach. Both approaches (manual and automatic) applied steps including processing, analysing, and extracting acyclic nets and transitions from the stories to construct SONS.

The results varied in terms of information and consistency among the manual modellers. While there was notable consistency in modelling certain acyclic nets, such as those representing people's names and witnesses, inconsistencies occurred in modelling weapons, law enforcement, and crime locations. For instance, weapons mentioned in crime stories A, C, D, and E were modelled inconsistently. Modeller 1 assigned a "gun" acyclic net to all stories except Story A. Similarly, Modeller 4 included a "gun" acyclic net for these stories but excluded it from Story D. In contrast, Modeller 3 consistently avoided constructing acyclic nets for weapons, and Modeller 2 created a "gun" acyclic net only for Story D. These variations indicate that human modellers were inconsistent in representing weapons across all stories.

Similarly, crime locations were inconsistently modelled. Although crime locations were present in nearly all stories, Modellers 1 and 2 constructed acyclic nets for locations in Stories C, D, and E, while Modellers 3 and 4 did not represent locations at all. These differences highlight variability in how modellers perceived or interpreted the crime details.

Moreover, manual modellers tend to compensate for building acyclic nets by augmenting transition labels with additional information. While this may be convenient for producing more compact (smaller) SONS, it risked excluding important acyclic net details. For example, the same modeller might represent a weapon as an acyclic net in one instance but include it as information in a transition label in another model or neglect mentioning them in other times. Additionally, Modeller 3 consistently included an acyclic net labelled "incident", which serves as a behavioural abstraction of the shooting incident. The exact word "incident"

in most cases were not explicitly mentioned in the text but was added by the modeller for clarification.

The automatic extraction, in most cases, shows a higher degree of consistency and completeness in entity identification. Across almost all stories, the automatic extractor accurately identified people's names, weapons, witnesses, and locations. However, it struggled with certain nouns, such as "bag" in Story A, which most manual modellers represented as an acyclic net. This limitation maybe attributed to the inherent difficulty to recognise as it is a common noun not a proper noun. A potential solution could possibly be identifying nouns that directly follow verbs, but further studies and analysis are needed to understand what type of entities commonly follow verbs in crime reports and how they could be labelled effectively.

While the automatic extractor provided a more comprehensive and consistent list of entities compared to manual modellers, it showed limitations in identifying items like "bag", "traffic light", or "gas pump". These were represented manually but not captured by the extractor. Despite these shortcomings, the automatic extractor reliably identified people's names, weapon mentions, witnesses, and, in most cases, locations. This consistency can be attributed to the use of a customised trained NER model.

Overall, the automatic extractor showed greater consistency than manual modellers, especially in recognising entities like weapons and locations across different stories. Manual modellers often produced structurally varied nets due to differences in interpretation and approach. Automatic extraction, while consistent, showed some limitations that could potentially be mitigated by further training the NER model using diverse datasets.

5.7 Conclusions

This chapter demonstrated the process of extracting and constructing acyclic nets and transitions from textual sources using NLP based on manual modelling. Moreover, it explained the steps taken to propose a custom NER model by introducing new labels and identifying verbs using proposed patterns. Initially, we engaged individuals to model stories manually in

SONs. This step provided a foundational understanding of how transitions and acyclic nets are structured from crime stories. The manual modelling allowed us to identify and define new entity labels which were used to train a custom NER model. The model was validated using stories from the same dataset used in training the dataset. Then the model was tested on a different dataset to evaluate the model's generalisability to varied data input.

We introduced a new method for verb identification by analysing patterns derived from human-identified verbs in the manual models. This method enhanced our ability to capture and categorise verbs accurately, which is crucial for understanding the actions and interactions within the narratives. By leveraging these verb patterns, we could develop a more detailed and effective approach to verb identification, which in turn contributed to the overall accuracy of the entity-verb extraction process.

The next step involved establishing communication links between different entities, which was achieved by identifying similar verbs across different acyclic nets. This approach allowed us to map out interactions between acyclic nets providing deeper insights into the dynamics of the crime story narratives. Creating these communication links is significant as it highlights the interactions between entities, and facilitates a better understanding of the underlying story structures and their complexities.

Chapter 6

Coloured Acyclic Nets (CA-nets)

Coloured Acyclic Nets (CA-nets) are an extension of SON acyclic nets employing coloured tokens and transition evaluation capability. This chapter introduces the concept of CA-nets, detailing their structural enhancements and advantages over the standard acyclic nets. We explore the application of CA-nets in designing a classifier and present an experiment to demonstrate their effectiveness in classification tasks leveraging the unique features of CA-nets.

In addition to their application in crime modelling, SONs can also be used to analyse cybercrime. This chapter builds on the flexibility of SONs to demonstrate that they are a robust and adaptable modelling framework applicable to diverse fields. With the increasing occurrence of cybercrime, there is a pressing need to propose solutions to reduce cybercriminal activities. This chapter addresses this need by introducing coloured acyclic nets capable of detecting and classifying fake social media accounts.

6.1 Introduction

In this chapter, we introduce a SON model called Coloured Acyclic Nets (CA-nets), which serve as an extension of acyclic nets discussed in Chapter 3. The CA-net model differs from acyclic nets by several key structural enhancements. One of the most significant differences is the incorporation of colours, which allows for *coloured tokens*, *arc expressions*, *transition*

guards, and *codes* associated with transitions. These features enable the CA-net to represent and manage more complex processes with greater flexibility.

Another notable advantage of CA-nets is their structure, which results in fewer nodes compared to the standard acyclic nets when modelling the same system behaviour. The CA-net model offers a potential approach to modelling complex systems including classifiers. This chapter discusses the specifics of the CA-net structure and the design of a classifier for social media accounts as an application.

Detecting *fake* accounts on the social media platform X is important to minimise their often malicious and deceptive purposes. For example, spreading misinformation, political manipulation, advertisement and spam, cyber attacks, and often artificially inflating follower counts. One such approach to identify online fake accounts is through the use of Structured Occurrence Nets (SONs) [36]. This study aims to use a specific SON approach, which can analyse and classify data inspired by its effectiveness in analysing and visualising system behaviour (see, e.g., [6, 7, 3, 5, 40]). The objective of this study is to investigate the capability of SONs to model, evaluate, and classify a large number of accounts using *coloured tokens* to distinguish between *genuine* and *fake* accounts. This study introduces a CA-net model developed to filter, evaluate, and detect fake accounts on the social media platform X, which for the purposes of critically analysing previous research, will be referred to as ‘Twitter’, the platform’s previous name.

Upon creation of each Twitter account, specific features, such as its name, ID, number of followers, number of following, profile background and text colours, are assigned. These assigned features will be used to distinguish between genuine and fake Twitter accounts. The CA-net model presented here will refer to each Twitter account as a *token* with various feature values. Each token will then be evaluated and classified in stages, leading to the final decision. To sum-up, the work here aims to introduce a coloured extension of the SON model developed for the classification of Twitter user accounts into genuine and fake classes. Subsequently, the model’s performance is assessed using a confusion matrix.

The remainder of this chapter is organised as follows. Section 6.2 explains the dataset and the feature selection used in the design and analysis of the classifier. Section 6.3 discusses the

definitions, design, and analysis of the CA-net model. Section 6.4 discusses the experimental results. Section 6.5 provides an overview of the research background and related work, before concluding the chapter in Section 6.6.

6.2 Datasets

6.2.1 Dataset analysis

In this study, we secured access to the MIB dataset [19] and analysed a total of 5301 user accounts, consisting of 1950 *genuine* accounts and 3351 *fake* accounts. These accounts were sourced from five different sources, and our analysis incorporated all features outlined in Table 6.1.

The dataset contains various profile features or attributes. Each account is identified by a unique ID, name, and the user's Twitter handle. Engagement measures such as follower and following counts, tweet counts, and favourites counts are included to show user activity. The account creation date provides information on how long the account has existed, while language preference, location, and time zone indicate user demographics.

The dataset includes a variety of profile customisation, such as background colour and profile links, along with profile URLs that link to the user's profile and associated images. Verification and protection statuses are also included, which are crucial for assessing account authenticity. These features offer a comprehensive view of user behaviour, engagement patterns, and account settings, which are useful for classification tasks.

It should be noted that while Chapters 4 and 5 addressed unstructured crime data, this chapter transitions to structured datasets to evaluate features rather than identify events. This is because crimes can vary significantly due to the presence of different events, the people or things involved, or patterns. The dataset used in the first contribution was used to identify crime events, such as what happened and who or what was involved. In contrast, the dataset used to build the CA-net for representing cybercrime is characterised by a more consistent structure, with differences in the feature values. This consistency can be attributed to the nature of cyber tools or profiles, such as online accounts, computer protocols, and systems

Table 6.1: List of all account features from the dataset. Default feature values are assigned to every account upon creation.

| Performance Measures | | | |
|----------------------|------------|----------------------------|------------------------------------|
| ID | Created at | profile_image_url | profile_sidebar_border_color |
| Name | URL | profile_banner_url | profile_background_image_url |
| Screen name | Lang | profile_image_url_https | profile_use_background_image |
| Status count | Time zone | profile_background_color | profile_background_image_url_https |
| Followers count | Location | profile_link_color | Default_profile_image |
| Friends count | verified | profile_text_color | Default_profile |
| Favourite count | protected | profile_sidebar_fill_color | |
| Listed count | Geo_enable | profile_background_tile | |
| description | updated | utc_offset | |

which are generally built based on standardised frameworks. However, variations in user or application behaviour remain critical for distinguishing between normal and abnormal activities. By addressing these differences, this chapter contributes to the overall aim of the thesis by developing flexible modelling applications for both crime and cybercrime.

6.2.2 Feature selection

Permission was granted to access the MIB dataset. The dataset was subsequently processed to identify the features that may allow to differentiate between genuine and fake accounts. The analysis primarily focused on identifying features that exhibit significant similarities and differences between the two categories. To perform this analysis, features were allocated into two groups, ‘user data’ and ‘account profile’. When analysing ‘user data’, we focused on the following: followers, following, favourite count, and whether the account was verified by Twitter. Upon analysing the ‘account profile’ we examined whether the time zone and profile banner URL were default settings or edited by the user. The final analysis was based on six features: ID, Following, Follower, Favourite, Time Zone, Profile Banner URL, and Verified, presented in Table 6.2 with definitions.

A key feature that was also added was the ratio between the number of followers and the following count, referred to as the Follower-Following Ratio (FFR) in Table 6.2. FFR is important due to the discrepancy between the number of followers and following count that

Table 6.2: Selected account features we used in building the classifier model.

| Selected feature | Feature definition |
|---------------------------|---|
| ID | represents account ID |
| Followers count | number of accounts following this account |
| Friends count (following) | number of accounts followed by this account |
| FFR | represents followers/following ratio |
| Verified | true if the account is verified |
| Favourite count | number of liked tweets by the account |
| Time Zone | the specified time zone for the account |
| Profile Banner URL | indicate Twitter's default profile photo |

is commonly identified in fake accounts. A list of prior work on follower-following ratio approaches has been overviewed in [19]. Figure 6.1 illustrates the relationship between the number of followers and the number of following for both genuine and fake accounts. The plots show that genuine accounts tend to have a more diverse range of follower/following ratios, spread across the entire graph with a higher density in areas of moderate follower counts. In contrast, fake accounts are densely clustered at the lower end of the follower/following scale indicating that they typically have fewer followers. This highlights a pattern where fake accounts often have fewer followers and a high number of followings compared to genuine accounts, providing a clear basis for distinguishing between the two types of accounts. This pattern highlights the use of the FFR as a feature for classifiers in detecting fake accounts.

Previous analyses conducted by our group have also identified that including a minimum number of features is crucial for minimising the complexity of the classifier. All floating numbers were rounded to integers, and all empty cells, were adjusted to 0 for numeric cells and null for string cells.

6.3 Model Design

6.3.1 Modelling

The classifier was constructed using SONS, aiming for the fewest possible nodes while maintaining satisfactory efficiency, ease of management, and minimising the number of

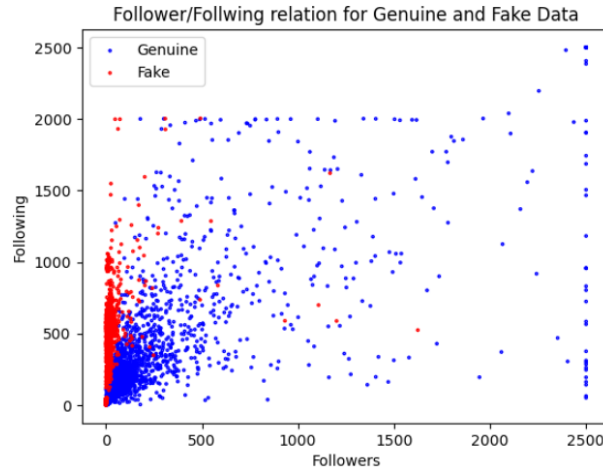


Fig. 6.1 Genuine and fake account follower/following ratio comparison using a sample from the dataset in [17].

features to reduce overall complexity. Further reduction in the number of nodes was facilitated by extending net transitions to support decision-making. This section discusses Coloured Acyclic Nets and the proposed transition extensions. It is worth noting that, to the best of our knowledge, no prior endeavours within the domain of SONS have been undertaken to identify fake social media accounts.

Approaches using current versions of SONS can model classifiers with conflicting choices. However, such modelling may not consider the feature value evaluation of tokens and is likely to increase the number of nodes. To design a binary classification using the standard acyclic nets, each decision is represented by a place, followed by two transitions that lead to a conflict, where only one transition can forward the token to the next place. This modelling could result in three places and two transitions for a single decision. We propose to decrease the number of nodes to three places and one transition by introducing transitions that ‘decide’ where to place decision tokens. Therefore, the classifier model based on CA-nets will feature fewer nodes compared to the standard SON-based modelling as depicted in Figure 6.2.

Figure 6.3 illustrates how social media platforms, such as Twitter and Facebook, are modelled using colour sets. Note that “colour” does not refer to colours literally but rather denotes a class, type, or some features. Colours can represent both social media accounts and their attributes. For example, a colour set for Twitter might be defined as:

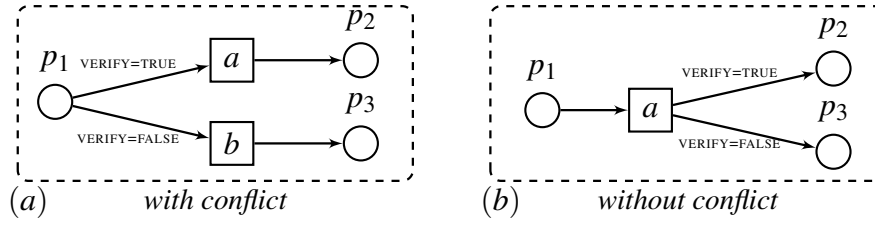


Fig. 6.2 Two different designs: (a) with conflict between a and b ; and (b) without conflict.

$Colset\ Twitter = (username : string \times FollowerCount : integer \times FriendsCount : integer)$

Where, the username represents the account's username (e.g., "User1", "Twitter55"), the FollowerCount is an integer number that indicates the number of followers for the account, and the FriendsCount refers to the number of accounts the user follows.

Similarly, a colour set can be defined for Facebook accounts. Tokens are instances of these colour sets. For instance:

- A token for a Twitter account could be ("User1", 180, 350).
- A token for a Facebook account could be ("User2", 120, 330).

Guards in the model can check a token's colour (representing its type or attributes) to ensure that only specific social media accounts can enable a transition. For example, a guard might only allow tokens representing Twitter accounts to enable a particular transition. This is because tokens are explicitly assigned to colour set, which can either represent Twitter or Facebook. Guards can evaluate the token's type or specific attributes to enable transitions. This can help the model differentiate between social media platforms and provide control over tokens based on the type of social media accounts.

To sum-up, In SONS, colours define the data type of a token. This provides robust abstraction for modelling systems with data to enhance simulation that supports a more realistic and complex systems analysis. This makes SON highly suitable for analysing complex systems like workflows, communication protocols, and distributed systems.

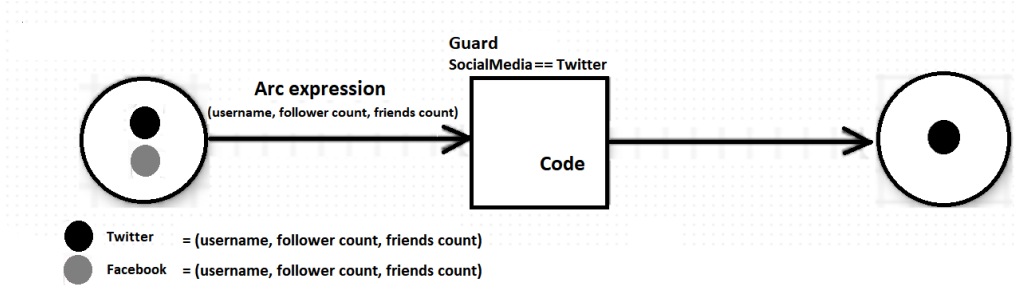


Fig. 6.3 Example showing different colours where only tokens of colour Twitter enables the transition.

6.3.2 Coloured acyclic nets

Initially, we can assume that Col is a non-empty set of colours with data values serving as the tokens. It is important to note that each colour can be structured as an ordered sequence of values, referred to as a tuple, of simpler values. Additionally, we can assume that variables, expressions, and code segments can be used over the set Col .

Definition 6.3.1 (coloured acyclic net) A coloured acyclic net (or CA-net) is a tuple $canet = (P, T, F, Col, ex, gd, C)$, where $P (= P_{canet})$ and $T (= T_{canet})$ are disjoint finite sets of places and transitions respectively, and $F (= F_{canet})$ is the flow relation included in $(P \times T) \cup (T \times P)$ such that:

1. (P, T, F) is an acyclic net.
2. col is a mapping assigning nonempty finite set $col(p) \subseteq Col$ to every place p .
3. ex is a function assigning an expression $ex(x)$ of type $col(p)$ to every arc $x = (p, t)$ or $x = (t, p)$.
4. gd is a mapping assigning a boolean expression (guard) to each transition. Here guards ensure that the input tokens for transitions are of the same colour.
5. C is a mapping assigning a code segment $C(t)$ to every transition t . ◇

A marking of $canet$ is a mapping M assigning a set of tokens $M(p) \subseteq col(p)$ to every place p . An initial marking is such that there are no tokens in the non-initial places of

(P, T, F) . Finally, the enabling and firing rules for the transitions of *canet* are as in the standard Coloured Petri Nets.

6.3.3 Classifier design

Model structural analysis

Classifiers can have any number of nodes that are based on the design specification. The classifier model proposed in this study was built using a CA-net with six places and four transitions, as depicted in Figure 6.4. Each transition received one arc from the pre-place and outputs tokens through the outgoing arcs to two post-places. In this design, c_0 was the initial place, and c_3 and c_4 were the terminal places, which is where all the tokens will eventually reside. All inner places acted as repositories to store tokens, prior to moving them forward to the next inner place repository based on the transition classification. This process continued until all tokens reached the terminal places c_3 and c_4 . In this model, places are defined as $P = \{c_0, c_1, c_2, c_3, c_4, c_5\}$ and transitions are defined as $T = \{e_0, e_1, e_2, e_3\}$.

Transitions are used to evaluate tokens received from pre-places before forwarding them to their post-places. For example, e_0 receives a token from c_0 , where the token's value is checked for *verified*. If the value of *verified* is true, then the token will be moved forward to c_3 . Otherwise, the token will move forward to c_1 . This approach will be used for subsequent transitions. Structurally, the model depicted in Figure 6.4 can be described as a simple acyclic net. However, in this model, the movement of the tokens makes it different from other net models. Typically, in net models, once a transition is enabled through placing tokens in all its pre-places, the transition moves the tokens to all the post-places. However, in the model presented in Figure 6.4, the transitions will only send meaningful tokens to specific post-places and not necessarily to all of them.

Each token in the model we construct has the value from the following colour set:

$$Col = \{\perp\} \cup (\text{INT} \times \text{BOOL} \times \text{INT} \times \text{INT} \times \text{STRING} \times \text{STRING})$$

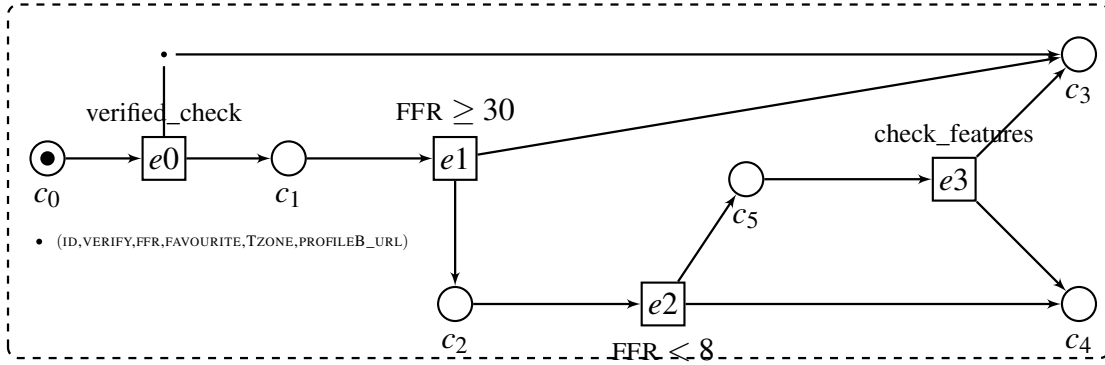


Fig. 6.4 Classifier design using CA-net.

The colour \perp is special in the sense that it is a ‘dummy colour’ introduced purely to adhere to the standard definitions of Coloured Petri Nets. The ‘dummy’ tokens are never used or consumed by transitions and are completely disregarded by the behavioural analysis (note that when a transition fires, it should output tokens to all post places). In this design, each transition is followed by two post places and whenever it fires, it sends the coloured token (after evaluation) to one post-place, and the other post-place receives a dummy token. The meaning of meaningful tokens, which belong to $\text{INT} \times \text{BOOL} \times \text{INT} \times \text{INT} \times \text{STRING} \times \text{STRING}$, is as follows:

- INT representing account identification.
- BOOL indicating if the account is verified.
- INT representing the followingfollowers ratio.
- INT representing favourite count.
- STRING representing the account time zone.
- STRING representing whether the profile banner

An example of a *token* from the dataset is:

(77, false, 17, 28, rome, https://si0.twimg.com/profile_banners/32204918/1355686088).

Table 6.3: The explanation of all nodes from the proposed classifier model.

| Node | Explanation |
|-------|--|
| c_0 | initially holds all tokens |
| e_0 | evaluate <i>tokens</i> based on the verified value, whether enabled or not. |
| c_1 | contains all <i>tokens</i> with verified not enable |
| c_3 | holds all <i>tokens</i> with verified enable, or if FFR values ≥ 30 , or e_3 check is false |
| e_1 | evaluate <i>token</i> based on FFR value, if ≥ 30 |
| c_2 | holds all <i>tokens</i> with FFR values < 30 |
| e_2 | evaluate <i>tokens</i> based on FFR value if < 8 |
| c_4 | holds all <i>tokens</i> with FFR values < 8 |
| e_3 | evaluate <i>tokens</i> based on favourite, time zone, and profile banner URL values |
| c_5 | holds all <i>tokens</i> with FFR values > 8 , or e_3 check is true. |

The account ID will make sure tokens are unique and related to specific accounts as collected from the dataset. Furthermore, tokens will carry a verified boolean value to show if the account is verified or not and will be evaluated in transition e_0 . All other values carried by a token will be evaluated in one of the transitions as we will discuss next.

Behavioural analysis

We now demonstrate the model behaviour and explain how tokens move from the starting place c_0 to the terminal places c_3 and c_4 . The movement of tokens is determined by their feature values and the evaluation of transitions which ultimately leads to the classification of tokens. Each transition will receive tokens and then forward them to the appropriate post place. Tokens will carry a list of values which will be checked at different stages of the model by transitions.

Transition e_0 : Receives a token from place c_0 then checks the value of the *verified* field. If the value is true, we assume that the token is genuine since verified accounts are very likely to be genuine accounts. Therefore, the token will be forwarded to c_3 . If, however, the verified field is false, then the token will be moved to c_1 .

Transition e_1 : Receives a token from c_1 and checks and classifies it according to the FFR value. Based on our analysis and observations, a threshold is devised to determine whether

Algorithm 3 checking if the account is *verified* [e_0]

```

1: Input: place  $\leftarrow M(c_0)$ 
2: Output: places  $\leftarrow M(c_1)$  and  $M(c_3)$ 
3: Input and output flow:
4: open (input) as  $M(c_0)$ 
5: open (output) as  $M(c_1)$  and  $M(c_3)$ 
6: do
7: input  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_0)$ 
8: out put1  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_3)$ 
9: out put2  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_1)$ 
10: for token in buffer do
11:   if verified is true then
12:     out put1.put(id, verified, FFR, favourite, tzone, profURL)
13:     out put2.put( $\perp$ )
14:   else
15:     out put1.put( $\perp$ )
16:     out put2.put(id, verified, FFR, favourite, tzone, profURL)

```

accounts have FFR values at or exceeding 30, where we assume the token are genuine and therefore moved to place c_3 . Otherwise, the token is moved to c_2 .

Algorithm 4 Second transition [e_1] FFR value check

```

1: Input: place  $\leftarrow M(c_1)$ 
2: Output: places  $\leftarrow M(c_3)$  and  $M(c_2)$ 
3: Input and output flow:
4: open (input) as  $M(c_1)$ 
5: open (output) as  $M(c_3)$  and  $M(c_2)$ 
6: do
7: input  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_1)$ 
8: out put1  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_3)$ 
9: out put2  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_2)$ 
10: for token in buffer do
11:   if FFR  $\geq 30$  then
12:     out put1.put(id, verified, FFR, favourite, tzone, profURL)
13:     out put2.put( $\perp$ )
14:   else
15:     out put1.put( $\perp$ )
16:     out put2.put(id, verified, FFR, favourite, tzone, profURL)

```

Transition e_2 : Mainly focuses on filtering and classifying tokens with FFR values less than 8. This is to move all fake tokens to place c_4 . Based on our observations, almost all fake

accounts have values less than 8. Therefore, tokens will be treated as fa and moved to c_4 . Otherwise, tokens are moved to c_5 for more assessments.

Algorithm 5 Accounts checking in $[e_2]$

```

1: Input: place  $\leftarrow M(c_2)$ 
2: Output: places  $\leftarrow M(c_4)$  and  $M(c_5)$ 
3: Input and output flow:
4: open (input) as  $M(c_2)$ 
5: open (output) as  $M(c_4)$  and  $M(c_5)$ 
6: do
7: input  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_2)$ 
8: output1  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_4)$ 
9: output2  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_5)$ 
10: for token in buffer do
11:   if FFR < 8 then
12:     output1.put(id, verified, FFR, favourite, tzone, profURL)
13:     output2.put( $\perp$ )
14:   else
15:     output1.put( $\perp$ )
16:     output2.put(id, verified, FFR, favourite, tzone, profURL)

```

Transition e_3 : Receives tokens from c_5 for the last stage of token classification. In this stage, tokens will be classified based on the values of the favourite count, time zone setting, and profile banner URL value. Based on our observation of tokens at this stage, most tokens with no favourite counts (favourite = 0), time zone not set, and profile banner URL kept to the default, come from fake accounts. Therefore, all tokens with such features will be evaluated and moved to c_4 . All other tokens are expected to come from genuine accounts and are therefore put in c_3 .

Reachability analysis

Recall that in reachability analysis we ignore the *dummy tokens* as they do not represent meaningful values.

The initial marking is where c_0 has tokens, and all other places are empty. When e_0 fires, it consumes tokens from c_0 and produces (meaningful) tokens in either c_1 or c_3 . Subsequently, when e_1 fires, it consumes tokens from c_1 and produces tokens in either c_3 or c_2 . Tokens in c_2 enable e_2 which forwards tokens to either c_5 or c_4 .

Algorithm 6 Last account evaluation and classification [e_3]

```

1: Input: place  $\leftarrow M(c_5)$ 
2: Output: places  $\leftarrow M(c_3)$  and  $M(c_4)$ 
3: Input and output flow:
4: open (input) as  $M(c_5)$ 
5: open (output) as  $M(c_3)$  and  $M(c_4)$ 
6: do
7: input  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_5)$ 
8: out put1  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_4)$ 
9: out put2  $\leftarrow$  (id, verified, FFR, favourite, tzone, profURL) from  $M(c_3)$ 
10: for token in buffer do
11:   if favourites_count = 0 and time_zone = null and profile_banner = null then
12:     out put1.put(id, verified, FFR, favourite, tzone, profURL)
13:     out put2.put( $\perp$ )
14:   else
15:     out put1.put( $\perp$ )
16:     out put2.put(id, verified, FFR, favourite, tzone, profURL)

```

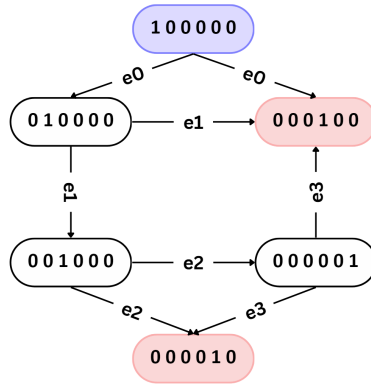


Fig. 6.5 Schematic reachability graph.

Finally, e_3 receives tokens from c_5 and produces tokens in either c_3 or c_4 . The schematic reachability graph in Figure 6.5 shows potential system states when processing a single initial token represented by ‘1’ (note that the lack of token is represented by ‘0’, and each marking shows the content of the places in the order $c_0 \dots c_5$).

6.4 Experiment

After completing design and analyses of the CA-net model, we carried out its experimental evaluation on the selected dataset comprising 5301 accounts. As mentioned previously, the

aim was to analyse and classify data using the proposed CA-net classifier. Figure 6.6 shows the distribution and evaluation of tokens within the designed classifier.

Initially, all 5301 tokens moved from place c_0 to transition e_0 , where they are evaluated for whether the account is verified or not. This is done by checking if the token's *verified* value is set to 1. Since none of the accounts were verified, all tokens proceeded to transition e_1 . Here, the tokens were evaluated based on whether the FFR was greater than or equal to 30. Out of the total, 3888 tokens had an FFR less than 30 and were therefore moved to place c_2 . The remaining 1413 tokens, with an FFR of 30 or more, were transferred to place c_3 , a terminating place, indicating that they represent genuine accounts.

The 3888 tokens in place c_2 were then evaluated in transition e_2 to check whether the FFR was less than 8. From this evaluation, 3127 tokens were found to have an FFR less than 8 and were moved to place c_4 , another terminating place holding tokens representing fake accounts. The remaining 761 tokens forwarded to place c_5 for further evaluation.

These 761 tokens underwent another evaluation in transition e_3 . This time, 579 tokens were moved to place c_3 , and the remaining 182 tokens were moved to place c_4 , concluding the token evaluation process.

The model effectively demonstrates the use of transitions and places to systematically evaluate and process tokens based on specific criteria. By moving through different transitions, tokens are filtered and directed to their respective terminating places. This ensure that each token is thoroughly assessed according to the established rules. This structured approach not only highlights the efficiency of the model in handling large datasets but also underscores its potential in applications where systematic evaluation and processing are crucial. The results from this experiment provide valuable insights into the model's performance and its ability to handle complex evaluation processes.

6.4.1 Evaluation and discussion

Our primary goal was to demonstrate that SON can be used to model classifiers based on coloured tokens. After the completion of the experiment, the results were analysed using a confusion matrix. The model effectively demonstrates the use of transitions and places

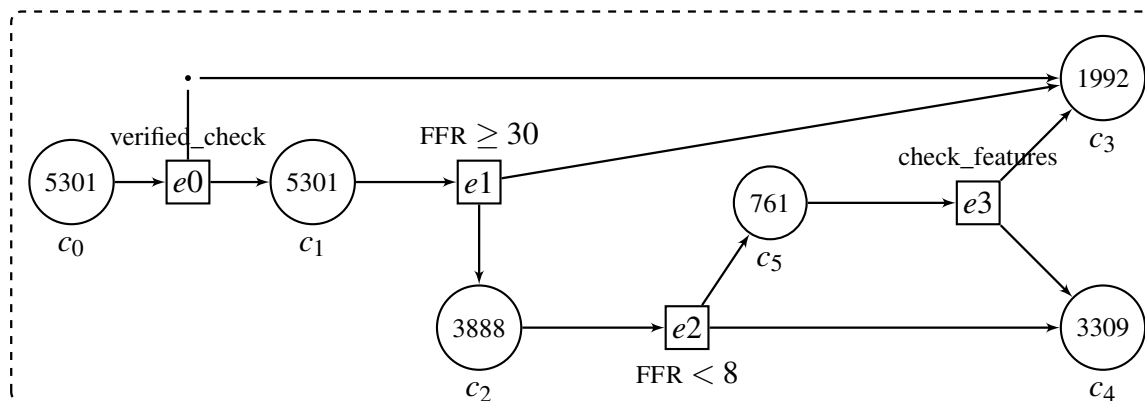


Fig. 6.6 Tokens distribution in the CA-net.

to evaluate and classify tokens representing Twitter accounts as either genuine or fake. By moving through different transitions, tokens are filtered and directed to their respective places. This ensures that each token is thoroughly assessed according to the established rules. Genuine accounts are classified into place c_3 , while fake accounts are classified into place c_4 .

The confusion matrix results show that the classification model is very reliable. The sensitivity measurement has a robust performance of 97.13%, while specificity is also high at 97.08%. Precision, reflecting the true positives among all positive predictions is 95.08%, which indicates a low rate of false positives. The overall accuracy of the model stands at 97.09%. The F1 score, the balance between precision and sensitivity, is 96.09%. The model's efficacy is further supported by the Matthews Correlation Coefficient of 93.79%, which acts as a reliable measure of classification performance despite dataset imbalance.

Table 6.4: Performance evaluation metrics for the classifier.

| Measure | Score |
|----------------------------------|---------|
| Sensitivity | 97.13 % |
| Specificity | 97.08 % |
| Precision | 95.08 % |
| Accuracy | 97.09 % |
| F1 Score | 96.09 % |
| Matthews Correlation Coefficient | 93.79 % |

6.5 Related Work

In recent years, efforts have been devoted to identify fake social media accounts through the implementation of various methods and techniques. One example is the use of Machine Learning (ML) which led to the publication of numerous novel approaches and algorithms. In a recent study [22], the authors evaluated 71 properties extracted from profile and timeline information to assess the effectiveness of common supervised ML methods in detecting malicious Twitter accounts for classification purposes. The authors employed various methods, including Bayesian classifiers, feature selection algorithms, decision trees, random forests, and neural networks to construct detection devices for identifying and screening automated fake accounts on Twitter. Through this approach, the authors identified that a random forest operating with 19 features achieved an average accuracy of 94%, highlighting the crucial role of feature quality and feature subset selection in prediction accuracy. An alternative approach to distinguish between fake and genuine Twitter accounts was presented by [30]. The authors combined ML algorithms such as J48, Random Forest, Naive Bayes, and KiNN with reduction techniques, including PCA and correlations. By applying three phases, starting with data preprocessing, reduction, and classification, they demonstrated that combining the Random Forest algorithm with correlation-data reduction achieved an accuracy of 98.6%.

An important study focuses on the MIB dataset [19]. The authors discuss the limitations of current methods and propose a novel approach using ML classifiers and a baseline dataset consisting of verified human and fake follower accounts. The authors were able to evaluate numerous features and rules proposed by academia to identify the most effective features for detecting fake followers. The aim of the research was to enhance the detection of fake Twitter followers and offer insight into the characteristics that can optimise approaches to distinguish fake from genuine followers. Using the baseline dataset, they found that algorithms based on classification rules were less effective at detecting fake followers compared to ML classifiers.

An additional layer of complexity in attempting to distinguish between genuine and fake Twitter accounts through the use of algorithms is the challenge of detecting bots. In a recent review [57], the authors discussed the difficulties of bot detection on Twitter. The authors explored the behaviour and characteristics of bots on Twitter, comparing different

databases that can be used for detection. They identified several complexities in detecting bots on Twitter, a challenge accentuated further due to the evolving nature and complexity of bots. The ability to detect bots on Twitter is crucial to identify fake accounts and minimise their malicious or unwanted intentions. Further research is required to identify an optimal approach to this issue.

One approach to identifying fake Twitter accounts and bots involves analysing posted tweets. The authors in [65] utilised Petri Nets modelling to evaluate and rank the credibility of 1000 tweets on Twitter. They incorporated the uniform resource locator (URL) feature in tweets and designed a Petri net model to visualise the ranking process. The authors then assessed the effectiveness of the Petri net model in classifying tweets into credible and untrustworthy categories based on specific features of the tweet. They revealed that the proposed Petri Net model outperformed Support Vector Machine (SVM) based methods in terms of the Normalised Discounted Cumulative Gain (NDCG) metric - an evaluation of the quality of recommendation and retrieval system - with a score of 0.86 compared to the SVM-based method 0.37. An extension of Petri Nets modelling is through the use of Coloured Petri Nets (CPNs), which possess useful properties of Petri Nets but can also extend the baseline formalism and optimise the distinction between tokens. In a more recent study [66], the authors proposed a novel CPN model and evaluated its ability to detect and block the propagation of misleading information through online social networks based on 1003 newsworthy tweets. The authors compared their novel CPNs to the existing approaches in the literature and identified several limitations of previous approaches. They revealed that their proposed CPN model was able to effectively detect misleading information tokens in the analysed dataset, while blocking their propagation, achieving the Precision of 86%, Recall of 91%, and Accuracy of 85%.

Malicious account detection on social networks is addressed in [14] by creating a feature engineering strategy to improve machine learning models' ability to distinguish fake accounts. The method involves augmenting social network datasets with new features that capture behavioural patterns indicative of malicious activity. This process includes data collection, feature development, training diverse machine learning models, and validating

their effectiveness on real-world datasets from Twitter and Instagram. The experimental results demonstrate enhanced detection of fake accounts, showcasing the strategy's efficacy in ensuring online safety and integrity. Whereas [54] introduces a novel system, the Publically Privacy Protection System (3PS), aims to enhance security in online social networks (OSNs) by detecting malicious profiles. This system uses an Enhanced Support Vector Machine-Neural Network (E_SVM-NN) classifier with feature reduction techniques for more effective identification of suspicious accounts based on unusual activity patterns and excessive content sharing. The effectiveness of this methodology is confirmed through empirical tests with fake OSN accounts which indicate a potential to improve the security posture of users against the threats posed by malicious profiles. Both methodologies are distinguished in the field of machine learning; nonetheless, this study identifies a deficiency in visualisation, which we have addressed through the utilisation of formal method tools based on graphs.

A review of the existing literature suggests that there is still room for further research in the design of classifiers using formal methods. Consequently, we have presented a model based on SONS for the evaluation and classification of Twitter accounts as genuine or fake.

6.6 Conclusions

A coloured extension of the SON model (CA-nets) has been introduced and developed for the classification of Twitter user accounts into genuine and fake classes. The design of the classifier was based on a thorough manual analysis of a sizable dataset comprising approximately 5301 accounts, comprising both genuine and fake classes. The outcomes of our comprehensive analysis revealed that the CA-net model attains exceptional performance metrics, namely, the Precision rate of 95%, Accuracy level of 97%, and F1 Score of 96% (as shown in Table 6.4).

The primary objective of our study was to introduce a CA-net model capable of analysing coloured token values. This enhancement has been achieved by equipping the model's transitions with evaluation capabilities. This development allows for a more sophisticated classification process, ensuring that each token is assessed according to a set of defined rules.

Our approach involved manually analysing the dataset to understand the key features that distinguish genuine and fake accounts, which forms the basis for the design of the CA-net model and its evaluation mechanisms.

Moreover, structurally the CA-net model has fewer nodes compared to traditional SON modelling approaches, leading to a more efficient classification process. Additionally, the CA-net model offers more modelling options for SONS, allowing for greater flexibility and adaptability in various scenarios. This structural efficiency and flexibility make the CA-net model a powerful tool for handling classification tasks.

Chapter 7

Concluding Remarks

In this thesis, we have proposed and developed methods to extract SON components from unstructured text. The thesis proposes and discusses approaches for mapping and extracting acyclic nets and transitions from unstructured text. Based on practical experiments, mapping rules between SONs and natural language are proposed. To do that, a custom NER model tailored to identify crime-related entities within the text is proposed. Moreover, methods to accurately identify and extract verbs have been proposed and experimented with. The thesis also introduces a Coloured Acyclic Net (CA-net) model which is a SON extension that incorporates additional attributes to help improve visualisation.

Chapter 4 presents an analysis and proposes a mapping of natural language texts into SON models. The design introduces a construction of acyclic nets representing sequences of events representing proper nouns, and the construction of transitions representing actions or state changes representing verbs. Furthermore, three extraction methods were introduced, including extracting only the main (root) verbs in sentences, and extracting main verbs and common crime-related verbs extracted from crime stories. The last method involves extracting all verbs present in the text. This mapping is crucial to accurately represent narrative structures and ensure that the SON models reflect the underlying stories correctly.

Chapter 5 builds on the findings of Chapter 4 and presents the design of a customised NER model capable of identifying acyclic nets. Furthermore, it proposes a syntax-based pattern for identifying verbs, inspired by human modelling approaches. This process involves

the analysis of the syntactic structure of sentences in order to extract verbs that indicate transitions in SON models. This approach aims to improve the precision and relevance of the identified actions. In this method of verb identification, verbs were identified from manual models and compared to the presence of verbs in the text. The syntactical structure of the verb in the sentence was considered for evaluation using the dependency parser. Seven patterns were identified, where any verb that has a similar syntactical pattern is considered to represent transition.

Finally, Chapter 6 introduces coloured acyclic nets, which add extra detail, complexity, and scalability to the SON acyclic net modelling process. In this chapter, acyclic nets have been extended to improve visualisation by incorporating coloured features and codes on transitions leading to a reduction in acyclic net size.

7.1 Limitations

Even though the automatic extractor has shown greater consistency in extraction compared to manual modelling, it has its own limitations. The extractor has successfully identified comprehensive list of verbs from text, however, it assigns verbs to entities within the same sentence. This can result in the construction of additional transitions, which are not always accurate. A more precise verb-entity assignment could be achieved by analysing dependency relation between verbs and entities within the same sentence.

Moreover, while the NER model performed well in identifying entities, it showed reduced identification accuracy when tested on a new dataset. This suggests that further training of the NER model is needed, using a more diverse crime-related dataset.

Additionally, the modelling process lacks the capability to recognise concurrent events (verbs) within acyclic nets. Addressing this limitation could enable the representation of events occurring simultaneously, particularly when an entity is involved in multiple activities at the same time. For instance, a person might be following a criminal while reporting to the police on the phone.

Furthermore, the extractor assumes that all communication links are synchronous, meaning events are happening at the sometime or one after another. However, this does not always show the exact sequences of events. As a result, the automatic extractor lacks the ability to model asynchronous communication, which could provide a more accurate representation of event sequences.

7.2 Future work

Several research and development opportunities can be identified for future work, including:

Enhancing the capabilities of the NER model. This can be achieved by training the model on a more diverse set of crime stories to improving the model's comprehensiveness and robustness. This can aid in the extraction from different narrative complexities and contexts, which will improve the generalisability and make the tool suitable for real-world applications.

Developing methods to extract action and stative verbs. This approach involves analysing the linguistic properties of verbs to distinguish between action verbs and stative verbs. Accurate identification of these verbs will improve the identification and modelling of *transitions* as well as *places*. This will provide a clearer and more precise representation of events and states within the SON framework.

Developing further rules based on synchronous communication. Different actions that occur simultaneously, such as driving and make a call at the same time, could be represented by two transitions receiving tokens involved in synchronous communication.

Developing rules based on asynchronous communication. Asynchronous communication in crime modelling could be used to model sequences of events. For example, it could capture action sequences like: `entering_a_house` → `stealing_goods` → `fleeing_the_house`, showing the precedence of, e.g., `entering_a_house` before `stealing_goods`. This would also reflect implicit actions, for example, `fleeing_the_house` implying that the burglar was inside the house.

Integrating the proposed extraction method as a SONCraft plug-in. SONCraft is a tool for building SON models, and incorporating extractor capabilities will significantly enhance the tool's functionality. SONCraft will become more powerful for users interested in rapidly constructing and analysing SON models directly from text sources. Additionally, investigating the identification and construction of concurrent and conflicting events within *acyclic nets* is an important area of future research. Developing methods to detect and represent events that occur concurrently and events with conflicting choices will allow for a better understanding of complex scenarios and relationships within text resources.

Enhancing the Coloured Acyclic Net (CA-net) model. Integrating the classifier into the SONCraft tool will diversify the usability of the tool. Furthermore, exploring the possibility of automating feature selection using machine learning algorithms and techniques could improve the model's performance. Investigating the use of ML algorithms in feature selection will provide users with more flexibility in constructing and analysing SON models.

Appendix A

A.1 Modelling examples

A.1.1 More models for STORY A

| Method | | | |
|---------|----------|-----------|----------|
| 1 | 2 | 3 | 4 |
| shot | shot | shot | shot |
| say | killed | killed | killed |
| ran | say | say | saw |
| walked | saw | saw | enter |
| shot | left | enter | left |
| found | ran | left | ran |
| added | shot | ran | shot |
| is | walked | shot | walked |
| gone | fired | walked | fired |
| know | shot | collapsed | picked |
| charged | found | fired | carrying |
| | added | shot | fled |
| | carrying | found | gone |
| | is | added | meet |
| | carrying | picked | buy |
| | gone | carrying | know |
| | know | fled | |
| | charged | carrying | |
| | | gone | |
| | | meet | |
| | | buy | |
| | | know | |
| | | charged | |

Table A.1 Comparison of verb extraction methods applied to STORY A.

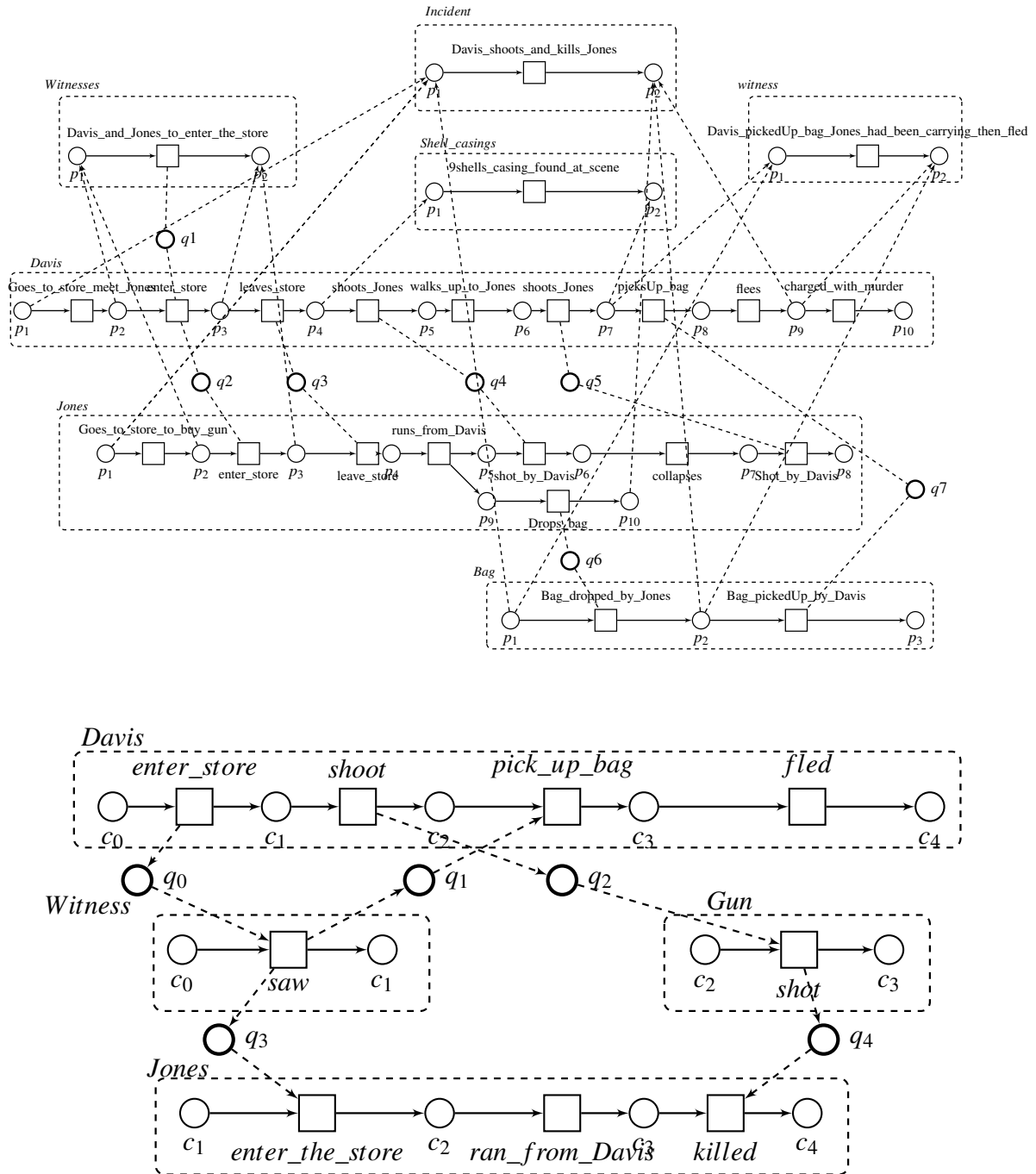


Fig. A.1 Two models by two modellers for STORY A

| Entities | | | | |
|-----------|-----------|---------------|-----------|--------------------|
| MODELLER1 | MODELLER2 | MODELLER3 | MODELLER4 | Automatic |
| Davis | Davis | Davis | Davis | Davis |
| Jones | Jones | Jones | Jones | Jones |
| witness | Witnesses | Witnesses | Witnesses | Witnesses |
| Bag | Store | A witness | Gun | Parking_lot |
| | Bag | Bag | | A witness |
| | | Shell casings | | 09mm_handgun |
| | | incident | | 45_caliber_handgun |

Table A.2 Extracted Entities from STORY A.

A.1.2 Crime B story Models and analysis

“Marvin Arrell Stanton, 49, shot and killed Jessie James Hamilton at a Raceway gas station in Texarkana, Arkansas. The shooting took place after a 25-second physical confrontation in the gas pump area of the business. According to the prosecution, the confrontation took place over a parking space after Stanton told Hamilton to "move [his] f–ing truck." According to Detective Jason Haak, Stanton started the physical confrontation by pushing Hamilton. After Hamilton gained the upper hand in the fight, he was pulled off Stanton by a female bystander. Immediately after, Stanton fired at Hamilton from a distance of approximately three feet. According to law enforcement, Stanton attempted to render medical aid to Hamilton before the police arrived by applying pressure to his wounds. When police arrived, Stanton’s 45 caliber pistol was still holstered to his belt and he had two magazines with 26 rounds of ammunition. Stanton had a permit to carry a concealed weapon and was certified to teach concealed carry classes. Stanton was found guilty of murder and was sentenced to life in prison plus 15 years.” [15]

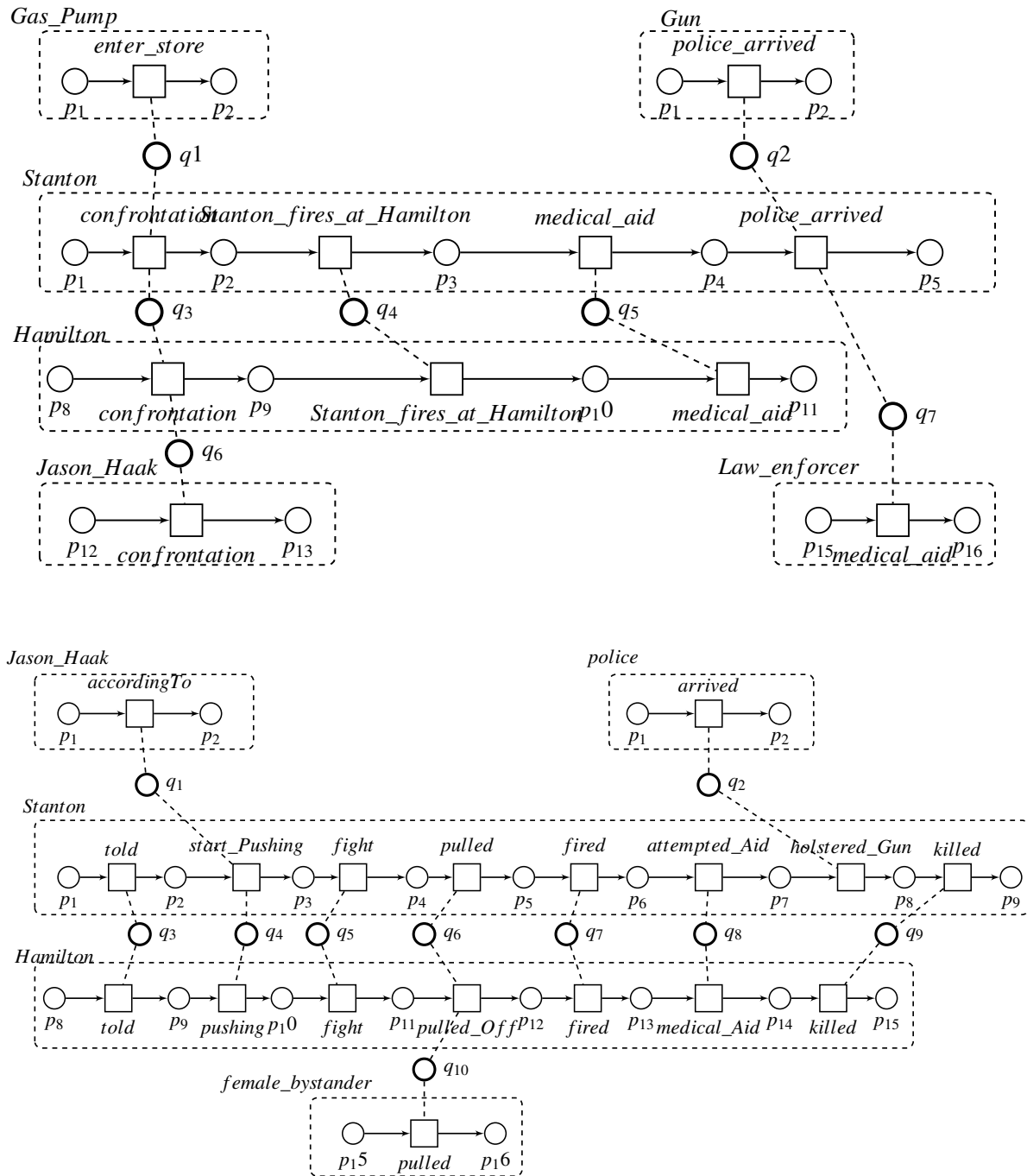


Fig. A.2 Two models created by different SON expert users depict the events from STORY B and demonstrate close entity identification.

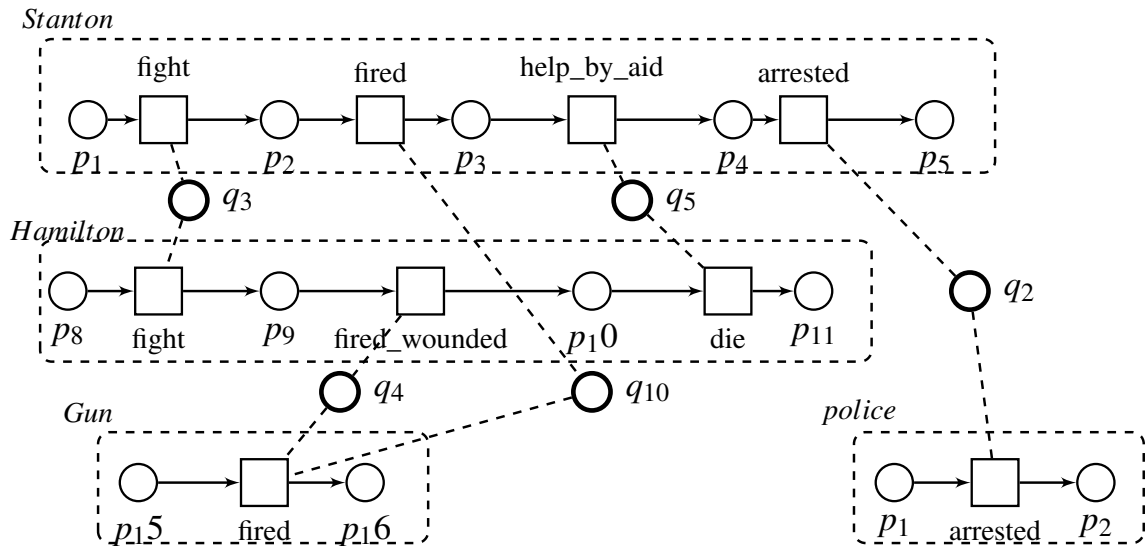
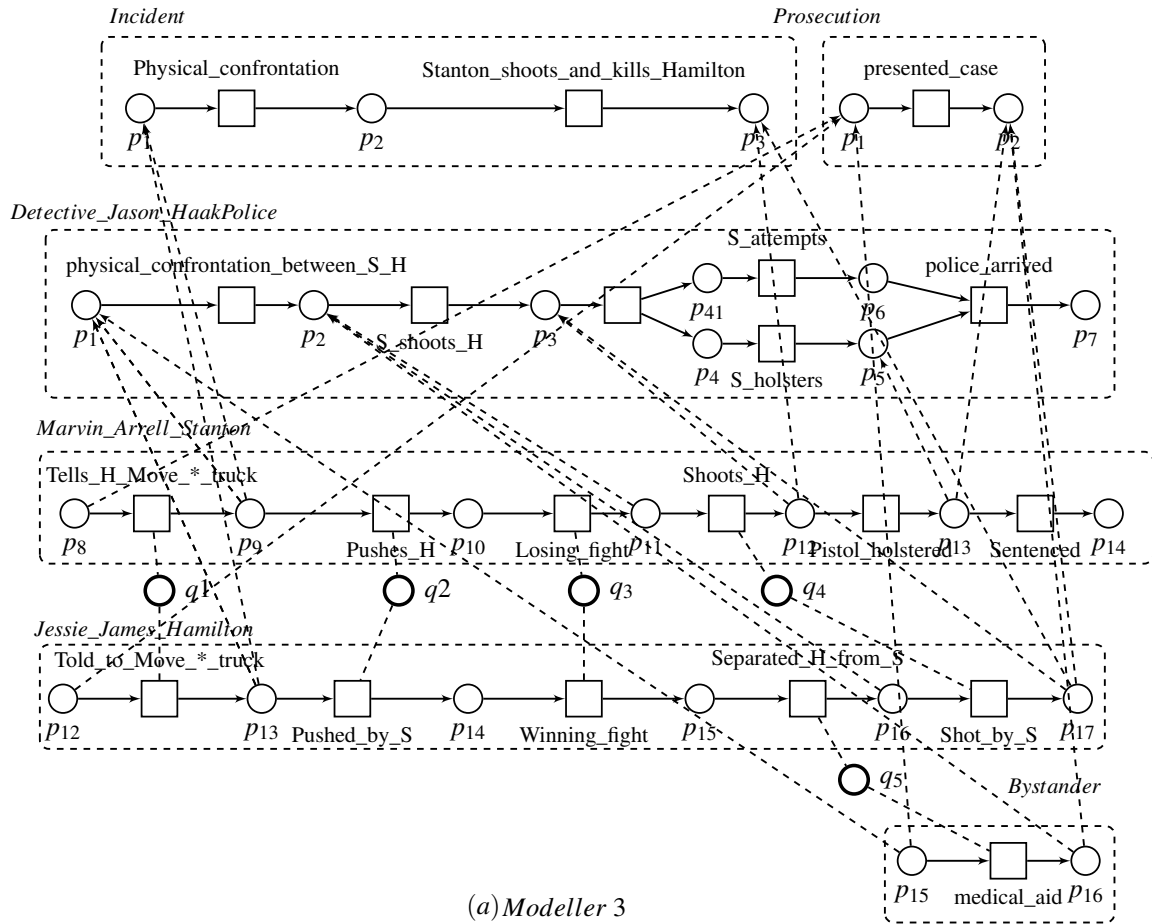


Fig. A.3 Two models created by different SON expert users depict the events from STORY B and demonstrate close entity identification.

| Method | | | |
|-----------|-----------|-----------|-----------|
| 1 | 2 | 3 | 4 |
| shot | shot | shot | shot |
| took | killed | killed | killed |
| took | took | took | took |
| started | According | According | According |
| pulled | took | took | took |
| fired | told | told | told |
| attempted | According | move | move |
| holstered | started | ing | ing |
| had | pulled | According | According |
| found | fired | started | started |
| | According | pushing | pushing |
| | attempted | gained | gained |
| | arrived | pulled | pulled |
| | arrived | fired | fired |
| | holstered | According | According |
| | had | attempted | attempted |
| | carry | render | render |
| | concealed | arrived | arrived |
| | concealed | applying | applying |
| | carry | arrived | arrived |
| | found | holstered | holstered |
| | sentenced | carry | teach |
| | | concealed | found |
| | | certified | sentenced |
| | | teach | |
| | | concealed | |
| | | found | |
| | | sentenced | |

Table A.3 Comparison of verb extraction methods applied to STORY B.

| Entities | | | | |
|--------------|------------------|-------------|-----------|--------------------|
| MODELLER1 | MODELLER2 | MODELLER3 | MODELLER4 | Automatic |
| Gas_Pump | Haak | Incident | Stanton | Jason |
| Stanton | Stanton | Prosecution | Hamilton | Stanton |
| Hamilton | Hamilton | Haak_police | Gun | Hamilton |
| Jason_Haak | Female_bystander | Stanton | police | police |
| Gun | police | Hamilton | | 45_caliber_handgun |
| Law_enforcer | | Bystander | | |

Table A.4 Extracted Entities from STORY B.

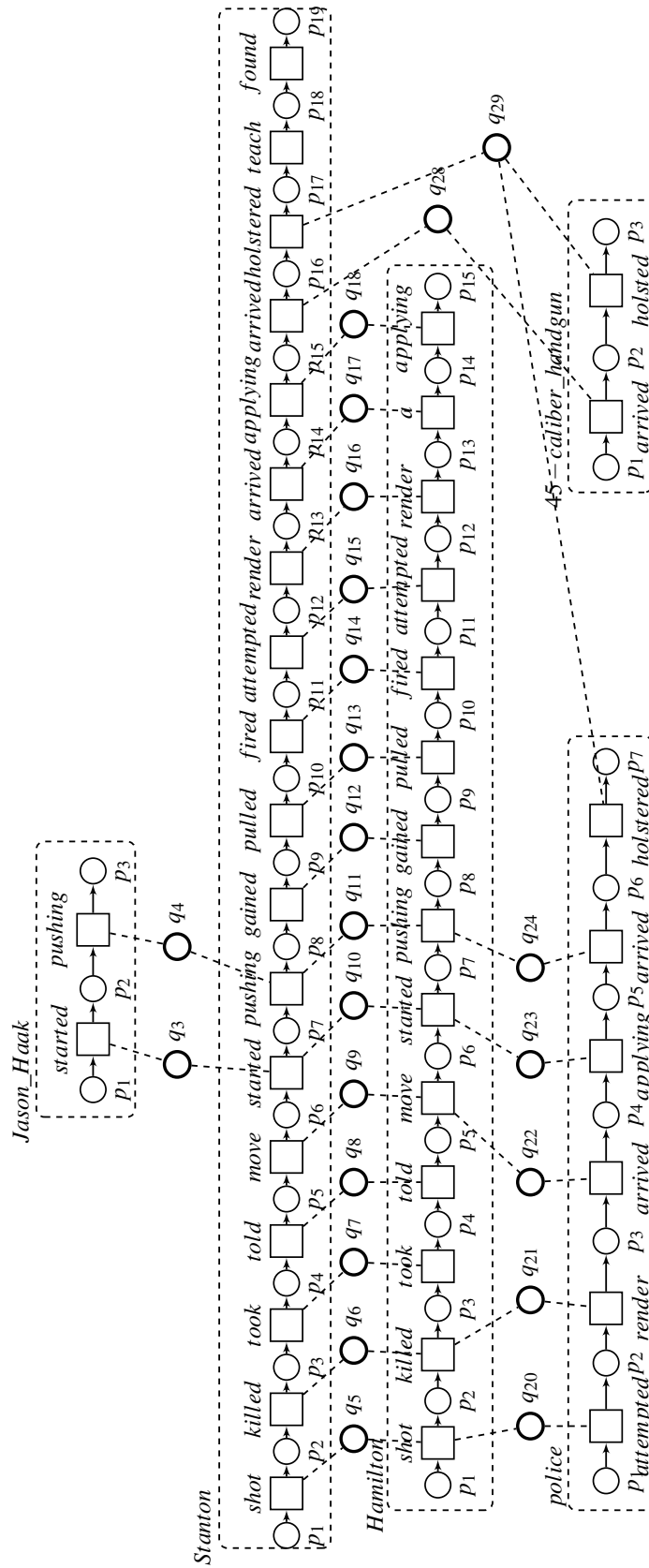


Fig. A.4 A potential SON model could be developed based on the results of the extraction for the same story (STORY B). The extraction approach identified more entities and verbs compared to human identification in Figures A.2 and A.3.

A.1.3 Crime C story Models and analysis

“Andre Sinclair, 22, shot and killed Keith Byrne, 41, during a road rage incident in Davie, Florida. Byrne was on the phone with a friend while driving when he accidentally cut off a BMW driven by Sinclair’s girlfriend. According to the friend on the phone, Byrne wanted to apologize for the mistake at the next light. However, despite pleas from his girlfriend to stay in the car with her and their toddler daughter, Sinclair got out of his car and shot Byrne in the chest. Byrne, who was also a concealed handgun permit holder, fired back at Sinclair. Byrne died at the scene and Sinclair died of his injuries two days later. Police Sgt. Mark Leone called the incident “pointless and silly.” He added that if Sinclair had survived, the police would have identified him as the primary aggressor and charged him with murder.” [15]

| Method | | | |
|--------|-----------|------------|------------|
| 1 | 2 | 3 | 4 |
| shot | shot | shot | shot |
| was | killed | killed | killed |
| wanted | was | driving | driving |
| got | According | cut | cut |
| fired | wanted | driven | driven |
| died | got | According | According |
| Sgt | shot | wanted | wanted |
| called | concealed | apologize | apologize |
| added | fired | stay | stay |
| | died | got | got |
| | died | shot | shot |
| | Sgt | concealed | concealed |
| | called | fired | fired |
| | added | died | died |
| | charged | died | died |
| | | injuries | called |
| | | called | identified |
| | | added | charged |
| | | survived | |
| | | would | |
| | | identified | |
| | | charged | |

Table A.5 Comparison of verb extraction methods applied to STORY C.

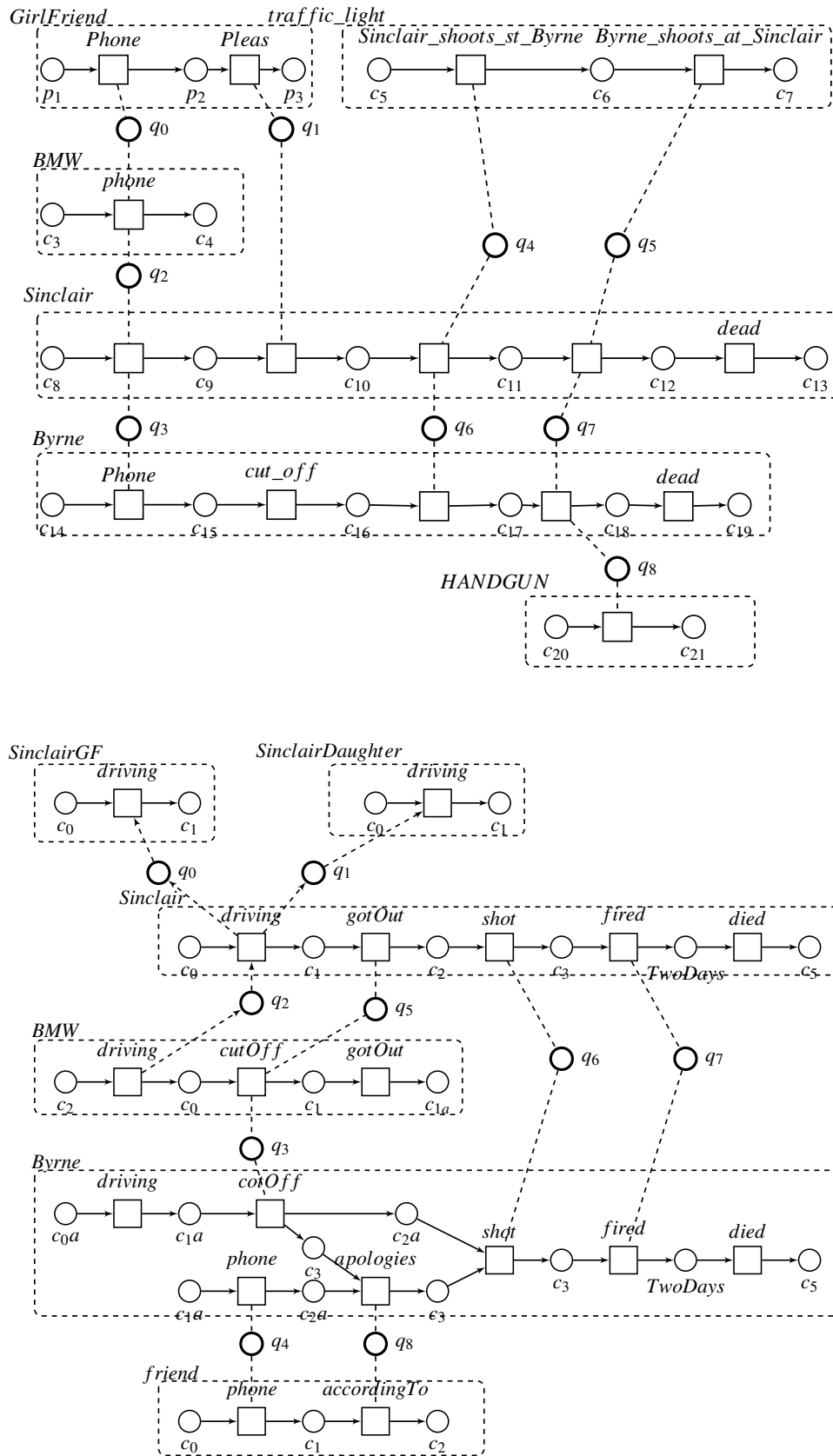


Fig. A.5 Two models created by different SON expert users depict the events from STORY C.

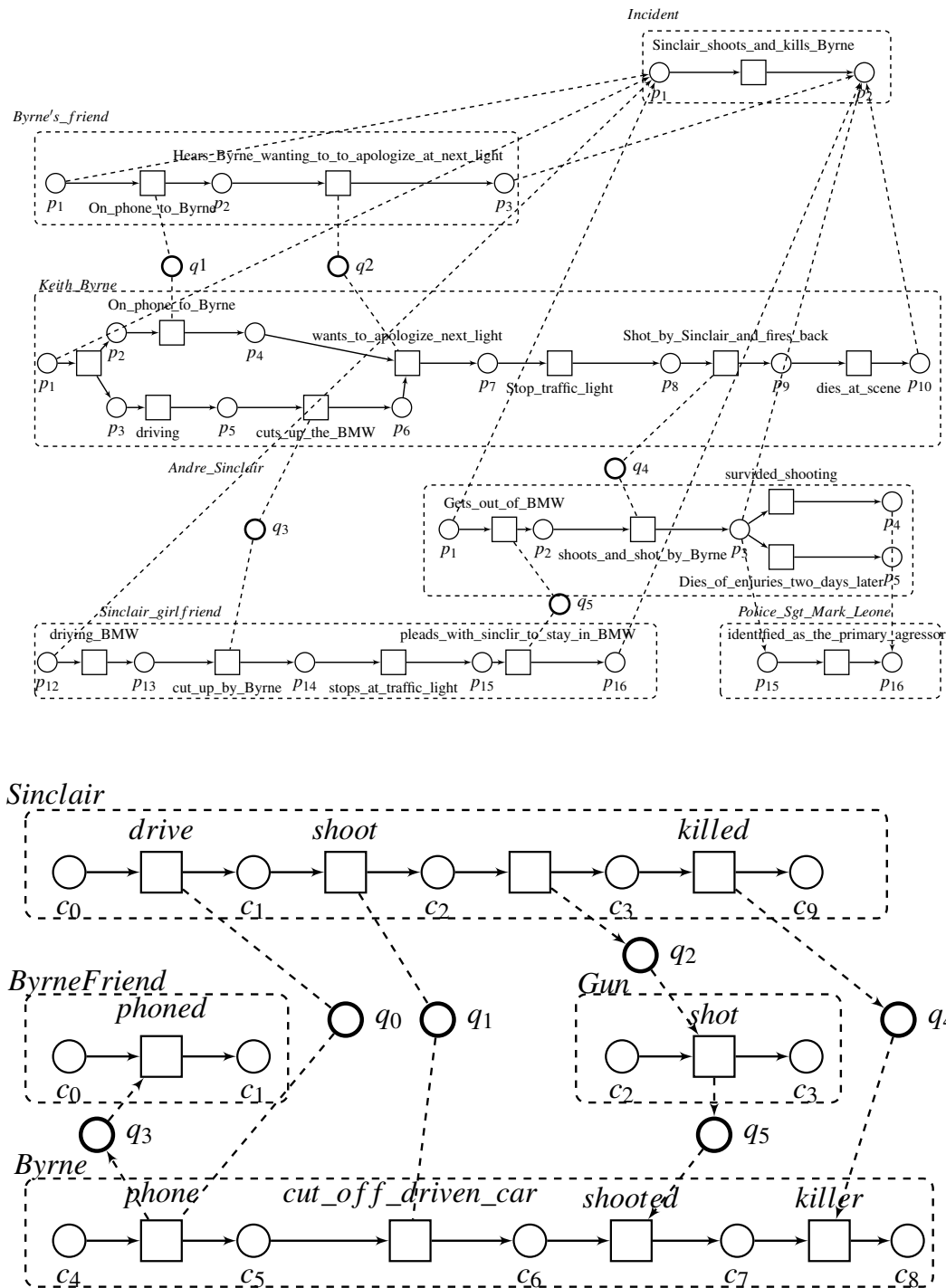


Fig. A.6 Another two models created by different SON expert users depict the events from STORY C.

| Entities | | | | |
|------------------------|---------------------|---------------------|-------------|------------------------|
| MODELLER1 | MODELLER2 | MODELLER3 | MODELLER4 | Automatic |
| Girlfriend and Toddler | Sinclair_Girlfriend | Incident | Sinclair | Sinclair |
| BMW | Sinclair_daughter | Byrne_friend | ByrneFriend | Byrne |
| Sinclair | Sinclair | Keith_Byrne | Byrne | Davie, Florida |
| Traffic light | BMW | Mark_Leone | Gun | Sinclair's girlfriend |
| Byrne | Byrne | Andre_Sinclair | | their toddler daughter |
| Handgun | friend | Sinclair_girlfriend | | Mark Leone |

Table A.6 Extracted Entities from STORY C.

A.1.4 STORY D models and analysis

“On February 21, 2010, 30-year-old Lashawn Davis was allegedly shot and killed by Vincent Williams, 31, outside an E-Z Mart convenience store in Jefferson County. Several witnesses told police that the two men had shot one another. According to a police report, Davis was found slumped in the driver’s seat of a 1999 gold Mercury Grand Marquis with his feet hanging out of the car and his body facing south. A .45 handgun was found nearby. Police found Williams on the south side of the store, bleeding, with a 9mm handgun nearby. Both men were transported to a local hospital where Davis was pronounced dead and Williams was taken into surgery. Williams reportedly told police that Davis had tried to steal his car. Williams had a concealed handgun permit.” [15]

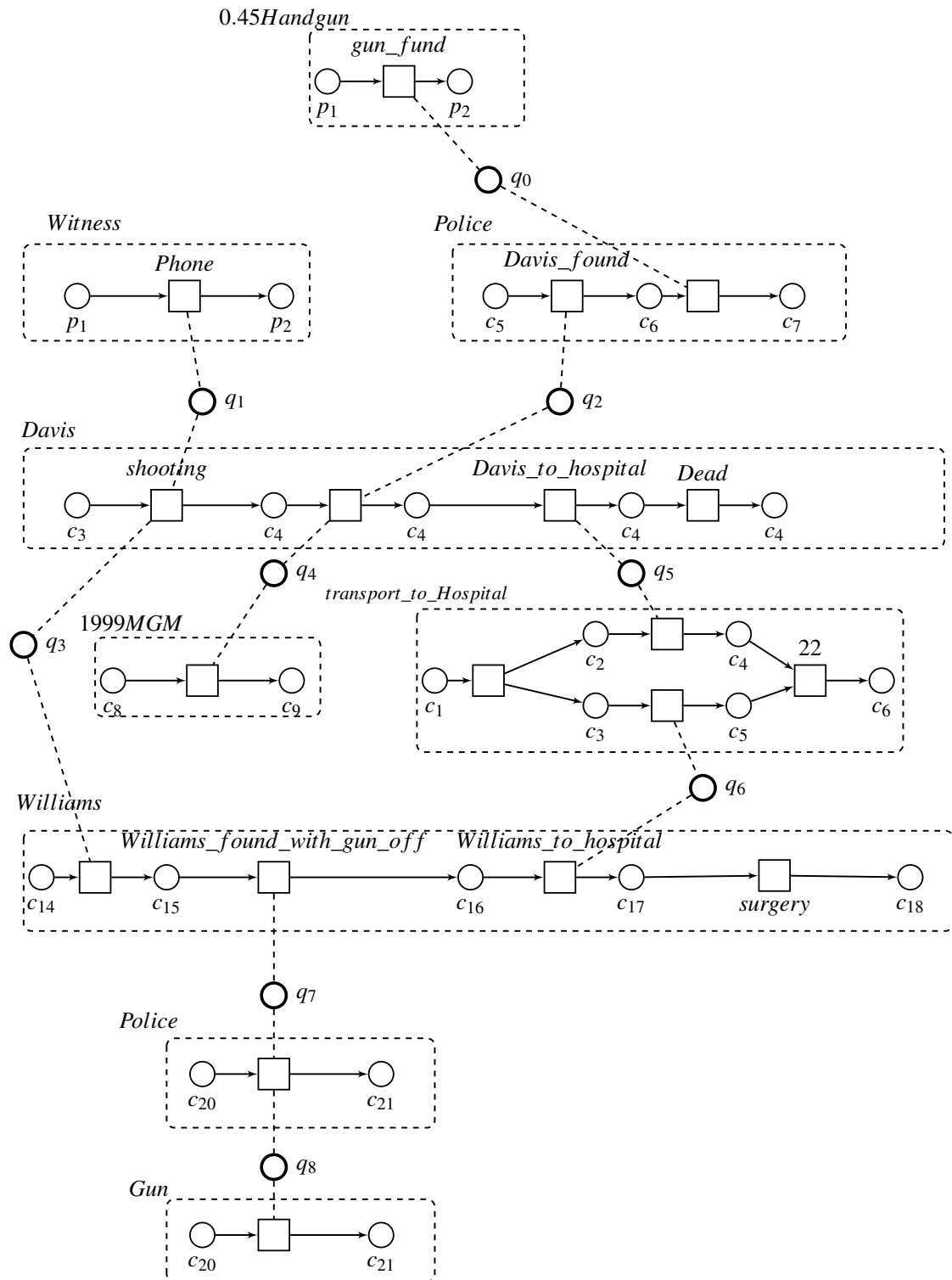


Fig. A.7 A models created by different SON expert user depict the events from STORY D.

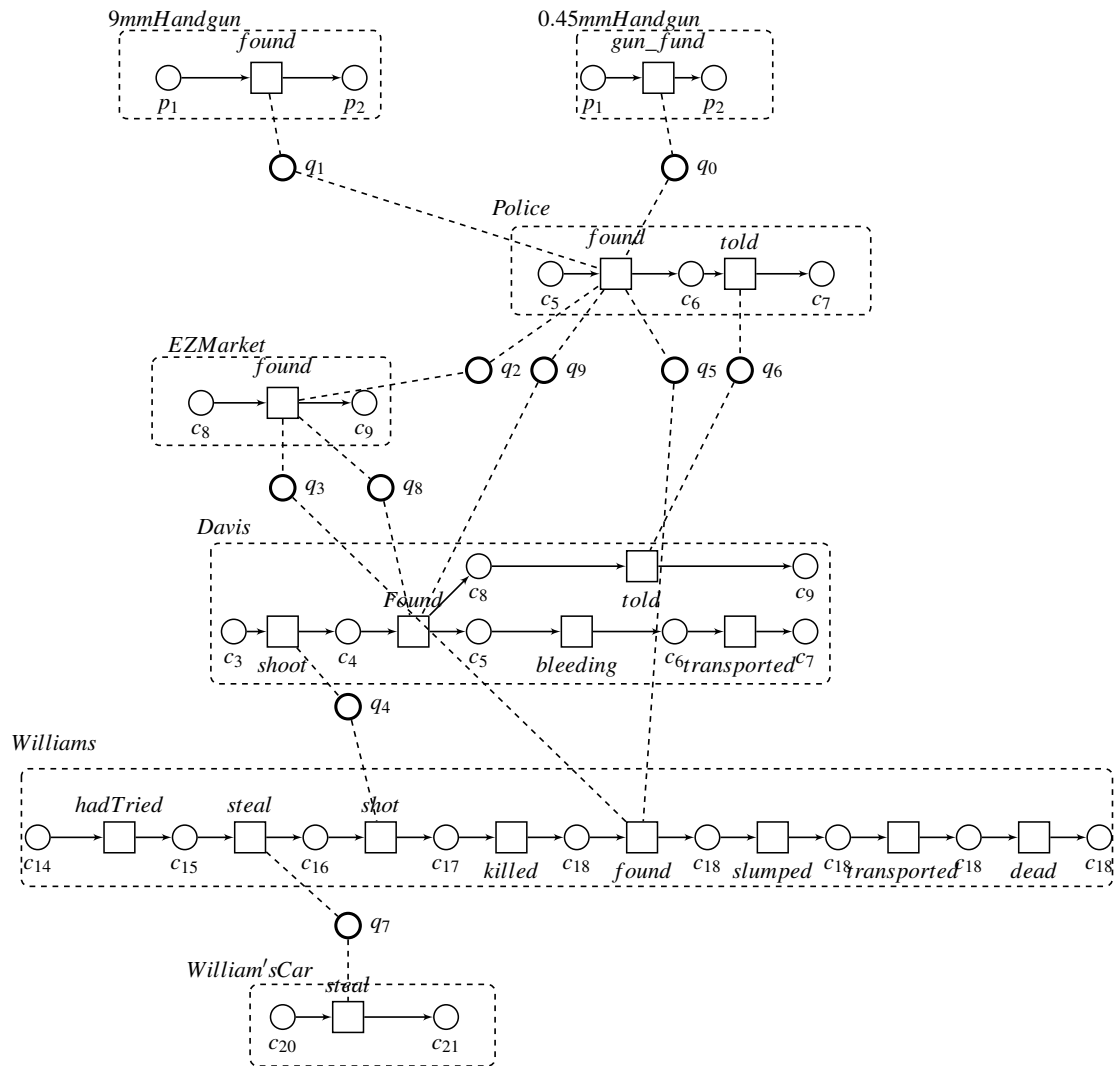


Fig. A.8 A models created by different SON expert user depict the events from STORY D.

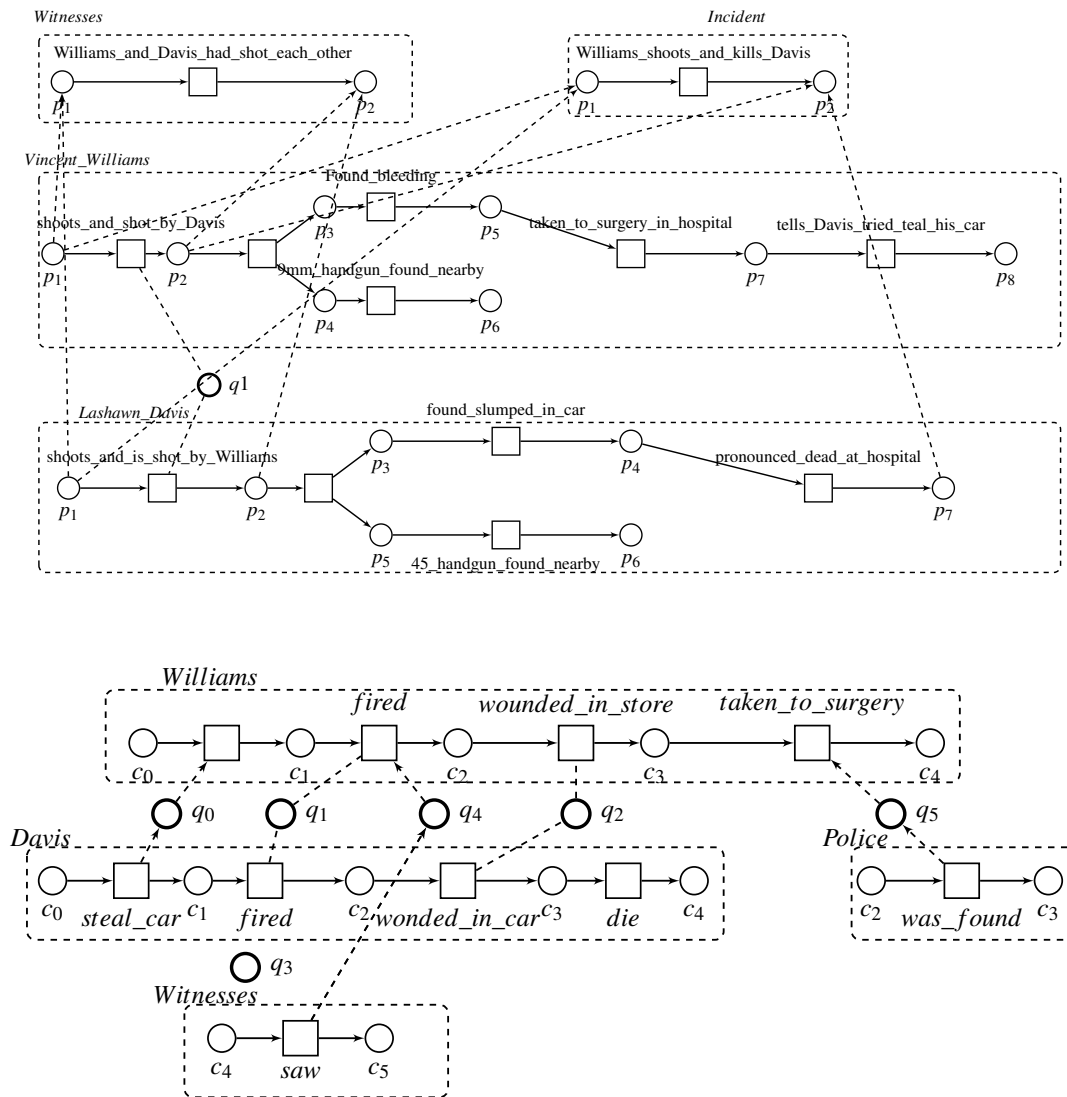


Fig. A.9 Two models created by different SON expert users depict the events from STORY D.

| Method | | | |
|-------------|-------------|-------------|------------|
| 1 | 2 | 3 | 4 |
| shot | shot | shot | shot |
| told | killed | killed | killed |
| slumped | told | told | told |
| found | shot | shot | According |
| found | According | According | facing |
| transported | found | found | found |
| told | slumped | slumped | found |
| had | facing | hanging | bleeding |
| | found | facing | pronounced |
| | found | found | taken |
| | transported | found | told |
| | taken | bleeding | tried |
| | told | transported | steal |
| | had | pronounced | |
| | concealed | taken | |
| | | told | |
| | | tried | |
| | | steal | |
| | | concealed | |

Table A.7 Comparison of verb extraction methods applied to STORY D.

| Entities | | | | |
|-----------------------|---------------|------------------|-----------|----------------------------|
| MODELLER1 | MODELLER2 | MODELLER3 | MODELLER4 | Automatic |
| 0.45 Gun | 9mmHandgun | incident | Williams | Davis |
| Witness | 45mmHandgun | Witnesses | Davis | William |
| police | Police | Vincent_Williams | Police | EZ Mart convenience store |
| Davis | EZ_Mart_store | Lashawn_Davis | Witnesses | several witnesses |
| 1999 MGM | William | | | police |
| Williams | Davis | | | gold Mercury Grand Marquis |
| transport_to_hospital | William's_car | | | A .45 handgun |
| Police | | | | his feet |
| Gun | | | | 9mm handgun |
| | | | | local hospital |

Table A.8 Extracted Entities from STORY D.

A.1.5 STORY E models and analysis

“Yvonne Hiller, 43, allegedly shot and killed co-workers Tanya Renee Wilson, 47, and LaTonya Sharon Brown, 36, following an argument at the Philadelphia Kraft Foods plant where they worked. Hiller, a 15-year Kraft employee, had fought with Brown and Wilson for at least two years, accusing the two of throwing chemicals at her and talking behind her back. Following her latest accusations, Hiller’s supervisor suspended her and told her to leave the plant.

Hiller then went to her car, retrieved a .357 Magnum handgun, and returned to the plant, pointing the handgun at two security guards as she demanded re- entry. Once back inside the plant, she found Brown, Wilson, and two other co-workers in a third-floor break room. She allowed one woman to leave, stating that she had no quarrel with her. She then allegedly opened fire, shooting Brown once in the head at close range and Wilson in the side. A third employee, Bryant Dalton, 39, was shot in the neck and wounded. Hiller then reportedly hunted down the supervisor who had suspended her in the hallway and fired at him, but missed. Hiller also fired at a mechanic who followed her throughout the building, shouting at co-workers to flee and reporting her movements to 911 via a cell phone. Hiller then allegedly fired at officers as they arrived on the scene. Forty minutes after the shooting began, SWAT officers stormed the building, apprehended Hiller, and freed her co-workers. Hiller was charged with two counts of murder, one count of attempted murder, aggravated assault, and other charges.” [15]

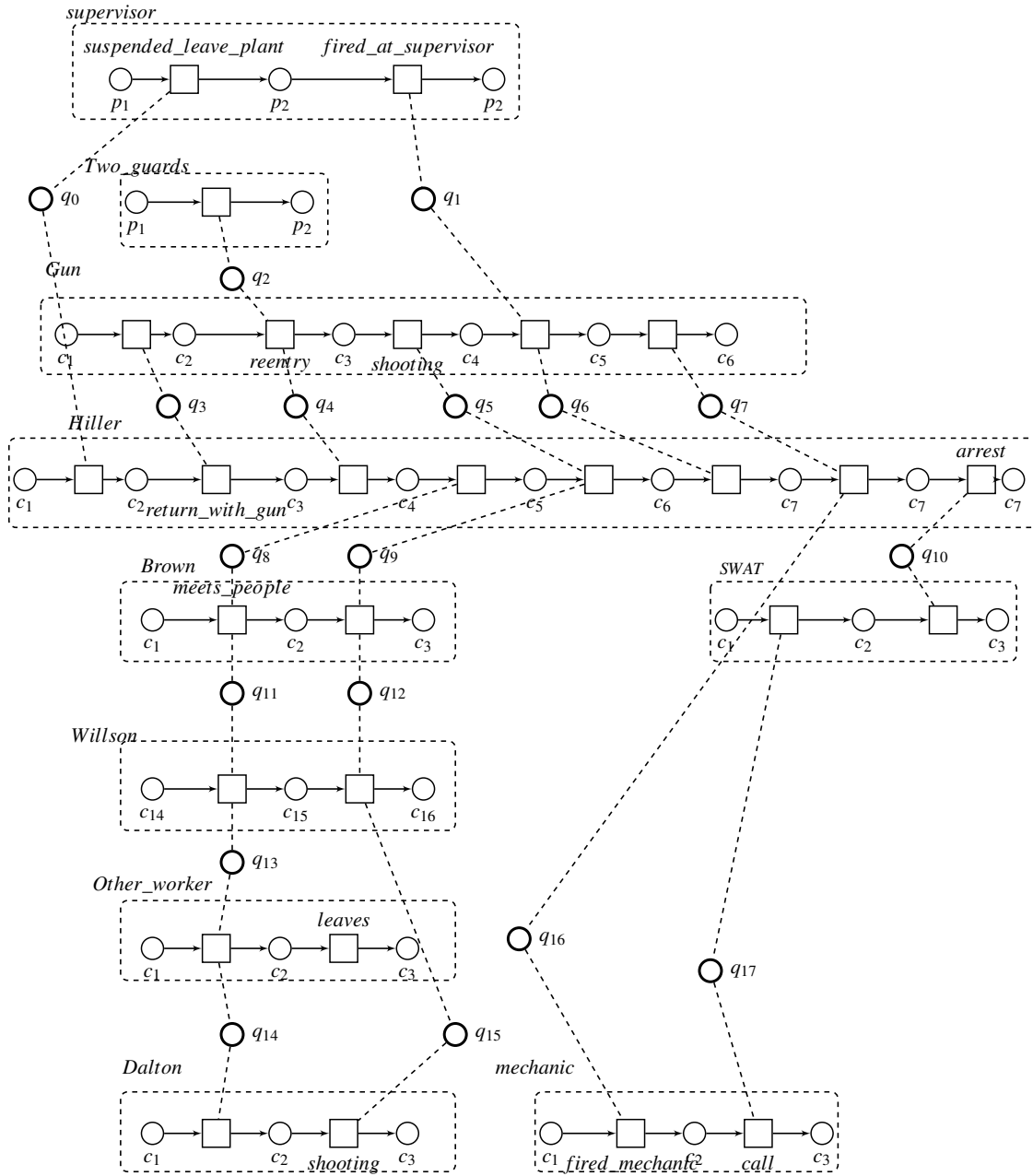


Fig. A.10 A model created by SON expert user depicts the events from STORY E.

| Method | | | |
|-----------|-----------|-------------|-----------|
| 1 | 2 | 3 | 4 |
| shot | shot | shot | shot |
| fought | killed | killed | killed |
| suspended | following | following | following |
| went | fought | worked | worked |
| found | Following | fought | fought |
| allowed | suspended | accusing | throwing |
| opened | told | throwing | talking |
| hunted | leave | talking | Following |
| fired | went | Following | suspended |
| stormed | returned | suspended | told |
| shot | found | told | leave |
| fought | allowed | leave | went |
| suspended | leave | worked | retrieved |
| went | stating | went | returned |
| hunted | opened | retrieved | pointing |
| fired | shooting | returned | demanded |
| stormed | shot | worked | found |
| charged | hunted | pointing | allowed |
| shot | fired | demanded | opened |
| fought | fired | worked | shooting |
| found | followed | found | shot |
| opened | fired | allowed | hunted |
| | arrived | leave | suspended |
| | stormed | stating | fired |
| | charged | opened | fired |
| | attempted | shooting | followed |
| | shot | shooting | shouting |
| | wounded | flee | |
| | hunted | reporting | |
| | suspended | fired | |
| | fired | arrived | |
| | missed | stormed | |
| | fired | apprehended | |
| | followed | freed | |
| | shouting | shouting | |
| | flee | fleeing | |
| | reporting | reporting | |
| | fired | fired | |
| | | arrived | |
| | | began | |
| | | stormed | |
| | | apprehended | |
| | | freed | |
| | | charged | |
| | | attempted | |

Table A.9 Comparison of verb extraction methods applied to STORY E.

| Entities | | | | |
|------------------|------------------|------------------|------------------|---------------------|
| MODELLER1 | MODELLER2 | MODELLER3 | MODELLER4 | Automatic |
| Supervisor | Supervisor | incident | Hiller | Hiller |
| Two_Guards | Hiller | Yvonne_Hiller | Handgun | Brown |
| Gun | Kraft_Food | LaTonya_Brown | Wilson | Wilson |
| Hiller | Mechanic | Tanya_Wilson | Dalton | Kraft |
| Brown | Security_guards | Mechanic | Brown | .357 Magnum handgun |
| Wilson | Dalton | Supervisor | officer | one woman |
| Other_workers | Woman | SWAT | | Third employee |
| Dalton | Wilson | | | Bryant Dalton |
| Mechanic | Brown | | | a mechanic |
| SWAT | co-workers | | | to 911 |
| | Officer | | | officers |
| | | | | SWAT officers |

Table A.10 Extracted Entities from STORY E.

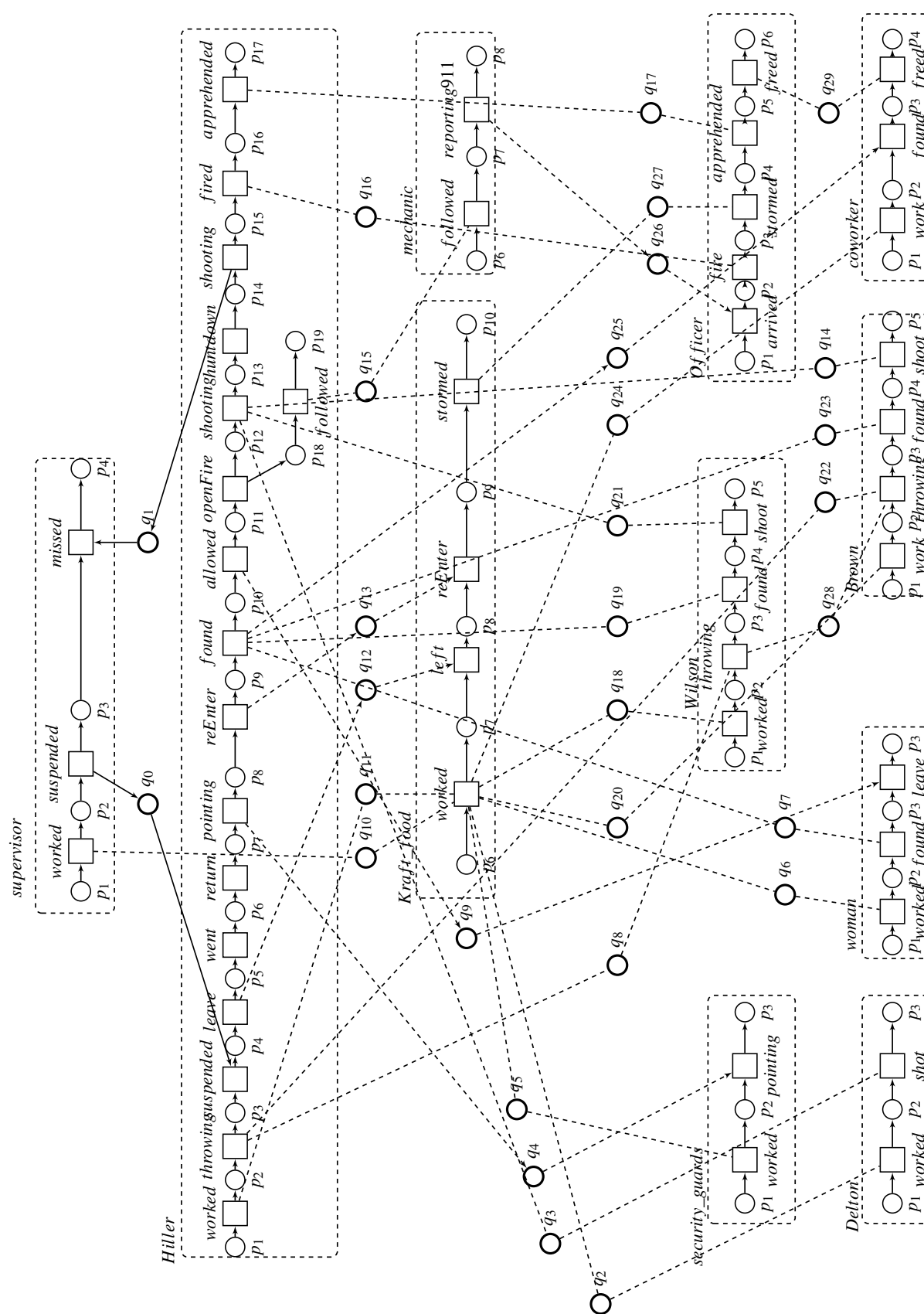


Fig. A.11 A model created by SON expert user depicts the events from STORY E.

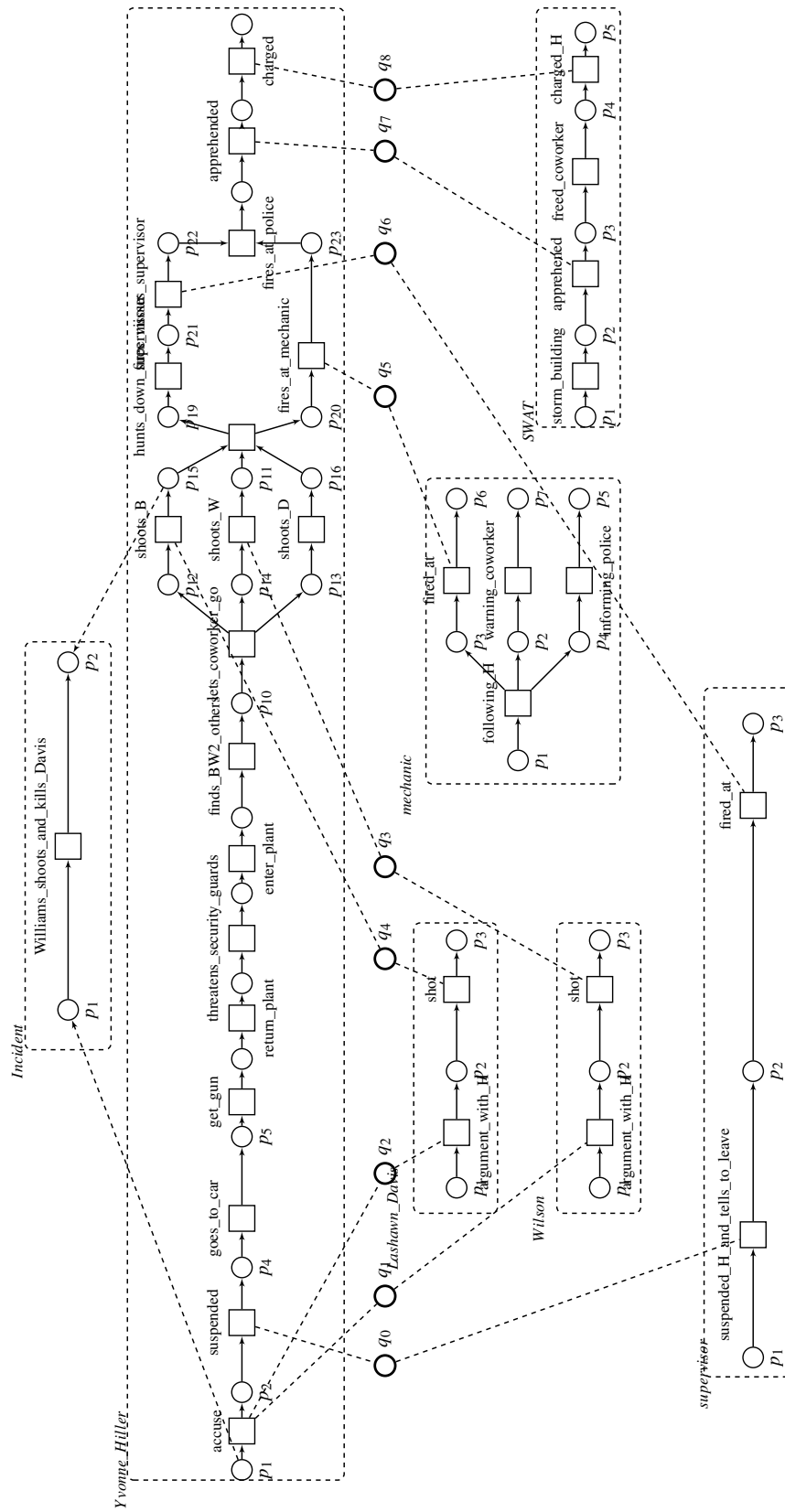


Fig. A.12 A model created by SON expert user depicts the events from STORY E.

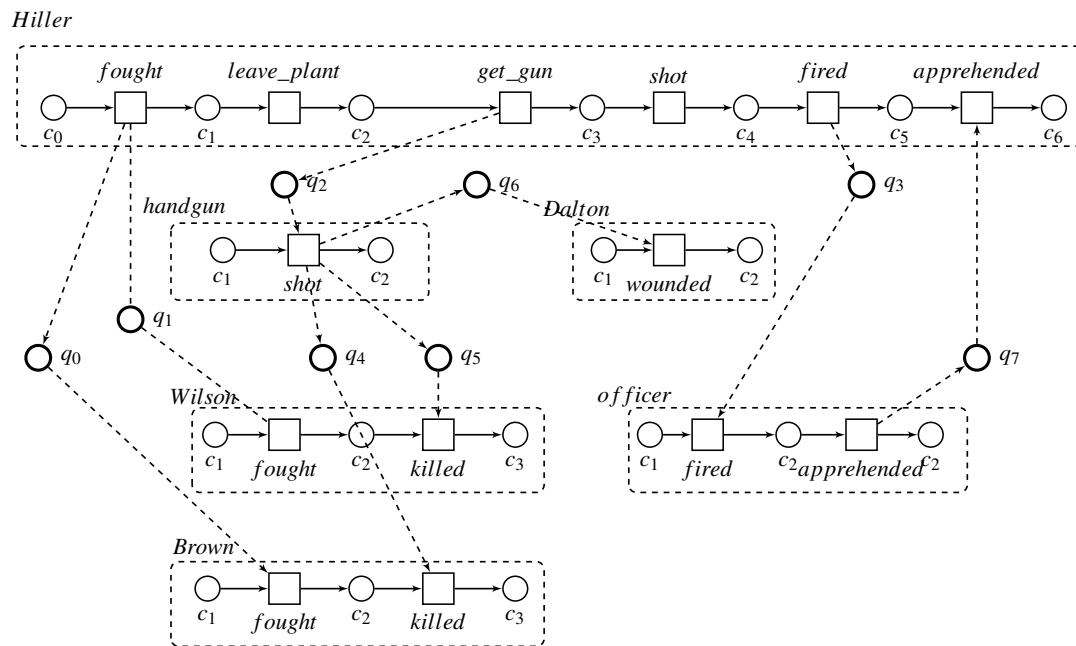


Fig. A.13 A model created by SON expert user depicts the events from STORY E.

References

- [1] AI, E. (2024). Explosion: NLP Technology and Open-Source AI Software. <https://explosion.ai/>. Accessed: 2022.
- [2] Al-Janabi, K. B. S. (2011). A proposed framework for analyzing crime data set using decision tree and simple k-means mining algorithms. *Journal of Kufa for Mathematics and Computer*, 1(3):8–24.
- [3] Alahmadi, M. (2023). Detecting SYN Flood Attack using CSA-Nets. In *CS & IT Conference Proceedings*, volume 13. CS & IT Conference Proceedings.
- [4] Alahmadi, M., Alharbi, S., Alharbi, T., Almutairi, N., Alshammari, T., Bhattacharyya, A., Koutny, M., Li, B., and Randell, B. (2024). Structured Acyclic Nets. *CoRR*, abs/2401.07308.
- [5] Alharbi, S. (2023). Hierarchical Simulation of Timed Behaviours of Structured Occurrence Nets. In Köhler-Bussmeier, M., Moldt, D., and Rölke, H., editors, *Proceedings of the 2023 International Workshop on Petri Nets and Software Engineering (PNSE 2023) co-located with the 44th International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2023), June 27, 2023, Lisbon, Portugal*, volume 3430 of *CEUR Workshop Proceedings*, pages 143–166. CEUR-WS.org.
- [6] Alharbi, T. and Koutny, M. (2019). Domain Name System (DNS) Tunneling Detection using Structured Occurrence Nets (SONs). In *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE 2019)*, CEUR Workshop Proceedings, pages 93–108. CEUR-WS.org.
- [7] Almutairi, N. (2022). Probabilistic communication structured acyclic nets. In Köhler-Bussmeier, M., Moldt, D., and Rölke, H., editors, *Petri Nets and Software Engineering 2022 co-located with the 43rd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2022), Bergen, Norway, June 20th, 2022*, volume 3170 of *CEUR Workshop Proceedings*, pages 168–187. CEUR-WS.org.
- [8] Almutairi, N. and Koutny, M. (2021). Verification of communication structured acyclic nets using sat. *PNSE@ Petri Nets*, 2907:175–194.
- [9] Altinok, D. (2021). *Mastering spaCy: An end-to-end practical guide to implementing NLP applications using the Python ecosystem*. Packt Publishing Ltd.
- [10] Arulanandam, R., Savarimuthu, B. T. R., and Purvis, M. A. (2014). Extracting crime information from online newspaper articles. In *Proceedings of the second australasian web conference-volume 155*, pages 31–38.

- [11] Bird, S. (2006). NLTK: the natural language toolkit. In Calzolari, N., Cardie, C., and Isabelle, P., editors, *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics.
- [12] Brauer, W. and Reisig, W. (2009). Carl adam petri and “petri nets”. *Fundamental concepts in computer science*, 3(5):129–139.
- [13] Brunelle, J. F. and Boonthum-Denecke, C. (2012). Natural language processing tools. In *Cross-disciplinary advances in applied natural language processing: Issues and approaches*, pages 9–23. IGI Global.
- [14] Caruccio, L., Cimino, G., Cirillo, S., Desiato, D., Polese, G., and Tortora, G. (2023). Malicious Account Identification in Social Network Platforms. *ACM Trans. Internet Techn.*, 23(4):57:1–57:25.
- [15] Center, V. P. (2020). Violence Policy Center. [online] Available at: <https://concealedcarrykillers.org/>. Accessed: (2022).
- [16] Chakravorty, S., Daripa, S., Saha, U., Bose, S., Goswami, S., and Mitra, S. (2015). Data mining techniques for analyzing murder related structured and unstructured data. *American Journal of Advanced Computing*, 2(2):47–54.
- [17] Chen, T. M., Sánchez-Aarnoutse, J. C., and Buford, J. F. (2011). Petri net modeling of cyber-physical attacks on smart grid. *IEEE Trans. Smart Grid*, 2(4):741–749.
- [18] Chiola, G., Marsan, M. A., Balbo, G., and Conte, G. (1993). Generalized stochastic petri nets: A definition at the net level and its implications. *IEEE Trans. Software Eng.*, 19(2):89–107.
- [19] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. (2015). Fame for sale: Efficient detection of fake Twitter followers. *Decision Support Systems*, 80:56–71.
- [20] Dasgupta, D., Akhtar, Z., and Sen, S. (2022). Machine learning in cybersecurity: a comprehensive survey. *The Journal of Defense Modeling and Simulation*, 19(1):57–106.
- [21] Dasgupta, T., Naskar, A., Saha, R., and Dey, L. (2017). Crimeprofiler: Crime information extraction and visualization from news media. In *Proceedings of the international conference on web intelligence*, pages 541–549.
- [22] David, I., Siordia, O. S., and Moctezuma, D. (2016). Features combination for the detection of malicious Twitter accounts. In *2016 IEEE international autumn meeting on power, electronics and computing (ROPEC)*, pages 1–6. IEEE.
- [23] Elhadad, M. (2010). Book review: Natural language processing with python by steven bird, ewan klein, and edward loper. *Computational Linguistics*, 36(4).
- [24] Encounters, F. (2024). Fatal Encounters. [online] Available at: <https://fatalencounters.org/>. Accessed: (2024).

- [25] Freytag, T. and Allgaier, P. (2018). WoPeD goes NLP: Conversion between Workflow Nets and Natural Language. In van der Aalst et. al, W. M. P., editor, *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018*, CEUR Workshop Proceedings, pages 101–105. CEUR-WS.org.
- [26] Group, S. N. (2024). Stanford CoreNLP. Accessed: 2022.
- [27] Hand, F. (2019). An Evaluation of Structured Occurrence Nets for Crime Investigation. Master’s thesis, School of Computing Newcastle University.
- [28] Harifi, S. and Nakhjavanlo, B. B. (2016). Decision support system based on petri net for a police vehicle command and control system. In *2016 Third International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*, pages 1–6. IEEE.
- [29] Hirschberg, J. and Manning, C. D. (2015). Advances in natural language processing. *Science*, 349(6245):261–266.
- [30] Homsy, A., Al Nemri, J., Naimat, N., Kareem, H. A., Al-Fayoumi, M., and Snober, M. A. (2021). Detecting Twitter Fake Accounts using Machine Learning and Data Reduction Techniques. In *DATA*, pages 88–95.
- [31] Jasiul, B., Szpyrka, M., and Sliwa, J. (2014). Detection and modeling of cyber attacks with petri nets. *Entropy*, 16(12):6602–6623.
- [32] Jensen, K. (1981). Coloured petri nets and the invariant-method. *Theoretical computer science*, 14(3):317–336.
- [33] Jensen, K. (1998). An introduction to the practical use of coloured petri nets. *Lectures on Petri Nets II: Applications: Advances in Petri Nets 3*, pages 237–292.
- [34] Jensen, K. and Kristensen, L. M. (2009). *Coloured Petri Nets - Modelling and Validation of Concurrent Systems*. Springer.
- [35] Jha, S., Sheyner, O., and Wing, J. (2002). Two formal analyses of attack graphs. In *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, pages 49–63. IEEE.
- [36] Koutny, M. and Randell, B. (2009). Structured occurrence nets: A formalism for aiding system failure prevention and analysis techniques. *Fundamenta Informaticae*, page 41–91.
- [37] Ku, C. H., Iriberri, A., and Leroy, G. (2008). Crime information extraction from police and witness narrative reports. In *2008 IEEE Conference on Technologies for Homeland Security*, pages 193–198. IEEE.
- [38] Lallie, H. S., Debattista, K., and Bal, J. (2018). An empirical evaluation of the effectiveness of attack graphs and fault trees in cyber-attack perception. *IEEE Trans. Inf. Forensics Secur.*, 13(5):1110–1122.
- [39] Lallie, H. S., Debattista, K., and Bal, J. (2020). A review of attack graph and attack tree visual syntax in cyber security. *Comput. Sci. Rev.*, 35:100219.
- [40] Li, B. (2017). *Visualisation and Analysis of Complex Behaviours using Structured Occurrence Nets*. PhD thesis, School of Computing, Newcastle University.

- [41] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.
- [42] Mansour Salah, M. and Xia, K. (2022). Big crime data analytics and visualization. In *Proceedings of the 2022 6th International Conference on Compute and Data Analysis*, pages 24–28.
- [43] Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proc. IEEE*, 77(4):541–580.
- [44] Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W. (2011). Natural language processing: an introduction. *J. Am. Medical Informatics Assoc.*, pages 544–551.
- [45] Nasridinov, A., Ihm, S., and Park, Y. (2013). A decision tree-based classification model for crime prediction. In Park, J. J. H., Barolli, L., Xhafa, F., and Jeong, H., editors, *Information Technology Convergence - Security, Robotics, Automations and Communication, 5th International Conference on Information Technology Convergence and Services, ITCIS 2013, Fukuoka, Japan, July 8-10, 2013*, volume 253 of *Lecture Notes in Electrical Engineering*, pages 531–538. Springer.
- [46] NeuralCoref (2022). NeuralCoref 4.0: Fast Coreference Resolution in spaCy with Neural Networks. <https://github.com/huggingface/neuralcoref>. Accessed: 2022.
- [47] Noel, S., Harley, E., Tam, K. H., Limiero, M., and Share, M. (2016). Cygraph: graph-based analytics and visualization for cybersecurity. In *Handbook of Statistics*, volume 35, pages 117–167. Elsevier.
- [48] Norouzi, Y. (2022a). Spatial, temporal, and semantic crime analysis using information extraction from online news. In *2022 8th International Conference on Web Research (ICWR)*, pages 40–46. IEEE.
- [49] Norouzi, Y. (2022b). Spatial, temporal, and semantic crime analysis using information extraction from online news. In *2022 8th International Conference on Web Research (ICWR)*, pages 40–46. IEEE.
- [50] Psarologou, A. (2016). *A Stochastic Petri Net based NLU Scheme for Technical Documents Understanding*. PhD thesis, Wright State University.
- [51] Qazi, N. and Wong, B. L. W. (2019). An interactive human centered data science approach towards crime pattern analysis. *Inf. Process. Manag.*, 56(6).
- [52] Rabzelj, M., Bohak, C., Juznic, L. S., Kos, A., and Sedlar, U. (2023). Cyberattack graph modeling for visual analytics. *IEEE Access*, 11:86910–86944.
- [53] Rahma, F. and Romadhony, A. (2021). Rule-based crime information extraction on indonesian digital news. In *2021 International Conference on Data Science and Its Applications (ICoDSA)*, pages 10–15. IEEE.
- [54] Raja, M. S. and Raj, L. A. (2022). Detection of Malicious Profiles and Protecting Users in Online Social Networks. *Wirel. Pers. Commun.*, 127(1):107–124.

- [55] Randell, B. (2018). Soncraft challenges. Unpublished manuscript.
- [56] Randell, B. and Koutny, M. (2009). Structured occurrence nets: Incomplete, contradictory and uncertain failure evidence. *School of Computing Science Technical Report Series*.
- [57] Samper-Escalante, L. D., Loyola-González, O., Monroy, R., and Medina-Pérez, M. A. (2021). Bot Datasets on Twitter: Analysis and Challenges. *Applied Sciences*, 11(9):4105.
- [58] Sharma, M. (2014). Z-crime: A data mining tool for the detection of suspicious criminal activities based on decision tree. In *2014 International Conference on Data Mining and Intelligent Computing (ICDMIC)*, pages 1–6. IEEE.
- [59] Siriaraya, P., Zhang, Y., Wang, Y., Kawai, Y., Mittal, M., Jeszenszky, P., and Jatowt, A. (2019). Witnessing Crime through Tweets: A Crime Investigation Tool based on Social Media. In *Proceedings of the 27th ACM International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, pages 568–571. ACM.
- [60] spaCy (2021). <https://spacy.io>. Accessed: 2021.
- [61] spaCy Pipeline (2022). <https://spacy.io/usage/spacy-101>. Accessed: 2022.
- [62] Stan, O., Bitton, R., Ezrets, M., Dadon, M., Inokuchi, M., Ohta, Y., Yagyu, T., Elovici, Y., and Shabtai, A. (2022). Extending attack graphs to represent cyber-attacks in communication protocols and modern IT networks. *IEEE Trans. Dependable Secur. Comput.*, 19(3):1936–1954.
- [63] Sukthanker, R., Poria, S., Cambria, E., and Thirunavukarasu, R. (2020). Anaphora and coreference resolution: A review. *Information Fusion*, 59:139–162.
- [64] TagEditor, I. (2022). TagEditor Annotation Tool. <https://github.com/d5555/TagEditor>. Accessed: 2022.
- [65] Torky, M., Baberse, R., Ibrahim, R., Hassanien, A. E., Schaefer, G., Korovin, I., and Zhu, S. Y. (2016). Credibility investigation of newsworthy tweets using a visualising Petri net model. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 003894–003898. IEEE.
- [66] Torky, M., Meligy, A., Ibrahim, H., and Hassanein, A. E. (2018). Colored Petri Net Model for Blocking Misleading Information Propagation in Online Social Networks. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017*, pages 600–609. Springer.
- [67] Umair, A., Sarfraz, M. S., Ahmad, M., Habib, U., Ullah, M. H., and Mazzara, M. (2020). Spatiotemporal analysis of web news archives for crime prediction. *Applied Sciences*, 10(22):8220.
- [68] Vasiliev, Y. (2020). *Natural Language Processing with Python and spaCy: A Practical Introduction*. No Starch Press.

-
- [69] Wang, Y., Peng, X., and Bian, J. (2014). Computer crime forensics based on improved decision tree algorithm. *J. Networks*, 9(4):1005–1011.
- [70] Yao, S., Wei, M., Yan, L., Wang, C., Dong, X., Liu, F., and Xiong, Y. (2020). Prediction of crime hotspots based on spatial factors of random forest. In *2020 15th International Conference on Computer Science & Education (ICCSE)*, pages 811–815. IEEE.
- [71] Zheng, J., Chapman, W. W., Crowley, R. S., and Savova, G. K. (2011). Coreference resolution: A review of general methodologies and applications in the clinical domain. *Journal of biomedical informatics*, 44(6):1113–1122.