



SAPIENZA
UNIVERSITÀ DI ROMA

On the Mining of Artful Processes

Dipartimento di Ingegneria Informatica, Automatica e Gestionale
ANTONIO RUBERTI

Dottorato di Ricerca in Ingegneria Informatica – XXV Ciclo

Candidate

Claudio Di Ciccio
ID number 795152

Thesis Committee

Prof. Tiziana Catarci
Dr. Massimo Mecella
Prof. Stefano Leonardi
Prof. Marlon Dumas (thesis reviewer)
Dr. Marco Montali (thesis reviewer)

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy in Ingegneria Informatica
2013, September the 4th

Thesis defended on 2013, October the 7th
in front of a Board of Examiners composed by:

Prof. Andrea Clementi (chairman)

Prof. Paolo Ferragina

Prof. Roberto Basili

On the Mining of Artful Processes

Ph.D. thesis. Sapienza – University of Rome

© 2013 Claudio Di Ciccio. All rights reserved

This thesis has been typeset by L^AT_EX and the Sapthesis class.

Version: 2013, October the 7th

Website: <http://www.dis.uniroma1.it/cdc>

Author's email: cdc@dis.uniroma1.it

Extended abstract

Motivation and rationale

Process (“prō-, -ses”, noun: *a series of actions or operations conducing to an end*¹) is the term by which we denote the sequence of activities performed by human or non-living beings to yield a result. Business process management is based on the observation that each product that a company provides to the market is the outcome of a number of activities performed. Nowadays, business processes are the key instrument to organize these activities and improve the understanding of their interrelationships. For a long time, formal business processes (characterizing, e.g., public administrations, insurance/financial institutions, etc.) have been the main subject of workflow related research.

Informal processes, a.k.a. “artful processes”, are carried out by those people whose work is mental rather than physical (managers, professors, researchers, etc.), the so called “knowledge workers”. With their skills, experience and knowledge, they are used to performing difficult tasks, which require complex, rapid decisions among multiple possible strategies, in order to fulfill specific goals. In contrast to business processes, which are formal and standardized, often informal processes are not even written down, let alone defined formally, and can vary from person to person even when those involved are pursuing the same objective. Knowledge workers create informal processes “on the run” to cope with many of the situations that arise in their daily work. Though informal processes are frequently repeated, they are not exactly reproducible even by their originators – since they are not written down – and can not be easily shared either. Artful processes have goals and methods that change quickly over time, making them difficult to codify in the context of an enterprise application. We denote these kinds of processes “artful” in the sense that there is an art to their execution. In some processes, it is primarily the content in each process instance – rather than the process itself – that determines the outcome. Furthermore, many highly specialized processes are developed or refined locally at the individual or small-team level such that the process cannot easily be separated from the specific people who perform it. Thus, while the framing process may be stable at an abstract level, the key details are not. They depend on the skills, experience, and judgment of the primary actors. As an example, we can consider the coordination of an international research project. Some deadlines are fixed, such as review meetings or annual budgeting reports, but the rest of the steps made to meet the project’s requirements vary from case to case. The publication of deliverables, the set-up of possible demos, the outcome of task-forces and work packages depend

¹Merriam-Webster dictionary

on the objective of the projects, the partners involved, contingencies, and so forth.

Knowledge workers are often used to convey their creativity and intuition into their job, so to come up with innovative results. Inventing is the “fun” aspect of their work, in most of the cases. But the “amusing” phases are intertwined with routine tasks, which are typical of the everyday job. Though not necessarily boring, they cannot be avoided. For instance, managing a Work Package entails both the activities of (i) defining the architecture of a new software system for a research project, and (ii) reminding a partner of the same project that the deadline for the submission of their contribution to the next deliverable is expiring. The routine phases could be allocated, at least partly, to automated systems, since they often require no creativity. Such a system could assist knowledge workers to complete them.

Knowledge workers cannot be realistically expected to instruct the assistive system by modeling their artful processes: it would be time-consuming both in the initial definition and in the potential continuous revisions. To make things worse, time is the crucial resource that usually knowledge workers indeed lack.

The repetitive patterns could be learnt by machines, tracking the operations of a human. Smart solutions to known artful processes frequently change instead, thus computers are by far less likely to infer the inventions. On the other hand, knowledge workers would not find any interference in their creative work useful. Therefore, a system which is able to learn the routine part of ever-changing processes, and assist the users accordingly, may leave more time to the knowledge workers for the creative aspects of their work.

Outcomes and ideas are often shared by means of email conversations, which are a fast, reliable, permanent way of keeping a track of the activities that knowledge workers fulfill. Despite the advent of structured case management tools, many enterprise processes are still “run” over emails. Thus, reverse engineering workflows of such processes and their integration with artefacts and other structured processes can accurately depict the enterprise’s process landscape. A system able to infer the models of the processes laying behind the email messages exchanged would be valuable and the result could materialize almost freely: the set-up time would be heavily reduced indeed.

This is the purpose of our approach, which is the core of this thesis and is named MAILOFMINE. Its investigation mainly resides in the Machine Learning area. More specifically, it relates to *Information Retrieval* (IR) and *Process Mining* (PM). We adopted well-known IR techniques in order to extract the activities out of the email messages. We proposed new algorithms for PM in order to discover the temporal rules that such activities adhere to. The set of such rules, intended as temporal constraints, constitute the so called *declarative modeling* of workflows. *Declarative* models differ from the *imperative* in that they do not explicitly represent every possible execution that a process can be enacted through. I.e., there is no graph-like structure determining the whole evolution of a process instance, from the beginning to the end. It just establishes a set of constraints that must hold true, whatever the evolution of the process instance will be. What is not explicitly declared to be respected, is allowed. The reader can easily see that it is better suited to processes subject to frequent changes, with respect to the classical approach.

Analyzing the problem from a more abstract perspective, this work challenges the problem of discovering highly flexible workflows (such as artful processes), out

of semi-structured information (such as email messages).

Research contribution

During the development of our approach, we contributed to the research fields of Knowledge Discovery and Process Mining. Here we briefly summarize our research contribution in these areas.

- R1** In order to define semantics of the state-of-the-art taxonomy of constraints for declarative workflow models, namely Declare, LTL (Linear Temporal Logic) over finite strings (LTL_f) was the standard. We made use of regular expressions, instead. Such a choice allowed us to exploit the rich scientific machinery of tools and algorithms for modeling, altering and updating finite state automata in order to model, alter and update declarative process models.
- R2** We developed a new algorithm, named MINERful, able to discover a declarative process model out of logs. Its main innovation resides in the two-step nature of the execution, where the first phase builds a knowledge base containing statistical information on every single task read, and the second discovers the validity of Declare constraints over the log by querying that knowledge base. Two main versions of MINERful have been developed.
- (a) The first release was able to infer whether constraints of the declarative workflow held or not, and returned a process model accordingly.
 - (b) The second release improved the preceding one, by associating a numerical Support to each constraint, rather than asserting whether it was verified or not in a boolean fashion. Namely, it computed a statistical-based value, according to which constraints were more or less likely to be verified in the discovered process. Moreover, it associates every constraint to metrics estimating a level of relevance, on the basis of the number of appearances of the involved activities in the log.

We proved MINERful to be polynomial in the size of the input, and faster than the current state-of-the-art algorithms. We checked the performances of MINERful over both synthetic logs (Research Contribution **R3**), synthetic error-injected logs (**R4**), and case study benchmarks.

- R3** We realized an automated tool for the generation of synthetic logs, defined on the basis of either Declare constraints or any other regular expression (see Research Contribution **R1**). Usually, the tool adopted for this kind of task was CPNTools, which is a software suite for the management of Colored Petri Nets – hence, more appropriate for imperative rather than declarative models.
- R4** Also thanks to the results obtained in Research Contributions **R1** and **R3**, we could create a controlled error injector for logs, i.e., a software module able to insert user-tunable noise into existing logs. This tool has been utilized in order to study how mined processes change when logs are affected by errors.
- R5** We applied document indexing techniques in order to find out the possible witnesses of the execution of activities inside email messages.

- R6** We assessed the quality of our approach, in terms of compliance of the overall discovered process to the expected behavior. To this extent, we conducted an extensive analysis together with an expert, over a real case study concerning the management of several European Research Projects.
- R7** We proposed a novel visualization framework for declarative workflows. Rather than visualizing only the whole process model at once, i.e., in a single graph, we also considered an alternative view, focusing on single activities and the constraints specifically related to them. To this aim, we used a paradigm for the representation of constraints essentially based on the concept of Cartesian diagrams, putting causality on the abscissae and time on the ordinates. We called it “local view”, as opposed to the “global view” representing the whole interconnection of constraints over activities. We adopted the global view as an bird’s eye perspective on the process. We made use of the local view to show the executable alternatives, once a given activity was assumed to be done.
- R8** We designed a modular architecture for extracting activities out of email archives (mailboxes) and then discover the models of processes of which those email messages reported the execution.

Parts of our work were published in the following papers. Each is labeled with the research challenges addressed.

- [35] Di Ciccio, C., Mecella, M. “Studies on the Discovery of Declarative Control Flows from Error-prone Data”. Full paper at 2013 International Symposium on Data-Driven Process Discovery and Analysis, SIMPDA 2013, Riva del Garda, Italy, August 30, 2013.
Research Contributions **R1**, **R3**, **R2b**.
- [36] Di Ciccio, C., Mecella, M. “A Two-Step Fast Algorithm for the Automated Discovery of Declarative Workflows”. Full paper at 2013 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 15-19 April 2013.
Research Contributions **R1**, **R3**, **R2b**.
- [32] Di Ciccio, C., Marrella, A., Russo, A., “Knowledge-intensive Processes: An Overview of Contemporary Approaches”. Proceedings of the 1st International Workshop on Knowledge-intensive Business Processes, KiBP 2012, Rome, Italy, June 15, 2012.
R2a, **R7**.
- [34] Di Ciccio, C., Mecella, M., “Mining Constraints for Artful Processes”. in Proceedings of the 15th International Conference on Business Information Systems, BIS 2012, Vilnius, Lithuania, May 21-23, 2012.
Research Contributions **R1**, **R3**, **R2a**.
- [33] Di Ciccio, C., Mecella, M., “MINERful, a Mining Algorithm for Declarative Process Constraints in MailOfMine”. Technical Report of Dipartimento di Ingegneria Informatica, Automatica e Gestionale “Antonio Ruberti” - SAPIENZA, Università di Roma. March 2012.
Research Contributions **R1**, **R3**, **R2a**.

-
- [41] Di Ciccio, C., Mecella, M., Scannapieco, M., Zardetto, D., Catarci, T., “MailOfMine – Analyzing Mail Messages for Mining Artful Collaborative Processes”. *Data-Driven Process Discovery and Analysis*, Springer, 55-81 (extended paper for SIMPDA 2011 post-conference proceedings [40]).
Research Contributions **R1**, **R8**, **R6**.
- [31] Di Ciccio, C., Mecella, M., Catarci, T., “Representing and Visualizing Mined Artful Processes in MailOfMine”. *HCI-KDD workshop, Information Quality in e-Health - 7th Conference of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2011*, Graz, Austria, November 25-26, 2011.
Research Contributions **R1**, **R7**.
- [38] Di Ciccio, C., Mecella, M., Scannapieco, M., Zardetto, D., “Groupware Mail Messages Analysis for Mining Collaborative Processes”. *Proceedings of the 19th Italian Symposium on Advanced Database Systems, SEBD 2011*, Maratea, Italy, June 26-29, 2011 (poster paper).
Research Contribution **R8**.
- [39] Di Ciccio, C., Mecella, M., Scannapieco, M., Zardetto, D., Catarci, T., “Groupware Mail Messages Analysis for Mining Collaborative Processes”. Technical report of Dipartimento di Ingegneria Informatica, Automatica e Gestionale “Antonio Ruberti” - SAPIENZA, Università di Roma. January 2011.
Research Contribution **R8**.

Thesis Outline

Chapter 1 introduces this work, with more insights on the problem and the motivation for our research.

Chapter 2 analyzes the current state of the art in the areas of (i) Information Retrieval, with a deeper insight on email-related tools, techniques and approaches, (ii) Process Modeling, especially focusing on declarative approaches such as Declare, and (iii) Process Mining.

Chapter 3 depicts the modular architecture of MAILOFMINE, seen as a system. There, the discovery of process models out of logs extracted by email archives is subdivided in functional elements, abstracted as high-level software components. A brief explanation of the techniques adopted in each of them is thus provided, in order to link them to a given stage of the computation. There, the software tools that we implemented or integrated as a realization are specified. We used as third-party components *MySQL* for storing data, *JBoss Application Server* for containing the components deployed as Java Enterprise Beans, *Apache Solr* for the information retrieval task of determining which email messages could be the evidence of an executed activity, `dk.bricks.automaton` library for the generation and management of finite state automata.

Chapter 4 details the new Process Mining algorithm that we introduced, for the discovery of declarative workflows. We present it into two versions: the first, determining whether any constraint among the possible Declare templates

hold or not. The second, giving each constraint a degree of Support, in terms of likelihood for a constraint to hold true, together with Confidence Level and Interest Factor for estimating the relevance of the mined constraints. Both compute their result on the basis of the same statistical information gathered from the log. We prove that, even though the second is more sophisticated, the complexity remains the same.

Chapter 5 shows the results of performance tests over synthetic data, the observation of the results gathered when dealing with artificially error-injected logs, and the evaluation of our system applied to a real case study, based on the mailboxes of people involved in this research work. Finally,

Chapter 6 concludes the discussion and traces some future developments that might arise from the basis of this work.

Additional work

The author of this thesis has also been involved in another research project during the earlier stages of the Ph.D. programme. He has been a member of the SM4ALL² FP7 European Research Project (2009-2011) team, for SAPIENZA, University of Rome. The aim of SM4ALL was the development of a comprehensive system, able to assist differently abled people in their everyday life at home. As a distributed pervasive computing architecture, it was thought to orchestrate the domotic devices spread around the house, so to let users achieve their goals independently. I.e., it had to guarantee the least degree of intervention possible for nurses. In other words, it aimed at being an innovative way to overcome the barriers that disabilities might oppose to the autonomy of users. The contribution of SAPIENZA was the development of an automated composition engine, applied to domotic sensors and actuators abstracted as services. We report it here in terms of published papers for sake of completeness, though the themes covered in their subject is out of scope for this thesis.

- [26] De Giacomo, G., Di Ciccio, C., Felli, P., Hu, Y., Mecella, M., “Goal-based Composition of Stateful Services for Smart Homes”. Proceedings of the 20th International Conference on Cooperative Information Systems, CoopIS 2012, On the Move to Meaningful Internet Systems (OTM 2012) Confederated International Conferences , Rome, Italy, September 10-14, 2012.
- [14] Caruso, M., Di Ciccio, C., Iacomussi, E., Kaldeli, E., Lazovik, A., Mecella, M., “Service Ecologies for Home/Building Automation”. Proceedings of the 10th International IFAC Symposium on Robot Control, SYROCO 2012, Dubrovnik, Croatia, September 05-07, 2012.
- [37] Di Ciccio, C., Mecella, M., Caruso, M., Forte, V., Iacomussi, E., Rasch, K., Santucci, G., Tino, G., “The Homes of Tomorrow: Service Composition and Advanced User Interfaces”. ICST Transactions on Ambient Systems, 11 (10-12), e2, 2011.

²<http://www.sm4all-project.eu/>

- [15] Catarci, T., Di Ciccio, C., Forte, V., Iacomussi, E., Mecella, M., Santucci, G., Tino, G., “Service Composition and Advanced User Interfaces in the Home of Tomorrow: the SM4All Approach”. Proceedings of the 2nd International ICST Conference on Ambient Media and Systems, AMBI-SYS 2011, Porto, Portugal, March 24-25, 2011.
- [27] De Masellis, R., Di Ciccio, C., Mecella, M., Patrizi, F., “Smart Home Planning Programs”. Proceedings of the 7th International Conference on Service Systems and Service Management, ICSSSM 2010, Tokyo, Japan, June 28-30, 2010.
- [7] Baldoni, R., Di Ciccio, C., Mecella, M., Patrizi, F., Querzoni, L., Santucci, G., et al., “An Embedded Middleware Platform for Pervasive and Immersive Environments for-All”. Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2009, Rome, Italy, June 22-26, 2009.

Contents

Extended abstract	iii
1 Introduction and rationale	1
2 Background and State of the Art	5
2.1 Information Retrieval and Text Mining	5
2.1.1 Analysis of email messages	6
2.2 Process Modeling	7
2.2.1 Automaton-based models	7
2.2.2 Petri-Net-based models	9
2.2.3 Declarative models	11
2.2.4 Constraint Templates in Declare	15
2.3 Process Mining	23
2.3.1 Analysis of email messages	26
3 Architecture and design	27
3.1 Architecture of MAILOFMINE as a software system	27
3.2 Database	31
3.3 Specification of declarative workflows as constraints	34
3.3.1 An example	36
3.4 Process visualization	40
3.4.1 Process schema	40
3.4.2 Running instances	45
4 The Workflow Discovery Algorithm	49
4.1 MINERful	49
4.1.1 MINERfulKB	50
4.1.2 The algorithm: a bird’s eye view	53
4.1.3 Construction of the MINERfulKB	53
4.1.4 Discovery of constraints	60
4.1.5 Discovery of constraints and their metrics	65
4.1.6 The complexity of the MINERful algorithm	77
5 Experiments and evaluation	79
5.1 Experiments	79
5.1.1 Experiments over artificial error-injected logs	86
5.2 Evaluation on a real case study	88

6	Conclusions	105
6.1	Further development	105
6.1.1	Distance computing in Relation Constraints	105
6.1.2	Refinement of constraints filtering	106
6.1.3	Uncertain logs	106
6.1.4	Branching Declare	107
6.1.5	Biochemistry and forensics	107
A	From indicia to log	109
A.1	The SQL query	109
A.2	The XML result of the query for creating the log	111
A.3	The XSLT stylesheet to transform the XML log into the XES format	146
A.4	The XES log	149
B	The discovered process	169
B.1	The local Finite State Automata, generated on the basis of the discovered process' constraints	169
B.2	The discovered process' Finite State Automaton	178
B.3	The discovered process, as in the output of the run of MAILOFMINE	179

Chapter 1

Introduction and rationale

For a long time, formal business processes (e.g., the ones of public administrations, of insurance/financial institutions, etc.) have been the main subject of workflow related research.

Process management systems (PMSs) hold the promise of facilitating the everyday operation of many enterprises and work environments. However, PMSs remain especially useful in a limited range of applications where business processes can be described with relative ease. Current modeling techniques are used to codify processes that are completely predictable: all possible paths along the process are well-understood, and the process participants never need to make a decision about what to do next, since the workflow is completely determined by their data entry or other attributes of the process. This kind of highly-structured work includes mainly production and administrative processes. However, most business functions involve collaborative features and unstructured processes that do not have the same level of predictability as the routine structured work [93].

In [52] processes have been classified on the basis of their “degree of structure”. Traditional PMSs perform well with fully *structured processes* and controlled interactions between participants. A major assumption is that such processes, after having been modeled, can be repeatedly instantiated and executed in a predictable and controlled manner. However, even for structured processes, the combination and sequence of tasks may vary from instance to instance due to changes in the execution context such as user preferences, or modifications in the environment such as exceptions and changes in the business rules. In such cases (*structured processes with ad hoc exceptions*), processes should be adapted accordingly (e.g. by adding, removing or generating an alternative sequence of activities). In general, structured processes can be described by an explicit and accurate model. But in scenarios where processes are to a large extent unclear and/or unstructured, process modeling cannot be completed prior to execution (due to lack of domain knowledge a priori or to the complexity of task combinations). Hence the classical axiom “first model, then execute” – valid for the enactment of structured processes – fails. As processes are executed and knowledge is acquired via experience, it is needed to go back to the process definitions and correct them according to work practices. This is the case of *unstructured processes with predefined fragments*, where processes cannot be anticipated, and thus cannot be studied or modeled as a whole. Instead, what can be done is to identify and study a set of individual activities, and then try to understand the ways in which these activities can precede or follow each other. At

the end of the classification lies the category of *unstructured processes*, where it is impossible to define a priori the exact steps to be taken in order to complete an assignment. Since there is no pre-defined view of the process, process steps are discovered as the process scenario unfolds, and might involve decisions not based on some “codified policy”, but on the user expertise applied on the scenario at hand.

The class of *knowledge-intensive processes* is transversal with respect to the classification proposed in [52]. In the literature, different definitions have been proposed about what does “knowledge-intensive” mean for a business process. In [46] a process is defined as knowledge intensive if its value can only be created through the fulfillment of the knowledge requirements of the process participants, while Davenport recognizes the knowledge intensity by the diversity and uncertainty of process input and output [24]. In our view, a knowledge-intensive process is characterized by activities that can not be planned easily, may change on the fly and are driven by the contextual scenario that the process is embedded in. The scenario dictates who should be involved and who is the right person to execute a particular step, and the set of users involved may be not formally defined and be discovered as the process scenario unfolds. Collaborative interactions among the users typically is a major part of such processes, and new process steps might have to be defined at run time on the basis of contextual changes. Despite the popularity of commercial PMSs, there is still a lack of maturity in managing such processes, i.e., a lack of a semantic associated to the models or an easy way to reason about that semantic.

Informal processes, a.k.a. “artful processes” [50], are carried out by those people whose work is mental rather than physical (managers, professors, researchers, engineers, etc.), the so called “knowledge workers” [102]. With their skills, experience and knowledge, they are used to perform difficult tasks, which require complex, rapid decisions among multiple possible strategies, in order to fulfill specific goals. In contrast to business processes that are formal and standardized, often informal processes are not even written down, let alone defined formally, and can vary from person to person even when those involved are pursuing the same objective. Knowledge workers create informal processes “on the fly” to cope with many of the situations that arise in their daily work. Though informal processes are frequently repeated, they are not exactly reproducible even by their originators – since they are not written down – and can not be easily shared either. Their outcomes and information are exchanged very often by means of email conversations, which are a fast, reliable, permanent way of keeping track of the activities that they fulfill.

The idea to apply process mining in the context of workflow management systems was introduced in [2]. There, processes were modelled as directed graphs where vertices represented individual activities and edges stood for dependencies between them. Cook and Wolf, at the same time, investigated similar issues in the context of software engineering processes. In [21], they described three methods for process discovery: one using neural networks, another with a purely algorithmic technique, the last adopting a Markovian approach. The authors consider the latter two the most promising approaches. The purely algorithmic approach builds a finite state machine where states are fused if their futures (in terms of possible behavior in the next k steps) are identical. The Markovian approach uses a mixture of algorithmic and statistical methods and is able to deal with noise. Note that the results presented in [21] are limited to sequential behavior. They further extended their work to the discovery of concurrent processes in [22]. There, they also propose specific metrics,

i.e., entropy, event type counts, periodicity, and causality, so to use them in order to discover models out of event streams.

[102] describes the “ACTIVE” EU collaborative project, coordinated by British Telecom. Such project addresses the need for greater knowledge worker productivity by providing more effective and efficient tools. Among the main objectives, it aims at helping users to share and reuse informal processes, even by learning those processes from the user’s behavior.

Understanding artful processes involving knowledge workers is becoming crucial in many scenarios. Here we mention some of them:

- *personal information management (PIM)*, i.e., how to organize one’s own activities, contacts, etc. through the use of software on laptops and smart devices (iPhones/iPads, smartphones, tablets). Here, inferring artful processes in which a person is involved allows the system to be proactive and thus drive the user through its own tasks (on the basis of the past) [102, 16];
- *information warfare*, especially in supporting anti-crime intelligence agencies: let us suppose that a government bureau is able to access the email account of a suspected person. People planning a crime or an act out of law are used to speak a language of their own to express duties and next moves, where meanings may not match with the common sense. Although, a system might build the processes that lay behind their communications anyway, exposing the activities and the role of the actors. At that point, translating the sense of misused words becomes an easier task for investigators, and allows to infer the criminal activities of the person(s) under suspicion;
- *enterprise engineering*: in design and engineering, it is important to preserve more than just the actual documents making up the product data. Preserving the “soft knowledge” of the overall process (the so-called product life-cycle) is of critical importance for knowledge-heavy industries. Hence, the idea here is to take to the future not only the designs, but also the knowledge about processes, decision making, and people involved [51, 49, 85].

The objective of the approach, proposed here, is to automatically build a set of workflow models that represent the artful processes laying behind the knowledge workers activities, on top of a collection of email messages. There are many advantages out of this work. First of all, the unspecified agile processes that are autonomously used become formalized: since such models are not defined *a priori* by experts but rather inferred from real-life scenarios that actually took place, they are guaranteed to respect the true executions (often Business Process Management tools are used to show the discrepancy between the supposed and the concrete workflows). Moreover, such models can be shared, compared, preserved, so that the best practices might be put in evidence from the community of knowledge workers, to the whole business benefit. Finally, an analysis over such processes can be done, so that bottlenecks and delays in actual executions can be found out.

The approach we want to pursue essentially involves two research fields, concerning different phases of the overall processing. Through *information extraction* procedures, the tasks, which the email messages are about, are found. *Process mining* techniques are used to abstract process models representing the workflows, which the sets of subsumed tasks were considered traces of.

Our approach is named MAILOFMINE.

The following chapters outline how we realized it.

Chapter 2 analyzes the current state of the art in the areas of Information Retrieval, Process Modeling and Process Mining. Chapter 3 depicts the modular architecture of MAILOFMINE, seen as a system. The actual realization and implementation of the components sketched is outlined there as well. Chapter 4 details the new Process Mining algorithms we introduced for the discovery of declarative workflows Chapter 5 shows the results of testings over synthetic data and the user evaluations of a real case study, based on real mailboxes of people involved in this research work. Finally, Chapter 6 concludes the discussion and traces some future developments that might arise from the basis of this work.

Chapter 2

Background and State of the Art

MAILOFMINE is an approach mainly concerning different topics in the wide area of Machine Learning. Machine Learning [64] deals with the challenge of how to construct computer programs that automatically improve with experience. Our objective is to extract information out of semi-structured data, provided by email conversations, so to infer the process model laying behind the execution of the activities, which the email messages were the proof of. Then, more in detail, this work relates with Information Retrieval, since we aim at finding which email messages traced the fulfillment of activities, and Process Mining, due to the final discovery of workflows on the basis of the gathered information. In the following, we outline the related work in, resp., Sections 2.1 and 2.3. Since our work mainly focused on Process Mining, the latter is delved more in depth. We also dedicate Section 2.2 to the comparative analysis of the main process modeling foundations in literature.

2.1 Information Retrieval and Text Mining

Information Retrieval (IR) is finding material of unstructured nature (usually text) satisfying an information need, from within large collections of documents ([58]). *Text Mining*, or *Knowledge Discovery from Text* ([1]), deals with the machine supported analysis of text. It refers generally to the process of extracting interesting information and knowledge from unstructured text. It is a field in the intersection of related areas such as information retrieval, machine learning, statistics, computational linguistics and especially data mining.

Natural language text contains much information that is not directly suitable for automatic analysis by a computer. However, computers can be used to sift through large amounts of text and extract useful information from single words, phrases or passages. Therefore, text mining can be interpreted in the sense of an information extraction activity, i.e. as a restricted form of full natural language understanding, where we know in advance what kind of semantic information we are looking for. The main task in this case is to extract parts of text and assign specific attributes to it.

A particular field, related to our work, is *Text Categorization* (TC, also known as *Text Classification* or *Topic Spotting*), i.e., assigning *categories* (symbolic labels),

from a given set, to natural language texts, on the basis of *endogenous* knowledge only. I.e., knowledge is extracted from the documents only and not from other possible external sources [84]. The categories in the given set can be two (*Binary TC*, i.e., a document can belong to a category or its complement) or more. TC is applied in many contexts, such as document filtering, automated metadata generation, spam filtering. Apache Solr, MG4J[12] and Terrier [70] are famous Information Retrieval platforms.

2.1.1 Analysis of email messages

[81] proposes a method employing text mining techniques to analyze email messages collected at a customer center. The method uses two kinds of domain-dependent knowledge: (i) a key concept dictionary manually provided by human experts, and (ii) a concept relation dictionary automatically acquired by a fuzzy inductive learning algorithm. Based on [80], the mentioned method takes as input the subject and the body of an email, decides a text class for the email, extracts key concepts from email messages and finally presents their statistical information. This work shows how to cope with the information extraction in the context of email messages, for the construction of a key concept dictionary. Our aim is not restricted to the extraction of the key concept dictionary, but rather deals with the mining of activities performed on top of them; on the other side, in MAILOFMINE, we assume to rely on a user-provided dictionary of keywords.

[42] shows a technique aimed at classifying email messages into activities. Activities, in turn, can represent structured sequences of elementary units of work, as explained in their work [53]. Their tool is capable of associating the given email to one of them, on the basis of the text within the message and the people involved, represented by their email addresses appearing as senders or recipients. Our approach, conversely, attempts to learn which the structure of processes laying behind email messages is, when the activities and the workflows are unknown.

[11] focuses on the problem of finding matching addresses, i.e., clustering them so to identify contacts having more than one email address. Their methodology adopts a similarity measure based on a mixture of Levenshtein distances of surnames, names, email address base (i.e., the part which precedes the '@' character). Once the "actors" involved are identified, a social network connecting the shared work of people is built.

As the aforementioned work of Bird et al [11] investigated collaborations looking at people involved in the discussion, Cohen et al [20] looked at the problem from the viewpoint of the "Speech Act". Speech Act ([83]) are verbal utterances, which belong to the class of "illocutionary", i.e., whose significance reside in the assertion to have done something or in the request, promise or suggestion of doing something, with a "performative" value [6] – roughly speaking, *saying* to have done something, e.g., is the proof that something is done or going to be done. Thus, they had the intuition that searching for the evidence of performed activities within email messages corresponds to the analysis of the verbal contents of the conversation. In particular, they look for correspondences of some words in particular, divided into *verbs* and *nouns*, which belong to a taxonomy of terms that, mixed together, make Speech Acts. In order to properly categorize the messages, according to such "mail acts", as they call them, they compare different categorization and learning

techniques to texts that are cleaned up from quoted phrases and signatures, as in [25]. In this work, we will take inspiration from the intuition of [20] and actually make use of the implementation of [25] to filter redundant information out of email texts.

Two related famous industrial tools for e-mail content processing

Here are recalled just two famous examples of email content processing tools, which are related to our research proposal.

The first is the *Priority Inbox*¹ for *GMail*, i.e., an add-on that automatically infers which email messages are to be considered “important” and which are the so called “bologna” (i.e., less important mail) or “Bacn” (e.g., news alerts, social network messages and that kind of messages that the user is subscribed to or interested in but can stay in the inbox for a long time, unread). On the basis of such a partitioning within the set of email messages, the user interface shows the two groups separately. Thereby, the user can see the important messages at a glimpse, on top of a vertical tab that graphically distinguishes them from the rest of email.

The second is the *Xobni*² plug-in for *Microsoft Outlook*. One of the feature it provides, which is very interesting to the scope of this document, is the tracing of the contents in email messages. It is provided with the ability to create connections among people that the user conversed with and analyze their behavior in terms of time of response, frequency of communications and so on.

2.2 Process Modeling

2.2.1 Automaton-based models

A process is intended as a stateful artifact characterized by its conversational behavior, that is, its potential evolutions resulting from the interaction with some external system, such as a client service. Such conversational behavior can be represented as **Finite State Automata** (FSAs), i.e., a *transition system* whose transitions are labeled by process’ activities, under the assumption that each legal run of the system corresponds to a conversation supported by the process.

A *deterministic* process behavior is represented by a finite deterministic transition system $\mathcal{S} = \langle \mathcal{A}, S, s_0, \delta, S^f \rangle$, where:

- \mathcal{A} is the finite alphabet of activities;
- S is the finite set of states;
- $s_0 \in S$ is the initial state;
- $\delta : S \times \mathcal{O} \rightarrow S$ is the transition function;
- $S^f \subseteq S$ is the set of final states.

¹for further readings, the reader may visit the related Gmail Help page <http://mail.google.com/mail/help/priority-inbox.html> or the Google software engineer Doug Aberdeen’s post on the official Google blog, at <http://gmailblog.blogspot.com/2010/08/email-overload-try-priority-inbox.html>

²<http://www.xobni.com/>

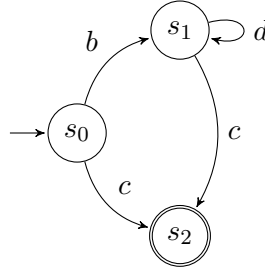


Figure 2.1. A process behavior as a deterministic automaton

The initial and final states respectively correspond to a legal initialization and termination of the process lifecycle. Thus, in Figure 2.1, the process would admit the process to either (i) perform as many b activities as she want, and finally d , then terminate, or (ii) perform c once and terminate. The transition system depicted can be formalized as follows:

$$\mathcal{S} = \langle \{a, b, c, d\}, \{s_0, s_1, s_2\}, s_0, \{ \langle \langle s_0, b \rangle, s_1 \rangle, \langle \langle s_1, d \rangle, s_1 \rangle, \langle \langle s_1, c \rangle, s_2 \rangle, \langle \langle s_0, c \rangle, s_2 \rangle \}, \{s_2\} \rangle$$

Transition systems may also be *non-deterministic*, i.e., allow a partially unpredictable behavior when a certain activity is performed. The definition changes as follows.

A *non-deterministic* process behavior is represented by a finite non-deterministic transition system $\mathcal{S} = \langle \mathcal{A}, S, s_0, \Delta, S^f \rangle$, where:

- \mathcal{A} is the finite alphabet of activities;
- S is the finite set of states;
- $s_0 \in S$ is the initial state;
- $\Delta \subseteq S \times \mathcal{O} \rightarrow S$ is the transition relation;
- $S^f \subseteq S$ is the set of final states.

Non-determinism lies in the δ function turned into a *relation* Δ , thus allowing the behavior to reach one among multiple possible states, given the execution of an activity.

An example is provided in Figure 2.2. The transition system depicted can be formalized as follows:

$$\mathcal{S} = \langle \{b, c, d\}, \{s_0, s_1, s_2\}, s_0, \{ \langle s_0, b, s_1 \rangle, \langle s_1, d, s_1 \rangle, \langle s_1, c, s_2 \rangle, \langle s_0, b, s_2 \rangle \}, \{s_2\} \rangle$$

It differs from the behavior allowed in Figure 2.1 in that it is unpredictable whether, once b is performed, other b 's, d and c can be executed, or the process terminates.

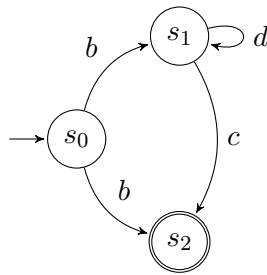


Figure 2.2. A process behavior as a non-deterministic automaton

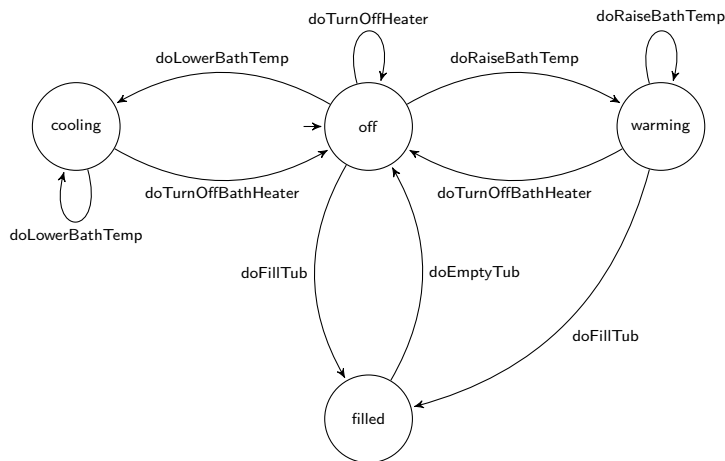


Figure 2.3. An automaton-based representation of a process in a case study [26]

Figure 2.3 draws the automaton of a case study, taken from the SM4ALL European Research Project and presented by the author of this thesis in [26].

We can consider finite state automata to be for process modeling what Assembly is for computer programming. We will make use of them to manage the representation of constraints in the declarative workflows, indeed (Sections 3.3, 3.4 and 5.2), even though not explicitly shown as a process model. Transition systems are simple and valuable in terms of expressive power, but have problems expressing concurrency succinctly. Suppose that there are n parallel activities, i.e., all n activities need to be executed but any order is allowed. There are $n!$ possible execution sequences. The transition system requires $2n$ states and $n \times 2n - 1$ transitions. This is an example of the well-known “state explosion” problem.

2.2.2 Petri-Net-based models

Petri Nets (PN’s – see Figure 2.4) originated from the Ph.D. thesis of Carl Adam Petri [76] (see [67] for a history of Petri nets and an extensive bibliography). A Petri Net is a directed bipartite graph. Its vertices can be divided into two disjoint finite sets consisting of *places* and *transitions*. Every arc of a Petri net connects a place to a transition or viceversa, but no two places neither two transitions can be directly connected. Formally, a Petri Net is a tuple $\mathcal{P} = \langle P, T, F \rangle$, where:

- P is a finite set of *places*;

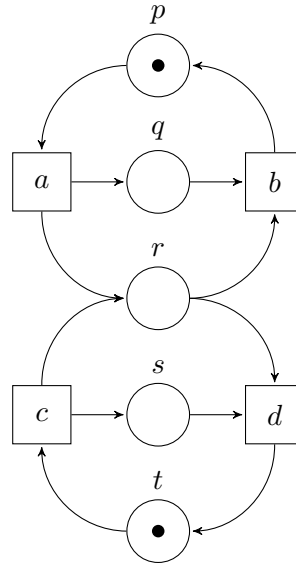


Figure 2.4. A Petri Net with its initial marking

- T is a finite set of *transitions*;
- $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*.

From a graphical point of view, places are represented by circles and transitions by rectangles. Places in a PN may contain a discrete number of marks called *tokens*. Any distribution of tokens over the places represents a configuration of the net called *marking*. Markings assign tokens (graphically represented as black dots) to places; they represent a state of the system.

Thus, modeling a workflow process definition in terms of a Petri Net is rather straightforward:

1. *tasks* are modeled by *transitions*;
2. *conditions* are modeled by *places*;
3. *cases* are modeled by *tokens*.

Formally, a marking of a Petri net is a multiset of its places, i.e., a mapping $M : P \rightarrow \mathbb{N}$. We say the marking assigns to each place a number of tokens. A transition of a Petri Net may *fire* whenever there are sufficient tokens at the start of all input arcs; when it fires, it consumes these tokens, and puts tokens at the end of *all* output arcs. Firings are atomic, i.e., single non-interruptible steps.

PN's are designed for modeling concurrency as well, as the reader can see in Figure 2.4. There, the Petri Net is

$$\mathcal{P} = \langle \{p, q, r, s, t\}, \{a, b, c, d\}, \\ \{ \langle a, q \rangle, \langle q, b \rangle, \langle a, r \rangle, \langle r, b \rangle, \langle r, d \rangle, \langle r, b \rangle, \langle b, p \rangle, \langle p, a \rangle, \\ \langle c, r \rangle, \langle c, s \rangle, \langle s, d \rangle, \langle d, t \rangle, \langle t, c \rangle \} \rangle$$

and its initial marking

$$M_0(\varphi) = \begin{cases} 1 & \text{if } \varphi \in \{p, t\} \\ 0 & \text{otherwise} \end{cases}$$

The PN in Figure 2.4 represents the parallel evolution of two separate branches of the execution, one involving a loop of c 's and d 's, the other involving loops of a ' and b 's. A possible evolution of the status of such Petri Net is depicted in Figure 2.5.

As an example, here we specify the marking for the PN in Figure 2.5b:

$$M(\varphi) = \begin{cases} 2 & \text{if } \varphi \in \{r\} \\ 1 & \text{if } \varphi \in \{q, s\} \\ 0 & \text{if } \varphi \in \{p, t\} \end{cases}$$

Workflow Nets

Typically, workflow models are drawn by means of a subset of PN's [89], i.e., the Workflow Nets (WFN's – see [88]), explicitly designed to represent the control-flow dimension of a workflow. WFN's impose structural restrictions on classical PN's, since: (i) they always have two special places, usually named i and o ; respectively, they represent a source place and a sink place, which correspond to the beginning and termination of the processing of a case; (ii) for each transition t (place p) there should be a directed path from place i to o via t (p).

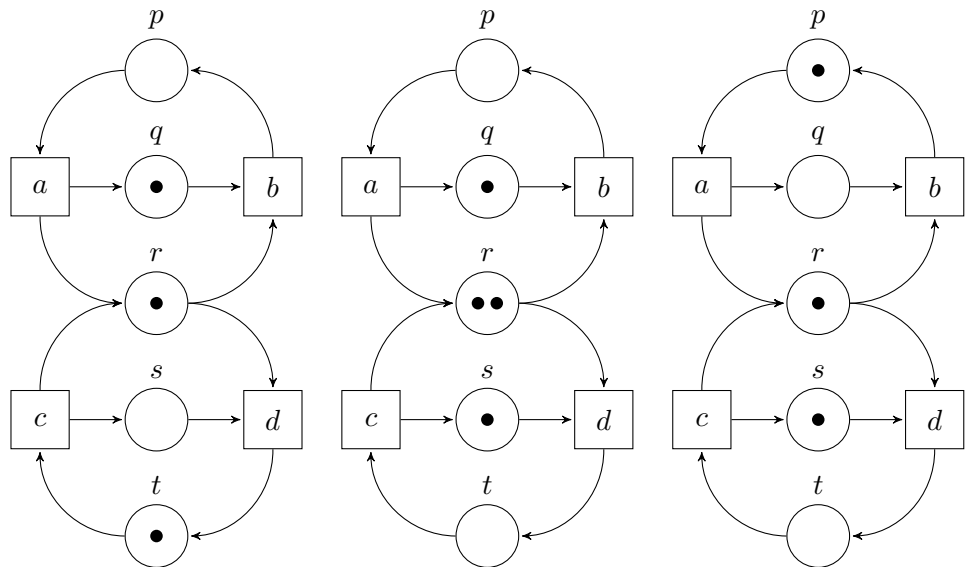
Figure 2.6 shows an example of marked Workflow Net. There, the reader can note the presence of a *parallel split* (also known as AND-split) of the workflow, in correspondence with the a transition. Firing a implies the change in the number of tokens, from one to two, put in p_1 and q_1 . From then onwards, there will be two branches running in parallel for the workflow. One will traverse p_1 , p_2 and p_3 , due to the firing of b_1 and b_2 (a *sequence*). The other instead encounters the so-called *exclusive choice* (or XOR-split) in correspondence with the q_1 place. After q_1 , the following transition to fire can be either c_1^1 or c_1^2 , but only one between them. The XOR-split is *balanced* from the following XOR-join (*simple merge*) in q_3 , as the AND-split is balanced from the following AND-join (*synchronization*) in correspondence of the d transition. Such Workflow Net is therefore called *balanced*. For further information on the workflow patterns (as sequence, XOR-split, XOR-join, AND-split and AND-join are), the reader is referred to [95].

BPMN (Business Process Modeling Notation [29]) and YAWL (Yet Another Workflow Language [94]) are two examples of widely used WFN-derived languages. For an extensive analysis of how they are used in business process modeling, the reader can refer to [106, 43] and [86].

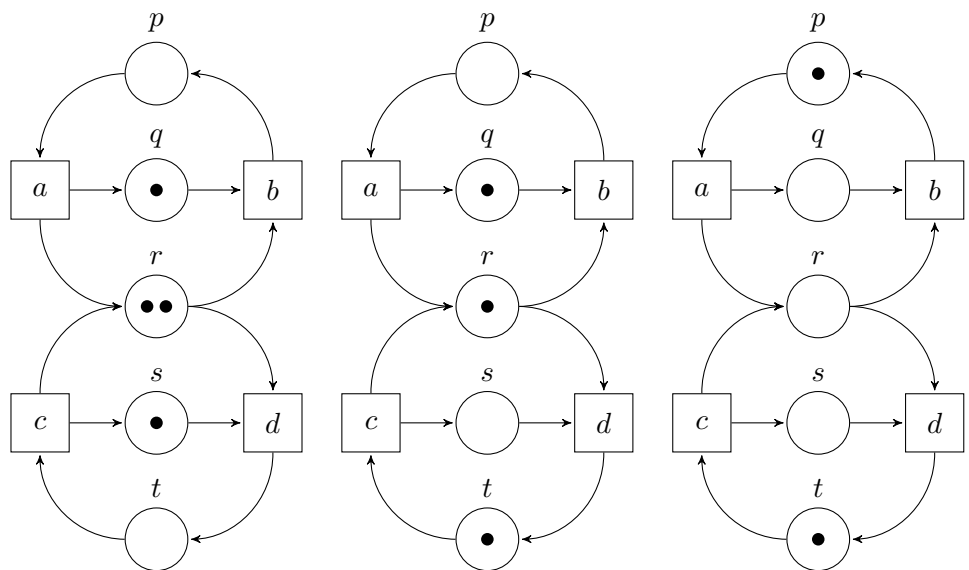
2.2.3 Declarative models

The need for flexibility in the definition of some types of process, such as artful business processes, has recently led to an alternative to the classical “imperative” approach: the “declarative” one.

The classical approach is called “imperative” because it explicitly represents every step allowed by the process model at hand, by means of transitions (the



(a) PN of Figure 2.4 after firing a (b) PN of Figure 2.5a after firing c (c) PN of Figure 2.5b after firing b



(d) PN of Figure 2.5c after firing a (e) PN of Figure 2.5d after firing d (f) PN of Figure 2.5e after firing b

Figure 2.5. A possible evolution of the status of the Petri Net in Figure 2.4

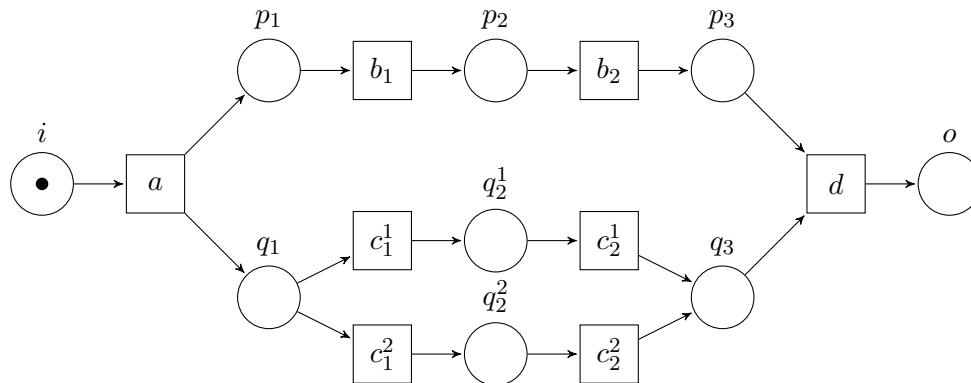


Figure 2.6. A Workflow Net, with its initial marking

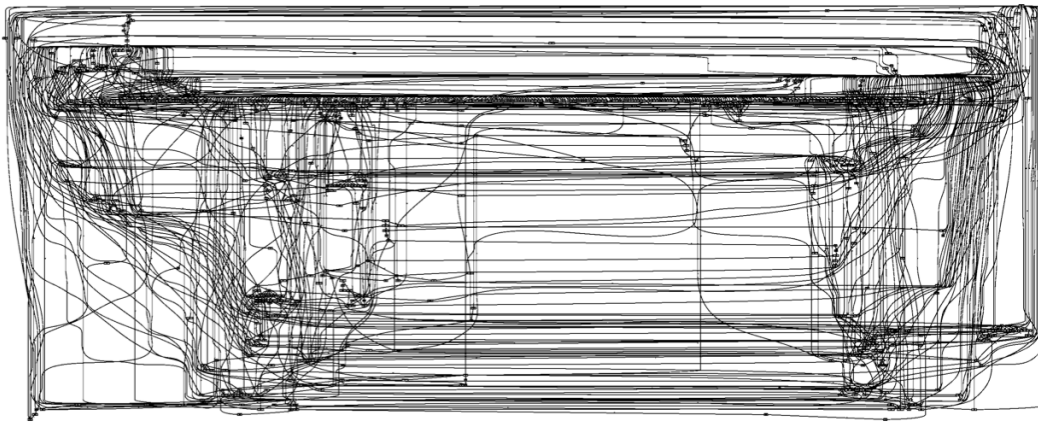


Figure 2.7. Spaghetti process describing the diagnosis and treatment of 2765 patients in a Dutch hospital. The process model was constructed based on an event log containing 114,592 events. There are 619 different activities (taking event types into account) executed by 266 different individuals (doctors, nurses, etc.). The image is taken from [90]

possible actions to do) among places/states (the legal situations where the process can wait or term). This leads to the likely increase of graphical objects as the process allows more alternative executions (see Figure 2.7). The size of the model, though, has undesirable effects on understandability and likelihood of errors (see [63] for an insight of the Seven Process Modeling Guidelines): larger models tend to be more difficult to understand [62], not to mention the higher error probability which they suffer from, with respect to small models [61].

Rather than using a procedural language for expressing the allowed sequences of activities, it is based on the description of workflows through the usage of constraints: the idea is that every task can be performed, except what does not respect them [74]. [93] shows how the declarative approach can help in obtaining a fair trade-off between flexibility in managing collaborative processes and support in controlling and assisting the enactment of workflows. DecSerFlow [92] and ConDec [75], now unified under the name of Declare [73], are languages which define an extendible set of *templates* for constraints. Their semantics are expressed as formulations of Linear Temporal Logic ([19]) as of [86], but a graphic representation for each constraint

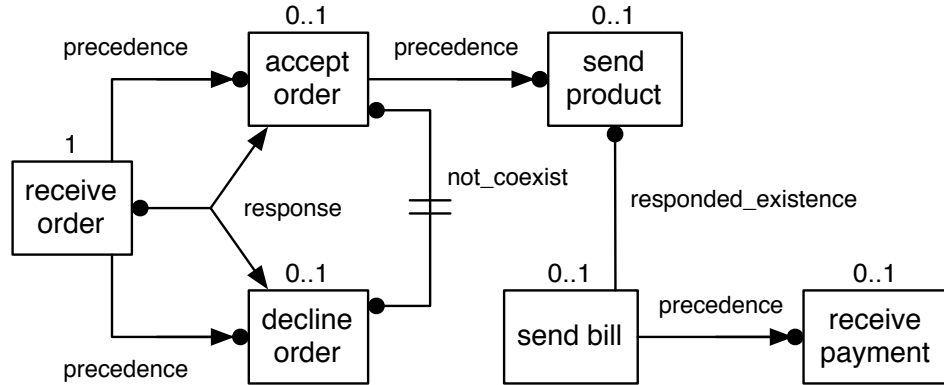


Figure 2.8. Example of a Declare constraint model [75]

template is provided as well, so to make it easily understandable by users who are not familiar with LTL.

The interested reader can find an extensive description in [71].

Declare

A constraint-based declarative Declare model is defined by the specification of a set of tasks and constraints. *Tasks* represent atomic units of work in the context of a process and are considered as single events occurring during the execution of a process instance; *constraints* define relationships between tasks and represent “rules” to be followed during execution, specifying the possible ways of executing tasks. Constraints are defined as Linear Temporal Logic (LTL) formulae (see Section 2.2.4 for the details on LTL syntax and semantics) and can be classified as (i) 1. *mandatory* constraints, i.e., constraints that a correct process execution must not violate in order to complete, and 2. *optional* constraints, i.e., constraints that can be violated.

Definition 1 (Constraint model). A constraint model (or Declare) \mathcal{CM} is a triple $\langle \mathcal{T}, \mathcal{C}_m, \mathcal{C}_o \rangle$, where:

- \mathcal{T} is the set of tasks in the model
- \mathcal{C}_m is the set of mandatory constraints, where every element $c \in \mathcal{C}_m$ is a well-formed LTL formula over \mathcal{T}
- \mathcal{C}_o is the set of optional constraints, where every element $c \in \mathcal{C}_o$ is a well-formed LTL formula over \mathcal{T}

The definition of a Declare model basically requires to:

1. identify and define the set of relevant tasks to be executed to achieve one or more intended goals;
2. identify and define the sets of mandatory and optional constraints (as LTL formulae) that reflect business constraints and restrict the set of supported execution traces.

When not explicitly specified, in this report the term *constraints* refers to *mandatory* constraints defined in a constraint model.

2.2.4 Constraint Templates in Declare

To avoid requiring users to directly define constraints using LTL and manipulate temporal logical formulae, the definition of constraints is supported via so called *constraint templates*. Constraint templates define various types of dependencies between activities at an abstract level, via a graphical syntax that allows modeling typical constraints in processes and workflows. The ConDec language can thus be considered as a collection of constraint templates. From this perspective, the idea of constraint templates is similar to the concept of workflow patterns identified for procedural languages [79]. A constraint template is defined by:

1. a unique name;
2. a graphical representation;
3. a formal specification of its semantics as an LTL formula

The set of Declare constraint templates can be classified in three main groups [92]: existence constraints, relation constraints and negation constraints. Figure 2.9 shows the main constraint templates provided by the Declare language. Tasks are graphically represented as boxes, whereas the graphical representations of possible relationships have been defined according to the following principles [66]:

- the number of lines used to interconnect two activities indicates how much tight is the dependency between them;
- the position of the • element determines which activity (called *source* of the constraint) has the ability of triggering the dependency;
- the presence of an arrow and its relative position with respect to the • element denote the qualitative temporal constraint associated to the relation.

In Figure 2.8 an example of a graphical constraint model defined for a product purchase process is provided. The Declare language is extensible, as new constraint templates can be added by defining their name and graphical representation, and providing the corresponding LTL-based formalization.

Existence constraints. Existence constraints are unary cardinality constraints defining how many times an activity can or must be executed. They can be used to represent either the minimal, the exact or the maximum number of executions (cardinality) of tasks. Among them, the *init* constraint is used to identify the first, starting activity of the model.

Relation constraints. Relation constraints define relations and dependencies between two activities. They are binary constraints which impose the presence of a certain activity when some other activity is performed, possibly imposing also qualitative temporal constraints between them.























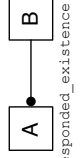
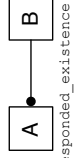
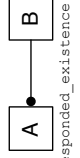
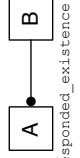
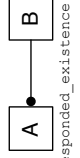
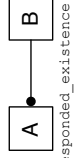
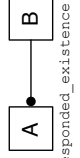
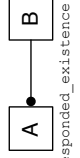
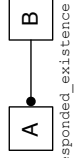
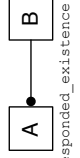
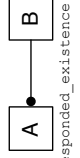
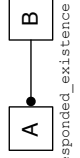
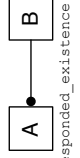
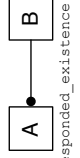
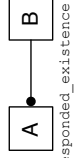
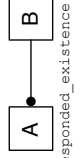
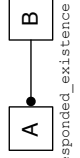
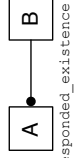
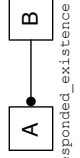
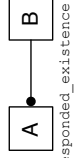
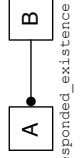
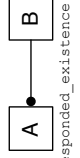
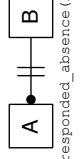
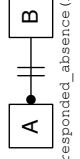
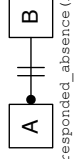
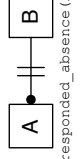
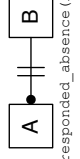
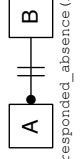
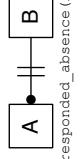
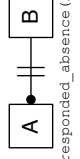
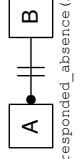
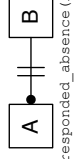
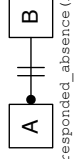
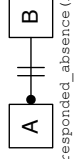
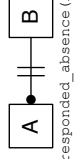
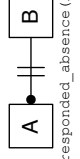
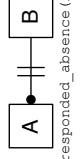
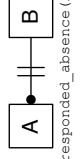
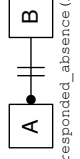
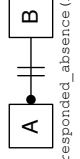
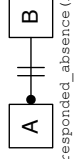
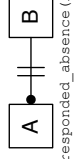
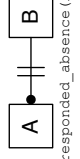
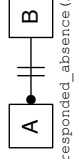
init 	LTL: A				
init (A) 	Activity A must be the first executed activity				
1... 	LTL: $\Diamond A$				
existence (A) 	Activity A must be executed at least once				
2... 	LTL: $\Diamond(A \wedge \bigcirc \text{existence}(A))$				
existence2 (A) 	Activity A must be executed at least 2 times				
N... 	LTL: $\Diamond(A \wedge \bigcirc \text{existence}_{N-1}(A))$				
existenceN (A) 	Activity A must be executed at least N times				
0 	LTL: $\Box(\neg A)$				
absence (A) 	Activity A cannot be executed				
0..1 	LTL: $\neg \text{existence}2(A)$				
absence2 (A) 	Activity A can be executed at most once, i.e. the execution trace cannot contain 2 occurrences of A				
0..2 	LTL: $\neg \text{existence}3(A)$				
absence3 (A) 	Activity A can be executed at most 2 times, i.e. the execution trace cannot contain 3 occurrences of A				
N... 	LTL: $\neg \text{existence}_{N+1}(A)$				
absenceN+1 (A) 	Activity A can be executed at most N times, i.e. the execution trace cannot contain N+1 occurrences of A				
1 	LTL: $\text{existence}(A) \wedge \text{absence}2(A)$				
exactly (A) 	Activity A must be executed exactly once				
2 	LTL: $\text{existence}2(A) \wedge \text{absence}3(A)$				
exactly2 (A) 	Activity A must be executed exactly two times				
N 	LTL: $\text{existence}_N(A) \wedge \text{absence}_{N+1}(A)$				
exactlyN (A) 	Activity A must be executed exactly N times				
	LTL: $\text{existence}(A) \Rightarrow \text{existence}(B)$				
responded_existence (A, B) 	If A is executed, then B must be executed before or after A				
	LTL: $\text{existence}(A) \Leftrightarrow \text{existence}(B)$				
coexistence (A, B) 	Neither A nor B is executed, or they are both executed				
	LTL: $\Box(A \Rightarrow \text{existence}(B))$				
response (A, B) 	If A is executed, then B must be eventually executed after A				
	LTL: $\text{existence}(B) \Rightarrow ((\neg B)UA)$				
precedence (A, B) 	B can be executed only if A has been previously executed				
	LTL: $\text{response}(A, B) \wedge \text{precedence}(A, B)$				
succession (A, B) 	A and B must be executed in succession, i.e., B must follow A and A must precede B				
	LTL: $\text{response}(A, B) \wedge \Box(A \Rightarrow \bigcirc(\text{precedence}(B, A)))$				
alternate_response (A, B) 	B is response of A and between every two executions of A, B must be executed at least once				
	LTL: $\text{precedence}(A, B) \wedge \Box(B \Rightarrow \bigcirc(\text{precedence}(A, B)))$				
alternate_precedence (A, B) 	A is precedence of B and between every two executions of B, A must be executed at least once				
	LTL: $\text{alternate_precedence}(A, B) \wedge \text{alternate_response}(A, B)$				
alternate_succession (A, B) 	B is alternate response of A, and A is alternate precedence of B				
	LTL: $\Box(A \Rightarrow \bigcirc B)$				
chain_response (A, B) 	If A is executed, then B must be executed next (immediately after A)				
	LTL: $\text{precedence}(A, B) \wedge \Box(\bigcirc B \Rightarrow A)$				
chain_precedence (A, B) 	If B is executed, then A must have been executed immediately before B				
	LTL: $\text{chain_response}(A, B) \wedge \text{chain_precedence}(A, B)$				
chain_succession (A, B) 	A and B must be executed in sequence (next to each other)				
	LTL: $\text{existence}(A) \Rightarrow \text{absence}(B)$				
responded_absence (A, B) 	If A is executed, then B can never be executed				
	LTL: $\text{responded_absence}(A, B) \wedge \text{responded_absence}(B, A)$				
not_coexistence (A, B) 	A and B exclude each other: if A is executed, then B can never be executed and vice versa				
	LTL: $\Box(A \Rightarrow \text{absence}(B))$				
negation_response (A, B) 	B cannot be executed after A				
	LTL: $\Box(\text{existence}(B) \Rightarrow \neg A)$				
negation_precedence (A, B) 	A cannot be executed before B				
	LTL: $\neg \text{response}(A, B) \wedge \text{neg_precedence}(A, B)$				
negation_succession (A, B) 	A and B cannot be executed in succession				
	LTL: $\Box(A \Rightarrow \bigcirc(A \Rightarrow \neg BU(A)))$				
neg_alt_response (A, B) 	B cannot be executed between any two occurrences of A				
	LTL: $\Box(B \Rightarrow \bigcirc(\bigcirc B \Rightarrow \neg AU(B)))$				
neg_alt_precedence (A, B) 	A cannot be executed between any two occurrences of B				
	LTL: $\neg \text{neg_alt_precedence}(A, B) \wedge \text{neg_alt_response}(A, B)$				
neg_alt_succession (A, B) 	B cannot be executed between any two occurrences of A and vice versa				
	LTL: $\Box(A \Rightarrow \bigcirc \neg B)$				
neg_chain_response (A, B) 	If A is executed, then B cannot be executed next (immediately after A)				
	LTL: $\Box(\bigcirc B \Rightarrow \neg A)$				
neg_chain_precedence (A, B) 	A cannot be executed immediately before B				
	LTL: $\neg \text{neg_chain_resp}(A, B) \wedge \text{neg_chain_prec}(A, B)$				
neg_chain_succession (A, B) 	A and B cannot be executed in sequence (next to each other)				

Figure 2.9. Existence, relation and negation constraint templates

Negation constraints. Negation constraints can be considered as the negated versions of the relation constraints: when a negation constraint is triggered by its source activity, then it forbids the execution of the target activity within certain time bounds, determined by the the specific constraint. Basically, negation constraints allow modeling undesired behaviors.

Branching of templates and choice constraints. The behavior of a constraint model defined by multiple constraints is given by the *conjunction* of all constraints. The *disjunction* of constraints can be specified by assigning more than two tasks to one parameter in a template [71] (in such a case, the parameter *branches*). In case of branching, the parameter is replaced (i) by multiple arcs to all branched tasks in the graphical representation and (ii) by a disjunction of branched tasks in the LTL formula. The semantics of branching depends on the LTL formula of the template. Choice templates are branching templates that can be used to specify a choice between tasks. They are n-ary constraints specifying that some activities inside a set of possible choices must be performed. They can be considered as an extension of the **existenceN** and **exactlyN** constraints, replacing a single activity with a set of activities, i.e., at least n distinct activities out of m must be performed or exactly n distinct activities out of m must be performed.

Execution traces When considering a process model, traces represent sequences of tasks executed during possible enactments of the process. Each trace represents a possible execution alternative as a sequence of events corresponding to the execution of tasks.

Definition 2 (Execution trace). *Given the set \mathcal{T} of tasks defined in a constraint model \mathcal{CM} , an execution trace $\sigma \in \mathcal{T}^*$ is a finite sequence of tasks $\langle t_0, t_1, \dots, t_{n-1} \rangle$, where \mathcal{T}^* is the set of all possible traces composed of zero or more elements (tasks) of \mathcal{T} . $|\sigma| = n$ is the length of the trace, σ_i is used to denote the i -th element of the trace, and σ^i is used to denote the suffix of σ starting at i , i.e., $\sigma^i = \langle t_i, t_{i+1}, \dots, t_{n-1} \rangle$.*

In a process instance, the execution of tasks generates a history trace for that instance, as a chronologically ordered list of events that occurred in the instance [74].

The Role of Linear Temporal Logic in Constraint Models

Languages such as Declare use LTL formulae to define constraints that implicitly identify possible executions of a model as sequences of events corresponding to performed tasks. As described later in this report, the LTL representation of Declare models can be exploited for the verification of model properties and for the enactment and monitoring of model instances. However, LTL deals with *infinite* traces, whereas the executions of process instances eventually terminate and thus correspond to *finite* execution traces (see Definition 2). As a consequence, the infinite semantics of LTL can not be directly applied to Declare models.

This section provides an introduction to LTL syntax and semantics, and then focuses on how its infinite semantics can be adapted to deal with finite traces corresponding to the execution of Declare models.

LTL Syntax and Semantics Temporal logics are a special class of modal logics where modalities are interpreted as temporal operators, used to describe and reason about how the truth values of assertions vary over time [66]. In this class, LTL can be considered as being: (i) 1. *propositional*, as formulae are defined from atomic propositions, whose truth values change over time; 2. *linear*, as temporal operators predicate on the truth of propositions along a single timeline and each moment has a single next future moment; 3. *qualitative*, as temporal operators are used to express qualitative time relations between propositions; 4. *point-based*, as temporal operators and propositions are evaluated over points in time; 5. *discrete*, as the present moment corresponds to the current state of the system and the next moment to the immediate successor state induced by the occurrence of an event (i.e., time is discrete); 6. *future-tense*, as temporal operators predicate on the occurrence of events in the future.

Basically, LTL formulae are defined using atomic propositions (with *true* and *false* constants), propositional connectives (\neg , \wedge , \vee , \Rightarrow), and temporal operators (\bigcirc next time, \square globally, \diamond eventually, \mathcal{U} until).

Definition 3 (LTL syntax). *Given a finite set \mathcal{P} of atomic propositions, the set of LTL-formulae over \mathcal{P} is inductively defined as follows:*

- every $p \in \mathcal{P}$ is a formula
- true and false are formulae
- if φ is a formula, then $\neg\varphi$ is a formula
- if φ and ψ are formulae, then $\varphi \wedge \psi$, $\varphi \vee \psi$ and $\varphi \Rightarrow \psi$ are formulae
- if φ is a formula, then $\bigcirc\varphi$, $\square\varphi$ and $\diamond\varphi$ are formulae
- if φ and ψ are formulae, then $\varphi\mathcal{U}\psi$ is a formula

LTL models and traces. The semantics of LTL is defined with respect to an LTL model in a specific state and temporal formulae are interpreted in a discrete, linear model of time. Given the set \mathcal{P} of atomic propositional formulae, an LTL model is represented by $\mathcal{M} = \langle \mathbb{N}, I \rangle$, where $I : \mathbb{N} \mapsto 2^{\mathcal{P}}$ maps each moment in time (represented by a natural number) to a set of propositions that represents all the propositions $p \in \mathcal{P}$ that hold in that moment in time. From an other perspective, an LTL model or interpretation of an LTL formula can be considered as an *infinite* trace $\pi = \pi_0, \pi_1, \dots$ having \mathbb{N} as time structure and \mathcal{P} as the set of atomic propositions; each element $\pi_i \in 2^{\mathcal{P}}$ of the trace is defined by I and thus refers to the set of propositions that hold at the i -th moment in time (i.e., $\pi_i = I(i)$). At some time point $i \in \mathbb{N}$ a proposition p is true iff $p \in \pi_i$. $\langle \pi, i \rangle \models \varphi$ means that a formula φ is true at time i in π (i.e., trace π satisfies φ), and \models denotes the logical entailment (or satisfaction relation).

Definition 4 (LTL semantics). *Given an infinite execution trace π and an LTL formula φ , $\pi \models \varphi$ is inductively defined on the structure of the formulae as follows:*

- $\pi \models \varphi$ iff $\langle \pi, 0 \rangle \models \varphi$

- $\langle \pi, i \rangle \models p$ iff $p \in \pi_i$
- $\langle \pi, i \rangle \models \text{true}$ and $\langle \pi, i \rangle \not\models \text{false}$
- $\langle \pi, i \rangle \models \neg\varphi$ iff $\langle \pi, i \rangle \not\models \varphi$
- $\langle \pi, i \rangle \models (\varphi \wedge \psi)$ iff $\langle \pi, i \rangle \models \varphi$ and $\langle \pi, i \rangle \models \psi$
- $\langle \pi, i \rangle \models (\varphi \vee \psi)$ iff $\langle \pi, i \rangle \models \varphi$ or $\langle \pi, i \rangle \models \psi$
- $\langle \pi, i \rangle \models (\varphi \Rightarrow \psi)$ iff $\langle \pi, i \rangle \not\models \varphi$ or $\langle \pi, i \rangle \models \psi$
- $\langle \pi, i \rangle \models \bigcirc\varphi$ iff $\langle \pi, i+1 \rangle \models \varphi$
- $\langle \pi, i \rangle \models \Box\varphi$ iff $\forall j \geq i \langle \pi, j \rangle \models \varphi$
- $\langle \pi, i \rangle \models \Diamond\varphi$ iff $\exists j \geq i$ s.t. $\langle \pi, j \rangle \models \varphi$
- $\langle \pi, i \rangle \models (\varphi \mathcal{U} \psi)$ iff $\exists k \geq i$ s.t. $\langle \pi, k \rangle \models \psi$ and $\forall i \leq j < k \langle \pi, j \rangle \models \varphi$

The operators \neg , \wedge , \bigcirc and \mathcal{U} are the basic operators, and the other operators can be expressed in terms of the basic ones; the following equivalences hold:

- $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$
- $\text{true} \equiv \varphi \vee \neg\varphi$
- $\text{false} \equiv \neg\text{true}$
- $\varphi \Rightarrow \psi \equiv \neg\varphi \vee \psi$
- $\varphi \Leftrightarrow \psi \equiv (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$
- $\Diamond\varphi \equiv \text{true} \mathcal{U} \varphi$
- $\Box\varphi \equiv \neg\Diamond\neg\varphi$

From LTL to LTL for Finite Traces As discussed in the previous Section, an interpretation of an LTL formula is an *infinite* trace $\pi = \pi_1, \pi_2, \dots$, where each element π_i of the trace identifies the set of all propositions $p \in \mathcal{P}$ that are true at time i . When considering Definition 2 it is possible to identify two main differences between a trace representing an LTL interpretation and a trace representing an execution trace of a constraint model [72]:

1. LTL considers *infinite* traces, whereas the execution of a process model generates a *finite* execution trace as a sequence of executed tasks: the temporal dimension in process executions is *bounded*, and therefore their execution traces are always finite;

2. each element of an LTL trace can refer to a *set* of propositions, whereas each element in a finite execution trace of a Declare model refers to exactly one event corresponding to the execution of a task t_i , i.e., only *one* proposition holds at one moment.

The notion of finite trace can be considered in LTL setting as follows.

Definition 5 (Finite LTL trace). *An LTL execution trace π is finite if and only if there exists a state n such that in all the states belonging to π^{n+1} (i.e., all states belonging to the infinite suffix of π starting at $n+1$) the set of propositions that hold is empty, i.e.:*

$$\exists n \in \mathbb{N} \text{ s.t. } \pi_n \neq \emptyset \wedge \forall i > n, \pi_i = \emptyset$$

Note that, according to this definition, a *finite* LTL trace still has an *infinite* suffix composed by empty elements. However, standard algorithms and techniques typically used in the LTL settings with infinite traces are not directly applicable with finite execution traces. The LTL semantics is given for infinite traces and assumes \mathbb{N} as the underlying infinite time structure. To bridge the gap between finite traces and the LTL semantics for infinite traces, two different solutions have been proposed for Declare models:

1. introduce an additional task and a corresponding constraint as an LTL formula that allows specifying that each execution of the model will eventually terminate;
2. redefine LTL models as finite traces and update the LTL semantics accordingly.

Termination property. The first solution does not require to modify the LTL semantics, but requires to introduce in each Declare model an *ending* task e and a *termination constraint* that specifies that (i) the *end* task will be eventually executed and (ii) after the execution of this task, the task is executed infinitely in each following state (and no other task is executed); the corresponding LTL formula is:

$$\diamond e \wedge (\square(e \Rightarrow \circ e))$$

As a result, each execution trace will have an infinite suffix containing only the *ending* task e . From another perspective, the *termination constraint* that formalizes the finiteness requirement for a Declare execution trace can be expressed as a *termination property*.

Definition 6 (Termination property). *Given a constraint model \mathcal{CM} , the LTL termination property of \mathcal{CM} , denoted by $\text{term}(\mathcal{CM})$, states that there must eventually be a state starting from which no event will occur (i.e., no task in the model will be executed):*

$$\text{term}(\mathcal{CM}) \equiv \diamond \square \left(\bigwedge_{\forall t \in \mathcal{T}} \neg t \right)$$

Finite LTL. The second solution requires to directly consider finite LTL traces as the only possible models for the logic, and revise the semantics of temporal operators accordingly. Different finite trace semantics for LTL have been proposed and a complete overview of them can be found in [8]. In the context of Declare the authors rely on the finite trace semantics proposed in [45] and [48]. The proposed semantics deals with finite execution traces, which can be considered either as the observed portion of a model's execution (i.e., a partial execution trace) or as a complete finite run.

Basically, the semantics of LTL temporal operators (\diamond , \square and \mathcal{U}) is modified in order to limit the scope to the finite set of states of an execution trace. Moreover, when interpreting LTL formulae over finite traces, it is necessary to define the semantics of the *next* \circ operator on a trace with a single element (or similarly on the last element of a finite trace) for which no next position exists to evaluate a formula. According to [8], the classical way to deal with this situation is to consider \circ as a *strong* next operator, which evaluates to *false* if no further position exists in the trace. The strong \circ operator is thus used to express with $\circ\varphi$ that (i) 1. a next state must exist, and 2. this next state has to satisfy property φ .

Definition 7 (LTL for finite traces). *Given a finite set of atomic propositions \mathcal{P} , every $p \in \mathcal{P}$ is a well-formed LTL formula. If φ and ψ are well-formed LTL formulae, then true, false, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$, $\circ\varphi$, $\square\varphi$, $\diamond\varphi$ and $\varphi\mathcal{U}\psi$ are well-formed LTL formulae.*

Let $\sigma = \langle \sigma_0, \sigma_1, \dots, \sigma_{n-1} \rangle$ be a finite trace of length n , with $\sigma \neq \varepsilon$, where ε is the empty trace. σ^i denotes the suffix $\langle \sigma_i, \sigma_{i+1}, \dots, \sigma_{n-1} \rangle$ for $0 \leq i < n$ and the empty string ε for $i \geq n$. The semantics of LTL is defined as follows³:

- $\sigma \models p$ iff $p \in \sigma_0$, for $p \in \mathcal{P}$
- $\sigma \models \neg\varphi$ iff $\sigma \not\models \varphi$
- $\sigma \models (\varphi \wedge \psi)$ iff $\sigma \models \varphi$ and $\sigma \models \psi$
- $\sigma \models (\varphi \vee \psi)$ iff $\sigma \models \varphi$ or $\sigma \models \psi$
- $\sigma \models (\varphi \Rightarrow \psi)$ iff $\sigma \not\models \varphi$ or $\sigma \models \psi$
- $\sigma \models \circ\varphi$ iff $\sigma^1 \neq \varepsilon$ and $\sigma^1 \models \varphi$
- $\sigma \models \square\varphi$ iff $\forall i, 0 \leq i < n, \sigma^i \models \varphi$
- $\sigma \models \diamond\varphi$ iff $\exists i, 0 \leq i < n$, s.t. $\sigma^i \models \varphi$
- $\sigma \models (\varphi\mathcal{U}\psi)$ iff $\exists k, 0 \leq k < n$, s.t. $\sigma^k \models \psi$ and $\forall i, 0 \leq i < k, \sigma^i \models \varphi$

The LTL semantics for finite traces provided in the previous definition is used for the mandatory and optional constraints (expressed as LTL formulae) defined in Declare models.

³Note that the semantics is not defined for the empty trace ε

Mandatory Formula and Supported Traces

According to Section 2.2.4, a Declare model \mathcal{CM} can be built by defining the set of relevant tasks and the optional and mandatory constraints among them, exploiting constraint templates. Each constraint template can be specified by an LTL formula, whose semantics is given according to Definition 7. When considering the set \mathcal{C}_m of mandatory constraints in a Declare model (cf. Definition 1), it is possible to define a single LTL formula for the whole model, as the conjunction of the LTL formulae defining the mandatory constraints.

Definition 8 (Mandatory formula). *Given a constraint model $\mathcal{CM} = \langle \mathcal{T}, \mathcal{C}_m, \mathcal{C}_o \rangle$, the mandatory formula for \mathcal{CM} is defined as:*

$$f_{\mathcal{CM}} = \begin{cases} \text{true} & \text{if } \mathcal{C}_m = \emptyset \\ \bigwedge_{c \in \mathcal{C}_m} c & \text{otherwise} \end{cases}$$

Basically, the semantics of LTL entailment \models can be used to define the notion of *compliance* of an execution trace $\sigma \in \mathcal{T}^*$ with a constraint formula c .

Definition 9 (Constraint satisfying trace). *Given a constraint model $\mathcal{CM} = \langle \mathcal{T}, \mathcal{C}_m, \mathcal{C}_o \rangle$ and an execution trace $\sigma \in \mathcal{T}^*$, σ is compliant with a constraint formula c (i.e., σ satisfies c) if and only if $\sigma \models c$.*

Starting from Definition 9, it is possible to define the set of all executions traces compliant with a constraint c as the set of all traces satisfying the constraint.

Definition 10 (Constraint satisfying traces). *Given a constraint model $\mathcal{CM} = \langle \mathcal{T}, \mathcal{C}_m, \mathcal{C}_o \rangle$, let c be a constraint for \mathcal{CM} . The set $\mathcal{T}_{\models c}^* \subseteq \mathcal{T}^*$ of constraint satisfying traces is defined as*

$$\mathcal{T}_{\models c}^* = \{\sigma \in \mathcal{T}^* \mid \sigma \models c\}$$

These definitions can be easily extended to the mandatory formula for a constraint model.

Definition 11 (Constraint model satisfying trace). *Given a constraint model $\mathcal{CM} = \langle \mathcal{T}, \mathcal{C}_m, \mathcal{C}_o \rangle$, let $f_{\mathcal{CM}}$ be the mandatory formula for \mathcal{CM} . An execution trace $\sigma \in \mathcal{T}^*$ is compliant with \mathcal{CM} if and only if $\sigma \models f_{\mathcal{CM}}$, i.e., $\forall c \in \mathcal{C}_m, \sigma \models c$.*

Note that if assuming an LTL setting on infinite traces with the additional termination property given in Definition 6, an execution trace σ is *compliant* with \mathcal{CM} if and only if

$$\sigma \models f_{\mathcal{CM}} \wedge \text{term}(\mathcal{CM})$$

Starting from Definition 11, it is then possible to define the set of all executions traces compliant with a constraint model \mathcal{CM} as the set of all traces satisfying the mandatory formula for the model.

Definition 12 (Constraint model satisfying traces). *Given a constraint model $\mathcal{CM} = \langle \mathcal{T}, \mathcal{C}_m, \mathcal{C}_o \rangle$, let $f_{\mathcal{CM}}$ be the mandatory formula for \mathcal{CM} . The set $\mathcal{T}_{\models \mathcal{CM}}^* \subseteq \mathcal{T}^*$ of constraint model satisfying traces (or supported traces) is defined as*

$$\mathcal{T}_{\models \mathcal{CM}}^* = \{\sigma \in \mathcal{T}^* \mid \sigma \models f_{\mathcal{CM}}\}$$

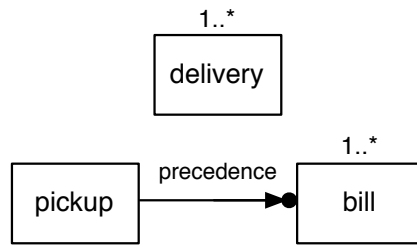


Figure 2.10. A simple constraint model [86]

From the definitions it follows that if in a constraint model the set of mandatory constraints is empty ($\mathcal{C}_m = \emptyset$), all possible execution traces over \mathcal{T} satisfy the mandatory formula for the model, i.e., $\mathcal{T} \models_{\mathcal{C}\mathcal{M}} \mathcal{T}^*$.

Figure 2.10 shows a simple constraint model. Assuming that all represented constraint are mandatory, the constraint model $\mathcal{C}\mathcal{M} = \langle \mathcal{T}, \mathcal{C}_m, \mathcal{C}_o \rangle$ has:

- $\mathcal{T} = \{\text{pickup}, \text{bill}, \text{delivery}\}$
- $\mathcal{C}_m = \{\textit{existence}(\text{delivery}), \textit{existence}(\text{bill}), \textit{precedence}(\text{pickup}, \text{bill})\}$
- $\mathcal{C}_o = \emptyset$

The mandatory formula for the model is defined as:

$$\begin{aligned} f_{\mathcal{C}\mathcal{M}} &= \textit{existence}(\text{delivery}) \wedge \textit{existence}(\text{bill}) \wedge \textit{precedence}(\text{pickup}, \text{bill}) \\ &= (\diamond \text{delivery}) \wedge (\diamond \text{bill}) \wedge (\diamond \text{bill} \Rightarrow (\neg \text{bill} \mathcal{U} \text{pickup})) \end{aligned}$$

Considering the constraint $c = (\diamond \text{bill} \Rightarrow (\neg \text{bill} \mathcal{U} \text{pickup}))$, the following traces are some of the constraint satisfying traces:

- $\sigma_1 = \langle \text{pickup}, \text{bill}, \text{delivery} \rangle$
- $\sigma_2 = \langle \text{pickup}, \text{pickup}, \text{pickup}, \text{delivery}, \text{delivery}, \text{bill}, \text{bill} \rangle$
- $\sigma_2 = \langle \text{pickup}, \text{pickup}, \text{delivery}, \text{delivery} \rangle$

2.3 Process Mining

Process Mining [90], a.k.a. *Workflow Mining* [89], is the set of techniques that allow the extraction of structured process descriptions, stemming from a set of recorded real executions. Such executions are intended to be stored in so called *event logs*, i.e., textual representations of a temporarily ordered linear sequence of tasks. There, each recorded *event* reports the execution of a *task* (i.e., a well-defined step in the workflow) in a *case* (i.e., a workflow instance). Events are always recorded sequentially, even though tasks could be executed in parallel: it is up to the algorithm to infer the actual structure of the workflow that they are traces of, identifying the causal dependencies between tasks (*conditions*). ProM [96] is one of the most used plug-in based software environment for implementing workflow mining techniques.

The idea to apply process mining in the context of workflow management systems was introduced in [2]. There, processes were modelled as directed graphs where vertices represented individual activities and edges stood for dependencies between them. Cook and Wolf, at the same time, investigated similar issues in the context of software engineering processes. In [21] they described three methods for process discovery: (i) neural network-based, (ii) purely algorithmic, (iii) adopting a Markovian approach. The authors considered the latter two as the most promising. The purely algorithmic approach built a finite state machine where states were fused if their *futures* (in terms of possible behaviors for the next k steps) were identical. The Markovian approach used a mixture of algorithmic and statistical methods and is able to deal with noise. Although, the results presented in [21] were limited to sequential behavior only.

From [2] onwards, many techniques have been proposed, in order to address specific issues: pure algorithmic (e.g., α algorithm, drawn in [97] and its evolution α^{++} [104]), heuristic (e.g., [103]), genetic (e.g., [28]), etc. Indeed, heuristic and genetic algorithms have been introduced to cope with noise, which the pure algorithmic techniques were not able to manage. Whereas algorithmic processes rely on footprints of traces (i.e., tables reporting whether events appeared before or afterwards, if decidable) to determine the workflow net that could have generated them, heuristic approaches build a representation similar to causal nets, taking frequencies of events and sequences into account when constructing the process model, in order to ignore infrequent paths. Genetic process mining adopts an evolutionary approach to the discovery and differs from the other two in that its computation evolves in a non-deterministic way: the final output, indeed, is the result of a simulation of a process of natural selection and evolutionary reproduction of the procedures used to determine the final outcome. [13] discusses in depth the user-tunable metrics adopted for the genetic algorithm, in order to make it return qualitatively better workflows: replay fitness, precision, generalization and simplicity. The accurate results are valuable, though such an algorithm suffers from unpredictability in terms of the process returned, which can change from run to run, due to the nature of evolutionary algorithms itself, and the time it might take, which is generally high.

A very smart extension to the previous research work has been recently achieved by the two-steps algorithm proposed in [87].

Differently from previous works, which typically provide a single process mining step, it splits the computation in two phases: (i) the tunable mining of a Transition System (TS) representing the process behavior and (ii) the automated construction of a Petri Net bisimilar to the TS [23, 30].

The first phase was made “tunable”, so that it could be either more strictly adhering or more permissive to the analyzed log traces behavior, i.e., the expert could determine a balance between “overfitting” and “underfitting”.

Indeed, past execution traces are not the whole universe of possible ones that may run: hence, the extracted process model should be valid for future unpredictable cases, on one hand, nevertheless checking whether the latter actually adhere to the common behavior, on the other hand. We also remark that a little percentage of the whole log might represent erroneous deviations from the natural flow of tasks. The second phase had no parameter to set, since its only aim was to synthesize the TS into an equivalent Workflow Net. Thus, it was fixed, while the former step could be realized exploiting one among many of the previously proposed “one-step”

algorithms (for instance, [103] was claimed to integrate well).

The need for flexibility in the definition of some types of process, such as artful business processes, lead to an alternative to the classical “imperative” approach: the “declarative”. Rather than using a procedural language for expressing the allowed sequences of activities, it is based on the description of workflows through the usage of constraints: the idea is that every task can be performed, except what does not respect them. [93] showed how the declarative approach (such as the one of Declare [73]) can help in obtaining a fair trade-off between flexibility in managing collaborative processes and support in controlling and assisting the enactment of workflows.

[57] outlines an algorithm for mining Declare processes, implemented in ProM. The approach works as follows. The user is asked to specify a set of Declare constraint templates. Then, the system generates all of the possible constraints stemming from them, i.e., obtained by the application of those templates to all of the possible subsets of activities in the process. The user is also required to set a parameter named PoE (Percentage of Events), ranging from 0 to 100, so to avoid the generation of constraints involving those activities that appear, in percentage, less than PoE times in the log (i.e., rare activities, according to the user preferences). Then, every candidate constraint is translated into the related accepting finite automata, according to the rules defined in [72]. For the optimization of this task, the tool is integrated with the technique described in [105]. Traces are thus replayed on the resulting automata. Each constraint among the candidates becomes part of the discovered process if and only if the percentage of traces accepted by the related automaton exceeds a user-defined threshold, named PoI (Percentage of Instances).

[56] proposes an evolution of [57]: there, a two-phase approach is adopted. The first phase is based on the Apriori algorithm, developed by Agrawal and Srikant for mining association rules [3]. During this preliminary step, the correlated activity sets are identified. The candidate constraints are computed on the basis of the correlated activity sets only. During the second phase, the candidate constraints are checked as in [57]. Therefore, the search space for the second phase is reduced.

In output, constraints constituting the discovered process are weighted according to their *Support*, i.e., the probability of such constraint to hold in the mined workflow. There, it is calculated as the proportion of traces where the constraint is satisfied. To filter out irrelevant constraints, more metrics are introduced, based on the appearances of the activities involved within the log: they are *Confidence*, *Interest Factor* and *CPIR* (Conditional-Probability Increment Ratio). Since we also adopted such metrics, with slight modifications though, they will be described further in this thesis.

[54, 17] describes the usage of inductive logic programming techniques to mine models expressed as a SCIFF [4] first-order logic theory, consisting in a set of implication rules named Social Integrity Constraints (IC’s for short). [18] shows how ConDec/DecSerFlow can be translated into SCIFF and a subset of SCIFF can be translated into Condec/Decserflow. Finally, the learned theory is automatically translated into the ConDec [75] notation.

[17] proposes the implementation of the framework, named DPML (Declarative Process Model Learner [55]) as a ProM plug-in. [10, 9] extends this technique by weighting in a second phase the constraints with a probabilistic estimation. The learned IC’s are indeed translated from SCIFF, discovered by DPML, into Markov

Logic [77] formulae, so that their probabilistic-based weighting is computed by the Alchemy tool. They both rely on the availability of compliant and non-compliant traces of execution, w.r.t. the process to mine. For instance, [54] takes a real log from cervical cancer screening careflows. All the traces have been analyzed by a domain expert and labeled as compliant or non compliant with respect to the protocol, adopted in the screening center. [10] takes as a case study the records of the careers which belonged to their affiliated university's students. In that case, positive traces were represented by graduated students, whilst negative traces were related with students who did not finish their studies. For a comprehensive insight on the logic-based approach to declarative workflow mining, the reader can refer to [66].

As the aforementioned logic-based approaches, we preferred to elaborate a technique which avoided the replay of the traces on automata, as in [57], so to diminish the time for computing the result. At the same time, we had to deal with traces which were not labeled as positive or negative.

2.3.1 Analysis of email messages

EMailAnalyzer [91] is an integrated ProM plug-in for mining processes from email logs, that are XML files compatible with the ProM framework, built through the analysis of *(i)* senders and receivers, in order to guess the actors involved, and *(ii)* tags on email messages and subjects, for extracting the task. email messages are extracted from an Outlook archive. Our approach shares similar ideas for the disambiguation of actors and sociograms, and aims at extending it by *(i)* building a plug-in based platform capable to retrieve email messages from multiple archives (not only Outlook), and *(ii)* extracting cases and relations among tasks from a more comprehensive analysis of email fields (headers, body, subjects, etc.).

Chapter 3

Architecture and design

3.1 Architecture of MailOfMine as a software system

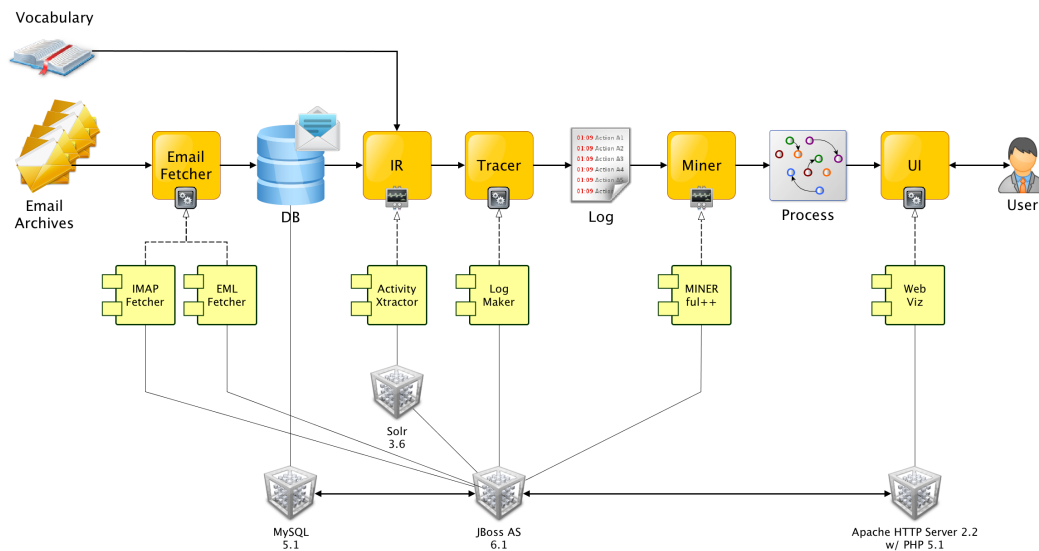


Figure 3.1. The MAILOFMINE approach

The MAILOFMINE approach, and the tool, adopts a modular architecture, the components of which allow to incrementally refine the mining process, as in Figure 3.1.

The main input are email archives; an archive is a stored collection of email messages. First of all, we need to extract email messages out of the given archive(s). Since archives are compliant to different standards, according to the email client in use, the component for accessing (“fecthing”) the messages (**Email Fetcher** in Figure 3.1) is intended to be plug-in based: At this stage of the implementation, the system is able to process archives which can be either IMAP folders (through the IMAP Fetcher), or compressed files with .eml documents in (EML Fetcher). In the future, it will handle Thunderbird storage files, POP mailboxes, etc. Before getting stored, email messages text parts are cleaned up from signatures and quotations

citing some text already written in another message within the thread, by the usage of a combination of techniques aiming at this [25, 69]. For what the methodology of [25] is concerned, we made use of its Java implementation, named Jangada¹. The outcome is the population of a database, on the basis of which all the subsequent steps are carried out. In our implementation, the database is managed by the well known MySQL Server, Community Edition, version 5.1².

In order to extract the activities whose execution was proven by the flow of messages, we adopt an approach based on the concept of Speech Acts ([83]) applied to email messages. The detection of Speech Acts needs to be assisted by knowledge workers/users, who are required to provide a dictionary of words of their domain field, as in [20]. The provided words have to be divided into *verbs* and *objects* – say, “write” and “submit” for verbs, “draft” and “deliverable” for objects. Each object, indeed, is concatenated with each verb and the resulting strings are kept in a collection of *expressions* (“write draft”, “submit draft”, “write deliverable”, “submit deliverable” in the example). For each expression, an Information Retrieval (IR) tool is used in order to search them within the email messages, considering their cleaned subjects and bodies. Only those expressions that are found in at least one email are considered *activities* for the process to discover: meaningless or irrelevant expressions, possibly created by the automated juxtaposition of verbs and objects, are thereby likely filtered out. If, for instance, “submit draft” is never found in any email, the collection of activities is composed by “write draft”, “write deliverable”, “submit deliverable” in the example. The so called *process alphabet*, namely, the collection of activities, is thus compiled. We developed the Activity Xtractor software component to this aim. All of the email messages where an expression is found are called *indicia*.

Each *indiciu*m is a possible evidence of the execution of the activity that the expression relates to. We also consider the *reliability* of an *indiciu*m, which corresponds to the *score* that the IR assigned to the email, when searching for the related activity’s expression.

The Activity Xtractor component makes use of the industry-strong search platform Apache Solr, version 3.6.2³. Apache Solr was particularly suitable to our case, because of three main reasons: (i) its ability to index textual information stored not only in text files but also in database; (ii) its rich and well-documented API making it possible for us to automatically submit run-time queries, and store their results; (iii) the availability of tuning parameters for queries, such as the Levenshtein distance for metrics and the customizable language-dependent synonyms. Its relevance assessment core for indexing makes use of an altered version of the TF-IDF metrics ([68, 58]), as explained in [44].

Once *indicia* are found, a *log* (i.e., a collection of *traces*) is created from the Tracer module:

1. each archive is taken as a trace, i.e., a list of events reporting the execution of an activity;
2. each *indiciu*m (i.e., each email) whose reliability is higher than a user-customizable threshold is taken as an event;

¹<http://www.cs.cmu.edu/~vitor/codeAndData.html>

²<http://dev.mysql.com/>

³<http://lucene.apache.org/solr/>

3. the events are ordered with respect to the date and time when the related emails were sent.

As a simplistic assumption, when more than one indicium is found to match a single email, we associate the email to the highest scored indicium. Thus, we have one indicium only per email and no couple of events occurs at the same timestamp in a trace. We have already planned to remove this assumption and deal with contemporaneity of events in logs for future work (see 6.1). The construction of the log is obtained through the processing of the evaluation of a proper SQL query (see Appendix A.1) over the shared database, whose results are formatted in XML and further turned into a XES (eXtensible Event Stream [47]) log, by means of a XSLT (eXtensible Stylesheet Language Transformation [101]) transformation – the XSLT document is available in Appendix A.3. XES is an XML-based standard for machine-readable event logs [100].

Being email messages associated to the email addresses of people in the conversation, the system infers who are the most likely contacts involved in the activities, simply observing what are the email addresses appearing as senders and recipients.

The XES log is passed further to the Process Mining tool (Miner, which we implemented as a realization of MINERful (see Chapter 4 for further details) in the MINERful++ component. The output is a process model, discovered from the given log. Such a process model is a declarative workflow (Section 2.2.3), i.e., expressed in terms of temporal constraints on the activities. Further details on the proposed declarative process model are provided in Section 3.3.

This model serves as a reference for the execution of the next processes, by means of the User Interface module, UI, namely the visualization tool of MAILOFMINE (see Figure 3.3). The users are requested to provide an initial setup, specifying the terms in the vocabulary of the domain of interest, and finally a process model is returned as a result. They could even ignore the output in terms of constraints over activities, and just take advantage of suggestions that the UI suggests her at run-time (Figure 3.2 depicts an example of run-time interaction, with suggestions about the subject to insert, the activity to link, etc.). Such suggestions are elaborated on the basis of the mined workflow, though it can be kept transparent to the user. The rationale behind the realization is to make it as less intrusive as possible. In case she wanted to take control or have a feedback on the structure of the process which the UI follows, a panel for the analysis of activities is provided as well, as depicted in Figure 3.3. The rationale and design of the process visualization is discussed in Section 3.4. Some screenshots of the Web Viz component, realizing the UI module, are in Figures 3.2 and 3.3.

We decided to design the MAILOFMINE tool as a modular architecture so to keep functionalities tied with specific components. This way, the separate steps in the computation can be demanded to specific artifacts which can in turn be substituted or refined with no or few impacts on the rest of the system. For further development, we might change the IR core, from Apache Solr to Terrier (see Section 2.1), because of its higher flexibility (multiple scoring functions are available), but this would not affect, say, neither the UI nor the Miner. All of the software modules were encoded in Java and deployed on an installation of JBoss Application Server, Community Edition, v. 6.1, besides the Web Viz component, written in PHP and running on an Apache HTTP Server, version 2.2.

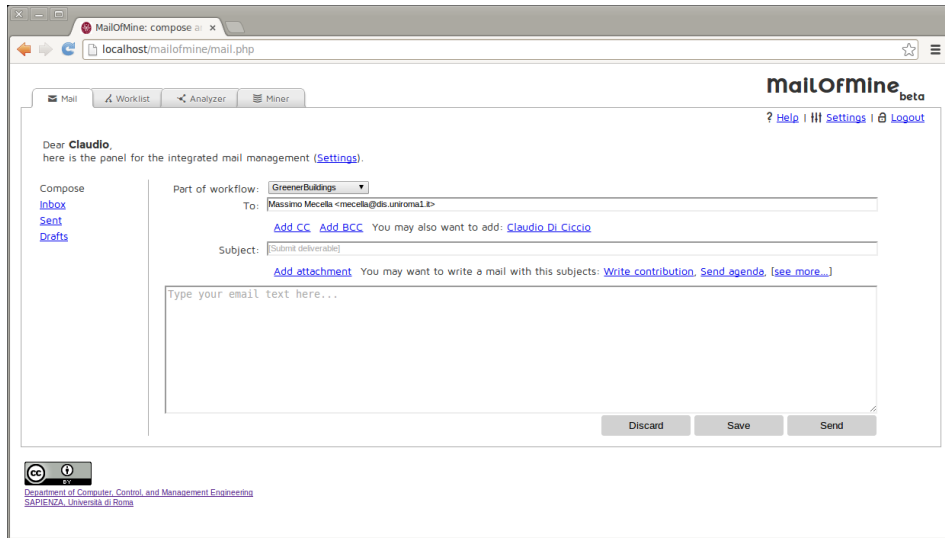


Figure 3.2. Screenshots of MAILOFMINE: the “compose” window

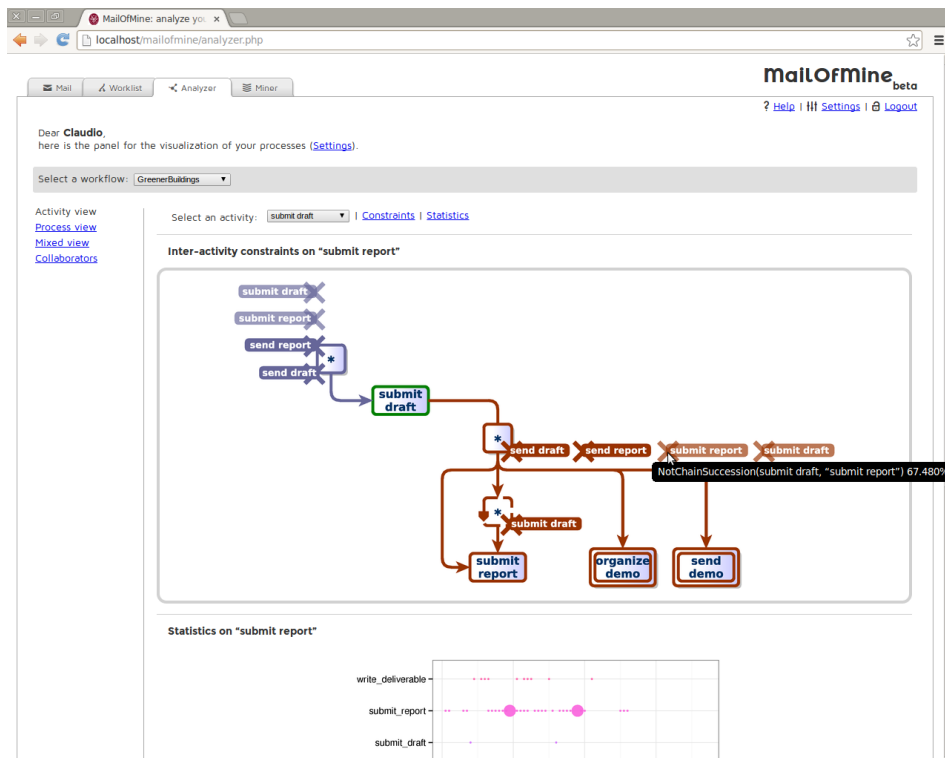


Figure 3.3. Screenshots of MAILOFMINE: the activity analyzer window

3.2 Database

MAILOFMINE stores its own data in a database, managed by a MySQL Server v1.5 installation. In Figure 3.4 we outline it at a conceptual level, as an ER-like simplified diagram. The data-types of the attributes, along with some relationships, are omitted for sake of readability. Here we want to make the reader aware of the core data which MAILOFMINE manages.

In the following, we discuss its structure so to provide a more detailed description of the data of interest for the analyses performed by MAILOFMINE. We store, for every `Message`, its unique `Mail-ID` (as retrieved from its raw POP headers) (`id`, primary key) together with the `dateTime` it was sent at (again, as inferred by reading the POP headers). Each `Message` is linked to textual parts, which are its `Body` and `Subject`. The body we save is either its simple text version, if available, or otherwise its HTML-formatted version, cleaned from tags. We also recall here that both are further re-processed, so to remove the quoted texts and signatures (in bodies), and text such as “Re:”, “FWD:” (in subjects), since we assume that they rarely add any bit of information about the real contents of the email. From the POP headers of the email messages we also extract `Senders` and `Recipients`, to which we associate the email addresses and the identifying name (if provided). For `Recipients`, we specify a `type`, specifying whether they were among the main recipients (“To:”), they received the Copy-Carbon of the email (“CC:”), or they were not among the publicly visible recipients (“BCC:”). Through the application of the techniques described in [11], we cluster them into single `Contacts`, so to reduce the space of possible actors involved in the process.

The email messages are collected from `Archives`, which can be, in turn, IMAP folders, or compressed files containing several `eml` files, etc. The user is also asked to provide, together with the source for email archives, a `Vocabulary`, containing `Words` divided into `Verbs` and `Objects`. Concatenated together in couples, they represent the possible `Expressions` to search for, within the email messages.

Every `Expression` which is actually found in an email at least, is considered a possible `Activity` for the `Process` to mine. Each email `Message` where the `Expression` is discovered is named `Indicium`. The `score` that the Information Retrieval tool in use assigns to the `Indicium` is stored as well.

Once the `Indicia` are gathered, the log is built as follows. For every `Indicium`, an event is considered. The timestamp for the event is the `dateTime` value of the `Message` linked to the `Indicium`. The events are thus temporally ordered. The events are clustered into traces, according to the `Archive` that the related `Messages` belong to – i.e., every `Archive` constitutes the source for a trace. The user is thought to select a collection of `Archives` to mine the `Process` from. The set of traces constitute the log that will be analyzed further for the discovery of the workflow.

Being the `Process` meant to be represented as a declarative model, the `Constraints` constituting such model are mined. They can be either `ExistenceConstraints` or `RelationConstraints`. `ExistenceConstraints` affect single activities only, whereas `RelationConstraints` are connected to couples of activities at a time. We will discuss more in detail about the declarative model of `Processes` in terms of `Constraints` in Section 3.3. The portion of the diagram related to the `Constraint` constitutes a hook to Figure 3.6, in fact. Figure 3.5 summarizes which parts the components described in Section 3.1 work on.

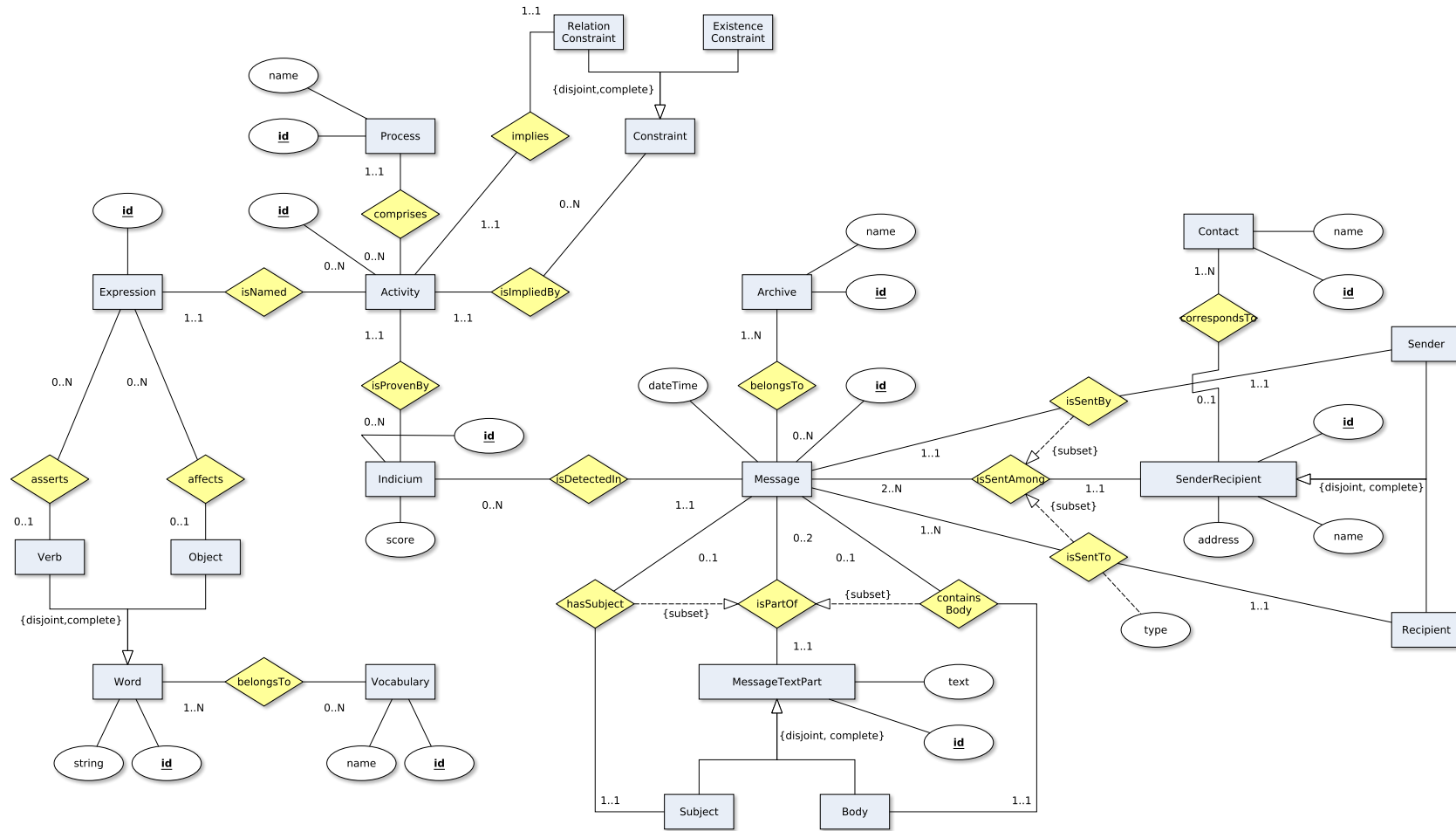


Figure 3.4. The database schema of MAILOfMINE

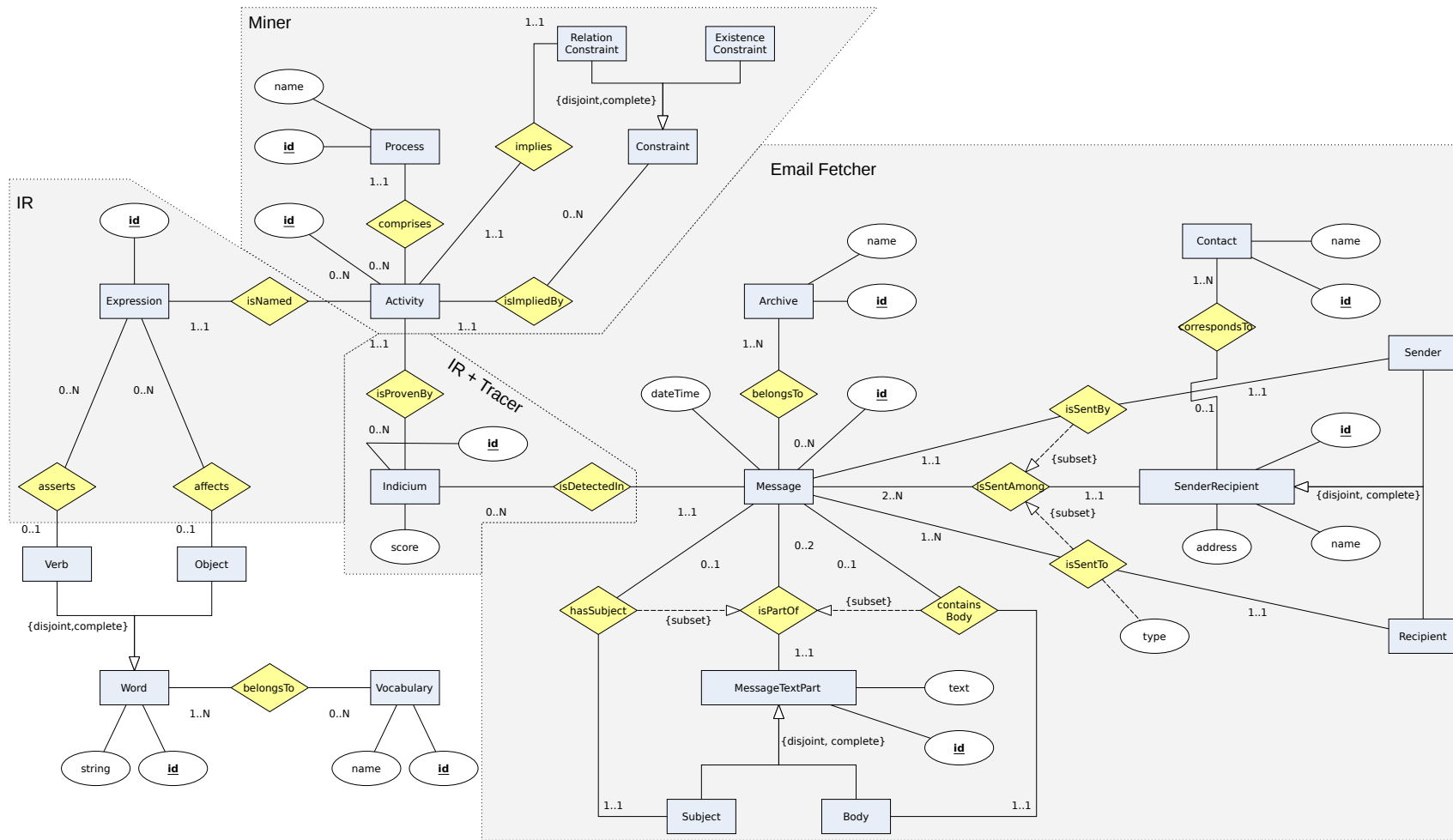


Figure 3.5. The database schema of MAILOFMINE, along with the components managing its stored values

3.3 Specification of declarative workflows as constraints

Here we abstract activities as symbols (e.g., ρ , σ) of an alphabet Σ , appearing in finite strings, which, in turn, represent process traces. We will interchangeably use the terms “activity”, “character” and “symbol”, as well as “trace” and “string”, then. We adopt the subset of Declare taxonomy of constraints for modeling processes, as in [57].

Constraints are temporal rules constraining the execution of activities. E.g., $Response(\rho, \sigma)$ is a constraint on the activities ρ and σ , forcing σ to be executed if the ρ activity was completed before. Such rules are meant to adhere to specific constraint templates. $RespondedExistence$ is the template of $RespondedExistence(\rho, \sigma)$. We further categorize constraint templates into *constraint types*. For instance, $RespondedExistence$ belongs to the *RelationConstraint* type, as far as *MutualRelation* and *NegativeRelation*.

In the following, we briefly summarize the constraint templates that Declare is based upon (see Table 3.1). The reader can find further information in [57, 73] and Section 2.2.3. Figure 3.6 depicts the subsumption hierarchy of Declare constraints. The hierarchy of subsumptions will be exploited in the algorithm explained further in this thesis (Section 4.1.4), in order to prune out redundant constraints.

Declare constraints are always referred to an activity at least, which we call “implying”: if it is executed, the constraint is triggered – vice-versa, if it does not appear in the trace, the constraint has no effect on the trace itself. The $Existence(M, \rho)$ constraint imposes ρ to appear at least M times in the trace. We rename $Existence(1, \rho)$ as $Participation(\rho)$. The $Absence(N, \rho)$ constraint holds if ρ occurs at most $N - 1$ times in the trace. We call $Absence(2, \rho)$ as $Uniqueness(\rho)$. $Init(\rho)$ makes each trace start with ρ .

The aforementioned constraints fall under the type of *ExistenceConstraints*, as they relate to an “implying” activity only. The following are named *RelationConstraints*, since the execution of the implying imposes some conditions on another activity, namely the “implied”.

$RespondedExistence(\rho, \sigma)$ holds if, whenever ρ is read, σ was either already read or going to occur (i.e., no matter if before or afterwards). Instead, $Response(\rho, \sigma)$ enforces it by requiring a σ to appear after ρ , if ρ was read. $Precedence(\rho, \sigma)$ forces σ to occur after ρ as well, but the condition to be verified is that σ was read - namely, you can not have any σ if you did not read a ρ before. $AlternateResponse(\rho, \sigma)$ and $AlternatePrecedence(\rho, \sigma)$ strengthen respectively $Response(\rho, \sigma)$ and $Precedence(\rho, \sigma)$ by stating that *each* ρ (σ) must be followed (preceded) by at least one occurrence of σ (ρ). The “alternation” is in that you can not have two ρ s (σ s) in a row before σ (after ρ). $ChainResponse(\rho, \sigma)$ and $ChainPrecedence(\rho, \sigma)$, in turn, specialize $AlternateResponse(\rho, \sigma)$ and $AlternatePrecedence(\rho, \sigma)$, both declaring that no other symbol can occur between ρ and σ . The difference between the two is in that the former is verified for each occurrence of ρ , the latter for each occurrence of σ . The reader should note that the hierarchy under the *Precedence* constraint template does not inherit the base and implied symbols from the *RespondedExistence* parent; it overrides them both by inverting the two, instead. This is due to the semantics of the constraints themselves.

The *MutualRelation* constraints follow: they are verified iff two

Constraint	Regular expression	Example
Existence constraints		
$Existence(n, a)$	Activity a occurs at least n times in the process instance	
$Participation(a) \equiv Existence(1, a)$	a occurs at least <i>once</i>	bc<u>a</u>ac
$Absence(m + 1, a)$	a occurs at most $n + 1$ times	
$Uniqueness(a) \equiv Absence(2, a)$	a occurs at most <i>once</i> for each trace	bc<u>a</u>c
$Init(a)$	a is the <i>first</i> to occur in each process instance	<u>a</u>ccbbbaba
$End(a)$	a is the <i>last</i> to occur in each process instance	bcaaccbb<u>b</u>aba
Relation constraints		
$RespondedExistence(a, b)$	If a occurs in the process instance, then b occurs as well	bc<u>a</u>accbb<u>b</u>aba
$Response(a, b)$	If a occurs, then b occurs after a	bc<u>a</u>accbb<u>b</u>ab
$AlternateResponse(a, b)$	Each time a occurs, then b occurs afterwards, before a recurs	bc<u>a</u>ccbb<u>b</u>ab
$ChainResponse(a, b)$	Each time a occurs, then b occurs immediately afterwards	bc<u>a</u>bb<u>b</u>ab
$Precedence(a, b)$	b occurs in the process instance only if preceded by a	caacc<u>b</u>bb<u>a</u>ba
$AlternatePrecedence(a, b)$	Each time b occurs, it is preceded by a and no other b can recur in between	caacc<u>b</u>aba
$ChainPrecedence(a, b)$	Each time b occurs, then a occurs immediately beforehand	ca<u>b</u>aba
$CoExistence(a, b)$	If b occurs in the process instance, then a occurs, and viceversa	<u>b</u>caaccbb<u>a</u>ba
$Succession(a, b)$	a occurs if and only if it is followed by b	caaccbb<u>a</u>ab
$AlternateSuccession(a, b)$	a and b if and only if the latter follows the former, and they alternate each other in the trace	ca<u>c</u>cb<u>a</u>ab
$ChainSuccession(a, b)$	a and b occur in the process instance if and only if the latter immediately follows the former	ca<u>b</u>ab
Negative relation constraints		
$NotChainSuccession(a, b)$	a and b occur in the process instance if and only if the latter does not immediately follow the former	bc<u>a</u>accbb<u>b</u>ba
$NotSuccession(a, b)$	a can never occur before b	bc<u>a</u>acca
$NotCoExistence(a, b)$	a and b never occur together	ca<u>a</u>ca

Table 3.1. Semantics of Declare constraints

RespondedExistence (or descendant) constraints (resp., *forward* and *backward*, in Figure 3.6) are satisfied. *CoExistence*(ρ, σ) holds if both *RespondedExistence*(ρ, σ) and *RespondedExistence*(σ, ρ) hold. *Succession*(ρ, σ) is valid if *Response*(ρ, σ) and *Precedence*(ρ, σ) are verified. The same holds with *AlternateSuccession*(ρ, σ), equivalent to the conjunction of *AlternateResponse*(ρ, σ) and *AlternatePrecedence*(ρ, σ), and *ChainSuccession*(ρ, σ), with respect to *ChainResponse*(ρ, σ) and *ChainPrecedence*(ρ, σ).

Finally, we consider *NegativeRelation* constraints: they are satisfied iff the related *MutualRelations* (*negated*, in Figure 3.6) are not. *NotChainSuccession*(ρ, σ) expresses the impossibility for σ to occur immediately after ρ (the opposite of *ChainSuccession*(ρ, σ)). *NotSuccession*(ρ, σ) generalizes the previous by imposing that, if ρ is read, no other σ can be read until the end of the trace (*Succession*(ρ, σ) is the *negated* constraint). *NotCoExistence*(ρ, σ) is even more restrictive: if ρ appears, not any σ can be in the same trace (the contrary of *CoExistence*(ρ, σ)).

In Table 3.2, the semantics of Declare constraints are reported for sake of clarification. Semantics are expressed by means of regular expressions. This translation has been useful to the automated generation of synthetic traces, complying to a given model, which the proposed algorithm could be tested on top of (see Section 5.1). For sake of brevity, there we used the POSIX standard shortcuts. Therefore, in addition to the known Kleene star (*), alternation (|) and concatenation (.) operators, we make use here of (i) the . and [\sim x] shortcuts for respectively matching any character in the alphabet, or any character but x , (ii) the + and ? operators for respectively matching from one to any, or none to one, occurrences of the preceding expression, and (iii) the { n, m } notation, where n (resp. m) denotes the minimum (maximum) number of repetitions of the preceding pattern. Examples are provided so to give a hint on the sense of such constraints. The underlined characters are the “implying” symbols. Strongly emphasized characters are those checked in order to verify the constraint on the string. The translation of constraints into regular expressions of Table 3.2 could be optimized further, although some redundancies are kept in order to provide a better readability and give confidence to the reader in finding similarities between constraints. E.g., you can easily see how *Response*, *AlternateResponse* and *ChainResponse* strengthen along the hierarchy, similarly to *Precedence*, *AlternatePrecedence* and *ChainPrecedence*.

3.3.1 An example

Here we outline a brief example (cf. [41]). We want to model the process of defining an agenda for a research project meeting. The schedule is discussed by email among the participants. We suppose that a final agenda will be committed (“confirm” – n) after that requests for a new proposal (“request” – r), proposals themselves (“propose” – p) and comments (“comment” – c) have been circulated.

The aforementioned tasks and activities are bound to the following constraints (cf. Process Description 1).

If a request is sent, then a proposal is expected to be prepared afterwards (cf. *Response*(r, p)). The presence of comments, in case, is due to a delay in the presentation of an expected proposal, or as a review of the previous. Thus, the presence of c in the trace is constrained to the presence of p (cf. *RespondedExistence*(c, p)). A confirmation is supposed to be mandatorily given after the proposal, and vice-versa

Process Description 1 The example process

Response(r, p)*RespondedExistence*(c, p)*Succession*(p, n)*Participation*(n), *Uniqueness*(n), *End*(n)

any proposal is expected to precede a confirmation (cf. *Succession*(p, n)). We suppose the confirmation to be the *final* activity (cf. *End*(n)). This mandatory task (cf. *Participation*(n)) is not expected to be executed more than once (cf. *Uniqueness*(n)).

As an example, the following traces would be compliant to the given model: pn, pcn, rpcn, rpcpn, rrpcrcrcpcn, rpprpcccrpcn.

Constraint	Regular expression	Example
Existence constraints		
$Existence(n, a)$	$[\hat{a}]^*(a[\hat{a}]^*)\{n, \}+[\hat{a}]^*$	
$Participation(a) \equiv Existence(1, a)$	$[\hat{a}]^*(a[\hat{a}]^*)+[\hat{a}]^*$	<u>b</u> <u>c</u> <u>a</u> <u>a</u> <u>c</u>
$Absence(m + 1, a)$	$[\hat{a}]^*(a[\hat{a}]^*)\{0, m\}+[\hat{a}]^*$	
$Uniqueness(a) \equiv Absence(2, a)$	$[\hat{a}]^*(a)?[\hat{a}]^*$	<u>b</u> <u>c</u> <u>a</u>
$Init(a)$	$a.*$	<u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u> <u>b</u> <u>a</u>
$End(a)$	$.*a$	<u>b</u> <u>c</u> <u>a</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u> <u>b</u> <u>a</u> <u>a</u>
Relation constraints		
$RespondedExistence(a, b)$	$[\hat{a}]^*((a.*b.*) (b.*a.*))*[\hat{a}]^*$	<u>b</u> <u>c</u> <u>a</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u> <u>b</u> <u>a</u>
$Response(a, b)$	$[\hat{a}]^*(a.*b)*[\hat{a}]^*$	<u>b</u> <u>c</u> <u>a</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u> <u>b</u>
$AlternateResponse(a, b)$	$[\hat{a}]^*(a[\hat{a}]^*b[\hat{a}]^*)*[\hat{a}]^*$	<u>b</u> <u>c</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u> <u>b</u>
$ChainResponse(a, b)$	$[\hat{a}]^*(ab[\hat{a}]^*)*[\hat{a}]^*$	<u>b</u> <u>c</u> <u>a</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u> <u>b</u>
$Precedence(a, b)$	$[\hat{b}]^*(a.*b)*[\hat{b}]^*$	<u>c</u> <u>a</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u> <u>b</u> <u>a</u>
$AlternatePrecedence(a, b)$	$[\hat{b}]^*(a[\hat{b}]^*b[\hat{b}]^*)*[\hat{b}]^*$	<u>c</u> <u>a</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>a</u> <u>b</u> <u>a</u>
$ChainPrecedence(a, b)$	$[\hat{b}]^*(ab[\hat{b}]^*)*[\hat{b}]^*$	<u>c</u> <u>a</u> <u>b</u> <u>a</u> <u>b</u> <u>a</u>
$CoExistence(a, b)$	$[\hat{ab}]^*((a.*b.*) (b.*a.*))*[\hat{ab}]^*$	<u>b</u> <u>c</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u> <u>b</u> <u>a</u>
$Succession(a, b)$	$[\hat{ab}]^*(a.*b)*[\hat{ab}]^*$	<u>c</u> <u>a</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u> <u>b</u>
$AlternateSuccession(a, b)$	$[\hat{ab}]^*(a[\hat{ab}]^*b[\hat{ab}]^*)*[\hat{ab}]^*$	<u>c</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>a</u> <u>b</u>
$ChainSuccession(a, b)$	$[\hat{ab}]^*(ab[\hat{ab}]^*)*[\hat{ab}]^*$	<u>c</u> <u>a</u> <u>b</u> <u>a</u> <u>b</u>
Negative relation constraints		
$NotChainSuccession(a, b)$	$[\hat{a}]^*(aa*[\hat{ab}]^*[\hat{a}]^*)*([\hat{a}]^* a)$	<u>b</u> <u>c</u> <u>a</u> <u>a</u> <u>c</u> <u>c</u> <u>b</u> <u>b</u> <u>b</u> <u>b</u> <u>a</u>
$NotSuccession(a, b)$	$[\hat{a}]^*(a[\hat{b}]^*)*[\hat{ab}]^*$	<u>b</u> <u>c</u> <u>a</u> <u>a</u> <u>c</u> <u>a</u>
$NotCoExistence(a, b)$	$[\hat{ab}]^*((a[\hat{b}]^*) (b[\hat{a}]^*))?$	<u>c</u> <u>a</u> <u>a</u> <u>c</u> <u>a</u>

Table 3.2. Semantics of Declare constraints as regular expressions

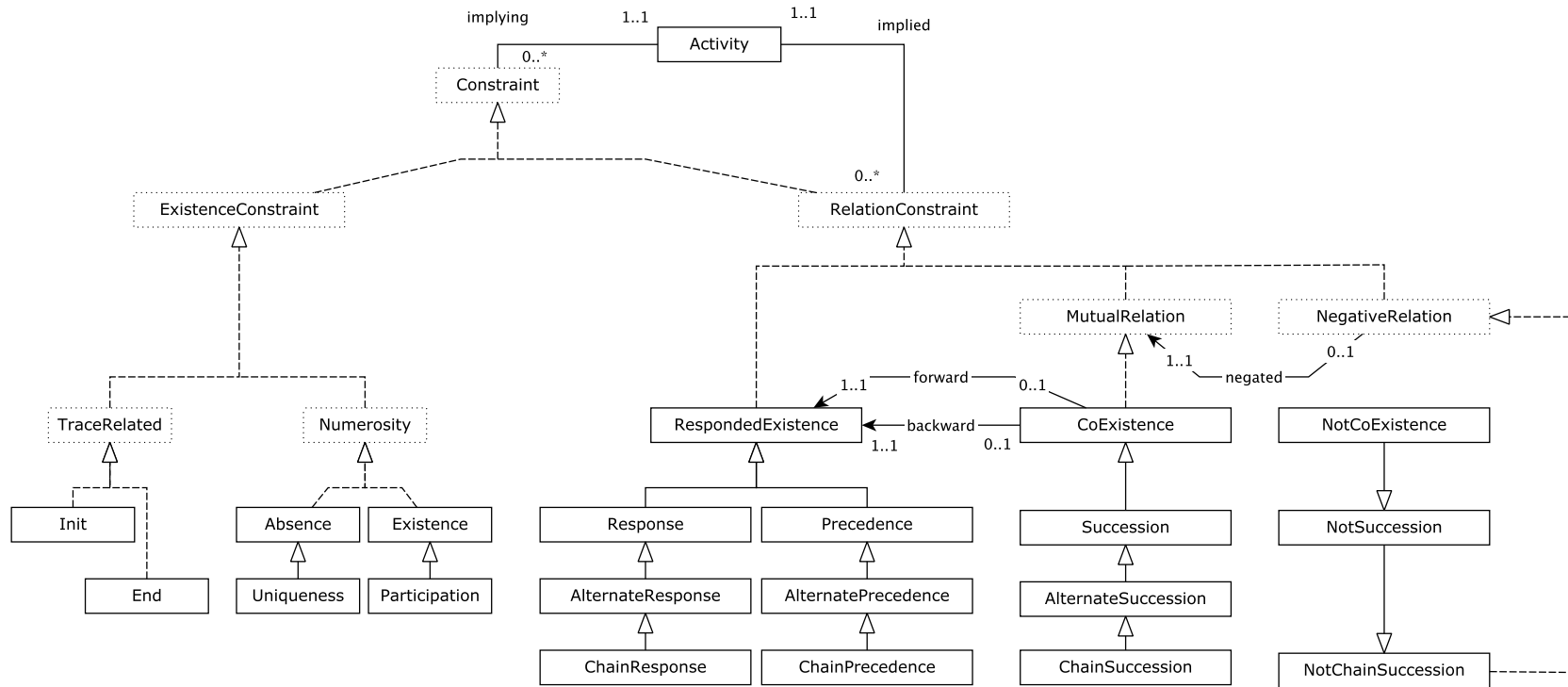


Figure 3.6. The declarative process model’s hierarchy of constraints. Taking into account the UML Class Diagram graphical notations, the Generalization (“is-a”) relationships represent the subsumption between constraint templates. The subsumed is on the tail, the subsuming on the head. The Realization relationships indicate that the constraint template (and the subsumed in the hierarchy) belong to a specific type. Constraint templates are drawn as solid boxes, whereas the constraint types’ boxes are dashed.

3.4 Process visualization

The literature dealing with the representation of processes typically aims at visualizing the process all at once, by means of diagrams that show the complete grid of interconnections among activities. Here we propose a change in the viewpoint. We want to model artful processes as a collection of constraints, through the declarative approach. Being highly flexible, this kind of representation does not necessarily impose a pre-defined strict order on activities, neither explicit nor implicit. For instance, one can state that an activity **a** implies the execution of another activity **b** afterwards (see Section 3.3.1), with no specification provided if **a** is not performed, meaning that **b** can be done or not during the process instance run. In other words, the process schema itself can change according to the things that may have happened before. This is why we do not consider as the best suitable solution adopting a static graph-based global representation alone, on one hand: a local view should work better in conjunction with it. On the other hand, no knowledge worker is expected to be able to read and understand the process by reading the list of regular-expression based constraints: a graphical representation, easy to understand at a first glimpse, must be used.

The process schema and the running processes are respectively modeled through (i) a set of diagrams, representing constraints on workflows (static view: Section 3.4.1) and (ii) an interactive evolutionary graphical representation for the visualization of running instances (dynamic view: Section 3.4.2). Furthermore, we propose two complementary views on constraints: (i) a local view focusing on one activity at a time and (ii) a global view providing a bird-eye sketch of the whole process schema.

For sake of simplicity and readability of figures, we adopt one-character identifiers for activities, as the graphical representation of artful processes presented here is intended to be the core of the user interface for the visualization of the mined processes in MAILOFMINE.

3.4.1 Process schema

The local view

It is very hard to show a process schema all at once and keep it easily readable, due to the high flexibility of the declarative representation. Thus, given that the declarative approach is based on constraints, we collect all of those related to every single activity, i.e., where the activity (e.g., **e**) is either (i) directly implied (e.g., if **d** is done, then **e** must be done), or (ii) directly implying (e.g., if **e** is done, no matter when, **f** was done before or must be done in the future). The *directly* adverb is used due to the need not to make things too much complicated and to follow the idea of having a local view only. For instance, the process is such that if **d** is done, then **e** must be performed; moreover, the enactment of **c** implies that **d** can not be done further. If we look at the constraints directly affecting **e**, the latter rule is not taken into account. In fact, if **c** is not performed, nothing is imposed on **d**. This is an example providing a hint on the rationale: for sake of readability, we want to avoid the confusion coming from too many cross-implications to consider at a time.

The representation of relation constraints is based on three main degrees of freedom, namely (i) time, (ii) implication, (iii) repeatability. The time is considered here as a discrete ordered set of steps the activities can take place in. We ideally

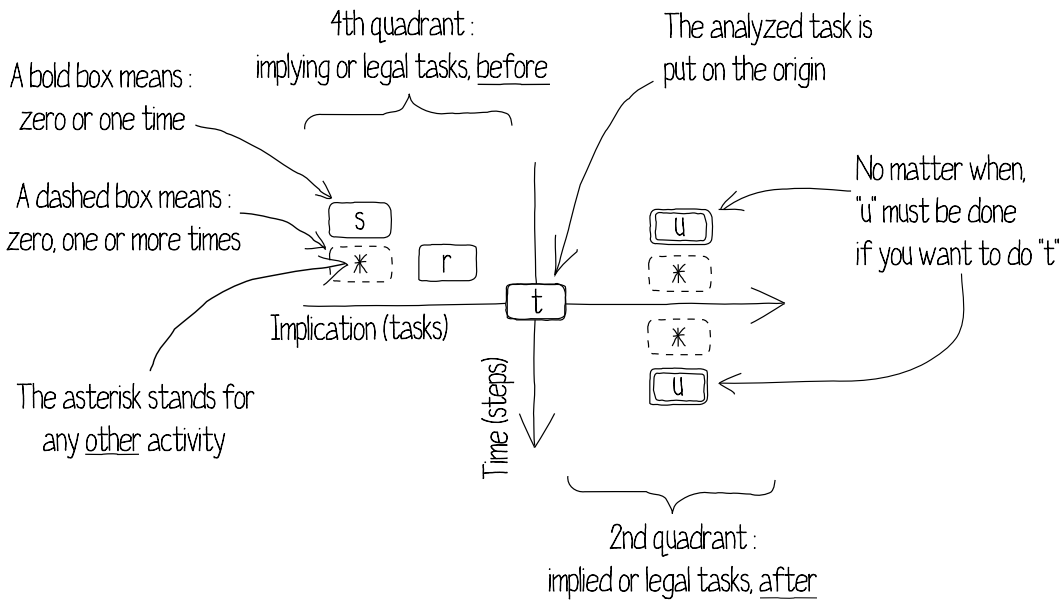


Figure 3.7. The rationale of the local view design

consider each activity as spending a single unit in this conception of time. The notion of implication is based on two values (implying, implied). The repeatability is given by the specification of one among four values, standing for the number of times a activity can be consequently fulfilled: (i) zero, one or more times; (ii) zero, or one time; (iii) exactly once; (iv) zero times.

Our graphical notation represents time and implication as the coordinates of a bidimensional drawing, where time is on the ordinates (see Figure 3.7 for a sketch of the rationale). This ideal y axis divides the plane space into two separate regions: one for each value of the implication dimension (implying, implied), on the abscissae. The x axis divides the plane space into two regions: upwards, what can (or can not) happen *before* the activity is executed, and, downwards, what can (or can not) happen after. On the origin of this chart, inspired to the cartesian coordinate system, we put the activity under examination. The y axis is oriented towards the bottom, in order to follow the reading directionality. For the same reason, the implication relation order flows from the left to the right. Of course, the orientation of the axes can change according to the localization of the software running: e.g., users from Arabic countries might prefer a mirrored version, where the implied activities are on the left, the implying on the right.

The repeatability is expressed by the thickness of the boundaries around the boxes representing activities: dashed for activities that can be done zero, one or more times, solid for zero or one times, double-line for exactly one time. The activity box turns into a cross shape when repeatability is zero. The repeatability is referred to the quadrant the box appears in. For instance, u must appear once either before or after e took place. We recall here that the scope of repeatability, as all of the other degrees of freedom, is not extended to the whole process instance existence, but only for what concerns the time surrounding the single activity under analysis.

For sake of readability, we do not explicitly mention every possible activity the process can be composed of, on the graph. Instead, we render only such activities

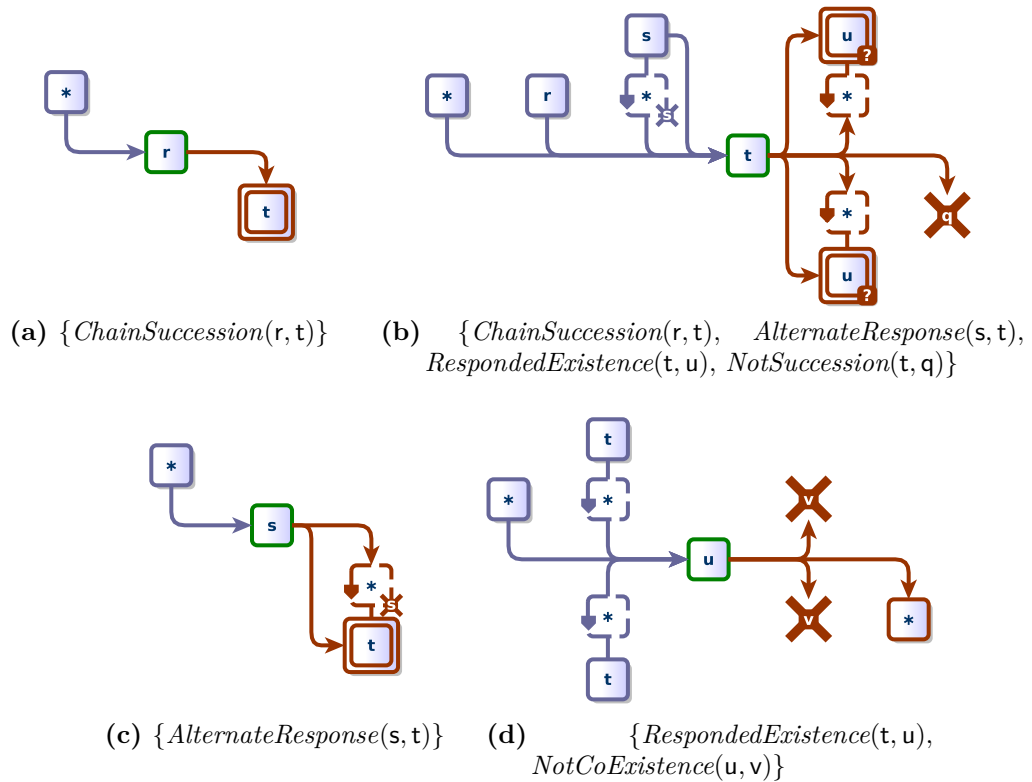


Figure 3.8. The MAILOFMINE local static constraint diagrams

that are interested in focused constraints, so to address the potential problem of too many nodes and arcs connecting one another in a scarcely readable spaghetti-like diagram. Though, visualizing the activities involved in constraints only, might look like a way to force the actor to execute nothing else than the ones that are shown. On the contrary, declarative models allow to do more: roughly speaking, what is not mentioned, is possible. Thus, we make use of a wildcard (*) not intended as “every activity” in the usual all-comprehensive form, but in the typical human conception: “any activity”, where it is understood that the other rules remain valid (e.g., if it is stated that q can not be executed after t , a $*$ after t means “any activity, except q ”). Examples of the diagrams are in Figure 3.8. The constraints depicted here are described in Section 3.3.

The graphical notation is enforced by arrows, easing the user to go across the flow of activities, from the implying before to the implied afterwards. Colors are used for sake of readability and comprehensibility, as additional arrows making a loop on zero-one-more-repeatable activities, though the idea is that such diagrams must be kept easy to be sketched by a pen, as well.

Figure 3.8a shows the only constraint pertaining r in an imaginary process, namely $ChainSuccession(r, t)$. It states that immediately after (i.e., below, on the diagram) the implying r (on the center) you must (double line) execute t , as an implication (on the right). Nothing is directly told about r as an implied activity in a constraint: thus, a $*$ wildcard is put on the left of r . Then, Figure 3.8c focuses

This is the restricted basic graphical syntax used in Figure 3.10a. Indeed, it is not explicitly expressed how strong the constraint is (e.g., whether other activities can be performed between the implying and the implied), in order to tidy the diagram up and provide a fast view of the overall process, without entering in details that are likely better explained through the local views: they can rely, in fact, on dimensions spread on axes the cartesian way, not as in graphs.

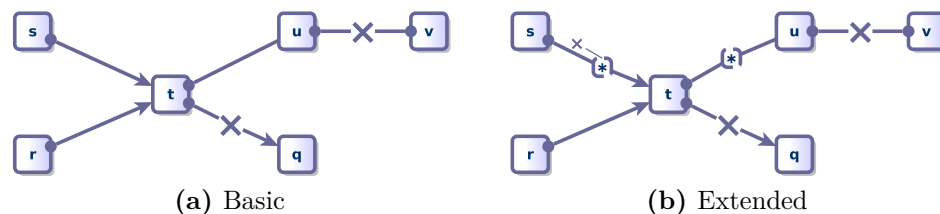


Figure 3.10. The MAILOFMINE global static constraints diagram

Nonetheless, skilled users might want to have a complete vision of the constraints involved, even though it might result in a reduced readability, due to the unavoidable increase of graphical symbols to draw in the diagram. Thus, a richer graphical syntax is needed. Its design rationale is to extend the basic, though keeping coherence with (i) the visual language terms used and (ii) the graph structure. This allows the user to be required of a minimal cognitive effort in order to learn its semantics, on one hand, and lets her toggle between the basic and the extended view. Indeed, only arcs are loaded with new symbols, as depicted on Figure 3.10b: no additional shape nor any change in the graph topology are required.

Both diagrams in Figure 3.10 draw the same set of constraints that were locally represented by example in Figure 3.8.

The global view is inspired to the graphical syntax of [93], though its design focuses on the basic relations (before/after, implying/implied, existence/absence) between activities in a binary constraint.

Coupling this diagram with the local view is useful for avoiding the misunderstanding that could arise by the usage of oriented graphs. Indeed, Finite State Automata, Petri Nets, State Transition Networks, UML Activity Diagrams, Flowcharts, and so forth, all share a common interpretation: roughly speaking, nodes are places to traverse one by one, following a path that respects the direction given by arrows along the arcs. Here, it is not the case: e.g., considering Figure 3.10a, one could intuitively suppose that, done *s*, the next activity is *t*. It is not true: after *s*, *r* or *u* could be performed, even many times, and after some further passages finally *t*.

A GUI sketch

Figure 3.11 draws a prototype of the window showing a local view, on the activity *t*. The additional information regarding the cardinality of the activity, as far as the actors involved and so forth, is located on the bottom of the window. The global view, put on the right, is used as a navigation tool on the process schema. Conversely, at any point in time it will be possible to activate the local view of a activity selected on the global view screen, in order to freely switch from one to another.

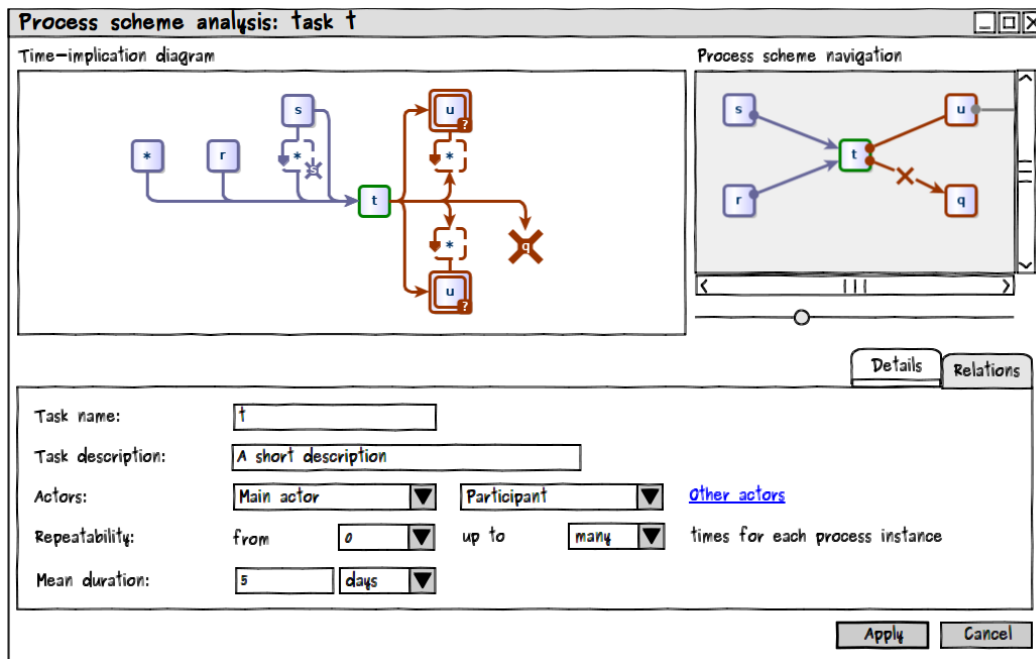


Figure 3.11. The activity's details panel

3.4.2 Running instances

A dynamic view is associated to the static process scheme, for the management of running instances. Such a view is designed to be interactive, i.e., to let the user play with the model, so to control the evolution of the running process. Moreover, she can better learn the constraints mechanism by looking at the process evolving. Indeed, it is based on the same visual notation provided for the visualization of constraints (see Figure 3.9), based in turn on local view diagrams. This choice is made in order to remark the user that global views do not explicitly express the evolution of the system over time, whereas local views do. Figure 3.12 depicts a sample evolution of a process instance.

From a starting step onwards, the user is asked to specify which the next activity to perform is. At each step, the following activities that can be enacted are shown, by means of the same visual language used for static views. After one of them is fired, all the *possible* and *mandatory* following activities are shown. And so forth. We recall here that the recognition of the possible initial activities, as far as the evolution which follows, is a view on the current state of the FSA obtained as the intersection of all the FSA's expressing the constraints in the process scheme. Figure 3.13 is a prototype sketch.

It remarks two main features. The first is that users can adapt the timing in two different ways: either (i) as if every activity lasts a time unit only, ignoring pauses between the preceding and the following, or (ii) showing the actual time consumption for both activities completion and pauses in between. The former is useful for a compact representation, the latter for a realistic snapshot of the time the running process is taking, with the evidence of delays. The second remarked

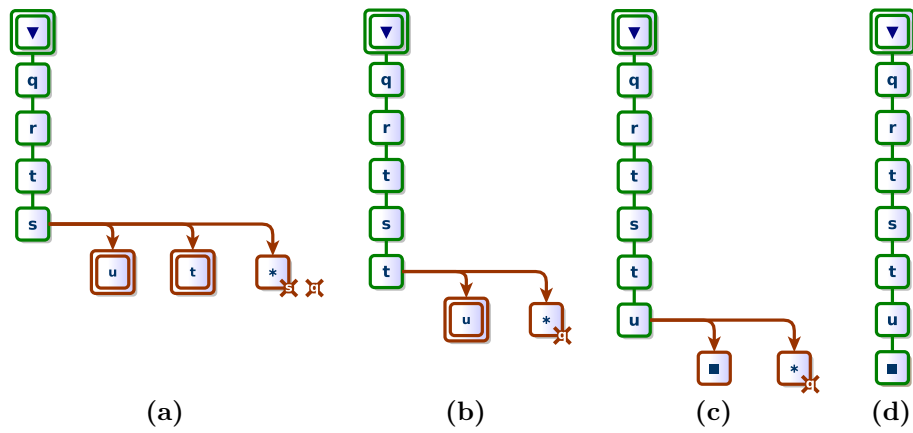


Figure 3.12. The MAILOFMINE dynamic process view

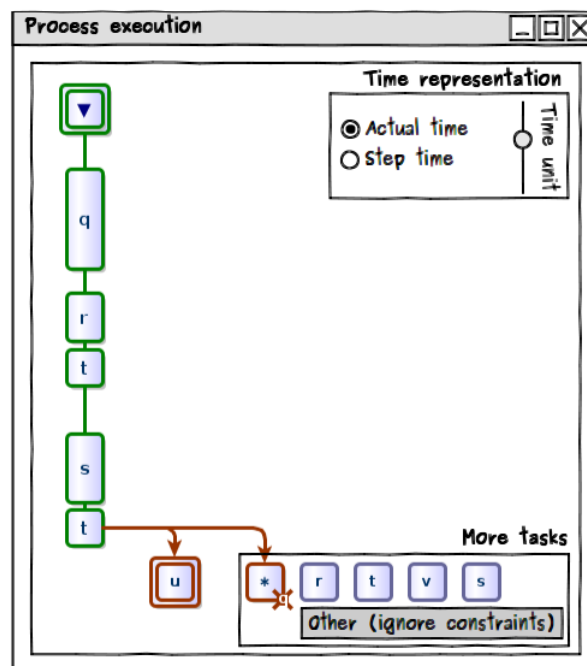


Figure 3.13. The process execution management window

feature is that users may even violate constraints: artful processes are subject to frequent changes, thus imposing a strict respect of constraints could be frustrating for the user who would like to do something unpredicted. This, on the other hand, can be a useful information for the process miner, since it can in turn refine the evolution of the process scheme itself, if a sufficient quantity of deviations from the expected paths are detected. For the next activities to take over, the user will be asked to choose whether she wants to *(i)* delete the violated constraint from the overall process scheme, or *(ii)* proceed as if it was a point deviation only, namely keeping the constraints untouched.

On the Realization of the Process Visualization Approach

The implementation of the tool applying the described visualization approach is still ongoing. Here, though, we outline the rationale behind its realization. Basically, the whole approach is based on FSA. We have already shown how to translate every constraint template in a regular expression, in fact – see Table 3.2. Regular expressions are a succinct way to describe regular grammars, which, in turn, recognize the same languages FSA's accept. Hence, in order to have the complete automaton underlying the declarative model, the conjunction of the set of its constraints is made. Of course, this automaton is likely to become rather unreadable for a human when ten or more activities are considered. Nonetheless, it is the basis to compute the dynamic process view. It actually is a view on the transitions allowed, given the current state, obtained by replaying the history of performed actions on the process FSA.

Also the possible change in the process, caused by choices which do not respect the given constraints, is manageable thanks to the fact that each constraint is equivalent to an FSA: this allows to immediately identify the violated constraint, on one hand, and recalculate the updated process scheme, on the other hand. During the execution of the process instance, in fact, not only the validity of the path on the global intersection FSA is considered: every step is monitored by the evolution of the individual constraint FSA components as well. So, when one or more of them are violated, they can be deleted from the set, on top of which the intersection FSA is computed. Once removed, the FSA is recalculated. Finally, the same history, up to the deviation, is enacted back on the new FSA. The next possible activities to perform are shown accordingly. A thorough analysis of the reconfiguration of declarative process models based on the omission of constraints is provided in the work of Schunselaar et al [82].

Local views are also based on FSA's. Their related FSA's are obtained by intersecting only those constraints which refer to the focused activity, as either the implying or the implied. Then, the resulting automaton is further intersected with another, accepting a limited number of optional transitions, before and after the focused activity. E.g., if the focused activity is *a* and we want to show two possible activities to do before and after, such regular expression is “ $[\hat{a}]0,2a[\hat{a}]0,2$ ”, according to the POSIX notation (see Section 3.3). The look-and-feel given by Figure 3.8 can be obtained by grouping activities and using the *** wildcard when many transitions lead to the same state in the FSA.

An example of this implementation, still bare-bone (i.e., drawing FSA's only), is given in Figure 5.19 and Appendix B.1.

Chapter 4

The Workflow Discovery Algorithm

Here we describe MINERful, namely our algorithm for the discovery of declarative workflows.

Throughout the following Section, we will show its two-steps nature. The first aims at building a knowledge base, named MINERfulKB, where statistical information taken from the log are collected. The second executes several queries to infer which constraints hold in the log. Due to its two-steps nature, two different versions exist of MINERful, sharing the same MINERfulKB – thus, only the querying step changes. The simplest version specifies which, among the possible constraints, hold for every trace in the log. The more complex associates a reliability metric to such assessment, thus confirming not only the constraints which were always verified by the log, but also assessing a so called “Support” for each constraint. Support is a real number ranging from 0 to 1, which gets higher as bigger is the portion of cases in which it is confirmed. An analysis of the way in which such measure is affected by controlled errors is drawn in Section 5.1.1. Moreover, it associates two metrics on more, useful to guess how inferred constraints were of any interest, based on the number of times the activities affected by the constraint actually appeared as events in the log: Confidence Level and Interest Factor. The way they have an impact on a real case study is discussed further in Section 5.2. Here we will also prove that both version are polynomial in the size of the input, w.r.t. their time complexity. The tests providing an experimental proof of this argument are described in Section 5.1.

4.1 MINERful

MINERful is our proposed algorithm for mining declarative constraints out of finite traces of activities (namely, logs). MINERful is based on the concept of *MINERfulKB*: it holds all of the useful information extracted from the given traces and tailored to the further discovery of constraints that might lay behind. The first step of MINERful consists in the construction of that knowledge base (Section 4.1.3), in order to easily infer the declarative model on top of it during the second step (Section 4.1.4). The final output is thus a set of constraints, verified on the the knowledge base. In the following, formal definitions are provided.

4.1.1 MINERfulKB

Let us consider a finite alphabet Σ . The symbols in the alphabet are meant to correspond to the *activities* in a process. Therefore, we will interchangeably use the terms “activity”, “character” and “symbol”. A log is a collection of traces, i.e., a finite sequences of activities. We will consider $T \subset \Sigma^*$ as the log, and thereby interchangeably use the terms “trace” and “string” for denoting every $t \in T$.

Next, we will describe six functions mapping a log $T \subset \Sigma^*$ and either one character $\rho \in \Sigma$ or two characters $\rho, \sigma \in \Sigma$, to integers. Such numbers will be interpreted in MINERful as the result of specific quantitative analyses performed on collections of strings (logs).

We will call ρ and σ , resp., as *pivot* and *searched* characters, due to their sense in the definition of the following functions.

In the examples, we will assign $\Sigma = \{a, b, c\}$ to the alphabet Σ . The pivot ρ will be assigned as **a** and the searched σ as **b**. The assigned log $T \subset \{a, b, c\}^*$ for T will change, case by case.

Definition 13 (MINERful interplay). *A tuple $\mathcal{I} = \langle \Sigma, \delta, \beta^{\rightarrow}, \beta^{\leftarrow} \rangle$, where Σ is the process alphabet (i.e., set of activities’ identifiers) and:*

$\delta(T, \rho, \sigma, d)$ $\delta : \Sigma^* \times \Sigma \times \Sigma \times \mathbb{Z} \rightarrow \mathbb{N}^+$ is the distances function, mapping a distance¹ $d \in \mathbb{N}^+$ between the pivot $\rho \in \Sigma$ and the searched $\sigma \in \Sigma$ to the number of cases they appeared at distance d in the traces of the log T (e.g., $\delta(T, a, b, 2) = 4$ means that we have the evidence of a searched **b** appearing 2 characters after the pivot **a** in 4 cases, given $T = \{cacbcc, acbcacba, acbaaa\}$); we recall that \mathbb{N}^+ is the set of natural integers excluding zero²;

$\beta^{\rightarrow}(T, \rho, \sigma)$ $\beta^{\rightarrow} : \Sigma^* \times \Sigma \times \Sigma \rightarrow \mathbb{N}$ is the in-between onwards appearances function, counting the number of cases in T where, between the occurrence of the pivot ρ and the occurrence of the searched σ , at least one more ρ was read (e.g., if $\beta^{\rightarrow}(T, a, b) = 2$, it means that the pivot **a** appeared 2 times between the preceding occurrence of **a** and the following first occurrence of the searched **b**, as in $T = \{accaacb\}$);

$\beta^{\leftarrow}(T, \rho, \sigma)$ $\beta^{\leftarrow} : \Sigma^* \times \Sigma \times \Sigma \rightarrow \mathbb{N}$ is the in-between backwards appearances function, counting the number of cases where between the occurrence of the pivot ρ and the occurrence of the searched σ , at least one more ρ was read, scanning each string contrariwise in T (e.g., if $\beta^{\leftarrow}(T, a, b) = 3$, it means that the pivot **a** appeared 3 times between the following occurrence of **a** and the preceding last occurrence of the searched **b**, as in $T = \{bcacacabcaa\}$);

Definition 14 (MINERful ownplay). *A tuple $\mathcal{O} = \langle \Sigma, \gamma, \alpha, \omega \rangle$, where Σ is the process alphabet (i.e., set of activities’ identifiers) and:*

$\gamma(T, \rho, n)$ $\gamma : \Sigma^* \times \Sigma \times \mathbb{N} \rightarrow \mathbb{N}$ is the global appearances function, mapping the pivot ρ and a natural number $n \in \mathbb{N}$ to the number of traces of T in

¹The *distance* represents the number of characters between ρ and σ . It is a positive value if σ follows ρ , negative if σ precedes ρ .

²Thus, we do not consider the contemporaneity of events in logs, i.e., no couple of characters is read in the same position.

Function	Extended	Abbreviated
<i>Distances</i>	$\delta(T, \rho, \sigma, d)$	$\delta_{\rho, \sigma}(d)$
<i>In-between onwards appearances</i>	$\beta^{\rightarrow}(T, \rho, \sigma)$	$\beta_{\rho, \sigma}^{\rightarrow}$
<i>In-between backwards appearances</i>	$\beta^{\leftarrow}(T, \rho, \sigma)$	$\beta_{\rho, \sigma}^{\leftarrow}$
<i>Global appearances</i>	$\gamma(T, \rho, n)$	$\gamma_{\rho}(n)$
<i>Initial appearances</i>	$\alpha(T, \rho)$	α_{ρ}
<i>Final appearances</i>	$\omega(T, \rho)$	ω_{ρ}

Table 4.1. Abbreviations for the functions of MINERfulKB

which ρ was read n times (e.g., $\gamma(T, \mathbf{a}, 4) = 2$ means that it happened to the pivot \mathbf{a} to be read exactly four times in two strings only in the log, as in $T = \{\mathbf{aabbabccaabab}, \mathbf{babacaa}\}$);

$\alpha(T, \rho)$ $\alpha : \Sigma^* \times \Sigma \rightarrow \mathbb{N}$ is the initial appearances function, which represents the number of strings where the pivot ρ appeared as the initial symbol (e.g., if $\alpha(T, \mathbf{a}) = 5$, five traces started with \mathbf{a} , as in $T = \{\mathbf{abc}, \mathbf{abbc}, \mathbf{aca}, \mathbf{aa}, \mathbf{a}\}$);

$\omega(T, \rho)$ $\omega : \Sigma^* \times \Sigma \rightarrow \mathbb{N}$ is the final appearances function, which represents the number of strings where the pivot ρ appeared as the last symbol (e.g., if $\omega(T, \mathbf{a}) = 0$, no trace ended with \mathbf{a} , as in $T = \{\mathbf{abc}, \mathbf{abbc}\}$);

Definition 15 (MINERfulKB). A tuple $\mathcal{KB} = \langle \mathcal{I}, \mathcal{O} \rangle$ where $\mathcal{I} = \langle \Sigma, \delta, \beta^{\rightarrow}, \beta^{\leftarrow} \rangle$ is the MINERful interplay, and $\mathcal{O} = \langle \Sigma, \gamma, \alpha, \omega \rangle$ is the MINERful ownplay.

In the definition above, \mathcal{I} and \mathcal{O} are intended to agree on the same process alphabet, Σ .

Notational conventions

For sake of readability, we put input characters as indexes in the subscript of the function symbols. We will remove the explicit reference to the log T , as well. Hence, we will have the abbreviations listed in Table 4.1.

With a slight abuse of notation, we consider $\delta_{\rho, \sigma}(+\infty)$, i.e., $\delta(T, \rho, \sigma, +\infty)$, and $\delta_{\rho, \sigma}(-\infty)$, $\delta(T, \rho, \sigma, -\infty)$, to denote the number of cases in which the searched σ , respectively, did not appear in a string *after* the pivot ρ , and did not appear in a string *before* ρ .

$\delta_{\rho, \sigma}(\pm\infty)$, alias $\delta(T, \rho, \sigma, \pm\infty)$, represents the number of cases in which the searched σ did not appear at all in the strings where ρ occurred, i.e., neither before nor after.

For sake of brevity, here we also define the following function:

$$\Gamma_{\rho} = \sum_{n > 0} \gamma_{\rho}(n) \cdot n$$

It is meant to count the number of appearances of the pivot ρ in the log T .

	$-\infty$	\dots	-5	-4	-3	-2	-1	$\pm\infty$	$+1$	$+2$	$+3$	$+4$	$+5$	\dots	$+\infty$		
$\delta_{a,b}$	2	0	0	0	0	1	1	0	1	2	1	0	0	0	1	$\beta_{a,b}^{\rightarrow} = 1;$	$\beta_{a,b}^{\leftarrow} = 0$
$\delta_{a,c}$	3	0	0	0	0	0	0	0	1	0	0	1	1	0	0	$\beta_{a,c}^{\rightarrow} = 2;$	$\beta_{a,c}^{\leftarrow} = 0$
$\delta_{b,a}$	0	0	0	0	1	2	1	0	1	1	0	0	0	0	0	$\beta_{b,a}^{\rightarrow} = 1;$	$\beta_{b,a}^{\leftarrow} = 1$
$\delta_{b,c}$	2	0	0	0	0	0	0	0	0	1	1	0	0	0	0	$\beta_{b,c}^{\rightarrow} = 1;$	$\beta_{b,c}^{\leftarrow} = 0$
$\delta_{c,a}$	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	$\beta_{c,a}^{\rightarrow} = 0;$	$\beta_{c,a}^{\leftarrow} = 0$
$\delta_{c,b}$	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	$\beta_{c,b}^{\rightarrow} = 0;$	$\beta_{c,b}^{\leftarrow} = 0$

Table 4.2. An example of MINERful interplay, interpreted over aabbac

As an example, let us suppose to interpret the MINERfulKB over a singleton $T = \{\text{aabbac}\}$. Then, for what a, b and c are concerned, \mathcal{I} is shown in Table 4.2, and \mathcal{O} is as it follows:

$$\left\langle \gamma_{\mathbf{a}}(n) = \begin{cases} 1 & n = 3 \\ 0 & n \in \mathbb{N} \setminus \{3\} \end{cases}, \alpha_{\mathbf{a}} = 1, \omega_{\mathbf{a}} = 0 \right\rangle$$

$$\left\langle \gamma_{\mathbf{b}}(n) = \begin{cases} 1 & n = 2 \\ 0 & n \in \mathbb{N} \setminus \{2\} \end{cases}, \alpha_{\mathbf{b}} = 0, \omega_{\mathbf{b}} = 0 \right\rangle$$

$$\left\langle \gamma_{\mathbf{c}}(n) = \begin{cases} 1 & n = 1 \\ 0 & n \in \mathbb{N} \setminus \{1\} \end{cases}, \alpha_{\mathbf{c}} = 0, \omega_{\mathbf{c}} = 1 \right\rangle$$

4.1.2 The algorithm: a bird's eye view

Algorithm 1 presents a bird-eye view of the technique. The different steps will be detailed in the following sections.

Algorithm 1 The MINERful pseudo-code algorithm, in its simplest form (bird's eye view)

```

 $\mathcal{KB} \leftarrow \text{COMPUTEKBONWARDS}(T, \Sigma, \emptyset)$ 
 $\mathcal{KB} \leftarrow \text{COMPUTEKBBACKWARDS}(T, \Sigma, \mathcal{KB})$ 
 $\mathcal{B} \leftarrow \text{DISCOVERCONSTRAINTS}(\mathcal{KB}, \Sigma, |T|)$ 

```

In the following, we will describe how the MINERfulKB is computed (Section 4.1.3). Then, we will show two versions of the DISCOVERCONSTRAINTS procedure: the simplest, explained in Section 4.1.4, returns \mathcal{B} , namely a bag of constraints which will compose the process; the more complex DISCOVERCONSTRAINTS⁺ (see Algorithm 2), explained in Section 4.1.5, returns an extended version of \mathcal{B} , namely \mathcal{B}^+ , where constraints are associated to *Support* as a reliability measure, along with metrics estimating their degree of significance: *Confidence Level* (or *Confidence* for short) and *Interest Factor*.

The MINERfulKB is designed in order to be tailored to the further reasoning for constraints discovery. Thus, the latter step becomes easier and faster, rather than analyzing it directly from the raw data (the collection of strings). At the same time, it must be fast: moving the whole complexity to that step would take no advantage to the overall technique. This first phase of the algorithm is built to be completely on-line, i.e., it refines the MINERfulKB as new strings occur and as new characters in the string are read, with no need to go back on already processed data in the end.

Algorithm 2 The MINERful pseudo-code algorithm, with the computation of reliability and interest metrics (bird-eye watching)

```

 $\mathcal{KB} \leftarrow \text{COMPUTEKBONWARDS}(T, \Sigma, \emptyset)$ 
 $\mathcal{KB} \leftarrow \text{COMPUTEKBBACKWARDS}(T, \Sigma, \mathcal{KB})$ 
 $\mathcal{B}^+ \leftarrow \text{DISCOVERCONSTRAINTS}^+(\mathcal{KB}, \Sigma, |T|)$ 

```

4.1.3 Construction of the MINERfulKB

The input of this algorithm is an alphabet of activities and a log, which are possible assignments for (resp.) Σ and T , referring back to the definitions of Section 4.1.1.

Here we call the input log L and the input process alphabet A . For each activity $a \in A$, a unique identifier is considered. For each trace $l \in L$, a string of unique activities' identifiers is taken into account.

We recall that Σ pertains the interpretation of the MINERful interplay \mathcal{I} and the MINERful ownplay \mathcal{O} , whereas T is the collection of strings passed as a parameter to all of the functions in \mathcal{I} and \mathcal{O} . Therefore, the algorithm computes an interpretation function $(\mathcal{L}, \mathcal{A})$ for the MINERfulKB \mathcal{KB} over L and A , considering the identifiers of the activities in A as the characters of Σ and the sequences of identifiers representing the traces of L as the strings of T . At the end of the run, we have the interpretations for both the MINERful interplay and the MINERful ownplay on the basis of L and A , i.e.,

$$\mathcal{L}, \mathcal{A} \mathcal{KB} = \langle \mathcal{L}, \mathcal{A} \mathcal{I}, \mathcal{L}, \mathcal{A} \mathcal{O} \rangle$$

where

$$\mathcal{L}, \mathcal{A} \mathcal{I} = \langle \mathcal{L}, \mathcal{A} \Sigma, \mathcal{L}, \mathcal{A} \delta, \mathcal{L}, \mathcal{A} \beta^{\rightarrow}, \mathcal{L}, \mathcal{A} \beta^{\leftarrow} \rangle$$

and

$$\mathcal{L}, \mathcal{A} \mathcal{O} = \langle \mathcal{L}, \mathcal{A} \Sigma, \mathcal{L}, \mathcal{A} \gamma, \mathcal{L}, \mathcal{A} \alpha, \mathcal{L}, \mathcal{A} \omega \rangle$$

(see Section 4.1.1).

For the remainder, although, we will omit the \mathcal{L}, \mathcal{A} . notation, thus implicitly referring to the interpreted MINERfulKB, when mentioning \mathcal{KB} and all the related functions, in order to ease the reading of text and formulae.

Explanation of Algorithm 3

Before starting the description of the code in Algorithm 3, we resume here the notation adopted. *Sets* differ from *lists* in that they can not have multiple copies of the same value. Therefore, if, e.g., $X = \{x, y\} \implies X \cup \{x\} = \{x, y\}$, i.e., unions are implicitly meant to be distinct: the reader has to keep this in mind when looking at instructions like $\mathbf{R} := \mathbf{R} \cup \{\sigma\}$ (see line 13 in Algorithm 3). Lists, though, have an explicit positional indexing over the values inserted. Hence, $\vec{p}_\rho[j]$ (see line 27 in Algorithm 3), is pointing at the j -th element in the \vec{p}_ρ list. Strings are considered as lists of characters: thus, $t[i]$ refers to the i -th character in the string t (see line 12 in Algorithm 3), where i ranges from 1 to $|t|$. Lists and strings are provided with a concatenation function, \circ : for instance, the effect of $\vec{p}_\sigma \leftarrow \vec{p}_\sigma \circ \{i\}$ is to add i as the last element in \vec{p}_σ (see line 14 in Algorithm 3). For pointing at a specific element in a map (indexed multi-set), we specify the ‘‘coordinates’’ between couples of brackets, as for a bi-dimensional array: e.g., $\mathbf{N}[r][s]$ is the element in \mathbf{N} corresponding to r and s (see line 7 in Algorithm 3). When pointing at the whole sub-map corresponding to a single character, we insert the target symbol only, as selecting a row in a bi-dimensional array: e.g., $\mathbf{N}[r]$ is the sub-map in \mathbf{N} corresponding to r (see line 41 in Algorithm 3).

In order to ease the reader to distinguish between assignments of temporary variables (like the ones from line 6 to line 10 in Algorithm 3) and the update of the interpretation for the MINERfulKB (see e.g., line 5 in Algorithm 3), we denote the former with $:=$, the latter with \leftarrow .

Algorithm 3 The COMPUTEKBONWARDS procedure's pseudo-code

```

1: procedure COMPUTEKBONWARDS( $T, \Sigma, \mathcal{KB}$ )
2:    $\forall d \in \mathbb{Z} \ \forall \rho \in \Sigma \ \forall \sigma \in \Sigma. \ \delta_{\rho, \sigma}(d) \leftarrow 0$ 
3:    $\forall n \in \mathbb{N}^+ \ \forall \rho \in \Sigma. \ \gamma_{\rho}(x) \leftarrow 0$ 
4:   for all  $t \subseteq T$  do
5:      $\alpha_{t[1]} \leftarrow \alpha_{t[1]} + 1$ 
6:      $\mathbf{R} := \emptyset$    #  $\mathbf{R}$ : set of characters already appeared in  $t$ 
7:      $\forall r, s \in \Sigma. \ \mathbf{N}[r][s] := 0$    #  $\mathbf{N}$ : bi-indexed map, counting the missing  $s$ 's after  $r$ 
8:      $\forall r \in \Sigma. \ \vec{\mathbf{p}}_r := \{\}$    #  $\vec{\mathbf{p}}_r$ : list of indexes where  $r$  appears in  $t$ 
9:      $\forall r, s \in \Sigma. \ \widehat{\mathbf{W}}[r][s] := 0$    #  $\widehat{\mathbf{W}}$ : counts the  $r$ 's repeated before the next  $s$ 
10:     $\forall r, s \in \Sigma. \ \widehat{\mathbf{W}}[r][s] := \perp$    #  $\widehat{\mathbf{W}}$ : flags granting the update of  $\mathbf{W}$ 
11:    for  $i = 1 \rightarrow |t|$  do
12:       $\sigma := t[i]$ 
13:       $\mathbf{R} := \mathbf{R} \cup \{\sigma\}$ 
14:       $\vec{\mathbf{p}}_{\sigma} := \vec{\mathbf{p}}_{\sigma} \circ \{i\}$ 
15:      for all  $\rho \in \mathbf{R}$  do
16:        if  $\rho = \sigma$  then
17:          for all  $s \in \Sigma \setminus \{\rho\}$  do
18:             $\mathbf{N}[\rho][s] := \mathbf{N}[\rho][s] + 1$ 
19:            if  $\widehat{\mathbf{W}}[\rho][s] = \perp$  then
20:               $\widehat{\mathbf{W}}[\rho][s] := \top$ 
21:            else
22:               $\mathbf{W}[\rho][s] := \mathbf{W}[\rho][s] + 1$ 
23:            end if
24:          end for
25:        else
26:          for  $j = 1 \rightarrow |\vec{\mathbf{p}}_{\rho}|$  do
27:             $\delta_{\rho, \sigma}(i - \vec{\mathbf{p}}_{\rho}[j]) \leftarrow \delta_{\rho, \sigma}(i - \vec{\mathbf{p}}_{\rho}[j]) + 1$ 
28:          end for
29:           $\mathbf{N}[\rho][\sigma] := 0$ 
30:          if  $\widehat{\mathbf{W}}[\rho][\sigma] = \top$  then
31:             $b_{\rho, \sigma}^{\rightarrow} \leftarrow b_{\rho, \sigma}^{\rightarrow} + \mathbf{W}[\rho][\sigma]$ 
32:             $\widehat{\mathbf{W}}[\rho][\sigma] := \perp, \mathbf{W}[\rho][\sigma] := 0$ 
33:          end if
34:        end if
35:      end for
36:    end for
37:    for all  $r \in \mathbf{R}$  do
38:      for all  $\bar{s} \in \Sigma \setminus \mathbf{R}$  do
39:         $\delta_{r, \bar{s}}(\pm\infty) \leftarrow \delta_{r, \bar{s}}(\pm\infty) + |\vec{\mathbf{p}}_r|$ 
40:      end for
41:      for all  $\bar{s} \in \Sigma \setminus \{r\}$  do
42:         $\delta_{r, \bar{s}}(+\infty) \leftarrow \delta_{r, \bar{s}}(+\infty) + \mathbf{N}[r][\bar{s}]$ 
43:      end for
44:      if  $|\vec{\mathbf{p}}_r| = 1$  then
45:         $\delta_{r, r}(+\infty) \leftarrow \delta_{r, r}(+\infty) + 1$ 
46:      end if
47:    end for
48:    for all  $s \in \Sigma$  do
49:       $\gamma_s(|\vec{\mathbf{p}}_s|) \leftarrow \gamma_s(|\vec{\mathbf{p}}_s|) + 1$ 
50:    end for
51:     $\omega_{t[|t|]} \leftarrow \omega_{t[|t|]} + 1$ 
52:  end for
53:  return  $\mathcal{KB}$ 
54: end procedure

```

From line 2 to line 3, the interpretations of the γ and δ functions are initialized, supposing that they are constant and equal to 0, whatever the value the variables assume. Then, for each string t in T (line 4), the first character appearing ($t[0]$) is recorded into the related $\alpha_{t[0]}$ as the first (line 5). After the initialization of auxiliary data structures, whose role is briefly explained in-line on the code itself and further in this Section, the analysis of the single characters in the string begins (line 11). First of all, the encountered character σ is added to the set of appeared characters in t , namely \mathbf{R} (if it is not already in – see the discussion on the set union at the beginning of Section 4.1.3). Next, the current index is concatenated (\circ operation) to the list of positions where σ was read in t ($\vec{\mathbf{p}}_\sigma$), at line 14. On line 15 the algorithm starts the computation of interleaving statistics between characters.

For each of the characters already found in the string, ρ , the algorithm proceeds differently, depending on whether an already appeared character is read again ($\rho = \sigma$) or not (line 16).

In the first case, the temporary counter for cases in which s (where s is any other character in Σ but ρ) did not appear anymore after an occurrence of ρ ($\mathbf{N}[r][s]$) is incremented by 1 (line 18). This is due to the fact that such counter will be reset if s appears afterwards (see line 29), and its value is going to be “flushed” to $\delta_{r,\bar{s}}(+\infty)$ at the end of the string t (see line 42). From line 19 to line 23, the algorithm updates the counters for repeated occurrences of ρ before the next occurrence of s : $\widehat{\mathbf{W}}[\rho][s]$ is the flag for incrementing the $\mathbf{W}[\rho][s]$ counter; hence, if it is set to false, it gets true, whereas if it is already true, $\mathbf{W}[\rho][s]$ is incremented by one. This is due to the fact that when the next occurrence of s is found in the string, the value of $\mathbf{W}[\rho][s]$ will be flushed as an increment to $\beta_{\rho,s}^{\rightarrow}$ (see line 31), before $\widehat{\mathbf{W}}[\rho][s]$ and $\mathbf{W}[\rho][s]$ are reset, respectively, to \perp and 0, (line 32).

If the encountered σ differs from ρ in the loop over \mathbf{R} , then the value assumed by $\delta_{\rho,\sigma}$ at the current distance between ρ and σ has to be incremented by 1. Though, we may have not only one position where ρ occurred, but many. Think to `aacccacab...`, for instance: there, the pivot `a` was read at position 1, 2, 7 and 9, and the searched `b` at position 10. Thus, `b` must be recorded to appear at distance 1, 3, 8 and 9 from `a`. Reminding that $\vec{\mathbf{p}}_\rho$ collects all of the indexes where ρ is read (see line 14), this is what happens at line 27, actually – repeated for each position of ρ in $\vec{\mathbf{p}}_\rho$, i.e., inside the loop starting at line 26. This is probably one of the most difficult steps of the algorithm, though it prevents the analysis to be repeated like a transitive closure on each string for each appeared character. As we said at the beginning of Section 4.1.3, the trace analysis for the interpretation of the MINERful interplay had to be local to each occurrence of ρ , but the construction of the MINERfulKB was required not to be too complex: this is the most noticeable example of how we managed both the prerequisites.

The final part of the outermost cycle updates counters on the basis of the previously gathered information. The instruction of line 39 records the number of times in which the read character r occurred in t , but \bar{s} did not: since $\bar{s} \in \Sigma \setminus \mathbf{R}$, it was not read, hence for all of the r 's in the trace ($|\vec{\mathbf{p}}_r|$), a \bar{s} missed. The statement at line 45 is due to the need of recording that if r appeared once and then no more in the string, then $\delta_{r,r}(+\infty)$ must be incremented by one in the interpretation. This is the only case where this operation makes sense. If we had used for $\delta_{r,r}(+\infty)$ the technique adopted at line 39, it would have been meaningless, since a last r always occurs, and no more r are read afterwards (this is the reason why the cycle for

$\langle N, \delta_{.,.}(+\infty) \rangle \setminus \sigma \in t$	a	a	b	b	a	c	
$\langle N[a][b], \delta_{a,b}(+\infty) \rangle$	$\langle 1, - \rangle$	$\langle 2, - \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 1, - \rangle$	$\langle 1, - \rangle$	$\langle 0, +1 \rangle$
$\langle N[a][c], \delta_{a,c}(+\infty) \rangle$	$\langle 1, - \rangle$	$\langle 2, - \rangle$	$\langle 2, - \rangle$	$\langle 2, - \rangle$	$\langle 3, - \rangle$	$\langle 0, - \rangle$	$\langle 0, +0 \rangle$
$\langle N[b][a], \delta_{b,a}(+\infty) \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 1, - \rangle$	$\langle 2, - \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 0, +0 \rangle$
$\langle N[b][c], \delta_{b,c}(+\infty) \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 1, - \rangle$	$\langle 2, - \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 0, +0 \rangle$
$\langle N[c][a], \delta_{c,a}(+\infty) \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 1, - \rangle$	$\langle 0, +1 \rangle$
$\langle N[c][b], \delta_{c,b}(+\infty) \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 0, - \rangle$	$\langle 1, - \rangle$	$\langle 0, +1 \rangle$

Table 4.3. The evolution of N and $\delta_{.,.}(+\infty)$, over the reading of a string $t = \text{aabbac}$

computing the value of $N[r][s]$ is executed for each $s \neq r$ – see line 17). On line 49, the function distributing the number of appearances of each character s in the alphabet Σ over T is updated: the number of occurrences of s in t , namely $|\vec{p}_s|$, is the argument, and the referred value is incremented by 1. $|\vec{p}_s|$ can be 0 as well, if it was never read in t . In the end (line 51), the counter for the appearances as last for the ending character of t ($\omega_{t[|t|]}$) is incremented by 1.

A running example for the computation of the δ and β functions

Since the work of the algorithm on δ and β (thus respectively on N , and on W and \widehat{W}) can lead to some difficulties in the understanding, we explain it through an example. Suppose to have a t string like this:

aabbac.

Taking into account the analysis of **a** only as the pivot ρ , for sake of simplicity, the evolution of N throughout the algorithm is reported on the following Table 4.3, as W and \widehat{W} evolve as on Table 4.4.

$\langle \widehat{w}, w, \beta_{\cdot, \cdot}^{\rightarrow} \rangle \setminus \sigma \in t$	a	a	b	b	a	c
$\langle \widehat{w[a][b]}, w[a][b], \beta_{a,b}^{\rightarrow} \rangle$	$\langle \langle \top, 0 \rangle, - \rangle$	$\langle \langle \top, 1 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, +1 \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \top, 0 \rangle, - \rangle$	$\langle \langle \top, 0 \rangle, - \rangle$
$\langle \widehat{w[a][c]}, w[a][c], \beta_{a,c}^{\rightarrow} \rangle$	$\langle \langle \top, 0 \rangle, - \rangle$	$\langle \langle \top, 1 \rangle, - \rangle$	$\langle \langle \top, 1 \rangle, - \rangle$	$\langle \langle \top, 1 \rangle, - \rangle$	$\langle \langle \top, 2 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, +2 \rangle$
$\langle \widehat{w[b][a]}, w[b][a], \beta_{b,a}^{\rightarrow} \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \top, 0 \rangle, - \rangle$	$\langle \langle \top, 1 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, +1 \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$
$\langle \widehat{w[b][c]}, w[b][c], \beta_{b,c}^{\rightarrow} \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \top, 0 \rangle, - \rangle$	$\langle \langle \top, 1 \rangle, - \rangle$	$\langle \langle \top, 1 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, +1 \rangle$
$\langle \widehat{w[c][a]}, w[c][a], \beta_{c,a}^{\rightarrow} \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \top, 0 \rangle, - \rangle$
$\langle \widehat{w[c][b]}, w[c][b], \beta_{c,b}^{\rightarrow} \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \perp, 0 \rangle, - \rangle$	$\langle \langle \top, 0 \rangle, - \rangle$

Table 4.4. The evolution of \widehat{w} , w , and $\beta_{\cdot, \cdot}^{\rightarrow}$ over the reading of a string $t = aabbac$

On the computeKBBackwards procedure.

This algorithm is called twice: (i) first, for reading strings onwards (COMPUTEKBONWARDS), i.e. from left to the right (according to the Western Latin standard); (ii) then, backwards (COMPUTEKBBACKWARDS).

Here we reported the pseudo-algorithm of COMPUTEKBONWARDS only, since the latter differs to the former in few details. The only differences are in that COMPUTEKBBACKWARDS:

- does not update either the γ , nor the α nor the ω functions (namely, it does not contribute to give an interpretation to the MINERful ownplay, being this task already fulfilled by COMPUTEKBONWARDS);
- does not update the δ function for 0 values (since COMPUTEKBBACKWARDS already detected characters never appeared in the string, if any);
- reverses the sign of i , the counter of the current index in the string (namely, it is initialized with -1 and proceeds being decremented by 1 at each step);
- updates the δ function for $-\infty$ values, instead of $+\infty$, whenever the same conditions of lines 41 and 44 in Algorithm 3 are verified.

Discussion on the complexity.

In the following, we discuss the complexity of Algorithm 3.

Lemma 1. *The procedure for building the knowledge base of the MINERful is (i) linear time w.r.t. the number of strings in the testbed, (ii) quadratic time w.r.t. the size of strings in the testbed, (iii) quadratic time w.r.t. the size of the alphabet; therefore, the complexity is $O(|T| \cdot |t_{max}|^2 \cdot |\Sigma|^2)$.*

Proof. The outermost cycle (line 4) is repeated exactly $|T|$ times, the following inner cycle (line 11) is executed $|t|$ times for each $t \in T$. In the worst case, i.e., assuming that each string is as long as the longest, it loops $|t_{max}|$ times, where $|t_{max}| = \max_{t \in T} |t|$. Actually, such couple of loops let the instructions be repeated exactly $\sum_{t \in T} |t|$ times, where T is likely to be the most significant part the input, in terms of its size. At line 15, we have a cycle whose number of repetitions grows as new characters are found in the analyzed string. The number of loops depends on the size of the string $|t|$ and on the size of the alphabet Σ at the same time. In fact, we might assume that each character read was not found before in the string. So, as soon as a new character is read, you have one loop more. You might say that, hence, the instructions in the block are executed $1 + 2 + 3 + \dots + |t|$ times as the cursor in the string moves on. If it was so, loops starting at line 11 and line 15) had run at most

$$\frac{|t| \times (|t| + 1)}{2}$$

times, as in the formula for counting the sum of the first $|t|$ natural numbers. Although, the maximum amount of “new” characters is bounded by the characters you can actually have. Therefore, assuming the worst case, i.e., all of the characters of Σ in every string, it runs at most $1 + 2 + 3 + \dots + |\Sigma|$ times. Then, we have

to subtract the number of loops that are not executed due to the limitation of the alphabet size (if the alphabet size is smaller than the size of the strings, which is likely). Let:

$$\Delta_{|t|,|\Sigma|} = |t| - |\Sigma|$$

Thus, the number of loops is equal to:

$$\frac{|t| \times (|t| + 1)}{2} - \frac{\Delta_{|t|,|\Sigma|} \times (\Delta_{|t|,|\Sigma|} + 1)}{2} \cdot \Theta(\Delta_{|t|,|\Sigma|} - 1)$$

where $\Theta(x)$ is the Heaviside step function (equal to 0, and thus deleting the second term in the subtraction if $\Delta_{|t|,|\Sigma|}$, i.e., if $|t| < |\Sigma|$, otherwise equal to 1). If we suppose that $|t| > |\Sigma| + 1$, then we can simplify terms of the multiplications and subtractions, up to

$$\frac{2|\Sigma||t| - |\Sigma|^2 + |\Sigma|}{2} = \frac{2\Delta_{|t|,|\Sigma|}|\Sigma| + |\Sigma|}{2} \leq |\Sigma||t|$$

Depending on the condition at line 16, the algorithm enters one of the two innermost loops, one starting at line 17, the other at line 26.

The first is executed exactly $|\Sigma| - 1$ times, no matter the outer cycles. The second, instead, is such that the more repetitions of the same character in the string we had, the more it loops. If we had strings composed by concatenations of the same character (the worst case for such cycle) after a prefix comprising the whole alphabet (the worst case for the outer cycle), this would lead to $|t|$ loops, asymptotically.

The instructions in the bodies of the loops are readings and writings in memory³ so they do not add any relevant degree of complexity to the algorithm.

Summing up this computation analysis, we have that the worst-case complexity of the algorithm is

$$O\left(\underbrace{|T|}_{\text{loop at 4}} \underbrace{|t_{max}||\Sigma|}_{\text{loops at 11 and 15}} \left(\underbrace{|\Sigma|}_{\text{loop at 17}} + \underbrace{|t_{max}|}_{\text{loop at 26}}\right)\right)$$

□

4.1.4 Discovery of constraints

Artful processes are represented by means of a set of constraints, imposing the rules that each process instance must follow, whatever the execution trace is. The set of mined constraints are those described in Table 3.2. Here, we express such constraints like predicates over the MINERfulKB, which are easily transposed into instructions for a verification algorithm. A refined version of the discovery is discussed in the following Section 4.1.5, where constraints are associated to a Support, i.e., a value ranging from 0 to 1 assessing the probability of a constraint to hold in the discovered process, and the relevance-assessing metrics of Confidence Level and Interest Factor.

³The reader might ask the opportunity to analyze the complexity of searching the datum to overwrite in the temporary data structures, such as N . Although, considering that (i) we can exploit the alphanumeric ordering function for ordering the couples of characters, and (ii) the alphabet of characters is known a priori, we can easily make use of a hashing function, so that reaching the datum and overwriting it is $O(1)$.

Constraints as predicates

Here we provide a list of predicates assessing whether a constraint holds or not in the MINERfulKB. Actually, a hierarchy between constraints exists (see Figure 3.6): the conjunctions inside, e.g., $ChainPrecedence(\rho, \sigma)$, involving $AlternatePrecedence(\rho, \sigma)$, is explicitly mentioned on purpose. This way, the reader can have an immediate evidence of the fact that, e.g., once $ChainPrecedence(\rho, \sigma)$ is known to hold, $AlternatePrecedence(\rho, \sigma)$, and recursively $Precedence(\rho, \sigma)$ as well, hold too.

Existence constraints

$$\begin{aligned} Participation(r) &\equiv Existence(1, r) \\ &\equiv \left(\min_{\langle o, p \rangle \in \gamma_r | p > 0} o > 0 \right) \end{aligned} \quad (4.1)$$

Each string has at least 1 occurrences of r in.

$$\begin{aligned} Uniqueness(r) &\equiv Absence(2, r) \\ &\equiv \left(\max_{\langle o, p \rangle \in \gamma_r | p > 0} o \leq 2 \right) \end{aligned} \quad (4.2)$$

There is no string with more than 1 occurrence of r in.

$$Init(r) \equiv (|T| \leq \alpha_r) \quad (4.3)$$

Every string starts with r .

$$End(r) \equiv (|T| \leq \omega_r) \quad (4.4)$$

Every string ends with r .

Rather than giving the exact number of times a task can be done (in a range from the lower to the upper), so to specify that all of the $Existence(n, \rho)$ constraints hold, for n ranging from 0 to $\min_{\gamma_\rho(n) > 0} n$ (and dually consider $Absence(m + 1, \rho)$ valid for each m from $\max_{\gamma_\rho(m) > 0} m$ onwards), we preferred to introduce a looser couple of constraints, stating whether a task ρ must be executed ($Participation(\rho)$) or not, and whether it must not be done more than once ($Uniqueness(\rho)$). We believe that providing the minimum and the maximum for ranges would have been for artful processes too overfitting, when mined, or too restrictive, when enacted.

Relation constraints

$$RespondedExistence(\rho, \sigma) \equiv \neg(\delta_{\rho, \sigma}(\pm\infty) > 0) \quad (4.5)$$

There is no string such that σ was not read if ρ was.

$$\begin{aligned} Response(\rho, \sigma) &\equiv RespondedExistence(\rho, \sigma) \\ &\quad \wedge \neg(\delta_{\rho, \sigma}(+\infty) > 0) \end{aligned} \quad (4.6)$$

There is no string such that σ does not succeed ρ .

$$\begin{aligned} \textit{AlternateResponse}(\rho, \sigma) &\equiv \textit{Response}(\rho, \sigma) \\ &\wedge \beta_{\rho, \sigma}^{\rightarrow} = 0 \end{aligned} \quad (4.7)$$

There is no string such that ρ appears again before the subsequent σ .

$$\begin{aligned} \textit{ChainResponse}(\rho, \sigma) &\equiv \textit{AlternateResponse}(\rho, \sigma) \\ &\wedge \delta_{\rho, \sigma}(1) \geq \Gamma_{\rho} \end{aligned} \quad (4.8)$$

Each time you have an occurrence of ρ , the total amount of which is given by Γ_{ρ} , there is always a new σ immediately following (i.e., at a distance equal to 1).

Dually, we have the following *Precedence*-based constraints.

$$\begin{aligned} \textit{Precedence}(\rho, \sigma) &\equiv \textit{Response}(\rho, \sigma) \\ &\wedge \neg(\delta_{\rho, \sigma}(-\infty) > 0) \end{aligned} \quad (4.9)$$

There was no string such that σ did not precede ρ .

$$\begin{aligned} \textit{AlternatePrecedence}(\rho, \sigma) &\equiv \textit{Precedence}(\rho, \sigma) \\ &\wedge \beta_{\rho, \sigma}^{\leftarrow} = 0 \end{aligned} \quad (4.10)$$

There was no string such that ρ appeared again before the preceding σ .

$$\begin{aligned} \textit{ChainPrecedence}(\rho, \sigma) &\equiv \textit{AlternatePrecedence}(\rho, \sigma) \\ &\wedge \delta_{\rho, \sigma}(-1) \geq \Gamma_{\sigma} \end{aligned} \quad (4.11)$$

Each time you have an occurrence of σ , the total amount of which is computed as Γ_{σ} , there is always a new ρ immediately preceding (i.e., at a distance equal to -1).

The next formulae follow by the definition of *MutualRelation* constraints (see Section 3.3).

$$\begin{aligned} \textit{CoExistence}(\rho, \sigma) &\equiv \textit{RespondedExistence}(\rho, \sigma) \\ &\wedge \textit{RespondedExistence}(\sigma, \rho) \end{aligned} \quad (4.12)$$

$$\begin{aligned} \textit{Succession}(\rho, \sigma) &\equiv \textit{Response}(\rho, \sigma) \\ &\wedge \textit{Precedence}(\rho, \sigma) \end{aligned} \quad (4.13)$$

$$\begin{aligned} \textit{AlternateSuccession}(\rho, \sigma) &\equiv \textit{AlternateResponse}(\rho, \sigma) \\ &\wedge \textit{AlternatePrecedence}(\rho, \sigma) \end{aligned} \quad (4.14)$$

$$\begin{aligned} \textit{ChainSuccession}(\rho, \sigma) &\equiv \textit{ChainResponse}(\rho, \sigma) \\ &\wedge \textit{ChainPrecedence}(\rho, \sigma) \end{aligned} \quad (4.15)$$

Negative relation constraints

$$\text{NotChainSuccession}(\rho, \sigma) \equiv \neg(\delta_{\rho, \sigma}(1) > 0) \quad (4.16)$$

It never happens that, after ρ , σ follows unless you have at least another character in the middle (i.e., σ never appears at distance 1 from ρ).

$$\begin{aligned} \text{NotSuccession}(\rho, \sigma) &\equiv \text{NotChainSuccession}(\rho, \sigma) \\ &\wedge \Gamma_{\rho} \leq \delta_{\rho, \sigma}(+\infty) \end{aligned} \quad (4.17)$$

It never happens that, after ρ , σ follows.

$$\begin{aligned} \text{NotCoExistence}(\rho, \sigma) &\equiv \text{NotSuccession}(\rho, \sigma) \\ &\wedge \Gamma_{\rho} \leq \delta_{\rho, \sigma}(\pm\infty) \end{aligned} \quad (4.18)$$

It never happens that, if ρ is in a string, σ appears in the same one, neither before nor afterwards.

The algorithm

In the pseudo-code of the algorithm for guessing the relation constraints (Algorithms 6, 8, 7), we assume to rely on procedures which share the same name of the aforementioned predicates, explicated from Formula 4.1 to Formula 4.18. Their input is MINERfulKB, \mathcal{KB} , interpreted over the given log and alphabet (as in Section 4.1.3, see Algorithm 3), the searched ρ , and optionally the pivot σ (if the procedure is verifying a Relation Constraint). Each procedure returns the boolean *true* value if and only if the conditions given in the homonim Formula holds, *false* otherwise. Hence, for instance, we have that the RESPONDEDEXISTENCE and RESPONSE procedure behaves like what drawn by the Algorithm 4 and Algorithm 5's pseudocodes, respectively.

Algorithm 4 The pseudo-code of the RESPONDEDEXISTENCE procedure

```

1: procedure RESPONDEDEXISTENCE( $\mathcal{KB}, \rho, \sigma$ )
2:   if  $\neg(\delta_{\rho, \sigma}(\pm\infty) > 0)$  then    # See Formula 4.5
3:     return true
4:   else
5:     return false
6:   end if
7: end procedure

```

Algorithm 5 The pseudo-code of the RESPONSE procedure

```

1: procedure RESPONSE( $\mathcal{KB}, \rho, \sigma$ )
2:   if  $\text{RESPONDEDEXISTENCE}(\mathcal{KB}, \rho, \sigma) \wedge \neg(\delta_{\rho, \sigma}(+\infty) > 0)$  then    # See Formula 4.6
3:     return true
4:   else
5:     return false
6:   end if
7: end procedure

```

With a slight conceptual modification w.r.t. COMPUTEKBONWARDS, then, we suppose the algorithm to fill a bag of predefined constants (each symbol corresponding

Constraint	Symbol
Existence constraints	
<i>Participation</i> (ρ)	
<i>Uniqueness</i> (ρ)	\top_{ρ}^{1-}
<i>Init</i> (ρ)	\top_{ρ}^i
<i>End</i> (ρ)	\top_{ρ}^e
Relation constraints	
<i>RespondedExistence</i> (ρ, σ)	$\top_{\rho, \sigma}$
<i>Response</i> (ρ, σ)	$\top_{\rho, \sigma}^{\rightarrow}$
<i>AlternateResponse</i> (ρ, σ)	$\top_{\rho, \sigma}^{\rightarrow\rightarrow}$
<i>ChainResponse</i> (ρ, σ)	$\top_{\rho, \sigma}^{\Rightarrow}$
<i>Precedence</i> (ρ, σ)	$\top_{\rho, \sigma}^{\leftarrow}$
<i>AlternatePrecedence</i> (ρ, σ)	$\top_{\rho, \sigma}^{\leftarrow\leftarrow}$
<i>ChainPrecedence</i> (ρ, σ)	$\top_{\rho, \sigma}^{\Leftarrow}$
<i>CoExistence</i> (ρ, σ)	\top_{σ}^{ρ}
<i>Succession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\leftrightarrow}$
<i>AlternateSuccession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\leftrightarrow\leftrightarrow}$
<i>ChainSuccession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\Leftrightarrow}$
Negative relation constraints	
<i>NotCoExistence</i> (ρ, σ)	\perp_{σ}^{ρ}
<i>NotSuccession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\nleftrightarrow}$
<i>NotChainSuccession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\nLeftrightarrow}$

Table 4.5. Symbols expressing the validity of constraints

to the validity of a constraint for the given set of traces T), rather than giving an interpretation to each predicate (see Table 4.5 for a reference). As the reader can see in Algorithms 6, 8, 7 and the further discussion, such constants will be added to the bag avoiding redundancies. The user who wants to understand the discovered artful process is not interested in reading trivial deductions. For instance, it is enough to say that $ChainPrecedence(\rho, \sigma)$ holds, rather than explicitly returning as verified constraints $ChainPrecedence(\rho, \sigma)$, $AlternatePrecedence(\rho, \sigma)$, $Precedence(\rho, \sigma)$ and $RespondedExistence(\rho, \sigma)$ (where the last three are directly implied by the first). In this example, e.g., reporting the successful verification of all the subsumed constraints would add no bit of information and rather make the result far less readable – which is definitely to avoid, in our case, being knowledge workers the target of our approach, i.e., people with a little amount of time to dedicate to the process analysis. For the same reason, characters never appeared in the testbed are not involved neither in existence constraints nor in relation constraints related to them as implying (see line 4 in Algorithm 6).

This is the rationale underlying the nested *if* structure of the Algorithms.

Discussion on the complexity In the following we discuss the complexity of the concurrent execution of Algorithms 6, 8 and 7.

Lemma 2. *The procedure for discovering the constraints of processes out of the MINERful knowledge base is (i) quadratic time w.r.t. the size of the alphabet, (ii) linear in the number of constraint templates, which is fixed and equal to 18 (thus constant); therefore, the complexity is $O(|\Sigma_T|^2)$.*

Proof. We have two nested cycles, both for ρ ranging over the characters of the alphabet Σ (see lines 3 and 3 in Algorithm 6), thus both looping for $|\Sigma|$, at most, due to the presence of the check at line 4 in Algorithm 6), so to analyze every possible couple. The nested **if** statements check whether a constraint holds or not. At most, it means that, for each couple of characters ρ and σ , you have up to 14 checks (for the innermost loop). The outermost loop calls up to 4 procedures for checking existence constraints. \square

4.1.5 Discovery of constraints and their metrics

Declarative processes are modeled by a set of constraints, imposing the rules that each process instance must follow, whatever the execution trace is. The set of mined constraints is listed in Table 4.6. Table 4.6 shows the functions used in order to compute the Support for the constraints, with respect to the MINERfulKB. Here we call “Support” the value, ranging from 0 to 1, that represents the normalized fraction of cases in which the constraint is verified, over the set of traces T . Such functions are all based on mathematical operations performed on data coming from the MINERfulKB only, plus the information about the size of T , i.e., how many strings were read. In order to ease the readability, we omit the \mathcal{KB} and $|T|$ parameters from the list of each function, since they can be considered as a common shared knowledge.

Support is a metric adopted in [56] as well, but with a slight difference in the computation. There, it corresponds to the number of traces where the constraint is non-vacuously satisfied, w.r.t. the number of traces in the log. Here, instead, we

Algorithm 6 The DISCOVERCONSTRAINTS procedure's pseudo-code

```

1: procedure DISCOVERCONSTRAINTS( $\mathcal{KB}, \Sigma, |T|$ )
2:    $\mathcal{B} \leftarrow \emptyset$    # Inizialization of the bag of constraints
3:   for all  $\rho \in \Sigma$  do
4:     if ( $\max_{(o,p) \in \gamma_r | p > 0} \sigma \leq 0$ ) then
5:        $\mathcal{B} \leftarrow$  DISCOVEREXISTENCECONSTRAINTS( $\rho, \mathcal{B}, \mathcal{KB}$ )
6:       for all  $\sigma \in \Sigma$  do
7:         if RESPONDEDEXISTENCE( $\mathcal{KB}, \rho, \sigma$ ) then
8:           if RESPONSE( $\mathcal{KB}, \rho, \sigma$ ) then
9:             if ALTERNATERESPONSE( $\mathcal{KB}, \rho, \sigma$ ) then
10:              if CHAINRESPONSE( $\mathcal{KB}, \rho, \sigma$ ) then
11:                if CHAINPRECEDENCE( $\mathcal{KB}, \rho, \sigma$ ) then
12:                   $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\leftrightarrow}\}$ 
13:                else
14:                   $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\Rightarrow}\}$ 
15:                end if
16:              else
17:                if ALTERNATEPRECEDENCE( $\mathcal{KB}, \rho, \sigma$ ) then
18:                   $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\leftrightarrow}\}$ 
19:                else
20:                   $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\Rightarrow}\}$ 
21:                end if
22:              end if
23:            else
24:              if PRECEDENCE( $\mathcal{KB}, \rho, \sigma$ ) then
25:                 $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\leftrightarrow}\}$ 
26:              else
27:                 $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\rightarrow}\}$ 
28:              end if
29:            end if
30:          end if
31:          if PRECEDENCE( $\mathcal{KB}, \rho, \sigma$ ) then
32:            if ALTERNATEPRECEDENCE( $\mathcal{KB}, \rho, \sigma$ ) then
33:              if CHAINPRECEDENCE( $\mathcal{KB}, \rho, \sigma$ )  $\wedge$   $\neg$ CHAINRESPONSE( $\mathcal{KB}, \rho, \sigma$ ) then
34:                 $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\Leftarrow}\}$ 
35:              else
36:                if  $\neg$ ALTERNATERESPONSE( $\mathcal{KB}, \rho, \sigma$ ) then
37:                   $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\Leftarrow}\}$ 
38:                end if
39:              end if
40:            else
41:              if  $\neg$ RESPONSE( $\mathcal{KB}, \rho, \sigma$ ) then
42:                 $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\leftarrow}\}$ 
43:              end if
44:            end if
45:          end if
46:          if  $\neg$ (RESPONSE( $\mathcal{KB}, \rho, \sigma$ )  $\vee$  PRECEDENCE( $\mathcal{KB}, \rho, \sigma$ )) then
47:             $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}\}$ 
48:          end if
49:        end if
50:        if  $\top_{\rho, \sigma} \in \mathcal{B} \wedge \top_{\sigma, \rho} \in \mathcal{B}$  then
51:           $\mathcal{B} \leftarrow \mathcal{B} \setminus \{\top_{\rho, \sigma}, \top_{\sigma, \rho}\} \cup \{\top_{\sigma}^{\rho}\}$ 
52:        end if
53:         $\mathcal{B} \leftarrow$  DISCOVERNEGATIVECONSTRAINTS( $\rho, \sigma, \mathcal{B}, \mathcal{KB}$ )
54:      end for
55:    end if
56:  end for
57:  return  $\mathcal{B}$ 
58: end procedure

```

Algorithm 7 The DISCOVEREXISTENCECONSTRAINTS procedure's pseudo-code

```

1: procedure DISCOVEREXISTENCECONSTRAINTS( $\rho, \mathcal{B}, \mathcal{KB}$ )
2:   if PARTICIPATION( $\mathcal{KB}, \rho$ ) then
3:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho}^{1+}\}$ 
4:   end if
5:   if UNIQUENESS( $\mathcal{KB}, \rho$ ) then
6:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho}^{1-}\}$ 
7:   end if
8:   if INIT( $\mathcal{KB}, \rho$ ) then
9:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho}^i\}$ 
10:  end if
11:  if END( $\mathcal{KB}, \rho$ ) then
12:     $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho}^e\}$ 
13:  end if
14:  return  $\mathcal{B}$ 
15: end procedure

```

Algorithm 8 The DISCOVERNEGATIVECONSTRAINTS procedure's pseudo-code

```

1: procedure DISCOVERNEGATIVECONSTRAINTS( $\rho, \sigma, \mathcal{B}, \mathcal{KB}$ )
2:   if NOTCOEXISTENCE( $\mathcal{KB}, \rho, \sigma$ ) then
3:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\perp_{\rho}^{\rho}\}$ 
4:   else
5:     if NOTSUCCESSION( $\mathcal{KB}, \rho, \sigma$ ) then
6:        $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\leftrightarrow}\}$ 
7:     else
8:       if NOTCHAINSUCCESSION( $\mathcal{KB}, \rho, \sigma$ ) then
9:          $\mathcal{B} \leftarrow \mathcal{B} \cup \{\top_{\rho, \sigma}^{\oplus}\}$ 
10:      end if
11:    end if
12:  end if
13:  return  $\mathcal{B}$ 
14: end procedure

```

changed the perspective for Relation Constraints, making Support being related to single events rather than to the entire trace. Thus, e.g., if there are two occurrences of an activity a in a trace, and the first respects the *Response* constraint with b , whilst the other does not (e.g., in $accbca$), we consider the support for $Response(a, b)$ equal to 0.5. [56] would consider it equal to 0 instead.

Constraint	Support function	Constraint	Support function
$Existence(n, \rho)$	$1 - \frac{\sum_{i=0}^{n-1} \gamma_{\rho}(i)}{ T }$	$Participation(\rho)$	$1 - \frac{\gamma_{\rho}(0)}{ T }$
$Absence(m, \rho)$	$\frac{\sum_{i=0}^m \gamma_{\rho}(i)}{ T }$	$Uniqueness(\rho)$	$\frac{\gamma_{\rho}(0) + \gamma_{\rho}(1)}{ T }$
$Init(\rho)$	$\frac{\alpha_{\rho}}{ T }$	$End(\rho)$	$\frac{\omega_{\rho}}{ T }$
$RespondedExistence(\rho, \sigma)$	$1 - \frac{\delta_{\rho, \sigma}(\pm\infty)}{\Gamma_{\rho}}$		
$Response(\rho, \sigma)$	$1 - \frac{\delta_{\rho, \sigma}(+\infty)}{\Gamma_{\rho}}$	$Precedence(\rho, \sigma)$	$1 - \frac{\delta_{\sigma, \rho}(-\infty)}{\Gamma_{\sigma}}$
$AlternateResponse(\rho, \sigma)$	$1 - \frac{\beta_{\rho, \sigma}^{\rightarrow} + \delta_{\rho, \sigma}(+\infty)}{\Gamma_{\rho}}$	$AlternatePrecedence(\rho, \sigma)$	$1 - \frac{\delta_{\sigma, \rho}(-\infty) + \beta_{\rho, \sigma}^{\leftarrow}}{\Gamma_{\sigma}}$
$ChainResponse(\rho, \sigma)$	$\frac{\delta_{\rho, \sigma}(1)}{\Gamma_{\rho}}$	$ChainPrecedence(\rho, \sigma)$	$\frac{\delta_{\sigma, \rho}(-1)}{\Gamma_{\sigma}}$
$CoExistence(\rho, \sigma)$	$1 - \frac{\delta_{\rho, \sigma}(\pm\infty) + \delta_{\rho, \sigma}(\pm\infty)}{\Gamma_{\rho} + \Gamma_{\sigma}}$	$NotCoExistence(\rho, \sigma)$	$\frac{\delta_{\rho, \sigma}(\pm\infty) + \delta_{\sigma, \rho}(\pm\infty)}{\Gamma_{\rho} + \Gamma_{\sigma}}$
$Succession(\rho, \sigma)$	$1 - \frac{\delta_{\rho, \sigma}(+\infty) + \delta_{\sigma, \rho}(-\infty)}{\Gamma_{\rho} + \Gamma_{\sigma}}$	$NotSuccession(\rho, \sigma)$	$\frac{\delta_{\rho, \sigma}(+\infty) + \delta_{\sigma, \rho}(-\infty)}{\Gamma_{\rho} + \Gamma_{\sigma}}$
$AlternateSuccession(\rho, \sigma)$	$1 - \frac{\beta_{\rho, \sigma}^{\rightarrow} + \delta_{\rho, \sigma}(+\infty) + \beta_{\rho, \sigma}^{\leftarrow} + \delta_{\sigma, \rho}(-\infty)}{\Gamma_{\rho} + \Gamma_{\sigma}}$		
$ChainSuccession(\rho, \sigma)$	$\frac{\delta_{\rho, \sigma}(1) + \delta_{\sigma, \rho}(-1)}{\Gamma_{\rho} + \Gamma_{\sigma}}$	$NotChainSuccession(\rho, \sigma)$	$1 - \frac{\delta_{\rho, \sigma}(1) + \delta_{\sigma, \rho}(-1)}{\Gamma_{\rho} + \Gamma_{\sigma}}$

Table 4.6. Functions computing the Support of constraints

Taking inspiration from [56], we also associated to Support the Confidence Level (or Confidence for short) and Interest Factor metrics. Our adaptation of such metrics does not completely match the definition given in [56], though. Both estimate a level of relevance for a constraint, based on the assumption that the more the constrained activities appear in the log, the more their constraints should be taken into account. Roughly speaking, if an activity a appears once in the whole log, made of hundreds of thousands events, it is likely to be a glitch in the normal execution. Reminding the definition of *implying* and *implied* activities, given in Section 3.3, we define

1. as the Confidence Level, the product of a constraint's Support and the fraction of traces where the implying activity appears once at least,
2. as the Interest Factor, the product of the Confidence Level and either
 - (a) the fraction of traces where the implying activity appears once at least, if the constraint is an *ExistenceConstraint*, or
 - (b) the fraction of traces where the implied activity appears once at least, if the constraint is a *RelationConstraint* besides *NotCoExistence*, or
 - (c) the fraction of traces where the implied activity does not appear, if the constraint is a *NotCoExistence*.

The reasons why the definition of Interest Factor changes according to the type of the constraint template are that:

- (a) *ExistenceConstraints* have no implied activities,
- (b) *RelationConstraints* tie couple of activities, but...
- (c) ... *NotCoExistence*'s Support increases as the occurrence of an activity in a trace excludes the other in that couple.

The reader can find the procedure computing such metrics in Algorithm 10.

The overall DISCOVERCONSTRAINTS⁺ algorithm is presented in Algorithm 9. It consists of three procedure calls.

Algorithm 9 The pseudo-code of the DISCOVERCONSTRAINTS⁺ algorithm

Require: $\tau = 1.0$ # An optional user-defined threshold

- 1: **procedure** DISCOVERCONSTRAINTS⁺($\mathcal{KB}, \Sigma, |T|$)
- 2: $\mathcal{B}^+ \leftarrow \text{CALCMETRICSFORCONSTRAINTS}(\mathcal{KB}, \Sigma, |T|)$
- 3: $\mathcal{B}^+ \leftarrow \text{CLEANOUTPUT}(\mathcal{B}^+)$
- 4: $\mathcal{B}^+ \leftarrow \text{FILTEROUTPUTBYTHRESHOLD}(\mathcal{B}^+, \tau)$
- 5: **return** \mathcal{B}^+
- 6: **end procedure**

The first, CALCMETRICSFORCONSTRAINTS (Algorithm 10), populates the \mathcal{B}^+ bag. \mathcal{B}^+ is a collection of tuples $\langle b, s_b, c_b, i_b \rangle$, each associating to a constraint b the related

1. Support, s_b ,
2. Confidence Level, c_b ,

Algorithm 10 The pseudo-code of the CALCMETRICSFORCONSTRAINTS procedure

```

1: procedure CALCMETRICSFORCONSTRAINTS( $\mathcal{KB}, \Sigma, |T|$ )
2:    $\mathcal{B}^+ \leftarrow \emptyset$    # Initialization of the extended bag of constraints
3:   for all  $\rho \in \Sigma$  do
4:     if  $\Gamma_\rho > 0$  then
5:       for all  $b^x \sqsubseteq \text{ExistenceConstraint}$  do
6:          $s_b^x \leftarrow \text{CALCSUPPORT}(b^x, \rho)$ 
7:          $c_b^x \leftarrow s_b^x \cdot \left(1 - \frac{\gamma_\rho(0)}{|T|}\right)$ 
8:          $i_b^x \leftarrow c_b^x \cdot \left(1 - \frac{\gamma_\rho(0)}{|T|}\right)$ 
9:          $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \cup \langle b^x, s_b^x, c_b^x, i_b^x \rangle$ 
10:      end for
11:     for all  $\sigma \in \Sigma$  do
12:       for all  $b^y \sqsubseteq \text{RelationConstraint}$  do
13:          $s_b^y \leftarrow \text{CALCSUPPORT}(b^y, \rho, \sigma)$ 
14:          $c_b^y \leftarrow s_b^y \cdot \left(1 - \frac{\gamma_\rho(0)}{|T|}\right)$ 
15:         if  $\neg(b^y \sqsubseteq \text{NotCoExistence})$  then
16:            $i_b^y \leftarrow c_b^y \cdot \left(1 - \frac{\gamma_\sigma(0)}{|T|}\right)$ 
17:         else
18:            $i_b^y \leftarrow c_b^y \cdot \left(\frac{\gamma_\sigma(0)}{|T|}\right)$ 
19:         end if
20:          $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \cup \langle b^y, s_b^y, c_b^y, i_b^y \rangle$ 
21:       end for
22:     end for
23:   end if
24: end for
25:   return  $\mathcal{B}^+$ 
26: end procedure

```

3. Interest Factor, i_b .

For each constraint, let it be b^x if it belongs to the type of *ExistenceConstraints* (line 5), b^y otherwise (line 12), the Support value is computed by the `CALCSUPPORT` procedure (resp., lines 6 and 13). Here we do not report its pseudo-code, since it is meant to apply the functions listed in Table 4.6 for the calculation, according to the constraint template that the b^x (or b^y) constraint belongs to. The \subseteq operator specifies whether a given constraint belongs to, or is subsumed by, a constraint template, or a constraint type.

The computation of the Confidence Level is listed in lines 7, for *ExistenceConstraints*, and 14, for *RelationConstraints*. The reader can notice that the Confidence Level is actually the product of any constraint’s Support and the *Participation*’s Support of its implying activity, referring back to Table 4.6. The Interest Factor is assigned in lines 7, for *ExistenceConstraints*, 16, for *RelationConstraints* excluding *NotCoExistence*, and 18, for *NotCoExistence* Constraints.

Finally, we want to focus the attention on line 4. The condition stated there avoids those characters never appeared in the log from being the implying of any inferred constraint. Given that *ex falso quod libet*, a character that was never read might be declared as supporting each constraint, though it would be senseless to the mining purpose, as it would add no bit of information to the gathered knowledge.

In order to filter the irrelevant constraints out of the output, we make use of two methods, the aim of which is: (i) not to show trivially deducible constraints⁴; (ii) let the user decide a threshold of reliability, i.e., decide what is the least Support for a constraint to be considered valid.

The former objective is managed by Algorithm 11, `CLEANOUTPUT`, which requires no user intervention. The latter is obtained by Algorithm 12, `FILTEROUTPUTBYTHRESHOLD`, which expects a parameter to be optionally provided by the user: τ , i.e., the threshold, which is equal to 1.0 by default.

In Algorithm 11, the block between lines 4 and 17 involves every Relation Constraint (see the hierarchy in Figure 3.6).

There, the constraints that are subsumed by others are removed, when they have less Support. By definition (see Table 4.6), the Support of a subsumed constraint is less than or equal to the subsuming’s. Therefore, the block from line 6 to line 9 raises along the hierarchy of Figure 3.6, from the current constraint to the subsuming. Due to the monotonic increase of Support along the hierarchy, the loop from line 7 to line 9 stops when either (i) a subsuming constraint has a Support which is greater than the constraint under analysis, or (ii) no more “ancestors” along the hierarchy exist (i.e., the whole hierarchy share the same Support). In the first case, the current constraint is removed from the \mathcal{B}^+ bag. In the second case, its “parent” is deleted. Applying this selection to all of the constraints, it is ensured that only one constraint along the hierarchy is kept in the \mathcal{B}^+ bag. Therefore, the number of returned constraints is dramatically reduced.

The *RelationConstraints* that are also based on the conjunction of other two (which we called *MutualRelation*), are managed within the block from line 18 to line 25: see Figure 3.6 to see how they are connected. In that case, if a *MutualRelation*

⁴e.g., it is enough to say that *ChainPrecedence(a, b)* holds, rather than explicitly return as valid constraints *ChainPrecedence(a, b)*, *AlternatePrecedence(a, b)* and *Precedence(a, b)* – where the latter couple is directly implied by the first: see Figure 3.6

constraint is known to have a Support which is at least greater than *both* of the involved *RelationConstraints*, the latter couple can be removed. Otherwise, no action is taken. Finally, from line 26 to line 33, a selection between each *NegativeRelation* constraint and its negated is given, on the basis of the respective Support: the one which has the least between the two, is removed.

Algorithm 11 The pseudo-code of the CLEANOUTPUT procedure

```

1: procedure CLEANOUTPUT( $\mathcal{B}^+$ )
2:    $\overline{\mathcal{B}^+} := \text{clone } \mathcal{B}^+$ 
3:   for all  $\langle b, s_b, c_b, i_b \rangle \in \overline{\mathcal{B}^+}$  do
4:     if  $b \sqsubseteq \text{RelationConstraint}$  then
5:       if  $\text{hasParent}(\overline{\mathcal{B}^+}, b)$  then
6:          $p := b$ 
7:         repeat
8:            $\langle p, s_p, c_p, i_p \rangle := \text{getParent}(\overline{\mathcal{B}^+}, p)$ 
9:           until  $(s_p = s_b) \wedge \text{hasParent}(\overline{\mathcal{B}^+}, p)$ 
10:          if  $s_p > s_b$  then
11:             $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \setminus \{ \langle b, s_b, c_b, i_b \rangle \}$ 
12:          else
13:             $\langle p, s_p, c_p, i_p \rangle := \text{getParent}(\overline{\mathcal{B}^+}, b)$ 
14:             $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \setminus \{ \langle p, s_p, c_p, i_p \rangle \}$ 
15:          end if
16:        end if
17:      end if
18:      if  $b \sqsubseteq \text{MutualRelation}$  then
19:         $\langle f, s_f, c_f, i_f \rangle := \text{getForward}(\overline{\mathcal{B}^+}, b)$ 
20:         $\langle r, s_r, c_r, i_r \rangle := \text{getBackward}(\overline{\mathcal{B}^+}, b)$ 
21:        if  $s_f < s_b \wedge s_r < s_b$  then
22:           $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \setminus \{ \langle f, s_f, c_f, i_f \rangle \}$ 
23:           $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \setminus \{ \langle r, s_r, c_r, i_r \rangle \}$ 
24:        end if
25:      end if
26:      if  $b \sqsubseteq \text{NegativeRelation}$  then
27:         $\langle n, s_n \rangle := \text{getNegated}(\overline{\mathcal{B}^+}, b)$ 
28:        if  $s_n \leq s_b$  then
29:           $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \setminus \{ \langle n, s_n, c_n, i_n \rangle \}$ 
30:        else
31:           $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \setminus \{ \langle b, s_b, c_b, i_b \rangle \}$ 
32:        end if
33:      end if
34:    end for
35:    return  $\mathcal{B}^+$ 
36: end procedure

```

The *hasParent*, *getParent*, *getForward*, *getBackward* and *getNegated* functions explore the subsumptions and the associations between constraints as drawn in Figure 3.6:

- *hasParent* and *getParent* traverse the subsumption hierarchy;
- *getForward* and *getBackward* return the tuples of the \mathcal{B}^+ bag (or its clone) referred to the constraints which are implied by *MutualRelations*, e.g., between *CoExistence* and the two related *RespondedExistence* (the same applies to the hierarchies below, as for *ChainSuccession* wrt. *ChainResponse* and *ChainPrecedence*);

- *getNegated* returns the *MutualRelation* constraint (like *CoExistence*) that is negated by the *NegativeRelation* constraint (like *NotCoExistence*).

These functions do not depend on the interpretation of the MINERful interplay and the MINERful ownplay, but only on the semantics of constraints.

The behaviour of *hasParent*, *getParent*, *getForward*, *getBackward* and *getNegated* functions is detailed in Table 4.7.

Constraint Name	Symbol	<i>hasParent</i>	<i>getParent</i>	<i>getNegated</i>	<i>getForward</i>	<i>getBackward</i>
<i>RespondedExistence</i> (ρ, σ)	$\top_{\rho, \sigma}$	<i>false</i>	-	-	-	-
<i>Response</i> (ρ, σ)	$\top_{\rho, \sigma}^{\rightarrow}$	<i>true</i>	$\top_{\rho, \sigma}$	-	-	-
<i>AlternateResponse</i> (ρ, σ)	$\top_{\rho, \sigma}^{\Rightarrow}$	<i>true</i>	$\top_{\rho, \sigma}^{\rightarrow}$	-	-	-
<i>ChainResponse</i> (ρ, σ)	$\top_{\rho, \sigma}^{\Leftrightarrow}$	<i>true</i>	$\top_{\rho, \sigma}^{\Rightarrow}$	-	-	-
<i>Precedence</i> (ρ, σ)	$\top_{\rho, \sigma}^{\leftarrow}$	<i>true</i>	$\top_{\sigma, \rho}$	-	-	-
<i>AlternatePrecedence</i> (ρ, σ)	$\top_{\rho, \sigma}^{\Leftarrow}$	<i>true</i>	$\top_{\rho, \sigma}^{\leftarrow}$	-	-	-
<i>ChainPrecedence</i> (ρ, σ)	$\top_{\rho, \sigma}^{\Leftrightarrow}$	<i>true</i>	$\top_{\rho, \sigma}^{\Leftarrow}$	-	-	-
<i>CoExistence</i> (ρ, σ)	\top_{σ}^{ρ}	<i>false</i>	-	-	$\top_{\rho, \sigma}$	$\top_{\sigma, \rho}$
<i>Succession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\leftrightarrow}$	<i>true</i>	\top_{σ}^{ρ}	-	$\top_{\rho, \sigma}^{\rightarrow}$	$\top_{\sigma, \rho}^{\leftarrow}$
<i>AlternateSuccession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\Leftrightarrow}$	<i>true</i>	$\top_{\rho, \sigma}^{\leftrightarrow}$	-	$\top_{\rho, \sigma}^{\Rightarrow}$	$\top_{\sigma, \rho}^{\Leftarrow}$
<i>ChainSuccession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\Leftrightarrow}$	<i>true</i>	$\top_{\rho, \sigma}^{\Leftrightarrow}$	-	$\top_{\rho, \sigma}^{\Leftrightarrow}$	$\top_{\sigma, \rho}^{\Leftrightarrow}$
<i>NotChainSuccession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\nleftrightarrow}$	<i>false</i>	-	$\top_{\rho, \sigma}^{\Leftrightarrow}$	-	-
<i>NotSuccession</i> (ρ, σ)	$\top_{\rho, \sigma}^{\nleftrightarrow}$	<i>true</i>	$\top_{\rho, \sigma}^{\nleftrightarrow}$	$\top_{\rho, \sigma}^{\leftrightarrow}$	-	-
<i>NotCoExistence</i> (ρ, σ)	\perp_{σ}^{ρ}	<i>true</i>	$\top_{\rho, \sigma}^{\nleftrightarrow}$	\top_{σ}^{ρ}	-	-

Table 4.7. The functions navigating the constraints' hierarchy of subsumptions

Algorithm 12 The pseudo-code of the FILTEROUTPUTBYTHRESHOLD procedure

```

1: procedure FILTEROUTPUTBYTHRESHOLD( $\mathcal{B}^+, \tau$ )
2:   for all  $\langle b, s_b, c_b, i_b \rangle \in \mathcal{B}^+$  do
3:     if  $\neg(|2 \cdot \tau - 1| \leq s_b \leq 1)$  then
4:        $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \setminus \{\langle b, s_b, c_b, i_b \rangle\}$ 
5:     else
6:       if  $\tau < 1$  then
7:          $\mathcal{B}^+ \leftarrow \mathcal{B}^+ \setminus \{\langle b, s_b, c_b, i_b \rangle\} \cup \{\langle b, \frac{s_b - \tau}{1.0 - \tau}, c_b, i_b \rangle\}$ 
8:       end if
9:     end if
10:  end for
11:  return  $\mathcal{B}^+$ 
12: end procedure

```

The FILTEROUTPUTBYTHRESHOLD procedure (Algorithm 12) finally filters out those constraints whose Support is below a given threshold. Actually, the algorithm returns not only the constraints which strictly respect that soil. Such razor-blade approach would not take into account the *pleiotropic*⁵ behavior of constraints. I.e., there might be a diapason-like effect, due to the interaction of two or more constraints acting at the same time: it might leverage the Support of another, up to limits that could hide the original from the attention of the user. Thus (see line 3), we prefer to keep not only the b constraints such that the related Support s_b is greater than (or equal to) τ (the user-defined threshold). We also give the evidence of those constraints whose Support is even below that value: see line 3. If (line 6) τ is not the default one (i.e., 1.0) then, the Support given in output is recalculated and shown with respect to the soil of τ , thus assuming either a negative or positive value, scaled to the distance from τ to 1.0 (line 6).

Discussion on the complexity In the following we discuss the complexity of the concurrent execution of Algorithm 9 by separately analyzing Algorithms 10, 11 and 12, i.e., the procedures it is based upon.

Lemma 3. *The procedure for discovering the constraints of processes out of the MINERful knowledge base, along with their Support, Confidence Level and Interest Factor is (i) quadratic in time w.r.t. the size of the alphabet, (ii) linear in time w.r.t. the number of constraint templates, which is fixed and equal to 18 (thus, constant); therefore, the complexity is $O(|\Sigma|^2)$.*

Proof. Algorithm 10 essentially executes two nested cycles: the outer from line 3 to line 24, the inner, from line 11 to line 22. Inside them both, only mathematical operations are performed, as many as the constraint templates are. Such calculi are executed on top of a fixed number of entries in the MINERfulKB. The complexity of CALCMETRICSFORCONSTRAINTS is $O(|\Sigma|^2)$, then. The number of entries in the bag \mathcal{B}^+ is $O(|\Sigma|^2)$ as well.

Algorithm 11 begins with the cloning of that bag (line 2). Therefore, it is a $O(|\Sigma|^2)$ procedure by itself. Afterwards, each of the elements in the clone of \mathcal{B}^+ is subject to some checks and modifications. The cycle from line 3 to line 34 iterates $O(|\Sigma|^2)$ times. Remembering what stated in the explanation of the procedure, the

⁵In chemical biology, pleiotropy occurs when one gene influences multiple phenotypic traits. Consequently, a mutation in a pleiotropic gene may have an effect on some or all traits simultaneously.

functions invoked (*hasParent*, *getParent*, etc.) can be considered as invocations of an oracle, being based on the knowledge about semantics of the constraint templates and not on the MINERfulKB. The loop from line 7 to line 9 is executed, at most, a limited number of times. Such limit does not depend on the input, but on the hierarchy of subsumptions in the constraint templates. By visual inspection of Figure 3.6 or Table 4.7, it is evident that such limit is fixed and less than or equal to 4. Thus, CLEANOUTPUT is $O(|\Sigma|^2)$ too.

Finally, Algorithm 12's complexity is affected by the iteration on the elements of \mathcal{B}^+ . Hence, it is $O(|\Sigma|^2)$ like the other two procedures.

Therefore, DISCOVERCONSTRAINTS is a $O(|\Sigma|^2)$ algorithm. \square

4.1.6 The complexity of the MINERful algorithm

Theorem 1. *The MINERful algorithm is (i) linear time w.r.t. the number of strings in the testbed, (ii) quadratic time w.r.t. the size of strings in the testbed, (iii) quadratic time w.r.t. the size of the alphabet; therefore, the complexity is $O(|T| \cdot |t_{max}|^2 \cdot |\Sigma_T|^2)$.*

Proof. The Theorem directly follows from Lemma 1, for the computation of the MINERfulKB, and either (i) Lemma 2, if we consider the discovery of constraints *without* any estimation of reliability (Support) and relevance (Confidence Level and Interest Factor), or (ii) Lemma 3, for the discovery of constraints *with* the estimation of reliability (Support) and relevance (Confidence Level and Interest Factor) metrics. \square

Chapter 5

Experiments and evaluation

In order to evaluate MAILOFMINE, we considered *(i)* its efficiency, in terms of computation time, and *(ii)* its efficacy, in terms of conformance of the discovered processes to reality.

We conducted the performance experiments on its Process Mining module (Section 5.1). To this extent, we first produced synthetic logs, stemming from predefined workflow models. Then, we processed such logs through the Process Mining module. For every log, we measured the time it took to discover the originating workflow model, and analyzed its performance with respect to the input size. The IR module's efficiency was not tested by itself, being it a third-party component.

In order to inspect the quality of results and validate the approach, we verified the whole MAILOFMINE system on a real case study (Section 5.2). There, data were extracted from the mailbox of an authors' colleague, known to be an expert in the area of the process to discover. As usual for artful processes, the process behind the analyzed email messages was not known a priori. Therefore, we could not apply an automated comparison between the resulting workflow model and the originating process, since no definition for the originating process was available at all. Thus, the expert was requested to analyze and assess the discovered workflow model by categorizing the mined constraints.

5.1 Experiments

Experiments on the proposed technique have been conducted on both synthetic and real data. All of the tests were performed on a Sony VAIO VGN-FE11H (Intel Core Duo T2300 1.66 GHz, 2 MB L2 cache, with 2 GB of DDR2 RAM at 667 Mhz), having Ubuntu Linux 10.04 as the operating system and Java JRE v1.6. In order to produce synthetic logs, we first considered the example workflow, outlined in Section 3.3.1. Random strings were created integrating our tool with Xeger¹, a Java open-source library for generating random text from regular expressions, based on [65].

We tested the algorithm by varying the input in terms of alphabet size (different symbols appearing in the traces), number of constraints, range of the number of characters per string (see Setup T1 in Table 5.1). The constraints ranged

¹<http://code.google.com/p/xeger/>

Setup	Min. length	Max. length	Number of traces	Alphabet size	Total runs
T1	[0,8]	[5,20]	$[10^2, 10^6]$	[2, 5]	29 000
T2	[0,2]	[10,25]	$[10^3, 16 \cdot 10^3]$	[2, 50]	13 536

Table 5.1. Experiments' setup

from a minimal set of four (*Unique*(n), *Participation*(n), *End*(n), *Succession*(p, n)) to the maximal set of seven (including *Response*(r, p), *RespondedExistence*(c, p), *AlternatePrecedence*(r, c)). In order to consider the performances' degradation over increasing alphabets, we also executed a new set of experiments, according to Setup T2 (Table 5.1).

Source	Tasks	Traces	Events processed	Total comp. time	Engine
Synth. log, Setup T1 (intermediate case)	5	100 000	1 676 447 (avg. 16.764)	00:00:15	MINERful
				00:13:39	Declare Miner [57]
Synth. log, Setup T2 (worst case)	52	16 000	296 277 (avg. 18.517)	00:00:25	MINERful
				00:21:24	Declare Miner [57]
Financial log [99]	24	13 087	262 200 (avg. 20.035)	00:00:08	MINERful
				00:08:54	Declare Miner [57]
Hospital log [98]	624	1 143	150 291 (avg. 131.488)	00:04:34	MINERful
				03:39:13	Declare Miner [57]

Table 5.2. Performances of MINERful over synthetic and real cases

Figure 5.1 shows the time taken by the algorithm to run, in comparison with the number of traces in the logs. As theoretically proven in Theorem 1, according to which there is a linear dependence between the two, the graph draws a section of straight line.

The time taken for the algorithm to complete, with respect to the average length of the strings is depicted in Figure 5.4. There, the alphabet size is fixed and equal to 5. The dependency is quadratic as expected.

The time taken by the algorithm to discover the workflow model, with respect to the size of the input (namely, the total number of events read) is depicted in Figure 5.3. There, each curve correspond to a different number of activities in the log. The reader can see four sections of a parabola, confirming that the algorithm is quadratic w.r.t. the size of the strings.

The shape of each parabola is very flatten due to the nature of its non-linearity: looking back to the algorithm (Section 4.1.3 and Algorithm 3), it is caused by a loop cycling more as characters are repeated (for defining the $\delta_{\rho,\sigma}$ function), nested in a loop cycling more as different characters are encountered (i.e., executing the former loop for each unread ρ) – both start to grow together in terms of cycles only if strings tend to be far longer than the size of the alphabet.

Given that the graph in Figure 5.3 showed how parabolae were shifted onwards as the number of activities considered raised, we studied the dependency of the running time w.r.t. the size of the alphabet. The resulting graph is depicted in Figure 5.4. Basing on Setup T2 (Table 5.1), we fixed the parameters specifying the minimum and maximum length of each string to their upper extremal (i.e., resp. 2 and 25), as far as the number of traces (again, the topmost value, that is 16000), letting the number of characters vary. Again, the shape of the fitting curve is the section of a parabola, confirming what theoretically predicted by Theorem 1.

Figure 5.5 re-elaborates what depicted in Figure 5.4. It separates the analysis of the time taken by the algorithm for its computation into its two main procedures: (i) 1. the construction of the MINERfulKB (Section 4.1.3) and 2. the discovery of constraints, obtained by queries over the MINERfulKB itself (Section 4.1.4). The graph let the reader infer clearly how faster is the second phase, with respect to the first. At first sight, the curve fitting the graph related to the querying phase might look like a straight line. If it were so, it would have contradicted the theoretical results of Lemma 2, according to which the report between the execution time and the alphabet size was quadratic.

Figure 5.6 is a zoom on the second curve, then. It shows that the aforementioned Lemma 2 is confirmed by experimental verification. The flattened shape derived from the deformation given by the more convex curve in the comprehensive view of Figure 5.5.

In order to test the efficiency of MINERful when dealing with real-life cases, we tested it with two well known benchmarks, taken from the latest Business Process Intelligence Challenges (BPIC): a Dutch academic hospital log [98] and a Dutch financial institute log [99]. Table 5.2 reports some interesting results, taken from the experiments on both synthetic and real logs. For each test, we performed the analysis on Declare Miner [57], publicly available as a ProM plug-in². We tuned Declare Miner so to check the same constraints MINERful discovers³. Table 5.2 shows that

²<http://www.win.tue.nl/declare/declare-miner/>

³PoE equal to 100%, PoIW equal to 0%, PoI equal to 100%, all constraints selected excluding

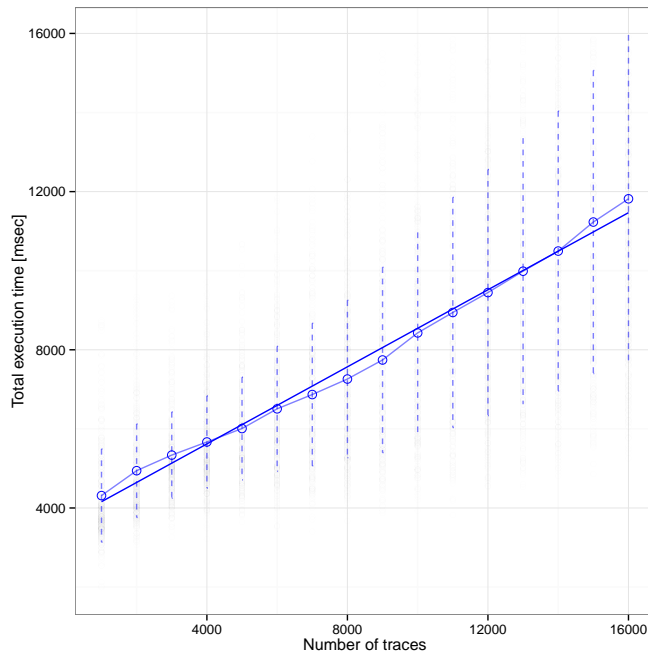


Figure 5.1. Experimental results of MINERful: time needed for the execution, with respect to the number of traces (from Setup T2 – Table 5.1): only the tests where the size of the alphabet is greater than 25 are considered)

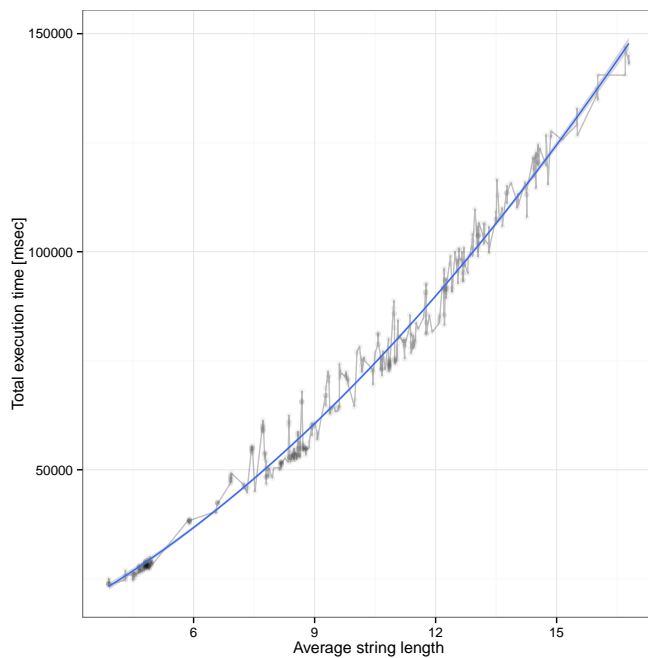


Figure 5.2. Experimental results of MINERful: time needed for the execution, with respect to the string length – from Setup T1 (Table 5.1): only the tests where the size of the alphabet is equal to 5 are plotted

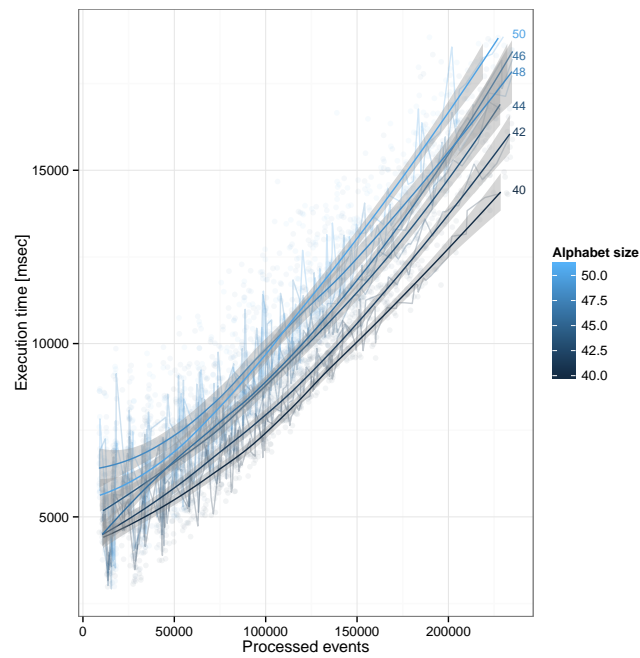


Figure 5.3. Experimental results of MINERful: time needed for the execution, with respect to the number of events; each curve corresponds to a given size of the alphabet – from Setup T2 (Table 5.1)

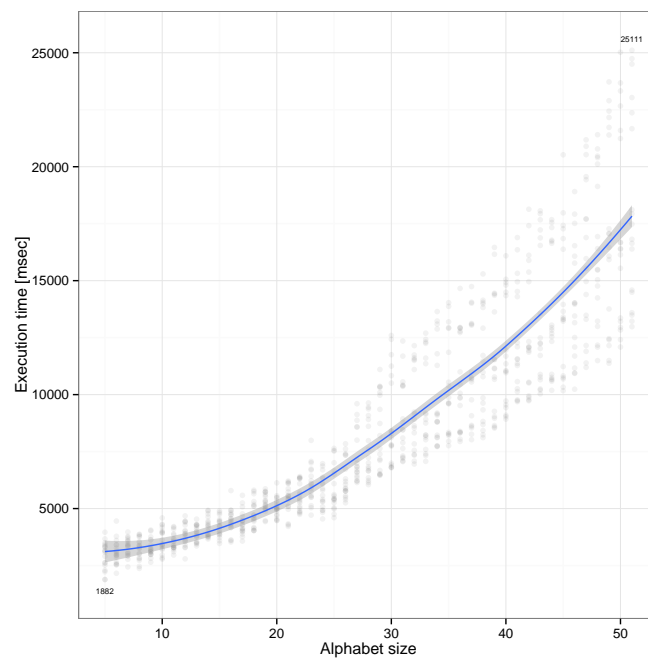


Figure 5.4. Experimental results of MINERful: time needed for the execution, with respect to the size of the alphabet – from Setup T2 (Table 5.1)

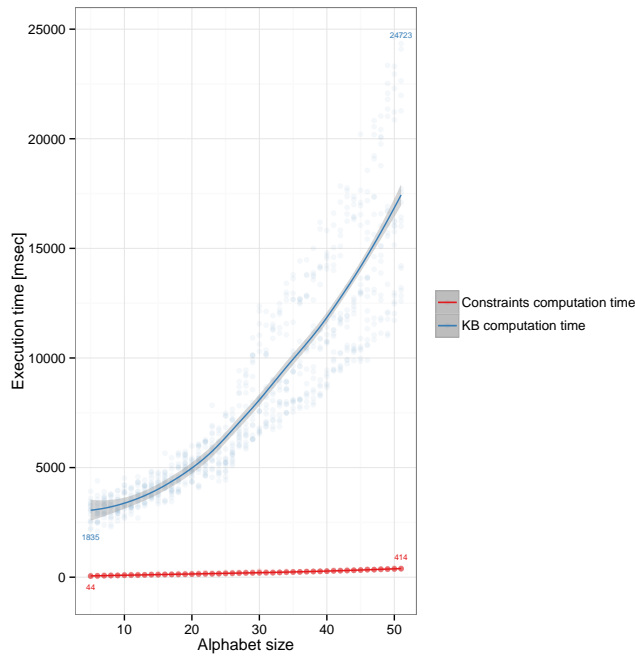


Figure 5.5. Experimental results of MINERful: time taken for the construction of the MINERfulKB and the discovery of constraints by queries over the MINERfulKB, with respect to the size of the alphabet – from Setup T2 (Table 5.1)

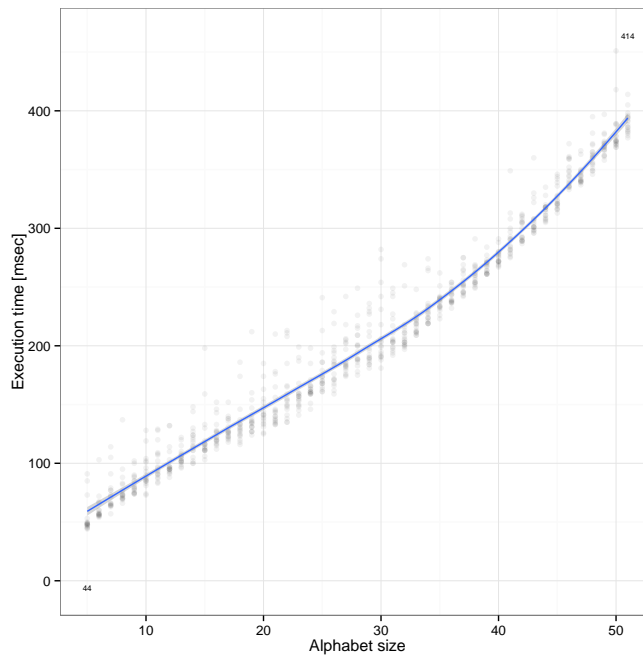


Figure 5.6. Experimental results of MINERful: time taken for the construction the discovery of constraints, given the MINERfulKB, with respect to the size of the alphabet – from Setup T2 (Table 5.1)

the execution of MINERful is faster than Declare Miner. We recall here that Declare Miner could potentially discover any LTL-expressible constraint, whereas MINERful is not customizable at this level from the user. Nonetheless, MINERful is completely unsupervised, as the user has not to select in advance the constraints that have to be checked. She is (optionally) requested to provide a threshold to filter out some loosely-supported constraints, just in the end, for sake of her ease to access the extracted information, with no impact on the performances of the algorithm.

5.1.1 Experiments over artificial error-injected logs

In order to test the robustness of MINERful with respect to the presence of errors in logs, we built an additional testing module, which injected a controlled noise in the sequences of traces.

We identified three possible *types of error injection*:

1. insertion of spurious events in the log;
2. deletion of events from the log;
3. random insertion/deletion of events.

The errors were spread according to a given percentage⁴. The tester could also specify whether errors had to refer to a given activity, or not. In the latter case, every insertion or deletion was applied to an event picked each time at random.

In order to define how many errors had to be injected, and where, a *spreading policy* was requested too. It could be either:

1. to calculate the number of errors to inject w.r.t. the whole log, and distribute the error injections accordingly, or
2. to calculate the number of errors to inject w.r.t. every single trace, case by case.

In the latter case, every trace was made affected by a number of errors, computed on the number of target events in that trace. This reproduces a systematic error, done in every enactment of the process by the executor. In the former, some traces could remain untouched. That pretends to be a little more realistic scenario, where a given number of mistakes were made in the execution of the process could be done.

We remark that, in our scenario, mistakes made by the actor were not the only possible cause of errors in logs. Since we extract logs out of conversation through an Information Retrieval module, the error could be possibly caused by false positives (inserted spurious events) or false negatives (deleted events) during the detection of the indicia (see Section 3.1).

Thus, we conducted an extensive analysis on the reaction of MINERful, through an experiment set up as summarized in Table 5.3.

We created 18 groups of 9300 synthetic logs each. Every group was generated so to comply to one constraint at a time, among the 18 templates involving **a**, as the implying activity, and (optionally) **b**, as the implied (i.e., *Participation(a)*,

those not covered in Table 3.2.

⁴In case the the calculated number of errors to inject resulted in a non-integer number, the actual amount of errors was rounded up to the next integer (e.g., 0.2 was rounded to 1 error to inject).

Activities (target)	8 (1)	Spreading policies	3
Generating constraints	18	Error types	3
Trace length	[0, 30]	Runs per combination	50
Log size	1 000	Error injection percentage	[0, 30]
		Total runs	167 400

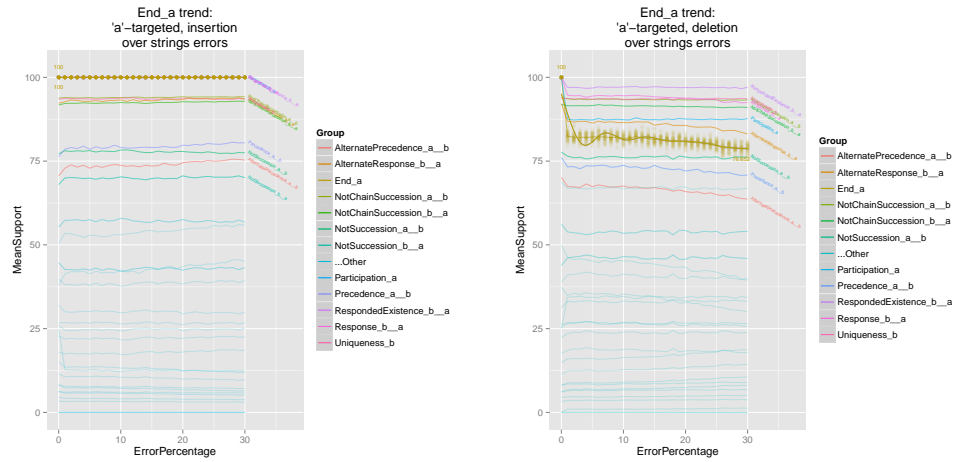
Table 5.3. Setup of the experiments for monitoring the reaction of MINERful to the controlled error injection into logs

$Uniqueness(a)$, \dots , $RespondedExistence(a, b)$, $Response(a, b)$, \dots). The alphabet comprised 6 more non-constrained activities (c, d, \dots , h), totalling 8. We chose a as the target activity for the injection of errors, being it the implying in all of the generating constraints. This way, we could have a stronger evidence of the effect produced on the discovery of declarative workflows, when errors occur. Then, we injected errors in the synthetic logs, with all of the possible combinations of the aforementioned parameters (*i*) insertion, deletion or random error type, (*ii*) over-string or over-collection spreading policy, (*iii*) error injection percentage ranging between 0 and 30%) and ran MINERful on the resulting altered logs. We collected the results and, for each of the 18 groups of logs, analyzed the trend of the support for the generating constraint. I.e., we looked at how the Support for the only constraint which had to be verified all over the log lowered, w.r.t. the increasing percentage of errors injected. We also highlighted those other constraints whose topmost computed Support exceeded the value of 0.75⁵, being them the most likely candidates to be false positives in the discovery. We did not consider Confidence Level nor Interest Factor since the number of appearances of events were not determined by actual enactments, but rather by strings generated uniformly at random. Such values, hence, tended to be equal for all events, unless they were directly constrained by the generating constraints.

The analysis of within-trace error-injected logs revealed to be more effective in stressing the resilience of constraints with respect to certain types of errors. In other words, it conversely showed the structural weaknesses of constraint templates w.r.t. some error types even for small percentages of injected errors. For instance, the Support of $End(a)$'s (Figure 5.7) is not affected by the insertion of spurious a's in the traces (see Figure 5.7a), whereas it suffers from deletions of a's (Figure 5.7b). The hierarchy of $NegativeRelation$ constraints (see Figure 3.6) reveals to be extremely "resistant" to errors of any kind, as shown in Figure 5.10.

In Section 3.3 we described the mechanism tying $MutualRelation$ constraints to *forward* and *backwards* related constraints, as in the case of $AlternateSuccession$ w.r.t. $AlternateResponse$ and $AlternatePrecedence$. Then, here we remark that since (*i*) the Support for $AlternateResponse(a, b)$ remains unchanged in case of spurious inserted spurious a's (Figure 5.8a), but not in case of deleted a's (Figure 5.8b), whilst (*ii*) conversely, the Support for $AlternatePrecedence(a, b)$ remains unchanged in case of spurious deleted a's (Figure 5.8c), but not in case of inserted spurious a's

⁵We recall that assigning a constraint the Support of 0.5 is equivalent to asserting that such constraint will hold if, tossing a coin, a cross is shown in the end. Thus, 0.75 is the least value of the topmost half of the "reliable" range.



(a) The trend of Support for $End(a)$, w.r.t. the percentage of spurious events insertion errors, injected into every string

(b) The trend of Support for $End(a)$, w.r.t. the percentage of events deletion errors, injected into every string

Figure 5.7. The trend of the Support for End , w.r.t. the errors injected in the log, within every trace.

(Figure 5.8d), *AlternateSuccession* somehow takes the sensitivity towards errors of both, resulting in a decreasing Support for both faulty insertions and deletions of a 's (Figure 5.9).

The analysis of over-collection error-injected logs showed smoother changes in curves, since errors are spread on a wider area of appearances, for the targeted activity. Thus, it reveals a more realistic trend for the assessment of discovered constraints in presence of errors. We reasonably expect to have sparse errors in logs, rather than a fixed percentage of faults for every trace, as a matter of fact.

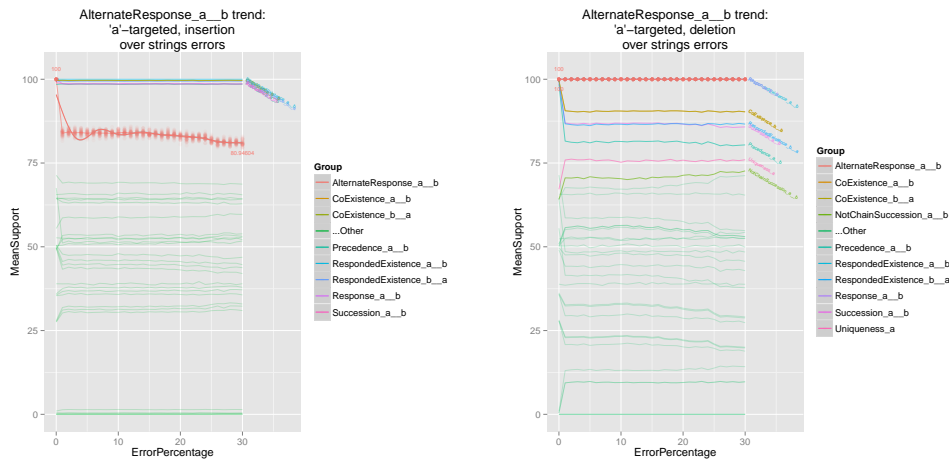
Along a branch in the constraints hierarchy, we expect for a constraint that the more it is restrictive, the more its Support decreases as deviations from the expected behavior are collected. We can prove it by evidence in, e.g., Figure 5.11, where the curve's slope gets steeper as we analyze the subsumed constraints along the *MutualRelation* constraints (i.e., *CoExistence*, *Succession*, *AlternateSuccession*, *ChainSuccession*).

The interested reader can download the whole collection of graphs summing the gathered results at the following address:

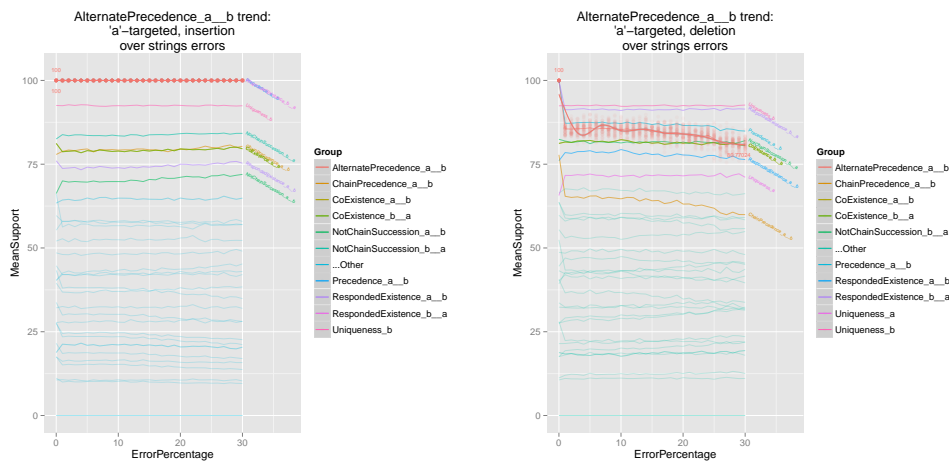
<http://www.dis.uniroma1.it/~lcdc/code/minerful/latest/errorinjectiontestresults.zip>

5.2 Evaluation on a real case study

As a case study, we took 6 mailbox IMAP folders containing email messages which concerned the management of 5 different European Research Projects (Table 5.3a). 5 over 6 such folders belonged to a domain expert, and one to the writer of this thesis. Our aim was to use MAILOFMINE in order to discover the artful process of

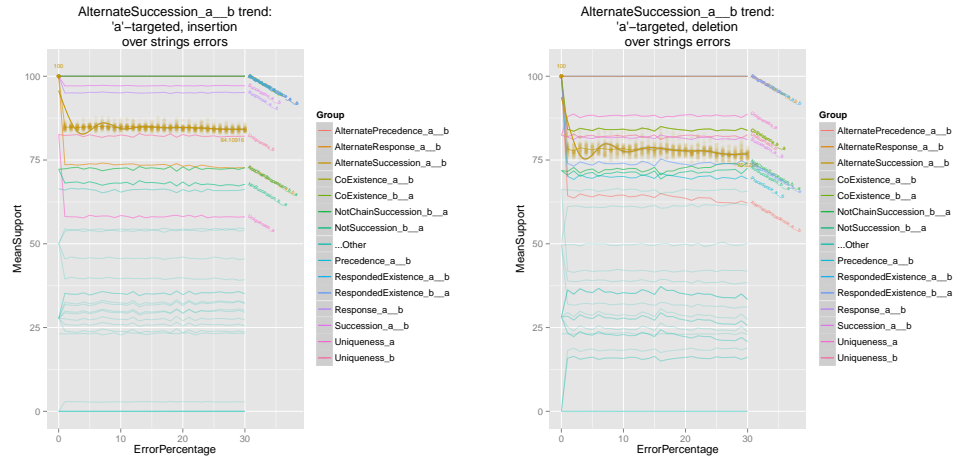


- (a) The trend of Support for $AlternateResponse(a, b)$, w.r.t. the percentage of spurious events insertion errors, injected into every string
- (b) The trend of Support for $AlternateResponse(a, b)$, w.r.t. the percentage of spurious events deletion errors, injected into every string

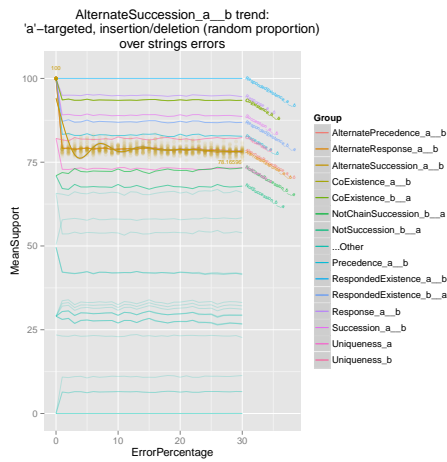


- (c) The trend of Support for $AlternatePrecedence(a, b)$, w.r.t. the percentage of events deletion errors, injected into every string
- (d) The trend of Support for $AlternatePrecedence(a, b)$, w.r.t. the percentage of events deletion errors, injected into every string

Figure 5.8. The trend of Support for $AlternateResponse$ and $AlternatePrecedence$, w.r.t. the errors injected in the log. The error injection policies under exam are both the insertion and deletion of a events, within each trace.

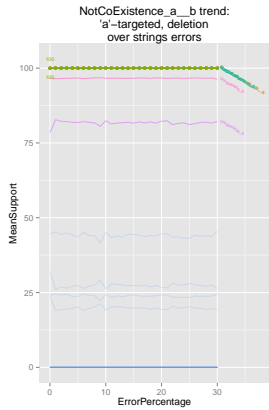


- (a) The trend of Support for $AlternateSuccession(a, b)$, w.r.t. the percentage of spurious events insertion errors, injected into every string
- (b) The trend of Support for $AlternateSuccession(a, b)$, w.r.t. the percentage of events deletion errors, injected into every string

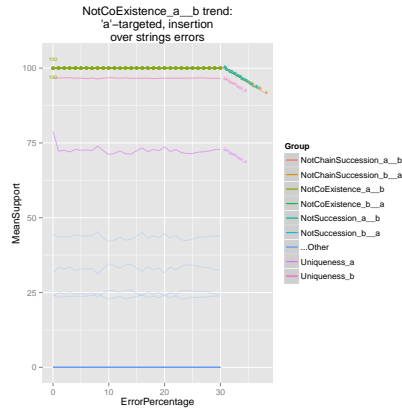


- (c) The trend of Support for $AlternateSuccession(a, b)$, w.r.t. the percentage of both events deletion and insertion errors, injected into every string

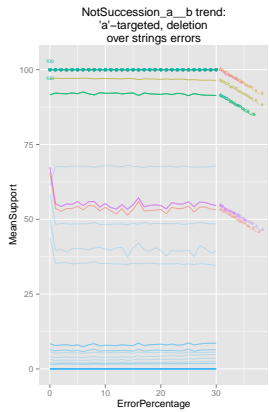
Figure 5.9. The trend of Support for $AlternateSuccession$, w.r.t. the errors injected in the log, within each trace.



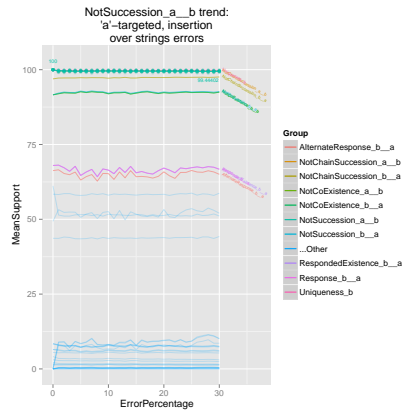
(a) The trend of Support for *NotCoExistence(a, b)*, w.r.t. the percentage of events deletion errors, injected into every string



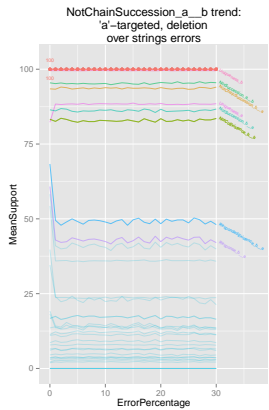
(b) The trend of Support for *NotCoExistence(a, b)*, w.r.t. the percentage of spurious events insertion errors, injected into every string



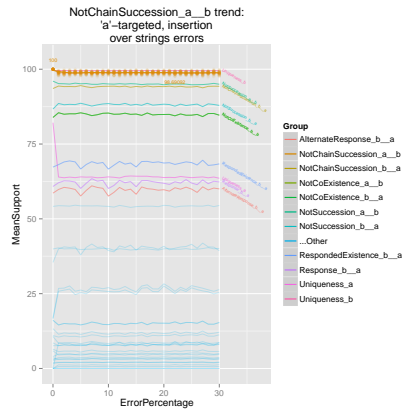
(c) The trend of Support for *NotSuccession(a, b)*, w.r.t. the percentage of events deletion errors, injected into every string



(d) The trend of Support for *NotSuccession(a, b)*, w.r.t. the percentage of spurious events insertion errors, injected into every string

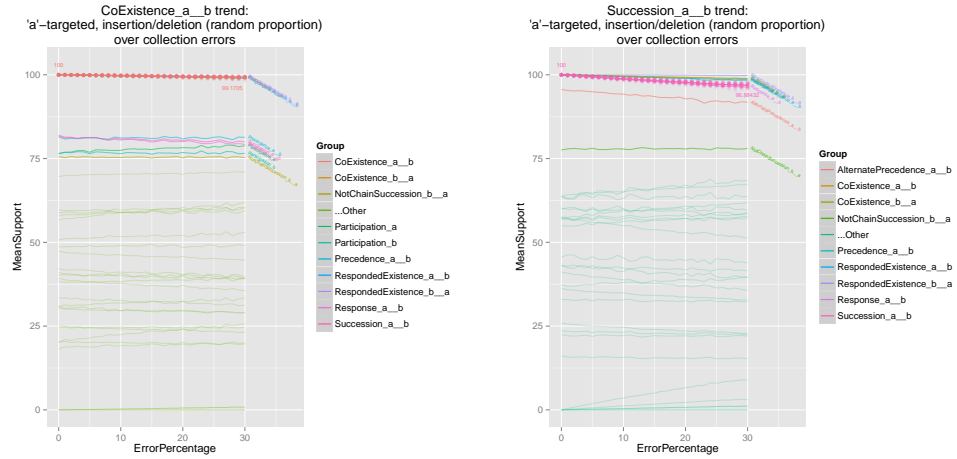


(e) The trend of Support for *NotChainSuccession(a, b)*, w.r.t. the percentage of events deletion errors, injected into every string



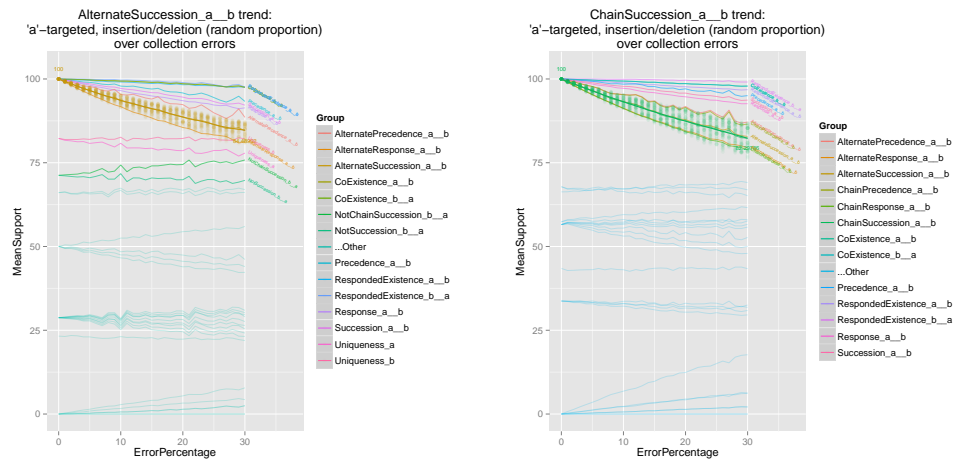
(f) The trend of Support for *NotChainSuccession(a, b)*, the percentage of spurious events insertion errors, injected into every string

Figure 5.10. The trend of the support for *NegativeRelation* constraints, w.r.t. the errors injected in the log, within each trace.



(a) The trend of Support for $CoExistence(a, b)$, w.r.t. the percentage of both events deletion and insertion errors, spread into the whole log

(b) The trend of Support for $Succession(a, b)$, w.r.t. the percentage of both events deletion and insertion errors, spread into the whole log



(c) The trend of Support for $AlternateSuccession(a, b)$, w.r.t. the percentage of injected errors

(d) The trend of Support for $ChainSuccession(a, b)$, w.r.t. the percentage of both events deletion and insertion errors, spread into the whole log

Figure 5.11. The trend of Support for the *MutualRelation* constraints, w.r.t. the errors injected in the log. The error injection policy under exam is the random insertion/deletion of a events, over the whole log.

(a) The input

		Mailbox						
		1	2	3	4	5	6	Total
Messages		3523	39	844	4746	1479	60	8770

(b) Retrieved information

Setup	Verbs	Objects	Words	Expressions	Activities	Indicia
E1 (extended)	4	36	40	144	55	317
E2 (reduced)	4	6	10	24	13	139

(c) Mined process

Setup	Constraints found	Constraints shown	Comp. time [msec]
E1 (extended)	4533	3074	1077
E2 (reduced)	378	218	324

Table 5.4. Evaluation of MAILOFMINE on a case study: preliminary setups and gathered data

managing European Research Projects and validate the result.

Together with the expert, we also defined a vocabulary of 40 domain-specific words, divided into 36 objects and 4 verbs (the list is in Table 5.5).

Once the Email Fetcher (Section 3.1) stored the email messages in the database, the IR module extracted the activities, i.e., those expressions appearing in at least an email. We recall here that we call *indicia* those email messages proving the execution of an activity. In this case, 8.998% of the total amount of email messages were considered related to the execution of an activity. The result of the Information Retrieval task is quantitatively summarized in Table 5.3b.

The Tracer module turned the ordered *indicia* into a log. That log was passed to the Miner, which discovered more than 4500 constraints to hold non-redundantly for the log. Seen the results gathered experimentally (see Section 5.1.1), we agreed with the expert that a Support of 80% could be a reasonable amount to filter outliers out of the set of discovered constraints. The reported constraints consequently diminished to c.a 3000. Though high, the number of constraints shown (c.a 31 constraints per activity, on average) is significantly less than the amount of relations verified (257580, i.e., 1894 per activity). This is due to the simplification techniques adopted by MINERful, based on formulae taking into account the number of traces where the constraints were verified and the subsumptions and associations among the Declare ([73]) taxonomy of constraints.

Nonetheless, we agreed with the user to reduce the complexity of the mined process, in terms of the number of constraints. In order to do so, we simplified the process by narrowing the set of words used in the domain vocabulary (see Table 5.6).

Therefore, activities diminished and it was easier for the user to examine the discovered constraints. We recall here that declarative constraints mined by MINERful establish a connection between pairs of activities, hence the less the activities, the less the constraints. Thus, we showed the expert only those constraints whose Support exceeded that threshold.

We ran MAILOFMINE again according to the new setup, which is called E2 in

Verbs	Objects		
write	deliverable	deadline	publication
send	report	task force	proposal
submit	demo	submission	document
organize	agenda	objective	invitation
	meeting	note	cost
	draft	contribution	finance
	presentation	slide	management
	integration	release	form
	requirement	review	comment
	payment	paper	strategy
	video	commitment	periodic
	showcase	call	dow

Table 5.5. Evaluation of MAILOFMINE on a case study: the extended vocabulary

Verbs	Objects
write	deliverable
send	report
submit	demo
organize	agenda
	meeting
	draft

Table 5.6. Evaluation of MAILOFMINE on a case study: the restricted vocabulary

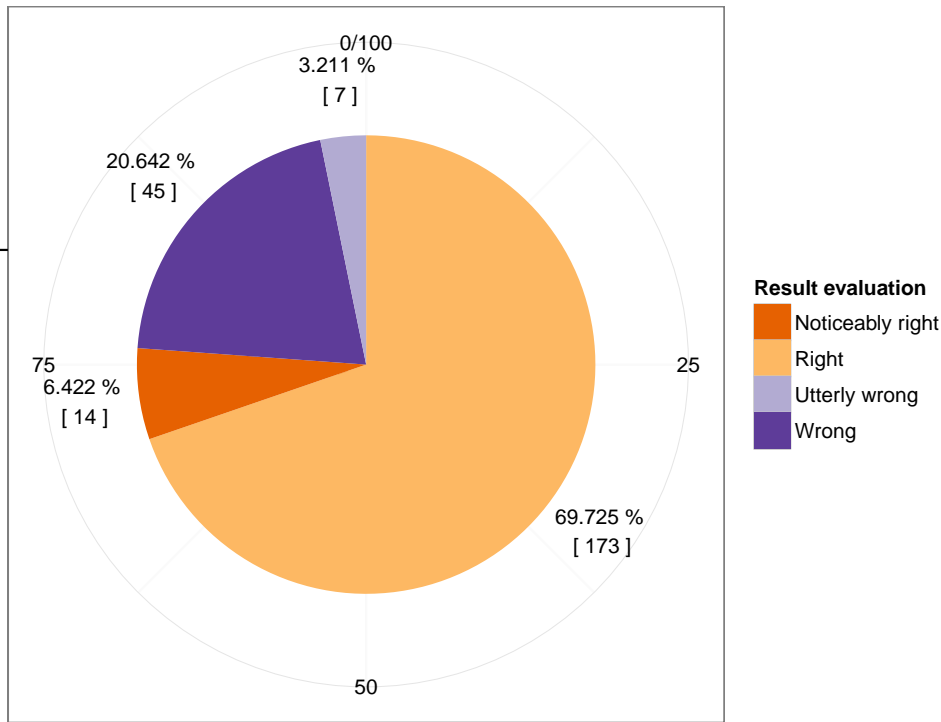


Figure 5.12. Evaluation of MAILOFMINE: appropriateness of the discovered results in the case study

Table 5.4.

The list of discovered constraints is reported in Appendix B.3. In order to assess the validity of the mined process, we checked every shown constraint with the expert. This allowed us to have a quantitative evaluation, which an “imperative” process model would not have eased.

For each constraint in the list, we asked him whether it was either: *(i)* right, i.e., it made sense with respect to his experience; *(ii)* noticeably right, i.e., it not only made sense but also suggested some unexpected mechanisms in the workflow; *(iii)* wrong, i.e., not necessarily corresponding to reality; *(iv)* utterly wrong, i.e., not corresponding to reality, unreasonable.

Luckily, the last level was assigned to few constraints (7 out of 173), a half of how many were considered noticeably right (14). The model is not known a priori, but the expert could classify as right or wrong a guessed constraint. Then, the analysis helped us find only true positives (*TP*, i.e., right or noticeably right, which in the following we will also refer to as “confirmed”) and false positives (*FP*, i.e., wrong or utterly wrong, which in the following we will also refer to as simply “wrong” altogether). It reproduces a real case, where the artful process was not formalized ever. Recalling that

$$Precision = \frac{TP}{TP + FP}$$

the algorithm was proved to obtain a *Precision* degree of 0.794 over the real case study.

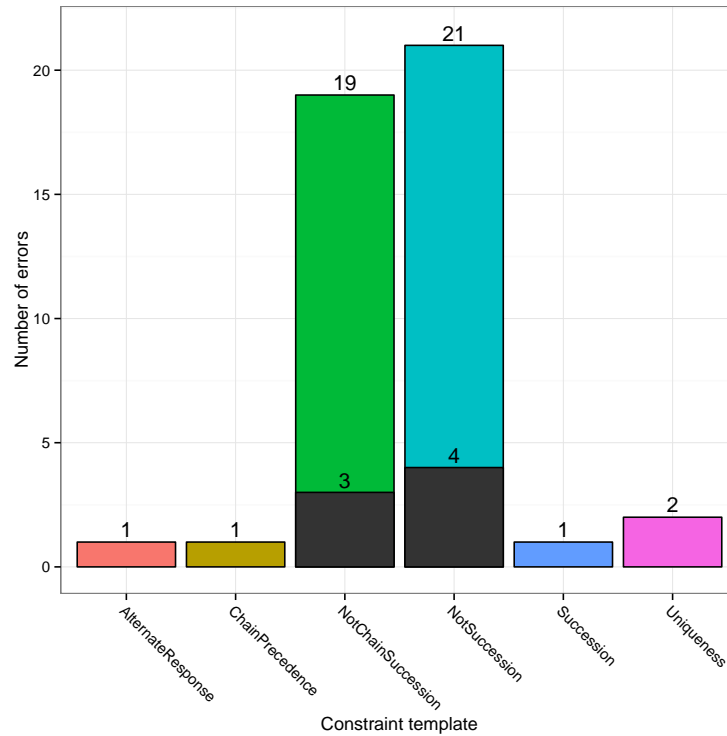


Figure 5.13. Evaluation of MAILOFMINE on a case study: errors w.r.t. constraint templates

Figure 5.12 summarizes the encouraging results of this real case study evaluation. More than 75% of the constraints inferred were compliant to a realistic model of the process. Having this qualitative measure, we performed some quantitative analyses on the gathered results.

First, we looked for a correlation between the constraint template and the number of wrong assumptions. As the reader can notice in Figure 5.13, the family of the Negative Relations seem to be the less reliable. This makes sense, since the less an activity is read in the log, the more likely the assumptions about its absence are confirmed.

Thus, we searched for a correlation between the implying activity and the amount of errors. The results are drawn on Figure 5.14.

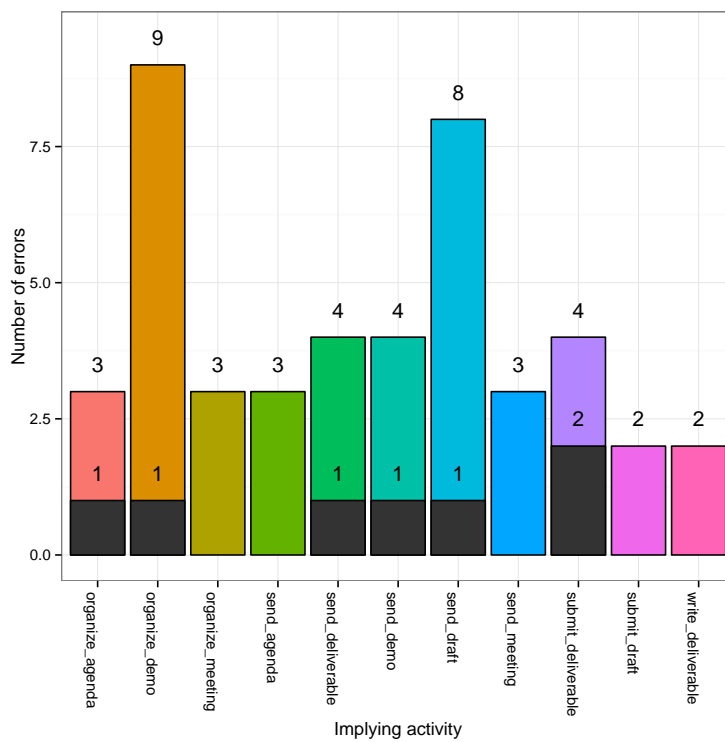
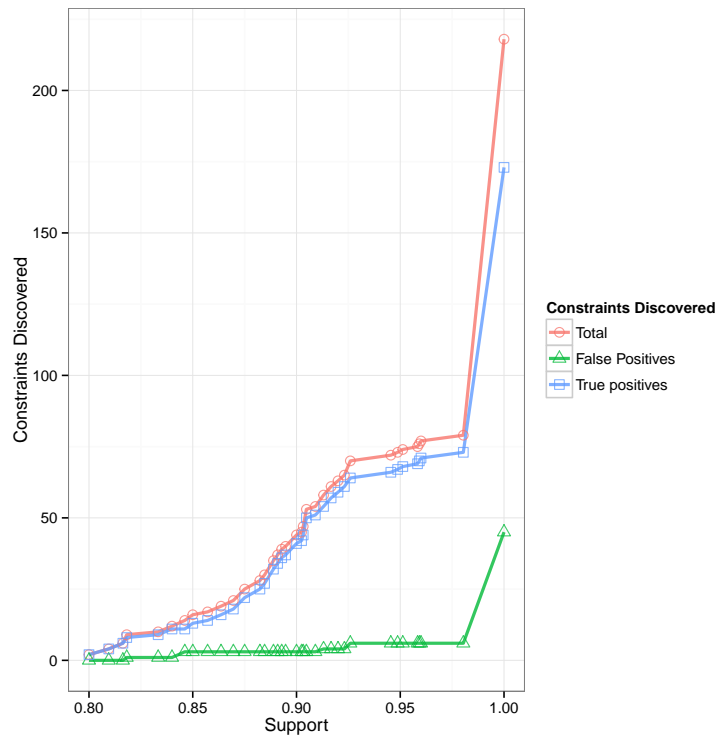
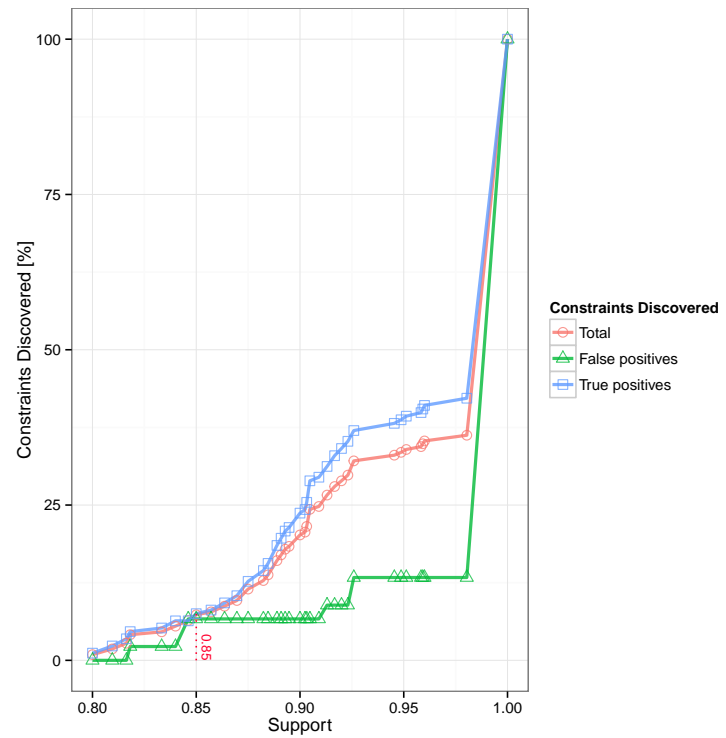


Figure 5.14. Evaluation of MAILOFMINE on a case study: errors w.r.t. implying activities

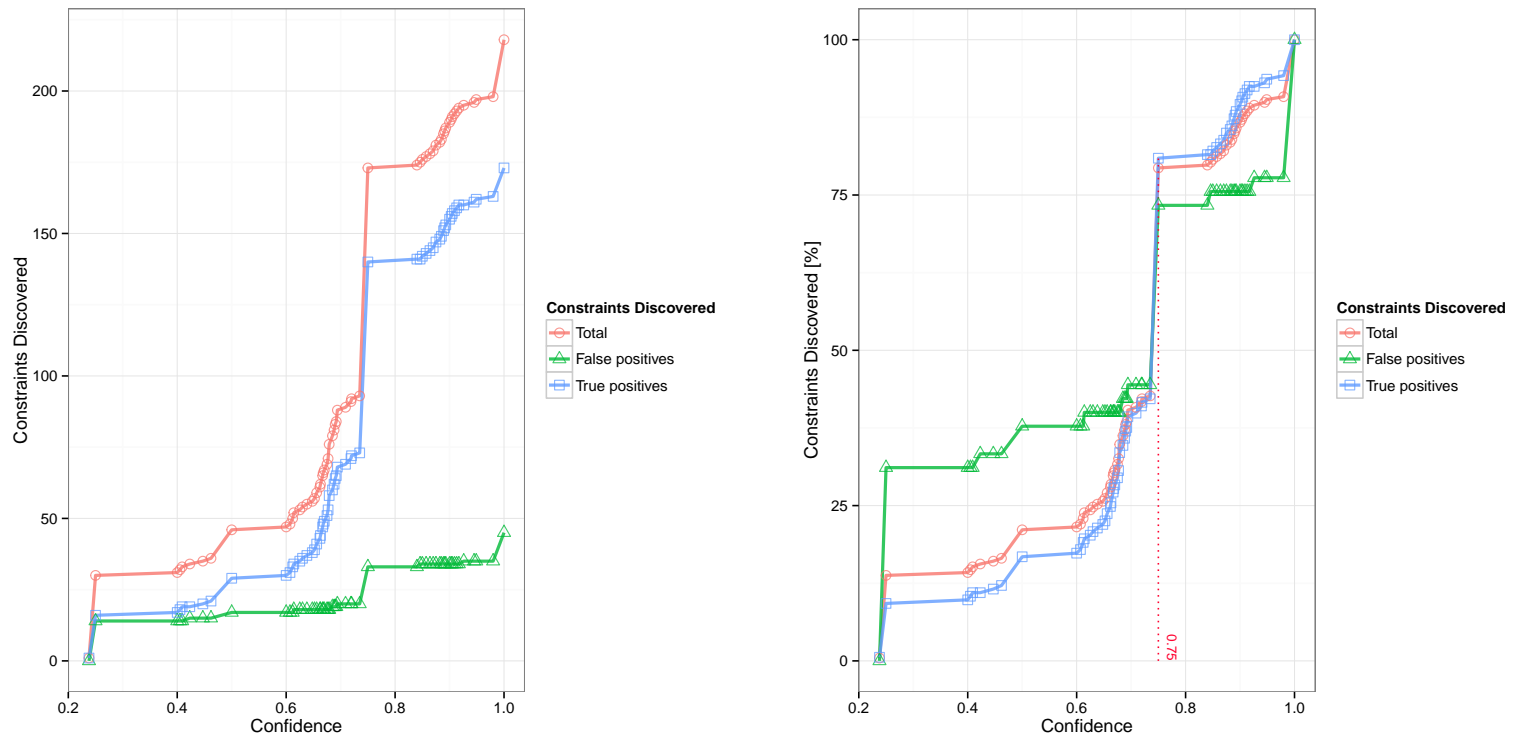


(a) The trend of the quality of the cumulative sum of constraints discovered, w.r.t. the assigned Support



(b) The trend of the quality of the cumulative sum of constraints discovered, scaled by their total amount, w.r.t. the assigned Support

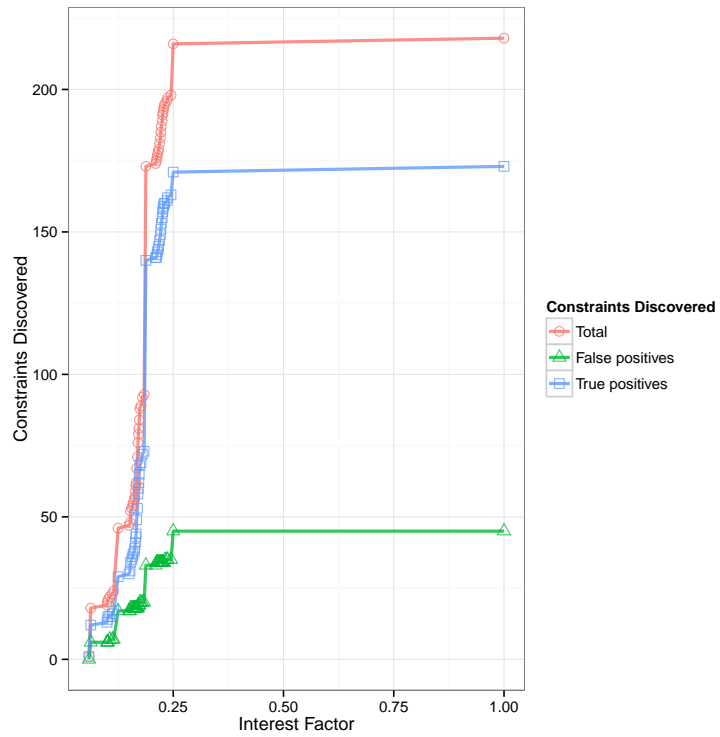
Figure 5.15. Evaluation of MAILOFMINE on a case study: trend of the quality of the process w.r.t. the Support of constraints



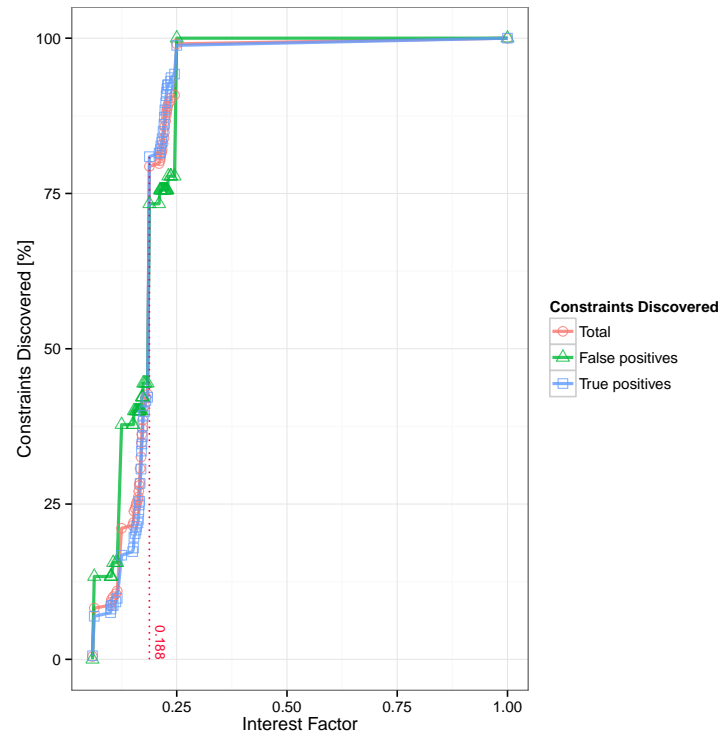
(a) The trend of the quality of the cumulative sum of constraints discovered, w.r.t. the assigned Confidence Level

(b) The trend of the quality of the cumulative sum of constraints discovered, scaled by their total amount, w.r.t. the assigned confidence level

Figure 5.16. Evaluation of MAILoFMINE on a case study: trend of the quality of the process w.r.t. the Confidence Level of constraints



(a) The trend of the quality of the cumulative sum of constraints discovered, w.r.t. the assigned Interest Factor



(b) The trend of the quality of the cumulative sum of constraints discovered, scaled by their total amount, w.r.t. the assigned Interest Factor

Figure 5.17. Evaluation of MAILOFMINE on a case study: trend of the quality of the process w.r.t. the Interest Factor of constraints

Constraint templates and activities constitute an unordered enumeration of values, whereas Support, Confidence Level and Interest Factor are numeric values. Hence, a quantitative analysis based on trends can be performed on the latter. Figures 5.15, 5.16 and 5.17 show the trend of true positives, false positives and overall (i.e., the sum of the preceding) constraints found. Respectively, such trends are calculated with respect to the Support, the Confidence Level and the Interest Factor. The quantities on the ordinates are cumulative, i.e., they represent the sum of the values which are gained up to the current value on the abscissae. Each of the aforementioned Figures contains two graphs. One considers the absolute values on the y-axis, whilst the other scales quantities with respect to the topmost, in percentage. The former is useful in that it shows how the distance between the curves rise as the value on the x-axis grows. We make use of the latter to add a marker that puts in evidence where the relative percentage of confirmed constraints overtakes the wrong, i.e., a “breakpoint” after which the rate of hits, in terms of accepted guesses, is higher than the rate of misses, in terms of wrong guesses. Filtering out the constraints falling under a value for Support (resp. Confidence Level, or Interest Factor) below the breakpoint guarantees an expectance of negligible amount of errors, compared to the correctly discovered constraints. On the ordinates, nonetheless, you see the number of verified guesses which you lose, in case. Thereby, Figure 5.15b shows that such breakpoint corresponds to a Support value of 0.85 (i.e., 25% higher than the threshold established a priori constraints to show to the user), which is little enough to limit the number of true positives below that soil to less than 10%. The same graph, although, depicts that more than 85% of errors are given a Support value of 100%. It reveals to be not very useful in practice, then. As in Figure 5.16b, the breakpoint value for Confidence corresponds to 0.75%. As drawn in Figure 5.17b, the breakpoint corresponds to an Interest Factor of 0.188. In both cases, almost half of the errors are given an Interest Factor below that soil. Sadly, almost half of the true positives are included in the same range, too.

The trend of the degree of Precision is drawn on Figure 5.18, with respect to (i) Support (Figure 5.18a), (ii) Confidence Level (Figure 5.18b) and (iii) Interest Factor (Figure 5.18c).

Again, Support (Figure 5.18a) is proven not to be a good measure for filtering misinterpretations away. The Precision indeed tends to decrease fast near to the support level of 1.0. On the other hand, deleting those constraints whose Support is equal to the maximum would be senseless. With respect to Confidence (Figure 5.18b) and Interest Factor (Figure 5.18c), the curve of Precision tends to grow monotonically.

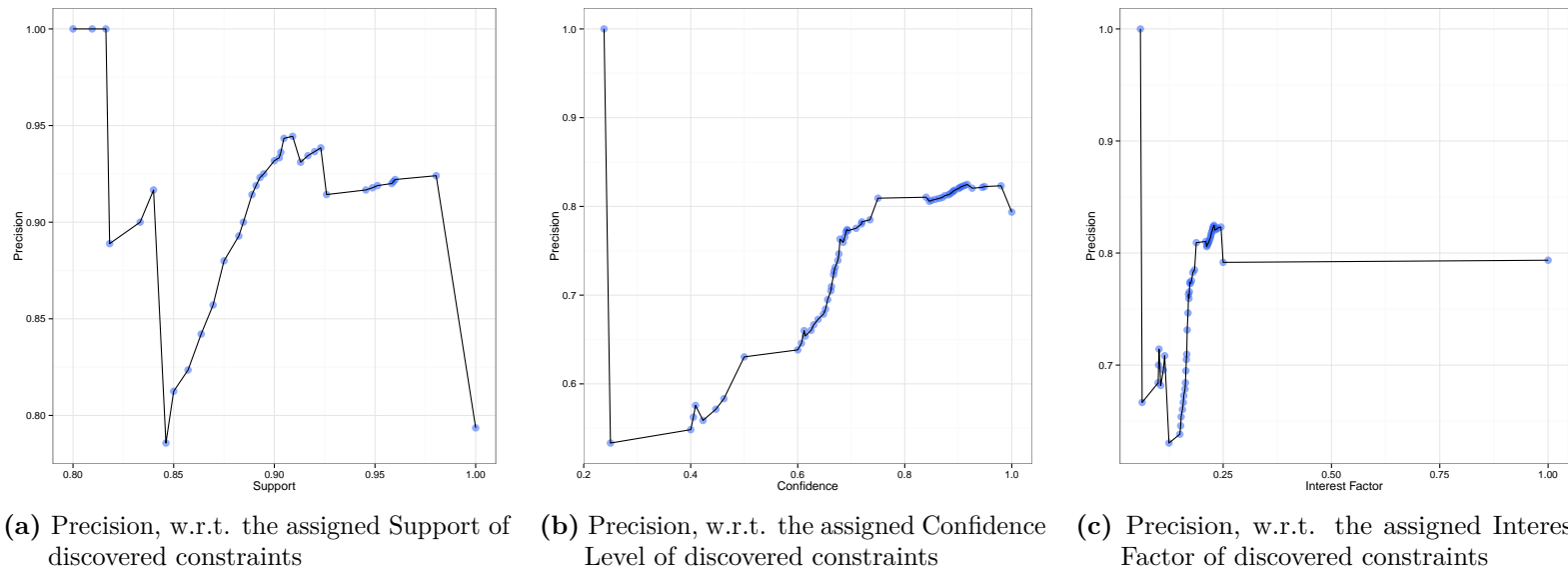
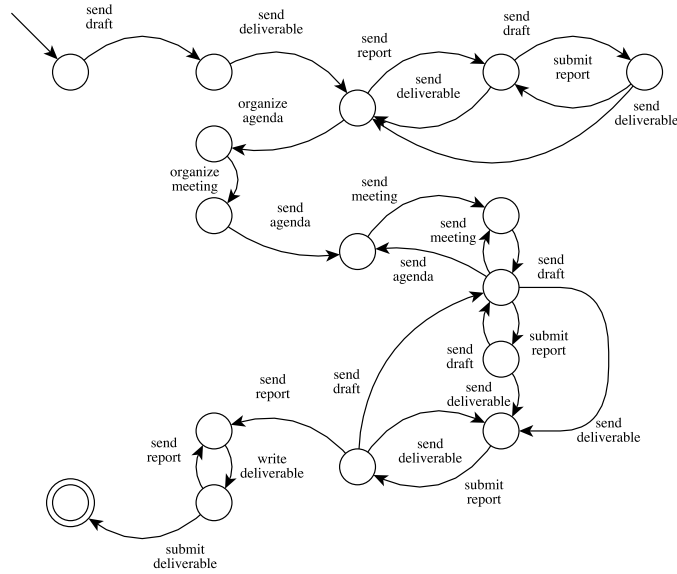


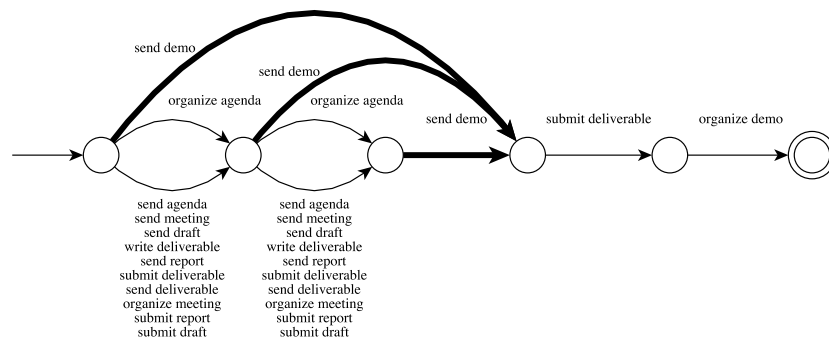
Figure 5.18. Evaluation of MAILOFMINE on a case study: trend of Precision w.r.t. the numerical assessment of discovered constraints

For sake of completeness, here we show the Finite State Automaton describing the inferred process, in Figure 5.19 (see Section 2.2.1). Such FSAs have been created according to the technique described in Section 3.4.2).

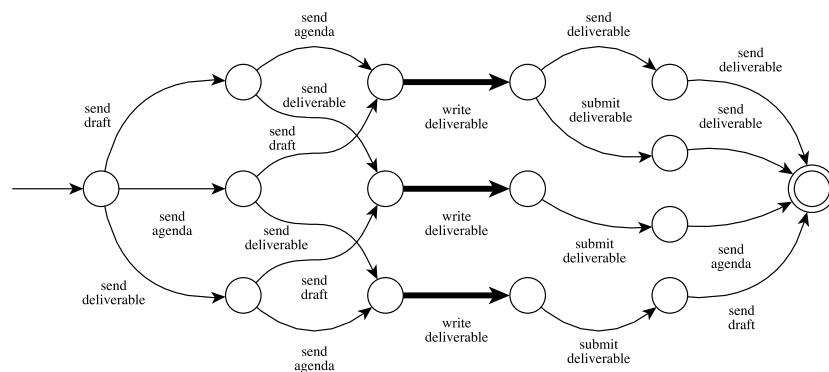
Figure 5.19a depicts the FSA representing the whole process at once, whereas Figures 5.19b and 5.19c are sub-automata, respectively related to the execution of “send demo” and “write deliverable”, preceded and followed by be the execution of two (optional) other activities. The full list of automata is provided in Appendix B.1. We recall here that sub-automata might not be correct, because they approximate the global automaton without computing it as a whole. I.e., since such sub-automata consider the constraints affecting the focused activity only, they could ignore the side-effects of those constraints which relate other activities (see the discussion in Section 3.4.2).



(a) The discovered process: a global view on the workflow, drawn as a Finite State Automaton



(b) The discovered process: a local view on the constraints constraining the “send demo” activity, drawn as a Finite State Automaton



(c) The discovered process: a local view on the constraints constraining the “write deliverable” activity, drawn as a Finite State Automaton

Figure 5.19. Evaluation of MAILOFMINE on a case study: global and local views on the discovered process, depicted as Finite State Automata

Chapter 6

Conclusions

Throughout this work, we described how we addressed the problem of discovering flexible processes out of semistructured sources of information, applied to the context of declarative models for artful processes, stemmed out of email conversations. The approach we proposed and the tool which we implemented it through is named MAILOFMINE. After an insight on the current state of the art in the fields of Information Retrieval, Process Modeling and Process Mining, which our research project concerns, we described how MAILOFMINE was designed as a modular software system, what was the expected data format and how the output was expressed. Then, we described in detail how our Process Mining algorithm, MINERful, worked. We proved its efficiency through both (i) a formal proof stating it is at most quadratic in the size of the input, (ii) experimental performance tests, over synthetic logs, (iii) comparison experiments, w.r.t. the current state-of-the-art algorithm in the field. Finally, we evaluated our tool on a real dataset of email messages. Together with an expert in the domain which the email conversations were about, we have been able to assess the quality of the discovered process.

6.1 Further development

6.1.1 Distance computing in Relation Constraints

We took a precious feedback from the expert involved in the evaluation on the real case study 5.2. He observed that several times, if there was an activity, say a , followed by another activity, say b , usually not immediately after, but after some other events, the resulting process model comprised this couple:

1. $Response(a, b)$
2. $NotChainSuccession(a, b)$

(see Appendix B.3). It suggests that b always come after a though not immediately, but how long after is it going be done? The interviewed user would have preferred to know a little more detailed information, like how many events are expected to be enacted between a and the next b . Luckily, this reasonable requirement is addressable with no modification on the MINERfulKB. We can rely on the δ function (see Section 4.1.1), which maps the number of occurrences of a searched character at a given distance from the pivot. We are currently in the process of developing such

extension to the algorithm, based on the usage of the “Student’s T distribution”, applied to the expected distance from the *pivot* to the *searched* task, given the δ function.

6.1.2 Refinement of constraints filtering

One of the key requirements for a good process model to be acceptable is its simplicity. The number of constraints shown to the user is often still too big. Indeed, we had to simplify the mined workflow by reducing the number of activities extracted from the email archive, in order to conduct the evaluation with the expert (Section 5.2). To our experience, the user gets confused and bored on checking too many rules altogether. Moreover, some false positives were signalled notwithstanding the fact they were given a maximum value for Support, i.e., equal to 100%. As we saw, Confidence and Interest Factor may help the system remove the wrong guesses, but, no lunch for free, also true positives were kept off this way. MINERful, though, was proven to be very effective in terms of time consumption and capable to return the whole set of possible constraints, along with its qualifying metrics, no matter their level. Filtering by threshold, as explained in Section 4.1.5, was an optional further step. Moreover, we were able to calculate the trend of Support in presence of errors. We may make the same with all of the metrics associated to the returned constraints, and see their trends. On the basis of such numbers, we could take advantage of Machine Learning techniques to refine the workflow returned by the mining algorithm. I.e., we could create synthetic train-and-validation sets with controlled errors within, so to automatically tune the thresholding parameters, then apply again the algorithm on the real test case, filter the result by applying the optimal parameters learnt, and verify whether they improved the returned results.

6.1.3 Uncertain logs

IR tools associate a relevance score to the indexed document when searching for an expression in the data set. We assumed to ignore such relevance score, unless it was useful to discriminate among multiple activities, when many searched terms were associated to the same document (i.e., email). Although, we believe that such score could be useful to improve the reliability of the input traces passed to MINERful. Being the nature of the analysis performed by MINERful intrinsically statistic, we might stop considering logs as collections of *sequences* of events, by linking events to a *reliability level*, indicating how likely an event read is to be the sign of an activity performed for real, or not. Thus, metrics such as Support, Confidence Level and Interest Factor for constraints could be scaled with the reliability level of the events involved in the discovery. Such a modification could be obtained by a slight modification of the knowledge base that MINERful is based upon, i.e., turning the co-domains of MINERful *interplay* and MINERful *ownplay* functions (see Section 4.1.1) from Integers to Real numbers. Such Real numbers would be calculated on the basis of the scores assigned to events by the IR tool. We still have to investigate Moreover, this extension could lead to the treatment of another challenge: contemporaneity of events, when more than one event is linked to the same email.

6.1.4 Branching Declare

Up to now, MINERful can deal with Declare constraints, though it is not able to discover Branching Declare constraints. I.e., it is not capable of determining whether a constraint hold or not between disjuncted *sets* of activities, as in the Branching Declare constraint templates. Nonetheless, we have already extended the MINERful's knowledge base so to have statistical information which, properly queried, could be address this problem as well. Roughly speaking, we have the intuition that it is enough to increment the variables currently used to store the correlation measures to sets of characters, rather than single characters, to break through this. We are currently working on this idea, since up to now we can obtain the list of constraints verified on the log along with their assigned Support, though it is still a time-consuming task. Moreover, we have to define a new strategy to remove the redundant constraints. To give a hint on it, if, e.g., $Response(a, \{b, c\})$ holds with an estimated Support equal to 1.0, the reader can see that computing the Support of $Response(a, \{b, c, d\})$ is irrelevant.

6.1.5 Biochemistry and forensics

MINERful addresses the problem of discovering declarative workflows out of logs by translating each trace in a string of single characters, each identifying a given activity, so to analyze the correlation between characters. Conversely, declarative models could be used not only for ruling the execution of activities in a process, but also for finding flexible patterns which hold in biological sequences, like in genomes and proteomes. E.g., finding whether a rule like “G always appears in the trace after A, if A is read” can be useful not only if G and A are the identifiers of “write deliverable” and “submit report”, but also if G and A stand for Guanine and Adenine in a DNA sequence. We are going to collaborate with biochemistry researchers so to check whether this intuition is valid.

Appendix A

From indicia to log

A.1 The SQL query

Listing A.1. The SQL query to create a log in tabular format

```

1 SELECT
2     WE.'word' AS 'verb',
3     WO.'word' as 'object',
4     A.'id' as 'activityId',
5     I.score, S.'text' as 'subject',
6     DATE_FORMAT(M.'dateTime', '%Y-%m-%dT%TZ') AS dateTime,
7     R.'name' AS 'archiveName'
8 FROM
9     'Indicium' I,
10    'Activity' A,
11    'VocabularyExpression' V,
12    'VocabularyObject' O,
13    'VocabularyVerb' E,
14    'VocabularyWord' WO,
15    'VocabularyWord' WE,
16    'Message' M,
17    'Subject' S,
18    'Archive' R,
19    'MessagesInArchive' MR
20 WHERE
21    I.'activityId' = A.'id'
22    AND A.'expressionId' = V.'id'
23    AND O.'id' = V.'referredObjectId'
24    AND E.'id' = V.'referredVerbId'
25    AND WO.'id' = O.'id'
26    AND WE.'id' = E.'id'
27    AND M.'id' = I.'messageId'
28    AND S.'messageId' = M.'id'
29    AND MR.'messageId' = M.'id'
30    AND MR.'archiveId' = R.'id'
31    AND A.'processId' = ?
32 AND NOT EXISTS
33 (
34     SELECT I2.'score'
35     FROM 'Indicium' I2
36     WHERE I2.'activityId' != I.'activityId'
37           AND I2.'messageId' = I.'messageId'
38           AND I2.'score' > I.'score'

```

```
39 )  
40 ORDER BY  
41 MR.'archiveId', M.'dateTime '  
42 ;
```

A.2 The XML result of the query for creating the log

Listing A.2. The XML result of the query for creating the log

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!--
3 - XML Dump
4 - version 3.3.2debiubuntu1
5 -
6 - Host: localhost
7 - Generation Time: Jan 10, 2013 at 12:14 PM
8 - Server version: 5.1.66
9 -->
10
11 <pma_xml_export version="1.0">
12   <!--
13   - Database: 'MailOfMineKB'
14   -->
15   <database name="MailOfMineKB">
16     <!-- Table Indicium -->
17     <table name="Indicium">
18       <column name="verb">send</column>
19       <column name="object">agenda</column>
20       <column name="activityId">14</column>
21       <column name="score">0.102464</column>
22       <column name="subject">telephone conference</column>
23       <column name="dateTime">2009-07-09T17:44:59Z</column>
24       <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
25     </table>
26     <table name="Indicium">
27       <column name="verb">send</column>
28       <column name="object">meeting</column>
29       <column name="activityId">18</column>
30       <column name="score">0.0329731</column>
31       <column name="subject">RE: [Sm4all] System requirements Task Force</column>
32       <column name="dateTime">2009-07-14T22:24:43Z</column>
```

```

33     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
34 </table>
35 <table name="Indicium">
36     <column name="verb">send</column>
37     <column name="object">draft</column>
38     <column name="activityId">22</column>
39     <column name="score">0.0795226</column>
40     <column name="subject">[SM4All] TF Scenarios: reminder for conf call on monday</column>
41     <column name="dateTime">2009-09-11T17:05:50Z</column>
42     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
43 </table>
44 <table name="Indicium">
45     <column name="verb">send</column>
46     <column name="object">draft</column>
47     <column name="activityId">22</column>
48     <column name="score">0.0795226</column>
49     <column name="subject">RE: [SM4All] TF Scenarios: reminder for conf call on monday</column>
50     <column name="dateTime">2009-09-14T10:21:42Z</column>
51     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
52 </table>
53 <table name="Indicium">
54     <column name="verb">send</column>
55     <column name="object">draft</column>
56     <column name="activityId">22</column>
57     <column name="score">0.0795226</column>
58     <column name="subject">[sm4all] WP4, Plans and timelines</column>
59     <column name="dateTime">2009-10-12T21:31:49Z</column>
60     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
61 </table>
62 <table name="Indicium">
63     <column name="verb">write</column>
64     <column name="object">deliverable</column>
65     <column name="activityId">1</column>
66     <column name="score">0.0459903</column>
67     <column name="subject">WP6 at the upcoming meeting</column>
68     <column name="dateTime">2010-01-12T23:16:34Z</column>
69     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>

```

```

70 </table>
71 <table name="Indicium">
72   <column name="verb">send</column>
73   <column name="object">report</column>
74   <column name="activityId">6</column>
75   <column name="score">0.0673094</column>
76   <column name="subject">[SM4ALL]: 2nd RP: Activity report</column>
77   <column name="dateTime">2010-01-13T16:00:58Z</column>
78   <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
79 </table>
80 <table name="Indicium">
81   <column name="verb">write</column>
82   <column name="object">deliverable</column>
83   <column name="activityId">1</column>
84   <column name="score">0.0536553</column>
85   <column name="subject">RE: [sm4all] D4.1 chapter example and location component</column>
86   <column name="dateTime">2010-02-01T14:23:35Z</column>
87   <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
88 </table>
89 <table name="Indicium">
90   <column name="verb">write</column>
91   <column name="object">deliverable</column>
92   <column name="activityId">1</column>
93   <column name="score">0.0459903</column>
94   <column name="subject">RE: [sm4all] D4.1 chapter example and location component</column>
95   <column name="dateTime">2010-02-01T14:54:15Z</column>
96   <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
97 </table>
98 <table name="Indicium">
99   <column name="verb">submit</column>
100  <column name="object">deliverable</column>
101  <column name="activityId">3</column>
102  <column name="score">0.141749</column>
103  <column name="subject">D6.2 final draft</column>
104  <column name="dateTime">2010-02-26T01:45:06Z</column>
105  <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
106 </table>

```

```

107 <table name="Indicium">
108   <column name="verb">send</column>
109   <column name="object">deliverable</column>
110   <column name="activityId">2</column>
111   <column name="score">0.0955151</column>
112   <column name="subject">PDFs of deliverables</column>
113   <column name="dateTime">2010-03-12T05:27:28Z</column>
114   <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
115 </table>
116 <table name="Indicium">
117   <column name="verb">write</column>
118   <column name="object">deliverable</column>
119   <column name="activityId">1</column>
120   <column name="score">0.0766504</column>
121   <column name="subject">[sm4all] Start D4.2</column>
122   <column name="dateTime">2010-04-21T05:29:47Z</column>
123   <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
124 </table>
125 <table name="Indicium">
126   <column name="verb">submit</column>
127   <column name="object">deliverable</column>
128   <column name="activityId">3</column>
129   <column name="score">0.0944994</column>
130   <column name="subject">[SM4ALL] deliverable 5.2 tentative document outline</column>
131   <column name="dateTime">2010-04-26T14:38:50Z</column>
132   <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
133 </table>
134 <table name="Indicium">
135   <column name="verb">submit</column>
136   <column name="object">deliverable</column>
137   <column name="activityId">3</column>
138   <column name="score">0.0708746</column>
139   <column name="subject">[Sm4all] [SM4ALL] deliverable 5.2 tentative document outline</column>
140   <column name="dateTime">2010-04-26T20:51:43Z</column>
141   <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
142 </table>
143 <table name="Indicium">

```

```

144     <column name="verb">submit</column>
145     <column name="object">deliverable</column>
146     <column name="activityId">3</column>
147     <column name="score">0.0590621</column>
148     <column name="subject">[Sm4all] [SM4ALL] deliverable 5.2 tentative document outline</column>
149     <column name="dateTime">2010-04-27T12:17:16Z</column>
150     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
151 </table>
152 <table name="Indicium">
153     <column name="verb">submit</column>
154     <column name="object">deliverable</column>
155     <column name="activityId">3</column>
156     <column name="score">0.0472497</column>
157     <column name="subject">RE: R: [Sm4all] [SM4ALL] deliverable 5.2 tentative document outline</column>
158     <column name="dateTime">2010-04-27T14:29:14Z</column>
159     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
160 </table>
161 <table name="Indicium">
162     <column name="verb">submit</column>
163     <column name="object">deliverable</column>
164     <column name="activityId">3</column>
165     <column name="score">0.119533</column>
166     <column name="subject">contributions to deliverable 5.2</column>
167     <column name="dateTime">2010-05-11T10:11:43Z</column>
168     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
169 </table>
170 <table name="Indicium">
171     <column name="verb">submit</column>
172     <column name="object">deliverable</column>
173     <column name="activityId">3</column>
174     <column name="score">0.0522959</column>
175     <column name="subject">contributions to deliverable 5.2</column>
176     <column name="dateTime">2010-05-11T11:35:52Z</column>
177     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
178 </table>
179 <table name="Indicium">
180     <column name="verb">organize</column>

```

```

181     <column name="object">meeting</column>
182     <column name="activityId">20</column>
183     <column name="score">0.131719</column>
184     <column name="subject">[SM4ALL]: Participants to the Innsbruck meeting</column>
185     <column name="dateTime">2011-01-11T19:20:19Z</column>
186     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
187 </table>
188 <table name="Indicium">
189     <column name="verb">organize</column>
190     <column name="object">meeting</column>
191     <column name="activityId">20</column>
192     <column name="score">0.0878127</column>
193     <column name="subject">RE: [SM4ALL]: Participants to the Innsbruck meeting</column>
194     <column name="dateTime">2011-01-17T09:51:45Z</column>
195     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
196 </table>
197 <table name="Indicium">
198     <column name="verb">send</column>
199     <column name="object">report</column>
200     <column name="activityId">6</column>
201     <column name="score">0.0673094</column>
202     <column name="subject">[SM4ALL]: 3rd RP: Activity report</column>
203     <column name="dateTime">2011-02-09T18:36:58Z</column>
204     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
205 </table>
206 <table name="Indicium">
207     <column name="verb">send</column>
208     <column name="object">meeting</column>
209     <column name="activityId">18</column>
210     <column name="score">0.0549552</column>
211     <column name="subject">RV: SM4ALL: performance tests for proxies</column>
212     <column name="dateTime">2011-02-11T08:20:25Z</column>
213     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
214 </table>
215 <table name="Indicium">
216     <column name="verb">send</column>
217     <column name="object">meeting</column>

```



```

218     <column name="activityId">18</column>
219     <column name="score">0.0688148</column>
220     <column name="subject">SM4ALL: Cost claim of the 4th period</column>
221     <column name="dateTime">2012-02-22T15:46:56Z</column>
222     <column name="archiveName">dc.claudio@gmail.com/Uniroma1/SM4All</column>
223 </table>
224 <table name="Indicium">
225     <column name="verb">send</column>
226     <column name="object">draft</column>
227     <column name="activityId">22</column>
228     <column name="score">0.0596419</column>
229     <column name="subject">RE: [Workpad-Tech] [Workpad] D5.3 - Addressing reviewers comments</column>
230     <column name="dateTime">2009-11-30T12:39:16Z</column>
231     <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
232 </table>
233 <table name="Indicium">
234     <column name="verb">send</column>
235     <column name="object">draft</column>
236     <column name="activityId">22</column>
237     <column name="score">0.0497016</column>
238     <column name="subject">RE: [Workpad-Tech] RE: [Workpad] D5.3 - Addressing reviewers comments</column>
239     <column name="dateTime">2009-11-30T13:14:26Z</column>
240     <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
241 </table>
242 <table name="Indicium">
243     <column name="verb">send</column>
244     <column name="object">draft</column>
245     <column name="activityId">22</column>
246     <column name="score">0.0397613</column>
247     <column name="subject">RE: [Workpad-Tech] [Workpad] Reminder D5.3 - Addressing reviewers comments</column>
248     <column name="dateTime">2009-12-03T14:29:55Z</column>
249     <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
250 </table>
251 <table name="Indicium">
252     <column name="verb">send</column>
253     <column name="object">deliverable</column>
254     <column name="activityId">2</column>

```

255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291

```
<column name="score">0.059697</column>
<column name="subject">[Workpad-Tech] [WORKPAD] - Reminder for missing deliverables and sentences for the letter to re
<column name="dateTime">2009-12-09T11:02:15Z</column>
<column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
</table>
<table name="Indicium">
  <column name="verb">send</column>
  <column name="object">deliverable</column>
  <column name="activityId">2</column>
  <column name="score">0.0417879</column>
  <column name="subject">[Workpad-Tech] [WORKPAD] - Reminder for missing deliverables and sentences for the letter to re
  <column name="dateTime">2009-12-09T12:32:03Z</column>
  <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
</table>
<table name="Indicium">
  <column name="verb">send</column>
  <column name="object">deliverable</column>
  <column name="activityId">2</column>
  <column name="score">0.0417879</column>
  <column name="subject">[Workpad-Tech] [WORKPAD] - Reminder for missing deliverables and sentences for the letter to re
  <column name="dateTime">2009-12-11T21:46:41Z</column>
  <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
</table>
<table name="Indicium">
  <column name="verb">submit</column>
  <column name="object">report</column>
  <column name="activityId">7</column>
  <column name="score">0.0368513</column>
  <column name="subject">RE: [WORKPAD] -- Resubmission of deliverables plus clarifications</column>
  <column name="dateTime">2010-01-12T19:06:09Z</column>
  <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
</table>
<table name="Indicium">
  <column name="verb">submit</column>
  <column name="object">report</column>
  <column name="activityId">7</column>
  <column name="score">0.0322449</column>
```

```

292     <column name="subject">RE: [WORKPAD] -- Resubmission of deliverables plus clarifications</column>
293     <column name="dateTime">2010-01-13T16:35:08Z</column>
294     <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
295 </table>
296 <table name="Indicium">
297     <column name="verb">submit</column>
298     <column name="object">report</column>
299     <column name="activityId">7</column>
300     <column name="score">0.0460642</column>
301     <column name="subject">[WORKPAD] -- Resubmission of deliverables plus clarifications</column>
302     <column name="dateTime">2010-01-14T15:01:09Z</column>
303     <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
304 </table>
305 <table name="Indicium">
306     <column name="verb">submit</column>
307     <column name="object">report</column>
308     <column name="activityId">7</column>
309     <column name="score">0.0460642</column>
310     <column name="subject">RE: [WORKPAD] Status of the documents</column>
311     <column name="dateTime">2010-02-04T10:40:04Z</column>
312     <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
313 </table>
314 <table name="Indicium">
315     <column name="verb">submit</column>
316     <column name="object">report</column>
317     <column name="activityId">7</column>
318     <column name="score">0.130289</column>
319     <column name="subject">[Fwd: Final report on the distribution of the Union financial
contribution]</column>
320     <column name="dateTime">2010-09-22T11:01:06Z</column>
321     <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
322 </table>
323 <table name="Indicium">
324     <column name="verb">submit</column>
325     <column name="object">report</column>
326     <column name="activityId">7</column>
327     <column name="score">0.104231</column>

```

```

328         <column name="subject">[Fwd: Final report on the distribution of the Union financial
contribution]</column>
329         <column name="dateTime">2010-09-22T11:07:55Z</column>
330         <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
331     </table>
332     <table name="Indicium">
333         <column name="verb">submit</column>
334         <column name="object">report</column>
335         <column name="activityId">7</column>
336         <column name="score">0.104231</column>
337         <column name="subject">[Fwd: Final report on the distribution of the Union financial
contribution]</column>
338         <column name="dateTime">2010-09-22T12:37:01Z</column>
339         <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
340     </table>
341     <table name="Indicium">
342         <column name="verb">submit</column>
343         <column name="object">report</column>
344         <column name="activityId">7</column>
345         <column name="score">0.104231</column>
346         <column name="subject">Final report on the distribution of the Union financial contribution</column>
347         <column name="dateTime">2010-10-04T13:09:53Z</column>
348         <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
349     </table>
350     <table name="Indicium">
351         <column name="verb">send</column>
352         <column name="object">report</column>
353         <column name="activityId">6</column>
354         <column name="score">0.0673094</column>
355         <column name="subject">WORKPAD www site</column>
356         <column name="dateTime">2010-10-20T21:35:54Z</column>
357         <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
358     </table>
359     <table name="Indicium">
360         <column name="verb">send</column>
361         <column name="object">report</column>
362         <column name="activityId">6</column>

```

```

363     <column name="score">0.0471166</column>
364     <column name="subject">RE: WORKPAD www site</column>
365     <column name="dateTime">2010-10-22T10:37:45Z</column>
366     <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
367 </table>
368 <table name="Indicium">
369     <column name="verb">send</column>
370     <column name="object">report</column>
371     <column name="activityId">6</column>
372     <column name="score">0.0842847</column>
373     <column name="subject">Workpad - annual report for third year</column>
374     <column name="dateTime">2012-02-20T10:55:31Z</column>
375     <column name="archiveName">mecella@dis.uniroma1.it/FP6/WORKPAD</column>
376 </table>
377 <table name="Indicium">
378     <column name="verb">submit</column>
379     <column name="object">report</column>
380     <column name="activityId">7</column>
381     <column name="score">0.041201</column>
382     <column name="subject">RE: [SM4ALL]: Declaration on actual costs</column>
383     <column name="dateTime">2009-10-09T12:14:34Z</column>
384     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
385 </table>
386 <table name="Indicium">
387     <column name="verb">submit</column>
388     <column name="object">report</column>
389     <column name="activityId">7</column>
390     <column name="score">0.041201</column>
391     <column name="subject">SM4ALL : payment Period nÃř1 - Confirmation ...</column>
392     <column name="dateTime">2009-10-09T17:38:30Z</column>
393     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
394 </table>
395 <table name="Indicium">
396     <column name="verb">send</column>
397     <column name="object">draft</column>
398     <column name="activityId">22</column>
399     <column name="score">0.0795226</column>

```

```

400     <column name="subject">[sm4all] WP4, Plans and timelines</column>
401     <column name="dateTime">2009-10-12T21:31:49Z</column>
402     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
403 </table>
404 <table name="Indicium">
405     <column name="verb">send</column>
406     <column name="object">report</column>
407     <column name="activityId">6</column>
408     <column name="score">0.0403856</column>
409     <column name="subject">RE: [SM4All] - Agenda of the Wien meeting</column>
410     <column name="dateTime">2009-10-14T10:12:59Z</column>
411     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
412 </table>
413 <table name="Indicium">
414     <column name="verb">send</column>
415     <column name="object">draft</column>
416     <column name="activityId">22</column>
417     <column name="score">0.0754417</column>
418     <column name="subject">Precondition and effect language specification</column>
419     <column name="dateTime">2009-10-14T10:34:14Z</column>
420     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
421 </table>
422 <table name="Indicium">
423     <column name="verb">submit</column>
424     <column name="object">report</column>
425     <column name="activityId">7</column>
426     <column name="score">0.055277</column>
427     <column name="subject">RE: &quot;Green light&quot; from TID (on behalf of Massimo)</column>
428     <column name="dateTime">2009-10-15T11:54:49Z</column>
429     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
430 </table>
431 <table name="Indicium">
432     <column name="verb">submit</column>
433     <column name="object">report</column>
434     <column name="activityId">7</column>
435     <column name="score">0.0460642</column>
436     <column name="subject">RE: &quot;Green light&quot; from TID (on behalf of Massimo)</column>

```

```

437         <column name="dateTime">2009-10-15T12:16:07Z</column>
438         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4A11</column>
439     </table>
440     <table name="Indicium">
441         <column name="verb">submit</column>
442         <column name="object">draft</column>
443         <column name="activityId">23</column>
444         <column name="score">0.175298</column>
445         <column name="subject">caise'10</column>
446         <column name="dateTime">2009-11-25T18:57:32Z</column>
447         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4A11</column>
448     </table>
449     <table name="Indicium">
450         <column name="verb">write</column>
451         <column name="object">deliverable</column>
452         <column name="activityId">1</column>
453         <column name="score">0.0459903</column>
454         <column name="subject">WP6 at the upcoming meeting</column>
455         <column name="dateTime">2010-01-12T23:16:34Z</column>
456         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4A11</column>
457     </table>
458     <table name="Indicium">
459         <column name="verb">send</column>
460         <column name="object">report</column>
461         <column name="activityId">6</column>
462         <column name="score">0.0673094</column>
463         <column name="subject">[SM4ALL]: 2nd RP: Activity report</column>
464         <column name="dateTime">2010-01-13T16:00:58Z</column>
465         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4A11</column>
466     </table>
467     <table name="Indicium">
468         <column name="verb">write</column>
469         <column name="object">deliverable</column>
470         <column name="activityId">1</column>
471         <column name="score">0.0342791</column>
472         <column name="subject">Personal comments</column>
473         <column name="dateTime">2010-01-28T20:15:41Z</column>

```

```

474         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
475 </table>
476 <table name="Indicium">
477     <column name="verb">write</column>
478     <column name="object">deliverable</column>
479     <column name="activityId">1</column>
480     <column name="score">0.0536553</column>
481     <column name="subject">RE: [sm4all] D4.1 chapter example and location component</column>
482     <column name="dateTime">2010-02-01T14:23:35Z</column>
483     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
484 </table>
485 <table name="Indicium">
486     <column name="verb">write</column>
487     <column name="object">deliverable</column>
488     <column name="activityId">1</column>
489     <column name="score">0.0459903</column>
490     <column name="subject">RE: [sm4all] D4.1 chapter example and location component</column>
491     <column name="dateTime">2010-02-01T14:54:15Z</column>
492     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
493 </table>
494 <table name="Indicium">
495     <column name="verb">send</column>
496     <column name="object">report</column>
497     <column name="activityId">6</column>
498     <column name="score">0.0673094</column>
499     <column name="subject">[Fwd: letter]</column>
500     <column name="dateTime">2010-02-04T10:46:18Z</column>
501     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
502 </table>
503 <table name="Indicium">
504     <column name="verb">submit</column>
505     <column name="object">report</column>
506     <column name="activityId">7</column>
507     <column name="score">0.106381</column>
508     <column name="subject">SM4ALL - 2nd Financial report</column>
509     <column name="dateTime">2010-02-17T17:24:36Z</column>
510     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>

```



```

511 </table>
512 <table name="Indicium">
513   <column name="verb">submit</column>
514   <column name="object">report</column>
515   <column name="activityId">7</column>
516   <column name="score">0.130289</column>
517   <column name="subject">TID financial report</column>
518   <column name="dateTime">2010-02-25T12:46:25Z</column>
519   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
520 </table>
521 <table name="Indicium">
522   <column name="verb">submit</column>
523   <column name="object">report</column>
524   <column name="activityId">7</column>
525   <column name="score">0.182405</column>
526   <column name="subject">SM4ALL - Re: TID financial report</column>
527   <column name="dateTime">2010-02-25T15:56:24Z</column>
528   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
529 </table>
530 <table name="Indicium">
531   <column name="verb">send</column>
532   <column name="object">draft</column>
533   <column name="activityId">22</column>
534   <column name="score">0.119284</column>
535   <column name="subject">SM4ALL - Re: TID financial report</column>
536   <column name="dateTime">2010-02-25T16:47:23Z</column>
537   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
538 </table>
539 <table name="Indicium">
540   <column name="verb">submit</column>
541   <column name="object">report</column>
542   <column name="activityId">7</column>
543   <column name="score">0.156347</column>
544   <column name="subject">SM4ALL - TID financial report</column>
545   <column name="dateTime">2010-03-02T16:20:31Z</column>
546   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
547 </table>

```

```
548 <table name="Indicium">
549   <column name="verb">submit</column>
550   <column name="object">draft</column>
551   <column name="activityId">23</column>
552   <column name="score">0.107348</column>
553   <column name="subject">SM4ALL:</column>
554   <column name="dateTime">2010-03-08T16:48:58Z</column>
555   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
556 </table>
557 <table name="Indicium">
558   <column name="verb">submit</column>
559   <column name="object">report</column>
560   <column name="activityId">7</column>
561   <column name="score">0.127657</column>
562   <column name="subject">SM4ALL - TID financial report</column>
563   <column name="dateTime">2010-03-10T15:42:22Z</column>
564   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
565 </table>
566 <table name="Indicium">
567   <column name="verb">submit</column>
568   <column name="object">report</column>
569   <column name="activityId">7</column>
570   <column name="score">0.0841013</column>
571   <column name="subject">RE: SM4ALL - TID financial report</column>
572   <column name="dateTime">2010-03-10T15:55:47Z</column>
573   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
574 </table>
575 <table name="Indicium">
576   <column name="verb">submit</column>
577   <column name="object">report</column>
578   <column name="activityId">7</column>
579   <column name="score">0.067281</column>
580   <column name="subject">RE: SM4ALL - TID financial report</column>
581   <column name="dateTime">2010-03-11T12:43:18Z</column>
582   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
583 </table>
584 <table name="Indicium">
```

```

585         <column name="verb">submit</column>
586         <column name="object">report</column>
587         <column name="activityId">7</column>
588         <column name="score">0.067281</column>
589         <column name="subject">[Fwd: RE: SM4ALL - TID financial report]</column>
590         <column name="dateTime">2010-03-11T19:06:56Z</column>
591         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
592     </table>
593     <table name="Indicium">
594         <column name="verb">send</column>
595         <column name="object">deliverable</column>
596         <column name="activityId">2</column>
597         <column name="score">0.0955151</column>
598         <column name="subject">PDFs of deliverables</column>
599         <column name="dateTime">2010-03-12T05:27:28Z</column>
600         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
601     </table>
602     <table name="Indicium">
603         <column name="verb">submit</column>
604         <column name="object">report</column>
605         <column name="activityId">7</column>
606         <column name="score">0.067281</column>
607         <column name="subject">RE: [Fwd: RE: SM4ALL - TID financial report]</column>
608         <column name="dateTime">2010-03-12T10:02:47Z</column>
609         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
610     </table>
611     <table name="Indicium">
612         <column name="verb">send</column>
613         <column name="object">deliverable</column>
614         <column name="activityId">2</column>
615         <column name="score">0.0716364</column>
616         <column name="subject">PDFs of deliverables</column>
617         <column name="dateTime">2010-03-12T11:41:55Z</column>
618         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
619     </table>
620     <table name="Indicium">
621         <column name="verb">send</column>

```

```

622     <column name="object">agenda</column>
623     <column name="activityId">14</column>
624     <column name="score">0.119542</column>
625     <column name="subject">Stockholm meeting</column>
626     <column name="dateTime">2010-04-06T17:41:57Z</column>
627     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
628 </table>
629 <table name="Indicium">
630     <column name="verb">send</column>
631     <column name="object">agenda</column>
632     <column name="activityId">14</column>
633     <column name="score">0.085387</column>
634     <column name="subject">Stockholm meeting</column>
635     <column name="dateTime">2010-04-09T14:12:39Z</column>
636     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
637 </table>
638 <table name="Indicium">
639     <column name="verb">write</column>
640     <column name="object">deliverable</column>
641     <column name="activityId">1</column>
642     <column name="score">0.0766504</column>
643     <column name="subject">[sm4all] Start D4.2</column>
644     <column name="dateTime">2010-04-21T05:29:47Z</column>
645     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
646 </table>
647 <table name="Indicium">
648     <column name="verb">send</column>
649     <column name="object">demo</column>
650     <column name="activityId">10</column>
651     <column name="score">0.0728515</column>
652     <column name="subject">Invitation to participate in a special Poster & Demo Session at IEEE SECON '10, 21-25 June, 2010,
653     <column name="dateTime">2010-04-23T18:05:03Z</column>
654     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
655 </table>
656 <table name="Indicium">
657     <column name="verb">submit</column>
658     <column name="object">deliverable</column>

```

```

659     <column name="activityId">3</column>
660     <column name="score">0.0944994</column>
661     <column name="subject">[SM4ALL] deliverable 5.2 tentative document outline</column>
662     <column name="dateTime">2010-04-26T14:38:50Z</column>
663     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
664 </table>
665 <table name="Indicium">
666     <column name="verb">submit</column>
667     <column name="object">deliverable</column>
668     <column name="activityId">3</column>
669     <column name="score">0.0708746</column>
670     <column name="subject">[Sm4all] [SM4ALL] deliverable 5.2 tentative document outline</column>
671     <column name="dateTime">2010-04-26T20:51:43Z</column>
672     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
673 </table>
674 <table name="Indicium">
675     <column name="verb">submit</column>
676     <column name="object">deliverable</column>
677     <column name="activityId">3</column>
678     <column name="score">0.0590621</column>
679     <column name="subject">[Sm4all] [SM4ALL] deliverable 5.2 tentative document outline</column>
680     <column name="dateTime">2010-04-27T12:17:16Z</column>
681     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
682 </table>
683 <table name="Indicium">
684     <column name="verb">submit</column>
685     <column name="object">deliverable</column>
686     <column name="activityId">3</column>
687     <column name="score">0.0472497</column>
688     <column name="subject">RE: R: [Sm4all] [SM4ALL] deliverable 5.2 tentative document outline</column>
689     <column name="dateTime">2010-04-27T14:29:14Z</column>
690     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
691 </table>
692 <table name="Indicium">
693     <column name="verb">submit</column>
694     <column name="object">deliverable</column>
695     <column name="activityId">3</column>

```

```

696         <column name="score">0.119533</column>
697         <column name="subject">contributions to deliverable 5.2</column>
698         <column name="dateTime">2010-05-11T10:11:43Z</column>
699         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
700     </table>
701     <table name="Indicium">
702         <column name="verb">submit</column>
703         <column name="object">deliverable</column>
704         <column name="activityId">3</column>
705         <column name="score">0.0522959</column>
706         <column name="subject">contributions to deliverable 5.2</column>
707         <column name="dateTime">2010-05-11T11:35:52Z</column>
708         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
709     </table>
710     <table name="Indicium">
711         <column name="verb">submit</column>
712         <column name="object">report</column>
713         <column name="activityId">7</column>
714         <column name="score">0.182405</column>
715         <column name="subject">SM4ALL: question about costs</column>
716         <column name="dateTime">2010-06-30T11:53:34Z</column>
717         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
718     </table>
719     <table name="Indicium">
720         <column name="verb">submit</column>
721         <column name="object">report</column>
722         <column name="activityId">7</column>
723         <column name="score">0.104231</column>
724         <column name="subject">RE: SM4ALL: question about costs</column>
725         <column name="dateTime">2010-07-20T15:45:19Z</column>
726         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
727     </table>
728     <table name="Indicium">
729         <column name="verb">submit</column>
730         <column name="object">report</column>
731         <column name="activityId">7</column>
732         <column name="score">0.0781735</column>

```

```

733     <column name="subject">[Fwd: RE: SM4ALL: question about costs]</column>
734     <column name="dateTime">2010-07-20T18:55:49Z</column>
735     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
736 </table>
737 <table name="Indicium">
738     <column name="verb">send</column>
739     <column name="object">deliverable</column>
740     <column name="activityId">2</column>
741     <column name="score">0.167151</column>
742     <column name="subject">Software Location in deliverables</column>
743     <column name="dateTime">2010-09-09T15:00:28Z</column>
744     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
745 </table>
746 <table name="Indicium">
747     <column name="verb">organize</column>
748     <column name="object">agenda</column>
749     <column name="activityId">16</column>
750     <column name="score">0.14481</column>
751     <column name="subject">URGENT INVITATION: Meeting of the Monitoring and Control Cluster on SMART BUILDING /SMART SPACE
752     <column name="dateTime">2010-10-29T10:14:49Z</column>
753     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
754 </table>
755 <table name="Indicium">
756     <column name="verb">organize</column>
757     <column name="object">meeting</column>
758     <column name="activityId">20</column>
759     <column name="score">0.131719</column>
760     <column name="subject">[SM4ALL]: Participants to the Innsbruck meeting</column>
761     <column name="dateTime">2011-01-11T19:20:19Z</column>
762     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
763 </table>
764 <table name="Indicium">
765     <column name="verb">organize</column>
766     <column name="object">meeting</column>
767     <column name="activityId">20</column>
768     <column name="score">0.0878127</column>
769     <column name="subject">RE: [SM4ALL]: Participants to the Innsbruck meeting</column>

```

```

770         <column name="dateTime">2011-01-17T09:51:45Z</column>
771         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
772     </table>
773     <table name="Indicium">
774         <column name="verb">send</column>
775         <column name="object">draft</column>
776         <column name="activityId">22</column>
777         <column name="score">0.0973948</column>
778         <column name="subject">3rd and 4th SM4ALL reviews confirmed!</column>
779         <column name="dateTime">2011-01-17T10:48:06Z</column>
780         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
781     </table>
782     <table name="Indicium">
783         <column name="verb">send</column>
784         <column name="object">agenda</column>
785         <column name="activityId">14</column>
786         <column name="score">0.102464</column>
787         <column name="subject">3rd and 4th SM4ALL reviews confirmed!</column>
788         <column name="dateTime">2011-01-17T11:13:07Z</column>
789         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
790     </table>
791     <table name="Indicium">
792         <column name="verb">send</column>
793         <column name="object">agenda</column>
794         <column name="activityId">14</column>
795         <column name="score">0.085387</column>
796         <column name="subject">RE: 3rd and 4th SM4ALL reviews confirmed!</column>
797         <column name="dateTime">2011-01-17T11:13:53Z</column>
798         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
799     </table>
800     <table name="Indicium">
801         <column name="verb">organize</column>
802         <column name="object">demo</column>
803         <column name="activityId">12</column>
804         <column name="score">0.0774897</column>
805         <column name="subject">URGENT INVITATION: Poster and Demo Session at EWSN 11, 23-25 February, Bonn, Germany</column>
806         <column name="dateTime">2011-01-31T10:52:34Z</column>

```



```

807         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
808     </table>
809     <table name="Indicium">
810         <column name="verb">send</column>
811         <column name="object">meeting</column>
812         <column name="activityId">18</column>
813         <column name="score">0.0659463</column>
814         <column name="subject">SM4ALL: performance tests for proxies</column>
815         <column name="dateTime">2011-02-02T15:45:21Z</column>
816         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
817     </table>
818     <table name="Indicium">
819         <column name="verb">send</column>
820         <column name="object">meeting</column>
821         <column name="activityId">18</column>
822         <column name="score">0.0439642</column>
823         <column name="subject">SM4All FOI concerns</column>
824         <column name="dateTime">2011-05-17T14:56:30Z</column>
825         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
826     </table>
827     <table name="Indicium">
828         <column name="verb">send</column>
829         <column name="object">draft</column>
830         <column name="activityId">22</column>
831         <column name="score">0.162325</column>
832         <column name="subject">Agenda for the final review of SM4ALL in Rome</column>
833         <column name="dateTime">2011-07-29T16:43:37Z</column>
834         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
835     </table>
836     <table name="Indicium">
837         <column name="verb">send</column>
838         <column name="object">agenda</column>
839         <column name="activityId">14</column>
840         <column name="score">0.102464</column>
841         <column name="subject">Agenda of the SM4All final review and meeting for the book</column>
842         <column name="dateTime">2011-09-21T12:14:46Z</column>
843         <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>

```

```

844 </table>
845 <table name="Indicium">
846   <column name="verb">send</column>
847   <column name="object">meeting</column>
848   <column name="activityId">18</column>
849   <column name="score">0.0688148</column>
850   <column name="subject">SM4ALL: Cost claim of the 4th period</column>
851   <column name="dateTime">2012-02-22T15:46:56Z</column>
852   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
853 </table>
854 <table name="Indicium">
855   <column name="verb">send</column>
856   <column name="object">meeting</column>
857   <column name="activityId">18</column>
858   <column name="score">0.0491534</column>
859   <column name="subject">Status update and requests</column>
860   <column name="dateTime">2012-03-13T12:10:18Z</column>
861   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
862 </table>
863 <table name="Indicium">
864   <column name="verb">send</column>
865   <column name="object">draft</column>
866   <column name="activityId">22</column>
867   <column name="score">0.0843464</column>
868   <column name="subject">Call for paper :)</column>
869   <column name="dateTime">2012-03-19T13:28:59Z</column>
870   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
871 </table>
872 <table name="Indicium">
873   <column name="verb">send</column>
874   <column name="object">meeting</column>
875   <column name="activityId">18</column>
876   <column name="score">0.0294921</column>
877   <column name="subject">RE: URGENT for SM4All (please react) -- Status update and requests</column>
878   <column name="dateTime">2012-03-27T10:54:02Z</column>
879   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
880 </table>

```

```

881 <table name="Indicium">
882   <column name="verb">send</column>
883   <column name="object">meeting</column>
884   <column name="activityId">18</column>
885   <column name="score">0.0245767</column>
886   <column name="subject">RE: [Sm4all] URGENT for SM4All (please react) -- Status update and requests</column>
887   <column name="dateTime">2012-03-28T16:22:43Z</column>
888   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
889 </table>
890 <table name="Indicium">
891   <column name="verb">send</column>
892   <column name="object">meeting</column>
893   <column name="activityId">18</column>
894   <column name="score">0.0196614</column>
895   <column name="subject">RE: [Sm4all] URGENT for SM4All (please react) -- Status update and requests</column>
896   <column name="dateTime">2012-03-28T16:48:32Z</column>
897   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
898 </table>
899 <table name="Indicium">
900   <column name="verb">send</column>
901   <column name="object">report</column>
902   <column name="activityId">6</column>
903   <column name="score">0.0235583</column>
904   <column name="subject">VB: Fwd: SM4All (224332) : Periodic Report and Cost Claim submission in NEF</column>
905   <column name="dateTime">2012-07-02T13:35:28Z</column>
906   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
907 </table>
908 <table name="Indicium">
909   <column name="verb">send</column>
910   <column name="object">deliverable</column>
911   <column name="activityId">2</column>
912   <column name="score">0.168848</column>
913   <column name="subject">[Sm4all] RED CARD: the missing deliverable D8.1.d</column>
914   <column name="dateTime">2012-07-05T10:46:45Z</column>
915   <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
916 </table>
917 <table name="Indicium">

```

```

918     <column name="verb">send</column>
919     <column name="object">deliverable</column>
920     <column name="activityId">2</column>
921     <column name="score">0.0506546</column>
922     <column name="subject">RE: [Sm4all] RED CARD: the missing deliverable D8.1.d</column>
923     <column name="dateTime">2012-07-06T12:15:31Z</column>
924     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
925 </table>
926 <table name="Indicium">
927     <column name="verb">send</column>
928     <column name="object">deliverable</column>
929     <column name="activityId">2</column>
930     <column name="score">0.0506546</column>
931     <column name="subject">RE: [Sm4all] RED CARD: the missing deliverable D8.1.d</column>
932     <column name="dateTime">2012-07-10T12:22:14Z</column>
933     <column name="archiveName">mecella@dis.uniroma1.it/FP7/SM4All</column>
934 </table>
935 <table name="Indicium">
936     <column name="verb">organize</column>
937     <column name="object">agenda</column>
938     <column name="activityId">16</column>
939     <column name="score">0.261674</column>
940     <column name="subject">Confirmation of next GB meeting</column>
941     <column name="dateTime">2010-10-24T23:53:41Z</column>
942     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
943 </table>
944 <table name="Indicium">
945     <column name="verb">send</column>
946     <column name="object">draft</column>
947     <column name="activityId">22</column>
948     <column name="score">0.0994032</column>
949     <column name="subject">[GreenerBuildings][D1.3] First internal deadline towards D1.3</column>
950     <column name="dateTime">2010-10-29T14:45:28Z</column>
951     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
952 </table>
953 <table name="Indicium">
954     <column name="verb">send</column>

```

```

955     <column name="object">meeting</column>
956     <column name="activityId">18</column>
957     <column name="score">0.117968</column>
958     <column name="subject">GreenerBuildings: draft meeting agenda</column>
959     <column name="dateTime">2010-12-01T21:26:57Z</column>
960     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
961 </table>
962 <table name="Indicium">
963     <column name="verb">write</column>
964     <column name="object">deliverable</column>
965     <column name="activityId">1</column>
966     <column name="score">0.0685582</column>
967     <column name="subject">[GreenerBuildings Tech] [D1.1] SOA review - initiative distribution</column>
968     <column name="dateTime">2011-01-25T00:05:52Z</column>
969     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
970 </table>
971 <table name="Indicium">
972     <column name="verb">send</column>
973     <column name="object">deliverable</column>
974     <column name="activityId">2</column>
975     <column name="score">0.119394</column>
976     <column name="subject">[GreenerBuildings Tech] [D1.1] Author list</column>
977     <column name="dateTime">2011-03-04T09:25:34Z</column>
978     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
979 </table>
980 <table name="Indicium">
981     <column name="verb">organize</column>
982     <column name="object">agenda</column>
983     <column name="activityId">16</column>
984     <column name="score">0.114482</column>
985     <column name="subject">RE: [GreenerBuildings Tech] Data for teleconference ...</column>
986     <column name="dateTime">2011-05-17T11:08:35Z</column>
987     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
988 </table>
989 <table name="Indicium">
990     <column name="verb">send</column>
991     <column name="object">agenda</column>

```

```

992         <column name="activityId">14</column>
993         <column name="score">0.0925967</column>
994         <column name="subject">[GreenerBuildings Admin] Rome meeting agenda planning</column>
995         <column name="dateTime">2011-05-22T19:16:49Z</column>
996         <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
997     </table>
998     <table name="Indicium">
999         <column name="verb">send</column>
1000         <column name="object">meeting</column>
1001         <column name="activityId">18</column>
1002         <column name="score">0.0589841</column>
1003         <column name="subject">[GreenerBuildings Admin] Rome meeting agenda and accommodation.</column>
1004         <column name="dateTime">2011-05-30T17:49:30Z</column>
1005         <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1006     </table>
1007     <table name="Indicium">
1008         <column name="verb">organize</column>
1009         <column name="object">meeting</column>
1010         <column name="activityId">20</column>
1011         <column name="score">0.107548</column>
1012         <column name="subject">[GreenerBuildings Admin] GreenerBuildings: Rome Meeting Accomodation</column>
1013         <column name="dateTime">2011-06-02T17:58:21Z</column>
1014         <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1015     </table>
1016     <table name="Indicium">
1017         <column name="verb">organize</column>
1018         <column name="object">meeting</column>
1019         <column name="activityId">20</column>
1020         <column name="score">0.188209</column>
1021         <column name="subject">[GreenerBuildings Admin] agenda planning for GB meeting September 28-30 Eindhoven Philips Resea
1022         <column name="dateTime">2011-08-29T19:57:18Z</column>
1023         <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1024     </table>
1025     <table name="Indicium">
1026         <column name="verb">organize</column>
1027         <column name="object">meeting</column>
1028         <column name="activityId">20</column>

```

```

1029     <column name="score">0.188209</column>
1030     <column name="subject">[GreenerBuildings Tech] agenda planning for GB meeting September 28-30 Eindhoven Philips Research
1031     <column name="dateTime">2011-08-30T08:59:47Z</column>
1032     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1033 </table>
1034 <table name="Indicium">
1035     <column name="verb">organize</column>
1036     <column name="object">meeting</column>
1037     <column name="activityId">20</column>
1038     <column name="score">0.134435</column>
1039     <column name="subject">[GreenerBuildings Tech] agenda planning for GB meeting September 28-30 Eindhoven Philips Research
1040     <column name="dateTime">2011-09-07T16:07:50Z</column>
1041     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1042 </table>
1043 <table name="Indicium">
1044     <column name="verb">send</column>
1045     <column name="object">agenda</column>
1046     <column name="activityId">14</column>
1047     <column name="score">0.0793686</column>
1048     <column name="subject">[GreenerBuildings Tech] [GreenerBuildings Admin] December meeting in Madrid</column>
1049     <column name="dateTime">2011-10-19T12:22:31Z</column>
1050     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1051 </table>
1052 <table name="Indicium">
1053     <column name="verb">submit</column>
1054     <column name="object">report</column>
1055     <column name="activityId">7</column>
1056     <column name="score">0.0851045</column>
1057     <column name="subject">[GreenerBuildings Tech] GreenerBuildings - periodic report P1 - technical part</column>
1058     <column name="dateTime">2011-12-01T13:12:16Z</column>
1059     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1060 </table>
1061 <table name="Indicium">
1062     <column name="verb">submit</column>
1063     <column name="object">report</column>
1064     <column name="activityId">7</column>
1065     <column name="score">0.0744664</column>

```

```
1066     <column name="subject">[GreenerBuildings Admin] GreenerBuildings - periodic report P1 - financial part</column>
1067     <column name="dateTime">2011-12-08T10:12:10Z</column>
1068     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1069 </table>
1070 <table name="Indicium">
1071     <column name="verb">submit</column>
1072     <column name="object">report</column>
1073     <column name="activityId">7</column>
1074     <column name="score">0.0744664</column>
1075     <column name="subject">[GreenerBuildings Tech] GreenerBuildings - periodic report P1 - technical part</column>
1076     <column name="dateTime">2012-01-06T11:39:10Z</column>
1077     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1078 </table>
1079 <table name="Indicium">
1080     <column name="verb">submit</column>
1081     <column name="object">report</column>
1082     <column name="activityId">7</column>
1083     <column name="score">0.0531903</column>
1084     <column name="subject">RE: [GreenerBuildings Tech] GreenerBuildings - periodic report P1 - technical part</column>
1085     <column name="dateTime">2012-01-12T16:39:41Z</column>
1086     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1087 </table>
1088 <table name="Indicium">
1089     <column name="verb">submit</column>
1090     <column name="object">report</column>
1091     <column name="activityId">7</column>
1092     <column name="score">0.0638284</column>
1093     <column name="subject">RE: [GreenerBuildings Admin] GreenerBuildings - periodic report P1 - financial part</column>
1094     <column name="dateTime">2012-01-12T16:42:36Z</column>
1095     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1096 </table>
1097 <table name="Indicium">
1098     <column name="verb">submit</column>
1099     <column name="object">report</column>
1100     <column name="activityId">7</column>
1101     <column name="score">0.0638284</column>
1102     <column name="subject">[GreenerBuildings Admin] FW: GreenerBuildings - periodic report P1 - financial part</column>
```

180

A. From indicia to log


```

1103         <column name="dateTime">2012-01-13T08:49:53Z</column>
1104         <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1105     </table>
1106     <table name="Indicium">
1107         <column name="verb">write</column>
1108         <column name="object">deliverable</column>
1109         <column name="activityId">1</column>
1110         <column name="score">0.0867201</column>
1111         <column name="subject">[GreenerBuildings Tech] D6.2 status</column>
1112         <column name="dateTime">2012-01-13T09:24:46Z</column>
1113         <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1114     </table>
1115     <table name="Indicium">
1116         <column name="verb">send</column>
1117         <column name="object">draft</column>
1118         <column name="activityId">22</column>
1119         <column name="score">0.0562309</column>
1120         <column name="subject">mail per contributi D2.2</column>
1121         <column name="dateTime">2012-01-14T15:48:50Z</column>
1122         <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1123     </table>
1124     <table name="Indicium">
1125         <column name="verb">send</column>
1126         <column name="object">draft</column>
1127         <column name="activityId">22</column>
1128         <column name="score">0.0562309</column>
1129         <column name="subject">[GreenerBuildings] - D2.2 Table of Content and Request for Contributions</column>
1130         <column name="dateTime">2012-01-14T17:50:22Z</column>
1131         <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1132     </table>
1133     <table name="Indicium">
1134         <column name="verb">submit</column>
1135         <column name="object">report</column>
1136         <column name="activityId">7</column>
1137         <column name="score">0.0531903</column>
1138         <column name="subject">[GreenerBuildings Tech] [GreenerBuildings Office] GreenerBuildings - periodic report P1 - technical
1139         <column name="dateTime">2012-01-23T15:24:44Z</column>

```

```

1140     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1141 </table>
1142 <table name="Indicium">
1143     <column name="verb">send</column>
1144     <column name="object">draft</column>
1145     <column name="activityId">22</column>
1146     <column name="score">0.139164</column>
1147     <column name="subject">Definite OK for Rome meeting on 8 & 9 Feb.</column>
1148     <column name="dateTime">2012-01-23T16:22:49Z</column>
1149     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1150 </table>
1151 <table name="Indicium">
1152     <column name="verb">send</column>
1153     <column name="object">deliverable</column>
1154     <column name="activityId">2</column>
1155     <column name="score">0.0844242</column>
1156     <column name="subject">[GreenerBuildings Admin] Deliverables submission for the review meeting.</column>
1157     <column name="dateTime">2012-01-25T21:36:21Z</column>
1158     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1159 </table>
1160 <table name="Indicium">
1161     <column name="verb">submit</column>
1162     <column name="object">report</column>
1163     <column name="activityId">7</column>
1164     <column name="score">0.0531903</column>
1165     <column name="subject">RE: [GreenerBuildings Tech] GreenerBuildings - periodic report P1 - technical part</column>
1166     <column name="dateTime">2012-01-31T20:10:27Z</column>
1167     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1168 </table>
1169 <table name="Indicium">
1170     <column name="verb">submit</column>
1171     <column name="object">report</column>
1172     <column name="activityId">7</column>
1173     <column name="score">0.0531903</column>
1174     <column name="subject">RE: [GreenerBuildings Tech] GreenerBuildings - periodic report P1 - technical part</column>
1175     <column name="dateTime">2012-01-31T20:15:43Z</column>
1176     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>

```

```

1177 </table>
1178 <table name="Indicium">
1179   <column name="verb">submit</column>
1180   <column name="object">report</column>
1181   <column name="activityId">7</column>
1182   <column name="score">0.0851045</column>
1183   <column name="subject">[GreenerBuildings Board] Contributions of ITRI to GB</column>
1184   <column name="dateTime">2012-02-09T11:31:08Z</column>
1185   <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1186 </table>
1187 <table name="Indicium">
1188   <column name="verb">submit</column>
1189   <column name="object">deliverable</column>
1190   <column name="activityId">3</column>
1191   <column name="score">0.188999</column>
1192   <column name="subject">D2.2a progress</column>
1193   <column name="dateTime">2012-02-21T14:16:07Z</column>
1194   <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1195 </table>
1196 <table name="Indicium">
1197   <column name="verb">submit</column>
1198   <column name="object">report</column>
1199   <column name="activityId">7</column>
1200   <column name="score">0.0851045</column>
1201   <column name="subject">[GreenerBuildings Tech] GreenerBuildings - periodic report P2 - technical part</column>
1202   <column name="dateTime">2012-10-02T14:05:45Z</column>
1203   <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1204 </table>
1205 <table name="Indicium">
1206   <column name="verb">submit</column>
1207   <column name="object">report</column>
1208   <column name="activityId">7</column>
1209   <column name="score">0.0638284</column>
1210   <column name="subject">RE: [GreenerBuildings Tech] GreenerBuildings - periodic report P2 - technical part</column>
1211   <column name="dateTime">2012-10-03T10:54:46Z</column>
1212   <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1213 </table>

```

```

1214 <table name="Indicium">
1215     <column name="verb">submit</column>
1216     <column name="object">report</column>
1217     <column name="activityId">7</column>
1218     <column name="score">0.0744664</column>
1219     <column name="subject">[GreenerBuildings Tech] GreenerBuildings - periodic report P2 - technical part</column>
1220     <column name="dateTime">2012-10-11T09:24:31Z</column>
1221     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1222 </table>
1223 <table name="Indicium">
1224     <column name="verb">submit</column>
1225     <column name="object">report</column>
1226     <column name="activityId">7</column>
1227     <column name="score">0.0744664</column>
1228     <column name="subject">[GreenerBuildings Tech] GreenerBuildings - periodic report P2 - technical part</column>
1229     <column name="dateTime">2012-10-19T15:50:57Z</column>
1230     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1231 </table>
1232 <table name="Indicium">
1233     <column name="verb">submit</column>
1234     <column name="object">report</column>
1235     <column name="activityId">7</column>
1236     <column name="score">0.0638284</column>
1237     <column name="subject">[GreenerBuildings Tech] GreenerBuildings - periodic report P2 - technical part</column>
1238     <column name="dateTime">2012-10-27T14:23:12Z</column>
1239     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1240 </table>
1241 <table name="Indicium">
1242     <column name="verb">submit</column>
1243     <column name="object">deliverable</column>
1244     <column name="activityId">3</column>
1245     <column name="score">0.141749</column>
1246     <column name="subject">[GreenerBuildings Tech] D6.2B Second Draft Version</column>
1247     <column name="dateTime">2012-10-30T14:42:33Z</column>
1248     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1249 </table>
1250 <table name="Indicium">

```

```
1251     <column name="verb">send</column>
1252     <column name="object">agenda</column>
1253     <column name="activityId">14</column>
1254     <column name="score">0.105825</column>
1255     <column name="subject">[GreenerBuildings Admin] Review meeting agenda</column>
1256     <column name="dateTime">2012-10-31T21:22:59Z</column>
1257     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1258 </table>
1259 <table name="Indicium">
1260     <column name="verb">send</column>
1261     <column name="object">agenda</column>
1262     <column name="activityId">14</column>
1263     <column name="score">0.0661405</column>
1264     <column name="subject">[GreenerBuildings Admin] Review meeting agenda</column>
1265     <column name="dateTime">2012-11-01T11:51:37Z</column>
1266     <column name="archiveName">mecella@dis.uniroma1.it/FP7/GreenerBuildings</column>
1267 </table>
1268 </database>
1269 </pma_xml_export>
```

A.3 The XSLT stylesheet to transform the XML log into the XES format

Listing A.3. The XSLT stylesheet to transform the XML log into the XES format

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet
3     version="1.0"
4     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
5     xmlns="http://www.xes-standard.org/"
6 >
7     <xsl:output
8         indent="yes"
9         encoding="utf-8"
10        method="xml"
11        omit-xml-declaration="no"
12    />
13    <xsl:template match="/">
14        <log xes:version="1.0" xes:features="nested-attributes" openxes:version="1.0RC7">
15            <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
16            <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
17            <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
18            <classifier name="Event Name" keys="concept:name"/>
19            <string key="concept:name" value="Email Log"/>
20            <string key="lifecycle:model" value="standard"/>
21
22            <xsl:apply-templates select="//table"/>
23        </log>
24    </xsl:template>
25
26    <xsl:template match="table">
27        <xsl:variable name="archiveName" select="./column[@name='archiveName']/text()" ></xsl:variable>
28        <xsl:if test="
29            not(preceding::table)
30            or
31            (preceding::table[1]/column[@name='archiveName'] != $archiveName)">
32            <trace>
```

```

33         <xsl:element name="string">
34             <xsl:attribute name="key">
35                 <xsl:text>concept:name</xsl:text>
36             </xsl:attribute>
37             <xsl:attribute name="value">
38                 <xsl:value-of select="./column[@name='archiveName']/text()" />
39             </xsl:attribute>
40         </xsl:element>
41         <xsl:apply-templates select="self::node()" mode="IN_TRACE" />
42         <xsl:apply-templates select="following::table[column/@name='archiveName' and column/text() = $arch
43     </trace>
44 </xsl:if>
45 </xsl:template>
46
47 <xsl:template match="table" mode="IN_TRACE">
48     <event>
49         <string key="lifecycle:transition" value="complete"/>
50         <xsl:element name="string">
51             <xsl:attribute name="key">
52                 <xsl:text>concept:name</xsl:text>
53             </xsl:attribute>
54             <xsl:attribute name="value">
55                 <xsl:value-of select="./column[@name='verb']/text()" />
56                 <xsl:text> </xsl:text>
57                 <xsl:value-of select="./column[@name='object']/text()" />
58             </xsl:attribute>
59         </xsl:element>
60         <xsl:element name="date">
61             <xsl:attribute name="key">
62                 <xsl:text>time:timestamp</xsl:text>
63             </xsl:attribute>
64             <xsl:attribute name="value">
65                 <xsl:value-of select="./column[@name='dateTime']/text()" />
66             </xsl:attribute>
67         </xsl:element>
68     </event>
69 </xsl:template>

```

```
70 </xsl:stylesheet>
```


A.4 The XES log

Listing A.4. The XES log

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <log xmlns="http://www.xes-standard.org/" xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7
3   <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.xesext"/>
4   <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
5   <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
6   <classifier name="Event Name" keys="concept:name"/>
7   <string key="concept:name" value="Email Log"/>
8   <string key="lifecycle:model" value="standard"/>
9   <trace>
10    <string key="concept:name" value="dc.claudio@gmail.com/Uniroma1/SM4A11"/>
11    <event>
12      <string key="lifecycle:transition" value="complete"/>
13      <string key="concept:name" value="send agenda"/>
14      <date key="time:timestamp" value="2009-07-09T17:44:59Z"/>
15    </event>
16    <event>
17      <string key="lifecycle:transition" value="complete"/>
18      <string key="concept:name" value="send meeting"/>
19      <date key="time:timestamp" value="2009-07-14T22:24:43Z"/>
20    </event>
21    <event>
22      <string key="lifecycle:transition" value="complete"/>
23      <string key="concept:name" value="send draft"/>
24      <date key="time:timestamp" value="2009-09-11T17:05:50Z"/>
25    </event>
26    <event>
27      <string key="lifecycle:transition" value="complete"/>
28      <string key="concept:name" value="send draft"/>
29      <date key="time:timestamp" value="2009-09-14T10:21:42Z"/>
30    </event>
31    <event>
32      <string key="lifecycle:transition" value="complete"/>
```

```
33     <string key="concept:name" value="send draft"/>
34     <date key="time:timestamp" value="2009-10-12T21:31:49Z"/>
35 </event>
36 <event>
37     <string key="lifecycle:transition" value="complete"/>
38     <string key="concept:name" value="write deliverable"/>
39     <date key="time:timestamp" value="2010-01-12T23:16:34Z"/>
40 </event>
41 <event>
42     <string key="lifecycle:transition" value="complete"/>
43     <string key="concept:name" value="send report"/>
44     <date key="time:timestamp" value="2010-01-13T16:00:58Z"/>
45 </event>
46 <event>
47     <string key="lifecycle:transition" value="complete"/>
48     <string key="concept:name" value="write deliverable"/>
49     <date key="time:timestamp" value="2010-02-01T14:23:35Z"/>
50 </event>
51 <event>
52     <string key="lifecycle:transition" value="complete"/>
53     <string key="concept:name" value="write deliverable"/>
54     <date key="time:timestamp" value="2010-02-01T14:54:15Z"/>
55 </event>
56 <event>
57     <string key="lifecycle:transition" value="complete"/>
58     <string key="concept:name" value="submit deliverable"/>
59     <date key="time:timestamp" value="2010-02-26T01:45:06Z"/>
60 </event>
61 <event>
62     <string key="lifecycle:transition" value="complete"/>
63     <string key="concept:name" value="send deliverable"/>
64     <date key="time:timestamp" value="2010-03-12T05:27:28Z"/>
65 </event>
66 <event>
67     <string key="lifecycle:transition" value="complete"/>
68     <string key="concept:name" value="write deliverable"/>
69     <date key="time:timestamp" value="2010-04-21T05:29:47Z"/>
```

```
70 </event>
71 <event>
72   <string key="lifecycle:transition" value="complete"/>
73   <string key="concept:name" value="submit deliverable"/>
74   <date key="time:timestamp" value="2010-04-26T14:38:50Z"/>
75 </event>
76 <event>
77   <string key="lifecycle:transition" value="complete"/>
78   <string key="concept:name" value="submit deliverable"/>
79   <date key="time:timestamp" value="2010-04-26T20:51:43Z"/>
80 </event>
81 <event>
82   <string key="lifecycle:transition" value="complete"/>
83   <string key="concept:name" value="submit deliverable"/>
84   <date key="time:timestamp" value="2010-04-27T12:17:16Z"/>
85 </event>
86 <event>
87   <string key="lifecycle:transition" value="complete"/>
88   <string key="concept:name" value="submit deliverable"/>
89   <date key="time:timestamp" value="2010-04-27T14:29:14Z"/>
90 </event>
91 <event>
92   <string key="lifecycle:transition" value="complete"/>
93   <string key="concept:name" value="submit deliverable"/>
94   <date key="time:timestamp" value="2010-05-11T10:11:43Z"/>
95 </event>
96 <event>
97   <string key="lifecycle:transition" value="complete"/>
98   <string key="concept:name" value="submit deliverable"/>
99   <date key="time:timestamp" value="2010-05-11T11:35:52Z"/>
100 </event>
101 <event>
102   <string key="lifecycle:transition" value="complete"/>
103   <string key="concept:name" value="organize meeting"/>
104   <date key="time:timestamp" value="2011-01-11T19:20:19Z"/>
105 </event>
106 <event>
```

```
107     <string key="lifecycle:transition" value="complete"/>
108     <string key="concept:name" value="organize meeting"/>
109     <date key="time:timestamp" value="2011-01-17T09:51:45Z"/>
110 </event>
111 <event>
112     <string key="lifecycle:transition" value="complete"/>
113     <string key="concept:name" value="send report"/>
114     <date key="time:timestamp" value="2011-02-09T18:36:58Z"/>
115 </event>
116 <event>
117     <string key="lifecycle:transition" value="complete"/>
118     <string key="concept:name" value="send meeting"/>
119     <date key="time:timestamp" value="2011-02-11T08:20:25Z"/>
120 </event>
121 <event>
122     <string key="lifecycle:transition" value="complete"/>
123     <string key="concept:name" value="send meeting"/>
124     <date key="time:timestamp" value="2012-02-22T15:46:56Z"/>
125 </event>
126 </trace>
127 <trace>
128     <string key="concept:name" value="mecella@dis.uniroma1.it/FP6/WORKPAD"/>
129     <event>
130         <string key="lifecycle:transition" value="complete"/>
131         <string key="concept:name" value="send draft"/>
132         <date key="time:timestamp" value="2009-11-30T12:39:16Z"/>
133     </event>
134     <event>
135         <string key="lifecycle:transition" value="complete"/>
136         <string key="concept:name" value="send draft"/>
137         <date key="time:timestamp" value="2009-11-30T13:14:26Z"/>
138     </event>
139     <event>
140         <string key="lifecycle:transition" value="complete"/>
141         <string key="concept:name" value="send draft"/>
142         <date key="time:timestamp" value="2009-12-03T14:29:55Z"/>
143     </event>
```

```
144 <event>
145   <string key="lifecycle:transition" value="complete"/>
146   <string key="concept:name" value="send deliverable"/>
147   <date key="time:timestamp" value="2009-12-09T11:02:15Z"/>
148 </event>
149 <event>
150   <string key="lifecycle:transition" value="complete"/>
151   <string key="concept:name" value="send deliverable"/>
152   <date key="time:timestamp" value="2009-12-09T12:32:03Z"/>
153 </event>
154 <event>
155   <string key="lifecycle:transition" value="complete"/>
156   <string key="concept:name" value="send deliverable"/>
157   <date key="time:timestamp" value="2009-12-11T21:46:41Z"/>
158 </event>
159 <event>
160   <string key="lifecycle:transition" value="complete"/>
161   <string key="concept:name" value="submit report"/>
162   <date key="time:timestamp" value="2010-01-12T19:06:09Z"/>
163 </event>
164 <event>
165   <string key="lifecycle:transition" value="complete"/>
166   <string key="concept:name" value="submit report"/>
167   <date key="time:timestamp" value="2010-01-13T16:35:08Z"/>
168 </event>
169 <event>
170   <string key="lifecycle:transition" value="complete"/>
171   <string key="concept:name" value="submit report"/>
172   <date key="time:timestamp" value="2010-01-14T15:01:09Z"/>
173 </event>
174 <event>
175   <string key="lifecycle:transition" value="complete"/>
176   <string key="concept:name" value="submit report"/>
177   <date key="time:timestamp" value="2010-02-04T10:40:04Z"/>
178 </event>
179 <event>
180   <string key="lifecycle:transition" value="complete"/>
```

```
181     <string key="concept:name" value="submit report"/>
182     <date key="time:timestamp" value="2010-09-22T11:01:06Z"/>
183 </event>
184 <event>
185     <string key="lifecycle:transition" value="complete"/>
186     <string key="concept:name" value="submit report"/>
187     <date key="time:timestamp" value="2010-09-22T11:07:55Z"/>
188 </event>
189 <event>
190     <string key="lifecycle:transition" value="complete"/>
191     <string key="concept:name" value="submit report"/>
192     <date key="time:timestamp" value="2010-09-22T12:37:01Z"/>
193 </event>
194 <event>
195     <string key="lifecycle:transition" value="complete"/>
196     <string key="concept:name" value="submit report"/>
197     <date key="time:timestamp" value="2010-10-04T13:09:53Z"/>
198 </event>
199 <event>
200     <string key="lifecycle:transition" value="complete"/>
201     <string key="concept:name" value="send report"/>
202     <date key="time:timestamp" value="2010-10-20T21:35:54Z"/>
203 </event>
204 <event>
205     <string key="lifecycle:transition" value="complete"/>
206     <string key="concept:name" value="send report"/>
207     <date key="time:timestamp" value="2010-10-22T10:37:45Z"/>
208 </event>
209 <event>
210     <string key="lifecycle:transition" value="complete"/>
211     <string key="concept:name" value="send report"/>
212     <date key="time:timestamp" value="2012-02-20T10:55:31Z"/>
213 </event>
214 </trace>
215 <trace>
216     <string key="concept:name" value="mecella@dis.uniroma1.it/FP7/SM4A11"/>
217 </event>
```

```
218     <string key="lifecycle:transition" value="complete"/>
219     <string key="concept:name" value="submit report"/>
220     <date key="time:timestamp" value="2009-10-09T12:14:34Z"/>
221 </event>
222 <event>
223     <string key="lifecycle:transition" value="complete"/>
224     <string key="concept:name" value="submit report"/>
225     <date key="time:timestamp" value="2009-10-09T17:38:30Z"/>
226 </event>
227 <event>
228     <string key="lifecycle:transition" value="complete"/>
229     <string key="concept:name" value="send draft"/>
230     <date key="time:timestamp" value="2009-10-12T21:31:49Z"/>
231 </event>
232 <event>
233     <string key="lifecycle:transition" value="complete"/>
234     <string key="concept:name" value="send report"/>
235     <date key="time:timestamp" value="2009-10-14T10:12:59Z"/>
236 </event>
237 <event>
238     <string key="lifecycle:transition" value="complete"/>
239     <string key="concept:name" value="send draft"/>
240     <date key="time:timestamp" value="2009-10-14T10:34:14Z"/>
241 </event>
242 <event>
243     <string key="lifecycle:transition" value="complete"/>
244     <string key="concept:name" value="submit report"/>
245     <date key="time:timestamp" value="2009-10-15T11:54:49Z"/>
246 </event>
247 <event>
248     <string key="lifecycle:transition" value="complete"/>
249     <string key="concept:name" value="submit report"/>
250     <date key="time:timestamp" value="2009-10-15T12:16:07Z"/>
251 </event>
252 <event>
253     <string key="lifecycle:transition" value="complete"/>
254     <string key="concept:name" value="submit draft"/>
```

```
255     <date key="time:timestamp" value="2009-11-25T18:57:32Z"/>
256 </event>
257 <event>
258   <string key="lifecycle:transition" value="complete"/>
259   <string key="concept:name" value="write deliverable"/>
260   <date key="time:timestamp" value="2010-01-12T23:16:34Z"/>
261 </event>
262 <event>
263   <string key="lifecycle:transition" value="complete"/>
264   <string key="concept:name" value="send report"/>
265   <date key="time:timestamp" value="2010-01-13T16:00:58Z"/>
266 </event>
267 <event>
268   <string key="lifecycle:transition" value="complete"/>
269   <string key="concept:name" value="write deliverable"/>
270   <date key="time:timestamp" value="2010-01-28T20:15:41Z"/>
271 </event>
272 <event>
273   <string key="lifecycle:transition" value="complete"/>
274   <string key="concept:name" value="write deliverable"/>
275   <date key="time:timestamp" value="2010-02-01T14:23:35Z"/>
276 </event>
277 <event>
278   <string key="lifecycle:transition" value="complete"/>
279   <string key="concept:name" value="write deliverable"/>
280   <date key="time:timestamp" value="2010-02-01T14:54:15Z"/>
281 </event>
282 <event>
283   <string key="lifecycle:transition" value="complete"/>
284   <string key="concept:name" value="send report"/>
285   <date key="time:timestamp" value="2010-02-04T10:46:18Z"/>
286 </event>
287 <event>
288   <string key="lifecycle:transition" value="complete"/>
289   <string key="concept:name" value="submit report"/>
290   <date key="time:timestamp" value="2010-02-17T17:24:36Z"/>
291 </event>
```



```
292 <event>
293   <string key="lifecycle:transition" value="complete"/>
294   <string key="concept:name" value="submit report"/>
295   <date key="time:timestamp" value="2010-02-25T12:46:25Z"/>
296 </event>
297 <event>
298   <string key="lifecycle:transition" value="complete"/>
299   <string key="concept:name" value="submit report"/>
300   <date key="time:timestamp" value="2010-02-25T15:56:24Z"/>
301 </event>
302 <event>
303   <string key="lifecycle:transition" value="complete"/>
304   <string key="concept:name" value="send draft"/>
305   <date key="time:timestamp" value="2010-02-25T16:47:23Z"/>
306 </event>
307 <event>
308   <string key="lifecycle:transition" value="complete"/>
309   <string key="concept:name" value="submit report"/>
310   <date key="time:timestamp" value="2010-03-02T16:20:31Z"/>
311 </event>
312 <event>
313   <string key="lifecycle:transition" value="complete"/>
314   <string key="concept:name" value="submit draft"/>
315   <date key="time:timestamp" value="2010-03-08T16:48:58Z"/>
316 </event>
317 <event>
318   <string key="lifecycle:transition" value="complete"/>
319   <string key="concept:name" value="submit report"/>
320   <date key="time:timestamp" value="2010-03-10T15:42:22Z"/>
321 </event>
322 <event>
323   <string key="lifecycle:transition" value="complete"/>
324   <string key="concept:name" value="submit report"/>
325   <date key="time:timestamp" value="2010-03-10T15:55:47Z"/>
326 </event>
327 <event>
328   <string key="lifecycle:transition" value="complete"/>
```

```
329     <string key="concept:name" value="submit report"/>
330     <date key="time:timestamp" value="2010-03-11T12:43:18Z"/>
331 </event>
332 <event>
333     <string key="lifecycle:transition" value="complete"/>
334     <string key="concept:name" value="submit report"/>
335     <date key="time:timestamp" value="2010-03-11T19:06:56Z"/>
336 </event>
337 <event>
338     <string key="lifecycle:transition" value="complete"/>
339     <string key="concept:name" value="send deliverable"/>
340     <date key="time:timestamp" value="2010-03-12T05:27:28Z"/>
341 </event>
342 <event>
343     <string key="lifecycle:transition" value="complete"/>
344     <string key="concept:name" value="submit report"/>
345     <date key="time:timestamp" value="2010-03-12T10:02:47Z"/>
346 </event>
347 <event>
348     <string key="lifecycle:transition" value="complete"/>
349     <string key="concept:name" value="send deliverable"/>
350     <date key="time:timestamp" value="2010-03-12T11:41:55Z"/>
351 </event>
352 <event>
353     <string key="lifecycle:transition" value="complete"/>
354     <string key="concept:name" value="send agenda"/>
355     <date key="time:timestamp" value="2010-04-06T17:41:57Z"/>
356 </event>
357 <event>
358     <string key="lifecycle:transition" value="complete"/>
359     <string key="concept:name" value="send agenda"/>
360     <date key="time:timestamp" value="2010-04-09T14:12:39Z"/>
361 </event>
362 <event>
363     <string key="lifecycle:transition" value="complete"/>
364     <string key="concept:name" value="write deliverable"/>
365     <date key="time:timestamp" value="2010-04-21T05:29:47Z"/>
```

```
366 </event>
367 <event>
368   <string key="lifecycle:transition" value="complete"/>
369   <string key="concept:name" value="send demo"/>
370   <date key="time:timestamp" value="2010-04-23T18:05:03Z"/>
371 </event>
372 <event>
373   <string key="lifecycle:transition" value="complete"/>
374   <string key="concept:name" value="submit deliverable"/>
375   <date key="time:timestamp" value="2010-04-26T14:38:50Z"/>
376 </event>
377 <event>
378   <string key="lifecycle:transition" value="complete"/>
379   <string key="concept:name" value="submit deliverable"/>
380   <date key="time:timestamp" value="2010-04-26T20:51:43Z"/>
381 </event>
382 <event>
383   <string key="lifecycle:transition" value="complete"/>
384   <string key="concept:name" value="submit deliverable"/>
385   <date key="time:timestamp" value="2010-04-27T12:17:16Z"/>
386 </event>
387 <event>
388   <string key="lifecycle:transition" value="complete"/>
389   <string key="concept:name" value="submit deliverable"/>
390   <date key="time:timestamp" value="2010-04-27T14:29:14Z"/>
391 </event>
392 <event>
393   <string key="lifecycle:transition" value="complete"/>
394   <string key="concept:name" value="submit deliverable"/>
395   <date key="time:timestamp" value="2010-05-11T10:11:43Z"/>
396 </event>
397 <event>
398   <string key="lifecycle:transition" value="complete"/>
399   <string key="concept:name" value="submit deliverable"/>
400   <date key="time:timestamp" value="2010-05-11T11:35:52Z"/>
401 </event>
402 <event>
```

```
403     <string key="lifecycle:transition" value="complete"/>
404     <string key="concept:name" value="submit report"/>
405     <date key="time:timestamp" value="2010-06-30T11:53:34Z"/>
406 </event>
407 <event>
408     <string key="lifecycle:transition" value="complete"/>
409     <string key="concept:name" value="submit report"/>
410     <date key="time:timestamp" value="2010-07-20T15:45:19Z"/>
411 </event>
412 <event>
413     <string key="lifecycle:transition" value="complete"/>
414     <string key="concept:name" value="submit report"/>
415     <date key="time:timestamp" value="2010-07-20T18:55:49Z"/>
416 </event>
417 <event>
418     <string key="lifecycle:transition" value="complete"/>
419     <string key="concept:name" value="send deliverable"/>
420     <date key="time:timestamp" value="2010-09-09T15:00:28Z"/>
421 </event>
422 <event>
423     <string key="lifecycle:transition" value="complete"/>
424     <string key="concept:name" value="organize agenda"/>
425     <date key="time:timestamp" value="2010-10-29T10:14:49Z"/>
426 </event>
427 <event>
428     <string key="lifecycle:transition" value="complete"/>
429     <string key="concept:name" value="organize meeting"/>
430     <date key="time:timestamp" value="2011-01-11T19:20:19Z"/>
431 </event>
432 <event>
433     <string key="lifecycle:transition" value="complete"/>
434     <string key="concept:name" value="organize meeting"/>
435     <date key="time:timestamp" value="2011-01-17T09:51:45Z"/>
436 </event>
437 <event>
438     <string key="lifecycle:transition" value="complete"/>
439     <string key="concept:name" value="send draft"/>
```

```
440     <date key="time:timestamp" value="2011-01-17T10:48:06Z"/>
441 </event>
442 <event>
443   <string key="lifecycle:transition" value="complete"/>
444   <string key="concept:name" value="send agenda"/>
445   <date key="time:timestamp" value="2011-01-17T11:13:07Z"/>
446 </event>
447 <event>
448   <string key="lifecycle:transition" value="complete"/>
449   <string key="concept:name" value="send agenda"/>
450   <date key="time:timestamp" value="2011-01-17T11:13:53Z"/>
451 </event>
452 <event>
453   <string key="lifecycle:transition" value="complete"/>
454   <string key="concept:name" value="organize demo"/>
455   <date key="time:timestamp" value="2011-01-31T10:52:34Z"/>
456 </event>
457 <event>
458   <string key="lifecycle:transition" value="complete"/>
459   <string key="concept:name" value="send meeting"/>
460   <date key="time:timestamp" value="2011-02-02T15:45:21Z"/>
461 </event>
462 <event>
463   <string key="lifecycle:transition" value="complete"/>
464   <string key="concept:name" value="send meeting"/>
465   <date key="time:timestamp" value="2011-05-17T14:56:30Z"/>
466 </event>
467 <event>
468   <string key="lifecycle:transition" value="complete"/>
469   <string key="concept:name" value="send draft"/>
470   <date key="time:timestamp" value="2011-07-29T16:43:37Z"/>
471 </event>
472 <event>
473   <string key="lifecycle:transition" value="complete"/>
474   <string key="concept:name" value="send agenda"/>
475   <date key="time:timestamp" value="2011-09-21T12:14:46Z"/>
476 </event>
```

```
477 <event>
478   <string key="lifecycle:transition" value="complete"/>
479   <string key="concept:name" value="send meeting"/>
480   <date key="time:timestamp" value="2012-02-22T15:46:56Z"/>
481 </event>
482 <event>
483   <string key="lifecycle:transition" value="complete"/>
484   <string key="concept:name" value="send meeting"/>
485   <date key="time:timestamp" value="2012-03-13T12:10:18Z"/>
486 </event>
487 <event>
488   <string key="lifecycle:transition" value="complete"/>
489   <string key="concept:name" value="send draft"/>
490   <date key="time:timestamp" value="2012-03-19T13:28:59Z"/>
491 </event>
492 <event>
493   <string key="lifecycle:transition" value="complete"/>
494   <string key="concept:name" value="send meeting"/>
495   <date key="time:timestamp" value="2012-03-27T10:54:02Z"/>
496 </event>
497 <event>
498   <string key="lifecycle:transition" value="complete"/>
499   <string key="concept:name" value="send meeting"/>
500   <date key="time:timestamp" value="2012-03-28T16:22:43Z"/>
501 </event>
502 <event>
503   <string key="lifecycle:transition" value="complete"/>
504   <string key="concept:name" value="send meeting"/>
505   <date key="time:timestamp" value="2012-03-28T16:48:32Z"/>
506 </event>
507 <event>
508   <string key="lifecycle:transition" value="complete"/>
509   <string key="concept:name" value="send report"/>
510   <date key="time:timestamp" value="2012-07-02T13:35:28Z"/>
511 </event>
512 <event>
513   <string key="lifecycle:transition" value="complete"/>
```

```

514     <string key="concept:name" value="send deliverable"/>
515     <date key="time:timestamp" value="2012-07-05T10:46:45Z"/>
516 </event>
517 <event>
518     <string key="lifecycle:transition" value="complete"/>
519     <string key="concept:name" value="send deliverable"/>
520     <date key="time:timestamp" value="2012-07-06T12:15:31Z"/>
521 </event>
522 <event>
523     <string key="lifecycle:transition" value="complete"/>
524     <string key="concept:name" value="send deliverable"/>
525     <date key="time:timestamp" value="2012-07-10T12:22:14Z"/>
526 </event>
527 </trace>
528 <trace>
529     <string key="concept:name" value="mecella@dis.uniroma1.it/FP7/GreenerBuildings"/>
530     <event>
531         <string key="lifecycle:transition" value="complete"/>
532         <string key="concept:name" value="organize agenda"/>
533         <date key="time:timestamp" value="2010-10-24T23:53:41Z"/>
534     </event>
535     <event>
536         <string key="lifecycle:transition" value="complete"/>
537         <string key="concept:name" value="send draft"/>
538         <date key="time:timestamp" value="2010-10-29T14:45:28Z"/>
539     </event>
540     <event>
541         <string key="lifecycle:transition" value="complete"/>
542         <string key="concept:name" value="send meeting"/>
543         <date key="time:timestamp" value="2010-12-01T21:26:57Z"/>
544     </event>
545     <event>
546         <string key="lifecycle:transition" value="complete"/>
547         <string key="concept:name" value="write deliverable"/>
548         <date key="time:timestamp" value="2011-01-25T00:05:52Z"/>
549     </event>
550 </event>

```

```
551     <string key="lifecycle:transition" value="complete"/>
552     <string key="concept:name" value="send deliverable"/>
553     <date key="time:timestamp" value="2011-03-04T09:25:34Z"/>
554 </event>
555 <event>
556     <string key="lifecycle:transition" value="complete"/>
557     <string key="concept:name" value="organize agenda"/>
558     <date key="time:timestamp" value="2011-05-17T11:08:35Z"/>
559 </event>
560 <event>
561     <string key="lifecycle:transition" value="complete"/>
562     <string key="concept:name" value="send agenda"/>
563     <date key="time:timestamp" value="2011-05-22T19:16:49Z"/>
564 </event>
565 <event>
566     <string key="lifecycle:transition" value="complete"/>
567     <string key="concept:name" value="send meeting"/>
568     <date key="time:timestamp" value="2011-05-30T17:49:30Z"/>
569 </event>
570 <event>
571     <string key="lifecycle:transition" value="complete"/>
572     <string key="concept:name" value="organize meeting"/>
573     <date key="time:timestamp" value="2011-06-02T17:58:21Z"/>
574 </event>
575 <event>
576     <string key="lifecycle:transition" value="complete"/>
577     <string key="concept:name" value="organize meeting"/>
578     <date key="time:timestamp" value="2011-08-29T19:57:18Z"/>
579 </event>
580 <event>
581     <string key="lifecycle:transition" value="complete"/>
582     <string key="concept:name" value="organize meeting"/>
583     <date key="time:timestamp" value="2011-08-30T08:59:47Z"/>
584 </event>
585 <event>
586     <string key="lifecycle:transition" value="complete"/>
587     <string key="concept:name" value="organize meeting"/>
```



```
588     <date key="time:timestamp" value="2011-09-07T16:07:50Z"/>
589 </event>
590 <event>
591   <string key="lifecycle:transition" value="complete"/>
592   <string key="concept:name" value="send agenda"/>
593   <date key="time:timestamp" value="2011-10-19T12:22:31Z"/>
594 </event>
595 <event>
596   <string key="lifecycle:transition" value="complete"/>
597   <string key="concept:name" value="submit report"/>
598   <date key="time:timestamp" value="2011-12-01T13:12:16Z"/>
599 </event>
600 <event>
601   <string key="lifecycle:transition" value="complete"/>
602   <string key="concept:name" value="submit report"/>
603   <date key="time:timestamp" value="2011-12-08T10:12:10Z"/>
604 </event>
605 <event>
606   <string key="lifecycle:transition" value="complete"/>
607   <string key="concept:name" value="submit report"/>
608   <date key="time:timestamp" value="2012-01-06T11:39:10Z"/>
609 </event>
610 <event>
611   <string key="lifecycle:transition" value="complete"/>
612   <string key="concept:name" value="submit report"/>
613   <date key="time:timestamp" value="2012-01-12T16:39:41Z"/>
614 </event>
615 <event>
616   <string key="lifecycle:transition" value="complete"/>
617   <string key="concept:name" value="submit report"/>
618   <date key="time:timestamp" value="2012-01-12T16:42:36Z"/>
619 </event>
620 <event>
621   <string key="lifecycle:transition" value="complete"/>
622   <string key="concept:name" value="submit report"/>
623   <date key="time:timestamp" value="2012-01-13T08:49:53Z"/>
624 </event>
```

```
625 <event>
626   <string key="lifecycle:transition" value="complete"/>
627   <string key="concept:name" value="write deliverable"/>
628   <date key="time:timestamp" value="2012-01-13T09:24:46Z"/>
629 </event>
630 <event>
631   <string key="lifecycle:transition" value="complete"/>
632   <string key="concept:name" value="send draft"/>
633   <date key="time:timestamp" value="2012-01-14T15:48:50Z"/>
634 </event>
635 <event>
636   <string key="lifecycle:transition" value="complete"/>
637   <string key="concept:name" value="send draft"/>
638   <date key="time:timestamp" value="2012-01-14T17:50:22Z"/>
639 </event>
640 <event>
641   <string key="lifecycle:transition" value="complete"/>
642   <string key="concept:name" value="submit report"/>
643   <date key="time:timestamp" value="2012-01-23T15:24:44Z"/>
644 </event>
645 <event>
646   <string key="lifecycle:transition" value="complete"/>
647   <string key="concept:name" value="send draft"/>
648   <date key="time:timestamp" value="2012-01-23T16:22:49Z"/>
649 </event>
650 <event>
651   <string key="lifecycle:transition" value="complete"/>
652   <string key="concept:name" value="send deliverable"/>
653   <date key="time:timestamp" value="2012-01-25T21:36:21Z"/>
654 </event>
655 <event>
656   <string key="lifecycle:transition" value="complete"/>
657   <string key="concept:name" value="submit report"/>
658   <date key="time:timestamp" value="2012-01-31T20:10:27Z"/>
659 </event>
660 <event>
661   <string key="lifecycle:transition" value="complete"/>
```

```
662     <string key="concept:name" value="submit report"/>
663     <date key="time:timestamp" value="2012-01-31T20:15:43Z"/>
664 </event>
665 <event>
666     <string key="lifecycle:transition" value="complete"/>
667     <string key="concept:name" value="submit report"/>
668     <date key="time:timestamp" value="2012-02-09T11:31:08Z"/>
669 </event>
670 <event>
671     <string key="lifecycle:transition" value="complete"/>
672     <string key="concept:name" value="submit deliverable"/>
673     <date key="time:timestamp" value="2012-02-21T14:16:07Z"/>
674 </event>
675 <event>
676     <string key="lifecycle:transition" value="complete"/>
677     <string key="concept:name" value="submit report"/>
678     <date key="time:timestamp" value="2012-10-02T14:05:45Z"/>
679 </event>
680 <event>
681     <string key="lifecycle:transition" value="complete"/>
682     <string key="concept:name" value="submit report"/>
683     <date key="time:timestamp" value="2012-10-03T10:54:46Z"/>
684 </event>
685 <event>
686     <string key="lifecycle:transition" value="complete"/>
687     <string key="concept:name" value="submit report"/>
688     <date key="time:timestamp" value="2012-10-11T09:24:31Z"/>
689 </event>
690 <event>
691     <string key="lifecycle:transition" value="complete"/>
692     <string key="concept:name" value="submit report"/>
693     <date key="time:timestamp" value="2012-10-19T15:50:57Z"/>
694 </event>
695 <event>
696     <string key="lifecycle:transition" value="complete"/>
697     <string key="concept:name" value="submit report"/>
698     <date key="time:timestamp" value="2012-10-27T14:23:12Z"/>
```

```
699 </event>
700 <event>
701   <string key="lifecycle:transition" value="complete"/>
702   <string key="concept:name" value="submit deliverable"/>
703   <date key="time:timestamp" value="2012-10-30T14:42:33Z"/>
704 </event>
705 <event>
706   <string key="lifecycle:transition" value="complete"/>
707   <string key="concept:name" value="send agenda"/>
708   <date key="time:timestamp" value="2012-10-31T21:22:59Z"/>
709 </event>
710 <event>
711   <string key="lifecycle:transition" value="complete"/>
712   <string key="concept:name" value="send agenda"/>
713   <date key="time:timestamp" value="2012-11-01T11:51:37Z"/>
714 </event>
715 </trace>
716 </log>
```

Appendix B

The discovered process

B.1 The local Finite State Automata, generated on the basis of the discovered process' constraints

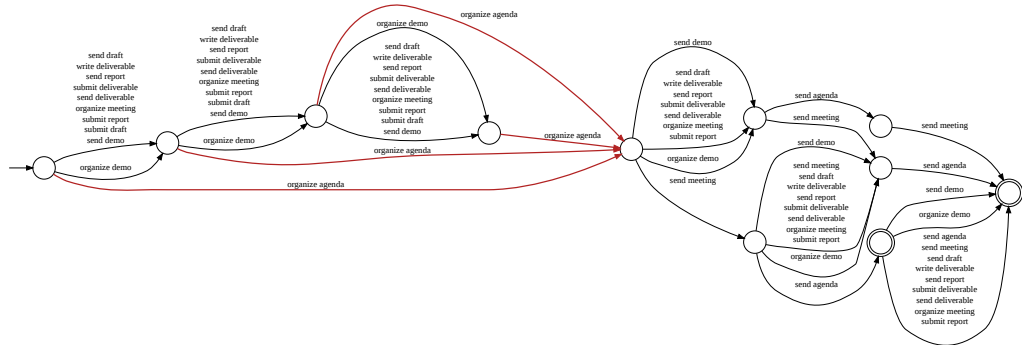


Figure B.1. The local automaton for the “organize agenda” activity

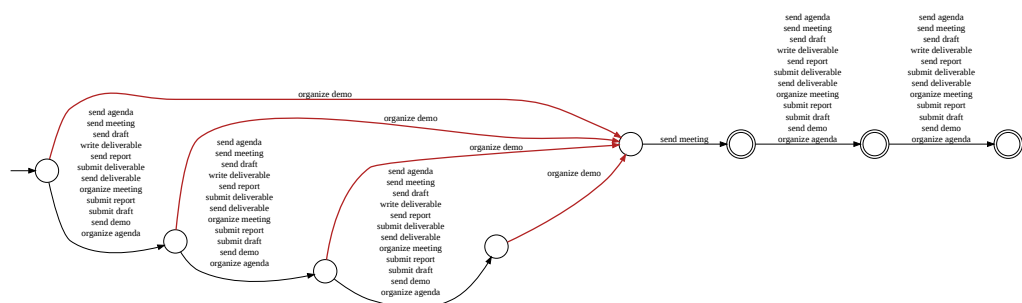


Figure B.2. The local automaton for the “organize demo” activity

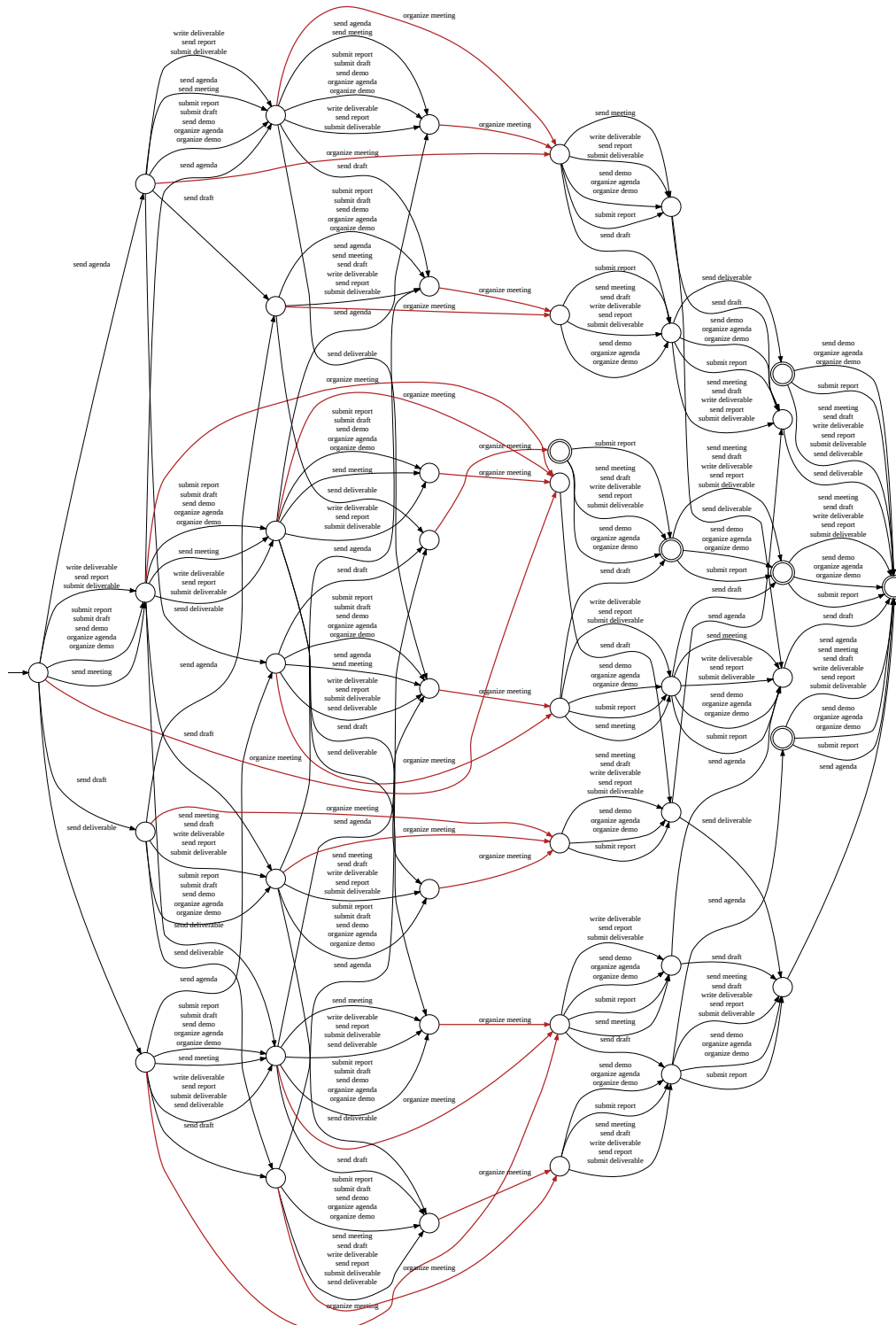


Figure B.3. The local automaton for the “organize meeting” activity

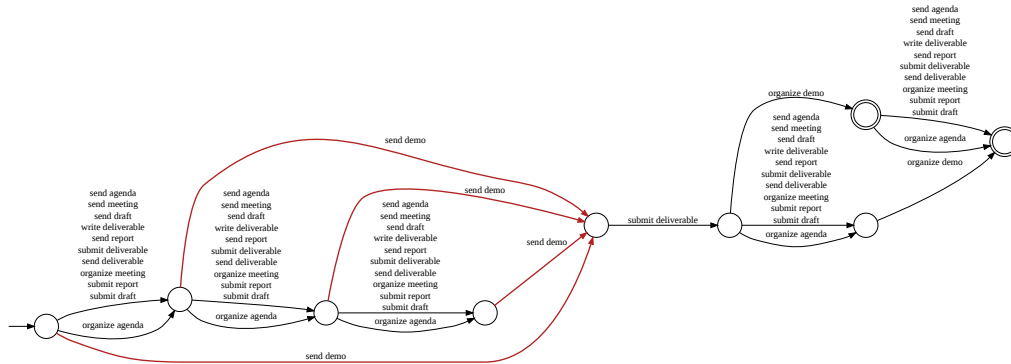


Figure B.6. The local automaton for the “send demo” activity

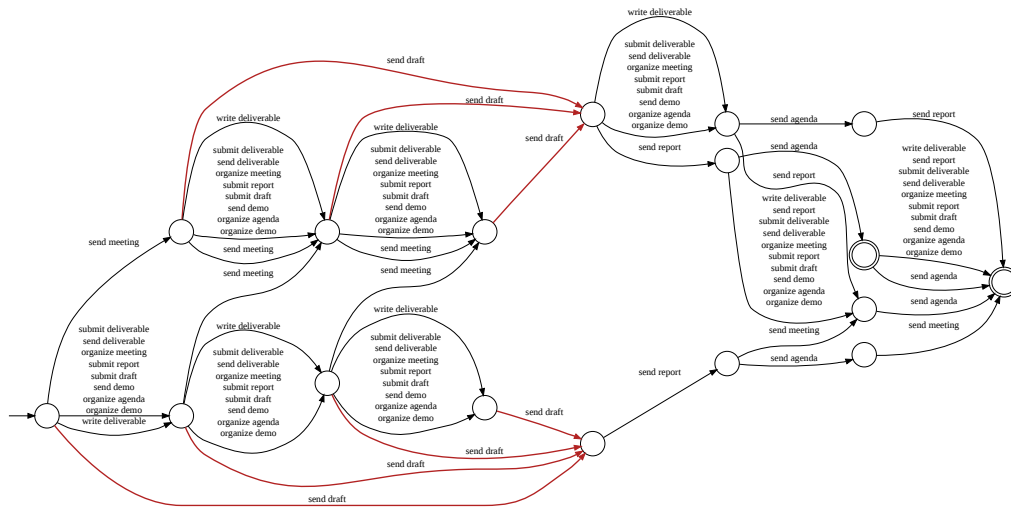


Figure B.7. The local automaton for the “send draft” activity

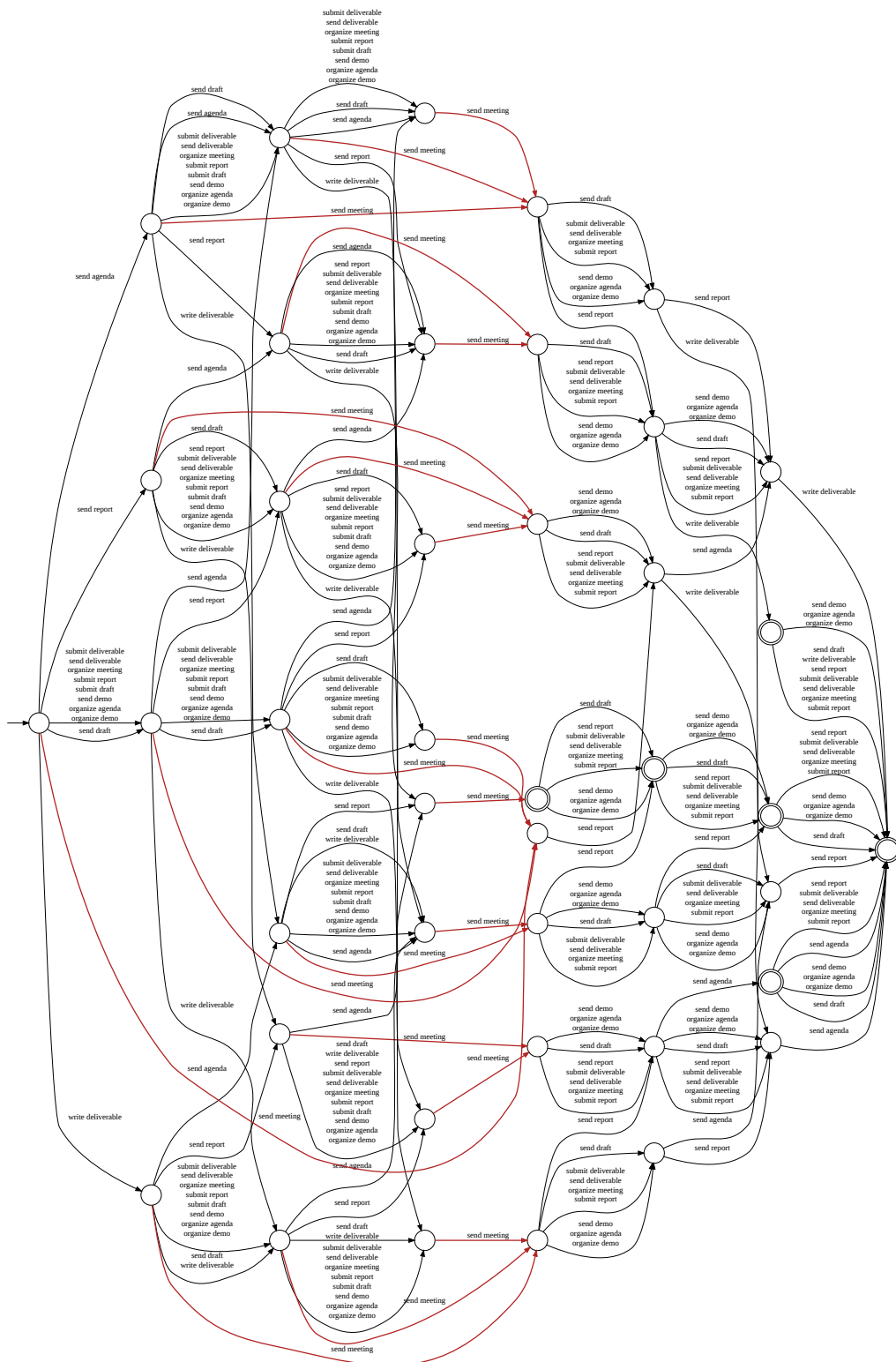


Figure B.8. The local automaton for the “send meeting” activity

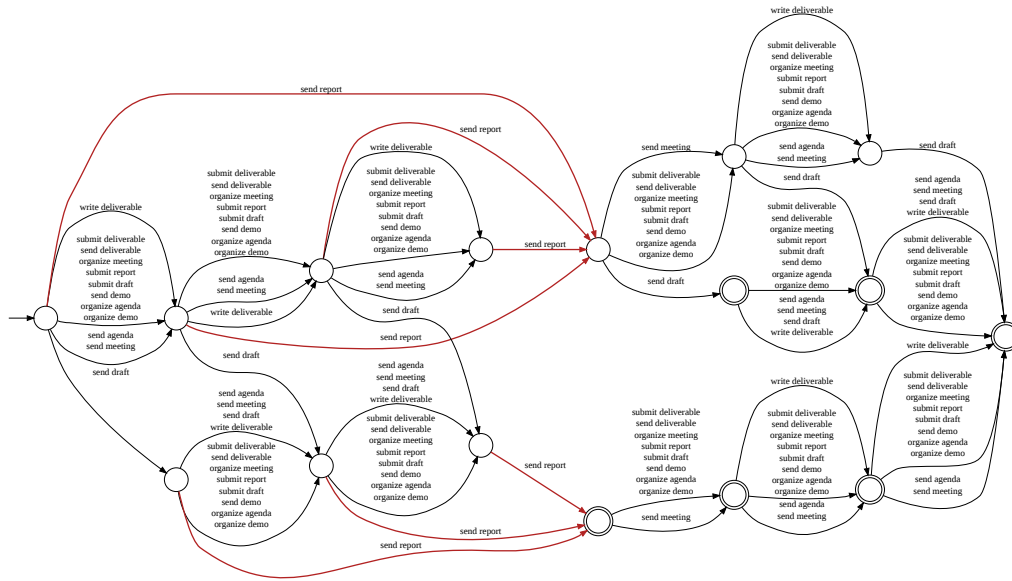


Figure B.9. The local automaton for the “send report” activity

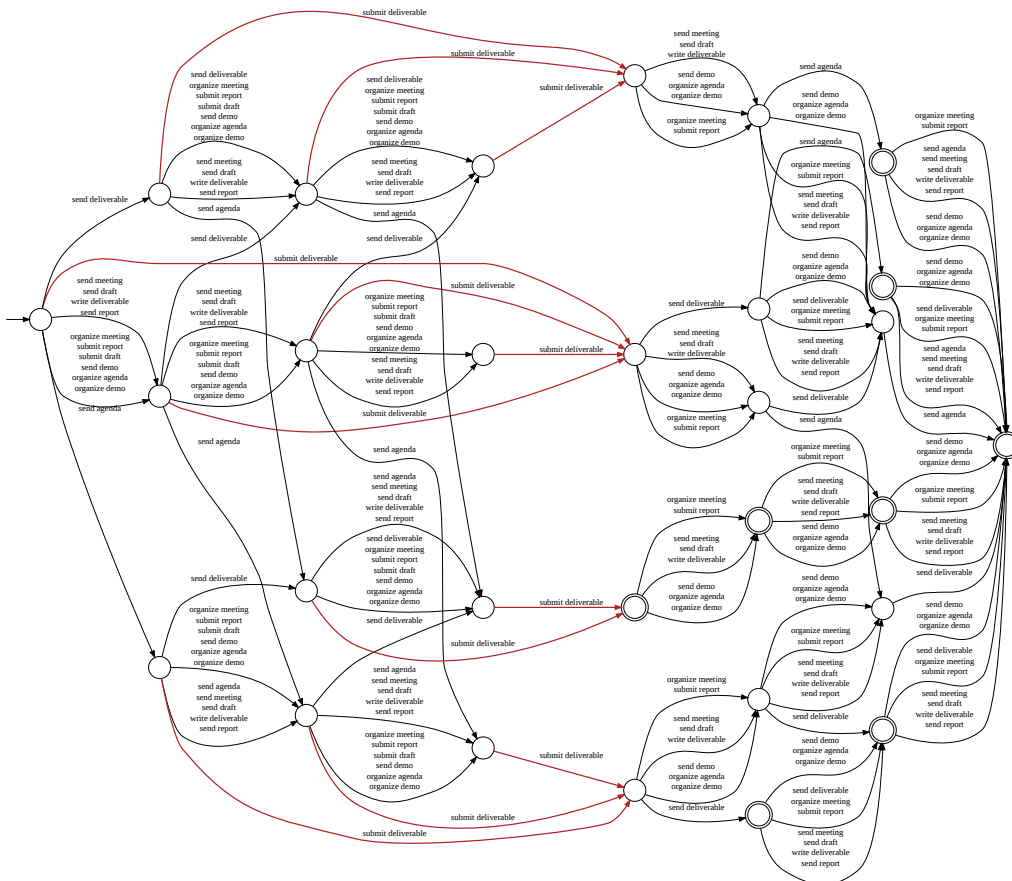


Figure B.10. The local automaton for the “submit deliverable” activity

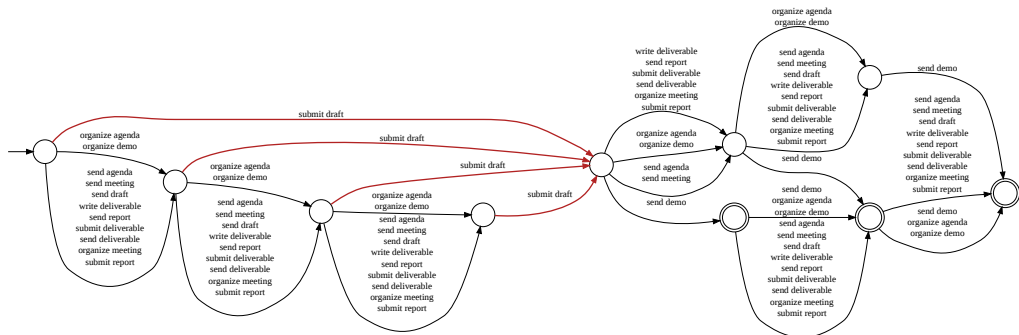


Figure B.11. The local automaton for the “submit draft” activity

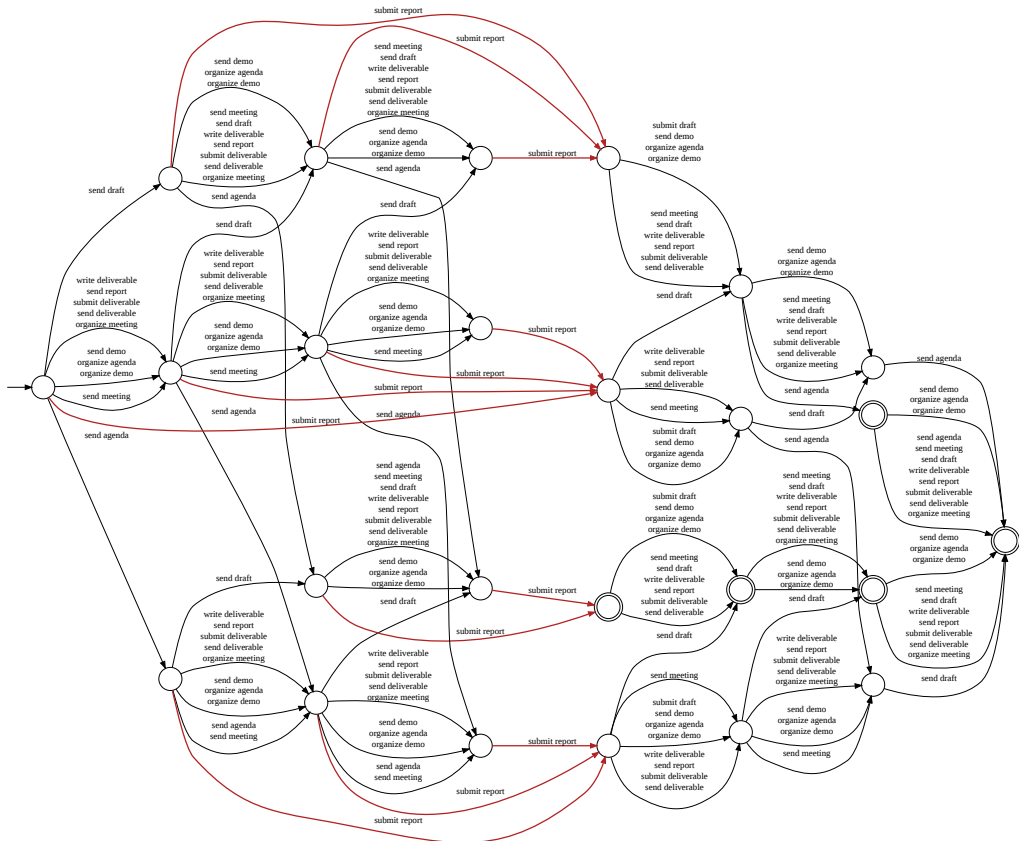


Figure B.12. The local automaton for the “submit report” activity

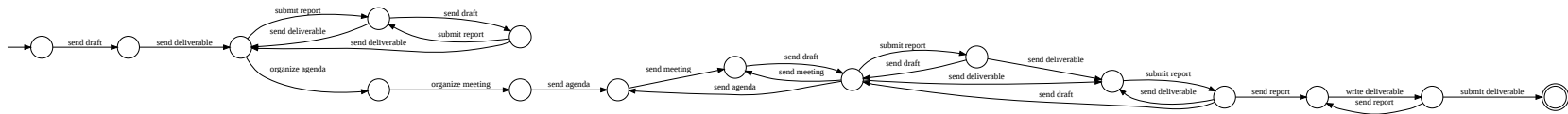


Figure B.14. The global automaton for the discovered process

B.2 The discovered process' Finite State Automaton

B.3 The discovered process, as in the output of the run of MailOfMine

Listing B.1. The discovered process

```

1 [send agenda] => {
2   100.000% RespondedExistence(send agenda, send meeting)      100.000% ||||| conf.: 0.750; int'f: 0.188;
3   91.667% Precedence(send agenda, send meeting)                58.333% ||||| conf.: 0.688; int'f: 0.172;
4   100.000% CoExistence(send agenda, send meeting)             100.000% ||||| conf.: 0.750; int'f: 0.188;
5   81.818% Succession(send agenda, send meeting)                9.091%   conf.: 0.614; int'f: 0.153;
6   100.000% NotChainSuccession(send agenda, send draft)         100.000% ||||| conf.: 0.750; int'f: 0.188;
7   100.000% RespondedExistence(send agenda, send draft)         100.000% ||||| conf.: 0.750; int'f: 0.188;
8   80.000% Response(send agenda, send draft)                    0.000%   conf.: 0.600; int'f: 0.150;
9   88.462% CoExistence(send agenda, send draft)                 42.308% ||||| conf.: 0.663; int'f: 0.166;
10  90.476% NotChainSuccession(send agenda, write deliverable)    52.381% ||||| conf.: 0.679; int'f: 0.170;
11  100.000% RespondedExistence(send agenda, write deliverable)  100.000% ||||| conf.: 0.750; int'f: 0.188;
12  100.000% CoExistence(send agenda, write deliverable)          100.000% ||||| conf.: 0.750; int'f: 0.188;
13  100.000% NotChainSuccession(send agenda, send report)         100.000% ||||| conf.: 0.750; int'f: 0.188;
14  100.000% NotChainSuccession(send agenda, submit deliverable)  100.000% ||||| conf.: 0.750; int'f: 0.188;
15  100.000% RespondedExistence(send agenda, submit deliverable)  100.000% ||||| conf.: 0.750; int'f: 0.188;
16  100.000% Precedence(send agenda, submit deliverable)          100.000% ||||| conf.: 0.750; int'f: 0.188;
17  100.000% CoExistence(send agenda, submit deliverable)          100.000% ||||| conf.: 0.750; int'f: 0.188;
18  80.000% Succession(send agenda, submit deliverable)           0.000%   conf.: 0.600; int'f: 0.150;
19  100.000% NotChainSuccession(send agenda, send deliverable)    100.000% ||||| conf.: 0.750; int'f: 0.188;
20  100.000% RespondedExistence(send agenda, send deliverable)    100.000% ||||| conf.: 0.750; int'f: 0.188;
21  80.000% Response(send agenda, send deliverable)               0.000%   conf.: 0.600; int'f: 0.150;
22  86.364% CoExistence(send agenda, send deliverable)            31.818% |||   conf.: 0.648; int'f: 0.162;
23  100.000% NotChainSuccession(send agenda, organize meeting)    100.000% ||||| conf.: 0.750; int'f: 0.188;
24  100.000% RespondedExistence(send agenda, organize meeting)    100.000% ||||| conf.: 0.750; int'f: 0.188;
25  100.000% Precedence(send agenda, organize meeting)             100.000% ||||| conf.: 0.750; int'f: 0.188;
26  100.000% CoExistence(send agenda, organize meeting)           100.000% ||||| conf.: 0.750; int'f: 0.188;
27  95.918% NotChainSuccession(send agenda, submit report)         79.592% ||||| conf.: 0.719; int'f: 0.180;
28  90.000% RespondedExistence(send agenda, submit report)         50.000% ||||| conf.: 0.675; int'f: 0.169;
29  81.633% CoExistence(send agenda, submit report)                8.163%   conf.: 0.612; int'f: 0.153;
30  100.000% NotChainSuccession(send agenda, submit draft)         100.000% ||||| conf.: 0.750; int'f: 0.188;
31  100.000% NotSuccession(send agenda, submit draft)              100.000% ||||| conf.: 0.750; int'f: 0.188;
32  100.000% NotChainSuccession(send agenda, send demo)            100.000% ||||| conf.: 0.750; int'f: 0.188;
33  100.000% Precedence(send agenda, send demo)                    100.000% ||||| conf.: 0.750; int'f: 0.188;
34  100.000% AlternatePrecedence(send agenda, send demo)           100.000% ||||| conf.: 0.750; int'f: 0.188;
35  100.000% NotChainSuccession(send agenda, organize agenda)      100.000% ||||| conf.: 0.750; int'f: 0.188;
36  90.000% RespondedExistence(send agenda, organize agenda)       50.000% ||||| conf.: 0.675; int'f: 0.169;
37  92.308% CoExistence(send agenda, organize agenda)              61.538% ||||| conf.: 0.692; int'f: 0.173;
38  81.818% NotChainSuccession(send agenda, organize demo)         9.091%   conf.: 0.614; int'f: 0.153;
39  100.000% Precedence(send agenda, organize demo)                100.000% ||||| conf.: 0.750; int'f: 0.188;
40  100.000% AlternatePrecedence(send agenda, organize demo)       100.000% ||||| conf.: 0.750; int'f: 0.188;
41  100.000% ChainPrecedence(send agenda, organize demo)           100.000% ||||| conf.: 0.750; int'f: 0.188;
42
43 }
44
45 [send meeting] => {
46  100.000% NotChainSuccession(send meeting, send agenda)         100.000% ||||| conf.: 0.750; int'f: 0.188;
47  100.000% RespondedExistence(send meeting, send agenda)         100.000% ||||| conf.: 0.750; int'f: 0.188;
48  100.000% CoExistence(send meeting, send agenda)                100.000% ||||| conf.: 0.750; int'f: 0.188;
49  100.000% RespondedExistence(send meeting, send draft)          100.000% ||||| conf.: 0.750; int'f: 0.188;
50  89.286% CoExistence(send meeting, send draft)                  46.429% ||||| conf.: 0.670; int'f: 0.167;
51  91.304% NotChainSuccession(send meeting, write deliverable)    56.522% ||||| conf.: 0.685; int'f: 0.171;

```

```

52 100.000% RespondedExistence(send meeting, write deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
53 100.000% CoExistence(send meeting, write deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
54 90.476% NotChainSuccession(send meeting, send report) 52.381% ||||| conf.: 0.679; int'f: 0.170;
55 83.333% RespondedExistence(send meeting, send report) 16.667% | conf.: 0.625; int'f: 0.156;
56 100.000% NotChainSuccession(send meeting, submit deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
57 100.000% RespondedExistence(send meeting, submit deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
58 100.000% CoExistence(send meeting, submit deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
59 100.000% NotChainSuccession(send meeting, send deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
60 100.000% RespondedExistence(send meeting, send deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
61 83.333% Response(send meeting, send deliverable) 16.667% | conf.: 0.625; int'f: 0.156;
62 87.500% CoExistence(send meeting, send deliverable) 37.500% ||| conf.: 0.656; int'f: 0.164;
63 90.000% NotChainSuccession(send meeting, organize meeting) 50.000% ||||| conf.: 0.675; int'f: 0.169;
64 100.000% RespondedExistence(send meeting, organize meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
65 100.000% CoExistence(send meeting, organize meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
66 100.000% NotChainSuccession(send meeting, submit report) 100.000% ||||| conf.: 0.750; int'f: 0.188;
67 100.000% NotChainSuccession(send meeting, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
68 100.000% NotSuccession(send meeting, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
69 100.000% NotChainSuccession(send meeting, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
70 100.000% NotSuccession(send meeting, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
71 100.000% NotChainSuccession(send meeting, organize agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
72 86.667% NotSuccession(send meeting, organize agenda) 33.333% ||| conf.: 0.650; int'f: 0.163;
73 80.000% CoExistence(send meeting, organize agenda) 0.000% | conf.: 0.600; int'f: 0.150;
74 100.000% NotChainSuccession(send meeting, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
75 100.000% NotSuccession(send meeting, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
76
77 }
78
79 [send draft] => {
80 100.000% Participation(send draft) 100.000% ||||| conf.: 1.000; int'f: 1.000;
81 84.615% NotChainSuccession(send draft, send agenda) 23.077% || conf.: 0.846; int'f: 0.212;
82 81.250% RespondedExistence(send draft, send agenda) 6.250% | conf.: 0.813; int'f: 0.203;
83 90.000% Precedence(send draft, send agenda) 50.000% ||||| conf.: 0.900; int'f: 0.225;
84 88.462% CoExistence(send draft, send agenda) 42.308% ||||| conf.: 0.885; int'f: 0.221;
85 85.714% NotChainSuccession(send draft, send meeting) 28.571% || conf.: 0.857; int'f: 0.214;
86 81.250% RespondedExistence(send draft, send meeting) 6.250% | conf.: 0.813; int'f: 0.203;
87 91.667% Precedence(send draft, send meeting) 58.333% ||||| conf.: 0.917; int'f: 0.229;
88 89.286% CoExistence(send draft, send meeting) 46.429% ||||| conf.: 0.893; int'f: 0.223;
89 92.593% NotChainSuccession(send draft, write deliverable) 62.963% ||||| conf.: 0.926; int'f: 0.231;
90 81.250% RespondedExistence(send draft, write deliverable) 6.250% | conf.: 0.813; int'f: 0.203;
91 100.000% Precedence(send draft, write deliverable) 100.000% ||||| conf.: 1.000; int'f: 0.250;
92 88.889% CoExistence(send draft, write deliverable) 44.444% ||||| conf.: 0.889; int'f: 0.222;
93 92.000% NotChainSuccession(send draft, send report) 60.000% ||||| conf.: 0.920; int'f: 0.230;
94 100.000% Precedence(send draft, send report) 100.000% ||||| conf.: 1.000; int'f: 0.250;
95 84.000% CoExistence(send draft, send report) 20.000% | conf.: 0.840; int'f: 0.210;
96 84.000% Succession(send draft, send report) 20.000% | conf.: 0.840; int'f: 0.210;
97 100.000% NotChainSuccession(send draft, submit deliverable) 100.000% ||||| conf.: 1.000; int'f: 0.250;
98 81.250% RespondedExistence(send draft, submit deliverable) 6.250% | conf.: 0.813; int'f: 0.203;
99 100.000% Precedence(send draft, submit deliverable) 100.000% ||||| conf.: 1.000; int'f: 0.250;
100 90.323% CoExistence(send draft, submit deliverable) 51.613% ||||| conf.: 0.903; int'f: 0.226;
101 80.645% Succession(send draft, submit deliverable) 3.226% | conf.: 0.806; int'f: 0.202;
102 85.714% NotChainSuccession(send draft, send deliverable) 28.571% || conf.: 0.857; int'f: 0.214;
103 100.000% RespondedExistence(send draft, send deliverable) 100.000% ||||| conf.: 1.000; int'f: 0.250;
104 100.000% Response(send draft, send deliverable) 100.000% ||||| conf.: 1.000; int'f: 0.250;
105 100.000% Precedence(send draft, send deliverable) 100.000% ||||| conf.: 1.000; int'f: 0.250;
106 100.000% CoExistence(send draft, send deliverable) 100.000% ||||| conf.: 1.000; int'f: 0.250;
107 100.000% Succession(send draft, send deliverable) 100.000% ||||| conf.: 1.000; int'f: 0.250;
108 100.000% NotChainSuccession(send draft, organize meeting) 100.000% ||||| conf.: 1.000; int'f: 0.250;

```



```

109 81.250% RespondedExistence(send draft, organize meeting) 6.250%
110 100.000% Precedence(send draft, organize meeting) 100.000%
111 87.500% CoExistence(send draft, organize meeting) 37.500%
112 89.091% NotChainSuccession(send draft, submit report) 45.455%
113 81.250% RespondedExistence(send draft, submit report) 6.250%
114 94.872% Precedence(send draft, submit report) 74.359%
115 94.545% CoExistence(send draft, submit report) 72.727%
116 85.455% Succession(send draft, submit report) 27.273%
117 100.000% NotChainSuccession(send draft, submit draft) 100.000%
118 100.000% Precedence(send draft, submit draft) 100.000%
119 100.000% AlternatePrecedence(send draft, submit draft) 100.000%
120 100.000% NotChainSuccession(send draft, send demo) 100.000%
121 100.000% Precedence(send draft, send demo) 100.000%
122 100.000% AlternatePrecedence(send draft, send demo) 100.000%
123 100.000% NotChainSuccession(send draft, organize agenda) 100.000%
124 100.000% NotChainSuccession(send draft, organize demo) 100.000%
125 100.000% Precedence(send draft, organize demo) 100.000%
126 100.000% AlternatePrecedence(send draft, organize demo) 100.000%
127
128 }
129
130 [write deliverable] => {
131 100.000% NotChainSuccession(write deliverable, send agenda) 100.000%
132 100.000% RespondedExistence(write deliverable, send agenda) 100.000%
133 90.000% Precedence(write deliverable, send agenda) 50.000%
134 100.000% CoExistence(write deliverable, send agenda) 100.000%
135 100.000% NotChainSuccession(write deliverable, send meeting) 100.000%
136 100.000% RespondedExistence(write deliverable, send meeting) 100.000%
137 90.909% Response(write deliverable, send meeting) 54.545%
138 83.333% Precedence(write deliverable, send meeting) 16.667%
139 100.000% CoExistence(write deliverable, send meeting) 100.000%
140 86.957% Succession(write deliverable, send meeting) 34.783%
141 92.593% NotChainSuccession(write deliverable, send draft) 62.963%
142 100.000% RespondedExistence(write deliverable, send draft) 100.000%
143 88.889% CoExistence(write deliverable, send draft) 44.444%
144 81.818% RespondedExistence(write deliverable, send report) 9.091%
145 81.818% Response(write deliverable, send report) 9.091%
146 84.615% NotChainSuccession(write deliverable, submit deliverable) 23.077%
147 100.000% RespondedExistence(write deliverable, submit deliverable) 100.000%
148 100.000% Response(write deliverable, submit deliverable) 100.000%
149 100.000% Precedence(write deliverable, submit deliverable) 100.000%
150 100.000% CoExistence(write deliverable, submit deliverable) 100.000%
151 100.000% Succession(write deliverable, submit deliverable) 100.000%
152 91.304% NotChainSuccession(write deliverable, send deliverable) 56.522%
153 100.000% RespondedExistence(write deliverable, send deliverable) 100.000%
154 90.909% Response(write deliverable, send deliverable) 54.545%
155 86.957% CoExistence(write deliverable, send deliverable) 34.783%
156 82.609% Succession(write deliverable, send deliverable) 13.043%
157 100.000% NotChainSuccession(write deliverable, organize meeting) 100.000%
158 100.000% RespondedExistence(write deliverable, organize meeting) 100.000%
159 90.909% Response(write deliverable, organize meeting) 54.545%
160 100.000% Precedence(write deliverable, organize meeting) 100.000%
161 100.000% CoExistence(write deliverable, organize meeting) 100.000%
162 94.737% Succession(write deliverable, organize meeting) 73.684%
163 100.000% NotChainSuccession(write deliverable, submit report) 100.000%
164 100.000% NotChainSuccession(write deliverable, submit draft) 100.000%
165 83.333% NotChainSuccession(write deliverable, send demo) 16.667%

```

```

166 100.000% Precedence(write deliverable, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
167 100.000% AlternatePrecedence(write deliverable, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
168 100.000% ChainPrecedence(write deliverable, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
169 100.000% NotChainSuccession(write deliverable, organize agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
170 100.000% NotChainSuccession(write deliverable, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
171 100.000% Precedence(write deliverable, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
172 100.000% AlternatePrecedence(write deliverable, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
173
174 }
175
176 [send report] => {
177 100.000% NotChainSuccession(send report, send agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
178 90.476% NotChainSuccession(send report, send meeting) 52.381% ||||| conf.: 0.679; int'f: 0.170;
179 92.000% NotChainSuccession(send report, send draft) 60.000% ||||| conf.: 0.690; int'f: 0.173;
180 100.000% RespondedExistence(send report, send draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
181 84.000% CoExistence(send report, send draft) 20.000% | conf.: 0.630; int'f: 0.158;
182 80.000% NotChainSuccession(send report, write deliverable) 0.000% | conf.: 0.600; int'f: 0.150;
183 100.000% NotChainSuccession(send report, submit deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
184 86.667% Precedence(send report, submit deliverable) 33.333% ||| conf.: 0.650; int'f: 0.163;
185 90.476% NotChainSuccession(send report, send deliverable) 52.381% ||||| conf.: 0.679; int'f: 0.170;
186 100.000% RespondedExistence(send report, send deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
187 90.476% CoExistence(send report, send deliverable) 52.381% ||||| conf.: 0.679; int'f: 0.170;
188 100.000% NotChainSuccession(send report, organize meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
189 95.833% NotChainSuccession(send report, submit report) 79.167% ||||| conf.: 0.719; int'f: 0.180;
190 100.000% NotChainSuccession(send report, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
191 100.000% Precedence(send report, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
192 100.000% AlternatePrecedence(send report, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
193 100.000% NotChainSuccession(send report, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
194 100.000% Precedence(send report, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
195 100.000% AlternatePrecedence(send report, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
196 100.000% NotChainSuccession(send report, organize agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
197 100.000% NotChainSuccession(send report, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
198 100.000% Precedence(send report, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
199 100.000% AlternatePrecedence(send report, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
200
201 }
202
203 [submit deliverable] => {
204 92.000% NotChainSuccession(submit deliverable, send agenda) 60.000% ||||| conf.: 0.690; int'f: 0.173;
205 100.000% RespondedExistence(submit deliverable, send agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
206 100.000% CoExistence(submit deliverable, send agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
207 100.000% NotChainSuccession(submit deliverable, send meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
208 100.000% RespondedExistence(submit deliverable, send meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
209 86.667% Response(submit deliverable, send meeting) 33.333% ||| conf.: 0.650; int'f: 0.163;
210 100.000% CoExistence(submit deliverable, send meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
211 81.481% Succession(submit deliverable, send meeting) 7.407% | conf.: 0.611; int'f: 0.153;
212 100.000% NotChainSuccession(submit deliverable, send draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
213 100.000% RespondedExistence(submit deliverable, send draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
214 90.323% CoExistence(submit deliverable, send draft) 51.613% ||||| conf.: 0.677; int'f: 0.169;
215 100.000% NotChainSuccession(submit deliverable, write deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
216 92.308% NotSuccession(submit deliverable, write deliverable) 61.538% ||||| conf.: 0.692; int'f: 0.173;
217 100.000% RespondedExistence(submit deliverable, write deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
218 100.000% CoExistence(submit deliverable, write deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
219 100.000% NotChainSuccession(submit deliverable, send report) 100.000% ||||| conf.: 0.750; int'f: 0.188;
220 86.667% RespondedExistence(submit deliverable, send report) 33.333% ||| conf.: 0.650; int'f: 0.163;
221 86.667% Response(submit deliverable, send report) 33.333% ||| conf.: 0.650; int'f: 0.163;
222 92.593% NotChainSuccession(submit deliverable, send deliverable) 62.963% ||||| conf.: 0.694; int'f: 0.174;

```

```

223 100.000% RespondedExistence(submit deliverable, send deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
224 88.889% CoExistence(submit deliverable, send deliverable) 44.444% ||| conf.: 0.667; int'f: 0.167;
225 91.304% NotChainSuccession(submit deliverable, organize meeting) 56.522% ||||| conf.: 0.685; int'f: 0.171;
226 100.000% RespondedExistence(submit deliverable, organize meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
227 86.667% Response(submit deliverable, organize meeting) 33.333% ||| conf.: 0.650; int'f: 0.163;
228 100.000% CoExistence(submit deliverable, organize meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
229 92.593% NotChainSuccession(submit deliverable, submit report) 62.963% ||||| conf.: 0.694; int'f: 0.174;
230 100.000% NotChainSuccession(submit deliverable, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
231 100.000% NotSuccession(submit deliverable, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
232 100.000% NotChainSuccession(submit deliverable, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
233 100.000% NotSuccession(submit deliverable, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
234 100.000% NotChainSuccession(submit deliverable, organize agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
235 100.000% NotChainSuccession(submit deliverable, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
236 100.000% Precedence(submit deliverable, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
237 100.000% AlternatePrecedence(submit deliverable, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
238
239 }
240
241 [send deliverable] => {
242 100.000% Participation(send deliverable) 100.000% ||||| conf.: 1.000; int'f: 1.000;
243 90.909% NotChainSuccession(send deliverable, send agenda) 54.545% ||||| conf.: 0.909; int'f: 0.227;
244 90.000% Precedence(send deliverable, send agenda) 50.000% ||||| conf.: 0.900; int'f: 0.225;
245 86.364% CoExistence(send deliverable, send agenda) 31.818% ||| conf.: 0.864; int'f: 0.216;
246 100.000% NotChainSuccession(send deliverable, send meeting) 100.000% ||||| conf.: 1.000; int'f: 0.250;
247 83.333% Precedence(send deliverable, send meeting) 16.667% | conf.: 0.833; int'f: 0.208;
248 87.500% CoExistence(send deliverable, send meeting) 37.500% ||| conf.: 0.875; int'f: 0.219;
249 100.000% NotChainSuccession(send deliverable, send draft) 100.000% ||||| conf.: 1.000; int'f: 0.250;
250 100.000% RespondedExistence(send deliverable, send draft) 100.000% ||||| conf.: 1.000; int'f: 0.250;
251 100.000% CoExistence(send deliverable, send draft) 100.000% ||||| conf.: 1.000; int'f: 0.250;
252 91.304% NotChainSuccession(send deliverable, write deliverable) 56.522% ||||| conf.: 0.913; int'f: 0.228;
253 86.957% CoExistence(send deliverable, write deliverable) 34.783% ||| conf.: 0.870; int'f: 0.217;
254 100.000% NotChainSuccession(send deliverable, send report) 100.000% ||||| conf.: 1.000; int'f: 0.250;
255 83.333% RespondedExistence(send deliverable, send report) 16.667% | conf.: 0.833; int'f: 0.208;
256 90.476% CoExistence(send deliverable, send report) 52.381% ||||| conf.: 0.905; int'f: 0.226;
257 100.000% NotChainSuccession(send deliverable, submit deliverable) 100.000% ||||| conf.: 1.000; int'f: 0.250;
258 93.333% Precedence(send deliverable, submit deliverable) 66.667% ||||| conf.: 0.933; int'f: 0.233;
259 88.889% CoExistence(send deliverable, submit deliverable) 44.444% ||||| conf.: 0.889; int'f: 0.222;
260 100.000% NotChainSuccession(send deliverable, organize meeting) 100.000% ||||| conf.: 1.000; int'f: 0.250;
261 100.000% Precedence(send deliverable, organize meeting) 100.000% ||||| conf.: 1.000; int'f: 0.250;
262 85.000% CoExistence(send deliverable, organize meeting) 25.000% ||| conf.: 0.850; int'f: 0.213;
263 88.235% NotChainSuccession(send deliverable, submit report) 41.176% ||||| conf.: 0.882; int'f: 0.221;
264 91.667% RespondedExistence(send deliverable, submit report) 58.333% ||||| conf.: 0.917; int'f: 0.229;
265 98.039% CoExistence(send deliverable, submit report) 90.196% ||||| conf.: 0.980; int'f: 0.245;
266 100.000% NotChainSuccession(send deliverable, submit draft) 100.000% ||||| conf.: 1.000; int'f: 0.250;
267 100.000% NotSuccession(send deliverable, submit draft) 100.000% ||||| conf.: 1.000; int'f: 0.250;
268 100.000% NotChainSuccession(send deliverable, send demo) 100.000% ||||| conf.: 1.000; int'f: 0.250;
269 100.000% Precedence(send deliverable, send demo) 100.000% ||||| conf.: 1.000; int'f: 0.250;
270 100.000% AlternatePrecedence(send deliverable, send demo) 100.000% ||||| conf.: 1.000; int'f: 0.250;
271 100.000% NotChainSuccession(send deliverable, organize demo) 100.000% ||||| conf.: 1.000; int'f: 0.250;
272 100.000% Precedence(send deliverable, organize demo) 100.000% ||||| conf.: 1.000; int'f: 0.250;
273 100.000% AlternatePrecedence(send deliverable, organize demo) 100.000% ||||| conf.: 1.000; int'f: 0.250;
274
275 }
276
277 [organize meeting] => {
278 88.889% NotChainSuccession(organize meeting, send agenda) 44.444% ||||| conf.: 0.667; int'f: 0.167;
279 100.000% RespondedExistence(organize meeting, send agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;

```

```

280 100.000% CoExistence(organize meeting, send agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
281 100.000% NotChainSuccession(organize meeting, send meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
282 100.000% RespondedExistence(organize meeting, send meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
283 100.000% CoExistence(organize meeting, send meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
284 91.667% NotChainSuccession(organize meeting, send draft) 58.333% ||||| conf.: 0.688; int'f: 0.172;
285 100.000% RespondedExistence(organize meeting, send draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
286 87.500% CoExistence(organize meeting, send draft) 37.500% ||| conf.: 0.656; int'f: 0.164;
287 100.000% NotChainSuccession(organize meeting, write deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
288 100.000% RespondedExistence(organize meeting, write deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
289 100.000% CoExistence(organize meeting, write deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
290 88.235% NotChainSuccession(organize meeting, send report) 41.176% ||||| conf.: 0.662; int'f: 0.165;
291 100.000% NotChainSuccession(organize meeting, submit deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
292 100.000% RespondedExistence(organize meeting, submit deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
293 100.000% CoExistence(organize meeting, submit deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
294 100.000% NotChainSuccession(organize meeting, send deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
295 100.000% RespondedExistence(organize meeting, send deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
296 85.000% CoExistence(organize meeting, send deliverable) 25.000% ||| conf.: 0.638; int'f: 0.159;
297 100.000% NotChainSuccession(organize meeting, submit report) 100.000% ||||| conf.: 0.750; int'f: 0.188;
298 100.000% NotChainSuccession(organize meeting, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
299 100.000% NotSuccession(organize meeting, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
300 100.000% NotChainSuccession(organize meeting, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
301 100.000% NotSuccession(organize meeting, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
302 100.000% NotChainSuccession(organize meeting, organize agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
303 100.000% NotSuccession(organize meeting, organize agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
304 81.818% CoExistence(organize meeting, organize agenda) 9.091% ||| conf.: 0.614; int'f: 0.153;
305 100.000% NotChainSuccession(organize meeting, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
306 100.000% Precedence(organize meeting, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
307 100.000% AlternatePrecedence(organize meeting, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
308
309 }
310
311 [submit report] => {
312 100.000% NotChainSuccession(submit report, send agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
313 81.633% CoExistence(submit report, send agenda) 8.163% ||| conf.: 0.612; int'f: 0.153;
314 100.000% NotChainSuccession(submit report, send meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
315 89.091% NotChainSuccession(submit report, send draft) 45.455% ||||| conf.: 0.668; int'f: 0.167;
316 100.000% RespondedExistence(submit report, send draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
317 94.545% CoExistence(submit report, send draft) 72.727% ||||| conf.: 0.709; int'f: 0.177;
318 96.000% NotChainSuccession(submit report, write deliverable) 80.000% ||||| conf.: 0.720; int'f: 0.180;
319 95.833% NotChainSuccession(submit report, send report) 79.167% ||||| conf.: 0.719; int'f: 0.180;
320 92.593% NotChainSuccession(submit report, submit deliverable) 62.963% ||||| conf.: 0.694; int'f: 0.174;
321 88.235% NotChainSuccession(submit report, send deliverable) 41.176% ||||| conf.: 0.662; int'f: 0.165;
322 100.000% RespondedExistence(submit report, send deliverable) 100.000% ||||| conf.: 0.750; int'f: 0.188;
323 98.039% CoExistence(submit report, send deliverable) 90.196% ||||| conf.: 0.735; int'f: 0.184;
324 100.000% NotChainSuccession(submit report, organize meeting) 100.000% ||||| conf.: 0.750; int'f: 0.188;
325 90.244% NotChainSuccession(submit report, submit draft) 51.220% ||||| conf.: 0.677; int'f: 0.169;
326 100.000% Precedence(submit report, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
327 100.000% AlternatePrecedence(submit report, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
328 100.000% ChainPrecedence(submit report, submit draft) 100.000% ||||| conf.: 0.750; int'f: 0.188;
329 100.000% NotChainSuccession(submit report, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
330 100.000% Precedence(submit report, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
331 100.000% AlternatePrecedence(submit report, send demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
332 100.000% NotChainSuccession(submit report, organize agenda) 100.000% ||||| conf.: 0.750; int'f: 0.188;
333 80.952% CoExistence(submit report, organize agenda) 4.762% ||| conf.: 0.607; int'f: 0.152;
334 100.000% NotChainSuccession(submit report, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
335 100.000% Precedence(submit report, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;
336 100.000% AlternatePrecedence(submit report, organize demo) 100.000% ||||| conf.: 0.750; int'f: 0.188;

```

```

337
338 }
339
340 [submit draft] => {
341     100.000% NotChainSuccession(submit draft, send agenda)      100.000% ||||||||| conf.: 0.250; int'f: 0.063;
342     100.000% RespondedExistence(submit draft, send agenda)    100.000% ||||||||| conf.: 0.250; int'f: 0.063;
343     100.000% Response(submit draft, send agenda)              100.000% ||||||||| conf.: 0.250; int'f: 0.063;
344     100.000% NotChainSuccession(submit draft, send meeting)   100.000% ||||||||| conf.: 0.250; int'f: 0.063;
345     100.000% RespondedExistence(submit draft, send meeting)  100.000% ||||||||| conf.: 0.250; int'f: 0.063;
346     100.000% Response(submit draft, send meeting)            100.000% ||||||||| conf.: 0.250; int'f: 0.063;
347     100.000% NotChainSuccession(submit draft, send draft)    100.000% ||||||||| conf.: 0.250; int'f: 0.063;
348     100.000% RespondedExistence(submit draft, send draft)    100.000% ||||||||| conf.: 0.250; int'f: 0.063;
349     100.000% Response(submit draft, send draft)              100.000% ||||||||| conf.: 0.250; int'f: 0.063;
350     100.000% AlternateResponse(submit draft, send draft)     100.000% ||||||||| conf.: 0.250; int'f: 0.063;
351     84.615% NotChainSuccession(submit draft, write deliverable) 23.077% ||         conf.: 0.212; int'f: 0.053;
352     100.000% RespondedExistence(submit draft, write deliverable) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
353     100.000% Response(submit draft, write deliverable)        100.000% ||||||||| conf.: 0.250; int'f: 0.063;
354     100.000% AlternateResponse(submit draft, write deliverable) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
355     100.000% NotChainSuccession(submit draft, send report)    100.000% ||||||||| conf.: 0.250; int'f: 0.063;
356     100.000% RespondedExistence(submit draft, send report)   100.000% ||||||||| conf.: 0.250; int'f: 0.063;
357     100.000% Response(submit draft, send report)              100.000% ||||||||| conf.: 0.250; int'f: 0.063;
358     100.000% AlternateResponse(submit draft, send report)     100.000% ||||||||| conf.: 0.250; int'f: 0.063;
359     100.000% NotChainSuccession(submit draft, submit deliverable) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
360     100.000% RespondedExistence(submit draft, submit deliverable) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
361     100.000% Response(submit draft, submit deliverable)       100.000% ||||||||| conf.: 0.250; int'f: 0.063;
362     100.000% NotChainSuccession(submit draft, send deliverable) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
363     100.000% RespondedExistence(submit draft, send deliverable) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
364     100.000% Response(submit draft, send deliverable)         100.000% ||||||||| conf.: 0.250; int'f: 0.063;
365     100.000% NotChainSuccession(submit draft, organize meeting) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
366     100.000% RespondedExistence(submit draft, organize meeting) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
367     100.000% Response(submit draft, organize meeting)         100.000% ||||||||| conf.: 0.250; int'f: 0.063;
368     95.122% NotChainSuccession(submit draft, submit report)   75.610% ||         conf.: 0.238; int'f: 0.059;
369     100.000% RespondedExistence(submit draft, submit report)  100.000% ||||||||| conf.: 0.250; int'f: 0.063;
370     100.000% Response(submit draft, submit report)            100.000% ||||||||| conf.: 0.250; int'f: 0.063;
371     100.000% AlternateResponse(submit draft, submit report)   100.000% ||||||||| conf.: 0.250; int'f: 0.063;
372     100.000% NotChainSuccession(submit draft, send demo)      100.000% ||||||||| conf.: 0.250; int'f: 0.063;
373     100.000% RespondedExistence(submit draft, send demo)     100.000% ||||||||| conf.: 0.250; int'f: 0.063;
374     100.000% Response(submit draft, send demo)                100.000% ||||||||| conf.: 0.250; int'f: 0.063;
375     100.000% Precedence(submit draft, send demo)              100.000% ||||||||| conf.: 0.250; int'f: 0.063;
376     100.000% AlternatePrecedence(submit draft, send demo)    100.000% ||||||||| conf.: 0.250; int'f: 0.063;
377     100.000% CoExistence(submit draft, send demo)            100.000% ||||||||| conf.: 0.250; int'f: 0.063;
378     100.000% Succession(submit draft, send demo)              100.000% ||||||||| conf.: 0.250; int'f: 0.063;
379     100.000% NotChainSuccession(submit draft, organize agenda) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
380     100.000% RespondedExistence(submit draft, organize agenda) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
381     100.000% Response(submit draft, organize agenda)          100.000% ||||||||| conf.: 0.250; int'f: 0.063;
382     100.000% NotChainSuccession(submit draft, organize demo)  100.000% ||||||||| conf.: 0.250; int'f: 0.063;
383     100.000% RespondedExistence(submit draft, organize demo)  100.000% ||||||||| conf.: 0.250; int'f: 0.063;
384     100.000% Response(submit draft, organize demo)            100.000% ||||||||| conf.: 0.250; int'f: 0.063;
385     100.000% Precedence(submit draft, organize demo)         100.000% ||||||||| conf.: 0.250; int'f: 0.063;
386     100.000% AlternatePrecedence(submit draft, organize demo) 100.000% ||||||||| conf.: 0.250; int'f: 0.063;
387     100.000% CoExistence(submit draft, organize demo)        100.000% ||||||||| conf.: 0.250; int'f: 0.063;
388     100.000% Succession(submit draft, organize demo)          100.000% ||||||||| conf.: 0.250; int'f: 0.063;
389
390 }
391
392 [send demo] => {
393     100.000% Uniqueness(send demo)                            100.000% ||||||||| conf.: 0.250; int'f: 0.063;

```

394	100.000%	NotChainSuccession(send demo, send agenda)	100.000%		conf.:	0.250;	int'f:	0.063;
395	100.000%	RespondedExistence(send demo, send agenda)	100.000%		conf.:	0.250;	int'f:	0.063;
396	100.000%	Response(send demo, send agenda)	100.000%		conf.:	0.250;	int'f:	0.063;
397	100.000%	AlternateResponse(send demo, send agenda)	100.000%		conf.:	0.250;	int'f:	0.063;
398	100.000%	NotChainSuccession(send demo, send meeting)	100.000%		conf.:	0.250;	int'f:	0.063;
399	100.000%	RespondedExistence(send demo, send meeting)	100.000%		conf.:	0.250;	int'f:	0.063;
400	100.000%	Response(send demo, send meeting)	100.000%		conf.:	0.250;	int'f:	0.063;
401	100.000%	AlternateResponse(send demo, send meeting)	100.000%		conf.:	0.250;	int'f:	0.063;
402	100.000%	NotChainSuccession(send demo, send draft)	100.000%		conf.:	0.250;	int'f:	0.063;
403	100.000%	RespondedExistence(send demo, send draft)	100.000%		conf.:	0.250;	int'f:	0.063;
404	100.000%	Response(send demo, send draft)	100.000%		conf.:	0.250;	int'f:	0.063;
405	100.000%	AlternateResponse(send demo, send draft)	100.000%		conf.:	0.250;	int'f:	0.063;
406	100.000%	NotChainSuccession(send demo, write deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
407	100.000%	NotSuccession(send demo, write deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
408	100.000%	RespondedExistence(send demo, write deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
409	100.000%	NotChainSuccession(send demo, send report)	100.000%		conf.:	0.250;	int'f:	0.063;
410	80.000%	NotSuccession(send demo, send report)	0.000%		conf.:	0.200;	int'f:	0.050;
411	100.000%	RespondedExistence(send demo, send report)	100.000%		conf.:	0.250;	int'f:	0.063;
412	100.000%	Response(send demo, send report)	100.000%		conf.:	0.250;	int'f:	0.063;
413	100.000%	AlternateResponse(send demo, send report)	100.000%		conf.:	0.250;	int'f:	0.063;
414	87.500%	NotChainSuccession(send demo, submit deliverable)	37.500%		conf.:	0.219;	int'f:	0.055;
415	100.000%	RespondedExistence(send demo, submit deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
416	100.000%	Response(send demo, submit deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
417	100.000%	AlternateResponse(send demo, submit deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
418	100.000%	ChainResponse(send demo, submit deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
419	100.000%	NotChainSuccession(send demo, send deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
420	100.000%	RespondedExistence(send demo, send deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
421	100.000%	Response(send demo, send deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
422	100.000%	AlternateResponse(send demo, send deliverable)	100.000%		conf.:	0.250;	int'f:	0.063;
423	100.000%	NotChainSuccession(send demo, organize meeting)	100.000%		conf.:	0.250;	int'f:	0.063;
424	100.000%	RespondedExistence(send demo, organize meeting)	100.000%		conf.:	0.250;	int'f:	0.063;
425	100.000%	Response(send demo, organize meeting)	100.000%		conf.:	0.250;	int'f:	0.063;
426	100.000%	AlternateResponse(send demo, organize meeting)	100.000%		conf.:	0.250;	int'f:	0.063;
427	100.000%	NotChainSuccession(send demo, submit report)	100.000%		conf.:	0.250;	int'f:	0.063;
428	90.000%	NotSuccession(send demo, submit report)	50.000%		conf.:	0.225;	int'f:	0.056;
429	100.000%	RespondedExistence(send demo, submit report)	100.000%		conf.:	0.250;	int'f:	0.063;
430	100.000%	Response(send demo, submit report)	100.000%		conf.:	0.250;	int'f:	0.063;
431	100.000%	AlternateResponse(send demo, submit report)	100.000%		conf.:	0.250;	int'f:	0.063;
432	100.000%	NotChainSuccession(send demo, submit draft)	100.000%		conf.:	0.250;	int'f:	0.063;
433	100.000%	NotSuccession(send demo, submit draft)	100.000%		conf.:	0.250;	int'f:	0.063;
434	100.000%	RespondedExistence(send demo, submit draft)	100.000%		conf.:	0.250;	int'f:	0.063;
435	100.000%	CoExistence(send demo, submit draft)	100.000%		conf.:	0.250;	int'f:	0.063;
436	100.000%	NotChainSuccession(send demo, organize agenda)	100.000%		conf.:	0.250;	int'f:	0.063;
437	100.000%	RespondedExistence(send demo, organize agenda)	100.000%		conf.:	0.250;	int'f:	0.063;
438	100.000%	Response(send demo, organize agenda)	100.000%		conf.:	0.250;	int'f:	0.063;
439	100.000%	AlternateResponse(send demo, organize agenda)	100.000%		conf.:	0.250;	int'f:	0.063;
440	100.000%	NotChainSuccession(send demo, organize demo)	100.000%		conf.:	0.250;	int'f:	0.063;
441	100.000%	RespondedExistence(send demo, organize demo)	100.000%		conf.:	0.250;	int'f:	0.063;
442	100.000%	Response(send demo, organize demo)	100.000%		conf.:	0.250;	int'f:	0.063;
443	100.000%	AlternateResponse(send demo, organize demo)	100.000%		conf.:	0.250;	int'f:	0.063;
444	100.000%	Precedence(send demo, organize demo)	100.000%		conf.:	0.250;	int'f:	0.063;
445	100.000%	AlternatePrecedence(send demo, organize demo)	100.000%		conf.:	0.250;	int'f:	0.063;
446	100.000%	CoExistence(send demo, organize demo)	100.000%		conf.:	0.250;	int'f:	0.063;
447	100.000%	Succession(send demo, organize demo)	100.000%		conf.:	0.250;	int'f:	0.063;
448	100.000%	AlternateSuccession(send demo, organize demo)	100.000%		conf.:	0.250;	int'f:	0.063;
449								
450	}							

```

451
452 [organize agenda] => {
453     84.615% NotChainSuccession(organize agenda, send agenda)      23.077% ||          conf.: 0.423; int'f: 0.106;
454     100.000% RespondedExistence(organize agenda, send agenda)    100.000% ||||||||| conf.: 0.500; int'f: 0.125;
455     100.000% Response(organize agenda, send agenda)               100.000% ||||||||| conf.: 0.500; int'f: 0.125;
456     92.308% CoExistence(organize agenda, send agenda)             61.538% |||||||   conf.: 0.462; int'f: 0.115;
457     100.000% NotChainSuccession(organize agenda, send meeting)    100.000% ||||||||| conf.: 0.500; int'f: 0.125;
458     100.000% RespondedExistence(organize agenda, send meeting)    100.000% ||||||||| conf.: 0.500; int'f: 0.125;
459     100.000% Response(organize agenda, send meeting)              100.000% ||||||||| conf.: 0.500; int'f: 0.125;
460     100.000% AlternateResponse(organize agenda, send meeting)     100.000% ||||||||| conf.: 0.500; int'f: 0.125;
461     80.000% CoExistence(organize agenda, send meeting)            0.000%   conf.: 0.400; int'f: 0.100;
462     80.000% Succession(organize agenda, send meeting)             0.000%   conf.: 0.400; int'f: 0.100;
463     89.474% NotChainSuccession(organize agenda, send draft)       47.368% ||||      conf.: 0.447; int'f: 0.112;
464     100.000% RespondedExistence(organize agenda, send draft)     100.000% ||||||||| conf.: 0.500; int'f: 0.125;
465     100.000% Response(organize agenda, send draft)                 100.000% ||||||||| conf.: 0.500; int'f: 0.125;
466     100.000% AlternateResponse(organize agenda, send draft)       100.000% ||||||||| conf.: 0.500; int'f: 0.125;
467     100.000% NotChainSuccession(organize agenda, write deliverable) 100.000% ||||||||| conf.: 0.500; int'f: 0.125;
468     100.000% RespondedExistence(organize agenda, write deliverable) 100.000% ||||||||| conf.: 0.500; int'f: 0.125;
469     100.000% NotChainSuccession(organize agenda, send report)     100.000% ||||||||| conf.: 0.500; int'f: 0.125;
470     83.333% NotSuccession(organize agenda, send report)           16.667% |         conf.: 0.417; int'f: 0.104;
471     100.000% NotChainSuccession(organize agenda, submit deliverable) 100.000% ||||||||| conf.: 0.500; int'f: 0.125;
472     100.000% RespondedExistence(organize agenda, submit deliverable) 100.000% ||||||||| conf.: 0.500; int'f: 0.125;
473     100.000% NotChainSuccession(organize agenda, send deliverable) 100.000% ||||||||| conf.: 0.500; int'f: 0.125;
474     100.000% RespondedExistence(organize agenda, send deliverable) 100.000% ||||||||| conf.: 0.500; int'f: 0.125;
475     100.000% Response(organize agenda, send deliverable)          100.000% ||||||||| conf.: 0.500; int'f: 0.125;
476     100.000% AlternateResponse(organize agenda, send deliverable) 100.000% ||||||||| conf.: 0.500; int'f: 0.125;
477     81.818% NotChainSuccession(organize agenda, organize meeting) 9.091%   conf.: 0.409; int'f: 0.102;
478     100.000% RespondedExistence(organize agenda, organize meeting) 100.000% ||||||||| conf.: 0.500; int'f: 0.125;
479     100.000% Response(organize agenda, organize meeting)           100.000% ||||||||| conf.: 0.500; int'f: 0.125;
480     81.818% CoExistence(organize agenda, organize meeting)        9.091%   conf.: 0.409; int'f: 0.102;
481     81.818% Succession(organize agenda, organize meeting)          9.091%   conf.: 0.409; int'f: 0.102;
482     100.000% NotChainSuccession(organize agenda, submit report)    100.000% ||||||||| conf.: 0.500; int'f: 0.125;
483     100.000% RespondedExistence(organize agenda, submit report)    100.000% ||||||||| conf.: 0.500; int'f: 0.125;
484     80.952% CoExistence(organize agenda, submit report)           4.762%   conf.: 0.405; int'f: 0.101;
485     100.000% NotChainSuccession(organize agenda, submit draft)     100.000% ||||||||| conf.: 0.500; int'f: 0.125;
486     100.000% NotSuccession(organize agenda, submit draft)          100.000% ||||||||| conf.: 0.500; int'f: 0.125;
487     100.000% NotChainSuccession(organize agenda, send demo)        100.000% ||||||||| conf.: 0.500; int'f: 0.125;
488     100.000% NotSuccession(organize agenda, send demo)             100.000% ||||||||| conf.: 0.500; int'f: 0.125;
489     100.000% NotChainSuccession(organize agenda, organize demo)    100.000% ||||||||| conf.: 0.500; int'f: 0.125;
490     100.000% Precedence(organize agenda, organize demo)            100.000% ||||||||| conf.: 0.500; int'f: 0.125;
491     100.000% AlternatePrecedence(organize agenda, organize demo)  100.000% ||||||||| conf.: 0.500; int'f: 0.125;
492
493 }
494
495 [organize demo] => {
496     100.000% Uniqueness(organize demo)                             100.000% ||||||||| conf.: 0.250; int'f: 0.063;
497     100.000% NotChainSuccession(organize demo, send agenda)        100.000% ||||||||| conf.: 0.250; int'f: 0.125;
498     81.818% NotSuccession(organize demo, send agenda)              9.091%   conf.: 0.205; int'f: 0.102;
499     100.000% RespondedExistence(organize demo, send agenda)        100.000% ||||||||| conf.: 0.250; int'f: 0.125;
500     100.000% Response(organize demo, send agenda)                  100.000% ||||||||| conf.: 0.250; int'f: 0.125;
501     100.000% AlternateResponse(organize demo, send agenda)         100.000% ||||||||| conf.: 0.250; int'f: 0.125;
502     84.615% NotChainSuccession(organize demo, send meeting)        23.077% ||          conf.: 0.212; int'f: 0.106;
503     100.000% RespondedExistence(organize demo, send meeting)      100.000% ||||||||| conf.: 0.250; int'f: 0.125;
504     100.000% Response(organize demo, send meeting)                 100.000% ||||||||| conf.: 0.250; int'f: 0.125;
505     100.000% AlternateResponse(organize demo, send meeting)        100.000% ||||||||| conf.: 0.250; int'f: 0.125;
506     100.000% ChainResponse(organize demo, send meeting)            100.000% ||||||||| conf.: 0.250; int'f: 0.125;
507     100.000% NotChainSuccession(organize demo, send draft)         100.000% ||||||||| conf.: 0.250; int'f: 0.125;

```



```

508      82.353% NotSuccession(organize demo, send draft)          11.765% |          conf.: 0.206; int'f: 0.103;
509      100.000% RespondedExistence(organize demo, send draft) 100.000% |          conf.: 0.250; int'f: 0.125;
510      100.000% Response(organize demo, send draft)             100.000% |          conf.: 0.250; int'f: 0.125;
511      100.000% AlternateResponse(organize demo, send draft)    100.000% |          conf.: 0.250; int'f: 0.125;
512      100.000% NotChainSuccession(organize demo, write deliverable) 100.000% |          conf.: 0.250; int'f: 0.125;
513      100.000% NotSuccession(organize demo, write deliverable) 100.000% |          conf.: 0.250; int'f: 0.125;
514      100.000% RespondedExistence(organize demo, write deliverable) 100.000% |          conf.: 0.250; int'f: 0.125;
515      100.000% NotChainSuccession(organize demo, send report)   100.000% |          conf.: 0.250; int'f: 0.125;
516      80.000% NotSuccession(organize demo, send report)         0.000%   |          conf.: 0.200; int'f: 0.100;
517      100.000% RespondedExistence(organize demo, send report) 100.000% |          conf.: 0.250; int'f: 0.125;
518      100.000% Response(organize demo, send report)             100.000% |          conf.: 0.250; int'f: 0.125;
519      100.000% AlternateResponse(organize demo, send report)    100.000% |          conf.: 0.250; int'f: 0.125;
520      100.000% NotChainSuccession(organize demo, submit deliverable) 100.000% |          conf.: 0.250; int'f: 0.125;
521      100.000% NotSuccession(organize demo, submit deliverable) 100.000% |          conf.: 0.250; int'f: 0.125;
522      100.000% RespondedExistence(organize demo, submit deliverable) 100.000% |          conf.: 0.250; int'f: 0.125;
523      100.000% NotChainSuccession(organize demo, send deliverable) 100.000% |          conf.: 0.250; int'f: 0.125;
524      100.000% RespondedExistence(organize demo, send deliverable) 100.000% |          conf.: 0.250; int'f: 0.125;
525      100.000% Response(organize demo, send deliverable)       100.000% |          conf.: 0.250; int'f: 0.125;
526      100.000% AlternateResponse(organize demo, send deliverable) 100.000% |          conf.: 0.250; int'f: 0.125;
527      100.000% NotChainSuccession(organize demo, organize meeting) 100.000% |          conf.: 0.250; int'f: 0.125;
528      100.000% NotSuccession(organize demo, organize meeting) 100.000% |          conf.: 0.250; int'f: 0.125;
529      100.000% RespondedExistence(organize demo, organize meeting) 100.000% |          conf.: 0.250; int'f: 0.125;
530      100.000% NotChainSuccession(organize demo, submit report) 100.000% |          conf.: 0.250; int'f: 0.125;
531      100.000% NotSuccession(organize demo, submit report)     100.000% |          conf.: 0.250; int'f: 0.125;
532      100.000% RespondedExistence(organize demo, submit report) 100.000% |          conf.: 0.250; int'f: 0.125;
533      100.000% NotChainSuccession(organize demo, submit draft) 100.000% |          conf.: 0.250; int'f: 0.125;
534      100.000% NotSuccession(organize demo, submit draft)     100.000% |          conf.: 0.250; int'f: 0.125;
535      100.000% RespondedExistence(organize demo, submit draft) 100.000% |          conf.: 0.250; int'f: 0.125;
536      100.000% CoExistence(organize demo, submit draft)        100.000% |          conf.: 0.250; int'f: 0.125;
537      100.000% NotChainSuccession(organize demo, send demo)     100.000% |          conf.: 0.250; int'f: 0.125;
538      100.000% NotSuccession(organize demo, send demo)         100.000% |          conf.: 0.250; int'f: 0.125;
539      100.000% RespondedExistence(organize demo, send demo)   100.000% |          conf.: 0.250; int'f: 0.125;
540      100.000% CoExistence(organize demo, send demo)          100.000% |          conf.: 0.250; int'f: 0.125;
541      100.000% NotChainSuccession(organize demo, organize agenda) 100.000% |          conf.: 0.250; int'f: 0.125;
542      100.000% NotSuccession(organize demo, organize agenda) 100.000% |          conf.: 0.250; int'f: 0.125;
543      100.000% RespondedExistence(organize demo, organize agenda) 100.000% |          conf.: 0.250; int'f: 0.125;
544
545 }

```


Bibliography

- [1] AGGARWAL, C. C. AND ZHAI, C. (eds.). *Mining Text Data*. Springer (2012). ISBN 978-1-4419-8462-3.
- [2] AGRAWAL, R., GUNOPULOS, D., AND LEYMANN, F. Mining process models from workflow logs. In *Advances in Database Technology – EDBT’98* (edited by H.-J. Schek, G. Alonso, F. Saltor, and I. Ramos), vol. 1377 of *Lecture Notes in Computer Science*, pp. 467–483. Springer Berlin / Heidelberg (1998). 10.1007/BFb0101003. Available from: <http://dx.doi.org/10.1007/BFb0101003>.
- [3] AGRAWAL, R. AND SRIKANT, R. Fast algorithms for mining association rules in large databases. In *VLDB* (edited by J. B. Bocca, M. Jarke, and C. Zaniolo), pp. 487–499. Morgan Kaufmann (1994). ISBN 1-55860-153-8. Available from: <http://www.vldb.org/conf/1994/P487.PDF>.
- [4] ALBERTI, M., CHESANI, F., GAVANELLI, M., LAMMA, E., MELLO, P., AND TORRONI, P. Verifiable agent interaction in abductive logic programming: The SCIFF framework. *ACM Trans. Comput. Log.*, **9** (2008), 29:1. Available from: <http://doi.acm.org/10.1145/1380572.1380578>, doi: 10.1145/1380572.1380578.
- [5] ALONSO, G., DADAM, P., AND ROSEMAN, M. (eds.). *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, vol. 4714 of *Lecture Notes in Computer Science*. Springer (2007). ISBN 978-3-540-75182-3.
- [6] AUSTIN, J. L. *How to do things with words*. Harvard University Press, Cambridge, Mass. (1975).
- [7] BALDONI, R., ET AL. An embedded middleware platform for pervasive and immersive environments for-all. In *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2009, Rome, Italy, June 22-26, 2009*. IEEE (2009). ISBN 978-1-4244-2907-3. doi:10.1109/SAHCNW.2009.5172921.
- [8] BAUER, A., LEUCKER, M., AND SCHALLHART, C. Comparing ltl semantics for runtime verification. *J. Log. Comput.*, **20** (2010), 651. doi:10.1093/logcom/exn075.
- [9] BELLODI, E., RIGUZZI, F., AND LAMMA, E. Probabilistic declarative process mining. In *KSEM* (edited by Y. Bi and M.-A. Williams), vol.

- 6291 of *Lecture Notes in Computer Science*, pp. 292–303. Springer (2010). ISBN 978-3-642-15279-5. Available from: http://dx.doi.org/10.1007/978-3-642-15280-1_28, doi:10.1007/978-3-642-15280-1_28.
- [10] BELLODI, E., RIGUZZI, F., AND LAMMA, E. Probabilistic logic-based process mining. In *CILC* (edited by W. Faber and N. Leone), vol. 598 of *CEUR Workshop Proceedings*. CEUR-WS.org (2010). Available from: <http://ceur-ws.org/Vol-598/paper17.pdf>.
- [11] BIRD, C., GOURLEY, A., DEVANBU, P. T., GERTZ, M., AND SWAMINATHAN, A. Mining email social networks. In *MSR* (edited by S. Diehl, H. Gall, and A. E. Hassan), pp. 137–143. ACM (2006). ISBN 1-59593-397-2. Available from: <http://doi.acm.org/10.1145/1137983.1138016>, doi:10.1145/1137983.1138016.
- [12] BOLDI, P. AND VIGNA, S. MG4J at TREC 2005. In *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings* (edited by E. M. Voorhees and L. P. Buckland), no. SP 500-266 in Special Publications. NIST (2005). <http://mg4j.di.unimi.it/>.
- [13] BUIJS, J. C. A. M., VAN DONGEN, B. F., AND VAN DER AALST, W. M. P. On the role of fitness, precision, generalization and simplicity in process discovery. In Meersman et al. [60]. On the Move to Meaningful Internet Systems (OTM 2012) Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012.
- [14] CARUSO, M., DI CICCIO, C., IACOMUSSI, E., KALDELI, E., LAZOVIK, A., AND MECELLA, M. Service ecologies for home/building automation. In *10th International IFAC Symposium on Robot Control, SYROCO 2012, Dubrovnik, Croatia, September 05-07, 2012* (edited by I. Petrovic and P. Korondi), vol. 10 of *Robot Control*, pp. 467–472. IFAC Papers On Line (2012). doi:10.3182/20120905-3-HR-2030.00191.
- [15] CATARCI, T., DI CICCIO, C., FORTE, V., IACOMUSSI, E., MECELLA, M., SANTUCCI, G., AND TINO, G. Service composition and advanced user interfaces in the home of tomorrow: the SM4All approach. In *2nd International ICST Conference on Ambient Media and Systems, AMBI-SYS 2011, Porto, Portugal, March 24-25, 2011* (edited by S. Gabrielli, D. Elias, and K. Kahol), vol. 70 of *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, pp. 12–19 (2011). ISBN 978-3-642-23901-4. doi:10.1007/978-3-642-23902-1_2.
- [16] CATARCI, T., DIX, A. J., KATIFORI, A., LEPOURAS, G., AND POGGI, A. Task-centred information management. In *DELOS Conference*, vol. 4877 of *Lecture Notes in Computer Science*, pp. 197–206. Springer (2007). ISBN 978-3-540-77087-9.
- [17] CHESANI, F., LAMMA, E., MELLO, P., MONTALI, M., RIGUZZI, F., AND STORARI, S. Exploiting inductive logic programming techniques for declarative process mining. *T. Petri Nets and Other Models of Concurrency*, **2** (2009), 278. Available from: http://dx.doi.org/10.1007/978-3-642-00899-3_16.

- [18] CHESANI, F., MELLO, P., MONTALI, M., AND STORARI, S. Towards a DecSerFlow declarative semantics based on computational logic. Tech. rep., DEIS, Università degli Studi di Bologna (2007).
- [19] CLARKE, E. M., GRUMBERG, O., AND PELED, D. *Model Checking*. MIT Press (2001). ISBN 978-0-262-03270-4.
- [20] COHEN, W. W., CARVALHO, V. R., AND MITCHELL, T. M. Learning to classify email into “speech acts”. In *EMNLP*, pp. 309–316. ACL (2004). Available from: <http://www.aclweb.org/anthology/W04-3240>.
- [21] COOK, J. E. AND WOLF, A. L. Discovering models of software processes from event-based data. *ACM Trans. Softw. Eng. Methodol.*, **7** (1998), 215. Available from: <http://doi.acm.org/10.1145/287000.287001>, doi:10.1145/287000.287001.
- [22] COOK, J. E. AND WOLF, A. L. Event-base detection of concurrency. In *SIGSOFT FSE*, pp. 35–45 (1998). Available from: <http://doi.acm.org/10.1145/288195.288214>, doi:10.1145/288195.288214.
- [23] CORTADELLA, J., KISHINEVSKY, M., LAVAGNO, L., AND YAKOVLEV, A. Deriving petri nets from finite transition systems. *Computers, IEEE Transactions on*, **47** (1998), 859. doi:10.1109/12.707587.
- [24] DAVENPORT, T. H., JARVENPAA, S. L., AND BEERS, M. C. Improving knowledge work processes. *Sloan Management Review*, **37** (1996), 53. Available from: <http://sloanreview.mit.edu/the-magazine/articles/1996/summer/3744/improving-knowledge-work-processes>.
- [25] DE CARVALHO, V. R. AND COHEN, W. W. Learning to extract signature and reply lines from email. In *CEAS* (2004). Available from: <http://www.ceas.cc/papers-2004/135.pdf>.
- [26] DE GIACOMO, G., DI CICCIO, C., FELLI, P., HU, Y., AND MECELLA, M. Goal-based composition of stateful services for smart homes. In *20th International Conference on Cooperative Information Systems, CoopIS 2012, On the Move to Meaningful Internet Systems (OTM 2012) Confederated International Conferences, Rome, Italy, September 10-14, 2012* (edited by R. Meersman, H. Panetto, T. S. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, and I. F. Cruz), vol. 7565 of *Lecture Notes in Computer Science*, pp. 194–211. Springer (2012). doi:10.1007/978-3-642-33606-5_13.
- [27] DE MASELLIS, R., DI CICCIO, C., MECELLA, M., AND PATRIZI, F. Smart home planning programs. In *7th International Conference on Service Systems and Service Management, ICSSSM 2010, Tokyo, Japan, June 28-30, 2010* (edited by J. Chen), pp. 377–382. IEEE (2010). doi:10.1109/ICSSSM.2010.5530212.
- [28] DE MEDEIROS, A. K. A., WEIJTERS, A. J. M. M., AND VAN DER AALST, W. M. P. Genetic process mining: an experimental evaluation. *Data Min.*

- Knowl. Discov.*, **14** (2007), 245. Available from: <http://dx.doi.org/10.1007/s10618-006-0061-7>, doi:10.1007/s10618-006-0061-7.
- [29] DECKER, G., DIJKMAN, R. M., DUMAS, M., AND GARCÍA-BAÑUELOS, L. The business process modeling notation. In ter Hofstede et al. [86], pp. 347–368. Available from: <http://www.springer.com/computer+science/database+management+%26+information+retrieval/book/978-3-642-03120-5>, doi:10.1007/978-3-642-03121-2_13.
- [30] DESEL, J. AND REISIG, W. The synthesis problem of petri nets. *Acta Informatica*, **33** (1996), 297. 10.1007/s002360050046. Available from: <http://dx.doi.org/10.1007/s002360050046>.
- [31] DI CICCIO, C., CATARCI, T., AND MECELLA, M. Representing and visualizing mined artful processes in MailOfMine. In *Information Quality in e-Health - 7th Conference of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2011, Graz, Austria, November 25-26, 2011* (edited by A. Holzinger and K.-M. Simonic), vol. 7058 of *Lecture Notes in Computer Science*, pp. 83–94. Springer (2011). ISBN 978-3-642-25363-8. doi:10.1007/978-3-642-25364-5_9.
- [32] DI CICCIO, C., MARRELLA, A., AND RUSSO, A. Knowledge-intensive processes: An overview of contemporary approaches. In *1st International Workshop on Knowledge-intensive Business Processes, KiBP 2012, Rome, Italy, June 15, 2012* (edited by A. H. ter Hofstede, M. Mecella, S. Sardina, and A. Marrella), vol. 861, pp. 33–47. CEUR Workshop Proceedings (2012). Available from: http://ceur-ws.org/Vol-861/KiBP2012_paper_2.pdf.
- [33] DI CICCIO, C. AND MECELLA, M. MINERful, a mining algorithm for declarative process constraints in MailOfMine. Tech. rep., Dipartimento di Ingegneria Informatica, Automatica e Gestionale “Antonio Ruberti” – SAPIENZA, Università di Roma (2012). Available from: http://ojs.uniroma1.it/index.php/DIS_TechnicalReports/issue/view/416.
- [34] DI CICCIO, C. AND MECELLA, M. Mining constraints for artful processes. In *15th International Conference on Business Information Systems, BIS 2012, Vilnius, Lithuania, May 21-23, 2012* (edited by W. Abramowicz, D. Kriksciuniene, and V. Sakalauskas), vol. 117 of *Lecture Notes in Business Information Processing*, pp. 11–23. Springer (2012). ISBN 978-3-642-30358-6. doi:10.1007/978-3-642-30359-3_2.
- [35] DI CICCIO, C. AND MECELLA, M. Studies on the discovery of declarative control flows from error-prone data. In *3rd International Symposium on Data-Driven Process Discovery and Analysis, SIMPDA 2013, Riva del Garda, Italy, August 30, 2013* (edited by R. Accorsi, P. Ceravolo, and P. Cudre-Mauroux), vol. 1027 of *CEUR Workshop Proceedings*, pp. 31–45 (2013). Available from: <http://ceur-ws.org/Vol-1027/paper3.pdf>.
- [36] DI CICCIO, C. AND MECELLA, M. A two-step fast algorithm for the automated discovery of declarative workflows. In *4th IEEE Symposium on Computational*

- Intelligence and Data Mining, CIDM 2013, Singapore, April 16-19, 2013*, pp. 135–142. IEEE (2013). doi:10.1109/CIDM.2013.6597228.
- [37] DI CICCIO, C., MECELLA, M., CARUSO, M., FORTE, V., IACOMUSSI, E., RASCH, K., QUERZONI, L., SANTUCCI, G., AND TINO, G. The homes of tomorrow: Service composition and advanced user interfaces. *ICST Transactions on Ambient Systems*, **11** (2011), e2. doi:10.4108/trans.amsys.2011.e2.
- [38] DI CICCIO, C., MECELLA, M., SCANNAPIECO, M., AND ZARDETTO, D. Groupware mail messages analysis for mining collaborative processes. In *19th Italian Symposium on Advanced Database Systems, SEBD 2011, Maratea, Italy, June 26-29, 2011* (edited by G. Mecca and S. Greco), pp. 397–404 (2011).
- [39] DI CICCIO, C., MECELLA, M., SCANNAPIECO, M., ZARDETTO, D., AND CATARCI, T. Groupware mail messages analysis for mining collaborative processes. Tech. rep., Dipartimento di Informatica e Sistemistica ANTONIO RUBERTI – SAPIENZA – Università di Roma (2011). Available from: http://ojs.uniroma1.it/index.php/DIS_TechnicalReports/article/view/8966.
- [40] DI CICCIO, C., MECELLA, M., SCANNAPIECO, M., ZARDETTO, D., AND CATARCI, T. MailOfMine – analyzing mail messages for mining artful collaborative processes. In *1st International Symposium on Data-Driven Process Discovery and Analysis, SIMPDA 2011, Campione d’Italia, Italy, June 29 - July 1st, 2011* (edited by K. Aberer, E. Damiani, and T. Dillon), pp. 45–59 (2011). ISBN 978-88-903120-2-1.
- [41] DI CICCIO, C., MECELLA, M., SCANNAPIECO, M., ZARDETTO, D., AND CATARCI, T. MailOfMine – analyzing mail messages for mining artful collaborative processes. In *Data-Driven Process Discovery and Analysis* (edited by K. Aberer, E. Damiani, and T. Dillon), vol. 116 of *Lecture Notes in Business Information Processing*, pp. 55–81. Springer (2012). ISBN 978-3-642-34043-7. doi:10.1007/978-3-642-34044-4_4.
- [42] DREDZE, M., LAU, T. A., AND KUSHMERICK, N. Automatically classifying emails into activities. In *IUI* (edited by C. Paris and C. L. Sidner), pp. 70–77. ACM (2006). ISBN 1-59593-287-9. Available from: <http://doi.acm.org/10.1145/1111449.1111471>, doi:10.1145/1111449.1111471.
- [43] DUMAS, M., LA ROSA, M., MENDLING, J., AND REIJERS, H. A. *Fundamentals of Business Process Management*. Springer (2013). ISBN 978-3-642-33143-5.
- [44] FOUNDATION, T. A. S. Apache lucene – scoring. On-line. Available from: http://lucene.apache.org/core/3_6_2/scoring.html.
- [45] GIANNAKOPOULOU, D. AND HAVELUND, K. Automata-based verification of temporal properties on running programs. In *ASE*, pp. 412–416. IEEE Computer Society (2001). ISBN 0-7695-1426-X. Available from: <http://doi.ieeecomputersociety.org/10.1109/ASE.2001.989841>.

- [46] GRONAU, N. AND WEBER, E. Management of knowledge intensive business processes. In *Business Process Management* (edited by J. Desel, B. Pernici, and M. Weske), vol. 3080 of *Lecture Notes in Computer Science*, pp. 163–178. Springer (2004). ISBN 3-540-22235-9. doi:10.1007/978-3-540-25970-1_11.
- [47] GÜNTHER, C. W. AND VERBEEK, E. Xes standard definition (2012). Available from: http://www.xes-standard.org/_media/xes/xesstandarddefinition-1.4.pdf.
- [48] HAVELUND, K. AND ROSU, G. Testing linear temporal logic formulae on finite execution traces. Tech. rep., Research Institute for Advanced Computer Science (RIACS) (2001).
- [49] HEUTELBECK, D. Preservation of enterprise engineering processes by social collaboration software (2011). Personal communication.
- [50] HILL, C., YATES, R., JONES, C., AND KOGAN, S. L. Beyond predictable workflows: Enhancing productivity in artful business processes. *IBM Systems Journal*, **45** (2006), 663. Available from: <http://dx.doi.org/10.1147/sj.454.0663>.
- [51] INNOCENTI, P., ROSS, S., MACECIUVITE, E., WILSON, T., LUDWIG, J., AND PEMPE, W. Assessing digital preservation frameworks: the approach of the SHAMAN project. In *MEDES* (edited by R. Chbeir, Y. Badr, E. Kapetanios, and A. J. M. Traina), pp. 412–416. ACM (2009). ISBN 978-1-60558-829-2. Available from: <http://doi.acm.org/10.1145/1643823.1643899>.
- [52] KEMSLEY, S. The changing nature of work: From structured to unstructured, from controlled to social. In Rinderle-Ma et al. [78], p. 2. Available from: http://dx.doi.org/10.1007/978-3-642-23059-2_2, doi:10.1007/978-3-642-23059-2_2.
- [53] KUSHMERICK, N., LAU, T. A., DREDZE, M., AND KHOUSSAINOV, R. Activity-centric email: A machine learning approach. In *AAAI*, pp. 1634–1637. AAAI Press (2006). Available from: <http://www.aaai.org/Library/AAAI/2006/aaai06-268.php>.
- [54] LAMMA, E., MELLO, P., MONTALI, M., RIGUZZI, F., AND STORARI, S. Inducing declarative logic-based models from labeled traces. In Alonso et al. [5], pp. 344–359. Available from: http://dx.doi.org/10.1007/978-3-540-75183-0_25, doi:10.1007/978-3-540-75183-0_25.
- [55] LAMMA, E., MELLO, P., RIGUZZI, F., AND STORARI, S. Applying inductive logic programming to process mining. In *ILP* (edited by H. Blockeel, J. Ramon, J. W. Shavlik, and P. Tadepalli), vol. 4894 of *Lecture Notes in Computer Science*, pp. 132–146. Springer (2007). ISBN 978-3-540-78468-5. Available from: http://dx.doi.org/10.1007/978-3-540-78469-2_16, doi:10.1007/978-3-540-78469-2_16.
- [56] MAGGI, F. M., BOSE, R. P. J. C., AND VAN DER AALST, W. M. P. Efficient discovery of understandable declarative process models from event

- logs. In *CAiSE* (edited by J. Ralyté, X. Franch, S. Brinkkemper, and S. Wrycza), vol. 7328 of *Lecture Notes in Computer Science*, pp. 270–285. Springer (2012). ISBN 978-3-642-31094-2. Available from: http://dx.doi.org/10.1007/978-3-642-31095-9_18.
- [57] MAGGI, F. M., MOOIJ, A. J., AND VAN DER AALST, W. M. P. User-guided discovery of declarative process models. In *CIDM*, pp. 192–199. IEEE (2011). ISBN 978-1-4244-9925-0. Available from: <http://dx.doi.org/10.1109/CIDM.2011.5949297>.
- [58] MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. *Introduction to information retrieval*. Cambridge University Press (2008). ISBN 978-0-521-86571-5.
- [59] MEERSMAN, R. AND TARI, Z. (eds.). *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I*, vol. 4803 of *Lecture Notes in Computer Science*. Springer (2007). ISBN 978-3-540-76846-3.
- [60] MEERSMAN, R., ET AL. (eds.). *20th International Conference on Cooperative Information Systems, CoopIS 2012, Rome, Italy, September 10-14, 2012*, vol. 7565 of *Lecture Notes in Computer Science*. Springer (2012). On the Move to Meaningful Internet Systems (OTM 2012) Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012.
- [61] MENDLING, J., NEUMANN, G., AND VAN DER AALST, W. M. P. Understanding the occurrence of errors in process models based on metrics. In Meersman and Tari [59], pp. 113–130. Available from: http://dx.doi.org/10.1007/978-3-540-76848-7_9, doi:10.1007/978-3-540-76848-7_9.
- [62] MENDLING, J., REIJERS, H. A., AND CARDOSO, J. What makes process models understandable? In Alonso et al. [5], pp. 48–63. Available from: http://dx.doi.org/10.1007/978-3-540-75183-0_4, doi:10.1007/978-3-540-75183-0_4.
- [63] MENDLING, J., REIJERS, H. A., AND VAN DER AALST, W. M. P. Seven process modeling guidelines (7PMG). *Information & Software Technology*, **52** (2010), 127. Available from: <http://dx.doi.org/10.1016/j.infsof.2009.08.004>, doi:10.1016/j.infsof.2009.08.004.
- [64] MITCHELL, T. M. *Machine Learning*. McGraw Hill series in computer science. McGraw-Hill, Inc., New York, NY, USA, 1 edn. (1997). ISBN 0070428077, 9780070428072.
- [65] MØLLER, A. dk.bricks.automaton (2011). Available from: <http://www.brics.dk/automaton/index.html>.
- [66] MONTALI, M. *Specification and Verification of Declarative Open Interaction Models: a Logic-Based Approach*, vol. 56 of *Lecture Notes in Business Information Processing*. Springer (2010). ISBN 978-3-642-14537-7. Available from: <http://www.springer.com/computer/database+management+%26+>

- [information+retrieval/book/978-3-642-14537-7?cm_mmc=sgw_-_ps_-_book_-_978-3-642-14537-7](http://dx.doi.org/10.1007/978-3-642-14537-7?cm_mmc=sgw_-_ps_-_book_-_978-3-642-14537-7), doi:10.1007/978-3-642-14538-4.
- [67] MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, **77** (1989), 541 . doi:10.1109/5.24143.
- [68] MUTSCHKE, P., MAYR, P., SCHAEER, P., AND SURE, Y. Science models as value-added services for scholarly information systems. *Scientometrics*, **89** (2011), 349. Available from: <http://dx.doi.org/10.1007/s11192-011-0430-x>, doi:10.1007/s11192-011-0430-x.
- [69] MYERS, E. W. An O(ND) difference algorithm and its variations. *Algorithmica*, **1** (1986), 251. Available from: <http://dx.doi.org/10.1007/BF01840446>.
- [70] OUNIS, I., AMATI, G., PLACHOURAS, V., HE, B., MACDONALD, C., AND LIOMA, C. Terrier: A high performance and scalable information retrieval platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)* (2006).
- [71] PESIC, M. *Constraint-based Workflow Management Systems: Shifting Control to Users*. Ph.D. thesis, Technische Universiteit Eindhoven (2008). Available from: <http://repository.tue.nl/638413>.
- [72] PESIC, M., BOSNACKI, D., AND VAN DER AALST, W. M. P. Enacting declarative languages using ltl: Avoiding errors and improving performance. In *SPIN* (edited by J. van de Pol and M. Weber), vol. 6349 of *Lecture Notes in Computer Science*, pp. 146–161. Springer (2010). ISBN 978-3-642-16163-6. Available from: http://dx.doi.org/10.1007/978-3-642-16164-3_11, doi:10.1007/978-3-642-16164-3_11.
- [73] PESIC, M., SCHONENBERG, H., AND VAN DER AALST, W. M. P. Declare: Full support for loosely-structured processes. In *EDOC*, pp. 287–300. IEEE Computer Society (2007). Available from: <http://doi.ieeecomputersociety.org/10.1109/EDOC.2007.25>.
- [74] PESIC, M., SCHONENBERG, M. H., SIDOROVA, N., AND VAN DER AALST, W. M. P. Constraint-based workflow models: Change made easy. In Meersman and Tari [59], pp. 77–94. Available from: http://dx.doi.org/10.1007/978-3-540-76848-7_7.
- [75] PESIC, M. AND VAN DER AALST, W. M. P. A declarative approach for flexible business processes management. In *Business Process Management Workshops* (edited by J. Eder and S. Dustdar), vol. 4103 of *Lecture Notes in Computer Science*, pp. 169–180. Springer (2006). ISBN 3-540-38444-8. Available from: http://dx.doi.org/10.1007/11837862_18.
- [76] PETRI, C. A. *Kommunikation mit Automaten*. Ph.D. thesis, Institut für instrumentelle Mathematik, Bonn (1962).
- [77] RICHARDSON, M. AND DOMINGOS, P. Markov logic networks. *Machine Learning*, **62** (2006), 107. Available from: <http://dx.doi.org/10.1007/s10994-006-5833-1>, doi:10.1007/s10994-006-5833-1.

- [78] RINDERLE-MA, S., TOUMANI, F., AND WOLF, K. (eds.). *Business Process Management - 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30 - September 2, 2011. Proceedings*, vol. 6896 of *Lecture Notes in Computer Science*. Springer (2011). ISBN 978-3-642-23058-5. Available from: <http://dx.doi.org/10.1007/978-3-642-23059-2>.
- [79] RUSSELL, N., TER HOFSTEDÉ, A. H. M., VAN DER AALST, W. M. P., AND MULYAR, N. Workflow control-flow patterns: a revised view. Tech. Rep. BPM-06-22, BPMcenter.org (2006).
- [80] SAKURAI, S., ICHIMURA, Y., SUYAMA, A., AND ORIHARA, R. Acquisition of a knowledge dictionary for a text mining system using an inductive learning method. In *Proceedings of IJCAI 2001 Workshop on Text Learning: Beyond Supervision*, pp. 45–52 (2001).
- [81] SAKURAI, S. AND SUYAMA, A. An e-mail analysis method based on text mining techniques. *Appl. Soft Comput.*, **6** (2005), 62. Available from: <http://dx.doi.org/10.1016/j.asoc.2004.10.007>.
- [82] SCHUNSELAAR, D. M. M., MAGGI, F. M., SIDOROVA, N., AND VAN DER AALST, W. M. P. Configurable Declare: Designing customisable flexible process models. In Meersman et al. [60], pp. 20–37. On the Move to Meaningful Internet Systems (OTM 2012) Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012. Available from: http://dx.doi.org/10.1007/978-3-642-33606-5_3, doi:10.1007/978-3-642-33606-5_3.
- [83] SEARLE, J. *A Taxonomy of Illocutionary Acts*, pp. 334–369. University of Minnesota Press, Minneapolis (1975).
- [84] SEBASTIANI, F. Machine learning in automated text categorization. *ACM Comput. Surv.*, **34** (2002), 1. Available from: <http://doi.acm.org/10.1145/505282.505283>.
- [85] SMART VORTEX CONSORTIUM. Smart Vortex – management and analysis of massive data streams to support large-scale collaborative engineering projects. FP7 IP Project (2010). Available from: <http://www.smartvortex.eu/>.
- [86] TER HOFSTEDÉ, A. H. M., VAN DER AALST, W. M. P., ADAMNS, M., AND RUSSELL, N. (eds.). *Modern Business Process Automation: YAWL and its Support Environment*. Springer (2010). ISBN 978-3-642-03120-5. Available from: <http://www.springer.com/computer+science/database+management+%26+information+retrieval/book/978-3-642-03120-5>.
- [87] VAN DER AALST, W. M. P., RUBIN, V., VERBEEK, E., VAN DONGEN, B. F., KINDLER, E., AND GÜNTHER, C. W. Process mining: a two-step approach to balance between underfitting and overfitting. *Software and Systems Modeling*, **9** (2010), 87. 10.1007/s10270-008-0106-z. Available from: <http://dx.doi.org/10.1007/s10270-008-0106-z>.
- [88] VAN DER AALST, W. M. P. Verification of workflow nets. In ICATPN (edited by P. Azéma and G. Balbo), vol. 1248 of *Lecture Notes in Computer*

- Science*, pp. 407–426. Springer (1997). ISBN 3-540-63139-9. Available from: http://dx.doi.org/10.1007/3-540-63139-9_48.
- [89] VAN DER AALST, W. M. P. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, **8** (1998), 21. Available from: <http://dx.doi.org/10.1142/S0218126698000043>, doi:10.1142/S0218126698000043.
- [90] VAN DER AALST, W. M. P. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer (2011). ISBN 978-3-642-19344-6. doi:10.1007/978-3-642-19345-3.
- [91] VAN DER AALST, W. M. P. AND NIKOLOV, A. Mining e-mail messages: Uncovering interaction patterns and processes using e-mail logs. *IJIT*, **4** (2008), 27. Available from: <http://www.igi-global.com/Bookstore/Article.aspx?TitleId=2437>.
- [92] VAN DER AALST, W. M. P. AND PESIC, M. DecSerFlow: Towards a truly declarative service flow language. In *WS-FM* (edited by M. Bravetti, M. Núñez, and G. Zavattaro), vol. 4184 of *Lecture Notes in Computer Science*, pp. 1–23. Springer (2006). ISBN 3-540-38862-1. Available from: http://dx.doi.org/10.1007/11841197_1, doi:10.1007/11841197_1.
- [93] VAN DER AALST, W. M. P., PESIC, M., AND SCHONENBERG, H. Declarative workflows: Balancing between flexibility and support. *Computer Science - R&D*, **23** (2009), 99. Available from: <http://dx.doi.org/10.1007/s00450-009-0057-9>, doi:10.1007/s00450-009-0057-9.
- [94] VAN DER AALST, W. M. P. AND TER HOFSTEDÉ, A. H. M. YAWL: yet another workflow language. *Inf. Syst.*, **30** (2005), 245. Available from: <http://dx.doi.org/10.1016/j.is.2004.02.002>, doi:10.1016/j.is.2004.02.002.
- [95] VAN DER AALST, W. M. P., TER HOFSTEDÉ, A. H. M., KIEPUSZEWSKI, B., AND BARROS, A. P. Workflow patterns. *Distributed and Parallel Databases*, **14** (2003), 5. Available from: <http://dx.doi.org/10.1023/A:1022883727209>.
- [96] VAN DER AALST, W. M. P., VAN DONGEN, B. F., GÜNTHER, C. W., ROZINAT, A., VERBEEK, E., AND WEIJTERS, T. ProM: The process mining toolkit. In *BPM (Demos)* (edited by A. K. A. de Medeiros and B. Weber), vol. 489 of *CEUR Workshop Proceedings*. CEUR-WS.org (2009). Available from: <http://ceur-ws.org/Vol-489/paper3.pdf>.
- [97] VAN DER AALST, W. M. P., WEIJTERS, T., AND MARUSTER, L. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, **16** (2004), 1128. Available from: <http://csdl.computer.org/comp/trans/tk/2004/09/k1143abs.htm>.
- [98] VAN DONGEN, B. F. Real-life event logs – a hospital log. First International Business Process Intelligence Challenge (BPIC’11) (2011). Available from: <http://dx.doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffc54>, doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffc54.

- [99] VAN DONGEN, B. F. Real-life event logs – a loan application process. Second International Business Process Intelligence Challenge (BPIC'12) (2012). Available from: <http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f>, doi:10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.
- [100] VERBEEK, H. M. W., BUIJS, J. C. A. M., VAN DONGEN, B. F., AND VAN DER AALST, W. M. P. XES, XESame, and ProM 6. In *CAiSE Forum* (edited by P. Soffer and E. Proper), vol. 72 of *Lecture Notes in Business Information Processing*, pp. 60–75. Springer (2010). ISBN 978-3-642-17721-7. Available from: http://dx.doi.org/10.1007/978-3-642-17722-4_5, doi:10.1007/978-3-642-17722-4_5.
- [101] W3C, <http://www.w3.org/TR/xslt.html>. *XSL Transformations (XSLT) Version 1.0 W3C Recommendation* (1999).
- [102] WARREN, P., ET AL. Improving knowledge worker productivity - the Active integrated approach. *BT Technology Journal*, **26** (2009), 165.
- [103] WEIJTERS, A. J. M. M. AND VAN DER AALST, W. M. P. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, **10** (2003), 151. Available from: <http://iospress.metapress.com/content/8puq22eumrva7vyp/>.
- [104] WEN, L., VAN DER AALST, W. M. P., WANG, J., AND SUN, J. Mining process models with non-free-choice constructs. *Data Min. Knowl. Discov.*, **15** (2007), 145. Available from: <http://dx.doi.org/10.1007/s10618-007-0065-y>.
- [105] WESTERGAARD, M. Better algorithms for analyzing and enacting declarative workflow languages using ltl. In Rinderle-Ma et al. [78], pp. 83–98. Available from: http://dx.doi.org/10.1007/978-3-642-23059-2_10.
- [106] WOHED, P., VAN DER AALST, W. M. P., DUMAS, M., TER HOFSTEDE, A. H. M., AND RUSSELL, N. On the suitability of BPMN for business process modelling. In *BPM* (edited by S. Dustdar, J. L. Fiadeiro, and A. P. Sheth), vol. 4102 of *Lecture Notes in Computer Science*, pp. 161–176. Springer (2006). ISBN 3-540-38901-6. Available from: http://dx.doi.org/10.1007/11841760_12, doi:10.1007/11841760_12.

List of Figures

2.1	A process behavior as a deterministic automaton	8
2.2	A process behavior as a non-deterministic automaton	9
2.3	An automaton-based representation of a process in a case study	9
2.4	A Petri Net	10
2.5	A possible evolution of the status of a Petri Net	12
2.6	A Workflow Net	13
2.7	A spaghetti process	13
2.8	Example of a Declare constraint model [75]	14
2.9	Existence, relation and negation constraint templates	16
2.10	A simple constraint model [86]	23
3.1	The MAILOFMINE approach	27
3.2	Screenshots of MAILOFMINE: the “compose” window	30
3.3	Screenshots of MAILOFMINE: the activity analyzer window	30
3.4	The database schema of MAILOFMINE	32
3.5	The database schema of MAILOFMINE, along with the components managing its stored values	33
3.6	The declarative process model’s hierarchy of constraints	39
3.7	The rationale of the local view design	41
3.8	The MAILOFMINE local static constraint diagrams	42
3.9	A activities subtrace constraints diagram	43
3.10	The MAILOFMINE global static constraints diagram	44
3.11	Prototype of the activity’s details panel	45
3.12	The MAILOFMINE dynamic process view	46
3.13	Prototype of the execution management window	46
5.1	Experimental results of MINERful: time w.r.t. number of traces	83
5.2	Experimental results of MINERful: time w.r.t. trace’s length	83
5.3	Experimental results of MINERful: time w.r.t. number of events read	84
5.4	Experimental results of MINERful: time w.r.t. size of the alphabet	84
5.5	Experimental results of MINERful: time taken by the algorithm for the construction of the MINERfulKB and the discovery of constraints by queries over the MINERfulKB, w.r.t. size of the alphabet	85
5.6	Experimental results of MINERful: time taken by the algorithm for the construction of the MINERfulKB, given the MINERfulKB, w.r.t. size of the alphabet	85
5.7	The trend of the support for <i>End</i> , w.r.t. the errors injected in the log	88

5.8	The trend of Support for <i>AlternateResponse</i> and <i>AlternatePrecedence</i> , w.r.t. the errors injected in the log	89
5.9	The trend of Support for <i>AlternateSuccession</i> , w.r.t. the errors injected in the log	90
5.10	The trend of the support for <i>NegatedRelation</i> constraints, w.r.t. the errors injected in the log	91
5.11	The trend of Support for the <i>MutualRelation</i> constraints, w.r.t. the errors injected in the log	92
5.12	Evaluation of MAILOFMINE on a case study: appropriateness of the discovered constraints in the case study	95
5.13	Evaluation of MAILOFMINE on a case study: errors w.r.t. constraint templates	96
5.14	Evaluation of MAILOFMINE on a case study: errors w.r.t. implying activities	97
5.15	Evaluation of MAILOFMINE on a case study: trend of the quality of the process w.r.t. the Support of constraints	98
5.16	Evaluation of MAILOFMINE on a case study: trend of the quality of the process w.r.t. the Confidence Level of constraints	99
5.17	Evaluation of MAILOFMINE on a case study: trend of the quality of the process w.r.t. the Interest Factor of constraints	100
5.18	Evaluation of MAILOFMINE on a case study: trend of Precision	102
5.19	Evaluation of MAILOFMINE on a case study: global and local views on the discovered process, depicted as Finite State Automata	104
B.1	The local automaton for the “organize agenda” activity	170
B.2	The local automaton for the “organize demo” activity	170
B.3	The local automaton for the “organize meeting” activity	171
B.4	The local automaton for the “send agenda” activity	172
B.5	The local automaton for the “send deliverable” activity	172
B.6	The local automaton for the “send demo” activity	173
B.7	The local automaton for the “send draft” activity	173
B.8	The local automaton for the “send meeting” activity	174
B.9	The local automaton for the “send report” activity	175
B.10	The local automaton for the “submit deliverable” activity	175
B.11	The local automaton for the “submit draft” activity	176
B.12	The local automaton for the “submit report” activity	176
B.13	The local automaton for the “write deliverable” activity	177
B.14	The global automaton for the discovered process	178

List of Tables

3.1	Semantics of Declare constraints	35
3.2	Semantics of Declare constraints as regular expressions	38
4.1	Abbreviations for the functions of MINERfulKB	51
4.2	An example of MINERful interplay, interpreted over <code>aabbac</code>	52
4.3	The evolution of N and $\delta_{\cdot, (+\infty)}$, over the reading of a string $t = \text{aabbac}$	57
4.4	The evolution of \widehat{W} , \bar{W} , and $\beta_{\cdot, \cdot}^{\rightarrow}$ over the reading of a string $t = \text{aabbac}$	58
4.5	Symbols expressing the validity of constraints	64
4.6	Functions computing the Support of constraints	69
4.7	The functions navigating the constraints' hierarchy of subsumptions	75
5.1	Experiments' setup	80
5.2	Performances of MINERful over synthetic and real cases	81
5.3	Setup of the experiments for monitoring the reaction of MINERful to the controlled error injection into logs	87
5.4	Evaluation of MAILOFMINE on a case study: preliminary setups and gathered data	93
5.5	Evaluation of MAILOFMINE on a case study: the extended vocabulary	94
5.6	Evaluation of MAILOFMINE on a case study: the restricted vocabulary	94