



Althubiti, Ebtihal Abdulmaeen (2025) *Formalising privacy regulations with bigraphs*. PhD thesis.

<https://theses.gla.ac.uk/85505/>

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses

<https://theses.gla.ac.uk/>

research-enlighten@glasgow.ac.uk

Formalising Privacy Regulations with Bigraphs

Ebtihal Abdulmaeen Althubiti

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Computing Science
College of Science and Engineering
University of Glasgow



University
of Glasgow

June 2025

Abstract

With many governments regulating the handling of user data—the General Data Protection Regulation (GDPR), the California Consumer Privacy Act (CCPA), and the Saudi Arabian Personal Data Protection Law (PDPL)—ensuring system compliance with data privacy legislation is critically important. Organisations’ failure to comply with these regulations can result in severe financial penalties. Checking compliance is a complex process, and often includes manual procedures. We propose that formal methods, that model systems mathematically, can provide strong guarantees to help organisations *prove* their adherence to legislation.

This thesis introduces a formal privacy framework based on Milner’s Bigraphical Reactive Systems (BRSs), a universal formalism that captures both spatial (placement) and non-spatial (connectivity) relationships between entities. BRSs evolve over time via user-specified rewriting rules that can be defined algebraically and diagrammatically. The rewrite rules describe system behaviour and allow flexibility in integrating privacy policies with user-defined systems. To increase usability, we advocate a diagrammatic approach, where privacy experts can explicitly *visualise* the systems and describe updates. We focus on modelling the following privacy concepts: *cross-border data transfer constraints*, *providing consent*, *withdrawing consent*, *purpose limitations*, *the right to access and sharing data with third-parties*.

For formal verification, the framework combines static analysis (via inductive reasoning and sorting) with automated model checking, which uses predefined properties expressed in Computation Tree Logic (CTL). To support practical adoption, user interfaces (UIs) are developed to automatically generate well-formed initial states (the starting configurations of a system model) for the framework. The effectiveness and generality of the framework are demonstrated through real-world systems deployed by WhatsApp, Monzo Bank, and Fitbit.

Contents

Abstract	i
Acknowledgements	vii
1 Introduction	1
1.1 Motivation	4
1.2 Research Questions	5
1.3 Thesis Statement	5
1.4 Thesis Contributions	6
1.5 Thesis Publications	7
1.6 Thesis Structure	7
2 Privacy Regulations	9
2.1 Privacy Regulations	10
2.1.1 Common Principles	10
2.1.2 Distinct Principles	11
2.2 Solove Taxonomy	12
2.3 Mapping Solove’s Taxonomy to Privacy Regulations Requirements	14
2.4 Summary	15
3 Informal and Formal Privacy Methods	17
3.1 Informal Methods	18
3.1.1 Non-Regulatory Methods for Privacy Protection	18
3.1.2 Regulatory Methods for Compliance with Privacy Regulations	21
3.1.3 Limitations	24

3.2	Formal Methods	24
3.2.1	Non-Regulatory Methods for Privacy Protection	25
3.2.2	Regulatory Methods for Compliance with Privacy Regulations	31
3.2.3	Limitations	35
3.3	Summary	36
4	Bigraph Reactive Systems	38
4.1	Bigraphs	38
4.2	Constituents of bigraphs	39
4.2.1	Place Graph	39
4.2.2	Link Graph	41
4.3	Bigraph Reactive Systems (BRSs)	43
4.4	Concrete Example: Modelling a System with BRSs	44
4.5	Key Features of BRSs	45
4.5.1	Multi-perspective Modelling	46
4.5.2	Spatial Properties	46
4.5.3	Sorting Scheme	46
4.5.4	Priority Classes	48
4.5.5	Paramaterised Entities and Rules	49
4.5.6	Instantiation Maps	49
4.5.7	Instantaneous rules	49
4.5.8	Conditions	51
4.5.9	Bigraph Patterns	51
4.6	Bigraph Evaluator and Rewriting	52
4.7	Summary	55
5	Privacy Compliance in Cross-border Data Transfers	56
5.1	Modelling Privacy Visually	57
5.1.1	Specifying Sender and Receiver Types	58
5.1.2	Specifying Sender and Receiver Locations	58
5.2	Checking privacy requirements	59
5.2.1	Checking Regions	59
5.2.2	Entities are in the Same Region	61

5.2.3	Checking Adequacy	61
5.2.4	Checking SCCs	62
5.2.5	Checking Certification	62
5.2.6	Result of Checking Certification	63
5.2.7	Checking the Expiry Date	64
5.2.8	Withdrawing Certification	64
5.3	Integration of Privacy Models and Specific System	65
5.4	Discussion	72
5.5	Summary	73
6	Privacy Regulations Model	74
6.1	A Visual Approach to Privacy Modelling	75
6.1.1	Modelling Data Governance Entity (DGE)	76
6.1.2	Specifying Agent And Data Types	77
6.1.3	Specifying Agent Locations	79
6.1.4	Consent Perspective	79
6.2	Example: Agent, Data Types And Locations for the Banking System	80
6.3	Privacy Dynamics	83
6.3.1	Providing Consent	83
6.3.2	Permission Checking	88
6.3.3	Right to Access	89
6.3.4	Sharing with Third-Parties	89
6.3.5	Withdrawing Consent	90
6.4	Integrating Privacy and System-Specific Behaviour	92
6.4.1	Sharing Data Based on Consent	94
6.4.2	Transferring Data based on GDPR Requirements for International Transfer	95
6.4.3	Withdrwing Consent	98
6.5	Reusability: Modelling a Home Healthcare System	100
6.6	Discussion	103
6.7	Summary	105

7	Verification	106
7.1	Static Analysis	106
7.1.1	Sorting	107
7.1.2	Inductive Reasoning	119
7.2	Model Checking	122
7.2.1	Cross-border Data Transfers Model	123
7.2.2	Privacy Regulation Model	124
7.3	Detecting Privacy Violations	128
7.4	Discussion	130
7.5	Summary	131
8	Enhancing Usability	132
8.1	BIDTRA Interface	132
8.1.1	System and Privacy File Integration	133
8.1.2	Building the Initial State	134
8.1.3	Run BigraphER	138
8.2	BPRA Interface	139
8.2.1	System and Privacy File Integration	139
8.2.2	Adding New Permissions and Purposes	139
8.2.3	Building the Initial State	145
8.2.4	Run BigraphER	146
8.3	Discussion	148
8.4	Summary	150
9	Discussion and Conclusion	151
9.1	Summary	151
9.2	Research Questions Revisited	151
9.3	Review of Thesis Contributions	155
9.4	Discussion	156
9.5	Future Work	158
A	Additional rules for the banking example	160
B	Additional system rules for the healthcare example	163

C	Algebraic definition of the privacy rules	165
D	User Interfaces	167
D.1	BIDTRA Interface	167
D.2	BPRA Interface	168
D.2.1	Banking Example	168
D.2.2	Healthcare Example	168

Acknowledgements

To my supervisor, Michele Sevegnani — thank you for your patience, guidance, and constant encouragement. You were not only a supervisor, but a trusted supporter who made this journey feel less daunting. Your impact on my academic and personal growth will stay with me always.

To Blair Archibald — thank you for your thoughtful feedback and support throughout this work.

To Northern Border University — thank you for the scholarship and continued support that made this journey possible.

To my father, Abdulmaeen — my hero and source of strength. Your sacrifices, prayers, and words of love carried me through. Your constant questions about when I would finish my PhD and return to Saudi reflected how much you missed me. That longing became my greatest motivation to keep going, so I could return to you with this achievement.

To my mother, Hamdah — though you are no longer here, your legacy continues to shape who I am becoming. I wish you could see this moment — you are in every part of it.

To my soulmates, Numa and Shatha — you have both been sources of love and strength. Numa, your care filled the space my mother left behind, and your belief in me carried me through. Shatha, your laughter and friendship lit up my darkest moments. I am deeply grateful to you both.

To the brothers who are my heart, Saeed and Hatim — you gave me a deep sense of safety, pride, and belonging. In every challenge, knowing you had my back gave me the strength to keep going. You reminded me, without saying a word, that I was never alone.

To my nephews and nieces — your joy and laughter reminded me of life's simple beauty. You gave me hope and energy just when I needed it most.

To my grandmother — your memory, love, and prayers still guide me. I carry your memory with pride and love.

*To the silent hope behind every
consent box and privacy setting —
this is my effort to give you meaning.*

Chapter 1

Introduction

Enhancements in sectors including education, governance, healthcare and finance [115], depend on sensing and data aggregation techniques to collect information about users. The amount and types of personal data collected make privacy a significant concern [29], and even collecting only non-personal information presents privacy risks through the ability to predict sensitive information about individuals [84].

To alleviate privacy concerns, governments have imposed regulations that protect users' private information and clarify their rights. A large number of regulations exist, including the Australian Privacy Principles (APPs) [106], the European Union (EU) General Data Protection Regulation (GDPR) [75], the California Consumer Privacy Act (CCPA) [126], the Saudi Arabian Personal Data Protection Law (PDPL) [22], the Georgia Computer Data Privacy Act (GCDPA) [45], and the American Data Privacy and Protection Act (ADPPA) [32].

Failing to adhere to these regulations can result in fines for organisations. Under the GDPR, organisations may face fines of up to 4% of their global annual income or 20 million euro, whichever is higher [74]. For example, in 2025, TikTok was fined 485 million euro by the Irish Data Protection Commission for failing to comply with the GDPR requirements by transferring the personal data of European Economic Area (EEA) users to China [76].

Given the range of regulations, how do organisations ensure information systems are robustly designed to avoid privacy violations? This issue is further compounded by the fact the regulations are non-formalised, *e.g.* textual, subject to change, and written by and for lawyers rather than system designers [52].

Formal methods play a key role in enabling companies to prove their compliance with

privacy legislation [128]. These methods are techniques used to model, analyse and verify systems' specifications mathematically, and allow guarantees/proofs of, for example, correctness, safety, and security [135]. They allow a shift away from text-based regulations and into a form amenable to exhaustive verification via automatic tooling.

Building on the role of formal methods in verifying privacy regulation compliance, we propose a novel approach to privacy modelling based on Milner's Bigraphical Reactive Systems (BRSs) [96]. BRSs are a computational model that is visual in nature, and that specifies systems based on the spatial and non-spatial relationships between entities. BRSs can evolve over time via user-specified rewriting rules that can be defined algebraically and diagrammatically. To increase usability we advocate a diagrammatic approach, where privacy experts can explicitly *visualise* the systems and describe updates, via rewrite rules, that describe system behaviour.

Figure 1.1 summarises the proposed framework. The framework consists of pre-defined (but modifiable) privacy entities and reaction rules. Users define system entities, using bigraphs, tailored to their domain (*e.g.* a banking system). The privacy entities can be customised by adding or removing entities as needed. Once linked to the system entities, users define the system's reaction rules and integrate them with the privacy reaction rules to model privacy regulations. Users can further customise the privacy reaction rules by selecting and removing unnecessary reaction rules. The combined model, consisting of both system and privacy entities along with their reaction rules, is executable using the BigraphER tool [117]. The framework also provides an extendable set of privacy properties that should be verified *e.g.* providing consent. These privacy properties are expressed using the Computation Tree Logic (CTL) and can be *automatically* verified using the PRISM model checker [85]. Although these properties are predefined, end-users must specify system-related aspects, *e.g.* sharing data with a third-party system entity.

To enhance the framework's usability, we develop user interfaces (UIs) that automatically link system entities with privacy entities. End-users subsequently integrate the linked entities with the privacy and system reaction rules to obtain the combined model, after which they follow the workflow in Figure 1.1: run BigraphER to derive the labelled transition system, verify the selected CTL properties in PRISM, and—if any property fails—revise the model and iterate.

The framework's primary users are developers specialising in bigraphs. It is applied during the design phase of the software development life cycle, after the system's goals

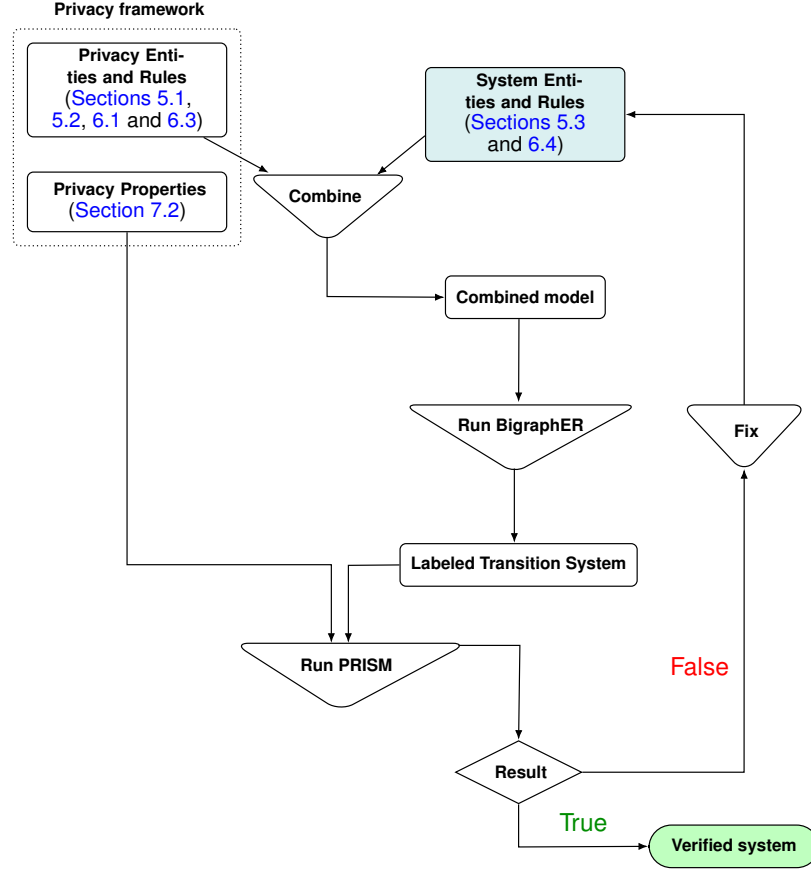


Figure 1.1: Bigraph-based privacy framework: The teal box represents the user-specified elements. These elements are combined with the privacy rules and entities to form a unified model. BigraphER then analyses this unified model to generate a labelled transition system. The model is subsequently verified using the PRISM model checker to assess its compliance with the specified privacy properties. If the verification result is positive, the system is considered to meet the privacy properties. If the result is negative, the system must be revised to address the identified issues, after which it undergoes reanalysis. This iterative process continues until the system successfully passes verification.

and privacy requirements are identified. At this stage, developers may use standard design methods such as UML and translate them into bigraphs. For example, class diagrams can be mapped to bigraph entities and initial states, while sequence diagrams can inform the reaction rules. The resulting bigraphical model allows developers to formally verify the compliance of the system's structure and behaviour with privacy regulations. The diagrammatic nature of bigraphs also makes collaboration with privacy experts practical, enabling them to review formal-analysis outcomes and provide legal or regulatory interpretation without engaging in the technical modelling.

We believe the approach is applicable to a variety of privacy-related scenarios. It is particularly applicable to those regulations with notions of *providing consent*, *withdrawing consent*, *purpose limitations*, *the right to access*, and *sharing data with third parties*. These notions are key principles required by privacy regulations to give users control over their data [114].

1.1 Motivation

Although several formal approaches are proposed to check compliance with privacy regulations [79], [131], [133], the majority only model the GDPR and a limited subset, *i.e.* providing/withdrawing consent or defining purposes. Regulations restricting cross-border data transfer are seldom covered, as these methods lack support for spatial properties. They also often require significant updates when privacy policies change, as they are typically defined in terms of a fixed set of rules.

Given these limitations, and the fact that organisations are legally required to ensure their systems adhere to privacy regulations, this thesis introduces a privacy framework based on bigraphs. Bigraphs can evolve over time through rewriting using reaction rules, allowing privacy updates to be modelled *e.g.* to describe consent being given or withdrawn, and data movement.

Bigraphs have several benefits: (1) they are flexible, as entities and reaction rules are user-specified, *e.g.* the same formalism can capture financial and healthcare domains; (2) reaction rules enable us to extend and amend models easily based on the changes that may occur in the privacy regulations or the underlying system; (3) they have a diagrammatic notation, not unlike you might draw on a whiteboard, that is suitable for system designers to understand and describe the model; (4) they natively express spatial properties, *e.g.* containment relation,

that enables us to model the GDPR requirements for cross-border data transfer; (5) they allow multi-perspective modelling where privacy concerns can be modelled independently of a specific system but interact with it via explicit links; (6) bigraph sorts allow categorising entities and links by means of sorts, ensuring that bigraphs are well-formed and preventing invalid compositions. These properties motivate our development of a pre-defined, reusable bigraph framework coupled with automated verification to provide formal assurance of privacy compliance.

1.2 Research Questions

This thesis contributes to ongoing research efforts that aim to formally model and verify system compliance with privacy regulations. The thesis answers the following questions:

1. How do bigraphs contribute to the formal design and verification of privacy-compliant systems?
2. Can a reusable, domain-independent bigraph framework be defined where modellers only specify system-specific components while covering different privacy regulations (the GDPR and the CCPA).
3. Can the spatial properties of bigraphs be used to model privacy regulations based on cross-border data transfer constraints?
4. Which privacy regulation requirements cannot be directly modelled with bigraphs, and what extensions are needed to support them?
5. How does the sorting discipline help modellers detect and correct structural errors during manual model construction?

1.3 Thesis Statement

This thesis presents a bigraph-based framework for modelling and verifying compliance with privacy regulations such as the GDPR and the CCPA. The framework supports key privacy principles—including consent, withdrawal, purpose limitation, access rights, and third-party data sharing. By exploiting bigraphs’ spatial properties, it captures location-based

restrictions in cross-border data-transfer requirements. Static techniques (inductive reasoning and sorting) ensure model well-formedness, while model checking provides automated dynamic verification. Examples inspired by real systems deployed by WhatsApp, Monzo, and Fitbit demonstrate the framework’s applicability and generality.

1.4 Thesis Contributions

- **Novel bigraph-based framework for privacy.** The framework consists of two models. The first model leverages the inherent spatial modelling capabilities of bigraphs to explicitly capture cross-border data transfers and support rigorous spatial verification. The second model allows notions of providing/withdrawing consent, purposes limitation, the right to access and sharing data with third-parties, all of which are required by most privacy regulations including the GDPR and the CCPA. We also show how we merge these two models to represent regulations that restrict international data transfers, *e.g.* the GDPR, in cases where data is shared with third-parties across borders.
- **Assessment the effectiveness of our approach via three real-world examples.** We demonstrate the applicability of the first model by modelling WhatsApp’s privacy policies. The applicability of the second model is illustrated by applying it to two examples: a bank notification system based on the GDPR requirements inspired by Monzo Bank’s privacy policy [97] and a cloud-based home healthcare system example based on the CCPA requirements derived from the privacy policies of the Fitbit app [42]. Although we abstract away certain system-specific aspects, the framework is comprehensive enough to capture the privacy regulations and express the privacy violation properties discussed in Chapter 7. Our goal is to show how the privacy model is constructed and utilised, rather than how to build a good system model (which is largely independent of the privacy perspective; see Fig. 1.1).
- **Formal verification of privacy properties through static and dynamic analysis.** As we generalise the model, we conduct static analysis: sorting and inductive reasoning. Sorting plays a structural role by enforcing well-formedness constraints on the model to prevent invalid configurations that could lead to privacy violations during execution. Inductive reasoning is used to prove the correctness of the privacy rules independently of any specific system instance. We also identify common privacy properties and

show how the model, combined with model checking, enables them to be formally and automatically verified.

- **User Interfaces (UIs) for automatically generating correct-by-construction initial states.** We develop two UIs: *Bigraph International Data Transfer Regulation Assistant* (BIDTRA) and *Bigraph Privacy Regulations Assistant* (BPRA). The former is used to define initial states for the first model, *i.e.* modelling the GDPR requirements for cross-border data transfers. The latter generates initial states for the second model, which captures providing and withdrawing consent, purpose limitation, the right to access, and sharing data with third parties. These interfaces help mitigate the risk of manually defining incorrect initial states, as discussed in [Section 5.4](#) and [Section 6.6](#).

1.5 Thesis Publications

The contents of [Chapters 5, 6](#) and [7](#) are presented in the following peer-reviewed publications:

- Ebtihal Althubiti, Blair Archibald, and Michele Sevegnani, "*Formalising privacy regulations with bigraphs*". *Software and Systems Modeling*, 2025, (doi:10.1007/s10270-025-01293-2).
- Ebtihal Althubiti and Michele Sevegnani, "*Modelling Privacy Compliance in Cross-border Data Transfers with Bigraphs*". In: *15th International Workshop on Graph Computation Models (GCM 2024)*, (doi: 10.4204/EPTCS.417.2).

1.6 Thesis Structure

This thesis is organised into 9 chapters:

- [Chapter 2](#) discusses the privacy principles that regulations such as the APPs, GDPR, PDPL, GCDPA, ADPPA, and CCPA share in common, as well as those that differ.
- [Chapter 3](#) presents related work by grouping existing approaches into two main categories: informal methods and formal methods. The informal methods are further divided into two types: those focused on general privacy protection without emphasis

on regulatory compliance, and those aimed at demonstrating adherence to legal frameworks. Likewise, formal methods are categorised into non-regulatory and regulatory approaches.

- [Chapter 4](#) introduces the theory of bigraphs and bigraphical reactive systems. The chapter also explains preliminary concepts that serve as a basis for the later chapters.
- [Chapter 5](#) presents the first model, which captures the GDPR requirements for cross-border data transfers. The chapter explains how we model system privacy using a multi-perspective approach. It also shows how the dynamic nature of privacy is captured through reaction rules, *e.g.* performing transfer requirements checking. We further demonstrate how the privacy framework is integrated with a specific system using an example based on WhatsApp’s privacy policies.
- [Chapter 6](#) introduces the second model, capturing privacy principles such as consent, withdrawal, purpose limitation, access rights, and third-party data sharing. To show the model’s reusability, it is applied to two examples: one based on Monzo’s privacy policy under the GDPR, and another based on Fitbit’s privacy policy under the CCPA. The chapter also shows how the first model is reused to capture the GDPR’s cross-border data transfer requirements in the Monzo example.
- [Chapter 7](#) details the verification techniques used to validate the models, including static analysis and model checking.
- [Chapter 8](#) describes the development of two interfaces that assist users in defining well-formed initial states for the models.
- [Chapter 9](#) highlights the features and the limitations of the framework, and concludes the thesis with suggestions for future work.

Chapter 2

Privacy Regulations

As mentioned previously in [Chapter 1](#), governments around the world have introduced various regulations aimed at protecting individuals' privacy, including the APPs, GDPR, CCPA, PDPL, GCDPA and ADPPA. However, these regulations differ due to the absence of a universally agreed-upon definition of privacy, which is shaped by cultural values and governments' rules [[130](#)]. This diversity makes it challenging to create a universal understanding of privacy and, consequently, to develop a unified framework that ensures compliance with these regulations¹.

To address this challenge, Solove proposes a widely accepted taxonomy [[121](#)] that classifies privacy violations into distinct categories. This taxonomy provides a shared conceptual framework for understanding privacy, which can be leveraged to develop a unified approach for formally proving compliance with key regulatory requirements.

In this chapter, we discuss the common privacy principles shared across the APPs, GDPR, CCPA, PDPL, GCDPA and ADPPA in [Section 2.1.1](#). The distinct principles that vary between these regulations are explained in [Section 2.1.2](#). We then introduce Solove's taxonomy of privacy violations in [Section 2.2](#). Finally, [Section 2.3](#) presents a mapping between Solove's taxonomy and the key requirements embedded in these regulations, allowing a foundation to be established for designing a unified framework to ensure compliance within systems.

¹Voluntary frameworks such as *ISO/IEC 27701* [[59](#)] and the *NIST Privacy Framework* [[122](#)] aim to align privacy practices across jurisdictions. However, as they are not legally binding and do not impose penalties for non-compliance, this thesis focuses on enforceable privacy regulations.

2.1 Privacy Regulations

The APPs, GDPR, CCPA, PDPL and ADPPA share common privacy principles but also include distinct ones. In this section, we present both the common and the distinct principles.

2.1.1 Common Principles

1. Providing and Withdrawing Consent

These regulations require organisations to obtain consent before processing individuals' data and allowing users to withdraw their consent at any time. They stipulate that consent must be informed. *Informed* consent implies that individuals must be explicitly made aware of the type of data being collected, the purposes for data collection and the specific processing activities that will be performed on their data [22], [32], [45], [73], [107].

2. Right to Access

A core objective of these regulations is to give individuals full control over their personal information, including the right to access the data held about them by organisations at any time [22], [32], [45], [61], [107], [126].

3. Purpose Limitation

Under these regulations, organisations should collect personal data for explicit, legal, and specific purposes. Furthermore, the collected data must be processed only in accordance with these defined purposes [22], [32], [45], [70], [108], [126]. This principle helps prevent unauthorised or unintended processing by clearly limiting the scope of data usage.

4. Sharing with Third Parties

A *third party* refers to any organisation or entity that did not originally collect users' data but is granted access to it by the primary data collector. Sharing personal data with third parties is a critical aspect of user privacy, as it directly affects individuals' control over their personal information. Once data is shared with third parties, users lose a degree of control over how that data is used. For instance, data may be processed in ways that were not originally consented to, potentially violating privacy. It may also be further shared without the individual's permission, with each additional recipient creating a new

opportunity for misuse [28]. To mitigate these risks, the privacy regulations obligate organisations to obtain users consent before sharing their data with third parties. These regulations also require that consent be informed. This means users must be made aware of key details, *e.g.* the specific permissions granted to third parties, the purposes for which the data will be used and what data will be collected.

2.1.2 Distinct Principles

1. Notice at Collection

The CCPA obligates organisations to provide notice to users at or before the point of data collection. This notice should include details about the data being collected, the purposes for the collection and the applicable privacy policies [126]. Users also have the right to opt out by sending a request to organisations, asking them to stop using their information [126]. The key difference between providing consent and notice at collection is that the former requires users to explicitly accept the privacy terms, whereas the latter assumes agreement by default.

2. Consent Evidence

This principle requires organisations to keep records that demonstrate a user's consent. Under the GDPR, data controllers (the entities that determine privacy permissions and the purposes of processing users' data) must be able to prove that the data subject (the user) has provided consent [56]. This requirement ensures compliance and can be used as evidence for regulatory authorities during audits.

3. Requirements for Cross-Border Data Transfers

The GDPR is one of the most stringent privacy regulations [88]. It imposes specific legal bases for data transfers to non-EU countries [72]. This is because such transfers may affect EU citizens' privacy or national security, especially if the transferred data is sensitive [24]. The GDPR permits international transfers only when one of the following legal bases is met:

- (a) **Adequacy decisions:** this legal basis ensures that the recipient country provides a suitable level of data protection. The European Commission has specified a set of countries that provide an adequate level of protection, *e.g.* US, Canada, Japan *etc.*².

(b) **Appropriate safeguards:** if the adequacy decision does not cover the receiver country, then receiver country should provide appropriate safeguards. The GDPR specifies many safeguards. One is adopting *Standard contractual clauses (SCCs)*, a set of predefined standards and rules identified by the European Commission to protect users' data when it is transferred outside EU countries. These rules should be incorporated into the contract between the sender and receiver organisation [25]. Another safeguard is using the *certification mechanism*. The certification should meet certain criteria (scheme) related to transparent processing and users' rights, *e.g.* clarifying the storage mechanisms of the data, data subjects rights to access *etc.* . The certification also should be approved by certification bodies that the GDPR specifies, *e.g.* the European Data Protection Board (EDPB) [40].

(c) **Derogations:** the GDPR identifies some special situations in which data transfer is permitted, *e.g.* when transferring data relates to the public interest ³.

This thesis models the transfer based on the adequacy decision and the appropriate safeguards. We focus on modelling the *SCCs and certification mechanism* as the SCCs are commonly used among organisations, and the certification should meet some criteria that need to be checked [41].

2.2 Solove Taxonomy

Solove [121] introduces a taxonomy of privacy violations. This taxonomy is also explained and summarised by [81] and [130]. It organises privacy violations into four categories. In this section, we clarify each of these categories.

1. Invasion group:

This group includes two types of privacy violations: intrusion and decisional interference. As handling this category requires specifying a full physical security model, we only focus on the following three categories.

²To view the full list of countries, see https://commission.europa.eu/law/law-topic/data-protection/international-dimension-data-protection/adequacy-decisions_en.

³For further details, see https://www.edpb.europa.eu/sme-data-protection-guide/international-data-transfers_en.

2. Information collection violations include:

- (a) Surveillance: It means collecting users' data without their consent.
- (b) Interrogation: This means putting users in an embarrassing situation so they cannot refuse to answer a private question.

3. Information processing violations classified as:

- (a) Aggregation: This concept states that knowing a single piece of information about data subjects may not affect their privacy as much as knowing a group of information. For example, aggregating the collected information about users' activities, locations, and preferred purchases could affect their privacy. In other words, combining this information may provide more details about the users' personalities and lives.
- (b) Identification: It is the association between pieces of information and users' identities. The difference between aggregation and identification is that, in aggregation, the possibility of identifying an individual is low, as more information needs to be collected. Conversely, in identification, there is a high possibility of linking a specific user by using an identifier, *e.g.* a user's national ID number or address.
- (c) Insecurity: Relates to data holders' responsibilities to protect users' data.
- (d) Secondary Use: It refers to using users' data for purposes other than those to which they consented.
- (e) Exclusion: This violation occurs when users are not given their rights. For example, if users cannot access their data or if they do not know what types of data will be collected.

4. Information dissemination:

- (a) Breach of confidentiality: It occurs when a user's data is leaked by a trusted data holder, *e.g.* when a doctor discloses a patient's data.
- (b) Disclosure: This violation occurs when user data is disclosed to unauthorised parties. Unlike a breach of confidentiality, it does not involve a violation of trust.
- (c) Exposure: Revealing embarrassing information about a user.

- (d) Increased accessibility: This arises when data holders disseminate non-confidential information of users widely and it becomes more accessible, *e.g.* disseminating a list of users' emails could expose them to receiving spam emails.
- (e) Blackmail: The threat to disclose or disseminate users' private information.
- (f) Appropriation: Using users' identities to achieve specific interests without their consent, *e.g.* associating a known person's name with a certain product without that person's consent to increase a company's income.
- (g) Distortion: Spreading incorrect information about individuals that could affect their reputations.

2.3 Mapping Solove's Taxonomy to Privacy Regulations Requirements

This section maps key privacy requirements in the APPs, GDPR, PDPL, GCDPA, CCPA, and ADPPA to Solove's specific sub-categories of privacy violations. Each mapping is accompanied by an example that illustrates a violation resulting from non-compliance with the corresponding regulatory principle. The mapping is summarised in [Table 2.1](#) along with illustrative examples of violations.

Privacy laws such as the APPs, GDPR, PDPL, GCDPA, and ADPPA require organisations to obtain explicit user consent before handling their data, while the CCPA introduces the notion of notice at collection [\[126\]](#). These measures aim to prevent violations related to *surveillance*, a sub-category of information collection where data is collected without user awareness. A representative violation occurs when a banking app collects location data without obtaining users consent or informing them.

To address *secondary use*, most regulations incorporate purpose limitation principles [\[22\]](#), [\[32\]](#), [\[45\]](#), [\[73\]](#), [\[107\]](#), [\[126\]](#). These prohibit the repurposing of data beyond the original reason for collection without renewed consent. A violation of this principle can be seen in cloud healthcare platforms that reuse patient data—initially collected for health monitoring—to train AI models without informing the patients.

Regulations also guarantee users the right to access their data and withdraw consent [\[22\]](#), [\[32\]](#), [\[45\]](#), [\[73\]](#), [\[107\]](#), [\[126\]](#), helping prevent *exclusion* violations. A relevant violation occurs

when users of cloud-connected health monitoring systems are unable to view their stored data or revoke consent.

Finally, Solove’s *disclosure* sub-category concerns the unauthorised sharing of data. To counter this, these regulations restrict sharing with third-parties [22], [32], [44], [45], [108], [126]. A notable example of this type of violation is a company sharing user metadata with its parent company for advertising purposes without the user’s knowledge. Regulations such as the GDPR impose additional requirements, *e.g.* Standard Contractual Clauses (SCCs), when data is disclosed to third-parties across borders. Violations of these requirements can pose national security risks, *e.g.* if sensitive data belonging to government personnel or critical infrastructure employees is transferred to countries with weak data protection, the data could be accessed or exploited by foreign entities.

As these privacy regulations share common privacy requirements, this alignment can support the development of unified frameworks to detect violations and formally verify compliance.

2.4 Summary

This chapter provides an overview of several privacy regulations (APPs, GDPR, PDPL, GCDPA, ADPPA, and CCPA), highlighting their shared principles and distinctive requirements (Sections 2.1.1 and 2.1.2, respectively).

To provide a structured view of privacy, we introduced Solove’s taxonomy (Section 2.2), which classifies privacy violations into four categories. Finally, we mapped these categories to regulatory requirements (Section 2.3), establishing a unified perspective that supports the development of a formal compliance framework.

Table 2.1: Mapping privacy regulatory requirements to Solove’s taxonomy, with examples of violations.

Solove Category	Regulatory Principle	Violation Example
Information collection: <i>surveillance</i>	Provide consent / Notice at collection	A banking app collects location data without obtaining permission or informing users.
Information processing: <i>secondary use</i>	Purpose limitation	Cloud health data is reused for AI training without patient consent
Information processing: <i>exclusion</i>	Right to access / Withdraw consent	Users cannot view their stored health data or revoke their consent
Information Dissemination: <i>disclosure</i>	Third-party sharing	A company sharing user metadata with its parent company for advertising purposes without the user’s knowledge.

Chapter 3

Informal and Formal Privacy Methods

The importance of privacy leads to extensive research into methods that ensure and verify privacy across different systems. These methods range from informal approaches, which rely on non-mathematical techniques, to formal methods, which use mathematical models for rigorous analysis of privacy properties. Researchers explore various solutions aimed either at safeguarding personal data from unauthorised access or at verifying a system’s compliance with privacy regulations, *e.g.* GDPR.

[Fig. 3.1](#) provides a visual representation of the structure of this chapter. We present the relevant work by classifying existing approaches into two main categories: *informal* ([Section 3.1](#)) and *formal* ([Section 3.2](#)). Within the informal category, we distinguish between methods developed primarily for privacy protection—without an explicit focus on regulatory adherence—as outlined in [Section 3.1.1](#), and those designed to demonstrate compliance with legal frameworks ([Section 3.1.2](#)). Both subcategories share the use of blockchain as a common technique. Likewise, formal methods designed for privacy protection are discussed in [Section 3.2.1](#), while [Section 3.2.2](#) presents formal techniques for verifying system adherence to privacy regulations. Both *formal* non-regulatory and regulatory methods share several formal techniques, including theorem proving, formal systems, formal logic, and model checking, which we will discuss in [Section 3.2](#). Finally, we discuss the limitations of both informal and formal approaches in their respective sections and conclude with a summary in [Section 3.3](#).

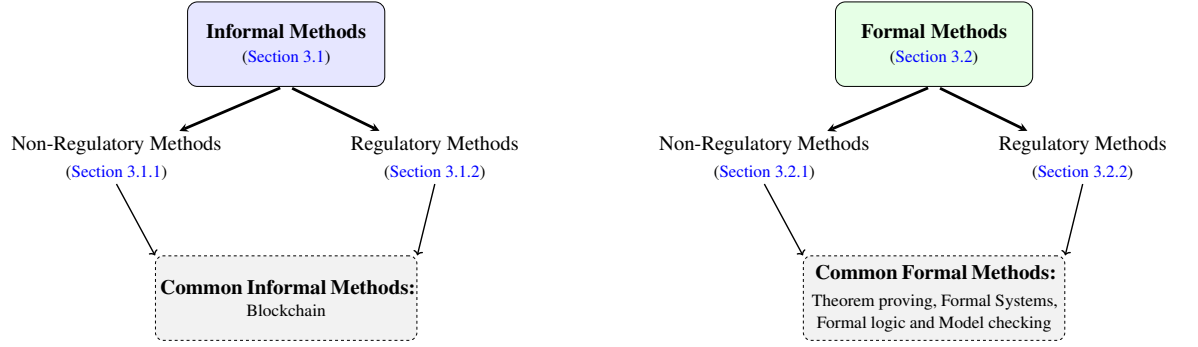


Figure 3.1: Chapter structure overview.

3.1 Informal Methods

Various informal approaches are proposed to enforce privacy within systems. These methods do not rely on formal mathematical techniques but still provide effective strategies to address privacy concerns. In the following sections, we explore two key types of these methods: non-regulatory strategies for privacy protection and approaches that support adherence to privacy regulations.

3.1.1 Non-Regulatory Methods for Privacy Protection

Non-regulatory methods focus on technical mechanisms that enhance privacy but are not explicitly designed to comply with privacy regulations, *e.g.* GDPR, CCPA *etc.* This section explores these approaches.

Blockchain

Researchers extensively investigate blockchain to protect the privacy and integrity of personal data¹. The concept of blockchain originated in the context of bitcoin transactions [100]. The idea of blockchain is to store bitcoin transactions in a chain of blocks, using cryptographic hash functions to secure them. Each block contains bitcoin transactions and a cryptographic hash of the previous block. This linking mechanism ensures the integrity of bitcoin because any change to a previous block would alter its hash, invalidating all subsequent blocks. This

¹Data integrity means protecting personal data from unlawful tampering or damage [69].

tamper-evident structure of blockchain strongly motivates its use for storing and protecting the integrity of sensitive data, such as patient records in the healthcare sector[109].

Another approach that utilises blockchain in the healthcare sector is the blockchain-based decentralised authentication system [138]. Traditional healthcare authentication systems store sensitive authentication details in centralised databases, which increases the risk of data breaches and unauthorised access. Blockchain-based decentralised authentication addresses this concern by leveraging the distributed nature of blockchain to provide a secure and transparent authentication process. Since authentication records are stored on the blockchain, authorised hospitals can securely access these records without relying on a central authority. This also eliminates the need for patients to repeat authentication steps when visiting these hospitals, thereby reducing authentication time.

Sharma et al. [119] propose a blockchain-based distributed application for IoT healthcare devices to enhance patient privacy and security. The system enables patients, physicians, and hospitals to store and generate medical certificates on a blockchain network, ensuring data integrity and preventing unauthorised modifications. Remote access to these certificates eliminates the need for physical documents, thereby reducing the risk of fraud in healthcare documentation.

While the immutable nature of blockchain provides robust protection against data tampering, it poses significant challenges under the right to erasure outlined in the GDPR, which requires the ability to remove personal information upon request [140]. To address this challenge, researchers have explored solutions such as off-chain storage as will be discussed in [Section 3.1.2](#).

Privacy by Design

Cavoukian introduced the concept of Privacy by Design [26], aiming to integrate privacy considerations into system design. Cavoukian provides seven guiding principles:

1. *Proactive, not reactive; preventative, not remedial*: this principle requires organisations and software engineers to design systems in a way that prevents privacy breaches rather than addressing them after they occur.
2. *Privacy as the default*: users should not be required to take action to protect their data, as the system should be designed in a way that protects their data automatically.

3. *Privacy embedded into design*: engineers should integrate privacy into the design phase of the system as a core component, not as an add-on requirement.
4. *Full functionality, positive-sum, not zero-sum*: developers must adopt a positive-sum approach, not a zero-sum approach, during system design. A positive-sum approach means improving privacy without sacrificing security; otherwise, it would become a zero-sum approach.
5. *End-to-end lifecycle protection*: this principle ensures that data is protected throughout its lifecycle, *i.e.* from the point it is collected until it is deleted.
6. *Visibility and transparency*: organisations' privacy policies must be demonstrated to users through their established practices and procedures.
7. *Respect for user privacy*: organisations must prioritise individuals' privacy rights by allowing users to easily access privacy settings and enabling intuitive controls.

Later, Cavoukian *et al.* [27] link these principles to the concept of accountability. Accountability means that organisations are responsible not only for protecting users' data but also for managing any risks that arise from failures in privacy protection. The authors also provide an example of how organisations can achieve accountability by applying privacy by design principles.

Although these guiding principles help software engineers maintain a focus on privacy, their broad nature raises the question of how each principle can be effectively implemented in practice. This ambiguity may result in overly complex system architectures when engineers attempt to satisfy the high-level guidelines.

Edge Computing

Edge computing [120] is a technique that allows data to be processed at the edge of the network, near the device where it is generated, instead of sending it to remote cloud servers. This decreases bandwidth costs and latency, while increasing the speed of data processing.

Since edge computing enables local data processing, it can enhance user privacy in IoT devices. Databox [141] leverages edge computing to process patient data collected via wearable devices locally, preventing the transmission of raw user data to third-parties. Databox also allows end-users to control the disclosure of sensitive information to third-parties, enhancing

privacy protection. This design mitigates the risk of data leaks caused by potentially weak security measures on third-party servers. However, Databox introduces usability challenges, particularly for older adults or individuals with limited technical experience, as it requires an additional layer of local data management that may be difficult for some users to navigate.

Similarly, Li *et al.* [87] introduce Mobile Edge Computing (MEC) as a solution to avoid sending users' raw data to a central cloud server for aggregation. In this approach, when data is generated by a terminal device (a user's device), it is transmitted to an edge server located close to the terminal device for local aggregation. The aggregated data is then transmitted to the public cloud centre. However, the aggregation process in this approach involves multiple data sources, creating a challenge in maintaining consistent security across them, which could lead to privacy violations.

Privadome

Privadome [113] is a system designed to assist e-commerce companies that utilise delivery drones in protecting citizens' privacy. Drones pose a privacy threat as they are equipped with cameras that capture videos during navigation processes, potentially recording individuals' footage. To implement Privadome, it is integrated with government agencies that regulate drones. This enables citizens to use their mobile devices to communicate with nearby drones (*i.e.* the system reveals their identities) and determine whether their footage has been recorded. Individuals can also request a tracking record of drone activities and verify whether privacy-preserving measures are applied to process the footage, *e.g.* blurring faces or deleting recordings. Although Privadome is designed to protect citizens' privacy, it may pose a privacy and security risk to delivery companies. This is because the system shares the identities of drones with citizens, allowing them to identify and track nearby drones, which could potentially be misused to monitor drone movements across the city.

3.1.2 Regulatory Methods for Compliance with Privacy Regulations

Unlike non-regulatory methods, several approaches are specifically developed to design systems that comply with privacy regulations. This section presents these approaches.

Blockchain and Smart Contracts

To address the previously mentioned limitation of blockchain regarding data storage and the GDPR right to erasure (see [Blockchain](#) in [Section 3.1](#)), Truong *et al.* [129] propose combining blockchain with smart contracts² to record all activities on user data, providing verifiable proof of compliance. The idea of this strategy is to store the actual data off-chain, and the reference to that data is stored on-chain. This method allows any operation on the data to be recorded in an immutable ledger and ensures that only authorised parties can access or process the data via smart contracts.

Although this combination of blockchain and smart contracts aids in detecting potential privacy violations, the approach introduces another challenge: the smart contracts themselves must also be verified to ensure their compliance with GDPR requirements. This adds complexity to regulatory adherence, as both the contract logic and its execution must be evaluated for compliance with GDPR.

Privacy-Aware Data

In response to the Privacy by Design approach [26] (see [Privacy by Design](#) in [Section 3.1.1](#)), Antignac *et al.* [9] introduced the Privacy-Aware Data Flow Diagram (PA-DFD), an extension of the Data Flow Diagram (DFD). The DFD is a graphical approach used to model the data flow of software systems. Antignac *et al.* extend the DFD by incorporating additional entities, enabling software engineers to represent GDPR principles graphically. PA-DFD allows software engineers to model two key GDPR principles: purpose limitation and retention time—ensuring that data is not retained once its processing purpose is fulfilled and, if no longer needed, is either deleted or anonymised.

Antignac *et al.* do not introduce an algorithm that allows software engineers to convert a DFD into a PA-DFD. To address this gap, Alshareef *et al.* [4] propose an algorithm for transforming a DFD into a PA-DFD. This approach enhances the ability of software designers to develop systems that comply with these two GDPR principles.

There are two limitations of PA-DFD: (1) its unsuitability for complex systems, as system designers must manually examine the PA-DFD, and (2) applying this approach to large-scale systems can be challenging, as the resulting model would also be large and difficult to manage [3].

²Smart contracts are programs that execute automatically when certain conditions are met [129].

Systematic Analysis

Guamán *et al.* [48] define a method that systematically analyses the compliance of Android mobile apps with the GDPR notion of international data transfer. The method analyses the app's behaviour to extract the personal information that will be transferred, the type of the recipient and the destination to specify if the transfer is covered by the adequacy decision. The privacy policy of the app is also analysed to identify the outlined transferring policy. The app's behaviour is checked against its data transfer policy, and both are checked against the GDPR requirements for international data transfers. This method is not entirely automated, but later it is converted to an automated tool by using Machine Learning (ML) and rule-based classifiers to perform the extraction process of the information and the privacy policy automatically [49]. Importantly, this technique checks transfers based on adequacy decisions and binding corporate rules but does not address certification safeguards. It is also limited to assessing the compliance of Android mobile apps, excluding other systems such as IoT devices.

Hunter

Pascual *et al.* [111] introduce Hunter, an automated tool designed for tracking data routing via the anycast communication protocol, which directs data to the nearest server regardless of its physical location. This method is developed based on the approach proposed by Guamán *et al.* [48]. Hunter identifies the destination of data routed through anycast communication within Android apps. Additionally, it analyses the privacy policies of these apps to determine their data transfer policies and assess whether the specified destination aligns with the app's stated policy—specifically, whether the app designates the destination as adequate. It then verifies whether both the app's stated policy and the actual data transfer comply with GDPR adequacy requirements. A key limitation of this approach is that the compliance analysis process is not fully automated, requiring manual intervention, which restricts scalability and efficiency. Furthermore, Hunter considers only one legal basis for cross-border data transfers (the adequacy decision) without addressing other legally recognised bases.

3.1.3 Limitations

The approaches discussed in the previous section are undoubtedly valuable for protecting users' privacy and facilitating compliance with relevant regulations. However, they primarily concentrate on implementation-level mechanisms, meaning they are applied after the design phase rather than as part of it. This means they overlook the crucial question of **how** to guarantee robust privacy protections **during** the system design phase. Specifically, none of these approaches provide a systematic way to prove the absence of design errors or vulnerabilities that could lead to privacy breaches at an early stage of development.

Even privacy-by-design guidelines, those proposed by Cavoukian [26] (see [Privacy by Design](#)), do not provide detailed methodologies for translating these principles into practice nor do they illustrate how to rigorously verify compliance with them. This lack of a formal, provable foundation creates a gap between high-level privacy goals and their concrete implementation in systems.

To address this gap, several *formal* approaches are proposed in the literature. Formal methods provide a mathematical framework for proving privacy requirements, enabling rigorous analysis to ensure that a system's design meets the intended privacy specifications from the outset of its development. Unlike informal methods, which rely on potentially error-prone reasoning, formal methods can provide mathematical proofs that a system satisfies its intended privacy requirements [128]. In the following sections, we present some of these formal approaches.

3.2 Formal Methods

As we mentioned previously, formal methods are approaches provide logical foundations to analyse and prove compliance with privacy requirements. The following sections explore how different formal methods are used to design systems that meet privacy requirements. We categorise these methods based on their aim into non-regulatory and regulatory methods. However, there are common formal methods across these categories:

1. Theorem proving: it is a branch of mathematical logic used to prove that a given property holds within a system [102].
2. Formal Systems: in the context of this thesis, we refer to formal systems as mathematical

techniques used to reason about systems, which include: π -calculus, which models the concurrent interaction and communication between processes and components; session types, which define structured communication protocols and their sequencing within such systems; and λ -calculus, which models sequential computations.

3. Formal logic: formal logic is a formal language that consists of logical constructs and rules used to define systems in a structured way, enabling precise understanding, rigorous reasoning and thorough analysis of their properties.
4. Model checking: it is a type of formal verification used to automatically verify whether the intended properties (*e.g.* privacy requirements) hold across all possible configurations (states) of a system at any given time.

3.2.1 Non-Regulatory Methods for Privacy Protection

In this section, we discuss formal techniques used to ensure that systems are robustly designed to protect user privacy without ensuring compliance with privacy regulations.

Differential Privacy

Differential privacy is a formal technique designed to protect users' private data when querying databases [38]. The objective of this approach is to enable analysts to extract statistical information from a database without revealing individuals' identities. This is achieved by adding a controlled amount of noise to the query results.

Later, differential privacy is applied to various applications, including its use in IoT devices to protect user data [55]. There are two main techniques for implementing differential privacy in IoT systems: (1) adding noise to raw user data on the server, known as *Centralized Differential Privacy (CDP)*, or (2) adding noise to raw user data on the user's device before transmission to the server, known as *Local Differential Privacy (LDP)*. LDP offers an additional layer of security because, in CDP, the server must be fully trusted to protect the data. Otherwise, the server could be compromised, leading to a breach of user privacy.

Although differential privacy enhances data protection, the added noise may reduce the reliability of the results [39]. Because the amount of noise required to ensure privacy can reduce data accuracy, it can impact analysis processes that require high precision. Another key limitation of differential privacy is that small changes to the dataset, *e.g.* the addition

or removal of a single individual, can affect the output of a computation. If the added noise does not adequately account for this change, the resulting mechanism may fail to satisfy the intended privacy guarantees.

Theorem Proving

Theorem proving is used in the literature to prove privacy properties within systems. One notable application of theorem proving in privacy verification is the Taint, Track, and Control (TTC) approach [54]. TTC is a formal technique based on the privacy by design principle ([Privacy by Design](#) in [Section 3.1](#)) to enforce user-defined privacy policies in online applications. The approach consists of three steps: marking user input with a unique identifier (Taint), tracking the dynamic flow of the marked input within systems (Track), and controlling outputs by checking privacy policies before releasing the data (Control). The semantics of the approach are encoded as theorems and their correctness is proven using Isabelle/HOL [103], an automated proof assistant framework. Although the effectiveness of the approach is demonstrated by applying it to online applications, it does not support communication between multiple components or services within a system, which is essential in modern software systems, *e.g.* cloud computing, distributed systems *etc.*

Sato et al. [116] introduce a formal approach to verifying differential privacy using the Isabelle/HOL theorem prover. Their work specifically addresses the key limitation of differential privacy: that small changes to a dataset can affect the output of a query and potentially violate privacy guarantees (see [Differential Privacy](#)). To address this, they formally define the mathematical properties of differential privacy, *i.e.* its core guarantee that outputs remain stable when a single individual is added/removed from the dataset, within Isabelle/HOL. They then prove that two widely used mechanisms, the Laplace mechanism and the report noisy max mechanism, behave correctly under such changes.

Theorem proving techniques are also used to verify the correctness of Cedar's privacy policies. Cedar is an authorisation policy language designed to enable fast and analysable access control decisions [33]. It allows developers to write privacy policy logic separately from application code and uses an engine to evaluate access requests. For formal guarantees, Cedar's design is modelled in the Lean proof assistant [99], enabling key system properties to be formally verified using proof-based methods. Cedar has been adopted in real-world systems, including *Amazon Web Services*, to demonstrate its effectiveness.

While theorem proving offers strong guarantees, it can be challenging for non-experts, as the process often involves manual effort and a certain level of expertise due to limited automation.

Formal Systems

Given that data mobility between processes is a key component of privacy, many approaches are based on Milner's π -calculus [95], a formal model of concurrency. For example, Kouzapas and Philippou [81] proposed privacy calculus, an extension of π -calculus to reason about privacy within systems. The concept of privacy calculus is based on π -calculus with groups [23]. π -calculus with groups is introduced to prevent information leaks outside a certain group (where a group represents a type of channels). In other words, the purpose of groups is to facilitate the exchange of secret information within a specific scope, ensuring that confidential data does not leak beyond the group. Kouzapas and Philippou leverage the concept of groups as they divide system's entities into groups and assign permissions to each entity. These groups are formally modelled using the syntax of privacy calculus. The authors also define a set of type rules to apply the type checking technique, a verification method that ensures the modelled system adheres to the defined type rules. This technique is used to verify that the formalised system complies with the specified privacy policies.

Another approach based on π -calculus is the ProVerif tool [90]. ProVerif is a theorem prover built on applied π -calculus that analyses security protocols [20] and allows unlinkability and anonymity properties to be specified. The author extended ProVerif to verify privacy-related properties using the equivalence-based approach. Checking equivalence-based technique is the hacker's inability to distinguish between the ideal protocol (the protocol that has the properties that intended to be proved) and the actual protocol that is aimed to be checked. If the adversary is unable to distinguish between them, it confirms that the protocol satisfies the required privacy properties. Otherwise, the protocol fails to maintain privacy.

Unlike π -calculus, λ -calculus [30], which is a formal model that represents computation using functions. It is used to develop a programming model that ensures system outputs comply with privacy policies defined by developers [137]. The model consists of two components: the core program and the declarative privacy policy. The former handles the main functionality of the program independently of privacy enforcement, while the latter defines rules that specify how and when sensitive data can be disclosed. The programming

model is implemented through the development of Jeeves. Jeeves is a functional constraint language based on λ -calculus. Jeeves represents private data as symbolic variables and privacy policies as constraints. Before displaying or sharing these variables, the system automatically checks the privacy constraints associated with these symbolic variables and adjusts the output accordingly. However, Jeeves has a limitation in terms of static checking³. The authors use a type-checking system to verify the correctness of Jeeves, but it does not provide detailed feedback on complex program properties, *e.g.* whether privacy policies are free of contradictions or cover all possible data usage scenarios.

Trace Semantics

Trace semantics is a formalism used to model actions that are observed during the process execution as a sequence (trace) [15]. Several privacy languages are developed using trace semantics. For example, S4P is a privacy language designed to allow users to define their privacy preferences and check whether systems comply with these preferences [17]. User policies and system behaviours are encoded using trace semantics to verify whether system behaviours adhere to user-defined policies. One limitation of this approach is that it cannot check dynamic changes required by privacy policies, *e.g.* providing and withdrawing consent.

Another privacy language developed based on trace semantics is SIMple Privacy Language (SIMPL) [86]. It is a natural language framework that allows both data subjects (users) and data controllers to specify privacy policies. Developers encode system implementations using trace semantics to ensure that SIMPL statements align with system functionalities. While this framework focuses on ensuring compliance with user-defined policies, it does not account for data sharing across multiple applications, which poses challenges when applying it to real-world scenarios.

Formal Logic

Several works use formal logic to model and enforce privacy policies within systems. One such approach is Rei is a privacy policy language developed by Kagal [78] to express and enforce user-defined privacy policies within systems. It is a flexible language that can be applied to various systems and is not limited to a specific domain. Rei is based on deontic logic, a formalism that models obligations, permissions, and rights, to define policies in a

³Static checking means that the correctness of code is checked without executing the program.

structured way. The author then encodes these privacy policies using First-Order Logic (FOL), a formal language used to logically describe systems, their components and their relationships, to facilitate their translation into a machine-readable format. A policy engine is developed to enforce these machine-readable policies within systems. A key limitation of this approach is that it requires further improvements to handle conflicts between privacy policies.

While Rei focuses on defining and enforcing privacy policies using deontic logic and FOL, other approaches, such as Description Logic (DL), explore privacy concerns from a knowledge representation aspect. DL is a fragment of first-order logic used in knowledge representation languages to formally define and reason about concepts and their relationships in a logical manner [101]. DL is widely used for reasoning about relationships in ontology-based systems. Unlike databases, which primarily store structured data, ontologies store and infer knowledge by defining concepts and their relationships [104]. Since ontologies store knowledge rather than structured data, they can be used to infer confidential information. Nuradiansyah [104] proposes a formalism using DL to assess the potential of ontologies to infer confidential information. This technique consists of three steps: (1) checking whether confidential information can be inferred from the ontology, (2) modifying the ontology to prevent privacy breaches while preserving its utility, and (3) re-evaluating the modified ontology to ensure that confidential information can no longer be inferred. Fixing the ontology using this technique does not fully guarantee privacy compliance, as an attacker could still infer confidential information by utilising other ontologies as external sources.

Another work that uses formal logic is the Obligation Specification Language (OSL). It is a framework designed to enforce user-defined privacy policies within distributed systems [51]. These privacy policies are formally defined using Linear Temporal Logic (LTL), a formal logic used to reason about temporal properties and the ordering of events over time within systems. OSL translates these encoded policies into Digital Rights Management (DRM) mechanisms, which legally manage access to digital content such as files and documents, to allow the enforcement of privacy policies using DRM tools. The authors evaluate their approach through a proof of concept using Microsoft's Rights Management Services (RMS), a system used to restrict accessing to information [93], to test OSL and verify whether the policies are properly enforced. However, a limitation of OSL is that it has only been tested under specific environmental conditions (*e.g.* user roles, device types), restricting its effectiveness to these conditions.

Model Checking

Model checking is commonly used for verifying privacy within systems. For instance, Lu *et al.* [89] introduce behaviour-aware privacy to ensure privacy compliance in web service composition using model checking. Web service composition refers to the integration of multiple web services to process and fulfil users' requests. This means that individuals' data may be shared across different web services. Accordingly, behaviour-aware privacy focuses on controlling and defining restrictions on accessing users' information during interactions between these web services. The authors propose a model-checking approach to verify whether web service compositions meet behaviour-aware privacy requirements. They define an interface automata to model the behaviour of web service composition and convert it into a Promela description. The SPIN model checker is then used to verify whether the modelled web service composition behaviour complies with the privacy requirements expressed in LTL. However, this approach primarily focuses on access control verification and does not address other privacy aspects, *e.g.* data integrity.

Another approach employing the SPIN model checker is PILOT [110]. It is a formal privacy policy language designed to assist data subjects and data controllers in defining, comparing and analysing their privacy policies. PILOT provides risk analysis assessments to evaluate the robustness of the defined privacy policies and to identify potential conflicts between them. This assessment increases users' awareness and helps them understand how they can edit the defined policies to better preserve their privacy. To perform the risk assessment, the authors model PILOT events and policies using Promela. The risk analysis questions are encoded using LTL. The SPIN model checker then checks all possible execution paths of events to detect violations of the LTL properties. The risk assessment is relatively simple as it does not consider the aggregation of different types of data, which could lead to privacy breaches, *e.g.* combining browsing history, purchase records and location data may reveal sensitive information about users.

The model checking technique is also utilised by Joshaghani and Mehrpouyan [77] to define the privacy management system, an approach proposed to address privacy issues arising from data-sharing processes. This approach allows users to define custom privacy policies and generates management rules based on user relationships and behaviours. The privacy policies are converted into a machine-readable format by breaking them down into four constructs: recipients, data, purpose, and conditions. The PAT model checker is then used to prove the

system's compliance with the machine-readable policies and to identify any conflicts between them. However, the approach does not resolve conflicts between user-defined policies, which may arise when multiple policies impose contradictory constraints on data access or sharing.

3.2.2 Regulatory Methods for Compliance with Privacy Regulations

Since formal methods provide a rigorous approach to proving system compliance with privacy regulations, this section presents formal techniques designed to ensure regulatory adherence.

Theorem Proving

Kammueßer [79] employs a theorem-proving technique to verify the compliance of IoT healthcare systems with the GDPR. The author uses Isabelle/HOL to formalise and prove GDPR principles. The proofs include users' right to access their data and their right to delete their data. These principles are encoded as theorems and verified using the Isabelle prover on a logically modelled system. Successfully proving these theorems confirms that the system meets GDPR principles.

As mentioned earlier in [Section 3.2.1](#), although theorem proving is highly accurate, it can be complex for non-experts to apply in practice, as the proving process is not fully automated.

Formal Systems

Privacy calculus that is introduced by Kouzapas and Philippou [81] (see [Formal Systems](#) in [Section 3.2.1](#)) is extended by Vanezi *et al.* [133] to reason about systems' compliance with the GDPR. They extend the privacy calculus syntax to model the GDPR notions of providing/withdrawing consent, right to erasure, purpose limitations and right to access. The authors also use type checking technique to statically analysis the systems and prove its compliance with the GDPR.

Further extensions, use multiparty session types [43], a formalism that describes the sequence and direction of interactions between system entities. Vanezi *et al.* [131] model the GDPR notion of processing user data for specific purposes by defining these purposes as multiple sessions types that can run in parallel or overlap. The authors introduce the *DiálogoP* tool, which provides a graphical representation of these purposes and automatically converts this representation into session types.

Building on *DiálogoP*, the researchers later developed another tool called *AnálisisP* [132]. This tool integrates the purpose of data processing into the requirements analysis phase of software engineering. It then converts these purpose-aware requirements into Unified Modelling Language (UML) use case and sequence diagrams. These diagrams are used as inputs for *DiálogoP*, which then transforms them into session types. A key limitation of these tools is the tool is designed to model only the GDPR notion of purpose limitation, without considering other privacy notions.

Layered Privacy Language

Layered Privacy Language (LPL) is a formal language developed to define and enforce GDPR principles within systems [46]. The language captures key privacy concepts such as user consent, purpose limitation, and data retention time. LPL focuses on query-based anonymisation, which enables data anonymisation when sharing data with third parties while still allowing useful data processing based on predefined privacy rules. For instance, when a doctor queries a patient database, they can access all patient details. Conversely, if researchers query the same database, they only receive anonymised data, *e.g.* average ages or patient gender distribution. The main limitation of LPL is its current inability to support real-time privacy rule changes, such as granting or withdrawing consent in real time.

Formal Logic

Privacy-aware Active Object Language (P-AOL) is proposed to handle users' personal data based on their consent [14]. It extends a core active object language—where objects have their own thread of execution [21]—by defining new privacy constructs that enforce the GDPR concept of consent in distributed systems. The authors provide operational semantics for the language and use FOL to prove its correctness. The approach supports checking the consent of multiple users. However, this method leads to considerable runtime overhead due to the extensive checks for each transition rule handling data. Another limitation of this method is that it only supports the notion of consent.

Model-driven privacy [83] is another approach uses formal logic to integrate GDPR principles of consent and purpose limitation into the implementation phase of software development. The authors introduce a privacy model that encodes these principles using FOL and defines rules for handling personal data based on these notions. They also provide model transfor-

mations that automatically enforce and integrate the privacy model into the implementation phase of web applications in C# and Python. However, the model transformations are not fully verified, which could introduce errors that may lead to privacy breaches.

Building on the idea of using formal logic for privacy compliance, Hublet *et al.* [53] propose a framework for automatically enforcing GDPR compliance based on the TTC approach [54] (see [Theorem Proving](#) in [Section 3.2.1](#)). The approach focuses on enforcing users' rights to access and erase their data, as well as their right to provide or withdraw consent within web applications. The authors formalise these principles using Metric First-Order Temporal Logic (MFOTL), a formal language used to model rules, system objects, and their changes over time with specific time restrictions (*e.g.* a password must be updated every 60 days). These formalised privacy principles are automatically enforced via WebTTC+, an enforcement environment that ensures compliance with the formalised principles in web applications and takes action if they are breached. The approach is evaluated through three case studies; however, its efficiency in large-scale contexts, such as high traffic and large volumes of data, remains untested. Additionally, the framework needs to be extended to capture other GDPR principles beyond access, erasure and consent.

Formal logic is also used to develop frameworks that enforce regulations other than the GDPR. DeYoung *et al.* [36] introduce PrivacyLFP, a logical language designed to enforce HIPAA (Health Insurance Portability and Accountability Act) and GLBA (Gramm-Leach-Bliley Act). The authors focus on modelling rules related to the purpose of disclosures, real-time constraints, and self-referential rules, *e.g.* rules that refer to themselves under specific conditions. These rules are encoded using Least Fixed Point Logic (LFP), an extension of First-Order Logic that supports expressing recursive constraints, *e.g.* accessing data multiple times under a specific condition. However, the framework only formalises HIPAA and GLBA without verifying system compliance with these regulations (*e.g.* via model checking).

Another framework that utilises formal logic to encode HIPAA, GLBA and COPPA (Children's Online Privacy Protection Act) is the Contextual Integrity (CI) framework [16]. This approach analyses user expectations regarding privacy and relates them to norms (rules) that govern the sharing and transformation of user data. These norms can be positive (allowing a process) or negative (preventing a process). The norms are encoded using LTL and applied to a model consisting of agents that communicate with each other to share data. The framework applies privacy norms to general information types rather than specific data values, *e.g.* patient information instead of an individual patient's name or ID. This limitation may affect its

applicability in modelling privacy regulations that require handling specific data values.

Model Checking

Privacy APIs [91] extend access control systems, which define who can access certain information within a system based on predefined rules, to enforce HIPAA compliance within systems. Privacy APIs model HIPAA rules using Promela and utilise the SPIN model checker to simulate these rules and detect undesired sequences. The authors also define an auditable privacy system, a formal approach for understanding the required operations to model privacy policies and identifying the minimal changes needed to implement these policies using traditional access control systems. However, the approach is limited to modelling HIPAA and captures only a subset of its rules.

The Model-Based approach to Identify Privacy Violations from software requirements (MBIPV) is a technique that uses model checking to detect GDPR violations *et al.* [139]. MBIPV defines GDPR requirements using UML class diagrams, which are then converted into a Kripke structure using the tool dom4j.jar[37]. The model checker NuSMV[105] is used to verify whether the resulting Kripke model satisfies the intended privacy properties, which are specified in Computation Tree Logic (CTL) [31]—a temporal logic used to determine if certain properties hold across all or some paths in a branching model. While MBIPV is effective for detecting privacy violations during early design phases, it does not support reasoning about GDPR concepts related to data sharing with third parties.

Beyond verifying system requirements, model checking are also used to encode GDPR principles directly into executable models. Data Protection Language (DPL), developed by Karami *et al.* [80], provides an object-oriented formalism based on multiset rewriting. It supports GDPR requirements: providing/withdrawing consent, purpose limitation, storage limitation, and the right to be forgotten. A formal model for DPL is given by encoding the operational semantics using multiset rewriting and the Maude model checker [92]. The checking process of the model was not entirely automated and required pen-and-paper proofs.

Another work based on the Maude model checker is the extension of P-AOL [13] (the original work on P-AOL [14] is discussed in [Formal Logic](#) in [Section 3.2.2](#)). The author extends P-AOL by providing a prototype implementation using the Maude formalism to validate the language's semantics and model-check the GDPR requirements, *i.e.* handling users' personal data based on their consent. However, the approach is still limited to checking

only the GDPR notion of consent.

Trustworthy autonomous agents are developed to manage user data collection decisions in compliance with the GDPR [136]. These agents aim to improve usability by automatically handling consent decisions, reducing the need for users to consent to repeated data-sharing requests. The approach uses cognitive decision-making, where agents model user beliefs and goals to decide whether to accept or reject data-sharing requests. To ensure correctness, the authors implement these agents using the Gwendolen language and verify their behaviour through the Model Checking Agent Programming Languages (MCAPL) framework [35]. Although effective, the approach has limitations, such as state space explosion during verification and the assumption that the agent remains trustworthy. Additionally, its fully autonomous nature reduces user involvement and transparency, which poses a potential conflict with the GDPR's transparency principle⁴ and may undermine user trust.

3.2.3 Limitations

Although non-regulatory methods can effectively protect privacy, they are not specifically tailored to model privacy regulations. The constructs these methods define do not support the dynamic nature of privacy legislation, *e.g.* real-time updates required in dynamic consent management or the complexities of cross-border data transfers [98].

In light of these limitations, specialised regulatory methods have emerged to effectively model and enforce privacy legislation. Table 3.1 summarises key characteristics of the formal approaches developed to capture these regulations. Notably, many of these methods employ dynamic verification, enabling the analysis of system behaviours in response to real-time regulatory changes. In contrast, static verification limits a system to a fixed configuration during the analysis phase, potentially reducing its ability to accommodate real-time updates. However, static verification remains vital for ensuring that desired properties are preserved, even in the presence of state space explosion⁵, which can make the verification process impractical.

The table also indicates that few existing approaches offer diagrammatic representation that help to enhance the accessibility of the approach. Additionally, few of these methods

⁴Transparency means individuals must be clearly informed about how their personal data is processed, in a concise and understandable way [60].

⁵State space explosion is a common challenge in model checking, where the number of system states grows rapidly and significantly complicates the verification process.

capture spatial properties—crucial when modelling privacy requirements in location-sensitive contexts. While cross-border data transfers are a major concern in practice (as discussed in [Chapter 1](#)), most surveyed methods do not address this issue directly.

Among the methods presented in the table, none simultaneously support both dynamic and static verification, provide diagrammatic representations and capture spatial properties for modelling cross-border data transfer regulations. This highlights the need for more comprehensive formal frameworks that support all these features.

In this thesis, we investigate the use of bigraph reactive systems to develop a framework that integrates both static and dynamic verification, diagrammatic modelling for representing and analysing privacy regulations, and explicit spatial representation.

3.3 Summary

In this chapter, we presented two categories of privacy-protection methods: informal methods ([Section 3.1](#)), which focus on either non-regulatory or regulatory compliance, and formal approaches ([Section 3.2](#)). In [Section 3.1.3](#), we showed that informal methods primarily address implementation-level mechanisms yet do not guarantee that a system’s design is free from potential privacy violations. This limitation is addressed by various formal methods—both non-regulatory and regulatory—introduced in [Section 3.2](#). However, they also have some limitations (*e.g.*, the inability to simultaneously capture spatial properties for modelling cross-border data transfer requirements, support both static and dynamic verification, and provide diagrammatic representation to enhance the approach accessibility) as discussed in [Section 3.2.3](#).

Table 3.1: Comparison of Formal Methods for Modelling Privacy Regulations. PA: Partially Automated, D: Dynamic, S: Static, (-): Not mentioned, ThP: Theorem Proving, GDPR: General Data Protection Regulation, HIPAA: The Health Insurance Portability and Accountability Act, COPPA: The Children’s Online Privacy Protection Act and GLBA: The Gramm–Leach–Bliley Act.

Approach	Regulation	Automated Verification	Verification Type	Spatial Properties	Cross-Border Transfer	Diagrams
PrivacyAPIs [91]	HIPAA	✓	D	×	×	×
MBIPV [139]	GDPR	✓	D	✓	×	✓
Isabelle/HOL [79]	GDPR	PA	ThP	×	×	×
Hublet [53] <i>et al.</i>	GDPR	✓	D	×	×	×
CI [16]	HIPAA, COPPA, GLBA	(-)	(-)	×	×	×
Model-driven [83]	GDPR	(-)	(-)	×	×	×
PrivacyLFP [36]	HIPAA, GLBA	(-)	(-)	×	×	×
DPL [80]	GDPR	PA	D/ThP	×	×	×
P-AOL [14]	GDPR	✓	D	×	×	×
Privacy calculus [133]	GDPR	(-)	S	×	×	×
DiálogoP [131]	GDPR	(-)	(-)	×	×	✓
LPL [46]	GDPR	(-)	(-)	×	×	×
Autonomous agents [136]	GDPR	✓	D	×	×	×

Chapter 4

Bigraph Reactive Systems

This chapter provides a foundational overview of bigraph reactive systems (BRSs). It begins by introducing the concept of bigraphs ([Section 4.1](#)) and their key constituents ([Section 4.2](#)). It then explains how these structures extend into BRSs through reaction rules that enable the representation of dynamic behaviour ([Section 4.3](#)). A concise example is provided to illustrate these principles ([Section 4.4](#)), followed by a discussion of the key features of BRSs ([Section 4.5](#)). The chapter concludes by introducing Bigraph Evaluator and Rewriting (BigraphER), an automatic tool for modelling, rewriting, and visualising bigraph models ([Section 4.6](#)).

4.1 Bigraphs

Bigraphs are a universal model of computation that describes systems based on how entities change their connectivity and locality (placement) as the system evolves [96]. A key benefit of bigraphs is that they have an equivalent algebraic and diagrammatic (visual) notation allowing them to be used by those without a formal background in mathematics [18], [94] without sacrificing mathematical rigour, *e.g.* their ability to prove system properties. An example of the diagrammatic representation of a bigraph B is provided in [Fig. 4.1a](#), where the first room (R1) contains a camera (C1). The room (R1) represents a communication channel to a central hub or another room via link x . Outside R1, there is another camera (C2) that is disconnected. The second room (R2) is located on another floor and contains an employee (Em). Outside R2, there is a camera (C3) that is linked to C1 and could potentially be linked to other cameras

or devices via link y . The equivalent algebraic form of the bigraph in Fig. 4.1a is given by:

$$/e ((R1_x.C1_e \mid /z C2_z) \parallel (C3_{e,y} \mid R2.(id \mid Em)))$$

For a detailed explanation of the algebraic notations used, refer to Table 4.1. In this thesis, we primarily focus on diagrammatic notations to enhance accessibility.

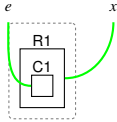
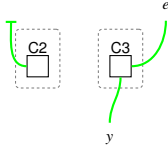
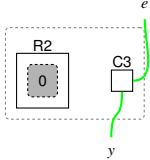
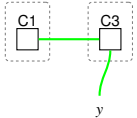
4.2 Constituents of bigraphs

The bigraph B consists of two interrelated structures: a *place graph*, illustrated in Figure Fig. 4.1b, and a *link graph*, depicted in Figure Fig. 4.1c. Both graphs share the same set of entities. The place graph forms a tree structure, while the link graph is a hypergraph. This dual representation enables the modelling of both spatial (placement) and non-spatial (connectivity) relationships between entities, as discussed in the following sections.

4.2.1 Place Graph

The place graph is a hierarchical tree structure that consists of a set of vertices (V), which represent nested entities, *i.e.* $V = \{v_0, v_1, v_2, v_3, v_4, v_5\}$. The place graph captures spatial and containment relationships, *e.g.* the camera $C1$ is located in the room $R1$, as shown in Fig. 4.1b. Entities are represented graphically via shapes, and we sometimes distinguish entities using colours. Each entity has a type called control. The controls in Fig. 4.1b are: $R1, C1, C2, C3, R2$ and Em . An entity is considered *atomic* if it has no children, *e.g.* $C1, C2, C3$ and Em . Grey dashed rectangles are *sites* that represent entities/bigraphs that have been abstracted away, that is, an unspecified bigraph can be placed there. For example, site 0 may contain another employee, *e.g.* ($Em2$). The dashed unfilled rectangles are *regions* that represent (disjoint) areas of the system, *e.g.* The room $R1$ is on a different floor than the room $R2$. Note: we do not need to say *where* it is, only that it is somewhere else. Parallel regions are often used to model *perspectives* where different concerns are modelled independently, *e.g.* privacy and systems concerns. We discuss this further in Chapters 5 and 6.

Table 4.1: Bigraphs algebraic notations and their equivalent diagrammatic forms.

Algebraic Notation	Term	Diagrammatic Notation
id	Identity	
1	Empty Region	
$/z \ C2_z$	Link closure	
$R1_x.C1_e$	Nesting	
$/z \ C2_z \parallel C3_{e,y}$	Parallel Product	
$R2.\text{id} \mid C3_{e,y}$	Merge Product	
$/e \ (C1_e \parallel C3_{e,y})$	Closed Link	
x	Idle name	x

4.2.2 Link Graph

Fig. 4.1c shows the link graph of bigraph B . The link graph models connectivity (non-spatial relationships) between entities through edges, *i.e.* $E = \{e_0, e_1, e_2, e_3\}$. Open links connect to names, *e.g.* x and y , that, like sites, specify this link *might* connect elsewhere. A closed link, *e.g.* the link connecting $C1$ with $C3$ as shown in Fig. 4.1c. Closed links may be one-to-zero hyperedges, *e.g.* the link attached to $C2$. An idle name is a link that exists, without being connected to any other entities or names. For ease of readability, we sometimes colour links connecting different types of entities.

As shown in Fig. 4.1a, bigraph B has outer and inner names which are x and y respectively. The outer names are written above the bigraph, while the inner names are written below the bigraph. Each entity has a fixed *arity*, defined as the number of ports associated with it. Ports determine the number of links an entity must have to connect with other entities in the bigraph. For example, the arity of the entity $C3$ is 2, while the arity of $R2$ is 0 as it has no ports.

Definition 3.1 (interface). Each bigraph has an interface that enables it to compose with other bigraphs. A bigraph's interface is defined by its inner face (the number of sites and inner names) and its outer face (the number of regions and outer names). For example, the interface of bigraph B (Fig. 4.1a) is given by:

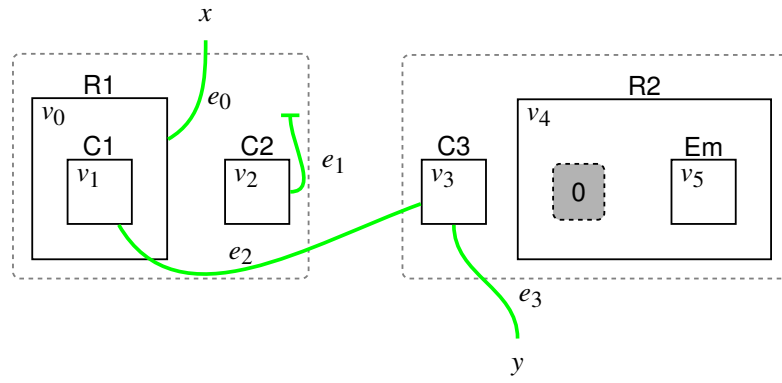
$$B : \langle 1, \{y\} \rangle \rightarrow \langle 2, \{x\} \rangle$$

Now consider another bigraph, G , with the following interface:

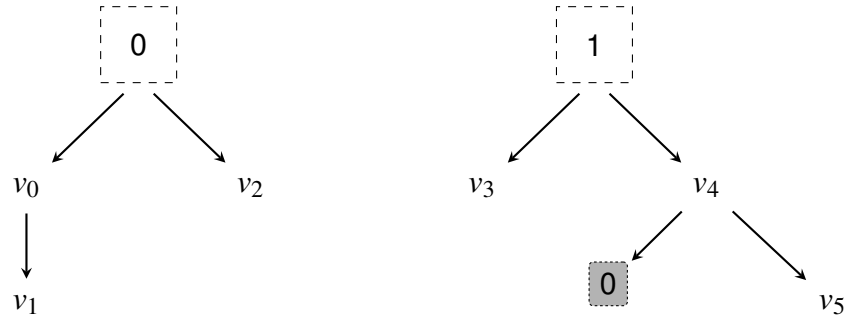
$$G : \langle 0, \{a\} \rangle \rightarrow \langle 1, \{y\} \rangle$$

To compose B and G , the number of sites in B must match the number of regions in G , and B 's inner name y must link to G 's outer name y .

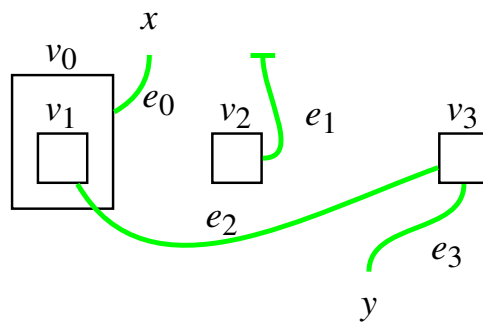
Definition 3.2 (signature). In bigraphs, a signature is a set whose entities are controls and the number of ports associated with each control. A signature is expressed as (κ, ar) , κ denotes the set of controls and ar assigns an arity to each control. In our example, the signature of bigraph B (Fig. 4.1a) is $\kappa = \{R1 : 1, C1 : 1, C2 : 1, C3 : 2, R2 : 0, Em : 0\}$.



(a) Bigraph B .



(b) Place graph.



(c) Link graph.

Figure 4.1: Constituents of Bigraphs.

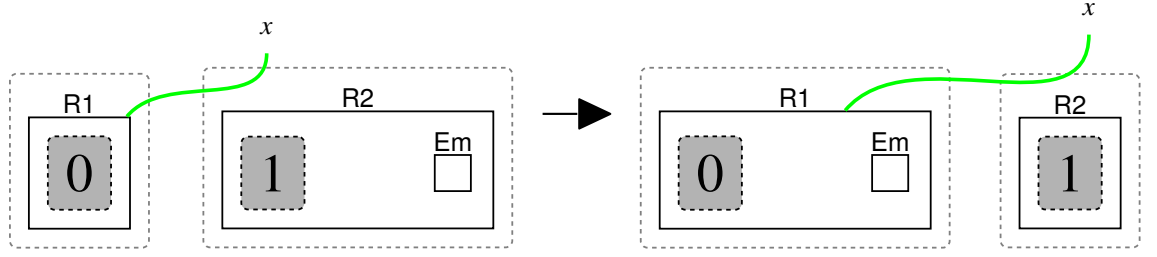
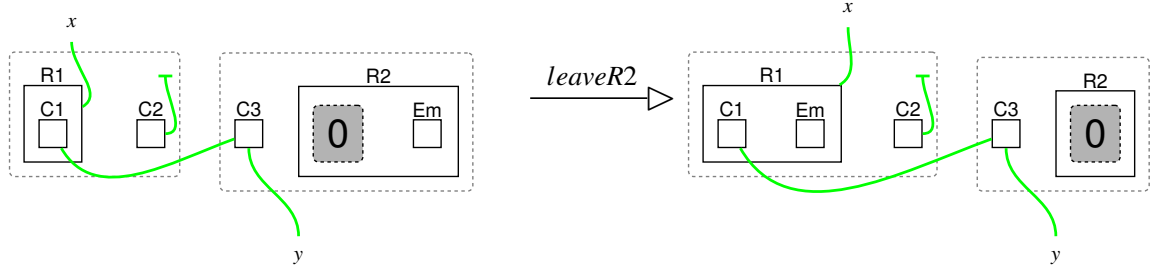


Figure 4.2: leaveR2: Movement of Em from R2 to R1.

Figure 4.3: The resultant bigraph B' after applying rule `leaveR2`, in which Em is moved from room R2 to room R1.

4.3 Bigraph Reactive Systems (BRSs)

We specify how bigraphs can evolve over time by using reaction rules. Throughout the thesis we use rule to mean reaction rule. Each rule, $L \rightarrow R$, consists of a left-hand side (L), representing the pattern that will be changed, and a right-hand side (R), representing the replacement. We use the notation $B \xrightarrow{r} B'$ to denote that B rewrites to B' when $r : L \rightarrow R$ is applied. An example rule is in Fig. 4.2. This rule models the movement of the employee (Em) from R2 to R1. By applying this rule to bigraph B shown in Fig. 4.1a, the part of the bigraph that matches the left-hand side of the rule is changed to match the right-hand side. Fig. 4.3 shows the result of applying rule `leaveR2` (Fig. 4.2) to bigraph B (Fig. 4.1a).

The match is performed based on the structure of the bigraphs, not the links' names. This means the links' names are not considered for matching the left-hand side of the rule, *e.g.* changing link x in rule `leaveR2` (Fig. 4.2) to z does not affect the application of the rule.

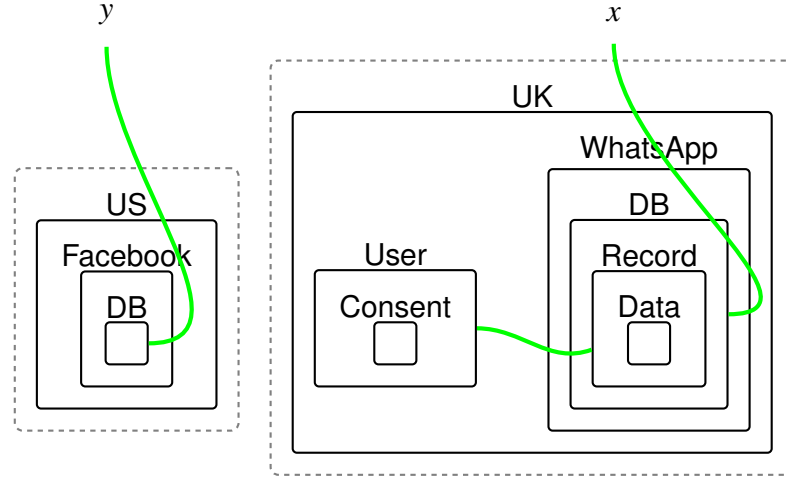


Figure 4.4: The initial state for the WhatsApp example.

4.4 Concrete Example: Modelling a System with BRSs

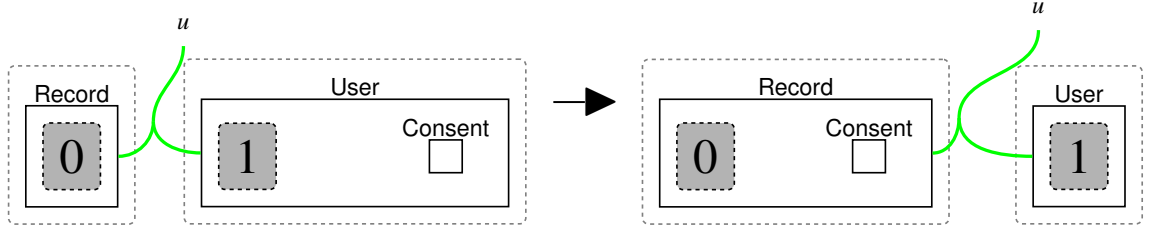
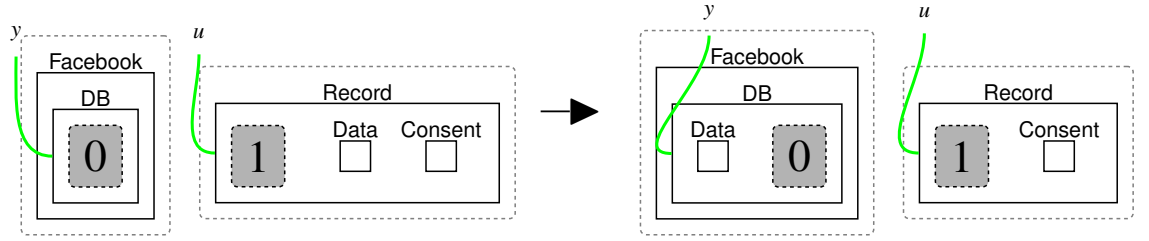
Consider a WhatsApp system that transfers a User’s Data from the UK to Facebook in the US. WhatsApp must obtain the User’s Consent before transferring the data, and the User must also be able to withdraw their Consent at any time. To model this scenario, we first define the initial state of the system as shown in Fig. 4.4. The initial state consists of two perspectives (regions): UK and US¹. The User is linked to the Record to express ownership. Other links (x and y) indicate that the DB is connected to one or more other entities, *e.g.* other databases.

So far, we have represented the state of the system at a given point in time (the initial state). To model its temporal evolution, we also need to define a set of reaction rules. For example, we define rule `obtainCons` (Fig. 4.5) to model the granting of user consent. This rule models the movement of Consent from the User to the DB of WhatsApp, indicating that the User provides their Consent to share their data.

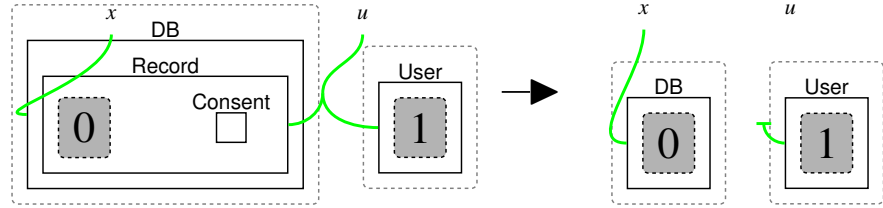
As the system obtains the Consent, we use rule `transD` (Fig. 4.6) to share the data with Facebook. In this rule, we explicitly match on the entity Consent to ensure that the transfer is performed only if the Consent is obtained, *i.e.* the Consent is nested within the user’s Record.

Rule `withd` (Fig. 4.7) models the process of withdrawing Consent. It removes the User’s Record and closes the link connecting the User to the Record, causing the name u to become

¹We refer to perspectives using the names of their top-level entities, *e.g.* the UK perspective and the US perspective.

Figure 4.5: `obtainCons` : obtaining user consent.Figure 4.6: `transD`: transferring data to Facebook.

`idle`². This rule has multiple matches: (1) immediately after `obtainCons` (Fig. 4.5), before any data is shared, and (2) at a later stage following data transfer by `transD` (Fig. 4.6). Therefore, the User is able to withdraw their consent at any time.

Figure 4.7: `withd`: withdrawing user consent.

4.5 Key Features of BRSs

BRSs provide a rigorous formalism for modelling and analysing dynamically evolving spatial and connectivity structures across diverse application domains. In this section, we highlight the key features that enhance their expressiveness and modelling capabilities.

²In diagrammatic representations, idle links appear as unconnected names (*e.g.* the name u on the right-hand side of the rule shown in Fig. 4.7).

4.5.1 Multi-perspective Modelling

As mentioned previously, each parallel region is considered a perspective that is used to separate different concerns. Entities within these regions (perspectives) communicate with each other through cross-region links. The effectiveness of this feature is demonstrated in modelling Savannah [18], a cyber-physical game, where the model comprises four perspectives: *human*, which represents how players manage their spaces and communicate with one another; *physical*, that models the area where players enter and exit; *technology*, which shows how GPS tracks physical spaces; and *computational*, that models the implementation of the game's rules.

Importantly, multi-perspective modelling has a key role in defining our models as it enables us to represent privacy concerns separately from specific system aspects. We will discuss this further in [Chapter 6](#) and [Chapter 5](#).

4.5.2 Spatial Properties

Bigraphs allow modelling spatial relationships between entities as they inherently express spatial properties such as containment. A clear practical application of this feature is the example presented in [Section 4.4](#), where geographical locations are represented as regions. This feature allows us to model GDPR requirements for cross-border data transfers by formally representing and analysing data flow across different jurisdictions, distinguishing between restricted (outside the EU) and unrestricted (within the EU) transfers. We will discuss this further in [Chapter 5](#).

4.5.3 Sorting Scheme

Sorting schemes ensure that bigraphs are well-formed by imposing constraints that prevent undesired or nonsensical models [96], *e.g.* preventing scenarios where US is incorrectly nested within Facebook, or where User is linked to WhatsApp instead of Record³.

We use the notation and grammar presented in [12] to define our sorts. Sorts can take two forms: `sort y` or `sort b = A`. The former specifies the sort of link ports, while the latter indicates that *A* is an atomic entity with sort *b*. If the entity *A* is linked to other entities and

³Currently, no automated tool supports these sorting schemes.

has children, its sort pattern is defined as:

$$\text{sort } b = A\{y \rightarrow z^*\} x + w$$

The sort y represents the sort of the port on A that connects to zero or more ports of sort z . Meanwhile, x and w refer to the sorts of A 's children, *i.e.* A has children of sort x *or* w . [Table 4.2](#) describes through some examples the sort notations used in this thesis.

For instance, we define sorts for the example presented in [Section 4.4](#) as:

```
sort a, sort b, sort y
sort d = Data
sort c = Consent
sort r = Record{y → b} d* + c
sort u = User{b → y} c + 1
sort db = DB{a → a*} r* × r
sort w = WhatsApp db
sort f = Facebook db
sort l = UK u × w | US f
```

We assign sorts a , b , and y to the ports on DB , $User$, and $Record$, respectively. The entities $Data$ and $Consent$ are atomic. $Data$ has sort d , while $Consent$ has sort c .

$Record$ may contain zero or more children of sort d ($Data$) or at most one child of sort c ($Consent$). Since a $Record$ must always be linked with a $User$, its port (of sort y) must be linked to a port of sort b , which is the port on $User$. Conversely, the port on $User$ must be linked to the port on $Record$. $User$ may either be empty or contain a single child of sort c ($Consent$).

The entity DB must always include at least one child of sort r ($Record$). Since DB can be linked to zero or more other DB entities, both its input and output ports share the same sort (a).

$WhatsApp$ and $Facebook$ must always contain exactly one entity of sort db . The sort l represents the sort of UK and US . UK must include children of sort w ($WhatsApp$) and u ($User$). Conversely, US must have only one child of sort f ($Facebook$).

Table 4.2: Description of sort notations.

Notation	Example	Description
s^*	$A s^*$	The entity A has <i>zero</i> or more children of sort s .
$s \times s^*$	$A s \times s^*$	The entity A has at least <i>one</i> child of sort s .
$s \times i$	$A\{a \rightarrow s \times i\}$	The entity A has a port of sort a , which is connected to ports of sorts s AND i .
$s + i$	$A\{a \rightarrow s + i\}$	The entity A has a port of sort a , which is connected to a port of sort s OR i .
$s + 1$	$A s + 1$	The entity A has children of sort s OR it does not have any (<i>i.e.</i> it is empty).

4.5.4 Priority Classes

Generally, reaction rules are applied whenever their left-hand side is matched, which can lead to a rule being applied in unintended contexts. Priority classes play a key role in preventing such cases as they enforce the ordering of rules, where each class representing a set of rules. Suppose we have two classes: P_1 and P_2 , with P_1 having a lower priority than P_2 . In this case, a rule in class P_1 can be applied only if no rules in class P_2 can be applied.

We use the example presented in [Section 4.4](#), to show how the priority classes can control the application of rules. In this example, we define two priority classes: $\{\text{obtainCons}\}$ has higher priority than $\{\text{transD}, \text{withd}\}$. In this case, rules `transD` and `withd` are triggered only if rule `obtainCons` cannot be applied.

Now suppose we define three priority classes for our reaction rules as follows: $\{\text{obtainCons}\}$ has the highest priority, $\{\text{withd}\}$ has medium priority, and $\{\text{transD}\}$ has the lowest priority. Under this priority scheme, rule `transD` will never be applied. The reason is that rule `obtainCons` (which provides Consent) is always applied first. Immediately afterward, rule `withd` (which withdraws) is triggered. Consequently, once `withd` is applied, there is no longer any configuration in the system that matches the left-hand side of `transD`. Since `transD` requires a specific pattern (*i.e.* the presence of Consent) that is now absent, `transD` can never be matched and therefore is never executed.

4.5.5 Paramaterised Entities and Rules

Bigraphs also support parameterised entities and rules. For example, we can define entities like $\text{User}(x)$ where x is drawn from a set of integers or strings, *e.g.* $x \in \{\text{ID}, \text{Name}\}$. This is equivalent to defining a set of entities, *e.g.* User_ID , User_Name , $\dots$. Similarly, rules can also be parameterised. For instance, we might write $\text{transD}(x)$, where x can be either $\{\text{ID}, \text{Name}\}$.

This feature allows us to define families of entities and rules, where each entity or rule in the family is an instance corresponding to a specific value of x .

4.5.6 Instantiation Maps

Instantiation maps are defined for rules to allow copying, swapping, or deletion of sites when a rule is applied. For example, to swap the content of the DB in Facebook and WhatsApp, we define an instantiation map as shown in [Fig. 4.8a](#), where:

- Site 1 on the right-hand side is mapped to site 1 on the left-hand side.
- Site 0 on the right-hand side is mapped to site 0 on the left-hand side.

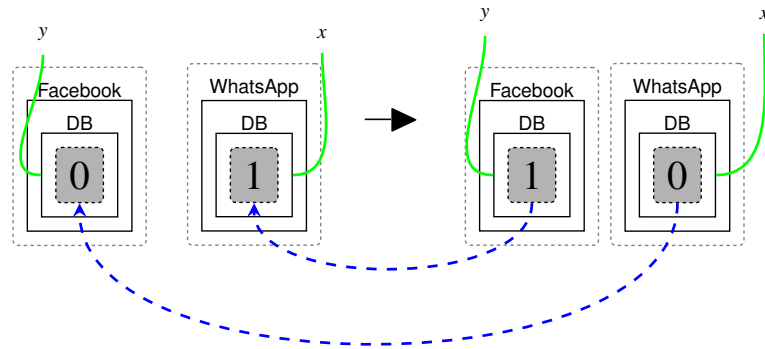
Likewise, [Fig. 4.8b](#) demonstrates how instantiation maps can copy the content of sites. Specifically, the content of site 1 on the left-hand side (*i.e.*, the DB of WhatsApp) is replicated to site 1 on the right-hand side. In contrast, instantiation maps can also delete sites as shown in [Fig. 4.8c](#). Here, site 0 appears on the left but not on the right, thus it is removed from the resulting bigraph.

In [Fig. 4.8](#), we use blue arrows as a visual aid to illustrate these mappings. However, in the rest of this thesis, we number sites on the left, and their positions after rule application are numbered right-hand sites, without using blue arrows.

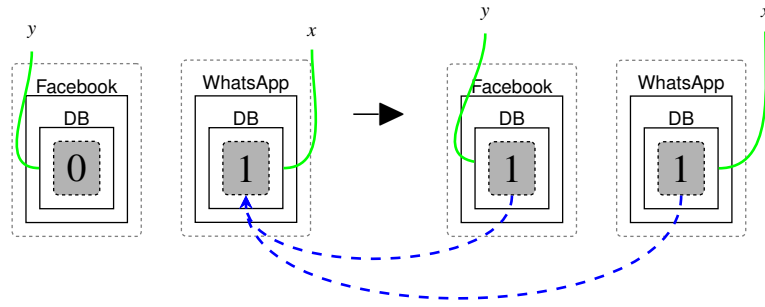
4.5.7 Instantaneous rules

Instantaneous rules are hidden rules that are executed but they do not explicitly appear in the generated transition system⁴. They are placed in instantaneous priority classes, represented

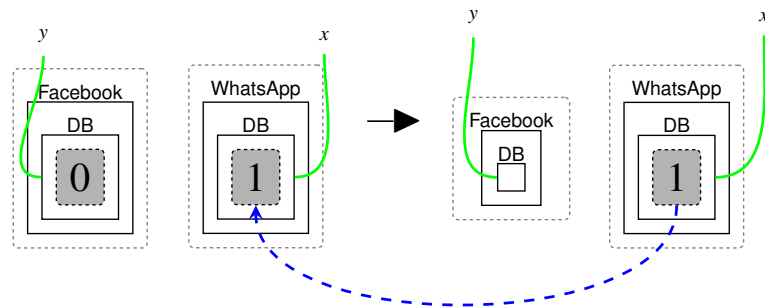
⁴A transition system consists of states (bigraphs) that describe the system configurations. Each state is a result of applying a rule (transition).



(a) swap: swapping Site 0 and Site 1.



(b) copy: copying Site 1.



(c) delete: deleting Site 0.

Figure 4.8: Examples of instantiation maps.

by parentheses: $(r1, r2)$. Importantly, instantaneous rules share execution priority with other rules from standard classes.

Instantaneous rules are useful for minimising the number of states in transition systems, especially in cases of state explosion, where the state space grows significantly. They can also act as garbage collection mechanisms, allowing for the removal of entities that are no longer needed in the model. For example, suppose that rule $r1$ needs to be executed n times. In this case we can define instantaneous rule(s) to reach the base case, where execution should stop by removing entities (*e.g.* tags) to prevent further matches of the left-hand side of the rule.

4.5.8 Conditions

We use conditional bigraphs⁵ [10], which allow rules to only apply in specific contexts. These are written after a rule in angle-bracket notation. For example, if $\langle -, \text{Consent}, \downarrow \rangle$ where $-$ indicates a negative condition (should not exist), Consent entity is the bigraph⁶ we want to disallow, and $\downarrow$ indicates we should disallow this inside the sites. We also allow positive $+$ (must exist), and contextual conditions $\uparrow$ (anywhere other than the sites/match).

Returning to the example in Section 4.4, consider a case where the User does not give their Consent to share data with Facebook. In this case, the transfer should be blocked. Rule `prevent` (Fig. 4.9) is defined to block the transfer if the Consent entity does not exist in either site 0 or site 1, as the condition checks **both** sites. This means that if Consent is present in site 0—even unintentionally—the transfer will still be allowed.

We should note that priority classes also restrict the application of rules, as discussed in Section 4.5.4. The difference between them and conditions is that priority classes determine the relationships among the entire set of rules, whereas conditions specify the purpose of a single rule.

4.5.9 Bigraph Patterns

To allow labelling of the transition system states, used within the logical specifications, *bigraph patterns* (that act like left-hand side matches in rewriting) are used. Often these are used to simply check for the presence of a specific entity in a state. We can also use them to

⁵We use bigraphs to mean conditional bigraphs throughout.

⁶In general this can be any bigraph, but is often a single entity.

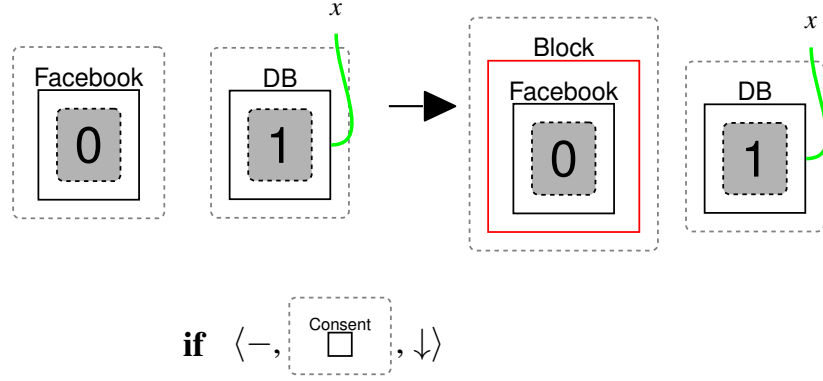
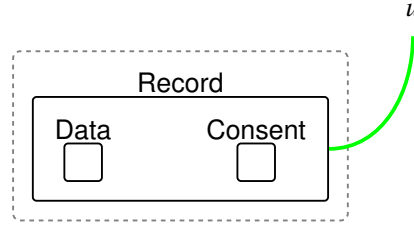
Figure 4.9: `prevent`: preventing data transfer to Facebook.

Figure 4.10: Example bigraph pattern.

detect design flaws by searching for specific conditions within the rules. Fig. 4.10 shows an example of a bigraph pattern, where each state contains this pattern is labelled, for example, with `const`.

4.6 Bigraph Evaluator and Rewriting

To analyse a BRSs we use Bigraph Evaluator and Rewriting (BigraphER) [117]. BigraphER is an open-source tool for modelling, rewriting, and visualising bigraphs.

Once we have a formal model, we utilise it to perform *model checking verification*, *i.e.* checking whether the system adheres to a specific regulation.

First, we create a transition system representing all possible updates the system can make. Given an initial state of the system, the transition system is *automatically* generated using BigraphER, with bigraphs representing states, and reaction rule applications representing transitions. For example, to generate the transition system of the example presented in Section 4.4, we define the model using BigraphER syntax. The model code includes declarations

of the controls used in the model, reaction rule definitions, bigraph patterns to be checked, and the definition of the initial state. The following illustrates the BigraphER syntax used to define these elements:

1. Declaring the model's controls, including their number of ports and types (atomic or non-atomic):

```

1 ctrl WhatsApp = 0;
2 ctrl Facebook = 0;
3 ctrl Block = 0;
4 ctrl UK = 0;
5 ctrl US = 0;
6 ctrl DB = 1;
7 ctrl Record = 1 ;
8 ctrl User = 1;
9 atomic ctrl Consent = 0;
10 atomic ctrl Data = 0;

```

2. Defining reaction rules:

```

1 react obtainCons =
2   User{u}.(id | Consent) || Record{u}.id
3   -[1]->
4   User{u}.id || Record{u}.(id | Consent);
5
6 react transD =
7   Record{u}.(id | Data | Consent) || Facebook .DB{y}.id
8   -[1]->
9   Record{u}.(id | Consent) || Facebook . DB{y}.(id | Data);
10
11 react withd=
12   User{u}. id || DB{x}.Record{u}.(id | Consent)
13   -[1]->
14   /u User{u}.id || DB{x}.id || {u};
15
16 # Conditional rule
17 react prevent =
18   DB{x}.id || Facebook.id
19   -[1]->
20   DB{x}.id || Block.Facebook.id
21   if ! Consent in param ;
22
23 # Instantiation map
24 react chgeDec=
25   Block.Facebook.(DB{y}.id) || User{u}.id || WhatsApp.(DB{x}.id)
26   -[1]->
27   Facebook.(DB{y}.1) || User{u}. (id | Consent) || WhatsApp.(DB{x}.Record{u}.Data)
28   @[1];

```

3. Specifying predicates (bigraph patterns):

```
1 big with = /u User{u}.id || DB{x}.id;
2 big prev = DB{x}.id || Block.Facebook.id;
```

4. Defining the initial state:

```
1 big a1 = UK. (User{u}.(Consent) | WhatsApp.(DB{x}.Record{u}.Data)) ;
2 big a2 = US.(Facebook.DB{y}.1);
3 big s0 = /u (a1 || a2 );
```

5. BRS's declaration:

```
1 begin pbrs
2   init s0;
3   rules = [
4     {chgDec},
5     {obtainCons},
6     {transD, withd},
7     {prevent}
8   ];
9   preds = {with, prev};
10 end
```

Fig. 4.11 shows the generated transition system for the WhatsApp example after running BigraphER. This transition system is relatively simple, as it contains only a few states. However, in real-world systems, the number of states can be considerably larger. Manually inspecting these states to verify properties is challenging due to the large state space.

To manage this complexity, we use the PRISM model checker [85], which is natively supported by BigraphER. For example, we can define the property we aim to check using CTL syntax as follows:

$$\mathbf{A}[\mathbf{G}(\text{with} \implies (\mathbf{F}\text{prev}))] \quad (4.1)$$

This property ensures that if consent is withdrawn (with), data transfer is prevented (prev). This is achieved using the PRISM model checker, which utilises CTL syntax to parse the transition system in Fig. 4.11 and verify whether this property holds or is violated. We will discuss this in depth in Chapter 7.

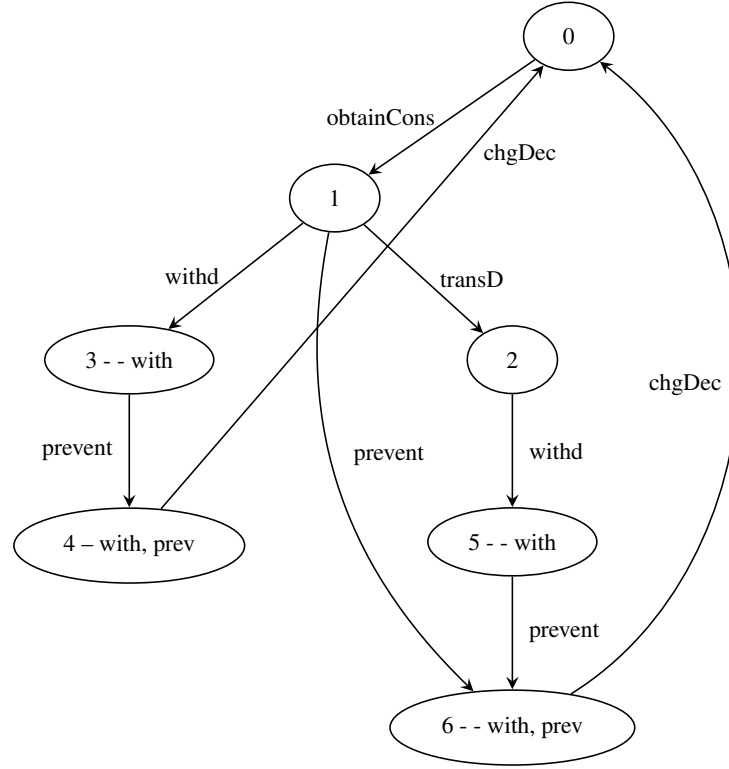


Figure 4.11: The transition system generated for the WhatsApp example using BigraphER. Rule `chgDec` allows users to change their decisions, *i.e.* providing consent again after withdrawing it.

4.7 Summary

In summary, this chapter provides a basic introduction to bigraph reactive systems (BRSs). It starts by describing bigraphs (Section 4.1), explaining how they capture both the place graph and the link graph (Section 4.2). It then shows how reaction rules turn these basic bigraphs into BRSs that can represent changes over time (Section 4.3). We also provide a simple example that demonstrates these ideas in action (Section 4.4), followed by a discussion of the main features that make BRSs useful (Section 4.5). Finally, the chapter introduces BigraphER, a tool that helps users define, analyse, and visualise bigraph models (Section 4.6).

Chapter 5

Privacy Compliance in Cross-border Data Transfers

Due to the advancement of information technologies, users' data is collected and shared among several organisations worldwide. As mentioned in [Chapter 2](#), such transfers may pose risks to user privacy and national security, particularly if the destination country lacks adequate protection measures. The GDPR imposes several requirements to regulate the international transfer of data and ensure that the recipient country provides an adequate level of data protection.

To assist organisations in ensuring their systems comply with these requirements and can demonstrate compliance to supervisory authorities, this chapter presents a bigraph framework for modelling the GDPR requirements for cross-border data transfers. The privacy aspects of our framework are predefined while system-specific aspects have to be specified for each application. The framework captures data transfers based on *adequacy decisions and appropriate safeguards (Standard Contractual Clauses (SCCs) and certification mechanisms)* that are introduced in [Chapter 2](#).

[Section 5.1](#) and [Section 5.2](#) present the privacy model. In [Section 5.3](#), we introduce an example that models WhatsApp's privacy policies to demonstrate how the privacy framework is integrated with a specific system. We then discuss key considerations regarding the model in [Section 5.4](#), followed by [Section 5.5](#), which summarises the chapter.

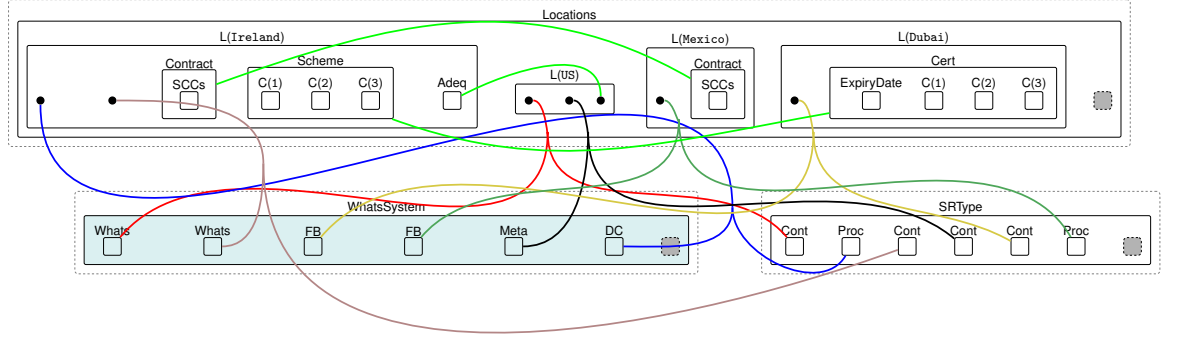


Figure 5.1: A partial initial state for the WhatsApp example. System-specific entities are shaded in teal. The predefined privacy model appears within the uncoloured regions. Sites are entities abstracted away for simplicity (but they are shown in Fig. 5.11). Solid black bullet points are entities (referred to as pointers) that allow end-users to assign an arbitrary number of links. Each system entity is linked to its corresponding type in SRTYPE and its location in Locations.

5.1 Modelling Privacy Visually

We now define our privacy model. A key goal is to separate the user-defined systems, *e.g.* WhatsApp’s system components, from the specific privacy entities in order to reuse as much of the model as possible in future scenarios. To enable this, we make use of multi-perspective modelling, where specific aspects of the system appear in their own regions. This approach has been used to good effect in [19], [112], [118].

We describe our approach by applying it to an example based on WhatsApp’s privacy policies. WhatsApp has two *data controllers* [134], one is located in Ireland to provide services to users in the EU countries, while the other is located in the US to serve users from other countries. These two controllers need to share users’ data with the parent company Meta, its data centres, and Facebook’s branches worldwide. A partial model of the example is shown in Fig. 5.1, while the other part of the model is presented in Section 5.3. The model consists of three perspectives: WhatsSystem, SRTYPE (for sender and receiver), and Locations¹. The specific system is modelled in the WhatsSystem perspective. It consists

¹The perspectives are referred to by the names of their top-level entities, *e.g.* WhatsSystem perspective, SRTYPE perspective, *etc.*

of the data controllers in the US and Ireland (Whats), the Facebook branches in Dubai and Mexico (FB), Meta company in the US, and the data centre in Ireland (DC). These entities are linked to their types in SRTYPE perspective, *e.g.* data controller (Cont) or processor (Proc), and to their locations in Locations perspective as discussed in [Section 5.1.1](#) and [Section 5.1.2](#), respectively.

5.1.1 Specifying Sender and Receiver Types

SRTYPE perspective is used to specify the types of each system-specific sender and receiver by means of linking, as shown in [Fig. 5.1](#). This perspective can only contain entities of types *data controller* (Cont) and *processor* (Proc). According to the GDPR, the *data controller* determines the privacy policies and the purposes of processing users' data, whereas the processor processes the data based on the controller's instructions [56]. This perspective can easily be extended to support other types, *e.g.* sub-processor, joint controller *etc.* In [Fig. 5.1](#), the system's entities that have the controller role (Whats, FB in Dubai, and Meta) are linked to their own Cont. Similarly, the system's processors (DC and FB in Mexico) are linked to their own Proc.

5.1.2 Specifying Sender and Receiver Locations

After assigning types to the sender and receiver entities, we need to specify their locations in the Locations perspective. We do so by linking them to an entity of type P (called *pointer*) nested within a location $L(x)$. This is shown in [Fig. 5.1](#) where, for instance, DC is linked to a P within $L(\text{Ireland})$. This approach allows for a great modelling flexibility as the number of entities in each location does not need to be specified and fixed beforehand. Also, by toggling between types P, P' and Ps, it is easy to define rules to check the adequacy decision and the safeguards when the sender and receiver are in different regions as we will explain in [Section 5.2.1](#).

[Table 5.1](#) describes the entities of this perspective. The parameterised entity $L(x)$ allows end-users to add new countries, *e.g.* $L(\text{Japan})$. The entity SCCs is nested within Contract. The Contract and Cert are nested within $L(x)$ to indicate that the safeguard provided by the organisation in country x is either the SCCs or certification, respectively. Consider the example in [Fig. 5.1](#). The entity $L(\text{Mexico})$ contains SCCs, which is nested within the

Table 5.1: Locations perspective entities.

Entity	Description
$L(x)$	Locations as parameterised entities where x is a country's name, <i>e.g.</i> UK
Adeq	Linked to adequate countries
Contract	The contract between the sender and the receiver
SCCs	The Standard Contractual Clauses
Scheme	The scheme specified by the GDPR, which the certification should comply with
ExpiryDate	The expiry date of the certification
$C(x)$	Criteria that should be checked to determine the valid certification. Parameter x is a criterion's identifier, <i>e.g.</i> 1, 2, Transparency, Right-to-access <i>etc.</i>
Cert	The certification provided by the receiver
P, P', Ps	Pointers for each entities within a location. Shown as solid black, blue and yellow bullets, respectively

Contract, meaning that the safeguard provided by Facebook in Mexico is the SCCs. Cert is nested within $L(\text{Dubai})$ to indicate that Facebook's branch in Dubai uses the certification safeguard. We do not allow the case when a company provides both safeguards as this is not included in the GDPR. Since, Ireland is the sender country in our example, it includes Adeq, Contract, and Scheme. The scheme contains three criteria, denoted as $C(x)$, where $x \in \{1, 2, 3\}$.

5.2 Checking privacy requirements

The modelling strategy we have described so far allows to only represent the status of a system at a given time. To model the temporal evolution of a system, we have also to define a set of reaction rules. In this section, we introduce predefined reaction rules encoding the processes of checking the GDPR requirements for cross-border data transfers.

5.2.1 Checking Regions

To check if data transfers are restricted, we need to check the region of the sender and the receiver. If both are located in the same region, the transfer is considered safe. Otherwise, it is a restricted transfer, and we need to check the adequacy decision and the safeguards.

The mechanism for checking the region is to tag the pointers linked to the sender and

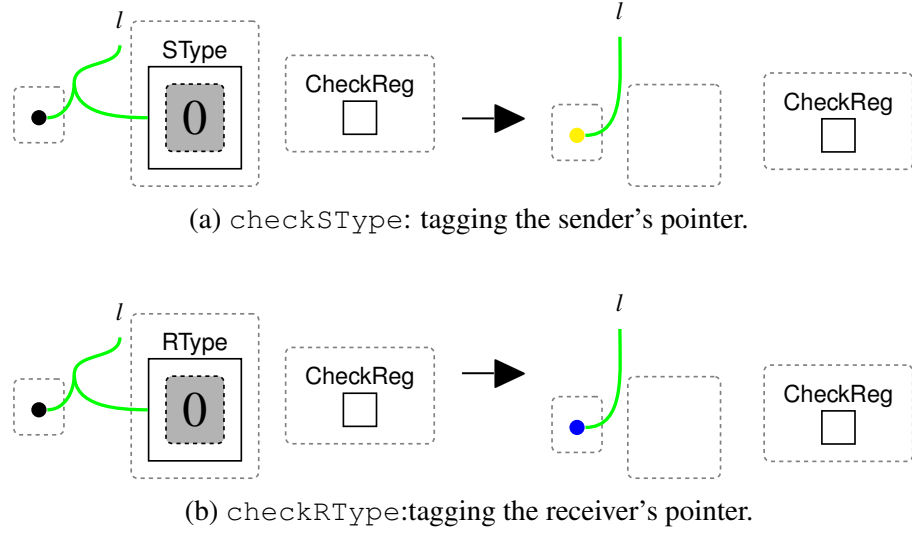


Figure 5.2: The process of checking the sender's and receiver's regions by tagging the pointers linked to their specified types.

recipient **type**. As these pointers are nested within the entity $L(x)$, tagging them enables us to specify if the sender and receiver are in the same or different regions as explained in [Section 5.2.2](#).

Rule *checkSType* ([Fig. 5.2a](#)) specifies the region of the sender, while rule *checkRType* ([Fig. 5.2b](#)) specifies the receiver's location. The entities *SType*, *RType*, and *CheckReg* are generated via system rule(s) in the system perspective, as outlined in [Section 5.3](#). *SType* and *RType* represent the types of the sender and receiver, respectively (*e.g.* site 0 can be either a Cont or Proc), and they are linked to the corresponding location of each. The entity *CheckReg* is generated by a system rule to trigger the privacy model.

Consider the controller Whats, located in Ireland, which needs to transfer data to the processor DC, also located in Ireland. To model this, we define system rules that specify the type of the sender (Whats) by generating *SType* around Cont and linking it to the hyper-edge that connects Cont with its pointer in $L(\text{Ireland})$. Similarly, we specify the type for DC, generating *RType* around Proc, as DC functions as a processor. Additionally, we generate the entity *CheckReg* to initiate the checking process.

As we generate the *SType* and *RType* entities and link them to the locations of the specified types, we apply rule *checkSType* and *checkRType* ([Fig. 5.2](#)) to tag the pointers linked

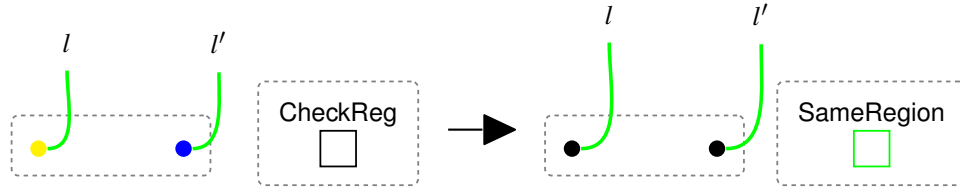


Figure 5.3: `sameReg`: the sender and the receiver are in the same region.

to `SType` and `RType`, represented by yellow and blue bullets, respectively.

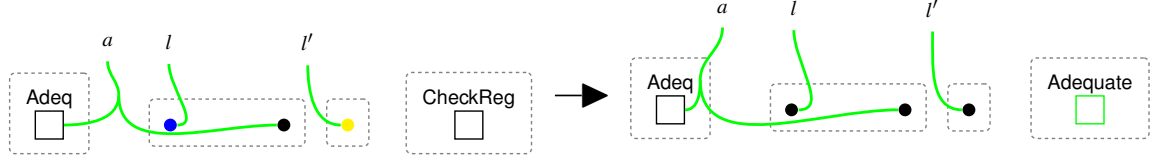
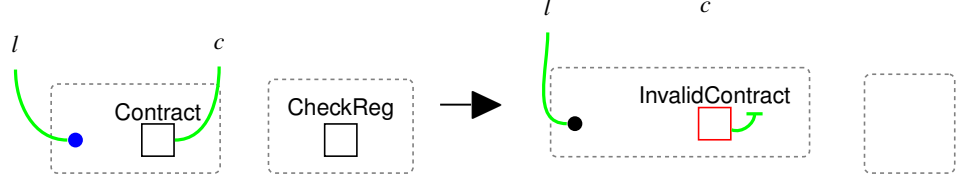
5.2.2 Entities are in the Same Region

If the tagged pointers (the yellow and blue pointers) are nested within the same region, then the transfer is safe. For instance, the tagged pointers in [Section 5.2.1](#) are nested within `L(Ireland)`, so we can verify that the sender and the receiver are both in Ireland. The rule that models the result of checking the regions is `sameReg` ([Fig. 5.3](#)). The entity `CheckReg` is replaced with `SameRegion` to indicate the result of checking the region, *i.e.* the sender and the recipient are in the same region, and terminate the checking process. We also untag the pointers to recheck the regions if needed.

5.2.3 Checking Adequacy

Suppose the data sender is Whats in Ireland, and the receiver is Meta in the US. By checking their regions as explained in [Section 5.2.1](#), we tag the pointers that are nested within the entities `L(Ireland)` and `L(US)`, respectively. In such a case, the transfer is restricted because the tagged pointers are nested within different regions, so we must check the adequacy.

Rule `checkingAdeq` in [Fig. 5.4](#) checks the adequacy requirement. This rule checks if there is a pointer in the relevant country linked to `Adeq`. If such a pointer exists, then the country is adequate. The entity `Adequate` is generated to show the result of checking the adequacy requirement. The sender's and the receiver's pointers (the yellow and blue bullets, respectively) are untagged to allow reuse of the rule if needed. Importantly, failing to find the match of the left-hand side of this rule means the country is inadequate, so we should start checking the safeguards as shown in [Section 5.2.4](#) and [Section 5.2.5](#) which have higher priority than rule `checkingAdeq` ([Fig. 5.4](#)).

Figure 5.4: `checkingAdeq`: checking the adequacy of the data importer country.Figure 5.5: `checkingSCCs`: tagging the contract as invalid if it does not include the SCCs.

5.2.4 Checking SCCs

As mentioned previously, we must check the safeguards when rule `checkingAdeq` (Fig. 5.4) is not applied. In case the provided safeguard is SCCs, we use rule `checkingSCCs` in Fig. 5.5 to check whether the receiver country incorporates the SCCs in their Contract².

For example, the Contract of FB in Mexico includes SCCs and is linked to the entity Contract nested within L(Ireland), as shown in Fig. 5.1, indicating that Mexico agrees to adhere to the SCCs specified by the sender, thereby confirming the validity of the Contract.

In contrast, if the Contract in the receiver country does not include SCCs, we use rule `checkingSCCs` (Fig. 5.5) to tag the Contract as `InvalidContract`. This rule also results in the closure of the link connected to the entity Contract in Ireland, *i.e.* link *c*.

5.2.5 Checking Certification

If the provided safeguard is a certification (Cert), it must be checked to prove that it meets the Scheme specified by the GDPR. To do so, we use rule `tagCriteria(x)` shown in Fig. 5.6 to check each criterion (*x*). Suppose that the Scheme has three criteria: *C*(1), *C*(2) and *C*(3). We replace the parameter *x* in rule `tagCriteria(x)` (Fig. 5.6) with the number of criterion, *e.g.* `tagCriteria(1)`, `tagCriteria(2)` *etc.*

Each rule checks the existence of the criterion in the Scheme and the provided Cert, *e.g.*

²We do not model the content of SCCs as they are pre-approved contractual clauses provided by the European Commission, eliminating the need to verify their validity.

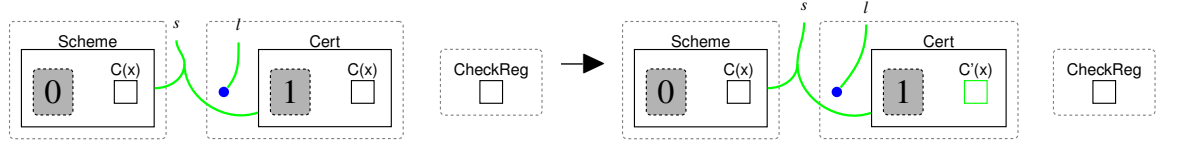


Figure 5.6: $\text{tagCriteria}(x)$: tagging each criterion that must be satisfied.

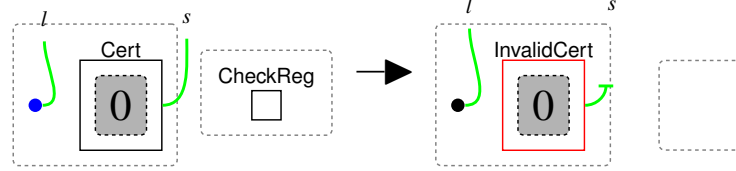


Figure 5.7: tagInvalidCert : tagging the certification as invalid if at least one criterion is not satisfied.

checking that $C(1)$ is nested within the Scheme and the Cert, by tagging the existing criterion (the green entity). Although the GDPR defines more than three criteria, we model only three as the modeller can extend the model by adding x number of criteria, *e.g.* $C(4)$, $C(5) \dots C(x)$.

If **one** of the defined family rules is not applied, the Cert does not meet the criteria. For example, if rule $\text{tagCriteria}(1)$ is not applied but rule $\text{tagCriteria}(2)$ is applied, it means the Cert does not fulfil the first criterion ($C(1)$). In such a case, we use rule tagInvalidCert (Fig. 5.7) to tag the Cert as invalid (InvalidCert) and close link s that is connected to the Scheme. We omit CheckReg to end the checking process.

5.2.6 Result of Checking Certification

Rule checkCertResult (Fig. 5.8) shows the result of checking the criteria. If the Cert meets the criteria, *i.e.* $C(1)$, $C(2)$ and $C(3)$ are tagged, the Cert is tagged as CompliantCert.

Based on the GDPR, the certification (Cert) is valid for only **three years** [66]. This requires checking its validation date as well [40]. Rule checkCertResult (Fig. 5.8) initialises the process of checking the expiration by replacing the entity CheckReg with CheckExp and specifying the current date (CurrentDate) to compare it with the expiry date of the Cert as shown in Section 5.2.7

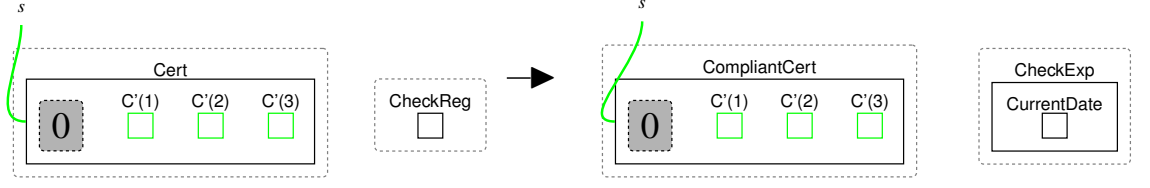


Figure 5.8: `checkCertResult`: result of checking the certification and initialising the process of checking its expiry date.

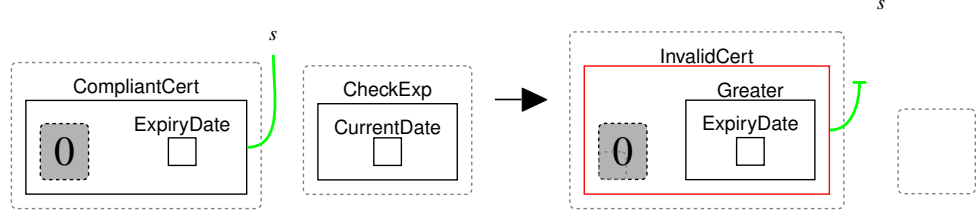


Figure 5.9: `expiredDate`: tagging the expired certification.

5.2.7 Checking the Expiry Date

As shown in Fig. 5.9, we can check the validation date of the `CompliantCert` by comparing the current date (`CurrentDate`) with the expiry date (`ExpiryDate`). If the `ExpiryDate` is **greater than** (`Greater`) the `CurrentDate`, it means the `CompliantCert` is expired and cannot be used as a safeguard. Otherwise, it is valid and can be used to transfer the data³.

Rule `expiredDate` (Fig. 5.9) tags the `CompliantCert` as `InvalidCert` and removes link `s` connecting the `CompliantCert` to the Scheme if the `ExpiryDate` is `Greater` than the `CurrentDate`. The entities `CheckExp` and `CurrentDate` are omitted to terminate the checking process.

5.2.8 Withdrawing Certification

Based on the GDPR, the provider of the compliant certification (`CompliantCert`) has the right to withdraw it [40]. Rule `processWithdReq` (Fig. 5.10) models the provider's withdrawal request by closing link `c` connecting the `CompliantCert` to the Scheme and replacing `CompliantCert` with `WithdrawnCert`. The entity `WithdReq` is generated through a system rule as explained in Section 5.3.

³To generalise the model, we do not use mathematical computation to perform the comparison. However, we model the two cases: when the certification is expired and when it is valid. See the full model [5].

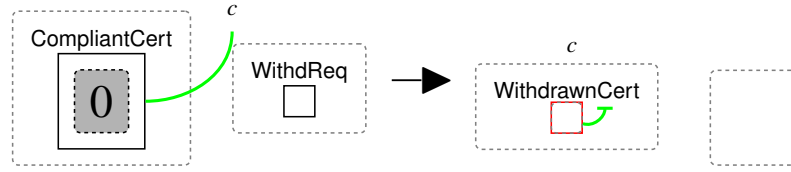


Figure 5.10: `processWithdReq`: processing the withdrawal request for the certification.

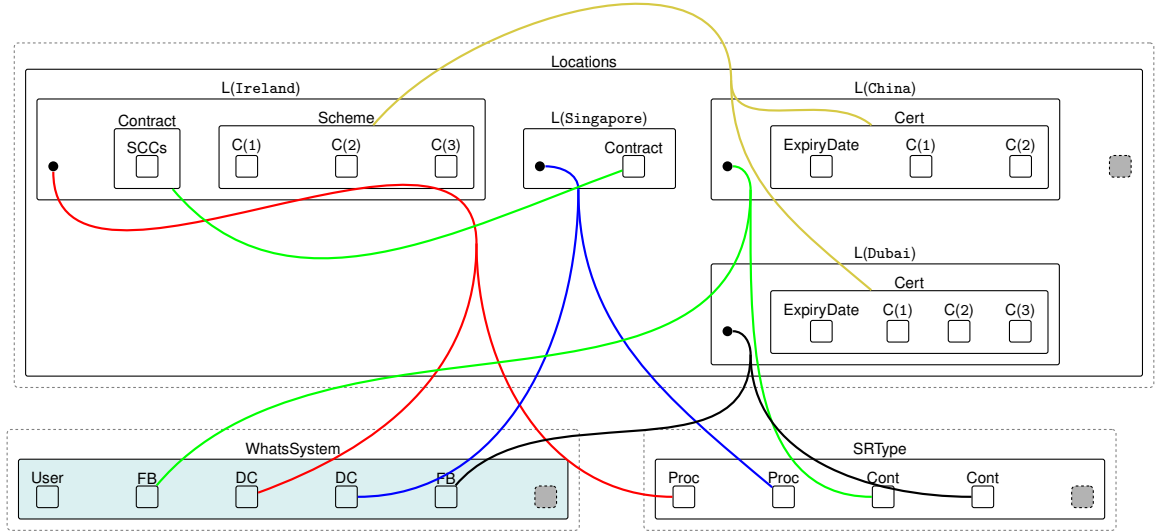


Figure 5.11: A Partial initial state for the example of transferring data to the data centre in Singapore and to Facebook in China and Dubai.

5.3 Integration of Privacy Models and Specific System

We introduce an example to show the privacy model's usability and how we can merge it with the specific system. Suppose that the data centre of WhatsApp in Ireland (DC) needs to transfer users' data to WhatsApp's data centre in Singapore (DC) and to the Facebook branches (FB) in China and Dubai. Fig. 5.11 shows the initial state for this example. We consider the DC in Singapore to be a processor (linked to Proc) that has a Contract with the DC in Ireland. Both FB entities are considered data controllers (linked to their respective Cont), and the safeguard they provide is a certification (Cert).

We suppose that we have one User and is located in Ireland. Rule `createAcc` (Fig. 5.12) models the step of creating a WhatsApp account by User (AccCreated). When the account is created, the user creates a status (CreateStatus) as shown in Fig. 5.13. We assume that the

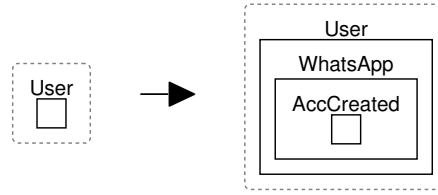


Figure 5.12: `createAcc`: the system's user starts using WhatsApp to create an account.

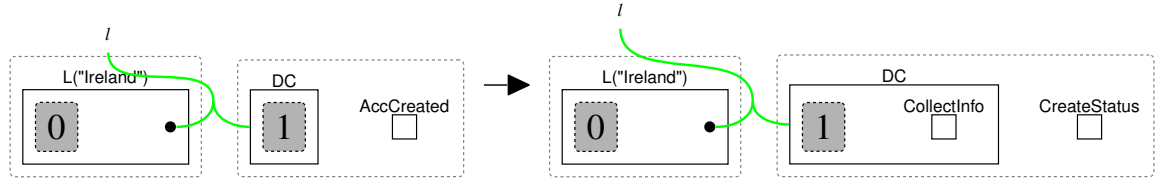


Figure 5.13: `createStatus`: the user creates a status.

DC in Ireland collects information about the user's status, *e.g.* status content type, location information, information about the device used to post the status *etc.* We model that by generating the entity `CollectInfo`. We do not specify exactly what the collected data is as our model focuses on checking the adequacy decision and the safeguards regardless of the type of transferred data. Once DC collects data, it initialises the transferring process as shown in Fig. 5.14 by replacing the entity `CollectInfo` with `StartTransfer`.

Now, we need to trigger the privacy model to check the sender and receiver regions and their safeguards. To do so, we should specify the type of the sender (`SType`) and the receivers (`RType`) and generate the entity `CheckReg` via system rules. Rule `dcIrType` (Fig. 5.15) specifies the type of DC in Ireland. As DC is the sender and is linked to `Proc`, `SType` is generated around `Proc` to indicate that the sender is a processor. We also link `SType` to the sender's pointer in `L(Ireland)`. `StartTransfer` is replaced with `SpecifyReceivers` to specify the receivers types before transferring the data.

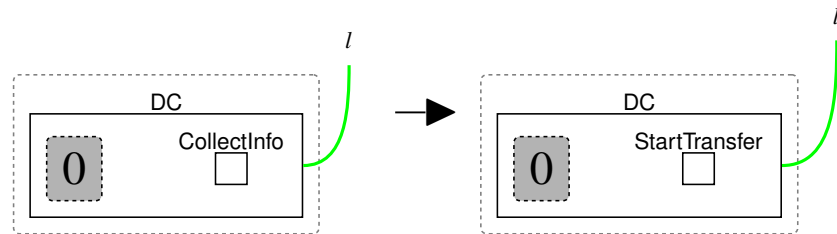
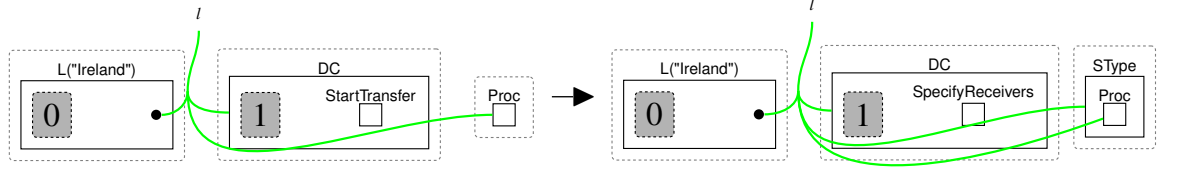
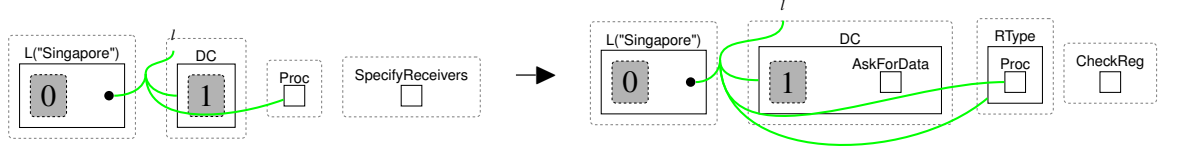
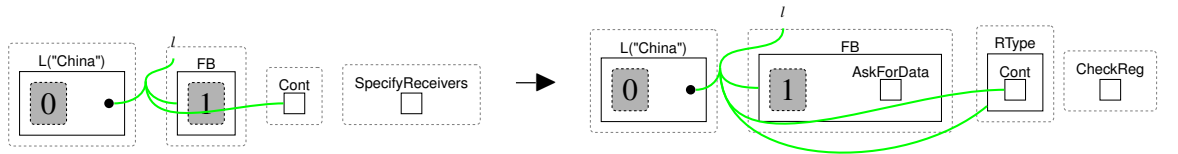


Figure 5.14: `initiTrnas`: DC initialises the transferring process.

Figure 5.15: `dcIrType`: specifying the type of the sender (DC in Ireland).Figure 5.16: `dcSingType`: specifying the type of the receiver (DC in Singapore).

Rules `dcSingType` (Fig. 5.16), `facebookCType` (Fig. 5.17) and `facebookDType` (Fig. 5.18) specify the type of the receivers in Singapore (DC), China and Dubai (FB), respectively. Since DC is linked to Proc, the RType is generated around Proc and linked to DC's location. The entity `AskForData` represents the receiver's request for the data. FB in China and Dubai are both data controllers, so RType is generated around Cont and linked to China and Dubai. We also replace `SpecifyReceivers` with `CheckReg`.

After specifying the sender and receivers types and generating `CheckReg`, we can start checking the regions by tagging the pointers linked to the specified types. We specify the region of the sender DC by using rule `checkSType` (Fig. 5.2a). We also use rule `checkRType` (Fig. 5.2b) to check the regions of the receivers DC and FB. By applying these rules, the pointers that will be tagged are the pointers that are located in `L(Dubai)`, `L(China)`, `L(Singapore)` and `L(Ireland)` as shown in Fig. 5.19. We can observe that the tagged pointers are located in different countries, *i.e.* the receivers' pointers are not located in Ireland alongside the sender's pointer. Therefore, the transfer is restricted.

Figure 5.17: `facebookCType`: specifying the type of FB in China.

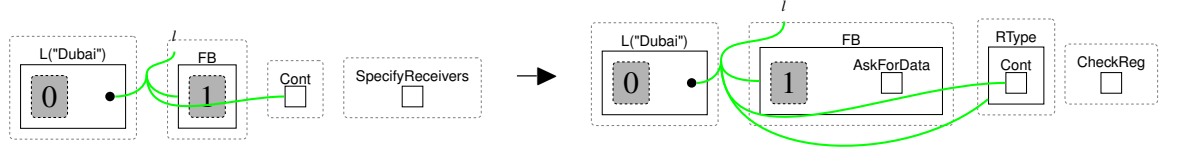


Figure 5.18: facebookDType: specifying the type of FB in Dubai.

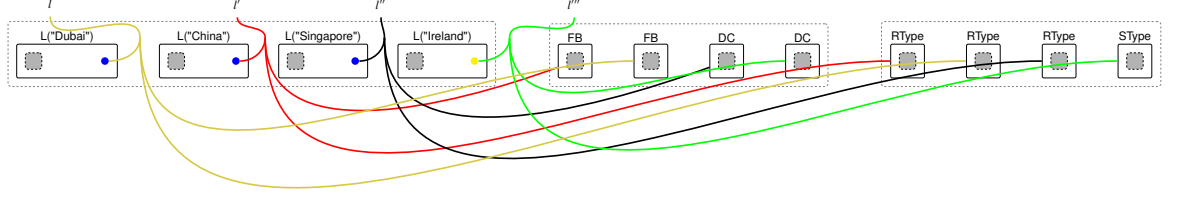


Figure 5.19: The tagged pointers after specifying DC in Ireland and Singapore, and FB in Dubai and China.

Based on Fig. 5.11, Singapore, China and Dubai, are not specified as adequate countries, *i.e.* no pointers in Singapore, China or Dubai linked to Adeq, as is the case with the US in Fig. 5.1, meaning that the safeguards must be checked. The entity Contract that is nested within L(Singapore) does not include SCCs. This indicates that DC in Singapore does not incorporate the SCCs in their contract, *e.g.* does not agree to adhere to the SCCs. Rule `checkingSCCs` (Fig. 5.5) is applied to tag the Contract as `InvalidContract` and close its link connected to the entity Contract in Ireland. The entity `InvalidContract` is used as a flag to prevent the transfer and block the DC in Singapore through the system rule `preventSing` (Fig. 5.20).

FB in China uses the certification mechanism that has only two criteria (C(1) and C(2)) as shown in Fig. 5.11. By applying rule `tagCriteria(x)` (Fig. 5.6), only two criteria are tagged. In this case, rule `tagInvalidCert` (Fig. 5.7) tags the Cert as `InvalidCert`. To block FB, we define a system rule that matches explicitly on `InvalidCert` as shown in Fig. 5.21. Because we prevented the transfer to DC and FB, we can safely discard the entities `InvalidContract` and `InvalidCert`.

Fig. 5.22 presents a partial transition system illustrating the interaction between the system

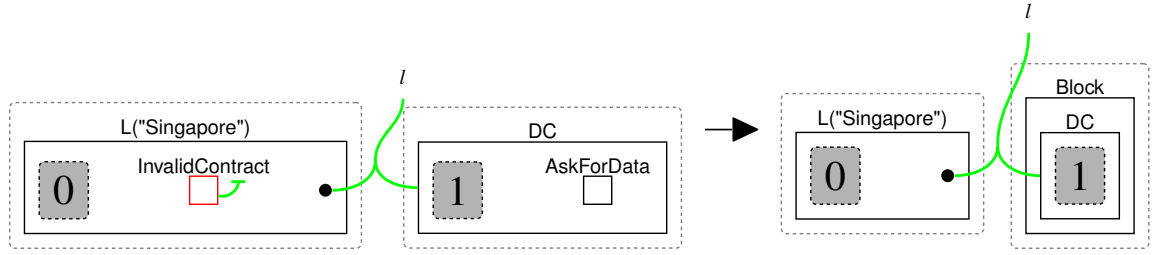


Figure 5.20: preventSing: preventing the transfer to the DC in Singapore.

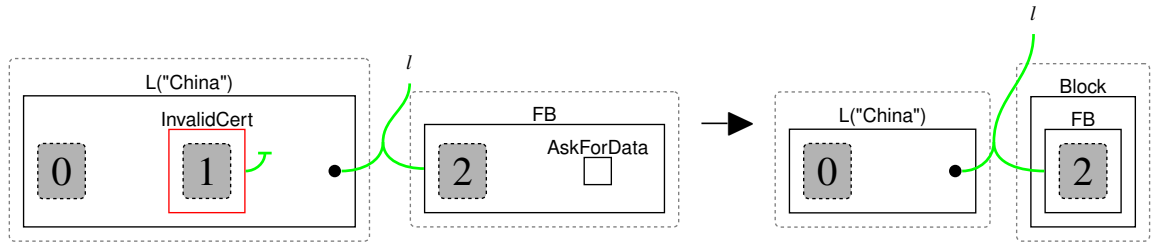


Figure 5.21: preventChina: preventing the transfer to the FB in China.

and privacy rules to identify the sender's and receivers' regions and validate their safeguards (Contract and Cert). Importantly, Fig. 5.22 shows only the transitions related to the receivers in Singapore and China, but the same modelling approach is applied to the receiver in Dubai as well. However, based on Fig. 5.11, the certification of FB in Dubai has three criteria. By applying rule `tagCriteria(x)` (Fig. 5.6), these criteria are tagged. Rule `checkCertResult` (Fig. 5.8) is then applied to tag the Cert as `CompliantCert`.

Before transferring data to FB in Dubai, it is necessary to verify the expiry date of the certification. We model two scenarios: one in which the certification is valid and another in which it has expired. In the former case, data transfer is permitted as demonstrated in (Fig. 5.23). In this case, FB can also request the withdrawal of its certification as illustrated in Fig. 5.24. The rule explicitly matches on the `CompliantCert` as we need to ensure that the entity requesting the withdrawal is using the certification, not the SCCs. Rule `processWithdReq` (Fig. 5.10) is then applied to process the withdrawal request.

In the second scenario (if the certification has expired), rule `expiredDate` (Fig. 5.9) tags it as `InvalidCert`, thereby preventing the transfer as shown in Fig. 5.25.

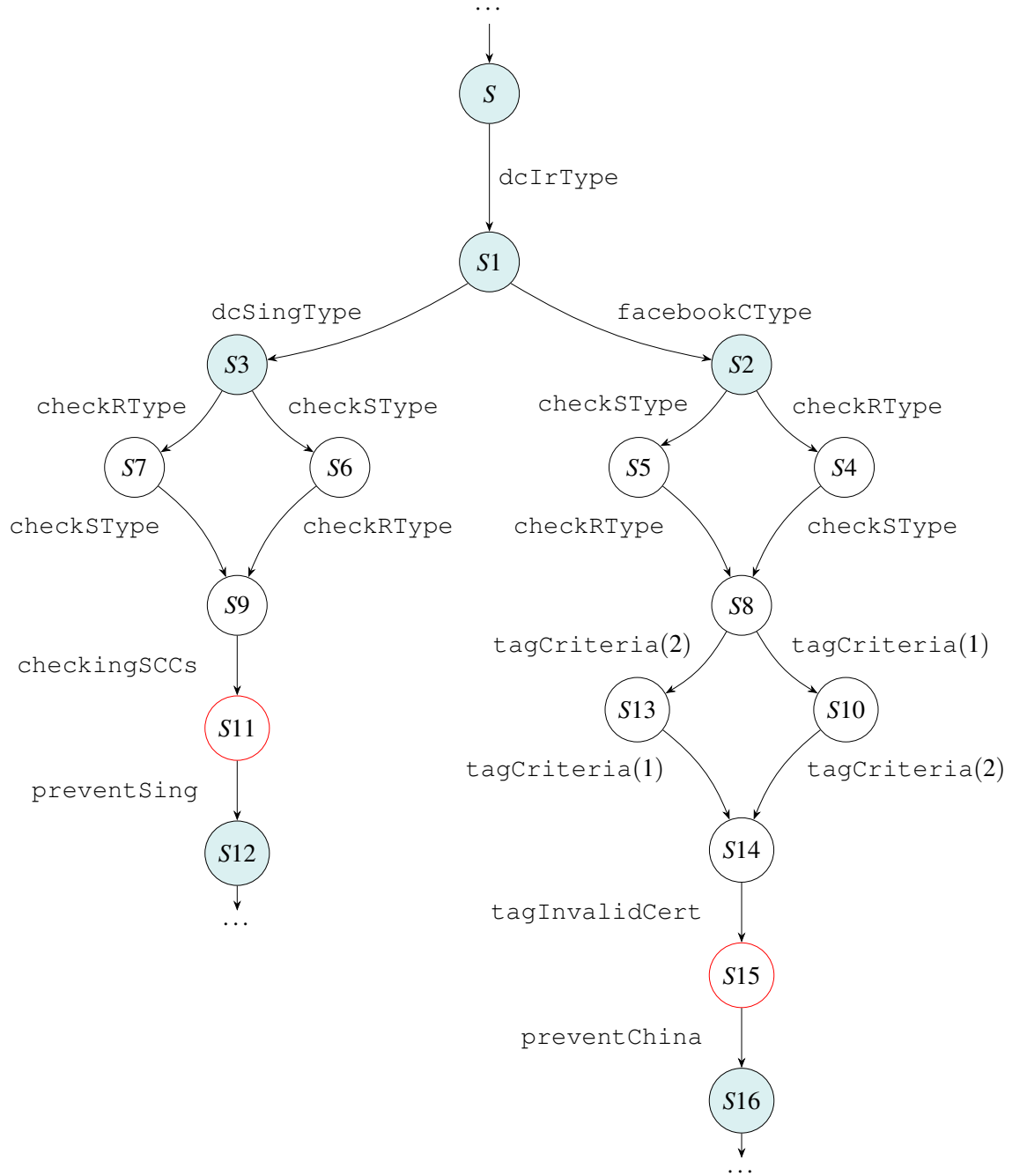
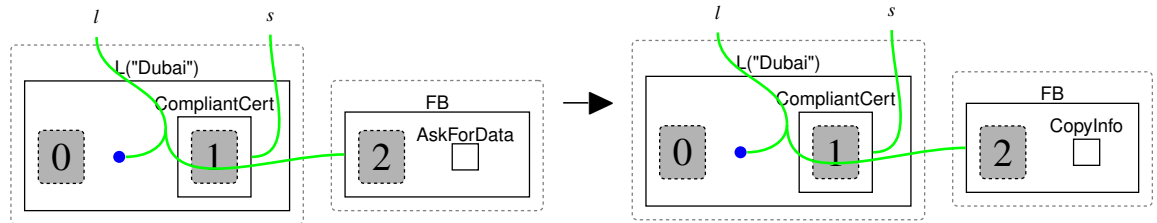
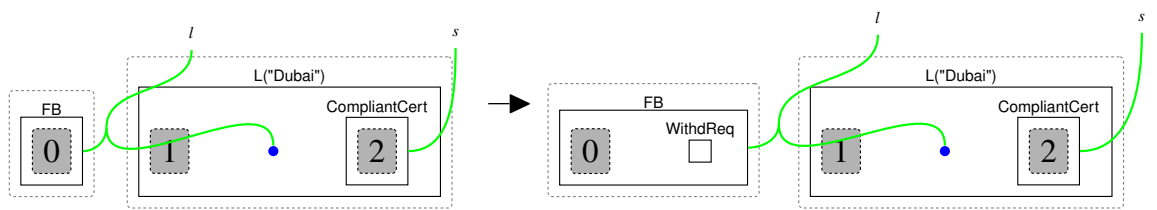
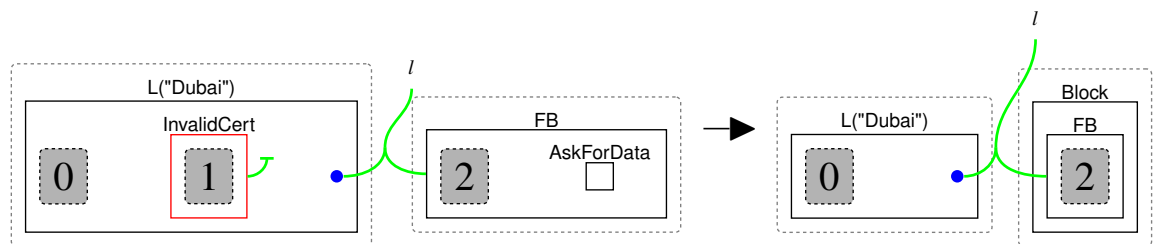


Figure 5.22: Partial transition system illustrating the interaction between system and privacy rules. Teal states are produced by applying a system rule.

Figure 5.23: `transValDate`: transferring data to FB in Dubai.Figure 5.24: `reqWithdCert`: FB in Dubai asks to withdraw its certification.Figure 5.25: `preventExpired`: prevent transferring data to FB in Dubai.

5.4 Discussion

In this chapter, we demonstrate how bigraphs, with their ability to represent spatial properties, can model the spatial relationships between entities. This capability allows us to model the GDPR notion of international data transfer and specify whether specific data transfers are restricted or permissible.

Although we illustrate the framework with a single example, the underlying formalism can be generalised to model a wide range of systems. By adopting a multi-perspectives approach, the framework captures the GDPR requirements for international data transfers without tying them to any particular implementation. This enhances the framework applicability and reusability. Moreover, the parameterisation within the model allows developers to represent different countries and to define as many transfer criteria as needed, making it flexible for diverse organisational settings. In the next chapter ([Chapter 6](#)), we will demonstrate how we can reuse this framework to model a banking notification example.

During the modelling phase, we abstract system-specific details (*e.g.* data collection and sharing mechanisms), since our focus is on modelling regulations rather than how to build a good system model. This approach also helps us avoid state space explosions, which occur when the number of possible system states grows uncontrollably, making analysis infeasible.

To enhance the model’s usability, we define an interface that allows end-users to generate the initial state automatically—rather than specifying it manually. We will present this interface in [Chapter 8](#). This helps mitigate the risk of incorrectly linking entities to their locations, which leads to applying the wrong reaction rules. For instance, if the data centre in Singapore is incorrectly linked with a pointer in Ireland, rule `sameReg` ([Fig. 5.3](#)) is triggered instead of rule `checkingSCCs` ([Fig. 5.5](#)), thereby incorrectly classifying the transfer as safe.

The aim of defining this model is to generate a transition system using BigraphER to conduct the *verification* step, *i.e.* proving that the transition system meets the GDPR requirements for international data transfers. In [Chapter 7](#), we will discuss how we perform this step to prove that the transition system holds the privacy properties we aim to verify (the requirements for cross-border data transfers).

5.5 Summary

In this chapter, we presented a bigraph-based framework for modelling the GDPR requirements for cross-border data transfers. The framework includes predefined privacy entities and rules, while leaving system-specific components to be defined according to each application's needs. It captures data transfers governed by adequacy decisions and appropriate safeguards (SCCs and certification mechanisms). In [Section 5.1](#), we outlined the framework's static structure, while its dynamic behaviour is described in [Section 5.2](#). We applied the framework to an example based on WhatsApp's privacy policies in [Section 5.3](#), illustrating how specific systems can be integrated with our framework. Finally, we discussed key points regarding the model in [Section 5.4](#).

Chapter 6

Privacy Regulations Model

In this chapter, we introduce a bigraph-based framework that captures key privacy notions: *providing and withdrawing consent, purpose limitation, the right to access, and data sharing with third parties*. All of these are core requirements of the GDPR and the CCPA. We also show how the cross-border data transfer model introduced in [Chapter 5](#) is integrated with the framework presented here.

To show the generality of the proposed framework, we apply it to two examples: a bank notification system, inspired by Monzo’s privacy policy, and a cloud-based home healthcare system based on the Fitbit app’s privacy policy. The first example is modelled based on the GDPR, while the second is aligned with the CCPA.

The chapter is structured as follows. [Section 6.1](#) explains how we model system privacy based on a multi-perspective approach. We show how these perspectives come together using the banking example in [Section 6.2](#). In [Section 6.3](#), we illustrate how, through reaction rules, we model the dynamic nature of privacy, *e.g.* performing permission checking. The integration between the privacy framework and system-specific aspects is explained in [Section 6.4](#). We show the approach is reusable in [Section 6.5](#) by applying it to a second system example. [Section 6.6](#) highlights some reflections on the model, and [Section 6.7](#) summarises the key points of the chapter.

6.1 A Visual Approach to Privacy Modelling

Similar to the cross-border data transfer model (Chapter 5), we use multi-perspective modelling to represent specific systems independently of privacy policies, enhancing the model's reusability across different scenarios.

We introduce our modelling approach with an example of a bank notification system inspired by Monzo Bank's privacy policy [97]. A user generates transactions using the Monzo mobile app. Monzo stores the user name and transaction details to calculate the total amount spent, and sends it as a push notification to the user. The bank can share transaction information with a third-party advertiser, *e.g. AnalogFolk company* [8], if the user has given consent. The parent company of the third-party is located in the UK and it needs to share transaction information with its branch in China [7]. We model from the view of a *single* user interacting with the system, as the data processing is equivalent for all users. This means however, we cannot model, for example, issues where data is sent to the wrong user.

Our framework consists of five perspectives: System perspective (Fig. 6.1), DGE perspective (Fig. 6.2), ADTypes perspective (Fig. 6.3), Locations perspective (Fig. 6.4) and Consent perspective (Fig. 6.5).

The System perspective (Fig. 6.1) models the system-specific entities of the banking application¹. It consists of a database (DB), a Notifier and an AdCompany. The Notifier reads the stored transactions, calculates the total amount spent, and sends it as a push notification to the User through the Monzo mobile App. The AdCompany is a third-party, whose main branch (MainBran) gathers users' transaction information (TransInfo) for advertising purposes. Marketing is the marketing branch of AdCompany with which MainBran needs to share the data. We use the general term *agent* to refer to system entities, *e.g.* DB and Notifier.

To use these system-specific entities and data with the privacy model, they must be mapped to general privacy types, *e.g.* data processors, third-parties, *etc.* This is handled by the ADTypes perspective, which is shown in Fig. 6.3. The agents' physical location must also be specified using the same modelling mechanism for locations as discussed in Chapter 5. The Locations perspective of this example is presented in Fig. 6.4. The specific policies applied to different entities and data types are then provided in the DGE perspective (Fig. 6.2) keeping the general-purpose privacy policies separate from the system being regulated. Finally, Fig. 6.5

¹Since Monzo policies intentionally lack technical details to make them more accessible to non-technical users, we inspire these system-specific entities from [133].

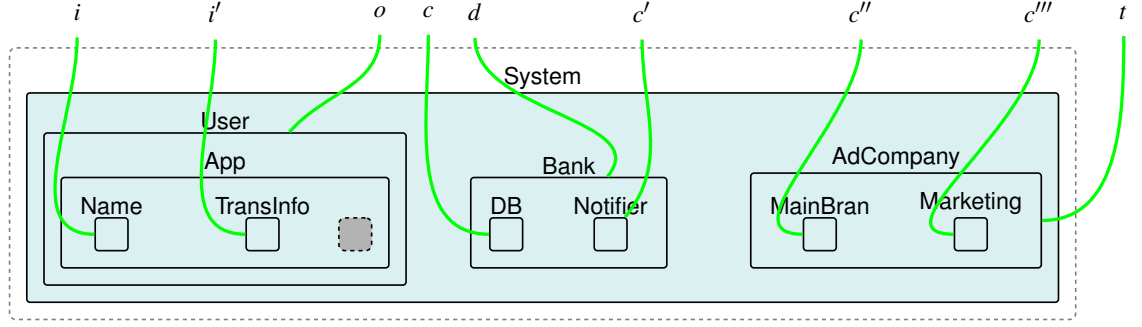


Figure 6.1: System perspective, which includes only specific-system entities. This perspective is defined by end-users.

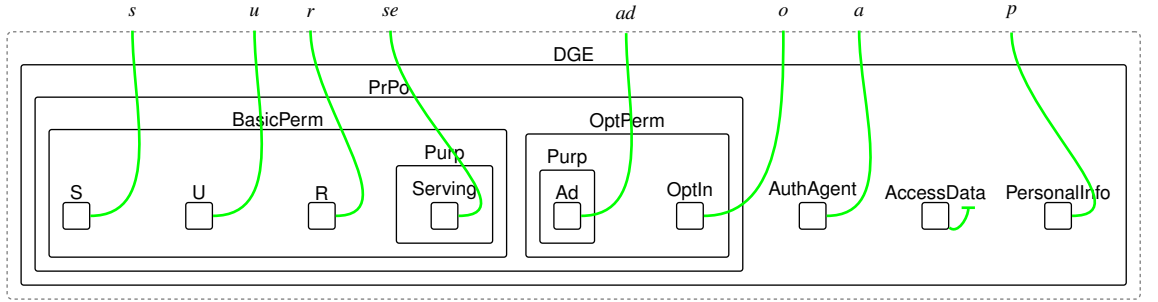


Figure 6.2: DGE perspective models the specific privacy policies applied to entities and data.

shows the Consent perspective that keeps track of the granted permissions and the accepted purposes as we will discuss in [Section 6.1.4](#) and [Section 6.3.1](#).

In the following sections, we will explain each perspective in more depth and provide a simple example to illustrate how we integrate them via links.

6.1.1 Modelling Data Governance Entity (DGE)

Data controllers under the GDPR and *businesses* under the CCPA [65], [126] are responsible for determining privacy policies, *e.g.* who has access to data and with what permissions [56], [126]. We use the term *Data Governance Entity* (DGE) to generalise across the GDPR and CCPA regulations. The entities of this perspective are listed in [Table 6.1](#).

We identify these parties and data types using hyper-links as *sets*. For example, an *AuthAgent* links to all agents in the *ADTypes* perspective that can access personal information. Conversely, if the entity is not linked to *AuthAgent* then it is unauthorised. *PersonallInfo* similarly identifies *sets* of data through linking. The classification of information into personal

Table 6.1: DGE perspective entities.

Entity	Description
PrPo	The privacy policies that are determined by the DGE: either basic permissions (BasicPerm) or optional permissions (OptPerm)
AuthAgent	Linked with in order to determine Authorised agents
PersonalInfo	Whether information needs stronger privacy
AccessData	Linked to when the user has a right to access data

and general is based on the definition of the personal information in the privacy regulation that we aim to model. For example, the GDPR and the CCPA classify any information that can *lead to* identification of a specific person as personal. The aim of specifying authorised entities and personal information is to prove there is no unauthorised access to personal data as we will discuss in [Chapter 7](#).

A core entity is the Privacy Policies (PrPo) which are either BasicPerm or OptPerm. BasicPerm contains the permissions the system **needs** consent for in order to provide its services to the user. Basic permissions are *store* (S), *update* (U) and *read* (R). The basic purpose (Purp) is Serving, *i.e.* serving the system's users. End-users of the model can add additional permissions and purposes according to their needs, *e.g.* writing permission and research purpose. OptPerm contains permissions that are not essential to provide services to the users, *e.g.* advertising only. Even if a user rejects these, the user is still able to use the system. Here we have additional, but extensible, permissions and purposes, including the OptIn permission, and Ad (advertising) purpose.

During modelling, hyper-edges are added to link *each* permission/purpose with: (1) the data that needs this permission, and (2) the entity that performs this processing.

AccessData is used to denote when an owner can access their data. This is described in detail in [Section 6.3.3](#).

6.1.2 Specifying Agent And Data Types

Agents and data types in privacy regulations are defined in abstract terms, *e.g.* a processor. To map these to system-specific agents or data, *e.g.* a database, we use the ADTypes perspective that maps, using links, system agents and data to general regulatory agents/terms. This provides flexibility in decoupling privacy rules from system rules.

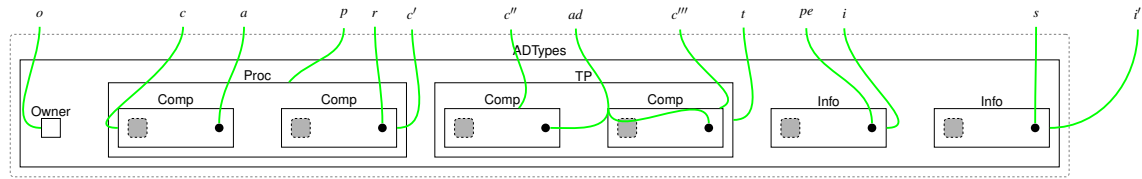


Figure 6.3: ADTypes perspective maps each system agent and data to their corresponding types via links. Sites nested within Comp and Info represent authentication levels (authorised entities or personal data) and permissions.

Table 6.2: Supported Agent and Data (AD) Types.

ADType	Description
Owner	The agent who owns the data: usually a user
Proc	Processor, the agent that needs to process data to provide a service according to the controller’s instructions
TP	Third-party, other agents who may hold user data for a specific purpose
Info	Specific data being managed

We support the common agent and data types specified in [Table 6.2](#), but the model is extensible and more could be added if necessary for a specific system/regulation pair, *e.g.* sub-processor.

Owners represent agents who own specific data, which usually corresponds to the users of the system². Agents acting as owners get their own Owner entity within the ADTypes region and the relationship is tracked through links. Data owners have specific operations in regards to consent and these are discussed in [Section 6.3.1](#), [Section 6.3.3](#), and [Section 6.3.5](#).

Data processors are responsible for the safe handling of owners data to fulfil a particular service. In the banking example, Bank is the main data processor and multiple sub components, *e.g.* databases, may fall under the remit of this main processor. We model this through nesting, with a Proc entity containing multiple processor components Comp. These components may have their own authorisation level and permissions. Each Comp is linked to the system entity that processes the data (*e.g.* DB), the authentication level (if it is authorised), and the component’s permissions in the DGE perspective.

As bigraph entities have a fixed number of ports, we use additional entities (pointers), denoted by solid black bullet points, to allow any number of links to be specified. For

²We use the term *owner* to generalise across different privacy frameworks, such as data subjects in the GDPR and consumers under the CCPA.

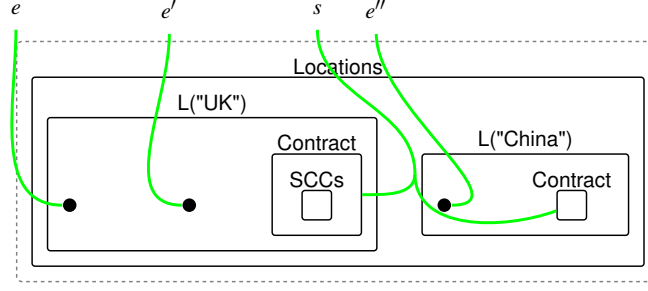


Figure 6.4: Locations perspective.

simplicity, we say these agents are linked to each other, even though there is an additional level of indirection in practice.

Third-parties are similar to data processors, but have reduced access to owner data. We model them in a similar way: with a TP entity nesting any sub-components of the specific third-party.

Finally, we abstract specific information, *e.g.* a users name, to a general Info data type. Info is linked with the specific properties we want of that data, for example, is it personal or general, and what permissions (*i.e.* store/read/update) does it have.

6.1.3 Specifying Agent Locations

Each agent is linked to a location in the Location perspective (Fig. 6.4) to allow checking the GDPR requirements for sharing data with third-parties outside the EU. We use the same mechanism to model locations as presented in Chapter 5. However, in this chapter, we present only one GDPR legal basis for cross-border data transfers (providing a safeguard: SCCs)³.

6.1.4 Consent Perspective

As consent is shared between both the data owner and the DGE, we track this within an additional region (that each can reference). It is initially an empty perspective as consent is not specified in the model a-priori, but constructed during a model run using the rules presented in Section 6.3.

³Other legal bases, such as adequacy and certifications, can also be modelled using the same mechanisms discussed in Chapter 5.

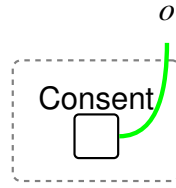


Figure 6.5: Consent perspective is dynamically constructed during model execution using reaction rules.

6.2 Example: Agent, Data Types And Locations for the Banking System

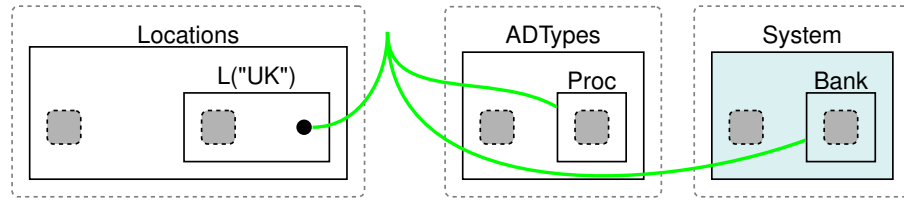
In this section, we gradually define a (partial) initial state of the banking system to demonstrate the mechanism for integrating the System perspective with the privacy perspectives.

In [Fig. 6.6a](#), the Bank which itself is made of several components (a database and a notifier) is a data processor (Proc) as shown by the link into the ADTypes perspective. We specify the Bank's location by linking it to the UK in the Locations perspective.

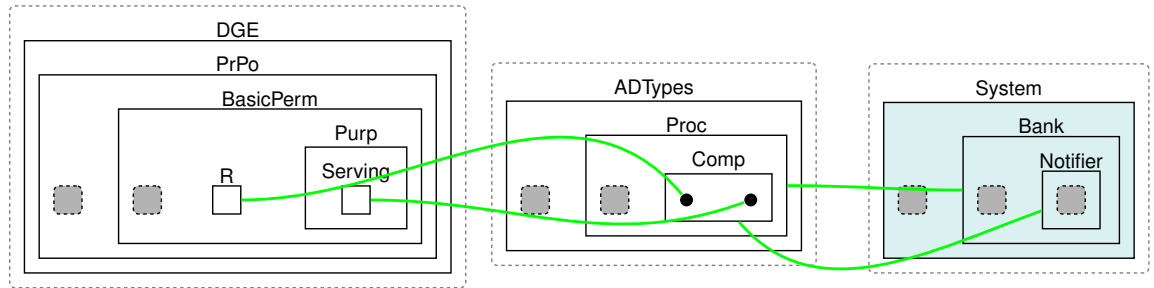
Specific components, *e.g.* Notifier, are created as nested components (Comp) inside Proc as shown in [Fig. 6.6b](#). Each Comp must be linked to: AuthAgent if authorised; the permissions it can exercise on specific data; and the purposes of accessing that data. To illustrate this, Comp, in [Fig. 6.6b](#), is not linked to AuthAgent in DGE perspective because Notifier must not access user's name. As Notifier needs to read user's transactions for serving purposes, we link Comp to R and the purpose Serving.

In [Fig. 6.6c](#), the information being managed include user transaction information (TransInfo) is assigned type Info in ADTypes. The links to the DGE specify how Info should be handled: TransInfo is general information as it does not linked to PersonallInfo and has read (R) permissions and Serving purposes. Likewise, User is assigned as a data Owner and linked to Consent to model the relationship.

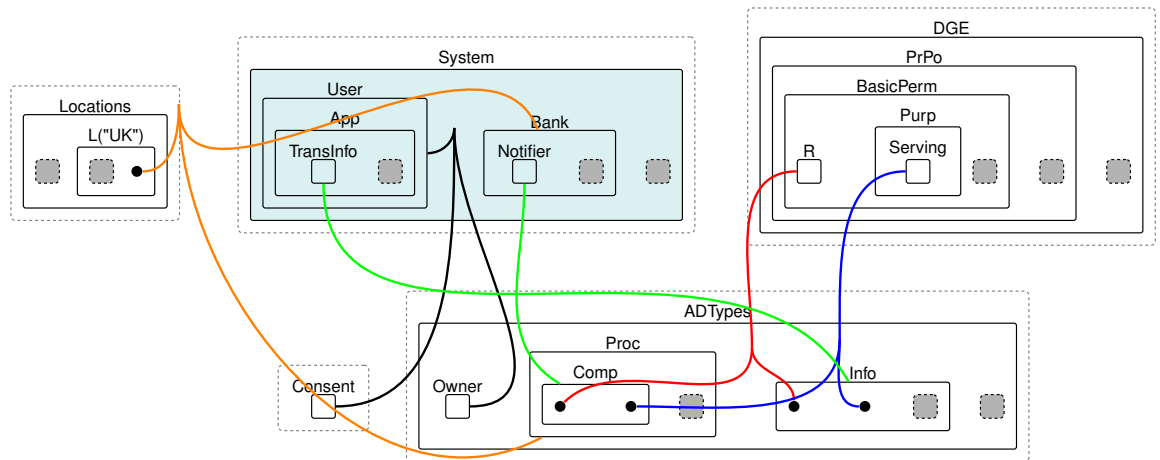
[Fig. 6.7](#) shows the full initial state of the banking example, where we use the same linking technique to integrate other system entities with privacy perspectives. The model is flexible, *e.g.* it is easy to remove AdCompany (and associated links/entities in the privacy perspective) if this service is no longer being required.



(a) Linking the system agent Bank with its type and location.



(b) Linking the Notifier component of the Bank with its type and permissions.



(c) Linking TransInfo with its type and permissions.

Figure 6.6: Partial initial bigraph (state) for the banking system scenario.

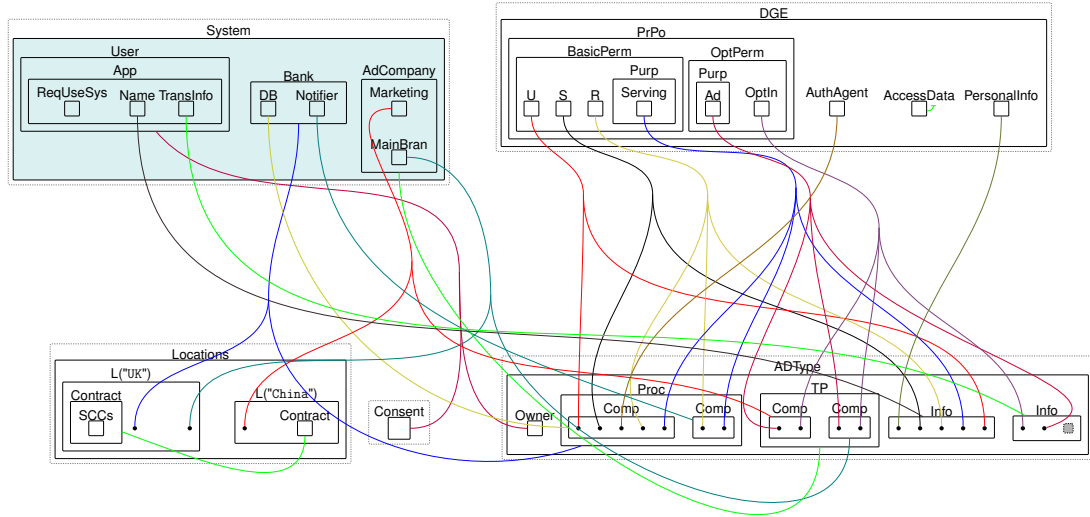


Figure 6.7: The initial state for the banking system scenario. System-specific entities are in teal, while uncoloured regions are part of the reusable privacy model. The entity *ReqUseSys* represents the user request to start using the system. Solid black bullet points are entities (referred to as pointers) that allow end-users to assign an arbitrary number of links. Each system entity is linked to its corresponding type in *ADTypes*, its location and its permissions in *DGE*. For example, the links denote *AdCompany* as third-party (*TP*). *Marketing* is a component of *AdCompany*, so it is linked to *Comp*, which is nested within *TP*. The location of *Marketing* is *China*. To link *Marketing* with its permissions, we connect the pointers nested within *Comp* to *OptIn* and the purpose of accessing the user's data, which is advertisements purpose (*Ad*). Similarly, the links denote *Name* as *PersonallInfo*, with store (*S*), read (*R*), and update (*U*) permissions for *Serving* purposes. The site nested within *Info* represents pointers linked to *R* and *Serving*, as shown in [Fig. 6.6c](#).

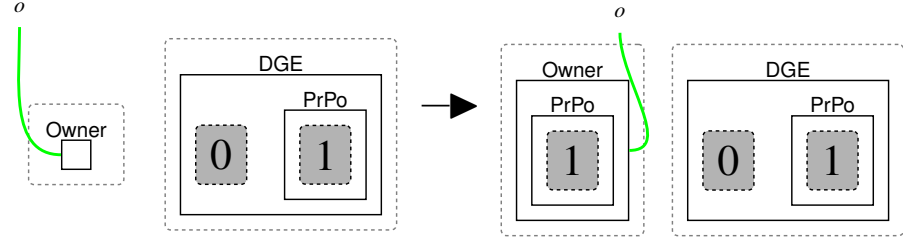


Figure 6.8: `sendPolicy`: sending privacy policies to the data owner.

6.3 Privacy Dynamics

The static model describes the system setup, *e.g.* emphasising what agent is authorised and the permissions, but does not describe interaction with the system. To encode interaction we develop a set of privacy reaction rules. As with the static model, we give general purpose rules we expect to apply to a wide variety of examples but bigraphs are open to extension and a modeller may add their own additional rules if required.

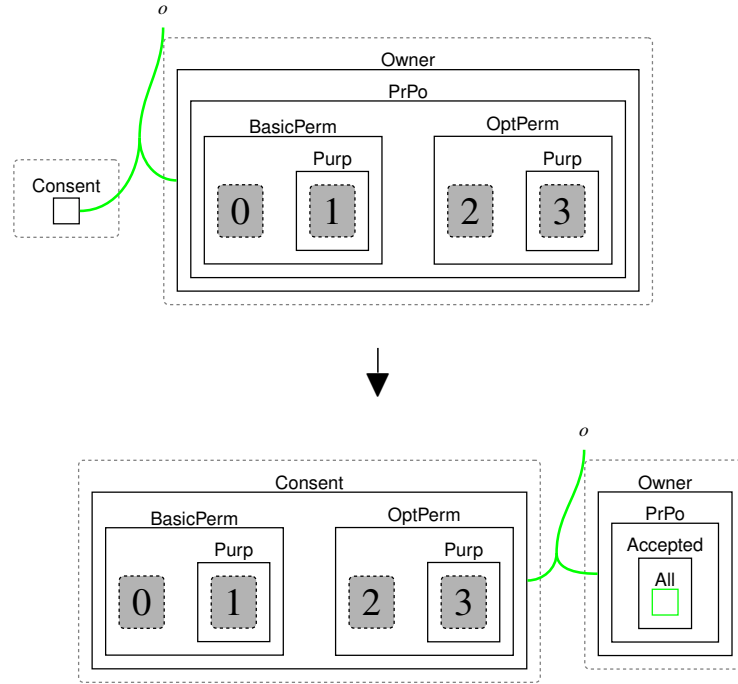
We categorise the rules into set for: (1) providing consent, (2) permission checking, (3) handling the right to access, (4) sharing data with a third-party, (5) withdrawal of consent.

6.3.1 Providing Consent

Before user data can be collected their consent must be obtained. Rule `sendPolicy` (Fig. 6.8) models the sending of a policy from the DGE to an owner. As we send policies to the abstract data owners, this same rule applies whether the owner is, for example, a banking customer or a patient in a healthcare system.

Based on the policies, an owner can decide on what consent to give. The consented terms are placed in the Consent perspective to explicitly express the notion of *freely given* and *genuine* consent [32], [58]. There are three cases:

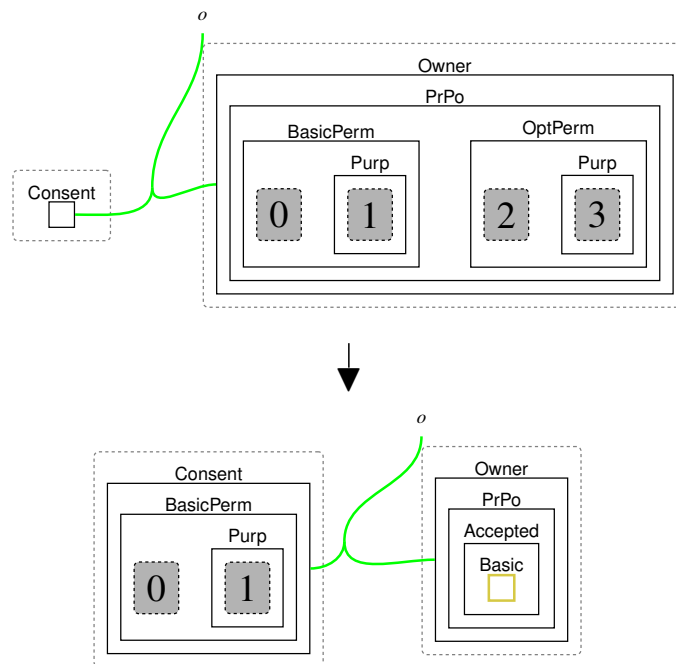
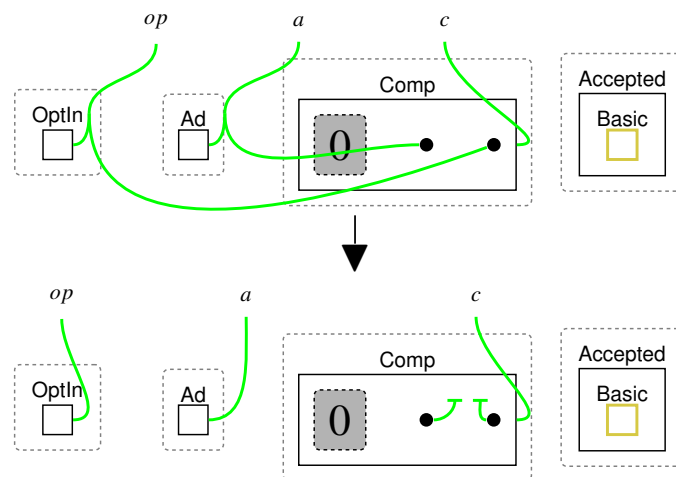
1. Accept all BasicPerm, OptPerm and Purp, as modelled by rule `acceptAll` (Fig. 6.9). Here, we track the owner has accepted all policies through a new entity `All`, and copy (through an instantiation map) the specific policies into the Consent entity for that owner.
2. Accept only BasicPerm, and Purp but reject optional permissions, as modelled by rule `acceptBasic` (Fig. 6.10). This is the case, for instance, when the data owner does

Figure 6.9: `acceptAll`: accepting all permissions and purposes.

not want to share their data with a third-party. As before, we track the acceptance with a new entity `Basic` and this time copy *only* the basic permissions and purposes to the `Consent` entity for this user. A second set of rules (Fig. 6.11) closes links between the optional permissions in the DGE perspective and the components of the third-party in the ADTypes perspective to indicate consent was rejected. This is a family of rules, and we define one per-type of optional purposes, *e.g.* one for Ad, one for Marketing *etc.*

Given that users may change their minds regarding their choices, we use rule `updateCons` (Fig. 6.12) to reset `Consent` by removing its content and generate `Update`, which represents the Owner's request to update the consent. The entity `Update` serves as a flag to apply rule `relinkPerm` (Fig. 6.13), re-establishing the links for the rejected permissions. This, in turn, allows rule `sendPolicy` (Fig. 6.8) to be re-executed, enabling the user to revise their choices.

3. Reject all policies, as modelled by rule `rejectAll` (Fig. 6.14). We use the entity `Rejected` to track the user's rejection. As we do not have partial permissions like the reject optional case, we do not need to remove links as the privacy policies are

Figure 6.10: `acceptBasic`: accepting only basic permissions and purposes.Figure 6.11: `closeLinks`: remove links between rejected permissions.

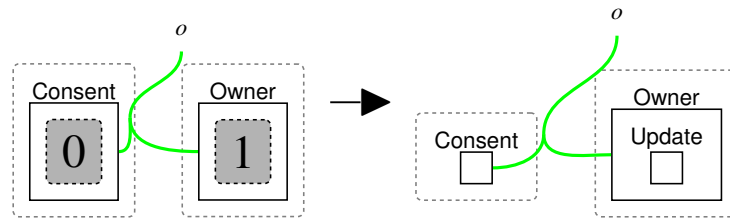


Figure 6.12: updateCons: Owner's request to update consent.

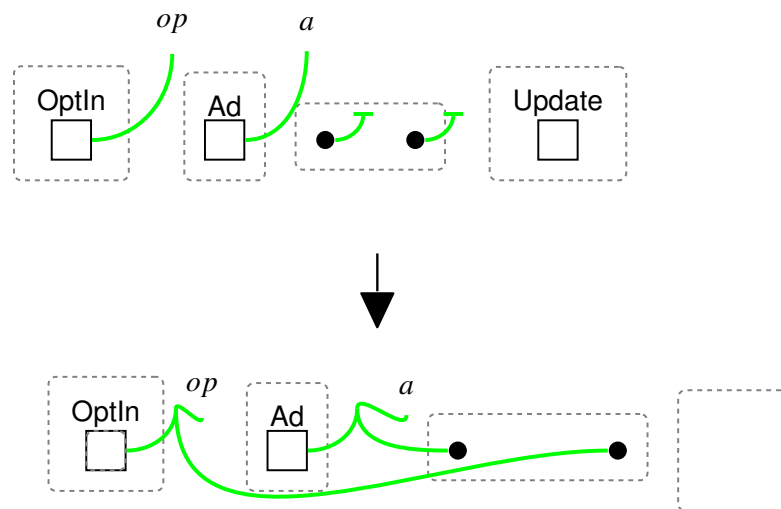
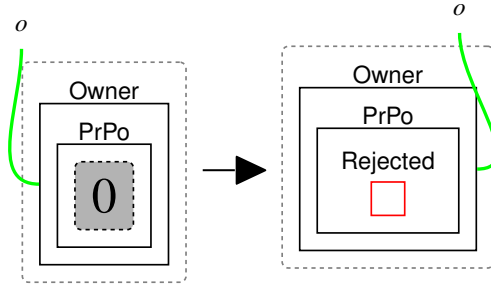
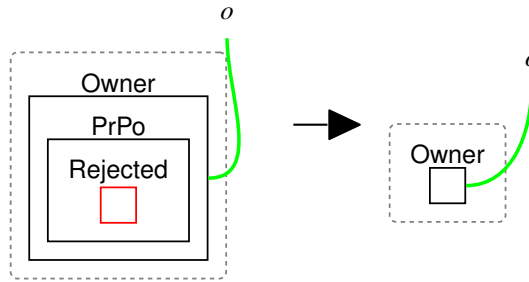
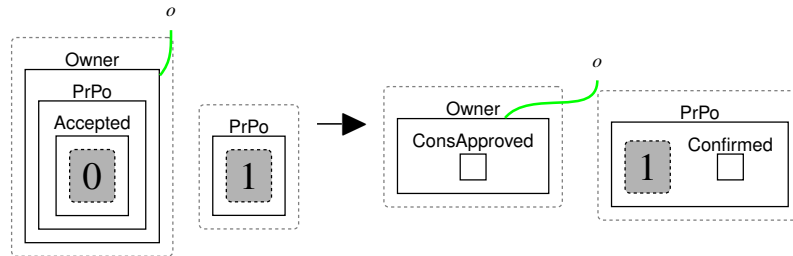


Figure 6.13: relinkPerm: Relink the rejected permissions with their corresponding Comp.

Figure 6.14: `rejectAll`: rejecting all permissions and purposes.Figure 6.15: `changeDec`: changing decisions after rejection all privacy policies.

never copied to the Consent perspective. To enable users to revise their decisions after the rejection, we can use rule `changeDec` (Fig. 6.15) that resets the bigraph to a state where rule `sendPolicy` can be reapplied. In our model, the system treats the absence of consent the same as an explicit rejection since neither grants any permissions. Nevertheless, the end-user can verify if no consent is provided by checking for All, Basic, or Rejected in the PrPo that is nested within Owner. If neither entity is present, no consent is given.

Once the owner has made their decision on the privacy policy, their acceptance (either full

Figure 6.16: `confirm`: confirming a DGE has evidence of data-owners consent.

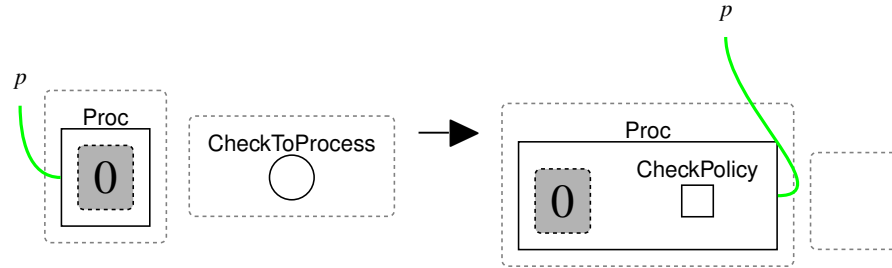


Figure 6.17: `startCheck`: the first rule that is used to start checking the permissions.

or non-optional only) is confirmed by the DGE. This is needed as some privacy regulations, *e.g.* GDPR [57], require data controllers to keep evidence that proves the users consent. Confirmation is modelled through rule `confirm` (Fig. 6.16) that simply adds entities (Confirmed and ConsApproved) to both the DGE and owner. Site 0 represents the entity All or Basic. It is deliberately not preserved in the right-hand side once the user's decision is confirmed. Similarly, the owner's copy of the privacy policy (PrPo) can be safely discarded, since the user's choices are already recorded in the consent perspective.

6.3.2 Permission Checking

Before the system starts accessing (storing/reading) data the user's consent must be checked to ensure they consented to these permissions and purposes.

As we set up links between the DGE, agents and data types, and consent perspectives based on the accepted policies, determining if a particular permission/purpose has been accepted is equivalent to checking a specific link exists. As we are interested in detecting incorrect accesses, we instead check for the absence of these links.

Checking the policies are valid follows a two step process: (1) initialise the checking phase by using rule `startCheck` (Fig. 6.17), (2) iteratively check all links using rule `changeTypeComp` (Fig. 6.18).

The entity `CheckToProcess` (Fig. 6.17) should be generated by a system rule in the system perspective (Bank) to start the checking process. The privacy checking phase then uses rule `startCheck` (Fig. 6.17) to place a `CheckPolicy` entity inside the `Proc`, indicating a policy check has started.

Once `CheckPolicy` is generated, the checking process starts to check the absence of the links using rule `changeTypeComp` (Fig. 6.18). This rule is used to change the type of any

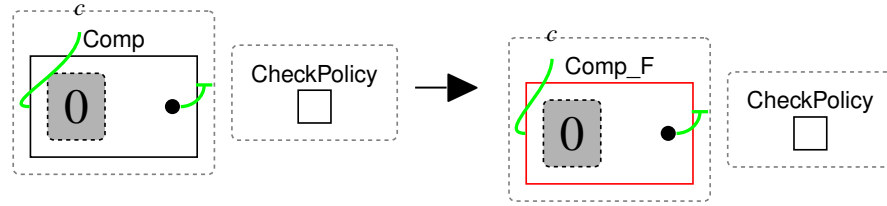


Figure 6.18: `changeTypeComp`: changing the component's type that has a rejected permission/purpose.

component that contains at least one permission/purpose with a closed link to `Comp_F`. For example, if the user rejected the optional permissions, the types of `Comp` linked to `MainBran` and `Marketing` are changed to `Comp_F` to identify that `MainBran` and `Marketing` should be prevented from accessing the data.

We can check the policies multiple times, *e.g.* before executing each process, by allowing the system to generate the `CheckToProcess` entity any time we need to check the consent. This handles, for example, policy updates over time.

6.3.3 Right to Access

Once the data is stored in the system, the user must be allowed to access their data at any time. Rule `rightToAccess` (Fig. 6.19) connects the owner to the DGE (via the `AccessData` entity) to indicate that the user can access the data. We will use this link to perform verification (Chapter 7).

Rule `rightToAccess` should be applied once the user's data are stored in the system. To ensure that, when the checking process is done and the system stores the data, the entity `CheckPolicy` should be replaced with `RightToAccess` to allow this rule to be applied as it will be discussed in Section 6.4. After applying this rule, the entity `RightToAccess` is deleted because there is no need to reuse the rule; the user can still access their data even if they update their permissions as permission updates do not affect the link between `AccessData` and `Owner`.

6.3.4 Sharing with Third-Parties

Before sharing `TransInfo` with the third party, we must check the user's consent. If the types of `Comp` linked to `Marketing` and `MainBran` are changed to `Comp_F`, sharing `TransInfo` must

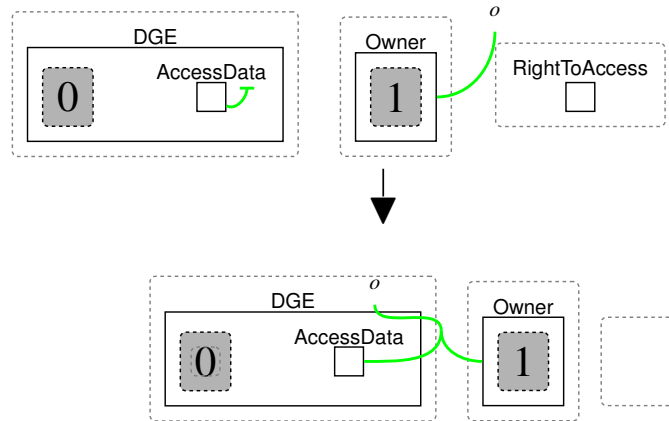


Figure 6.19: `rightToAccess`: allowing the owner to ask the DGE for data access.

be prevented, as the user has declined to share their information. Note: while `TransInfo` may not always be considered personal data, it can often identify individuals (*e.g.* through payment history). Therefore, under regulations like the GDPR, it should be treated as personal data. Checking consent also ensures that the user's data will be used in accordance with the purposes the user has accepted, in line with the purpose limitation principle.

Besides checking consent, some regulations (*e.g.* the GDPR) require safeguards for data transfers outside the EU. This means that before sharing data, we must determine whether the transfer is international (restricted) and ensure the appropriate safeguards are in place. To do this, we use the same mechanism and rules previously used to check the GDPR requirements for cross-border data transfers in [Chapter 5](#).

6.3.5 Withdrawing Consent

Just as users have the right to provide consent and access their data, they also have the right to withdraw their consent at any time. Users can withdraw both `OptPerm` and `BasicPerm` or choose to withdraw only the `OptPerm`. Importantly, the rules governing consent withdrawal must be applicable at any point during the system's state transitions. This is achieved through the use of priority classes, enabling users to revoke their consent—either fully or partially—at any stage of the system's evolution.

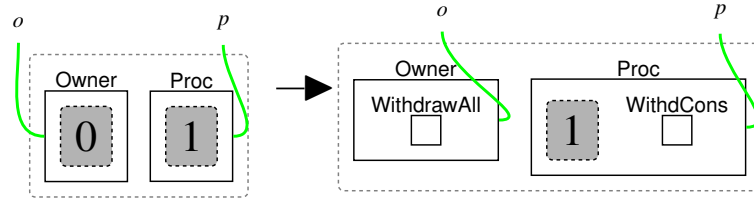


Figure 6.20: `withdrawReq`: the owner requests to withdraw all permissions.

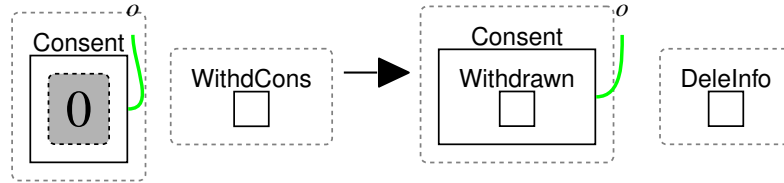


Figure 6.21: `processWithd`: the consent is withdrawn and the user's data should be deleted from the system.

Withdrawing All Permissions

Withdrawing consent is a three-step process (and uses three rules). First, a user sends a consent withdraw request through rule `withdrawReq` (Fig. 6.20). This places the entity `WithdCons` in the `Proc`. There is no need to specify which user should withdraw their consent as the model supports one user.

The processor then actions the withdrawal using reaction rule `processWithd` (Fig. 6.21) that removes all permissions from the consent perspective, by removing the site and replacing it with a `Withdrawn` entity, and discarding `WithdCons` as the process of withdrawing the consent is done. `DeleInfo` is used to delete existing data. Rule(s) to delete data are system-specific as they need information, for example, database models as discussed in Section 6.4. The deletion process does not necessarily occur immediately after the withdrawal process *e.g.* to allow companies to use batched deletions.

Finally, we confirm that consent is withdrawn using reaction rule `confirmWithdrawal` (Fig. 6.22). This step is not a specific requirement of the regulations, but is used to replace the confirmation of consent with the withdrawal. In this case, we simply change `Confirmed` to `ConfirmWithd` to record this operation.

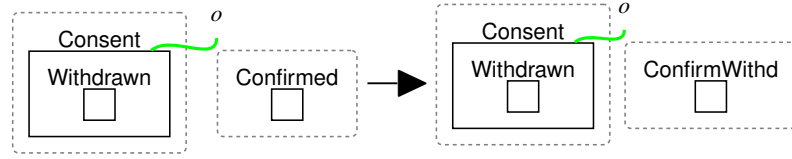


Figure 6.22: `confirmWithdrawal`: acknowledgement of successful consent withdrawal.

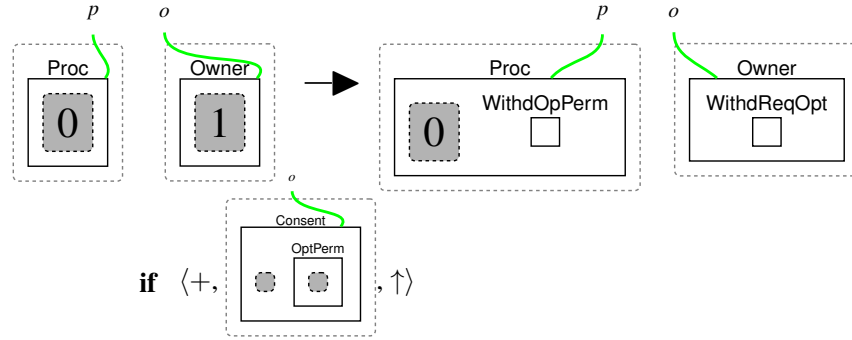


Figure 6.23: `withdrawReqOPT`: the owner requests to withdraw the optional permissions. The condition allows the rule to be applied only if the owner has already accepted the optional permissions.

Withdrawing Only Optional Permissions

Similar to withdrawing all the permissions, we have three steps to process withdrawing `OptPerm`. Rule `withdrawReqOPT` (Fig. 6.23) models the owner's request to withdraw the optional permission. This rule adds the entities `WithdReqOpt` in `Owner` and `WithdOpPerm` in `Proc` to allow the processor to start processing the partial withdrawal. The condition restricts the rule's application as the rule should be applied only if the user accepted all the permissions.

Rule `processWithdOpt` in Fig. 6.24 allows the user to update the consent by accepting only the `BasicPerm` and deleting `OptPerm` from the `Consent` perspective.

As the consent is updated, we must close the links of the rejected permissions and recheck the consent by generating the entity `CheckToProcess` as discussed in Section 6.4.

6.4 Integrating Privacy and System-Specific Behaviour

As shown in Fig. 1.1, a model consists of a reusable set of privacy entities/rules coupled to a system-specific model. We describe how the system model interacts with the privacy model

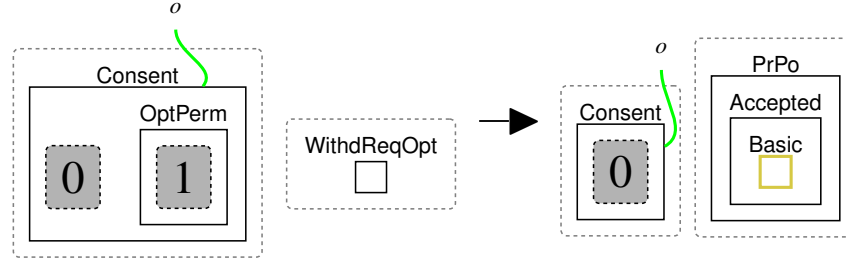


Figure 6.24: `processWithdOpt`: the optional permissions are withdrawn and regenerate Basic to recheck the consent.

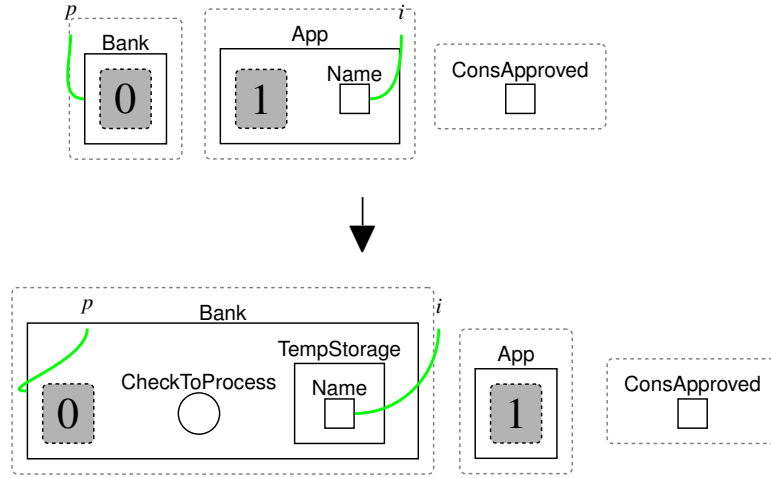


Figure 6.25: `registerRequest`: user's registration with the system.

by showing some of the rules needed in the banking example. We assume GDPR requirements and that the user has accepted the basic permissions required for interacting with the system.

Once the consent is approved (using privacy rules), the system can register a user by sending their Name to the Bank as shown in Fig. 6.25. The rule explicitly matches over `ConsApproved` that is generated by rule `confirm` (Fig. 6.16) to show a user has accepted some policies (but not the specific policies). Before starting to store or process user data, we must check the user's consent.

To do so, the user's Name is first temporarily stored in `TempStorage` before being stored in the database. The entity `CheckToProcess` is generated to tell the *privacy model* to begin the checking process by enabling rule `startCheck` (Fig. 6.17), *i.e.* `CheckToProcess` is a special flag that allows integration between the models.

After checking, the system continues based on the checking result. As the user accepted

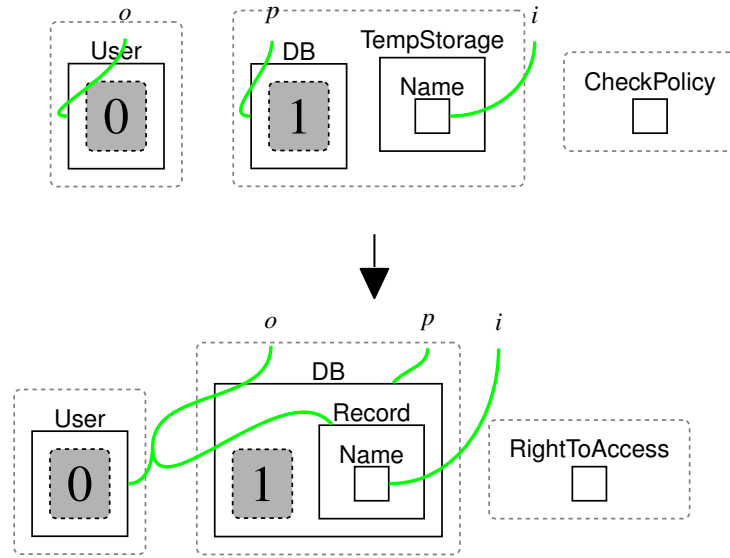


Figure 6.26: storingName: storing the user's name.

the basic permissions, the bank is now able to store the user's name using rule `storingName` (Fig. 6.26). To maintain provenance of the data, we link the User with the Record on link *o*. For the GDPR and other regulations with a notion of right to access, this rule must be followed by rule `rightToAccess` (Fig. 6.19), as the user must be allowed to access their data. To do that, the entity `RightToAccess` should be generated as shown in Fig. 6.26 to allow rule `rightToAccess` (Fig. 6.19) to be applied directly after storing the data.

The user now may ask to access their data. For example, rule `updateReq` (Fig. 6.27) models their request to retrieve and update their record. Here, `ToUpdate` denotes the request and `NeedUpdate` specifies which information must be changed. We omit `AllowServ` once processing begins and regenerate it upon completion, allowing the user to make further requests. The entity `AllowServ` is generated by a system (see Fig. A.1 Appendix A). By explicitly matching on `AccessData` (after linking it to `Owner`), we can ensure that the user can always access their data.

6.4.1 Sharing Data Based on Consent

Since the consent is checked before processing the user's data (as mentioned earlier), the `Comp` linked to `MainBran` and the `Comp` linked to `Marketing` are changed to `Comp_F` because the user has only accepted basic permissions. Rule `preventMainBran` (Fig. 6.28) and

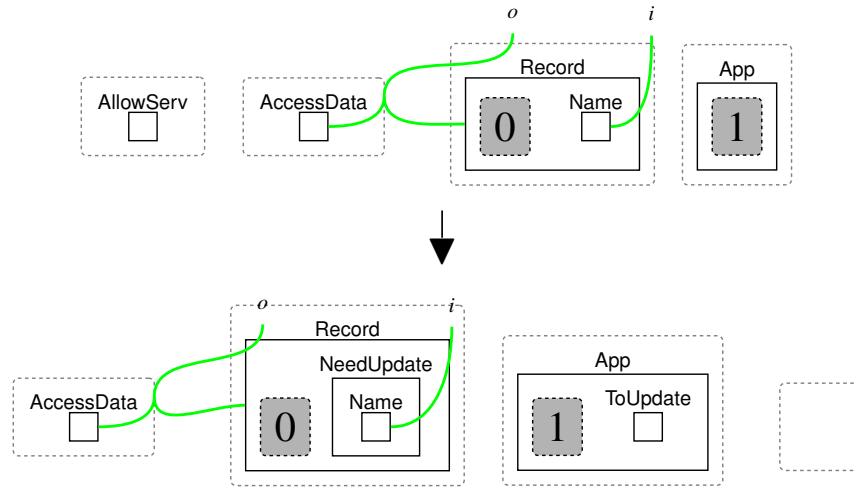


Figure 6.27: updateReq: user's request to update their data.

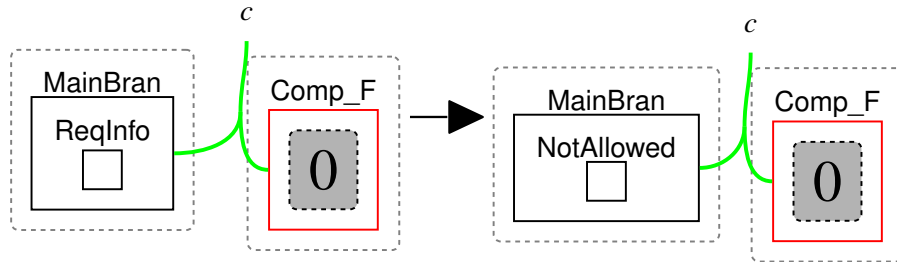


Figure 6.28: preventMainBran: prevent sharing the user's transaction information with the main branch (MainBran) of AdCompany as the user rejects the optional permissions.

preventMarBraCons (Fig. 6.29) are then applied to prevent information sharing with the main branch and marketing branch of AdCompany.

If there were no components with type Comp_F, then the user accepted the optional permissions, so we should start checking the second requirement.

6.4.2 Transferring Data based on GDPR Requirements for International Transfer

If the user accepts the optional permissions, we must determine whether the data transfer is international before proceeding. If so, we must check the GDPR requirements for cross-border data transfers. We use the same technique introduced in Chapter 5, which involves specifying the sender's and receiver's types, tagging their pointers to identify their regions, and verifying

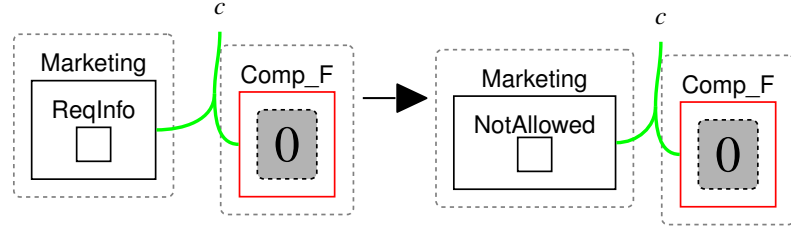


Figure 6.29: `preventMarBraCons`: prevent sharing the user's data with the marketing department.

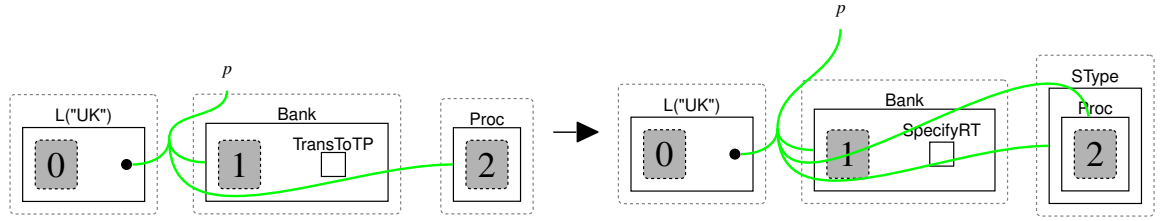


Figure 6.30: `senderType`: specifying the type of Bank.

the GDPR requirements if the transfer is international.

Rule `senderType` (Fig. 6.30) specifies the sender's type (Bank) by generating `SType` around the type and linking them together. `TransToTP` is generated by a system rule (see Figs. A.5 and A.6) to indicate that the system will transfer the data to a third party. It is replaced with `SpecifyRT` to start determining the type of the receiver(s). `SpecifyRT` is nested within the Bank because the Bank, as the sender, is responsible for specifying the receiver's type.

We have two receivers in this example: MainBran and Marketing. To specify the type of MainBran, we use rule `mainBrnType` (Fig. 6.31). As MainBran is a component of AdCompany, the entity `RType` is generated around Comp and linked to it. We replace `SpecifyRT` with `CheckReg`. Similarly, rule `marketType` (Fig. 6.32) specifies the type of Marketing.

Now, rule `checkSType` (Fig. 5.2a) and rule `checkRType` (Fig. 5.2b) are applied to tag the pointers linked to `SType` and `RType`, respectively. As shown in Fig. 6.33, the pointer linked to Marketing is located in China, while the pointer linked to Bank is located in the UK. In this situation, rule `checkingSCCs` (Fig. 5.5) is applied to change Contract that is nested within China into `InvalidContract`, because it does not include the SCCs. This means

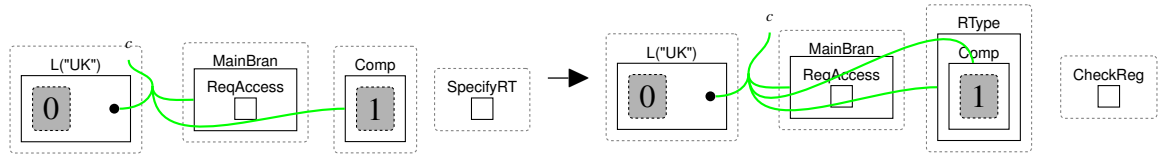


Figure 6.31: mainBrnType: specifying the type of MainBrn.

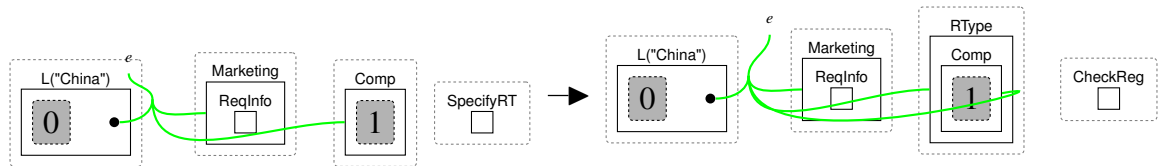


Figure 6.32: marketType: specifying the type of Marketing.

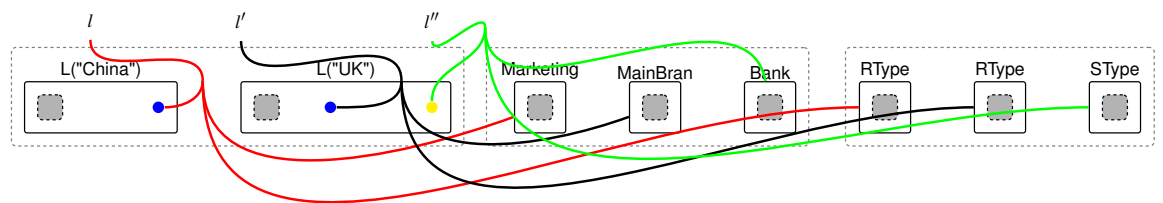


Figure 6.33: The tagged pointers after specifying the location of Bank, MainBrn and Marketing. The tagged pointers of Bank and MainBrn are nested within UK, while the tagging pointer of Marketing is in China.

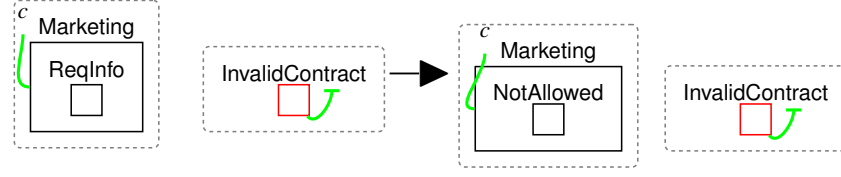


Figure 6.34: `prevMDSCCs`: prevent sharing the user's transaction information with Marketing as the Contract is invalid.

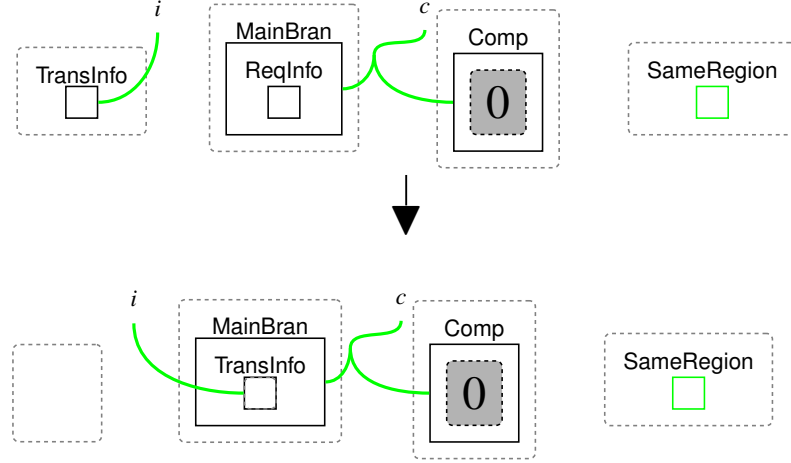


Figure 6.35: `shareMainBran`: sharing the data with MainBran.

we must not transfer the data to Marketing even if the user **consented** to do so as it does not meet the GDPR requirements for international data transfer. Rule `prevMDSCCs` (Fig. 6.34) prevents sharing the data with Marketing.

Conversely, MainBran and Bank are both located in the UK, as shown in Fig. 6.33. Since they are in the same region, rule `sameReg` (Fig. 5.3) is applied. We can then share the data without checking the SCCs by using rule `shareMainBran` (Fig. 6.35). This rule matches on Comp and SameRegion to ensure that data sharing proceeds only after checking the requirements (*i.e.*, checking consent and the requirements of the international data transfers).

6.4.3 Withdrwing Consent

The user also should be able to withdraw the consent at any time and the consent should be withdrawn once the user asks for that (rule `withdrawReq` in Fig. 6.20 and `processWithd` in Fig. 6.21). After withdrawing the consent, rule `deleteInfo` (Fig. 6.36) deletes the user's

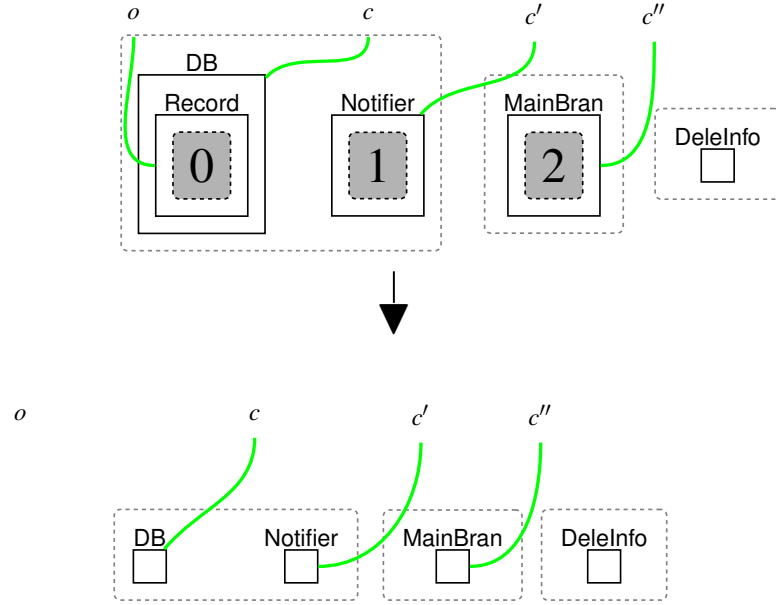


Figure 6.36: `deleteInfo`: deleting the user's information after withdrawing the consent.

information from DB, Notifier and MainBran. To ensure that rule `deleteInfo` (Fig. 6.36) is applied after withdrawing the consent, we explicitly match on `DeleInfo` that is generated by rule `processWithd` in Fig. 6.21. The last step is confirming the withdrawal by using rule `confirmWithdrawal` (Fig. 6.22).

The user also has the right to withdraw only the `OptPerm`. Rule `withdrawReqOPT` (Fig. 6.23) and `processWithdOpt` (Fig. 6.24) are applied to perform the withdrawal step of the optional permissions. After performing the partial withdrawal, we use rule `closeLinks` (Fig. 6.11) to close the links of the rejected permissions. Because the consent is updated, we should recheck it before starting processing the user's data. Rule `recheckCon` (Fig. 6.37) generates the flag `CheckToProcess` to trigger rule `startCheck` in Fig. 6.17 to start checking the consent. We match on `WithdOpPerm` (generated by rule `withdrawReqOPT` in Fig. 6.23) to allow rule `recheckCon` (Fig. 6.37) to be applied after withdrawing the optional permissions.

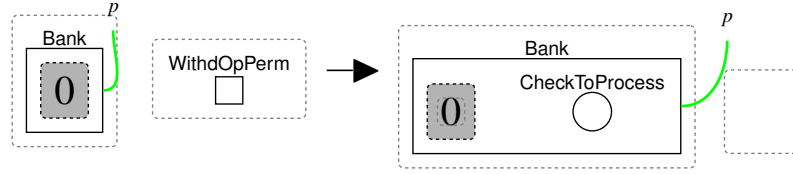


Figure 6.37: `recheckCon`: initialising the process of rechecking consent after withdrawing optional permissions.

6.5 Reusability: Modelling a Home Healthcare System

To demonstrate our approach is reusable and applicable to a variety of examples and regulations, we model a second example: a cloud-based home healthcare system. Instead of the GDPR, we apply the California Consumers Privacy Act (CCPA). The privacy aspects of this example are derived from the privacy policies of the Fitbit app for users in California [42]. Here, a patient uses Fitbit app that records their physical activity level (`ActInfo`). The data is stored on, and advice is generated on, a cloud server. Patient information can be shared with a third-party to inform studies in population health [47]: the statistics department (`StatDep`) at the National Regional Authority (NRA).

Modelling this system requires creating new system-specific entities⁴, e.g. NRA, Server, Patient, linking them to the appropriate agent and data types, and specifying new privacy policies for the DGE. An example initial state is in Fig. 6.38 where we use colours to differentiate the types of linking.

The Server is the data processor in this system and DB is its component. The third-party is NRA and `statDep` is its agent. DB is an authorised agent, while `statDep` is unauthorised. The DB can store the Patient’s Name and `ActInfo`.

End-users can amend the privacy framework to model the CCPA by removing unnecessary privacy entities. For example, since the CCPA does not specify requirements for cross-border data transfers, there is no need to use the Locations perspective, as shown in Fig. 6.38.

Alongside defining the initial state, they should use the privacy rules outlined in Section 6.3. To adapt the framework to their needs, the end-users can use only the privacy rules they require, discard the others, or adjust the priority classes of these rules. To illustrate this, the CCPA has the notion of *notice at collection*, which assumes users agree to the privacy policy by default,

⁴Similar to Monzo’s privacy policy, Fitbit’s privacy policies do not include technical details. As such, we adapt the system’s aspects from [34].

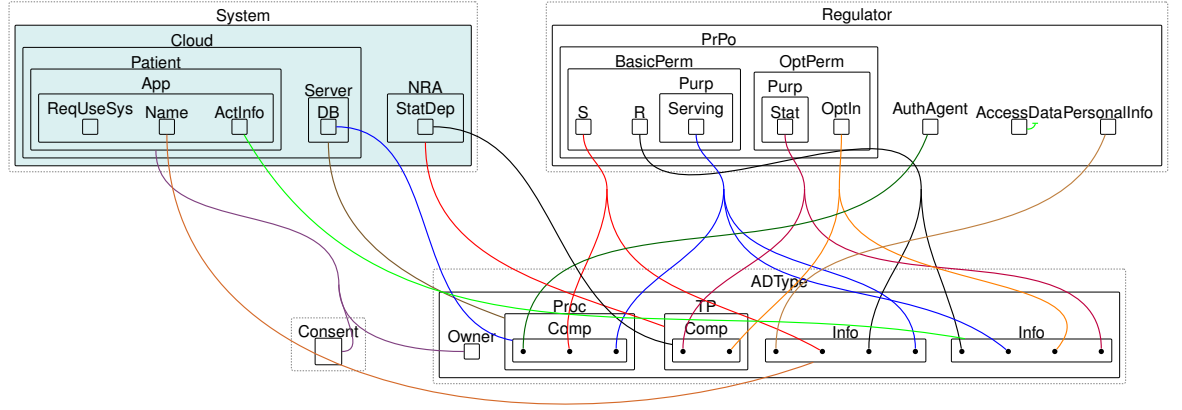


Figure 6.38: The initial state of the cloud-based home healthcare system.

but they are able to request that their data not be shared at any time⁵. To model this notion, we use rule `sendPolicy` (Fig. 6.8) once the user asks to use the system. Rule `acceptAll` (Fig. 6.9) is used by default, *e.g.* instantaneous rule, a higher priority rule, allowing it to be applied immediately and invisibly within the system. Rule `acceptBasic` (Fig. 6.10) and `confirm` (Fig. 6.16) can be discarded since the users cannot choose the policy they want to consent to, and consent confirmation is not a CCPA requirement.

Like the GDPR, the CCPA allows users to access their data at any time. After storing the patient's name, we use rule `rightToAccess` (Fig. 6.19) to explicitly match over `AccessData` in each system rule that represents a user's request to access their record as discussed in Section 6.4.

When the patient starts generating activities, the recorded information about their activities (`ActInfo`) is shared with `StatDep` if the user did not ask to stop sharing their data. Rule `shareActInfo` (Fig. 6.39) shares `ActInfo` with `StatDep`. It matches over `Comp` to ensure that the Patient did not ask data sharing to end.

Once the user asks to stop sharing their data, rules `withdrawReqOPT` (Fig. 6.23) and `processWithdOpt` (Fig. 6.24) should be applied. We also should use rule `closeLinks` (Fig. 6.11) (but we change the purpose to `Stat` as shown in Fig. 6.40.) to terminate the sharing permissions. The consent in this case should be rechecked to change the components' types of the third-party to `Comp_F`. We define a system rule shown in Fig. 6.41 that explicitly matches

⁵Since Fitbit does not explicitly state whether they employ notice at collection or obtain consent, we assume they adhere to the notice at collection practice as outlined in the CCPA.

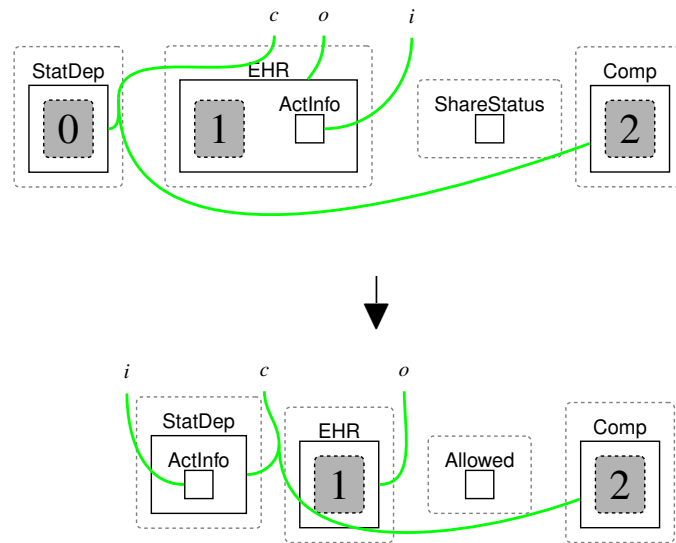


Figure 6.39: `shareActInfo`: sharing the patient's activities information with the statistical department at the NRA.

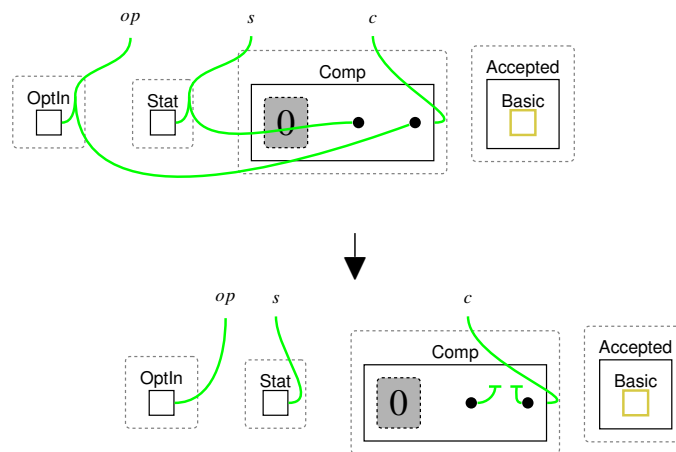


Figure 6.40: `closeLinks`: remove links between rejected permissions when the patient requests to opt out.

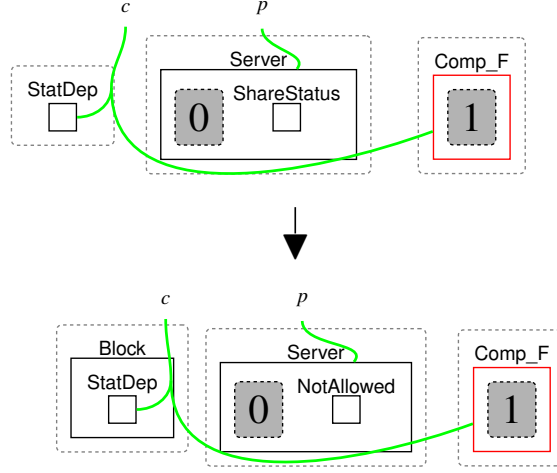


Figure 6.41: `prevshrActInfo`: prevent sharing the patient's activities information with the statistical department at the NRA.

over `Comp_F` to prevent sharing `ActInfo` with `StatDep`.

We can also use rule `updateCons` (Fig. 6.12) and rule `relinkPerm` (Fig. 6.13) to enable users to update their decisions after withdrawing optional permissions. In this case, we need to reset the type of `Comp_F` to `Comp`.

To model the notion of *right to delete* [126], we use rule `withdrawReq` (Fig. 6.20) and `processWithd` (Fig. 6.21). After applying these two rules, the entity `DeleInfo` is generated, which we use to define a system rule that deletes the Patient's electronic health record (EHR) as shown in Fig. 6.42.

6.6 Discussion

The proposed framework generates a model that captures the privacy concepts we consider in this thesis. However, the interaction between our privacy model and the system model defined by developers could generate bugs affecting privacy, such as those produced by conflicts between the privacy and system rules. Conflicts may arise due to improper handling of priority classes. For instance, if we assign a lower priority to rule `closeLinks` (Fig. 6.11) than to rules `confirm` (Fig. 6.16) and `registerRequest` (Fig. 6.25), this could result in rule `registerRequest` being executed after `confirm`, causing `closeLinks` to be skipped and not executed.

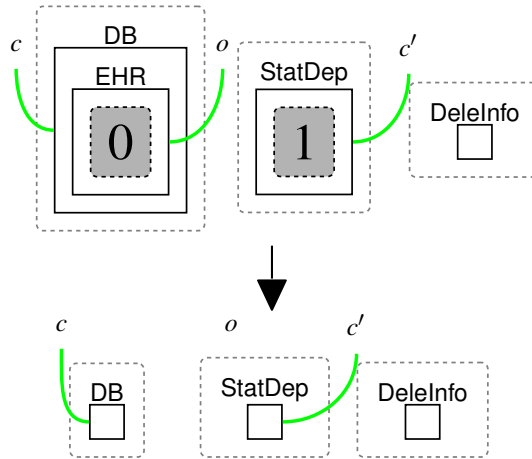


Figure 6.42: `deleteInfoEHR`: deleting the patient's information after withdrawing the consent.

Linking system entities to permissions for defining initial states is challenging, as it may result in mislinked permissions. For example, if we incorrectly link `Ad` and `OptIn` in Fig. 6.7 with a `Comp` other than the one linked to `MainBran`, `closeLinks` will close the link of the wrong `Comp`. This leads to sharing data with `MainBran`, even if the user has rejected sharing their data as the `Comp` linked to `MainBran` is not changed to `Comp_F`.

The developers need to use PRISM to *automatically* detect these bugs, which can then be fixed accordingly. To proactively address these potential issues, we have predefined a set of privacy properties for end-users to use directly, eliminating the need for them to define these properties themselves. This not only simplifies the process but also significantly reduces the risk of errors by the developers when defining privacy properties, ensuring the accuracy of the model. We will discuss this in depth in Chapter 7.

The model does not capture the case where a user is allowed to selectively accept or reject individual optional permissions when multiple optional permissions and purposes are present. However, the framework remains flexible, as end-users can define rules that enable data owners to selectively accept or reject individual optional permissions by tagging and recording the accepted ones in the consent perspective, while rejecting the others. Similarly, rules can be defined to allow users to select which optional permissions they wish to withdraw.

6.7 Summary

We presented a bigraph-based framework for modelling key privacy principles, including providing/withdrawing consent, purpose limitation, the right to access, and data sharing with third parties—all of which are fundamental notions in the GDPR and the CCPA.

The framework is applied to two examples: a bank notification system, modelled according to GDPR requirements, and a cloud-based home healthcare system, aligned with the CCPA. We also demonstrated how we incorporate the cross-border data transfer model introduced in [Chapter 5](#) into the framework, enabling spatially-aware privacy modelling.

In [Section 6.1](#), we presented the model's entities. [Section 6.2](#) provides an illustrative example based on the bank notification system scenario, demonstrating how these entities are linked. The dynamic behaviour of the model is described in [Section 6.3](#), while [Section 6.4](#) explains how system-specific entities and rules are integrated with the privacy model. Finally, [Section 6.5](#) demonstrates the reusability of the framework by modelling the CCPA through a cloud-based home healthcare system.

Chapter 7

Verification

The aim of defining the privacy models in [Chapters 5](#) and [6](#) is to conduct the *verification* step, proving that they comply with the privacy principles considered in this thesis.

We perform two types of analysis: (1) static analysis, which includes sorting and inductive reasoning, to show that certain states are unlikely to exist (or may not exist) in any model, and (2) model checking, which analyses our framework’s configurations to verify whether they satisfy privacy regulations in all possible executions.

Verification proceeds in a static-to-dynamic order. We begin with sorting (a typing discipline) to reject ill-formed initial states and rule applications. We then use inductive reasoning to prove the correctness of the reaction rules. After these pass, we generate the labelled transition system using BigraphER and perform CTL model checking.

The chapter is structured as follows. We begin with the static analysis of both models, *i.e.* the cross-border data transfers model and the privacy regulations model, via sorting ([Section 7.1.1](#)) and inductive reasoning ([Section 7.1.2](#)). We then present the model checking analysis ([Section 7.2](#)) for each model ([Sections 7.2.1](#) and [7.2.2](#)). In [Section 7.3](#), we illustrate how privacy violations can be detected through examples. We discuss the chapter’s key insights in [Section 7.4](#), and conclude with a summary in [Section 7.5](#).

7.1 Static Analysis

Static analysis refers to proving the correctness of the framework without executing the model. In this section we present two types of static analysis: *sorting* and *inductive reasoning*.

7.1.1 Sorting

We define sort schemes for the cross-border data transfers model and the privacy regulation model to preserve their integrity and to prevent ill-structured configurations.

Cross-border Data Transfers Model

We present the defined sorts for the cross-border data transfers model (Figs. 5.1 and 5.11). Specifically, we define sorts for each perspective of the model—the SRTYPE perspective, the Locations perspective, and the WhatsSystem perspective—as well as for entities that are generated through privacy reaction rules.

SRTYPE Perspective. We define the following sorts to ensure a well-formed perspective:

$$\begin{aligned}
 &\text{sort } a, \text{ sort } p, \text{ sort } et, \text{ sort } tp, \text{ sort } sy \\
 &\text{sort } ety = \text{Cont}\{a \rightarrow (p + tp) \times sy + et\} \\
 &\quad | \text{Proc}\{a \rightarrow (p + tp) \times sy + et\} \\
 &\text{sort } ent = \text{SType}\{et \rightarrow (p + tp) \times a \times sy\} ety \\
 &\quad | \text{RType}\{et \rightarrow (p + tp) \times a \times sy\} ety \\
 &\text{sort } srt = \text{SRTYPE } ety \times ety^*
 \end{aligned}$$

Figure 7.1: Sorts for SRTYPE perspective.

The entities Cont and Proc, both having the sort ety, each have a port of sort a. This port is always linked to:

- A port of sort p (for an untagged pointer) or tp (for a tagged pointer) along with a system entity via a port of sort sy,
- Or to a port of sort et on the entity SType or RType (Figs. 5.15 and 5.17).

The sort ent represents the sort of SType and RType. As these entities are used to specify the type of the sender and receiver (whether Cont or Proc), each must include at least one child of sort ety. Both SType and RType are linked to three ports: one on a tagged or untagged pointer, one on the entity type, and one on the system entity. The entity SRTYPE, with sort

`srt`, acts as the parent of `Cont` and `Proc`. This imposes a structural constraint where `SRTyp` contains one or more entities of sort `ety`, as illustrated in Figures 5.1 and 5.11.

By defining and applying these sorts, we ensure that `site 0`, nested within `SType` and `RType` in rule `checkSType` and `checkRType` (Fig. 5.2), either represents the entity `Cont` or `Proc`. We also ensure that `SType` and `RType` are linked to a pointer, with the other end of the link connected to a port on the system entity.

Locations Perspective. Similar to the `SRTyp` perspective, we define the following sorts for the entities of the Locations perspectives:

```

sort ce, sort i, sort t, sort p, sort st, sort d
sort cr = C(1) | C(2) | C(3)
sort tcr = C'(1) | C'(2) | C'(3)
sort e = ExpiryDate
sort g = Greater e
sort scss = SCCs
sort ade = Adeq{d → p × p*}
sort pnt = P{p → (sy × a) + d + et}
           | P'{tp → (sy × a) + et}
           | Ps{tp → (sy × a) + et}
sort scm = Scheme{sc → ce × ce*} cr × cr*
sort certf = Cert{ce → sc} (cr* + tcr*) × e
            | InvalidCert{ce → 1} tcr* × e
            | CompliantCert{ce → sc} (tcr × tcr*) × e
sort ctr = Contract{t → t × t*} scss + 1
sort inctr = InvalidContract{i → 1}
sort l = L(Ireland) (pnt* × pnt) × (ade + sc + ctr)
        | L(US) (pnt* × pnt)
        | L(Mexico) (pnt* × pnt) × (ctr + inctr)
        | L(Singapore) (pnt* × pnt) × (ctr + inctr)
        | L(Dubai) (pnt* × pnt) × certf
        | L(China) (pnt* × pnt) × certf
sort loc = Locations l* × l

```

Figure 7.2: Sorts for Locations perspective.

We assign the sort `pnt` to the pointers `P`, `P'`, and `Ps`, which represent the solid black, blue, and yellow bullets, respectively. Pointer `P` has a port of sort `p`, while pointers `P'` and `Ps` have

ports of sort *tp*. These pointers are linked to two ports: one on a system entity (with sort *sy*) and one on a corresponding entity type within the *SRT* perspective via a port of sort *a*. These pointers can also be linked to *S*Type or *R*Type through a port of sort *et* (Figs. 5.15, 5.17 and 5.19). Additionally, pointer *P* can be linked to a port of sort *d* on the entity *Adeq*, whereas pointers *P'* and *Ps* should not be linked to ports of this sort.

The entity *Scheme* is linked to one or more certifications (*Cert*) and must always include at least one criterion of sort *cr* (untagged criteria). The certifications, whether invalid or compliant, are assigned the sort *certf*. Both *Cert* and *CompliantCert* should be linked to a port on the entity *Scheme*, and both have a child of sort *e*. However, *Cert* may contain zero or more untagged or tagged criteria (*tcr*), while *CompliantCert* must always contain at least one *tagged* criterion to satisfy the criteria specified by the *Scheme*. In contrast, *InvalidCert* can include zero or more tagged criteria. As shown in Fig. 5.7 and Fig. 5.9, *InvalidCert* has a closed link, denoted by 1 in the defined sorts, indicating the absence of a connection to *Scheme*.

The sort of *Contract* is *ctr*. Since *Contract* is linked to one or more contracts, the output and input ports share the same sort (*t*). *Contract* may either have a child of sort *scss*, or it may be empty, in which case it can be tagged as *InvalidContract* by rule *checkingSCCs* (Fig. 5.5), which also closes its link.

The parameterised entity *L* is defined with the sort *l*. The sorts of its children vary based on its parameters. For example, *L(Ireland)* is the sender country and must contain a child of sort *ade*, *scm*, or *ctr* to allow the GDPR compliance check.

Receiver countries that use SSCs, *e.g.* *L(Mexico)* and *L(Singapore)*, must include the entity *Contract* (sort *ctr*) or *InvalidContract* (sort *inctr*) if the contract is empty, *i.e.* it does not contain SSCs. In contrast, countries like *L(Dubai)* and *L(China)* use a certification mechanism, so their children must be of sort *certf*. If the receiver country is adequate, it does not contain any safeguards, meaning it includes only tagged or untagged pointers.

Regardless of the value of the parameter for entity *L*, it must always contain at least one pointer (tagged or untagged) that links to system entities and their types. The parent of *L* is the entity *Locations* that has sort *loc* and it should contain at least one child of sort *l*.

WhatsSystem Perspective. The system consists of various entities such as *DC*, *FB*, and others, each of which has specific roles and children entities based on their functions in the data exchange process:

```

sort snd = SpecifyReceivers | CheckReg | SameRegion | Adequate
          | CollectInfo | StartTransfer
sort req = AskForData | WithdReq
sort info = CopyInfo
sort cd = CurrentDate
sort chex = CheckExp cd
sort syse = DC{sy → (p + tp) × a + et} snd + req + info + chex + 1
          | FB{sy → (p + tp) × a + et} req + info + 1
sort b = Block syse
sort w = WhatsSystem syse × syse*

```

Figure 7.3: Sorts for WhatsSystem perspective.

DC and FB are connected by a hyper-edge to a tagged or untagged pointer in the Locations perspective, as well as to their corresponding types, *e.g.* Cont. When the entities SType and RType are generated, they can be linked to DC or FB via a port of sort et. In our model, DC can act as both a sender and a receiver. When it is a sender, it contains the following entities: SpecifyReceivers, CheckReg, SameRegion, Adequate, CollectInfo or StartTransfer. When DC is a receiver, it can contain AskForData or WithdReq. FB and Other companies in the model, *e.g.* Whats, Meta *etc.*, contains children such as AskForData or CopyInfo, which are specifically related to their role in requesting and receiving data (see [Section 5.3](#)). The entity Block is used to block companies from accessing the data, thereby always containing one child of sort syse. WhatsSystem has sort w and always has at least one child of sort syse.

Privacy Regulation Model

Here, we define sorts for the privacy regulations model presented in [Figs. 6.7](#) and [6.38](#). Specifically, we define sorts for each privacy perspective (DGE, ADTypes, and Consent), as well as for the system perspectives in both our banking and healthcare examples. We also sort entities that are generated through the dynamic evolution of the system via privacy rules.

DGE Perspective. We define sorts to govern the structure of entities and links in the DGE perspective as follows:

```

sort s, sort u, sort r, sort po, sort o, sort se, sort ad, sort a, sort pe, sort ac
sort ow, sort us, sort re, sort c, sort st
sort prm = S{s → po × po*} | U{u → po × po*} | R{r → po × po*}
           | OptIn{o → po × po*}
sort pur = Serving{se → po × po*} | Ad{ad → po × po*} | Stat{st → po × po*}
sort prp = Purp pur × pur*
sort prmc = BasicPerm (prm × prm*) × prp
            | OptPerm (prm × prm*) × prp
sort cnf = Confirmed | ConfirmWithd
sort acpty = All | Basic
sort dcs = Accepted acpty | Rejected
sort prpo = PrPo (prmc × prmc*) + (dcs + cnf)
sort rst = AuthAgent{au → po × po*} | PersonalInfo{pe → po × po*}
sort acd = AccessData{ac → (ow × us × re × c) + 1}
sort dge = DGE prpo × rst × acd

```

Figure 7.4: Sorts for DGE Perspective.

Fig. 7.4 presents the sorts defined to sort DGE perspective. We assign the sort `prm` to the entities `S`, `U`, `R`, and `OptIn`. Each of these entities has one port whose sort is `s`, `u`, `r`, and `o`, respectively. Each port must connect to at least one port on the pointer entity (`P`)¹.

The purposes `Serving`, `Ad`, and `Stat` have the sort `pur`. Each of these purpose entities has exactly one port of sort `se`, `ad`, and `st`. Similar to permissions, each purpose's port should link to one or more pointer entities. The entity `Purp` has sort `prp` and serves as a parent for one or more purposes.

¹The sort of the pointer here is different from the one defined in the cross-border data transfers model (Fig. 7.2).

The sort `prmc` represents the categories of the permissions, namely `BasicPerm` and `OptPerm`. Since these entities contain permissions and the entity `Purp`, as shown in [Figs. 6.7](#) and [6.38](#), each category always has at least one child of sort `prm` and one child of sort `prp`.

Because `BasicPerm` and `OptPerm` are nested within `PrPo`, the entity `PrPo` must have at least one child of sort `prmc`. `PrPo` may also have children of sorts `dcs` or `cnf`, which represent entities generated through the system's dynamic evolution via reaction rules ([Figs. 6.9, 6.10, 6.14, 6.16](#) and [6.22](#)).

The entities `AuthAgent` and `PersonalInfo` have the sort `rst`. Each such entity has one port of sort `au` and `pe`, which must link to at least one pointer. Conversely, the entity `AccessData` has a single link that can either be closed or connected to ports of sorts `ow`, `us`, `re`, and `c`, corresponding to the ports on `Owner` ([Fig. 6.19](#)), `User/Patient`, `Record/EHR` ([Fig. 6.27](#)) and `Consent`, respectively.

Finally, the top-level entity in this perspective is `DGE`, which has sort `dge` and must contain three children of sorts `prpo`, `rst`, and `acd`.

```

sort a, sort e, sort i, sort d, sort cm, sort sy
sort req = Update | WithdrawAll | WithdReqOpt
sort func = checkPolicy | RightToAccess | DeleInfo
sort apr = ConsApproved
sort ds = Owner{ow → (us × c) + re + ac} prpo + apr + req + 1
sort pntr = P{po → (s + u + r + o) × (se + ad + st) + au + pe}
sort cpnt = P{po → 1}
sort cmp = Comp{cm → e × (p + tp + 1)}(pntr × pntr) × pntr*
           | Comp_F{cm → e × (p + tp + 1)} cpnt × cpnt*
sort ety = Proc{a → sy × (p + tp + 1)}(cmp × cmp*) + func
           | TP{a → sy × (p + tp + 1)} cmp × cmp*
sort ent = SType{et → (p + tp) × a × sy} ety
           | RType{et → (p + tp) × a × sy} ety
sort dty = Info{i → d}(pntr × pntr) × pntr*
sort adt = ADTypes (ety × ety*) × (dty × dty*)

```

Figure 7.5: Sorts for ADTypes perspective.

ADTypes Perspective We sort ADTypes perspective as shown in Fig. 7.5. The user requests that are modelled via the entities Update, WithdReqOpt and WithdrawAll are assigned to the sort req. The sort func specifies functions within the system, such as checkPolicy, RightToAccess and DeleInfo .

We assign the sort ds to the entity Owner, which has a single port of sort ow. This port links to ports of sort us and c, and it may also be linked to ports of sort ac or re. These target ports correspond to User/Patient, Consent, AccessData and Record/EHR. Internally, Owner can be empty or contain exactly one entity of sorts prpo (see the defined sorts for the DGE perspective in Fig. 7.4), apr and req. These sorts represent the privacy policy, the approval state of the consent and user's requests, respectively (Figs. 6.8, 6.12, 6.16, 6.20 and 6.23).

We define two sorts to handle pointer structures. The sort pntr is assigned to the entity P, which has a port of sort po. This port must link to either a port of sort (s + u + r + o) on a

permission entity (*e.g.* S, U, OptIn) and a port of sort (se + ad + st) on a purpose entity (*e.g.* Serving, Ad, Stat). The port po can also link to a port of sort au or pe, which belong to the entities AuthAgent and PersonallInfo, respectively.

Inactive or closed pointers are given the sort cpnt, which corresponds to the same P entity but with a closed link. This indicates that the pointer is disconnected from the permission and purpose entity.

The sort cmp represents the entity Comp, which has a port of sort cm connected to an agent's component (*e.g.* DB) through a port of sort e. Comp must contain at least two pnt_r-sorted pointers—one linked to a permission and one to a purpose. Conversely, the entity Comp_F must contain at least one pointer with a closed link (Fig. 6.18).

The entities Proc and TP are both assigned the sort ety. Each of them has a port of sort a, which is linked to a system agent, *e.g.* Bank and NRA, via a port of sort sy. Both entities Proc and TP contain at least one child of sort cmp. Proc can also have a child of sort func that represents functions the processor performs, *e.g.* checkPolicy, RightToAccess, and DeleteInfo.

To capture location-aware behaviour for GDPR-compliant data transfers, Comp, Proc, and TP may also connect to a pointer of sort pnt (defined in Fig. 7.2), either through a port of sort p or a port of sort tp. We also reuse the same sorts defined for SType and RType in Fig. 7.1. These entities are used when modelling GDPR requirements related to international data transfers.

The entity Info has sort dty and it connects to data in the system perspective via a port of sort d. Info must contain at least two pnt_r-sorted pointers—one linked to a permission and one to a purpose.

ADType is a top-level entity that has sort of adt and has one or more children of sort ety and dty.

```
sort wthcon = Withdrawn
sort cons = Consent{c → (ow × us) + re + ac} prpo + wthcon + 1
```

Figure 7.6: Sorts for Consent perspective.

Consent Perspective Consent perspective has the sorts presented in Fig. 7.6. The sort cons is assigned to the entity Consent. This entity has a single port of sort c, which must

be connected to one of the following: a combination of *ow* and *us*, or ports of sort *re* or *ac*. These ports represent the Owner, User, Record, and AccessData, respectively. Internally, the entity *Consent* must contain a child of sort *prpo*, which represents the privacy policy at the time the consent is created. It may also contain a child of sort *wthcon*, representing a withdrawal of consent or it may be empty.

```

sort ureq = ReqUseSys | ToUpdate | ToStore
sort pdata = Name{d → i}
sort gdata = TransInfo{d → i}
sort oper = CheckToProcess | AllowServ | TransToTP
           | SpecifyRT | TempStorage pdata
sort upd = NeedUpdate pdata
sort reco = Record{re → ow × c × ac × us} pdata + gdata + upd
sort ap = App (pdata × gdata) + ureq
sort usr = User{us → (ow × c) + re + ac} ap
sort sban = DB{e → cm × cm + (p + tp)} reco
           | Notifier{e → cm × cm + (p + tp)} gdata + 1
sort subad = Marketing{e → cm × cm + (p + tp)} gdata + 1
            | MainBran{e → cm × cm + (p + tp)} gdata + 1
sort agent = Bank{sy → a × a + (p + tp)} (sban × sban*) + oper*
            | AdCompany{sy → a × a + (p + tp)} subad
sort sys = System usr × agent

```

Figure 7.7: Sorts for System perspective in the banking example.

System Perspective (Banking Example) Fig. 7.7 shows the defined sorts for the System perspective in the banking example (Fig. 6.7). In this example, the top-level system is

represented by the sort *sys*, defined as the product $\text{usr} \times \text{agent}$. This captures a system composed of a user and a data-handling agent (*e.g.* a bank or an ad company).

The entities *Name* and *TransInfo* are assigned the sorts *pdata* and *gdata*, each having a port of sort *d* linked to a port of sort *i* on *Info* in ADTypes perspective.

The sort *usr* is assigned to the entity *User*, which has a port of sort *us* that must be linked to ports of sort *ow* and *c*, or to ports of sort *re* or *ac*. These links associate the user with the Owner, Consent, Record and AccessData, respectively. Internally, *User* contains an entity of sort *ap*, which represents the bank application (App) containing either personal and transactional data or the user request (sort *ureq*), *i.e.* *ReqUseSys*, *ToUpdate* and *ToStore*.

The entity *Record*, with sort *reco*, must be linked to the Owner, Consent, AccessData and User. It includes both personal and transaction data, and it may also contain an update indicator of sort *upd*, modelled by the entity *NeedUpdate* (Fig. 6.27).

The sort *oper* defines system operations that take the form of entities such as *CheckToProcess* (Fig. 6.25), *AllowServ* (Fig. A.1), *TransToTP* (Fig. 6.30), *SpecifyRT* (Fig. 6.30), or *TempStorage* (Fig. 6.25). The last of these also requires a child of sort *pdata*, representing the temporary storage of a user's name in order to check consent before storing it to the database.

The sort *sban* is assigned to *DB* and *Notifier*, which each have a port of sort *e* linked to a port of sort *cm* on *Comp*. *DB* contains *Record*, while a *Notifier* can be empty or hold transactional data. Similarly, *subad* represents *Marketing* and *MainBran*, which may contain transaction data or remain empty.

The sort *agent* includes either a *Bank* or an *AdCompany*. A *Bank* has a port of sort *sy*, which is linked to its type in the ADTypes perspective (*e.g.* *Proc*) via a port of sort *a*. It also contains one or more children of sort *sban* and may include zero or more system operations of sort *oper*. An *AdCompany*, on the other hand, also links from *sy* to a but contains only a single component of sort *subad*.

When modelling entities' locations to represent GDPR requirements for data transfers, *Bank*, *Marketing*, *MainBran* and *AdCompany* can also be linked to a pointer of sort *pnt* as introduced in Fig. 7.2.

```

sort preq = ReqUseSys
sort sopr = ShareStatus | Allowed | NotAllowed
sort pInfo = Name{d → i}
sort gInfo = ActInfo{d → i}
sort ehr = EHR{re → ow × c × ac × us} pInfo × gInfo
sort ap = App pdata × gInfo × preq
sort pa = Patient{us → (ow × c) + re + ac} ap
sort db = DB{e → cm} ehr
sort cldagt = Server{sy → a} db × sopr*
sort stdep = StatDep{e → cm} gInfo*
sort sysagt = NRA{sy → a} stdep
sort cld = Cloud cldagt × pa
sort sytm = System cld × sysagt

```

Figure 7.8: Sorts for System perspective in the healthcare example.

System Perspective (Healthcare Example) The sorts of the healthcare example (Fig. 6.38) is defined as shown Fig. 7.8. The system is defined by the sort `sytm`, composed of the cloud's entities and system's agent, represented as the product `cld × sysagt`. The cloud component, of sort `cld`, includes both a server (sort `cldagt`) and a patient (sort `pa`).

The sort `pa` is assigned to the entity Patient, which has a port of sort `us` linked to ports of sort `ow` and `c`, and optionally to `re` and `ac`. These ports are located on Owner, Consent, EHR, and AccessData, respectively. Internally, it contains an application entity (sort `ap`), which includes personal and activity data (sorts `pInfo` and `gInfo`) and a request of sort `preq` (e.g. `ReqUseSys`).

The entity EHR, with sort `ehr`, models an electronic health record. It links via a port of sort `re` to Owner, Consent, AccessData and Patient. EHR contains children of sorts `pInfo` and `gInfo`, representing the patient's name and activity details, respectively.

The sort `db` is assigned to the DB entity, which connects to Comp in ADTypes perspective via port `cm` and contains an EHR record. The sort `cldagt` is used for the Server, which

includes a port of sort *sy* linked to its type in ADTypes perspective via port *a*. It holds a child of sort *db* and a collection of operations, represented by *sopr*^{*}. The sort *sopr* defines the server's operations and includes the actions *ShareStatus*, *Allowed* or *NotAllowed*.

The entity *NRA* is assigned the sort *sysagt*. It connects via a port of sort *sy* to its type in the ADTypes perspective (*TP*) and contains the statistical department *StatDep*, which is assigned the sort *stdep*. Since *StatDep* is a component of *NRA*, it links to the *Comp* entity via a port of sort *cm* and may include zero or more *gInfo*-sorted children representing the patient's activity data.

7.1.2 Inductive Reasoning

As we generalise the framework, we apply inductive reasoning to prove the correctness of the reaction rules², thereby helping us ensure that the desired properties are preserved, even in the presence of state space explosion. Inductive reasoning proves that if a property holds for one state, it will continue to hold for all subsequent states, ensuring the property remains true throughout the system's execution. Meanwhile, invariant reasoning identifies properties that remain unchanged throughout the system's execution.

The correct application of the rules is tightly coupled to the priority classes that enforce the proper ordering of the rules. This means the analysis needs repeated for different regulations as these might change the rule priorities (Section 6.5). We use GDPR as an illustrative example.

The following analyses are proof sketches; a full formal proof would require more detailed examination, including all possible interleavings and conflict analysis.

In this section, we present the analysis of the reaction rules for both the cross-border data transfers model and the privacy regulations model.

Cross-border Data Transfers Model

- **Property 1:** No data transfer if the country is not adequate:

We can verify this property by examining the initial state and the reaction rule statically.

If a recipient country has no pointer linked to the *Adeq* entity and does not contain either *Contract* or *Cert*, then it is considered non-adequate and lacks safeguards for

²Currently there is no tool support for this and it must be done by hand.

lawful data transfer. In such cases, the data must not be transferred to that country. This is confirmed by inspecting the right-hand side of the reaction rules. If the non-adequate country appears on the left-hand side, then on the right-hand side, no data entity should be nested within that country. Equivalently, if a rule produces data transfer on its right-hand side, then its left-hand side must not contain a non-adequate country. This acts as a bigraphical invariant [50], *i.e.* a structural property that must be preserved across all rule applications to prevent invalid data transfers.

- **Property 2:** `invalidContra` does not lead to `dataTransfer`:

By inspecting the initial state, if a country contains a `Contract` entity but this `Contract` does not contain SCCs, this implies that data must not be shared with any company in that country. For example, this applies to DC in Singapore (see Fig. 5.11). We can verify this property by inspecting the right-hand side of the reaction rules to ensure that the company does not gain access to the user's data.

- **Property 3:** `invalidCert` does not lead to `dataTransfer`:

Similar to **Property 2**, by examining the initial state, we can identify a country that uses an invalid certification, *i.e.* a `Cert` entity that does not contain three criteria such as the certification used by China in Fig. 5.11. In such cases, we verify the reaction rule by inspecting its right-hand side to ensure that the data is not nested within the company or branch located in that country.

- **Property 4:** `withdrawCert` does not lead to `dataTransfer`:

Here we check the left-hand side of the reaction rules. If the entity `WithdrawnCert` appears on the left-hand side, then the data must not be transferred on the right-hand side to the country that contains the `WithdrawnCert` entity.

- **Property 5:** expired certification does not lead to `dataTransfer`:

Unlike **Property 3**, we cannot deduce that the certification is expired by inspecting the initial state. Therefore, to verify this property, we need to examine the reaction rules. If `InvalidCert` appears on the left-hand side of a reaction rule, then data must not be transferred on the right-hand side to the country that contains this `InvalidCert`.

Privacy Regulation Model

- **Property 1:** `rejAll` does not lead to confirmation:

Rule `rejectAll` (Fig. 6.14) does not lead to the confirmation of the consent. When rule `rejectAll` (Fig. 6.14) is applied, it generates the entity `Rejected`. Rule `confirm` (Fig. 6.16) requires the entity `Accepted` to be present on its left-hand side to be applicable. Since rule `rejectAll` (Fig. 6.14) produces the `Rejected` entity instead of the `Accepted` entity, the left-hand side of rule `confirm` (Fig. 6.16) cannot be matched. As a result, the system state that results from applying rule `rejectAll` (Fig. 6.14) cannot transition to a state where consent is confirmed. This ensures that the rejection of the consent does not eventually lead to its confirmation.

- **Property 2:** No `shareInfo` if `rejectOpt`:

This can be verified by visually examining the reaction rules. If `Comp_F` appears on the left-side of a rule, the user's data must not be nested within the agent linked to `Comp_F` on the right-side of the rule. We can also use predicates to automatically verify this by searching for `Comp_F` and checking if the agent linked to it contains the user's data.

- **Property 3:** `invalidContra` does not lead to `shareInfo`:

By inspecting the initial state in Fig. 6.7, we can deduce that the `Contract` nested within `China` is invalid because it does not contain SCCs. This implies that data should not be shared with the agent linked to a pointer nested within `China`, *i.e.* `Marketing`. We can verify this by examining the right-hand side of the rules to ensure that `Marketing` does not access the user's data, *e.g.* `prevMDSCCs` (Fig. 6.34).

- **Property 4:** No `unauthAccess`:

We can also detect unauthorised access by inspecting the initial state and the right-hand side of the rules. By checking the initial state, we confirm that `Notifier` is an unauthorised agent, as the `Comp` linked to it does not linked to `AuthAgent`. Similarly, `Name` is identified as personal information. We can prove that there is no unauthorised access because `Name` is not nested within `Notifier`, `MainBran` or `Marketing` on the right-hand side of the rules.

- **Property 5:** `withdAll` leads to `deletingInfo`:

`DeleInfo` is generated by rule `processWithd` (Fig. 6.21). This means whenever

DeleInfo appears on the left-hand side of a rule, it indicates that consent has been withdrawn. Consequently, the user’s data should not appear on the right-hand side of the rule. The right-hand side should not also contain any sites that might hold the user’s data, ensuring complete removal of the data post-withdrawal.

7.2 Model Checking

Once we have a formal model of privacy we can utilise it to perform *model checking verification*: checking the system, and its privacy policies, provably adheres to a specific regulation.

First, we create a transition system representing all possible updates the system can make. Given an initial state of the system, the transition system is *automatically* generated using BigraphER, with bigraphs representing states, and reaction rule applications representing transitions.

Manually inspecting the resulting transition system to prove privacy properties in our WhatsApp, banking and healthcare examples is challenging due to the large state spaces: 193 states in the WhatsApp example, 464 states in the banking example and 182 states in the healthcare example. To manage this complexity, we use the PRISM [85] model checker as it is supported natively by BigraphER.

Although PRISM supports probabilistic properties, our privacy specifications—based on the regulations—are written in the non-probabilistic fragment of PRISM’s property specification language which subsumes logics like CTL [31]. We use the CTL-like elements, and introduce the syntax/semantics as they are used.

Finally, to allow labelling of states, used within the logical specifications, *bigraph patterns* (that act like left-hand side matches in rewriting) are added to our model. Since the predicates are explained in [Section 4.5.9](#) and are trivial, we do not present them in this chapter, but they are available in the full model file [5]. Importantly, privacy predicates are predefined, *i.e.* they represent bigraph patterns of the right-hand side of the privacy rules. However, adjustments are needed for the predicates related to the system’s perspective, which are highlighted in teal in this section.

7.2.1 Cross-border Data Transfers Model

This section presents the model checking analysis of the cross-border data transfers model. We formalise the GDPR requirements for international data transfers (adequacy decisions, SCCs, and certification mechanism) using CTL to prove that these requirements hold in the model of the WhatsApp example.

The first requirement that we need to check is the adequacy requirement: *no data transfer if the country is inadequate*. The CTL syntax of this requirement is:

$$\mathbf{A}[\mathbf{G}(\neg \text{adequateCountry} \implies \neg \text{dataTransfer})] \quad (7.1)$$

`adequateCountry` is a predicate that labels all states that contain the match of the right-hand side of rule `checkingAdeq` (Fig. 5.4). The second predicate is `dataTransfer`, which labels the states that transfer the data to the receiver via system rules. **A** is a path quantifier, meaning *for all paths*. **G** is a state quantifier and it means *for all states* (globally). **AG** means the property should hold for all paths globally throughout the entire transition system. Here, we check that: for all paths (**A**), if `adequateCountry` does not hold ($\neg$), `dataTransfer` should not hold ($\neg$) in all states along that path (**G**).

The second requirement that we need to check is: *no data transfer if the contract is invalid*:

$$\mathbf{A}[\mathbf{G}(\text{invalidContra} \implies (\mathbf{X} \neg \text{dataTransfer}))] \quad (7.2)$$

As previously mentioned, `invalidContra` is a predicate that holds if the Contract is invalid. In this case, we check: for all paths of the transition system, if `invalidContra` holds in a state, then `dataTransfer` should not hold in the immediate next state along that path. We use the next-state operator **X** to represent the system's sequential behaviour, where a contract is first tagged as invalid and data transfer is prevented in the next state as a result.

The third requirement is: *no data transfer if the certification is invalid*:

$$\mathbf{A}[\mathbf{G}(\text{invalidCert} \implies (\mathbf{X} \neg \text{dataTransfer}))] \quad (7.3)$$

This property states that if the certification is tagged as invalid, *e.g.* expired or not meeting the required criteria, then the data transfer must not occur in the immediately following state.

The last requirement is: *no transfer should be performed if the certification is withdrawn*:

$$\mathbf{A}[\mathbf{G}(\text{withdrawCert} \implies (\mathbf{X} \neg \text{dataTransfer}))] \quad (7.4)$$

Here, we prove that for all paths, if `withdrawCert` is true, then `dataTransfer` should be false in all subsequent states on that path. Since `withdrawCert` only becomes true when a system rule generates `WithdReq` as discussed in [Section 5.2.8](#), we apply the next-state operator **X** to ensure that `dataTransfer` is false starting from the state immediately following the state that holds `withdrawCert`.

We can observe that there is a difference in formulation between property [7.1](#) and the others. This difference is not arbitrary or implementation-driven but reflects how the GDPR treats these legal bases. Under the GDPR, adequacy (property [7.1](#)) is expressed as a standing prohibition: if a country is not adequate, transfers must never take place along any path. In practice, adequacy decisions can be revoked, which means adequacy may change over time. However, in our framework we model adequacy as a global condition because such changes are infrequent and the GDPR considers adequacy as a *precondition* for transfer. By contrast, contracts and certifications (properties [7.2](#), [7.3](#) and [7.4](#)) can change status during execution, so we use the next-state operator (**X**) to ensure that transfers stop immediately once they become invalid.

To make the link to the GDPR explicit, [Table 7.1](#) maps each CTL property to the corresponding GDPR article(s) and indicates our coverage, *i.e. Full or Partial*. All properties are marked *Partial* because they check the core system-level prohibition, *i.e.* that transfers do not occur without a valid legal basis, but do not cover non-technical requirements (*e.g.* documentation) or alternative transfer bases (*e.g.* BCRs or derogations)³.

7.2.2 Privacy Regulation Model

Here, we formalise the privacy principles considered in this thesis to demonstrate how model checking can be used to verify these principles within the banking and healthcare models.

We first check a system does not assume consent unless the user explicitly accepts the policies. In this case, it is easier to verify the inverse statement: consent is never confirmed if

³These alternatives are not included in the current model but could be incorporated by extending the reaction rules and entities to represent them.

Table 7.1: Mapping of the CTL properties to GDPR cross-border data transfer requirements. *Partial* denotes that the property captures the core transfer-prohibition condition but does not cover non-technical requirements in the Articles or alternative legal bases (*e.g.* Binding Corporate Rules (BCRs), Art. 47; derogations, Art. 49).

CTL property number	GDPR article(s)	Coverage
7.1	Art. 45 (Adequacy decision) [67]	Partial
7.2	Art. 46(2)(c,d) (Standard Contractual Clauses) [68]	Partial
7.3	Art. 42 [66]; 46(2)(f) [68] (Certification as safeguard)	Partial
7.4	Art. 42(7) (Withdrawal of certification) [66]	Partial

a user rejects the policies. We use the following CTL formula:

$$\mathbf{A}[\mathbf{G}(\text{rejAll} \implies \neg \text{confirmation})] \quad (7.5)$$

Where `rejAll` and `confirmation` are bigraph predicates that label states whenever the left-hand sides of rules `rejectAll` (Fig. 6.14) and `confirm` (Fig. 6.16) are matched. **A** is a path quantifier meaning *for all possible (future) paths*, while **G** is *globally*, i.e. the property must hold for all future states on the path.

A related property, is that we must ensure consent is given before we start any data processing. Expressed in CTL:

$$\mathbf{A}[(\neg \text{process}) \mathbf{W} \text{checkingCons}] \quad (7.6)$$

Here we use **W** which is the *weak until* operator. In our case, this means that `process` (store/copy *etc.*) should never ($\neg$) hold until after `checkingCons` (consent has been checked) holds. The weakness means that `checkingCons` does not need to hold at any point, e.g. if the user rejects the policies then it is still true that there was no processing done before consent was checked.

We also ensure that the system does not share the user's data if the user withdraws the optional permissions using the following formula:

$$\mathbf{A}[\mathbf{G}(\text{partWithd} \implies (\mathbf{X} \neg \text{shareInfo}))] \quad (7.7)$$

This property ensures that for all paths, if the user withdraws optional permissions (`partWithd`), then in the very next state (**X**), data is not shared with third parties ($\neg \text{shareInfo}$) as soon as the permission is withdrawn. The aim of using next operator **X** to ensure that once the permissions are withdrawn, the system instantly ceases any sharing in the very next state.

Similar to property 7.7, we check that optional processing is handled correctly and that the user's data is not shared if the Contract is invalid:

$$\mathbf{A}[\mathbf{G}(\text{rejectOpt} \implies (\mathbf{X} \neg \text{shareInfo}))] \quad (7.8)$$

$$\mathbf{A}[\mathbf{G}(\text{invalidContra} \implies (\mathbf{X} \neg \text{shareInfo}))] \quad (7.9)$$

We check that users who store information always have the right to access it by using the

following formula:

$$\mathbf{A}[\mathbf{G}(\text{storeInfo} \implies (\mathbf{F}\text{accessRight}))] \quad (7.10)$$

This formula uses an additional operator **F** that denotes *eventually*. This means `accessRight` does not need to immediately hold, *i.e.* the action that stores the info (`storeInfo`) fires at some point and then there may be multiple steps before the access becomes available *e.g.* to create the access link; but it always will.

We can verify there's never unauthorised accesses, *e.g.* the links never connect where they should not or personal information is not nested within an unauthorised agent:

$$\mathbf{A}[\mathbf{G}\neg\text{unauthAccess}] \quad (7.11)$$

Here, the entire property is inverted, *i.e.* for all paths there is no an unauthorised access.

Finally, we check that the user's data is deleted when the consent is withdrawn:

$$\mathbf{A}[\mathbf{G}(\text{withdAll} \implies (\mathbf{F}\text{deletingInfo}))] \quad (7.12)$$

With this formula, we ensure that in every possible execution of the system, if consent is fully withdrawn at any point, the system will eventually delete the user's information, although the deletion may occur after several steps following the withdrawal of consent.

All these properties hold for both our banking and healthcare examples when checked with PRISM. This gives increased confidence⁴ the systems, and privacy policy implementations, operate in accordance with the privacy regulations: properties 7.5, 7.6 and 7.9 say we are GDPR compliant, while the remaining properties are shared by all the regulations we have considered.

Table 7.2 aligns each CTL property with the relevant GDPR articles and—where applicable—the CCPA topics, and indicates coverage within our modelling scope. Notably, properties 7.5 and 7.6 capture the GDPR consent framework, but there is no direct CCPA counterpart, as the CCPA relies on notice at collection rather than consent. Property 7.9 addresses Standard Contractual Clauses for cross-border transfers and likewise has no CCPA analogue. The remaining properties align with both the GDPR and the CCPA. Coverage is marked *Partial* because the CTL checks enforce core runtime prohibitions (*e.g.* no sharing

⁴Although we capture key features of privacy regulations, we cannot claim full adherence unless *every* aspect, *e.g.* retention time and data encryption, was fully modelled and checked; but we can operate with increased confidence.

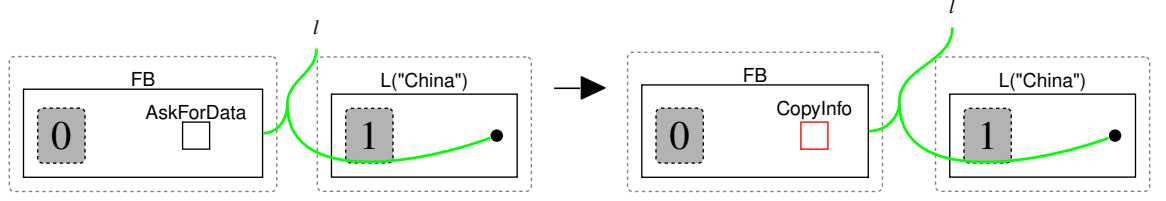


Figure 7.9: `privacyViolation`: system rule that leads to a privacy violation by transferring the data to FB in China .

after opt-out) but do not capture non-technical compliance duties or every alternative legal basis.

7.3 Detecting Privacy Violations

Malformed initial states or reaction rules can introduce privacy violations. For instance, according to the defined sorting constraints, $L(\text{Singapore})$ should contain only a single child of either sort `ctr` or `inctr`. However, if a malformed initial state or rule assigns $L(\text{Singapore})$ both sorts, this could lead to data being transferred to $L(\text{Singapore})$ during the execution of the transition system, violating the intended semantics.

Another example that highlights the vital role of sorts in ensuring the correctness of the model is when `Cont` is linked to a port of sort `sc` on `Scheme` instead of `p`. In this scenario, the process of checking the region cannot be performed, resulting in a bug in the model. Such an issue can lead to significant consequences, especially when data transfer is critical. To maintain the integrity of the model, `Cont` must be linked to ports of sort `p` and `sy`, as specified in Fig. 7.1.

Defining system rules can inadvertently lead to privacy violations. As discussed in Section 5.3, data must not be transferred to FB in China because the provided certification is invalid. However, modellers may define a system rule that allows data to be transferred to FB, as shown in Fig. 7.9. This violation occurs because the rule is defined without explicitly matching on `InvalidCert`. The PRISM model checker detects this violation as property 7.3 is not met (*i.e.* the certification is invalid, but the transfer occurs regardless). By inspecting the generated transition system, it is straightforward to identify the rule introducing the violation, *i.e.* the rule that generates a state holding `invalidCert`, followed by a state holding `dataTransfer`. To correct this rule, we redefine it as presented in Fig. 5.21.

Table 7.2: Mapping of the CTL properties to the GDPR and the CCPA. *Partial* denotes core runtime coverage only, while ‘-’ indicates no direct counterpart.

CTL property number	GDPR article(s)	Coverage	CCPA topic	Coverage
7.5	Art. 4(11); [65]; Art. 7 [71] (consent)	Partial	-	-
7.6	Art. 6(1)(a) [70]; Art. 7 [71] (consent before processing)	Partial	-	-
7.7 / 7.8	Art. 7(3) [71]; Art. 21 [63] (withdrawal)	Partial	Opt-out of sale/share [124]	Partial
7.9	Art. 46(2)(c,d) (Standard Contractual Clauses) [68]	Partial	-	-
7.10	Art. 15 (right of access) [61]	Partial	Right to know / access [125]	Partial
7.11	Art. 5(1)(f) [69]; Art. 32 [64] (security)	Partial	Reasonable security [123]	Partial
7.12	Art. 17 [62] (erasure)	Partial	Deletion right [127]	Partial

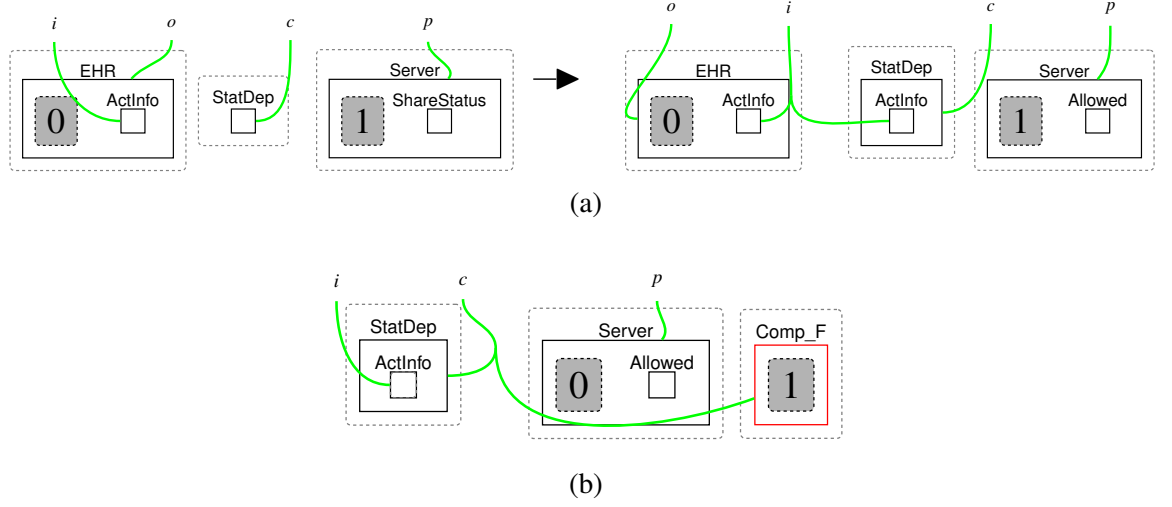


Figure 7.10: (a) System-specific rule for copying data to a third-party. As the consent was not checked (*i.e.* there is no match into the privacy perspectives), this may result in a privacy violation; (b) Partial model state showing the privacy violation: activity information has moved when consent was not given.

Similarly, [Fig. 7.10a](#) shows a system rule that transfers data from the database to the statistics department at the NRA. This is only valid if the user has accepted optional policies, but the rule has not checked consent before transferring the data. This means we might end up in the (partial) state shown in [Fig. 7.10b](#) where data has moved without a user's consent.

This is detected by property 7.8 as the property is not fulfilled in this case. The end-users need to fix that by matching over the privacy perspectives, specifically, matching over Comp_F to prevent sharing and Comp to share the information as shown in [Fig. 6.41](#) and [Fig. 6.39](#). For more complex examples, a model checker will give a full trace that can aid experts in determining a suitable fix (and the model checker can then prove the fix was correct).

7.4 Discussion

The verification process for our framework proceeds from static analysis to dynamic model checking. We conduct two forms of static analysis—sorting and inductive reasoning—to verify the structure of the models' initial states and reaction rules. Sorting plays a structural role by enforcing well-formedness constraints on the model, preventing invalid configurations

that could lead to privacy violations during execution. For example, the defined sorts enable us to ensure that data transfer occurs only if the Contract is valid, specifically when it includes SCCs. Without the use of these sorts, we risk the Contract containing unintended entities instead of SCCs, which would mean it fails to be marked as `InvalidContract` and could wrongly allow the transfer to proceed.

Inductive reasoning proves the correctness of the privacy reaction rules independently of any specific system instance. It ensures that each rule preserves well-formedness and the required privacy constraints. Consequently, applying the rules to a well-formed state (as enforced by sorting) cannot produce a state that breaches privacy.

After the static analysis, we use model checking to verify privacy properties over dynamic system configurations, ensuring that the specified properties are preserved throughout system evolution.

Although our framework primarily focuses on modelling the privacy notions we consider in this thesis rather than directly addressing adversary models, the scenario in [Fig. 7.10](#) illustrates how adversarial behaviour could be represented. While this scenario is examined through the lens of regulatory compliance, it highlights a situation in which bypassing consent checks could be considered adversarial. This violation can also be detected using property [7.8](#). Through the model checker, developers can adjust the system’s behaviour to prevent such actions, *e.g.* by triggering alerts if any attempt to bypass the consent check is detected and subsequently blocking `StatDep`.

7.5 Summary

In this chapter, we applied two verification techniques—static analysis ([Section 7.1](#)) and model checking ([Section 7.2](#)) to evaluate the privacy models introduced in [Chapters 5](#) and [6](#). Static analysis, including sorting and inductive reasoning are employed to ensure the correctness of the initial states and privacy rules. Model checking is used to verify that specific system configurations uphold privacy properties throughout system evolution. We also presented examples to show how we can detect potential privacy violations and how we can fix them ([Section 7.3](#)). Finally, we reflected on the chapter’s key points in [Section 7.4](#).

Chapter 8

Enhancing Usability

To mitigate the risk of incorrectly defining initial states, *e.g.* linking entities with improper permissions, we develop user interfaces (UIs) that allow end-users to automatically generate initial states that are correct by construction. The first UI is the *Bigraph International Data Transfer Regulation Assistant* (BIDTRA). It defines initial states for the first model, which captures the GDPR requirements for cross-border data transfers (adequacy decision, SCCs and certification mechanism). The second UI is the *Bigraph Privacy Regulations Assistant* (BPRA). It is designed to generate initial states for the second model, which focuses on providing and withdrawing consent, purpose limitation, the right to access and sharing data with third parties.

This chapter is organised into the following sections. [Section 8.1](#) introduces BIDTRA, followed by [Section 8.2](#), which presents BPRA. [Section 8.3](#) provides a discussion of the two UIs and highlights their limitations, while [Section 8.4](#) concludes the chapter with a summary of its main points.

8.1 BIDTRA Interface

BIDTRA is an interface designed to facilitate the modelling of privacy regulations, particularly those related to cross-border data transfers. [Fig. 8.1](#) shows the main window of the BIDTRA interface, which includes three primary buttons: (1) Combine System File with Privacy File, (2) Cross-Border Data Transfers, and (3) Run BigraphER.

The following sections describe the functions of these buttons and detail how BIDTRA

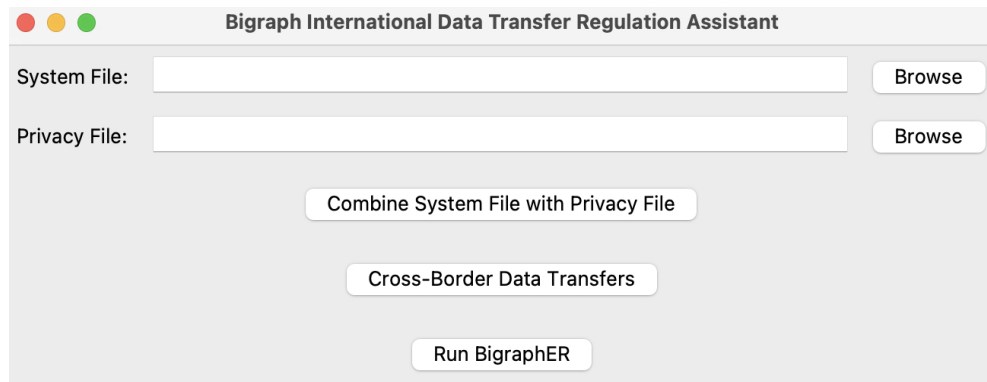


Figure 8.1: The main window of the BIDTRA interface.

integrates system components with the privacy framework to model the GDPR requirements for cross-border data transfers.

8.1.1 System and Privacy File Integration

The interface is structured around two core inputs: *System File* and *Privacy File*. The system file is created by end-users to define system-specific constructs as shown in Fig. 8.2. This file includes system entities (defined in lines 1–5) and a set of system rules (specified in lines 7–20). Line 22 defines the rule priority classes. Importantly, the System File deliberately excludes the initial state as this is automatically generated by the interface as we will discuss in Section 8.1.2.

Unlike the system file, the privacy file serves as a template that is included in the downloaded package of the interface. It includes all privacy-related entities and rules discussed in Chapter 5. A partial excerpt from this file is presented in Fig. 8.3, while the complete file is available in [5]. The privacy file includes privacy entities defined in lines 1–22 and privacy rules (introduced in Chapter 5) defined in lines 26–38. The initial states (a_1 , a_2 , and a_3) are declared in lines 41–44 and are dynamically populated based on user input via the interface. A string list, l , is defined in line 47 to represent the list of countries specified by the user. The priority class of the privacy rules is specified in line 51. Importantly, the privacy file contains the entity System (line 24), which represents the top-level entity of the system perspective. It is included in the privacy file because it is used to construct the initial state of the system (see line 41).

After the user provides the system file and the privacy file and clicks on Combine System

```

1 ctrl DC = 1;
2 ctrl FB = 1;
3 atomic ctrl AskForData = 0;
4 atomic ctrl SpecifyReceivers = 0;
5 atomic ctrl Data = 0;
6
7 react dcIrType=
8   Proc{1} || DC{1}.id || L("Ireland").(id | P{1})
9   -[1]->
10  SType{1}.Proc{1} || DC{1}.(id | SpecifyReceivers) || L("Ireland").(id | P{1});
11
12 react dcSingType=
13   SpecifyReceivers || Proc{1} || DC{1}.id || L("Singapore").(id | P{1})
14   -[1]->
15   CheckReg || RType{1}.Proc{1} || DC{1}.(id | AskForData) || L("Singapore").(id | P{1});
16
17 react validContract=
18   L("Singapore").(P{1} | Contract{s}.id) || DC{1}.(id | AskForData)
19   -[1]->
20   L("Singapore").(P{1} | Contract{s}.id) || DC{1}.(id | Data);
21
22 rules = [{dcIrType}, {dcSingType}, {validContract}];

```

Figure 8.2: System file used in the BIDTRA interface. This file is created by users to define system-specific entities, rules and rule priority classes.

File with Privacy File, the content of the system file is automatically inserted into the privacy file as shown in [Fig. 8.4](#).

8.1.2 Building the Initial State

When the user clicks the Cross-Border Data Transfers button, the window shown in [Fig. 8.5a](#) appears. This window allows the user to build the initial state for modelling cross-border data transfers. The user begins by entering the sender's country and selecting the sender's name from a dropdown list populated with the system controls defined in the system file. The sender's type (Processor or Controller) is then selected, followed by the number of receivers involved. When the user clicks on **Add Countries**, the window shown in [Fig. 8.5b](#) appears, allowing the user to specify:

1. The receiver's country and name, where the name is selected from the same dropdown list of system-defined controls.
2. The type of legal safeguard applied to the transfer, selected from a list that includes: Adequate Country, SCCs, Certification, or None.

```

1 #SRTYPE perspective
2 ctrl SRTYPE =0;
3 ctrl STYPE = 1;
4 ctrl RTYPE = 1;
5 atomic ctrl Proc = 1;
6 atomic ctrl Cont =1;
7 # locations perspective
8 ctrl Locations = 0;
9 atomic ctrl P=1;
10 fun ctrl L(l) = 0 ;
11 ctrl Contract = 1;
12 atomic ctrl SCCs =0;
13 ctrl Scheme = 1;
14 atomic fun ctrl C(c) = 0;
15 ctrl Cert=1;
16 atomic ctrl ExpiryDate =0;
17 # privacy entities
18 atomic ctrl CheckReg =0;
19 atomic ctrl SameRegion =0;
20 atomic ctrl P' = 1;
21 atomic ctrl Ps = 1;
22 atomic ctrl InvalidContract = 1;
23 #System perspective
24 ctrl System = 0;
25
26 react checkSType =
27   CheckReg || STYPE{l}.id || P{l} -[1]-> CheckReg || id || Ps{l};
28
29 react checkRTYPE =
30   CheckReg || RTYPE{l}.id || P{l} -[1]-> CheckReg || id || P'{l};
31
32 react sameReg =
33   CheckReg || ( Ps{a} | P'{a'}) -[1]-> SameRegion || ( P{a} | P{a'});
34
35 react checkingSCCs =
36   CheckReg || ( P'{a} | Contract{s}.1 )
37   -[1]->
38   1 || ( P{a} | /s InvalidContract{s}) || {s};
39
40 # Initial states
41 big a1 = System.();
42 big a2 = SRTYPE.();
43 big a3 = Locations.();
44 big s = (a1 || a2 || a3);
45
46 begin pbrs
47   string l = {};
48   # Criteria number
49   int c = {1 , 2 , 3};
50   init s;
51   rules = [ {checkSType}, {checkRTYPE}, {sameReg}, {checkingSCCs}];
52 end

```

Figure 8.3: Partial content of the privacy file, provided as a predefined template within the interface’s downloaded package.

```

1 # SRType perspective
2 ctrl SRType =0;
3 ctrl SType = 1;
4 ctrl RType = 1;
5 atomic ctrl Proc = 1;
6 atomic ctrl Cont =1;
7 # locations perspective
8 ctrl Locations = 0;
9 atomic ctrl P=1;
10 fun ctrl L(l) = 0 ;
11 ctrl Contract = 1;
12 atomic ctrl SCCs =0;
13 ctrl Scheme = 1;
14 atomic fun ctrl C(c) = 0;
15 ctrl Cert=1;
16 atomic ctrl ExpiryDate =0;
17 # privacy entities
18 atomic ctrl CheckReg =0;
19 atomic ctrl SameRegion =0;
20 atomic ctrl P' = 1;
21 atomic ctrl Ps = 1;
22 atomic ctrl InvalidContract = 1;
23 #System perspective
24 ctrl System = 0;
25 ctrl DC = 1;
26 ctrl FB = 1;
27 atomic ctrl AskForData = 0;
28 atomic ctrl SpecifyReceivers = 0;
29 atomic ctrl Data = 0;
30
31 react dcIrType=
32 Proc{l} || DC{l}.id || L("Ireland").(id | P{l})
33 -[l]->
34 SType{l}.Proc{l} || DC{l}.(id | SpecifyReceivers) || L("Ireland").(id | P{l});
35 react dcSingType=
36 SpecifyReceivers || Proc{l} || DC{l}.id || L("Singapore").(id | P{l})
37 -[l]->
38 CheckReg || RType{l}.Proc{l} || DC{l}.(id | AskForData) || L("Singapore").(id | P{l});
39 react validContract=
40 L("Singapore").(P'{l} | Contract{s}.id) || DC{l}.(id | AskForData)
41 -[l]->
42 L("Singapore").(P{l} | Contract{s}.id) || DC{l}.(id | Data);
43 react checkSType =
44 CheckReg || SType{l}.id || P{l} -[l]-> CheckReg || id || Ps{l};
45 react checkRType =
46 CheckReg || RType{l}.id || P{l} -[l]-> CheckReg || id || P'{l};
47 react sameReg =
48 CheckReg || ( Ps{a} | P'{a'}) -[l]-> SameRegion || ( P{a} | P{a'});
49 react checkingSCCs =
50 CheckReg || ( P'{a} | Contract{s}.1 )
51 -[l]->
52 1 || ( P{a} | /s InvalidContract{s}) || {s};
53
54 # Initial states
55 big a1 = System.() ;
56 big a2 = SRType.();
57 big a3 = Locations.();
58 big s = (a1 || a2 || a3);
59
60 begin pbrs
61 string l = {};
62 # Criteria number
63 int c = {1 , 2 , 3};
64 init s;
65 rules = [{dcIrType}, {dcSingType}, {validContract},
66         {checkSType}, {checkRType}, {sameReg}, {checkingSCCs}];
67 end

```

Figure 8.4: The privacy file after insertion of the system file content through the BIDTRA interface.

Cross-Border Data Transfers

Sender Country: Sender Name: Sender Type:

Receiver Number:

(a) Sender specification interface, where the user defines the sender's country, name, type and the number of intended receivers.

Cross-Border Data Transfers

Sender Country: Sender Name: Sender Type:

Receiver Number:

Country Name 1: Singapore	Add Requirement	Requirement Type: SCCs	Receiver Name: DC
		Receiver Type: Processor	
Country Name 2: China	Add Requirement	Requirement Type: Certification	Receiver Name: FB
		Receiver Type: Data Controller	

Receiver Name Dropdown:
 FB
 DC
 FB
 AskForData
 SpecifyReceivers

(b) Receiver specification interface, enabling the user to specify each receiver's country, applicable safeguard, type and name.

Figure 8.5: Building an initial state via the BIDTRA interface.

```

1 # Initial states
2 big a1 = System.(DC{y_E}.1 | FB{s2N}.1 | DC{g0}.1) ;
3 big a2 = SRType.(Proc{y_E} | Cont{s2N} | Proc{g0});
4 big a3 = Locations.(L("Ireland").(P{y_E} | Contract{c_}.SCCs
5                               | Scheme{o_}.(C(1) | C(2) | C(3)))
6                               | L("Singapore").(P{g0} | Contract{c_}.SCCs )
7                               | L("China").(P{s2N} | Cert{o_}.(ExpiryDate | C(1) | C(2) | C(3))));
8 big s =/s2N /g0 /y_E /o_ /c_ (a1 || a2 || a3);
9
10 begin pbrs
11   string l = {"Singapore", "China", "Ireland"};
12   # Criteria number
13   int c = {1 , 2 , 3};
14   init s;
15   rules = [{dcIrType}, {dcSingType}, {validContract},
16           {checkSType}, {checkRType}, {sameReg}, {checkingSCCs}];
17 end

```

Figure 8.6: The initial state generated by the BIDTRA interface.

3. The receiver's type, selected from a dropdown list with two options: Processor or Controller.

Once the user clicks on **Save**, the interface links these inputs and write them into the privacy file, populating the empty initial states and the string list *l* shown in Fig. 8.4. The resulting initial states are presented in Fig. 8.6, while the rest of the privacy file remains unchanged from Fig. 8.4. As shown in the result, the interface links the receiver DC with its type Proc and with a pointer nested within Singapore. The link is generated using a function that ensures uniqueness, avoiding link duplication. The interface also creates the Contract entity and links it, via link *c_*, with the corresponding Contract in Ireland. Other entities are linked in a similar manner and their associated safeguards are generated accordingly.

The Done button allows the user to return to the main window shown in Fig. 8.1, where they can create a new initial state or run BigraphER.

8.1.3 Run BigraphER

The Run BigraphER button enables the user to execute the generated model using the BigraphER tool. Clicking this button triggers a dialog asking the user to enter the maximum number of states, as shown in Fig. 8.7a. Once confirmed, the interface runs BigraphER using the privacy file¹. In parallel, BigraphER outputs detailed information in the terminal as shown

¹Priority classes are not set by the interface and must be edited manually in the privacy file before or after running BigraphER.

in Fig. 8.7b, including the number of states, transitions and execution time. The interface then displays a confirmation dialog indicating that the model is generated and saved in a file named *Output.svg* (see Fig. 8.7c). The content of this file is shown in Figure 8.8. Figure 8.8a presents the model's transition system, while Figure 8.8b (displayed when clicking state 0 in Figure 8.8a) shows the initial state defined by the BIDTRA interface.

8.2 BPRA Interface

BPRA is an interface designed to generate initial states for the privacy regulations model. The main window of BPRA, shown in Fig. 8.9, consists of five primary buttons: (1) Combine System & Privacy Files, (2) Add New Permissions, (3) Add New Purposes, (4) Link Initial States, and (5) Run BigraphER. The following sections describe the function of each button and the associated user workflow.

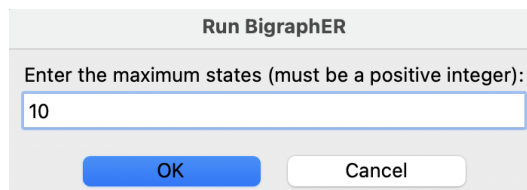
8.2.1 System and Privacy File Integration

The Combine System & Privacy Files button in the BPRA interface functions similarly to the corresponding button in the BIDTRA interface. It enables users to integrate their system-specific definitions with the predefined privacy template. The system file, created by end-users, defines the relevant system entities, reaction rules and priority classes as shown in Fig. 8.10. The privacy file is provided within the BPRA package. Partial content of this file is shown in Fig. 8.11². When the Combine System & Privacy Files button is clicked, the content of the system file is automatically inserted into the privacy file. The result of this integration is shown in Fig. 8.12.

8.2.2 Adding New Permissions and Purposes

The Add New Permissions and Add New Purposes buttons allow users to extend the predefined set of permissions and purposes. When either button is clicked, a dialog appears prompting the user to enter a new permission or purpose as shown in Figs. 8.13a and 8.13c. The user is then asked to specify whether the new permission or purpose is of type *basic* or *optional* as illustrated in Figs. 8.13b and 8.13d.

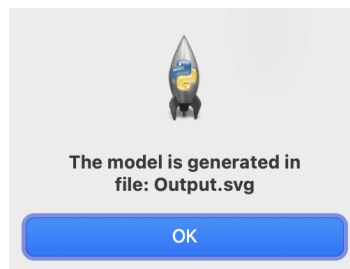
²The full content of the file is available in [5].



(a)

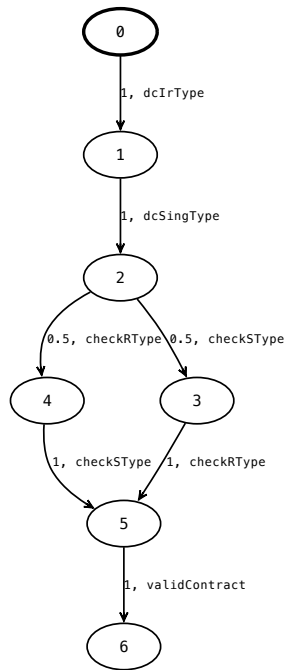


(b)

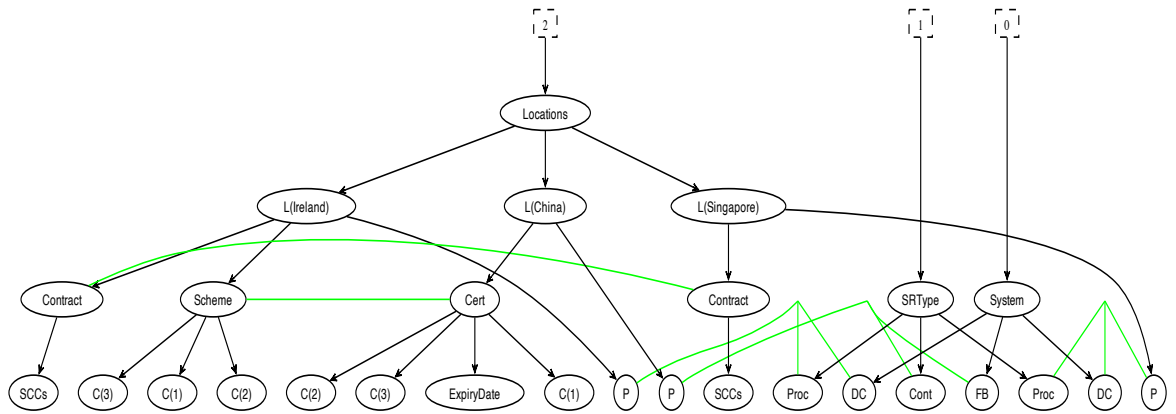


(c)

Figure 8.7: Running BigraphER through the BIDTRA interface: (a) prompt for entering the maximum number of states; (b) terminal output displaying model details; (c) confirmation dialog indicating that the model has been generated.



(a)



(b)

Figure 8.8: Content of the *Output.svg* file produced by the BIDTRA interface. (a) Partial transition system of the model; (b) the initial state automatically generated by the interface (displayed when clicking state 0 in (a)). The hierarchical structure shown in (b) is equivalent to the nesting used in the diagrammatic representations of initial states in previous chapters.

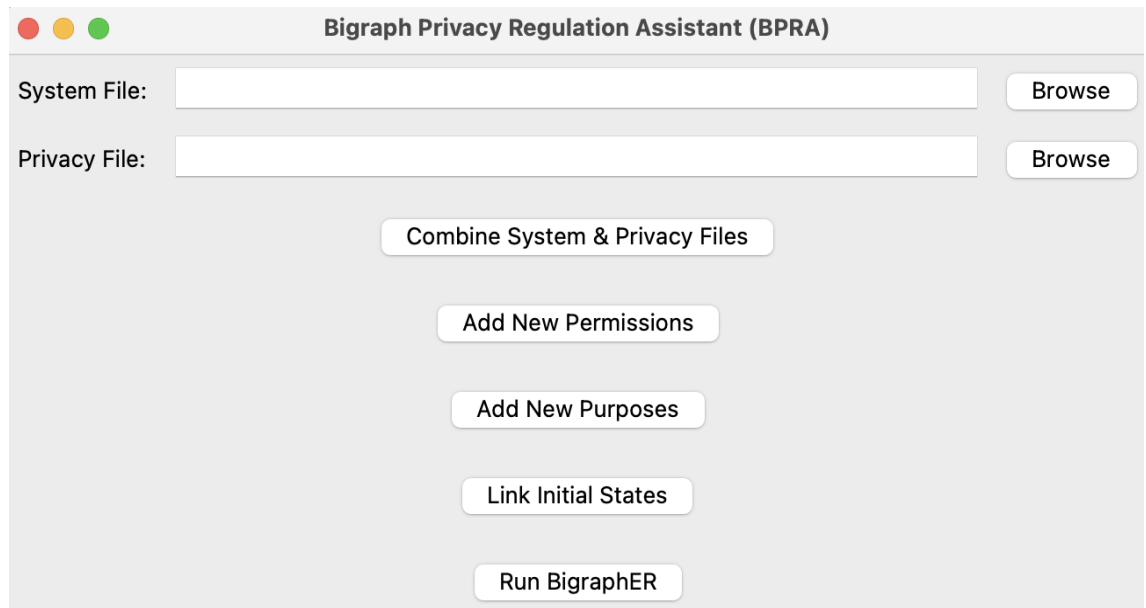


Figure 8.9: The main window of the BPRA interface.

```

1 ctrl User = 1;
2 ctrl App = 0;                                # Mobile banking application
3 atomic ctrl TransInfo = 1;                    # Transaction information
4 ctrl Bank = 1;                                # Banking institution
5 ctrl DB = 1;                                  # Database
6 ctrl AdCompany = 1;                           # Advertising company
7 ctrl Marketing = 1;                           # Marketing department
8 atomic ctrl ReqUseSys = 0;                    # Request to use the system
9 ctrl StatusReq = 0;                          # Status of the user's request
10 atomic ctrl SysAccept = 0;                   # System's acceptance of the user's request
11 ctrl ToStore = 0;                            # Control to initialise the storing process in the DB
12
13 react usingSystemReq =                      # Request to use the system
14   App.(id | ReqUseSys ) -[1]-> App.( id | StatusReq. SysAccept);
15 react generateTrans =                      # Generate transaction and prepare to store it
16   App.(id | TransInfo{i}) -[1]-> App.(id | ToStore. TransInfo{i});
17 rules = [{usingSystemReq}, {generateTrans}];

```

Figure 8.10: The system file used in the BPRA interface. It is created by end-users to define system-specific entities, reaction rules and rule priority classes.

```

1 # DGE perspective
2 ctrl DGE =0;
3 ctrl PrPo=0;
4 ctrl BasicPerm =0;
5 atomic ctrl S =1;
6 atomic ctrl P=1;
7 ctrl Purp = 0;
8 atomic ctrl Serving = 1 ;
9 ctrl OptPerm=0;
10 atomic ctrl OptIn = 1;
11 atomic ctrl Ad = 1;
12 atomic ctrl AuthAgent = 1;
13 atomic ctrl PersonalInfo = 1;
14 # ADType perspective
15 ctrl ADType=0;
16 ctrl Owner = 1;
17 ctrl Proc = 1;
18 ctrl TP = 1;
19 ctrl Info = 1;
20 ctrl Comp=1;
21 # Consent perspective
22 ctrl Consent =1;
23 # Privacy entities
24 ctrl Accepted =0;
25 atomic ctrl All =0;
26 # System perspective
27 ctrl System = 0;
28
29 react sendingPolicy=
30   DGE.(id | PrPo.id) || Owner{o}.1 -[1]-> DGE.(id | PrPo.id) || Owner{o}.PrPo.id
31   @[0,1,1] ;
32 react acceptingAll=
33   Owner{o}.PrPo.(BasicPerm.(id | Purp.id) | OptPerm.(id | Purp.id))
34   || Consent{o}.1
35   -[1]->
36   Owner{o}.PrPo.Accepted.All
37   || Consent{o}.(BasicPerm.(id | Purp.id) | OptPerm.(id | Purp.id));
38
39 # Initial states
40 big a1= DGE.(PrPo.(BasicPerm.(S{s} | Purp.(Serving{se}))
41           | OptPerm.(OptIn{op} | Purp.(Ad{ad})))
42           | AuthAgent{a} | PersonalInfo{pe});
43 big a2 = Consent{o}.1;
44 big a3= ADType.();
45 big a4= System.();
46 big s = /s /se /op /ad /a /pe /o (a1 || a2 || a3 || a4);
47
48 begin pbrs
49   init s;
50   rules = [{sendingPolicy}, {acceptingAll}];
51 end

```

Figure 8.11: Partial content of the privacy file provided in the BPRA package.

```

1 # DGE perspective
2 ctrl DGE =0;
3 ctrl PrPo=0;
4 ctrl BasicPerm =0;
5 atomic ctrl S =1;
6 atomic ctrl P=1;
7 ctrl Purp = 0;
8 atomic ctrl Serving = 1 ;
9 ctrl OptPerm=0;
10 atomic ctrl OptIn = 1;
11 atomic ctrl Ad = 1;
12 atomic ctrl AuthAgent = 1;
13 atomic ctrl PersonalInfo = 1;
14 # ADType perspective
15 ctrl ADType=0;
16 ctrl Owner = 1;
17 ctrl Proc = 1;
18 ctrl TP = 1;
19 ctrl Info = 1;
20 ctrl Comp=1;
21 # Consent perspective
22 ctrl Consent =1;
23 # Privacy entities
24 ctrl Accepted =0;
25 atomic ctrl All =0;
26 # System perspective
27 ctrl System = 0;
28 ctrl User = 1;
29 ctrl App = 0;
30 atomic ctrl TransInfo = 1;
31 ctrl Bank = 1;
32 ctrl DB = 1;
33 ctrl AdCompany = 1;
34 ctrl Marketing = 1;
35 atomic ctrl ReqUseSys = 0;
36 ctrl StatusReq = 0;
37 atomic ctrl SysAccept = 0;
38 ctrl ToStore = 0;
39
40 react usingSystemReq = # Request to use the system
41   App.( id | ReqUseSys ) -[1]-> App.( id | StatusReq. SysAccept );
42 react generateTrans = # Generate transaction and prepare to store it
43   App.(id | TransInfo{i}) -[1]-> App.(id | ToStore. TransInfo{i});
44 react sendingPolicy=
45   DGE.(id | PrPo.id) || Owner{o}.1 -[1]-> DGE.(id | PrPo.id) || Owner{o}.PrPo.id
46   @[0,1,1] ;
47 react acceptingAll=
48   Owner{o}.PrPo.(BasicPerm.(id | Purp.id) | OptPerm.(id | Purp.id))
49   || Consent{o}.1
50   -[1]->
51   Owner{o}.PrPo.Accepted.All
52   || Consent{o}.(BasicPerm.(id | Purp.id) | OptPerm.(id | Purp.id));
53
54 # Initial states
55 big a1= DGE.(PrPo.(BasicPerm.(S{s} | Purp.(Serving{se})))
56         | OptPerm.(OptIn{op} | Purp.(Ad{ad})))
57         | AuthAgent{a} | PersonalInfo{pe});
58 big a2 = Consent{o}.1;
59 big a3= ADType.();
60 big a4= System.();
61 big s = /s /se /op /ad /a /pe /o (a1 || a2 || a3 || a4);
62
63 begin pbrs
64   init s;
65   rules = [{usingSystemReq}, {generateTrans}, {sendingPolicy}, {acceptingAll}];
66 end

```

Figure 8.12: The privacy file after integration with the system file through the BPRA interface.

(a) Entering the permission name.

(b) Specifying the permission type (basic or optional).

(c) Entering the purpose name.

(d) Specifying the purpose type (basic or optional).

Figure 8.13: Sub-windows that appear when adding a new permission or purpose.

Once the user enters the permission and purpose, they are automatically added to the initial state of the privacy file as shown in [Fig. 8.14](#). The permission is added under the corresponding entity—either BasicPerm or OptPerm—depending on its type. Conversely, the purpose is added to the Purp entity nested within BasicPerm or OptPerm, also based on its specified type. Importantly, the interface defines the added permission and purpose as atomic controls and assigns a link to each of them (see lines 3 and 4 in [Fig. 8.14](#)) to connect system components and data to these permissions and purposes as will be discussed in [Section 8.2.3](#).

8.2.3 Building the Initial State

When the Link Initial States button is clicked, the interface shown in [Fig. 8.15a](#) appears. This view allows end-users to select a system agent or data from a drop-down list of system entities (*e.g.* Bank), specify its type (*Processor*, *Third Party*, *Owner*, or *Information*), and define the number and names of its associated components (*e.g.* DB).

Clicking the Add Privileges button opens a new window ([Fig. 8.15b](#)), where the user can assign privacy-relevant properties to the selected component. These include the authorisation level (Authorised or Unauthorised), applicable permissions (*e.g.* Store, Read, *etc.*) and associated purposes (*e.g.* Serving, Advertising, *etc.*). The interface also supports selection of

```

1 # DGE perspective
2 ctrl DGE =0;
3 atomic ctrl RE = 1;
4 atomic ctrl Wr = 1;
5 ctrl PrPo=0;
6 ctrl BasicPerm =0;
7 atomic ctrl S =1;
8 atomic ctrl P=1;
9 ctrl Purp = 0;
10 atomic ctrl Serving = 1 ;
11 ctrl OptPerm=0;
12 atomic ctrl OptIn = 1;
13 atomic ctrl Ad = 1;
14 atomic ctrl AuthAgent = 1;
15 atomic ctrl PersonalInfo = 1;
16
17 big a1= DGE.(PrPo.(BasicPerm.(Wr{wr_3} | S{s} | Purp.(Serving{se}))
18         | OptPerm.(OptIn{op} | Purp.(RE{re_3} | Ad{ad})))
19         | AuthAgent{a} | PersonalInfo{pe});

```

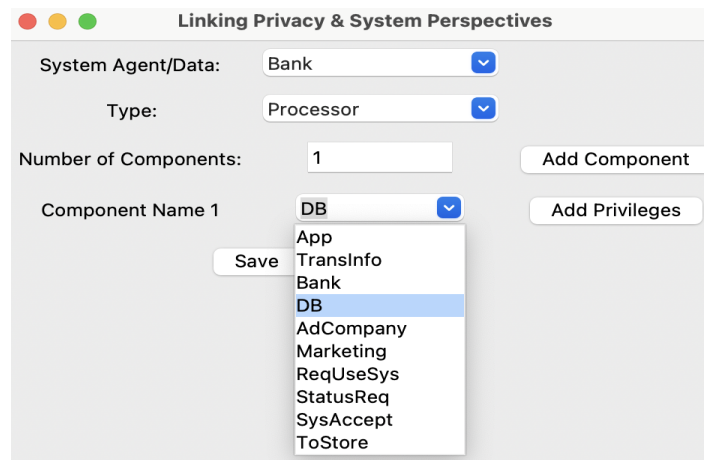
Figure 8.14: The privacy file after the user adds the write (Wr) permission as a basic permission and the research purpose (RE) as an optional purpose through the BPRA interface. Both are defined as atomic controls and added to the DGE perspective. The rest of the file remains unchanged and is unaffected by these additions.

previously added permissions and purposes (Wr and RE) through a drop-down menu enabled by the *Other* option.

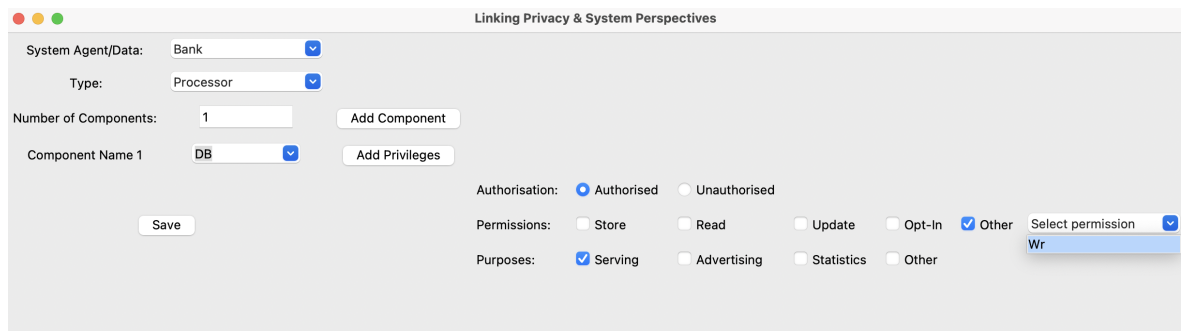
Once the selections are complete, clicking Save returns the user to the main window shown in Fig. 8.9 and writes the configuration directly into the privacy file, updating the initial state accordingly. To add a new entity, the user can click the Link Initial States button again and follow the same workflow. For example, after saving the previous entry, Fig. 8.15c shows how additional information (e.g. TransInfo) can be added by specifying its type (Personal or General), the required permissions and purposes for access and the system entities authorised to access it. Similarly, User and AdCompany can be added as an owner and a third party, respectively, using the same process (see Figs. D.3 and D.4). The resulting initial states, after all configurations are applied, are shown in Fig. 8.16.

8.2.4 Run BigraphER

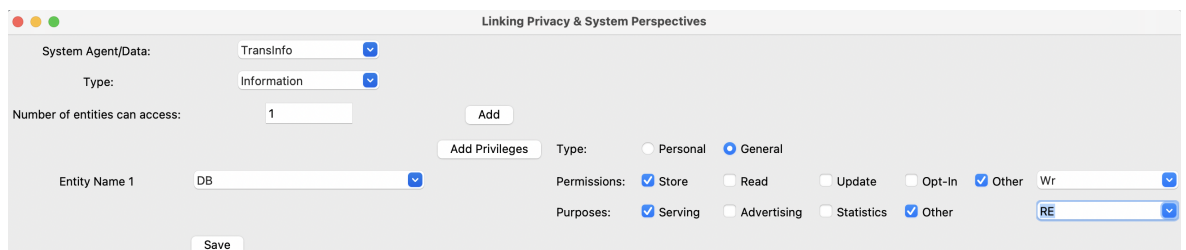
The Run BigraphER button in the BPRA interface behaves identically to the one in the BIDTRA interface. All dialogs—such as the prompt for entering the maximum number of states, the terminal output and the confirmation upon successful execution—are exactly the same as those described in Section 8.1.3 and shown in Fig. 8.7.



(a)



(b)



(c)

Figure 8.15: (a) Window for selecting a system agent or data, specifying its type, the number of associated component and their names; (b) Privilege assignment window for a selected system component; (c) Privilege assignment window for system data.

```

1 # Initial states
2 big a1= DGE.(PrPo.(BasicPerm.(Wr{wr_3} | S{s} | Purp.(Serving{se}))
3           | OptPerm.(OptIn{op} | Purp.(RE{re_3} | Ad{ad})))
4           | AuthAgent{a} | PersonalInfo{pe});
5 big a2 = Consent{o}.1;
6
7 big a3= ADType.(Owner{o}.1 | Info{jS}.(P{re_3} | P{se} | P{wr_3} | P{s}))
8           | TP{c_g}.(Comp{c_}.(P{re_3} | P{s}))
9           | Proc{xW}.(Comp{c6K}.(P{se} | P{wr_3} | P{a})));
10
11 big a4= System.(User{o}.1 | AdCompany{c_g}.(Marketing{c_}.1)
12           | Bank{xW}.(DB{c6K}.TransInfo{jS}));
13
14 big s =/jS /re_3/wr_3/c_ /c_g /c6K /xW /s /se /op /ad /a /pe /o (a1 || a2 || a3 || a4);

```

Figure 8.16: The initial states in the privacy file after linking system entities with selected permissions and purposes via the BPRA interface.

As in BIDTRA, once the initial state is defined, the user can trigger the execution, and the model is saved as *Output.svg*. Figure 8.17 shows the content of the *Output.svg* file produced by the BPRA interface.

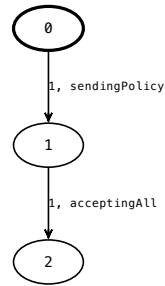
8.3 Discussion

The UIs presented in this chapter automate the generation of initial states while preserving BigraphER syntax, eliminating the need for users to manually correct structural formatting. They also increase confidence that the generated models preserve the sorting constraints defined in Chapter 7, ensuring structural correctness and reducing the risk of modelling errors.

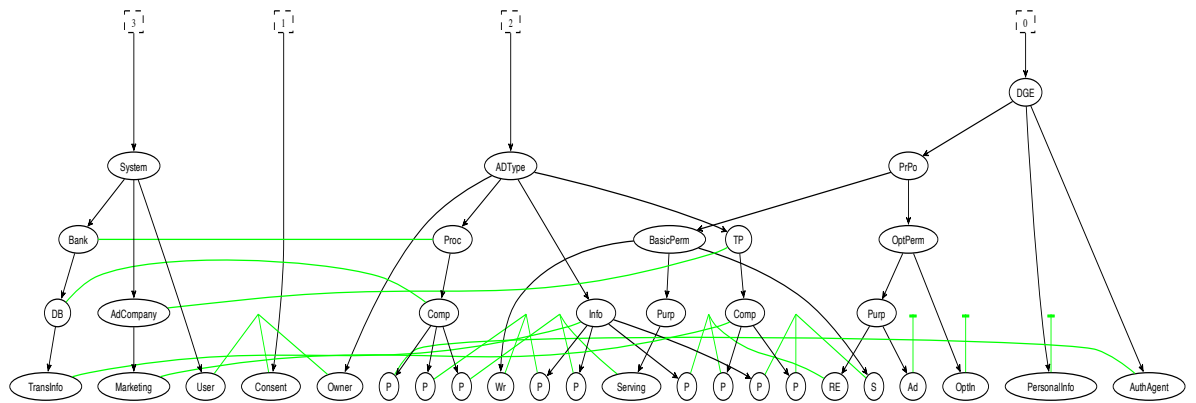
However, entities that do not require specific links or types, *e.g.* App, must be added manually, as the interfaces are primarily designed to support the definition of privacy-related entities and their associations with permissions, types, and locations. The UIs also do not currently support modifying rule priority classes, which must be adjusted manually if needed.

As a result, the primary users of both UIs are system developers familiar with bigraphs and the BigraphER tool, as they need to prepare system files and edit priority classes. Privacy specialists can also collaborate with bigraph experts to specify legal aspects, *e.g.* defining safeguards per country and setting purpose constraints.

Although the UIs have been successfully applied to the examples in this thesis, further evaluation on larger and more complex case studies is needed. Structured user studies are also essential to assess their usability and to guide future improvements.



(a)



(b)

Figure 8.17: Content of the *Output.svg* file produced by the BPRA interface. (a) Partial transition system of the model; (b) the initial state automatically generated by the interface (displayed upon clicking state 0 in (a)).

8.4 Summary

This chapter presents two UIs designed to assist users in correctly constructing initial states for our privacy models. [Section 8.1](#) introduces the BIDTRA interface, which supports the modelling of the GDPR requirements for cross-border data transfers. This is followed by [Section 8.2](#), which presents the BPRA interface to support the modelling of key privacy regulation principles such as consent, purpose limitation, the right to access and third-party data sharing. Both UIs automate the generation of initial states by integrating user-defined system components with our predefined privacy framework. This reduces the likelihood of incorrectly linking entities when defining initial states manually. The chapter also demonstrates how users interact with these UIs. Finally, [Section 8.3](#) highlights key insights and limitations of the developed UIs.

Chapter 9

Discussion and Conclusion

This chapter concludes the thesis by presenting a brief summary of its scope and focus. It then revisits the research questions and main contributions. The chapter also reflects on key aspects of the proposed approach and outlines directions for future work.

9.1 Summary

We have shown how to formally model privacy policies in a diagrammatic notation, based on bigraphs, that makes them amenable to automated verification through model checking. We believe the visual nature of the approach makes it suitable for a wide audience such as privacy experts who may not have the background knowledge to use complex mathematical modelling tools, *e.g.* those based on π -calculus or theorem proving. The approach is *reusable*, *i.e.* it can capture multiple case studies, and *extensible*, *i.e.* end-users can add/amend reaction rules to respond to changes in the underlying system or the regulatory environment.

9.2 Research Questions Revisited

1. *How do bigraphs contribute to the formal design and verification of privacy-compliant systems?*

Bigraphs contribute to the formal design and verification of privacy-compliant systems by enabling privacy policies to be expressed as rewrite rules within a formal, visual modelling framework. These rules allow flexible integration of privacy requirements

with system-specific behaviours and enable privacy updates to be modelled, *e.g.* to describe consent being given or withdrawn. Diagrammatic notations also can make the model more accessible to a wider range of system designers.

Bigraphs support static analysis through sorting and inductive reasoning. The sorting scheme enforces structural well-formedness and prevents invalid or unintended configurations. Inductive reasoning helps ensure the correctness of privacy rules, ensuring that the intended properties hold even in the presence of state space explosion. However, these analyses are conducted manually, as no automated tool currently supports them.

In addition to static analysis, bigraphs also allow automated dynamic verification of the regulatory notions modelled in this work using the PRISM model checker. These notions are expressed using CTL. The use of a multi-perspective modelling approach enables the generalisation of privacy rules. Consequently, regulatory properties are predefined, eliminating the need for end-users to define these properties themselves. This simplifies the process and significantly reduces the risk of errors when defining privacy properties, improving model accuracy. However, users must specify system-related aspects, *e.g.* sharing data with a third-party system entity.

Priority classes categorise actions by importance, ensuring that high-priority privacy operations, such as consent withdrawal, are handled immediately while lower-priority tasks continue seamlessly. This hierarchical management enables privacy checks to run concurrently with other activities, preventing system blocking and maintaining compliance without rendering the system unusable.

[Table 9.1](#) summarises the key features of our approach in comparison to the formal methods reviewed in [Chapter 3](#), highlighting bigraphs' contribution among related approaches.

2. *Can a reusable, domain-independent bigraph framework be defined where modellers only specify system-specific components while covering different privacy regulations (the GDPR and the CCPA)?*

The cross-border data transfer model consists of 10 predefined privacy reaction rules, 4 of which are reused in the banking system example. The privacy regulations model includes 17 predefined privacy reaction rules. The banking system example is modelled

Table 9.1: The comparison table from [Chapter 3](#), extended with our bigraph-based framework to position our contribution among related approaches.

Approach	Regulation	Automated Verification		Verification Type	Spatial Properties	Cross-Border Transfer Diagrams
PrivacyAPIs [91]	HIPAA	✓	D	×	×	×
MBIPV [139]	GDPR	✓	D	✓	×	✓
Isabelle/HOL [79]	GDPR	PA	ThP	×	×	×
Hublet [53] <i>et al.</i>	GDPR	✓	D	×	×	×
CI [16]	HIPAA, COPPA, GLBA	(-)	(-)	×	×	×
Model-driven [83]	GDPR	(-)	(-)	×	×	×
PrivacyLFP [36]	HIPAA, GLBA	(-)	(-)	×	×	×
DPL [80]	GDPR	PA	D/ThP	×	×	×
P-AOL [14]	GDPR	✓	D	×	×	×
Privacy calculus [133]	GDPR	(-)	S	×	×	×
DiálogoP [131]	GDPR	(-)	(-)	×	×	✓
LPL [46]	GDPR	(-)	(-)	×	×	×
Autonomous agents [136]	GDPR	✓	D	×	×	×
Bigraphical Framework	GDPR, CCPA	✓	D/S	✓	✓	✓

based on the GDPR and uses all 17 rules. Conversely, the healthcare system example is modelled based on the CCPA and requires only 13 of these rules. This demonstrates the framework’s ability to adapt to different domains by reusing predefined rules. While additional case studies are needed, these examples show that the framework can be repurposed with minimal modification, highlighting its reusability.

3. *Can the spatial properties of bigraphs be used to model privacy regulations based on cross-border data transfer constraints?*

The cross-border data transfer model shows how bigraphs’ spatial properties can be used to represent relationships between entities located in different jurisdictions. This capability enables the modelling of privacy regulations—such as the GDPR—that impose constraints on international data transfers, helping specify when such transfers are restricted or permitted.

4. *Which privacy regulation requirements cannot be directly modelled with bigraphs, and what extensions are needed to support them?*

Bigraphs can model key privacy principles such as consent management, third-party data sharing, and spatial constraints like cross-border data transfers. However, they are limited in expressing temporal requirements. For example, rules that impose deadlines—such as the GDPR’s requirement to notify users within 72 hours of a data breach—or enforce data retention limits¹ cannot be directly modelled using bigraphs. This is because bigraphs lack native temporal semantics and thus cannot express requirements based on time or duration. Although there is initial work on extending bigraphs to support temporal requirements using digital clocks [2], the approach has scalability limitations. In particular, as the number of clocks increases, the state space grows significantly, leading to state space explosion. This means that further development is needed to support these regulatory temporal requirements in a scalable and practical way.

5. *How does the sorting discipline help modellers detect and correct structural errors during manual model construction?*

¹Retention limits refer to the principle that personal data must not be retained once its processing purpose is fulfilled and must be deleted or anonymised if it is no longer needed.

Sorting enforces structural correctness by specifying which sorts of entities can be connected or contained within others. This helps modellers detect invalid configurations early during model construction. It allows end-users to catch structural errors that could lead to privacy violations, *e.g.* it ensures that a Contract includes SCCs, rather than any unintended entity, to be considered valid. Sorting also plays a key role in inductive reasoning, where verifying certain properties may require examining the initial state. This means that sorts can support the validity of the reasoning process. Although it is effective but currently manual, as no supporting tools exist yet.

9.3 Review of Thesis Contributions

- **Novel bigraph-based framework for privacy.** We developed a framework that capture the GDPR and the CCPA. The framework consists of two models:
 1. The cross-border data transfer model captures GDPR regulations that restrict international transfers of EU citizens' data. It models three legal bases for transferring data to non-EU countries: adequacy decisions, safeguards such as Standard Contractual Clauses (SCCs), and certification mechanisms. The model includes 10 predefined reaction rules that are customisable by end-users.
 2. The privacy regulations model captures key principles such as providing and withdrawing consent, purpose limitation, right of access, and sharing data with third parties. It consists of 17 privacy rules that users can selectively apply. We also demonstrated how this model can be integrated with the cross-border data transfer model to represent international data-sharing regulations based on the GDPR.
- **Assessment of the effectiveness of our approach via three real-world examples.** We demonstrated the applicability of the first model by modelling WhatsApp's privacy policies. The second model's applicability is illustrated through two examples: a bank notification system inspired by Monzo Bank's privacy policy and modelled according to the GDPR, and a cloud-based home healthcare system based on Fitbit's privacy policies and modelled according to the CCPA. Additionally, we demonstrated the framework's flexibility by integrating the second model with the cross-border data transfer model in

the banking example, to capture international data sharing with third parties.

- **Formal verification of privacy properties through static and dynamic analysis.** We formally analysed our framework by conducting two types of analysis:
 1. **Static analysis.** We defined sorting schemes to ensure the structural correctness of the models and prevent invalid compositions or links. We also conducted inductive reasoning to ensure the correctness of the reaction rules. This helps guarantee that the intended properties hold, despite system-specific initial states or the presence of state space explosion. However, both the inductive reasoning and sorting are performed manually, as no automated support currently exists.
 2. **Model checking.** We introduced a set of predefined privacy properties that are automatically checked via the PRISM model checker. These properties are encoded in CTL, a non-probabilistic temporal logic language, as our specifications are based on regulations and written in the non-probabilistic fragment.
- **User Interface (UIs).** We developed two UIs to automatically generate valid initial states and reduce the risks of incorrectly linking entities. We introduced *Bigraph International Data Transfer Regulation Assistant* (BIDTRA), which generates valid initial states for modelling the GDPR requirements for cross-border data transfers. The second UI is the *Bigraph Privacy Regulations Assistant* (BPRA), which defines correct-by-construction initial states for the second model, which focuses on providing and withdrawing consent, purpose limitation, the right to access, and sharing data with third parties.

9.4 Discussion

Collaboration Between Technical and Legal Experts. Collaboration between developers (who specialise in bigraphs) and privacy experts is essential for fully benefiting from the framework. While developers are responsible for applying bigraphs to ensure privacy compliance, privacy experts and policymakers do not directly engage with the technical model. Instead, they review the formal analysis and provide legal and regulatory guidance.

In certain instances, privacy experts or policymakers must comprehend specific technical dimensions of systems. However, the inherent complexity of these systems can impede

their complete understanding and limit their capacity to offer informed feedback on privacy concerns [52]. To address this challenge, developers can utilise diagrammatic representations to depict the system's structure and compliance with privacy regulations. This visual approach enables experts and policymakers to grasp the system's behaviour without necessitating extensive immersion in technical details.

System Abstraction and Implementation Gap. Our framework abstracts system complexity, *e.g.* modelling a single user instead of multiple users, and excludes system-specific details like data storage or operations, *e.g.* calculating spending in the banking system example. As a result, system-specific processes and interactions must be considered when translating the model into the actual implementation, ensuring correct application of privacy regulations.

Trust and Transparency in Privacy Systems. Informed consent has known limitations [82], *e.g.* users often fail to read privacy notices or fully understand data-collection policies. Our approach can address this by presenting verification results, *e.g.* formal certificates or regulatory approvals, to assure users that the system operates in accordance with regulations [66].

The framework promotes transparency by visually representing data flows and system behaviour, enabling users and designers to understand how data is used and shared. This can foster trust in privacy practices by reassuring data owners that their data is handled according to regulations, even beyond their immediate control. However, complete adherence to all privacy principles requires modelling and rigorous examination of each principle. The framework only guarantees compliance with the privacy aspects addressed in this thesis.

Computational complexity. In our framework, the computational cost arises mainly from: (1) matching reaction rules while generating the labelled transition system, and (2) checking CTL properties. Rule matching is *polynomial* because the rule patterns are *bounded in size*. Let K denote the structural size of the largest rule left-hand side (total entities, sites, and links). In our model, the largest is `acceptAll` (Fig. 6.9) with $K = 14$ (9 entities, 4 sites, 1 link). With K fixed, each match is found in polynomial time in n , *i.e.* the size of the current state, and the degree of the polynomial depends only on K . The total matching cost then scales with the number of candidate matches considered during generation. CTL checking runs in time proportional to the model size and the property length. Since our CTL properties are short and predefined, this phase is effectively linear in the model size. The dominant factor is the

number of reachable states, which can grow exponentially (state-space explosion). To control state growth, we generate a finite, bounded transition system by constraining exploration, *e.g.* limiting the total number of generated states and pruning rewrites via conditions and priority classes. We then verify our CTL properties in PRISM on this bounded transition system. Sorting and inductive reasoning are performed manually and add no algorithmic running time.

Scalability and Performance. While this thesis demonstrates the applicability of the proposed framework through three case studies, we recognise that scalability remains a significant challenge when applying the framework to larger and more complex systems. Like many model-checking approaches, the state space can grow considerably when applied to large systems [1]. However, this is not an issue in our work, as we abstract away system-specific aspects and focus on modelling privacy regulations.

To provide an initial scalability assessment, we measured PRISM’s time and memory usage for our examples. The WhatsApp model required a Resident Set Size (RSS) of 124.8 MB, a Virtual Memory Size (VSZ) of 394.7 MB, and an execution time of 0.002 seconds. The Bank model consumed an RSS of 129.9 MB, a VSZ of 394.7 MB, and an execution time of 0.008 seconds. The cloud-based home healthcare model required an RSS of 125.3 MB, a VSZ of 394.7 MB, and an execution time of 0.002 seconds. These results indicate that PRISM’s time and space requirements are manageable for the models considered.

To support larger deployments, we outline practical techniques to mitigate potential state-space growth: (1) employ conditional rules to prune irrelevant matches; (2) apply property-driven abstraction—model only the parts relevant to the property under check; (3) avoid duplicates by identifying and collapsing symmetric cases; and (4) use instantaneous rules to compress intermediate internal steps that the property does not need to observe (*e.g.* garbage collection), so no extra states are generated. Future work will apply the framework to larger systems to assess scalability and evaluate these techniques for mitigating state-space growth.

9.5 Future Work

While this thesis establishes a foundation for modelling and verifying privacy-compliant systems using bigraphs, several opportunities remain to extend and improve the framework. The following outlines potential directions for future research and development.

1. Extending the framework to capture security aspects, *e.g.* authentication failures and intrusion detection, by modelling security-relevant entities and behaviours. An authentication failure can be defined as a case where an unauthorised system entity attempts to access user data. Such failures can trigger transitions into compromised system states. Intrusion detection can be represented using monitoring components. For instance, when a suspicious pattern is detected (matched), the system transitions into a new state where a monitor issues an alert.
2. Extending the second model to support multiple users, allowing us to model violations where data is, for example, sent to the wrong person.
3. Applying the framework to complex case studies and investigate how the approach scales to these systems. This includes analysing how the framework performs when modelling complex data flows and handling a higher number of entities and interactions.
4. Conducting user studies to evaluate the usability of the framework for system designers and privacy experts. This includes assessing how the diagrammatic notations facilitate communication between them.
5. Extending the second model to capture concepts such as differential privacy [6]. As bigraphs support using probabilities [11], we can compute the probability of private information leakage when aggregate data about a group of users is shared with third-parties.
6. Reusing the framework to model other regulations—*e.g.* the PDPL, GCDPA, APPs, and ADPPA—and extending it to capture common privacy principles across them, *e.g.* modelling retention limits (see [Research Question 4](#)).
7. Improving tool to automatically perform the static analysis (sorting and inductive reasoning) or developing new domain specific languages which target bigraphs for further analysis.

Appendix A

Additional rules for the banking example

We present additional reaction rules for the banking example. These rules outline the main system-specific processes, *e.g.* updating user's name, generating, and storing transactions *etc.* (Figs. A.1, A.2, A.3, A.4, A.5 and A.6):

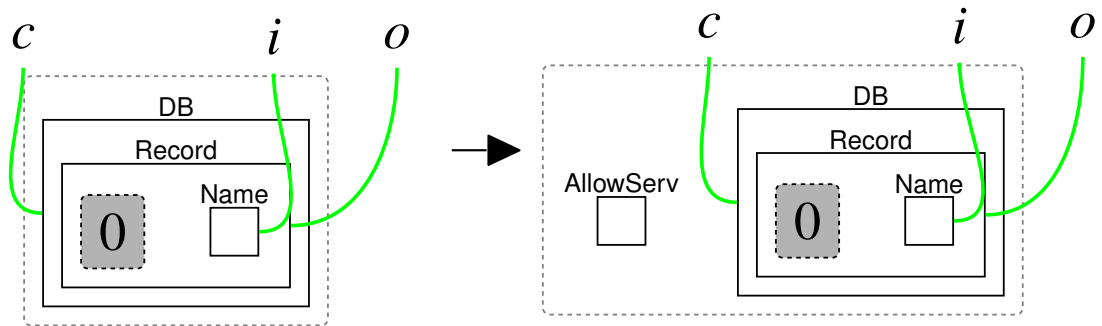


Figure A.1: `provideServices`: once the user registers with the system (their name is stored), then they are able to start using the system's services (`AllowServ`).

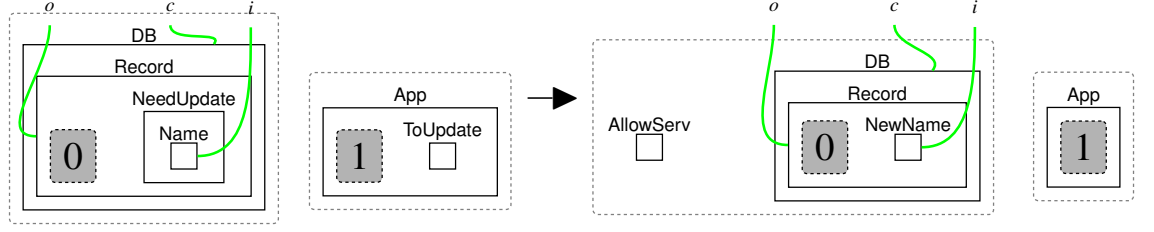


Figure A.2: `updatingName`: the system updates the user's name by replacing `Name` with `NewName`. Note that the new name should have the same permissions of the previous name as it is derived information. The entity `AllowServ` is generated again to allow the user to ask for a new service.

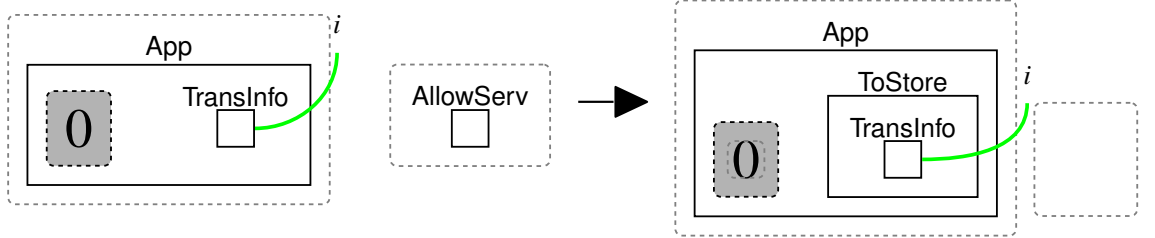


Figure A.3: `generateTrans`: when the transaction is generated, it should be stored in the DB. The entity `ToStore` is created to initialise the storing process in the DB.

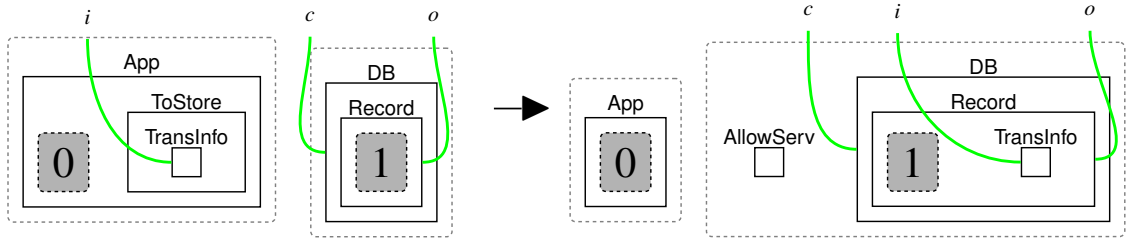


Figure A.4: `storeTransInfo`: the system stores the generated `TransInfo` in the user's `Record`. To allow the user to ask for a new service, `AllowServ` is created.

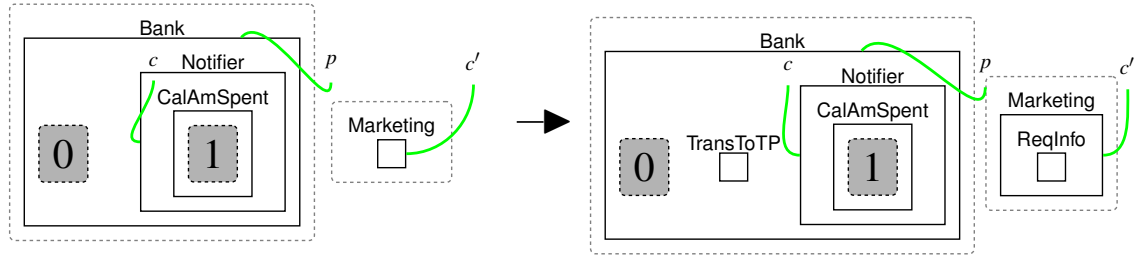


Figure A.5: *branchReqInfo*: Marketing requests the user's transaction details through *ReqInfo*. The entity *TransToTP* indicates that the system will transfer this data to Marketing.

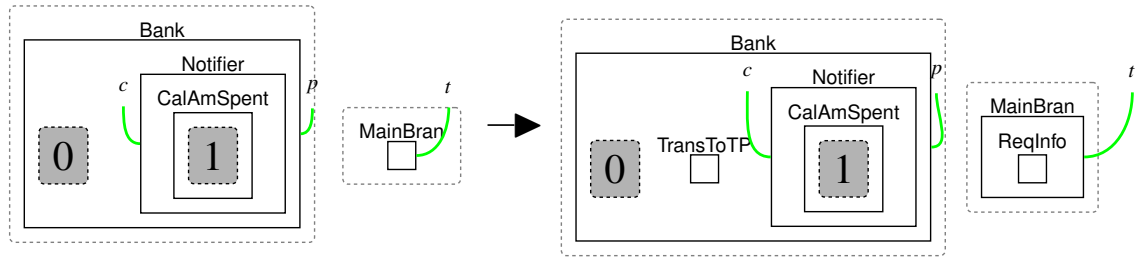


Figure A.6: *mainBrnReq*: MainBran requests the user's transaction details through *ReqAccess*. The entity *TransToTP* indicates that the system will transfer this data to MainBran.

Appendix B

Additional system rules for the healthcare example

The following illustrates the diagrammatic notations of some system-specific rules used to model the healthcare example based on the California Consumer Privacy Act ([Figs. B.1 and B.2](#)):

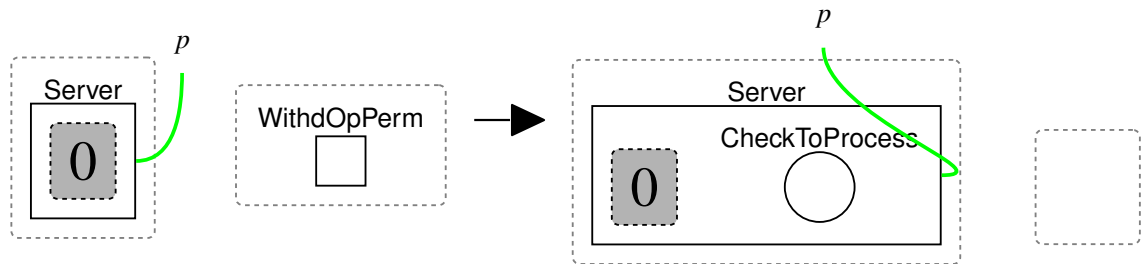


Figure B.1: `checkConOptOut`: after withdrawing the optional permissions, this rule generates the entity `CheckToProcess` to initiate the checking process of the permissions, allowing rule `startCheck` ([Fig. 6.17](#)) to be applied. This, in turn, changes the type of the third-party components to `Comp_F` (`changeTypeComp`, [Fig. 6.18](#)). By doing so, any rule that shares data must match over `Comp_F` to ensure that the data is no longer shared following the user's request to stop sharing their data as shown in [Fig. 6.41](#).

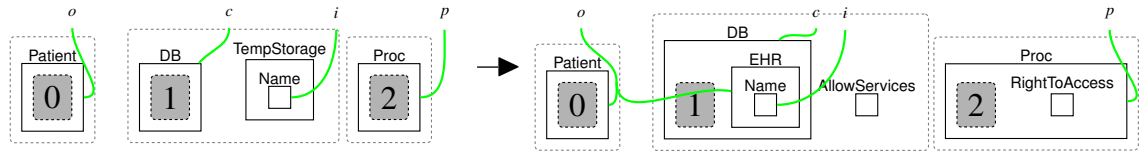


Figure B.2: `storingName`: as the CCPA considers the patient to have consented to share their data, the patient's name is stored directly once the user registers with the system. Unlike rule `storingName` Fig. 6.26, where the entity `CheckPolicy` is replaced with `RightToAccess`, this rule does not include `CheckPolicy` since no consent is required to check. Therefore, it generates the entity `RightToAccess` to allow rule `rightToAccess` (Fig. 6.19) to be applied.

Appendix C

Algebraic definition of the privacy rules

This appendix presents the algebraic definitions (equivalent to the diagrammatic notations) for some of the privacy rules discussed in this paper. The definitions for the remaining rules can be found in [\[5\]](#).

In the following definitions, we use η to denote instantiation maps.

$$\begin{aligned} \text{sendPolicy} &\stackrel{\text{def}}{=} \text{DGE}.(id_0 \mid \text{PrPo}.id_1) \parallel \text{Owner}_o.1 \\ &\rightarrow \text{DGE}.(id_0 \mid \text{PrPo}.id_1) \\ &\quad \parallel \text{Owner}_o.\text{PrPo}.id_2 \\ &\text{with } \eta = [0 \rightarrow 0, 1 \rightarrow 1, 1 \rightarrow 2] \end{aligned}$$

$$\begin{aligned} \text{acceptAll} &\stackrel{\text{def}}{=} \text{Owner}_o.\text{PrPo}.(\text{BasicPerm}.(id \mid \text{Purp}.id) \\ &\quad \mid \text{OptPerm}.(id \mid \text{Purp}.id)) \\ &\quad \parallel \text{Consent}_o.1 \\ &\rightarrow \text{Owner}_o.\text{PrPo}.\text{Accepted}.\text{All} \\ &\quad \parallel \text{Consent}_o.(\text{BasicPerm}.(id \mid \text{Purp}.id) \\ &\quad \mid \text{OptPerm}.(id \mid \text{Purp}.id)) \end{aligned}$$

$$\begin{aligned} \text{acceptBasic} &\stackrel{\text{def}}{=} \text{Owner}_o.\text{PrPo}.(\text{BasicPerm}.(id_0 \mid \text{Purp}.id_1) \\ &\quad \mid \text{OptPerm}.(id_2 \mid \text{Purp}.id_3)) \\ &\quad \parallel \text{Consent}_o.1 \\ &\rightarrow \text{Owner}_o.\text{PrPo}.\text{Accepted}.\text{Basic} \\ &\quad \parallel \text{Consent}_o.(\text{BasicPerm}.(id_0 \mid \text{Purp}.id_1)) \\ &\text{with } \eta = [0 \rightarrow 0, 1 \rightarrow 1] \end{aligned}$$

$$\begin{aligned} \text{closeLinks} &\stackrel{\text{def}}{=} \text{Accepted}.\text{Basic} \parallel \text{Comp}_c.(id \mid P_a \mid P_{op}) \\ &\quad \parallel \text{Ad}_a \parallel \text{OptIn}_{op} \\ &\rightarrow \text{Accepted}.\text{Basic} \parallel \text{Comp}_c.(id \mid /a P_a \\ &\quad \mid /op P_{op}) \\ &\quad \parallel /a \text{Ad}_a \parallel /op \text{OptIn}_{op} \parallel \{a\} \parallel \{op\} \end{aligned}$$

$$\begin{aligned} \text{confirm} &\stackrel{\text{def}}{=} \text{PrPo}.id_0 \parallel \text{Owner}_o.\text{PrPo}.\text{Accepted}.id_1 \\ &\rightarrow \text{PrPo}.(id_0 \mid \text{Confirmed}) \\ &\quad \parallel \text{Owner}_o.\text{ConsApproved} \\ &\text{with } \eta = [0 \rightarrow 0] \end{aligned}$$

$$\begin{aligned} \text{rejectAll} &\stackrel{\text{def}}{=} \text{Owner}_o.\text{PrPo}.id_0 \\ &\rightarrow \text{Owner}_o.\text{PrPo}.\text{Rejected} \\ &\text{with } \eta = [] \end{aligned}$$

Appendix D

User Interfaces

This appendix demonstrates how the BIDTRA and BPRA interfaces are used to construct the initial states for the examples presented in this thesis.

D.1 BIDTRA Interface

In this section, we define the initial state of the WhatsApp example that is presented in [Fig. 5.1](#) ([Figs. D.1](#) and [D.2](#)):

Country Name	Requirement Type	Receiver Type	Receiver Name
Country Name 1: US	Adequacy Decisions	Data Controller	Whats
Country Name 2: Ireland	None	Data Controller	Whats
Country Name 3: Dubia	Certification	Data Controller	FB
Country Name 4: US	Adequacy Decisions	Data Controller	Meta
Country Name 5: Mexico	SCCs	Processor	FB

Figure D.1: BIDTRA interface window for configuring cross-border data transfers after combining the system file with the privacy file. It shows the sender and receiver countries, their types, applicable legal safeguards (*e.g.* SCCs, Certification), and their associated system entities.

D.2 BPRA Interface

This section demonstrates how BPRA is used to build the initial state for the banking example (we show only how the AdCompany is defined as a third party because the definitions of the remaining entities are provided in [Chapter 8](#)) and the home healthcare system example.

D.2.1 Banking Example

The following figure shows how the BPRA interface is used in the banking example to define a third-party entity and assign its associated privileges ([Figs. D.3 and D.4](#)):

D.2.2 Healthcare Example

In this section, we define the initial state of the home healthcare example that is shown in [Fig. 6.38](#) ([Figs. D.5 and D.6](#)):

```

1 big a1 = System.(DC{s0Q}.1 | Meta{uN}.1 | FB{h0U}.1
2               | FB{uZ}.1 | Whats{u_C}.1 | Whats{n_x}.1) ;
3 big a2 = SRType.(Proc{s0Q} | Cont{uN} | Proc{h0U}
4               | Cont{uZ} | Cont{u_C} | Cont{n_x});
5 big a3 = Locations.(L("Ireland").(P{s0Q} | Contract{z_m}.SCCs | Adeq{vB}
6               | Scheme{c_}.(C(1) | C(2) | C(3)) | P{u_C})
7               | L("US").(P{n_x} | P{vB} | P{uN})
8               | L("Dubai").(P{uZ} | Cert{c_}.(ExpiryDate | C(1) | C(2) | C(3)))
9               | L("Mexico").(P{h0U} | Contract{z_m}.SCCs));
10 big s =/uN /h0U /uZ /u_C /n_x /s0Q /vB /z_m /c_ /vB (a1 || a2 || a3);

```

Figure D.2: The initial state generated by the BIDTRA interface as configured in Fig. D.1, reflecting the specified sender and receiver countries, their types, and legal safeguards.

Linking Privacy & System Perspectives

System Agent/Data: User

Type: Owner

Number of Components: 0

Add

Done

Figure D.3: Defining the User entity as a data owner without associated components.

Linking Privacy & System Perspectives

System Agent/Data: AdCompany

Type: Third Party

Number of Components: 1

Add Component

Component Name 1: Marketing

Add Privileges

Save

Authorisation: ☐ Authorised ☒ Unauthorised

Permissions: ☒ Store ☐ Read ☐ Update ☐ Opt-In ☐ Other

Purposes: ☐ Serving ☐ Advertising ☐ Statistics ☒ Other

Select purpose: RE

Figure D.4: BPRA interface window used to configure the third-party entity (AdCompany), specify its component (Marketing), and assign the component's permissions and purposes.

The dialog box 'Linking Privacy & System Perspectives' is shown. It contains the following fields and options:

- System Agent/Data:** Server (dropdown)
- Type:** Processor (dropdown)
- Number of Components:** 1 (text input) with an **Add** button.
- Component Name 1:** DB (dropdown) with an **Add Privileges** button.
- Save** button.
- Authorisation:** ☒ Authorised, ☐ Unauthorised
- Permissions:** ☒ Store, ☐ Read, ☐ Update, ☐ Opt-In, ☐ Other
- Purposes:** ☒ Serving, ☐ Advertising, ☐ Statistics, ☐ Other

(a) Defining Server as a processor (Proc) and specifying its component DB, which is authorised to store data for serving purposes.

The dialog box 'Linking Privacy & System Perspectives' is shown. It contains the following fields and options:

- System Agent/Data:** NRA (dropdown)
- Type:** Third Party (dropdown)
- Number of Components:** 1 (text input) with an **Add** button.
- Component Name 1:** StatDep (dropdown) with an **Add Privileges** button.
- Save** button.
- Authorisation:** ☐ Authorised, ☒ Unauthorised
- Permissions:** ☐ Store, ☐ Read, ☐ Update, ☒ Opt-In, ☐ Other
- Purposes:** ☐ Serving, ☐ Advertising, ☒ Statistics, ☐ Other

(b) Defining NRA as a third party (TP) and specifying its component StatDep, which is unauthorised to access the patient's name but permitted to access activity information for statistical purposes.

The dialog box 'Linking Privacy & System Perspectives' is shown. It contains the following fields and options:

- System Agent/Data:** Patient (dropdown)
- Type:** Owner (dropdown)
- Number of Components:** 0 (text input) with an **Add** button.
- Done** button.

(c) Defining the Patient entity as a data owner without associated components.

The screenshot shows a window titled "Linking Privacy & System Perspectives". It contains the following fields and options:

- System Agent/Data:** A dropdown menu with "Name" selected.
- Type:** A dropdown menu with "Information" selected.
- Number of entities can access:** A text input field with "1" entered.
- Add:** A button.
- Entity Name 1:** A dropdown menu with "Patient" selected.
- Add Privileges:** A button.
- Type:** Radio buttons for "Personal" (selected) and "General".
- Permissions:** Checkboxes for "Store" (checked), "Read" (checked), "Update" (unchecked), "Opt-In" (unchecked), and "Other" (unchecked).
- Purposes:** Checkboxes for "Serving" (checked), "Advertising" (unchecked), "Statistics" (unchecked), and "Other" (unchecked).
- Save:** A button.

(d) Defining Name as personal information, specifying its parent entity (Patient), and the privileges associated with it.

The screenshot shows the same window titled "Linking Privacy & System Perspectives" with the following fields and options:

- System Agent/Data:** A dropdown menu with "ActInfo" selected.
- Type:** A dropdown menu with "Information" selected.
- Number of entities can access:** A text input field with "1" entered.
- Add:** A button.
- Entity Name 1:** A dropdown menu with "Patient" selected.
- Add Privileges:** A button.
- Type:** Radio buttons for "Personal" (unchecked) and "General" (selected).
- Permissions:** Checkboxes for "Store" (unchecked), "Read" (checked), "Update" (unchecked), "Opt-In" (checked), and "Other" (unchecked).
- Purposes:** Checkboxes for "Serving" (checked), "Advertising" (unchecked), "Statistics" (checked), and "Other" (unchecked).
- Save:** A button.

(e) Defining ActInfo as general information, specifying its parent entity (Patient), and the privileges associated with it.

Figure D.5: BPRA interface configurations for the home healthcare example. Each sub-figure illustrates the configuration of a specific system agent/data, including its type, components, and associated permissions and purposes. After configuring each entity, clicking **Save** writes the respective settings to the privacy file.

```

1 # Initial states
2 big a1= DGE.(PrPo. (BasicPerm. ( S{s} | U{up} | R{r}
3                               | Purp.(Serving{se} ))
4                               | OptPerm.(OptIn{op} | Purp.(Ad{ad} | Stat{st})) ) )
5                               | AuthAgent{a}
6                               | /o AccessData{o}
7                               | PersonalInfo{pe});
8 big a2 = Consent{o}.1;
9 big a3= ADType.(Info{v_y}.(P{st} | P{se} | P{op} | P{r}))
10          | Info{c4}.(P{se} | P{r} | P{s} | P{pe})
11          | Owner{o}.1
12          | TP{f8E}.(Comp{b_p}.(P{st} | P{op}))
13          | Proc{b_}.(Comp{g_m}.(P{se} | P{s} | P{a})));
14 big a4= System.(Cloud.(Patient{o}.MobileApp.(ReqUseSyst | ActInfo{v_y} | Name{c4}))
15          | NRA{f8E}.(StatDep{b_p}.1)
16          | Server{b_}.(DB{g_m}.1));
17 big s=/v_y /c4 /b_p /f8E /g_m /b_ /s /up /r /se /op /ad /st /a /o /pe (a1 || a2 || a3 ||
    a4);

```

Figure D.6: The resulting initial state generated after saving all inputs configured in Fig. D.5. Entities highlighted in brown are added manually because they do not have associated types or links.

Bibliography

- [1] T. Abe, T. Ugawa, T. Maeda, and K. Matsumoto, “Reducing state explosion for software model checking with relaxed memory consistency models,” in *Dependable Software Engineering: Theories, Tools, and Applications: Second International Symposium, SETTA 2016, Beijing, China, November 9-11, 2016, Proceedings 2*, Springer, 2016, pp. 118–135.
- [2] M. Albalwe, B. Archibald, and M. Sevegnani, “Modelling real-time systems with bigraphs,” in *Proceedings of the Fourteenth and Fifteenth International Workshop on Graph Computation Models*, Leicester, UK and Enschede, the Netherlands, 18 July 2023 and 9 July 2024, J. Endrullis, D. Grzelak, T. Heindel, and J. Kosiol, Eds., ser. Electronic Proceedings in Theoretical Computer Science, vol. 417, Open Publishing Association, 2025, pp. 96–116. DOI: 10.4204/EPTCS.417.6.
- [3] H. Alshareef, S. Stucki, and G. Schneider, “Refining privacy-aware data flow diagrams,” in *International Conference on Software Engineering and Formal Methods*, Springer, 2021, pp. 121–140.
- [4] H. Alshareef, S. Stucki, and G. Schneider, “Transforming data flow diagrams for privacy compliance.,” in *MODELSWARD*, 2021, pp. 207–215.
- [5] E. Althubiti, *Phd_thesis: Formalising privacy regulations with bigraphs*, Jun. 2025. DOI: 10.5281/zenodo.15642352. [Online]. Available: <https://doi.org/10.5281/zenodo.15642352>.
- [6] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, P. Degano, and C. Palamidessi, “Differential privacy: On the trade-off between utility and information leakage,” in *Formal Aspects of Security and Trust: 8th International Workshop, FAST 2011, Leuven, Belgium, September 12-14, 2011. Revised Selected Papers 8*, Springer, 2012, pp. 39–54.

- [7] AnalogFolk, *Analogfolk*. [Online]. Available: <https://analogfolk.com/contact>.
- [8] AnalogFolk, *Analogfolk appointed as monzo's digital and customer experience agency of record*, 2024. [Online]. Available: https://analogfolk.com/news/analogfolk-appointed-as-monzos-digital-and-customer-experience-agency-of-record?utm_source=chatgpt.com.
- [9] T. Antignac, R. Scandariato, and G. Schneider, "A privacy-aware conceptual model for handling personal data," in *International Symposium on Leveraging Applications of Formal Methods*, Springer, 2016, pp. 942–957.
- [10] B. Archibald, M. Calder, and M. Sevegnani, "Conditional bigraphs," in *International Conference on Graph Transformation*, Springer, 2020, pp. 3–19.
- [11] B. Archibald, M. Calder, and M. Sevegnani, "Probabilistic bigraphs," *Formal Aspects Comput.*, vol. 34, no. 2, pp. 1–27, 2022. DOI: 10.1145/3545180. [Online]. Available: <https://doi.org/10.1145/3545180>.
- [12] B. Archibald and M. Sevegnani, "A bigraphs paper of sorts," in *Graph Transformation - 17th International Conference, ICGT 2024, Held as Part of STAF 2024, Enschede, Netherlands, Proceedings*, To Appear, 2024.
- [13] C. P. Baramashetru, S. L. Tapia Tarifa, and O. Owe, "Assuring gdpr conformance through language-based compliance," in *IFIP International Summer School on Privacy and Identity Management*, Norway: Springer, 2023, pp. 46–63.
- [14] C. P. Baramashetru, S. L. T. Tarifa, and O. Owe, "Integrating data privacy compliance in active object languages," in *Active Object Languages: Current Research Trends*, ser. Lecture Notes in Computer Science, F. S. de Boer, F. Damiani, R. Hähnle, E. B. Johnsen, and E. Kamburjan, Eds., vol. 14360, Switzerland: Springer, 2024, pp. 263–288. DOI: 10.1007/978-3-031-51060-1_10.
- [15] R. Barbuti, N. De Francesco, A. Santone, and G. Vaglini, "Abstract interpretation of trace semantics for concurrent calculi," *Information processing letters*, vol. 70, no. 2, pp. 69–78, 1999.
- [16] A. Barth, A. Datta, J. Mitchell, and H. Nissenbaum, "Privacy and contextual integrity: Framework and applications," in *2006 IEEE Symposium on Security and Privacy (S&P'06)*, 2006, 15 pp.–198. DOI: 10.1109/SP.2006.32.

- [17] M. Y. Becker, A. Malkis, L. Bussard, *et al.*, “S4P: A generic language for specifying privacy preferences and policies,” *Microsoft Research*, vol. 167, 2010.
- [18] S. Benford, M. Calder, T. Rodden, and M. Sevegnani, “On lions, impala, and bigraphs: Modelling interactions in physical/virtual spaces,” *ACM Trans. Comput.-Hum. Interact.*, vol. 23, no. 2, 9:1–9:56, May 2016. DOI: 10.1145/2882784. [Online]. Available: <http://doi.acm.org/10.1145/2882784>.
- [19] L. Birkedal, S. Debois, E. Elsborg, T. Hildebrandt, and H. Niss, “Bigraphical models of context-aware systems,” in *Foundations of Software Science and Computation Structures: 9th International Conference, FOSSACS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-31, 2006. Proceedings 9*, Springer, 2006, pp. 187–201.
- [20] B. Blanchet, “Automatic verification of security protocols in the symbolic model: The verifier proverif,” in *Foundations of security analysis and design VII*, Springer, 2013, pp. 54–87.
- [21] F. D. Boer *et al.*, “A survey of active object languages,” *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–39, 2017.
- [22] Bureau Of Experts At The Council Of Ministers, *Personal Data Protection Law*, 2023. [Online]. Available: <https://laws.boe.gov.sa/BoeLaws/Laws/LawDetails/b7cf8e89-828e-4994-b167-adaa00e37188/1>.
- [23] L. Cardelli, G. Ghelli, and A. D. Gordon, “Secrecy and group creation,” in *International Conference on Concurrency Theory*, Springer, 2000, pp. 365–379.
- [24] F. Casalini and J. L. González, “Trade and cross-border data flows,” *OECD Trade Policy Papers*, 2019.
- [25] F. Casalini, J. L. González, and T. Nemoto, “Mapping commonalities in regulatory approaches to cross-border data transfers,” *OECD Trade Policy Papers*, 2021.
- [26] A. Cavoukian, “Privacy by design: The definitive workshop. a foreword by ann cavoukian, ph. d,” *Identity in the Information Society*, vol. 3, no. 2, pp. 247–251, 2010.

- [27] A. Cavoukian, S. Taylor, and M. E. Abrams, “Privacy by design: Essential for organizational accountability and strong business practices,” *Identity in the Information Society*, vol. 3, pp. 405–413, 2010.
- [28] Y. Cheng, J. Park, and R. Sandhu, “Preserving user privacy from third-party applications in online social networks,” in *Proceedings of the 22nd international conference on world wide web*, 2013, pp. 723–728.
- [29] S. Chong *et al.*, *Report on the NSF workshop on formal methods for security*, 2016. arXiv: 1608.00678 [cs.CR]. [Online]. Available: <https://arxiv.org/abs/1608.00678>.
- [30] A. Church, “A set of postulates for the foundation of logic,” *Annals of Mathematics*, vol. 34, no. 4, pp. 839–864, 1933. [Online]. Available: <http://www.jstor.org/stable/1968702> (visited on 01/18/2025).
- [31] E. M. Clarke and E. A. Emerson, “Design and synthesis of synchronization skeletons using branching time temporal logic,” in *Workshop on logic of programs*, Springer, 1981, pp. 52–71.
- [32] CONGRESS.GOV, *American Data Privacy and Protection Act*, 2024. [Online]. Available: <https://www.congress.gov/bill/117th-congress/house-bill/8152/text>.
- [33] J. W. Cutler *et al.*, “Cedar: A new language for expressive, fast, safe, and analyzable authorization,” *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA1, Apr. 2024. DOI: 10.1145/3649835. [Online]. Available: <https://doi.org/10.1145/3649835>.
- [34] M. Deng, M. Petkovic, M. Nalin, and I. Baroni, “A home healthcare system in the cloud—addressing security and privacy challenges,” in *2011 IEEE 4th International Conference on Cloud Computing*, IEEE, 2011, pp. 549–556.
- [35] L. A. Dennis, M. Fisher, M. P. Webster, and R. H. Bordini, “Model checking agent programming languages,” *Automated software engineering*, vol. 19, pp. 5–63, 2012. [Online]. Available: <https://doi.org/10.1007/s10515-011-0088-x>.

- [36] H. DeYoung, D. Garg, L. Jia, D. Kaynar, and A. Datta, “Experiences in the logical specification of the HIPAA and GLBA privacy laws,” in *Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society*, ser. WPES ’10, Chicago, Illinois, USA: Association for Computing Machinery, 2010, pp. 73–82. DOI: 10.1145/1866919.1866930.
- [37] *Dom4j*. [Online]. Available: <https://dom4j.github.io/>.
- [38] C. Dwork, “Differential privacy,” in *International Colloquium on Automata, Languages, and Programming*, Springer, 2006, pp. 1–12.
- [39] F. Eigner and M. Maffei, “Differential privacy by typing in security protocols,” in *2013 IEEE 26th Computer Security Foundations Symposium*, IEEE, 2013, pp. 272–286.
- [40] European Data Protection Board, *Guidelines 07/2022 on certification as a tool for transfers*, 2023. [Online]. Available: https://www.edpb.europa.eu/our-work-tools/our-documents/guidelines/guidelines-072022-certification-tool-transfers_en.
- [41] European Data Protection Board, *International data transfers*. [Online]. Available: https://www.edpb.europa.eu/sme-data-protection-guide/international-data-transfers_en.
- [42] Fitbit app, *Fitbit privacy policy*, 2024. [Online]. Available: <https://support.google.com/product-documentation/answer/14815921#anchor2>.
- [43] S. Gay and M. Hole, “Subtyping for session types in the pi calculus,” *Acta Informatica*, vol. 42, no. 2, pp. 191–225, 2005.
- [44] GDPR.EU, *Data sharing and GDPR compliance: Bounty UK shows what not to do*, 2024. [Online]. Available: <https://gdpr.eu/data-sharing-bounty-fine/>.
- [45] Georgia General Assembly, *SB 394 Georgia Computer Data Privacy Act*. [Online]. Available: <https://www.legis.ga.gov/legislation/61417>.
- [46] A. Gerl, N. Bennani, H. Kosch, and L. Brunie, “LPL, Towards a GDPR-Compliant Privacy Language: Formal Definition and Usage,” in *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVII*, A. Hameurlain and R. Wagner, Eds.

- Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 41–80. DOI: 10.1007/978-3-662-57932-9_2.
- [47] Google Fitbit, *Help enhance population health with data from fitbit devices*, 2025. [Online]. Available: <https://enterprise.fitbit.com/blog/help-enhance-population-health-with-data-from-fitbit-devices/>.
- [48] D. S. Guamán, J. M. Del Alamo, and J. C. Caiza, “GDPR compliance assessment for cross-border personal data transfers in Android apps,” *IEEE Access*, vol. 9, pp. 15 961–15 982, 2021.
- [49] D. S. Guamán, D. Rodriguez, J. M. del Alamo, and J. Such, “Automated GDPR compliance assessment for cross-border personal data transfers in Android applications,” *Computers & Security*, vol. 130, p. 103 262, 2023.
- [50] R. Heckel and G. Taentzer, *Graph transformation for software engineers*. Springer, 2020, vol. 10, pp. 978–3.
- [51] M. Hilty, A. Pretschner, D. Basin, C. Schaefer, and T. Walter, “A policy language for distributed usage control,” in *Computer Security—ESORICS 2007: 12th European Symposium On Research In Computer Security, Dresden, Germany, September 24–26, 2007. Proceedings 12*, Springer, 2007, pp. 531–546.
- [52] S. A. Horstmann *et al.*, ““Those things are written by lawyers, and programmers are reading that.” mapping the communication gap between software developers and privacy experts,” *Proceedings on Privacy Enhancing Technologies*, vol. 1, pp. 151–170, 2024.
- [53] F. Hublet, D. Basin, and S. Krstić, “Enforcing the GDPR,” in *Computer Security – ESORICS 2023*, G. Tsudik, M. Conti, K. Liang, and G. Smaragdakis, Eds., Cham: Springer Nature Switzerland, 2024, pp. 400–422.
- [54] F. Hublet, D. Basin, and S. Krstić, “User-controlled privacy: Taint, track, and control,” *Proceedings on Privacy Enhancing Technologies*, 2024.
- [55] M. A. Husnoo, A. Anwar, R. K. Chakraborty, R. Doss, and M. J. Ryan, “Differential privacy for IoT-enabled critical infrastructure: A comprehensive survey,” *IEEE Access*, vol. 9, pp. 153 276–153 304, 2021. DOI: 10.1109/ACCESS.2021.3124309.

- [56] Information Commissioner's Office (ICO), *How do you determine whether you are a controller or processor?* [Online]. Available: <https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/controllers-and-processors/controllers-and-processors/how-do-you-determine-whether-you-are-a-controller-or-processor/>.
- [57] Information Commissioner's Office (ICO), *Consent*. [Online]. Available: <https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/lawful-basis/a-guide-to-lawful-basis/consent/#ib4>.
- [58] Information Commissioner's Office (ICO), *What is valid consent?* [Online]. Available: <https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/lawful-basis/consent/what-is-valid-consent/>.
- [59] International Organization for Standardization, *ISO/IEC 27701:2019 - security techniques — extension to ISO/IEC 27001 and ISO/IEC 27002 for privacy information management — requirements and guidelines*, 2019. [Online]. Available: <https://www.iso.org/standard/71670.html>.
- [60] Intersoft Consulting, *Art.12 GDPR Transparent information, communication and modalities for the exercise of the rights of the data subject*. [Online]. Available: <https://gdpr-info.eu/art-12-gdpr/>.
- [61] Intersoft Consulting, *Art.15 GDPR Right of access by the data subject*. [Online]. Available: <https://gdpr-info.eu/art-15-gdpr/>.
- [62] Intersoft Consulting, *Art.17 GDPR Right to erasure ('right to be forgotten')*, n.d. [Online]. Available: <https://gdpr-info.eu/art-17-gdpr/>.
- [63] Intersoft Consulting, *Art.21 GDPR Right to object*, n.d. [Online]. Available: <https://gdpr-info.eu/art-21-gdpr/>.
- [64] Intersoft Consulting, *Art.32 GDPR Security of processing*, n.d. [Online]. Available: <https://gdpr-info.eu/art-32-gdpr/>.
- [65] Intersoft Consulting, *Art.4 GDPR Definitions*. [Online]. Available: <https://gdpr-info.eu/art-4-gdpr/>.

- [66] Intersoft Consulting, *Art.42 GDPR Certification*. [Online]. Available: <https://gdpr-info.eu/art-42-gdpr/>.
- [67] Intersoft Consulting, *Art.45 GDPR Transfers on the basis of an adequacy decision*, n.d. [Online]. Available: <https://gdpr-info.eu/art-45-gdpr/>.
- [68] Intersoft Consulting, *Art.46 GDPR Transfers subject to appropriate safeguards*, n.d. [Online]. Available: <https://gdpr-info.eu/art-46-gdpr/>.
- [69] Intersoft Consulting, *Art.5 GDPR Principles relating to processing of personal data*, <https://gdpr-info.eu/art-5-gdpr/>.
- [70] Intersoft Consulting, *Art.6 GDPR Lawfulness of processing*. [Online]. Available: <https://gdpr-info.eu/art-6-gdpr/>.
- [71] Intersoft Consulting, *Art.7 GDPR Conditions for consent*, n.d. [Online]. Available: <https://gdpr-info.eu/art-7-gdpr/>.
- [72] Intersoft Consulting, *Chapter 5 Transfers of personal data to third countries or international organisations*, n.d. [Online]. Available: <https://gdpr-info.eu/chapter-5/>.
- [73] Intersoft Consulting, *GDPR Consent*, 2024. [Online]. Available: <https://gdpr-info.eu/issues/consent/>.
- [74] Intersoft Consulting, *GDPR Fines / Penalties*, n.d. [Online]. Available: <https://gdpr-info.eu/issues/fines-penalties>.
- [75] Intersoft Consulting, *General Data Protection Regulation GDPR*, 2024. [Online]. Available: <https://gdpr-info.eu/>.
- [76] Irish Data Protection Commission, *Irish data protection commission fines tiktok €530 million and orders corrective measures following inquiry into transfers of eea user data to china*, 2025. [Online]. Available: <https://www.dataprotection.ie/en/news-media/latest-news/irish-data-protection-commission-fines-tiktok-eu530-million-and-orders-corrective-measures-following>.
- [77] R. Joshaghani and H. Mehrpouyan, “A model-checking approach for enforcing purpose-based privacy policies,” in *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, IEEE, 2017, pp. 178–179.

- [78] L. Kagal, “Rei : A policy language for the me-centric project,” HP Labs, Tech. Rep., Sep. 2002. [Online]. Available: <http://www.hpl.hp.com/techreports/2002/HPL-2002-270.html>.
- [79] F. Kammüller, “Formal modeling and analysis of data protection for GDPR compliance of iot healthcare systems,” in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2018, pp. 3319–3324.
- [80] F. Karami, D. Basin, and E. B. Johnsen, “DPL: A language for GDPR enforcement,” in *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*, IEEE, 2022, pp. 112–129.
- [81] D. Kouzapas and A. Philippou, “Privacy by typing in the π -calculus,” *Log. Methods Comput. Sci.*, vol. 13, no. 4, 2017. DOI: 10.23638/LMCS-13(4:27)2017. [Online]. Available: [https://doi.org/10.23638/LMCS-13\(4:27\)2017](https://doi.org/10.23638/LMCS-13(4:27)2017).
- [82] F. Kreuter, G.-C. Haas, F. Keusch, S. Bähr, and M. Trappmann, “Collecting survey and smartphone sensor data with an app: Opportunities and challenges around privacy and informed consent,” *Social Science Computer Review*, vol. 38, no. 5, pp. 533–549, 2020.
- [83] S. Krstić, H. Nguyen, and D. Basin, “Model-driven privacy,” *Proceedings on Privacy Enhancing Technologies*, vol. 2024, no. 1, pp. 314–329, 2024.
- [84] N. Kshetri, “Big data’s impact on privacy, security and consumer welfare,” *Telecommunications Policy*, vol. 38, no. 11, pp. 1134–1145, 2014.
- [85] M. Kwiatkowska, G. Norman, and D. Parker, “PRISM 4.0: Verification of probabilistic real-time systems,” in *Proc. 23rd International Conference on Computer Aided Verification (CAV’11)*, G. Gopalakrishnan and S. Qadeer, Eds., ser. LNCS, vol. 6806, US: Springer, 2011, pp. 585–591.
- [86] D. Le Métayer, “A formal privacy management framework,” in *Formal Aspects in Security and Trust*, P. Degano, J. Guttman, and F. Martinelli, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 162–176.
- [87] X. Li, S. Liu, F. Wu, S. Kumari, and J. J. P. C. Rodrigues, “Privacy preserving data aggregation scheme for mobile edge computing assisted iot applications,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4755–4763, 2019. DOI: 10.1109/JIOT.2018.2874473.

- [88] J. N. Lisowski, “California data privacy law and automated decision-making,” *J. Corp. L.*, vol. 49, p. 701, 2023.
- [89] J. Lu, Z. Huang, and C. Ke, “Verification of behavior-aware privacy requirements in web services composition,” *J. Softw.*, vol. 9, no. 4, pp. 944–951, 2014.
- [90] L. I. Mancini, “Formal verification of privacy in pervasive systems,” Ph.D. dissertation, University of Birmingham, 2015.
- [91] M. J. May, C. A. Gunter, and I. Lee, “Privacy APIs: Access control techniques to analyze and verify legal privacy policies,” in *19th IEEE Computer Security Foundations Workshop (CSFW’06)*, IEEE, 2006, 13–pp.
- [92] J. Meseguer, “Twenty years of rewriting logic,” *J. Log. Algebr. Program.*, vol. 81, no. 7-8, pp. 721–781, 2012. DOI: 10.1016/j.jlap.2012.06.003. [Online]. Available: <https://doi.org/10.1016/j.jlap.2012.06.003>.
- [93] Microsoft, *9.8 rights management services*, Jun 14 2022. [Online]. Available: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-wpo/1c11d4ee-98c7-46a9-99e0-d6899d4522b9.
- [94] R. Milner, “Bigraphs and their algebra,” *Electronic Notes in Theoretical Computer Science*, vol. 209, pp. 5–19, 2008.
- [95] R. Milner, *Communicating and mobile systems: the π -calculus*. Cambridge university press, 1999.
- [96] R. Milner, *The space and motion of communicating agents*. Cambridge: Cambridge University Press, 2009.
- [97] Monzo Bank, *Version 1.22 customer privacy notice*, 2024. [Online]. Available: <https://monzo.com/legal/privacy-notice/#your-rights->.
- [98] V. Morel and R. Pardo, “SoK: Three Facets of Privacy Policies,” in *Proceedings of the 19th Workshop on Privacy in the Electronic Society*, ser. WPES’20, Virtual Event, USA: Association for Computing Machinery, 2020, pp. 41–56. DOI: 10.1145/3411497.3420216.
- [99] L. d. Moura and S. Ullrich, “The lean 4 theorem prover and programming language,” in *Automated Deduction – CADE 28*, A. Platzer and G. Sutcliffe, Eds., Cham: Springer International Publishing, 2021, pp. 625–635.

- [100] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized Business Review*, p. 21 260, 2008.
- [101] D. Nardi and R. J. Brachman, “An introduction to description logics,” in *The Description Logic Handbook: Theory, Implementation and Applications*, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge University Press, 2007, pp. 1–44.
- [102] M. S. Nawaz, M. Malik, Y. Li, M. Sun, and M. I. U. Lali, “A survey on theorem provers in formal methods,” *CoRR*, vol. abs/1912.03028, 2019. [Online]. Available: <http://arxiv.org/abs/1912.03028>.
- [103] T. Nipkow, M. Wenzel, and L. C. Paulson, *Isabelle/HOL: a proof assistant for higher-order logic*. Springer, 2002.
- [104] A. Nuradiansyah, “Reasoning in description logic ontologies for privacy management,” *KI-Künstliche Intelligenz*, vol. 34, no. 3, pp. 411–415, 2020.
- [105] *Nusmv: A new symbolic model checker*, 2023. [Online]. Available: <https://nusmv.fbk.eu/index.html>.
- [106] Office of the Australian Information Commissioner (OAIC), *Australian Privacy Principles*. [Online]. Available: <https://www.oaic.gov.au/privacy/australian-privacy-principles>.
- [107] Office of the Australian Information Commissioner (OAIC), *Chapter 1: APP 1 — Open and transparent management of personal information*, 2024. [Online]. Available: <https://www.oaic.gov.au/privacy/australian-privacy-principles/australian-privacy-principles-guidelines/chapter-12-app-12-access-to-personal-information>.
- [108] Office of the Australian Information Commissioner (OAIC), *Chapter 6: APP 6 Use or disclosure of personal information*, 2019. [Online]. Available: <https://www.oaic.gov.au/privacy/australian-privacy-principles/australian-privacy-principles-guidelines/chapter-6-app-6-use-or-disclosure-of-personal-information#holds-use-disclose-and-purpose>.

- [109] M. M. H. Onik, S. Aich, J. Yang, C.-S. Kim, and H.-C. Kim, “Blockchain in healthcare: Challenges and solutions,” in *Big data analytics for intelligent healthcare management*, Elsevier, 2019, pp. 197–226.
- [110] R. Pardo and D. Le Métayer, “Analysis of privacy policies to enhance informed consent,” in *Data and Applications Security and Privacy XXXIII*, S. N. Foley, Ed., Cham: Springer International Publishing, 2019, pp. 177–198.
- [111] H. Pascual, J. M. Del Alamo, D. Rodriguez, and J. C. Dueñas, “Hunter: Tracing anycast communications to uncover cross-border personal data transfers,” *Computers & Security*, vol. 141, p. 103 823, 2024.
- [112] E. Pereira, C. M. Kirsch, J. B. de Sousa, and R. Sengupta, “Bigactors: A model for structure-aware computation,” in *ACM/IEEE 4th International Conference on Cyber-Physical Systems (with CPS Week 2013), ICCPS '13, April 8-11, 2013*, C. Lu, P. R. Kumar, and R. Stoleru, Eds., Philadelphia, PA, USA: ACM, 2013, pp. 199–208. DOI: 10.1145/2502524.2502551.
- [113] G. M. Pillai, A. Suresh, E. Gupta, V. Ganapathy, and A. Patra, “Privadome: Delivery drones and citizen privacy,” *Proceedings on Privacy Enhancing Technologies*, 2024.
- [114] S. Quach, P. Thaichon, K. D. Martin, S. Weaven, and R. W. Palmatier, “Digital technologies: Tensions in privacy and data,” *Journal of the Academy of Marketing Science*, vol. 50, no. 6, pp. 1299–1323, 2022.
- [115] M. N. I. Sarker, M. Wu, and M. A. Hossin, “Smart governance through bigdata: Digital transformation of public agencies,” in *2018 international conference on artificial intelligence and big data (ICAIBD)*, IEEE, 2018, pp. 62–70.
- [116] T. Sato and Y. Minamide, “Formalization of differential privacy in Isabelle/HOL,” in *Proceedings of the 14th ACM SIGPLAN International Conference on Certified Programs and Proofs*, ser. CPP ’25, Denver, CO, USA: Association for Computing Machinery, 2025, pp. 67–82. DOI: 10.1145/3703595.3705875. [Online]. Available: <https://doi.org/10.1145/3703595.3705875>.
- [117] M. Sevegnani and M. Calder, “BigraphER: Rewriting and analysis engine for bi-graphs,” in *International Conference on Computer Aided Verification*, Springer, 2016, pp. 494–501.

- [118] M. Sevegnani, M. Kabac, M. Calder, and J. McCann, “Modelling and verification of large-scale sensor network infrastructures,” in *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2018, pp. 71–81. DOI: 10.1109/ICECCS2018.2018.00016.
- [119] P. Sharma, S. Namasudra, N. Chilamkurti, B.-G. Kim, and R. Gonzalez Crespo, “Blockchain-based privacy preservation for IoT-enabled healthcare system,” *ACM Trans. Sen. Netw.*, vol. 19, no. 3, Mar. 2023. [Online]. Available: <https://doi.org/10.1145/3577926>.
- [120] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016. DOI: 10.1109/JIOT.2016.2579198.
- [121] D. J. Solove, “A taxonomy of privacy,” *U. Pa. L. Rev.*, vol. 154, p. 477, 2005.
- [122] N. I. of Standards and Technology, “Privacy framework: A tool for improving privacy through enterprise risk management,” NIST, Tech. Rep. NISTIR 8062, 2020, Accessed: 2024-05-04. [Online]. Available: <https://www.nist.gov/privacy-framework>.
- [123] State of California Department of Justice, *A. general information about the ccpa*, 2025. [Online]. Available: <https://oag.ca.gov/privacy/ccpa#sectiona>.
- [124] State of California Department of Justice, *B. right to opt-out of sale or sharing*, 2025. [Online]. Available: <https://oag.ca.gov/privacy/ccpa#sectionb>.
- [125] State of California Department of Justice, *C. requests to know*, 2025. [Online]. Available: <https://oag.ca.gov/privacy/ccpa#sectionc>.
- [126] State of California Department of Justice, *California Consumer Privacy Act (CCPA)*, 2024. [Online]. Available: <https://oag.ca.gov/privacy/ccpa>.
- [127] State of California Department of Justice, *D. requests to delete*, 2025. [Online]. Available: <https://oag.ca.gov/privacy/ccpa#sectiond>.
- [128] V. T. Ta, *Privacy by design: On the formal design and conformance check of personal data protection policies and architectures*, 2018. arXiv: 1511.01249 [cs.CR]. [Online]. Available: <https://arxiv.org/abs/1511.01249>.

- [129] N. B. Truong, K. Sun, G. M. Lee, and Y. Guo, “GDPR-compliant personal data management: A blockchain-based solution,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1746–1761, 2019.
- [130] M. C. Tschantz and J. M. Wing, “Formal Methods for Privacy,” in *FM 2009: Formal Methods*, A. Cavalcanti and D. R. Dams, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–15.
- [131] E. Vanezi, G. M. Kapitsaki, D. Kouzapas, A. Philippou, and G. A. Papadopoulos, “Diálogop-a language and a graphical tool for formally defining GDPR purposes,” in *International Conference on Research Challenges in Information Science*, Springer, 2020, pp. 569–575.
- [132] E. Vanezi, G. M. Kapitsaki, and A. Philippou, “What’s your purpose? an approach to incorporating GDPR purposes into requirements analysis,” in *ICISSP*, 2024, pp. 907–914.
- [133] E. Vanezi, D. Kouzapas, G. M. Kapitsaki, and A. Philippou, “Towards GDPR compliant software design: A formal framework for analyzing system models,” in *International Conference on Evaluation of Novel Approaches to Software Engineering*, Springer, 2019, pp. 135–162.
- [134] WhatsApp, *Who is providing your whatsapp services*, 2023. [Online]. Available: https://faq.whatsapp.com/523679699550284/?helpref=hc_fnave&locale=en_U.
- [135] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, “Formal methods: Practice and experience,” *ACM computing surveys (CSUR)*, vol. 41, no. 4, pp. 1–36, 2009.
- [136] M. Xu, L. A. Dennis, and M. A. Mustafa, “Safeguard privacy for minimal data collection with trustworthy autonomous agents,” in *AAMAS’24: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2024, pp. 1966–1974.
- [137] J. Yang, K. Yessenov, and A. Solar-Lezama, “A language for automatically enforcing privacy policies,” in *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL ’12, Philadelphia,

- PA, USA: Association for Computing Machinery, 2012, pp. 85–96. DOI: 10.1145/2103656.2103669.
- [138] A. Yazdinejad, G. Srivastava, R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and M. Aledhari, “Decentralized authentication of distributed patients in hospital networks using blockchain,” *IEEE journal of biomedical and health informatics*, vol. 24, no. 8, pp. 2146–2156, 2020.
- [139] T. Ye, Y. Zhuang, and G. Qiao, “MBIPV: A model-based approach for identifying privacy violations from software requirements,” *Software and Systems Modeling*, vol. 22, no. 4, pp. 1251–1280, 2023.
- [140] F. Zemler and M. Westner, “Blockchain and GDPR: Application scenarios and compliance requirements,” in *2019 Portland International Conference on Management of Engineering and Technology (PICMET)*, IEEE, 2019, pp. 1–8.
- [141] Y. Zhao, H. Haddadi, S. Skillman, S. Enshaeifar, and P. Barnaghi, “Privacy-preserving activity and health monitoring on databox,” in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, 2020, pp. 49–54.