Modelling Privacy Compliance in Cross-border Data Transfers with Bigraphs

Ebtihal Althubiti
Northern Border University, Arar, Saudi Arabia
ebtihal.althubiti@nbu.edu.sa
University of Glasgow, Glasgow, United Kingdom
e.althubiti.1@research.gla.ac.uk

Michele Sevegnani
University of Glasgow, Glasgow, United Kingdom
michele.sevegnani@glasgow.ac.uk

Due to the advancement of information technologies, users' data is collected and shared among several organisations worldwide. The cross-border transfer of the users' data can threaten their privacy, especially when the destination country lacks adequate protection measures. Many privacy legislation are imposed to regulate the international transfer of data, particularly the European General Data Protection Regulation (GDPR). Several companies are fined for their failure to comply with these regulations. To help these companies ensure compliance, we propose a privacy framework that proves systems compliance with these regulations. The proposed framework is defined based on Milner's Bigraphical Reactive Systems (BRSs), which are a universal formalism that models systems based on the spatial (placement) and non-spatial (connectivity) relationships between entities. BRSs can evolve over time via rewriting user-specified reaction rules. The key feature of BRSs is their ability to define the reaction rules algebraically and diagrammatically. In this paper, we mainly rely on diagrammatic notations to enable end-users and privacy experts with less background in formal modelling to use the framework. Our privacy framework consists of predefined privacy reaction rules that model the GDPR requirements for international data transfers. To prove these requirements within the systems, we define a set of properties encoded using Computation Tree Logic (CTL) to be checked using the PRISM model checker. We instantiate the framework by using WhatsApp privacy policies as an example.

1 Introduction

Transferring data across various jurisdictions worldwide is one of the requirements for advancing digital systems [26]. For example, WhatsApp states that they need to share users' data with Meta's data centres and Facebook's branches around the world to improve their services [32]. Transferring data to different countries can potentially threaten users' privacy as the receiver country may not use sufficient mechanisms to protect the data, *e.g.* encryption techniques [9].

Several regulations have been imposed to restrict such transfers, for example, the European Union (EU) General Data Protection Regulation (GDPR) [14], and the Australian Privacy Principles (APPs) [4]. The main aim of these regulations is to guarantee that the recipient country provides an adequate data protection level compared to the sender country's protection level.

According to the GDPR, there are two main ways to restrict data transfer towards non-EU countries: transferring based on the *adequacy decision* or transferring based on *appropriate safeguards* [8]. The former permits the transfer to specific countries that have an adequate level of data protection as specified by the European Commission. The latter requires safeguards, *e.g. Standard Contractual Clauses (SCCs)* or a certification, if the adequacy decision does not cover the receiver country. Besides these two

Submitted to:
GCM 2024

methods, the GDPR identifies some exceptional cases where transferring data is permitted, *e.g.* if the transfer is important for the public interest. We discuss all these ways further in Section 2.

Organisations must adhere to the above requirements to transfer data from the EU to third countries. Otherwise, they can be at risk of being fined by an EU authority. For example, Facebook has been fined 1.2 billion euros by the Irish Data Protection Authority (IE DPA) for not adhering to the GDPR requirements for transferring personal data from EU countries to the US [1] [6]. Such breaches could occur for many reasons. One is that the regulations are text-based and subject to update, which can create challenges for developers to convert them into technical requirements [16]. Another reason is that tools used by developers or supervisory authorities to check compliance are not automated, *e.g.* Transfer Impact Assessment (TIA) [29], [16]. This means there is a need for an automated approach that enables organisations to ensure their systems comply with the privacy regulations and serve as evidence of their compliance for supervisory authorities.

Formal methods are mathematical techniques that have been used in various domains to solve these issues thanks to their ability to model systems and provide a rigorous analysis of properties of interest such as security and safety [34]. They can also be used as proof of fulfilling these specifications because they enable automated and comprehensive verification processes. To the best of our knowledge, they have yet to be used to prove the systems' compliance with the GDPR requirements for cross-border personal data transfers. In this paper, we propose a framework based on Milner's Bigraphs [24] to model systems and prove for the first time their compliance to these aspects of privacy regulations. Bigraphs are a universal formalism for the modelling of interacting systems that evolve in their connectivity and space via rewriting rules called *reaction rules*. Compared to other formalisms, reaction rules are user-specified allowing for greater flexibility to model a wide variety of systems. Bigraphs and reaction rules can be specified algebraically or through an equivalent diagram notation. This feature enables system designers to collaborate with privacy experts, *e.g.* privacy lawyers, who might not be expert in formal modelling and verification. Another advantage of using bigraphs is that modelling the flow of data among different jurisdictions requires modelling the spatial relationships between entities, and bigraphs can natively express spatial properties such as containment relation.

Our approach is described in Fig. 1. The privacy aspects of our framework are predefined while system-specific aspects have to be specified for each application. The framework also provides a set of privacy properties that should be verified, *e.g.* providing safeguards, the validity of them *etc*. These privacy properties are expressed using the Computation Tree Logic (CTL) and can be checked *automatically* using off-the-shelf verification tools such as PRISM [21].

We provide the following contributions:

- We define a bigraph framework to model the GDPR requirements for cross-border data transfers. The framework captures *data transfers based on adequacy decisions and appropriate safeguards (Standard Contractual Clauses (SCCs) and certification mechanism)*. To the best of our knowledge, this is the first framework that models the GDPR notion of *international data transfers*.
- We apply the framework to an example based on WhatsApp's privacy policies and showcase the capabilities of bigraphs to model complex systems diagrammatically.
- We define the GDPR requirements for international data transfers using CTL and show how they can be proven *formally and automatically* using PRISM.

The rest of this paper is structured as follows: Section 2 presents the GDPR requirements for international data transfers, while Section 3 gives an overview of Bigraphical Reactive Systems. Section 4 and

[1] This is before considering the US as an adequate country.

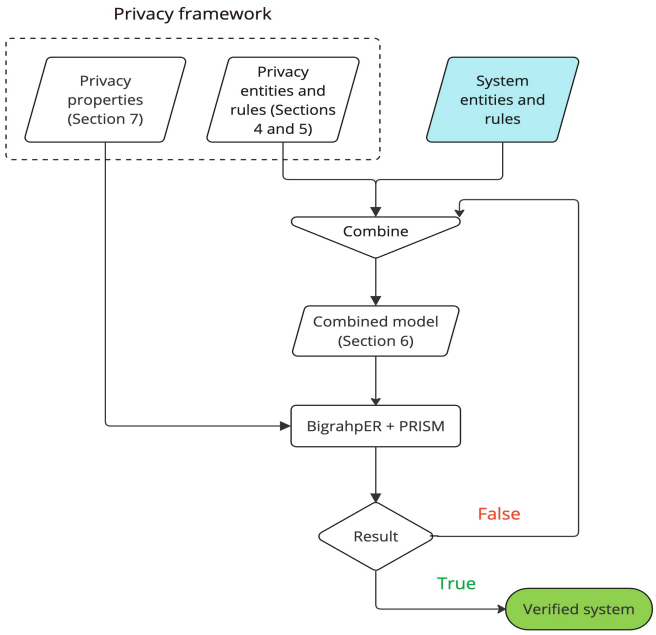


Figure 1: Overview of our bigraph-based privacy framework. The blue box represents the system entities and rules that end-users should specify.

Section 5 introduce the proposed privacy model, followed by Section 7, which discusses the formal verification of the model. Section 8 and Section 9 present the related work, and the conclusion, respectively.

2 Privacy Regulations for Cross-Border Data Transfers

Many governments have imposed legislation to regulate transferring data outside their countries or territories. They specify certain conditions to transfer data internationally as such transfers could affect their citizens' privacy or national security (especially if the transferred data is sensitive) [9].

For example, the GDPR allows international transfers if the transfer is done based on one of the following:

1. **Adequacy decisions:** this requirement ensures that the recipient country provide 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].

2. **Appropriate safeguards:** if the adequacy decision does not cover the receiver country, then it should provide appropriate safeguards. The GDPR specifies many safeguards. One of these safeguards is *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 [10]. Another safeguard is using the *certification mechanism*. The certification should meet

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

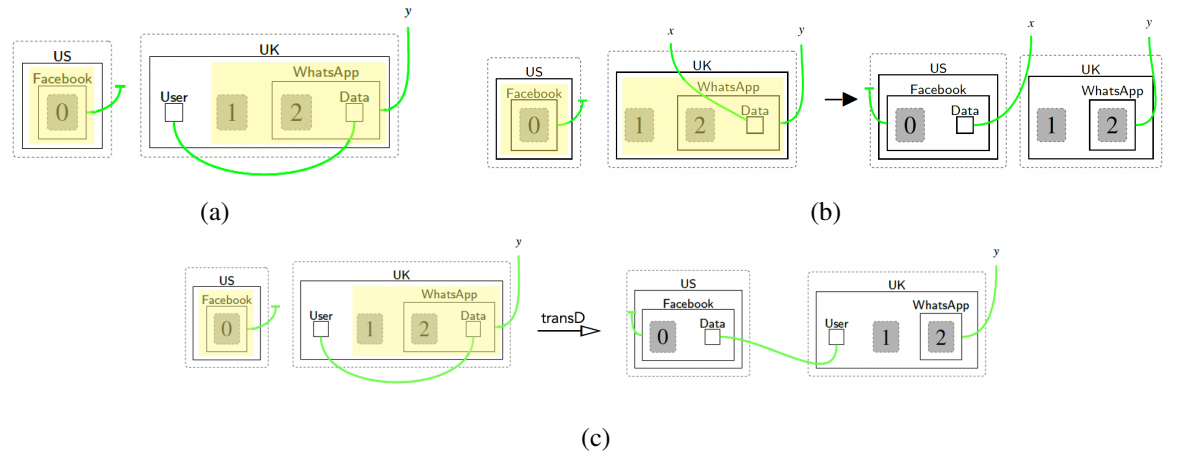


Figure 2: (a) A bigraph representing the initial state of a WhatsApp system that transfers Data to Facebook; (b) Rule `transD` transfers the Data from WhatsApp to Facebook; (c) The result of applying rule `transD` to the initial state, where Facebook acquires the Data.

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) [7].

3. **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 paper 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 [8].

3 Bigraphical Reactive Systems

An initial bigraph and a set of reaction rules specifying the systems' evolution over time define a Bigraphical Reactive Systems (BRSs). Unlike other formalisms, BRSs model systems diagrammatically and algebraically. In this paper, we only use the diagrammatic representation, but the equivalent algebraic specification is available online [2].

An example bigraph is in Fig. 2a: a WhatsApp system that transfers User's Data from the UK to Facebook in the US. Shapes represent entities, *e.g.* US, UK *etc*. The shaded rectangles are *sites* that represent components that have been abstracted away as they are irrelevant to the context, *e.g.* site 0 could be a data centre of Facebook.

The dashed rectangles are called *regions*. Each parallel region can be considered as a modelling perspective that is used to split between different concerns [5, 28], *e.g.* the US is modelled in one perspective and the UK in another.

The green *links* are used to connect entities with each other. Links can be open to indicate the possibility of being linked to other unspecified entities *e.g.* link y, or closed to express for example

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

exlusive ownership, *e.g.* the link that connects the User with the Data. Another way to close the links is making them one-to-zero hyperedge, *e.g.* the link that is attached to Facebook. We sometimes use coloured links for readability and visual clarity.

Bigraphs can evolve over time by using reaction rules. From now on, we will use rules to indicate reaction rules. Each rule contains two parts: the left-hand side (lhs) and the right-hand side (rhs). The lhs represents the pattern that will be changed, whereas the rhs represents the changed pattern. For example, rule `transD` in Fig. 2b models a transfer of the user Data to the US. By applying this rule to bigraph B shown in Fig. 2a, the part of the bigraph that matches the lhs of the rule (highlighted in yellow) is changed to match the rhs [4]. Fig. 2c shows the result of applying rule `transD` (Fig. 2b) to bigraph B (Fig. 2a). 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 lhs of the rule, *e.g.* changing link y in rule `transD` (Fig. 2b) to z does not affect the application of the rule.

Instantiation maps are a feature of BRS that allows us to copy, swap, or delete sites when the rules are applied. In this paper, we number sites to represent the instantiation maps. For instance, the sites are copied if shown in the lhs and the rhs of the rule as presented in Fig. 2b. Conversely, they are deleted if they appear in the lhs but do not exist in the rhs.

We specify and analyse our models using BigraphER [27], an open-source tool for modelling, rewriting, and visualising bigraphs. BigraphER offers two useful features: (1) enforcement of ordering over the rules via priority classes, *i.e.* each class is a set of rules and any rule in a class with lower priority can only be applied when no rules in any higher priority class cannot be applied, and (2) parameterised entities and rules, *e.g.* Data(x) and `transD`(x) where $x \in UserData = \{\texttt{Name}, \texttt{ID}, \texttt{Age}\}$, to allow sets of entities and rules (each entity or rule uses one value x).

4 Modelling Privacy Visually

We apply our approach to WhatsApp's privacy policies. WhatsApp has two *data controllers* [33], 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. For space, we present the data centre that is located in Ireland and Facebook's branches in Dubai and Mexico. A partial model of the example is shown in Fig. 3, while the full model is [2]. The model consists of three perspectives: WhatsSystem, SRType (for sender and receiver), and Locations. The specific system is modelled in the WhatsSystem perspective. It consists of the data controller in the US (WhatsLLC), the data controller in Ireland (WhatsIreland), the Facebook branch in Dubai (FacebookD), the Facebook branch in Mexico (FacebookM), Meta company in the US, and the data centre in Ireland (DC_Ir). 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 4.1 and Section 4.2.

4.1 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. 3. 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

[4] As this is an illustrative example, we transfer (or optionally duplicate) the data to the US without verifying the GDPR requirements for international data transfers.

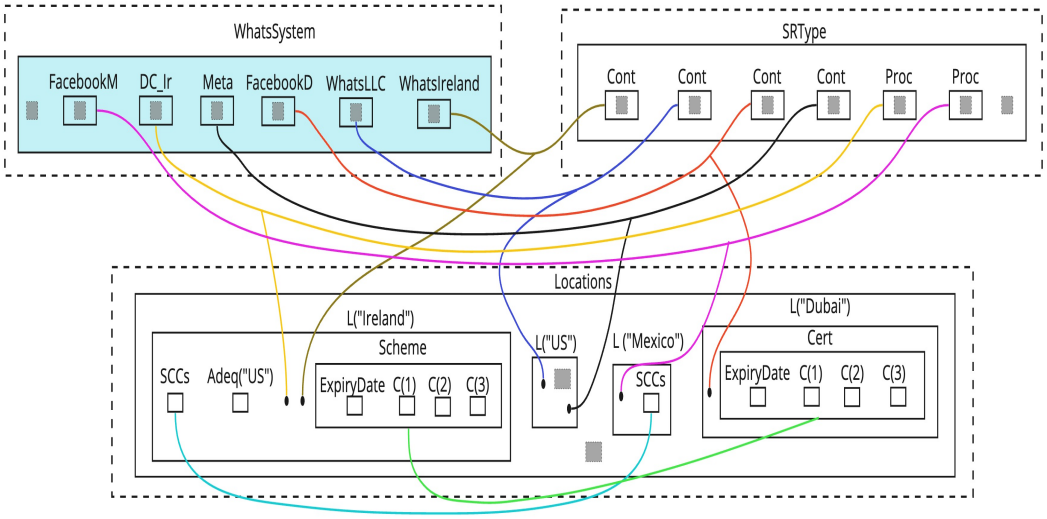


Figure 3: A partial initial state for the WhatsApp example. System-specific entities are shaded in blue. The predefined privacy model appears within the uncoloured regions.

purposes of processing users' data, whereas the processor processes the data based on the controller's instructions [17]. This perspective can easily be extended to support other types, *e.g.* sub-processor, joint controller *etc*.

In Fig. 3, the system's entities that have the controller role (WhatsIreland, WhatsLLC, FacebookD, and Meta) are linked to their own Cont. Similarly, the system's processors (DC_Ir and FacebookM) are linked to their own Proc.

4.2 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 $\mathsf{L}(x)$. This is shown in Fig. 3 where, for instance, DC_Ir is linked to a P within $\mathsf{L}(\texttt{Ireland})$. 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 P and P′, 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 5.1.

Table 1 describes the entities of this perspective. Parameterised entities $\mathsf{L}(x)$ and $\mathsf{Adeq}(x)$ allow end-users to add new countries, *e.g.* $\mathsf{Adeq}(\texttt{Japan})$. Entities SCCs and Cert are nested within $\mathsf{L}(x)$ to indicate that the safeguard provided by the organisation in country x is the standard contractual clauses or the certification, respectively. Consider the example in Fig. 3. Entity $\mathsf{L}(\texttt{Mexico})$ contains SCCs as the safeguard provided by Facebook in Mexico is the SCCs. Cert is nested within $\mathsf{L}(\texttt{Dubai})$ 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. Finally, entity $\mathsf{L}(\texttt{Ireland})$ contains $\mathsf{Adeq}(\texttt{US})$, SCCs, Scheme and three criteria, *i.e.* $\mathsf{C}(x)$ with $1 \leq x \leq 3$.

Table 1: Locations perspective entities.

Entity	Description
$\mathsf{L}(x)$	Locations as parameterised entities where x is a country's name, *e.g.* `UK`.
$\mathsf{Adeq}(x)$	Parameterised entity where x is the name of an adequate country.
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.
$\mathsf{C}(x)$	Criteria that should be checked to determine the valid certification. Parameter x is a criterion's identifier, *e.g.* 1, 2, `Transparency`, `Right-to-access`.
Cert	The certification provided by the receiver.
P, P′	Pointers for each sender/receiver within a location. Shown as solid black and blue bullets, respectively.

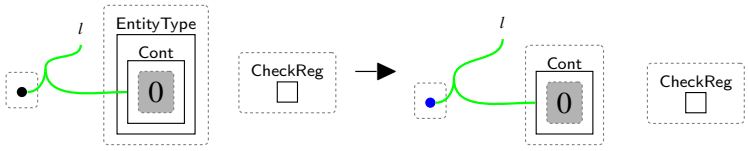


Figure 4: `checkingReg`: checking the sender/receiver region by tagging the linked pointers.

5 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 the reaction rules encoding the processes required to check various privacy requirements. Unlike system-specific rules (see Section 6), they can be utilised across systems without changes.

5.1 Checking Regions

To check if data transfers are restricted, we need to check the region of the sender and the receiver. If they are in the same region, then the transfer is 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 recipient **type**. As these pointers are nested within the entity $\mathsf{L}(\mathtt{x})$, tagging them enables us to specify if the sender and receiver are in the same or different regions as explained in Section 5.2.

Rule `checkingReg` (Fig. 4) checks the region of the data controller (Cont), either a sender or receiver[5], by tagging the linked pointer. This is a set of rules, so we need to define one for each entity type. For example, to specify the regions of the processor (Proc) or a sub-processor (SubProc), we use rule `checkingReg` (Fig. 4), but we replace the Cont with the Proc or the SubProc, respectively (see Fig. 21a and Fig. 21b in Appendix A). This allows us to model several situations of data transfers: from controller to controller, from processor to processor, from controller to processor *etc*.

EntityType and CheckReg are generated via systems rules as explained in Section 6. EntityType is

[5]Rules that specify senders and receivers are specific systems.

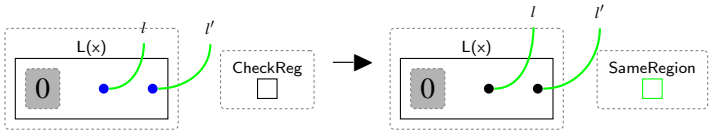


Figure 5: `sameReg(x)`: the sender and the receiver in the same region. The parameter `x` is the name of the country.

generated to specify the type of the entity, *e.g.* Cont, SubProc *etc*. The entity CheckReg is generated by a system rule to trigger the privacy model.

Consider the controller WhatsIreland needs to transfer data to the processor DC_Ir. We specify the sender and the receiver type by generating EntityType around Cont and Proc via system rule(s). We also need to generate the entity CheckReg to start the checking process. As we specified the type of the data exporter (Cont) and the data importer (Proc), we use rule `checkingReg` (Fig. 4) to tag the pointer linked to the Cont. We also use the same rule to tag the pointer of the Proc, but we replace the entity Cont with Proc. We discard EntityType after applying the rule as the pointers are tagged.

5.2 Entities are in the Same Region

If the tagged pointers are nested within the same entity L(`x`), then the sender and the receiver are in the same region. For instance, the tagged pointers in Section 5.1 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(x)` (Fig. 5). 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. The parameter `x` should be replaced with the country's name when we specify the priority classes in BigraphER, *e.g.* `Ireland`, `Dubai` *etc*.

5.3 Checking Adequacy

Suppose the data sender is WhatsIreland in `Ireland`, and the receiver is Meta in the `US`. By checking their regions as explained in Section 5.1, 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(x)` in Fig. 6 checks the adequacy requirement. To use this rule, we need to replace the parameter `x` with the **receiver** country's name, *e.g.* the `US`. If the rule is applied (the match of the left-hand side is found), then the country is adequate. We match on the tagged pointer as we need to specify the country we aim to check its adequacy. The entity Adequate is generated to show the result of checking the adequacy requirement. The pointer is untagged to reuse 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 5.4 and Section 5.5.

5.4 Checking SCCs

As mentioned previously, we must check the safeguards when rule `checkingAdeq(x)` (Fig. 6) is not applied. In case the provided safeguard is SCCs, we use rule `tagInvalidSCCs` in Fig. 7 to check its

Figure 6: `checkingAdeq(x)`: checking the adequacy of the data importer country where the parameter `x` is its name.

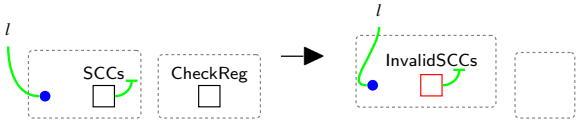


Figure 7: `tagInvalidSCCs`: tagging the invalid SSCs.

validity. For example, the safeguard that FacebookM in `Mexico` provides is SCCs and it is linked to the entity SCCs that is nested within L(`Ireland`) as shown in Fig. 3. The aim of linking them is to show that the SCCs that FacebookM provides are *active and accepted* by the controller in `Ireland`. If the link is closed, rule `tagInvalidSCCs` (Fig. 7) is applied to indicate that the SCCs are invalid (InvalidSCCs) and we discard CheckReg to terminate the checking process. Conversely, if the two entities (SCCs) are linked, the system can safely transfer the data. We do not model the content of the SCCs since they are part of the contract between the sender and receiver, so our focus is only on verifying their acceptance by the receiver. Additionally, the GDPR does not require their evaluation as they are *pre-approved* by the European Commission [12].

5.5 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. 8 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. 8) 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.* checking that C(`1`) is nested within the Scheme and the Cert, by tagging the existing criterion (the green token). 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`) $\cdots$ C(`x`).

If **one** of the defined family rules is not applied, the Cert does not meet the criteria. For example, if rule `tagCriteria(1)` is not applied but rule `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. 9) 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.

5.6 Result of Checking Certification

Rule `checkingCertResult` (Fig. 10) 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** [13]. This requires checking

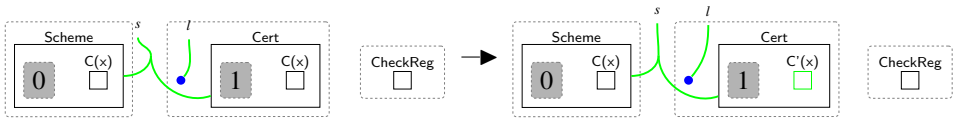


Figure 8: `tagCriteria(x)`: tagging each criterion that must be satisfied.

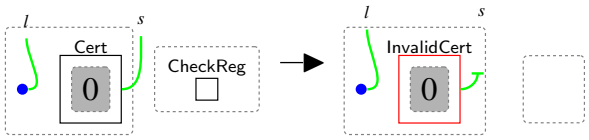


Figure 9: `tagInvalidCert`: tagging the certification as invalid if at least one criterion is not satisfied.

its validation date as well [7]. Rule `checkingCertResult` (Fig. 10) 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 5.7

5.7 Checking the Expiry Date

As shown in Fig. 11, 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 data [6].

Rule `ExpiredDate` (Fig. 11) 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.

5.8 Withdrawing Certification

Based on the GDPR, the provider of the compliant certification (CompliantCert) has the right to withdraw it [7]. We model the case of withdrawing the CompliantCert through three steps: initialising the withdrawal process, asking for withdrawal and processing the withdrawal request.

To initialise the withdrawal process, we need to enable the provider to withdraw the CompliantCert. Rule `enabelWithd` in Fig. 12a models this step by replacing the entities CheckExp and CurrentDate

[6]To 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 [2].

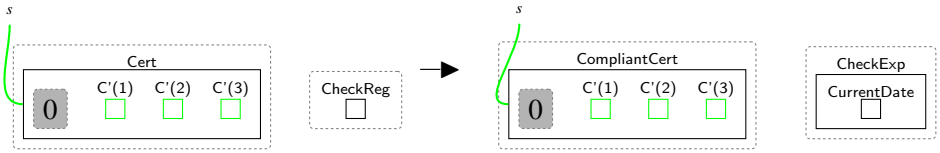


Figure 10: `checkingCertResult`: result of checking the certification and initialising the process of checking its expiry date.

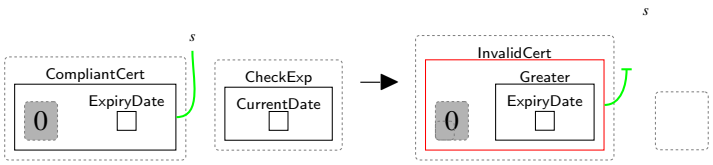
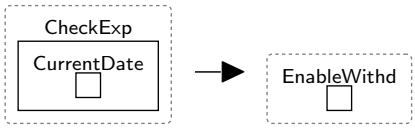
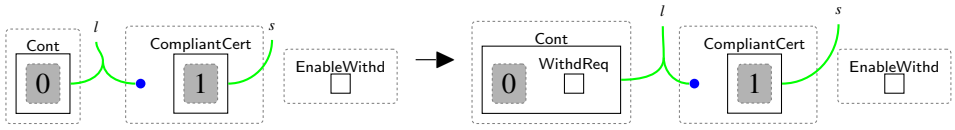


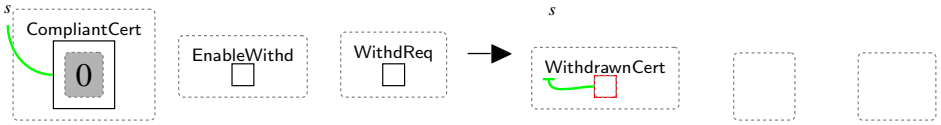
Figure 11: `ExpiredDate`: tagging the expired certification.



(a) `enabelWithd`: enabling the provider of the certification to ask for withdrawing.



(b) `askingForWithd`: the data controller asks for withdrawing the certification.



(c) `processWithdReq`: the certification is withdrawn.

Figure 12: The process of withdrawing the certification.

with EnableWithd. Importantly, this rule is only applied if the certification has been checked and has yet to expire.

The providers of the CompliantCert, *i.e.* Cont, Proc *etc*., are now able to ask to withdraw their CompliantCert. Rule `askingForWithd` (Fig. 12b) models the withdrawal request (WithdReq) of the Cont. If the provider is a Proc or a SubProc, we use the same rule (`askingForWithd` in Fig. 12b), but we should replace the Cont with the Proc or the SubProc, respectively (see Fig. 22 in Appendix A). The rule explicitly matches on the CompliantCert as we need to ensure that the entity who asked for the withdrawal uses the certification not the SCCs. The rule also matches on the provider's location (the tagged pointer) to specify the entity's location.

Rule `processWithdReq` (Fig. 12c) models the last step. It closes link *s* connecting the CompliantCert to the Scheme and tagging the CompliantCert as WithdrawnCert. We can safely discard EnableWithd and WithdReq as the request has been processed.

6 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_Ir) needs to transfer users' data to WhatsApp's data centre in Singapore (DC_Si) and to the Facebook branch in China (FacebookC). Fig. 13 shows a partial initial state for this example. We consider DC_Si a sub-processor (linked to

SubProc) that uses SCCs. FacebookC is considered as a data controller (linked to Cont), and the safeguard that it provides is a certification (Cert).

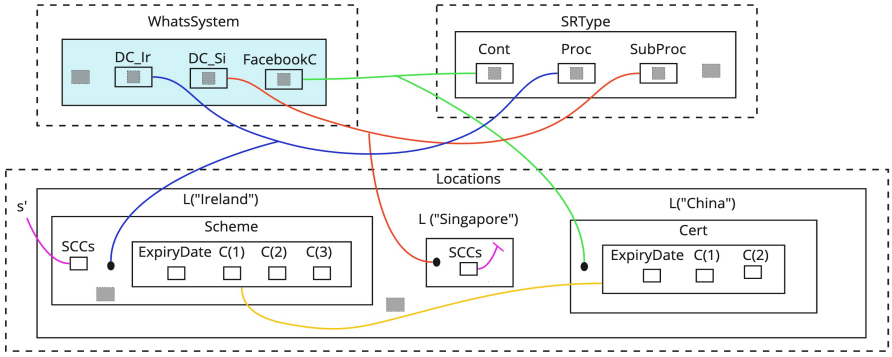


Figure 13: A Partial initial state for the example of transferring data to the data centre in Singapore and to Facebook in China.

We need to trigger the privacy model to check the sender and receiver regions and their safeguards. To do so, we should generate two entities: EntityType and CheckReg via system rules. Rule `dcIrType` (Fig. 14) specifies the DC_Ir type. As DC_Ir is linked to Proc, EntityType is generated around Proc to indicate that the sender is a processor. The entity StartTransfer is generated by a system rule (shown in Fig. 25 Appendix B). It is nested within DC_Ir as DC_Ir is the entity initiating the transfer. StartTransfer is replaced with SpecifyReceivers to specify the receivers types before transferring the data.

Rules `dcSingType` (Fig. 15) and `FacebookCType` (Fig. 16) specify the type of DC_Si and FacebookC, respectively. As DC_Si is linked to SubProc, the EntityType is generated around SubProc. The entity AskForData represents the receiver's request for the data. FacebookC is a data controller, so EntityType is generated around the token Cont. We also replace SpecifyReceivers with CheckReg.

After generating EntityType and CheckReg, we can start checking the regions by tagging the pointers linked to the specified entity types. We specify the region of the sender DC_Ir by using rule `checkingReg` (Fig. 4). However, we change the entity Cont to Proc (see Fig. 21a in Appendix A) because the specified type of DC_Ir is Proc as shown in Fig. 14. We also use the same rule to check the region of DC_Si, but we replace Cont with SubProc (Fig. 21b in Appendix A). The specified type of FacebookC is Cont, so we use rule `checkingReg` (Fig. 4) as it is. By applying these rules, the pointers that will be tagged are the pointers that are located in L(`Ireland`), L(`Singapore`) and L(`China`) as shown in Fig. 17. We can observe that the tagged pointers are located in different countries, so the transfer is restricted.

As shown in Fig. 13, Singapore and China are not specified as adequate countries (like the US in Fig. 3). DC_Si provides the SCCs, but the entity SCCs has a closed link as presented in Fig. 13. Rule `tagInvalidSCCs` (Fig. 7) is applied to tag SCCs as InvalidSCCs. The entity InvalidSCCs is used as a

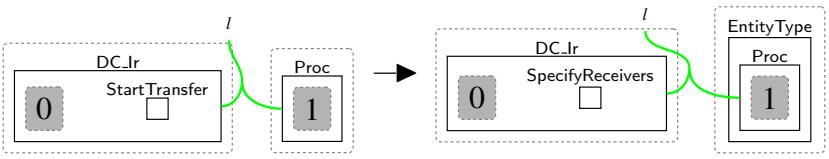


Figure 14: `dcIrType`: specifying DC_Ir type.

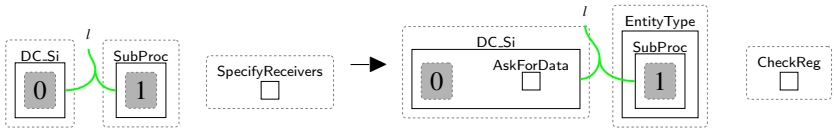


Figure 15: `dcSingType`: specifying DC_Si type.

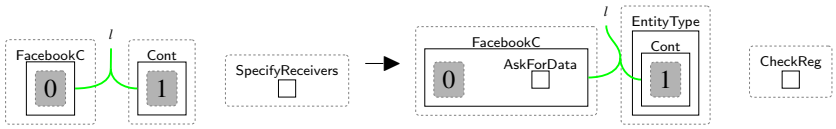


Figure 16: `FacebookCType`: specifying FacebookC type.

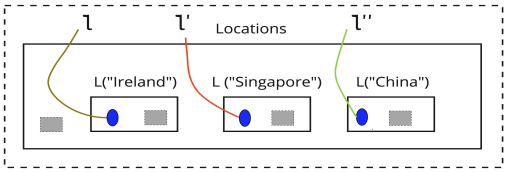


Figure 17: The tagged pointers after specifying DC_Ir, DC_Si and FacebookC locations.

flag to prevent the transfer and block DC_Si through the system rule `preventSing` (Fig. 18). Similarly, FacebookC uses the certification mechanism that has only two criteria (C(1) and C(2)) as shown in Fig. 13. By applying rule `tagCriteria(x)` (Fig. 8), only two criteria are tagged. In this case, rule `InvalidCert` (Fig. 9) tags the Cert as InvalidCert. To block FacebookC, we define a system rule that matches explicitly on InvalidCert as shown in Fig. 19. Because we prevented the transfer to DC_Si and FacebookC, we can safely discard the entities InvalidSCCs and InvalidCert.

7 Verification

The aim of defining the privacy model is to generate a transition system used to conduct the *verification* step, *i.e.* proving the GDPR requirements for international data transfer within the system. BigraphER utilises the initial state, *e.g.* Fig. 3, and the reaction rules (privacy rules and system rules) to *automatically* generate the transition system. The produced transition system consists of states (bigraphs) that describe the system configurations. Each state is a result of applying a rule (transition). Manually checking privacy properties, *i.e.* the GDPR requirements, within the transition system is challenging as the number of states can be considerable. For example, in the WhatsApp example, the number of the generated states is 160, while the number of the transitions is 207.

Model checkers are tools used to *automatically* prove properties within the transition system. We use PRISM [21] model checker as BigraphER supports it. To use PRISM, the properties that we aim to check should be encoded using a logic language. We use a non-probabilistic temporal logic language as the requirements we aim to check are specified based on *regulations*. In particular, we use Computation Tree Logic (CTL) [11] to encode the GDPR requirements for international data transfer because CTL quantifies the properties over all paths. This allows us to prove that the GDPR requirements hold across all possible execution paths. We also define a set of predicates using BigraphER to label the states.

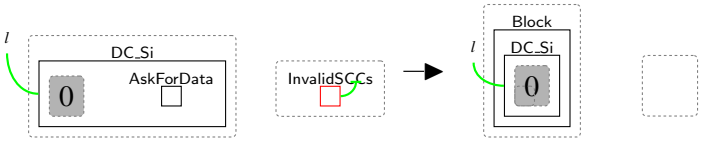


Figure 18: `preventSing`: preventing the transfer to the DC_Si.

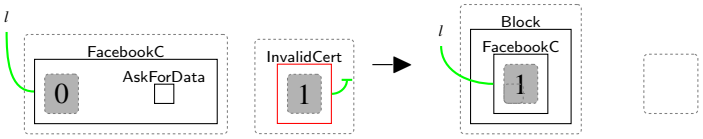


Figure 19: `preventChina`: preventing the transfer to the FacebookC.

These predicates represent the right-hand sides of the reaction rules, and PRISM utilises them to parse the transition system. For instance, we define predicate `invalidSCCs` to label the states that have the match of the right-hand side of rule `tagInvalidSCCs` (Fig. 7).

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:

$$\mathbf{A}\,[\,\mathbf{G}\,(\,\neg\texttt{adequateCountry} \implies \neg\texttt{dataTransfer})\,] \tag{1}$$

`adequateCountry` is a predicate that labels all states that contain the match of the right-hand side of rule `checkingAdeq(x)` (Fig. 6). 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 SCCs is invalid*:

$$\mathbf{A}\,[\,\mathbf{G}\,(\texttt{invalidSCCs} \implies \neg\texttt{dataTransfer})\,] \tag{2}$$

As previously mentioned, `invalidSCCs` is a predicate that holds if the SCCs are invalid, so in this case we check: for all paths of the transition system, if `invalidSCCs` holds in a state, `dataTransfer` should not hold in the all states throughout the path.

The third requirement is: *no transfer should be performed if the certification is invalid*:

$$\mathbf{A}\,[\,\mathbf{G}\,(\texttt{invalidCert} \implies \neg\texttt{dataTransfer})\,] \tag{3}$$

This property proves that: it is always the case that if the certification is invalid, *e.g.* expired or does not meet the criteria, the transfer is never performed.

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

$$\mathbf{A}\,[\,\mathbf{G}\,(\texttt{withdrawCert} \implies \neg\,\texttt{dataTransfer})\,] \tag{4}$$

Here, we prove that for all paths, if `withdrawCert` is true, `dataTransfer` should be false in all sates on the path.

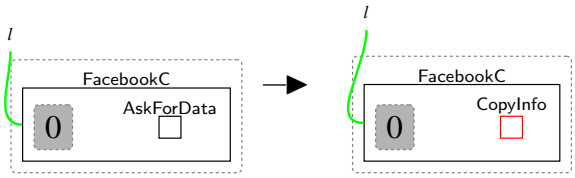


Figure 20: `privacyViolation`: system rule that leads to a privacy violation by transferring the data to FacebookC.

Privacy Violations Detection: defining systems rules could lead to privacy violations. As shown in Fig. 19, the data must not be transferred to FacebookC as the provided certification is invalid. However, the data is transferred in Fig. 20. The reason for such a violation is that modellers define the rule without explicitly matching on InvalidCert. This violation is detected by property (3) as is not met, *i.e.* the certification is invalid, but the transfer is performed. To correct the rule, we have to redefine it as presented in Fig. 19.

8 Related Work

There is a lack of using formal methods to prove systems' compliance with the GDPR requirements for international data transfer. However, multiple works have been proposed to prove the systems' compliance with other GDPR notions. For example, Karami *et al.* [19] define data protection language (DPL) as an object-oriented language that handles the GDPR notions of purpose and storage limitation, the right to be forgotten, and providing/withdrawing consent. They use multiset rewrite rules to formally model the operational semantics of DPL. The resulting model is checked using the model checker Maude [22], but the checking process was *partially automated* as they also provide a pen-and-paper proof. Another tool that checks these notions is the Model-Based approach to Identify Privacy Violations in software requirements (MBIPV) [35]. It is a *fully automatic* tool that detects GDPR violations by converting UML class diagrams into a Kripke model. The NuSMV [1] model checker is used to detect the violations in the defined Kripke model.

Kammueller [18] uses theorem prover Isabelle (HOL) to prove the compliance of IoT healthcare systems with the GDPR requirements for the users' right to access their data, their right to delete their data and the compliance of systems' functions with these rights. However, the proving process is not entirely automated.

Milner's π-calculus [23] is extended [20] by defining a set of privacy calculus syntax. Later, the privacy calculus is improved to model the GDPR notion of providing and withdrawing consent [31]. Another work based on π-calculus involves using multiparty session types to develop D´alogoP tool that proves the GDPR notion of processing purposes within systems [30].

All the mentioned approaches use formal techniques to evaluate the systems' compliance with specific aspects of the GDPR. However, they are limited in verifying the GDPR requirements for international data transfers because they need to support *spatial properties* as bigraphs. [24].

Recent *informal* approaches are proposed to prove the GDPR requirements for international transfer. Guamán *et al.* [15] define a method that systematically analyses the compliance of Android mobile apps with the GDPR notion of international data transfer. Later, the method is converted to an automated tool [16] but does not handle the certification mechanisms as our model does. Pascual *et al.* [25] provide *Hunter*, an automated tool for tracking anycast communications protocol that routes the data to the

nearest server regardless of location. The proposed approach mainly relies on the adequacy decision but does not cover other requirements for international data transfer, *e.g.* the use of safeguards. Unlike our approach, which *formally* proves the GDPR requirements for cross-border data transfers, these methods use *informal* techniques, potentially resulting in a less rigorous analysis of these requirements.

9 Conclusion

We provide a bigraphical framework that captures the GDPR requirements for cross-border data transfer. We encode these requirements using CTL and prove them by PRISM.

Although we use only one example to define the framework, we believe it can capture different systems. The multi-perspectives approach enables us to model the GDPR requirements for international data transfers independently from the specific system. This supports the framework's ability to model several systems. Using parameters also allows various developers to model different countries and criteria.

Using our framework requires collaboration between software engineers with a background in bigraphs theory and privacy experts. The engineers build the model and explain the data flow to the privacy experts. The privacy experts advise on the certification criteria and the validity of the SCC, or they even amend the model if the regulations are updated. Organisations can use our framework to prove their adherence to the international transfer requirements.

In future, we aim to extend the framework to capture other safeguards, *e.g.* Binding Corporate Rules (BCRs) and Codes of conduct. We also aim to model the case of transferring data based on the user's consent. The model need to be applied to more examples and real-world systems to show its generality. An automated tool is also needed to check the correctness of defining initial states using sorting schemes (assigning types and constraints for entities and links) [3].

References

[1] (2023): *NuSMV: a new symbolic model checker*. Available at `https://nusmv.fbk.eu/index.html`.

[2] Ebtihal Althubiti & Michele Sevegnani (2024): *Modelling Privacy Compliance in Cross-border Data Transfers with Bigraphs*, doi:10.5281/zenodo.11215226. Available at `https://doi.org/10.5281/zenodo.11215226`.

[3] Blair Archibald & Michele Sevegnani (2024): *A Bigraphs Paper of Sorts*. In: *Graph Transformation - 17th International Conference, ICGT 2024, Held as Part of STAF 2014, Enschede, Netherlands, Proceedings*. To Appear.

[4] Office of the Australian Information Commissioner (OAIC) (2019): *Sending personal information overseas*. Available at `https://www.t.ly/oIcI7`.

[5] Steve Benford, Muffy Calder, Tom Rodden & Michele Sevegnani (2016): *On Lions, Impala, and Bigraphs: Modelling Interactions in Physical/Virtual Spaces*. *ACM Trans. Comput.-Hum. Interact.* 23(2), pp. 9:1–9:56, doi:10.1145/2882784. Available at `http://doi.acm.org/10.1145/2882784`.

[6] European Data Protection Board (2023): *1.2 billion euro fine for Facebook as a result of EDPB binding decision*. Available at `https://www.t.ly/HRqTR`.

[7] European Data Protection Board (2023): *Guidelines 07/2022 on certification as a tool for transfers*. Available at `https://www.t.ly/OPhXd`.

[8] European Data Protection Board (n.d.): *International data transfers*. Available at `https://www.t.ly/HDXO0`.

[9] Francesca Casalini & Javier López González (2019): *Trade and cross-border data flows*. *OECD Trade Policy Papers*.

[10] Francesca Casalini, Javier López González & Taku Nemoto (2021): *Mapping commonalities in regulatory approaches to cross-border data transfers*. *OECD Trade Policy Papers*.

[11] Edmund M Clarke & E Allen Emerson (1981): *Design and synthesis of synchronization skeletons using branching time temporal logic*. In: *Workshop on logic of programs*, Springer, pp. 52–71.

[12] European Commission (n.d.): *Standard Contractual Clauses (SCC)*. Available at `https://www.t.ly/7HnF6`.

[13] Intersoft Consulting (n.d.): *Certification*. Available at `https://gdpr-info.eu/art-42-gdpr/`.

[14] Intersoft Consulting (n.d.): *Transfers of personal data to third countries or international organisations*. Available at `https://gdpr-info.eu/chapter-5/`.

[15] Danny S Guamán, Jose M Del Alamo & Julio C Caiza (2021): *GDPR compliance assessment for cross-border personal data transfers in android apps*. *IEEE Access* 9, pp. 15961–15982.

[16] Danny S Guamán, David Rodriguez, Jose M del Alamo & Jose Such (2023): *Automated GDPR compliance assessment for cross-border personal data transfers in android applications*. *Computers & Security* 130, p. 103262.

[17] Information Commissioner's Office (ICO) (2023): *How do you determine whether you are a controller or processor?* Available at `t.ly/MPvgu`.

[18] Florian Kammueller (2018): *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, pp. 3319–3324.

[19] Farzane Karami, David Basin & Einar Broch Johnsen (2022): *DPL: A Language for GDPR Enforcement*. In: *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*, IEEE, pp. 112–129.

[20] Dimitrios Kouzapas & Anna Philippou (2017): *Privacy by typing in the π-calculus*. *Log. Methods Comput. Sci.* 13(4), doi:10.23638/LMCS-13(4:27)2017. Available at `https://doi.org/10.23638/LMCS-13(4:27)2017`.

[21] M. Kwiatkowska, G. Norman & D. Parker (2011): *PRISM 4.0: Verification of Probabilistic Real-time Systems*. In G. Gopalakrishnan & S. Qadeer, editors: *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, *LNCS* 6806, Springer, pp. 585–591.

[22] José Meseguer (2012): *Twenty years of rewriting logic*. *J. Log. Algebr. Program.* 81(7-8), pp. 721–781, doi:10.1016/j.jlap.2012.06.003. Available at `https://doi.org/10.1016/j.jlap.2012.06.003`.

[23] Robin Milner (1999): *Communicating and mobile systems: the pi calculus*. Cambridge university press.

[24] Robin Milner (2009): *The space and motion of communicating agents*. Cambridge University Press.

[25] Hugo Pascual, Jose M Del Alamo, David Rodriguez & Juan C Dueñas (2024): *Hunter: Tracing anycast communications to uncover cross-border personal data transfers*. *Computers & Security* 141, p. 103823.

[26] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, Phillipa Gill et al. (2018): *Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem*. In: *The 25th Annual Network and Distributed System Security Symposium (NDSS 2018)*.

[27] Michele Sevegnani & Muffy Calder (2016): *BigraphER: rewriting and analysis engine for bigraphs*. In: *International Conference on Computer Aided Verification*, Springer, pp. 494–501.

[28] Michele Sevegnani, Milan Kabác, Muffy Calder & Julie A. McCann (2018): *Modelling and Verification of Large-Scale Sensor Network Infrastructures*. In: *23rd International Conference on Engineering of Complex Computer Systems, ICECCS 2018, Melbourne, Australia, December 12-14, 2018*, pp. 71–81, doi:10.1109/ICECCS2018.2018.00016. Available at `https://doi.org/10.1109/ICECCS2018.2018.00016`.

[29] European Data Protection Supervisor (n.d.): *International transfers*. Available at `https://www.t.ly/GPuii`.

[30] Evangelia Vanezi, Georgia M Kapitsaki, Dimitrios Kouzapas, Anna Philippou & George A Papadopoulos (2020): *DiálogoP-A Language and a Graphical Tool for Formally Defining GDPR Purposes*. In: *International Conference on Research Challenges in Information Science*, Springer, pp. 569–575.

[31] Evangelia Vanezi, Dimitrios Kouzapas, Georgia M Kapitsaki & Anna Philippou (2019): *Towards GDPR Compliant Software Design: A Formal Framework for Analyzing System Models*. In: *International Conference on Evaluation of Novel Approaches to Software Engineering*, Springer, pp. 135–162.

[32] WhatsApp (2024): *WhatsApp Privacy Policy*. Available at `https://www.t.ly/uGjmN`.

[33] WhatsApp (2024): *Who is providing your WhatsApp services*. Available at `https://www.t.ly/NqTfZ`.

[34] Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui & John Fitzgerald (2009): *Formal methods: Practice and experience*. *ACM computing surveys (CSUR)* 41(4), pp. 1–36.

[35] Tong Ye, Yi Zhuang & Gongzhe Qiao (2023): *MBIPV: a model-based approach for identifying privacy violations from software requirements*. *Software and Systems Modeling* 22(4), pp. 1251–1280.

A Additional Privacy Rules

We present the family rules of rule `checkingReg` and `askingForWithd` shown in Fig. 4 and Fig. 12b, respectively:

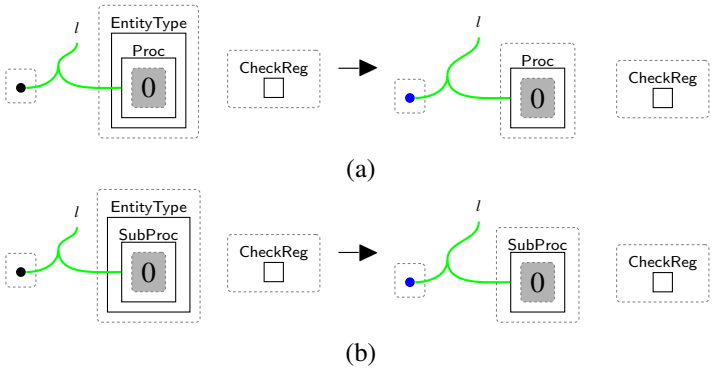


Figure 21: (a) `checkProcReg`: checking the region of the processor (whether it is a sender or receiver); (b) `checkSubProcReg`: checking the region of the sub-processor (whether it is a sender or receiver).

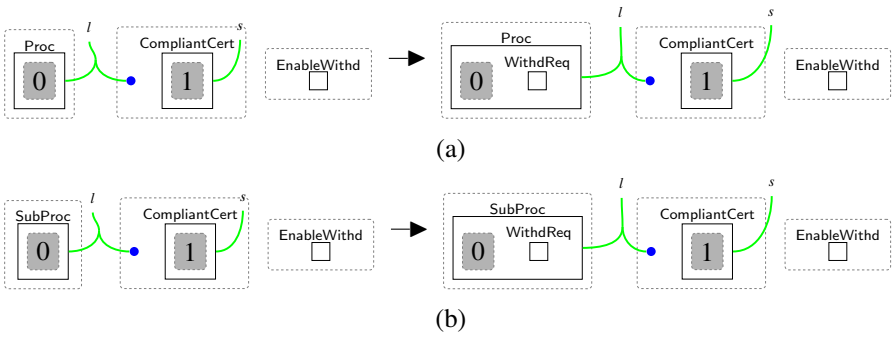


Figure 22: (a) `procAsksWithd`: the processor asks to withdraw the certification; (b) `subProcAsksWithd`: the sub-processor asks to withdraw the certification.

B System Rules

We present some specific system rules for the WhatsApp example:

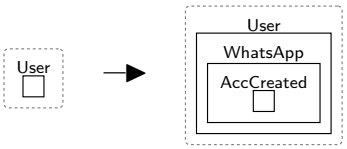


Figure 23: `creatingAccount`: the system's user starts using WhatsApp to create an account.

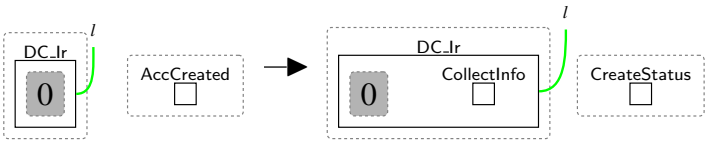


Figure 24: `creatingStatus`: the user creates a status, so the entity AccCreated is replaced with CreateStatus. We assume that the processor DC_Ir collects information about the user's status, *e.g.* status content type, location information, information about the device used to post the status *etc*. 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.

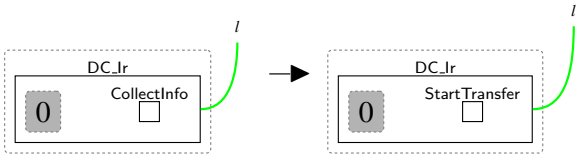


Figure 25: `initialisingTrnas`: DC_Ir initialises the transferring process, so we replace the entity CollectInfo with SpecifyReceivers to start applying rule `dcIrType` in Fig. 14