Domain-Specific Languages and Normative Reasoning

L. Thomas van Binsbergen

Complex Cyber Infrastructure, University of Amsterdam
ltvanbinsbergen@acm.org

September 23, 2021
SAE Digital Data Steering Group

UNIVERSITY OF AMSTERDAM

Overview

1. Regulated systems
   - Relating normative and computational concepts
   - DSLs and model-driven engineering
2. The eFLINT language
   - eFLINT 1.0
   - eFLINT 2.0
3. Reflections

Section 1

Regulated systems

Regulated data exchange:

data exchange systems governed by regulations, agreements and policies

as an instance of

Regulated systems:

distributed software systems with embedded regulatory services derived from norm specifications that monitor and/or enforce compliance

NWO-funded: SSPDDP – Secure and scalable, policy-driven data exchange

NWO-funded: DL4LD – Data Logistics for Logistics Data

EFRO-funded: AMDEX Fieldlab – neutral data-exchange infrastructure

Towards regulated systems

Monolithic programs

Towards regulated systems

Monolithic programs $\xrightarrow{\textit{distribution}}$ Service-oriented architectures

Towards regulated systems

$$\begin{tikzcd}
\text{Monolithic programs} \arrow[r, "\textit{distribution}"] \arrow[d, "\textit{AI}"] & \text{Service-oriented architectures} \\
\text{Autonomous systems} &
\end{tikzcd}$$

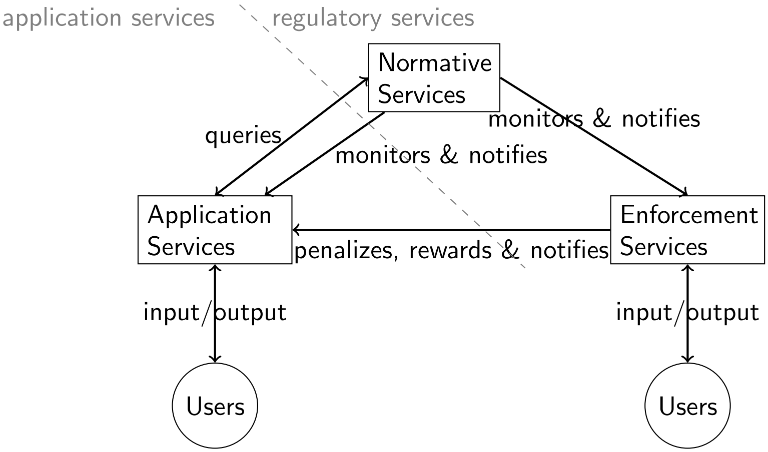
Towards regulated systems

$$
\begin{tikzcd}
\text{Monolithic programs} \arrow[r, "\textit{distribution}"] \arrow[d, "\textit{AI}"] & \text{Service-oriented architectures} \arrow[d, "\textit{AI}"] \\
\text{Autonomous systems} \arrow[r, "\textit{distribution}"] & \text{Social software systems}
\end{tikzcd}
$$

Towards regulated systems

$$
\begin{tikzcd}
\text{Monolithic programs} \arrow[r, "\textit{distribution}"] \arrow[d, "\textit{AI}"] & \text{Service-oriented architectures} \arrow[d, "\textit{AI}"] & \\
\text{Autonomous systems} \arrow[r, "\textit{distribution}"] & \text{Social software systems} \arrow[dr, "\textit{norms} + \textit{enforcement}"] & \\
 & & \text{Regulated (software) systems}
\end{tikzcd}
$$

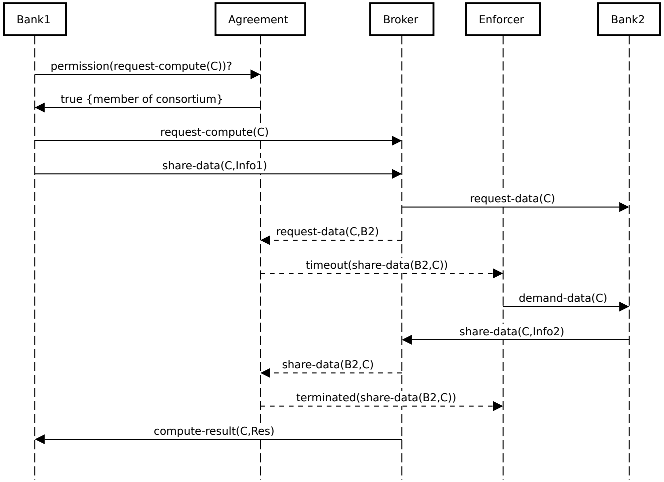
Regulated system = application services + regulatory services



Dynamic enforcement examples – sharing agreement

(Article 1) A member of the consortium has the right to request a risk assessment computation from the broker for any (potential) client

(Article 2) The data broker has the power to oblige members of the consortium to share information about any client the member does business with



Regulated systems – points to address

Formalization of applicable norms: reusable, modular and dynamically updateable

Different methods of embedding and enforcing norms:

- Static ex-ante: verify and apply norms during software production
  e.g. correct-by-construction arguments, model checking
- Dynamic ex-ante: apply rules at run-time, guaranteeing compliance
  enables decisions (behavioral, normative) that depend on input
- Embedded ex-post enforcement: specified responses to violations
  enables (regulated) non-compliant behavior, e.g. based on risk assessment by agent
- External ex-post enforcement: external responses to violations
  e.g. auditing, conformance checking
  enables (human-)intervention in running system

Production of diagnostic reports and/or audit trails to enable evaluation and reflection

Regulated *systems* – points to address

Derivation of regulatory services from formalization of norms

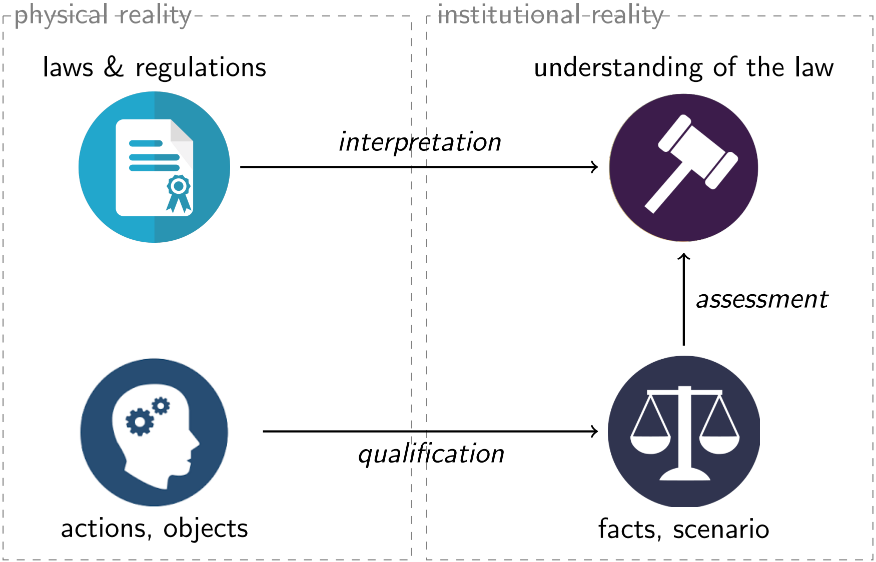
Interfacing between application and regulatory services:

- Monitoring (communicated and silent) behavior of services
  difficulties: fallible and subject to manipulation
- Regulatory services responding to queries about normative positions
  e.g. do I have permission to...? or the obligation to... ?
- Application services verifying facts on behalf of regulatory services
  e.g. verifying credentials or certificates
- Regulatory services communicating changes in normative positions
  e.g. gaining/losing powers, holding/satisfying obligations, violations

Challenges: different interpretations of norms and different qualifications of situations

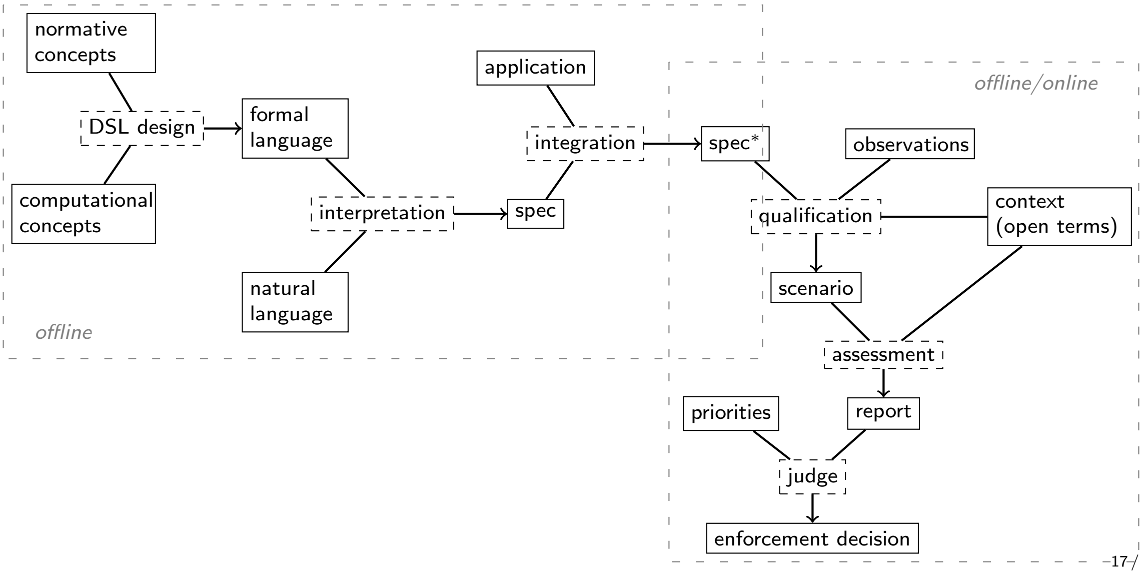
Subsection 1

Relating normative and computational concepts



"If the facts are against you, argue the law. If the law is against you, argue the facts. If the law and the facts are against you, pound the table ..." -Carl Sandburg

Normative reasoning – information flow



Foundational, normative & computational concepts

computational

state

$\mathit{parent}(A, B) = \mathit{true}$
$\ldots$

Foundational, normative & computational concepts

computational

state

$parent(A, B) = true$
...

transitions

$parent(A, B) = true$
...

$parent(A, B) = false$
...

Foundational, normative & computational concepts

computational

state

$$\boxed{\begin{array}{l} parent(A, B) = true \\ \ldots \end{array}}$$

transitions

$$\boxed{\begin{array}{l} parent(A, B) = true \\ \ldots \end{array}}$$

$$\boxed{\begin{array}{l} parent(A, B) = false \\ \ldots \end{array}}$$

- Violations of state and transition

Foundational, normative & computational concepts

deontic

computational

state

$$\boxed{\begin{array}{l} \mathit{parent}(A, B) = \mathit{true} \\ \ldots \end{array}}$$

prohibitions

transitions

permissions

$$\boxed{\begin{array}{l} \mathit{parent}(A, B) = \mathit{true} \\ \ldots \end{array}}$$

$$\boxed{\begin{array}{l} \mathit{parent}(A, B) = \mathit{false} \\ \ldots \end{array}}$$

obligations

- Violations of state and transition

Foundational, normative & computational concepts

deontic

computational

prohibitions

permissions

obligations

of

of

state

$parent(A, B) = true$
...

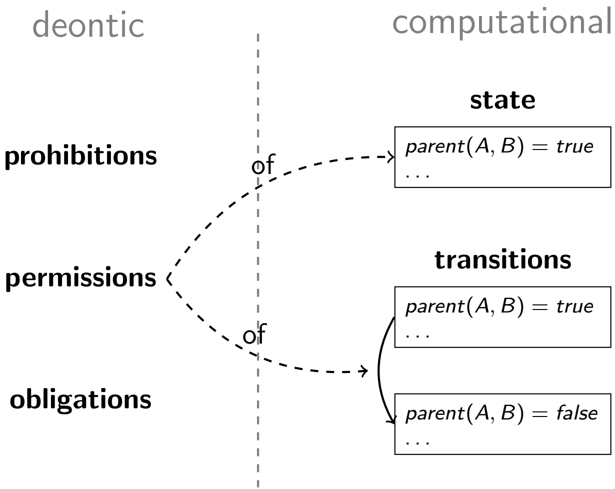
transitions

$parent(A, B) = true$
...

$parent(A, B) = false$
...

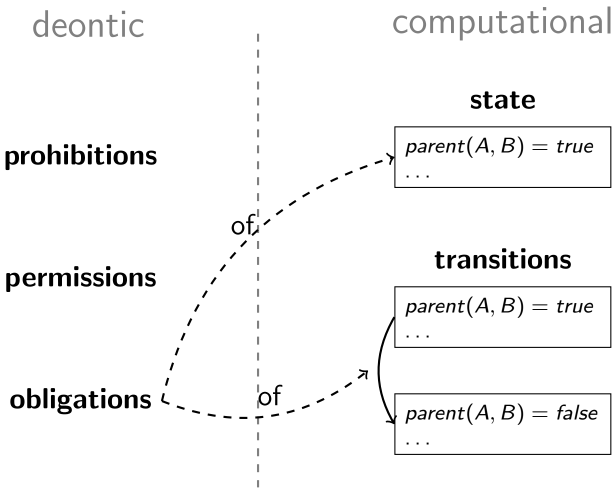
- Violations of state and transition

Foundational, normative & computational concepts



- Violations of state and transition

Foundational, normative & computational concepts



- Violations of state and transition

Foundational, normative & computational concepts

deontic	computational	potestative
	state	
prohibitions	$parent(A, B) = true$ $\ldots$	**powers**
	transitions	
permissions	$parent(A, B) = true$ $\ldots$	
obligations	$parent(A, B) = false$ $\ldots$	

of

of

- Violations of state and transition

Foundational, normative & computational concepts

deontic	computational	potestative
	state	
prohibitions	$parent(A, B) = true$ $\ldots$	**powers**
	transitions	
permissions	$parent(A, B) = true$ $\ldots$	
obligations	$parent(A, B) = false$ $\ldots$	

of

of

- Powers have (normative) consequences

- Violations of state and transition

Foundational, normative & computational concepts

deontic

prohibitions

permissions

obligations

of

of

computational

state

$\boxed{\begin{array}{l} \mathit{parent}(A,B) = \mathit{true} \\ \ldots \end{array}}$

transitions

$\boxed{\begin{array}{l} \mathit{parent}(A,B) = \mathit{true} \\ \ldots \end{array}}$

$\boxed{\begin{array}{l} \mathit{parent}(A,B) = \mathit{false} \\ \ldots \end{array}}$

- Violations of state and transition

potestative

powers

- Powers have (normative) consequences
- Deontic and potestative terms are first-class
- Powers modify truth-assignments to variables

Foundational, normative & computational concepts

deontic

prohibitions

permissions

obligations

of

of

computational

state

$parent(A, B) = true$
...

transitions

$parent(A, B) = true$
...

$parent(A, B) = false$
...

- Violations of state and transition

potestative

powers

- Powers have (normative) consequences
  - Deontic and potestative terms are first-class
  - Powers modify truth-assignments to variables
- Implicit vs explicit permissions

Normative relations between actors

- A deontic term is associated with several actors:
  - The **holder** of the prohibition, obligation or prohibition
  - Zero or more **claimants** to the prohibition or obligation
  - The actor who **assigned** the prohibition, obligation or permission
- A potestative term is associated with several actors
  - The **performing** actor
  - One or more **recipients** being affected by the power
  - The actor who **assigned** the power

Subsection 2

DSLs and model-driven engineering

Domain-specific languages

Domain-specific languages empower domain-experts to producing programs, specifications, or models without having to rely on software engineers

DSLs have constructs and syntax (sometimes visual) relevant to their domain

Figure: MySQL

Figure: PlantUML

Figure: DOT

Model-driven engineering

Generate implementations from *models* of the desired system:

- Specify the essence, abstracting away from implementation details
- Visualisation, inspection, and checking of model in isolation
- Applied by low-code/no-code platforms

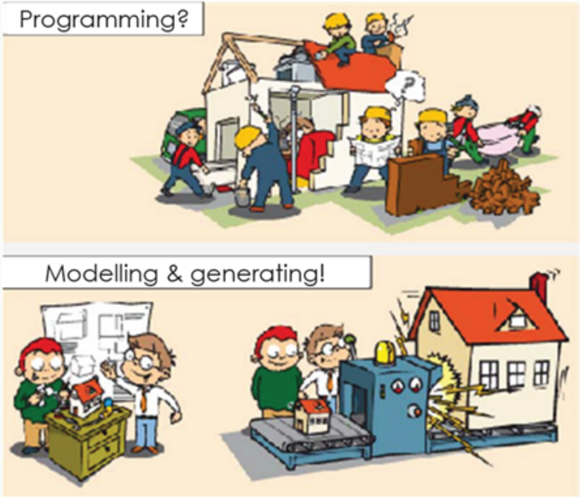
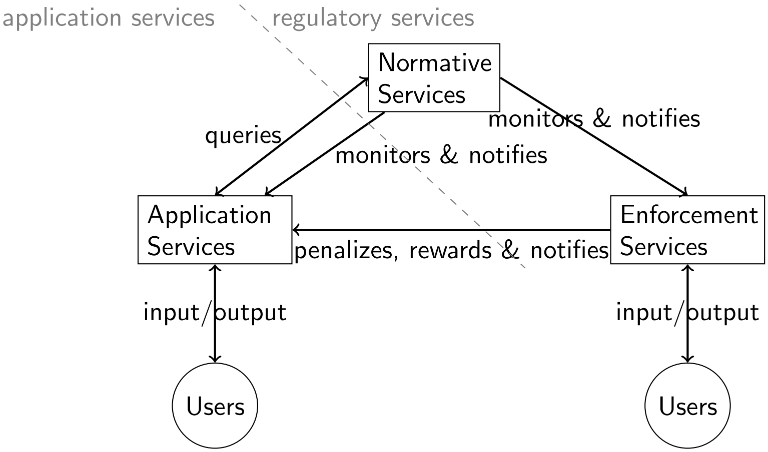


Figure: by Johan den Haan, CTO at Mendix

Model-driven experimentation with regulated system



Our languages for model-driven experimentation

eFLINT – formalization of norms from a variety of sources

declarative reasoning about facts, actions and duties

reactive component for integration in software systems

including actor-based implementation

AgentScriptCC – specification of services as agents

reactive BDI agents,

compiled to actor-based implementation,

used for both application and enforcement services

eFLINT: A Domain-Specific Language for Executable Norm Specifications

L. Thomas van Binsbergen
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
ltvanbinsbergen@acm.org

Lu-Chi Liu
University of Amsterdam
Amsterdam, The Netherlands
l.liu@uva.nl

Robert van Doesburg
Leibniz Institute, University of Amsterdam / TNO
Amsterdam, The Netherlands
robertvandoesburg@uva.nl

Tom van Engers
Leibniz Institute, University of Amsterdam / TNO
Amsterdam, The Netherlands
vanengers@uva.nl

published @ SPLASH 2020

Run, Agent, Run

Architecture and Benchmarking of Actor-based Agents

Mostafa Mohajeri Parizi
m.mohajeriparizi@uva.nl
Informatics Institute, University of Amsterdam
Amsterdam, the Netherlands

Giovanni Sileno
g.sileno@uva.nl
Informatics Institute, University of Amsterdam
Amsterdam, the Netherlands

Tom van Engers
vanengers@uva.nl
Informatics Institute, University of Amsterdam
Amsterdam, the Netherlands

Sander Klous
s.klous@uva.nl
Informatics Institute, University of Amsterdam
Amsterdam, the Netherlands

published @ SPLASH 2020

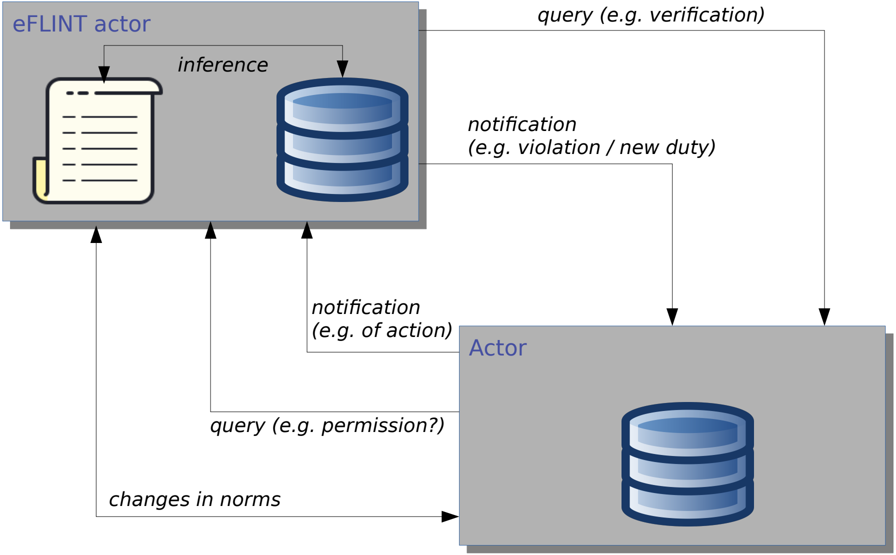
Actor-oriented programming in the Akka framework:

`https://akka.io/`

actor systems modelling social software systems

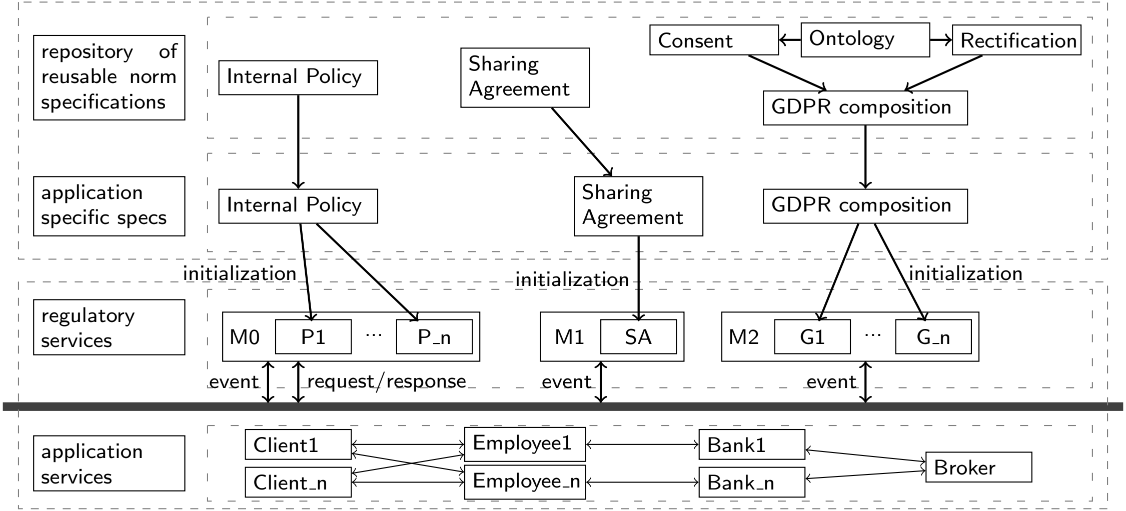
akka

eFLINT actors



Regulated systems for Know Your Customer case study

policy construction (offline)



distributed system (online)

Section 2

The eFLINT language

Subsection 1

eFLINT 1.0

Example – knowledge representation

(Toy Article 1) a natural person is a legal parent of another natural person if:

- *they are a natural parent, or*
- *they are an adoptive parent*

```
Fact person Identified by String
Placeholder parent    For person
Placeholder child     For person

Fact natural-parent   Identified by parent * child
Fact adoptive-parent  Identified by parent * child

Fact legal-parent     Identified by parent * child
  Holds when adoptive-parent(parent,child)
          || natural-parent(parent,child)
```

Example – powers and duties

(Toy Article 2) a child has the power to ask a legal parent for help with their homework, resulting in a duty for the parent to help.

```
Act ask-for-help
  Actor       child
  Recipient   parent
  Creates     help-with-homework(parent,child)
  Holds when  legal-parent(parent,child)

Duty help-with-homework
  Holder         parent
  Claimant       child
  Violated when  homework-due(child)

Fact homework-due Identified by child

Act help
  Actor       parent
  Recipient   child
  Terminates  help-with-homework(parent,child)
  Holds when  help-with-homework(parent,child)
```

Example – scenario

```
Fact person Identified by Alice, Bob, Chloe, David
```

Listing 1: Domain specification

```
+natural-parent(Alice, Bob).
+adoptive-parent(Chloe, David).
```

Listing 2: Initial state

```
ask-for-help(Bob, Alice).                     // permitted: Alice is Bob's legal
    parent
+homework-due(Bob).                           // homework deadline passed
?Violated(help-with-homework(Alice,Bob)).     // query confirms duty is violated
help(Alice,Bob).                              // duty terminated
```

Listing 3: Scenario

eFLINT online!

frames

```
Fact person Identified by String
Placeholder parent    For person
Placeholder child     For person

Fact natural-parent   Identified by parent * child
Fact adoptive-parent  Identified by parent * child
Fact legal-parent     Identified by parent * child
  Holds when adoptive-parent(parent,child)
          || natural-parent(parent,child)

Act ask-for-help
  Actor      child
  Recipient  parent
  Creates    help-with-homework(parent,child)
  Holds when legal-parent(parent,child)

Fact homework-due Identified by child

Duty help-with-homework
  Holder        parent
  Claimant      child
  Violated when homework-due(child)

Act help
  Actor      parent
  Recipient  child
  Terminates help-with-homework(parent,child)
  Holds when help-with-homework(parent,child)
```

domains

```
Fact person Identified by Alice, Bob, Chloe, David
```

initial state

```
natural-parent(Alice, Bob).
adoptive-parent(Chloe, David).
```

Examples

Knowledge representation: Vehicles | Departments | Count Votes | Cast Votes

GPCE2020 paper examples: Help with homework | GDPR

Various: Buyer/Seller (v1) | Buyer/Seller (v2) | Buyer/Seller (v3) | Permit Applications | Permit Applications (v2) | Multiple taxpayers | Voting

Load file: Browse... No file selected.

scenario

```
ask-for-help(Bob, Alice).
+homework-due(Bob).  // homework deadline passed
?Violated(help-with-homework(Alice,Bob)).
help(Alice,Bob).
```

Run Reset Save model name

response

```
* Duty violated at step 2
    ("Alice":person,"Bob":person):help-with-homework
```

output

Step 0: initial state

Step 1: ("Bob":person,"Alice":person):ask-for-help

+("Alice":person,"Bob":person):help-with-homework

Step 2: ("Bob":person):homework-due

+("Bob":person):homework-due

Step 3: query

Step 4: ("Alice":person,"Bob":person):help

-("Alice":person,"Bob":person):help-with-homework

Subsection 2

eFLINT 2.0

REPLization of eFLINT

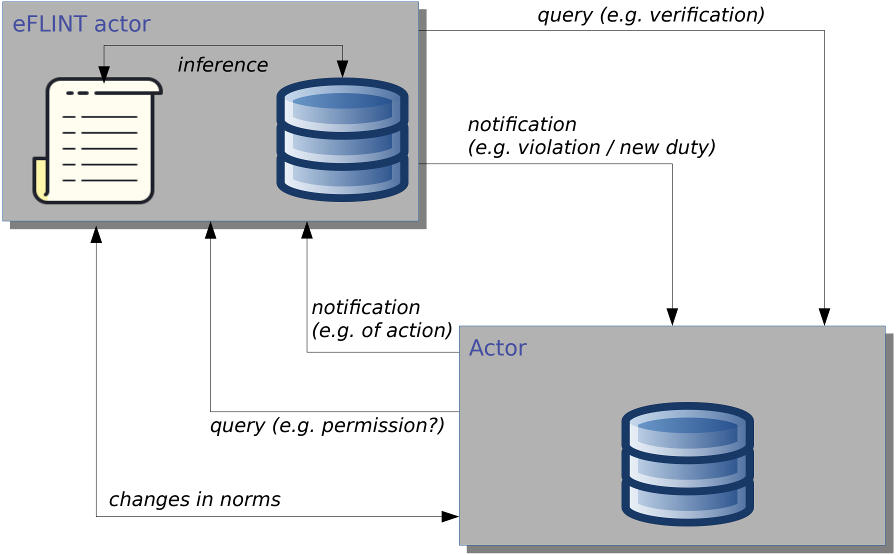
1. eFLINT 2.0: REPLization applied to eFLINT 1.0
    - the different kinds of declarations and statements can be mixed freely
    - dynamic scenario construction and assessment; dynamic policy construction
    - enables implementation of ‘eFLINT actors’

REPLization of eFLINT

1. eFLINT 2.0: REPLization applied to eFLINT 1.0
   - the different kinds of declarations and statements can be mixed freely
   - dynamic scenario construction and assessment; dynamic policy construction
   - enables implementation of 'eFLINT actors'

2. Extensions to the eFLINT syntax
   - The `Extends` keyword to modularly extend existing declarations
     Enables rule-based formalisation of articles
   - The `Syncs with` keyword to trigger multiple actions simultaneously
     Enables the qualification of one action as an instance of another

eFLINT actors

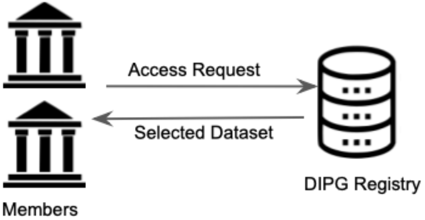


The DIPG case – Compliance questions

According to the GDPR and the DIPG regulatory document:

1. What conditions need to be fulfilled by a member before making data available?

2. What conditions need to be fulfilled when accessing data from the registry?



Modular GDPR specification

Dynamic generation of access control policies from social policies

L. Thomas van Binsbergen[1,a], Milen G. Kebede[a], Joshua Baugh[b], Tom van Engers[a], Dannis G. van Vuurden[b]

[a]*Informatics Institute, University of Amsterdam, 1090GH Amsterdam, The Netherlands*
[b]*Princess Maxima Center for Pediatric Oncology, Department of Neuro-oncology, Utrecht, The Netherlands*

Figure: ICTH2021

```
Act collect-personal-data
  Actor controller
  Recipient subject
  Related to data, processor, purpose
    Where subject-of(subject, data)
  Creates processes(processor, data, controller, purpose)
```

Article 5 – processing conditions

Article 5

Principles relating to processing of personal data

1. Personal data shall be:

(a) processed lawfully, fairly and in a transparent manner in relation to the data subject ('lawfulness, fairness and transparency');

(b) collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1), not be considered to be incompatible with the initial purposes ('purpose limitation');

(c) adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed ('data minimisation');

(d) accurate and, where necessary, kept up to date; every reasonable step must be taken to ensure that personal data that are inaccurate, having regard to the purposes for which they are processed, are erased or rectified without delay ('accuracy');

```
Fact minimal-for-purpose Identified by processes
Extend Act collect-personal-data Conditioned by minimal-for-purpose(data, purpose)
```

Listing 4: Member (1c)

```
Fact accurate-for-purpose Identified by data * purpose
Extend Act collect-personal-data Conditioned by accurate-for-purpose(data, purpose)
```

Listing 5: Member (1d)

Article 6 – legal processing

Article 6

Lawfulness of processing

1. Processing shall be lawful only if and to the extent that at least one of the following applies:

(a) the data subject has given consent to the processing of his or her personal data for one or more specific purposes;

(b) processing is necessary for the performance of a contract to which the data subject is party or in order to take steps at the request of the data subject prior to entering into a contract;

(c) processing is necessary for compliance with a legal obligation to which the controller is subject;

```
Fact consent Identified by subject * controller * purpose * data
Extend Act collect-personal-data
  Holds when consent(subject, controller, purpose, data)
```

Listing 6: Member (1a)

```
Fact has-legal-obligation Identified by processes
Extend Act collect-personal-data
  Holds when has-legal-obligation(controller, purpose)
```

Listing 7: Member (1c)

Compliance Question 1

DIPG Regulatory document – Article 4(2):

Members should transfer data to the DIPG registry in a coded form only

```
Fact coded Identified by dataset

Act make-data-available
  Actor institution
  Recipient dcog
  Related to dataset
  Conditioned by coded(dataset)
  Holds when member(institution)
```

Compliance Question 1

```
Extend Act make-data-available Syncs with (Foreach donor:
  collect-personal-data(controller = institution
                       ,subject    = donor
                       ,data       = dataset
                       ,processor  = "DCOG"
                       ,purpose    = "DIPGResearch")
    When subject-of(donor, dataset))
```

An institution can make a dataset available when (for each donor (subject) in the dataset):

- The institution is a member (DIPG Regulatory Document – Article 4(2))
- Data is coded (DIPG Regulatory Document – Article 4(2))
- Consent is given by the donor for the processing of their personal data by the DCOG for the purpose of DIPGResearch (GDPR – Article 6)
- Data should be accurate for the purpose DIPGResearch (GDPR – Article 5)

Section 3

Reflections

Bounded vs open-ended domains

Static analyses

- eFLINT 1.0 enabled automated scenarios assessment in finite domain
- Future work: applying model checking, and/or property-based testing

Dynamic enforcement

- eFLINT 2.0 enabled dynamic interpretation, qualification and assessment
- Domain and scenario established at runtime, based on the contents of the knowledge base

Design decision:

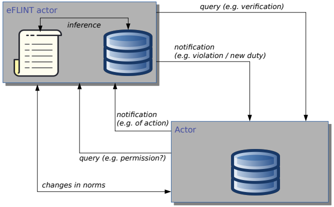
When enumerating instances, first check domain of type, then knowledge base

```
// opt1: Fact person Identified by Alice, Bob, Chloe, David
// opt2: +person(Alice). +person(Bob). +person(Chloe). +person(David).

?(Forall person: !homework-due(person))
```

Two approaches to enforcing social policies

Embedding eFLINT specifications as eFLINT actors, akin to ‘policy decision point’:



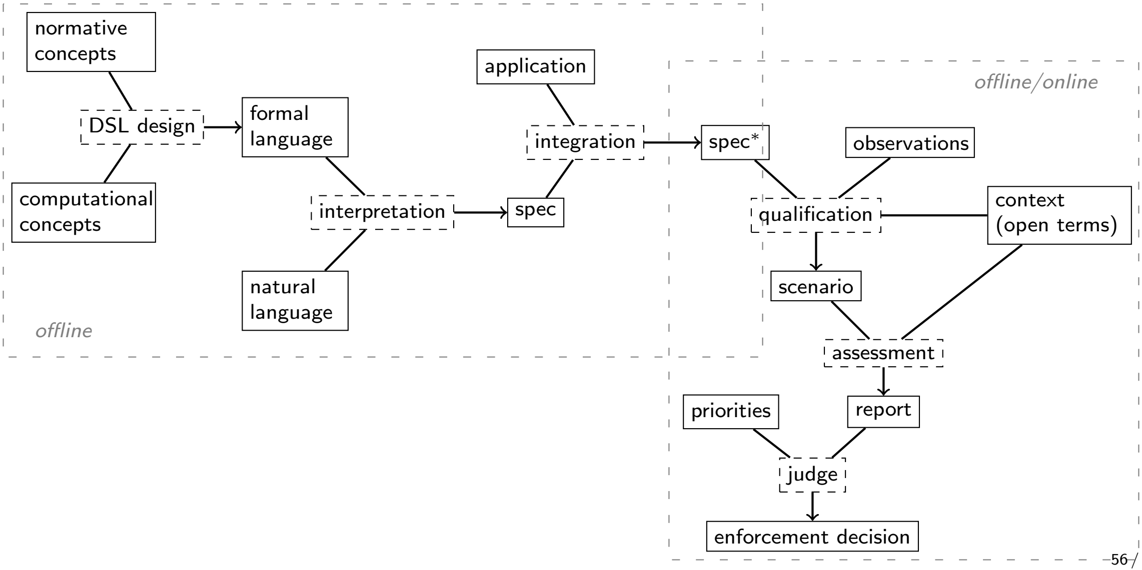
Generating system-level policies, akin to ‘policy administration point’

Dynamic generation of access control policies from social policies

L. Thomas van Binsbergen[1,a], Milen G. Kebede[a], Joshua Baugh[b], Tom van Engers[a], Dannis G. van Vuurden[b]

[a]*Informatics Institute, University of Amsterdam, 1090GH Amsterdam, The Netherlands*
[b]*Princess Maxima Center for Pediatric Oncology, Department of Neuro-oncology, Utrecht, The Netherlands*

Potential benefits from standardisation



Standardisation efforts

Amsterdam Data Exchange (AMdEX)

- Project running till June 2023, initiative > 2023
- Standardisation of: specification (of interpretations) and assessment reports

Legal Engineering and TNO and UvA

- All aspects of legal engineering are in scope
- Standardisation of: specifications, scenarios (cases), reports, decisions, ...

Takeaway messages

At the Complex Cyber Infrastructure group, we are experimenting with approaches to enforcing laws, regulations, agreements and contracts in (distributed) systems

Takeaway messages

At the Complex Cyber Infrastructure group, we are experimenting with approaches to enforcing laws, regulations, agreements and contracts in (distributed) systems

The eFLINT DSL serves as a tool to demonstrate and experiment with various aspects of our approach, with a focus on runtime enforcement using 'regulatory services'

Takeaway messages

At the Complex Cyber Infrastructure group, we are experimenting with approaches to enforcing laws, regulations, agreements and contracts in (distributed) systems

The eFLINT DSL serves as a tool to demonstrate and experiment with various aspects of our approach, with a focus on runtime enforcement using 'regulatory services'

We are currently working on a prototype to demonstrate our approach in data exchange systems such as the Amsterdam Data Exchange (AMdEX)

Takeaway messages

At the Complex Cyber Infrastructure group, we are experimenting with approaches to enforcing laws, regulations, agreements and contracts in (distributed) systems

The eFLINT DSL serves as a tool to demonstrate and experiment with various aspects of our approach, with a focus on runtime enforcement using 'regulatory services'

We are currently working on a prototype to demonstrate our approach in data exchange systems such as the Amsterdam Data Exchange (AMdEX)

Regulated (data exchange) systems involve several information processing steps that can benefit from standardisation

Domain-Specific Languages and Normative Reasoning

L. Thomas van Binsbergen

Complex Cyber Infrastructure, University of Amsterdam
ltvanbinsbergen@acm.org

September 23, 2021
SAE Digital Data Steering Group

University of Amsterdam