# eFLINT
## An action-based language for reasoning about norms

L. Thomas van Binsbergen[1] and Tom van Engers[2]

[1] Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands `ltvanbinsbergen@acm.org`
[2] Leibniz Institute, University of Amsterdam / TNO, Amsterdam, The Netherlands `vanengers@uva.nl`

**Abstract.** The CALCULEMUS-FLINT project is a collaboration between several Dutch organizations, including the University of Amsterdam, TNO, ICTU, the Ministry of Justice and Security and the Ministry of Finance, and aims to reduce the efforts made by governmental organizations towards implementing the rules and services described in laws and policies. The CALCULEMUS method uses the FLINT language to capture the norms embedded in laws and policies as formal specification and involves tools for reasoning about cases and scenarios in a way that makes algorithmic decisions explainable to affected individuals and the general public. This presentation discusses an executable variant of FLINT, called eFLINT, and its application within projects of the University of Amsterdam. For example, the prototype implementation of eFLINT serve as references for the reasoners developed as part of the CALCULEMUS-FLINT project. Furthermore, the formalization of eFLINT is the first step in ongoing research projects aimed at enabling normative reasoning into application domains such as multi-agent systems, policy-aware data sharing and software engineering.

**Keywords:** normative reasoning, policy specification, executable specification, testing, verification

## 1 Norm representation

The CALCULEMUS-FLINT project, and in particular the FLINT language [1], is based on principles derived from the legal framework constructed by Hohfeld for analyzing courthouse activities [2]. Central to the framework is the observation that the 'normative position' of an individual, such as the individual being deemed to having a 'duty' or 'liability', is always with respect to another individual; if person $X$ has the duty to do $A$, then there is a person $Y$ having a claim to $A$ being done (reaping potential benefits of $A$). We refer to such a correlative connection between two parties as a 'normative relation'. The normative positions are divided in two groups, the *deontic* positions – duty, claim, privilege and no-claim – and the *potestative* positions – power, liability, disability and immunity. Hohfeld defines the positions through a scheme of opposites and correlatives. For example, power and disability are opposites, whereas duty and claim are correlatives (in the sense exemplified before). A more detailed discussion on Hohfeld's definitions is provided by Thompson [3].

An important aspect of the FLINT language is that the normative positions of actors are derived from the actions they can (powers) or are expected to (duties) perform. The benefit of this approach is that relating a scenario or implementation to a normative specification written in FLINT is more straightforward, compared to normative specifications written directly in terms of normative positions, because FLINT specifications, scenarios and implementations are inherently action-based. Potential applications of FLINT include the automated analysis of policies, the automatic construction of software implementations compliant with policies [4] and the analysis of (legal) cases [1, 5].

**Norms in eFLINT** An eFLINT specification consists of type definitions representing abstract concepts such as 'person', 'car' and 'license plate'. The values of these types are instances of the concepts, e.g. the person named "Alice", the license plate with identification number "12-XYZ-3" and the car with the aforementioned license plate. The state of the world at a particular moment in time is modeled by an assignment of truth-values to instances, determining which instances represent facts in that state. For example, the term $car(Alice,$ `"12-XYZ-3"`$)$ could represent that Alice owns a car with license plate "12-XYZ-3".

A specification declares certain types as representing *actors*, *acts* or *duties* and uses these types to express norms. An action – an instance of an act-type – contains at least the actor that can perform the action and the actor liable to the outcome of the action, among other values that uniquely identify the action. The precondition associated with an action determines whether it is enabled in a certain state, in which case the actors of the action are considered to be in a Hohfeldian power-liability relation. The post-condition associated with an action determines the effects of the action, changing the truth-assignment of the values in the current state. The actions of a specification give rise to a transition graph, with states labeling nodes and actions labeling edges such that for every edge from a state $\sigma_1$ to a $\sigma_2$, labeled with action $a$, it holds that $a$ is enabled in $\sigma_1$ and executing $a$ in $\sigma_1$ gives $\sigma_2$. Algorithms for normative reasoning work by querying the transition graph.

A duty – an instance of a duty-type – indicates that one of the so-called terminating actions for the duty is to be performed before a state is reached in which the violation condition associated with the duty holds. A duty contains at least the actor holding the duty and the actor making a claim to the duty. If a duty holds in a certain state, then the actors of the duty are in a Hohfeldian duty-claim relation. When a duty is violated, the duty holder is susceptible to penalties, delivered through the effects of so-called enforcement actions.

## 2 Reasoning about norms

The University of Amsterdam (UvA) and Centrum Wiskunde & Informatica (CWI) are collaborating with TNO and ICTU to develop software supporting the CALCULEMUS methodology. The eFLINT language has been developed to guide these efforts, providing reference material in the form of a formalization of the operational and static semantics of the language and a prototype implementation based directly on the formalization. The abstract syntax is defined in the style of Pierce [6] and the semantics as transition relations in the style of Kahn's natural semantics [7], based on Plotkin's Structural Operational Semantics [8].

Reasoning about norms is made possible by eFLINT in a variety of ways, typically through language extensions that describe alternative ways of querying the transition graph embedded in a specification. Two forms of reasoning are currently supported by the prototype implementation: simulation and testing. Firstly, a specification is refined to ensure that all types are finite. This is achieved by listing for all types the values that are considered relevant to the scenarios or cases the user wants to reason about. Secondly, an initial state is described by giving a list of facts, i.e. the observations, acts or duties deemed to hold. The simulator starts an interactive session in which the user can iteratively choose an action modifying the current state, starting at the initial state, effectively exploring paths in the transition graph.

A specification is tested by describing and executing scenarios. A scenario is a sequences of actions and events. An event modifies states in the same way an action does, but an event does not need to have actors associated with it. Events can be used to model changes to the world caused by non-compliant behavior or by external processes such as natural disasters or the passing of time. A test is performed by executing a scenario and reporting any violations, such as an action being performed while its precondition does not hold or a duty that is not terminated before its violation-condition is satisfied.

**Future work** The reasoning algorithms described above are useful for gaining confidence in the correctness of a specification and can also be used to analyze individual cases and resolve disputes. Additional reasoning algorithms are being developed to extend the capabilities of eFLINT as well as to apply eFLINT in other domains. For example, the scenario language can be extended with placeholder variables to allow users to describe 'scenario patterns'. A matching algorithm looks for instances of the pattern in the transition graph, returning zero or more substitutions for the placeholders such that applying these substitutions to the pattern yields concrete scenarios without violations. Scenario patterns are expected to be useful in multi-agent systems, allowing agents to form plans based on the compliant behavioral alternatives. Another extension being considered is to include LTL or CTL formulas in the scenario language and to enable model checking via a translation into an existing formalism for model checking.

In software engineering, successful software design processes are characterized by the specification of the software system at multiple layers of abstractions and the incremental introduction of the details necessary to transform a high-level specification to an implementation. To make it possible to use eFLINT as part of such a process, it is necessary to express norms at several layers of abstraction. Moreover, several regulations and policies are likely to govern the software system being implemented. To enable reasoning over a multitude of normative specifications, we are investigating rule formats for, effectively, translating behavior (paths in transition graphs) between specifications.

### Demonstration

This proposal for an oral presentation is accompanied by a proposal for a demonstration titled "Trust in the digital government" about the application of the CALCULEMUS method in governmental organizations.

### Acknowledgments

This work has been supported by the NWO project Secure Scalable Policy-enforced Distributed Data Processing (628.009.014) which is part of the NWO research program Big Data: Real Time ICT for Logistics

## References

1. Van Doesburg, R., Van der Storm, T., Van Engers, T.M. 2016: CALCULEMUS: Towards a Formal Language for the Interpretation of Normative Systems. In: AI4J Workshop at ECAI 2016, pp. 73–77.
2. Hohfeld, W.N. 1913: Some Fundamental Legal Conceptions as Applied in Judicial Reasoning. In: Yale Law Journal, vol. 23(1), pp. 59–64.
3. Thompson, J.C. 2018: The Rights Network: 100 Years of the Hohfeldian Rights Analytic. In: Laws 2018, 7, 28. MDPI. https://doi.org/10.3390/laws7030028
4. Van Engers, T.M., Van Doesburg, R. 2015: At Your Service, On the Definition of Services from Sources of Law. In: Proceedings of the 15th International Conference on Artificial Intelligence and Law (ICAIL 2015), pp. 221-0225. ACM. https://doi.org/10.1145/2746090.2746115
5. Van Doesburg, R., Van Engers, T.M. 2019: The False, the Former, and the Parish Priest. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law (ICAIL 2019), pp. 194–198. ACM. https://doi.org/10.1145/3322640.3326718
6. Pierce, B.C. 2002: Types and Programming Languages. 1st edn. The MIT Press.
7. Kahn, G. 1987: Natural Semantics. In: Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1987), pp. 22–39. Springer-Verlag.
8. Plotkin, G.D. 2004: A Structural Approach to Operational Semantics. In: Journal of Logic and Algebraic Programming, vol. 60-61, pp. 17–139. Elsevier. https://doi.org/10.1016/j.jlap.2004.05.001