GRCTC Draft

August 2016

Financial Industry Regulatory Ontology: High Level (FIRO-H)

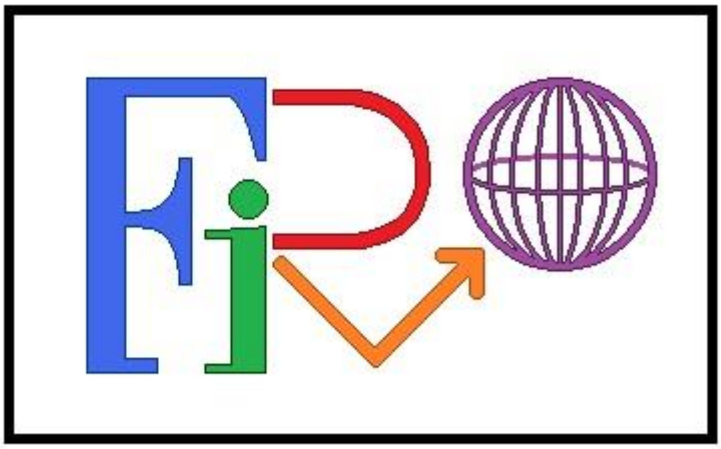
Marcello Ceci

Firas Al Khalil

Leona O'Brien

1. Introduction

The present document describes the Financial Industry Regulatory Ontology (FIRO). This ontology was developed by the Governance, Risk and Compliance Technology Centre to support the knowledge extraction and enhance the knowledge base for its research projects focused on knowledge management systems for regulatory compliance in the financial industry.



2. The Financial Industry Regulatory Ontology (FIRO)

2.1. Overview

FIRO-H(ighLevel)

FIRO-HighLevel (FIRO-H) is a core legal ontology about regulatory compliance. It is centred around the concept of Requirement (Rule Statement) and the concept of Action, and defined in OWL.

FIRO-S(tructure)

FIRO-Structure (FIRO-S) deals with the structure and the semantics of the source document. It accounts for legal and non-legal documents alike. The purpose of FIRO-S is to integrate information from the source text part of Mercury to allow querying, Regulatory Change Management, and reasoning.

FIRO-S relies on LegalDocML for the representation of the structure and the semantics of the legal document. FIRO-S is not formalized in OWL.

FIRO-D(omain)

FIRO-Domain (FIRO-D) identifies the domain ontologies based on FIRO-H. Each contains one rulebook and the related vocabulary. This means that different rulebooks result in different instances of FIRO-Domain, and any common rule (or vocabulary entry) will be present in all relevant instances. Its possible applications include:

- Extract the rules valid for a particular point in time (exploiting RCM of FIRO-S).
- Classify instances of **RegulatoryStatement**s as exceptions to other **RegulatoryStatement**s.
- Classify **BusinessRule**s as ensuring compliance with **LegalRule**s

FIRO-P(urpose)S(pecific)

FIRO-PurposeSpecific (FIRO-PS) is the ontology used for performing reasoning towards a specific application. It is a specialization of one or more FIRO-Ds. It may contain **Factor** instances to represent (either real or fictional) data. Its possible applications include:

- Classify **Event**s (instances of **Action**s) on the basis of their relation to **RegulatoryStatement**s, as either "relevant", "complying", "allowed", "breaching", "exempted".

2.2. Namespace Definitions

Prefix and Namespaces for referenced/external vocabularies

Metadata Term	Value

Prefix and Namespaces for FIRO

Metadata Term	Value
sm:filename	Firo_H_Beta_1.owl
sm:fileAbbreviation	firo-h
OntologyIRI	
owl:versionIRI	
sm:dependsOn	

3. References

LegalDocML: The LegalDocumentXML Specs provides a common legal document standard for the specification of parliamentary, legislative and judicial documents, for their interchange between institutions anywhere in the world and for the creation of a common data and metadata model that allows experience, expertise, and tools to be shared and extended by all participating peers, courts, Parliaments, Assemblies, Congresses and administrative branches of governments. The standard aims to provide a format for long-term storage of and access to parliamentary, legislative and judicial documents that allows search, interpretation and visualization of documents.

The specification of the standard is based on the experience of the Akoma Ntoso language, and for this reason the specification keeps the name "Akoma Ntoso" and the root of the XML-schema will be "akomaNtoso".

The LegalDocumentXML Specs examine the relationships between the proposed XML vocabulary and other similar efforts especially those that already have gained national acceptance or are included in other LegalXML vocabularies (e.g. eContracts). In particular, the CEN Metalex standard is recognized as offering a conceptual meta-model that is appropriate for the management of the compliancy issues between different XML national standards. Akoma Ntoso is from the very beginning compliant with CEN Metalex as an explicit design choice. CEN Metalex compliancy will be considered as the first and most important requirement for comparison between the XML language approved by the TC and any other XML standard.

One of the topics about whom the LegalDocumentXML Specs provides a standardization is a URI-based syntax for legal citations for all types of documents produced by Parliaments and Courts and managed by the XML vocabulary, called the naming convention. The LegalDocumentXML Spec aims to examine and, as much as possible, accept past experiences and decisions of the Akoma Ntoso technical team, which have been consistently using an URI-based syntax. This approach appears similar to the one chosen by the European Legislation Identifier and is also consistent with the http based URIs of the URN:LEX syntax, appendix D.

https://www.oasis-open.org/committees/legaldocml/

FIBO: The Financial Industry Business Ontology (FIBO) is a business conceptual ontology developed by the members of the EDM Council. FIBO provides a description of the structure and contractual obligations of financial instruments, legal entities and financial processes. FIBO is used for harmonization of data across repositories as a common language (i.e. Rosetta stone) for risk analysis and business process automation. FIBO is expressed in the triplestore language of the Web (RDF/OWL) for machine readable inference processing and UML for people readable analysis.

http://www.edmcouncil.org/financialbusiness

Time ontology:

OWL: https://www.w3.org/OWL/

XML: https://www.w3.org/XML/

4. Symbols, Abbreviations and Notation

The following abbreviations are used in this specification:

- SM – Specification Metadata (an OMG standard).
- OWL – Ontology Web Language.
- SME – Subject Matter Expert. Is the legal expert that enriches the legal text with the metadata which, in turn, are used to populate the ontology.
- STE – Semantic Technology Expert. Is the knowledge engineer that extends and populates the ontology.
- URI – Uniform Resource Identifier.
- IRI – Internationalized Resource Identifier.
- XML – eXtensible Markup Language.
- TLC – Top Level Class.

5. Ontology: FIRO-H

5.1. Overview

FIRO-H Metadata

Metadata Term	Value

5.2. Usage Scenarios

FIRO-H is designed to be used in the following scenarios:

- Querying a rulebook on the basis of the characteristics of the action being required (or forbidden, or allowed);
- Reasoning on legal rules to find any exception to any given rule;
- Reasoning on business rules to find if they ensure compliance with a given legal rule;
- Reasoning on data to find out which of them are breaching (or complying to) given legal rules.

5.3. Module: Requirement

Requirement is a TLC and the main element of FIRO-H. It represents a norm, intended as a single rule, as contained in the Mercury Rulebook. Every rule in the Mercury Rulebook corresponds to *at least one* requirement.

The main attributes (data properties) identifying the requirement are taken from Mercury-ML and are the following:

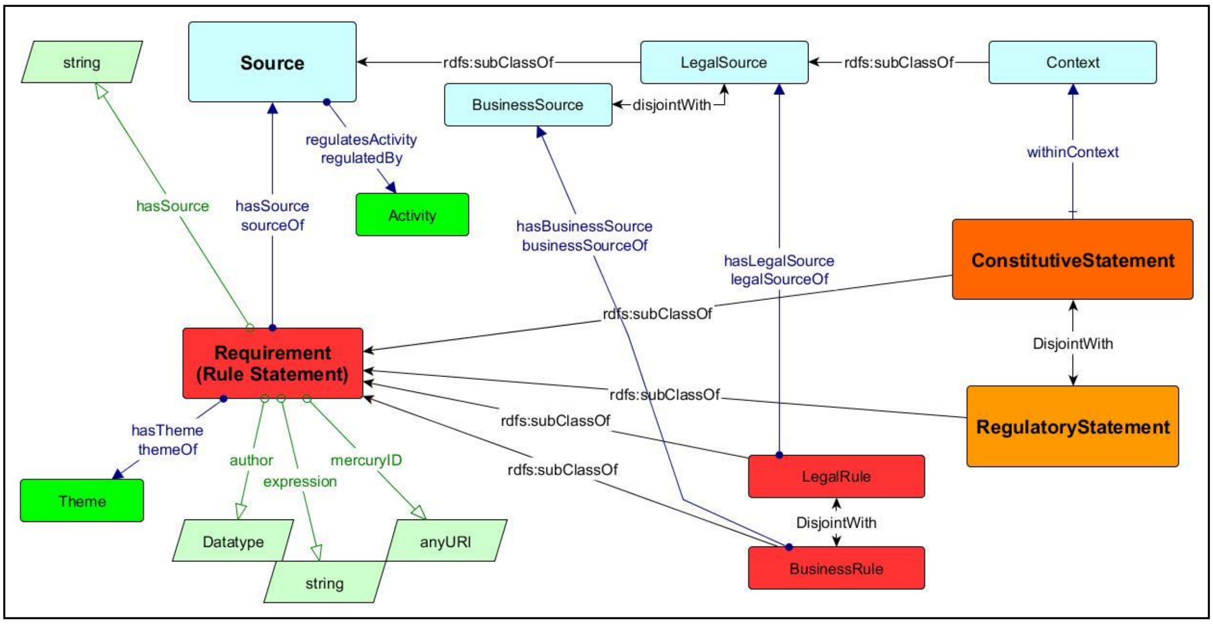
- Author
- originalText
- expression
- id (refID)
- ref (ruleURI)

Every rule has a **Source** (indicated by the ***hasSource*** property), either indicated as a string or as a hierarchical or HTML element in FIRO-S. Entities in FIRO-S are identified by a namespace in the LegalDocML format [CIT].

Rules can also be assigned **Theme**s, if their source has any, with the ***hasTheme*** property.

The Requirement class includes any norm, both legal and non-legal ones. These two categories are represented as subclasses of Requirement: **LegalRule** and **BusinessRule**. Every requirement must be either a legal rule or a business rule. In FIRO, the distinction between legal rules and business rules is not at the level of their contents (both can be either ConstitutiveStatements or RegulativeStatements) but rather at the level of their sources. Legal rules have a special Source (indicated by the ***legalCitation*** property) that is always a **LegalSource** (referenced in the LegalDocML format), while business rules have no particular property. The distinction between legal and business rules is mostly relevant when dealing with their effects (the breach of regulative statements, the context of constitutive statements)

Requirements are also classified based on their content, in two types: constitutive statements (see next paragraph) or regulative statements (see the paragraph after the next).



Classes

Name	Annotations	Class Expressions
Requirement (Rule Statement)	Definition: A norm, i.e. an abstract command either imposing a behaviour or modifying the legal status of reality. The content of a normative text, either with or without legal value. Explanatory note: We extend the notion of legal rule to encompass also non-legal (business) rules. Definition origin: [Gordon et al. 2009] Adapted From:	SubclassOf: hasSource *some* Source hasTheme *min 0* Theme mercuryID some xsd:anyURI
RegulatoryStatement	Definition: a regulatory norm, i.e. an abstract command imposing a behaviour. Explanatory note: see below Definition origin: [Biagioli e Sartor 1993] Adapted From:	Equivalent Class: (hasDeonticModality *value* Obligation *and* requires *some* Action) *or* (hasDeonticModality *value* Prohibition and prohibits *some* Action) *or* (hasDeonticModality *value* Permission *and* allows *some* Action) SubclassOf: Requirement hasCondition *min 1* Condition hasDeonticModality *exactly 1* DeonticModality
ConstitutiveStatement	Definition: an abstract command modifying the legal status of reality. Explanatory note: see below Definition origin: [Austin 1962] Adapted From:	SubClassOf: hasAlethicModality *exactly 1* AlethicModality hasContext *min 1* Context
LegalRule	Definition: a legal rule is a requirement which has legal value in one or more jurisdictions. The important characteristic distinguishing them from non-legal requirements is the liability of the offender to a penalty. Explanatory note: Secondary legislation (e.g. technical specifications coming from official sources) should still be classified as Legal. Definition origin: [Hart 1961] Adapted From:	SubClassOf: Requirement hasLegalSource *some* LegalSource
BusinessRule	Definition: a statement that defines or constrains some aspect of the business. This must be either a term or fact, a constraint, or a derivation. It is 'atomic' in that it cannot be broken down or decomposed further into more detailed business	SubClassOf: Requirement hasBusinessSource *some* BusinessSource

	rules. If reduced any further, there would be loss of important information about the business. A Business Rule is a requirement which has no legal value in any jurisdiction, i.e. its violation does not lead to any “external” penalty. Explanatory note: internal policies are not “Legal Rules”, because the violation of those norms does not entail any liability by the company, but only some internal penalty for some employee. Definition origin: Chapter 3 of the paper "Defining Business Rules ~ What Are They Really?", produced by the Business Rules Group Adapted From:	
Theme	Definition: the theme of a requirement, corresponding to the theme of the source’s unit(s) of analysis. See Mercury-CL. Explanatory note: themes are using for classifying text fragment and retrieving the relevant legislation for a specific business process. Definition origin: Adapted From:	
Source	Definition: the medium that contains the requirement. Explanatory note: Definition origin: Adapted From:	
LegalSource	Definition: the legal text that contains the legal requirement. It is represented in LegalDocML and defined in FIRO-S. Explanatory note: Definition origin: Adapted From:	SubClassOf: Source DisjointWith: BusinessSource
BusinessSource	Definition: the text that contains the business rule. It is represented in Mercury-ML and defined in FIRO-S. Explanatory note: Definition origin: Adapted From:	SubClassOf: Source DisjointWith: LegalSource
Activity	Definition: an activity as concerned by a business rule. It involves several actions. Definition origin:	SubClassOf: hasResponsible min 0 Person

Properties

Name	Annotations	Property Axioms
hasTheme	Identifies the Theme of a Requirement.	InverseOf:

		themeOf Domain: Requirement Range: Theme
themeOf	Identifies the Requirement with a Theme	InverseOf: hasTheme Domain: Theme Range: Requirement
hasSource	Identifies the Source of a Requirement.	InverseOf: sourceOf Domain: Requirement Range: Source
sourceOf	Identifies the Requirement contained in the Source.	InverseOf: hasSource Domain: Source Range: Requirement
hasLegalSource	Identifies the Legal Source of a LegalRule.	SubPropertyOf: hasSource DisjointWith: hasBusinessSource Domain: LegalRule Range: LegalSource
legalSourceOf	Identifies the Legal Rule contained in a Legal Source	SubPropertyOf: hasSource DisjointWith: businessSourceOf Domain: LegalSource Range: LegalRule
hasBusinessSource	Identifies the Business Source of a Business Rule.	SubPropertyOf: hasSource DisjointWith: hasLegalSource Domain: BusinessRule Range: BusinessSource
businessSourceOf	Identifies the Business Rule contained in a Business Source	SubPropertyOf: hasSource DisjointWith: legalSourceOf Domain: BusinesSource

		Range: BusinessRule
refersTo	Specifies the Legal Source that is referred in the Business Source. This property should exist in FIRO-S instead of FIRO-H.	Domain: BusinessSource Range: LegalSource
contains	Specifies the document that is part of the domain instance. This property should exist in FIRO-S instead of FIRO-H.	Domain: Source Range: Source
containedIn	Specifies the document that the domain instance is a part of. This property should exist in FIRO-S instead of FIRO-H.	Domain: Source Range: Source
enactmentDate	Indicates the date on which the rule enters into force. Because this information is in the Source, this property should be superfluous. The reason for including it is mostly for covering the case of commencement rules that target a part of legal text that is smaller than the smallest structural element. If FIRO-S can identify units of analysis smaller than the smallest structural element, this property can be definitely removed.	
originalText	Indicates the original caption for the rule. It is superfluous if the source text is always provided with the rulebook. This should be in FIRO-S.	Domain: Requirement Range: xsd:string
expression	Contains the text of the rule as written by its author.	Domain: Requirement Range: xsd:string
mercuryID	A complete URI assigned to the rule in Mercury-ML.	Range: xsd:anyURI
author	The author of the rule (the SME who first interpreted the legal text and came out with that representation in the rulebook)	Domain: Requirement Range: To be defined (string, uri, class “author”, …)
regulatesActivity	Identifies the Activity regulated by a Policy.	InverseProperty: regulatedBy Domain: Source Range: Activity
regulatedBy	Identifies the Policy that regulates an activity	InverseProperty: regulatesActivity Domain: Activity Range: Source

5.4. Module: Regulative Norm

Regulative norms are the result of *directive acts* [Biagioli e Sartor 1993]. These norms regulate behaviours by introducing deontic modalities (it is obligatory that…, it is prohibited that…, it is permitted that…) for their addressees. Their application is not categorical, but conditional: they specify all their applicability conditions. In order to correctly identify and represent them, it is necessary to identify:

a. The addressee (the subject of the law);
b. the conditions which trigger the rule (also called *conditions of applicability* in [Gordon et al. 2009])
c. the deontic modality (permission, obligation, prohibition);
d. the behaviour being regulated;

A critical aspect of regulative norms is represented by exceptions: under a logical perspective, these are rules whose subsequent is incompatible with another rule. In FIRO-H, an exception is a regulative rule which targets the same or a similar behaviour of another rule with a different deontic modality: for example, if the other rule introduces an obligation, the exception would be either a permission or a prohibition. When modelling rules, the SME must does not need to explicitly state whether a rule is an exception to another rule: the overlapping of targeted behaviours and the difference in modality will allow the OWL reasoner to infer such a relationship.

When a permission targets a subset of a prohibited (or obligatory) behaviour we have a special kind of exception, and the behaviour being allowed is called exempt. This solution is used to represent explicit exceptions (i.e., exceptions explicitly stated within the general rule, and not just incidentally overlapping with another rule).

Reasoning with exceptions also raises specific issues related to the logics used, as they require a non-monotonic (or defeasible) kind of reasoning in order to be properly represented [CIT].

Another specific type of regulative norms is contrary-to-duty obligation, that is, obligations that trigger only when another obligation (the main duty) is breached.

In LegalRuleML regulative rules are represented through the *DeonticSpecification* General Concept and the *PrescriptiveStatement* Node Element.

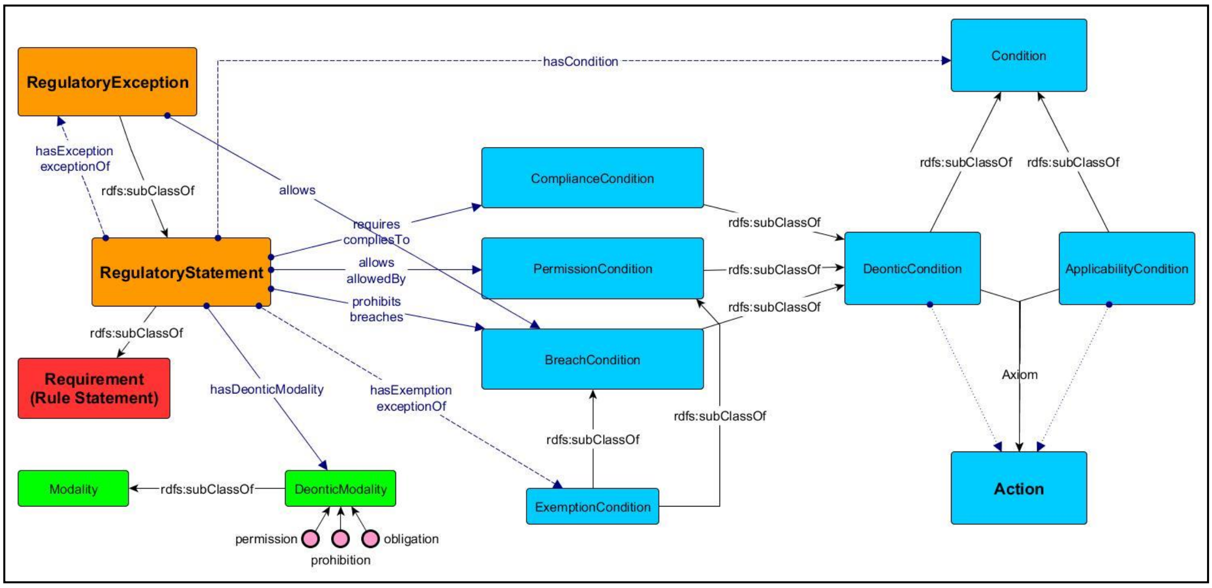
In SBVR, deontic statements are exhaustively represented through operative (or behavioural) rules. Allowed deontic statements include obligation statements, prohibition statements and restricted permission statements. These statements are equivalent, in the sense that the same statement can be expressed in any of the three deontic modalities.

In FIRO, Regulatory Statements are a subclass of Requirement, sibling of Constitutive Statement. Their initial modality (that attributed to it by the SME) is indicated by the *hasModality* property. The relationship with the action, however, is not restricted to that expressed by the initial modality: this means that an obligation will *require* some action, but can also *prohibit* some other action (see *identifying deontic opposites*).

The actions that are required, prohibited, or allowed by a regulatory statement are classified as subclasses of respectively Comply, Breach and Allowed called *Comply_ruleX*, *Breach_ruleX* and *Allowed_ruleX*. The Comply, Breach and Allowed classes are in turn subclasses of Condition.

The Condition class is used to classify all actions relevant for the rule application, either being the behaviour being regulated (they would then be also subclasses of either Comply, Breach or Allowed) or some circumstances in presence of which the main action is relevant for the law ("applicability conditions" in [Gordon et al. 2009]). All conditions are subclasses of the corresponding Action and constitute a specification of it. "Simple" Actions represent a single verb concept: thus a Mercury-ML vocabulary entry and/or one or more instances of Event are classified under this Action (see below 5.4). Conditions, instead, are not used to classify Mercury-ML vocabulary entries: what they do is adding an additional requirement for an instance of Event to become its member. What they require is one of the roles of the Event (corresponding to a verb concept role in the "simple" Action-verb concept) to be involved in another role in another Event (corresponding to another verb concept role to another "simple" Action-verb concept). See document on "how to write rules in FIRO-H" for a detailed explanation of the process.

Exceptions are represented in FIRO either as conditions within operative rules, or through advices of permission that contradict the operative rule. In the latter case, a rule hierarchy is necessary to sort out these contradictions in an automatic way.



Classes

Name	Annotations	Class Expressions
RegulatoryStatement	See above	
Modality	Description: the modality chosen by the editor for the original representation of the regulatory statement in Mercury.	
DeonticModality	Description: the modality chosen by the editor for the original representation of the Mercury regulative rule (SBVR operative rule). Its members are Obligation, Permission, Prohibition. Explanatory note: Every statement can be expressed as an obligation or as a prohibition or as a restricted permission. The modality class just indicates the original representation chosen by the editor.	SubClassOf: Modality Enumeration: obligation, permission, prohibition.
Action	Description: a category of events, or an abstraction of them,	SubClassOf: hasResponsible *min 0* Person

	defined arbitrarily, as expressed by a verb, complete with the subject and (if present) its direct complements (e.g. object, indirect object, location). Actions have at least one factor (the subject). It is the result of the interpretation on the behavior required by the rule. Actions are expressed by SBVR verb concepts. <u>Explanatory note</u>: instances of Action correspond to verb concepts in the Mercury Vocabulary. hasResponsible property and Agent will be added in the future.	hasVerb *min 1* Verb hasSubject *exactly 1* Factor incompatibleWith *min 0* Action partOfActivity *min 0* Activity mercuryID some xsd:anyURI
Condition	<u>Description</u>: a condition is an action used in a rule. A condition shares the same properties of its general action and may restrict factors by specifying: 1. its scope or value, or 2. The role it plays in another condition (grammatical complement). <u>Explanatory note</u>:	<u>SubClassOf</u>: Action
Deontic Condition	<u>Description</u>: a condition that determines if a relevant event complies/breaches a rule.	<u>SubClassOf</u>: Condition
Applicability Condition	<u>Description</u>: a condition that determines if a given event is relevant to a given rule, or not.	<u>SubClassOf:</u> Condition
ComplianceCondition	<u>Description</u>: A Condition that is deontically qualified by an obligation, or a condition that is incompatible with a breach of an obligation. <u>Explanatory note</u>: Every regulatory statement that is not a Permission will have at least one subclass of the Comply class linked to it by the "compliesTo" property. In case of a RegStatement with modality "obligation", this subclass constitute the main field of application of the requirement. In case of a RegStatement with modality "prohibition", this subclass constitute an indirect field of application of the requirement (for actions that are	<u>SubClassOf</u>: Deontic Condition compliesTo *min 1* RegulatoryStatement

	"incompatibleWith" a Breach class)	
BreachCondition	Description: A Condition that is deontically qualified by a proibition, or a condition that is incompatible with a compliance to an obligation. Explanatory note: Every regulatory statement that is not a Permission will have at least one subclass of the Breach class linked to it by the "breaches" property. In case of a RegStatement with modality "prohibition", this subclass constitute the main field of application of the requirement. In case of a RegStatement with modality "obligation", this subclass constitute an indirect field of application of the requirement (for actions that are "incompatibleWith" a Comply class)	SubClassOf: Deontic Condition Breaches *min 1* RegulatoryStatement hasDefence *min 0* Defence
PermissionCondition	Description: A Condition that is deontically qualified by a permission. Explanatory note:	SubClassOf: Deontic Condition allowedBy min 1 RegulatoryStatement
ContraryToDuty	Description: a regulatory statement which applies to those responsible for the actions that breach another ("main") regulatory statement. Explanatory note:	
RegulatoryException	Description: a regulatory statement that allows an action that is a subset of a Breach or a Comply. Explanatory note:	EquivalentTo: allows *min 1* BreachCondition SubClassOf: RegulatoryStatement ExceptionOf *min 1* RegulatoryStatement
ExemptionCondition	Description: An action that is the target of an exception (i.e., an action that is Allowed and also a subset of a Breach). Explanatory note:	SubClassOf: PermissionCondition ExemptionOf *min 1* Condition
hasDeonticModality	Identifies the original modality of the regulatory statement (the one assigned by its author).	Domain: RegulatoryStatement Range: DeonticModality
hasCondition		InverseProperty: conditionOf Domain: Requirement

		Range: Condition
conditionOf		InverseProperty: hasCondition Domain: Condition Range: Requirement
hasApplicabilityCondition		SubPropertyOf: hasCondition InverseProperty: applicabilityConditionOf Domain: Requirement Range: ApplicabilityCondition
applicabilityConditionOf		SubPropertyOf: conditionOf InverseProperty: hasApplicabilityCondition Domain: ApplicabilityCondition Range: Requirement
requires	Identifies the Comply class that is deontically qualified by the regulatory statement	SubPropertyOf: hasCondition Domain: RegulatoryStatement Range: ComplianceCondition InverseProperty: complyTo
complyTo	Identifies the regulatory statement that specific Comply class complies with.	SubPropertyOf: conditionOf Domain: ComplianceCondition Range: RegulatoryStatement
Prohibits	Identifies the Breach class that is deontically qualified by the regulatory statement	SubPropertyOf: hasCondition Domain: RegulatoryStatement Range: BreachCondition Property chain: requires 0 incompatibleWith InverseProperty: breaches
Breaches	Identifies the regulatory statement that specific Breach class breaches.	SubPropertyOf: conditionOf Domain: BreachCondition

		Range: RegulatoryStatement
Allows	Identifies the Allowed class that is deontically qualified by the regulatory statement	SubPropertyOf: hasCondition Domain: RegulatoryStatement Range: PermissionCondition InverseProperty: allowedBy
allowedBy	Identifies the regulatory statement that allows the specific Allowed class.	SubPropertyOf: conditionOf Domain: PermissionCondition Range: RegulatoryStatement
hasExemption	Identifies a condition that is subject to an "allows" deontic qualification that is different from that of the domain regulatory statement, for a subset of the target of the domain regulatory statement.	Domain: BreachCondition Range: ExemptCondition
exemptionOf	Identifies the rule that imposes a different deontic qualification from "allows" for a superset of the domain condition.	InverseProperty: hasExemption Domain: ExemptCondition Range: BreachCondition
hasException	Identifies the rule that allows a subset of the condition targeted by the domain regulatory statement	Domain: RegulatoryStatement Range: RegulatoryException
exceptionOf	Identifies the rule that requires or prohibits a superset of the condition targeted by the domain regulatory statement	Domain: RegulatoryException Range: RegulatoryStatement

5.5. Module: Constitutive Norm

The concept of constitutive norms, as distinguished from regulative rules as defined above, was refined by John R. Searle, who provided the following definition:

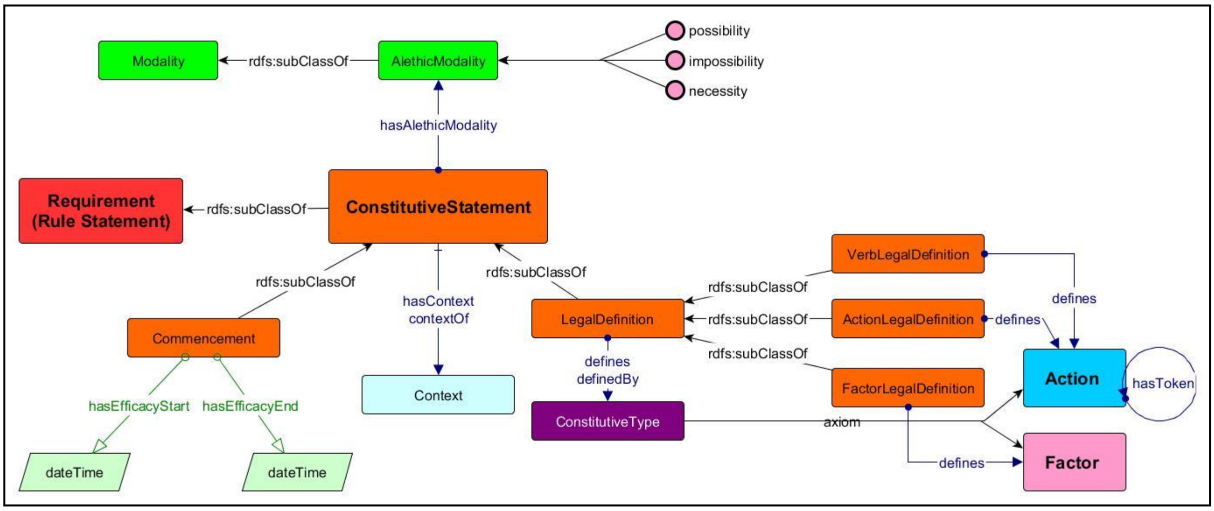
> [R]egulative rules regulate antecedently or independently existing forms of behaviour […]. But constitutive norms do not merely regulate, they create or define new forms of behaviour. The rules of football or chess, for example […] create the very possibility of playing such games. (1969, p. 33).

Constitutive norms are the result of *declarative acts* [Biagioli e Sartor 1993]. These norms introduce new abstract classifications of existing facts and entities. Those classifications are called institutional facts (e.g. marriage, money, private property) and they emerge from an independent ontology of "brute" physical facts. Differently from regulative rules, constitutive norms have no deontic content: they do not introduce obligations, prohibitions or permissions. Instead, they tipically take the following form:

a counts as *b* in context *c*

In order to capture these rules, it is thus necessary to identify three elements:

- a material (or previously identified) phenomenon (*a*);
- an abstract concept that is created by the constitutive rule itself (*b*);
- the context (*c*).



Constitutive norms are called "determinative rules" in [T. F. Gordon, G. Governatori and A. Rotolo, Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain, in: A. Paschke, G. Governatori, J. Hall, eds., Rule Interchange and Applications, Berlin, Springer, 2009, pp. 282-296]. In LegalRuleML, that class is currently represented as *ConstitutiveStatement* Node Element. In Normative Multi-Agent Systems they are formalized as belief rule of normative agents: from a knowledge representation point of view, they behave as data abstraction in programming languages [Boella and Van der Torre 2004].

The Italian and Scandinavian school of legal philosophy introduced further distinctions within constitutive rules [CIT Carcaterra, Conte, Roversi], identifying i.e.

- rules that directly constitute new entities, introducing "sufficient" conditions for the new entity to exist (*thetic constitutive rules*);
- rules that mere create the *possibility* of new entities, introducing "necessary" conditions for the new entity to exist (*eidetic constitutive rules*);
- rules that without constituting new entities yet prescribe necessary conditions for them to exist (*anankastic-constitutive rules*).

FIRO-H models constitutive rules according to this doctrine, not with a general model (as this would be too abstract for an SME to use) but rather modelling single types of constitutive rules, one at a time[1]. In this way, the SME easily understands what template to use for representing a specific type of constitutive rule, without having to worry about eidetic, thetic or anankastic rules.

SBVR's restricted language does not explicitly include institutional facts or constitutive norms, although it has a rule category called structural (or definitional) rules that can be used for the purpose. These rules are represented in the language through necessity statements instead of deontic modalities.

[1] E.g. commencement rules are *thetic constitutive rules*, legal definitions are *eidetic constitutive rules*, and relative necessities are *anankastic constitutive rules*.

In FIRO-H, constitutive norms are represented as subclass of Requirement called ConstitutiveStatement. Version 1 of FIRO-H supports the following types of constitutive statement:

- Legal definitions
- Commencement rules

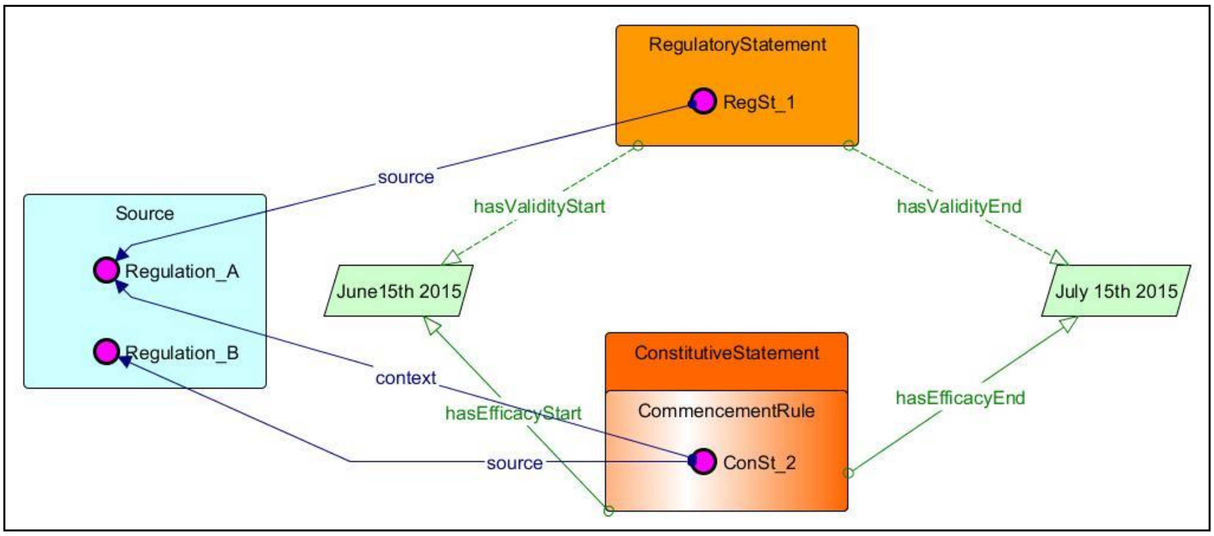
Every Constitutive Statement has exactly 1 constitutive modality (property **hasModality**, range **CostModality**): possibility, impossibility, necessity. This property indicates the modality in which the original rule entry in the rulebook has been modelled. In fact, a necessity statement can be transformed into impossibility statement, and vice versa, but the hasModality property will always indicate the original modality that was assigned to the rule by the SME.

Every Constitutive Statement has a **Context**. In legal theory, the context of a constitutive rule is used to identify the limits within which the constitutive effects of the rule take place. The concept of context is used in FIRO-H in a slightly different way: it represents the domains where the rule is relevant. This difference becomes evident when dealing with commencement rules (see below): while in the legal theory the context of a commencement rule is the entire legal system (jurisdiction), in FIRO-H **Context** is used to indicate which legal fragments have their coming into force date affected by the **CommencementRule**. Context can be specified in terms of **Theme**s, **Activit**ies, **Rulebook**s, or **Source**s. For legal rules, the context must include a **LegalSource**.

Commencement Rules

Commencement rules include a efficacy start date and/or an efficacy end date, indicating the point in time when the effects of the law (indicated as range of the hasLegalContext property) begin and/or end, respectively. These dates are indicated by the datatype properties hasEfficacyStart and hasEfficacyEnd, respectively.

A CommencementRule can indicate more than one begin and/or end dates. It can also express the date in relative terms (e.g. 30th day after a given date).



Legal Definitions

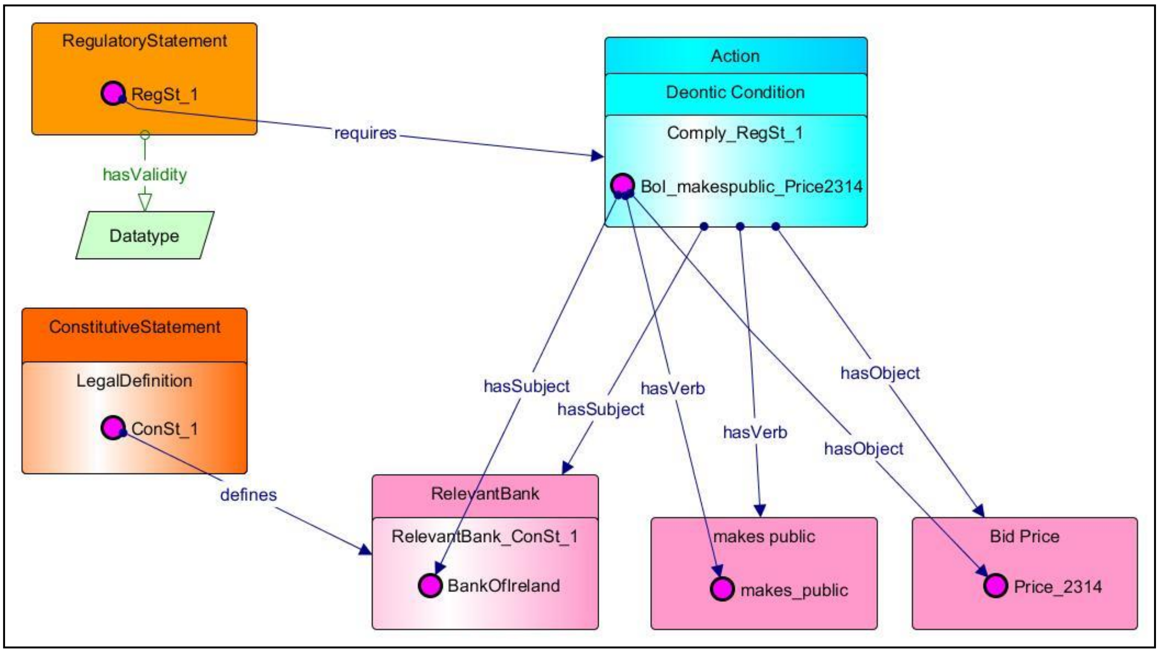
Legal definitions are used to introduce specific (and legally valid) classification for the following elements of FIRO-H:

- **Factor**s (noun concepts);
- **Action**s (verb concepts, including verb concept roles as factors);

- **Verb**s (verb symbol). In the Mercury model, it is not possible to define verbs independently. For overcoming this, FIRO creates an abstraction of the most general verb concept using that verb (i.e. Thing *verb* Thing) and defines that as if it were an Action.

Legal Definition of Factors

When the element defined by the legal definition is an abstract entity (represented in Mercury as a general noun concept and in FIRO as a factor), the effect of the rule (the new, abstract concept being defined, the definiendum) is represented as a new factor. The concrete, previously identified phenomenon used to define the new concept (the definiens) is indicated as equivalent class through an axiom, in a way similar to the way the Deontic Condition of regulative rules is represented, with the difference that, while deontic conditions specify actions, these axioms identify a factor.



Add example showing the definiendum and the axiom, showing that the axiom identifies a factor and not an action.

Legal Definition of Verb Concept with different Verbs but the same Complements

This solution covers the case when the legal definition defines a new verb that describes the same action as the definiens, but specifying the conditions when the definiens can be labelled with this new verb. For example, a rule such as “shoot well: a person shoots well if he shoots the ball and the ball goes forward” defines the verb “person shoots well ball” starting from the verb “person shoots ball”: it thus only specifies a new verb for the same action involving the same complements.

In Mercury, verbs don’t have their own entries, and thus they cannot be defined on their own. To represent this type of legal definition, we thus still need to create a new verb concept for the definiendum. However, because the complements do not change, these don’t need to be specified in FIRO: because the class for the definiendum is inferred as subclass of the definiens, it will inherit the axiom of the definiens, including the indication of its complements.

Legal Definition of Verb Concepts

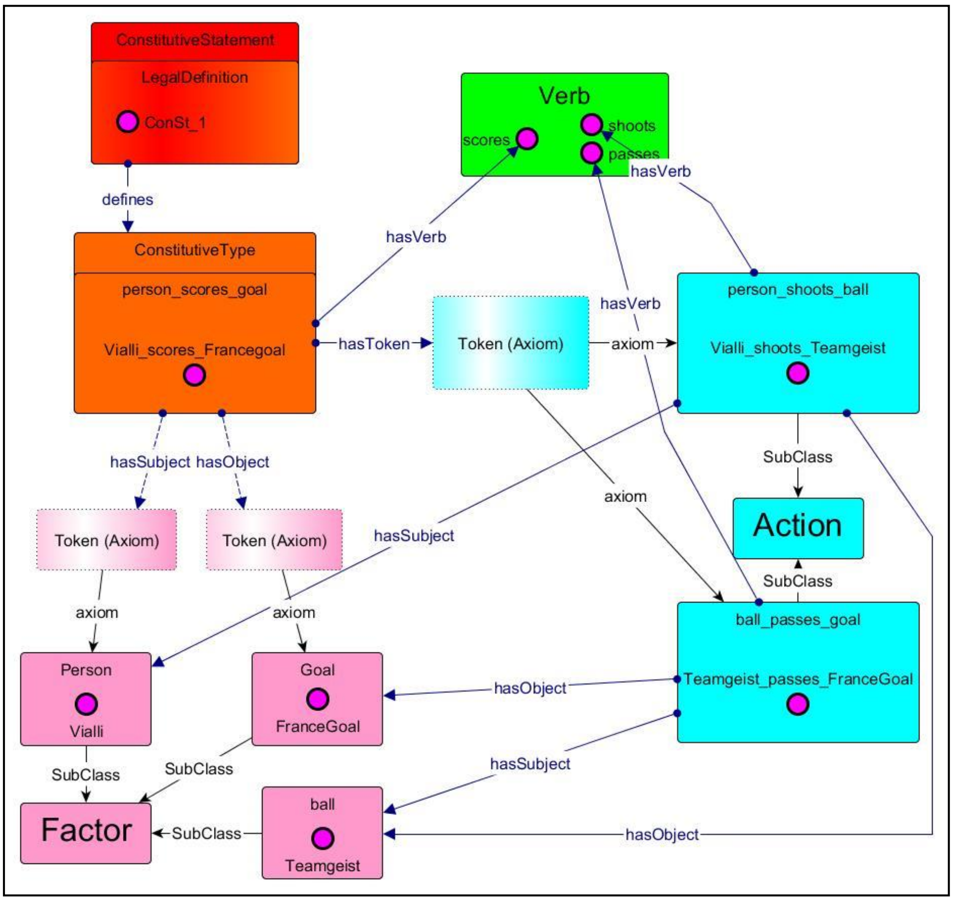
Sometimes a legal definition defines a new verb that represents an action whose complements are different from any of the actions present in the definiens. Let's consider the following example:

Necessity: (recipient financial institution *notifies* recipient *of* the receipt *of* the order) *counts as* (recipient financial institution *accepts* order) *in* **Chapter X general**

In this case, the definiendum (recipient financial institution accepts order) has verb concept roles (recipient financial institution and order) that are present in the definiens, but not all in the same action: more precisely, "recipient financial institution" is present in the verb concept "recipient financial institution notifies recipient of receipt" and "order" is present in the verb concept "order has receipt".

To represent this kind of definiendum it is not possible to create a subclass of the definiens, as this would entail that the definiendum inherits all the properties of the definiens, including those that do not pertain to it. It is instead necessary to create a distinct subclass of Action for the definiendum, and link it to the definiens' conditions through the property hasToken. In this way, the definiens is referred by the definiendum, but without inheritance of properties. The Condition is linked to the definiendum through an axiom.

Defining verb concept roles. Because the new verb concept does not inherit any complement from its definiens, it is necessary to specify them through an axiom that assigns the appropriate subproperty of hasFactor to the correct verb concept role, in a way similar to that which is used for the legal definitions of factors.



Possibility Statements (need to expand that)

Possibility statements are used either for representing exemptions (either in legal definitions or in commencement rules) or for representing *statements of facts with legal relevance* e.g. the fact that "ESMA will publish technical standards on what constitute a prevalent market condition".

Classes

Name	Annotations	Class Expressions
ConstitutiveStatement	See above	See above
AlethicModality	Description: Indicates the modality chosen by the editor for the original representation of the Mercury constitutive statement. Its members are Necessity, Impossibility, Possibility. Explanatory note: Every statement can be expressed as an obligation or as a prohibition or as a restricted permission. The modality class just indicates the original representation chosen by the editor.	SubClassOf: Modality Enumeration: possibility, necessity, impossibility.
LegalDefinition	Explanatory note: These rules, often contained in the first article of regulations, specify the meaning of specific terms (*b*) that are found throughout the regulative text or in a subpart of it (*c*). When terms specifically appear in *legal definitions*, the interpretation of their meaning cannot be arbitrary: every time they occur in the text, they must be understood as meaning the exact combination of words (or sentences) that appear in the definition (*a*).	SubClassOf: ConstitutiveStatement defines min 1 ConstitutiveType
FactorLegalDefinition		SubClassOf: LegalDefinition defines min 1 Factor
VerbLegalDefinition		SubClassOf: LegalDefinition defines min 1 Action
ActionLegalDefinition		SubClassOf: LegalDefinition defines min 1 (Action and hasToken min 1 Action)
Commencement	Explanatory note: these rules indicate a (directly or indirectly identified) time paramenter (*a*) as the starting point for the validity (*b*) of the regulation (or part of it) (*c*).	SubClassOf: ConstitutiveStatement hasCommencementDate min 1 xsd:dateTime
ConstitutiveType	An action or a factor that is defined by a LegalDefinition.	SubClassOf: definedBy min 1 LegalDefinition
Context		SubclassOf: Source
LegalContext	The LegalSource that constitutes the "area of relevance" of a certain constitutive rule. E.g., terms that are classified under a certain ConstitutiveType will be labelled as such only if contain within the LegalContext	SubclassOf: LegalSource Context

	of the Legal Definition related to that ConstitutiveType.	

Properties

Name	Annotations	Property Axioms
hasAlethicModality	Identifies the original modality of the constitutive statement (the one assigned by its author).	Domain: ConstitutiveStatement Range: AlethicModality
hasContext		
contextOf		
hasLegalContext	Identifies the context of a constitutive statement.	SubPropertyOf: hasContext Domain: ConstitutiveStatement Range: Context
legalContextOf		SubPropertyOf: contextOf Domain: Context Range: ConstitutiveStatement
defines	Identifies the constitutive type defined by a Legal Definition	Domain: ConstitutiveStatement Range: ConstitutiveType
definedBy		Domain: ConstitutiveType Range: ConstitutiveStatement
hasToken	Identifies the verb concept used by the legal definition to define the brand new verb concept.	Domain: ConstitutiveType and Action Range: Action
tokenOf		Domain: Action Range: ConstitutiveType and Action
hasCommencementDate		Domain: Commencement Range: xsd:dateTime
hasEfficacyStart	Identifies the commencement date for the law defined as Context, as defined by a Commencement.	SubPropertyOf : hasCommencementDate Domain : Commencement Range : xsd :dateTime
hasEfficacyEnd	Identifies the date when the effects of the Context cease, as defined by a Commencement.	SubPropertyOf : hasCommencementDate Domain :

	Commencement Range : xsd :dateTime

5.6. Module: Action

Requirements in FIRO-H express their effects in terms of **Conditions**. **Conditions**, in turn, are specifications of **Actions**. The specification happens either through a function (keyword) or by identifying factors in common among them. Similarly, Factors specify noun concepts by identifying the **Actions** (or **Conditions**) they are involved in.

Actions represent abstract **Event**s as SBVR verb concepts. Every **Action** thus refers to a verb concept entry in the Mercury Vocabulary, and includes at least one verb concept role and exactly one verb symbol.

Verb symbols are represented as individuals and classified under the **Verb** class.

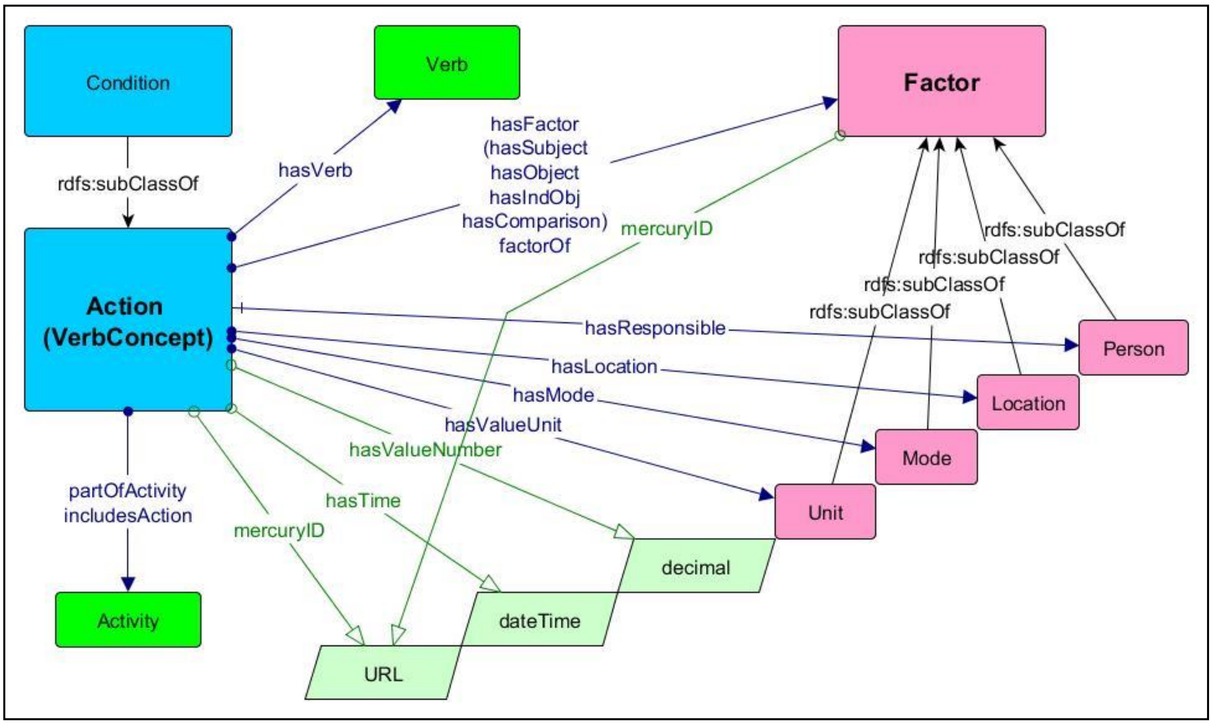
Verb concept roles (roles played by noun concepts in a verb concept) are classified under the **Factor** class. The Factor instances in FIRO also contain attributes of their related noun concepts: all **Factor**s are classified under the **Factor** (general noun concept) indicated as “general concept” in the Mercury Vocabulary entry. General and Unitary verb concept roles are represented as subclasses of **Factor**, while Individual noun concepts roles are represented as individuals, members of the **Factor** class.

The specific role played by verb concept roles in a specific verb concept is expressed by the object property linking the **Action** to the **Factor**. The following complements are supported by the current version of FIRO-H:

- Subject
- Object
- Indirect Object
- Comparison
- Location
- Mode
- Value
- Time

Two more object properties are used to enrich the semantics of the action beyond the mere verb concept they express:

- hasResponsible: indicates the person that has the responsibility for a specific action in a specific environment (e.g. a company).
- involvedIn: indicates the activity which encompasses the action. Used to retrieve all the actions (and related requirements) that are relevant for a certain activity.



Classes

Name	Annotations	Class Expressions
Action	Description: see above Explanatory note: instances and subclasses of Action (except Conditions) correspond to verb concepts in SBVR. The link to the Mercury Vocabulary is ensured by the property hasVocEntry.	See above
Verb	Description: The predicate expressing the Action. Explanatory note: It corresponds to the verb symbol in SBVR.	
Factor	Description: a factor is a (generic or specific) entity that plays a role in one or more actions contained in the same rule. It is a result of the interpretation of the entities involved in the rule. Factors are expressed by SBVR verb concept roles. Explanatory note: It corresponds to the verb concept role in SBVR. The link to the noun concept in the Mercury Vocabulary is ensured by the hasVocEntry property. General and unitary noun concepts are represented as subclasses, while individual noun concepts are represented as individuals. The "General concept" attribute in SBVR is translated into the rdfs:subClassOf predicate.	SubClassOf: mercuryID some xsd:anyURI
Person	Description: a (natural or legal) person.	rdfs:subClassOf Factor

Location	Description: a (physical or abstract) location.	rdfs:subClassOf Factor
Mode	Description: a modality of the action.	rdfs:subClassOf Factor
Unit	Description: a measuring unit, including a legal tender.	rdfs:subClassOf Factor
Activity	Definition: an activity as concerned by a business rule. It involves several actions.	See above

Properties

Name	Annotations	Property Axioms
hasVerb	Identifies the verb symbol of a verb concept.	InverseProperty: verbOf Domain: Action Range: Verb
verbOf		Domain: Verb Range: Action
hasFactor	Identifies generically a verb concept role of a verb concept	InverseProperty: factorOf Domain: Action Range: Factor
factorOf		Domain: Factor Range: Action
mercuryID	Data property. Identifies the ID of the vocabulary entry. Its range corresponds to Mercury-ML's nounConceptID for noun concepts, and to Mercury-ML's verbConceptID for verb concepts.	See above
partOfActivity	Identifies the activity related to an action.	InverseProperty: includesAction Domain: Action Range: Activity
includesAction	Identifies the actions that an activity involves.	Domain: Activity Range: Action
hasResponsible	Identifies the agent responsible for an action.	SubPropertyOf: hasFactor InverseProperty: responsibleOf Domain: Action Range:

		Person
responsibleOf	Identifies the action that an agent is responsible for.	SubPropertyOf: factorOf Domain: Person Range: Action
hasSubject	Identifies the subject of an action.	SubPropertyOf: hasFactor InverseProperty: subjectOf Domain: Action Range: Agent
subjectOf		SubPropertyOf: factorOf Domain: Agent Range: Action
hasObject	Identifies the object of an action.	SubPropertyOf: hasFactor InverseProperty: objectOf Domain: Action Range: Factor
objectOf		SubPropertyOf: factorOf Domain: Factor Range: Action
hasIndirectObject	Identifies the indirect object of an action.	SubPropertyOf: hasFactor InverseProperty: indObjectOf Domain: Action Range: Factor
indirectObjectOf		SubPropertyOf: factorOf Domain: Factor Range: Action
hasComparison	Identifies the second term of comparison of an action. The first term of comparison is always the subject.	SubPropertyOf: hasFactor InverseProperty: comparisonOf

		Domain: Action Range: Factor
comparisonOf		SubPropertyOf: factorOf Domain: Factor Range: Action
hasLocation	Identifies the location of an action.	SubPropertyOf: hasFactor InverseProperty: locationOf Domain: Action Range: Location
locationOf		SubPropertyOf: factorOf Domain: Location Range: Action
hasMode	Identifies the modality of an action.	SubPropertyOf: hasFactor InverseProperty: modeOf Domain: Action Range: Mode
modeOf		SubPropertyOf: factorOf Domain: Mode Range: Action
hasValueUnit	Identifies the unit in which the value of an action is expressed.	SubPropertyOf: hasFactor InverseProperty: value_cOf Domain: Action Range: Unit
valueUnitOf		SubPropertyOf: factorOf InverseProperty: hasValue_c Domain: Unit Range:

		Agent
hasValueNumber	Data property. Identifies the value of an action.	SubPropertyOf: hasFactor Domain: Action Range: xsd:decimal
hasTime	Data property. Identifies the time of an action.	SubPropertyOf: hasFactor Domain: Action Range: xsd:dateTime