Functional Design of MirrorMe

Rieks Joosten (rieks.joosten@tno.nl)

15 July 2017

Contents

Chapter 1

Introduction

This document[1] defines the functionality of an information system called 'MirrorMe'. It defines the database and the business services of MirrorMe by means of business rules[2]. Those rules are listed in chapter 2, ordered by theme.

The diagnosis in chapter 3 is meant to help the authors identify shortcomings in their Ampersand script.

[1]This document was generated at 15-7-2017 on 20:12:38, using Ampersand-v3.8.6 [HEAD:49946cb70], build time: 08-Jul-17 09:11:28 Coordinated Universal Time.

[2]Rule based design characterizes the Ampersand approach, which has been used to produce this document.

Chapter 2

Shared Language

This chapter describes functional requirements for 'MirrorMe' in natural language. It contains definitions and agreements. The purpose of this chapter is to create shared understanding among stakeholders. All definitions and agreements have been numbered for the sake of traceability.

2.1 Arguments

The structure of legal arguments has been studied by scholars such as Toulmin~[**?**], Verheij (name some more). To help legal professionals construct such arguments, the structure of arguments must be formalized. For this purpose, we can consider each (legal) case as a set of statements. Each statement can be considered true or

Definition 1: a set of TTexts that are controlled by a single authority and (together) serve a particular purpose *Scope*

If people make claims, they do observations, and they will reason about them. In order to decide whether a claim is true, statements are organised in a structure that represents the argument. The concept "statement" is used to represent claims, observations, and all other utterances that can be considered true or false. Statements and Claims are modeled/implemented as TTexts.

Definition 2: the collection of one template string, one instance thereof, and one value within some scope *TText*

Let us treat a statement made by an individual as a claim of that individual that the statement is true. For this reason, each individual who makes the claim can be registered in the relation "claimant". The `claimant` 'claims' (a) the validity of the `ttTemplate` and (b) the veracity of `ttValue`. Thus, the `claimaint` is solely authorized to change `ttValue` and `ttTemplate`.

Agreement 1 A claimant is the party that makes a claim, especially one that is legally cognizable.

Agreement 2 the Actor that is the claimant of all toplevel TTexts.

Agreement 3 The application can register a moment as the time a case has been created.

Agreement 4 A statement that needs another statement to be true, is registered in the relation `requires`.

2.2 Case Management

This pattern contains some basic administration for cases.

Als iemand inlogt in het systeem moet diens 'context' worden geactiveerd, d.w.z. de gegevens over de persoon die het systeem nodig heeft om te kunnen berekenen wat hij/zij wel en niet mag doen, en welke gegevens van het systeem daarbij mogen worden gebruikt. Om zulke computations te kunnen maken wordt een aantal zaken geregistreerd en aan één persoon gekoppeld.

Definition 3: een verzameling gegevens die (een deel van) de gebruikerscontext van één gebruiker binnen het systeem beschrijft — *Account*

Een persoon gebruikt een gegevensruimte (en heet dan 'user') door met een browser (bijv. Chrome of Firefox) het systeem te benaderen dat de gegegensruimte beheert. Als meerdere personen een gegevensruimte delen, moet het systeem de context van elk van hen kunnen onderscheiden, bijvoorbeeld om:

- de interactie 'klein' te houden, d.w.z. alleen gegevens te laten zien die voor hem/haar relevant zijn;
- ervoor te zorgen dat een user niet ziet wat hij niet mag zien;
- te kunnen bijhouden welke persoon, of welk(e) organisatie(onderdeel) verantwoordelijk is voor een zekere transactie;
- automatisch gegevens betreffende de user of zijn context aan transacties toe te kunnen voegen

We gebruiken de term 'SESSION' of 'session' om de verzameling van gegevens betreffende één (actieve) user mee aan te geven. Deze term correspondeert met de gelijknamige term browsers gebruiken, nl. 'een verbinding (door de browser) met een webservice (die een URL heeft)'. Het systeem houdt één session bij voor elke actieve user, d.w.z. voor elke browser die het systeem benadert. Merk op dat dit in het bijzonder geldt als de user in verschillende tabbladen van dezelfde browser het systeem benadert - er is dan toch maar één session (en één user).

Definition 4: een verzameling van gegevens die de context beschrijven waarin één persoon het systeem gebruikt — *SESSION*

Agreement 5 A SESSION may be linked to an Account, which specifies the (user)context of the session.

2.3 Placeholder Extraction and TText Hierarchy creation Service

This service ensures that - `ttTemplateParsedText`=`ttTemplate` AND - `ttTemplatePlaceholders` contains all `TTName`s that are mentioned in the `ttTemplate` of a `TText`.

The idea is that the specification of a TText is parsed to see if it contains names of `TText`s. Such names are recognized by the fact that they are surrounded by square brackets (`[` and `]`).

So, - `ttTemplateParsedText` stores the text that has been parsed (in PHP) to produce the contents of `ttTemplatePlaceholders`. This implies that whenever `ttTemplateParsedText` is empty for some `TText`, `ttTemplatePlaceholders` is empty for that same atom. - whenever `ttTemplateParsedText` differs from `ttTemplate`, - it is first removed, and `ttTemplatePlaceholders` are discarded - then `ttTemplate` is being parsed, and `ttTemplateParsedText` and `ttTemplatePlaceholders` are filled again.

All of this happens in the same `Scope`, i.e. in the scope to which the `TTexts` belong. Note that for this to work, all `TTexts` that are mentioned in a `ttTemplate` must exist.

Definition 5: a label used to identify a TText (within a scope) *TTName*

Definition 6: a sequence of words *TTPhrase*

Agreement 6 Statements that are in a given scope are considered valid within that scope.

Agreement 7: TTexts that have no placeholders detected, yet have a ttTemplate, must have been parsed

Agreement 8: Whenever a template phrase changes, the parsed text must be deleted so that the template is parsed again

Agreement 9: TTexts whose template text is not parsed, must not specify placeholders that are detected

2.4 Placeholder Replace-By-TTValue Service

This service ensures that the `ttInstance` phrase of a TText is the `ttTemplate` phrase of that TText, where every occurrence of a reference to another TText in that template (i.e. occurences of `[<TTName>]`, called 'placeholder's) **is replaced by the `ttValue`** of that other TText, provided that the `ttValue` is populated for that TText.

Definition 7: (the representation of) a value of a TText *TTValue*

The property ttInstanceResetReq is used to start reconstructing an instance phrase of a TText.

Agreement 10 If a TText has the property ttInstanceResetReq, this means that its instance phrase needs to be (re)constructed from scratch.

Agreement 11 SRC TText has been used to replace placeholders in the TGT TText instance phrase

Agreement 12 References to this TText (i.e.: placeholders) exist, and have been replaced with the specified TTValue

Agreement 13: If a TText has a value, and is used in a TText that is not being re-initialized, then it must appear in the `ttValueIsUsedInInstanceOfTText` of the TText.

Agreement 14: When the value of a TText differs from the value used in replacements, or has become void, then the instance-texts of all TTexts in which the replacements took place must be discarded and recreated.

2.5 TText Reset Service

This service relinquishes any depencies that the ttInstance of a TText has on other TTexts (as registered in `ttValueIsUsedInInstanceOfTText`). This condition will be realized by setting the property `ttInstanceResetReq` for that TText.

Agreement 15: If a TText has a template phrase and no instance phrase, the TText must be reset/initialized, thus allowing its instance phrase to be constructed

Agreement 16: Resetting a TText means that the registration of replaced placeholders (for that TText) must be reset (cleared/deleted).

Before reconstructing an instance phrase of a TText, the ExecEngine must have discarded all administration related to that TText.

Agreement 17: A TText that needs to be (re-)initialized and does not use values of TTexts in its UsedValue, must have the specification-text as its UsedValue, which completes the (re-)initialization.

2.6 Validity

In order to talk about true and false statements in a precise way, we need the idea of contexts.

Consider the statement "John has blond hair". If this statement is known to be true in some context `c`, the tuple ("John has blond hair", c) can exist in the relation `true`. However, if this tuple is not in the relation `true`, it does not follow that John does not have blond hair. The truth of the TText value (as decided by its claimant) is accepted by the owner of the Scope

Agreement 18 A statement that is considered true within a context is registered in the relation `true`.

The relation `false` is dual to `true`.

Agreement 19 A statement that is considered false within a context is registered in the relation `false`.

For reconstructing events, it can be necessary to administer the moment an observation or a claim is made. For this reason, we use the relation "observed".

Agreement 20 The application can register a moment as the time a statement has been made.

Every statement (TText) that is in scope is considered valid. This means that it can be true or false, given that all its placeholders have been instantiated.

Agreement 21: A statement can be true or false in a context only if it is valid within that context.

Every statement (TText) that is in scope is considered valid. This means that it can be true or false, if all its placeholders have been instantiated.

Agreement 22: A statement cannot be both true and false in the same context.

2.7 Wetsartikelen

Chapter 3

Diagnosis

This chapter provides an analysis of the Ampersand script of *'MirrorMe'*. This analysis is intended for the author(s) of this script. It can be used to complete the script or to improve possible flaws.

MirrorMe does not specify which roles may change the contents of which relations.

MirrorMe assigns rules to roles. The following table shows the rules that are being maintained by a given role.

Rule	ExecEngine	SYSTEM
caseAuthor	×	
TText template parsing - extract placeholders	×	
TText template parsing - delete parsed templates	×	
TText template parsing - delete placeholders used in template	×	
Compute transitive closure of 'ttIsUsedBy'	×	
Create TTexts for placeholders (if necessary)	×	
When the name of a TText is in the list of placeholders of another TText, the first TText is said to be used by the second TText	×	
A (first) TText is only used by a (second) TText if the name of the first is in the list of placeholders of the second	×	
When a TText is used by another TText, it inherits the latter's Scope	×	
Register that the value of a TText has been used to replace placeholders	×	

Rule	ExecEngine	SYSTEM
After a value update, all TTexts that used the value must be reset	×	
When a TText value is NOT used in an instance of (another) TText, then it may not say that its value is used to replace a placeholder	×	
Request TText instance phrase reset if the template phrase does, and the instance phrase does not exist	×	
Resetting a TText implies that the relations with other TTexts in which it is actually used for replacement, are broken	×	
Resetting a TText is complete if other TTexts are not used by this TText	×	

Concepts Scope, Scope, TTPhrase, TTName, TTValue, Actor, Claim, LegalThing, Moment, RechtsGrond, URL, Regeling, Bijlage, Aanwijzing, Artikel, Lid, Sub, BasisWettenBestand, Titel, Aantal, Domein, Regelingsoort, Concept, Procesflow, and Tekst remain without a purpose.

The relations $object[Claim * LegalThing]$, $ttTemplateParsedText[TText * TTPhrase]$, $ttTemplatePlaceholders[TText * TTName]$, $ttIsUsedBy[TText]$, $ttIsUsedByCopy[TText]$, $ttIsUsedByStar[TText]$, $url[RechtsGrond * URL]$, $regeling[RechtsGrond * Regeling]$, $bijlage[RechtsGrond * Bijlage]$, $aanwijzing[RechtsGrond * Aanwijzing]$, $artikel[RechtsGrond * Artikel]$, $lid[RechtsGrond * Lid]$, $sub[RechtsGrond * Sub]$, $bwb[Regeling * BasisWettenBestand]$, $titel[Regeling * Titel]$, $artikelen[Regeling * Aantal]$, $url[Regeling * URL]$, $gedelegeerdUit[Regeling]$, $regelgeving[Domein * Regeling]$, $soort[Regeling * Regelingsoort]$, $onderwerp[Regeling * Concept]$, $procesflow[Regeling * Procesflow]$, $afkorting[Regeling * Tekst]$, $class[TText * Concept]$, $legalGround[TText * LegalGround]$, $evidence[TText * Evidence]$, $autoLoginAccount[Account]$, $loginUserid[SESSION * UserID]$, $loginPassword[SESSION * Password]$, $logoutRequest[SESSION]$, $isoLevelGTE[ISOLevel]$, $sessionReqdISOLevel[SESSION * ISOLevel]$, $sessionAuthISOLevel[SESSION * ISOLevel]$, $moduleName[Module * ModuleName]$, $moduleVsnMajor[Module * ModuleVsnMajor]$, $moduleVsnMinor[Module * ModuleVsnMinor]$, $personOrg[Person * Organization]$, $uidUserid[UID * UserID]$, $uidIssuer[UID * IdP]$, $siamcompFirstName[SIAMPersonRefComputation * FirstName]$, $siamcompLastName[SIAMPersonRefComputation * LastName]$, $siamcompPersonRef[SIAMPerson$ $loginCreateAccount[SESSION]$, $registerPassword[SESSION * Password]$, $registerOrgRef[SESSION * OrgRef]$, $loginReq[SESSION]$, $loginISOLevel[SESSION * ISOLevel]$, $sessionDev[SESSION]$, $ttTrace[TText * Assignment]$, $asmVar[Assignment * TText]$, $asmVal[Assignment * TTValue]$, $asmPOT[Assignment * PointOfTime]$, $asmHasPred[Assignment]$, $compVar[Computation * TText]$, $compArg[Computation * Assignment]$, $compRes[Computation * TTValue]$, $sessionLoginAssist[SESSION]$, $sessionLogoutAssist[SESSION]$, $sessionSRI[SESSION]$, $sessionSIA[SESSION]$,

$scopeID[Scope * ScopeID]$, $scopeOwner[Scope * Account]$, $scopeDescr[Scope * ScopeDescr]$, $ttName[TText * TTName]$, $ttValue[TText * TTValue]$, $ttTemplate[TText * TTPhrase]$, $ttInstance[TText * TTPhrase]$, $ttICO[TText]$, $ttICCO[TText]$, $ttDescr[TText * TTPhrase]$, and $ttOwner[TText * Account]$ all lack both a purpose and a meaning.

The purpose of relations $owner[Scope * Actor]$, $created[Scope * Moment]$, $requires[TText]$, $ttValueUsedToReplacePlaceholders[TText * TTValue]$, $ttValueIsUsedInInstanceOfTText[TText]$, $ttScope[TText * Scope]$, $accUserid[Account * UserID]$, $accPassword[Account * Password]$, $sessionAccount[SESSION * Accoun$ $accIsInitialized[Account]$, $accIsActive[Account]$, $accDeactivateReq[Account]$, $sessionUserid[SESSION * UserID]$, $orgAbbrName[Organization * OrgAbbrName]$, $orgFullName[Organization * OrgFullName]$, $accOrg[Account * Organization]$, $sessionOrg[SESSION * Organization]$, $personFirstName[Person * FirstName]$, $personLastName[Person * LastName]$, $accPerson[Account * Person]$, $sessionPerson[SESSION * Person]$, $accAllowedRoles[Account * Role]$, $accDefaultRoles[Account * Role]$, $sessionAllowedRoles[SESSION * Role]$, $sessionActiveRoles[SESSION * Role]$, $accUID[Account * UID]$, $personRef[Person * PersonRef]$, $accPersonRef[Account * PersonRef]$, $sessionPersonRef[SESSION * PersonRef]$, $accOrgRef[Account * OrgRef]$, $sessionOrgRef[SESSION * OrgRef]$, $scopeIII[Scope]$, and $scopeIsaCC[Scope]$ is not documented.

Relations claimant, object, created, requires, ttIsUsedByStar, observed, url, regeling, bijlage, aanwijzing, artikel, lid, sub, bwb, titel, artikelen, url, gedelegeerdUit, regelgeving, soort, onderwerp, procesflow, and afkorting are not used in any rule.

fig. 3.1 shows a conceptual diagram with all relations declared in 'Arguments'.

fig. 3.2 shows a conceptual diagram with all relations declared in 'Placeholder Extraction and TText Hierarchy creation Service'.

fig. 3.3 shows a conceptual diagram with all relations declared in 'Validity'.

fig. 3.4 shows a conceptual diagram with all relations declared in 'Wetsartikelen'.

Rules are defined without documenting their purpose:

- *caseAuthor*

  line 155:5, file C:\Afstuderen\repos\ampersandmodels\MirrorMe\MirrorMe.adl

- *TText template parsing - extract placeholders*

  line 91:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *TText template parsing - delete parsed templates*

  line 103:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *TText template parsing - delete placeholders used in template*

  line 109:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *TTexts can only use TTexts that are in the same ttScope*

  line 122:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *Compute transitive closure of 'ttIsUsedBy'*

  line 129:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *Create TTexts for placeholders (if necessary)*

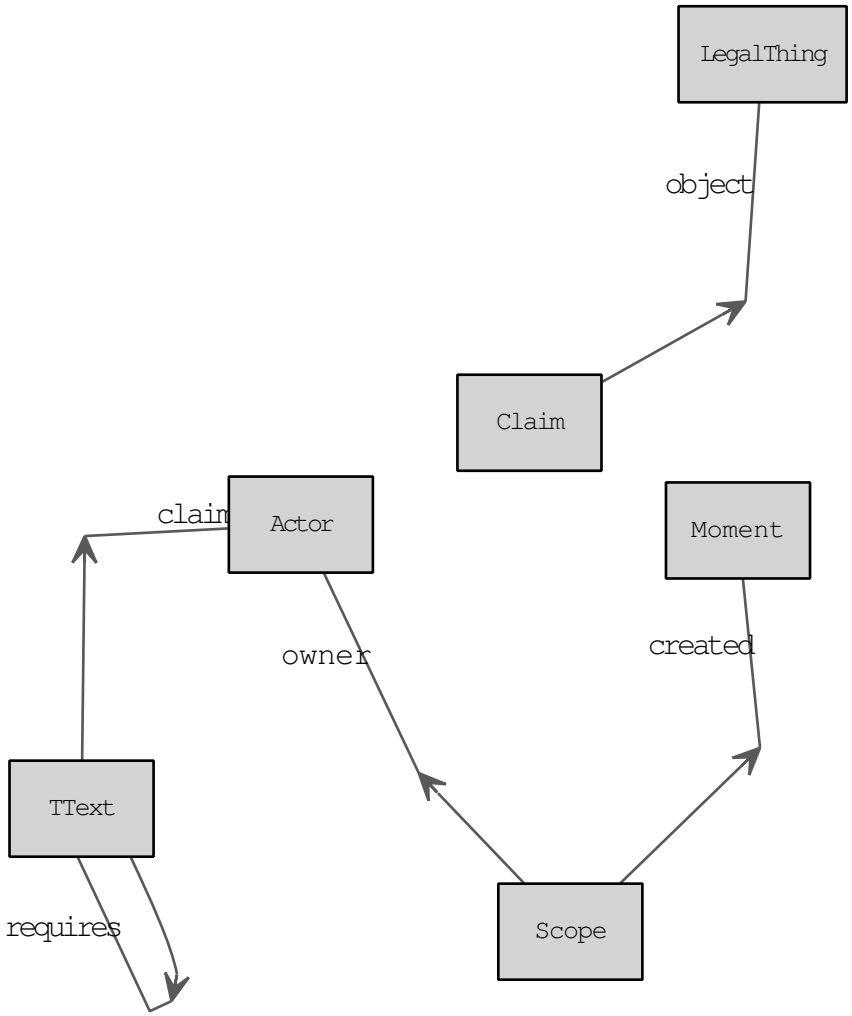

Figure 3.1: Concept diagram of relations in Arguments

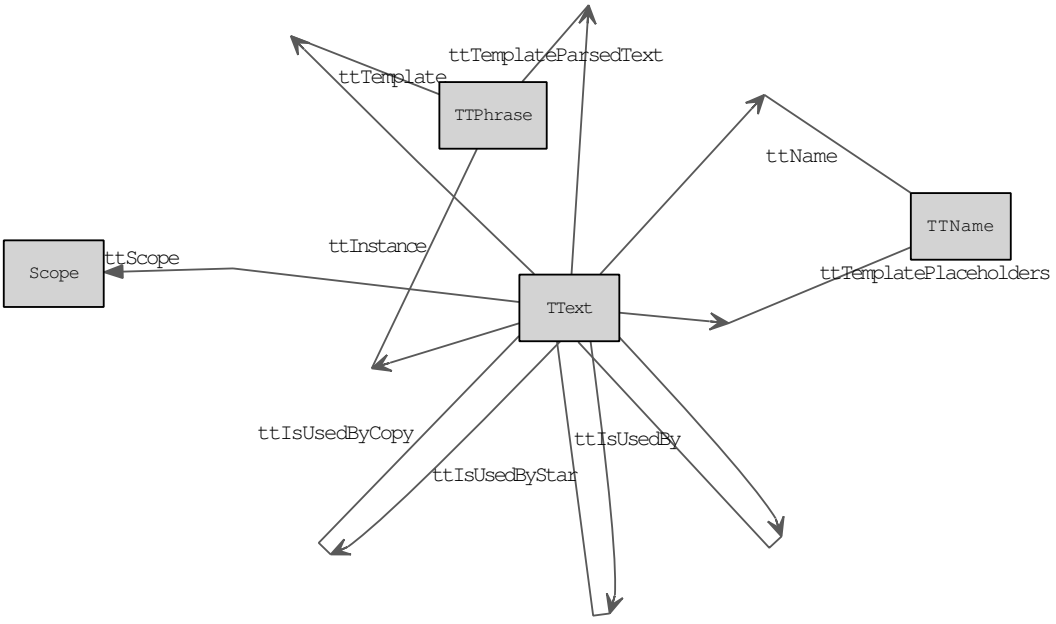

Figure 3.2: Concept diagram of relations in Placeholder Extraction and TText Hierarchy creation Service

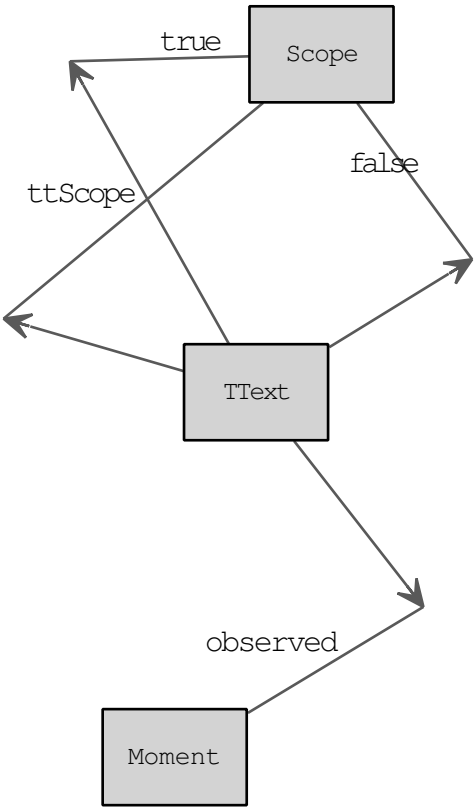

Figure 3.3: Concept diagram of relations in Validity

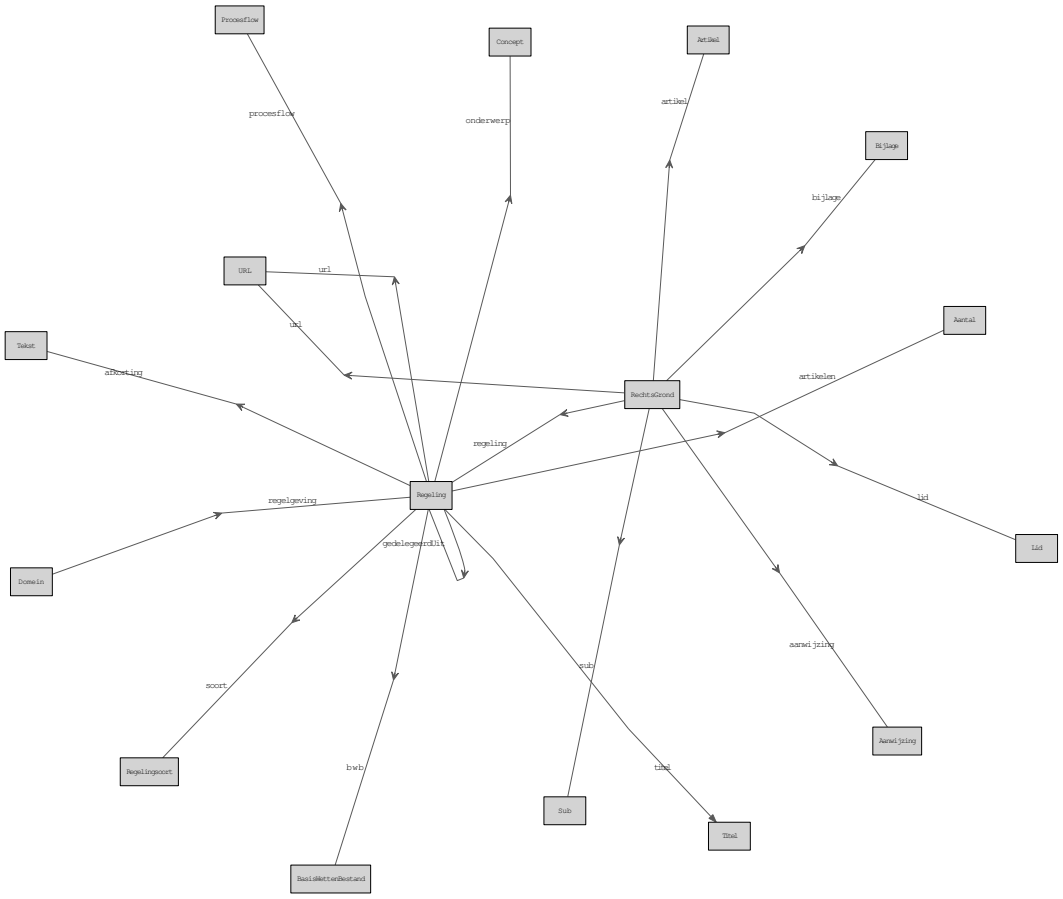

Figure 3.4: Concept diagram of relations in Wetsartikelen

line 133:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *When the name of a TText is in the list of placeholders of another TText, the first TText is said to be used by the second TText*

  line 140:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *A (first) TText is only used by a (second) TText if the name of the first is in the list of placeholders of the second*

  line 144:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *When a TText is used by another TText, it inherits the latter's Scope*

  line 150:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *ttValueUsedToReplacePlaceholders integrity*

  line 19:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcReplace.svc

- *Register that the value of a TText has been used to replace placeholders*

  line 22:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcReplace.svc

- *After a value update, all TTexts that used the value must be reset*

  line 38:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcReplace.svc

- *When a TText value is NOT used in an instance of (another) TText, then it may not say that its value is used to replace a placeholder*

  line 50:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcReplace.svc

- *ttValueIsUsedInInstanceOfTText integrity*

  line 31:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *Request TText instance phrase reset if the template phrase does, and the instance phrase does not exist*

  line 35:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *Resetting a TText implies that the relations with other TTexts in which it is actually used for replacement, are broken*

  line 41:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

Rules are defined, the meaning of which is documented by means of computer generated language:

- *caseAuthor*

  line 155:5, file C:\Afstuderen\repos\ampersandmodels\MirrorMe\MirrorMe.adl

- *TTexts can only use TTexts that are in the same ttScope*

  line 122:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *Compute transitive closure of 'ttIsUsedBy'*

  line 129:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *Create TTexts for placeholders (if necessary)*

  line 133:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *When the name of a TText is in the list of placeholders of another TText, the first TText is said to be used by the second TText*

  line 140:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *A (first) TText is only used by a (second) TText if the name of the first is in the list of placeholders of the second*

  line 144:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *When a TText is used by another TText, it inherits the latter's Scope*

  line 150:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

- *ttValueUsedToReplacePlaceholders integrity*

  line 19:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcReplace.svc

- *When a TText value is NOT used in an instance of (another) TText, then it may not say that its value is used to replace a placeholder*

  line 50:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcReplace.svc

- *ttValueIsUsedInInstanceOfTText integrity*

  line 31:1, file C:\Afstuderen\repos\ampersandmodels\TText\TText_PlcExtract.svc

The table below shows for each theme (i.e. process or pattern) the number of relations and rules, followed by the number and percentage that have a reference. Relations declared in multiple themes are counted multiple times.

Theme	Relations	With reference	%	Rules	Entire context	%
Arguments	5	1	20%	0	0	-
Case Management	0	0	-	1	0	0%
Placeholder Extraction and TText Hierarchy creation Service	5	0	0%	9	0	0%
Placeholder Replace-By-TTValue Service	1	0	0%	4	0	0%
TText Reset Service	2	0	0%	4	0	0%
Validity	4	0	0%	2	0	0%
Wetsartikelen	17	0	0%	0	0	-
Entire context	110	1	0%	99	0	0%

The table below shows the signal rules per role.

role	in pattern	rule
ExecEngine	Case Management	caseAuthor
ExecEngine	Placeholder Extraction and TText Hierarchy creation Service	TText template parsing - extract placeholders
ExecEngine	Placeholder Extraction and TText Hierarchy creation Service	TText template parsing - delete parsed templates

role	in pattern	rule
ExecEngine	Placeholder Extraction and TText Hierarchy creation Service	TText template parsing - delete placeholders used in template
ExecEngine	Placeholder Extraction and TText Hierarchy creation Service	Compute transitive closure of 'ttIsUsedBy'
ExecEngine	Placeholder Extraction and TText Hierarchy creation Service	Create TTexts for placeholders (if necessary)
ExecEngine	Placeholder Extraction and TText Hierarchy creation Service	When the name of a TText is in the list of placeholders of another TText, the first TText is said to be used by the second TText
ExecEngine	Placeholder Extraction and TText Hierarchy creation Service	A (first) TText is only used by a (second) TText if the name of the first is in the list of placeholders of the second
ExecEngine	Placeholder Extraction and TText Hierarchy creation Service	When a TText is used by another TText, it inherits the latter's Scope
ExecEngine	Placeholder Replace-By-TTValue Service	Register that the value of a TText has been used to replace placeholders
ExecEngine	Placeholder Replace-By-TTValue Service	After a value update, all TTexts that used the value must be reset
ExecEngine	Placeholder Replace-By-TTValue Service	When a TText value is NOT used in an instance of (another) TText, then it may not say that its value is used to replace a placeholder
ExecEngine	TText Reset Service	Request TText instance phrase reset if the template phrase does, and the instance phrase does not exist
ExecEngine	TText Reset Service	Resetting a TText implies that the relations with other TTexts in which it is actually used for replacement, are broken

role	in pattern	rule
ExecEngine	TText Reset Service	Resetting a TText is complete if other TTexts are not used by this TText
ExecEngine	MirrorMe	A TText that is not in a Scope may not exist
ExecEngine	MirrorMe	Account activation/initialization
ExecEngine	MirrorMe	Activate 'Developer' role
ExecEngine	MirrorMe	Activate roles in a session
ExecEngine	MirrorMe	Activation of session Organization
ExecEngine	MirrorMe	Active accounts are initialized
ExecEngine	MirrorMe	Add arguments to computations
ExecEngine	MirrorMe	Add to Assignments history
ExecEngine	MirrorMe	Assign computation result to TText (provided actual arguments were used)
ExecEngine	MirrorMe	Auto DelPair sessionSIA
ExecEngine	MirrorMe	Auto activate auto-login accounts
ExecEngine	MirrorMe	Auto create SIAMPersonRefComputation
ExecEngine	MirrorMe	Auto maintain 'accPersonRef' relation
ExecEngine	MirrorMe	Auto maintain 'personRef' relation
ExecEngine	MirrorMe	Automatically depopulate personOrg
ExecEngine	MirrorMe	Automatically login
ExecEngine	MirrorMe	Automatically populate personOrg
ExecEngine	MirrorMe	By default, a session authentication level is '1'
ExecEngine	MirrorMe	Clear allowed session roles
ExecEngine	MirrorMe	Clear the OrgRef in a session
ExecEngine	MirrorMe	Create Account upon request
ExecEngine	MirrorMe	Create Assignment for TTexts that have a (new) value

role	in pattern	rule
ExecEngine	MirrorMe	Create Assignment for TTexts that have no value
ExecEngine	MirrorMe	Create computations
ExecEngine	MirrorMe	Create/Update copied ttDescr
ExecEngine	MirrorMe	Create/Update copied ttOwner
ExecEngine	MirrorMe	Deactivate 'Developer' role
ExecEngine	MirrorMe	Deactivate roles in a session
ExecEngine	MirrorMe	Deactivation of session Organization
ExecEngine	MirrorMe	Default roles must be allowed roles
ExecEngine	MirrorMe	Delete computations (that have results) of which one or more arguments have changed
ExecEngine	MirrorMe	Delete obsolete Assignments
ExecEngine	MirrorMe	Delete obsolete Computations
ExecEngine	MirrorMe	Determine the 'userid' in a session
ExecEngine	MirrorMe	Determine the OrgRef in a session
ExecEngine	MirrorMe	Determine the acting person in a session
ExecEngine	MirrorMe	Determine the personRef in a session
ExecEngine	MirrorMe	Disable automated login
ExecEngine	MirrorMe	Every TText with scope and name can be used as a binding
ExecEngine	MirrorMe	Initialize copied ttDescr
ExecEngine	MirrorMe	Initialize copied ttOwner
ExecEngine	MirrorMe	InsPair sessionAuthISOLevel (MPTrx specific)
ExecEngine	MirrorMe	Limit history size to 2 predecessors
ExecEngine	MirrorMe	Logout
ExecEngine	MirrorMe	Registering the first authenticated session account

role	in pattern	rule
ExecEngine	MirrorMe	Reset login help
ExecEngine	MirrorMe	SURttName
ExecEngine	MirrorMe	SURttPhrase
ExecEngine	MirrorMe	Service U/PW Login request
ExecEngine	MirrorMe	Service deactivation request
ExecEngine	MirrorMe	Sessions with an inactive sessionaccount may not exist
ExecEngine	MirrorMe	Set allowed session roles
ExecEngine	MirrorMe	Set default Scope ownership
ExecEngine	MirrorMe	Set default TText ownership
ExecEngine	MirrorMe	The userid of a customer is its (initial) personref
ExecEngine	MirrorMe	U/PW Login
ExecEngine	MirrorMe	Undetermine the 'userid' in a session
ExecEngine	MirrorMe	Undetermine the acting person in a session
ExecEngine	MirrorMe	Undetermine the personRef in a session
ExecEngine	MirrorMe	Unowned TTexts that are used by an owned TText, will be assigned the same owner
ExecEngine	MirrorMe	Update arguments of computations that do not have a result
ExecEngine	MirrorMe	Update deleted ttDescr
ExecEngine	MirrorMe	Update deleted ttOwner

The population in this script does not specify any work in progress.

The population in this script violates no rule.

Chapter 4

Conceptual Analysis

This chapter defines the formal language, in which functional requirements of 'MirrorMe' can be analysed and expressed.The purpose of this formalisation is to obtain a buildable specification. This chapter allows an independent professional with sufficient background to check whether the agreements made correspond to the formal rules and definitions.

4.1 Theme: 'Arguments'

The structure of legal arguments has been studied by scholars such as Toulmin~[**?**], Verheij (name some more). To help legal professionals construct such arguments, the structure of arguments must be formalized. For this purpose, we can consider each (legal) case as a set of statements. Each statement can be considered true or

fig. 4.1 shows a conceptual diagram of this pattern.

4.2 Theme: 'Case Management'

This pattern contains some basic administration for cases.

fig. 4.2 shows a conceptual diagram of this pattern.

4.3 Theme: 'Placeholder Extraction and TText Hierarchy creation Service'

This service ensures that - `ttTemplateParsedText`=`ttTemplate` AND - `ttTemplatePlaceholders` contains all `TTName`s that are mentioned in the `ttTemplate` of a `TText`.

The idea is that the specification of a TText is parsed to see if it contains names of `TText`s. Such names are recognized by the fact that they are surrounded by square brackets (`[` and `]`).

So, - `ttTemplateParsedText` stores the text that has been parsed (in PHP) to produce the contents of `ttTemplatePlaceholders`. This implies that whenever

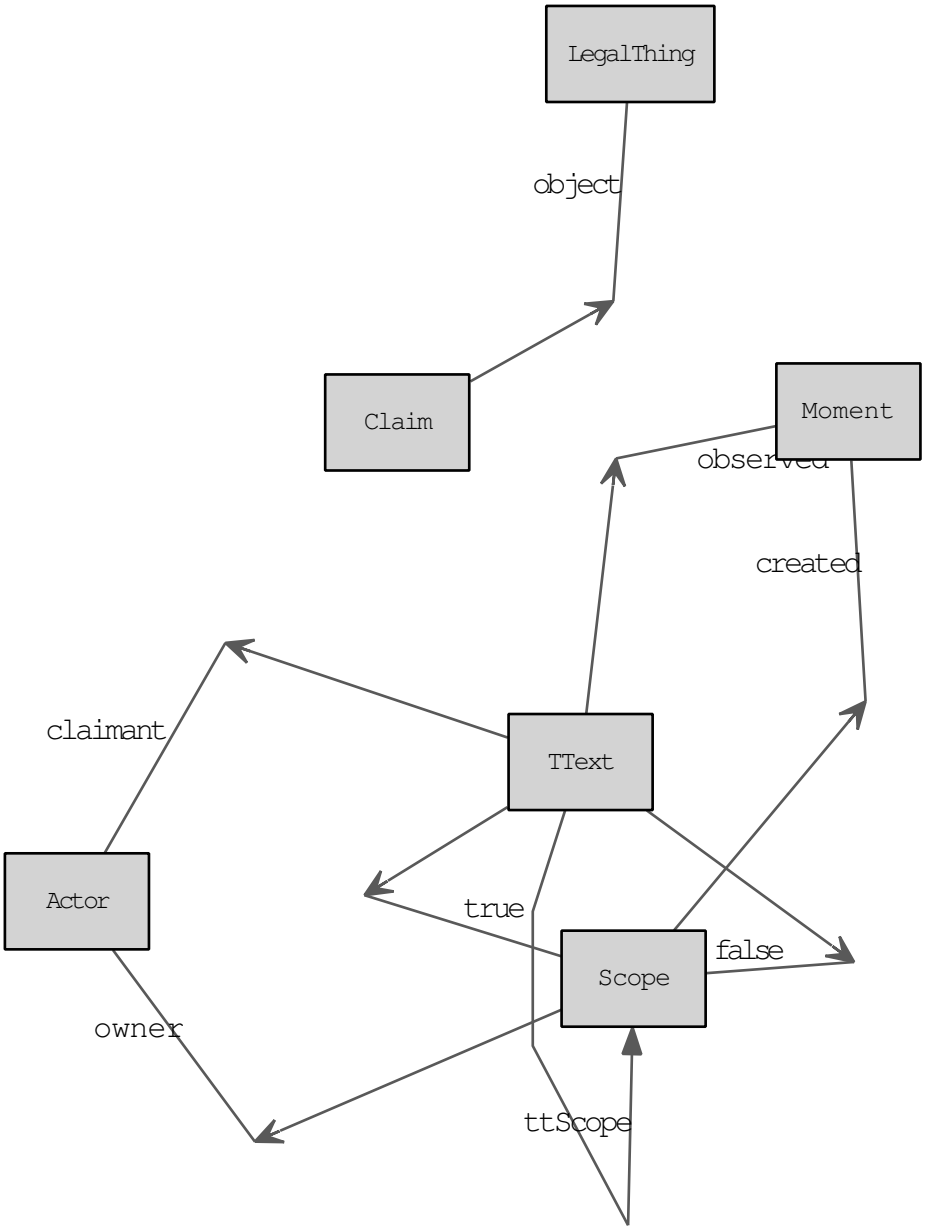

Figure 4.1: Concept diagram of the rules in Arguments

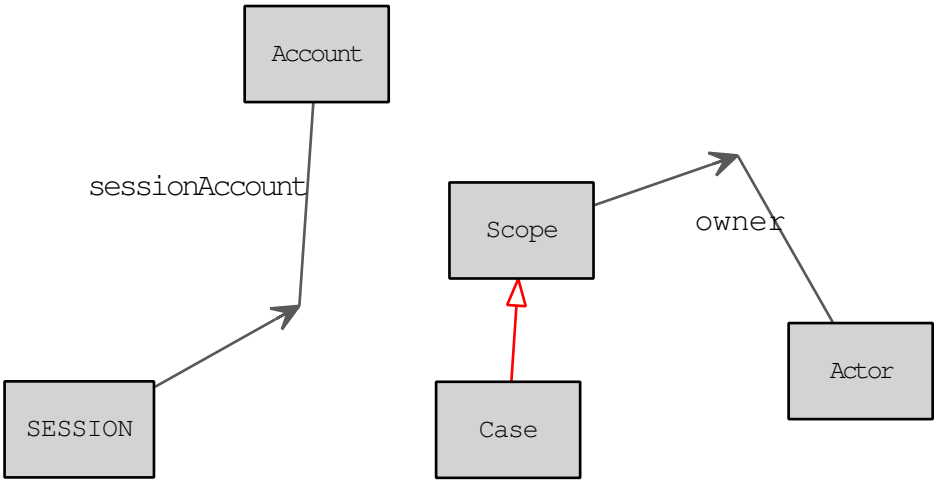

Figure 4.2: Concept diagram of the rules in Case Management

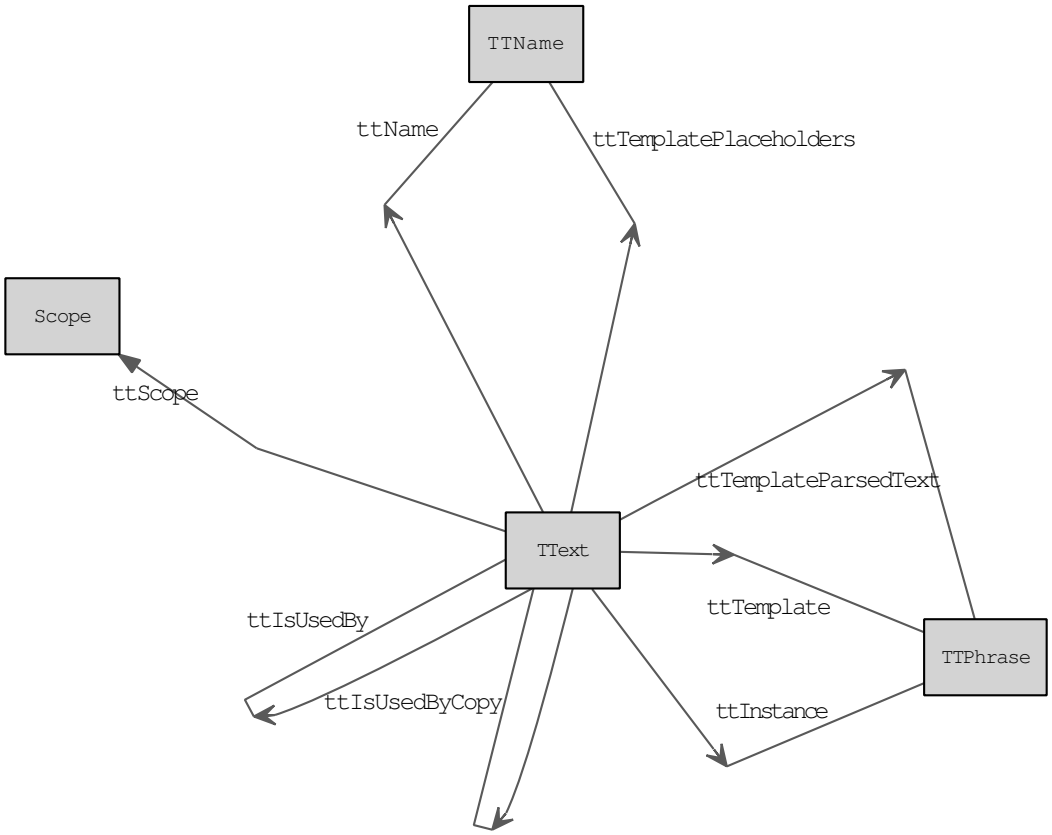

Figure 4.3: Concept diagram of the rules in Placeholder Extraction and TText Hierarchy creation Service

`ttTemplateParsedText` is empty for some `TText`, `ttTemplatePlaceholders` is empty for that same atom. - whenever `ttTemplateParsedText` differs from `ttTemplate`, - it is first removed, and `ttTemplatePlaceholders` are discarded - then `ttTemplate` is being parsed, and `ttTemplateParsedText` and `ttTemplatePlaceholders` are filled again.

All of this happens in the same `Scope`, i.e. in the scope to which the `TTexts` belong. Note that for this to work, all `TTexts` that are mentioned in a `ttTemplate` must exist.

fig. 4.3 shows a conceptual diagram of this pattern.

4.3.1 Rules

This section itemizes the rules with a reference to the shared language of stakeholders for the sake of traceability.

agreement **¿lst:SharedLangrule:TTexts_32can_32only_32use_32TTexts_32that_32are_32i**
has been made:
Using relations relation **¿eq:ConceptualAnalysisdeclaration:ttIsUsedBy_91TText_42TText**
(ttIsUsedBy), relation **¿eq:ConceptualAnalysisdeclaration:ttScope_91TText_42Scope_93?**
(ttScope), this is formalized as
an undocumented agreement: $\text{ttIsUsedBy} \vdash \text{ttScope}; \text{ttScope}^{\smile}$
fig. 4.4 shows a conceptual diagram of this rule.

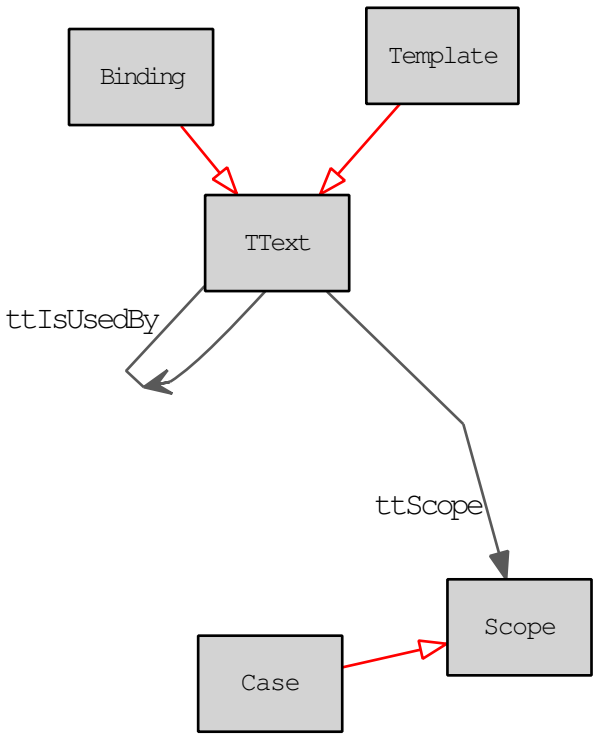

Figure 4.4: Concept diagram of rule TTexts can only use TTexts that are in the same ttScope

4.4 Theme: 'Placeholder Replace-By-TTValue Service'

This service ensures that the `ttInstance` phrase of a TText is the `ttTemplate` phrase of that TText, where every occurrence of a reference to another TText in that template (i.e. occurences of `[<TTName>]`, called 'placeholder's) **is replaced by the `ttValue`** of that other TText, provided that the `ttValue` is populated for that TText.

fig. 4.5 shows a conceptual diagram of this pattern.

4.4.1 Rules

This section itemizes the rules with a reference to the shared language of stakeholders for the sake of traceability.

agreement **¿lst:SharedLangrule:ttValueUsedToReplacePlaceholders_32integrity?**
has been made:
Using relations relation **¿eq:ConceptualAnalysisdeclaration:ttValueUsedToReplacePlacehol**
(ttValueUsedToReplacePlaceholders), relation **¿eq:ConceptualAnalysisdeclaration:ttValue_91T**
(ttValue), this is formalized as
an undocumented agreement: ttValueUsedToReplacePlaceholders ⊢ ttValue
fig. 4.6 shows a conceptual diagram of this rule.

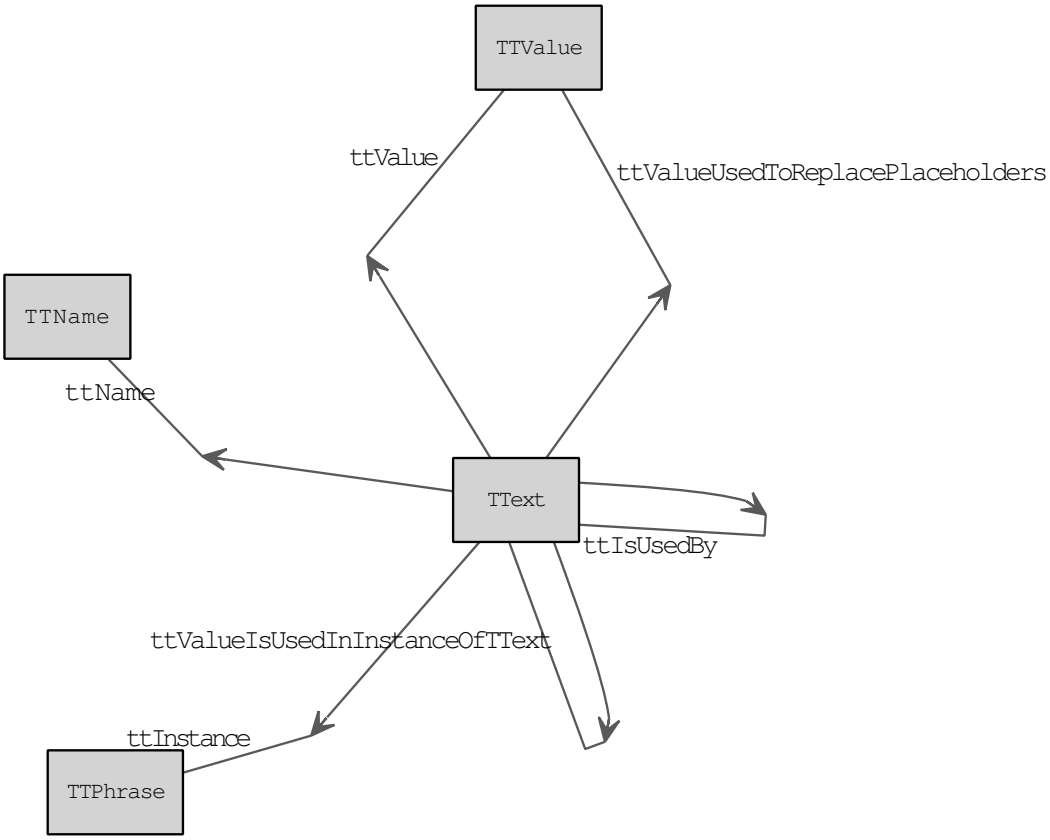

Figure 4.5: Concept diagram of the rules in Placeholder Replace-By-TTValue Service

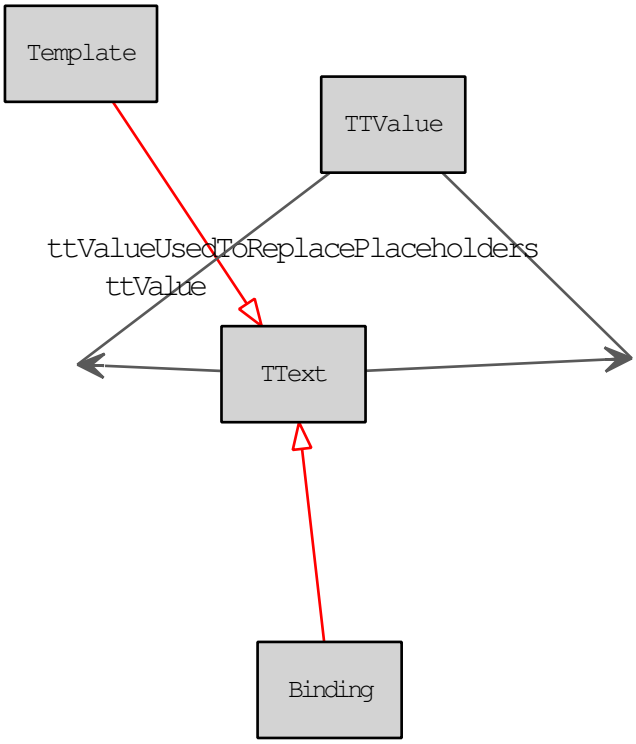

Figure 4.6: Concept diagram of rule ttValueUsedToReplacePlaceholders integrity

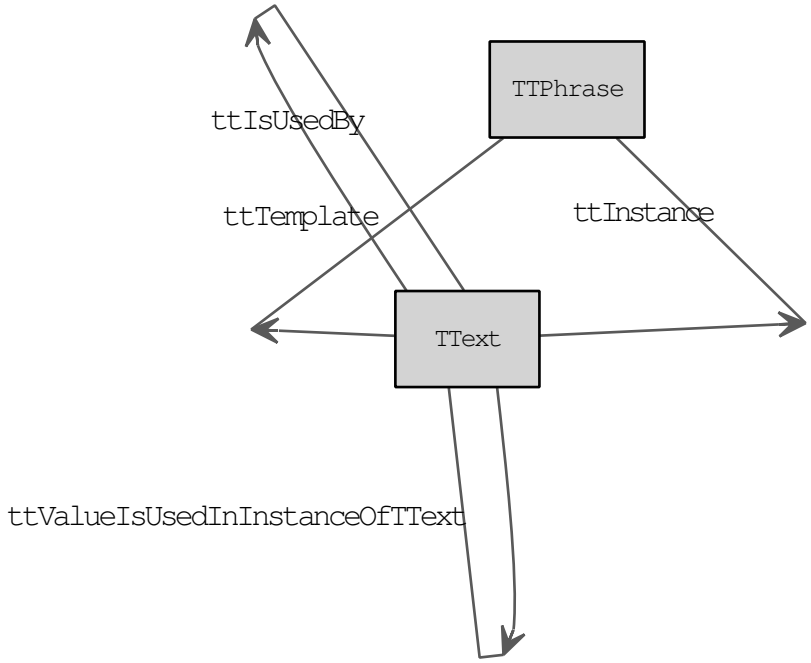

Figure 4.7: Concept diagram of the rules in TText Reset Service

4.5 Theme: 'TText Reset Service'

This service relinquishes any depencies that the ttInstance of a TText has on other TTexts (as registered in `ttValueIsUsedInInstanceOfTText`). This condition will be realized by setting the property `ttInstanceResetReq` for that TText.

fig. 4.7 shows a conceptual diagram of this pattern.

4.5.1 Rules

This section itemizes the rules with a reference to the shared language of stakeholders for the sake of traceability.

agreement **¿lst:SharedLangrule:ttValueIsUsedInInstanceOfTText_32integrity?**
has been made:
Using relations relation **¿eq:ConceptualAnalysisdeclaration:ttValueIsUsedInInstanceOfTTe**
(ttValueIsUsedInInstanceOfTText), relation **¿eq:ConceptualAnalysisdeclaration:ttIsUsedBy_9**
(ttIsUsedBy), this is formalized as
an undocumented agreement: ttValueIsUsedInInstanceOfTText $\vdash$ ttIsUsedBy
fig. 4.8 shows a conceptual diagram of this rule.

4.6 Theme: 'Validity'

In order to talk about true and false statements in a precise way, we need the idea of contexts.

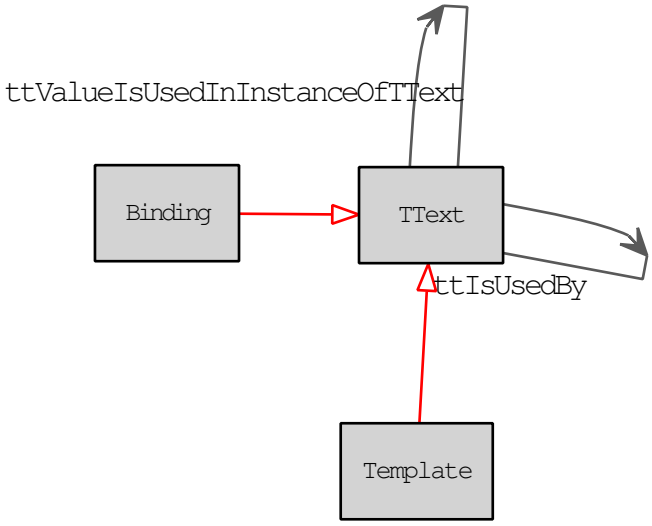

Figure 4.8: Concept diagram of rule ttValueIsUsedInInstanceOfTText integrity

fig. 4.9 shows a conceptual diagram of this pattern.

4.6.1 Rules

This section itemizes the rules with a reference to the shared language of stakeholders for the sake of traceability.

Every statement (TText) that is in scope is considered valid. This means that it can be true or false, given that all its placeholders have been instantiated.

Therefore agreement **¿lst:SharedLangrule:TrueFalseValid?** exists:

A statement can be true or false in a context only if it is valid within that context.

Using relations relation **¿eq:ConceptualAnalysisdeclaration:true_91TText_42Scope_93?** (true), relation **¿eq:ConceptualAnalysisdeclaration:false_91TText_42Scope_93?** (false), relation **¿eq:ConceptualAnalysisdeclaration:ttScope_91TText_42Scope_93?** (ttScope), this is formalized as

Rule 21: : $\text{true} \cup \text{false} \vdash \text{ttScope}$

fig. 4.10 shows a conceptual diagram of this rule.

Every statement (TText) that is in scope is considered valid. This means that it can be true or false, if all its placeholders have been instantiated.

Therefore agreement **¿lst:SharedLangrule:Inconsistency?** exists:

A statement cannot be both true and false in the same context.

Using relations relation **¿eq:ConceptualAnalysisdeclaration:true_91TText_42Scope_93?** (true), relation **¿eq:ConceptualAnalysisdeclaration:false_91TText_42Scope_93?** (false), this is formalized as

Rule 22: : $\text{true} \cap \text{false} \vdash \overline{V_{[\text{TText}*\text{Scope}]}}$

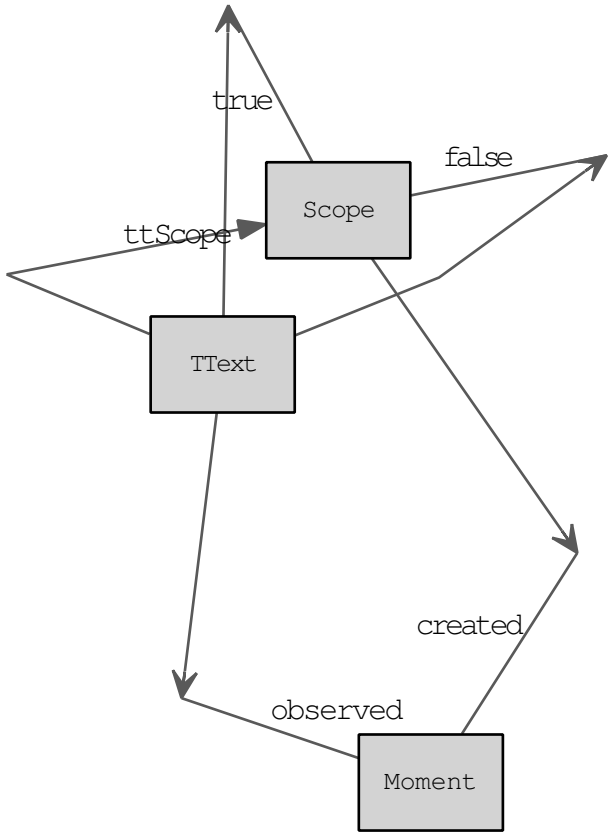

Figure 4.9: Concept diagram of the rules in Validity

fig. 4.11 shows a conceptual diagram of this rule.

4.7 Theme: 'Wetsartikelen'

fig. 4.12 shows a conceptual diagram of this pattern.

The definitions of concepts can be found in the glossary.

4.7.1 Declared relations

This section itemizes the declared relations with properties and purpose.

Let us treat a statement made by an individual as a claim of that individual that the statement is true. For this reason, each individual who makes the claim can be registered in the relation "claimant". The `claimant` 'claims' (a) the validity of the `ttTemplate` and (b) the veracity of `ttValue`. Thus, the `claimaint` is solely authorized to change `ttValue` and `ttTemplate`.

http://www.thefreedictionar

For this purpose, the following relation has been defined

Relation 1: : claimant[TText $*$ Actor]

A claimant is the party that makes a claim, especially one that is legally cognizable.

The following univalent relation has been defined

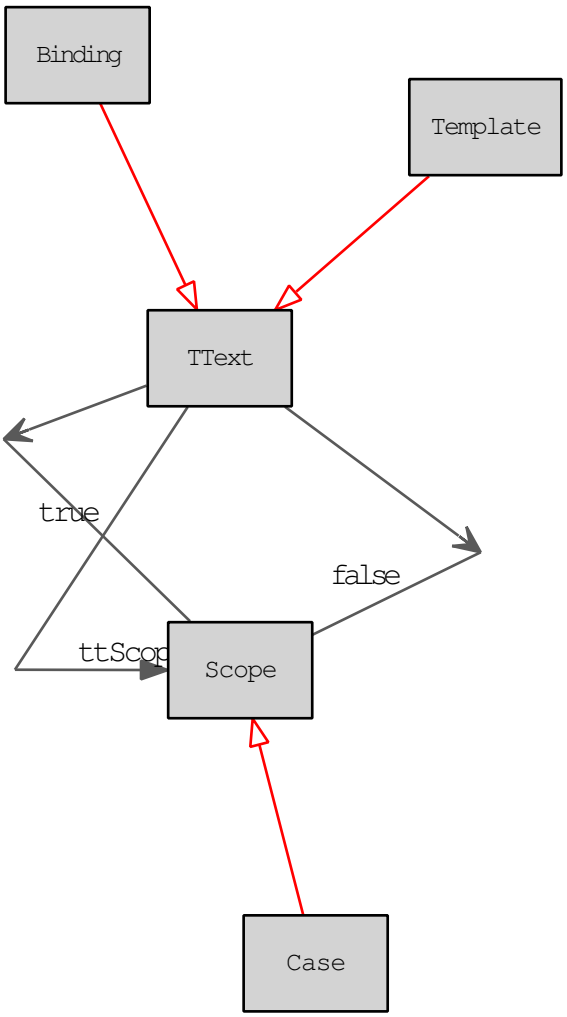

Figure 4.10: Concept diagram of rule TrueFalseValid

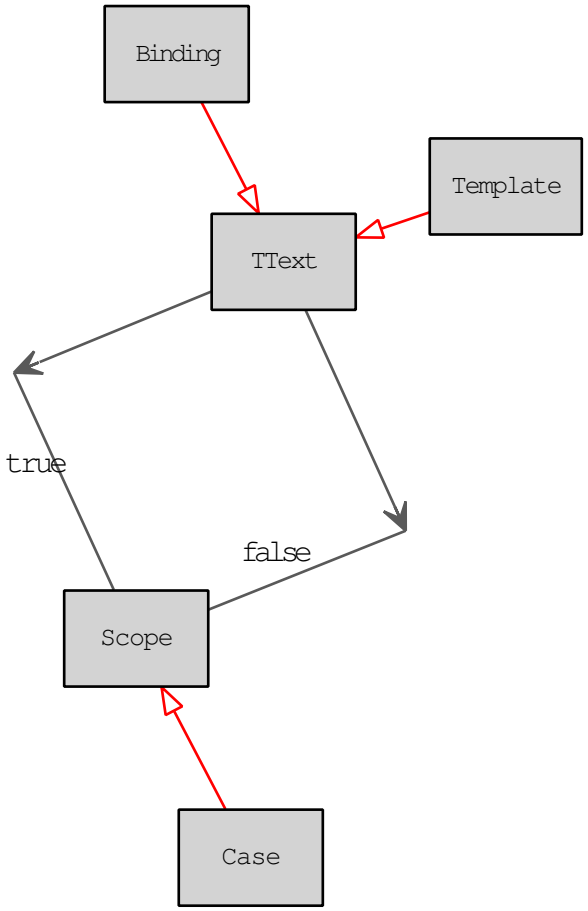

Figure 4.11: Concept diagram of rule Inconsistency

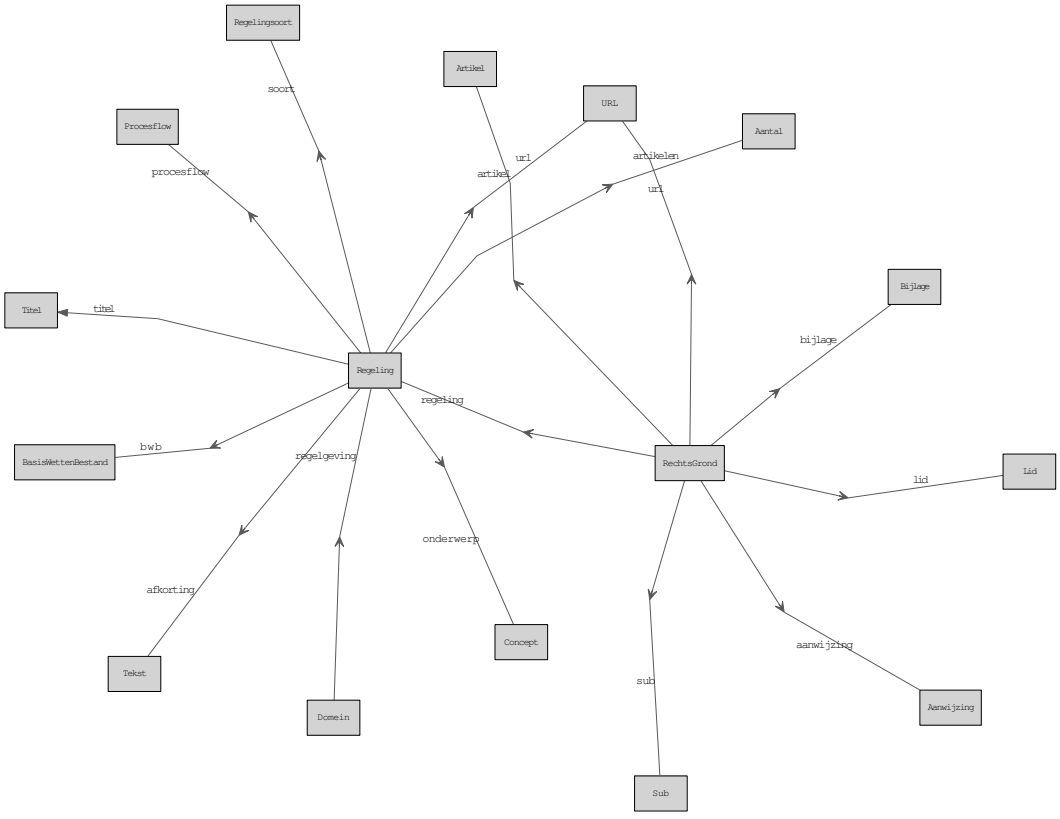

Figure 4.12: Concept diagram of the rules in Wetsartikelen

an undocumented relation: object[Claim ∗ LegalThing]

(No meaning has been specified)

The following univalent relation has been defined

Relation 2: : owner[Scope ∗ Actor]

the Actor that is the claimant of all toplevel TTexts.

The following univalent relation has been defined

Relation 3: : created[Scope ∗ Moment]

The application can register a moment as the time a case has been created.

The following relation has been defined

Relation 4: : requires[TText ∗ TText]

A statement that needs another statement to be true, is registered in the relation `requires`.

The following univalent relation has been defined

an undocumented relation: ttTemplateParsedText[TText ∗ TTPhrase]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: ttTemplatePlaceholders[TText ∗ TTName]

(No meaning has been specified)

The following irreflexive and antisymmetric relation has been defined

an undocumented relation: ttIsUsedBy[TText ∗ TText]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: ttIsUsedByCopy[TText ∗ TText]

(No meaning has been specified)

The following irreflexive, antisymmetric, and transitive relation has been defined

an undocumented relation: ttIsUsedByStar[TText ∗ TText]

(No meaning has been specified)

The following univalent relation has been defined

Relation 12: : ttValueUsedToReplacePlaceholders[TText ∗ TTValue]

References to this TText (i.e.: placeholders) exist, and have been replaced with the specified TTValue

The property ttInstanceResetReq is used to start reconstructing an instance phrase of a TText.

For this purpose, the following symmetric, antisymmetric, univalent, and injective relation has been defined

Relation 10: : ttInstanceResetReq[TText ∗ TText]

If a TText has the property ttInstanceResetReq, this means that its instance phrase needs to be (re)constructed from scratch.

The following irreflexive and antisymmetric relation has been defined

Relation 11: : ttValueIsUsedInInstanceOfTText[TText ∗ TText]

SRC TText has been used to replace placeholders in the TGT TText instance phrase

Consider the statement "John has blond hair". If this statement is known to be true in some context `c`, the tuple ("John has blond hair", c) can exist in the relation `true`. However, if this tuple is not in the relation `true`, it does not follow that John does not have blond hair. The truth of the TText value (as decided by its claimant) is accepted by the owner of the Scope

For this purpose, the following univalent relation has been defined

Relation 18: : true[TText ∗ Scope]

A statement that is considered true within a context is registered in the relation `true`.

The relation `false` is dual to `true`.

For this purpose, the following univalent relation has been defined

Relation 19: : false[TText ∗ Scope]

A statement that is considered false within a context is registered in the relation `false`.

The following function has been defined

Relation 6: : ttScope[TText ∗ Scope]

Statements that are in a given scope are considered valid within that scope.

For reconstructing events, it can be necessary to administer the moment an observation or a claim is made. For this reason, we use the relation "observed".

For this purpose, the following univalent relation has been defined

Relation 20: : observed[TText ∗ Moment]

The application can register a moment as the time a statement has been made.

The following relation has been defined

an undocumented relation: url[RechtsGrond ∗ URL]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: regeling[RechtsGrond ∗ Regeling]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: bijlage[RechtsGrond ∗ Bijlage]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: aanwijzing[RechtsGrond ∗ Aanwijzing]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: artikel[RechtsGrond ∗ Artikel]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: lid[RechtsGrond ∗ Lid]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: sub[RechtsGrond ∗ Sub]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: bwb[Regeling ∗ BasisWettenBestand]

(No meaning has been specified)

The following function has been defined

an undocumented relation: titel[Regeling ∗ Titel]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: artikelen[Regeling ∗ Aantal]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: url[Regeling ∗ URL]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: gedelegeerdUit[Regeling ∗ Regeling]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: regelgeving[Domein ∗ Regeling]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: soort[Regeling ∗ Regelingsoort]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: onderwerp[Regeling ∗ Concept]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: procesflow[Regeling ∗ Procesflow]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: afkorting[Regeling ∗ Tekst]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: class[TText ∗ Concept]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: legalGround[TText ∗ LegalGround]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: evidence[TText ∗ Evidence]

(No meaning has been specified)

The following symmetric, antisymmetric, univalent, and injective relation has been defined

an undocumented relation: autoLoginAccount[Account ∗ Account]

(No meaning has been specified)

The following injective function has been defined

Relation 23: : accUserid[Account ∗ UserID]

An Account registers a pseudonym for whatever the Account applies to

The following univalent relation has been defined

Relation 24: : accPassword[Account ∗ Password]

An Account registers a password for whatever the Account applies to

The following univalent relation has been defined

Relation 5: : sessionAccount[SESSION ∗ Account]

A SESSION may be linked to an Account, which specifies the (user)context of the session.

The following symmetric, antisymmetric, univalent, and injective relation has been defined

Relation 25: : accIsInitialized[Account ∗ Account]

An account may have the property that it is initialized (e.g. some default values have been set)

The following symmetric, antisymmetric, injective, and surjective function has been defined

Relation 26: : accIsActive[Account ∗ Account]

An Account may have the property that it is active, meaning that it can be used to login

The following symmetric, antisymmetric, univalent, and injective relation has been defined

Relation 27: : accDeactivateReq[Account ∗ Account]

A request may exist to deactivate the account (as soon as it is no longer a sessionAccount)

The following univalent relation has been defined

Relation 28: : sessionUserid[SESSION ∗ UserID]

In a SESSION, the userid (i.e. the name/text that identifies the Account that is used to login) may be known.

The following univalent relation has been defined

an undocumented relation: loginUserid[SESSION ∗ UserID]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: loginPassword[SESSION ∗ Password]

(No meaning has been specified)

The following symmetric, antisymmetric, univalent, and injective relation has been defined

an undocumented relation: logoutRequest[SESSION ∗ SESSION]

(No meaning has been specified)

The following reflexive, antisymmetric, and transitive relation has been defined

an undocumented relation: isoLevelGTE[ISOLevel ∗ ISOLevel]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: sessionReqdISOLevel[SESSION ∗ ISOLevel]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: sessionAuthISOLevel[SESSION ∗ ISOLevel]

(No meaning has been specified)

The following function has been defined

an undocumented relation: moduleName[Module ∗ ModuleName]

(No meaning has been specified)

The following function has been defined

an undocumented relation: moduleVsnMajor[Module∗ModuleVsnMajor]

(No meaning has been specified)

The following function has been defined

an undocumented relation: moduleVsnMinor[Module∗ModuleVsnMinor]

(No meaning has been specified)

The following injective and univalent relation has been defined

Relation 29: : orgAbbrName[Organization ∗ OrgAbbrName]

An organization may have a short name by which it can be identified

The following injective and univalent relation has been defined

Relation 30: : orgFullName[Organization ∗ OrgFullName]

An organization has a long (full) name by which it may be identified

The following univalent relation has been defined

Relation 31: : accOrg[Account ∗ Organization]

An Account may contain (web)service-specific attributes regarding a specific Organization

The following univalent relation has been defined

Relation 32: : sessionOrg[SESSION ∗ Organization]

In a session, the (accountable) Organization may be known.

The following relation has been defined

an undocumented relation: personOrg[Person ∗ Organization]

(No meaning has been specified)

The following function has been defined

Relation 33: : personFirstName[Person ∗ FirstName]

The first name of a Person that is registered.

The following function has been defined

Relation 34: : personLastName[Person ∗ LastName]

The last name of a Person that is registered.

The following univalent relation has been defined

Relation 35: : accPerson[Account ∗ Person]

An Account may contain (web)service-specific attributes regarding a specific Person

The following univalent relation has been defined

Relation 36: : sessionPerson[SESSION ∗ Person]

In a SESSION, the user (i.e. the Person that acts 'at the other side') may be known.

The following relation has been defined

Relation 37: : accAllowedRoles[Account ∗ Role]

An Account registers the Roles that MAY be activated in a SESSION to which the Account is assigned

The following relation has been defined

Relation 38: : accDefaultRoles[Account ∗ Role]

An Account registers the Roles that ARE activated in a SESSION once the Account is assigned

The following relation has been defined

Relation 39: : sessionAllowedRoles[SESSION ∗ Role]

Within a SESSION, a Role may be activated

The following relation has been defined

Relation 40: : sessionActiveRoles[SESSION ∗ Role]

A SESSION has activated a Role

In order to enable the suspension of sessions, the first sessionaccount must be remembered since it must be used to verify that the user that logs in to a suspended session in order to reactivate it, is the same as the user that did the original login.

For this purpose, the following univalent relation has been defined

Relation 41: : firstSessionAccount[SESSION ∗ Account]

The Account that is authenticated first in a session, must be remembered.

The following function has been defined

an undocumented relation: uidUserid[UID ∗ UserID]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: uidIssuer[UID ∗ IdP]

(No meaning has been specified)

The following injective relation has been defined

Relation 42: : accUID[Account ∗ UID]

An Account registers a UID for whatever the Account applies to

The following univalent relation has been defined

Relation 43: : personRef[Person ∗ PersonRef]

A textstring to refer to this person

The following univalent relation has been defined

Relation 44: : accPersonRef[Account ∗ PersonRef]

An Account may have a textstring that refers to the Person for which the account has attributes

The following univalent relation has been defined

Relation 45: : sessionPersonRef[SESSION ∗ PersonRef]

In a SESSION, a textstring that refers to the user (i.e. the Person that acts 'at the other side') may be known.

The following univalent relation has been defined

an undocumented relation: siamcompFirstName[SIAMPersonRefComputation∗ FirstName]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: siamcompLastName[SIAMPersonRefComputation∗ LastName]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: siamcompPersonRef[SIAMPersonRefComputation∗ PersonRef]

(No meaning has been specified)

The following univalent relation has been defined

Relation 46: : accOrgRef[Account ∗ OrgRef]

An Account may have a textstring that refers to the Organization to which the Account is related

The following univalent relation has been defined

Relation 47: : sessionOrgRef[SESSION ∗ OrgRef]

In a SESSION, a textstring that refers to the Organization ('at the other side') may be known.

The following symmetric, antisymmetric, univalent, and injective relation has been defined

an undocumented relation: loginCreateAccount[SESSION ∗ SESSION]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: registerPassword[SESSION ∗ Password]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: registerOrgRef[SESSION ∗ OrgRef]

(No meaning has been specified)

The following symmetric, antisymmetric, univalent, and injective relation has been defined

an undocumented relation: loginReq[SESSION ∗ SESSION]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: loginISOLevel[SESSION ∗ ISOLevel]

(No meaning has been specified)

The following symmetric, antisymmetric, univalent, and injective relation has been defined

an undocumented relation: sessionDev[SESSION ∗ SESSION]

(No meaning has been specified)

The following injective function has been defined

an undocumented relation: ttTrace[TText ∗ Assignment]

(No meaning has been specified)

The following function has been defined

an undocumented relation: asmVar[Assignment ∗ TText]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: asmVal[Assignment ∗ TTValue]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: asmPOT[Assignment ∗ PointOfTime]

(No meaning has been specified)

The following univalent and injective relation has been defined

an undocumented relation: asmHasPred[Assignment ∗ Assignment]

(No meaning has been specified)

The following injective function has been defined

an undocumented relation: compVar[Computation ∗ TText]

(No meaning has been specified)

The following relation has been defined

an undocumented relation: compArg[Computation ∗ Assignment]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: compRes[Computation ∗ TTValue]

(No meaning has been specified)

The following symmetric, antisymmetric, univalent, and injective relation has been defined

an undocumented relation: sessionLoginAssist[SESSION∗SESSION]

(No meaning has been specified)

The following symmetric, antisymmetric, univalent, and injective relation has been defined

an undocumented relation: sessionLogoutAssist[SESSION ∗ SESSION]

(No meaning has been specified)

The following symmetric, antisymmetric, univalent, and injective relation has been defined

an undocumented relation: sessionSRI[SESSION ∗ SESSION]

(No meaning has been specified)

The following symmetric, antisymmetric, univalent, and injective relation has been defined

an undocumented relation: sessionSIA[SESSION ∗ SESSION]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: scopeID[Scope ∗ ScopeID]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: scopeOwner[Scope ∗ Account]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: scopeDescr[Scope ∗ ScopeDescr]

(No meaning has been specified)

The following irreflexive and antisymmetric relation has been defined

Relation 48: : scopeIII[Scope ∗ Scope]

The structured content of a (SRC) Scope may be included in that of a (TGT) Scope

The following symmetric, antisymmetric, univalent, and injective relation has been defined

Relation 49: : scopeIsaCC[Scope ∗ Scope]

A Scope may have the property of being a carbon copy of another Scope

The following univalent relation has been defined

an undocumented relation: ttName[TText ∗ TTName]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: ttValue[TText ∗ TTValue]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: ttTemplate[TText ∗ TTPhrase]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: ttInstance[TText ∗ TTPhrase]

(No meaning has been specified)

The following univalent, irreflexive, and antisymmetric relation has been defined

an undocumented relation: ttICO[TText ∗ TText]

(No meaning has been specified)

The following univalent, irreflexive, and antisymmetric relation has been defined

an undocumented relation: ttICCO[TText ∗ TText]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: ttDescr[TText ∗ TTPhrase]

(No meaning has been specified)

The following univalent relation has been defined

an undocumented relation: ttOwner[TText ∗ Account]

(No meaning has been specified)

Chapter 5

Data structure

This chapter contains the result of the data analysis. It is structured as follows:

We start with the classification model, followed by a list of all relations, that are the foundation of the rest of the analysis. Finally, the logical and technical data model are discussed.

5.1 Classifications

A number of concepts is organized in a classification structure. This is shown in fig. 5.1.

5.2 Rules

In this section an overview of all rules is given. For every rule, the formal expression is shown. The process rules are given first, followed by the invariants.

5.2.1 Process rules

Process rules are rules that are signalled.

Process rule: *caseAuthor*

$V_{[\text{Case}*\text{SESSION}]}; \text{"\_SESSION"}; \text{sessionAccount} \vdash \text{owner}; V_{[\text{Actor}*\text{Account}]}$

Process rule: *TText template parsing - extract placeholders*

TTexts that have no placeholders detected, yet have a ttTemplate, must have been parsed

$(I_{[\text{TText}]} - \text{ttTemplatePlaceholders}; \text{ttTemplatePlaceholders}^{\smile}); \text{ttTemplate} \vdash \text{ttTemplateParsedText}$

Process rule: *TText template parsing - delete parsed templates*

Whenever a template phrase changes, the parsed text must be deleted so that the template is parsed again

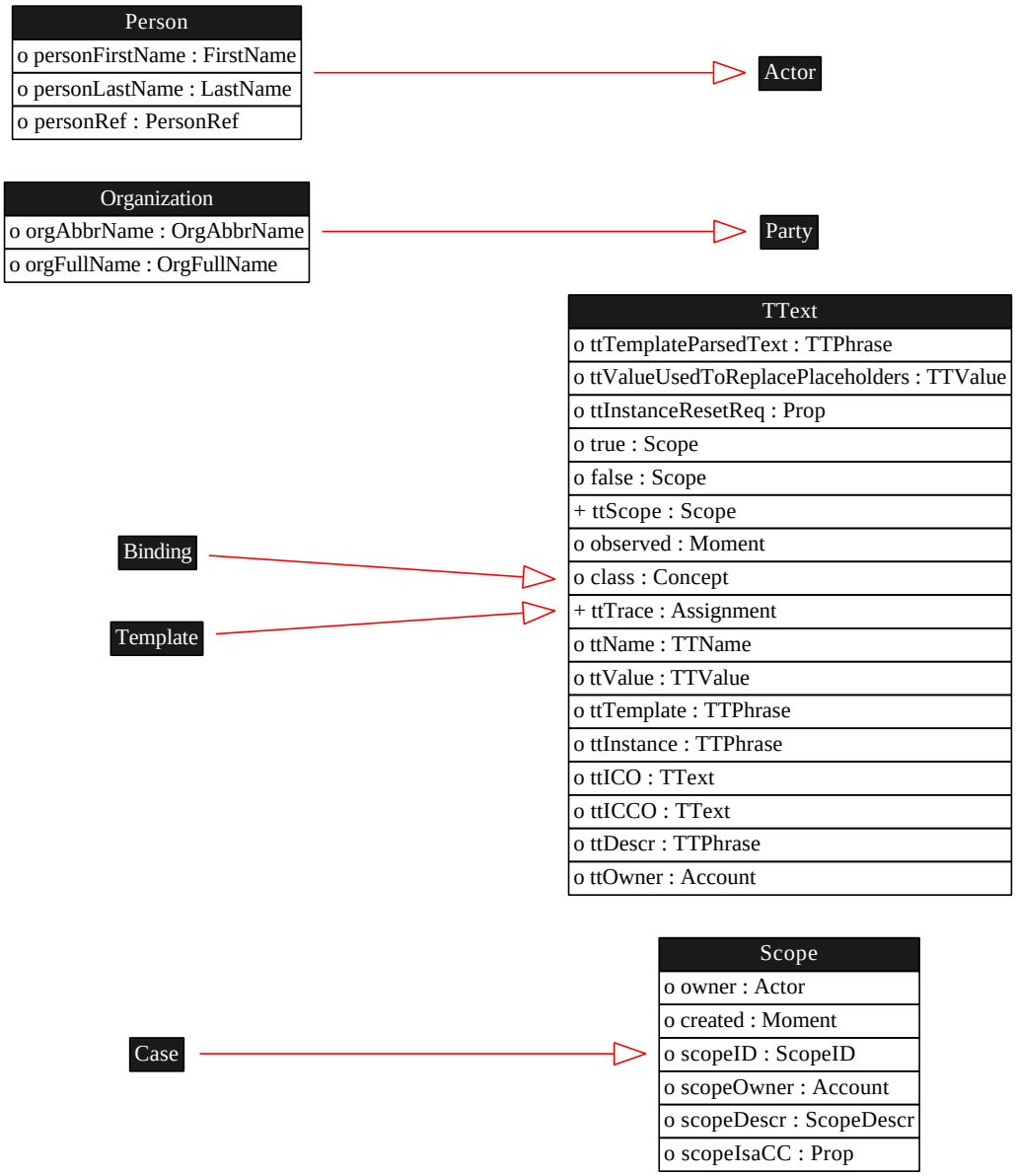

Figure 5.1: Classification of MirrorMe

$\mathrm{ttTemplateParsedText} \vdash \mathrm{ttTemplate}$

Process rule: ***TText template parsing - delete placeholders used in template***

TTexts whose template text is not parsed, must not specify placeholders that are detected

$(I_{[\mathrm{TText}]} - \mathrm{ttTemplateParsedText}; \mathrm{ttTemplateParsedText}^{\smile}); \mathrm{ttTemplatePlaceholders} \vdash \overline{V_{[\mathrm{TText} * \mathrm{TTName}]}}$

Process rule: ***Compute transitive closure of 'ttIsUsedBy'***

$\mathrm{ttIsUsedBy} = \mathrm{ttIsUsedByCopy}$

Process rule: ***Create TTexts for placeholders (if necessary)***

$\mathrm{ttScope}^{\smile}; \mathrm{ttTemplatePlaceholders} \vdash \mathrm{ttScope}^{\smile}; \mathrm{ttName}$

Process rule: ***When the name of a TText is in the list of placeholders of another TText, the first TText is said to be used by the second TText***

$\mathrm{ttName}; \mathrm{ttTemplatePlaceholders}^{\smile} \cap \mathrm{ttScope}; \mathrm{ttScope}^{\smile} \vdash \mathrm{ttIsUsedBy}$

Process rule: ***A (first) TText is only used by a (second) TText if the name of the first is in the list of placeholders of the second***

$\mathrm{ttIsUsedBy} \vdash \mathrm{ttName}; \mathrm{ttTemplatePlaceholders}^{\smile} \cap \mathrm{ttScope}; \mathrm{ttScope}^{\smile}$

Process rule: ***When a TText is used by another TText, it inherits the latter's Scope***

$\mathrm{ttIsUsedBy}; \mathrm{ttScope} \vdash \mathrm{ttScope}$

Process rule: ***Register that the value of a TText has been used to replace placeholders***

If a TText has a value, and is used in a TText that is not being re-initialized, then it must appear in the `ttValueIsUsedInInstanceOfTText` of the TText.

$(I_{[\mathrm{TText}]} \cap \mathrm{ttValue}; \mathrm{ttValue}^{\smile}); \mathrm{ttIsUsedBy}; (I_{[\mathrm{TText}]} - \mathrm{ttInstanceResetReq}) \vdash \mathrm{ttValueIsUsedInInstanceOfTText}$

Process rule: ***After a value update, all TTexts that used the value must be reset***

When the value of a TText differs from the value used in replacements, or has become void, then the instance-texts of all TTexts in which the replacements took place must be discarded and recreated.

$(I_{[\mathrm{TText}]} \cap (\mathrm{ttValueUsedToReplacePlaceholders}; \overline{I_{[\mathrm{TTValue}]}}; \mathrm{ttValue}^{\smile} \cup \overline{(\mathrm{ttValue}; \mathrm{ttValue}^{\smile})})); \mathrm{ttValueIsUsedInIn}$
$\mathrm{ttValueIsUsedInInstanceOfTText}; \mathrm{ttInstanceResetReq}$

Process rule: ***When a TText value is NOT used in an instance of (another) TText, then it may not say that its value is used to replace a placeholder***

ttValueUsedToReplacePlaceholders $\vdash$ ttValueIsUsedInInstanceOfTText; $V_{[\text{TText}*\text{TTValue}]}$

Process rule: ***Request TText instance phrase reset if the template phrase does, and the instance phrase does not exist***

If a TText has a template phrase and no instance phrase, the TText must be reset/initialized, thus allowing its instance phrase to be constructed

$(I_{[\text{TText}]} - \text{ttInstance}; \text{ttInstance}^{\smile}); \text{ttTemplate} \vdash \text{ttInstanceResetReq}; \text{ttTemplate}$

Process rule: ***Resetting a TText implies that the relations with other TTexts in which it is actually used for replacement, are broken***

Resetting a TText means that the registration of replaced placeholders (for that TText) must be reset (cleared/deleted).

$\text{ttValueIsUsedInInstanceOfTText}; \text{ttInstanceResetReq} \vdash \overline{V_{[\text{TText}*\text{TText}]}}$

Process rule: ***Resetting a TText is complete if other TTexts are not used by this TText***

Before reconstructing an instance phrase of a TText, the ExecEngine must have discarded all administration related to that TText.

A TText that needs to be (re-)initialized and does not use values of TTexts in its UsedValue, must have the specification-text as its UsedValue, which completes the (re-)initialization.

$(\text{ttInstanceResetReq} - \text{ttValueIsUsedInInstanceOfTText}^{\smile}; \text{ttValueIsUsedInInstanceOfTText}); \text{ttTemplate} \vdash \text{ttInstance}$

Process rule: ***A TText that is not in a Scope may not exist***

TTexts are destroyed when they are not assigned a ttScope, or their ttScope ceases to exist

$I_{[\text{TText}]} \vdash \text{ttScope}; \text{ttScope}^{\smile}$

Process rule: ***Account activation/initialization***

$I_{[\text{Account}]} \vdash \text{accIsInitialized}$

Process rule: ***Activate 'Developer' role***

$\text{sessionDev} \vdash \text{sessionActiveRoles}; \text{“Developer”}; \text{sessionActiveRoles}^{\smile}$

Process rule: ***Activate roles in a session***

$(I_{[\text{SESSION}]} - \text{sessionActiveRoles}; \text{sessionActiveRoles}^{\smile}); \text{sessionAccount}; \text{accDefaultRoles} \vdash \text{sessionActiveRoles}$

Process rule: ***Activation of session Organization***

The Organization of the user of the session may be known.

$\text{sessionAccount}; \text{accOrg} \vdash \text{sessionOrg}$

Process rule: ***Active accounts are initialized***

$\text{accIsActive} \vdash \text{accIsInitialized}$

Process rule: ***Add arguments to computations***

If the computation should uses (the value of) a TText as an argument (because it is used by the TText for which the computation computes its result), then the latest assignment of that TText should become an argument unless there is an ambiguity (i.e. the computation already has an argument that is an assignment for that TText).

$\text{compVar}; \text{ttIsUsedBy}^{\smile}; \text{ttTrace} \cap \overline{(\text{compArg}; \text{asmVar}; \text{asmVar}^{\smile})} \vdash \text{compArg}$

Process rule: ***Add to Assignments history***

$\text{asmVar} \vdash \text{asmHasPred}^{\smile}; \text{asmVar} \cup \text{ttTrace}^{\smile}$

Process rule: ***Assign computation result to TText (provided actual arguments were used)***

$\text{compVar}^{\smile}; \text{compRes} \vdash \text{ttValue}$

Process rule: ***Auto DelPair sessionSIA***

$\text{sessionSIA} \vdash V_{[\text{SESSION}*\text{Account}]}; (I_{[\text{Account}]} - \text{accIsActive}); V_{[\text{Account}*\text{SESSION}]}$

Process rule: ***Auto activate auto-login accounts***

$\text{autoLoginAccount} \vdash \text{accIsActive}$

Process rule: ***Auto create SIAMPersonRefComputation***

$\text{personFirstName}^{\smile}; \text{personLastName} \vdash \text{siamcompFirstName}^{\smile}; \text{siamcompLastName}$

Process rule: ***Auto maintain 'accPersonRef' relation***

$\text{accPerson}; \text{personRef} \vdash \text{accPersonRef}$

Process rule: ***Auto maintain 'personRef' relation***

$(\text{personFirstName}; \text{siamcompFirstName}^{\smile} \cap \text{personLastName}; \text{siamcompLastName}^{\smile}); \text{siamcompPersonRef} \vdash \text{personRef}$

Process rule: ***Automatically depopulate personOrg***

$\text{personOrg} \vdash \text{accPerson}^{\smile}; \text{accOrg}$

Process rule: ***Automatically login***

$\text{autoLoginAccount} \cap \text{accIsActive} \cap V_{[\text{Account}*\text{SESSION}]}; (\text{"\_SESSION"} - \text{sessionAccount}; \text{sessionAccount}^{\smile}); V_{[\text{SESSION}*\text{Account}]} \vdash \overline{V_{[\text{Account}*\text{Account}]}}$

Process rule: ***Automatically populate personOrg***

$\text{accPerson}^{\smile}; \text{accOrg} \vdash \text{personOrg}$

Process rule: ***By default, a session authentication level is '1'***

$I_{[\text{SESSION}]} \cap \text{sessionAccount}; \text{sessionAccount}^{\smile} \vdash \text{sessionAuthISOLevel}; \text{sessionAuthISOLevel}^{\smile}$

Process rule: ***Clear allowed session roles***

$\text{sessionAllowedRoles} \vdash \text{sessionAccount}; \text{accAllowedRoles}$

Process rule: ***Clear the OrgRef in a session***

$(I_{[\text{SESSION}]} \cap \text{sessionAccount}; \text{sessionAccount}^{\smile}); \text{sessionOrgRef} \vdash \text{sessionAccount}; \text{accOrgRef}$

Process rule: ***Create Account upon request***

$\text{loginCreateAccount} \cap \overline{(\text{sessionAccount}; \text{sessionAccount}^{\smile})} \cap \text{loginUserid}; \text{loginUserid}^{\smile} \cap$
$\text{registerPassword}; \text{registerPassword}^{\smile} \cap \text{sessionPersonRef}; \text{sessionPersonRef}^{\smile} \vdash$
$(\text{loginUserid}; \text{accUserid}^{\smile} \cap \text{registerPassword}; \text{accPassword}^{\smile} \cap \text{sessionPersonRef}; \text{accPersonRef}^{\smile}); V_{[\text{Account} * \text{SESS}}$

Process rule: ***Create Assignment for TTexts that have a (new) value***

$\text{ttValue} \vdash \text{ttTrace}; \text{asmVal}$

Process rule: ***Create Assignment for TTexts that have no value***

$I_{[\text{TText}]} - \text{ttValue}; \text{ttValue}^{\smile} \vdash \text{ttTrace}; (I_{[\text{Assignment}]} - \text{asmVal}; \text{asmVal}^{\smile}); \text{ttTrace}^{\smile}$

Process rule: ***Create computations***

$I_{[\text{TText}]} \cap \text{ttIsUsedBy}^{\smile}; \text{ttIsUsedBy} \vdash \text{compVar}^{\smile}; \text{compVar}$

Process rule: ***Create/Update copied ttDescr***

$\text{ttICCO}; \text{ttDescr} \vdash \text{ttDescr}$

Process rule: ***Create/Update copied ttOwner***

$\text{ttICCO}; \text{ttOwner} \vdash \text{ttOwner}$

Process rule: ***Deactivate 'Developer' role***

$\text{sessionActiveRoles}; \text{“Developer”} \vdash \text{sessionDev}; \text{sessionActiveRoles}; \text{“Developer”}$

Process rule: ***Deactivate roles in a session***

$\text{sessionActiveRoles} \vdash \text{sessionAccount}; V_{[\text{Account} * \text{Role}]}$

Process rule: ***Deactivation of session Organization***

The Organization of the user of the session may be known.

$\text{sessionOrg} \vdash \text{sessionAccount}; \text{accOrg}$

Process rule: ***Default roles must be allowed roles***

$\text{accDefaultRoles} \vdash \text{accAllowedRoles}$

Process rule: ***Delete computations (that have results) of which one or more arguments have changed***

$\overline{(I_{[\text{Computation}]} \cap \text{compRes}; \text{compRes}^{\smile}); \text{compArg}; (I_{[\text{Assignment}]} - \text{asmVar}; \text{ttTrace}) \vdash V_{[\text{Computation} * \text{Assignment}]}}$

Process rule: ***Delete obsolete Assignments***

$I_{[\text{Assignment}]} \vdash \text{asmVar}; \text{asmVar}^{\smile}$

Process rule: ***Delete obsolete Computations***

$I_{[\text{Computation}]} \vdash \text{compVar}; \text{compVar}^{\smile}$

Process rule: ***Determine the 'userid' in a session***

$\text{sessionAccount}; \text{accUserid} \vdash \text{sessionUserid}$

Process rule: ***Determine the OrgRef in a session***

$\text{sessionAccount}; \text{accOrgRef} \vdash \text{sessionOrgRef}$

Process rule: ***Determine the acting person in a session***

$\text{sessionAccount}; \text{accPerson} \vdash \text{sessionPerson}$

Process rule: ***Determine the personRef in a session***

$\text{sessionAccount}; \text{accPersonRef} \vdash \text{sessionPersonRef}$

Process rule: ***Disable automated login***

$\overline{\text{autoLoginAccount} \cap V_{[\text{Account} * \text{SESSION}]}; (\text{"\_SESSION"} \cap \text{sessionAccount}; \text{sessionAccount}^{\smile}); V_{[\text{SESSION} * \text{Account}]} \vdash V_{[\text{Account} * \text{Account}]}}$

Process rule: ***Every TText with scope and name can be used as a binding***

$I_{[\text{TText}]} \cap \text{ttScope}; \text{ttScope}^{\smile} \cap \text{ttName}; \text{ttName}^{\smile} \vdash I_{[\text{Binding}]}$

Process rule: ***Initialize copied ttDescr***

$(I_{[\text{TText}]} - \text{ttDescr}; \text{ttDescr}^{\smile}); \text{ttICO}; \text{ttDescr} \vdash \text{ttDescr}$

Process rule: ***Initialize copied ttOwner***

$(I_{[\text{TText}]} - \text{ttOwner}; \text{ttOwner}^{\smile}); \text{ttICO}; \text{ttOwner} \vdash \text{ttOwner}$

Process rule: ***InsPair sessionAuthISOLevel (MPTrx specific)***

We set sessionAuthISOLevel to whatever value is specified at login

$(I_{[\text{SESSION}]} \cap \text{sessionAccount}; \text{sessionAccount}^{\smile}); \text{loginISOLevel} \vdash \text{sessionAuthISOLevel}$

Process rule: ***Limit history size to 2 predecessors***

$\text{asmHasPred}; \text{asmHasPred}; \text{asmHasPred} \vdash \overline{V_{[\text{Assignment}*\text{Assignment}]}}$

Process rule: ***Logout***

A request to logout in a session must be processed in that session.

$\text{"\_SESSION"}; \text{logoutRequest} \vdash \overline{V_{[\text{SESSION}*\text{SESSION}]}}$

Process rule: ***Registering the first authenticated session account***

Any account may be used to log into a session for the first time.

$(\text{"\_SESSION"} - \text{firstSessionAccount}; \text{firstSessionAccount}^{\smile}); \text{sessionAccount} \vdash \text{firstSessionAccount}$

Process rule: ***Reset login help***

$\text{sessionLoginAssist} \cap \text{sessionAccount}; \text{sessionAccount}^{\smile} \vdash \overline{V_{[\text{SESSION}*\text{SESSION}]}}$

Process rule: ***SURttName***

If a TTName is no longer used in a TText, it is registered no longer.

$I_{[\text{TTName}]} \vdash \text{ttName}^{\smile}; \text{ttName}$

Process rule: ***SURttPhrase***

If a TTPhrase is no longer used in a TText, it is registered no longer.

$I_{[\text{TTPhrase}]} \vdash \text{ttTemplate}^{\smile}; \text{ttTemplate} \cup \text{ttInstance}^{\smile}; \text{ttInstance}$

Process rule: ***Service U/PW Login request***

When a user is authenticated, the corresponding Account will become the sessionAccount (provided it is active).

$\text{loginReq}; (\text{loginUserid}; \text{accUserid}^{\smile} \cap \text{loginPassword}; \text{accPassword}^{\smile}); \text{accIsActive} \vdash \text{sessionAccount}$

Process rule: ***Service deactivation request***

$\text{accDeactivateReq} \vdash \text{sessionAccount}^{\smile}; \text{sessionAccount}$

Process rule: ***Sessions with an inactive sessionaccount may not exist***

$\text{sessionAccount} \vdash \text{sessionAccount}; \text{accIsActive}$

Process rule: ***Set allowed session roles***

$\text{sessionAccount}; \text{accAllowedRoles} \vdash \text{sessionAllowedRoles}$

Process rule: ***Set default Scope ownership***

Ownership of a Scope is assigned by default to the Account that has created the Scope

$(I_{[\text{Scope}]} - \text{scopeOwner}; \text{scopeOwner}^{\smile}); V_{[\text{Scope}*\text{SESSION}]}; \text{"\_SESSION"}; \text{sessionAccount} \vdash \text{scopeOwner}$

Process rule: ***Set default TText ownership***

Ownership of a TText is automatically assigned to the Account that has created the TText

$(I_{[\text{TText}]} - \text{ttOwner}; \text{ttOwner}^{\smile}); V_{[\text{TText}*\text{SESSION}]}; \text{"\_SESSION"}; \text{sessionAccount} \vdash \text{ttOwner}$

Process rule: ***The userid of a customer is its (initial) personref***

$(I_{[\text{Account}]} - \text{accUserid}; \text{accUserid}^{\smile}); \text{accPersonRef} \vdash \text{accUserid}; V_{[\text{UserID}*\text{PersonRef}]}$

Process rule: ***U/PW Login***

When a user is authenticated, the corresponding Account will become the sessionAccount (provided it is active).

$(\text{loginUserid}; \text{accUserid}^{\smile} \cap \text{loginPassword}; \text{accPassword}^{\smile}); \text{accIsActive} \vdash \text{sessionAccount}$

Process rule: ***Undetermine the 'userid' in a session***

$\text{sessionUserid} \vdash \text{sessionAccount}; \text{accUserid}$

Process rule: ***Undetermine the acting person in a session***

$\text{sessionPerson} \vdash \text{sessionAccount}; \text{accPerson}$

Process rule: ***Undetermine the personRef in a session***

$(I_{[\text{SESSION}]} \cap \text{sessionAccount}; \text{sessionAccount}^{\smile}); \text{sessionPersonRef} \vdash \text{sessionAccount}; \text{accPersonRef}$

Process rule: ***Unowned TTexts that are used by an owned TText, will be assigned the same owner***

$(I_{[\text{TText}]} - \text{ttOwner}; \text{ttOwner}^{\smile}); \text{ttIsUsedBy}; \text{ttOwner} \vdash \text{ttOwner}$

Process rule: ***Update arguments of computations that do not have a result***

$\overline{(I_{[\text{Computation}]} - \text{compRes}; \text{compRes}^{\smile}); \text{compArg}; (I_{[\text{Assignment}]} - \text{asmVar}; \text{ttTrace})} \vdash V_{[\text{Computation}*\text{Assignment}]}$

Process rule: ***Update deleted ttDescr***

$\text{ttICCO}; (I_{[\text{TText}]} - \text{ttDescr}; \text{ttDescr}^{\smile}) \vdash (I_{[\text{TText}]} - \text{ttDescr}; \text{ttDescr}^{\smile}); \text{ttICCO}$

Process rule: ***Update deleted ttOwner***

$\text{ttICCO}; (I_{[\text{TText}]} - \text{ttOwner}; \text{ttOwner}^{\smile}) \vdash (I_{[\text{TText}]} - \text{ttOwner}; \text{ttOwner}^{\smile}); \text{ttICCO}$

5.2.2 Invariants

Invariants are rules that are enforced by the database. It is guaranteed that violations cannot occur in the database.

Invariant: ***TTexts can only use TTexts that are in the same ttScope***

$\text{ttIsUsedBy} \vdash \text{ttScope}; \text{ttScope}^{\smile}$

Invariant: ***ttValueUsedToReplacePlaceholders integrity***

$\text{ttValueUsedToReplacePlaceholders} \vdash \text{ttValue}$

Invariant: ***ttValueIsUsedInInstanceOfTText integrity***

$\text{ttValueIsUsedInInstanceOfTText} \vdash \text{ttIsUsedBy}$

Invariant: ***TrueFalseValid***

Every statement (TText) that is in scope is considered valid. This means that it can be true or false, given that all its placeholders have been instantiated.

A statement can be true or false in a context only if it is valid within that context.

$\text{true} \cup \text{false} \vdash \text{ttScope}$

Invariant: ***Inconsistency***

Every statement (TText) that is in scope is considered valid. This means that it can be true or false, if all its placeholders have been instantiated.

A statement cannot be both true and false in the same context.

$\text{true} \cap \text{false} \vdash \overline{V_{[\text{TText}*\text{Scope}]}}$

Invariant: ***An account can only be created for users that are not logged in***

$\text{loginCreateAccount} \cap \text{sessionAccount}; \text{sessionAccount}^{\smile} \vdash \overline{V_{[\text{SESSION}*\text{SESSION}]}}$

Invariant: ***An account may only be created if it has not been previously registered***

$\text{loginCreateAccount}; \text{loginUserid}; \text{accUserid}^{\smile} \vdash \overline{V_{[\text{SESSION}*\text{Account}]}}$

Invariant: ***At least one account must exist that is active***

$V_{[\text{Account}*\text{Account}]}; \text{accIsActive}; V_{[\text{Account}*\text{Account}]}$

Invariant: ***Authenticate user***

$\text{loginUserid}^{\smile}; \text{loginPassword} \vdash \text{accUserid}^{\smile}; \text{accPassword}$

Violations of this rule will result in an error message for the user:

- <violation message should be printed here>

Invariant: ***AutoLoginAccounts must be active***

$\text{autoLoginAccount} \vdash \text{accIsActive}$

Invariant: ***Different computation arguments (assignments) must refer to different TTexts***

$\text{compArg}^{\smile}; \text{compArg} - I_{[\text{Assignment}]} \vdash \text{asmVar}; \overline{I_{[\text{TText}]}}; \text{asmVar}^{\smile}$

Invariant: ***Every assignment has a successor or represents the current value of a TText***

$V_{[\text{ONE}*\text{Assignment}]}; \text{asmHasPred} \cup V_{[\text{ONE}*\text{TText}]}; \text{ttTrace}$

Invariant: ***ISOLevels (LoAs) must be in the range [1-5]***

$I_{[\text{ISOLevel}]} = \text{“1”} \cup \text{“2”} \cup \text{“3”} \cup \text{“4”} \cup \text{“5”}$

Invariant: ***If one assignment precedes another, they affect the same TText***

$\text{asmVar}^{\smile}; \text{asmHasPred}; \text{asmVar} \vdash I_{[\text{TText}]}$

Invariant: ***Integrity of session accounts***

At most one user may log into any session so as to guarantee that it is unambiguously clear which account was used for doing things in the system.

At most one account may be activated in any session.

$(\text{”\_SESSION”} \cap \text{firstSessionAccount}; \text{firstSessionAccount}^{\smile}); \text{sessionAccount} \vdash \text{firstSessionAccount}$

Violations of this rule will result in an error message for the user:

- <violation message should be printed here>

Invariant: ***The trace of each TText starts with an assignment that has no successors***

$\overline{(\text{ttTrace}; \text{asmHasPred}^{\smile})}$

Invariant: ***The user must be authenticated with at least the required ISOLevel***

$\text{sessionAuthISOLevel}^{\smile}; \text{sessionReqdISOLevel} \vdash \text{isoLevelGTE}$

Invariant: ***The value of a TText is per definition the latest assignment value***

$\text{ttValue} = \text{ttTrace}; \text{asmVal}$

Invariant: ***This file expects to load SIAM version 2.x***

$\text{“SIAM”}; \text{moduleVsnMajor} \vdash \text{moduleVsnMajor}; \text{“2”}$

Violations of this rule will result in an error message for the user:

- <violation message should be printed here>

Invariant: ***You must specify a password to create an account***

loginCreateAccount $\vdash$ registerPassword; registerPassword$^\smile$

Invariant: ***You must specify a username to create an account***

loginCreateAccount $\vdash$ loginUserid; loginUserid$^\smile$

Invariant: ***UNI object[Claim*LegalThing]***

object[Claim*LegalThing] is univalent

object; $\overline{I_{[\text{LegalThing}]}}$; object$^\smile$ $\vdash$ $\overline{I_{[\text{Claim}]}}$

Invariant: ***UNI owner[Scope*Actor]***

owner[Scope*Actor] is univalent

owner; $\overline{I_{[\text{Actor}]}}$; owner$^\smile$ $\vdash$ $\overline{I_{[\text{Scope}]}}$

Invariant: ***UNI created[Scope*Moment]***

created[Scope*Moment] is univalent

created; $\overline{I_{[\text{Moment}]}}$; created$^\smile$ $\vdash$ $\overline{I_{[\text{Scope}]}}$

Invariant: ***UNI ttTemplateParsedText[TText*TTPhrase]***

ttTemplateParsedText[TText*TTPhrase] is univalent

ttTemplateParsedText; $\overline{I_{[\text{TTPhrase}]}}$; ttTemplateParsedText$^\smile$ $\vdash$ $\overline{I_{[\text{TText}]}}$

Invariant: ***IRF ttIsUsedBy[TText*TText]***

ttIsUsedBy[TText*TText] is irreflexive

ttIsUsedBy $\vdash$ $\overline{I_{[\text{TText}]}}$

Invariant: ***ASY ttIsUsedBy[TText*TText]***

ttIsUsedBy[TText*TText] is antisymmetric

ttIsUsedBy$^\smile$ $\cap$ ttIsUsedBy $\vdash$ $I_{[\text{TText}]}$

Invariant: ***IRF ttIsUsedByStar[TText*TText]***

ttIsUsedByStar[TText*TText] is irreflexive

ttIsUsedByStar $\vdash$ $\overline{I_{[\text{TText}]}}$

Invariant: ***ASY ttIsUsedByStar[TText*TText]***

ttIsUsedByStar[TText*TText] is antisymmetric

ttIsUsedByStar$^\smile$ $\cap$ ttIsUsedByStar $\vdash$ $I_{[\text{TText}]}$

Invariant: ***TRN ttIsUsedByStar[TText*TText]***

ttIsUsedByStar[TText*TText] is transitive

$\text{ttIsUsedByStar}; \text{ttIsUsedByStar} \vdash \text{ttIsUsedByStar}$

Invariant: *UNI ttValueUsedToReplacePlaceholders[TText*TTValue]*

ttValueUsedToReplacePlaceholders[TText*TTValue] is univalent

$\text{ttValueUsedToReplacePlaceholders}; \overline{I_{[\text{TTValue}]}}; \text{ttValueUsedToReplacePlaceholders}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *SYM ttInstanceResetReq[TText*TText]*

ttInstanceResetReq[TText*TText] is symmetric

$\text{ttInstanceResetReq} = \text{ttInstanceResetReq}^{\smile}$

Invariant: *ASY ttInstanceResetReq[TText*TText]*

ttInstanceResetReq[TText*TText] is antisymmetric

$\text{ttInstanceResetReq}^{\smile} \cap \text{ttInstanceResetReq} \vdash I_{[\text{TText}]}$

Invariant: *UNI ttInstanceResetReq[TText*TText]*

ttInstanceResetReq[TText*TText] is univalent

$\text{ttInstanceResetReq}; \overline{I_{[\text{TText}]}}; \text{ttInstanceResetReq}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *INJ ttInstanceResetReq[TText*TText]*

ttInstanceResetReq[TText*TText] is injective

$\text{ttInstanceResetReq}^{\smile}; \overline{I_{[\text{TText}]}}; \text{ttInstanceResetReq} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *IRF ttValueIsUsedInInstanceOfTText[TText*TText]*

ttValueIsUsedInInstanceOfTText[TText*TText] is irreflexive

$\text{ttValueIsUsedInInstanceOfTText} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *ASY ttValueIsUsedInInstanceOfTText[TText*TText]*

ttValueIsUsedInInstanceOfTText[TText*TText] is antisymmetric

$\text{ttValueIsUsedInInstanceOfTText}^{\smile} \cap \text{ttValueIsUsedInInstanceOfTText} \vdash I_{[\text{TText}]}$

Invariant: *UNI true[TText*Scope]*

true[TText*Scope] is univalent

$\text{true}; \overline{I_{[\text{Scope}]}}; \text{true}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *UNI false[TText*Scope]*

false[TText*Scope] is univalent

$\text{false}; \overline{I_{[\text{Scope}]}}; \text{false}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: ***UNI ttScope[TText*Scope]***

ttScope[TText*Scope] is univalent

$\text{ttScope}; \overline{I_{[\text{Scope}]}}; \text{ttScope}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: ***TOT ttScope[TText*Scope]***

ttScope[TText*Scope] is total

$I_{[\text{TText}]} \vdash \text{ttScope}; \text{ttScope}^{\smile}$

Invariant: ***UNI observed[TText*Moment]***

observed[TText*Moment] is univalent

$\text{observed}; \overline{I_{[\text{Moment}]}}; \text{observed}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: ***UNI regeling[RechtsGrond*Regeling]***

regeling[RechtsGrond*Regeling] is univalent

$\text{regeling}; \overline{I_{[\text{Regeling}]}}; \text{regeling}^{\smile} \vdash \overline{I_{[\text{RechtsGrond}]}}$

Invariant: ***UNI bijlage[RechtsGrond*Bijlage]***

bijlage[RechtsGrond*Bijlage] is univalent

$\text{bijlage}; \overline{I_{[\text{Bijlage}]}}; \text{bijlage}^{\smile} \vdash \overline{I_{[\text{RechtsGrond}]}}$

Invariant: ***UNI aanwijzing[RechtsGrond*Aanwijzing]***

aanwijzing[RechtsGrond*Aanwijzing] is univalent

$\text{aanwijzing}; \overline{I_{[\text{Aanwijzing}]}}; \text{aanwijzing}^{\smile} \vdash \overline{I_{[\text{RechtsGrond}]}}$

Invariant: ***UNI bwb[Regeling*BasisWettenBestand]***

bwb[Regeling*BasisWettenBestand] is univalent

$\text{bwb}; \overline{I_{[\text{BasisWettenBestand}]}}; \text{bwb}^{\smile} \vdash \overline{I_{[\text{Regeling}]}}$

Invariant: ***UNI titel[Regeling*Titel]***

titel[Regeling*Titel] is univalent

$\text{titel}; \overline{I_{[\text{Titel}]}}; \text{titel}^{\smile} \vdash \overline{I_{[\text{Regeling}]}}$

Invariant: ***TOT titel[Regeling*Titel]***

titel[Regeling*Titel] is total

$I_{[\text{Regeling}]} \vdash \text{titel}; \text{titel}^{\smile}$

Invariant: ***UNI artikelen[Regeling*Aantal]***

artikelen[Regeling*Aantal] is univalent

$\text{artikelen}; \overline{I_{[\text{Aantal}]}}; \text{artikelen}^{\smile} \vdash \overline{I_{[\text{Regeling}]}}$

Invariant: *UNI gedelegeerdUit[Regeling*Regeling]*

gedelegeerdUit[Regeling*Regeling] is univalent

$\text{gedelegeerdUit}; \overline{I_{[\text{Regeling}]}}; \text{gedelegeerdUit}^{\smile} \vdash \overline{I_{[\text{Regeling}]}}$

Invariant: *UNI soort[Regeling*Regelingsoort]*

soort[Regeling*Regelingsoort] is univalent

$\text{soort}; \overline{I_{[\text{Regelingsoort}]}}; \text{soort}^{\smile} \vdash \overline{I_{[\text{Regeling}]}}$

Invariant: *UNI procesflow[Regeling*Procesflow]*

procesflow[Regeling*Procesflow] is univalent

$\text{procesflow}; \overline{I_{[\text{Procesflow}]}}; \text{procesflow}^{\smile} \vdash \overline{I_{[\text{Regeling}]}}$

Invariant: *UNI afkorting[Regeling*Tekst]*

afkorting[Regeling*Tekst] is univalent

$\text{afkorting}; \overline{I_{[\text{Tekst}]}}; \text{afkorting}^{\smile} \vdash \overline{I_{[\text{Regeling}]}}$

Invariant: *UNI class[TText*Concept]*

class[TText*Concept] is univalent

$\text{class}; \overline{I_{[\text{Concept}]}}; \text{class}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *SYM autoLoginAccount[Account*Account]*

autoLoginAccount[Account*Account] is symmetric

$\text{autoLoginAccount} = \text{autoLoginAccount}^{\smile}$

Invariant: *ASY autoLoginAccount[Account*Account]*

autoLoginAccount[Account*Account] is antisymmetric

$\text{autoLoginAccount}^{\smile} \cap \text{autoLoginAccount} \vdash I_{[\text{Account}]}$

Invariant: *UNI autoLoginAccount[Account*Account]*

autoLoginAccount[Account*Account] is univalent

$\text{autoLoginAccount}; \overline{I_{[\text{Account}]}}; \text{autoLoginAccount}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: *INJ autoLoginAccount[Account*Account]*

autoLoginAccount[Account*Account] is injective

$\text{autoLoginAccount}^{\smile}; \overline{I_{[\text{Account}]}}; \text{autoLoginAccount} \vdash \overline{I_{[\text{Account}]}}$

Invariant: *INJ accUserid[Account*UserID]*

accUserid[Account*UserID] is injective

$\text{accUserid}^{\smile}; \overline{I_{[\text{Account}]}}; \text{accUserid} \vdash \overline{I_{[\text{UserID}]}}$

Invariant: ***UNI accUserid[Account*UserID]***

accUserid[Account*UserID] is univalent

$\text{accUserid}; \overline{I_{[\text{UserID}]}}; \text{accUserid}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: ***TOT accUserid[Account*UserID]***

accUserid[Account*UserID] is total

$I_{[\text{Account}]} \vdash \text{accUserid}; \text{accUserid}^{\smile}$

Invariant: ***UNI accPassword[Account*Password]***

accPassword[Account*Password] is univalent

$\text{accPassword}; \overline{I_{[\text{Password}]}}; \text{accPassword}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: ***UNI sessionAccount[SESSION*Account]***

sessionAccount[SESSION*Account] is univalent

$\text{sessionAccount}; \overline{I_{[\text{Account}]}}; \text{sessionAccount}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***SYM accIsInitialized[Account*Account]***

accIsInitialized[Account*Account] is symmetric

$\text{accIsInitialized} = \text{accIsInitialized}^{\smile}$

Invariant: ***ASY accIsInitialized[Account*Account]***

accIsInitialized[Account*Account] is antisymmetric

$\text{accIsInitialized}^{\smile} \cap \text{accIsInitialized} \vdash I_{[\text{Account}]}$

Invariant: ***UNI accIsInitialized[Account*Account]***

accIsInitialized[Account*Account] is univalent

$\text{accIsInitialized}; \overline{I_{[\text{Account}]}}; \text{accIsInitialized}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: ***INJ accIsInitialized[Account*Account]***

accIsInitialized[Account*Account] is injective

$\text{accIsInitialized}^{\smile}; \overline{I_{[\text{Account}]}}; \text{accIsInitialized} \vdash \overline{I_{[\text{Account}]}}$

Invariant: ***SYM accIsActive[Account*Account]***

accIsActive[Account*Account] is symmetric

$\text{accIsActive} = \text{accIsActive}^{\smile}$

Invariant: ***ASY accIsActive[Account*Account]***

accIsActive[Account*Account] is antisymmetric

$\text{accIsActive}^{\smile} \cap \text{accIsActive} \vdash I_{[\text{Account}]}$

Invariant: *UNI accIsActive[Account*Account]*

accIsActive[Account*Account] is univalent

$\text{accIsActive}; \overline{I_{[\text{Account}]}}; \text{accIsActive}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: *INJ accIsActive[Account*Account]*

accIsActive[Account*Account] is injective

$\text{accIsActive}^{\smile}; \overline{I_{[\text{Account}]}}; \text{accIsActive} \vdash \overline{I_{[\text{Account}]}}$

Invariant: *SYM accDeactivateReq[Account*Account]*

accDeactivateReq[Account*Account] is symmetric

$\text{accDeactivateReq} = \text{accDeactivateReq}^{\smile}$

Invariant: *ASY accDeactivateReq[Account*Account]*

accDeactivateReq[Account*Account] is antisymmetric

$\text{accDeactivateReq}^{\smile} \cap \text{accDeactivateReq} \vdash I_{[\text{Account}]}$

Invariant: *UNI accDeactivateReq[Account*Account]*

accDeactivateReq[Account*Account] is univalent

$\text{accDeactivateReq}; \overline{I_{[\text{Account}]}}; \text{accDeactivateReq}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: *INJ accDeactivateReq[Account*Account]*

accDeactivateReq[Account*Account] is injective

$\text{accDeactivateReq}^{\smile}; \overline{I_{[\text{Account}]}}; \text{accDeactivateReq} \vdash \overline{I_{[\text{Account}]}}$

Invariant: *UNI sessionUserid[SESSION*UserID]*

sessionUserid[SESSION*UserID] is univalent

$\text{sessionUserid}; \overline{I_{[\text{UserID}]}}; \text{sessionUserid}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: *UNI loginUserid[SESSION*UserID]*

loginUserid[SESSION*UserID] is univalent

$\text{loginUserid}; \overline{I_{[\text{UserID}]}}; \text{loginUserid}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: *UNI loginPassword[SESSION*Password]*

loginPassword[SESSION*Password] is univalent

$\text{loginPassword}; \overline{I_{[\text{Password}]}}; \text{loginPassword}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: *SYM logoutRequest[SESSION*SESSION]*

logoutRequest[SESSION*SESSION] is symmetric

$\text{logoutRequest} = \text{logoutRequest}^{\smile}$

Invariant: ***ASY logoutRequest[SESSION*SESSION]***

logoutRequest[SESSION*SESSION] is antisymmetric

$\text{logoutRequest}^{\smile} \cap \text{logoutRequest} \vdash I_{[\text{SESSION}]}$

Invariant: ***UNI logoutRequest[SESSION*SESSION]***

logoutRequest[SESSION*SESSION] is univalent

$\text{logoutRequest}; \overline{I_{[\text{SESSION}]}}; \text{logoutRequest}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***INJ logoutRequest[SESSION*SESSION]***

logoutRequest[SESSION*SESSION] is injective

$\text{logoutRequest}^{\smile}; \overline{I_{[\text{SESSION}]}}; \text{logoutRequest} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***RFX isoLevelGTE[ISOLevel*ISOLevel]***

isoLevelGTE[ISOLevel*ISOLevel] is reflexive

$I_{[\text{ISOLevel}]} \vdash \text{isoLevelGTE}$

Invariant: ***ASY isoLevelGTE[ISOLevel*ISOLevel]***

isoLevelGTE[ISOLevel*ISOLevel] is antisymmetric

$\text{isoLevelGTE}^{\smile} \cap \text{isoLevelGTE} \vdash I_{[\text{ISOLevel}]}$

Invariant: ***TRN isoLevelGTE[ISOLevel*ISOLevel]***

isoLevelGTE[ISOLevel*ISOLevel] is transitive

$\text{isoLevelGTE}; \text{isoLevelGTE} \vdash \text{isoLevelGTE}$

Invariant: ***UNI sessionReqdISOLevel[SESSION*ISOLevel]***

sessionReqdISOLevel[SESSION*ISOLevel] is univalent

$\text{sessionReqdISOLevel}; \overline{I_{[\text{ISOLevel}]}}; \text{sessionReqdISOLevel}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***UNI sessionAuthISOLevel[SESSION*ISOLevel]***

sessionAuthISOLevel[SESSION*ISOLevel] is univalent

$\text{sessionAuthISOLevel}; \overline{I_{[\text{ISOLevel}]}}; \text{sessionAuthISOLevel}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***UNI moduleName[Module*ModuleName]***

moduleName[Module*ModuleName] is univalent

$\text{moduleName}; \overline{I_{[\text{ModuleName}]}}; \text{moduleName}^{\smile} \vdash \overline{I_{[\text{Module}]}}$

Invariant: ***TOT moduleName[Module*ModuleName]***

moduleName[Module*ModuleName] is total

$I_{[\text{Module}]} \vdash \text{moduleName}; \text{moduleName}^{\smile}$

Invariant: ***UNI moduleVsnMajor[Module*ModuleVsnMajor]***

moduleVsnMajor[Module*ModuleVsnMajor] is univalent

$\text{moduleVsnMajor}; \overline{I_{[\text{ModuleVsnMajor}]}}; \text{moduleVsnMajor}^{\smile} \vdash \overline{I_{[\text{Module}]}}$

Invariant: ***TOT moduleVsnMajor[Module*ModuleVsnMajor]***

moduleVsnMajor[Module*ModuleVsnMajor] is total

$I_{[\text{Module}]} \vdash \text{moduleVsnMajor}; \text{moduleVsnMajor}^{\smile}$

Invariant: ***UNI moduleVsnMinor[Module*ModuleVsnMinor]***

moduleVsnMinor[Module*ModuleVsnMinor] is univalent

$\text{moduleVsnMinor}; \overline{I_{[\text{ModuleVsnMinor}]}}; \text{moduleVsnMinor}^{\smile} \vdash \overline{I_{[\text{Module}]}}$

Invariant: ***TOT moduleVsnMinor[Module*ModuleVsnMinor]***

moduleVsnMinor[Module*ModuleVsnMinor] is total

$I_{[\text{Module}]} \vdash \text{moduleVsnMinor}; \text{moduleVsnMinor}^{\smile}$

Invariant: ***INJ orgAbbrName[Organization*OrgAbbrName]***

orgAbbrName[Organization*OrgAbbrName] is injective

$\text{orgAbbrName}^{\smile}; \overline{I_{[\text{Organization}]}}; \text{orgAbbrName} \vdash \overline{I_{[\text{OrgAbbrName}]}}$

Invariant: ***UNI orgAbbrName[Organization*OrgAbbrName]***

orgAbbrName[Organization*OrgAbbrName] is univalent

$\text{orgAbbrName}; \overline{I_{[\text{OrgAbbrName}]}}; \text{orgAbbrName}^{\smile} \vdash \overline{I_{[\text{Organization}]}}$

Invariant: ***INJ orgFullName[Organization*OrgFullName]***

orgFullName[Organization*OrgFullName] is injective

$\text{orgFullName}^{\smile}; \overline{I_{[\text{Organization}]}}; \text{orgFullName} \vdash \overline{I_{[\text{OrgFullName}]}}$

Invariant: ***UNI orgFullName[Organization*OrgFullName]***

orgFullName[Organization*OrgFullName] is univalent

$\text{orgFullName}; \overline{I_{[\text{OrgFullName}]}}; \text{orgFullName}^{\smile} \vdash \overline{I_{[\text{Organization}]}}$

Invariant: ***UNI accOrg[Account*Organization]***

accOrg[Account*Organization] is univalent

$\text{accOrg}; \overline{I_{[\text{Organization}]}}; \text{accOrg}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: ***UNI sessionOrg[SESSION*Organization]***

sessionOrg[SESSION*Organization] is univalent

$\text{sessionOrg}; \overline{I_{[\text{Organization}]}}; \text{sessionOrg}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: *UNI personFirstName[Person*FirstName]*

personFirstName[Person*FirstName] is univalent

$\text{personFirstName}; \overline{I_{[\text{FirstName}]}}; \text{personFirstName}^{\smile} \vdash \overline{I_{[\text{Person}]}}$

Invariant: *TOT personFirstName[Person*FirstName]*

personFirstName[Person*FirstName] is total

$I_{[\text{Person}]} \vdash \text{personFirstName}; \text{personFirstName}^{\smile}$

Invariant: *UNI personLastName[Person*LastName]*

personLastName[Person*LastName] is univalent

$\text{personLastName}; \overline{I_{[\text{LastName}]}}; \text{personLastName}^{\smile} \vdash \overline{I_{[\text{Person}]}}$

Invariant: *TOT personLastName[Person*LastName]*

personLastName[Person*LastName] is total

$I_{[\text{Person}]} \vdash \text{personLastName}; \text{personLastName}^{\smile}$

Invariant: *UNI accPerson[Account*Person]*

accPerson[Account*Person] is univalent

$\text{accPerson}; \overline{I_{[\text{Person}]}}; \text{accPerson}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: *UNI sessionPerson[SESSION*Person]*

sessionPerson[SESSION*Person] is univalent

$\text{sessionPerson}; \overline{I_{[\text{Person}]}}; \text{sessionPerson}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: *UNI firstSessionAccount[SESSION*Account]*

firstSessionAccount[SESSION*Account] is univalent

$\text{firstSessionAccount}; \overline{I_{[\text{Account}]}}; \text{firstSessionAccount}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: *UNI uidUserid[UID*UserID]*

uidUserid[UID*UserID] is univalent

$\text{uidUserid}; \overline{I_{[\text{UserID}]}}; \text{uidUserid}^{\smile} \vdash \overline{I_{[\text{UID}]}}$

Invariant: *TOT uidUserid[UID*UserID]*

uidUserid[UID*UserID] is total

$I_{[\text{UID}]} \vdash \text{uidUserid}; \text{uidUserid}^{\smile}$

Invariant: *UNI uidIssuer[UID*IdP]*

uidIssuer[UID*IdP] is univalent

$\text{uidIssuer}; \overline{I_{[\text{IdP}]}}; \text{uidIssuer}^{\smile} \vdash \overline{I_{[\text{UID}]}}$

Invariant: ***INJ accUID[Account*UID]***

accUID[Account*UID] is injective

$\text{accUID}^{\smile}; \overline{I_{[\text{Account}]}}; \text{accUID} \vdash \overline{I_{[\text{UID}]}}$

Invariant: ***UNI personRef[Person*PersonRef]***

personRef[Person*PersonRef] is univalent

$\text{personRef}; \overline{I_{[\text{PersonRef}]}}; \text{personRef}^{\smile} \vdash \overline{I_{[\text{Person}]}}$

Invariant: ***UNI accPersonRef[Account*PersonRef]***

accPersonRef[Account*PersonRef] is univalent

$\text{accPersonRef}; \overline{I_{[\text{PersonRef}]}}; \text{accPersonRef}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: ***UNI sessionPersonRef[SESSION*PersonRef]***

sessionPersonRef[SESSION*PersonRef] is univalent

$\text{sessionPersonRef}; \overline{I_{[\text{PersonRef}]}}; \text{sessionPersonRef}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***UNI siamcompFirstName[SIAMPersonRefComputation*FirstName]***

siamcompFirstName[SIAMPersonRefComputation*FirstName] is univalent

$\text{siamcompFirstName}; \overline{I_{[\text{FirstName}]}}; \text{siamcompFirstName}^{\smile} \vdash \overline{I_{[\text{SIAMPersonRefComputation}]}}$

Invariant: ***UNI siamcompLastName[SIAMPersonRefComputation*LastName]***

siamcompLastName[SIAMPersonRefComputation*LastName] is univalent

$\text{siamcompLastName}; \overline{I_{[\text{LastName}]}}; \text{siamcompLastName}^{\smile} \vdash \overline{I_{[\text{SIAMPersonRefComputation}]}}$

Invariant: ***UNI siamcompPersonRef[SIAMPersonRefComputation*PersonRef]***

siamcompPersonRef[SIAMPersonRefComputation*PersonRef] is univalent

$\text{siamcompPersonRef}; \overline{I_{[\text{PersonRef}]}}; \text{siamcompPersonRef}^{\smile} \vdash \overline{I_{[\text{SIAMPersonRefComputation}]}}$

Invariant: ***UNI accOrgRef[Account*OrgRef]***

accOrgRef[Account*OrgRef] is univalent

$\text{accOrgRef}; \overline{I_{[\text{OrgRef}]}}; \text{accOrgRef}^{\smile} \vdash \overline{I_{[\text{Account}]}}$

Invariant: ***UNI sessionOrgRef[SESSION*OrgRef]***

sessionOrgRef[SESSION*OrgRef] is univalent

$\text{sessionOrgRef}; \overline{I_{[\text{OrgRef}]}}; \text{sessionOrgRef}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***SYM loginCreateAccount[SESSION*SESSION]***

loginCreateAccount[SESSION*SESSION] is symmetric

$\text{loginCreateAccount} = \text{loginCreateAccount}^{\smile}$

Invariant: ***ASY loginCreateAccount[SESSION*SESSION]***

loginCreateAccount[SESSION*SESSION] is antisymmetric

$\text{loginCreateAccount}^{\smile} \cap \text{loginCreateAccount} \vdash I_{[\text{SESSION}]}$

Invariant: ***UNI loginCreateAccount[SESSION*SESSION]***

loginCreateAccount[SESSION*SESSION] is univalent

$\text{loginCreateAccount}; \overline{I_{[\text{SESSION}]}}; \text{loginCreateAccount}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***INJ loginCreateAccount[SESSION*SESSION]***

loginCreateAccount[SESSION*SESSION] is injective

$\text{loginCreateAccount}^{\smile}; \overline{I_{[\text{SESSION}]}}; \text{loginCreateAccount} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***UNI registerPassword[SESSION*Password]***

registerPassword[SESSION*Password] is univalent

$\text{registerPassword}; \overline{I_{[\text{Password}]}}; \text{registerPassword}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***UNI registerOrgRef[SESSION*OrgRef]***

registerOrgRef[SESSION*OrgRef] is univalent

$\text{registerOrgRef}; \overline{I_{[\text{OrgRef}]}}; \text{registerOrgRef}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***SYM loginReq[SESSION*SESSION]***

loginReq[SESSION*SESSION] is symmetric

$\text{loginReq} = \text{loginReq}^{\smile}$

Invariant: ***ASY loginReq[SESSION*SESSION]***

loginReq[SESSION*SESSION] is antisymmetric

$\text{loginReq}^{\smile} \cap \text{loginReq} \vdash I_{[\text{SESSION}]}$

Invariant: ***UNI loginReq[SESSION*SESSION]***

loginReq[SESSION*SESSION] is univalent

$\text{loginReq}; \overline{I_{[\text{SESSION}]}}; \text{loginReq}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***INJ loginReq[SESSION*SESSION]***

loginReq[SESSION*SESSION] is injective

$\text{loginReq}^{\smile}; \overline{I_{[\text{SESSION}]}}; \text{loginReq} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***UNI loginISOLevel[SESSION*ISOLevel]***

loginISOLevel[SESSION*ISOLevel] is univalent

$\text{loginISOLevel}; \overline{I_{[\text{ISOLevel}]}}; \text{loginISOLevel}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: *SYM sessionDev[SESSION*SESSION]*

sessionDev[SESSION*SESSION] is symmetric

$\text{sessionDev} = \text{sessionDev}^{\smile}$

Invariant: *ASY sessionDev[SESSION*SESSION]*

sessionDev[SESSION*SESSION] is antisymmetric

$\text{sessionDev}^{\smile} \cap \text{sessionDev} \vdash I_{[\text{SESSION}]}$

Invariant: *UNI sessionDev[SESSION*SESSION]*

sessionDev[SESSION*SESSION] is univalent

$\text{sessionDev}; \overline{I_{[\text{SESSION}]}}; \text{sessionDev}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: *INJ sessionDev[SESSION*SESSION]*

sessionDev[SESSION*SESSION] is injective

$\text{sessionDev}^{\smile}; \overline{I_{[\text{SESSION}]}}; \text{sessionDev} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: *INJ ttTrace[TText*Assignment]*

ttTrace[TText*Assignment] is injective

$\text{ttTrace}^{\smile}; \overline{I_{[\text{TText}]}}; \text{ttTrace} \vdash \overline{I_{[\text{Assignment}]}}$

Invariant: *UNI ttTrace[TText*Assignment]*

ttTrace[TText*Assignment] is univalent

$\text{ttTrace}; \overline{I_{[\text{Assignment}]}}; \text{ttTrace}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *TOT ttTrace[TText*Assignment]*

ttTrace[TText*Assignment] is total

$I_{[\text{TText}]} \vdash \text{ttTrace}; \text{ttTrace}^{\smile}$

Invariant: *UNI asmVar[Assignment*TText]*

asmVar[Assignment*TText] is univalent

$\text{asmVar}; \overline{I_{[\text{TText}]}}; \text{asmVar}^{\smile} \vdash \overline{I_{[\text{Assignment}]}}$

Invariant: *TOT asmVar[Assignment*TText]*

asmVar[Assignment*TText] is total

$I_{[\text{Assignment}]} \vdash \text{asmVar}; \text{asmVar}^{\smile}$

Invariant: *UNI asmVal[Assignment*TTValue]*

asmVal[Assignment*TTValue] is univalent

$\text{asmVal}; \overline{I_{[\text{TTValue}]}}; \text{asmVal}^{\smile} \vdash \overline{I_{[\text{Assignment}]}}$

Invariant: ***UNI asmPOT[Assignment*PointOfTime]***

asmPOT[Assignment*PointOfTime] is univalent

$\text{asmPOT}; \overline{I_{[\text{PointOfTime}]}}; \text{asmPOT}^{\smile} \vdash \overline{I_{[\text{Assignment}]}}$

Invariant: ***UNI asmHasPred[Assignment*Assignment]***

asmHasPred[Assignment*Assignment] is univalent

$\text{asmHasPred}; \overline{I_{[\text{Assignment}]}}; \text{asmHasPred}^{\smile} \vdash \overline{I_{[\text{Assignment}]}}$

Invariant: ***INJ asmHasPred[Assignment*Assignment]***

asmHasPred[Assignment*Assignment] is injective

$\text{asmHasPred}^{\smile}; \overline{I_{[\text{Assignment}]}}; \text{asmHasPred} \vdash \overline{I_{[\text{Assignment}]}}$

Invariant: ***INJ compVar[Computation*TText]***

compVar[Computation*TText] is injective

$\text{compVar}^{\smile}; \overline{I_{[\text{Computation}]}}; \text{compVar} \vdash \overline{I_{[\text{TText}]}}$

Invariant: ***UNI compVar[Computation*TText]***

compVar[Computation*TText] is univalent

$\text{compVar}; \overline{I_{[\text{TText}]}}; \text{compVar}^{\smile} \vdash \overline{I_{[\text{Computation}]}}$

Invariant: ***TOT compVar[Computation*TText]***

compVar[Computation*TText] is total

$I_{[\text{Computation}]} \vdash \text{compVar}; \text{compVar}^{\smile}$

Invariant: ***UNI compRes[Computation*TTValue]***

compRes[Computation*TTValue] is univalent

$\text{compRes}; \overline{I_{[\text{TTValue}]}}; \text{compRes}^{\smile} \vdash \overline{I_{[\text{Computation}]}}$

Invariant: ***SYM sessionLoginAssist[SESSION*SESSION]***

sessionLoginAssist[SESSION*SESSION] is symmetric

$\text{sessionLoginAssist} = \text{sessionLoginAssist}^{\smile}$

Invariant: ***ASY sessionLoginAssist[SESSION*SESSION]***

sessionLoginAssist[SESSION*SESSION] is antisymmetric

$\text{sessionLoginAssist}^{\smile} \cap \text{sessionLoginAssist} \vdash I_{[\text{SESSION}]}$

Invariant: ***UNI sessionLoginAssist[SESSION*SESSION]***

sessionLoginAssist[SESSION*SESSION] is univalent

$\text{sessionLoginAssist}; \overline{I_{[\text{SESSION}]}}; \text{sessionLoginAssist}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***INJ sessionLoginAssist[SESSION*SESSION]***

sessionLoginAssist[SESSION*SESSION] is injective

$\text{sessionLoginAssist}^{\smile}; \overline{I_{[\text{SESSION}]}}; \text{sessionLoginAssist} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***SYM sessionLogoutAssist[SESSION*SESSION]***

sessionLogoutAssist[SESSION*SESSION] is symmetric

$\text{sessionLogoutAssist} = \text{sessionLogoutAssist}^{\smile}$

Invariant: ***ASY sessionLogoutAssist[SESSION*SESSION]***

sessionLogoutAssist[SESSION*SESSION] is antisymmetric

$\text{sessionLogoutAssist}^{\smile} \cap \text{sessionLogoutAssist} \vdash I_{[\text{SESSION}]}$

Invariant: ***UNI sessionLogoutAssist[SESSION*SESSION]***

sessionLogoutAssist[SESSION*SESSION] is univalent

$\text{sessionLogoutAssist}; \overline{I_{[\text{SESSION}]}}; \text{sessionLogoutAssist}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***INJ sessionLogoutAssist[SESSION*SESSION]***

sessionLogoutAssist[SESSION*SESSION] is injective

$\text{sessionLogoutAssist}^{\smile}; \overline{I_{[\text{SESSION}]}}; \text{sessionLogoutAssist} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***SYM sessionSRI[SESSION*SESSION]***

sessionSRI[SESSION*SESSION] is symmetric

$\text{sessionSRI} = \text{sessionSRI}^{\smile}$

Invariant: ***ASY sessionSRI[SESSION*SESSION]***

sessionSRI[SESSION*SESSION] is antisymmetric

$\text{sessionSRI}^{\smile} \cap \text{sessionSRI} \vdash I_{[\text{SESSION}]}$

Invariant: ***UNI sessionSRI[SESSION*SESSION]***

sessionSRI[SESSION*SESSION] is univalent

$\text{sessionSRI}; \overline{I_{[\text{SESSION}]}}; \text{sessionSRI}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***INJ sessionSRI[SESSION*SESSION]***

sessionSRI[SESSION*SESSION] is injective

$\text{sessionSRI}^{\smile}; \overline{I_{[\text{SESSION}]}}; \text{sessionSRI} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***SYM sessionSIA[SESSION*SESSION]***

sessionSIA[SESSION*SESSION] is symmetric

$\text{sessionSIA} = \text{sessionSIA}^{\smile}$

Invariant: ***ASY sessionSIA[SESSION*SESSION]***

sessionSIA[SESSION*SESSION] is antisymmetric

$\text{sessionSIA}^{\smile} \cap \text{sessionSIA} \vdash I_{[\text{SESSION}]}$

Invariant: ***UNI sessionSIA[SESSION*SESSION]***

sessionSIA[SESSION*SESSION] is univalent

$\text{sessionSIA}; \overline{I_{[\text{SESSION}]}}; \text{sessionSIA}^{\smile} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***INJ sessionSIA[SESSION*SESSION]***

sessionSIA[SESSION*SESSION] is injective

$\text{sessionSIA}^{\smile}; \overline{I_{[\text{SESSION}]}}; \text{sessionSIA} \vdash \overline{I_{[\text{SESSION}]}}$

Invariant: ***UNI scopeID[Scope*ScopeID]***

scopeID[Scope*ScopeID] is univalent

$\text{scopeID}; \overline{I_{[\text{ScopeID}]}}; \text{scopeID}^{\smile} \vdash \overline{I_{[\text{Scope}]}}$

Invariant: ***UNI scopeOwner[Scope*Account]***

scopeOwner[Scope*Account] is univalent

$\text{scopeOwner}; \overline{I_{[\text{Account}]}}; \text{scopeOwner}^{\smile} \vdash \overline{I_{[\text{Scope}]}}$

Invariant: ***UNI scopeDescr[Scope*ScopeDescr]***

scopeDescr[Scope*ScopeDescr] is univalent

$\text{scopeDescr}; \overline{I_{[\text{ScopeDescr}]}}; \text{scopeDescr}^{\smile} \vdash \overline{I_{[\text{Scope}]}}$

Invariant: ***IRF scopeIII[Scope*Scope]***

scopeIII[Scope*Scope] is irreflexive

$\text{scopeIII} \vdash \overline{I_{[\text{Scope}]}}$

Invariant: ***ASY scopeIII[Scope*Scope]***

scopeIII[Scope*Scope] is antisymmetric

$\text{scopeIII}^{\smile} \cap \text{scopeIII} \vdash I_{[\text{Scope}]}$

Invariant: ***SYM scopeIsaCC[Scope*Scope]***

scopeIsaCC[Scope*Scope] is symmetric

$\text{scopeIsaCC} = \text{scopeIsaCC}^{\smile}$

Invariant: ***ASY scopeIsaCC[Scope*Scope]***

scopeIsaCC[Scope*Scope] is antisymmetric

$\text{scopeIsaCC}^{\smile} \cap \text{scopeIsaCC} \vdash I_{[\text{Scope}]}$

Invariant: *UNI scopeIsaCC[Scope*Scope]*

scopeIsaCC[Scope*Scope] is univalent

$\text{scopeIsaCC}; \overline{I_{[\text{Scope}]}}; \text{scopeIsaCC}^{\smile} \vdash \overline{I_{[\text{Scope}]}}$

Invariant: *INJ scopeIsaCC[Scope*Scope]*

scopeIsaCC[Scope*Scope] is injective

$\text{scopeIsaCC}^{\smile}; \overline{I_{[\text{Scope}]}}; \text{scopeIsaCC} \vdash \overline{I_{[\text{Scope}]}}$

Invariant: *UNI ttName[TText*TTName]*

ttName[TText*TTName] is univalent

$\text{ttName}; \overline{I_{[\text{TTName}]}}; \text{ttName}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *UNI ttValue[TText*TTValue]*

ttValue[TText*TTValue] is univalent

$\text{ttValue}; \overline{I_{[\text{TTValue}]}}; \text{ttValue}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *UNI ttTemplate[TText*TTPhrase]*

ttTemplate[TText*TTPhrase] is univalent

$\text{ttTemplate}; \overline{I_{[\text{TTPhrase}]}}; \text{ttTemplate}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *UNI ttInstance[TText*TTPhrase]*

ttInstance[TText*TTPhrase] is univalent

$\text{ttInstance}; \overline{I_{[\text{TTPhrase}]}}; \text{ttInstance}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *UNI ttICO[TText*TText]*

ttICO[TText*TText] is univalent

$\text{ttICO}; \overline{I_{[\text{TText}]}}; \text{ttICO}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *IRF ttICO[TText*TText]*

ttICO[TText*TText] is irreflexive

$\text{ttICO} \vdash \overline{I_{[\text{TText}]}}$

Invariant: *ASY ttICO[TText*TText]*

ttICO[TText*TText] is antisymmetric

$\text{ttICO}^{\smile} \cap \text{ttICO} \vdash I_{[\text{TText}]}$

Invariant: *UNI ttICCO[TText*TText]*

ttICCO[TText*TText] is univalent

$\text{ttICCO}; \overline{I_{[\text{TText}]}}; \text{ttICCO}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: ***IRF ttICCO[TText*TText]***

ttICCO[TText*TText] is irreflexive

$\text{ttICCO} \vdash \overline{I_{[\text{TText}]}}$

Invariant: ***ASY ttICCO[TText*TText]***

ttICCO[TText*TText] is antisymmetric

$\text{ttICCO}^{\smile} \cap \text{ttICCO} \vdash I_{[\text{TText}]}$

Invariant: ***UNI ttDescr[TText*TTPhrase]***

ttDescr[TText*TTPhrase] is univalent

$\text{ttDescr}; \overline{I_{[\text{TTPhrase}]}}; \text{ttDescr}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: ***UNI ttOwner[TText*Account]***

ttOwner[TText*Account] is univalent

$\text{ttOwner}; \overline{I_{[\text{Account}]}}; \text{ttOwner}^{\smile} \vdash \overline{I_{[\text{TText}]}}$

Invariant: ***identity_TTexts***

Identity rule, following from identity TTexts

$\text{ttScope}; \text{ttScope}^{\smile} \cap \text{ttName}; \text{ttName}^{\smile} \vdash I_{[\text{TText}]}$

Invariant: ***identity_Scopes***

Identity rule, following from identity Scopes

$\text{scopeOwner}; \text{scopeOwner}^{\smile} \cap \text{scopeID}; \text{scopeID}^{\smile} \vdash I_{[\text{Scope}]}$

Invariant: ***identity_UIDs***

Identity rule, following from identity UIDs

$\text{uidIssuer}; \text{uidIssuer}^{\smile} \cap \text{uidUserid}; \text{uidUserid}^{\smile} \vdash I_{[\text{UID}]}$

Invariant: ***identity_SIAMPersonRefComputations***

Identity rule, following from identity SIAMPersonRefComputations

$\text{siamcompFirstName}; \text{siamcompFirstName}^{\smile} \cap \text{siamcompLastName}; \text{siamcompLastName}^{\smile} \vdash I_{[\text{SIAMPersonRefComputation}]}$

Invariant: ***identity_Computations***

Identity rule, following from identity Computations

$\text{compArg}; \text{compArg}^{\smile} \cap \text{compVar}; \text{compVar}^{\smile} \vdash I_{[\text{Computation}]}$

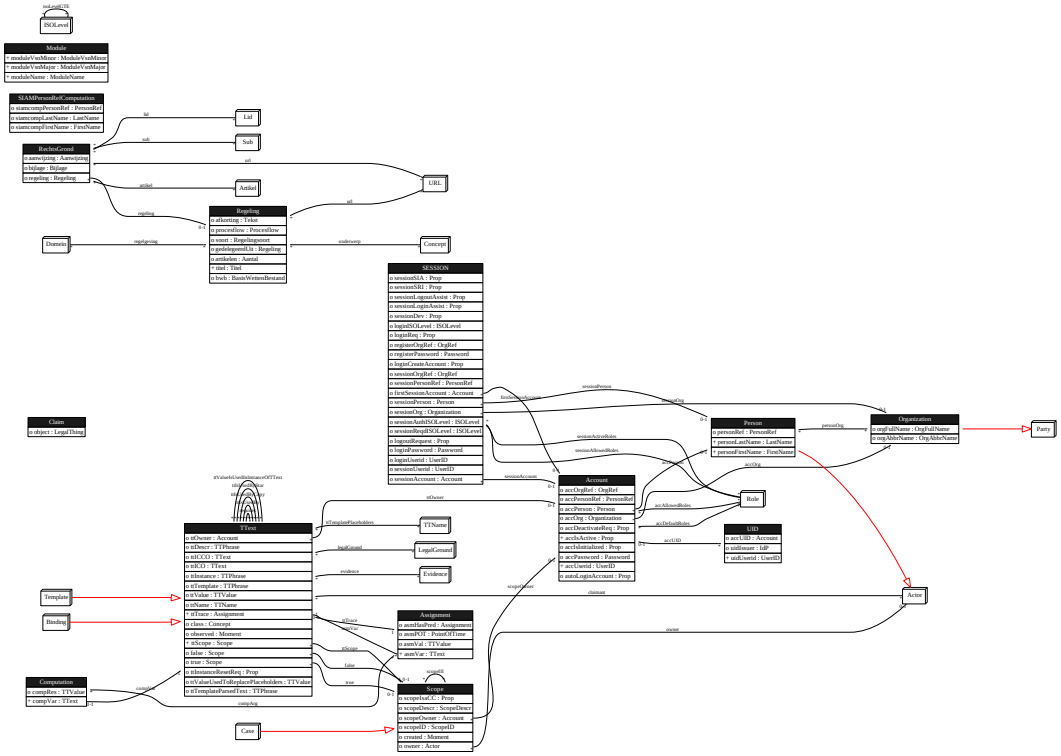

Figure 5.2: Logical data model of MirrorMe

5.3 Logical data model

The functional requirements have been translated into a data model. This model is shown by fig. 5.2.

There are 14 entity types.The details of each entity type are described (in alphabetical order) in the following two tables:

Logical entity types

Concept	Meaning	Type
Account	een verzameling gegevens die (een deel van) de gebruikerscontext van één gebruiker binnen het systeem beschrijft Als iemand inlogt in het systeem moet diens 'context' worden geactiveerd, d.w.z. de gegevens over de persoon die het systeem nodig heeft om te kunnen berekenen wat hij/zij wel en niet mag doen, en welke gegevens van het systeem daarbij mogen worden gebruikt. Om zulke computations te kunnen maken wordt een aantal zaken geregistreerd en aan één persoon gekoppeld.	
Assignment	the registration of an event where either a TText was assigned a value, or its value was deleted	
Claim		
Computation	(for a TText T): a set of Assignments for all TTexts T[i] that are mentioned in the ttTemplate of T, for the purpose of computing a ttValue for T	
Module		
Organization	an organized group of people with a particular purpose, such as a business or government department	
Person	a human body of flesh and blood	
RechtsGrond		
Regeling		

Concept	Meaning	Type
SESSION	een verzameling van gegevens die de context beschrijven waarin één persoon het systeem gebruikt Een persoon gebruikt een gegevensruimte (en heet dan 'user') door met een browser (bijv. Chrome of Firefox) het systeem te benaderen dat de gegegensruimte beheert. Als meerdere personen een gegevensruimte delen, moet het systeem de context van elk van hen kunnen onderscheiden, bijvoorbeeld om: • de interactie 'klein' te houden, d.w.z. alleen gegevens te laten zien die voor hem/haar relevant zijn; • ervoor te zorgen dat een user niet ziet wat hij niet mag zien; • te kunnen bijhouden welke persoon, of welk(e) organ-isatie(onderdeel) verantwoordelijk is voor een zekere transactie; • automatisch gegevens betreffende de user of zijn context aan transacties toe te kunnen voegen We gebruiken de term 'SESSION' of 'session' om de verzameling van gegevens betreffende één (actieve) user mee aan te geven. Deze term correspondeert met de gelijknamige term browsers gebruiken, nl. 'een verbinding (door de browser) met een	

Concept	Meaning	Type
SIAMPersonRefComputation		
Scope	a set of TTexts that are controlled by a single authority and (together) serve a particular purpose a set of TTexts that are controlled by a single authority and (together) serve a particular purpose	
TText	the collection of one template string, one instance thereof, and one value within some scope If people make claims, they do observations, and they will reason about them. In order to decide whether a claim is true, statements are organised in a structure that represents the argument. The concept "statement" is used to represent claims, observations, and all other utterances that can be considered true or false. Statements and Claims are modeled/implemented as TTexts.	
UID		

Other attributes

Concept	Meaning	Type
Aantal		
Aanwijzing		
Actor		
Artikel		
BasisWettenBestand		
Bijlage		
Binding		
Case		
Concept		
Domein		
Evidence		
FirstName		

Concept	Meaning	Type
ISOLevel	a 'level of assurance' as specified in the ISO/IEC 29115:2013 standard	
IdP		
IfcText		
LastName		
LegalGround		
LegalThing		
Lid		
ModuleName	the name (text) by which the module is known	
ModuleVsnMajor	the major version indicator of the module - incremented when changes are not backwards compatible	
ModuleVsnMinor	the minor version indicator of the module - incremented when changes exist that are backwards compatible	
Moment		
OrgAbbrName		
OrgFullName		
OrgRef		
Party		
Password	een rij karakters, die geheim gehouden kan worden door een persoon, en door die persoon gebruikt moet worden om toegang te krijgen tot het systeem Om het moeilijk te maken dat het Account van een zeker persoon door een ander wordt gebruikt, registreert het systeem wachtwoorden. Door een wachtwoord geheim te houden ontstaat enige mate van zekerheid dat het systeem gebruikt word door (dan wel met medeweten van) de persoon op wiens naam het wachtwoord is geregistreerd.	
PersonRef		
PointOfTime	a specific moment on a specific date	
Procesflow		
Regelingsoort		

Concept	Meaning	Type
Role	de naam voor het mogen inzien en/of wijzigen van zekere gegevens, die kan worden toegekend aan accounts en in sessions kan worden geactiveerd Niet iedereen mag alle gegevens uit een systeem inzien en/of wijzigen. Om de beheerslast te beperken die ontstaat als dit soort rechten per persoon wordt uitgegeven, gebruiken we het concept 'Role'. Een Role wordt enerzijds toegekend aan Accounts (en daarmee aan Personen) en anderzijds wordt hij gebruikt om gegevens in te zien en/of te wijzigen. Als een user inlogt worden de Rollen die aan hem zijn toegekend in de session geactiveerd (sessionrollen). Interfaces gebruiken deze sessionrollen om al dan niet gegevens te tonen en/of te editen.	
ScopeDescr	descriptive text or purpose of the Scope	
ScopeID	a short name that helps to refer to the Scope	
Sub		
TTName	a label used to identify a TText (within a scope)	
TTPhrase	a sequence of words	
TTValue	(the representation of) a value of a TText	
Tekst		
Template		
Titel		
URL		

Concept	Meaning	Type
UserID	een rij karakters (bijvoorbeeld het e-mailadres van de user), die een account identificeert binnen het systeem Het UserID (gebruikersnaam) van een account identificeert dat account en impliciet daarmee ook diens eigenaar (d.w.z. de persoon die als enige geacht wordt met dit account in te loggen. Een goed gebruik is om hiervoor een e-mailadres te gebruiken waarop de user van het account bereikbaar is.	

5.3.1 Entity type: *Account*

Als iemand inlogt in het systeem moet diens 'context' worden geactiveerd, d.w.z. de gegevens over de persoon die het systeem nodig heeft om te kunnen berekenen wat hij/zij wel en niet mag doen, en welke gegevens van het systeem daarbij mogen worden gebruikt. Om zulke computations te kunnen maken wordt een aantal zaken geregistreerd en aan één persoon gekoppeld.

This entity type has the following attributes:

Attribute	Type	
Id	Account	Primary key
accOrgRef	OrgRef	Optional
accPersonRef	PersonRef	Optional
accPerson	Person	Optional
accOrg	Organization	Optional
accDeactivateReq	Prop	Optional
accIsActive	Prop	Mandatory
accIsInitialized	Prop	Optional
accPassword	Password	Optional
accUserid	UserID	Mandatory
autoLoginAccount	Prop	Optional

Account has the following associations:

1. sessionAccount (from SESSION to Account).
2. accOrg (from Account to Organization).
3. accPerson (from Account to Person).
4. accAllowedRoles (from Account to Role).

5. accDefaultRoles (from Account to Role).
6. firstSessionAccount (from SESSION to Account).
7. accUID (from Account to UID).
8. scopeOwner (from Scope to Account).
9. ttOwner (from TText to Account).

5.3.2 Entity type: *Assignment*

This entity type has the following attributes:

Attribute	Type	
Id	Assignment	Primary key
asmHasPred	Assignment	Optional
asmPOT	PointOfTime	Optional
asmVal	TTValue	Optional
asmVar	TText	Mandatory

Assignment has the following associations:

1. ttTrace (from TText to Assignment).
2. asmVar (from Assignment to TText).
3. compArg (from Computation to Assignment).

5.3.3 Entity type: *Claim*

This entity type has the following attributes:

Attribute	Type	
Id	Claim	Primary key
object	LegalThing	Optional

Claim has no associations.

5.3.4 Entity type: *Computation*

This entity type has the following attributes:

Attribute	Type	
Id	Computation	Primary key
compRes	TTValue	Optional
compVar	TText	Mandatory

Computation has the following associations:

1. compVar (from Computation to TText).
2. compArg (from Computation to Assignment).

5.3.5 Entity type: *Module*

This entity type has the following attributes:

Attribute	Type	
Id	Module	Primary key
moduleVsnMinor	ModuleVsnMinor	Mandatory
moduleVsnMajor	ModuleVsnMajor	Mandatory
moduleName	ModuleName	Mandatory

Module has no associations.

5.3.6 Entity type: *Organization*

This entity type has the following attributes:

Attribute	Type	
Id	Organization	Primary key
orgFullName	OrgFullName	Optional
orgAbbrName	OrgAbbrName	Optional

Organization has the following associations:

1. accOrg (from Account to Organization).
2. sessionOrg (from SESSION to Organization).
3. personOrg (from Person to Organization).

5.3.7 Entity type: *Person*

This entity type has the following attributes:

Attribute	Type	
Id	Person	Primary key
personRef	PersonRef	Optional
personLastName	LastName	Mandatory
personFirstName	FirstName	Mandatory

Person has the following associations:

1. personOrg (from Person to Organization).
2. accPerson (from Account to Person).
3. sessionPerson (from SESSION to Person).

5.3.8 Entity type: *RechtsGrond*

This entity type has the following attributes:

Attribute	Type	
Id	RechtsGrond	Primary key
aanwijzing	Aanwijzing	Optional
bijlage	Bijlage	Optional
regeling	Regeling	Optional

RechtsGrond has the following associations:

1. url (from RechtsGrond to URL).
2. regeling (from RechtsGrond to Regeling).
3. artikel (from RechtsGrond to Artikel).
4. lid (from RechtsGrond to Lid).
5. sub (from RechtsGrond to Sub).

5.3.9 Entity type: *Regeling*

This entity type has the following attributes:

Attribute	Type	
Id	Regeling	Primary key
afkorting	Tekst	Optional
procesflow	Procesflow	Optional
soort	Regelingsoort	Optional
gedelegeerdUit	Regeling	Optional
artikelen	Aantal	Optional
titel	Titel	Mandatory
bwb	BasisWettenBestand	Optional

Regeling has the following associations:

1. regeling (from RechtsGrond to Regeling).
2. url (from Regeling to URL).
3. regelgeving (from Domein to Regeling).
4. onderwerp (from Regeling to Concept).

5.3.10 Entity type: *SESSION*

Een persoon gebruikt een gegevensruimte (en heet dan 'user') door met een browser (bijv. Chrome of Firefox) het systeem te benaderen dat de gegegensruimte beheert. Als meerdere personen een gegevensruimte delen, moet het systeem de context van elk van hen kunnen onderscheiden, bijvoorbeeld om:

- de interactie 'klein' te houden, d.w.z. alleen gegevens te laten zien die voor hem/haar relevant zijn;
- ervoor te zorgen dat een user niet ziet wat hij niet mag zien;
- te kunnen bijhouden welke persoon, of welk(e) organisatie(onderdeel) verantwoordelijk is voor een zekere transactie;

- automatisch gegevens betreffende de user of zijn context aan transacties toe te kunnen voegen

We gebruiken de term 'SESSION' of 'session' om de verzameling van gegevens betreffende één (actieve) user mee aan te geven. Deze term correspondeert met de gelijknamige term browsers gebruiken, nl. 'een verbinding (door de browser) met een webservice (die een URL heeft)'. Het systeem houdt één session bij voor elke actieve user, d.w.z. voor elke browser die het systeem benadert. Merk op dat dit in het bijzonder geldt als de user in verschillende tabbladen van dezelfde browser het systeem benadert - er is dan toch maar één session (en één user).

This entity type has the following attributes:

Attribute	Type	
Id	SESSION	Primary key
sessionSIA	Prop	Optional
sessionSRI	Prop	Optional
sessionLogoutAssist	Prop	Optional
sessionLoginAssist	Prop	Optional
sessionDev	Prop	Optional
loginISOLevel	ISOLevel	Optional
loginReq	Prop	Optional
registerOrgRef	OrgRef	Optional
registerPassword	Password	Optional
loginCreateAccount	Prop	Optional
sessionOrgRef	OrgRef	Optional
sessionPersonRef	PersonRef	Optional
firstSessionAccount	Account	Optional
sessionPerson	Person	Optional
sessionOrg	Organization	Optional
sessionAuthISOLevel	ISOLevel	Optional
sessionReqdISOLevel	ISOLevel	Optional
logoutRequest	Prop	Optional
loginPassword	Password	Optional
loginUserid	UserID	Optional
sessionUserid	UserID	Optional
sessionAccount	Account	Optional

SESSION has the following associations:

1. sessionAccount (from SESSION to Account).
2. sessionOrg (from SESSION to Organization).
3. sessionPerson (from SESSION to Person).
4. sessionAllowedRoles (from SESSION to Role).
5. sessionActiveRoles (from SESSION to Role).
6. firstSessionAccount (from SESSION to Account).

5.3.11 Entity type: *SIAMPersonRefComputation*

This entity type has the following attributes:

Attribute	Type	
Id	SIAMPersonRefComputation	Primary key
siamcompPersonRef	PersonRef	Optional
siamcompLastName	LastName	Optional
siamcompFirstName	FirstName	Optional

SIAMPersonRefComputation has no associations.

5.3.12 Entity type: *Scope*

This entity type has the following attributes:

Attribute	Type	
Id	Scope	Primary key
scopeIsaCC	Prop	Optional
scopeDescr	ScopeDescr	Optional
scopeOwner	Account	Optional
scopeID	ScopeID	Optional
created	Moment	Optional
owner	Actor	Optional

Scope has the following associations:

1. owner (from Scope to Actor).
2. true (from TText to Scope).
3. false (from TText to Scope).
4. ttScope (from TText to Scope).
5. scopeOwner (from Scope to Account).
6. scopeIII (from Scope to Scope).

5.3.13 Entity type: *TText*

If people make claims, they do observations, and they will reason about them. In order to decide whether a claim is true, statements are organised in a structure that represents the argument. The concept "statement" is used to represent claims, observations, and all other utterances that can be considered true or false. Statements and Claims are modeled/implemented as TTexts.

This entity type has the following attributes:

Attribute	Type	
Id	TText	Primary key
ttOwner	Account	Optional
ttDescr	TTPhrase	Optional
ttICCO	TText	Optional
ttICO	TText	Optional
ttInstance	TTPhrase	Optional
ttTemplate	TTPhrase	Optional
ttValue	TTValue	Optional

Attribute	Type	
ttName	TTName	Optional
ttTrace	Assignment	Mandatory
class	Concept	Optional
observed	Moment	Optional
ttScope	Scope	Mandatory
false	Scope	Optional
true	Scope	Optional
ttInstanceResetReq	Prop	Optional
ttValueUsedToReplacePlaceholders	TTValue	Optional
ttTemplateParsedText	TTPhrase	Optional

TText has the following associations:

1. claimant (from TText to Actor).
2. requires (from TText to TText).
3. ttTemplatePlaceholders (from TText to TTName).
4. ttIsUsedBy (from TText to TText).
5. ttIsUsedByCopy (from TText to TText).
6. ttIsUsedByStar (from TText to TText).
7. ttValueIsUsedInInstanceOfTText (from TText to TText).
8. true (from TText to Scope).
9. false (from TText to Scope).
10. ttScope (from TText to Scope).
11. legalGround (from TText to LegalGround).
12. evidence (from TText to Evidence).
13. ttTrace (from TText to Assignment).
14. asmVar (from Assignment to TText).
15. compVar (from Computation to TText).
16. ttOwner (from TText to Account).

5.3.14 Entity type: *UID*

This entity type has the following attributes:

Attribute	Type	
Id	UID	Primary key
accUID	Account	Optional
uidIssuer	IdP	Optional
uidUserid	UserID	Mandatory

UID has the following associations:

1. accUID (from Account to UID).

5.4 Technical datamodel

The functional requirements have been translated into a technical data model. This model is shown by fig. 5.3.

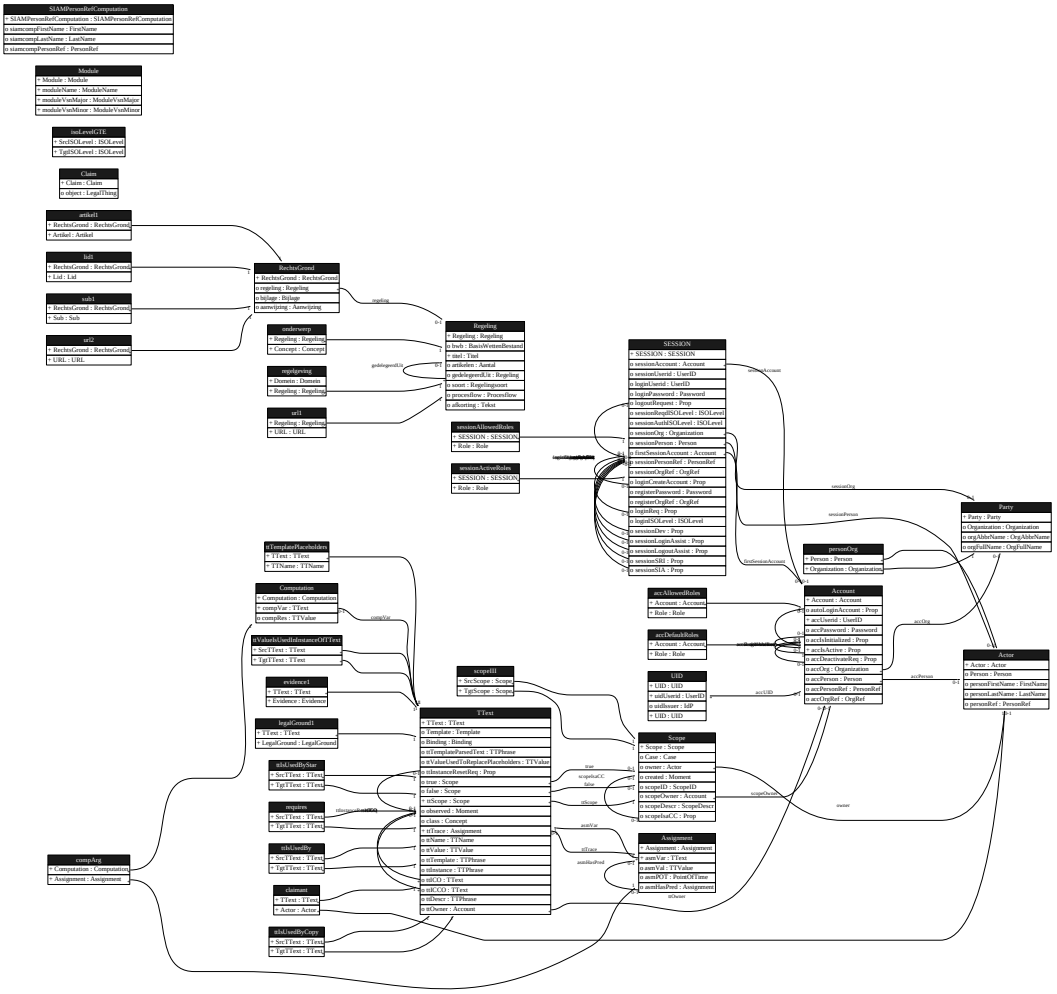

Figure 5.3: Technical data model of MirrorMe

The technical datamodel consists of the following 78 tables:

5.4.1 Table: Aantal

This table has the following 1 attributes:

- **Aantal**
  This attribute implements the identityrelation of *Aantal*.
  `INTEGER`, Mandatory, Unique.

5.4.2 Table: Aanwijzing

This table has the following 1 attributes:

- **Aanwijzing**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.3 Table: accAllowedRoles

This is a link-table, implementing the relation $Account \xrightarrow{accAllowedRoles} Role$. It contains the following columns:

- **Account**
  This attribute is a foreign key to Account
  `OBJECT`, Mandatory.
- **Role**
  This attribute is a foreign key to Role
  `OBJECT`, Mandatory.

5.4.4 Table: accDefaultRoles

This is a link-table, implementing the relation $Account \xrightarrow{accDefaultRoles} Role$. It contains the following columns:

- **Account**
  This attribute is a foreign key to Account
  `OBJECT`, Mandatory.
- **Role**
  This attribute is a foreign key to Role
  `OBJECT`, Mandatory.

5.4.5 Table: Account

This table has the following 11 attributes:

- **Account**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

- **autoLoginAccount**
  This attribute is a foreign key to Account
  `OBJECT`, Optional, Unique.
- **accUserid**
  This attribute is a foreign key to UserID
  `ALPHANUMERIC`, Mandatory, Unique.
- **accPassword**
  This attribute implements the relation $Account \xrightarrow{accPassword} Password$.
  `PASSWORD`, Optional.
- **accIsInitialized**
  This attribute is a foreign key to Account
  `OBJECT`, Optional, Unique.
- **accIsActive**
  This attribute is a foreign key to Account
  `OBJECT`, Mandatory, Unique.
- **accDeactivateReq**
  This attribute is a foreign key to Account
  `OBJECT`, Optional, Unique.
- **accOrg**
  This attribute is a foreign key to Organization
  `OBJECT`, Optional.
- **accPerson**
  This attribute is a foreign key to Person
  `OBJECT`, Optional.
- **accPersonRef**
  This attribute is a foreign key to PersonRef
  `ALPHANUMERIC`, Optional.
- **accOrgRef**
  This attribute is a foreign key to OrgRef
  `OBJECT`, Optional.

5.4.6 Table: Actor

This table has the following 5 attributes:

- **Actor**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **Person**
  This attribute implements the identityrelation of $Person$.
  `OBJECT`, Optional, Unique.
- **personFirstName**
  This attribute is a foreign key to FirstName
  `ALPHANUMERIC`, Optional.
- **personLastName**
  This attribute is a foreign key to LastName

ALPHANUMERIC, Optional.

- **personRef**
  This attribute is a foreign key to PersonRef
  ALPHANUMERIC, Optional.

5.4.7 Table: artikel1

This is a link-table, implementing the relation $RechtsGrond \xrightarrow{artikel} Artikel$. It contains the following columns:

- **RechtsGrond**
  This attribute is a foreign key to RechtsGrond
  OBJECT, Mandatory.
- **Artikel**
  This attribute is a foreign key to Artikel
  OBJECT, Mandatory.

5.4.8 Table: Artikel2

This table has the following 1 attributes:

- **Artikel**
  This attribute is the primary key.
  OBJECT, Mandatory, Unique.

5.4.9 Table: Assignment

This table has the following 5 attributes:

- **Assignment**
  This attribute is the primary key.
  OBJECT, Mandatory, Unique.
- **asmVar**
  This attribute is a foreign key to TText
  OBJECT, Mandatory.
- **asmVal**
  This attribute is a foreign key to TTValue
  ALPHANUMERIC, Optional.
- **asmPOT**
  This attribute is a foreign key to PointOfTime
  DATETIME, Optional.
- **asmHasPred**
  This attribute is a foreign key to Assignment
  OBJECT, Optional, Unique.

5.4.10 Table: BasisWettenBestand

This table has the following 1 attributes:

- **BasisWettenBestand**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.11 Table: Bijlage

This table has the following 1 attributes:

- **Bijlage**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.12 Table: Claim

This table has the following 2 attributes:

- **Claim**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **object**
  This attribute is a foreign key to LegalThing
  `OBJECT`, Optional.

5.4.13 Table: claimant

This is a link-table, implementing the relation $TText \xrightarrow{claimant} Actor$. It contains the following columns:

- **TText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.
- **Actor**
  This attribute is a foreign key to Actor
  `OBJECT`, Mandatory.

5.4.14 Table: compArg

This is a link-table, implementing the relation $Computation \xrightarrow{compArg} Assignment$. It contains the following columns:

- **Computation**
  This attribute is a foreign key to Computation
  `OBJECT`, Mandatory.
- **Assignment**
  This attribute is a foreign key to Assignment
  `OBJECT`, Mandatory.

5.4.15 Table: Computation

This table has the following 3 attributes:

- **Computation**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **compVar**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory, Unique.
- **compRes**
  This attribute is a foreign key to TTValue
  `ALPHANUMERIC`, Optional.

5.4.16 Table: Concept

This table has the following 1 attributes:

- **Concept**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.17 Table: Domein

This table has the following 1 attributes:

- **Domein**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.18 Table: evidence1

This is a link-table, implementing the relation $TText \xrightarrow{evidence} Evidence$. It contains the following columns:

- **TText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.
- **Evidence**
  This attribute is a foreign key to Evidence
  `OBJECT`, Mandatory.

5.4.19 Table: Evidence2

This table has the following 1 attributes:

- **Evidence**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.20 Table: FirstName

This table has the following 1 attributes:

- **FirstName**
  This attribute implements the identityrelation of $FirstName$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.21 Table: IdP

This table has the following 1 attributes:

- **IdP**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.22 Table: IfcText

This table has the following 1 attributes:

- **IfcText**
  This attribute implements the identityrelation of $IfcText$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.23 Table: ISOLevel

This table has the following 1 attributes:

- **ISOLevel**
  This attribute implements the identityrelation of $ISOLevel$.
  `INTEGER`, Mandatory, Unique.

5.4.24 Table: isoLevelGTE

This is a link-table, implementing the relation $ISOLevel \xrightarrow{isoLevelGTE} ISOLevel$.
It contains the following columns:

- **SrcISOLevel**
  This attribute is a foreign key to ISOLevel
  `INTEGER`, Mandatory.
- **TgtISOLevel**
  This attribute is a foreign key to ISOLevel
  `INTEGER`, Mandatory.

5.4.25 Table: LastName

This table has the following 1 attributes:

- **LastName**
  This attribute implements the identityrelation of $LastName$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.26 Table: legalGround1

This is a link-table, implementing the relation $TText \xrightarrow{legalGround} LegalGround$. It contains the following columns:

- **TText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.
- **LegalGround**
  This attribute is a foreign key to LegalGround
  `OBJECT`, Mandatory.

5.4.27 Table: LegalGround2

This table has the following 1 attributes:

- **LegalGround**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.28 Table: LegalThing

This table has the following 1 attributes:

- **LegalThing**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.29 Table: lid1

This is a link-table, implementing the relation $RechtsGrond \xrightarrow{lid} Lid$. It contains the following columns:

- **RechtsGrond**
  This attribute is a foreign key to RechtsGrond
  `OBJECT`, Mandatory.
- **Lid**
  This attribute is a foreign key to Lid
  `OBJECT`, Mandatory.

5.4.30 Table: Lid2

This table has the following 1 attributes:

- **Lid**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.31 Table: Module

This table has the following 4 attributes:

- **Module**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **moduleName**
  This attribute is a foreign key to ModuleName
  `ALPHANUMERIC`, Mandatory.
- **moduleVsnMajor**
  This attribute is a foreign key to ModuleVsnMajor
  `INTEGER`, Mandatory.
- **moduleVsnMinor**
  This attribute is a foreign key to ModuleVsnMinor
  `INTEGER`, Mandatory.

5.4.32 Table: ModuleName

This table has the following 1 attributes:

- **ModuleName**
  This attribute implements the identityrelation of $ModuleName$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.33 Table: ModuleVsnMajor

This table has the following 1 attributes:

- **ModuleVsnMajor**
  This attribute implements the identityrelation of $ModuleVsnMajor$.
  `INTEGER`, Mandatory, Unique.

5.4.34 Table: ModuleVsnMinor

This table has the following 1 attributes:

- **ModuleVsnMinor**
  This attribute implements the identityrelation of $ModuleVsnMinor$.
  `INTEGER`, Mandatory, Unique.

5.4.35 Table: Moment

This table has the following 1 attributes:

- **Moment**
  This attribute implements the identityrelation of $Moment$.
  `DATETIME`, Mandatory, Unique.

5.4.36 Table: onderwerp

This is a link-table, implementing the relation $Regeling \xrightarrow{onderwerp} Concept$. It contains the following columns:

- **Regeling**
  This attribute is a foreign key to Regeling
  `OBJECT`, Mandatory.
- **Concept**
  This attribute is a foreign key to Concept
  `OBJECT`, Mandatory.

5.4.37 Table: ONE

This table has the following 1 attributes:

- **ONE**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.38 Table: OrgAbbrName

This table has the following 1 attributes:

- **OrgAbbrName**
  This attribute implements the identityrelation of $OrgAbbrName$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.39 Table: OrgFullName

This table has the following 1 attributes:

- **OrgFullName**
  This attribute implements the identityrelation of $OrgFullName$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.40 Table: OrgRef

This table has the following 1 attributes:

- **OrgRef**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.41 Table: Party

This table has the following 4 attributes:

- **Party**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **Organization**
  This attribute implements the identityrelation of *Organization*.
  `OBJECT`, Optional, Unique.
- **orgAbbrName**
  This attribute is a foreign key to OrgAbbrName
  `ALPHANUMERIC`, Optional, Unique.
- **orgFullName**
  This attribute is a foreign key to OrgFullName
  `ALPHANUMERIC`, Optional, Unique.

5.4.42 Table: Password

This table has the following 1 attributes:

- **Password**
  This attribute implements the identityrelation of *Password*.
  `PASSWORD`, Mandatory, Unique.

5.4.43 Table: personOrg

This is a link-table, implementing the relation $Person \xrightarrow{personOrg} Organization$.
It contains the following columns:

- **Person**
  This attribute is a foreign key to Person
  `OBJECT`, Mandatory.
- **Organization**
  This attribute is a foreign key to Organization
  `OBJECT`, Mandatory.

5.4.44 Table: PersonRef

This table has the following 1 attributes:

- **PersonRef**
  This attribute implements the identityrelation of $PersonRef$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.45 Table: PointOfTime

This table has the following 1 attributes:

- **PointOfTime**
  This attribute implements the identityrelation of $PointOfTime$.
  `DATETIME`, Mandatory, Unique.

5.4.46 Table: Procesflow

This table has the following 1 attributes:

- **Procesflow**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.47 Table: RechtsGrond

This table has the following 4 attributes:

- **RechtsGrond**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **regeling**
  This attribute is a foreign key to Regeling
  `OBJECT`, Optional.
- **bijlage**
  This attribute is a foreign key to Bijlage
  `OBJECT`, Optional.
- **aanwijzing**
  This attribute is a foreign key to Aanwijzing
  `OBJECT`, Optional.

5.4.48 Table: regelgeving

This is a link-table, implementing the relation $Domein \xrightarrow{regelgeving} Regeling$. It contains the following columns:

- **Domein**
  This attribute is a foreign key to Domein
  `OBJECT`, Mandatory.
- **Regeling**
  This attribute is a foreign key to Regeling
  `OBJECT`, Mandatory.

5.4.49 Table: Regeling

This table has the following 8 attributes:

- **Regeling**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **bwb**
  This attribute is a foreign key to BasisWettenBestand
  `OBJECT`, Optional.

- **titel**
  This attribute is a foreign key to Titel
  `OBJECT`, Mandatory.
- **artikelen**
  This attribute is a foreign key to Aantal
  `INTEGER`, Optional.
- **gedelegeerdUit**
  This attribute is a foreign key to Regeling
  `OBJECT`, Optional.
- **soort**
  This attribute is a foreign key to Regelingsoort
  `OBJECT`, Optional.
- **procesflow**
  This attribute is a foreign key to Procesflow
  `OBJECT`, Optional.
- **afkorting**
  This attribute is a foreign key to Tekst
  `OBJECT`, Optional.

5.4.50 Table: Regelingsoort

This table has the following 1 attributes:

- **Regelingsoort**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.51 Table: requires

This is a link-table, implementing the relation $TText \xrightarrow{requires} TText$. It contains the following columns:

- **SrcTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.
- **TgtTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.

5.4.52 Table: Role

This table has the following 1 attributes:

- **Role**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.53 Table: Scope

This table has the following 8 attributes:

- **Scope**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **Case**
  This attribute implements the identityrelation of *Case*.
  `OBJECT`, Optional, Unique.
- **owner**
  This attribute is a foreign key to Actor
  `OBJECT`, Optional.
- **created**
  This attribute is a foreign key to Moment
  `DATETIME`, Optional.
- **scopeID**
  This attribute is a foreign key to ScopeID
  `ALPHANUMERIC`, Optional.
- **scopeOwner**
  This attribute is a foreign key to Account
  `OBJECT`, Optional.
- **scopeDescr**
  This attribute is a foreign key to ScopeDescr
  `ALPHANUMERIC`, Optional.
- **scopeIsaCC**
  This attribute is a foreign key to Scope
  `OBJECT`, Optional, Unique.

5.4.54 Table: ScopeDescr

This table has the following 1 attributes:

- **ScopeDescr**
  This attribute implements the identityrelation of *ScopeDescr*.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.55 Table: ScopeID

This table has the following 1 attributes:

- **ScopeID**
  This attribute implements the identityrelation of *ScopeID*.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.56 Table: scopeIII

This is a link-table, implementing the relation $Scope \xrightarrow{scopeIII} Scope$. It contains the following columns:

- **SrcScope**
  This attribute is a foreign key to Scope
  `OBJECT`, Mandatory.
- **TgtScope**
  This attribute is a foreign key to Scope
  `OBJECT`, Mandatory.

5.4.57 Table: SESSION

This table has the following 23 attributes:

- **SESSION**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **sessionAccount**
  This attribute is a foreign key to Account
  `OBJECT`, Optional.
- **sessionUserid**
  This attribute is a foreign key to UserID
  `ALPHANUMERIC`, Optional.
- **loginUserid**
  This attribute is a foreign key to UserID
  `ALPHANUMERIC`, Optional.
- **loginPassword**
  This attribute implements the relation $SESSION \xrightarrow{loginPassword} Password$.
  `PASSWORD`, Optional.
- **logoutRequest**
  This attribute is a foreign key to SESSION
  `OBJECT`, Optional, Unique.
- **sessionReqdISOLevel**
  This attribute is a foreign key to ISOLevel
  `INTEGER`, Optional.
- **sessionAuthISOLevel**
  This attribute is a foreign key to ISOLevel
  `INTEGER`, Optional.
- **sessionOrg**
  This attribute is a foreign key to Organization
  `OBJECT`, Optional.
- **sessionPerson**
  This attribute is a foreign key to Person
  `OBJECT`, Optional.

- **firstSessionAccount**
  This attribute is a foreign key to Account
  `OBJECT`, Optional.
- **sessionPersonRef**
  This attribute is a foreign key to PersonRef
  `ALPHANUMERIC`, Optional.
- **sessionOrgRef**
  This attribute is a foreign key to OrgRef
  `OBJECT`, Optional.
- **loginCreateAccount**
  This attribute is a foreign key to SESSION
  `OBJECT`, Optional, Unique.
- **registerPassword**
  This attribute implements the relation $SESSION \xrightarrow{registerPassword} Password$.
  `PASSWORD`, Optional.
- **registerOrgRef**
  This attribute is a foreign key to OrgRef
  `OBJECT`, Optional.
- **loginReq**
  This attribute is a foreign key to SESSION
  `OBJECT`, Optional, Unique.
- **loginISOLevel**
  This attribute is a foreign key to ISOLevel
  `INTEGER`, Optional.
- **sessionDev**
  This attribute is a foreign key to SESSION
  `OBJECT`, Optional, Unique.
- **sessionLoginAssist**
  This attribute is a foreign key to SESSION
  `OBJECT`, Optional, Unique.
- **sessionLogoutAssist**
  This attribute is a foreign key to SESSION
  `OBJECT`, Optional, Unique.
- **sessionSRI**
  This attribute is a foreign key to SESSION
  `OBJECT`, Optional, Unique.
- **sessionSIA**
  This attribute is a foreign key to SESSION
  `OBJECT`, Optional, Unique.

5.4.58 Table: sessionActiveRoles

This is a link-table, implementing the relation $SESSION \xrightarrow{sessionActiveRoles} Role$.
It contains the following columns:

- **SESSION**
  This attribute is a foreign key to SESSION
  `OBJECT`, Mandatory.
- **Role**
  This attribute is a foreign key to Role
  `OBJECT`, Mandatory.

5.4.59 Table: sessionAllowedRoles

This is a link-table, implementing the relation $SESSION \xrightarrow{sessionAllowedRoles} Role$.
It contains the following columns:

- **SESSION**
  This attribute is a foreign key to SESSION
  `OBJECT`, Mandatory.
- **Role**
  This attribute is a foreign key to Role
  `OBJECT`, Mandatory.

5.4.60 Table: SIAMPersonRefComputation

This table has the following 4 attributes:

- **SIAMPersonRefComputation**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **siamcompFirstName**
  This attribute is a foreign key to FirstName
  `ALPHANUMERIC`, Optional.
- **siamcompLastName**
  This attribute is a foreign key to LastName
  `ALPHANUMERIC`, Optional.
- **siamcompPersonRef**
  This attribute is a foreign key to PersonRef
  `ALPHANUMERIC`, Optional.

5.4.61 Table: sub1

This is a link-table, implementing the relation $RechtsGrond \xrightarrow{sub} Sub$. It contains the following columns:

- **RechtsGrond**
  This attribute is a foreign key to RechtsGrond
  `OBJECT`, Mandatory.
- **Sub**
  This attribute is a foreign key to Sub
  `OBJECT`, Mandatory.

5.4.62 Table: Sub2

This table has the following 1 attributes:

- **Sub**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.63 Table: Tekst

This table has the following 1 attributes:

- **Tekst**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.64 Table: Titel

This table has the following 1 attributes:

- **Titel**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.65 Table: TText

This table has the following 20 attributes:

- **TText**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **Template**
  This attribute implements the identityrelation of $Template$.
  `OBJECT`, Optional, Unique.
- **Binding**
  This attribute implements the identityrelation of $Binding$.
  `OBJECT`, Optional, Unique.
- **ttTemplateParsedText**
  This attribute implements the relation $TText \xrightarrow{ttTemplateParsedText} TTPhrase$.
  `BIGALPHANUMERIC`, Optional.
- **ttValueUsedToReplacePlaceholders**
  This attribute is a foreign key to TTValue
  `ALPHANUMERIC`, Optional.
- **ttInstanceResetReq**
  This attribute is a foreign key to TText
  `OBJECT`, Optional, Unique.
- **true**
  This attribute is a foreign key to Scope
  `OBJECT`, Optional.

- **false**
  This attribute is a foreign key to Scope
  `OBJECT`, Optional.
- **ttScope**
  This attribute is a foreign key to Scope
  `OBJECT`, Mandatory.
- **observed**
  This attribute is a foreign key to Moment
  `DATETIME`, Optional.
- **class**
  This attribute is a foreign key to Concept
  `OBJECT`, Optional.
- **ttTrace**
  This attribute is a foreign key to Assignment
  `OBJECT`, Mandatory, Unique.
- **ttName**
  This attribute is a foreign key to TTName
  `ALPHANUMERIC`, Optional.
- **ttValue**
  This attribute is a foreign key to TTValue
  `ALPHANUMERIC`, Optional.
- **ttTemplate**
  This attribute implements the relation $TText \xrightarrow{ttTemplate} TTPhrase$.
  `BIGALPHANUMERIC`, Optional.
- **ttInstance**
  This attribute implements the relation $TText \xrightarrow{ttInstance} TTPhrase$.
  `BIGALPHANUMERIC`, Optional.
- **ttICO**
  This attribute is a foreign key to TText
  `OBJECT`, Optional.
- **ttICCO**
  This attribute is a foreign key to TText
  `OBJECT`, Optional.
- **ttDescr**
  This attribute implements the relation $TText \xrightarrow{ttDescr} TTPhrase$.
  `BIGALPHANUMERIC`, Optional.
- **ttOwner**
  This attribute is a foreign key to Account
  `OBJECT`, Optional.

5.4.66 Table: ttIsUsedBy

This is a link-table, implementing the relation $TText \xrightarrow{ttIsUsedBy} TText$. It contains the following columns:

- **SrcTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.
- **TgtTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.

5.4.67 Table: ttIsUsedByCopy

This is a link-table, implementing the relation $TText \xrightarrow{ttIsUsedByCopy} TText$. It contains the following columns:

- **SrcTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.
- **TgtTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.

5.4.68 Table: ttIsUsedByStar

This is a link-table, implementing the relation $TText \xrightarrow{ttIsUsedByStar} TText$. It contains the following columns:

- **SrcTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.
- **TgtTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.

5.4.69 Table: TTName

This table has the following 1 attributes:

- **TTName**
  This attribute implements the identityrelation of $TTName$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.70 Table: TTPhrase

This table has the following 1 attributes:

- **TTPhrase**
  This attribute implements the identityrelation of $TTPhrase$.
  `BIGALPHANUMERIC`, Mandatory, Unique.

5.4.71 Table: ttTemplatePlaceholders

This is a link-table, implementing the relation $TText \xrightarrow{ttTemplatePlaceholders} TTName$.
It contains the following columns:

- **TText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.
- **TTName**
  This attribute is a foreign key to TTName
  `ALPHANUMERIC`, Mandatory.

5.4.72 Table: TTValue

This table has the following 1 attributes:

- **TTValue**
  This attribute implements the identityrelation of $TTValue$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.4.73 Table: ttValueIsUsedInInstanceOfTText

This is a link-table, implementing the relation $TText \xrightarrow{ttValueIsUsedInInstanceOfTText} TText$.
It contains the following columns:

- **SrcTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.
- **TgtTText**
  This attribute is a foreign key to TText
  `OBJECT`, Mandatory.

5.4.74 Table: UID

This table has the following 4 attributes:

- **UID**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.
- **uidUserid**
  This attribute is a foreign key to UserID
  `ALPHANUMERIC`, Mandatory.
- **uidIssuer**
  This attribute is a foreign key to IdP
  `OBJECT`, Optional.
- **accUID**
  This attribute is a foreign key to Account
  `OBJECT`, Optional.

5.4.75 Table: url1

This is a link-table, implementing the relation $Regeling \xrightarrow{url} URL$. It contains the following columns:

- **Regeling**
  This attribute is a foreign key to Regeling
  `OBJECT`, Mandatory.
- **URL**
  This attribute is a foreign key to URL
  `OBJECT`, Mandatory.

5.4.76 Table: url2

This is a link-table, implementing the relation $RechtsGrond \xrightarrow{url} URL$. It contains the following columns:

- **RechtsGrond**
  This attribute is a foreign key to RechtsGrond
  `OBJECT`, Mandatory.
- **URL**
  This attribute is a foreign key to URL
  `OBJECT`, Mandatory.

5.4.77 Table: URL3

This table has the following 1 attributes:

- **URL**
  This attribute is the primary key.
  `OBJECT`, Mandatory, Unique.

5.4.78 Table: UserID

This table has the following 1 attributes:

- **UserID**
  This attribute implements the identityrelation of $UserID$.
  `ALPHANUMERIC`, Mandatory, Unique.

5.5 Logical data model

Concept	C	R	U	D
Case		Case		
Case		Cases		
Artikel		Rechtsgrond		
Template		Case		
ScopeID		Cases		

Concept	C	R	U	D
Scope		Case		
Scope		Cases		
Scope		Garbage		
OrgRef		Login/Register		
OrgRef		ShowAcc		
URL		Rechtsgrond		
RechtsGrond		Rechtsgrond		
ISOLevel		Login/Register		
IfcText		Login/Register		
TTValue		Case		
LegalGround		Case		
LegalGround		Garbage		
Regeling		Rechtsgrond		
Evidence		Garbage		
Lid		Rechtsgrond		
Bijlage		Rechtsgrond		
TTName		Case		
Moment		Cases		
Moment		Garbage		
PersonRef		Login/Register		
PersonRef		ShowAcc		
Role		SessionRoles		
TTPhrase		Case		
TTPhrase		Garbage		
Aanwijzing		Rechtsgrond		
Password		Login/Register		
Sub		Rechtsgrond		