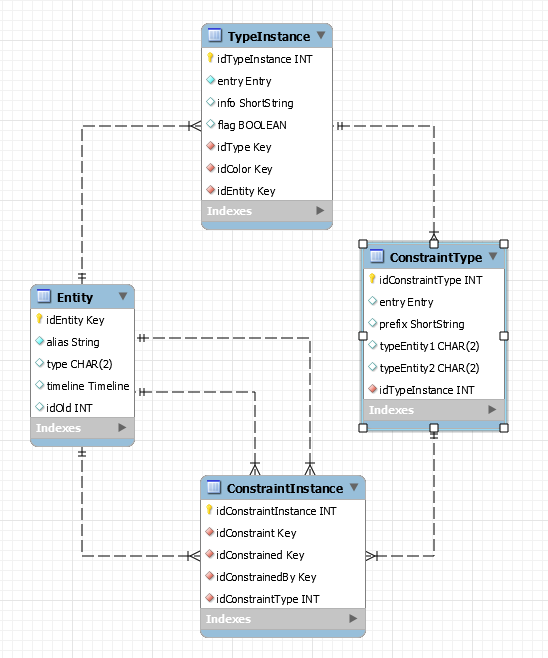
FNBr - Constraints

# Notes on the Constraints Data Structure - 20181209

* This note uses an AVM notation. In this context, a **Matrix** is a Frame/Construction, an **Attribute** is a FrameElement/ConstructionElement and a **Value** is an Entity from FNBr.
* A **Constraint** is a relation between Entity1 and Entity2 whose purpose is to provide added evidence for the semantic interpretation of Entity1.
* A **Constraint** represents two kinds of information:
  + *Relations* that must be hold between Attributes. Constraints as relations are applied to the Matrix.
  + *Values* for an Attribute. Constraints as values are applied to the Attribute.
* A **Constraint** has a specific **ConstraintType**. A specific application of a constraint is a **ConstraintInstance**.

Relational structure:



* A **Constraint** is represented as an Entity. They are registered only as a record in the **Entity** table (with type = CN). There is no specific table for constraints. The **idEntity** is used in the **ConstraintInstance** table to establish the relation between the *constrained entity* and the *constrainedBy entity*. In **ConstraintInstance**:
  + idConstraint: idEntity of Constraint (in Entity table)
  + idConstrained: idEntity of constrained entity – a Matrix or Attribute
  + idConstrainedBy: idEntity of constraining entity
* It is possible for a constraint to be (recursively) constrained. The chain of constraints is called *Constraint Set*. In this case, we have the following in the **ConstraintInstance** table:
  + idConstraint: idEntity of Constraint (in Entity table)
  + idConstrained: idEntity of constrained Constraint
  + idConstrainedBy: idEntity of constraining entity

## Constraints

### Types of constraints

|  |  |
| --- | --- |
| entry | info |
| typ\_con\_value | Value |
| typ\_con\_relation | Relation |

### Constraint Types (Relations)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| entry | prefix | typeEntity1 | typeEntity2 | Semantic |
| con\_constraint | cnn | \*\* | CN | A constraint is constrained by another constraint |
| con\_before | bef | CE | CE | An Attribute precedes another in the same Matrix |
| con\_meets | mee | CE | CE | An Attribute precedes another immediately in the same Matrix |
| con\_follows | flw | CE | CE | An Attribute follows another immediately in the same Matrix |
| con\_same | sam | CE | CE | An Attribute has the same words of another Attribute |
| con\_different | dif | CE | CE | All words of an Attribute are different from words of another Attribute |
| con\_dominance | dom | CE | CE | The head word of an Attribute is dominated (in Dependency Parser sense) by the head word of another attribute |
| con\_hasword | wrd | CE | CE | An Attribute has at least one word that is present in another Attribute |
| con\_and | and | CE | CE | Both Attributes must be hold to Matrix be valid |
| con\_xor | xor | CE | CE | One Attribute and not the other must be hold to Matrix be valid |
| con\_evokes | evk | CN | \*\* | Some word in the Attribute must evoke a specific Frame |
| con\_qualia\_formal | fml | CE | CE | An Attribute is related to another via a Formal Qualia |
| con\_qualia\_constitutive | cst | CE | CE | An Attribute is related to another via a Constitutive Qualia |
| con\_qualia\_agentive | agt | CE | CE | An Attribute is related to another via a Agentive Qualia |
| con\_qualia\_telic | tlc | CE | CE | An Attribute is related to another via a Telic Qualia |

### Constraint Types (Values)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| entry | prefix | typeEntity1 | typeEntity2 | Semantic |
| con\_cxn | cxn | CE | CX | A Attribute has a Cxn as value |
| con\_element | ele | CN | CE | A Constraint is constrained by an Attribute |
| con\_frame | frm | FE | FR | An Attribute has a Frame as value |
| con\_framefamily | fam | CE | FR | An Attribute slot is filled by Lexical Units evoking a Frame from a Frame Family (the inheritance network of a Frame) |
| con\_lemma | lem | CE | LM | A Attribute has a Lemma as value |
| con\_lexeme | lex | CE | LX | A Attribute has a Lexeme as value |
| con\_lu | lu | CE | LU | A Attribute has a LU as value |
| con\_semtype | sty | \*\* | ST | An Entity is mapped to a Semantic (Ontological) Type |
| con\_stlu | slu | CE | ST | The value of an Attribute must be of a specific LU SemanticType (biframal lu, bound lu, etc.) |
| con\_udfeature | udf | CE | UF | The value of an Attribute is restricted to a specific UD Feature (from TypeInstance table) |
| con\_udrelation | udr | CE | UR | The value of an Attribute is restricted to a specific UD Relation (from UDRelation table) |
| con\_wordform | wrf | CE | WF | A Attribute has a Wordform as value |

### Example

* the number inside brackets corresponds to idEntity

cxn:NP [*idCxn: 2000*]

ce:N [*idCe: 2001*]

cxn: VPComp *[idCxn: 1000*]

ce:Verb [*idCe: 1001*]

ce:Complement [*idCe: 1002*]

cxn:NP [*idConstraint: 4010, idConstrained: 1002, idConstrainedBy (CX): 2000*]

ele:N [*idConstraint:4011, idConstrained: 4010, idConstrainedBy (CE): 2001*]

cxn:SplitArgument [*idCxn:3000*]

ce:Subject [*idCe: 3001*]

cxn:NP [*idConstraint: 4000, idConstrained: 3001, idConstrainedBy(CX): 2000*]

ele:N [*idConstraint:4013, idConstrainted: 4000, idConstrainedBy(CE): 2001*]

bef:Predicate [*idConstraint: 4001, idConstrained: 3001, idConstrainedBY(CE): 3002*]

ce:Predicate [*idCe: 3002*]

cxn:VPComp [*idConstraint: 4002, idConstrained: 3002, idConstrainedBy(CX): 1000*]

ele:Verb [*idConstraint:4003, idConstrained: 4002, idConstrainedBy(CE): 1001*]

ele:Complement [*idConstraint:4004, idConstrained: 4002, idConstrainedBy(CE): 1002*]

cxn:NP [*idConstraint:4005, idConstrained: 4004, idConstrainedBy(CX): 2000*]

ele:N [*idConstraint:4012, idConstrained: 4005, idConstrainedBy(CE): 2001*]

cnt:Constitutive\_qualia [*idConstraint: 4007, idConstrained: 4013, idConstrainedBy(CN): 4012*]

ele:N [*idConstraint: 4013*]

ele:N [*idConstraint: 4012*]

Here, we have three constructions:

1) NP, with one CE (N)

2) VPComp, with two CEs (Verb and Complement). The CE Complement must be a NP.

3) SplitArgument, with two CEs (Subject and Predicate). CE Subject must be a NP and it must precede CE Predicate. CE Predicate must be a VPComp. This construction has a constraint stating that a qualia\_constitutive relation must hold between the Subject Noun and the Complement Noun of VPComp.

\* It's worth noting that while some constraints are added automatically by the system (e.g. the CEs for a construction used as constraints for a CE - the constraint 'ele' at figure) others must be annotated manually by the user (e.g. the association of a construction to a CE and the constitutive qualia relation between the nouns).

The following will be recorded in the database in the **ConstraintInstance** table for the constructions above (here only the relations concerning constraints are shown):

|  |  |  |  |
| --- | --- | --- | --- |
| RelationType | IdConstraint | IdConstrained | IdConstrainedBy |
| con\_cxn | 4010 | 1002 | 2000 |
| con\_element | 4011 | 4010 | 2001 |
| con\_cxn | 4000 | 3001 | 2000 |
| con\_cxn | 4002 | 3002 | 1000 |
| con\_element | 4013 | 4000 | 2001 |
| con\_before | 4001 | 3001 | 3002 |
| con\_element | 4003 | 4002 | 1001 |
| con\_element | 4004 | 4002 | 1002 |
| con\_cxn | 4005 | 4004 | 2000 |
| con\_element | 4012 | 4005 | 2001 |
| con\_qualia\_constitutive | 4015 | 4013 | 4012 |
| con\_constraint | 4007 | 3000 | 4015 |