# Supplementary Materials for: Raw Ontological Model Driven by Institutional Grammar – Attempt To Introduce A New Subclass of Ontologies for Policy Design Studies

Anna Wróblewska[1][0000−0002−3407−7570], Bartosz Pieliński[2][0000−0003−2664−2135], Karolina Seweryn[1][0000−0003−0617−7301], Sylwia Sysko-Romańczuk[1][0000−0001−5960−8131], Stanisław Giziński[1][0000−0002−3695−9809], Karol Saputa[1][0000−0002−8809−5248], and Aleksandra Wichrowska[1]

[1] Warsaw University of Technology, Warsaw, Poland
{anna.wroblewska1}@pw.edu.pl
[2] University of Warsaw,
Faculty of Political Science and International Studies, Warsaw, Poland
{b.pielinski}@uw.edu.pl

This document is a supplement to our paper. Here we describe in more detail our algorithms. Section 1 lists our contributed resources. Then, we present an annotation file, also available in the resources (see Section 2). Section 4 and Section 5 describe in detail algorithms for automatic tagger of Institutional Grammar and Raw Ontology Builder, respectively.

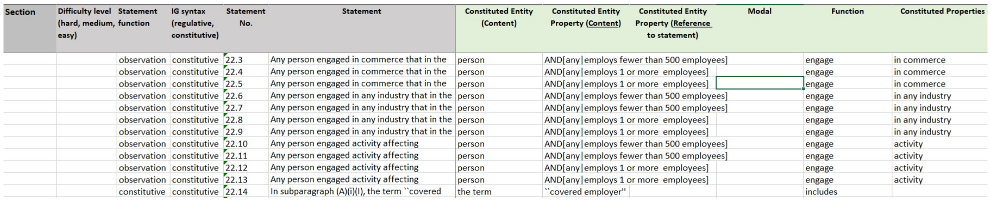
## 1 Resources Description

The following software resources make up our workflow:

1. Legal Acts annotated with Institutional Grammar and examplary ontology of COVID-19 sick leaves
`https://github.com/institutional-grammar-pl/Raw-Ontological-Model-Driven-by-Institutional-Grammar`
2. Dataset – COVID-19 use case
`https://github.com/institutional-grammar-pl/ig-annotations`
3. Institutional Grammar tagger
`https://github.com/institutional-grammar-pl/policydemic-annotator`
4. IG-based ontology builder
`https://github.com/institutional-grammar-pl/ig-ontology-builder`

## 2 Annotation Format

Manual annotations of legal acts with Institutional Grammar are prepared in Excel spreadsheet format. The format is easy for manual use and collaboration. This columnar format is also easy to process by software. As an additional resource, we include the file with the IG annotations of the "COVID-19 seek leaves" act.



| Section | Difficulty level (hard, medium, easy) | Statement function | IG syntax (regulative, constitutive) | Statement No. | Statement | Constituted Entity (Content) | Constituted Entity Property (Content) | Constituted Entity Property (Reference to statement) | Modal | Function | Constituted Properties |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | observation | constitutive | 22.3 | Any person engaged in commerce that in the | person | AND[any\|employs fewer than 500 employees] | | | engage | in commerce |
| | | observation | constitutive | 22.4 | Any person engaged in commerce that in the | person | AND[any\|employs 1 or more employees] | | | engage | in commerce |
| | | observation | constitutive | 22.5 | Any person engaged in commerce that in the | person | AND[any\|employs 1 or more employees] | | | engage | in commerce |
| | | observation | constitutive | 22.6 | Any person engaged in any industry that in the | person | AND[any\|employs fewer than 500 employees] | | | engage | in any industry |
| | | observation | constitutive | 22.7 | Any person engaged in any industry that in the | person | AND[any\|employs fewer than 500 employees] | | | engage | in any industry |
| | | observation | constitutive | 22.8 | Any person engaged in any industry that in the | person | AND[any\|employs 1 or more employees] | | | engage | in any industry |
| | | observation | constitutive | 22.9 | Any person engaged in any industry that in the | person | AND[any\|employs 1 or more employees] | | | engage | in any industry |
| | | observation | constitutive | 22.10 | Any person engaged activity affecting | person | AND[any\|employs fewer than 500 employees] | | | engage | activity |
| | | observation | constitutive | 22.11 | Any person engaged activity affecting | person | AND[any\|employs fewer than 500 employees] | | | engage | activity |
| | | observation | constitutive | 22.12 | Any person engaged activity affecting | person | AND[any\|employs 1 or more employees] | | | engage | activity |
| | | observation | constitutive | 22.13 | Any person engaged activity affecting | person | AND[any\|employs 1 or more employees] | | | engage | activity |
| | | constitutive | constitutive | 22.14 | In subparagraph (A)(i)(I), the term ``covered | the term | ``covered employer'' | | | includes | |

**Fig. 1.** A part of an annotated file with columns describing elements of sentence by Institutional Grammar.

## 3 Atomic Statement Classification

## 4 IG Tagger

The atomic institutional statements can be divided into two groups: regulative and constitutive. We built the tagger consisting of rules specific to mentioned types of sentences. It tags words based on their characteristics. Each word in the statement gets annotation containing lemma, part of speech tag, morphological features, and relation to other words (StanfordNLP package [2]).

Tags referring to each statement type are presented in our paper in Section on IG and Table 1.

| Regulative statements | Description | Constitutive statements | Description |
|---|---|---|---|
| Attribute | The addressee of the statement. | Constituted Entity | The entity being defined. |
| Aim | The action of addressee regulated by the statement. | Constitutive Function | A verb used to define Constituted Entity. |
| Deontic | An operator determining level of discretion or constraint associated with Aim. | Modal | An operator determining level of necessity and possibility of defining Constituted Entity. |
| Object | The receiver of the action described by Aim | Constituting Properties. | The entity against which Constituted Entity is defined. |
| Activation Condition | The setting to which the statements applies. | Activation Condition | The setting to which the statements applies. |
| Execution Constraint | Quality of action described by Aim | Execution Constraint | Quality of Constitutive Function. |

**Table 1.** IG main components depending on statement type (regulative or constitutive) based on [1, pp. 10-11].

### 4.1 Rules for Regulative Statements

Function *annotate_ all_ descendants(node, tag)* assigns *tag* to all descendants of *node*. Algorithms 1, 3, 2 shows rules of tagger to find aim, attribute, and deontic tag, respectively. Algorithm 4 (and points listed below) presents a way of annotating object and context.

**Algorithm 1:** AIM tagger.

```
Data: tree: LexicalTreeNode, root: tree.root, tags: list of tags
tags.append(AIM(root))
for child in root.children do
    if child.relation in ("aux:pass", "cop") then
        tags.append(AIM(child))
    if child.relation = "aux" and child.lemm in ("be", "have", "do") then
        tags.append(AIM(chid))
    if child.relation = "advmod" and child.lemm = "not" then
        tags.append(AIM(chid))
```

**Algorithm 2:** DEONTIC tagger.

```
Data: tree: LexicalTreeNode, root: tree.root, tags: list of tags
for child in root.children do
    if child.relation = "aux" and child.lemm in ("must", "should", "may",
      "might", "can", "could", "need", "ought", "shall") then
        tags.append(DEONTIC(child))
```

**Algorithm 3:** ATTRIBUTE and ATTRIBUTE_PROPERTY tagger.

```
Data: tree: LexicalTreeNode, root: tree.root, tags: list of tags
for child in tree.children do
    if child.relation in("nsubj", "nsubj:pass") then
        for desc in child.descendants do
            if desc.relation="det" and desc.parent=child.parent then
                tags.append(ATTR(desc))
            else
                tags.append(ATTR_PROP(desc))
    else if child.relation="conj" then
        annotate_all_descendants(desc, ATTR_PROP)
```

**Algorithm 4:** OBJECT AND CONTEXT tagger.

```
Data: tree: LexicalTreeNode, root: tree.root, tags: list of tags
for child in tree.children do
    if child.relation == "obj" then
        tags.append(OBJECT(child))
        for desc in child.children do
            if desc.relation="advcl" then
                annotate_all_descendants(desc, CONTEXT)
            else if desc.relation in ("det", "amod", "case", "compound")
             then
                tags.append(OBJECT(desc))
                annotate_all_descendants(desc, OBJECT_PROP)
            else if desc.relation in (nmod", "nmod:poss") then
                tags.append(OBJECT(desc))
                for child_desc in desc.descendants do
                    if child_desc.relation in ("case", "amod") then
                        tags.append(OBJECT(child_desc))
                    else
                        tags.append(OBJECT_PROP(child_desc))
            else
                annotate_all_descendants(desc, OBJECT_PROP)
    else if child.relation in ("advcl", "obl") or (child.relation ==
     "advmod" and child.lemm != "not") then
        annotate_all_descendants(desc, CONTEXT)
    else if child.relation="ccomp" then
        if any(child.children = "that" and
         child.children.relation="mark") then
            annotate_all_descendants(child, OBJECT)
        else
            annotate_all_descendants(child_desc, CONTEXT)
    else if child.relation="xcomp" then
        for desc in child.children do
            if desc.relation="obj" then
                tags.append(OBJECT(desc))
                annotate_all_descendants(child_desc, OBJECT_PROP)
            else
                annotate_all_descendants(child_desc, CONTEXT)
```

1. AIM
   (a) If a sentence contains one word with *root* tag, then annotate this word as AIM.
   Otherwise, stop the annotation and return empty results. (AIM is required in the regulative institutional statement and no AIM was found.)

   (b) If the word annotated in 1a has a child with *passive auxiliary* or *copula* relation, then annotate this child as AIM.
   (c) If the word annotated in 1a has a child with *auxiliary* relation and that child's lemma is one of "be", "have", "do", then annotate this child as AIM.
   (d) If the word annotated in 1a has a child with *adverb modifier* relation and that child's lemma is "not", then annotate this child as AIM.
2. DEONTIC
   If the word annotated in 1a has a child with *auxiliary* relation and that child's lemma is one of "must", "should", "may", "might", "can", "could", "need", "ought", "shall", then annotate this word as DEONTIC.
3. ATTRIBUTE
   (a) If the word annotated in 1a has a child with *nominal subject* or *passive nominal subject* relation, then annotate this child as ATTRIBUTE.
   Otherwise, stop the annotation and return empty results. (ATTRIBUTE is required in the regulative institutional statement and no ATTRIBUTE was found.)
   (b) If the word annotated in 3a has a child with *determiner* or *adjectival modifier* relation, then annotate this word as ATTRIBUTE.
   (c) Annotate remaining children of the word annotated in 3a as ATTRIBUTE PROPERTY
   (d) If the word annotated in 1a has a child with *conjunct* relation, then annotate this child and all descendants as ATTRIBUTE PROPERTY.

   If no attribute has been found after these steps, then stop the annotation and return empty results.
4. OBJECT and CONTEXT
   (a) If the word annotated in 1a has a child with *obj* relation, then annotate this word as OBJECT.
   (b) If the word annotated in 4a has a child with one of *det*, *amod*, *case*, *compound* relation, then annotate this child as OBJECT.
   (c) If the word annotated in 4a has a child with *adverbial clause modifier*, then annotate this child and all descendants as CONTEXT.
   (d) Annotate remaining children of the word annotated in 4a as OBJECT PROPERTY.
   (e) If the word annotated in 1a has a child with *open clausal complement* relation:
      - If this word has a child with *object* relation, then annotate this child as OBJECT and all child's descendants as OBJECT PROPERTY.
      - Annotate remaining children of the word found in 4e and all descendants as CONTEXT.
   (f) If the word annotated in 1a has a child with one of *adverbial clause modifier*, *obl*, *adverb modifier* or *clausal complement* relation, then annotate this word and all descendants as CONTEXT.

### 4.2 Rules for Constitutive Statements

MODAL tag is determined analogously to tag DEONTIC in regulative layer. The way the tagger works in constitutive statements is shown in Algorithm 5 and points listed below.

**Algorithm 5:** CONSTITUTIVE tagger.

```
Data: tree: LexicalTreeNode, root: tree.root, tags: list of tags,
      properties: root's children with relation "obl", "obj", or "advcl",
      entities: root's children with relation "nsubj", "nsubj:pass", or
      "expl", context: root's children with relation "advmod" or
      "xcomp"; csubj: root's children with relation "csubj", cop: root's
      children with relation "cop"; entities_noun: root's children with
      relation "det", "acl", or "nmod:npmod"
if root.tag="ADJ" or root.tag="VERB" then
    if root.tag="ADJ" then
        tags.append(CONST_PROP(root))
    else
        tags.append(CONST_FUNCTION(root))
    for entity in entities do
        tags.append(CONST_ENTITY(entity))
        for child in entity.children do
            if child.relation in ("det", "compound", "mark") then
                annotate_all_descendants(child, CONST_ENTITY)
            else
                annotate_all_descendants(child,
                  CONST_ENTITY_PROP)
    for property in properties do
        annotate_all_descendants(property, CONST_FUNCTION)
    for c in context do
        annotate_all_descendants(c, CONTEXT)
else if root.tag="NOUN" then
    if len(csubj)=1 then
        tags.append(CONST_FUNCTION(csubj))
        for child in csubj.children do
            if child.relation="obl then
                annotate_all_descendants(child, CONTEXT)
        for child in cop do
            annotate_all_descendants(child, CONST_ENTITY)
    else
        if len(cop)=1 then
            tags.append(CONST_FUNCTION(cop))
        tags.append(CONST_ENTITY(root))
        for entity in entities$_r$oot do
            for desc in entity.descendants do
                if desc.relation in ("det", "mark") then
                    annotate_all_descendants(desc, CONST_ENTITY)
                else
                    annotate_all_descendants(desc,
                      CONST_ENTITY_PROP)
        for child in properties do
            annotate_all_descendants(child, CONST_PROP)
```

1. If a sentence contains one word with *root* tag and this word is a verb or an adjective:
   (a) If the word founded in 1 is a noun, then annotate it as CONSTITUTIVE FUNCTION, otherwise as CONSTITUTING PROPERTIES.
   (b) If the word annotated in 1a has a child with *aux:pass* or *cop* relation, then annotate this child as CONSTITUTIVE FUNCTION.
   (c) If the word annotated in 1a has a child with *aux* relation and that child's lemma is one of "be", "have", "do", then annotate this child as CONSTITUTIVE FUNCTION.
   (d) If the word annotated in 1a has a child with one of *nsubj*, *nsubj:pass* or *expl* relation, then annotate this child as CONSTITUTED ENTITY.
   Otherwise, stop the annotation and return empty results. (CONSTITUTED ENTITY is required in the constitutive institutional statement, and no CONSTITUTED ENTITY was found.)
   (e) If the word annotated in **??** has a child with one of *det*, *compound*, *mark*, then annotate this child and all child's descendant as CONSTITUTED ENTITY. Annotate remaining children of the word annotated in **??** as CONSTITUTED ENTITY PROPERTY.
   (f) If the word annotated in 1a has a child with *obj* or *advcl* relation, then annotate this child and all child's descendants as CONSTITUTED PROPERTIES.
   (g) If the word annotated in 1a has a child with one of *obl*, *advmod*, *xcomp* relation, then annotate this child and all child's descendants as CONie wiem co wiecej o nim pisac, bo to model, ktory zbudowalam NTEXT.
2. If a sentence contains one word with *root* tag and this word is a noun, then:
   (a) Annotate the word founded in 2 as CONSTITUTIVE ENTITY.
   (b) If the word annotated in 2a has a child with one of *det*, *acl* or *nmod:npmod*, then annotate this child and all child's descendants as CONSTITUTIVE ENTITY.
   (c) If the word annotated in 2a has a child with *csubj* relation, then annotate this child as CONSTITUTIVE FUNCTION.
   (d) If no word was annotated in 2c and the word annotated in 2a has a child with *cop* relation, then annotate this child as CONSTITUTIVE FUNCTION.
   Otherwise, stop the annotation and return empty results. (FUNCTION is required in the constitutive institutional statement, and no FUNCTION was found.)
   (e) If any word was annotated in 2c and this word has a child with *obl* relation, then annotate this child and all child's descendants as CONTEXT.
   (f) If the word annotated in 2a has a child with *nsubj* or *nsubj:pass* relation, then annotate this child and all child's descendants as CONSTITUTING PROPERTIES.

## References

1. Frantz, C.K., Siddiki, S.: Institutional grammar 2.0: A specification for encoding and analyzing institutional design. Public Administration **n/a**(n/a) (2020). https://doi.org/10.1111/padm.12719, `http://dx.doi.org/10.1111/padm.12719`
2. Qi, P., Dozat, T., Zhang, Y., Manning, C.D.: Universal Dependency parsing from scratch. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. pp. 160–170. Association for Computational Linguistics, Brussels, Belgium (Oct 2018). https://doi.org/10.18653/v1/K18-2016, `https://www.aclweb.org/anthology/K18-2016`

# 5 Raw Ontology Builder

## 5.1 Defining Classes and Their Hierarchy

Class and subclass names are defined by unique combinations of IG tags values from statements. Table 2 describes which specific IG tags are used in which statement type to define classes and subclasses. Example:

```
[Constituted Entities] = employee,

[Function] = is,
[Constituted Properties] = unable,
[Constituted Properties Property] = to work
Class: employee,
Subclass: employeeThatIsUnableToWork
```

**Table 2.** IG tags used to define class and subclass by each statement type. Note that '+' denotes concatenation of IG tag value.

| statement type | class | subclasses |
|---|---|---|
| observation constitutive | *Constituted Entities* | *Constituted Entities* + 'that' + *Function* + *Constituted Properties* |
| proper constitutive | *Constituted Entity* | *Constituted Entity* + *Constituted Entity Property* + *Constituted Properties Property* |
| proper constitutive | *Constituted Properties* | *Constituted Properties* + *Constituted Properties Property* |
| proper/observation regulative | *Attributes* | *Attributes* + *Attributes property* |
| proper/observation regulative | *Direct Object* | *Direct Object* + *Direct Object Property* |
| proper/observation regulative | *Indirect Object* | *Indirect Object* + *Indirect Object Property* |

### 5.2 Defining Relations

Relations are defined based on *observations regulative*, *proper regulative* and *proper constitutive*. As raw ontology is supposed to be used for the analysis of possible relationships that emerge from regulation, we treat differently relations that are observational and relations that are *regulative* or *constitutive*. From *regulative observations*, we define *possible observed relations*, which are later modelled as antecedents in SWRL rules. Relation is named by *Aim* tag. If there is an non empty *Indirect Object* tag, we define the second relation, which name is created by transforming *Aim* to passive form. Then we define *regulative relations*. Relation name is defined by [*Deontic* + *Aim*]. *Deontic* is also stored as a relation property, which is later used for analysis. Simmilary, if there is *Indirect Object*, we define the second relation, named after [*Deontic* + passive(*Aim*)]. Finally, we define *constitutive modal relations* from proper constitutive statements using [*Modal* + *Function*] tags as the relation name. Details of domains and rages of each relation are provided in Table 3.

Both in *constitutive modal relations* and *regulative relations*, we treat each relation independently – that means that for every class domain we create different relations – reason for this, is again, that we want use this ontology primarily for quantitative analysis of such relations.

**Table 3.** IG tags used to define relations by each statement type.

| statement type | domain | relation name | range |
|---|---|---|---|
| observation regulative | *Atttribute + Attribute Property* | Aim | *Direct Object + Direct Object Property* |
| observation regulative | *Direct Object + Direct Object Property* | passive(Aim) | *Indirect Object + Indirect Object Property* |
| proper regulative | *Atttribute + Attribute Property* | *Deontic + Aim* | *Direct Object + Direct Object Property* |
| proper regulative | *Direct Object + Direct Object Property* | *Deontic* + passive(*Aim*) | *Indirect Object + Indirect Object Property* |
| proper constitutive | *Entity + Entity Properties* | *Modal + Function* | *Constituted property + Constituted Property Properties* |

### 5.3 Defining Axioms and SWRL Rules

Axioms and SWRL rules are build only on basis of proper regulative and proper constitutive statements – those statements which really constitute or regulate some part of reality. In those statements, if *Activation Condition* is empty, relation defined by such statement must always hold, then we add axiom that every subject of the class defined by this row must be in such relation with object (same for the relation between object and indirect object).

If there exists *Activation Condition* referring to some observational statements, we define SWRL rules where the antecedent is defined by referred statement and the consequent is statement that is referring. We iterate over all statements referred in activation condition tag. For every statement, we check whether it is *regulative observation* or *constitutive observation*. For *regulative observation* we add rule that has all relations defined in that statement in the antecedent. For *constitutive observation* antecedent, consist of the constraint on being a specific subclass defined by such statement. In consequent, there are all relations defined this proper regulative/constitutive statement that we are considering.

Example rule created automatically is as follows:

```
EmployeeThatIsUnableToWork(?x),
    EmployeeEmployedByEmployer(?x), Employer(?y),
    PaidSickTime(?q) -> shall_provide(?y,?q),
    shall_be_provided_to(?q,?x)
```