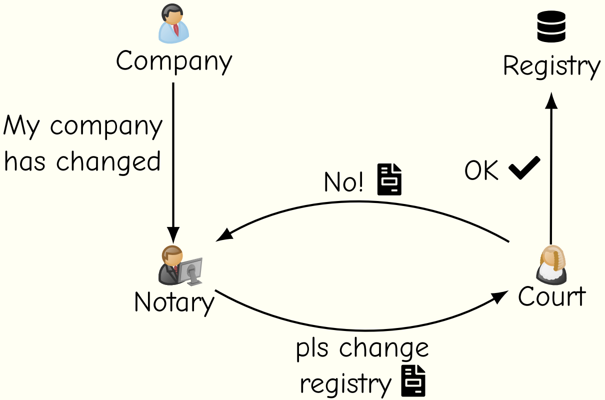
DIREGA: Formalizing German Register Law

Merlin Humml

28th October 2025

Register Application



Requirements

Dependability

- Legal context requires certainty

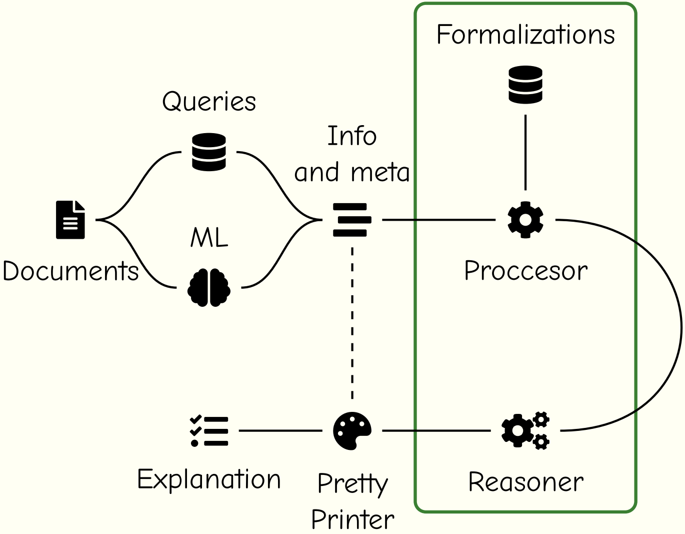
⇒ Cannot just be machine learning

Explainability

- Notary needs to make the finall call

⇒ System needs to explain its reasoning

The Pipeline



Challenges I

Counting

“If there are at least two executives, then the signature of two executives or one executive and one procurator is required.”

Equality

“The name and birthday of person X needs to be identical in all submitted documents.”

Free-form constructs

Many regulations a company can decide are free-form but need to be checked.

Challenges II

Time

Multiple states of the world are relevant as each document needs checking at the time of signing and might change the state of the world after signing.

Missing information

The system should work despite missing information.

Completeness

Multiple issues should be found in one pass.

Carneades

Based on

- Constraint Handling Rules (CHR)
- Formal argumentation

Cool features

- Easy to understand
- Explanation through labelling
- Time built in
- Finds multiple issues

Problems

- No counting
- No quantifiers
- Explanation misses last step in the negative case

Interlude: Formal Argumentation

Representation

- Arguments are nodes
- Edges are semantic relations (usually attack)
- Graph represents argumentation

Structure and Schemes

- *Structured* argumentation labels the nodes (with “formulas”)
- Argument *schemes* guide construction of arguments

Semantics

- Labelling (in, out, undecided)
- Computed through fixpoints

Carneades Under the Hood

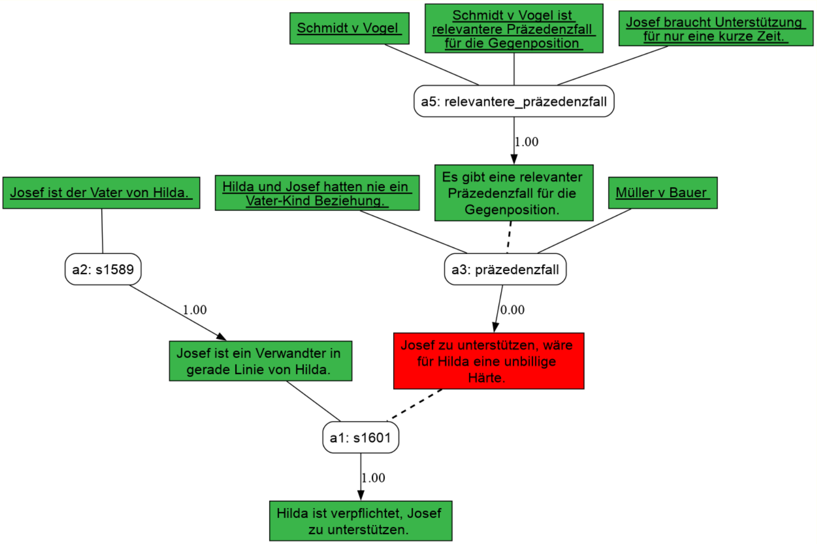
Building the graph

1. Explicit arguments are built into initial graph
2. Argument schemes are translated into CHR rules
3. CHR solver extends the graph by instantiating all possible arguments from the schemes given the arguments in the graph and assumptions

Labelling

- Arguments are labelled from the outside in
- Assumptions start as true everything else not the target of an argument is labelled out
- Propagation until fixed

Carneades Explanation


Schmidt v Vogel
Schmidt v Vogel ist relevantere Präzedenzfall für die Gegenposition
Josef braucht Unterstützung für nur eine kurze Zeit.
a5: relevantere_präzedenzfall
1.00
Josef ist der Vater von Hilda.
Hilda und Josef hatten nie ein Vater-Kind Beziehung.
Es gibt eine relevanter Präzedenzfall für die Gegenposition.
Müller v Bauer
a2: s1589
a3: präzedenzfall
1.00
0.00
Josef ist ein Verwandter in gerade Linie von Hilda.
Josef zu unterstützen, wäre für Hilda eine unbillige Härte.
a1: s1601
1.00
Hilda ist verpflichtet, Josef zu unterstützen.

sCASP

Based on

- Goal directed answer set programming (ASP)
- Constraint solving (C)
- Prolog (via metaprogramming)

Cool features

- Proper programming language (counting & equality for free)
- Time via embedding
- Explanations with custom messages
- Code generation possibly viable via LLMs

Problems

- Bugs
- Finding all issues instead of just the first

sCASP Under the Hood

Dualisation

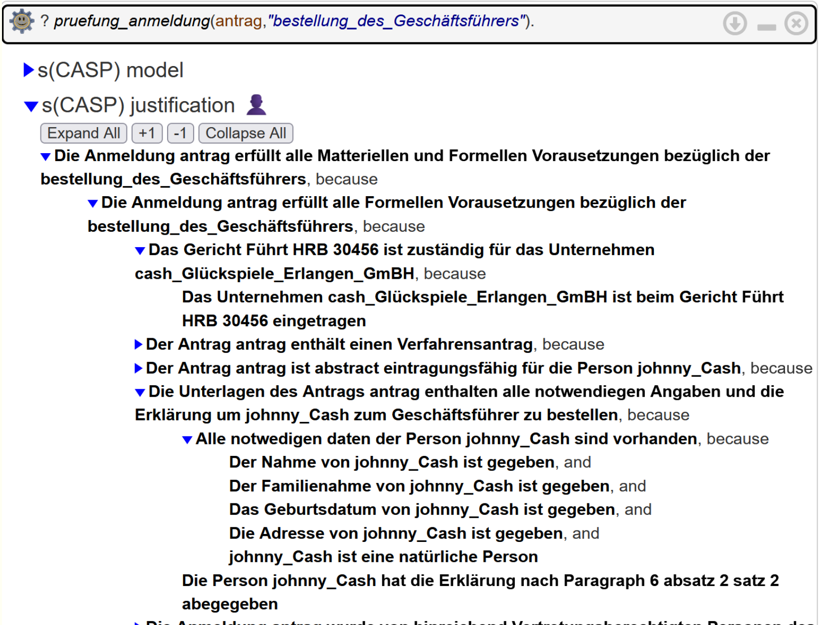
- Generates dual predicates via metaprogramming
- Some Prolog features are not dualizable
- `not <predicate>` is replaced by dual predicate

Verbalisation

- Custom format strings for predicate verbalisation

  ```
  #pred submitted_to(Court) :: 'Der Antrag
   ↪  richtet sich an das Gericht @(Court)'.
  ```
- Are used to format explanation trees

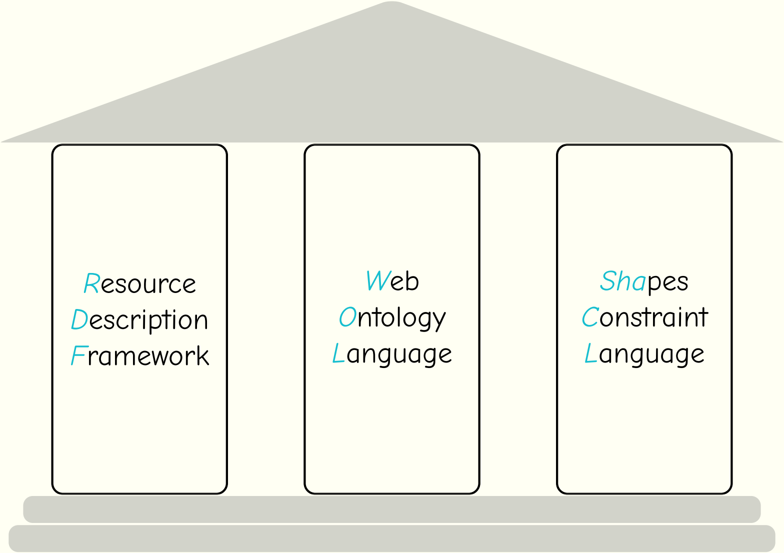
sCASP Explanation



```
? pruefung_anmeldung(antrag,"bestellung_des_Geschäftsführers").
```

- ▼ **Die Anmeldung antrag erfüllt alle Matteriellen und Formellen Vorausetzungen bezüglich der bestellung_des_Geschäftsführers**, because
  - ▼ **Die Anmeldung antrag erfüllt alle Formellen Vorausetzungen bezüglich der bestellung_des_Geschäftsführers**, because
    - ▼ **Das Gericht Führt HRB 30456 ist zuständig für das Unternehmen cash_Glückspiele_Erlangen_GmBH**, because
      - **Das Unternehmen cash_Glückspiele_Erlangen_GmBH ist beim Gericht Führt HRB 30456 eingetragen**
    - ▶ **Der Antrag antrag enthält einen Verfahrensantrag**, because
    - ▶ **Der Antrag antrag ist abstract eintragungsfähig für die Person johnny_Cash**, because
    - ▼ **Die Unterlagen des Antrags antrag enthalten alle notwendiegen Angaben und die Erklärung um johnny_Cash zum Geschäftsführer zu bestellen**, because
      - ▼ **Alle notwedigen daten der Person johnny_Cash sind vorhanden**, because
        - **Der Nahme von johnny_Cash ist gegeben**, and
        - **Der Familienahme von johnny_Cash ist gegeben**, and
        - **Das Geburtsdatum von johnny_Cash ist gegeben**, and
        - **Die Adresse von johnny_Cash ist gegeben**, and
        - **johnny_Cash ist eine natürliche Person**
      - **Die Person johnny_Cash hat die Erklärung nach Paragraph 6 absatz 2 satz 2 abegegeben**
    - ▶ **Die Anmeldung antrag wurde von hinreichend Vertretungsberechtigten Personen des**

The W3C Stack



Resource Description Framework

Web Ontology Language

Shapes Constraint Language

The W3C Stack — RDF

- Everything is a `<subject> <predicate> <object>` triple

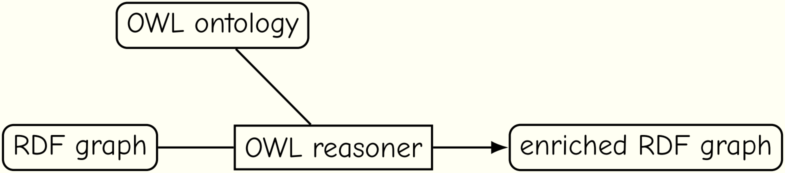
⇒ Directed graph with labelled edges

```
{
  ”@type”: ”direga:ShareholderList”,
  ”@id”: ”c:ShareholderList”,
  ”schema:dateCreated”: ”2024-07-05”,
  ”direga:shareholder”: [
    {
      ”direga:shareNumber”: 1,
      ”direga:holder”: ”c:JCash”,
      ”direga:shareValue”: 12500
    },
  ],
  ”direga:capital”: 25000
}
```

The W3C Stack — OWL

Description Logic

- Extension of multimodal graded modal logic
- Nominals
- Contraints on predicates/relations



The W3C Stack — SHACL

Shapes

- Distinct from OWL classes
- Have attached structural requirements and messages

```
:ShareholderListShape a sh:NodeShape ;
  sh:targetClass direga:ShareholderList ;
  sh:property :AboutOrganization ;
  sh:property [ sh:path schema:listItem ;
    sh:class direga:ShareholderInfo ;
    sh:message ”A shareholder list can contain only
  ] ;
  sh:property [ sh:path schema:listItem ;
    sh:minCount 1 ;
    sh:message ”A shareholder list has to contain a
  ] .
```

Modal Reasoning Spectrum

Synthesis

Formula ⇒ (pointed) frame and valuation satisfying formula

Satisfiability

Formula ⇒ exists (pointed) frame and valuation satisfying formula

Validation

Formula + frame ⇒ exists valuation satisfying formula in the given frame

Modelchecking

Formula + frame + valuation ⇒ formula satisfied in the given frame under the given valuation

Coalgebraic Validation?

Coalgebraic Valuation

- Atoms are seen als nullary modalities
- Valuation is part of the composed frame structure

Coalgebraic Generalisation?

Can there be a generic framework for leaving off part of a coalgebraic frame structure?

The W3C Stack — SHACL Extensions

SHACL-af (Advanced Features)

- Adds inference capabilities to SHACL
- Rules to construct triples pre-validation when preconditions match
- A bit more flexible than OWL due to complicated path expressions
- Blurs distinction between inference and validation

The W3C Stack — SPARQL

Queries

- If RDF models data, can we query it?

```
SELECT ?c ?r
WHERE
{
  SELECT ?c (GROUP_CONCAT(?n; separator=”, ”) AS ?r
  WHERE
  { ?c direga:executive ?p .
    ?p direga:person ?n .
  }
  GROUP BY ?c HAVING (COUNT(?p)>=1)
}
```

The W3C Stack — Star Extension

RDF-star + SPARQL-star

- “Merlin said that Lutz said we should use SHACL”
- «`”we should use SHACL” :saidBy :Lutz`» `:saidBy :Merlin`
- Useful for incorporating time
- Does *not* imply `”we should use SHACL” :saidBy :Lutz`
- No OWL support

Vertretungsbefugnis

Allgemeine Vertretungsbefugnis

Regulates power of representation of the company

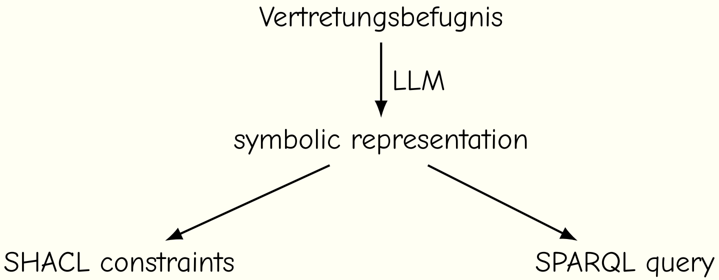
Spezifische Vertretungsbefugnis

Special rules attached to a specific executive

Use-cases

1. Notary wants to have a list of possible combinations of executive signatures
2. Our system gets fed a document and needs to check if the signatures suffice

Formal Power of Representation



Vertretungsbefugnis

LLM

symbolic representation

SHACL constraints

SPARQL query

Language of Representation

Language

- "If there are at least 2 executives ..."
- "...at least 2 executives need to sign"
- "One of the signatures can be from a procurator"
- In general arbitrary properties from the commercial register can be referenced

Formal Language

$$\phi, \psi := \phi \wedge \psi \mid \phi \vee \psi \mid \text{min } n \textit{ role } \text{signed} \mid \text{min } n \textit{ role } \text{exist}$$

$$\text{where } n \in \mathbb{N},\ \textit{role} \in \{\text{executive}, \text{procurator}, \text{liquidator}\}$$

Compiling to SPARQL I

Idea

- Use shared variables for company and candidate set
- Each formula becomes **`SELECT`** query
- Counting from **`GROUP BY`** `?company` with **`HAVING`** `COUNT()` restriction
- Join by using subselects

Damn Conjunction

- min n $role_1$ signed $\wedge$ min m $role_2$ signed
- The result would be the cross product of the two conjuncts result sets
- SPARQL knows sets only implicitly

Compiling to SPARQL II

Idea 2.0

- Compile whithout shared candidate set variable
- Pass the list of variables to accumulate to the surrounding statement
- Form the cross product “manually” while compiling
- Lots and lots of constraints to make it duplicate free
- Ugly but could work…

What amount of compilation logic is OK before the formalization degenerates to programming?

Compiling to SHACL

Future work

Rule Composition

Disjunctive

general: two signatures (one can be a procurator)

specific: Merlin can represent the company alone

Conjunctive

general: two signatures (one can be a procurator)

specific: Mr and Mrs Smith can not represent the company without another signature

181 BGB

Insichgeschäft

- Company A and B both have X as executive
- Can X sign for a deal between A and B?
- What about a deal between A and himself?

Formalization

- Representation: easy just two booleans
- Complicates all checks: nature of the deal relevant
- Requires multiple companies being in context

Time

Discrete but Unknown

- Every document and signature creates a point in time
- Not known in advance how many
- No evolving system characteristics

What formalism to encode?

- Fluents
- Situations/points in time
- *No* relevant actions/events

Summary

- Seemingly simple formalization turns out challenging
- All formalisms checked so far not ideal

Thanks!