Automated Reasoning with eFLINT and sCASP: Formalization of Commercial Registry Law

DIREGA

- DIgital REGister Assistant
- Research Project in Legal AI
  - Automated / supported decisions
  - Check correctness of applications
- Interdiciplinary Team
  - NLP, Symbolic KI, Subsymbolic KI, Notary

Commercial register

- Legitimate expectation (Vertrauensschutz)
  - Entries in register can be assumed to be correct
- Database on Companys and Tradespeople
  - Name, business address, owners, representation, capital, statutes
  - Numerus clausus (Typenzwang)
- Digital database / machine readable
  - Online accessible

Application checking

- Formal examination
  - Jurisdiction (locationally)
  - Necessary documents
  - Necessary information
  - ...
- Material examination
  - Consistency of material
  - Formal correctness of company resolution
  - Material correctness of company resolution

Without founded clues the registry court must only use information from the given documents in its checks

Legal Framework

- Large distributed domain of legal texts
  - Family law and others (FamGF)
  - Taxcode (Abgabenordnung)
  - Company Law (Gesellschaftsrecht)
- Common practice
- “Gustavus” as reference of cases

Working with Test schema from Notary

eFLINT

- Trace monitoring / checking tool
- Hofeld’s framework of legal fundamental concepts for normative formalizing
- Used e.g. for formalizing the GDPR
- System modeling
- Scenario execution
  - Single actor
  - Multiple actor

Syntax: eFLINT

State

- Fact
- Invariant
- Obligation

Transitions

- Action
- Event

Inference Rules / Conditions

- Disjunctive: Holds when / Derived by
- Conjunctive: Conditioned by

Syntax: eFLINT

```
Fact anmeldungsart Identified by Verlegung_Hauptsitz , Bestellung_Geschaeftsfuehrers


Fact plz Identified by Int
Placeholder adresse For plz


Invariant sitz_einzigartig When
    (Forall sitz1:
                !(Exists sitz2:
                            (sitz1.unternehmen == sitz2.unternehmen) &&
                            (sitz1.adresse != sitz2.adresse)
                )
    )
```

Syntax: eFLINT

```
Act umzug
    Actor unternehmen
    Related to adresse1, adresse2
    Holds when sitz(unternehmen,adresse1) && unternehmen && adresse1 && adresse2 && (adresse1 != adresse2)
    Creates sitz(unternehmen,adresse2)
    Terminates sitz(unternehmen,adresse1)
```

Syntax: eFLINT

```
Fact anzahl_geschaeftsfuehrer Identified by Int
    Derived from Count(Foreach geschaeftsfuehrer : geschaeftsfuehrer
                           When Holds(geschaeftsfuehrer) )

Fact vertreter Identified by person * unternehmen
    Derived from (Foreach person, unternehmen: geschaeftsfuehrer(person, unternehmen))


Fact vertreter Identified by person * unternehmen
    Holds when geschaeftsfuehrer(person, unternehmen)
    Holds when notar(person) && vertretungsbefugnis(person,unternehmen)

Fact vertreter Identified by person * unternehmen
    Conditioned by  notar(person)
    Conditioned by  vertretungsbefugnis(person,unternehmen)
```

- Two ideas considered for modeling as action:
  - Changes on company and entry into register
  - Check items from notary test schema

Example for modeling actions as changes / entry (1)

```
Act umzug
    Actor unternehmen
    Related to adresse1, adresse2
    Holds when sitz(unternehmen,adresse1) && unternehmen && adresse1 && adresse2 && (adresse1 != adresse2)
    Creates sitz(unternehmen,adresse2)
    Terminates sitz(unternehmen,adresse1)

Act anmelden_adresse
    Actor person
    Recipient register
    Related to adresse1,adresse2,unternehmen
    Holds when register && unternehmen &&
                sitz(unternehmen,adresse1) && eintrag(unternehmen,register,adresse2) &&
                !(adresse1 == adresse2) &&
                zustaendig(register,adresse2)&&
                vertreter(person,unternehmen)
    Creates eintrag(unternehmen,register,adresse1)
    Terminates eintrag(unternehmen,register,adresse2)
```

Example for modeling actions as changes / entry (2)

```
+ plz(1).
+ plz(2).
+ sitz(A,1).
+ register(N).
+ eintrag(A,N,1).
+ register_adresse(N,5).
+ person(M).
+ geschaeftsfuehrer(M,A).
umzug(unternehmen("A"),plz(1),plz(2)).
anmelden_adresse(person("M"),register("N"),plz(2),plz(1),unternehmen("A")).
```

```
executed transition:
umzug(unternehmen("A"),plz(1),plz(2)) (ENABLED)
~umzug(unternehmen("A"),plz(1),plz(2))
-sitz(unternehmen("A"),plz(1))
+sitz(unternehmen("A"),plz(2))
+umzug(unternehmen("A"),plz(2),plz(1))
+anmelden_adresse(person("M"),register("N"),plz(2),plz(1),unternehmen("A"))
```

```
executed transition:
anmelden_adresse(person("M"),register("N"),plz(2),plz(1),unternehmen("A")) (ENABLED)
~anmelden_adresse(person("M"),register("N"),plz(2),plz(1),unternehmen("A"))
-eintrag(unternehmen("A"),register("N"),plz(1))
+eintrag(unternehmen("A"),register("N"),plz(2))
```

conclusions

- Useful for monitoring of making entries into database

- No explainability of decisions
  - Not useful as assistant for notary

Example: eFLINT different category of move

```
Fact umzug_politischegemeinde Identified by umzug
    Holds when  umzug &&
                selbes_registergericht(umzug.ursprungs_adresse,umzug.ziel_adresse) &&
                selbe_politischegemeinde(umzug.ursprungs_adresse,umzug.ziel_adresse).

Fact umzug_registerbezirk Identified by umzug
    Holds when  umzug &&
                selbes_registergericht(umzug.ursprungs_adresse,umzug.ziel_adresse)&&
                !selbe_politischegemeinde(umzug.ursprungs_adresse,umzug.ziel_adresse).

Fact umzug_inland Identified by umzug
    Holds when  umzug &&
                ! selbes_registergericht(umzug1.ursprungs_adresse,umzug.ziel_adresse).
```

Conclusion

- Test result per item from notary test schema
  - No real explanation of acceptance / rejection
- De facto sequence of simple querys
  - First order reasoner could give justification

Deficciencies

- Lacks formulas over diferent states
  - No resoning over traces
- Actions change not dependent on conditions
- Deontic operation just aded or removed invariant

s(CASP)

- Constraint Answer Set Programming
- Top down / goal driven approach
- Human readable justification

- Non monotonic logic

Duality of Formulas

- Automatic generation of dual rules
  - Non monotonic rules
- Provides positive and negative queries

Syntax : sCASP

- Conjunction

```
alpha(X) :- beta(X),gamma(X).
```

- Disjunction

```
beta(X):- gamma(X).
beta(X):- theta(X).
```

- Negation

```
theta(X):- not alpha(X).
```

Standard solution: not (negation as failure)

Not possible with predicates having free variables

```
abstract_eintragungsfaehigkeit(Antrag, Per):-
        not exists_vertretungsbefugnis(Antrag, beschluss, Per).

exists_vertretungsbefugnis(Antrag, beschluss, Per):-
        vertretungsbefugnis(Antrag, beschluss, Per, _).
```

Rules of representation

- Checking lawfulness of given rule of representation
- Checking amount of signatures with given rule of representation
- Can be set almost arbitrarily
  - Few standard cases are predominant
    - Individual representation (Einzelvertretung)
    - Representation with fixt number of other managing directors
    - Representation with other managing director or prokurist

Example: cheking amount of signitures

Rules of representation for a fixed number of people

```
mlength([],L) :- L #= 0.
mlength([_|TL],L) :- L #> 0, mlength(TL,L - 1).


vertretung_und_unterschrift(Antrag,Per,Unternehmen):-
            vertretung(Per,Unternehmen,v1),
            unterschrift(Per,Antrag).

unterschriften_nach_vertretungsbefugnis(Antrag,Per,Unternehmen,ver3):-
            number(Unternehmen,N),
            findall(X, vertretung_und_unterschrift(Antrag,X,Unternehmen),L),
            mlength(L,M), M #> N.
```

Example

Decision criteria by majority vote with majority depending on company

```
summe([],L) :- L #= 0.
summe([N|TL],L) :- L #> 0, summe(TL,L - N).

mehrheit_fuer_beschluss(Antrag,Unternehmen):-
            findall(X,stimme_abgegeben(Per,X,Unternehmen,Antrag),L),
            mehrheit(Unternehmen,N),
            summe(L,LL), LL#>N.
```

Consistency checking

- Consistency of model as constraint

```
False:- familiename(Per1,Dock1,Name1),
        familiename(Per2,Dock2,Name2),
        equal(Per1,Per2),
        not equal(Name1 , Name2).
```

- sCASP does not find a model if inconsistent

- Consistency as part of checking

```
inkonsistenz_familiename(Antrag)  :-
            familiename(Per1,Dock1,Name1),
            familiename(Per2,Dock2,Name2),
            equal(Per1,Per2),
            not equal(Name1 , Name2).
```

- Gives explanantion for failure

Justification

- sCASP generates model and justification tree
  - Human readable justification
  - Own formalizations can be used
  - Detail level can be controlled

```
#pred fallgruppe_vert(ver1)::'die Vertretungsbefugnis ist der Fallgruppe Einzelvertretung zuzuordnen'.

#pred pruefung_anmeldung(Antrag, Gericht, Unternehmen, Per, Zweck)::
    'Die Anmeldung @(Antrag) erfüllt alle Materiellen und Formellen Voraussetzungen bezüglich der @(Zweck)'.
```

- Explanation ends at first counter example
  - Additional explanation can be found with external help

Explanation acceptance

```
?- ?(pruefung_anmeldung(antrag,'Fürth','Cash Glückspiele Erlangen GmbH','Eugen_Kolunin','Bestellung des Geschäftsführers')).
```

▶ s(CASP) model

▼ s(CASP) justification

Expand All | +1 | -1 | Collapse All

- ▼ **Die Anmeldung antrag erfüllt alle Materiellen und Formellen Voraussetzungen bezüglich der Bestellung des Geschäftsführers**, because
  - ▼ **Die Anmeldung antrag erfüllt alle Formellen Voraussetzungen bezüglich der Bestellung des Geschäftsführers**, because
    - ▼ **Das Gericht Fürth ist zuständig für das Unternehmen Cash Glückspiele Erlangen GmbH**, because
      - **Das Unternehmen Cash Glückspiele Erlangen GmbH ist beim Gericht Fürth mit der HRB 30456 eingetragen**
    - ▶ **Der Antrag antrag enthält einen Verfahrensantrag**, because
    - ▼ **Der Antrag antrag ist Abstract eintragungsfähig für die Person Eugen_Kolunin**, because
      - **der Antrag antrag enthält die Einzelvertretung für die Person Eugen_Kolunin**, and
      - **die Vertretungsbefugnis ist der Fallgruppe Einzelvertretung zuzuordnen**, and
      - ▼ **Die in der Anmeldung genannte Vertretungsbefugnis für Eugen_Kolunin unterscheidet sich von der in der Satzung genannten**, because
        - **der Antrag antrag enthält die Einzelvertretung für die Person Eugen_Kolunin**, and
        - **Die Satzung enthält die Vertretungsbefugnis ver2**
    - ▶ **Die Unterlagen des Antrags antrag enthalten alle notwendigen Angaben und die Erklärung um Eugen_Kolunin zum Geschäftsführer zu bestellen**, because
    - ▼ **Die Anmeldung antrag wurde von hinreichend Vertretungsberechtigten Personen des Unternehmens Cash Glückspiele Erlangen GmbH Unterschrieben**, because
      - **Der Gesellschafter Eugen_Kolunin hat das Dokument antrag unterschrieben**, justified above, and
      - ▼ **Der Antrag antrag wurde von Eugen_Kolunin unterschrieben der Einzelvertretungsbefugt ist**, because
        - **der Antrag antrag enthält die Einzelvertretung für die Person Eugen_Kolunin**, and
        - **Der Gesellschafter Eugen_Kolunin hat das Dokument antrag unterschrieben**, justified above
    - ▶ **Der Antrag antrag enthält alle notwendigen unterlagen für eine Bestellung des Geschäftsführers**, because
  - ▼ **Die Anmeldung antrag erfüllt alle Materiellen Voraussetzungen bezüglich der Bestellung des Geschäftsführers**, because
    - ▶ **die Unterlagen des Antrags antrag sind konsistent**, because
    - ▶ **Der Beschluss des Unternehmens Cash Glückspiele Erlangen GmbH ist wirksam**, because
    - ▼ **unterschriftsbeglaubigt** holds for antrag, because
      - ▶ **Alle nicht davon ausgenommenen Unterschriften wurden Notariell beglaubigt**, because

0.239 seconds cpu time

Explanation rejection

```
?- ?(not pruefung_anmeldung(antrag,'Fürth' ,'Cash Glückspiele Erlangen GmbH','Kaplan_Achmed','bestellung_des_Geschäftsführers')).
```

▶ s(CASP) model

▼ s(CASP) justification

Expand All | +1 | -1 | Collapse All

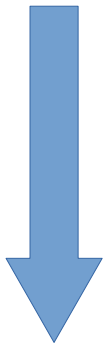
- ▼ **Die Anmeldung antrag erfüllt nicht alle Materiellen und Formellen Voraussetzungen bezüglich der bestellung_des_Geschäftsführers**, because
  - ▼ **Die Anmeldung antrag erfüllt nicht alle Formellen Voraussetzungen bezüglich der bestellung_des_Geschäftsführers**, because
    - ▼ **Das Gericht Fürth ist zuständig für das Unternehmen Cash Glückspiele Erlangen GmbH**, because
      - **Das Unternehmen Cash Glückspiele Erlangen GmbH ist beim Gericht Fürth mit der HRB 30456 eingetragen**
    - ▼ **Der Antrag antrag enthält einen Verfahrensantrag**, because
      - **Die Datei antrag geschäftsführer bestellung gehört zum Antrag antrag**, and
      - **Die Unterlage antrag geschäftsführer bestellung enthält die Formalisierung : In obiger Registersache wird zur Eintragung angemeldet**
    - ▼ **Der Antrag antrag ist Abstract eintragungsfähig für die Person Kaplan_Achmed**, because
      - **der Antrag antrag enthält die Einzelvertretung für die Person Kaplan_Achmed**, and
      - **die Vertretungsbefugnis ist der Fallgruppe Einzelvertretung zuzuordnen**, and
      - ▼ **Die in der Anmeldung genannte Vertretungsbefugnis für Kaplan_Achmed unterscheidet sich von der in der Satzung genannten**, because
        - **der Antrag antrag enthält die Einzelvertretung für die Person Kaplan_Achmed**, and
        - **Die Satzung enthält die Vertretungsbefugnis ver2**

    there is no evidence that **Der Antrag antrag enthält alle notwendigen Daten der Per Kaplan_Achmed und die Erklärung dass keine Umstände i.S.v. § 6 Abs. 2 Satz 2 Nr. 2 und 3 sowie Satz 3 und 4 GmbHG vorliegen.**

Next | 10 | 100 | 1,000 | Stop

Finding missing facts

```
:- include( "logic_Geschäftsführer.pl" ).
 :- include( "find_counterexample.pl" ).
 :- include( "Jonny_Cash_Fall1.pl" ).
```



By repeated quering an adding of missing predicates
all missing facts can be found

- Provides the first predicate option in the justification tree
- Only works on missing predicates

```
:- include( "logic_Geschäftsführer.pl" ).
:- include( "find_counterexample.pl" ).
:- include( "Jonny_Cash_Fall1.pl" ).

gesellschaftsbeschluss(beschluss) .
unterschrift(alexander, beschluss) .
```

Conclusion: sCASP

- Arithmetic operations possible
- Understandable explanations

- Extandable with other logical Frameworks
  - Defeasible logic
  - Argumentation framework