From (Deontic) Logic to (the Lawyer's) Practice

Tomer Libal

Why Applications?

- Deontic logic in 2020-2030
- 2000 - 2020: AI and law
  - Mainly machine learning
- 2020 - 2030: Automated reasoning and law
  - No bias
  - Explainable
  - Certified
- Why Deontic logic and applications?
  - Practical use might shed some insight into required properties for certain uses

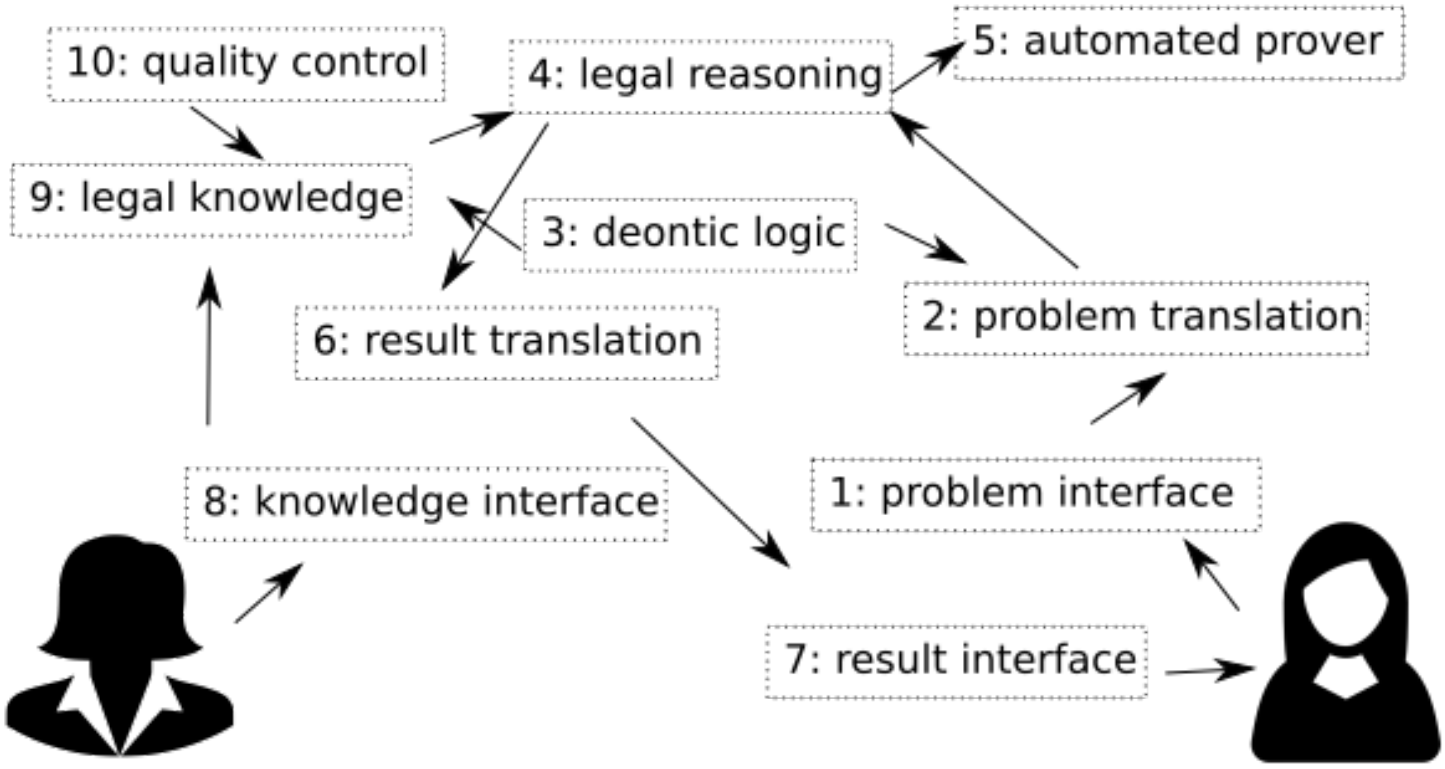
Deontic logic and applications

- Expressive logic
- Automated reasoning
- Human interfaces
  - Formalize normative/legal knowledge
  - Formalize normative/legal problems
  - Human interaction with the result
- In this talk
  - The whole picture
  - Different levels of abstraction
- Part of a proof of concept project

Two possible applications

- Interactive verification of normative/legal arguments
  - This approach is very successful in hardware/software verification
  - in this case, high logical expertise is required
- Replace some of legal expert work
  - in this case, we expect zero logic knowledge
- I will focus on the second

High-level view



1. Problem interface and 7. Result interface

- **Requirments:**
  - Correspond to a specific task of user
  - Small or no amount of learning required
- **Method:**
  - Collaboration with stakeholders
    - Identify best use case
- **Example use case 1:**
  - An auditor wants to get feedback about the compliance of a document
  - Auditor uploads the document
  - Auditor gets an annotated version of the document
    - Each decision is supported by full legal argument
- **Example use case 2:**
  - A lawyer wants to prepare for court proceedings
  - Lawyer uploads information about case
  - Lawyer obtains possible arguments and counter-arguments

2. Problem translation

- **Possible input**
  - Semi-structured data
    - Diagrams (e.g. BPMN)
    - Documents (e.g. excel sheets)
    - Software code
  - Unstructured data (e.g. contracts)
- **Required output**
  - Deontic logic

2. Problem translation: solutions

- Ontologies
  - Input: all types of input
  - Example: PrOnto
- Natural Legal Language Processing
  - Input: semi and unstructured data
  - Example: use existing solutions to generate semi-structured data, then use algorithms
- Algorithms
  - Input: semi-structured data
- (Dynamic) code analysis
  - Input: software
  - Example: logs-based

3. Legal reasoning

- **Requirements:**
  - Expressivity: CTD, exceptions, contexts, unknown requirements
  - Efficiency: answer validity, find counter models, etc. in seconds
- **Possible solutions:**
  - Deontic logics (?)
  - Automated theorem provers
- **Problems:**
  - Does such a logic exist?
  - If it does, do its properties have efficient mechanization?

3. Legal reasoning: Solution

- Use a high-level, but formal, language to capture the precise legal meaning
  - CDT, exceptions
  - Contexts
  - Unknown requirements
- When a specific query is asked
  - Translate from the high-level language into a specific logic
    - Most efficient logic
    - Sufficient expressivity for the question
- Other examples
  - TLA of Leslie Lamport
- Practicals
  - Using JSON

4. Automated provers

- **Requirements:**
  - Efficiency: reply within seconds
  - License: grants sometimes dislike restrictive licenses
  - Support for logic: next

5. Deontic logic

- Given our solution to legal reasoning requirements are simpler
- **Requirements:**
  - Capture specific problems in specific contexts
  - Be supported by a variety of (state-of-the-art) provers
    - Correctness
    - Efficiency
    - Different questions: validity, counter models

5. Deontic logic: one option - DL*

- Collaboration with Matteo Pascucci
  - Presented in ICAIL '19
- **Advantages**
  - Supports many "philosophical" requirements
    - e.g. 8 points of Carmo and Jones '02
  - Direct encoding into first-order multi-modal logic
    - MleanCoP state-of-the-art prover support
  - Theoretical support for variety
    - Model finders: research exists (tableaux) but no implementation
  - Variety: more provers exist of lesser efficiency
- **Disadvantage**
  - Non-monotonic reasoning is crucial and is currently not supported
    - Exceptions

5. Deontic logic: further solution

- **Problem:** get
  - Nice "philosophical" properties
  - Support non monotonic reasoning
  - Efficiency, variety, etc. of theorem provers
  - FOML seems to be on the boundaries of efficient theorem provers
    - Encoding non-monotonicity in FOML is challenging
- **Solution:**
  - Two (or more) phases of reasoning

5. Deontic logic: two-phase reasoning

- Current work with Pascucci, Gabbay and van der Torre
  - Future work: extend to FOL
- Phase 1: apply reasoning to compute monotonic sub-problems (in DL*)
  - SAT, SMT
- Phase 2: MleanCoP

6. Result translation

- **Requirements:**
  - Produce result in an informative way
  - Legal argument
- **Problems:**
  - MleanCoP produces a connection proof
    - `[d2:[ (1^d)^189^[]],-d2: -[ (2^d)^190^[]]]`
    - Counter-models don't look much better
- **Solutions:**
  - Use ontologies
  - Feasible

8. Knowledge interface

- **Requirements:**
  - Formalize complex and bountiful legal knowledge
  - Formalization understandable to legal experts
- **Solution:**
  - NAI: an annotations editor for legal texts
  - Collaboration with Alexander Steen
  - Expressive language for knowledge
    - JSON
  - Won 1st prize in IRIS 2020
- **Advantages** of an annotation editor:
  - Faithful to the original text
    - Understandable by legal experts
  - Well engineered: DRY, automating generation
    - Rigorous, correctness

9. Legal knowledge and 10. Quality control

- **Requirements:**
  - Quality of knowledge needs to be on par with legal experts
  - Knowledge changes all the time
  - Knowledge is never complete and added incrementally
- **Solution:**
  - New legal engineering practices

Legal engineering

- A new term describing the combination of legal and technology expertise
- Our meaning is different
  - Build and maintain legal knowledge using specific engineering methodologies
- Example: software engineers
  - Software developers who use specific engineering methodologies, such as
    - Behavior driven development: write executable test stories of how your code should behave
    - ...
- Ongoing work with Alexander Steen

icomplai.eu

- Current project to meet the needs of an industrial use case
  - Privacy law compliance checking
- Application for a 2 years FNR PoC project
- `icomplai.eu`
- Industrial collaboration with
  - CNPD - DPAs
  - KPMG - auditing firms
  - Some law firms

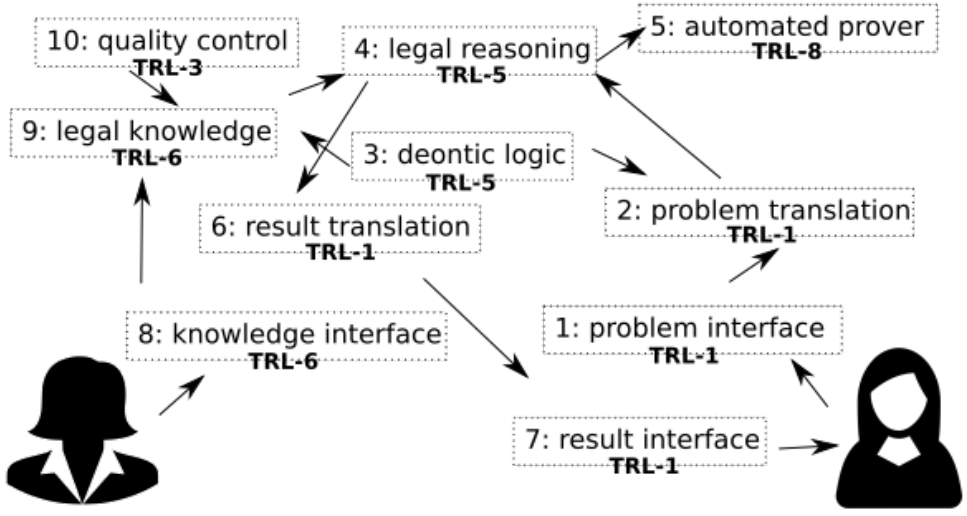
Main milestones

1. (12m) Having a basic prototype capable of processing business process diagrams for GDPR compliance based on the main GDPR articles
2. (18m) Integrate prototype within the specified use cases for fully automated support and interaction with users
3. (24m) icomplai can process various free text documents, such as privacy policies and data breach notices, provide legal expertise based on all relevant GDPR articles and many local laws and being optimized to process thousands of documents per day

Challenge

- For meeting these milestones, work on all 10 points must be done during the project

Summary



- **24 months later:** TRL-7 for all components

TRL 1	basic principles observed
TRL 2	technology concept formulated
TRL 3	experimental proof of concept
TRL 4	technology validated in lab
TRL 5	technology validated in relevant environment
TRL 6	technology demonstrated in relevant environment
TRL 7	system prototype demonstration in operational environment
TRL 8	system complete and qualified
TRL 9	actual system proven in operational environment