IDEal: A Legal Development Environment

Australian Submission

2020

Mamta Thaker — Joanna Chen — Mirhady Dorodjatun — Joshua Fourie

Contents

Part I

Theory

A legal matter processed in IDEAL traverses four states:

- *Generation* ($\mathcal{G}$).

- *Representation* ($\mathcal{R}$).

- *Transformation* ($\mathcal{T}$).

- *Presentation* ($\mathcal{P}$).

We can understand IDEAL as a system of *plug-ins* which either generate, or commonly access and transform a unified representation of a legal matter into derivative states. Consequently, the system is a series of machines mapping [$\alpha_i \in \mathcal{G} \cup \mathcal{T} \cup \mathcal{P}$] $\rightarrow \mathcal{R}$, or $\mathcal{R} \rightarrow$ [$\beta_i \in \mathcal{G} \cup \mathcal{T} \cup \mathcal{P}$]. We denote $\langle \alpha_i \rangle$ as a *generator* state which can generate $\mathcal{R}$, and β_i as a *producible* state, which can be produced by some action on $\mathcal{R}$.
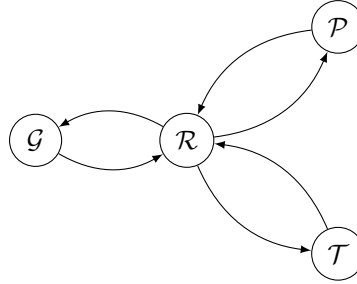

Figure 1: Visualising the interactions between the states.

In this section, we define $\mathcal{R}$, and provide an example of $(\mathcal{G} \rightarrow \mathcal{R})$ and $(\mathcal{R} \rightarrow \mathcal{P})$.

1 Defining the Representation

Our goal is to define a mathematical structure for $\mathcal{R}$ which encodes legal information and maximises the number of producible states. Given that the representation is driven by the encoded legal information, we begin with the idea that a lawyer is a mechanism for analysing the *existence* or *non-existence* of a legal relationship between *objects*.

Facts, Nodes and the NodeState. We describe a legal object as a `Node`, and define the admissible *factual* actions, attributes or character of the `Node` as the `NodeState`. The `NodeState`$_n$ is an unordered

set of n `Fact` objects which are generated and edited by an oracle, $\mathcal{F}$, in combination with a `SourceOfLaw`:

$$\mathcal{F} : (\texttt{SourceOfLaw}, \{ \texttt{Fact}_0, .., \texttt{Fact}_n \}) \rightarrow \texttt{NodeState}_n \tag{1.1}$$

$$\mathcal{F} : (\texttt{Fact}, \texttt{NodeState}_n) \rightarrow \texttt{NodeState}_{n'} \tag{1.2}$$

$$\mathcal{F} : \texttt{NodeState}_n \rightarrow \texttt{NodeState}_{n''} \tag{1.3}$$

Sources of Law. A `SourceOfLaw` defines the conditions, attributes or characteristics which are required for the oracle to generate both a `NodeState`, as well as the `Role` (1.1) associated with a `NodeState`. Concretely, these are objects such as legislation or common law which have the inherent capacity to generate legal rights or obligations relevant to an object.

1.1 The Role Object

A `Role` defines the *legal personality* of a `Node` by capturing the attributes, actions, or characterstics attributable under law. A `Node` can be subject to multiple `Role` objects of arbitrary complexity, provided they are distinct under (1.6). The `Role` associated with a `Node` is generated by a pair (`NodeState`, `SourceOfLaw`) under the oracle function, $\mathcal{F}$, and is transformable under the `Consequence` of a `Link` (1.2):

$$\mathcal{F} : \big[\ (\texttt{NodeState}, \texttt{SourceOfLaw}) \rightarrow \texttt{Role}\ \big] \tag{1.4}$$

$$\texttt{Consequence} : \big[\ \texttt{Role} \rightarrow \texttt{Role'}\ \big] \tag{1.5}$$

Equivalence of Roles. We define an equivalence relation on a pair (`Role`$_i$, `Role`$_j$) by comparing their generating states, such that they are only pairwise distinct where the generative facts and law diverge:

$$\big[\ \texttt{Role}_i = \texttt{Role}_j\ \big] \iff \big[\ (\texttt{NodeState}_i \iff \texttt{NodeState}_j) \wedge (\texttt{SourceOfLaw}_i \iff \texttt{SourceOfLaw}_j)\ \big] \tag{1.6}$$

Role Extension. The `NodeState` of a `Node` can generate multiple `Role` objects *iff* the (`NodeState`, `SourceOfLaw`) pair are distinct under (1.6). A `Role` is *reducible* when a subset of the generative `NodeState` can produce another distinct `Role`:

$$\big[\ N := \{\texttt{Fact}_0, .., \texttt{Fact}_n\}\ \big] \wedge \big[\ M := \{\texttt{Fact}_0, .., \texttt{Fact}_m\}\ \big] : \big[\ M \subset N\ \big] \tag{1.7}$$

$$\big[\ \texttt{NodeState}_i = \mathcal{F}(N, \texttt{SourceOfLaw}_i)\ \big] : \big[\ \texttt{Role}_i = \mathcal{F}(\texttt{NodeState}_i, , \texttt{SourceOfLaw}_{i'})\ \big] \tag{1.8}$$

$$\big[\ \texttt{NodeState}_j = \mathcal{F}(M, , \texttt{SourceOfLaw}_j)\ \big] : \big[\ \texttt{Role}_j = \mathcal{F}(\texttt{NodeState}_j, , \texttt{SourceOfLaw}_{j'})\ \big] \tag{1.9}$$

$$\implies \big[\ N \text{ is } \textit{reducible}\ \big] \tag{1.10}$$

A `Role` which is reducible is an *extension* of another `Role`, and the `Role` objects which are extended are called the *components of the extension*. The extended `Role` will automatically import any components

of the extension objects into its own definition. We denote an extension using subset notation, such that the following indicates Role_i is an extension of Role_j:

$$\text{Role}_j \subset \text{Role}_i \tag{1.11}$$

$$[\,\text{Role}_j \subset \text{Role}_i\,] : [\,\text{Role}_i \implies \text{Role}_j\,] \tag{1.12}$$

Given that an extension implies any components of the extension, a `Node` with multiple `Role` objects *may* replace any `Role` with an extension. We distinguish a `Role` extension from a `Role` *composition*, which is a set of distinct `Role` objects where there there does not exist a `Role` in the composition which is an extension of any other `Role`:

$$[\,N := \{\,\text{Role}_0, .., \text{Role}_n\,\}\,] : [\,\nexists(\text{Role}_i, \text{Role}_j) \in N : \text{Role}_i \subset \text{Role}_j\,] \tag{1.13}$$

1.2 Links

A `Link` is a directed, pairwise relationship between a source (Role_i) and a destination (Role_j), which has been generated with the assistance of a `SourceOfLaw`. Given a pair ($\text{Role}_i, \text{Role}_j$) and an associated `SourceOfLaw`, the oracle may draw a $\text{Link}_{i \to j}$:

$$\mathcal{F} : (\text{SourceOfLaw, Role}_i, \text{Role}_j) \to \text{Link}_{i \to j} \tag{1.14}$$

Hooking a Link. The `Hook` and `Anchor` objects are a specialisation of the `Role` object which represent the directed, relational requirements of a `Link`, and are generated under a (`SourceOfLaw, Role`) pairing. The oracle defines a `Link` by consuming a (`SourceOfLaw, Hook, Anchor`) triple:

$$[\,\mathcal{F}(\text{SourceOfLaw, Role}_i) = \text{Hook}\,] \,\wedge\, [\,\mathcal{F}(\text{SourceOfLaw, Role}_j) = \text{Anchor}\,] \tag{1.15}$$

$$\implies \mathcal{F} : (\text{SourceOfLaw, Role}_i, \text{Role}_j) \to (\text{Hook, Anchor}) \tag{1.16}$$

$$\mathcal{F} : (\text{SourceOfLaw, Hook, Anchor}) \to \text{Link}_{i \to j} \tag{1.17}$$

1.3 Consequence

Given a `Link` to a relationship, the oracle, $\mathcal{F}$, generates a `Consequence` from a `SourceOfLaw`:

$$\text{Link}_{i \to j} \implies [\,\mathcal{F} : \text{SourceOfLaw} \to \text{Consequence}\,] \tag{1.18}$$

A `Node` may be modified under the `Consequence` of a `Link`, changing the `NodeState` or associated `Role`:

$$[\,\mathcal{F}(\text{Consequence, Node}) = \text{Node'}\,] \implies \{\,[\,\text{NodeState} \to \text{NodeState'}\,] \vee [\,\text{Role}_i \to \text{Role}_i'\,]\,\} \tag{1.19}$$

The oracle subsequently *walks the consequence forward* to generate or decouple any `Hook` or `Anchor` objects which have been invalidated by a disturbance of the `Conditions` required by the `SourceOfLaw`.

1.4   Sources of Law

Given a `SourceOfLaw`, there exists a related set of `Conditions` which must be satisfied before the oracle function, $\mathcal{F}$, can evaluate any expression:

$$\texttt{SourceOfLaw} \implies \texttt{Conditions} \tag{1.20}$$

$$\mathcal{F} : \texttt{Conditions} \rightarrow \{\, T, F \,\} \tag{1.21}$$

Application to Links.   The `Hook` or `Anchor` required to draw a `Link` will have `Conditions` containing evaluations related to the `NodeState` of an object. The oracle, $\mathcal{F}$, will only generate a `Link` where the `Conditions` imposed by the `SourceOfLaw` are satisfied. Consequently, a `Link` will 'decouple' where a `Consequence` modifies the `NodeState` such that the evaluation of the `Conditions` of the relevant `SourceOfLaw` fail.

1.5   The Stage Controller

Time is encoded in $\mathcal{R}$ as an ordered set of n distinct `Interval` objects related by a `Transition`:

$$\texttt{IntervalSet} := \{\, \texttt{Interval}_0, .., \texttt{Interval}_n \,\} \tag{1.22}$$

$$\texttt{ForwardTransition} : \texttt{Interval}_i \rightarrow \texttt{Interval}_{i+1} \tag{1.23}$$

$$\texttt{BackwardTransition} : \texttt{Interval}_i \rightarrow \texttt{Interval}_{i-1} \tag{1.24}$$

The `BackwardTransition` and `ForwardTransition` are a specialisation of a `Transition` that form an identity map under function composition:

$$\texttt{BackwardTransition} \circ \texttt{ForwardTransition} : a \rightarrow a \tag{1.25}$$

The `TransitionSet` of an `Interval` is the minimum set of `ForwardTransition` and `BackwardTransition` objects required to reconstruct the `IntervalSet` under the oracle function, F:

$$\mathcal{F} : (\texttt{TransitionSet}, \texttt{Interval}) \rightarrow \texttt{IntervalSet} \tag{1.26}$$

Example.

Consider a boundary dispute between $\mathcal{A}lice$ and $\mathcal{B}ob$ on two adjacent plots of land: $\mathcal{L}_a$ and $\mathcal{L}_b$. In this case, $\mathcal{A}lice$ is the owner of $\mathcal{L}_a$, and $\mathcal{B}ob$ is the owner of $\mathcal{L}_b$. We are attempting to define whether it is permissible for $\mathcal{A}lice$ to build a structure by defining a boundary line delineating the properties, called `Bound`. The graphs below represents the information encoded in $\texttt{Interval}_0$.

Diagram 1.1 *$\mathcal{A}lice$ owns a parcel of land.*
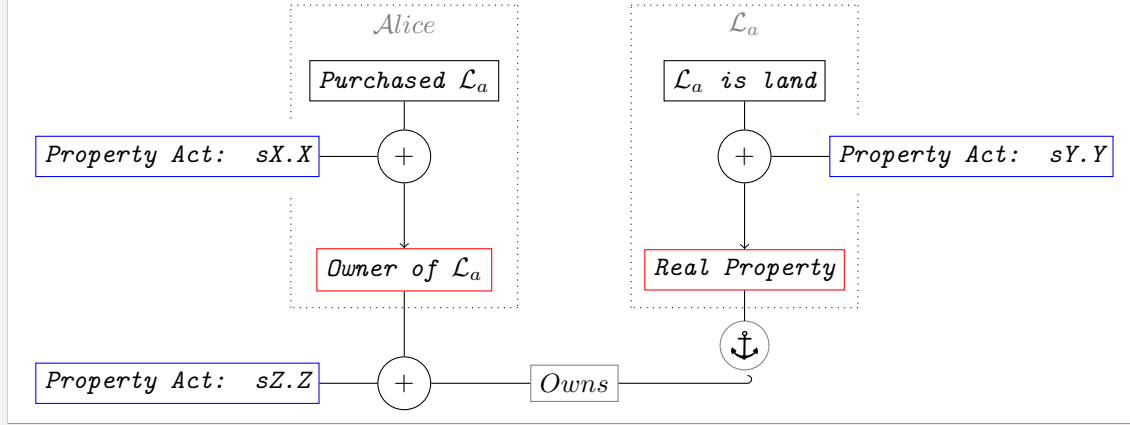
The above diagram (**1.1**) reflects the following:

1. $\mathcal{A}lice$ and $\mathcal{L}_a$ are both a `Node`.

2. [*Purchased $\mathcal{L}_a$*] and [*$\mathcal{L}_a$ is land*] are both instances of a `Fact`.

3. (*Property Act* : *sX.X, Purchased $\mathcal{L}_a$*) generates the `Role` [*Owner of $\mathcal{L}_a$*].

4. (*Property Act* : *sY.Y, $\mathcal{L}_a$ is land*) generates the `Role` [*Real Property*].

5. The (`Hook`, `Anchor`) pairing below generates a `Link` [*Owns*]:

   (a) (*Property Act* : *sZ.Z, Owner of $\mathcal{L}_a$*) generates a `Hook`.

   (b) [*Real Property*] generates an `Anchor`.

Compliance and Deviation. A lawyer constructs a `Link(Consequence)` by analysing the `Role` of an object relative to a set of legal conditions. We define both compliance and deviation as:

$$[\, \exists\, \texttt{Link(Existence, Consequence)}\,] \wedge \begin{cases} [\, \texttt{Role} \implies \texttt{Link(..)}\,], & \text{Compliance} \\ [\, \texttt{Role} \centernot\implies \texttt{Link(..)}\,], & \text{Deviation} \end{cases} \tag{1.27}$$

> **Example.** Continuing the previous example, we define compliance as Bob enforcing the terms of C against Alice, because Alice fulfils the relevant `Role`. Bob could not, however, enforce C against a third party, unless they also fulfilled a role which generated a `Link` to C.