On the Formal Representation of the Australian Spent Conviction Scheme

Guido Governatori[1,2] Pompeu Casanovas Romeu[2,3] and Louis de Koker[2]

[1] Data61, CSIRO, Australia
[2] Law School, La Trobe University, Australia
[3] Institute of Law and Technology, Autonomous University of Barcelona, Spain

Abstract. We discuss how to use Defeasible Deontic Logic to provide a formal representation of the Commonwealth of Australia spent conviction schema (Part VII C of the Crimes Act (1914)). The formalisation is directly written and implemented in Turnip (a modern implementation of Defeasible Deontic Logic).

1 Background: Spent Conviction

A "spent conviction" is a conviction that becomes hidden from public view after a set period of time but, depending on certain factors, still remains accessible for specific (public) purposes by specific interested parties. These schemes are mainly focused on convictions for less serious crimes and generally do not extend to convictions for violent sexual offences. The set period of time is also extended where the person has re-offended during the set period.

Australia has a spent conviction scheme operating at a Commonwealth level and Territories and States also have schemes, but the nature and rules of these schemes differ. Each regime has exemptions which permit the lawful disclosure of spent convictions in certain limited circumstances. These exemptions usually relate to employment in particularly sensitive positions (e.g., on application for appointment as a police official, teacher, or childcare worker). In this sense, unless an ex-offender falls within an exemption, spent conviction schemes operate to encourage the rehabilitation of ex-offenders and to reduce the potential for ongoing punishment or discrimination against them [6].The paper focuses on elements of the Commonwealth spent conviction scheme National Crime Check describes a "spent conviction" as follows for purposes of this scheme:

A "spent conviction" is a conviction of a Commonwealth, Territory, State or foreign offence that satisfies all of the following conditions: (i) it is 10 years since the date of the conviction (or 5 years for juvenile offenders); AND (ii) the individual was not sentenced to imprisonment or was not sentenced to imprisonment for more than 30 months; (iii) AND the individual has not re-offended during the 10 years (5 years for juvenile offenders) waiting period; (iv) AND a statutory or prescribed exclusion does not apply. (A full list of exclusions is available from the Office of the Australian Information Commissioner).[1]

[1] `https://www.nationalcrimecheck.com.au/resources/spent_convictions_information`

Exchanges of criminal records data among the jurisdictions in Australia are coordinated by and through the Australian Criminal Intelligence Commission (ACIC). It manages the processes and provides the system through which Australian police agencies and accredited bodies submit nationally coordinated criminal history checks. The ACIC operates the National Police Checking Service that assists organisations to screen and make informed decisions for example about prospective employees and volunteers, visa and citizenship applications and work-related due diligence relating to national security. The service is used by 251 accredited agencies and bodies. During the period 2017–18 the number of checks processed increased by 11.1% to 5.29 million, and 1.49 million checks were referred to police agencies for further assessment to determine whether the information may be disclosed in accordance with their spent convictions legislation and/or information release policies.[2] The aim of the paper is to explore the possibilities to automatise or, better, semi-automatise the service described above. The extensive number of checks referred to police agencies is directly linked to the complexity of the regime and inconsistencies among the different jurisdictional schemes. Accordingly, the objective of this work was to produce a proof of concept to partially model project use case solutions relating to the Spent Convictions Scheme, to lessen the current pressure on officials who need to process the checks.

For the modelling required to implement the proof of concept we selected Part VIIC (Pardons, Quashed Convictions and Spent Conviction) of the *Crimes Act 1914* (Cth). modelling of the Part of the Act we selected Defeasible Deontic Logic (DDL) for its ability:

- to integrate reasoning with exceptions,
- to model deontic concepts such us obligations, permissions, prohibitions, and
- to represent both prescriptive norms and definitional norms.

All such elements are present in the Commonwealth spent conviction scheme as set out in the *Crimes Act 1914* (Cth). The aim of the modelling was to understand to what extent formal models are suitable to represent legislation and if they offer suitable environment to support legal decision. The encoding of the selected section was done in the Turnip language and reasoner that implements DDL. The encoding was then tested with a few examples to provide an initial, small scale, validation of the approach.

2 Defeasible Deontic Logic

In this section we provide a brief outline of Defeasible Deontic Logic, for the full details of the logic see [4]. DDL is an extension of the Defeasible Logic [1] and [5] combining defeasibility for the natural handling of exceptions, deontic modalities for modelling legal provisions about obligations, prohibitions and permissions,

[2] ACIC. Annual Report 2017–18 `https://www.acic.gov.au/25-national-information-and-intelligence-sharing-services-0`.

and a non-classical compensation operator to model obligations in force after a violation. Also, DDL has been used for applications in the legal domain [3].

In DDL a rule is an IF... THEN ... statement where the IF part encodes the condition of applicability of the rule (where in general a rule corresponds to a norm or a part of a norm), and the THEN part models the effect of the norm. Rules can then be divided in constitutive rules that are used to provide the definitions of the terms used in a normative document, and normative rules that give the conditions (IF part) under which legal requirements (i.e., obligations, permissions, prohibitions, ...), THEN part, are in force. In DDL rules can be further classified according to their strength: thus we have strict rules, defeasible rules and defeaters. A strict rule is a rule in the classical sense. Defeasible rules are rules subject to exceptions: the conclusion of the rule holds unless there are other (applicable) rules (for the same conclusion) that defeat the rule. Finally, defeaters are a special kind of rules, they do not support conclusions, but prevent the conclusion to the opposite.

Constitutive rules are rules in standard defeasible logic, while for normative rules we consider prescriptive rules (setting when something is obligatory/forbidden) and permissive rules (rules making something explicitly permitted derogating rules for prohibitions or obligations to the contrary). Normative rules have the following form:

$$r\colon A_1, \ldots, A_n \hookrightarrow_{\Box} C_1 \odot \cdots \odot C_m$$

where $A_1, \ldots, A_n$ are the conditions of applicability of the rule and are expressed as literals or deontic literals (a literal in the scope a deontic modality: obligation O or permission P), $\Box$ is one of the deontic modalities, and the C_i are literals ($C_1 \odot \cdots \odot C_m$ is called a reparation chain). The mode of the rule $\Box$ determines the scope of the conclusion. In case the mode is O the meaning of the right-hand side of the rule is that when the rule applies $\mathsf{O}C_1$ is in force (C_1 is obligatory), and if it is violated, i.e., $\neg C_1$ holds, then $\mathsf{O}C_2$ is in force (C_2 is obligatory), and C_2 compensates for the violation of $\mathsf{O}C_1$. We can repeat the reasoning when $\mathsf{O}C_2$ is violated. The reasoning mechanism of DDL extends the proof theory of Defeasible Logic [1] and it is based on an argumentation like schema. To prove a conclusion, there must be an applicable rule for the conclusion we want to prove. A rule is applicable if all the elements of the antecedent of the rule hold (have already been proved). In addition, all counter-arguments are either rebutted or defeated. A counter-argument is a rule for a conflicting conclusion (the negation of the conclusion, or in case of deontic conclusions, conflicting deontic modalities). A counter-argument is rebutted if some of its premises do not hold (we proved that the premises do not hold) and the counter-argument is defeated when the rule is weaker than an applicable rule for the conclusion. For reparation chains, in addition to the normal defeasibility conditions, to prove $\mathsf{O}C_k$ we require that for all the elements before it in the chain, we are able to prove $\mathsf{O}C_l$ and $\neg C_l$.

Turnip[3] is a modern (typed) functional programming implementation of DDL written in Haskell. The aims of Turnip is to provide a reference implementation

[3] An online environment to run Turnip rulesets, with samples of the features it offers is available at `http://turnipbox.netlify.com/`.

of DDL and at the same time to offer features to facilitate the encoding of norms as rules. Turnip requires that the terms used in a set of rules are defined before they are used, where the definition of a term has the following form:

```
Type Name description_string
```

where `Type` is either a Boolean (`Atom`, string, numeric, date, datetime or duration. The description string is optional, and its main use is to provide the meaning of the term in natural language. Arithmetic operators (i.e., `+, -, *, /,`) can be used for numeric terms and values and comparison operators (i.e., `==, !=, <, <=, >, >=`) to create boolean types from numeric and duration terms. Also, Turnip provides conversion functions (e.g., `interval, toDays, after`) to operate on dates, times and duration terms. For example, the `interval` function takes two dates as input and returns a duration. Consider the snippet

```
case.date := 2019-09-01
conviction.date := 2010-12-03
interval(case.date,conviction.date)>=5y
```

where we use the assignment operator `:=` to give values to two terms of type date; then, we use the `interval` operator to compute the duration (time elapsed between the two dates), and we compare it with a given duration (5 years). Rules consist of a label (optional), a condition list, an arrow and a conclusion list.

```
label: condition_list => conclusion_list
```

where the arrow determines the type of rule (strict, defeasible or defeaters). Rules are meant to represent norms and it is reasonably common that a norm prescribes multiple (simultaneous) effects; similarly, the same effect can be prescribed by different norms. To ease the effort of writing the rules encoding such norms, the condition list can be either a conjunction (`&`) or a disjunction (`|`) of Boolean, while the conclusion list is either an assignment, a single Boolean, a conjunction of assignments, or a conjunction of Boolean. For the Boolean expression Turnip, in addition to Atom, their negation `~`, and expression constructed from numeric and temporal expression allows for deontic expressions, where a deontic expression is obtained from the combination of one the following deontic modalities `[O]`, `[P]`, `[F]`, `[E]` (standing, respectively for Obliged, Permitted, Forbidden, Exempt) and an atom. For the modalities, notice that: `[F]A` is equivalent to `[O]~A` (and `~[P]A`) and `[E]A` is equivalent to `[P]~A`.

3 Formal Modelling of the Spent Conviction Schema

The encoding of Part VIIC requires the extraction of the terms (or atoms in DDL parlance), corresponding to the concepts, used in the legislation. While the Turnip language supports different data types (e.g., Boolean, numeric, date and time, duration, ...), for the encoding of this part we only needed to use Boolean, duration and date. The representation of each atom comes with its textual description providing the meaning in natural language of the term. The atoms encode either factual information relevant for a case (e.g., the date when a person was convicted for an offence, or the whether a person was convicted or found guilty of an offence) or for information that can be obtained based on the

conditions defined in the Act (whether the waiting period for an offence ended). For example, we can create the following atoms

```
Date conviction.date "the date when the person was convicted"
Date case.date "the date when the current case is dealt with"
Atom minor "the person was a minor when the offence was dealt with"
Atom WaitingPeriodEnded "the waiting period for the offence ended"
```

that can then be used by the following constitutive rules

```
wp1: interval(conviction.date,case.date)>=5y & minor
    => WaitingPeriodEnded
wp2: interval(conviction.date,case.date)>=10y  => WaitingPeriodEnded
```

encoding the definition of waiting period given in Division 1 of Part VIIC, namely:

waiting period, in relation to an offence, means:

(a) if the person convicted of the offence was dealt with as a minor in relation to the conviction—the period of 5 years beginning on the day on which the person was convicted of the offence; or

(b) in any other case—the period of 10 years beginning on the day on which the person was convicted of the offence.

Similarly, Section 85ZM on the meaning of conviction and spent conviction, i.e.:

(1) For the purposes of this Part, a person shall be taken to have been convicted of an offence if:
   (a) the person has been convicted, whether summarily or on indictment, of the offence;
   (b) the person has been charged with, and found guilty of, the offence but discharged without conviction; or
   (c) the person has not been found guilty of the offence, but a court has taken it into account in passing sentence on the person for another offence.

(2) For the purposes of this Part, a person's conviction of an offence is spent if:
   (a) the person has been granted a pardon for a reason other than that the person was wrongly convicted of the offence; or
   (b) the person was not sentenced to imprisonment for the offence, or was not sentenced to imprisonment for the offence for more than 30 months, and the waiting period for the offence has ended.

can be encoded by the following atoms and (constitutive) rules:

```
Atom Person "an individual not a body corporate"
Atom ConvictionVII "a conviction according to Part VII"
Atom Conviction "a person has been convicted for an offence"
Atom Guilty "a person has been charged and found guilty"
Atom Discharged "a person has been discharged without a conviction"
Atom OtherOffence "a person has not been found guilty, but the court
  has taken it into account (for the conviction) for another offence"

s85ZM_1a: Person & Conviction => ConvictionVII
s85ZM_1b: Person & Guilty & Discharged => ConvictionVII
s85ZM_1c: Person & OtherOffence => ConvictionVII
```

where the ConvictionVII atom corresponds to the "institutional" fact that an event counts as a conviction for the purpose of applying Part VIIC to that event.

```
Atom SpentConviction "a conviction is considered spent"
Atom Pardon "a person has been granted a pardon"
Atom Imprisonment "a person was sentence to imprisonment"
Duration imprisonment_term "the length of the imprisonment"

s85ZM_2a: ConvictionVII & Pardon => SpentConviction
s85ZM_2b1: ConvictionVII & ~Imprisonment & WaitingPeriodEnded
    => SpentConviction
s85ZM_2b1: ConvictionVII & imprisonment_term <= 30m
    & WaitingPeriodEnded => SpentConviction
```

As one can notice, the rules in DDL bear a close resemblance with the textual provisions they are meant to encode. For examples of prescriptive rules we can consider the rules encoding Section 85ZS(1a-b)

(1) Subject to Division 6, but despite any other Commonwealth law or any State law or Territory law, where, under section 85ZR, a person is, in particular circumstances or for a particular purpose, to be taken never to have been convicted of an offence:
   (a) the person is not required, in those circumstances or for that purpose, to disclose the fact that the person was charged with, or convicted of, the offence;
   (b) it is lawful for the person to claim, in those circumstances, or for that purpose, on oath or otherwise, that he or she was not charged with, or convicted of, the offence;

```
s85ZS_1a: Person & PardonOrWronglyConvicted
    => [E] Disclose.charged & [E] Disclose.conviction
s85ZS_1b: Person & PardonOrWronglyConvicted
    => [E] Oath.not_charged & [E] Oath.not_conviction
```

Section 85ZW on Effect of right of non-disclosure

(b) anyone else who knows, or could reasonably be expected to know, that section 85ZV applies to the person in relation to the offence shall not:
   (i) without the person's consent, disclose the fact that the person was charged with, or convicted of, the offence to any other person, or to a Commonwealth authority or State authority, where it is lawful for the first-mentioned person not to disclose it to that other person or that authority;

is then encoded as follows:

```
Atom ExpectedKnow85ZV "the other entity knows or could reasonably be
      expected to know that Section 85ZW applies"

s85ZW_b: OtherEntity & ExpectedKnow85ZV & [E]Disclose.convictionInfo
      => [F] OtherDisclose.conviction & [F] OtherDisclose.charged
```

As Section 85ZS(1) recites, there are exemptions to the provisions allowing a person the right of non-disclosure (and preventing other entities to disclose such information). For instance, one can consider Subdivision B of Section 6, where Section 85ZZH Exclusions (a) and (g) provide that:

Division 3 does not apply in relation to the disclosure of information to or by, or the taking into account of information by a person or body referred to in one of the following paragraphs for the purpose specified in relation to the person or body:
(a) a law enforcement agency, for the purpose of making decisions in relation to prosecution or sentencing or of assessing:
   (i) prospective employees or prospective members of the agency; or

(ii) persons proposed to be engaged as consultants to, or to perform services for, the agency or a member of the agency;

...

(g) Commonwealth authority, for the purpose of assessing appointees or prospective appointees to a designated position;

These provisions can be modelled by the following DDL rules:

```
s85ZZHa_2: LawEnforcementAgency & PurposeOfEngagementWithAgency =>
    [O] Disclose.charged & [O] Disclose.conviction &
    [P] OtherDisclose.conviction & [P] OtherDisclose.charged
s85ZZHh: CommonwealthAutority & PurposeOfEngagementWithAgency =>
    [O] Disclose.charged & [O] Disclose.conviction &
    [P] OtherDisclose.conviction & [P] OtherDisclose.charged
```

where `CommonwealthAutority` and `LawEnforcementAgency` are defined by constitutive rules based on the definition of the terms in Section 85ZL. Similarly, `PurposeOfEngagementWithAgency` can be defined by an auxiliary constitutive rule based on the conditions in 85ZZH(a)(ii).

The rules for Section 85ZS and Section 85ZZH are in conflict with each other, DDL provides a mechanism (called superiority relation) to solve the conflict. Specifically, rule `s85ZZHa_2` overrides rule `s85ZS_1a` (same for rule `s85ZZhh`):

```
s85ZZHa_2 >> s85ZS_1a
s85ZZHh >> s85ZS_1a
```

Thus, in case both rules apply, i.e., a person who received a pardon for an offence, seeking to work for the Australian Federal Police has to disclose the conviction. Consider a case related to the vetting of Person A's appointment as a management consultant to the Australian Federal Police. Person A had two prior convictions for insider trading in 1998 and had been released on entering into a good behaviour bond for two years. For the sample case we have to determine (1) it there was a conviction, (2) whether the conviction has been spent or not, and in case the conviction is spent (3) to assess if any exclusions under Section 6 apply. Accordingly, we encoded the facts of the cases in Turnip as follows:

```
Person // Person A
Conviction // there was a conviction for insider trading
conviction.date := 1998-08-10 // when A was convicted
case.date := 2019-09-22 // when the case was examined
Guilty // A was found guilty
Discharged // A was discharged with 2y good conduct bond
~Imprisonment // A was discharged with 2y good conduct bond
proposedConsultant // A seeks to work as proposed consultant for
AustralianFederalPolice // the Australian Federal Police
```

based on the above facts, we obtain the following conclusions (excluding the given facts)

```
[O]Disclose.charged                 CommonwealthAutority
[O]Disclose.conviction              ConvictionVII
[P]OtherDisclose.charged            LawEnforcementAgency
[P]OtherDisclose.conviction         SpentConviction
                                    WaitingPeriodEnded
```

showing that the waiting period for the conviction ended and so the conviction is spent; but the person has to disclose the information for his application to work

as consultant and other entities (expected to know the conviction) are permitted to disclose the information to the relevant authority or agency (in this case the Australian Federal Police).

The encoding for Part VIIC in Turnip, with the facts of the case, is available at `https://turnipbox.netlify.com/fiddles/fcA5uXkUnQ0y4B9uVkIq`.

4 Conclusions and future work

This paper reported on the formal representation of elements of the Australian Spent Conviction Scheme, developed in the course of research of the Australian government-funded Data to Decisions Cooperative Research Centre (D2D CRC). The project [2], which concluded in June 2019 , focused on specific spent convictions use cases selected by the ACIC to produce a proof of concept on Compliance through Design (CtD) modelling [7].

The main conclusion is that the modelling of the Spent Convictions scheme can be used to process the most common cases at the federal level. The encoding has proved to be consistent both at the formal and empirical levels in the most common cases. But its implementation with legal effects in more complex scenarios would require embedding it into a broader legal context. This alignment between the extracted rules with the legal conceptual analysis for decision-making purposes at Commonwealth and State and Territory levels constitute the challenge that we are going to face in the immediate future.

References

1. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic **2**(2), 255–287 (2001)
2. Casanovas, P., de Koker, L., Keyzer, P., Mendelson, D., Watts, D., Barnes, J., O'Toole, S., Stammers, M., Parsons, D., Governatori, G., Rodríguez-Doncel, V., Hashmi, M., Gonzalez-Conejero, J.: Summary Legal and Technical Report on Spent Convictions. D.C3.7. DC25008. `doi://10.5281/zenodo.3271525` (Jul 2019)
3. Governatori, G.: Practical normative reasoning with defeasible deontic logic. In: d'Amato, C., Theobald, M. (eds.) Reasoning Web 2018, pp. 1–25. No. 11078 in LNCS, Springer International Publishing, Cham (2018)
4. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S.: Computing strong and weak permissions in defeasible logic. Journal of Philosophical Logic **42**(6), 799–829 (2013)
5. Governatori, G., Rotolo, A.: Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations. Australasian Journal of Logic **4**, 193–215 (2006)
6. Paterson, M., Naylor, B.: Australian spent convictions reform: A contextual analysis. UNSW Law Journal **34**, 939 (2011)
7. Stumptner, M., Mayer, W., Grossmann, G., Liu, J., Li, W., Casanovas, P., de Koker, L., Mendelson, D., Watts, D., Bainbridge, B.: An architecture for establishing legal semantic workflows in the context of integrated law enforcement. In: AICOL 2015–2017. LNCS, vol. 10791, pp. 124–139. Springer (2017)