Formalizing Legislation in the Event Calculus:
The Case of the Italian Citizenship Law

MSc Thesis (*Afstudeerscriptie*)

written by

Marcello Di Bello
(born October 13, 1982 in Milano, Italy)

under the supervision of **prof dr Michiel van Lambalgen**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
June 13, 2007	prof dr Joost Breuker
	prof dr Peter van Emde Boas
	prof dr Michiel van Lambalgen
	prof dr Frank Veltman

Institute for Logic, Language and Computation

Contents

1 Introduction

This thesis explores to what extent legal knowledge and legal reasoning can be encoded in the Event Calculus (hereafter, EC), and how EC needs to be extended (if at all) to accommodate legal reasoning and knowledge. The locutions 'legal knowledge' and 'legal reasoning' gather a variety of disparate referents. A working hypothesis of this thesis is that a considerable portion of legal knowledge and reasoning is contained in legislative texts (laws, decrees, regulations, directives, etc.). The primary task will thus be to formalize a piece of legislation in the language of EC.

The reasons for wanting to formalize a piece of legislation with the aid of a formal language like EC are numerous. I would like to remind the reader of three of them. First, formalization enhances the precise understanding of a text, in the sense that it helps single out ambiguities, unintended readings or interpretations, etc. Second, when an actual reasoning implementation is made available for the formalized piece of law, one can perform reasoning tasks, such as consistency checking, computation of implicit inferences, and instance checking or instance enumeration. Evidently, these are very valuable aids for law drafters. Third, the formalization of a piece of text is an enterprise that falls under the field of formal semantics as well. Legal texts might offer peculiar or new linguistic data that are in need of explanation by formal semanticists.

Several pieces of legislation have been formalized by researchers at the Imperial College London (see [14, 23, 3]) as early as the eighties. The most famous of these attempts, the formalization of the *British Nationality Act* in Sergot et al. [23], dates back to 1986. The language adopted to carry out the formalization was logic programming,[1] on the assumption that logic programming and legal texts share many affinities (see Kowalski [18]). The same assumption is endorsed in this thesis, and is pushed even further, by using a logic programming-based formalism, the EC, to formalize legislation. I will not argue abstractly, or by means of toy examples, whether or not EC is suitable for modeling the law. A great deal of evidence for applying EC to the legal domain is provided by

[1] An alternative approach uses description logic and OWL (Ontology Web Language) to formalize legislation and legal concepts; see Hoekstra et al. [13], and Breuker et al. [4]. For reasons of time and space, I will not be able to compare this approach with the one proposed here.

chapters 3 and 4. The former considers some recurrent features of legislative texts and shows how they can be formally represented in EC. Chapter 4 is more empirical. It considers as a case-study the Italian Citizenship Law [2], and a sample of the formalization is given and discussed in detail.

It was said at the beginning that a working hypothesis of this thesis is that legal knowledge and reasoning is contained in legislative texts. While this is certainly the case, one should not be induced to think that legal texts are self-sufficient, or that by formalizing a piece of law on Italian citizenship, it will become possible to compute mechanically whether or not someone qualifies for Italian citizenship. This is over-simplistic: the process of applying and interpreting the law plays a crucial role in making the law usable to deliver conclusions such as 'X is an Italian citizen'. Chapter 5 applies the formalization of the Italian Citizenship Law to a concrete case, which was discussed before a court of law. The chapter investigates how additional interpretative principles can be added to the formalization to yield or block certain conclusions from being drawn.

Before engaging into the formalization of the law, the next chapter gives a brief introduction to EC.

2 The Event Calculus

This chapter introduces the Event Calculus,[1] its syntax and its proof-system.[2] The Event Calculus (hereafter, EC) was initially designed to model commonsense knowledge and reasoning about actions and events (see Kowalski & Sergot [16] and Shanahan [24]). In order to achieve this, EC is divided into two parts: one is a general axiomatic theory (call it EC-$\mathcal{A}$) about events and causality; the other is given by micro-theories or scenarios that are specific to the domain being modeled. To illustrate, the knowledge and the reasoning needed to crack an egg can be taken as an example.[3] The axioms of EC suffice to capture the general theory of causality involved in cracking an egg, but the domain-dependent knowledge such as 'cracking an egg will cause it to break' must be made explicit elsewhere. Hence, the set of axioms is augmented with an additional set of formulas which is called a *scenario*, following the terminology in [28]. This means that any EC-formalization will contain the axiomatic part EC-$\mathcal{A}$ and a scenario—the former being domain independent and the latter domain dependent.

2.1 Syntax

Formulas of EC are either axioms in EC-$\mathcal{A}$ or scenario formulas, written in an appropriate scenario language (call it EC-$\mathcal{S}$). The complex EC-$\mathcal{A}$ + EC-$\mathcal{S}$ makes up the language of EC. For expository purposes, it is useful to introduce an underlying language (call it EC-$\mathcal{L}$), of which both EC-$\mathcal{A}$ and EC-$\mathcal{S}$ are subsets. This underlying language is a multi-sorted rule language. It is a *multi-sorted* language since the entities talked about are not only individuals, but also events, fluents and quantities, as we shall see. It is a *rule* language since only rule-like formulas count as well-formed formulas. However, EC-$\mathcal{L}$ cannot be used directly as the language in which EC-formulas are written, because it allows the construction of formulas which may lead to loops in the computations. For this reason, well-formed formulas of EC are restricted to EC-$\mathcal{A}$ +EC-$\mathcal{S}$.

Although EC-$\mathcal{L}$ is too expressive and its subset EC-$\mathcal{S}$ + EC-$\mathcal{A}$ is sufficient for modeling purposes, it is useful to add an integrity constraint language (call

[1] I will base my presentation on van Lambalgen & Hamm [28].

[2] The semantics is left out as it is not essential to understand the material presented in this thesis. The interested reader is referred to [28].

[3] This example is used in Shanahan [25].

it EC-$\mathcal{IC}$) which plays a role in modeling statements with deontic operators.

The presentation of the language of EC will so proceed: first, the underlying language EC-$\mathcal{L}$; next, the axiom system EC-$\mathcal{A}$ as well as the language for scenarios EC-$\mathcal{S}$; finally, the integrity constraint language EC-$\mathcal{IC}$.

2.1.1 EC-$\mathcal{L}$

The *alphabet* consists of the symbols:

Constants in lower case letters, e.g. `acquiring`, `italy`;
Variables in upper case letters, e.g. `X`, `T`;
The symbols: $=, <, +, \times$;
The numerals: 0 and 1;
Connectives: $\neg, \wedge, \leftarrow$;
Universal quantifier: $\forall$;
Auxiliary symbols: $(,)$.

Terms are either atomic or complex. *Atomic* terms are:

Constants for individuals, quantities, events and fluents;
Variables for individuals, quantities, events and fluents;
1-place predicate `initially`;
2-place predicates `happen` and `holdAt`;
3-place predicates `initiate, terminate, clipped, release`;
4-place predicate `trajectory`.

Two observations are in order. First, I shall use the convention of denoting variables for time points by '`T, S, T1, T2`', and variables for fluents and events by '`F, F1, F2, E, E1, E2`'; the symbols '`t, s, t1, t1, f, f1, f2, e, e1, e2`' are to be understood as metavariables for time points, fluents and events constants. Second, constants and variables for individuals can occur as arguments of fluent or event terms; for example, '`citizen(X)`' is a one-argument fluent term, with '`X`' a variable for individuals. If constants or variables for quantities occur within a fluent, then the fluent is called parametrized; for example, '`distance(N)`' is a parametrized fluent, with '`N`' a variable for a quantity.

For *complex* terms, two recursive clauses apply. First, if t is an event term with k arguments, and $t_1, \ldots, t_k$ are fluent or event terms, or a combination of both, then $t(t_1 \ldots, t_k)$ is an event term. Second, if t is a fluent term with k arguments, and $t_1, \ldots, t_k$ are fluent or event terms, or a combination of both, then $t(t_1 \ldots, t_k)$ is a fluent term.

The distinction between fluents and events is crucial. Intuitively, the former are extended over time (they are states); the latter are 'happenings' and they possess a more point-wise nature. Examples are, respectively, '`know`' and '`come-to-know`'; '`be-injured`' and '`hit`'. In addition, fluents and events can be embedded into other fluents or events, following the definition of complex terms. For example, '`know(hit)`' or '`come-to-know(hit)`' are allowed as constructions.

Formula	Intended meaning
`initially(f)`	Fluent '`f`' holds from time '0'
`happen(e, t)`	Event '`e`' occurs at time '`t`'
`holdAt(f,t)`	Fluent '`e`' holds at time '`t`'
`initiate(e,f,t)`	Fluent '`f`' starts to hold after event '`e`' at time '`t`'
`terminate(e,f,t)`	Fluent '`f`' ceases to hold after event '`e`' at time '`t`'
`clipped(t1,f,t2)`	Fluent '`f`' ceases to hold between time '`t1`' and '`t2`'
`release(e,f,t)`	Fluent '`f`' is not subject to inertia after event '`e`' at time `t`'
`trajectory(f1,t1,f2,d)`	If fluent '`f1`' starts to hold at time' `t1`', then fluent '`f2`' starts to hold at time '`t1+d`'

Table 2.1: Informal explanation of the meaning of EC-$\mathcal{L}$ atomic formulas (see [24]). Bear in mind that, properly speaking, these atomic formulas receive their meaning only in the context of the axioms of EC.

Formulas are either literals or rules. A *literal* L is either an atomic formula A or a negated atomic formula $\neg A$. *Atomic* formulas are:

Equalities between constants or variables;
Formulas in the language of the structure $\langle \mathbb{N}, <, +, \times, 0, 1\rangle$;[4]
`initially(f), happen(e,t), holdAt(f,t)`;
`initiate(e,f,t), terminate(e,f,t)`;
`clipped(t1,f,t2), release(e,f,t)`; and
`trajectory(f1,t,f2,d)`,

where '`f,f1,f2`' are fluents, '`e`' is an event, and '`t,t1,t2,d`' are quantities.

Rules are formulas of the shape: $A \leftarrow L_1 \wedge \cdots \wedge L_k$, where A is an atom and $L_1, \ldots, L_k$ are literals. An alternative notation is $A \leftarrow L_1, \ldots, L_k$, where commas replace conjunctions. It is customary to call the consequent A the *head* of the rule; and the antecedents $L_1, \ldots, L_k$ the *body* of the rule. Note that any variable occurring in a rule is bound by a universal quantifier scoping over the entire rule. A rule without a body is called a *fact*.

Definition **1** (Formulas)**.** The set of EC-$\mathcal{L}$ formulas is the smallest set containing literals and rules.

The reader unfamiliar with EC may benefit from consulting table 2.1 which spells out informally the intended meaning of EC-$\mathcal{L}$ atomic formulas.

[4] In [28], the structure $\langle \mathbb{R}, <, +, \times, 0, 1\rangle$ is used instead. I have chosen to use $\mathbb{N}$ and not $\mathbb{R}$, because the quantities I will be concerned with are days only, and these can be represented discretely by the natural numbers.

2.1.2 Axioms EC-$\mathcal{A}$

Intuitively speaking the axioms EC-$\mathcal{A}$ formalize the notions of causality and change typically employed in everyday reasoning. These assume a naïve formulation of the principle of inertia, which says: Normally, things don't change; they only change under the influence of an event or a force. As a matter of fact, two notions of change are involved in EC-$\mathcal{A}$, *instantaneous* and *continuous* change. The former is captured by axioms 2 and 3, and the latter by axiom 4. Before explaining the difference further, I will give the list of axioms. Each axiom is paired with an informal explanation of its intended meaning. Recall that variables (in uppercase letters) are meant to be universally quantified, since each axiom has the shape of a rule.

Ax1: `holdAt(F,0)` $\leftarrow$ `initially(F)`.

A fluent holds at time 0, provided it holds initially.

Ax2: `holdAt(F,T2)` $\leftarrow$ `T1<T2` $\wedge$ `holdAt(F,T1)` $\wedge$ $\neg$`clipped(T1,F,T2)`.

A fluent still holds at an instant in time, provided it started to hold previously, and the fluent has not been clipped in the meantime.[5]

Ax3: `holdAt(F,T2)` $\leftarrow$ `happen(E,T1)` $\wedge$ `initiate(E,F,T1)` $\wedge$ `T1<T2` $\wedge$ $\neg$`clipped(T1,F,T2)`.

A fluent still holds at an instant in time, provided an event previously occurred which initiated the fluent, and the fluent has not been clipped in the meantime.

Ax4: `holdAt(F2,T2)` $\leftarrow$ `happen(E,T1)` $\wedge$ `initiate(E,F1,T1)` $\wedge$ `T1<T2` $\wedge$ `T2=T1+D` $\wedge$ `trajectory(F1,T,F2,D)` $\wedge$ $\neg$`clipped(T,F1,T2)`.

A fluent '`F2`' holds at an instant in time, provided (i) an event previously occurred which initiated another fluent '`F1`'; (ii) the fluent '`F1`', as long as it has not been clipped in the meantime, has triggered fluent '`F2`'.

Ax5: `clipped(T1,F,T2)` $\leftarrow$ `happen(E,S)` $\wedge$ `T1<S<T2` $\wedge$ `(terminate(E,F,S)` $\vee$ `release(E,F,S))`.[6]

A fluent is clipped during a period of time, provided an event initiated the fluent but afterwards the same fluent was terminated or released (from the law of inertia), and this occurred during the considered period of time.

The first notion of change—instantaneous change—can be best illustrated with an example. Suppose someone plants a tree, or—in EC terminology—the event '`plant-a-tree`' initiates the fluent '`tree-be-on-the-ground`'. From this scenario, common sense sanctions the inference that, as long as nothing occurs which may interfere with the initial situation (e.g., someone uproots the tree), the tree will remain where it was planted, i.e., the fluent '`tree-be-on-the-ground`'

[5] This is a simplified version of axiom 2, as presented in [28].

[6] Strictly speaking, there is no disjunction in the language. So axiom 5 should be written as two axioms.

will still hold. This reasoning based on inertia is taken care of by axiom 3. Axiom 2 captures the same kind of reasoning, but without making reference to a fluent being initiated by an event.[7]

The second notion of change—continuous change—is related with changes taking place in concomitance with the continuous exercise of a force, which triggers a form of acceleration and 'trajectorial' change. If someone kicks a ball, the ball will start moving in a certain direction: a force is communicated and as long as it exercises its influence, the ball will trace a trajectory in space that is dependent upon the force being exercised. In the language of EC the situation can be phrased as follows. The event '`kick-the-ball`' initiates the fluent '`the-ball-move`'; this fluent, in turn, is connected with another fluent describing the position of the ball, e.g., the parametrized fluent '`distance(ball,N)`', with '`N`' the distance expressed in, say, meters. The value of '`N`' will increase continuously as time passes, and as long as nothing unexpected happens (e.g., the ball hits an obstacle). The reasoning based upon this conceptualization is taken care of by axiom 4. The remaining axiom 5 captures the ordinary understanding of phrases such as 'something that may interfere occurs' or 'something unexpected happens,' which is rendered in EC by the predicate '`clipped`'.

2.1.3 Scenarios EC-$\mathcal{S}$

As anticipated at the beginning of this chapter, the axioms only provide a general theory of change and causality, and they must be complemented with domain-dependent formulas. These are particular rule-like formulas which compose scenarios.

In the following, the definition of scenario is preceded by the definitions of state and of chain of a rule. The notion of chain of a rule is useful as to avoid the construction of cyclic programs such as $\{p \leftarrow p_1, p_1 \leftarrow p_2, p_2 \leftarrow p\}$ which may cause loops in the computations.

DEFINITION **2.** For a rule R, let $b[R]$ and $h[R]$ denote the set of formulas occurring in the body and head of R.

(a) The *chain* of a rule R is the smallest set of rules, such that: (i) if $h[R'] \subseteq b[R]$, then R' is in the the chain of R; (i) whenever a rule Q is in the chain of R, if $h[Q'] \subseteq b[Q]$, then Q' is in the chain of R.

(b) The *ordered chain* of R is a strict partial order $\prec$ such that $Q \prec Q'$ iff Q' is in the chain of Q, with Q, Q' in the chain of R.

DEFINITION **3.** *States* are conjunctions of formulas of the shape:
Atoms '`happen(e,t)`' and '`holdAt(f,t)`', or their negations;
Equalities between fluent or event terms;
Formulas in the language of the structure $\langle \mathbb{N}, <, +, \times, 0, 1 \rangle$.

DEFINITION **4.** *Scenario rules* are rules with the shape: `initially(f)`

[7] Although it is standard in EC to conceive of fluents as being initiated by events, there may be fluents which hold but have not been initiated by an event (e.g., parametrized fluents driven by a dynamics as given in axiom 4; see the second notion of change).

`happen(e,t)` $\leftarrow \sigma$`(t)`
`initiate(e,f,t)` $\leftarrow \sigma$`(t)`
`terminate(e,f,t)` $\leftarrow \sigma$`(t)`
`release(e,f,t)` $\leftarrow \sigma$`(t)`
`trajectory(f1,t,f2,d)` $\leftarrow \sigma$`(f1,f2,t,d)`,

where 'σ`(t)`' and 'σ`(f1,f2,t,d)`' are states with the constants '`f, f1, f2, t, d`' occurring in them. Furthermore, one restriction on the construction of scenario rules apply. The restriction uses the notion of ordered chain given by definition 2 and they are meant to avoid loops in the computations.

> If a scenario rule is of the shape '`happen(e,t)` $\leftarrow \sigma$`(t)`', then the event '`e`' cannot occur in $b[R]$, and in no formula in Q, where $R \prec Q$.

Whenever a piece of knowledge is being modeled, its EC-representation will consist of the axiom set EC-$\mathcal{A}$ and a set of scenario rules, or a *scenario*, which is a subset of EC-$\mathcal{S}$. Typically, a set of rules and facts is called a *program* P, so every scenario will have the form of a program.

2.1.4 Integrity Constraints EC-$\mathcal{IC}$

The addition of an integrity constraint language on top of EC-$\mathcal{A}$ + EC-$\mathcal{S}$ is motivated by the need to model deontic statements. For now, the only concern is with the syntactic presentation, but it suffices to say that the usage of integrity constraints to express deontic notions has been put forward by Kowalski [18].

Definition **5.** Given a rule $A \leftarrow L_1, \ldots, L_k$, *integrity constraint* formulas are:
$?A$ `succeed` $\leftarrow ?L_1, \ldots, L_k$ `succeed`;
$?A$ `succeed` $\leftarrow ?L_1, \ldots, L_k$ `fail`;
$?A$ `fail` $\leftarrow ?L_1, \ldots, L_k$ `succeed`;
$?A$ `fail` $\leftarrow ?L_1, \ldots, L_k$ `fail`.

2.2 Computing with EC

This section outlines the proof system of EC. The primary task of the proof system is as follows: Given the set of axioms EC-$\mathcal{A}$ and a scenario in the form of a program P, determine whether the literal L can be derived from P. If L is derivable from P using the axioms of EC, I shall write '$P \vdash_{\mathsf{EC}} L$'.

The derivation process exploits the method of *resolution* based on backward chaining. I shall first look at how resolution works for a restricted language. In the propositional case, resolution is *applicable* whenever, for a query $?L$ and a program P, there is a rule such that its consequent is identical with L. Now, suppose we are given the query $?A$ and the rule $A \leftarrow L_1, \ldots, L_k$ in P. Since A is identical with the consequent of the given rule, resolution—via backward chaining—yields the new query $?L_1, \ldots, L_k$. As long as there are new rules available in P to which resolution is applicable, new queries can be generated. When all applicable rules have been used up, resolution terminates;

which means: (a) we reached the *empty query*[8] or (b) we are left with a rule to which resolution cannot be applied.

To define the derivation process, we need some more machinery. Resolution can be thought of as the process of constructing rooted trees, in which each node is labeled with a query. Trees are constructed and labeled as follows:

1. The root is labeled with the initial query.
2. The successive nodes are labeled according to three expansion rules:
   i) Given a node labeled with the query $?A$, and an applicable rule such as $A \leftarrow L_1, \ldots, L_k$, a successor node is constructed and labeled with the query $?L_1, \ldots, L_k$. If $k > 1$ rules are applicable to $?A$, then the tree splits into k branches.
   ii) Given the query $?L_1, \ldots, L_k$, two sucessor nodes, on the same branch, are constructed and labeled with $?L$ and $?L_2, \ldots, L_k$.
   iii) Given the query $?\neg A$, a sucessor node is constructed and labeled with the query $?A$.
3. The leaves of the tree are labeled with '`succeed`', if the immediately preceding node is labeled with an empty query; if otherwise, they are labeled with '`fail`', provided the immediately preceding node is labeled with a query to which no expansion rule is applicable.

It is useful to single out paths of nodes over rooted trees. Maximal paths or branches—i.e., those going from the initial query up to one of the leaves—are of special interest, since they can be used to express success and failure of queries *qua* nodes. Preliminarily, a *maximal path is successful* iff it ends with '`success`' and contains an even number of negated formulas, or it ends with '`fail`' and contains an odd number of negated formulas. Likewise, a *maximal path is unsuccessful* iff it ends with '`success`' and contains an odd number of negated formulas, or it ends with '`fail`' and contains an even number of negated formulas. With this terminology in place, it is easy to define when an initial query succeeds or fails.

Definition **6.** The initial query $?A$ *fails* iff all maximal paths are unsuccessful; it *succeeds* iff some maximal paths are successful. Accordingly, A is *derivable* from P iff $?A$ succeeds w.r.t. P; A is not derivable iff $?A$ fails. In case of $\neg A$, the query $?\neg A$ succeeds iff $?A$ fails, and viceversa.

The fact that $?\neg A$ succeeds if $?A$ fails indicates that the negation used in EC is a form of negation as failure. I give an example of the resolution process to clarify. Consider the program $P = \{p \leftarrow \neg q_1, p \leftarrow r_1, q_1 \leftarrow q_2, r_1 \leftarrow r_2, r_2\}$. The tree generated by posing the query $?p$ is given in figure 2.1. One can see that the tree has two branches, both successful. The left-most one is successful because it ends with '`fail`' but contains an odd number of negated formulas. The right-most one is successful because it ends with '`succeed`' and contains an even number of negated formulas. Thus, the query $?\neg p$ will fail, as $?p$ succeeds.

Note that the above works for the propositional language with $\leftarrow, \neg$ and $\wedge$. EC is, in addition, a first-order language, and so the method of resolution

[8] For a program P, a query $?A$ is an empty query iff A is a fact of P.

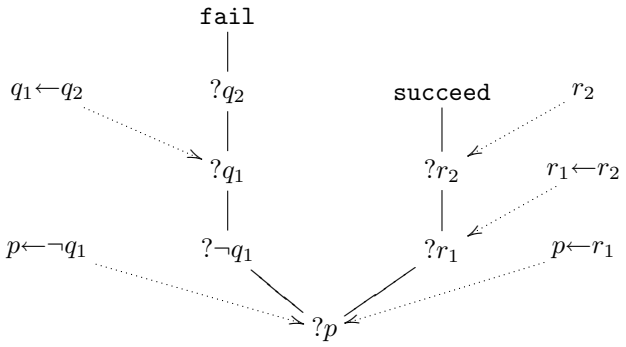

Figure 2.1: The resolution tree for the query $?p$ relative to the program $P = \{p \leftarrow \neg q_1, p \leftarrow r_1, q_1 \leftarrow q_2, r_1 \leftarrow r_2, r_2\}$. The dotted arrows connect a given query to the applicable rule or fact in P.

should be extended to handle first-order rules. Following the choice made in [28], I shall use *constraint logic programing* (hereafter, CLP).

2.2.1 Resolution in CLP

To explain CLP, a constraint language (hereafter EC-$\mathcal{C}$) will be extracted out of the language EC-$\mathcal{A}$ + EC-$\mathcal{S}$. Constraint formulas in EC-$\mathcal{C}$ are:

- formulas in the language of the structure $\langle \mathbb{N}, <, +, \times, 0, 1 \rangle$;
- formula securing the Unique Name Assumption;
- equalities or inequalities between terms;

and they will be denoted by C. I shall use over-lined symbols, e.g. '$\overline{x}$', to denote sequences of symbols, e.g., '$x_1, \ldots, x_k$'; the substitution of, say, the constant 'a' for the variable x will be denoted by '$[a/x]$'.

The main differences between resolution in the propositional case and in CLP are two. The first difference is due to CLP being a first-order language. In the propositional case, given a query $?L$, resolution is applicable if L is identical with some consequent L' of a rule in P; in CLP, we speak of L being unifiable with L':

Definition **7.** Two literals $L(\overline{a}, \overline{x}), L(\overline{y}, \overline{b})$ in EC-$\mathcal{L}$ (which are not in EC-$\mathcal{C}$) are *unifiable*, if there is a substitution, e.g., $[\overline{a}/\overline{y}]$ and $[\overline{b}/\overline{x}]$, which makes them identical.

Note that substitutions can almost equivalently be written as constraint formulas; for example, the substitution '$[a/x]$' can be written as the constraint '$a = x$'. This brings us to the second difference, which is related to the fact that CLP is a constraint language. For, in CLP, applications of resolution may result in the addition of a constraint formula 'C' in the new generated query, so that the iteration of resolution leads to an accumulation of constraints. In

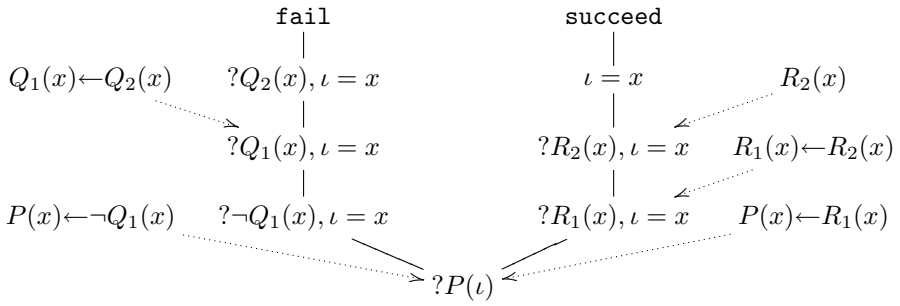

Figure 2.2: The resolution tree for the query $?P(\iota)$ relative to the program $P = \{P(x)\leftarrow\neg Q_1(x), P(x)\leftarrow R_1(x), Q_1(x)\leftarrow Q_2(x), R_1\leftarrow R_2(x), R_2(x)\}$. The dotted arrows connect a given query to the applicable rule or fact in P^*.

fact, resolution is dependent upon successful unification (corresponding to the addition of a constraint), as follows:

Definition **8.** Given the query $?A, C$, resolution is applicable iff there is a rule $A'\leftarrow L_1, \ldots, L_k, C'$, such that A and A' are unifiable under the constraint C''. As a result, the new query will have the shape $?L_1, \ldots, L_K, C, C', C''$.

Resolution in CLP will terminate when we have reached a query that only contains a set of satisfiable constraint formulas (in this case the successive node is labeled with '`succeed`') or a query to which resolution is not applicable (in this case the successive node is labeled with '`fail`'). The tree construction is the same as in the propositional case, and success, failure and derivability are defined as in definition 6.

Two Examples of Resolution in CLP

The *first* example (which uses rules written in a simple first-order language[9]) is only preparatory to the second (which uses formulas in the language of EC). Consider the program:

$$P^* = \{P(x)\leftarrow\neg Q_1(x), P(x)\leftarrow R_1(x), Q_1(x)\leftarrow Q_2(x), R_1\leftarrow R_2(x), R_2(x)\},$$

where variables are universally quantified. Posing the query $?P(\iota)$, for some individual ι, generates the tree in figure 2.2. The query $?P(\iota)$ succeeds, as the tree contains at least one successful branch, and so $?\neg P(\iota)$ fails.

For the *second* example in the language of EC, consider the program P^{**}:

```
initiate(plant(X,tree),be-on(tree,ground),T).
terminate(uproot(X,tree),be-on(tree,ground),T).
```

[9]There is not need to define this language precisely. It suffices to say it is a rule language with negation, where the literas composing rules are allowed to be first-order formulas.

happen(plant(ι_1,tree),1990).
happen(uproot(ι_2,tree),2006).

The program P^{**} describes the scenario in which someone planted a tree in 1990, and someone else uprooted it in 2006. The description is very simplistic (for example, it assumes that there is only one tree termed 'tree'), but it is good enough to show how resolution works in EC. In the previous two examples, the derivations were given in the form of trees (see figures 2.1 and 2.2). With formulas of EC, drawing trees becomes more difficult, and so I will use an alternative, more informal way to represent derivations. I shall illustrate it as I go on. Suppose we pose the query:

?holdAt(be-on(tree,ground),2005).

The expected result is that the query succeeds. There are four rules that are applicable to this query, namely Ax1-Ax4. This will correspond to the construction of a four-branched tree. However, the application of axiom 2 is what is needed. Axiom 2 can be applied, provided the constraints '?F=be-on(tree,ground)' and '?T2=2005' are given. So the new query is as follows:

?F=be-on(tree,ground)
?T2=2005
?happen(F,T1)
?initiate(E,F,T1)
?T1<T2
?¬Clipped(T1,F,T2)

The first two queries in the list contain the constraints, and the other queries contain the literals in the body of axiom 2. In order to understand the way I am representing resolution, two *conventions* must be born in mind:

(C1) One is that the constraints are always added first in the list of new queries, while the new queries (generated by the application of the rule) are added below the constraints.

(C2) The second convention is that if the same variable name occurs in two or more queries within the same list of queries, this means that the two variables are actually the same (they are implicitly unified). For example, '?T2' occurs in more than one query; in the second query above we have '?T2=2005'; this equality is implicitly transmitted to the variables '?T2' occurring in the other queries.

Note that the query '?happen(E,T1)' and '?initiate(E,F,T1)' can be eliminated, because the program P^* contains 'happen(plant(ι_1,tree).1990)', and 'initiate(plant(X,tree),be-on(tree,ground),T)'. To carry out the elimination, some constraints are to be added: 'E=plant(X,tree)', 'T=1990=T1', 'X=ι_1', 'F=be-on(tree,ground)'. So we have:

?F=be-on(tree,ground)

?T2=2005
?T1<T2
?¬Clipped(T1,F,T2)
?E=plant(X,tree)'
?T=1990=T1',
'?X=ι_1',

The new list of queries contains a set of satisfiable constraints. So, for the initial query to succeed, we only need to show that the query '?¬clipped(T1,F,T2)' succeeds. By negation as failure, this means that we need to show that '?clipped(T1,F,T2)' fails. Axiom 5 is the only applicable rule containing the 'clipped' predicate in the head. So we have:

?F=be-on(tree,ground)
?T2=2005
?T1<T2
?E=plant(X,tree)
?T=1990=T1,
?X=ι_1,
?happen(E1,T1)
?T1<S<T2
?terminate(E1,F,S) ∨ releases(E1,F,S)

Consider the disjunctive query in the list above. (Strictly speaking, there are no disjunctions in EC-$\mathcal{L}$, but these can be expressed via syntactic sugar.) In the program P^* there is no 'release' formula, but there is one 'terminate' formula. The last query above can be eliminated and replaced with the constraint 'E1=uproot(ι_2,tree)' and 'S=2006', we have:

?F=be-on(tree,ground)
?T2=2005
?T1<T2
?E=plant(X,tree)'
?T=1990=T1',
'?X=ι_1',
?T1<S<T2
?E1=uproot(ι_2,tree)
?S=2006

By rewriting the constraints, we have '1990<2006<2005', which is unsatisfiable. Now observe that the program P^* does not contain any other way to make the query '?terminate(E1,F,S)' succeed and, thus, that query must fail. By negation as failure, this yields that '?¬terminate(E1,F,S)' succeeds. This means that '?clipped(T1,F,T2)' fails, and so the initial query succeeds, as expected. This concludes the second example. Later on in the thesis, I will represent

derivations in the style of the second example, and adopt the two conventions (C1) and (C2) on page 13.

2.2.2 CLP with $\mathcal{IC}$

The language of EC includes integrity constraint formulas. Let I be an integrity constraint formula, evidently it is non-sense to pose a query such as $?I$. In fact, integrity formulas are taken to be true beforehand and the program P is forced to be updated so that the integrity constraints are satisfied. The notion of integrity satisfaction, as well as those satisfiability and non-satisfiability can be explained by means of the definition of failure and success of queries as given in definition 6.

I shall adopt the notational convention of denoting the consequent and antecedent part of I, deprived from the expression `succeed` or `fail`, by $c[I]$ and $a[I]$.

Definition **9.** Given a program P and an integrity constraint I, the integrity constraint can enjoy three status:

Satisfied: The outcome of the queries $?a[I]$ and $?c[I]$ is in accord with I; or the outcome of $a[I]$ is not in accord with I. In the latter case I is only vacuously satisfied.

Satisfiable: The formula I is not satisfied in P, but there is a way to update P into $P*$ such that I is satisfied in $P*$.

Non-satisfiable: The formula I is not satisfied in P, and from any $P*$ that satisfies I the contradiction is derivable.

Summary

This chapter introduces the syntax and the proof-system of EC. The presentation of the syntax in section 2.1 is split in four parts, corresponding to the different languages composing EC: the underlying language EC-$\mathcal{L}$, axioms EC-$\mathcal{A}$, language for scenarios EC-$\mathcal{S}$, and integrity constraint language EC-$\mathcal{IC}$. The proof system is briefly and informally outlined in section 2.2 by defining *resolution* (for the propositional case first, and then for the first-order case suitable for the language of EC). Three examples of derivations based on resolution are given (the first is based on a propositional rule language, the second on a simple first-order rule language, and the third on the language of EC). The chapter ends with definitions of the notions of satisfaction, satisfiability and unsatisfiability of integrity constraints.

3 Legislation and Event Calculus

This chapter illustrates how EC can be used to formalize legislation.[1] As introduced in the previous chapter, EC is a formal system that captures the ordinary, naïve understanding of change and causality. The examples given to illustrate the axioms of EC were about actions and changes occurring within (what may be called) the physical layer of reality—an egg being cracked, a tree being planted, or a ball being kicked. But many of the objects talked about in pieces of legislation—e.g., citizens, married couples, Presidents of the Republic—or legal changes—e.g., acquiring citizenship, becoming the proprietary of a house—are not included in the physical layer. So, the connection between EC and legislation does not seem to be straightforward.

However, the application of EC to the physical layer is one application among others—and this chapter should convince the reader that EC can be applied to different domains such as the legal one. A preliminary argument may point at the fact that changes occurring within (what may be called) the legal layer of reality follow more or less the same pattern as the changes occurring at the physical layer. If someone acquires citizenship, she then becomes a citizen, and she will continue to be so indefinitely, unless something occurs which may terminate or interfere with her citizenship. As the reader can see, this reasoning about a "legal change" is captured by axiom 3 of EC.

A concrete way to test more extensively the hypothesis that EC can be applied to the legal domain, is by taking a conceptualization of (portions of) the legal domain, and representing or modeling it with the language of EC. Pieces of legislation are such conceptualizations.[2] This means that the task of representing or modeling the legal domain becomes the task of formalizing a piece of (legal) text. The next chapter does this by taking as a case-study the Italian Citizenship Law (hereafter, ICL).

Engaging right away into the formalization of the text of ICL may be dispersive. It is best to first get acquainted, in a more organized way, with some recurrent expressions or features of legislative texts. A representative, though incomplete, list of such features is as follows:

[1] The application of EC to the legal domain is inspired by the use of Logic Programming to model the law: the affinities between Logic Programing and the law have been illustrated by Kowalski [18], while others have expressed qualms, e.g., Valente [27] and Leith [19]. Counter-evidence to their stance is given by the material presented in this and the following chapters.

[2] The legal domain has the advantage over other domains (like common-sense) that there exist explicit conceptualizations of it in the form of statutes, laws, decrees, etc.

1. static vs. dynamic view (or definitional vs. event-based);
2. time w.r.t. to issues such as validity and applicability of the law;
3. cross-referencing between laws, articles and clauses in the legislation; and
4. exceptions.[3]

In what follows, each feature will be examined more closely and modeled in the framework of EC. To this end, I shall consider these features *mainly* with respect to ICL. This will allow me to avoid unwarranted generalizations and at the same time gently introduce the topic of the next chapter: the formalization of ICL in EC.

3.1 Definitional and Event-based View

The ICL classifies the ways in which someone counts as an Italian citizen and the ways in which she does not. The law is almost entirely composed of articles in the form '*person 'a' is an Italian citizen, provided 'a' satisfies requirements 'b', 'c', etc.*' One could even contend that ICL provides nothing more than a classification of types of Italian citizenship. Although correct, this view is too coarse. I shall argue that two types of legal classification are to be distinguished—static and dynamic, or, *definitional* and *event-based*—and that EC, when properly extended, can handle both of them. Consider the following simplified version of articles 1 and 12 from the ICL:

(3.1) The child of an Italian national shall be an Italian citizen as well.

(3.2) The Italian national who serves in the army of a foreign country shall lose Italian citizenship.

The definitional renderings in first-order logic (hereafter, FOL) are:

(3.3) $\forall x, y : \text{child\_of}(x, y) \wedge \text{Italian\_citizen}(y) \rightarrow \text{Italian\_citizen}(x)$.

(3.4) $\forall x, y : \text{serves\_in}(x, y) \wedge \text{Italian\_citizen}(x) \wedge \text{foreign\_army}(y) \rightarrow \text{lose}(x, \text{Italian\_citizenship})$.

Problems occur with this formalization in the following scenario:

[3] A remarkable absentee in this list of features and expressions recurrent in the law is given by deontic notions (obligations, permissions, etc.). I cannot give a full treatment of deontic notions as of now, but simply point at two possible directions of research.

In Kowalski [18] and van Lambalgen & Hamm [28, p. 100], it is proposed that obligations (and deontic notions in general) be treated as integrity constraints on a database. On the one hand, a database registers what happens or has happened, and, on the other, a set of integrity constraints forces the database to be updated in a certain way, or it restricts the ways in which the database can be updated. This distinction parallels the one between a language for scenarios, EC-$\mathcal{S}$, and an integrity constraint language, EC-$\mathcal{IC}$ (see chapter 2). There is an alternative approach to deontics, based on deontic fluents. It has been worked out in the case of Situation Calculus (see Reiter [22] for an introduction) by Demolombe & Parra [9]. The same solution *mutatis mutandis* could be used for EC.

The abundant literature on the Chisholm's paradox could be used to test these proposed approaches, e.g., [5, 21].

(a) an individual 'ι' is a child of an Italian citizen, and
(b) 'ι' serves in a foreign army at a given time, say, '`T=1999`'.

The definitional FOL-formalization yields that, since 'ι' is a child of an Italian national, 'ι' is an Italian citizen by (3.3); and also that, since 'ι' serves in a foreign army, then 'ι' loses Italian citizenship by (3.4). Evidently, it is absurd that someone has Italian citizenship, while losing it. So the definitional formalization must be discarded as it sanctions an implausible conclusion.

On the other hand, a formalization of (3.1) and (3.2) that is restricted to an event-based approach is not adequate either. Suppose that (3.1) is interpreted as defining a condition that *initiates* Italian citizenship, and (3.2) as defining a condition that *terminates* it; then the formalization will be as follows:

(3.5) `initiate(start_Italian_citizen(X),italian_citizen(X),T).`
`happen(start_Italian_citizen(X),T)` $\leftarrow$
`holdAt(child_of(X,Y),T)` $\wedge$ `holdAt(Italian_citizen(Y),T).`

(3.6) `terminate(end_Italian_citizen(X),italian_citizen(X),T).`
`happen(end_Italian_citizen(X),T)` $\leftarrow$
`happen(serve_in(X,Y),T)` $\wedge$ `holdAt(foreign_army(Y),T).`

In the problematic scenario, composed of (a) and (b), the new formalization sanctions the right inferences.[4] However, an adequate formalization should not only capture the right inferences, but also resemble the wording of the law, especially if different wordings mark conceptual distinctions. In fact, while (3.2) appears to be adequately represented by (3.6), the formula (3.5) in the new event-based formalization is not a good rendering of (3.1). The sentence in (3.1) does not contain reference to any dynamics of initiating events; rather, it is a static representation of what a citizen is, for which the definitional approach seems to be more suitable. Note that the different wording of (3.1) and (3.2)—the latter contains reference to an event, and the former is *event-less*—is not limited to style or syntax: the different wording reflects a conceptual difference. To appreciate the importance of this distinction, the reader will have to wait until chapter 5, where a judgment based on the ICL at the Tribunal of Turin is analyzed. The text of the tribunal's sentence pointed exactly at the difference (in my terminology) between the definitional and event-based view of article 1 of the ICL.[5] To anticipate, it is worth quoting an excerpt:

[4] For if one poses the query '`?holdAt(Italian_cit(`ι`),2000)`', the query will fail. By fact (a), sentence (3.5) and axiom 3 the query *may* succeed, but fact (b) and sentence (3.6) make the '`clipped`' formula in axiom 3 succeed. This blocks axiom 3 from generating inferences, and hence the initial query fails, as excepted. (Incidentally, given '`T=1999`' the initial query will still succeed because fluents are actually terminated right after the terminating event has taken place. This is due to how the axioms of EC are set up. See [28, p. 63] for a discussion of this point.)

[5] The reader should be alerted that, in chapter 5, the assumption that there is only one correct interpretation of the law, and hence only one correct formalization, is strongly doubted. However, even if one interpretation (formalization) does not exclude the other (e.g., the definitional formalization does not esclude the event-based one), the point made in this section remains valid: there should a formal way to express definitional and event-based views of the law.

> On this matter, it should be made precise that ... the right to acquire Italian citizenship comes into being not at the event of the "birth" ..., but rather it is based on the situation of filiation from a parent who is a citizen.[6]

The moral is that we need to combine en event-based rendering of (3.2)—which we have already in (3.6)—and a definitional rendering of (3.1)—which we do not have yet. Obviously, the overall formalization should not yield unacceptable conclusions in the case of scenarios like (a)–(b). A straightforward, yet unsatisfactory, manner to formalize (3.1) in a definitional way is as follows:

(3.7) The child of an Italian national shall be an Italian citizen as well.
`holdAt(Italian_citizen(X),T)` ←
`holdAt(child_of(X,Y),T)` ∧ `holdAt(Italian_citizen(Y),T)`.

In the new formalization, the usage of initiating events is replaced by the representation of a static relation between being the child of an Italian national and being of Italian nationality. The formalization (3.7) is now textually precise, and one problem has been solved. (Note that there is no essential difference between the formalization (3.3) in FOL and the new formalization (3.7) in EC. The only, non-substantial, difference lies in the predicate '`holdAt`', because of technical reasons due to the EC-framework.)

Unfortunately, if one considers the new formalization overall —composed of (3.6) and (3.7)—an unacceptable conclusion, due to scenario (a)–(b), can be derived. Suppose both facts (a) and (b) are the case. The query
`?holdAt(Italian_cit(`ι`),2000)`
fails, by axiom 3, fact (b) and (3.6). However, the failure is overridden by the fact that the same query succeeds by (3.7) and fact (a). This is clearly counterintuitive. The solution I shall propose is to extend EC with the static predicate '`obtain`' and an additional axiom similar to axiom 2:

Ax6: `holdAt(F,T2)` ← `T1<T2` ∧ `obtain(F,T1)` ∧ ¬`clipped(T1,F,T2)`.

Axiom 6 says that if a fluent '`F`' *obtains* at an instant in time, then the same fluent will *hold* later on, unless something interferes with it in the meantime. The difference between '`obtain`' and '`holdAt`' is that the former is not subject to the principle of inertia and the latter is. Both '`obtain`' and '`holdAt`' apply to fluents, and so they are both *event-less*, static predicates. The difference between '`obtain`' and '`happen`' is that the predicate '`obtain`' is used for fluents and '`happen`' for events, but none is subject to the principle of inertia. The advantage of introducing the new predicate '`obtain`' can be immediately appreciated in this new, and satisfactory, formalization of (3.1):

(3.8) The child of an Italian national shall be an Italian citizen as well.
`obtain(Italian_citizen(X),T)` ←
`holdAt(child_of(X,Y),T)` ∧ `holdAt(Italian_citizen(Y),T)`.

[6] Deve essere precisato, in proposito, che il titolo di siffatto modo di acquisto ... della cittadinanza italiana è costituito, non già dall'evento "nascita" ..., bensì dalla situazione di filiazione da genitore cittadino. See [10].

With the new formalization, composed of (3.6) and (3.8), every problem seems to be sorted out: (3.6) has an event-based form and (3.8) has a definitional form, in accordance with their textual wordings. Furthermore, in case both (a) and (b) occurs, the intended conclusion can be derived. The query

`?holdAt(Italian_citizen(`ι`),2000)`

fails, because '`?clipped(1998,Italian_citizen(`ι`),2000)`' succeeds by fact (b), (3.6) and axiom 5. The '`clipped`' formula blocks any possible inference towards '`holdAt(Italian_citizen(`ι`),2000)`'.

The nature of the difficulties I have been confronted with while modeling (3.1) and (3.2) is twofold: on the one hand, it is essential to get the right inferences, and on the other it is preferable to model legislative articles in a way that is as close as possible to their natural language wording. Two requirements for good modeling can thus be stated:

(R1) An adequate modeling sanctions only the right or intended inferences.

(R2) An adequate modeling is textually precise.

The EC, properly extended, turns out to be a well-equipped formalism to cope with the interplay between a definitional and an event-based view of the legislation, yet two general questions are still open. One is whether EC is expressive enough to model *any* kind of definitional article. The other is whether EC is expressive enough to model *any* kind of event-like article. These are empirical questions, and a proper response would require one to model the entire available legislation.[7] More modestly, from the formalization of ICL given in the next chapter, it will become apparent that EC *is* expressive enough.

3.2 Time and Validity

A difficulty with the EC-based formalization of the law is that rules are assumed to hold unconditionally over time, without any temporal limitation.[8] In legal reasoning, instead, certain articles (translated as rules) may have limited applicability or validity. For instance, a law about social benefits can be valid or active from January 1, 2007, be applicable to those who satisfied certain requirements only between 1975–1980, and produce effects not before 2020 (e.g., the benefit can only be claimed after 2020). The example points out that there are at least three periods of time to which rules should be relativized: period of validity or activity, period of applicability, and period of effectiveness.[9]

[7] Clearly, EC cannot express sentences like '*if X satisfies Y, then X is* Z_1 *or X is* Z_2', because disjunctions are not allowed in the consequent of rules. This can be remedied by using extended logic programming. The problematic rule becomes, for example, '*If X satisfies Y, and X does* not *satisfy* Z_1*, then X satisfies* Z_2', where '*not*' is classical negation.

[8] This observation is taken from Marín & Sartor [20]. The authors provide an alternative solution to the one advocated here.

[9] In fact, the classification of three temporal periods is not complete. For example, it was neglected the period of time during which the law is part, as a piece of text, of the legislative

The period of validity or activity indicates when the law *qua* abstract object is part of the legal system and has normative force. The event of enactment[10] and repeal mark, respectively, the beginning and end of such a period. Yet the date of enactment does not say anything about the time period in which the rule is applicable or can produce effects. These are specified in the antecedent and consequent of rules (formalizing legislation):

$$A(t_1, t_2) \rightarrow C(t_3, t_4).$$

The temporal parameters t_1, t_2 in the antecedent determine the period of applicability of the rule. The period of effectiveness, instead, is specified in the consequent of the rule and demarcates the temporal beginning and end of the effects triggered by the rule, t_3, t_4. In the example above, the period of applicability was 1975-1980, and the period of effectiveness was 2020-onwards.

While validity, applicability and effectiveness should be kept apart, as they are *intensionally* different, it is often the case that they are *extensionally* the same, i.e., they cover the same set of points in time. For if an article does not contain explicitly its period of applicability and effectiveness, it will be assumed that these coincide (extensionally) with the period of validity, which should always be specified in legislative texts. But there are cases, as the previous example, in which the three periods are also extensionally different.

The question arises of how theses different temporal parameters are accommodated within the EC-framework. In one solution, they are attached to rules as meta-data, i.e., they are not expressed in the object-language itself, but they are specified at the meta-level. A more straightforward solution—the one I shall adopt—does not make any distinction between language and meta-language, and incorporates the temporal parameters within the rules themselves. I shall give a sketch of how this might work.

3.2.1 Applicability and Effectiveness

Periods of applicability and effectiveness can be easily embedded as temporal parameters within rules. Suppose an article applies between 1995-1999 and it is effective between 2005-2010. The following schema of formalization should take care of the *applicability* period 1995-1999, where 'σ`(T1,...,Tk)`' stands for the (proper) antecedent of the rule with the related temporal parameters:

`holdAt(cit(X),T)` $\leftarrow$
σ`(T1,..,Tk)` $\wedge$ `1995<T1,...,Tk<1999` .

The period of *effectiveness* 2005-2010 should be specified in the consequent, but consequents can only be atoms, not conjunctions. Fortunately, this syntactic

corpus. Publication and repeal (or abrogation) mark the beginning and end of such a period. The date of publication indicates when the law is added to the legislative corpus; date of repeal (or abrogation) indicates when the law is removed. Equivocally enough, the event of repeal marks also the end of the period of validity.

[10] There are quite a number of synonymous expressions denoting the event of enactment, e.g., coming into force, coming into effect. In the rest of the thesis, I shall use them more or less interchangeably.

restriction does no harm, as one can use an equivalent formalization by using three formulas instead of one, as follows:

```
2005<T1<2010.

T1=T2.

holdAt(cit(X,art1),T2) ← ...
```

A solution which distinguishes predicates of the object-language, e.g., '`cit`', from meta-level temporal attributes would be more elegant, while the solution here outlined confuses the two levels. Nonetheless, the solution seems to be acceptable enough in terms of the inferences that it sanctions, and in addition legislative texts themselves confuse object- and meta-level predicates or attributes.[11]

3.2.2 Validity or Activity

Defining the period of validity or activity of a rule is not straightforward. While in the case of applicability or effectiveness it was sufficient to add restrictions on the temporal variables in the head of a rule (for effectiveness), or in the body (for applicability); in the case of validity, temporal restrictions are meant to apply to the entire rule as a whole. A meta-level representation seems now unavoidable. Still, there is a way to define the period of validity in the object language.

Any legal qualification or predicate (e.g., being a citizen) will be indexed to an article name (typically a number). The article name uniquely identifies the article in the legal text which defines the meaning of the legal predicate, or which defines the conditions for the legal predicate to hold. So I shall write '`cit(X,ART)`' instead of simply '`cit(X)`'. The idea is to qualify the article number in the intended way, as to specify when the article (associated to a number) is valid. The events of repeal and enactment determines the extremities of the period of validity, and they can be specified as follows:

```
happen(repeal(ART),T);

happen(enact(ART),T).
```

By default, it is assumed that an article is made active or valid by the event of its enactment:

(act) `initiate(enact(ART),active(ART),T)`,

and it is made inactive or invalid by the event of its repeal:

(rep) `terminate(repeal(ART),active(ART),T)`.

[11] For example, here is how article 18, from ICL, mentions the period of applicability (underline added): *The persons ... who emigrated abroad before July 16, 1920 ... are deemed aliens of Italian origin.*

Clearly, as long as an article is not being repealed, it remains active. This form of reasoning based on inertia is taken care of by axiom 3.

The solution here proposed can define the validity of an article (translated as a rule or set of rules) within the object-language. Suppose we are given a rule saying that someone, who is found abandoned in Italy, acquires Italian citizenship:

`happen(acquire_cit(X,italy,art1),T)` ← `happen(found_abandoned(X,italy),T)`

Note that, *given any moment in time*, this rule grants acquisition of citizenship to anyone who is found abounded in the territory of Italy. However, such a "timeless" validity of the rule can be restricted by using (act) and (rep). For suppose we have derived the conclusion ($\star$) '`happen(acquire_cit(`ι`,italy,art1),1999)`' from the fact '`happen(found_abandoned(`ι`,italy),1999)`', by using the rule above. By using (act) and (rep), and some facts about when '`art1`' was enacted and repeled (if at all), one is able to check whether or not the conclusion ($\star$) is legally valid, i.e., whether it is based on a valid or active article. One only needs to pose the query '`?holdAt(active(art),1999)`'. In conclusion, rules as such have timeless validity, but the inferences that are generated using them can be distinguished in valid and invalid, depending upon the validity of the article on which the inference is based.

Validity vs. Classificatory Reasoning

By indexing '`cit`' to an article '`ART`', if the query

`?holdAt(cit(`ι`,art1),1999).`

succeeds, this does not tell us anything about 'ι' being en *actual* citizen at time 1999. Being an actual citizen for 'ι' would correspond to the success of the following query, where the article number has been dropped:

`?holdAt(cit(`ι`),1999).`

The above query will succeed provided there is an article granting Italian nationality to 'ι', and this article is valid at time 1999. To connect citizenship granted by an article with citizenship granted by a *valid* article (effective citizenship), this rule can be used:

(val) `holdAt(LEGAL_FLUENT(X),T)` ←

`holdAt(LEGAL_FLUENT(X,ART),T)`∧`holdAt(active(ART),T).`

The new rule[12] yields the conclusion '`holdAt(cit(`ι`),1999)`', provided 'ι' is an Italian national according to some *valid* article at time `1999`.

[12]Rule (val) does not comply with the syntax for scenario given at page 8, definition 4, and it is used here as an exceptional case. Except for the axioms, rules with '`holdAt`' predicates in the head should be used with extreme care. The problem with such rules is that, once their antecedent is satisfied, the '`holdAt`' will hold indefinitely because of axiom 2. In case of (val) this is no problem, but it may be troublesome in other cases.

The moral is that there are two kinds of reasoning: *classificatory reasoning* (definitional plus event-based; see section 3.1) and *validity reasoning*. The former consists in the attribution of a legal qualification or property to an individual at a given instant in time and according to a certain article of the law. The latter builds on the result of the classificatory reasoning and requires in addition that the articles used in the classificatory reasoning be valid.

The separation between classificatory and validity reasoning is a useful one, because, among other things, it allows one to perform simulations, namely drawing conclusions about what legal qualifications, rights, duties, etc., would hold, regardless of the validity of the law. Unfortunately, such a distinction yields a major complication. Suppose it has been proven that 'ι' is an Italian national, according to article 1 at time '`T=1999`', and that the proof involves articles 2 and 3. This means that the proof involved previous legal predicates which were defined by article 2 and 3. For example, the proof may have used the rule:

`happen(acquire_cit(X,italy,art1),T)` $\leftarrow$

`happen(approval(president,art2),T)`$\wedge$

`happen(approval(state_council,art3),T).`

In order to know whether or not 'ι' counts as an effective Italian citizen, one should verify the validity of articles 1, 2 and 3, and not only of article 1. For no valid conclusion can follow from premisses which used invalid articles. The difficulty is that, when 'ι' is proven to be Italian, according to article 1, there is no trace (encoded in the language) that other articles have been used during the proof. Yet, these other articles need to be checked for their validity.

One way out is to design a formal device that can dynamically keep track of articles invoked during proofs, so that, at the end of a proof, one can check the validity of the articles collected during the proof. Another approach to the problem is to require that, during the formalization of the law, whenever legally relevant fluents or events are written in the antecedent of rules, they should be paired with fluents guaranteeing their activity or validity. So the above rule should have been written:

`happen(acquire_cit(X,italy,art1),T)` $\leftarrow$

`happen(approval(president,art2),T) [`$\wedge$`holdAt(active(art2),T)]` $\wedge$

`happen(approval(state_council,art3),T) [`$\wedge$`holdAt(active(art3),T)].`

The two legal fluents related with '`art2,art3`' are paired with fluents between square brackets guaranteeing the validity of '`art2,art3`'. This solution solves the problems related with the validity reasoning, but destroys the possibility to perform independent classificatory reasoning. A good solution would then allow one to select which formulas to call during proofs. So, while performing validity reasoning the formulas between square brackets are used, but they are ignored while performing classificatory reasoning. How this can be achieved is an unsolved problem in this thesis[13]

[13] A concrete example of how this problem can manifest itself is given at the end of chapter 5.

3.3 Cross-referencing

A distinguishing feature of legislative texts is the use of cross-referencing between laws, articles, clauses, and phrases. Tentatively, *cross-referencing* is a textual phenomenon such that sections, articles, etc. refer to other sections, articles, etc., or to themselves. From the standpoint of formal linguistics, cross-referencing can also be seen as a variant of anaphoric (and cataphoric) reference.[14] Typically, anaphoric expressions such as pronouns refer to individual entities that have been talked about previously in the discorse. Assuming that legislation is an ongoing discourse, cross-referencing is the act of retrieving articles that have been talked about elsewhere in other portions of the legislative discourse (or legislative corpus). However, while anaphoric reference points at well-defined individual entities, it is not equally clear what cross-referenced entities are like in legislative texts. Certainly, they are articles—but what does that mean exactly?

Roughly speaking, cross-referencing can be split into two components. First, there is the proper reference, call it ref_n, to another article, section, etc. by use of a number. Second, ref_n is "framed" within a certain linguistic construction: 'ref_n *shall apply*', '*according to the definition of . . . contained in* ref_n', '*provided the provisions given by* ref_n *have been satisfied*', etc. Formalizing the linguistic frame of ref_n should not be more difficult than modeling any other natural language expression. The ultimate difficulty seems to reside in what the exact reference of each ref_n consists of, and how the referred entity interacts with the linguistic frame.

Unfortunately, there is no available categorization of the main linguistic constructions adopted in the multifaceted phenomenon of cross-referencing. This absence makes it impossible to show, with some degree of generality, how cross-referencing can be handled by EC. For the present purpose, it should suffice to present and model only one illustrative example.

Recall that legally relevant predicates are associated with the article number in which they are defined, e.g., '`cit(X,ART)`'. This device is simple and powerful enough to deal with some elementary cross-referencing phenomena. Suppose we are given the following articles (a simplified version of article 3 of the ICL, clause 1 and clause 2):

(A) Those who are adopted by an Italian citizen, shall acquire Italian citizenship.

(B) Article A applies also to those who were adopted before the date of enactment of this law.

Under one posible interpretation, article B secures that the period of applicability of article A can precede enactment, although by default it should initiate at the same moment as enactment. Note, however, that the period of effectiveness does not seem to be affected by what is stated in article B. An adequate

[14] I am grateful to Michiel van Lambalgen for suggesting this parallelism.

translation of (A) and (B) should capture this interplay between applicability and effectiveness:

(A*) `initiate(acquire_italian_cit(X,artA),italian_cit(X,artA),T).`
`happen(acquire_italian_cit(X,artA),T)` ←
`happen(adopt(Y,X),T)` ∧ `holdAt(italian_cit(Y),T).`

(B*) `happen(acquire_italian_cit(X,artA),T2)` ← `happen(enact(artA,ART),T1)`
∧ `T<T1` ∧ `happen(apply(X,artA),T)`
∧ `T1<T2` ∨ `T2=T1` [∧ `happen(adopt(Y,X),T)`].

Observe the constraints on the time variables '`T`', '`T1`' and '`T2`' in (B*). Variable '`T`' is bound to the applicability of (A), variable '`T1`' to the date of enactment, and variable '`T2`' to the period of applicability. So, rule (B*) says that the period of applicability of (A) extends itself retroactively before enactment—constraint '`T<T1`'—, but the effectiveness period is not modified by any retroactivity, and so it follows (or coincides with) the period of activity—constraint '`T1<T2` ∨ `T2=T1`'.

Rule (B*) contains the formula '`happen(apply(X,artA),T)`', which should be linked to article A as follows:

(B-A) `happen(apply(X,artA),T)` ←
`happen(adopt(Y,X),T)` ∧ `holdAt(italian_cit(Y),T).`

A few words of comment are in order. First, the part between square brackets in (B*) is superfluous, because it is already contained in the above rule. Second, '`apply(X,artA)`' is the key predicate used to handle the ref_A component of cross-referencing, while rules (B-A) and (B*) take care of the linguistic frame of ref_A. A more systematic and less *ad hoc* approach is called for, but as long as no comprehensive body of linguistic data is available (e.g., a systematic classification of cross-referencing), such an enterprise is unfeasible. Third, the proposed approach sanctions the right inferences. Consider the scenario in which the individual 'ι' is adopted by an Italian citizen in 1950, while the law is enacted in 1992. By applying (B*), (B-A), and finally (A*), the conclusion '`happen(italian_cit(`ι`,artA),1992)`' can be reached, as well as '`holdAt(italian_cit(`ι`,artA),1993)`' by axiom 3, although the conclusion '`happen(italian_cit(`ι`,artA),1955)`' does not hold, as expected.

I have only illustrated a simple example of cross-reference; other examples will be given in the next chapter when the text of ICL is formalized.

3.4 Exceptions

The main technical device I shall adopt to handle exceptions is negation as failure,[15] which has the following, exception-like, procedural meaning: the negated formulas are assumed not to hold, unless they are proven to obtain.

[15]The content of this section relies on and adapts ideas from Stenning & van Lambalgen [26] and Kowalski & Toni [17].

To begin with, I will outline a general and EC-independent framework for formalizing exceptions. Rules of the form '*if* φ*, then* ψ*, unless* χ', where the *unless*-clause marks an exception, can be easily rendered as follows:

$$\varphi \wedge \neg\chi \rightarrow \psi.$$

However, it is often the case that *unless*-clauses are not explicitly stated. An article may have the form '*if* φ*, then* ψ', and another article the form 'ψ *cannot obtain in case of* χ'. The latter article is an exception to the former, but no exception marker is used. This raises the question of how the rule '*if* φ*, then* ψ' should be rendered. To accomodate exceptions that are not explicitly introduced as exceptions in the text of the law, one may use exception markers, e.g, 'ex', in any formal rendering:

$$\varphi \wedge \neg ex \rightarrow \psi.$$

The formula 'ex' is assumed to fail, unless it is triggered by something else, such as:

$$\chi \wedge \neg ex' \rightarrow ex.$$

The drawback of this approach is that, by default, it introduces exception markers where the text of the legislation may not contain any. This is a problem with respect to requirement (R2) on page 20—that the modeling of the law should be textually precise, and reflect the way the law is worded. One could say that problems with (R2) arise only if we are seeking a literal formal translation of the law; but, if capturing the intended meaning of the law is all we are seeking, the formalization that uses default exception markers is adequate, albeit not literal.[16]

I will now look at exceptions in the framework of EC. The advantage of EC is that it can handle exceptions and be textually precise at the same time (also when explicit exception markers are not mentioned in the text of the law). Roughly speaking, heads of rules will (mainly) contain '`happen`' or '`obtain`' predicates.

If explicit exception markers are mentioned in the law, then one can use an exception-fluent, e.g., '`ex`', with at least two arguments, the article (number) where the exception is stated and the article (number) to which the exception applies. Rules with '`obtain`' predicates in the head will look like the following:

[16] One textually precise approach would be to use a priority ordering between rules. In this approach, rules '*if* φ*, then* ψ' and 'ψ *cannot obtain in case of* χ' are rendered by the rule-formulas '$\varphi \rightarrow \psi$' and '$\chi \rightarrow \neg\psi$'; subsequently, a priority ordering relation between the rule-formulas is added, saying for example that the second formula overrides the first, whenever conflicting outcomes arise. The main difficulty with this solution is that it adds a further complication to the formalism, by reasoning *about* the rules, and thus introducing a form of meta-level reasoning that I prefer to avoid. Besides, one may question whether the solution based on ordering of formulas is cognitively adequate. What do we do when we compute exceptions? Do we actually compute all the possible conclusions first, and then decide which conclusion actually follows, on the basis of the ordering? Rather, is it not that we immediately grasp one rule as an exception to the other, and so we compute only one inference and ignore the others?

```
obtain(legal_qualification(X,ART1),T) ←
...
not(holdAt(ex(X,ART1,ART2),T).
```

The '`ex`' fluent contains reference to two articles in the legislation: '`ART2`' is the article where the exception is defined, and '`ART1`' is the article affected by the exception. It may be the case that the two articles are the same one, or that the articles containing the exception are unknown; and in the latter case the fluent '`ex`' cannot hold. Since '`ex`' is a fluent, it may (but need not) be initiated by an event:

```
initiate(start_ex(X,ART1,AR2),ex(X,ART1,ART2),T).
```

The treatment of exceptions in the case of '`happen`' predicates in the head of a rule can be accomplished along the same lines as above.

When no exception markers are mentioned in the law, the axioms of EC can be used to block inferences, by adequately using the predicate '`clipped`'. Let us look at rules with the '`happen`' predicate in the head. For example, consider the sentence '*those who are born in Italian territory acquire Italian citizenship on the date of birth*', and its formal rendering:

(3.9) `initiate(acquire_cit(X,italy),cit(X,italy),T).`

`happen(acquire_cit(X,italy),T) ← happen(born_in(X,italy),T).`

Consider now the sentence '*those who possess another nationality shall not posses Italian nationality*', which can be seen *per se* as a prohibition, but also as an exception to the status of citizenship by birth in the territory of Italy. Using the predicate '`clipped`' allows one to formalize the exception:

(3.10) `clipped(T1,cit(X,italy),T2) ← hold(cit(X,Z),T) ∧`
`neg(Z=italy) ∧ T1<T<T2`[17].

Consider, now, the scenario in which the individual 'ι' is born in the territory of Italy in 1995, but also is a citizen of Germany. We pose the query:

`? holdAt(cit(`ι`,Italy),2000).`

By axiom 3 and (3.9), the query may succeed, but the exception-sentence (3.10) makes it fail; in the different scenario in which 'ι' was born in the territory of Italy without being German, the query will succeed by axiom 3 and (3.9), as expected.

The advantage of this alternative treatment of exceptions for '`happen`' predicates is that exception markers need not be written by default in the antecedent of rules, but they can be added subsequently by means of '`clipped`' clauses, such as (3.10). An analogous treatment of exceptions can be given in the case of the '`obtain`' predicate: the appropriate usage of axiom 5 and 6, together with the '`clipped`' predicate, should do. For this solution to work, it is essential to

[17] This rules does not respect the syntax of scenario formulas. This is not a problem, unless it causes unforeseen complications.

formalize articles as rules which contain either '`obtain`' or '`happen`' predicates in their heads. As shown in section 3.1 before, where (3.2) was an exception to (3.1), using the predicate '`holdAt`' destroys the possibility of handling exceptions.

Summary

This chapter explores to what extent recurrent features and expressions of legislative texts can be formalized in the language of EC. The features are: definitional vs. event-based view, time and validity, cross-referencing, exceptions.

Throughout the chapter, sparsely, some general observations are made. It is worth summing them up. In section 3.1, it is suggested the EC be extended with the predicate '`obtain`' and the axiom:

Ax6: `holdAt(F,T2)` $\leftarrow$ `T1<T2` $\wedge$ `obtain(F,T1)` $\wedge$ $\neg$`clipped(T1,F,T2).`

In the same section, two general maxims for good modeling are formulated:

(R1) An adequate modeling sanctions only the right or intended inferences.

(R2) An adequate modeling is textually precise.

In section 3.2, a distinction between classificatory and validity reasoning is made, and a rule to combine the two is given:

(val) `holdAt(LEGAL_FLUENT(X),T)`
$\leftarrow$`holdAt(LEGAL_FLUENT(X,ART),T)`$\wedge$`holdAt(active(ART),T).`

The distinction between classificatory and validity reasoning raises an open problem, namely: once the classificatory reasoning yields a certain (legal) conclusion (e.g., 'ι' is a citizen according to article 5), such a conclusion needs to be checked with respect to its legal validity (i.e., whether the drawn conclusion is based on valid articles). To perform the checking, the articles needed during the classificatory reasoning which yielded the conclusion, are to be collected. So the problem is to design a formal devise that can dynamically keep track of the articles used during the classificatory reasoning.

4 Legislation Formalized

This chapter presents a sample of the formalization of the Italian Citizenship Law.[1] For each of the legislative features talked about in the previous chapter (definitions and events, time and validity, cross-referencing, exceptions) concerte examples taken from the Italian Citizenship Law (hereafter, ICL) will be given and formalized. To begin with, some preliminaries about notation and methodology are in order.

4.1 Preliminaries

4.1.1 Notation

To facilitate the writing up of the formalization, I shall use a slightly different notation than the one in chapter 2. Formulas are written in a Prolog-style syntax. Constants (for predicates and individuals) begin with lower case letters, and variables with upper case letters. Mnemonic notation will help understand the intended use of variables, e.g., '`T`' is used for instants in time, '`ART`' for articles in the legislation, '`X`' or the like for individuals, '`N`' or the like for numbers. Sentential connectives are as follows: '`,`' for conjunction, '`;`' for disjunction, '`not`' for negation as failure, and '`:-`' for reversed implication. No specific symbol is used for quantifiers, even though rules are always universally quantified. Rules have the following shape:

```
consequent:-
          antecedent_1,
          antecedent_2,
          antecedent_3,
          ...
          antecedent_k.
```

No distinction between fluents and events is made apparent at the level of the syntax, but suggestive names are adopted, e.g., '`acquire_citizenship`' for an event and '`citizen`' for a fluent. The axioms of EC should be rewritten accordingly.

[1] The formalization is based on an English translation of the Italian text [2]. The unofficial English translation I followed can be found at http://www.legislationline.org/. However, I have amended it, whenever I considered it to be necessary.

4.1.2 Methodology

The methodological choices made for the formalization were essentially twofold.

Bottom-up vs. Top-down

First, the modeling of the law followed a bottom-up approach, supplemented with a trial-and-error procedure (following Sergot et al.[23]). A bottom-up methodology means that, first, each single article *qua* isolated piece of text is formalized, and subsequently the different bits of the formalization are assembled to form the global model of the law. Trial-and-error procedures are necessary whenever the isolated formalizations need to be adjusted, more or less drastically, to fit in the global model.[2] An alternative, top-down, approach—which could avoid most of the trial-and-error procedures—consists of constructing a preliminary representation of the entire legislation, and subsequently carrying out the formalization of single articles, in view of the preliminary representation.[3] This approach is certainly better in that it avoids the trial-and-error. However, it brings along the different complication of designing a method for constructing a preliminary model of the law.

One more reason to prefer a bottom-up methodology comes from a further requirement for a good formalization:[4]

(R3) A good formalization should be *maintainable*: as the texts of the law evolves, so does the formalization.

If the formalization is bound to a preliminary model of the *entire* law, whenever the law changes incrementally by means of the addition of new articles,[5] it may very well be that the existing formalization of old articles needs to be revised as well, despite the fact that these were not modified. Suppose a legal system is composed of two normative statements:

(R_1) Walking on the grass is prohibited.

(R_2) Owners of dogs are allowed to walk on the grass.

The two statements as a whole, in a top-down approach, can be formalized in FOL as follows:

$(R_{12})\ \neg(own(x,y) \wedge dog(y)) \rightarrow prohibited(walk\_on\_grass(x))$

But suppose a new statement is added:

(R_3) Owners of dogs younger than 6 months are *not* allowed to walk on the grass.

[2] It would be best to avoid trial-and-errors, but this cannot always be the case. Indeed, if the formalization of one article needs to be revised once the global picture is in place, this makes the approach partly, albeit not entirely, top-down.

[3] The distinction between top-down and bottom-up methodology is also used by Bench-Capon et al. [3, p.193], although the authors use the same terminology in the reversed way.

[4] Other requirements were spelled out in section 3.1, on page 20.

[5] To keep things simple, I disregard the case in which the law changes decrementally, i.e., when articles are repealed.

The addition will require to throw away rule (R_{12}) and replace it with:

(R_{123})
$$\neg(own(x,y) \wedge dog(y)) \vee (own(x,y) \wedge dog(y) \wedge younger\_than(y, 6\_months)) \rightarrow prohibited(walk\_on\_grass(x))$$

The difficulty with a top-down approach is that the formalization evolves in a way that is not uniform to the changes made to the legal text. In the example above, the existing statements (R_1) and (R_2) were not removed after the addition of (R_3), whereas the old formalization (R_{12}) had to be replaced by (R_{123}). The lack of uniformity between the law and its formal representation has drawbacks with regard to maintainability: when new articles are added to the legislation, the old formalization may have to be revised or replaced entirely. This raises questions as to how the revision should be carried out, which portions of the formalization should be given up, how to single them out efficiently, etc.

More elegantly, the addition of new articles should simply correspond to the addition of new rules in the formalization, with no further complication. Above that, it is true that when (R_3) was added, (R_1) and (R_3) were not modified as pieces of texts (as syntactic objects), but their "meaning" shifted after the addition. A formal representation should be able to mirror the same kind of phenomenon: after the addition of new formulas (formalizing added articles), the old formulas (formalizing existing articles) continue to be the way they were, but their "meaning" changes in concomitance with the addition. I contend that a bottom-up approach, combined with the use of negation as failure, is the most suitable candidate that can meet these desiderata.[6]

Statements (R_1) and (R_2) can be formalized as isolated units, in a bottom-up way, as follows:

(R_1^*) $prohibited(walk\_on\_grass(x))$

(R_2^*) $own(x,y) \wedge dog(y) \rightarrow allowed(walk\_on\_grass(x))$

Unfortunately, the combination of (R_1^*) and (R_2^*), applied to owners of dogs, yields the unacceptable conclusion that owners are prohibited *and* allowed to walk on the grass. This problem was avoided in the top-down approach by formalizing (R_1) and (R_2) in one go. A solution to this problem, within the bottom-up approach, is offered by negation as failure:

(R_1^{**}) $\neg ex_1 \rightarrow prohibited(walk\_on\_grass(x))$

(R_2^{**}) $own(x,y) \wedge dog(y) \wedge \neg ex_2 \rightarrow allowed(walk\_on\_grass(x))$

The formulas '$\neg ex_1$' and '$\neg ex_2$' have been added in the antecedent of (R_1^*) and (R_2^*). In general, '$\neg ex_i$' reads '*there is no proven exception to* R_i', where '$\neg$' is negation as failure. The final step to solve the problem is to make explicit that (R_2) is an exception to (R_1):

[6] What follows is based on the work by Stenning and van Lambalgen on the suppression task [26].

(ex_{12}) $own(x, y) \wedge dog(y) \rightarrow ex_1$

The combination of (R_1^*), (R_2^*) and (ex_{12}), when applied to dog owners, does not yield the prior unacceptable conclusion, because (ex_{12}) blocks the conclusion '$prohibited(walk\_on\_grass(x))$'.

The addition of (R_3) is unproblematic and corresponds to the addition of one rule formalizing (R_3), along with a rule specifying that (R_3) is an exception to (R_2):

(R_3^{**}) $own(x, y) \wedge dog(y) \wedge younger\_than(y, 6\_months) \wedge \neg ex_3$
$\rightarrow prohibited(walk\_on\_grass(x))$

(ex_{23}) $own(x, y) \wedge dog(y) \wedge younger\_than(y, 6\_months) \rightarrow ex_2$

Both desiderata are met: the addition of new article corresponds to the simple addition of new formulas, and the "meaning" of the formulas changes as new formulas are added. The latter point can be made more perspicuous if 'meaning' reads 'sanctioned inferences'. Given a scenario in which the individual 'ι' owns a dog younger than 6 months, the set of formulas given by $(R_1^{**}), (R_2^{**})$, and (ex_{12}) yields the conclusion that '$allowed(walk\_on\_the\_grass(\iota))$' and '$\neg$ *prohibited(walk_on_the_grass(ι))*'. After adding (R_3) and (ex_{23}), the new set of formulas yields the different conclusion that '$\neg allowed(walk\_on\_the\_grass(\iota))$' and '$prohibited(walk\_on\_the\_grass(\iota))$'. As expected, the meaning of the formulas—seen in terms of derivable inferences—changes as new formulas are added.[7]

[7] A counter-counter argument to the bottom-up approach defended here, is that the size of the formalization is larger than in a top-down approach (in the example above: 5 formulas instead of only 1). This, in turn, may result in a blow-up of the complexity of the reasoning needed to compute inferences. If the increase in size becomes a threat when inferences are computed, a tentative solution could suggest that the bottom-up representation be translated into a top-down one that will be used to compute inferences. A translation from bottom-up to top-down representations is available, and it is known as *completion* (of a program) (see [28, p.55]). Without entering into the technicalities, completion consists of two main steps: first, rules with the same head, e.g., $\varphi_1 \rightarrow p, \varphi_1 \rightarrow p$, are merged into one rule whose body contains the disjunction of the bodies of the original rules, e.g., $\varphi_1 \vee \varphi_2 \rightarrow p$; second, occurrences of '$\rightarrow$' are replaced by '$\leftrightarrow$'.

If completion is applied to 'ex_{12}' and 'ex_{23}', this yields two doubles implications.

(ex_{12}^c) $own(x, y) \wedge dog(y) \leftrightarrow ex_1$

(ex_{23}^c) $own(x, y) \wedge dog(y) \wedge younger\_than(y, 6\_months) \leftrightarrow ex_2$

Moreover if an ex-formula does not occur in the head of any rule, as in the case of ex_3, we set by default:

$(ex_{3\perp}^c)$ $\perp \leftrightarrow ex_3$

The equivalences given by (ex_{12}^c), (ex_{23}^c) and $ex_{3\perp}^c$ can be applied to the rules $(R_1^{**}), (R_2^{**})$, and (R_3^{**}), so we obtain:

(R_{1c}^{**}) $\neg(own(x, y) \wedge dog(y)) \rightarrow prohibited(walk\_on\_grass(x))$

(R_{2c}^{**}) $own(x, y) \wedge dog(y) \wedge \neg(own(x, y) \wedge dog(y) \wedge younger\_than(y, 6\_months)) \rightarrow$
$allowed(walk\_on\_grass(x))$

(R_{3c}^{**}) $own(x, y) \wedge dog(y) \wedge younger\_than(y, 6\_months) \wedge \top$
$\rightarrow prohibited(walk\_on\_grass(x))$

From the new set of rules, the formalization (R_{123}) can be derived immediately.

EC-specific Methodology

In the second place, the adopted methodology was shaped by the language used for modeling the law, the EC. Unfortunately, no experimented methodology for modeling law, that is specific to EC, is available. One trend in the literature is to use fragments of FOL to formalize pieces of legislative text (e.g., the language of Logic Programming as in [23, 18, 3]), but EC has never been applied extensively to the domain of law.[8] The EC-formalization of ICL that is presented here, thus, aims at being no more than a rudimentary prototype, based on a not-yet-finally-shaped methodology. The tentative EC-specific bottom-up methodology may be casted in this semi-algorithmic way:

1. While formalizing, consider each article in isolation, but according to the order of occurrence in the legislative text;
2. Formalize each article as a set of one or more rules:
   a. Pick out the part of the text which indicates what the head(s) of the rule(s) should look like;
   b. Since heads of rules are mainly of two kinds, '`obtain`' or '`happen`' predicates, textual evidence may suggest which one to prefer (e.g., if an event is mentioned, such as '*was born*' or '*acquire*', then the '`happen`' predicate is preferable, but if a static representation like a definition is expressed, then one should opt for the '`obtain`' predicate);
   b. Write up the head(s) of the rules(s);
3. Finally, construct the global model by coherently assembling the rules:
   a. Collect and group consequents of rules according to desired criteria;
   b. Add formulas to make the formalization work with the axioms of EC.

Points 1–3 will be clarified throughout the rest of the chapter. Sections 4.2.1, 4.2.2, and 4.2.3 provide plenty of illustration for points 1 and 2. In these sections, several articles from ICL are formalized in isolation (or, at the most, the isolated formalizations are connected only when an explicit cross-referencing from one article to the other is given by the text of the law). The stage of assembling the isolated formalizations in a coherent way for EC (point 3 above) is explained in section 4.2.4.

4.2 The Formalization

What follows should be seen as the empirical counterpart to the material presented in the previous chapter, in which issues such as cross-referencing, time and validity, exception, etc. are dealt with in a rather abstract and general way. In this section, the very same topics are encountered and discussed in the "lively flesh" of the legislative text. As it will soon become apparent, deontic notions will receive little or no attention;[9] this choice was deliberate and intended to show that, despite the *vulgata* in deontic logic, formalizing the legal domain

[8] The application of EC to the law has been suggested in e.g. Bench-Capon et al. [3].

[9] Permissions are formalized in case of article 3 (on citizenship by adoption).

does not only involve formalizing talks about obligations and permissions—it involves much more.

The formalization of each article assumes a short-cut which needs to be explained. In the previous chapter, it was suggested that each legal predicate, e.g., '`acquire_cit`' or '`cit`', should contain an argument indicating the number of the article in which it is defined. I shall adopt the convention of writing the article number which corresponds to most specific place in the legislation (number of section, article, clause, etc.) in which the predicate is being defined. For example, it is redundant to write two rules such as:

```
holdAt(cit(...,icl.art1), T) :- ...
holdAt(cit(...,icl), T) :- ...
```

The two rules indicate that the fluent '`cit`' is being defined and occurs in ICL, article 1, and also in the entire ICL. But the latter could be inferred from the former. The dot between '`icl`' and '`art1`' can be seen as a *navigation operator* through the hierarchical structure of the legislation: if a predicate occurs in a certain article of the legislation, it also occurs in the entire legislation (since the entire legislation hierarchically contains its own articles). In this way I shall minimize the number of times the same predicate is paired with different numbers.

I will divide the presentation of the formalized articles into three parts:

Acquiring Citizenship: This part include articles 1, 3, 4, 5 and 9. They determine how Italian citizenship is acquired: by birth (art. 1), by adoption (art. 3), by residence (art. 4), by marriage (art. 5), by decree of the President of the Republic (art. 9).

Exception and Cross-referencing: This part presents the formalization of articles 6 and 10. They concern ways in which acquisition if citizenship may be precluded.

Application, Repeal, and Enactment: This part presents articles 18, 26 and 27. Article contains provisions for acquiring or recovering Italian citizenship which are related to the territorial situation of Italy immediately after the second World War, and it was selected to show how applicability restrictions occur in the law. Articles 26 and 27 deal with enactment of ICL and repeal of previous legislation.

4.2.1 Acquiring Citizenship

The range of representational issues raised by the first group of articles is rather wide, e.g., definitional and event-based approaches (see, in particular, citizenship by birth in article 1), cross-referencing (articles 3 and 9), usage of the predicate '`trajectory`' to handle deadlines (articles 3, 4, 5 and 9). In what follows, for each article, I will quote the text of the article or portions thereof, then give the related formalization, and finally comment on it. The writing between brackets accompanying the number of the article is *not* part of the text of the law (it was added to enhance clarity).

Article 1 (birth):

1. *Citizen by birth is:*
   a) *the child of a father or a mother, who are Italian citizens;*
   b) *a person who was born in the territory of the Republic if both parents are unknown or stateless [...]*
2. *The child of unknown parents who is found abandoned in the territory of the Republic shall, provided possession of another citizenship is not proved, be deemed citizen by birth.*

```
%clause 1.a
obtain(cit(X,italy,birth,art1.1.a),T):-
          holdAt(child_of(X,Y),T),
          holdAt(cit(Y,italy),T),

%clause 1.b
happen(acquire_cit(X,italy,birth,art1.1.b),T):-
          happen(born_in(X,italy),T),
          (
          holdAt(unknown_child_of(X,Y),T1);
          (holdAt(child_of(X,Y),T) , holdAt(stateless(Y),T))
          ),
          (
          holdAt(unknown_child_of(X,Z),T1));
          (holdAt(child_of(X,Z),T) , holdAt(stateless(Z),T))
          ),
          not(Y=Z).

%clause 2
happen(acquire_deemed_cit(X,italy,birth,art1.2),T):-
          happen(found_in(X,Y),T),
          holdAt(abandoned_in(X,italy,T),
          holdAt(unknown_child_of(X,Y1),T)),
          holdAt(unknown_child_of(X,Y2),T)),
          not(Y1=Y2),
          not(holdAt(cit_of(X,Z)),T),
          not(Z=italy).

% Additional information
holdAt(unknown_child_of(X,Y),T):-
          not(holdAt(child_of(X,Y),T)).
```

Comments:

The interpretation of article 1 is affected by a difficulty I have discussed in section 3.1, with regard to the contrast definitional vs. event-based view. The choice made here is that clause 1.a is definitional, while clauses 1.b and 2 are event-base. Textual evidence warrants this choice: clause 1 does not contain reference to events in a tensed form, while the other clauses do. Clause 1.a

was, thus, formalized by using a static representation involving only '`obtain`' predicates; the remaining clauses, instead, were rendered, more dynamically, by using '`happen`' predicates in the consequents.

This discussion brings me to the problem of *non-initiated fluents*. Clause 1.a attempts to remain neutral as to when precisely the fluent '`cit`' comes into being, and so it was not formalized by means of a '`happen`' predicate marking an initiating event. Note, however, that article 1.a is not a definition *strictu sensu*, and it should be understood as involving a certain degree of dynamicity. For suppose it is a definition, then as soon as both Italian parents lose Italian nationality[10] their child stops being Italian. This is—I hold—an unintended inference, whence clause 1.a cannot be seen as a definition *strictu sensu* . Interestingly enough, in the EC the status of Italian citizen of the child would still hold, even when both its parents lose their Italian nationality. This is the case because of the principle of inertia captured by axioms 2 (once a fluent holds, it persists through time).

Another crucial issue concerns the application of negation as failure. It was used to express the sentence '*provided possession of another citizenship is not proved*', and so the following formulas occur in the antecedent of the third rule:

```
not(holdAt(cit_of(X,Z)),T1),
not(Z=italy).
```

The locution '*not proved*' is very close to the procedural meaning of '`not`' in the formula '`not`(φ)', where 'φ' holds if it is proved, and its negation holds if such a proof is missing (the query ?φ fails).

Negation as failure was used for expressing the fact that the parents of a child are unknown, i.e., by using '`not(holdAt(son_of(X,Y),T1))`'—the rationale being that if the unification of the variable '`Y`' with any constant fails, then no parent of '`X`' is known.

A third observation is required as to avoid possible confusion surrounding the meaning of the fluent '`cit`' or the event '`acquire_cit`'. The present formalisation implies, for example, that, when a child is born in Italy from unknown parents, it acquires Italian citizenship (second rule, clause 1.b). By contrast, some may hold that such an acquisition takes place *after* birth, once all official documents have been submitted and approved by the competent authority. This seems a minor point, but it is not. For suppose a child is born in Italy from unknown parents; at the time of the birth, both parents are unknown, but one of them shows up afterwards, and he or she holds another nationality. Does that mean that the child cannot be considered an Italian citizen according to article 1.1.a? The formalization tells us that it can. On the other hand, if citizenship initiates when all documents have been submitted and approved, the outcome may be different. If the parent shows up before the documents

[10] For example, one can lose Italian nationality if he serves in the army of a foreign country during the state of war, according to 12 of ICL.

have been submitted and approved for obtaining Italian nationality according to clause 1.b, then the child will not be entitled to acquire Italian nationality anymore. This example suggests that a further distinction needs to be made, the one between *being* or *becoming* Italian, and *qualifying for* being or becoming Italian. ICL takes care only of the latter. As a consequence, fluents such as '`cit`' (and similar events) should have been written in the form '`qualify_for_cit`'.[11]

Article 3 (adoption):

1. *An alien, who is a minor, shall acquire Italian citizenship if he is adopted by an Italian citizen.*
2. *Paragraph 1 also applies to persons who were adopted before this statute came into force.*
3. *When revocation of the adoption is based on a fact of the adopted he loses Italian citizenship,*[12] *if he possesses another citizenship [. . .].*
4. *In the other cases of revocation the adopted maintains Italian citizenship. Nevertheless, if revocation occurs when the adopted is of full age, the latter will be entitled to renounce Italian citizenship within one year after the revocation itself, if he possesses another citizenship or he regains it.*

```
%clause 1
happen(acquire_cit(X,italy,adoption,art3.1),T):-
          holdAt(minor(X),T),
          happens(adopt(Y,X),T),
          holdAt(cit(Y,italy),T).

%clause 2
happen(acquire_cit(X,italy,adoption,art3.2),T3):-
          happen(apply(X,art3.1),T1),
          happen(enact(icl,ART),T2),
          T1<T2,
          (T3=T2 ; T3>T2).

%clause 3
happen(lose_cit(ADOPTED,adoption_revocation,art3.3,T):-
          happen(revocation(adoption(ADOPTED,ADOPTER),fact_of_adopted),T),
          holdAt(cit(ADOPTED,Z),T),
          not(Z=italy).

%clause 4 (permission-part)
```

[11] For article 25 of ICL refers to further regulations to be implemented which finalize the acquisition of citizenship.

[12] The phrase 'based on a fact of the adopted' may sound obscure. It is a close rendering of the Italian 'per fatto dell'adottato'. The locution simply indicates the case in which the revocation of the adoption is due to a fact, an action, for which the adopted is responsible. The provision does not specify what this "fact" might be like, and, in fact, the revocation of the adoption (based on a fact of the adopted) is regulated by articles 305 and 306 of the Italian Civil Code, and by Law n. 184/1993, on adoption, articles 52 and 53. So the provision contains an implicit cross-reference to other pieces of legislation. Note that if the adoption is revoked because of a fact of the *adopter*, the adopted does not lose Italian nationality.

```
 happen(start_allowed(renounce(ADOPTED,cit(X,italy)),art3.4),T1):-
          happen(revocation(adoption(ADOPTED,ADOPTER),Z1),T1),
          not(Z1=fact_of_adopted),
          holdAt(full_age(ADOPTED),T1),
          (holdAt(cit(ADOPTED,Z2),T1) ; happen(acquire(cit(ADOPTED,Z2),T1)),
          not(Z2=italy).

%clause 4 (end permission-part)
happen(end_allowed(renounce(X,cit(X,italy)),art3.4),T1):-
          holdAt(allowed(renounce(X,cit(X,italy)),art3.4),T1),
          holdAt(time_passed(
                start_allowed(renounce(X,cit(X, italy)),art3.4),365),T1).

% Additional information
initiate(
          start_allowed(renounce(X,cit(X,italy)),art3.4),
          time_passed(start_allowed(renounce(X,cit(X,italy)),art3.4),0),
          T).

release(
          start_allowed(renounce(X,cit(X,italy)),art3.4),
          time_passed(start_allowed(renounce(X,cit(X, italy)),art3.4),0),
          T1).

trajectory(
          allowed(renounce(X,cit(X,italy)),art3.4),
          T,
          time_passed(start_allowed(renounce(X,cit(X, italy)),art3.4),N+D),
          D) :-
        holdAt(time_passed(start_allowed(renounce(X,cit(X,italy)),art3.4),N),T).
```

Comments:

I will look at each clause separately, except for clause 1 whose formalization is self-explanatory. Clause 2 states that the applicability of clause 1 is retroactive, and its formalization is as follows:

```
happen(acquire_cit(X,italy,adoption,art3.2),T3):-
      happen(apply(X,art3.1),T1),
      happen(enact(icl,ART),T2),
      T1<T2,
      (T3=T2 ; T3>T2).
```

As discussed and motivated in section 3.3 on cross-reference phenomena, the event '`apply`' has to be linked to article 3, clause 1, to which it refers:

```
happen(apply(X,art3.1),T):-
      happen(acquire_cit(X,italy,adoption,art3.1),T).
```

In clause 3, the predicate '`lose_cit`' is used to express losses of citizenship due to the revocation of the adoption. The loss of citizenship does not follow automatically from the revocation of the adoption, but it is restricted to cases in which the adopted has committed a serious crime (a fact of the adopted) to the detriment of the adopter, e.g., murder. This restriction is marked in the event '`revocation(adoption(ADOPTED,ADOPTER),fact_of_adopted)`', which has two arguments: the object or target of the revocation, '`adoption(ADOPTED,ADOPTER)`', and the reason for it, '`fact_of_adopted`', where variables in upper case letters have intuitive names.

Note, however, that the scope of the restriction—i.e., what kind of crime will cause the citizenship of the adopted to be revoked—is not specified by ICL, nor is provided any explicit reference to other pieces of legislation. We are only given a *semantic cross-reference* as opposed to a *syntactic cross-reference* in which the number of the law is provided. Clearly, we need to rely on the existing legislation concerning adoption, namely Law n. 184/1983 and Law n. 149/2001. Yet the reference to the two laws on adoption is not made explicit. The content of clause 3 and its exact meaning is made accessible, for the essential part, only to legal practitioners and specialists.[13] I consider this a fault of the legislator. In fact, a simple phrase such as '*according to law n. ... and its successive modifications*' would have sufficed.

Finally, clause 4 is interesting for two reasons: it requires one to use the predicate '`trajectory`' and, secondly, it introduces the deontic notion of permission with the fluent '`allowed`'.

The permission to renounce Italian citizenship is acquired by the adopted whenever her adoption is revoked and she is of full age, provided she possesses or regains another citizenship. However, such a permission lasts only for one year. To capture this, I have devised two events:

`start_allowed(renounce(X,cit(X,italy)),art3.4)` and
`end_allowed(renounce(X,cit(X,italy)),art3.4)`.

The meaning of the former event is straightforward, and it is defined clearly in clause 4, and also in the fourth rule in the formalization. The formalization of '`end_allowed`' is more laborious and requires the introduction of the parametrized fluent '`time_passed`', which contains two arguments: the first is the event with respect to which the passing time is measured, and the second argument registers the time passed (in days) and functions as a counter of the progression of time since the event contained in the first argument occurred. In the formalization, the event '`end_allowed`' obtains when the second argument of '`time_passed`' reaches the value of '365', namely 1 year:

```
happen(end_allowed(renounce(X,cit(X,italy)),art3.4),T1):-
      holdAt(allowed(renounce(X,cit(X,italy)),art3.4),T1),
```

[13] In general, it seems that there are two kinds of revocation of the adoption: One is based upon deficiencies from the part of the adopter (e.g, she cannot maintain the adopted); the other upon crimes committed by the adopted, as explained in article 51, Legge n. 184/1983, to which clause 3 should have contained a reference.

```
holdAt(time_passed(
      start_allowed(renounce(X,cit(X,italy)),art3.4),
      365),T1).
```

The formalization of the article should stop here, but some additional information on how the value '365' is arrived at is necessary (see the part under '`additional information`' in the formalization). The counter of the parametrized fluent is set at 0 when the permission to renounce citizenship is acquired, and the progression from 0 to 365 is made possible by releasing it from the law of inertia. This is represented by these two formulas:

```
initiate(
    start_allowed(renounce(X,cit(X,italy)),art3.4),
    time_passed(start_allowed(renounce(X,cit(X,italy)),art3.4),0),
    T).

release(
    start_allowed(renounce(X,cit(X,italy)),art3.4),
    time_passed(start_allowed(renounce(X,cit(X,italy)),art3.4),0),
    T1).
```

The rest of the job is done by the predicate '`trajectory`' which determines the progression of the value of the second argument in '`time_passed`':

```
trajectory(
       allowed(renounce(X,cit(X,italy)),art3.4),
       T,
       time_passed(start_allowed(renounce(X,cit(X,italy)),art3.4),N+D),
       D)  :-
   holdAt(time_passed(start_allowed(renounce(X,cit(X,italy)),art3.4),N),T).
```

The same pattern of formalization that adopts the fluent '`time_passed`' can be reused in many different cases, whenever *deadlines* or *cutoff points* are involved (see articles 4, 5, 9, and 10).

Article 4 (residency):

1. *An alien or a stateless person, whose father or mother or one of the ascendants in second degree were Italian nationals by birth, shall become Italian national: [...]*
   (c) *if, when he becomes of full age, he had his legal place of residence in Italy for at least two years and he declares, within one year after reaching the age of majority, that he wants to acquire Italian citizenship.*

```
% clause 1 (preamble)
happen(acquire_cit(X,italy,residence,art4.1),T1):-
         (holdAt(alien(X),T1) ; holdAt(stateless(X),T1)),
         (holdAt(child_of(X,Y),T1) ; holdAt(descendent(X,Y,second_degree)),T1),
```

```
          holdAt(cit(Y,italy,birth),T2),
          (T1=T2 ; T1>T2),
          happen(met_provision(X,art4.1),T1).

%clause 1, letter c)
happen(met_provision(X,art41.c),T1):-
          happen(reach(X,full_age),T3),
          (T1=T3 ; T1>T3),
          holdAt(time_passed(start_residence_in(X,italy),N),T3),
          (N=365*2 ; N>365*2),
          happen(declare(want(X,cit(X,italy)),T4),
          holdAt(time_passed(reach(X,full_age),N2),T4),
          (N2=365 ; N2<365),
          T3<T4.
```

Comments:

The text of the article cited above and the related formalization contains only clause (c), while clauses (a) and (b) were left out for brevity. The formalization went quite smoothly, and again it shows the usage of parametrized fluents for dealing with deadlines (see the expression '*within one year*' in 4.1.c). The fluent '`time_passed`' can be defined in similar ways as for article 3.

The point to be stressed is the introduction of the event '`met_provision`', which is intended to connect the rule formalizing the preamble of the article to the rule formalizing clause (c). Note that '`met_provision`' is useful to handle cross-references such as '*provided clauses (c) of article 4 are met*'.

Article 5 (marriage):

1. *The spouse, who is alien or stateless, of an Italian national acquires Italian citizenship when he has had legal residence in the Republic for at least six months, or after three years since the date of the marriage, provided dissolution, annulment, divorce did not take place, and provided legal separation is not the case.*

```
happen(acquire_cit(X,Y,italy,marriage,art5),T1) :-
          holdAt(married_with(X,Y),T1),
          holdAt(cit(Y,italy),T1),
          holdAt(time_passed(start_residence_in(X,italy),N1),T1),
          (N1=31*6 ; N1>31*6).

happen(acquire_cit(X,Y,italy,marriage,art5),T1) :-
          holdAt(married_with(X,Y),T1),
          holdAt(cit(Y,italy),T1),
          holdAt(time_passed(marriage_with(X,Y),N2),T1),
          (N2=365*3 ; N2>365*3),
          not(happen(dissolve_marriage(X,Y),T2)),
          not(happen(divorce(X,Y),T2)),
          not(happen(annul_marriage(X,Y),T2)),
```

```
not(holdAt(separation(X,Y),T1)),
happen(mariage_with(X,Y),T3),
T3<T2<T1.
```

Comments:

Article 5 shows, once again, the usage of parametrized fluents for counting the months of residency in Italy or the years of marriage that are necessary to acquire Italian citizenship. In this case we are not dealing with a deadline, but with the maturation of a certain amount of time (as married couple or as resident) that is necessary to acquire citizenship. The pattern used to define '`time_passed`' in article 3 can be applied again (with small adjustments).

Two interpretative choices had to be made, as a consequence of interpretative problems. The connective '*provided*' suffers from scope ambiguity, and it was interpreted as scoping only over '*after three years since the date of the marriage*', but not over '*when he has had legal residence in the Republic for at least six months*'. To solve the scope ambiguity, the article was formalized by two separate rules.

The second interpretive choice concerns *when* the events of dissolution, annulment, etc. count as relevant for blocking the inference towards acquisition of citizenship. It was assumed the these events are relevant if they occur between the time of the marriage and the time of 3 years of marriage.[14] As a consequence, the sentence beginning with '*provided*' could be rewritten this way: *[. . .] provided dissolution, annulment, divorce did not take place, in the meantime*.

What is more, the reference to dissolution, annulment, etc. seems redundant. For whenever a marriage is dissolved, annulled, etc., and hence terminated, the acquisition of citizenship by marriage cannot take place by definition (there is no valid marriage in the first place!), *unless* article 5 says that the acquisition of citizenship by marriage *can* take place also when the marriage has been terminated, except in the cases of dissolution, annulment, etc. This seems the only interpretation under which mentioning dissolution, annulment, etc. would not be redundant, but it is indeed an implausible one.

One argument to justify the reference to dissolution, annulment, etc. may be that article 5 is intended to rule out the cumulation of the years of marriage. For example, consider the case in which someone was married with an Italian citizen for 2 years, then got divorced, and finally got married again with the same person, but for only 1 more year. If this scenario is not to satisfy the requirement of 3 years of marriage contained in article 5, this is because dissolution, annulment, etc. took place in the meantime, even though the marriage still holds, and 3 (cumulative) years of marriage passed. A way to avoid the cumulation of years is by guaranteeing, while counting the years of marriage, that we are talking about the very same marriage, and not simply the marriage

[14] This explains the constraint '`T3<T2<T1`' in the second rule, where '`T3`' is bound to the date of the marriage, '`T2`' to dissolution, annulment etc., and '`T1`' to the acquisition of citizenship.

with the same person; an alternative consists of securing that the 3 years are uninterrupted.

Because of the above interpretive problems, article 5 may need to be rephrased in a way that avoids the scope ambiguity of '*provided*' and the problematic reference to dissolution, annulment, etc.:

1. *The alien or stateless person who is currently married with an Italian national acquires Italian citizenship, provided*
   a) *she has had legal residence in the Republic for at least six months, or*
   b) *an uninterrupted period of three years have passed since the date of the considered marriage.*

To be precise, the rewriting of letter b) is slightly redundant: either '*uninterrupted*' or '*considered*' should be used, where the former secures that the three years are uninterrupted, and the latter secures that we are talking about the very same marriage. As the criteria of identity of marriage may be controversial (e.g., one may claim that a marriage is the same iff it is with the same person), I hold that using '*uninterrupted*' would be the best solution.

Article 9 (decree):

1. *The President of the Republic, after consulting the State Council, on the recommendation of the Minister of Internal Affairs, grants Italian citizenship to:*
   a) *an alien, whose father or mother or one of the direct ascendants in II degree were Italian nationals by birth, or who was born in the territory of the Republic and, in both cases, has had his legal residence in Italy for at least 3 years, saving art. 4, clause 1, letter c) [...]*

```
% Clause 1, preamble
happen(grant(president,cit(X,italy),art9.1),T1):-
          holdAt(consulted(president,state_council,cit(X, italy)),T1),
          holdAt(recommended(president,interior_minister,cit(X, italy)),T1),
          happen(met_provision(X,art9.1.a),T1).

% Clause 1, letter a)
happen(met_provision(X,art9.1.a),T1):-
          holdAt(alien(X),T1),
          (
          holdAt(child_of(X,Y),T1) ;
          holdAt(descendent(X,Y,second_degree),T1);
          (happen(born_in(X,italy),T) , (T=T1;T<T1))
          ),
          holdAt(cit(Y,italy,birth)T2),
          T2<T1,
          holdAt(time_passed(start_residence(X,italy),N),T4),
          (N=365*3 ; N>365*3),
          T4<T1,
          [not(happen(met_provision(X,art4.1.c),T1))].
```

Comments:

It is remarkable that some of the patterns used in formalizing other articles can be applied quite mechanically in this case as well. These patterns are: the use of the fluent '`met_provision`' (see article 4), and of the fluent '`time_passed`' (see articles 3, 4, and 5).

The cross-reference '*saving art. 4, clause 1, letter c)*' deserves some discussion. It was rendered with negation as failure by placing the following formula in the body of the second rule:

```
[not(happen(met_provision(X,art4.1.c),T1))],
```

which implies that whenever the provisions of article 4.1.c are met, the conclusions deriving from article 9.1.a are blocked. This is not precisely what the expression '*saving ...*' means, but it is hard to see an appropriate formal rendering of it. At any rate, as suggested by the use of the square brackets, I will argue that the expression is unnecessary, yet its rationale is understandable at first: article 9 requires 3 years of residence, while article 4 requiers only 2 years. So if the two articles apply to the same person, they will give different outcomes, whence we add the expression '*saving art. 4, clause 1, letter c)*' to avoid divergent outcomes. However, article 3 on adoption could also apply to the same person (an alien who married an Italian, had residence in Italy, and was adopted by an Italian); article 3 states that the adopted becomes Italian citizen at the time of the adoption, so with 0 years of residence. As a consequence, one should add also '*saving art. 3*'—and there may be other articles which apply to the same individual and give different outcomes. Moreover, it does no harm if different articles give different outcomes as long as these are not conflicting with each other[15]—and the most favorable outcome is the one to prefer. For this reason, I consider the cross-reference '*saving art. 4, clause 1, letter c)*' a redundant expression.

4.2.2 Exceptions and Cross-referencing

As phenomena of exception and cross-referencing are closely related, I will deal with them at the same time. To say that article n is an exception to article m involves a cross-reference between n and m, implicitly or explicitly. Here, I shall focus on the formalization of (parts) of articles 6 and 10. They contain similar constructions indicating exceptions combined with cross-referencing. Other cross-referencing phenomena were modeled before, in article 3, clause 2, and article 9, clause 1, letter c).

Article 6 (preclusion to citizenship by marriage):

1. *Acquisition of citizenship pursuant to art.5 is precluded by:*
   a) *conviction for one of the offences provided by book II, title I, parts I, II, III, of the Penal Code; [...]*

[15] On conflicting outcomes, see section 4.2.3 below.

2. Rehabilitation avoids the results of the sentence. [...]

```
% Clause 1, preamble
happen(block(acquire_cit(X,italy,marriage,art5),art6.1),T):-
        happen(met_provision(X,art6.1),T).

% Clause 1, letter a)
happen(met_provision(X,art6.1.a),T):-
        happen(conviction(
                          X,
                          CRIME,
                          PUNISHMENT_GIVEN,
                          AUTHORITY,
                          YEARS1,
                          EDITTAL_PUNISHMENT,
                          YEARS2,
                          ART
                         ),
                          T1),
       (
       ART = penal_code.bookII.titleI.partI;
       ART = penal_code.bookII.titleI.partII;
       ART = penal_code.bookII.titleI.partIII
       ).

% Clause 2
happen(block(block(acquire_cit(X,italy,art5),art6.1),art6.2),T):-
       happen(rehabilitation(X,PUNISHMENT,ART),T),
       (
       ART = penal_code.bookII.titleI.partI;
       ART = penal_code.bookII.titleI.partII;
       ART = penal_code.bookII.titleI.partIII
       ).
```

Comments:

The expression '*precluded by*' was rendered by the event '`block`' in the first rule above. Yet it is unclear what such a preclusion amounts to. To spell it out, one can add the following rule:

```
clipped(T1,cit(X,italy,marriage,art5),T2) :-
     holdAt(blocked(acquire_cit(X,italy,marriage,art5)),T),
     T1<T<T2.
```

For the above to work, it is assumed that '`initiate(block(...),blocked(...),T)`'. So, if the fluent '`blocked`' is made true, this will terminate the fluent '`cit`'. This approach is susceptible to two criticisms. One is that the above formula is not allowed by the syntax of EC-$\mathcal{S}$. Unless extending the syntax in the required way leads to unforeseen complications, this should not be a major problem. Another

criticism is that preclusion mentioned in clause 1, letter a) is directed toward the *acquisition* of citizenship, and not to the citizenship itself, contrary to what the above formalization suggests.[16] So a better rendering could be as follows:

```
initiate(acquire_cit(X,italy,marriage,art5),T2)
                fail :-
       holdAt(blocked(acquire_cit(X,italy,marriage,art5)),T)
              succeed.
```

This rendering uses an integrity constraint, meaning: if the fluent '`blocked`' obtains, then the "initiation" of citizenship should fail. Unfortunately, this (more textually precise) rendering may lead to inconsistencies.[17] So the initial rendering should be preferred after all, in lack of better solutions.

The predicate '`conviction`' needs some explanation, because of the 8 arguments it contains. Fewer arguments would have been sufficient with respect to the piece of law under formalization, but I added more arguments in view of the other clauses of article 6, which are left out here for brevity.[18] I followed the idea of *case semantics* (which has been incorporated in EC; see Kowalski [15]). Most of the arguments are self-explanatory. Those that are in need of explanation are '`PUNISHMENT_GIVEN`', '`EDITTAL_PUNISHMENT`', ' `YEARS1`' and '`YEARS2`'. In jurisprudence, one distinguishes between the punishment which has been effectively inflicted by the competent authority (e.g., a court of law)

[16] For the predicate '`clipped`' applies to '`cit`' and not to '`acquire_cit`'.

[17] That inconsistencies may arise can be shown by an example. Suppose an individual ι_1 gets married with the Italian national ι_2 and satisfies, for `T=1998`, the requirements—pursuant to article 5—for becoming an Italian citizen. So the following holds:

`happen(acquire_cit(`ι_1`,`ι_2`,italy,marriage,art5),1998).`

It is to be assumed that the event '`acquire_cit`' initiates the fluent '`cit`'. (This is not specified in the formalization of article 5, but the formula '`initiate(acquire_cit(...),cit(...),T)`' is (and should be) added subsequently in order make the formalization work with EC. See point 3 in the methodology, p. 34.) Thus, we have:

`? initiate(acquire_cit(`ι_1`,`ι_2`,italy,marriage,art5)`

`cit(`ι_1`,`ι_2`,italy,marriage,art5),`

`1998) succeed.`

Moreover, suppose the individual ι_1 has been convicted for one of the offences of the Penal Code, according to article 6, clause 1, letter a). By the formalization of article 6, and the integrity constraints of the shape (C), the following should be the case:

`? initiate(acquire_cit(`ι_1`,`ι_2`,italy,marriage,art5)`

`cit(`ι_1`,`ι_2`,italy,marriage,art5),`

`1998) fail.`

This causes a loss of database integrity, because an (instance of) the integrity constraint has been violated.

[18] Relative to clause a), three arguments for '`conviction`' would have been enough: the person '`X`' who was convicted, the crime '`CRIME`' for which she was convicted, and the article '`ART`' of the law upon which the conviction is based. Clause b), instead, contains the notion of edittal punishment (as opposed to the punishment effectively inflicted), and that of numbers of years of punishment. This required to introduced additional argument: '`PUNISHMENT_GIVEN`', '`EDITTAL_PUNISHMENT`', '`YEARS1`', and '`YEARS2`'.

and the punishment as it is prescribed in the legislation (this type of punishment is called in Italian 'edittale', from which the English rendering 'edittal'). Likewise, the years of punishment will be distinguished according to the given and the edittal punishment: '`YEARS1`' and '`YEARS2`' respectively.

The last remark concerns clause 2, which is an exception to clause 1, which, in turn, is an exception to article 5—hence, clause 2 contains an exception to an exception. Such a second-degree exception was rendered, in the third rule, by the event '`block(block(acquire_cit(X,italy,art5),art6.1),art6.2)`'. As in the case of clause 1, a suitable rule needs to be added:

```
clipped(T1,blocked(cit(X,italy,marriage,art5)),T2) :-
     holdAt(blocked(block(acquire_cit(X,italy,marriage,art5))),T).
```

The rules says that second-degree exceptions[19] block first-degree exceptions from producing their effects.

Article 10 (oath of fidelity):

1. *The decree which grants Italian citizenship has no effects, provided the person concerned, within six months after the notification of the decree, does not take the oath of fidelity to the Republic, the Constitution and the national law.*

```
happen(block(grant(president,cit(X,italy),art9.1),art10),T):-
          not(
             happen(oath_of_fidelity(X,Y),T2),
             (Y=republic, Y=constitution, Y=national_law),
             holdAt(time_passed(grant(presiden,cit(X,italy),art10),N),T2),
             (N=31*6 ; N<31*6),
             T1<T2
             ).
```

Comments.

The proposed formalization uses the event '`block`' to render the meaning of '*has not effect*'. However, as in the case of article 6, the event '`block`' needs to be defined by an additional rule. The same pattern used for article 6 can be applied for article 10 as well:

```
clipped(T1,cit(X,italy,decree_of_president_of_republic,art5),T2) :-
     holdAt(blocked(grant(president,cit(X,italy),art9.1),art10),T).
```

4.2.3 Applicability, Repeal and Enactment

The final group of formalized articles I am going to present consists of articles 18, 26 and 27. Article 18 was selected because it contains an example of how periods of applicability are specified in legislative texts. Article 18 is interesting

[19] Once again, it is assumed that `initiate(block(...),blocked(...),T)`.

for two more reasons: it contains a new example of a cross-reference construct (different from the ones analyzed in articles 3, 6, and 9), and arguably it provides a piece of evidence that (unfortunately) the language of EC is not sufficient to model legislation.[20] The other articles, 26 and 27, are interesting because they mention the events of repeal (of other statutes) and enactment (of ICL).

Article 18 (applicability):

1. *The persons who resided in the territories which were under the House of Austria, and who emigrated abroad before 16 July 1920 [...] – pursuant to art.9, clause 1, letter a) – are deemed aliens of Italian origin or born in the territory of the Republic.*

As anticipated, this article exceeds the expressive power of EC, because the rule language upon which EC is based does not allow disjunctions to occur in the head of rules. Article 18, instead, requires such a disjunction, for its logical form can be given as follows:

> IF someone resided in the territories which were under the House of Austria and emigrated abroad before 16 July 1920 [...] – pursuant to art.9, clause 1, letter a) – ,
>
> THEN he will be deemed alien of Italian origin OR born in the territory of the Republic.

I will now formalize article 18 by assuming that a disjunction can occur in the head of a rule:

```
(
(  holdAt(child_of(X,Y),T1) ; holdAt(descendent(X,Y,second_degree),T1)),
   holdAt(cit(Y,italy,birth)  )
;
(  happen(born_in(X,italy),T)  )
                       :-
         holdAt(resident_at(X,Y),T2),
         holdAt(territory_in(Y,house_of_austria),T2),
         happen(migrate_abroad(X),T3),
         T2<T3<1920.
```

Comments:

Let us look carefully at the head of the rule. It consists of two disjuncts. The first one expresses '*alien of Italian origin*'. Article 18 does not state clearly what this means, but it cross-refers to article 9, clause 1, letter c), where an alien of Italian origin is presumably someone who is the child of an Italian citizen by birth or someone who is descendent in second degree from Italian nationals. Thus, the first disjunct is as follows:

[20] It has to be determined how widespread the syntactic construct used in article 18 is. If it is not widespread, then the fact the EC cannot formalize it properly should not represent a serious objection to the use of EC for modeling legislation. I will argue that this construct introduces a form of uncertainty in the law, which is undesired.

```
(holdAt(child_of(X,Y),T1) ; holdAt(descendent(X,Y,second_degree),T1)),
holdAt(cit(Y,italy,birth)
```

Syntactically speaking, the first disjunct is a conjunction where one of the conjuncts is a disjunction.[21] This complicated construction is needed to express '*alien of Italian origin*' in accordance with what is stated in article 9, clause 1, letter c). The second disjunct in the head of the rule is '`(happen(born_in(X,italy),T))`', and it easily expresses the notion of '*born in the territory of the Republic*'.

Why (if at all) should we need to have disjunctions in the head of rules formalizing legislation? Suppose an individual satisfies the body of the above rule, then which disjunct shall be derived? Which one should be picked? Does it make a difference? Placing a disjunction in the head of a rule seems to introduce an undesired uncertainty in the law. For instance, an article saying '*those who reach their nineteenth year of age are entitled to free medical insurance or to a 5 per cent increase in their pension*' does make sense, provided it grants the right to a choice between, say, φ_1 or φ_2; if so, the disjunction can be replaced by, for example, the formula '`choose_between(`φ_1, φ_2`)`'.

On the other hand, the same article, when interpreted as yielding a simple disjunction, commits us to the conclusion that there are disjunctive states of affair (e.g., the disjunction of being entitled to φ_1 or being entitled to φ_2). Without entering the controversial debate about disjunctive states of affairs, if we restrict our attention to the legal domain, it does not seem to be acceptable that laws talk about disjunctive state of affairs. Imagine an old man of nineteen years old entitled to free medical insurance *or* 5 per cent increase of his pension. He is entitled to the disjunction of the two, so that he is not really entitled to any of the two, only to their disjunction. That would be a rather peculiar way of living—being entailed to a disjunction!

So if the above reasoning is correct, disjunctions in the head of rules are to be avoided, or, at least, used with parsimony. However, a fuller treatment of the problem will require an in-depth discussion which falls outside the scope of this chapter. In addition, it remains to be seen whether articles of the form '$\varphi_1 \vee \varphi_2 \leftarrow \psi$' are widespread in the legislation or not. For the time being, it should suffice to have pointed out the problem.[22]

A further comment is that there is no explicit formalization of the expression '*pursuant to art.9, clause, 1, letter c)*', except that the head of the rule was written by looking at the body of the rule formalizing article 9. A way to make the cross-reference apparent in the formalization is by adding an argument, indicating the article number, to the predicates occurring in the body of the rule for article 9 (thus far, the policy has been to indicate article numbers only in formulas occurring in the head of rules).

[21] Recall that it is no problem to have conjunctions in the head of rules.

[22] One more aspect to take into account is that, in article 18, the disjunction in the consequent occurs in the context of the deeming provision '*be deemed aliens of Italian origin or born in the territory of the Republic*'.

Finally, as mentioned previously, the reason why article 18 was selected is that it contains a restriction on its period of applicability, given by the expression '*emigrated abroad before 16 July 1920*'. Rendering this construction was fairly straightforward, by adding these formulas in the body of the rule:

```
happen(migrate_abroad(X),T3),
T2<T3<1920.
```

Article 26 (repeal):

1. *Statute 13 June 1912, n.555, statute 31 January 1926, n.108, royal law-decree 1 December 1934, n.1997, converted by statute 4 April 1935, n.517, art.143-ter civil code, statute 21 April 1983, n.123, art.39 of statute 4 May 1983, n.184, statute 15 May 1986, n.180, and every other provision which is incompatible with this statute are repealed.[...]*

```
happen(repeal(OTHER_LAW,art26.1),T1) :-
          (OTHER_LAW=555 ; OTHER_LAW=108 ; OTHER_LAW=1997 ; OTHER_LAW=517 ;
          OTHER_LAW=civil_code_art.143.ter; OTHER_LAW=129 ;
          OTHER_LAW=statute4.art.39 ; OTHER_LAW=184 ; OTHER_LAW=180).

happen(repeal(OTHER_LAW,art26.1),T1) :-
          happen(conflict(OTHER_LAW,icl),T1).
          happen(enact(OTHER_Law),T2),
          happen(enact(icl);T3),
          T2<T3.
```

Comments:

When the number of articles or entire laws, which are to be repealed, is explicitly mentioned, the formalization goes smoothly: The event '`repeal`' is the inverse of '`enact`', and one needs to specify which laws or articles are repealed, by writing constraints of the form '`OTHER_LAW=555`'. Unfortunately, problems arise whenever no explicit mention of articles under repeal is given.[23] This is the case in the last sentence of clause 1, which says that if conflicts arise between ICL and older laws (actually it is written *every other provision*, so also future ones!), the older laws are treated as if they had been repealed. This principle—used in jurisprudence to cope with conflicting norms—is know as *lex posterior* (more recent legislative texts take priority over older ones).[24]

The challenge is to capture the meaning of the event '`conflict`', which is left under-specified in the formalization. Negation as failure can cause problems if it

[23] Not by chance, guidelines for legal drafting strongly advise not to use vague cross-referencing expressions such as '*in any other provision*'.

[24] Other principles used to cope with conflicting laws are *lex specialis* (more specific legislative texts take priority over more general ones) and *les superior* (more important legislative texts take priority over less important ones). It is debated which priority ordering hold between the three principles themselves.

is used without care. For suppose the query '`?holdAt(cit(ι,italy,icl),1999)`' succeeds, while the query '`?holdAt(cit(ι,other_law),1999))`' fails; then, there are two explanations why the second query failed: one is that '`other_law`' contains requirements to become Italian citizen that 'ι' does not met; the second explanation is that '`other_law`' is about a matter that is completely different from that of citizenship, and thus '`other_law`' cannot grant citizenship to anyone. In the former case, there is incompatibility, but not in the latter case. So, formulas of the shape '`holdAt(cit(X,law1),T)` ∧`not(holdAt(cit(X,law2),T))`', for some variable assignment, need not represent a conflict between '`art1`' and '`art2`'. The incompatibility between two pieces of law arises whenever they deliver incompatible outcomes about the same matter, and in addition the two pieces of law are about the very same matter (e.g., citizenship). A way to capture this is as follows:

```
happen(conflict(OTHER_Law,icl),T):-
      not(holdAt(F(X,OTHER_LAW),T)),
      holdAt(cit(X,icl),T),
      F=cit.
```

The key ingredient is the constraint '`F=cit`', which secures that a conflict is shown to exists only when the failure of '`holdAt(F(X,OTHER_LAW),T)`' concerns the matter of citizenship. If the constraint were not satisfied, there would be no conflict, as the event '`conflict`' would not be derivable from the rule.

Article 27 (enactment):

1. *This statute shall come into force after six months from its publication in the Gazzetta Ufficiale.*

```
happen(enact(icl,art27),T2):-
         happen(publish(icl),T1),
         holdAt(time_passed(publish(icl),N),T2),
         N=6*31,
         T1<T2.
```

Comments:

This article does not raise particular problems. There is one curious thing, however. Article 27 gives rise to a liar-like paradox. If article 27 comes into force (=is enacted) when ICL is published, then ICL is violated, since it prescribes of itself and also of article 27 to come into force *only* after six months. Hence, article 27 cannot come into force right after its publication; but if article 27 does not come into force, the entire law will not come into force either. So, does ICL come into force after all? One could say that article 26 only comes into force after six months, and yet in order to make sure that this will be the case, article 26 should have already come into force. This impasse is avoided in the formalization, as the formulas cannot talk about themselves.

4.2.4 The Global Picture

Point 2 of the semi-algorithmic methodology (see section 4.1.2) was illustrated at length by the previous sections. It is now time to see how the isolated pieces of the formalization modeling single articles come together in a global and coherent formal representation of the law.

Once point 2 is completed, the result is a rather large set of rules in the language of EC. These rules appear to be rather chaotic and in need of a systematic arrangement. A quick look at the rules suggests that their heads are of only two kinds, depending on the predicate, '`happen`' or '`obtain`', they contain. This syntactic distinction can be refined further. For instance, many of the rules with '`happen`' in their head are about acquisition of citizenship. A new rule can, thus, be devised which makes this commonality perspicuous:

```
happen(acquire_cit(X,italy,icl),T1):-
          happen(acquire_cit(X,italy,birth,art1.1),T1);
          happen(acquire_deemed(X,italy,birth,art1.2),T1);
          happen(acquire_cit(X,italy,adoption,art3.1),T1);
          happen(acquire_cit(X,italy,adoption,art3.2),T1);
          happen(acquire_cit(X,italy,residence,art4.1.c),T1);
          happen(acquire_cit(X,Y,italy,marriage,art5),T1).
```

The new rule provides a first general view of the law, relative to acquisition of citizenship. In a similar way other rules with systematizing purposes can be added. These new rules do not add anything to the content of the formalization, but they can enhance its usability. For example, if another piece of law contains a phrase such as '*the ways in which citizenship is acquired according to* ICL *shall be inapplicable to those born after 2010*', this can be rendered as follows:

```
terminate(inapplicable(cit(X,italy,icl)),cit(X,italy,icl),T).

happen(inapplicable(cit(X,italy,icl),T):-
          happen(born(X).T),
          T>2010.
```

This illustrates how point 3.a in the methodology should be pursued and what its application may be. As for point 3.b, once predicates in the consequents are systematized in a general picture, some EC-formulas should be added to make it possible to apply the axioms of EC. These formulas will mainly contain '`terminate`' and '`initiate`' predicates. Rather obviously, we want the event '`acquire_cit`' to initiate the fluent '`cit`':

```
initiate(acquire_cit(X,Italy,icl),cit(X,italy,icl),T).
```

One can decide the level of granularity in which '`initiate`' and '`terminate`' predicates are to be used. For example, the above formula can be used for all sub-types of acquisition of citizenship, or alternatively analogous formulas for each type could be added, as follows:

```
initiate(acquire(X,Italy,birth,art1.2),
```

```
          cit(X,italy,birth,art1.2),
          T).

initiate(acquire_cit(X,Italy,adoption,art3),
          cit(X,italy,adoption,art3),
          T).                ...
```

The second choice is more fine-grained and hence more flexible and precise, yet it is more laborious.[25]

The addition of '`initiate`' but also '`terminate`' predicates should be completed in the obvious way for other events. An extended list of examples should make this point clear. The initiating events '`start_allowed`' (art. 3), '`block`' (art. 6), '`enacted`' (art. 27) should be specified as follows:

```
initiate(block(...),blocked(...),T).
initiate(start_allowed(...),allowed(...),T).
initiate(enact(...),enacted(...),T).
```

The terminating events '`lose_cit`' (art. 3) and '`end_allowed`' (art. 3) should be specified as follows:

```
terminate(loose(...),(...),T).
terminate(end_allowed(...),allowed(...),T)
```

With these additions in place, the axioms of EC can be applied to perform reasoning, and the formalization is ready to be used.

Summary

This chapter presents a sample of the formalization of ICL in the language of EC (see section 4.2), The formalization is complemented by a discussion on methodological issues (see section 4.1.2). It is argued that a bottom-up approach is preferable over a top-down approach, as only the former meets the requirement of maintainability:

(R3) A good formalization should be *maintainable*: as the texts of the law evolves, so does the formalization.

While commenting on the formalization, several questions of detail emerged—the use of parametrized fluents to express deadlines (articles 3, 4, 9, and 27); the distinction between syntactic and semantic cross-referencing (article 3 on adoption); the problem of disjunctive formulas in the head of rules, and of disjunctive state of affairs (article 18); the problem of defining the notion of conflict between two pieces of legislation (article 26).

[25] Incidentally, in view of the laborious and repetitive procedures one may encounter during the formalization, the usage of an editor may facilitate the process considerably (more on the editor will be said in the conclusion of this thesis).

5 Jurisprudence[1] Formalized

In the previous chapter, ICL was modeled and formalized in the language of EC. This chapter puts the formalization into use by looking at a precedent involving the application of ICL. The case can be succinctly described as follows. An Argentinean citizen, X, child of Italian mother and Argentinean father, was refused recognition of Italian citizenship by the Italian interior ministry. This clearly contradicts the first article of ICL:

> Citizen by birth is the child of a father or a mother who are Italian citizens.

However, the interior ministry held that ICL could not be applied to X, since X was born before the date of enforcement of ICL. Furthermore, the previously enforced legislation on the same matter established that only a male citizen can transmit Italian citizenship to the descendants. So—the ministry concluded—in accordance with the old norm, X could not be considered an Italian national, given that he was not born from an Italian father. On the other hand, X objected that the Italian Constitutional Court declared the old norm discriminatory and unconstitutional, and maintained that both female and male citizens can transmit citizenship. On this ground, X claimed the right to acquire Italian citizenship, because the judgment of the Constitutional Court *retroactively* abolished the previous discriminatory legislation on the trasmission of citizenship. The Interior ministry, in turn, objected that the judgment of unconstitutionality has indeed retroactive validity, but such a validity cannot be extended before the date of enactment of the Italian Constitution. As a matter of fact, X was born before that date, and so the old legislation shall still apply.

The court in charge of the trial, the Tribunal of Turin of the Italian Republic (hereafter, also referred to as 'the court'), conceded that X was correct, and condemned the Italian interior ministry. Roughly, the motivation was that the date of birth is not the only time one can transmit citizenship. Although X's mother could not transmit Italian citizenship at the time of X's birth, because of the unfavourable legislation, she became entailed to do so as soon as a new legislation was enforced which conferred such a right to her (e.g., at the time in which the new ICL came into effect).

[1] Here 'jurisprudence' means 'case-law', not 'legal theory'. The term 'jurisprudence' is preferred over 'case-law', since the latter is mostly related to the English-speaking legal tradition of common-law. In the civil tradition (to which Italy belongs), case-law is typically called 'jurisprudence'.

In what follows, a more detailed description of the trial is given, in which the facts, arguments and legal data put forward by both parties, as well as the final judgment of the court, are reconstructed. The description will be in natural language first, and then followed by a formalization. The main goal of this chapter is to outline a "formal commentary" to the sentence of the Tribunal of Turin.

5.1 The Case

The main document I shall use for the reconstruction of the case is the sentence by the Tribunal of Turin (see [11]). The first part, 'svolgimento del processo' (development of the trial), contains a thorough exposition of the relevant legal facts that were taken into consideration in the court's final judgment.

The Argentinean citizen, X, requested to acquire Italian citizenship on the ground that:

(F1): He was born in the territory of the Argentinean Republic on August 18, 1942, with an Argentinean citizen as a father, and an Italian citizen as a mother.

(F2): He was subsequently recognized by both his parents.

(F3): His mother was born in Argentina, on March 9, 1906, as legitimate child of Italian citizens.

(F4): His mother has acquired Italian citizenship, albeit born in Argentina, because her parents were both Italian nationals.

The relevant legislation put forward by X comprises:

(L1): The old Italian Citizenship Law, Law n. 555/1912, precisely article 1, clause 1, as it was modified by the Constitutional Court with sentence n. 30 on February 9, 1983. The court amended the Law n. 555 in the part in which it prescribes that only male Italian citizens may transmit citizenship to their descendants.

(L2): The sentence n. 6227, on July 10, 1996, of the Italian Supreme Court (Corte di Cassazione), which rejected the previously held interpretation that the judgment of unconstitutionality is only valid from the date in which the constitution came into effect.

The interior minister did not show any counter-evidence against (F1)-(F4). It did not object to (L1) either, as this is a piece of law, amended by the Constitutional Court, and that information is unquestionable. The interior minister, instead, did move an objection against (L2), as the latter contains an *interpretation of the law*, which can be disputed. The objection to (L2) is made by referring to a different interpretation held by the supreme judiciary authority on administrative matters (citizenship included), the State Council:

(L3): The State Council, during consultation n. 105/83, claimed that only those born from an Italian mother *after* January 1, 1948 (when the Italian Constitution came into force) can acquire citizenship from the part of the mother.

5.1.1 Relevant Legislation

Upon first reading, the reason why (L3) is a rebuttal to (L2) may be obscure. It is, thus, useful to quote passages from the legislation, or interpretations thereof, that were invoked by both parties to support their claims. The unquestionable (L1) corresponds to an amended version of article 1, clause 1, of Law n. 555/1912 (hereafter, OLD-ICL). The original version of the article states:

> Citizen by birth is the child of a father who is an Italian citizen.[2]

OLD-ICL differs from ICL in that it does not comprise the case of mothers transmitting Italian citizenship.[3] As stated in article 20, OLD-ICL came into force on July 12, 1912. Subsequently, on February 9, 1983, the Constitutional Court, with sentence n. 30, established:

> ... article 1, clause 1, of the Law n. 555/1912, should be considered unconstitutional with respect to the part in which it excludes the case of a child of Italian mother becoming citizen by birth.[4]

As a result, the content of (L1) is given as follows:

> Citizen by birth is the child of a father or a mother who are Italian citizens.

Note that the above has been incorporated literally into article 1 of ICL. Now, we face the problem of when the validity and applicability of (L1) should begin. On this matter, qualified interpretations diverge, as is apparent from the conflict between (L2) and (L3). A good idea of the content of (L2) is given by the following excerpt of sentence n. 6227/1996, in which the Supreme Court, rather confusingly, claims:

> The declaration of constitutional illegitimacy of laws prior to the Constitution gives rise to the effects that are proper of such types of declaration—namely, the cessation of effectiveness, *erga omnes*[5] and with retroactive effect (which implies the general prohibition of application), of the norm declared constitutionally illegitimate, relative to situations or relationships, to which the same norm would still be applicable, if the sentence of unconstitutionality did not take place.[6]

[2] È cittadino per nascita: il figlio di padre cittadino. See [1].

[3] As a matter of fact, OLD-ICL allows mothers to transmit Italian citizenship only when the father cannot transmit his own citizenship (e.g., see article 1, clause 2, of OLD-ICL.)

[4] ... deve essere dichiarata la illegittimità costituzionale dell'art. 1, n. 1, della legge n. 555 del 1912, nella parte in cui non prevede che sia cittadino per nascita anche il figlio di madre cittadina. See [8].

[5] English renderings: in relation of everyone; towards all.

[6] La dichiarazione d'illegittimità costituzionale di leggi anteriori a Costituzione esplica gli effetti propri di tale tipo di pronuncia: e cioè la cessazione di efficacia erga omnes con effetto retroattivo (che implica il generale divieto di applicazione) della norma dichiarata costituzionalmente illegittima relativamente a situazioni o rapporti, cui sarebbe ancora applicabile la norma stessa, ove non fosse intervenuta la pronuncia di incostituzionalità. See [10].

In laymen terms, the Supreme Court contends that the judgment of unconstitutionality has *general* retroactive effects, i.e., also before the date of enactment of the Constitution. A different interpretative stance is offered by (L3). The essential idea is given by this passage:

> The validity of the sentence of the Constitutional Court cannot "retroact" beyond the moment in which the contrast between the law (or any legally binding act) and the Constitution has occurred ... that is, beyond January 1, 1948, the date of enactment of the latter.[7]

It should now be clear why, in support of their claims, X endorsed (L2) and the interior ministry endorsed (L3). For (L2) guarantees the unlimited retroactivity of the judgment of unconstitutionality, and hence the unlimited retroactive applicability of (L1). This entitles X to claim Italian citizenship. On the contrary, (L3) restricts the retroactive applicability of (L1) to events occurring *after* the date of enactment of the Constitution. Given that X was born before that date—and on the assumption that the event of birth counts to determine the applicability of (L1)—X is deprived of any right to claim Italian citizenship.

However, as anticipated, in view of the court's final decision, deciding for (L2) or (L3) was not crucial for deciding the issue at stake—the Italian citizenship of X. This is because the court held that the assumption that the event of birth determines the applicability of (L1), was misplaced. Let me explain this more precisely.

5.1.2 The Decision of the Court

X went to court because the interior ministry refused to grant him Italian nationality. From the text of the sentence, it is clear that both X and the interior ministry maintained that solving the case amounted to solving the problem of the unlimited vs. limited retroactive validity of the sentence of unconstitutionality—(L2) vs. (L3). The reasoning was along these lines: If the sentence of unconstitutionality has limited retractive validity, namely not further back than January 1948, then X cannot become an Italian national, given that X was born before January 1948 from a Argentinean father (and, to repeat, the valid legislation, back then, granted the privilege to transmit citizenship only to male citizens). If, on the contrary, the sentence of unconstitutionality has unlimited retroactive validity, then X can become an Italian national, because his Italian mother acquired—via the sentence of unconstitutionality—the right to transmit Italian citizenship, so to speak, *ab initio*.

[7] I was unable to retrieve the original text of the consultation n. 105/83 of the State Council. I cited, instead, a relevant excerpt from the sentence by the Tribunal of Turin. Here is the orignal Italian: L'efficacia del giudicato costituzionale non può in ogni caso retroagire oltre il momento in cui si è verificato il contrasto tra la norma di legge o di atto avente la forza di legge - anteriore all'entrata in vigore della Costituzione - dichiarata illegittima, e la norma o il principio della Costituzione, cioè non può retroagire oltre il 1 gennaio 1948, data di entrata in vigore di quest'ultima. See [11].

Note, however, that the above reasoning makes sense solely on the assumption (shared by X and the ministry) that the acquisition of citizenship can only take place at the time of the birth. Interestingly and surprisingly enough, the sentence of the Tribunal of Turin denied that shared assumption. This yielded, as a consequence, that X could acquire Italian citizenship as soon as a non discriminatory legislation came into effect, e.g., when the Constitution came into effect.

With respect to the dispute over (L2) or (L3), the court stood on the side of the ministry and opted for (L3), because—after all—it is non sense to say that the retroactive validity of unconstitutionality may overcome the scope of validity of the Constitution itself: how could there be any unconstitutionality without any Constitution being in place? Yet in its final judgment, the court stood on the side of X who was granted Italian nationality, given that acquisition of nationality is not limited to the time of the birth. More precisely, the court sentenced that X acquired Italian nationality when his mother acquired the right to transmit Italian citizenship, i.e., retroactively on January 1948 by means of the sentence of unconstitutionality of article 1 of OLD-ICL.

Why the court decided for (L3) rather than (L2) turned out to be an issue of secondary interest, contrary to the expectations of both the ministry and X. The crucial issue became, instead, the one of whether or not citizenship can be transmitted after the time of the birth. The court sentenced that it can, and based its decision on a previous sentence of the Supreme Court, n. 6297/96, and on textual considerations concerning how article 1 of ICL, equivalent to (L1), is phrased. First, the sentence of the Supreme Court (emphasis added):

> On this matter, it should be made precise that ... the right to acquire Italian citizenship comes into being not at the event of the "birth" ..., but rather it is based on the *situation of filiation from a parent who is a citizen*.[8]

Second, the textual considerations:

> Article 1, clauses 1 and 2, of Law n. 555/1912, as well as article 1, letter a), of the Law n. 91/1992, does not contain reference to the historical fact of the birth, but rather to the situation of filiation, as it is evident from the usage of the expression "the child" ... The expression "citizen by birth" ... presupposes the fact of the birth, but it is not equivalent to the expression "citizen at the time of the birth."[9]

[8] Deve essere precisato, in proposito, che il titolo di siffatto modo di acquisto ... della cittadinanza italiana è costituito, non già dall'evento "nascita" ..., bensì dalla situazione di filiazione da genitore cittadino. See [10].

[9] L'art. 1, punti 1 e 2 della legge n. 555 del 1912, al pari del vigente art. 1 lett. a della legge 5 febbraio 1992 n. 91 non include nella fattispecie il fatto storico della nascita, ma piuttosto la situazione di filiazione, come evidenzia l'uso dell'espressione "il figlio" ... L'espressione "cittadino per nascita" ... presuppone il fatto della nascita ma non equivale necessariamente all'espressione "cittadino dal momento della nascita". See [11].

5.2 Formalization

The court's sentence included two decisions: the one about (L2) vs. (L3), and the one about the status of X as an Italian citizen. As pointed out before, the first decision (in favour of (L3) and against the arguments by X) did not affect the second decision (in favour of X). So the first decision may be disregarded because of its neutrality with respect to the second decision, that is the crucial one. The two decisions, however, are interesting in their own, and I shall look at them separately.

5.2.1 Limited or Unlimited Retroactive Validity?

The discussion about the limited or unlimited retroactive validity is a discussion about the law, its validity or retroactivity—it is not a discussion that involves any reference to the events which occurred to X. For this reason, this section does not use facts (F1)-(F4), but only legal data (L1)-(L3)(and some more will be added).

As the name suggests, retroactivity is based on the notion of activity (or validity) of the law[10], but conceived of in a backward way—retroactively that is. In what follows, I shall speak of *retroactivity* or *retroactive validity* rather interchangeably, although the locution 'retroactive validity' is redundant. The notion of activity in the word 'retro*active*' contains already the notion of validity (for in section 3.2 activity and validity of the law are treated as synonymous notions), and thus saying 'retroactive validity' is the same as saying 'retroactive activity'. So, a better synonym for 'retroactivity' would be for example 'retrovalidity'.

Terminological worries aside, I will begin by giving a formal account of retroactivity in the framework of EC. We need to add a new axiom to EC, one that can express "retroactive initiation" of fluents:

Ax7: `holdAt(F,T1)` ← `happen(E,T2)` ∧ `retroinitiate(E,F,T2)` ∧ `T1<T2` ∧ ¬`clipped(T1,F,T2)`.

Axiom 7 expresses a form of backward inertia, as opposed to the forward inertia of axiom 3. The predicate '`retroinitiate`' can be seen as the reversed of '`initiate`'. Backward inertia as expressed by axiom 7 does not generally apply, but there is a restricted range of legal phenomena to which it does, those being for example the event '`retro_enact`' and the fluent '`retro_active`'. The following formulas, written symmetrically, give an account of retroactivity and activity of articles in the law:

(act$^-$) `retroinitiate(retro_enact(ART),retro_active(ART),T)`.

(act) `initiate(enact(ART),active(ART),T)`.

With this general machinery in place, the disagreement over (L2) and (L3) can be given a formal rendering. First, the disagreement is based upon the agreement on some trivial legal data:

[10] see section 3.2 on time and validity in chapter 3

(L4) a sentence of the Constitutional Court (sentence n. 30/83) took place on February 1983, and the sentence amended OLD-ICL in such a way that the first article was replaced by (L1);
(L5) the judgment of the Constitutional Court is retroactive.

The disagreement, instead, consists of whether the sentence is retroactive in a unlimited or limited manner: (L2) vs. (L3).

The agreed legal data can be formalized by ($L4^f$) and ($L5^f$) below.[11] The predicate '`sentence(CONT,AUTH,N)`' contains three variables: the content of the sentence, the authority issuing the sentence, and the number of the sentence. In ($L4^f$) below, the three variables are replaced by constants using the constraints '`N=30-83`', '`AUTH=const_court`' and '`CONT=L1`'—for the content of the sentence is given by (L1). In ($L5^f$) only the constraint '`AUTH=const_court`' is used because the intended meaning of the rule is that every sentence *of the Constitutional Court* is retroactive.

($L4^f$): `happen(sentence(L1,const_court,30-83),1983)`

($L5^f$): `happen(retro_enact(sentence(CONT,const_court,N)),T)` ←
`happen(sentence(CONT,const_court,N),T)`

Note that I have used (and will use) the convention to treat time instants in a rather simplified way: instead of writing February 9, 1983, I have written 1983. To be fully precise, one would need a function that progressively maps every date to a natural number. I will leave out this complication.

The disagreement over (L2) and (L3) can be finally formalized. (L2) does not pose any restriction on the extension of the retroactivity of the sentence of the Constitutional Court. It can be expressed as follows, where the superscript '^{f}' indicates that ($L2^f$) is the formal rendering of (L2):

($L2^f$) `holdAt(retro_active(sentence(L1,const_court,30-83)),T).`

The variable '`T`' is not restricted by anything—the retroactivity is unlimited. In fact, what is intended is that the retroactivity of the sentence of the Constitutional Court is limited by only '`T<1989`', where 1989 is the date of the sentence, as stated in ($L4^f$)-($L5^f$). Note that ($L2^f$)—limited to '`T<1989`'—is redundant, as it can be proven from the axioms of EC, axiom 7, and the agreed legal data ($L4^f$)-($L5^f$).[12]

(L3) is expressed by these two formulas

($L3^f$) `happen(limit_retro_active(sentence(L1,const_court,30-83),1948).`
`terminate(limit_retro_active(ART),retro_active(ART)),T).`

The first formula says that, at time 1948, the content (L1) of the sentence n. 80-83 of the Constitutional Court is subject to limited retroactivity (see the

[11] The superscript '^{f}' indicates that ($L4^f$) and ($L5^f$) are formal renderings of (L4) and (L5).
[12] The proof can be given as a modification of claim 1 below, that is, by unproblematically replacing the constraint '`T1<1948`' with '`T1<1989`' in claim 1, part (a).

predicate '`limit_retro_active`'). The second formula, more generally, says that the event '`limit_retro_active`' has the effect of terminating the fluent '`retro_active`'. Note that '`ART`' is a placeholder for any legislative content, inclusive of sentences of the Constitutional Court.

The disagreement of the two parties convened for the trial can be expressed by different stances on the query:

(Q1) `?holdAt(retro_active(sentence(L1,const_court,30-83)),T1),T1<1948.`

The interior ministry holds that (Q1) fails, and X holds that it succeeds. The ministry bases its claim on ($L3^f$), while the X's claim follows from the agreed legal data.

Claim **1.** Assume ($L4^f$)-($L5^f$):
(a) Then (Q1) succeeds (X's stance).
(b) If ($L3^f$) holds, then (Q1) fails (ministry's stance).

Proof. The proof is straightforward, and only a sketch is given. For representing derivations, I shall use the notational conventions (C1) and (C2) on page 13. Query (Q1) becomes:

```
? holdAt(retro_active(sentence(L1,const_court,30-83)),T1),
? T1<1948.
```

Set the constraint '`F=retro_active(sentence(L1,const_court,30-83))`'; then apply axiom 7. We have:

```
? T1<1948,
? F=retro_active(sentence(L1,const_court,30-83)),
? happen(E,T2),
? retroinitiate(E,F,T2),
? T1<T2
? not(clipped(T1,F,T2))
```

If we set '`E=retro_enact(sentence(CONT,const_court,N))`', by applying ($L5^f$), we can erase the query '`?happen(E,F)`', and replace it with the body of ($L5^f$), as follows:

```
? T1<1948,
? F=retro_active(sentence(L1,const_court,30-83)),
? retroinitiate(E,F,T2),
? T1<T2,
? not(clipped(T1,F,T2)),
? happen(sentence(CONT,const_court,N),T2).
```

The last query, given the constraints '`T2=1983`', '`CONT=L1`', and '`N=30-83`' can be eliminated by ($L4^f$). The query '`?retroinitiate(E,F,T2)`' succeeds by (act^-), given the constraints '`E=retro_enact(ART)`' and '`F=retro_active(ART)`'. Thus, we have:

```
? T1<1948,
? F=retro_active(sentence(L1,const_court,30-83)),
? T1<T2,
? not(clipped(T1,F,T2)),
? T2=1983,
? E=retro_enact(ART),
? F=retro_active(ART).
```

The query '`not(clipped(T2,F,T1))`' succeeds by negation as failure, as there is no information that makes the '`clipped`' formula true. Finally we have:

```
? T1<1948,
? F=retro_active(sentence(L1,const_court,30-83)),
? T1<T2,
? T2=1983,
? E=retro_enact(ART),
? F=retro_active(ART).
```

The variable '`E`' is unified to two different fluent terms. However, by putting '`ART=sentence(L1,const_court,30-83)`', we can see that the above list of constraints is satisfiable, whence (Q1) succeeds.

The proof for (b) is the same as for (a) up to a certain point, namely when the query '`?not(clipped(T1,F,T2))`' succeeds. In the proof of (b), instead, the same query will fail by ($L3^f$), and so does (Q1). To see this, consider the stage of the derivation of (a) where the formula '`clipped`' was not erased yet:

```
? T1<1948,
? F=retro_active(sentence(L1,const_court,30-83)),
? T1<T2,
? not(clipped(T1,F,T2)),
? T2=1983,
? E=retro_enact(ART),
? F=retro_active(ART).
```

The '`clipped`' formula can be removed by applying axiom 5. We now have:

```
? T1<1948,
? F=retro_active(sentence(L1,const_court,30-83)),
? T1<T2,
? T2=1983,
? E=retro_enact(ART),
? F=retro_active(ART),
? happen(E1,S),
? T1<S<T2,
? terminate(E1,F,S).
```

By ($L3^f$) the queries '`?happen(E1,S)`' and '`?terminate(E1,F,S)`' can be removed, given '`E1=limit_retro_active(sentence(L1,const_court,30-83))`' and '`S=1948`'. The updated list of queries is as follows:

```
? T1<1948,
? F=retro_active(sentence(L1,const_court,30-83)),
? T1<T2,
? T2=1983,
? E=retro_enact(ART),
? F=retro_active(ART),
? T1<S<T2,
? E1=limit_retro_active(sentence(L1,const_court,30-83)),
? S=1948.
```

The above list of constraints is satisfiable, so the query '`?clipped(T1,F,T2)`' succeeds; this means that '`not(clipped(T1,F,T2))`' fails, whence (Q1) fails. □

Some may lament that claim 1 does not settle the disagreement between the ministry and X because the real issue is which principle, (L2) or (L3), should be endorsed. Formally, the issue boils down to be whether (L3) should be accepted or not, because (L2), rendered as ($L2^f$), follows from the agreed legal data and axioms. Obviously, our formalism cannot *prove* which position to take, and it would be unreasonable to except a formalism to be able to do so.

Deciding between (L2) or (L3) would require to appeal to additional principles. For example, (L2) could override (L3) if we accept the principle that '*any rule based on more recent legislative sources shall take precedence over older rules*' (also known as *lex posterior*), along with the fact that (L2) is more recent than (L3). A fuller treatment of how to deal with conflicting rules or normative statements falls outside the scope of this chapter. One suggestion is to look at formal models of argumentation, where the choice of one rule over another is debated between two players in an argumentation game.[13]

5.2.2 The Crucial Issue

The crucial disagreement between X and the ministry concerns the attribution of citizenship. The crucial issue can then be captured by the query (Q2):

```
% Q2
    ? holdAt(cit(X,italy,TYPE,ART),T),
    ? holdAt(active(ART),T),
    ? T=1999
```

The ministry contends that (Q2) should fail, and X contends that it should succeed. For the query to succeed, there should be an article granting Italian citizenship to X and this article should be applicable to X for '`T=1999`' (date of the trial). From the sentence of the court, it was apparent that deciding on the query (Q2) did not require one to settle the dispute about (L2) or (L3), contrary

[13] The interested reader may consult, among others, Gordon [12], in which legal reasoning about rules in the context of argumentation theory is proposed.

to what both parties thought. This results is mirrored by the formal treatment that I am going to present.

To give a conclusive answer to (Q2), some more rules need to be written up. These will comprise a formal rendering of (F1)–(F4). I will use the convention to name X's mother, father, grandmother and grandfather with the constants '`mother, father, grandmother, grandfather`'.

(F1^f) `happen(born_in(X,argentina),1942).`

`holdAt(cit(father,argentina),1942).`

(F2^f) `happen(recognize(mother,child_of(X,mother),T)) ←T>1942.`

`happen(recognize(father,child_of(X,father),T)) ←T>1942.`

(F3^f) `happen(born_in(mother,argentina),1906).`

`holdAt(mother_of(grandmother,mother),1906).`

`holdAt(father_of(grandfather,father),1906).`

`holdAt(cit(grandother,italy),1906).`

`holdAt(cit(grandfather,italy),1906).`

(F4^f) `holdAt(cit(mother,italy),T)←T>1906.`

Let's now go back to the main point:

```
% Q2
    ? holdAt(cit(X,italy,TYPE,ART),T),
    ? holdAt(active(ART),T),
    ? T=1999
```

In accordance with the sentence of the court, this query succeeds:

Claim **2.** Assume (F1^f)–(F4^f). Then, query (Q2) succeeds.

Proof. First, let's concentrate on the first line of the query and apply axiom 6 (see page 19), by putting '`F=cit(X,italy,TYPE,ART)`'. This yields:

```
? F=cit(X,italy,TYPE,ART),
? obtain(F,T1),
? T1<T,
? not(clipped(T1,F,T)).
```

The query '`obtain(F,T1)`' can be made to succeed by applying (L1), whose content is given by the formalization of article 1.1.a of ICL, as explained in the previous chapter.[14] The formalization is repeated here for convenience:

```
obtain(cit(X,italy,birth,art1.1.a),T):-
      holdAt(child_of(X,Y),T),
      holdAt(cit(Y,italy),T).
```

[14] (L1) and article 1 of ICL boil down to the same.

By setting '`TYPE=birth`' and '`ART=art1.1.a`', the formalization of article 1.1.a can be applied. Thus we have:

```
? F=cit(X,italy,TYPE,ART),
? T1<T,
? not(clipped(T1,F,T)).
? TYPE=birth,
? ART=art1.1.a,
? holdAt(child_of(X,Y),T),
? holdAt(cit(Y,italy),T),
```

The formula '`not(clipped(T1,F,T2)`' succeeds by negation as failure, and can be erased. Moreover, given the convention (val), on page 23, that any fluent which is not indexed to an article must be reduced to one that is, the query

```
? holdAt(child_of(X,Y),T)
```

should be replaced with

```
? holdAt(child_of(X,Y,ART2),T),
? holdAt(active(ART2),T).
```

The article defining '`child`' is article 2.1. of ICL. Thus, we can set the constraint '`ART2=art2.1`'. The new query will be:

```
? TYPE=filiation,
? ART=art1.1.a,
? holdAt(cit(Y,italy),T),
? holdAt(child_of(X,Y,ART2),T),
? holdAt(active(ART2),T).
? ART2=art2.1.
```

After setting '`F=child_of(X,Y,art1.2)`', remove the query

```
? holdAt(child_of(X,Y, ART2),T),
```

and replace it with the suitable antecedents of axiom 3, thus having:

```
? TYPE=filiation,
? ART=art1.1.a,
? holdAt(cit(Y,italy),T),
? holdAt(active(ART2),T),
? ART2=art2.1,
? F=child_of(X,Y,art1.2),
? happen(E,T1),
? initiate(E,F,T1),
? T1<T,
? neg(clipped(T1,F,T)).
```

Now article 2.1. of ICL defines the event '`become_child`', as follows:

```
happen(become_child_of(X,Y,art2.1),T1):-
     happen(recognize(Y,child_of(X,Y)),T1),
     hold_at(minor(X),T1).
```

So we can set '`E=become_child_of(X,Y,art1.2)`' and replace the query

```
? happen(E,T1)
```

with the query:

```
 ? happen(recognize(Y,child_of(X, Y)),T1),
 ? hold_at(minor(X),T1).
```

The new query will be:

```
? TYPE=filiation,
? ART=art1.1.a,
? holdAt(cit(Y,italy),T),
? holdAt(active(ART2),T),
? ART2=art2.1,
? F=child_of(X,Y,art1.2),
? T1<T,
? neg(clipped(T1,F,T)),
? E=become_child_of(X,Y,art1.2),
? happen(recognize(Y,child_of(X,Y)),T1),
? hold_at(minor(X),T1).
```

The negated formula is true by negation as failure and the related query can be removed. The queries:

```
? holdAt(cit(Y,italy),T),
? happen(recognize(Y,child_of(X,Y)),T1),
```

can be removed by ($\mathsf{F4}^{\mathsf{f}}$) and ($\mathsf{F2}^{\mathsf{f}}$), and by setting '`Y=mother`', '`T>1906`' and '`T1>1942`'. The updated query looks as follows:

```
? TYPE=filiation,
? ART=art1.1.a,
? holdAt(active(ART2),T),
? ART2=art2.1,
? F=child_of(X,Y,art1.2),
? T1<T,
? E=become_child_of(X,Y,art1.2),
? hold_at(minor(X),T1),
? Y=mother,
? T1>1942,
? T>1906.
```

The query '`?hold_at(minor(X),T1)`' is problematic. Unfortunately, there is no information about X being a minor at '`T1`', the time of his recognition as a child. Strictly speaking, this should cause the query (Q2) to fail. However, a charitable interpretation of ($F2^f$) (see the statement in natural language) suggests that X was (almost) immediately recognized by his parents when he was born. Thus, it can be safely assumed that '`holdAt(minor(X),T1)`' obtains.

It is time now to consider the second and third line of (Q2), which were put aside at the beginning of the proof:

```
? holdAt(active(ART),T),
? T=1999.
```

The application of axiom 3 and article 27 of ICL:

```
initiate(start_active(icl,art27),active(icl,art27),T).
happen(start_active(icl,art 27),T):-
      T=1992.
```

makes the above query succeed, by '`ART=icl`' (where `icl` includes also any article of ICL). By the same token, the query '`?holdAt(active(ART2),T)`' succeeds. Thus, the final query is:

```
? TYPE=filiation,
? ART=art1.1.a,
? ART2=art2.1,
? F=child_of(X,Y,art1.2),
? T1<T,
? E=become_child_of(X,Y,art1.2),
? Y=mother,
? T1>1942,
? T>1906,
? T=1999.
```

This query is satisfiable, whence (Q2) succeeds. □

The first, obvious, question that arises is this: why was there a trial if a rather straightforward derivation mechanism could have delivered just the same conclusion? Recall that the main point in the court's final judgment was that the date of birth is not the only time citizenship can be transmitted, whereas both parties convened for the trial thought it was. In the proof of claim 2, the crucial step was the first one, in which axiom 6 and the following rule (call it ($L1^s$); 's' for static) was applied:

```
% L1^s
obtain(cit(X,italy,birth,art1.1.a),T):-
          holdAt(child_of(X,Y),T),
          holdAt(cit(Y,italy),T),
```

In accordance with the wording of article 1.1.a of ICL, $(\mathsf{L1}^{\mathsf{f}})$ does not contain any reference to the event of birth. By contrast, suppose a different formalization (call it $(\mathsf{L1}^{\mathsf{d}})$; 'd' for dynamic), were used:

```
% L1^d
    initiate(start_cit(X,italy,birth,art1.1.a),
                cit(X,italy,birth,art1.1.a),
                T).
    happen(start_cit(X,italy,birth,art1.1.a),T):-
          happen(born(X),T),
          holdAt(child_of(X,Y),T),
          holdAt(cit(Y,italy),T).
```

$(\mathsf{L1}^{\mathsf{d}})$ contains a reference to the event of birth. Rules $(\mathsf{L1}^{\mathsf{s}})$ and $(\mathsf{L1}^{\mathsf{d}})$ reflect the two different interpretations alluded by the Supreme Court. Let's read the passage one more time:

> On this matter, it should be made precise that ... the right to acquire Italian citizenship comes into being not at the event of the "birth" ..., but rather it is based on the situation of filiation from a parent who is a citizen.[15]

The contrast between a *dynamic interpretation* based on "the event of the birth" and a *static interpretation*[16] based on the "situation of filiation" is parallel by $(\mathsf{L1}^{\mathsf{d}})$ vs. $(\mathsf{L1}^{\mathsf{s}})$. However, the parallelism does not go as far as we would expect: it seems that, in the intention of the court, accepting the dynamic interpretation should yield the failure of query $(\mathsf{Q2})$, provided bounded retractive validity is granted, namely principle $(\mathsf{L3})$. Unfortunately, this is not the case from a formal point of view:

CLAIM **3.** First, assume the agreed principles, $(\mathsf{F1}^{\mathsf{f}})$–$(\mathsf{F4}^{\mathsf{f}})$ and $(\mathsf{L4}^{\mathsf{f}})$–$(\mathsf{L5}^{\mathsf{f}})$. Second, assume $(\mathsf{L1}^{\mathsf{d}})$. Then, query $(\mathsf{Q2})$ succeeds.

Proof. I will only give a very rough sketch of the proof, and leave the details to the reader. Now, X was born in 1942, so the application of $(\mathsf{L1}^{\mathsf{d}})$ (along with the agreed facts and principles) makes true the following:

```
happen(start_cit(X,italy,birth,art1.1.a),1942).
```

The above initiates the fluent:

```
cit(X,italy,birth,art1.1.a).
```

There are no interfering events on the way, so up until 1999 (date of the trial) the fluent still holds, by axiom 3. Furthermore, in 1999 the law containing article 1.1.a is active, so the query:

[15] Deve essere precisato, in proposito, che il titolo di siffatto modo di acquisto ... della cittadinanza italiana è costituito, non già dall'evento "nascita" ..., bensì dalla situazione di filiazione da genitore cittadino. See [10].

[16] Recall the discussion between definitional and event-based view in section 3.1.

```
? holdAt(active(art1.1.a),1999)
```

succeeds, whence (Q2) succeeds. □

The proof reveals how things developed for individual X, under our formalization. The fluent

```
cit(X,italy,birth,art1.1.a)
```

is extended from 1942 onwards. However, the fluent has lacked legal force for a while, because article 1.1.a was activated, by principle (L3), only from January 1948 (the date of enactment of the Constitution). This means that fluent '`cit`' went through a long period of legal inactivity, from which it was "liberated" and made active on January 1948. This picture is problematic because the event initiating '`cit`' was not legally active in 1942, and so it should not be possible for a fluent to become legally active if it was initiated by a legally inactive event.

The reader may have noticed that we face a variant of the problem described in section 3.2, on page 24, while dealing with the distinction between validity and classificatory reasoning. To repeat, the problem is as follows: If a statement of the form '*ι is a citizen according to article n*' is the result of a legal classification consisting of a proof where several articles are utilized; then, while performing validity reasoning one has to check the validity of all the articles utilized during the proof of '*ι is a citizen according to article n*'. The difficulty is that it is unclear how to keep track of the articles utilized during proofs. The variant of the problem here discussed adds a further element—it is not enough to check the validity of the articles used during proofs, one also needs to know with respect to which moments in time the validity of these article needs to be checked. In the example here at hand, article 1, clause 1, letter a) of ICL is used during the proof of claim 3, as follows:

```
? happen(start_cit(X,italy,birth,art1.1.a),1942).
```

At the end of the proof of claim 3, that very same article is checked for its validity at '`T=1999`'; and the validity checking yields a positive outcome. This is not the validity checking that we need: the validity of article 1, clause 1, letter a) should have been checked at time '`T=1942`', not '`T=1999`'. How the problem, and variants thereof, can be tackled remains an open question.

At any rate, suppose the technical problem of keeping track of the articles utilized during proofs is solved; then, (Q2) will not succeed under the interpretation ($\mathsf{L1}^{\mathsf{d}}$). This brings me to the question *why* ($\mathsf{L1}^{\mathsf{d}}$) should be preferred over ($\mathsf{L1}^{\mathsf{s}}$), or viceversa. The choice between the two should appeal to extra-principles which are not contained in the text of the law itself, and so the dispute over ($\mathsf{L1}^{\mathsf{d}}$) or ($\mathsf{L1}^{\mathsf{s}}$) cannot be solved by means of the sole formalization. Note, however, that the formalization given in the previous chapter of article 1.1.a used the interpretation ($\mathsf{L1}^{\mathsf{s}}$), which happened to be the same interpretation adopted by the court. Nonetheless, other courts may prefer the interpretation ($\mathsf{L1}^{\mathsf{d}}$), so

that ($\mathsf{L1}^{\mathsf{s}}$) is not the definitive word on the issue.[17] The conclusion that can be drawn from this is that a good formalism should be able to accomodate both interpretations, and leave it open to the interpreters which one to select (see the conclusion for a discussion).

Summary

This chapter attempts to apply the formalization of ICL to a concrete case that has been disputed before a court of law. The concrete case is described in the first part of the chapter, section 5.1, and subsequently the formal apparatus is introduced, section 5.2. By using the axioms of EC, formalizations of articles from ICL, facts that are specific to the case at hand, but also (sometimes conflicting) interpretative principles such as ($\mathsf{L2}^{\mathsf{f}}$) and ($\mathsf{L3}^{\mathsf{f}}$)—it was possible to answer simple queries about retroactivity (see claim 1), and X's Italian citizenship (see claim 2).

The axioms of EC are extended to deal with retroactivity, as follows:

Ax7: `holdAt(F,T1)` $\leftarrow$ `happen(E,T2)` $\wedge$ `retroinitiate(E,F,T2)` $\wedge$ `T1<T2` $\wedge$ $\neg$`clipped(T1,F,T2)`.

A recurrent theme of the chapter—which did not receive a systematic treatment, however—is the issue of how the law is interpreted (see for example the opposition between dynamic and static interpretations: ($\mathsf{L1}^{\mathsf{s}}$) vs. ($\mathsf{L1}^{\mathsf{d}}$)). The moral is that, in most cases, the text of the law, or a formalization thereof, is insufficient to decide about a problematic case. The text of the law is not self-contained. A formalism is thus needed which can handle the interaction between the formalization of a piece of law and its possibly conflicting interpretations.

[17] In fact if one consults the web-site of the Italian Consulate in Argentina, under the heading '*citizenship office*', one will find that the interpretation adopted by the consulate is ($\mathsf{L1}^{\mathsf{d}}$). On the web site, it is written: *children born before January 1, 1948, are Italian citizen only if they are born by Italian father, given that women can transmit Italian citizenship only beginning January 1, 1948.*

6 Conclusion

The reader should be convinced by now that EC can formalize non-trivial parts of legislative texts, while preserving elementary requirements for a good modeling, such as:
(R1) sanctioning the right or intended inferences;
(R2) precision and adherence to the text (sometimes termed "isomorphism");
(R3) maintainability.

Chapters 3 and 4 have shown how legislative texts can be formalized in EC—the former in a general and abstract way, and the latter by looking at a case-study, the Italian Citizenship Law (ICL). During the formalization, it was suggested to augment EC with two more axioms. They are needed to cope with definitional statements in the law (see section 3.1) and retroactivity (see section 5.2):

Ax6: `holdAt(F,T2)` $\leftarrow$ `T1<T2` $\wedge$ `obtain(F,T1)` $\wedge$ $\neg$`clipped(T1,F,T2)`.

Ax7: `holdAt(F,T1)` $\leftarrow$ `happen(E,T2)` $\wedge$ `retroinitiate(E,F,T2)` $\wedge$ `T1<T2` $\wedge$ $\neg$`clipped(T1,F,T2)`.

Chapter 5 can be seen as an attempt to model the process of applying and interpreting the law, by examining a case which involved the use of the ICL. The process of interpretation was roughly described as the addition or removal of certain interpretative principles, such as ($\mathsf{L2}^{\mathsf{f}}$) or ($\mathsf{L3}^{\mathsf{f}}$), or as the choice between two competing formal renderings of the same piece of legal text, as in the case of ($\mathsf{L1}^{\mathsf{s}}$) and ($\mathsf{L1}^{\mathsf{d}}$).

Future Work

One direction of future work concerns the *interpretation* of legal texts. In the last chapter an example of competing interpretations of the same piece of law is given, i.e., ($\mathsf{L1}^{\mathsf{s}}$) and ($\mathsf{L1}^{\mathsf{d}}$), the static and dynamic interpretation of article 1 of ICL. In our set-up, they were two opposite interpretations, so that the formalization could only contain one of them. To account for the interpretative process, the formalization should instead be open to both ($\mathsf{L1}^{\mathsf{s}}$) and ($\mathsf{L1}^{\mathsf{d}}$).

When the law is interpreted, the text of the law does not change. So, a desideratum for a theory of interpretation is that the formalization remains the same and interpretative principles (also in the form of integrity constraints) are added subsequently on a higher layer to constrain the formalization in the

intended way. Interpretative principles are retractable, while the formalization of the law should not be so (unless it is modified by the legislator). This requires that the underlying formalization be flexible enough and open to conflicting interpretations.

To this proposed direction of work, one may object that intending the formalization as open to interpretation will defeat the purpose of the formalization itself, as formalizing means *disambiguating*. A proper reply would require an in-depth discussion of the purposes and ranges of application of formalizations of the law. If one of the purposes is the understanding of how content and meaning develop in legislative texts, the issue of interpretation cannot be escaped. On the other hand, if the purpose of the formalization is to make the law fully precise and unambiguous, a theory of interpretation is certainly not needed. I hold that it is not feasible to aim at disambiguating the law completely, without dooming the formalization process to biases from a particular interpretation. Moreover, it follows by no means that formalizing coincides with *fully* disambiguating. In fact, there is formal work about under-specification in syntax and semantics (see for example Minimal Recursion Semantics [6, 7] to deal with scope ambiguity), and hence formal renderings, in principle, can be open to interpretation.

There are two more directions of future work I shall propose. On the technical side, an *implementation* is the most needed one. Proofs given in this thesis were straightforward and should be fully computational. The implementation should be paired with an *editor* which can automatically generate Prolog-like code for the implementation. Writing the formulas by hand is a tedious task, often repetitive and subject to human errors. An editor to assist the process of formalization will enable one to formalize larger pieces of legislation in the EC. In fact, the formalization of the law becomes interesting only when a large amount of legislation is formalized.

Other pieces of legislation to be formalized are those which are connected with ICL. The Italian citizenship law is not self-sufficient in that, for example, it does not specify which documents one should submit and which procedures one should follow to be granted Italian citizenship. These related pieces of law are mainly regulative decrees, precisely two decrees of the President of Republic, n. 572 on October, 12, 1993, and n. 362, on April 4, 1994. Formalizing these other pieces of law will contribute to testing the maintainability of the present formalization and, in addition, shape a more precise and full-fledged methodology to formalize large scale legal texts with EC.

Bibliography

[1] Legge 13 giugno 1912, n. 555 sulla cittadinanza italiana.

[2] Legge 5 febbraio 1992, n. 91. nuove norma sulla cittadinanza. In *Gazzetta Ufficiale, n. 38, 15 febbraio 1992.*

[3] T. J. M. Bench-Capon, G. O. Robinson, T. W. Routen, and M. J. Sergot. Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation. In *ICAIL '87: Proceedings of the 1st international conference on Artificial intelligence and law*, pages 190–198, New York, NY, USA, 1987. ACM Press.

[4] Joost Breuker, Rinke Hoekstra, Alexander Boer, Kasper van den Berg, Rossella Rubino, Giovanni Sartor, Monica Palmirani, Adam Wyner, and Trevor Bench-Capon. OWL ontology of basic legal concepts (LKIF-Core). Deliverable 1.4, Estrella, 2007.

[5] Roderick M. Chisholm. Contrary-to-duty imperatives and deontic logic. *Analysis*, 24(2):33–36, 1963.

[6] Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. Translation using Minimal Recursion Semantics. In *Proceedings of the 6th. International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95)*, Leuven, Belgium, July 1995.

[7] Ann Copestake, Daniel P. Flickinger, and Ivan A. Sag. Minimal recursion semantics: An introduction. Manuscript, Stanford University: CSLI, 1999.

[8] Corte Costituzionale. 9 febbraio 1983. sentenza n. 30.

[9] Robert Demolombe and Maria del Pilar Pozos Parra. The chisholm paradox and the situation calculus. In *ISMIS*, pages 425–434, 2005.

[10] Corte di Cassazione. 10 luglio 1996. sentenza n. 6297.

[11] Tribunale di Torino. Sentenza n. 12, aprile 1999.

[12] Thomas F. Gordon. Constructing arguments with a computational model of an argumentation scheme for legal rules (draft). In *ICAIL '07: Proceedings of the 11th International Conference on Artificial Intelligence and Law (to appear)*. ACM Press, 2007.

[13] Rinke Hoekstra, Joost Breuker, Marcello Di Bello, and Alexander Boer. The LKIF Core ontology of basic legal concepts. In Pompeu Casanovas, Maria Angela Biasiotti, Enrico Francesconi, and Maria Teresa Sagri, editors, *Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT 2007)*, June 2007.

[14] Robert Kowalski. The treatment of negation in logic programs for representing legislation. In E. Rissland, editor, *Proceedings of Second International Conference on AI and Law*, pages 11–15, Vancouver, Canada, 1989. ACM Press.

[15] Robert Kowalski. Database updates in the event calculus. *Journal of Logic Programming*, 12(1-2):121–146, 1992.

[16] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[17] Robert Kowalski and Francesca Toni. Abstract argumentation. *Artificial Intelligence & Law*, 4(3-4):275–296, 1996.

[18] Robert A. Kowalski. Legislation as logic programs. In *LPSS '92: Proceedings of the Second International Logic Programming Summer School on Logic Programming in Action*, pages 203–230, London, UK, 1992. Springer-Verlag.

[19] P. Leith. Fundamental errors in legal logic programming. *The Computer Journal*, 29(6):545–552, 1985.

[20] Rafaél Hernández Marín and Giovanni Sartor. Time and norms: a formalisation in the event-calculus. In *ICAIL '99: Proceedings of the 7th international conference on Artificial intelligence and law*, pages 90–99, New York, NY, USA, 1999. ACM Press.

[21] Henry Prakken and Marek Sergot. Contrary-to-duty obligations. *Studia Logica*, 57(1):91–115, 1996.

[22] Raymond Reiter. The frame problem in situation the calculus: a simple solution (sometimes) and a completeness result for goal regression. In *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, pages 359–380, San Diego, CA, USA, 1991. Academic Press Professional, Inc.

[23] M. Sergot, F. Sadri, R. Kowalski, F. Kriwaczek, P. Hammond, and T. Cory. The british nationality act as a logic program. *CACM*, 29(5):370–386, 1986.

[24] Murray Shanahan. The event calculus explained. In M.J.Wooldridge and M.Veloso, editors, *Artificial Intelligence Today*, pages 409–430. Springer, 1999.

[25] Murray Shanahan. An attempt to formalise a non-trivial benchmark problem in common sense reasoning. *Artificial Intelligence*, 153(1-2):141–165, 2004.

[26] K. Stenning and M. van Lambalgen. *Human reasoning and cognitive science (to appear)*. MIT Press, 2007.

[27] André Valente. *Legal Knowledge Management: An Engineering Approach*. IOS Press, 1995.

[28] Michiel van Lambalgen and Fritz Hamm. *The Proper Treatment of Events*. Blackwell, 2005.