Modelling Legal Acts by Means of Expert Systems, ECA Rules and High-level Petri Nets

Boštjan Berčič
Institute for Legal Informatics, 1000 Ljubljana, Slovenia
E-mail: bostjan.bercic@ipri-zavod.si

Keywords: expert systems, petri nets, legal informatics

Received: May 7, 2002

Modelling of legal acts often felt short of expectations because it didn't take into account legal theory. This paper proposes a different approach to modelling that is based on the theory of law. Legal theory (structure, hierarchy and types of legal rules) is considered a fundament and then interpreted with expert systems, high-level Petri nets and ECA rules. In particular, none of these methods alone is sufficiently strong to capture the semantics of legal rules. Put together, they represent a powerful means to overcome some difficulties legal modelers have encountered in the past. This paper takes into account procedural and substantive aspects of law, as well as factual and deontic ones. Methodology is presented alongside with notation and examples to clarify the idea.

1 Introduction

Modelers of legal procedures in the past often came from the field of informatics and had little knowledge of legal theory. They applied same techniques and methodologies to legal procedures as they would to business (or indeed any other) processes. Whereas this might work in certain cases, it certainly comes at a price: legal procedures are *sui generis* and not every legal rule can be expressed with business process methodology. There are complexities and peculiarities inherent to legal matter which can hardly be modelled by means of business processes.

For one thing, legal norms contain normative (deontic) and factual elements. Most modelers in the past focused their work on factual elements only. They made little or no reference to normative elements such as rights and duties.

Second, their approach was directed either at substantive or procedural law, but rarely at both. Substantive law defines contents of legal subject's rights and duties whereas procedural law defines procedures in which these rights and duties can be enforced in case legal subjects do not act in conformance with them.

Modelers dealt in the past mostly with procedural law, which can be modelled with state machines, workflow methods etc. Very often, one of these techniques was used to represent procedural part of legal rules as one can in general apply knowledge of (any kind of) processes to legal procedures.

On the other hand, modelers have often encountered problems with substantive legal norms. Technology of choice for substantive legal rules are expert systems. There are many legal expert systems shells available (such as Wysh, ICaR etc.) but general purpose expert system shells can be used as well. Legal rules are written in knowledge base, usually in form of PROLOG rules.

Some of these approaches are summarized bellow. They have been partial at least in some respect because none of them joined all (legal) elements under one hood.

Holt and Meldman (Meldman J.A., Holt A.W. (1971), Meldman J.A. (1978)) used Petri nets to model Federal Rules of Civil Procedure. They modelled only procedural rules and took into account only factual elements.

Sergot et al.(Sergot M.J., Sadri F., Kowalski R.A., Kriwaczek F., Hammond P., Cory H.T. (1986)) used PROLOG rules to express British Nationality Act (rules governing acquirement of British citizenship). These are substantive rules, but no mention is made of deontic elements (for example, what does it *mean* to be a British citizen in terms of rights and duties).

Lee et al.(Lee R.M., Bons R.W.H., Wrigley C.D., Wagenaar R. W. (1995)) used special kind of high-level Petri nets, called documentary Petri nets, to express procedural rules. Documentary Petri nets are high-level Petri nets augmented with deontic operators expressing obligations (duties) and rights (permissions) and special documentary places holding documents. Thus, they came a step closer in embedding factual and normative (deontic) contents within procedural framework. Separately, Lee et al. (Lee R. M., Ryu Y.U. (1994)) also considered deontic expert systems, with deontic operators augmented first-order logic (PROLOG knowledge base), but they have not integrated it in a single framework.

Burg and Van de Riet (Burg J.F.M., Van de Riet R.P. (1994)) devised a modelling technique called COLOR X. It can model procedural aspects of (deontic) rules with use of linguistics. They went a step further in introducing linguistic elements to modelling. Now, a fact is not merely a fact but a sentence in natural language which can be stripped down to words. Meaning of a word can be fully interpreted as every word can have its own existence. This technique is object-oriented and could form basis for legal ontology building if extended a little bit further. Technique prescribes dynamic (CEM) as well as static (CSOM) model of events and objects. This technique can model not just legal domains but, because of its root in linguistics, any domain expressed in natural language. So, COLOR X can represent factual as well as deontic elements, but it focuses on procedural aspects. Substantive knowledge is left out.

This paper proposes a different approach. Our models are based on legal theory. First, legal structures (legal acts, legal rules, hierarchy of legal rules and acts etc.) are taken into account and interpreted with different information technologies and notations. High-level Petri nets are used to represent procedural aspects of legal rules. Expert system knowledge base is used to represent substantive legal rules. Factual elements of legal rules are mapped to Petri net transitions and deontic elements to Petri net places. Petri net transitions are extended with ECA rules in order to be able to express factual and deontic pre- and post-conditions.

2 Legal Rules

2.1 Structure of Legal Rules

Legal rules are composed of three (optionally four) components: the norm addressee (norm subject), deontic modality, object of a norm (contents) and optionally norm conditions (Kralingen van R. (1997),Breuker J.,Valente A.,Winkels R. (1997),Visser P.R.S.,Bench-Capon J.M. (1997)).

Norm addressee is a subject addressed by the norm. Usually certain act or behavior is required from him.

There are two principle deontic modalities: obligation O and permission P which correspond to legal duties and legal rights. Every norm either prescribes or permits some behavior.

Object or theme of a norm is an act or behavior required from or allowed to norm subjects. Object of a norm is norm's contents. It prescribes what is allowed to do and what should be done.

Condition of a norm is norm's hypothesis. It describes state of affairs which must be satisfied in order for the norm to apply. Some norms have conditional part, some don't. Those that don't are unconditional. Conditional norms turn into unconditional ones once the condition is fulfilled.

We can write this succintly as:

$$F_1(X) \Rightarrow P(F_2(X)) \tag{1}$$

where:

- X is norm subject
- $F_1(X)$ is norm condition
- $P(\ldots)$ is deontic modality and
- $F_2(X)$ is object of a norm

If we take a closer look, we discover that there is a difference between norm's condition and norm's consequence. Norm's condition is *always* some state of affairs (legally relevant state of the world) and norm's consequence is *always* some deontic modality (defined on yet another, prescribed state of affairs), either obligation or permission.

2.2 States of Affairs

States of affairs are important elements of legal rules. They define legally relevant states of the world, i.e. states of the world that are of interest to legal order.

Legal rules feature states of affairs in both conditional and prescriptive part. States of affairs describe under which circumstances legal rules are applicable (rule condition) and what behavior is required or allowed. We can find them both in

$$F_1(X) \tag{2}$$

and in

$$P(F_2(X)) \tag{3}$$

States of affairs are often subject to different methods of legal exegesis.

2.3 Rights and Duties

Rights and duties (permissions and obligations) form prescriptive part of legal rules. They are deontic modalities and they cannot stand alone. They come in two flavors: O and P and they are *always* defined on some state of affairs.

$$O(F_1(X)) \tag{4}$$
$$P(F_2(X)) \tag{5}$$

In the first case (equation 4) person X is required to bring about state of affairs F_1.

In the second case (equation 5) person X is allowed to bring about state of affairs F_1.

Rights and duties can be of two types: *ought-to-do* and *ought-to-be* (*tun-sollen* and *sein-sollen*). Ought-to-do operators have subjects that they address whereas ought-to-be don't. This paper will deal only with first type operators.

2.4 Formal notation

So far, all components of legal rules have been formally defined . But rules are often interconnected. If legal rule is not obeyed there is usually another one that specifies what legal order should do in order to preserve rule conformity. This rule is called sanction.

This can be written as:

$$hypothesis \Rightarrow duty \tag{6}$$
$$secundary\ hypothesis \Rightarrow secundary\ duty \tag{7}$$

or, in common terms as:

$$hypothesis \Rightarrow duty \tag{8}$$
$$duty \wedge violation\ of\ duty \Rightarrow sanction \tag{9}$$

On general, if one denotes state of affairs with F(factual) and normative contents with N, one gets the following structure of legal rules:

$$\begin{aligned} F_1 &\Rightarrow N_1 \\ N_1 \wedge F_2 &\Rightarrow N_2 \\ N_2 \wedge F_3 &\Rightarrow N_3 \end{aligned} \tag{10}$$

Keyword here is structure. Legal rules don't stand alone. They are intertwined with each other in legal order. One rule's consequence may be another one's condition. If all rules from one legal system are put together in this manner, we obtain *legal order* in force.

Notice here the strict alternation of factual and deontic elements. According to legal theory, legal rules can have factual antecedent and factual consequence (e.g. legal definition) or even deontic antecedent and deontic consequence (another legal definition, e.g.: having one right means that you have another one, too). But legal rules can never have deontic antecedent and factual consequence (what should be is not the case by mere fact that it should be).

If we use our previous notation, we can now write legal order as:

$$\begin{aligned} F_1(X) &\Rightarrow P(F_2(X)) \\ P(F_2(X)) \wedge F_2(X) &\Rightarrow O(F_3(X)) \\ O(F_3(X)) \wedge \neg F_3(X) &\Rightarrow O(F_4(X)) \\ &\vdots \end{aligned} \tag{11}$$

2.5 Types of Legal Rules

2.5.1 Substantive vs. Procedural Rules

Substantive rules express contents of rights and duties that address legal subjects. They prescribe how legal subjects are to act in order to achieve desired results and what different states of affairs imply in terms of normative consequences.

Procedural rules, on the other hand, express procedural aspects of law enforcement. They prescribe procedures in which substantive rules can be enforced. Procedural rules govern legal processes such as criminal procedure and civil procedure which seek to remedy breaches (civil law) and crimes (penal law) committed by subjects of legal norms.

2.5.2 Legal Rules vs. Legal Definitions

Legal rules prescribe behavior that is required from norm addressees. Legal rules have conditional part and prescribed part. Conditional part contains description of states of affairs to which norm applies. Consequential part prescribes deontic modality (right,duty) and act, which is to be (duty) or is allowed to be (right) performed by the norm subject. Unconditional legal rules contains only second part. It is of constitutive importance for legal rules to contain normative (deontic) contents.

$$F \Rightarrow N \tag{12}$$

Legal definitions, by contrast, do not prescribe any behavior but rather define some legal term. Legal definitions do not contain hypothetical (conditional) part which would describe when to apply the norm, nor do they contain normative elements which would say what behavior is required. Rather, legal definitions apply to legal terms themselves (not states of affairs) and further clarify their meaning (e.g. somebody falling under provisions of certain law means that . . .).

$$F \Leftrightarrow F_1 \wedge \ldots F_n \tag{13}$$

Legal definitions can apply to legal terms which contain states of affairs and legal terms which contain deontic modalities.

2.5.3 Rules of Conduct vs. Rules of Competence

There is an important difference between rules of conduct, which describe what is permitted to do or must be done by the norm addressees in terms of factual behavior and rules of competence which describe what powers (liabilities) norm subjects have in respect to creating new legal rules.

Most rules are rules of conduct. They prescribe required behavior from norm subjects.

Rules of competence, on the other hand, empower or oblige norm subjects to create new legal rules. A legal subject can be empowered to create new normative contents (rights and duties, e.g. legislator who passes laws or contractor who creates new obligations). Legal subject can also be obliged to create new rules (e.g. if you apply for citizenship, under some conditions, public administration office has to issue it)(Allen L.E. (1997)).

2.5.4 Monotonic vs. Nonmonotonic Rules

Monotonicity is defined with respect to temporal aspects of legal rules.

Monotonic rules are those that hold regardless of the event (events) that triggered them (e.g. once somebody is dead, his heirs are entitled to heritage).

Nonmonotonic rules are those, whose validity constantly depends on validity of the triggering conditions (e.g. when you get sick you can apply for remedies from your health care insurance policy, but you have no right to do so while healthy).

Monotonicity of rules has to do with repeatability of norm's conditional part. If states of affairs that describe conditional part can vary over time, then we have nonmonotonic rule. If, on contrary, it is a one-time event, then we have monotonic rule.

The same applies for truth values of legal definitions. Nonmonotonic conditions can hold over intervals of time and change (e.g. marital status can change several times in one's lifetime).

Some procedural rules are typically monotonic while some substantive rules are typically nonomonotonic.

3 Expert Systems

Expert systems contain expert system shell and knowledge base. Knowledge base is used as an input to inference mechanism, which resides in expert system shell and infers consequences. Knowledge base can be expressed in PROLOG in form of Horn clauses (clauses with implicit existential quantifier)(Bratko I. (2001)):

$$\Phi : -\Psi_1, \ldots, \Psi_n \tag{14}$$

$$\Phi : -\Psi_1; \ldots; \Psi_n \tag{15}$$

These PROLOG clauses can be conveniently expressed in more familiar form:

$$\Phi \Leftarrow \Psi_1 \wedge \ldots \wedge \Psi_n \tag{16}$$

$$\Phi \Leftarrow \Psi_1 \vee \ldots \vee \Psi_n \tag{17}$$

Expert systems lend themselves to express nonmonotonic rules (rules whose truth values change over time because of changing states of affairs). Expert system can be interrogated by user in every moment about truth values of consequences of rules in knowledge base. If rule's conditions are satisfied, system infers its consequences.

Expert systems have been mostly used in substantitve law. They can be used in legal definitions:

$$F \Leftrightarrow F_1 \wedge \cdots F_n \tag{18}$$

where:

- F is complex state of affairs
- $F_1, \ldots, F_n$ are elementary states of affairs

and in legal rules as well:

$$N \Leftarrow F_1 \wedge \cdots F_n \tag{19}$$

where:

- N is norm object (of normative type)
- $F_1, \ldots, F_n$ are norm conditions (of factual type)

For the most part, this paper will refer to expert systems' knowledge base for expressing factual elements of legal rules (states of affairs). Although deontic elements could be expressed as well (and sometimes they will be), it is probably better to have a clear delineation on the level of implementation between states of affairs and their deontic consequences. Deontic consequences will be expressed by means of places in Petri Nets.

4 ECA Rules

ECA rules have general form (Dittrich K.R., Gatziu S. (1993)):

on <event>
if <condition>
do <action>

ECA rules stem from active database community and are today most widely used in active database research and event-driven programming.

Event detector detects events, checks whether conditions hold and if they do, fires corresponding action.

Events in ECA rules can be of simple and composite types; composite types can be, for example, expressed with rules in knowledge base. The same holds for conditions which are typically expressed as boolean combination of simple conditions read from some database (or knowledge base). ECA rules can thus be easily integrated with expert system's knowledge base.

ECA rules will be used to express various elements of states of affairs. States of affairs typically contain either some state or event or both. Event part of ECA rules will be used to specify events of some complex state of affairs and condition part of ECA rules will be used to express some static state of affairs. Example will clarify this:

If somebody wants to enter into a contract, he must perform some action (e.g. sign a contract). But in order for the contract to be valid, contactor must have competence to enter into it. Competence to sign a contract is static part and signing a contract active part of the state of affairs which must be fulfilled in order for the contract to be valid. Thus, we map signing a contract into event part and legal competence into condition part of ECA rule. Both should be there if the event is to trigger some consequences. Note that this paragraph doesn't deal with deontic consequences of these acts, which is left for later when Petri nets are considered).

Finally, the action part (which is optional) may express some change in the state of affairs (for example, if you marry someone, your marital status changes to married).

4.1 Events

Events can be primitive or composite events. Composite events are made up of primitive ones with the use of operators (disjunction, sequence,conjunction,periodicity) of event algebra (Gatziu S. (1993),Chakravarthy S., Mishra

D. (1991)) . Event algebra itself could be specified with Petri nets or with rules in knowledge base (boolean operators plus attribute for time). This paper implements it with knowledge base in order to have expert systems cover all factual elements.

4.2 Conditions

Conditions express conditions which must hold in order for the event to fire. Conditions can be any combination of first-order predicate logic statements. They are implemented with expert systems, as well.

4.3 Actions

Actions are an optional part of an ECA rule in our interpretation. They can express an update or change in the state of affairs. Actions always refer to state of affairs (matters of fact), never to deontic modalities (rights and duties). Petri net markings take care of the latter.

5 High-level Petri Nets

A HLP-net is a structure $HLPN = (P; T; CT; C; Pre; Post; M_0)$ (Billington J. (1997)) where:

- P is a finite set of elements called Places,
- T is a finite set of elements called Transitions, which are disjoint from P ($P \cap T = \emptyset$),
- $CT = \{N, F\}$ is a non-empty finite set of types (of places and transitions), where N denotes normative type and F denotes factual type
- C $C : P \cup T \rightarrow CT$ is a function used to type places and determine transition modes, such that $C(P) = N, C(T) = F$,
- Pre is a pre mapping $Pre_{(p,t)} : C(t) \rightarrow N^{|C(p)|}$,
- Post ia s post mapping $Post_{(p,t)} : C(t) \rightarrow N^{|C(p)|}$,
- M_0 is an initial marking of the net.

HLPN lend themselves to express procedural law.

They can be data(place) or transition driven. Data driven Petri nets fire transition whenever its preconditions are met (all input places contain tokens). Event driven Petri nets fire transition whenever its preconditions are met *and* event associated with it has occured.

Event driven Petri nets will be used in this paper because states of affairs will be mapped to events in Petri net transitions.

Petri nets are most appropriate for procedural rules (which contain implicit timeline). Sometimes they can be used for substantive rules as well.

They are also appropriate for monotonic rules, i.e. rules which don't change their truth values over time. Their truth space is monotonically increasing (once you file a complaint it will always be that you have filed it)

Petri net places and transitions are interpreted as follows.

5.1 Places

Places hold deontic contents (rights, duties). They are the only elements in whole structure that do so. Rights and duties are not expressed within knowledge base in this paper. Rather, they are all gathered at one place (consider implementation issues).

5.2 Transitions

Transitions hold events and are always of factual nature. They contain events that change rights and duties. In accordance with Petri net semantics, whenever event fires and its preconditions hold, postcondition hold after the event and preconditions stop to hold (e.g. if you have certain right and choose to exercise it, you lose that right (precondition) and possibly get another one (postcondition).

Transition events can be simple, composite or empty. Composite events are made up of simple events. Sometimes transitions can synchronize: if one fires, the others fire, too. This happens if one event is mapped onto many transitions.

Transitions in HLPN can fire in different modes. Some priority function must be defined over modes, just as some priority function must be defined over ordinary transitions if they happen to be enabled at the same time. We say that they are in conflict.

5.3 Step Semantics

Step semantics determines in detail what effect firing of a transition has. It prescribes detailed sequence of steps taken by the system in order to arrive at desired state (determination of enabled transitions and modes, retrieval of relevant data, updates to net marking,etc.)

Check which transitions are enabled in what modes. For each transition and for each mode do:

1. resolve mode and transition priority
2. check whether transition contains event or null event
   (a) if it contains event, check whether event has happened and if so: fire
   (b) if it contains null event: fire
3. withdraw tokens from appropriate input places
4. put tokens into appropriate output places

6 Petri Nets, Expert Systems and ECA Rules Integrated

Now we can make use of all three components in one integrated structure. HLPN will be taken as a starting point. Then, ECA rules will be mapped onto transitions, making them event/condition/action transitions. Event, condition and action parts of ECA rules will be furthermore mapped into expert system knowledge base rules that will be processed via expert system's shell inference mechanism. This will enable the expression of complex events, conditions and actions.

A HLP-net is a structure $HLPN = (P; T; CT; C; Pre; Post; M_0)$ where

– P is a finite set of elements called Places

– T is a finite set of elements called Transitions disjoint from P ($P \cap T = \emptyset$)

– $CT = \{N, F\}$ is a non-empty finite set of types (of places and transitions), where N denotes normative type and F denotes factual type

– Γ is a mapping $\Gamma : T \rightarrow \{E, Co, A\}$

– C $C : P \cup E \cup Co \cup A \rightarrow CT$ is a function used to type places and transitions such that $C(P) = N, C(E), C(Co), C(A) = F$

– Pre is a pre mapping $Pre_{(p,t)} : C(t) \rightarrow N^{|C(p)|}$

– Post ia s post mapping $Post_{(p,t)} : C(t) \rightarrow N^{|C(p)|}$

– M_0 is an initial marking of the net

– Δ is a mapping $\Delta : E \rightarrow \Phi : -\Phi_1, \ldots, \Phi_n$

– Λ is a mapping $\Lambda : C \rightarrow \Psi : -\Psi_1, \ldots, \Psi_n$

– Σ is a mapping $\Sigma : A \rightarrow \Omega : -\Omega_1, \ldots, \Omega_n$

High-level Petri net defines the global structure of the legal model. It contains places and transition. Places are of two (deontic) types: rights P and duties O. Transitions can fire in different modes and they contain states of affairs. Just as places and transitions strictly alternate in Petri net, so do factual and deontic elements in legal order. States of affairs trigger normative contents (rights and duties). It is never *vice versa*

Transitions represent states of affair. States of affair can be simple or composite. They can also be proper states or events.

This semantics is captured by ECA rules. Events part represent (potentially composite) active components of states of affairs and conditions represent (potentially composite) proper states of affairs. Action part permits rules to issue actions such as update on states of affairs.

Transitions are mapped to ECA rules, which are triples $\{E, Co, A\}$. All members of triple are of factual type. They contain events that trigger the transition, conditions which implement guard function as to when transition is allowed to fire and action which can be set off as a consequence of firing of transition.

Composition of events, conditions and actions in ECA rule is done by means of expert system rules.

Composite event can be described in usual knowledge base manner. Composite event name is head of the knowledge base rule, simple events which constitute composite events are its tail, coupled with corresponding boolean operators.

The same goes for conditions. Conditions are rules in knowledge base. Rule name (condition name) is head of the rule and represents consequence, whereas tail contains simple conditions.

Each time that system has to check whether certain event or condition is fulfilled (for example in order to fire a transition, the system asks user whether this or that has happened), it calls expert system knowledge base. Each ECA rule contains rule head, which is matched against rule head in knowledge base. Head is than expanded with rule body (tail) which in turn contains heads of other rules in knowledge base. This process continues iteratively until list contains only elementary facts that can be matched either against present knowledge base or (in case of absence) required from the user.

After all required elementary facts have been retrieved, rule can be evaluated to be either true or false (alternatively we could have any type of function, not just booleans, which would operate on supplied data). If both event part is true (composite event has happened) and condition part is true (required conditions hold), the transition can fire.

Firing a transition means subtracting tokens from its input places and putting them in its output places. This can be done after the event and when the condition part has been evaluated to be true. Also, at the same time, action part of ECA rule fires.

In legal semantics presented in this paper firing of transition means, that a legal event has happened (e.g. a person has committed an act), with all its normative preconditions (places) present (e.g. right of a person to commit that act) and with all its factual conditions present (e.g. person's competence to commit that act). Consequences are twofold: normative and factual. Normative consequences are new rights and duties which result out of act of a person committing that act (e.g. contractual obligations arise from signing the contract). Sometimes factual consequences arise as well (e.g. date of the contract is set). Normative consequences are represented by tokens in PN places, factual consequences are written as facts in knowledge base.

Thus, this methodology delimits very neatly the normative and factual contents of legal acts. Deontic elements all lie in PN places, while factual elements are all stored in knowledge base and are invoked via ECA rules from PN transitions.

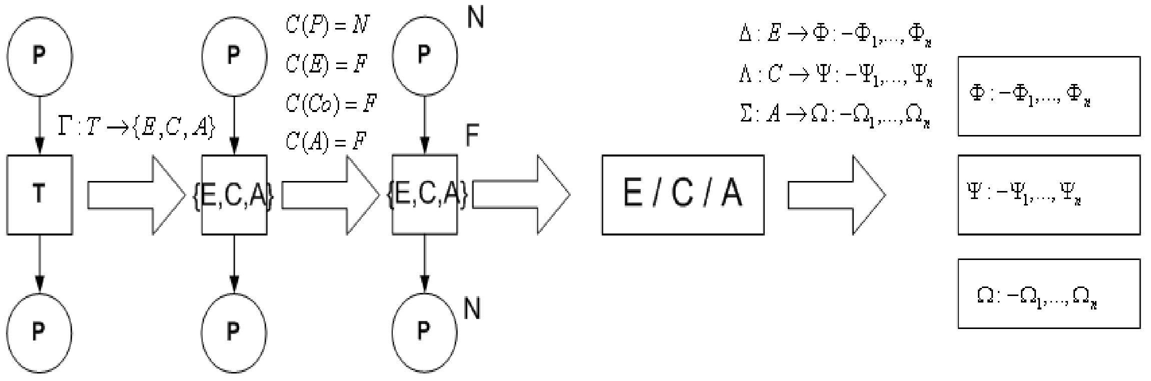
P
T
P
$\Gamma : T \rightarrow \{E,C,A\}$
P
{E,C,A}
P
$C(P) = N$
$C(E) = F$
$C(Co) = F$
$C(A) = F$
N
P
F
{E,C,A}
P
N
E / C / A
$\Delta : E \rightarrow \Phi : -\Phi_1,\ldots,\Phi_n$
$\Lambda : C \rightarrow \Psi : -\Psi_1,\ldots,\Psi_n$
$\Sigma : A \rightarrow \Omega : -\Omega_1,\ldots,\Omega_n$
$\Phi : -\Phi_1,\ldots,\Phi_n$
$\Psi : -\Psi_1,\ldots,\Psi_n$
$\Omega : -\Omega_1,\ldots,\Omega_n$

Figure 1: Mappings

6.1 Step Semantics

Step semantics can be now defined anew. Transitions are mapped to ECA rules and these are in turn mapped to expert system queries. Step semantics has to take into account possible composite nature of events, conditions and actions.

```
Check which transitions are enabled in what modes.For each transition and for each mode do:
  1. resolve mode and transition priority
  2. check whether transition contains simple event, complex event or null event
     (a) if it contains simple event, check whether event has happened and if so go to the condition part
     (b) if it contains complex event call ES inference machine
         i. infer simple events from a complex one
         ii. for each simple event:check it against the database or ask user
         iii. evaluate truth function of a complex event; if it evaluates to true:go to the condition part
     (c) if it contains null event: go to the condition part
  3. check whether condition is simple, complex or empty
     (a) if the condition is simple and is satisfied: fire
     (b) if the condition is complex than call ES inference machine
         i. infer simple conditions from a complex one
         ii. for each simple condition: check it against the database or ask user
         iii. evaluate truth function of complex condition; if it evaluates to true:fire
     (c) if the condition is empty:fire
  4. withdraw tokens from appropriate input places
  5. put tokens into appropriate output places
  6. set off appropriate action from action part of ECA rule
```

7 Examples

A real world example can now be presented that is based on this model. We will take first few chapters of Federal Rules of Civil Procedure and try to express them in our model. We take this example because it requires modelling of both procedural and substantive law. Also, these rules have already been modelled in (Meldman J.A., Holt A.W. (1971) and Meldman J.A. (1978)).

Procedure begins with plaintiff filing a complaint.

Rule 3 Commencement of action A civil action is commenced by filling a complaint with the court

Then, court issues summons.

Rule 4 Process Upon filing of the complaint the clerk shall forthwith issue a summons and deliver it for service to the marshal or to a person specially appointed to serve it.

Many events have been collected under one umbrella here: issue summons, deliver summons, and serve defendant with summons. System calls knowledge base and retrieves body of *summon* rule, which consists of three simple events: issue summons, deliver summons, serve the defendant. System matches these events with those written in the knowledge base and eventually asks user about them (is it the case that ...). Note that this is the event part of the ECA rule. Condition part is empty (null). We also have action part here, which sets latest respond time within which defendant must answer or else be confronted with default judgement.

As a consequence of this, defendant now has the right to answer with pleading, counterclaim, motion or default.

Rule 7 Pleadings allowed There shall be a complaint and an answer;[which may or may not contain a counterclaim, and a reply to a counterclaim.] No other pleadings shall be allowed...

Rule 12 Defenses: by pleading or motion A defendant shall serve his answer within 20 days after the service of the summons and complaint upon him...

Rule 55 Default When a party against whom a judgment for affirmative relief is sought has failed to plead or otherwise defend as provided by these rules...the clerk shall enter his default.

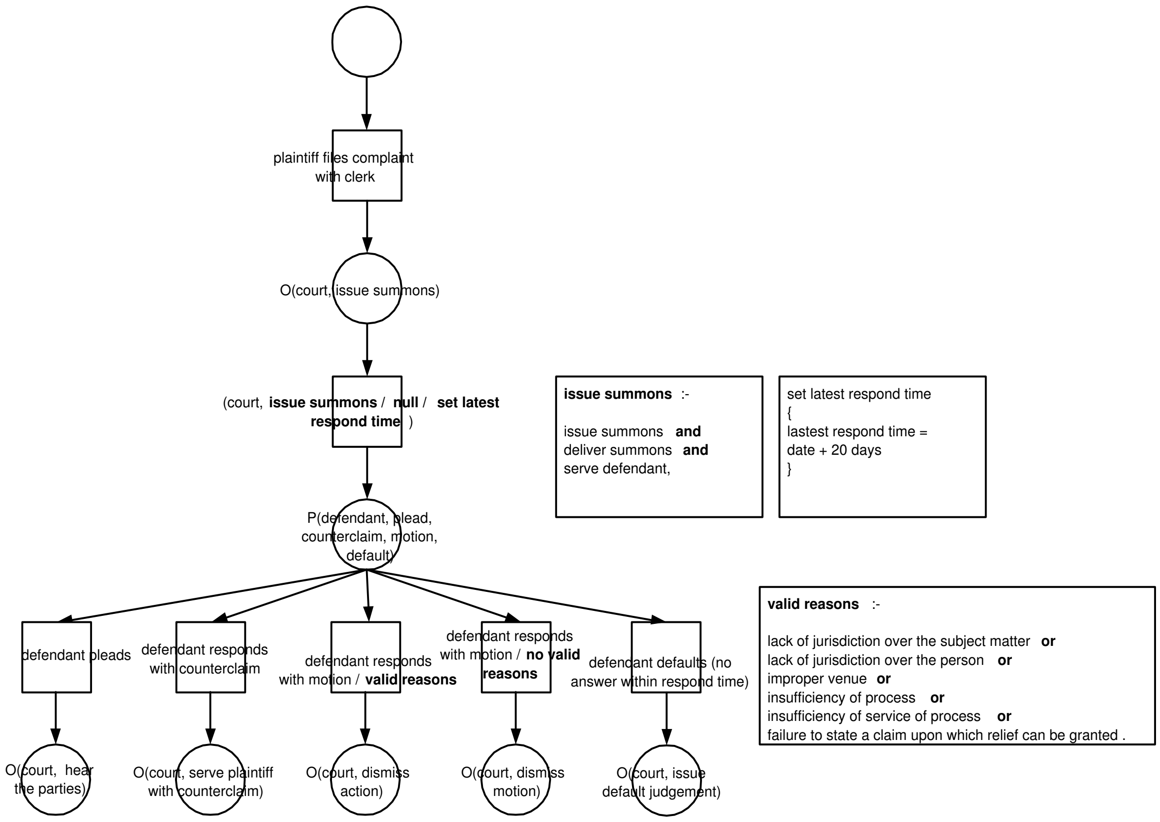
plaintiff files complaint with clerk
O(court, issue summons)
(court, **issue summons** / **null** / **set latest respond time**)
issue summons :- issue summons **and** deliver summons **and** serve defendant,
set latest respond time { lastest respond time = date + 20 days }
P(defendant, plead, counterclaim, motion, default)
defendant pleads
defendant responds with counterclaim
defendant responds with motion / **valid reasons**
defendant responds with motion / **no valid reasons**
defendant defaults (no answer within respond time)
valid reasons :- lack of jurisdiction over the subject matter **or** lack of jurisdiction over the person **or** improper venue **or** insufficiency of process **or** insufficiency of service of process **or** failure to state a claim upon which relief can be granted .
O(court, hear the parties)
O(court, serve plaintiff with counterclaim)
O(court, dismiss action)
O(court, dismiss motion)
O(court, issue default judgement)

Figure 2: Civil Procedure

If he chooses to plead, court proceeds with action. If he responds with counterclaim, court then serves plaintiff with counterclaim. If he defaults (doesn't answer in latest respond time,which the system has set before), court issues default judgement against him.

If he answers with motion, everything depends on validity of reasons for motion.

Rule 12b ... Every defense, in law or fact, to a claim for relief in any pleading, whether a claim, or counterclaim,... shall be asserted in the responsive pleading thereto... except that the following defenses may at the option of the pleader be made by motion: (1) lack of jurisdiction over the subject matter, (2) lack of jurisdiction over the person, (3) improper venue, (4) insufficiency of service of process, (6) failure to state a claim upon which relief can be granted...

These are conditions. System searches the knowledge base for *valid reasons for motion* and retrieves: lack of jurisdiction over subject matter, lack of jurisdiction over persons, improper venue, insufficiency of process, insufficiency of service of process, failure to state a claim upon which relief can be granted. Any of these reasons (conditions) make court dismiss action. If none of them is satisfied, court dismisses motion.

Here, the procedure goes on, of course. Model could be extended further, but we stop here because it serves demonstration purposes.

8 Conclusions

This paper has shown how high-level Petri nets, expert systems and ECA rules can be combined to represent semantics of legal rules. Different aspects of legal rules can be covered: factual, deontic, procedural and substantive. All of these are put into one picture.

Purpose of this paper is to show how semantics of legal rules can be mapped to different technologies and notations and how they can work together. This methodology has been applied to a few examples. Federal Rules of Civil Procedure, presented in this paper, is one of them. Both factual and deontic, procedural and substantive rules were covered in it.

Model could be extended further; one obvious way is to include linguistic elements as a micro structure (macro structure being Petri net). For example, if one wants to operate with finer elements than rights and duties (components like norm subjects, norm objects, prescribed behaviors) they must be interpreted individually. One way of doing this is by parsing text of legal rule and obtaining individual words or atoms like legal subjects, legal objects etc.). Then, not only facts, but single words also, could be subjects of queries and rules of expert systems (e.g. what is the meaning of word *due* in legal expression *due dilligence*?).

Also, model could be made executable. A mapping could be defined from the model to some programming language data types or DB schema. Models would than be migrated to this platform. Of course, a lot of implementation issues would have to be solved. Linguistics, again, could be of great help in determining atoms of such model.

Time aspects have only been very briefly touched in this paper. Since time is ubiquitous in information systems, model should be augmented with it. One way to do this is with Time or Timed Petri nets, where time is attributed to places or transitions or both.

Petri nets themselves could be replaced by more flexible structures. Petri nets semantics require strict alternation of transitions and places. Some real-world legal situations may escape this logic and we may very well find ourselves in need of a more flexible semantics.

References

[18] Federal Rules of Civil Procedure (2001) *www.house.gov/judiciary/civil00.pdf*

[18] Allen L.E. (1997) The Language of LEGAL RELATIONS (LLR):Useful in a Legal Ontologist's Toolkit?. *Proceedings of the First International Workshop on Legal Ontologies LEGONT'97*, Melbourne,Victoria, Australia, p. 47-60.

[18] Bratko I. (2001), Prolog Programming for Artificial Intelligence - 3rd edition, Addison-Wesley.

[18] Breuker J.,Valente A.,Winkels R. (1997) Legal Ontologies: A Functional View. *Proceedings of the First International Workshop on Legal Ontologies LEGONT'97*, Melbourne,Victoria, Australia, p. 23-36.

[18] Burg J.F.M., Van de Riet R.P. (1994) Syntax, Semantics and Pragmatics of COLOR-X Event Models Specifying the Dynamics of Information and Communication Systems. *Technical Report IR-365, Vrije Universiteit, Amsterdam, 1994*

[18] Burg J.F.M., Van de Riet R.P. (1994) COLOR-X: Object modelling profits from linguistics. *Technical Report IR-365, Vrije Universiteit, Amsterdam*

[18] Chakravarthy S., Mishra D. (1991) Snoop: An Expressive Event Specification Language For Active Databases. *Tech. Report UF-CIS-TR-93-007*, Gainesville, Florida.

[18] Dittrich K.R., Gatziu S. (1993) Time Issues in Active Database Systems. *Proceedings of International Workshop on an Infrastructure for Temporal Databases*, Arlington,Texas, p. 1-6.

[18] Fedorov S. (1991) ICaR project, - applying relational data model and expert systems technology to represent and use legal knowledge. *Industrial Report ICAIL-99 Conference, 1991*

[18] Gatziu S. (1993) Events in an Active Object-Oriented Database System. *Proceedings of the 1.st International Workshop on Rules in Database Systems*, Edinburg.

[18] Jonathan Billington (Editor) (1997) High-level Petri Nets - Concepts, Definitions and Graphical Notation. *Committee Draft ISO/IEC 15909*, October 2, 1997 Version 3.4

[18] Kralingen van R. (1997) A Conceptual Frame-based Ontology for the Law. *Proceedings of the First International Workshop on Legal Ontologies LEGONT'97*, Melbourne,Victoria, Australia, p. 15-22.

[18] Lee R. M., Ryu Y.U. (1994) DX: A Deontic Expert System. *Journal of Management Information Systems, Vol. 12, No. 1, 1995*, pp. 145-169.

[18] Lee R.M., Bons R.W.H., Wrigley C.D., Wagenaar R. W. (1995) Modelling Inter-organizational Trade Procedures Using Documentary Petri Nets. *Proceedings of the Hawaii International Conference on System Sciences 1995*.

[18] Meldman J.A., Holt A.W. (1971) Petri Nets and Legal System. *Jurimetrics Journal 12/2 1971*, ,p. 65-75.

[18] Meldman J.A. (1978) A Petri-Net Representation of Civil Procedure. *IDEA The Journal of Law and Technology 19/2 1978* , , p. 123-148.

[18] Sergot M.J., Sadri F., Kowalski R.A., Kriwaczek F., Hammond P., Cory H.T. (1986) The British Nationality Act as a logic program. *Communications of the ACM 370, 1986*

[18] Visser P.R.S.,Bench-Capon J.M. (1997) A Comparison of Two Legal Ontologies. *Proceedings of the First International Workshop on Legal Ontologies LEGONT'97*, Melbourne,Victoria, Australia, p. 37-46.