Time in Automated Legal Reasoning[*]

Lluís Vila[†] **Hajime Yoshino**

Technical Report #96-57

Information and Computer Science Dept.
University of California, Irvine
Irvine, CA, U.S.A.

Meiji Gakuin University
Tokyo, Japan

[*]**Acknowledgments.** We are thankful to Trevor Bench-Capon, Eddie Schwalb, Marek Sergot and anonymous referees for helpful remarks.

[†]Contact address: Lluís Vila, IIIA-CSIC, 08193 Cerdanyola, Catalonia, Spain; `vila@iiia.csic.es`; `http://www.iiia.csic.es~vila`. Partially supported by Spanish MEC grant EX-94 77909683 and by CICYT project *ARREL* (TIC92-0579-C02-01).

Keywords. Formalizing law, time, legal reasoning.

Abstract

Despite the ubiquity of time and temporal references in legal texts, their formalization has often been disregarded or addressed in an *ad hoc* manner. We address this issue from the standpoint of the research done in temporal representation and reasoning in AI. We identify the temporal requirements of legal domains and propose a temporal representation framework for legal reasoning independent of (i) the underlying representation language and (ii) the specific legal reasoning application. The approach is currently being used in a rule-based language for an application in commercial law.

Contents

1 Introduction

Automated legal reasoning systems require a proper formalization of time and temporal information [43, 32]. Quoting L. Thorne McCarty [32]:

> "...time and action are both ubiquitous in legal domains. ..."

Notions related to time are found in major legal areas such as labor law (e.g. the time conditions to compute benefit periods), commercial law (e.g. the time of the information used to establish the validity of agreements or to calculate damages[1] [7]), criminal law (e.g. the temporal information known about the various elements involved in the analysis of a criminal case), patent law (e.g. the time constraints formulated in regulations for applying for and obtaining a patent). Moreover, many *procedural codes* associated with these statutes usually require the management of time tables based on some temporal representation.

We elaborate on two representative examples. The first example is from the *United Nations Convention for International Sale of Goods* (CISG)[59].

Example 1 (CISG) Article 15*: An offer becomes effective when it reaches the offeree. An offer, even if it is irrevocable, may be withdrawn if the withdrawal reaches the offeree before or at the same time as the offer.*

This article contains various temporal aspects that are common in legal texts. We find denotations for events that *happen at a certain time* (e.g. "reach"), objects that have a certain *lifetime* (e.g. "offer", "withdrawal"), properties that *change over time* (e.g. "an offer is effective") and *temporal relations* (e.g. "before or at the same time").

We borrow the second example from [38].

Example 2 *Next two articles belong to the* Canadian Unemployment Insurance Law*:*
Section 9(1) *[...] A benefit period begins on the Sunday of the week in which (a) the interruption of earnings occurs, or (b) the initial claim for benefit is made, whichever the later.*
Section 7(1) *[...] the qualifying period of an insured person is the shorter of:*
(a) the period of fifty-two weeks that immediately precedes the commencement of a benefit period under subsection 9(1), and
(b) the period that begins on the commencement date of an immediately preceding benefit period and ends with the end of the week preceding the commencement of a benefit period under subsection 9(1).

In addition to denotations of temporal events (e.g. "interruption of earnings", "claim for benefits"), we find references to temporal units such as "qualification

[1]It can be the time before the tort, at the tort time, before the trial, until the damages have been paid or even after that.

period" and "benefit period", and temporal relations such as "begins", "ends", "period of fifty-two weeks", "the period that precedes", "the period that immediately precedes" and a rich variety of temporal operators such as "the shorter of ...", "the Sunday of the week...", "the later of ...".

This work belongs to the tradition of applying logic to formalize law. Despite the prominent presence of temporal references in legal texts, not much has been done in the *AI and law* community towards a *general* temporal representation framework. Time is an issue that legal reasoning projects have often disregarded or addressed in an *ad hoc* manner. The situation is more surprising if we take into account the intensive research done on temporal reasoning in AI during the past 15 years (see [50] for a survey). This may be due to the fact that, quoting Marek Sergot [43] "it looks like a *huge* topic" or that it requires techniques traditionally disconnected from legal reasoning such as constraint satisfaction.

Our goal is to provide a representation framework well-suited to formalizing the temporal aspects of law in its different areas. We build upon results from the area of *temporal reasoning about time in AI.*

We proceed by first identifying the requirements of legal domains (section 2). Then we outline the features that characterize a temporal representation framework and point our some of the choices studied in the area (section 3.1). Next we overview some related work (section 3.2) and, finally, we systematically discuss each feature and propose a choice that best addresses the requirements (section 4). We illustrate the adequacy of our proposal, called LTR, by revisiting the examples above.

Our requirement analysis gives us some guarantee about the applicability of LTR. This work does not address the issues of (i) periodic occurrences, (ii) handling time associated with legal provisions, and (iii) non-monotonic temporal reasoning.

The contribution of this paper is twofold: (i) as a reference for analyzing the temporal representation in existing legal reasoning systems, and (ii) as the foundation in building the "temporal component" of a legal reasoning application.

Terminology Before going ahead we define some few common terms used in the temporal reasoning literature and also throughout this paper. By *temporal expression* we mean an expression whose denotation is naturally associated with a specific time. In the above examples, "offer is effective" and "interruption of earnings" are temporal expressions. We shall distinguish between ***fluents*** when they are expressions that describe the state of affairs in a given domain ("offer is effective") and *events* when they represent occurrences that may change that state ("interruption of earnings")[2]. A *temporal proposition* is a logical proposition representing a temporal expression. By *temporal relation* we mean

[2]"Offer" can be modelled as an event, if refer to the offer object, or as a fluent if we refer to the "existence of the offer.

a relation whose arguments are all temporal, and by *temporal function* a function whose range is temporal[3].

2 Requirements

Generally speaking, the requirements for a computational representation language are two: (i) notational efficiency and (ii) computational efficiency. *Notational efficiency* comprises issues such as expressiveness, modularity, readability, compactness, flexibility, ... of the representation, whereas *computational efficiency* concerns the efficiency of the inference system in returning answers. Next we discuss them in the specific case of a language for formalizing law.

2.1 Notational Efficiency

Repeated Temporal References A *repeated temporal reference* is a temporal expression that includes a reference to another temporal expression. Let us have a closer look on example 1:

> "An **offer**, even if it is **irrevocable**, may be **withdrawn** if the **withdrawal reaches** the **offeree** <u>before or at the same time</u> as the **offer**."

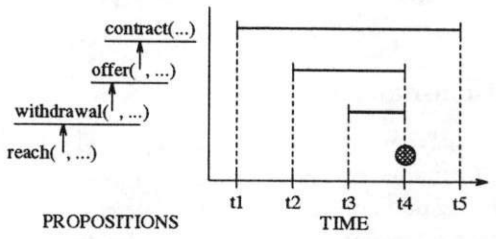


Figure 1: *Repeated temporal reference* example.

The "reach" event makes reference to a "withdrawal" of an "offer" of a "contract", all these being temporal objects with their own associated times (see figure 1). Although their times are different, there may be some implicit temporal constraints between them. For example, the "reach" cannot happen outside the lifetime of the offer.

Repeated temporal references abound in legal domains.

Temporal Operators Formalizing temporal domains involve a number of temporal operators such as, from example 1, an function that returns "the shorter of" two periods or "the latest of" two dates.

[3]As opposed to a function whose interpretation is time-dependent.

Precise and Indefinite Temporal Relations In addition to exact times and dates (e.g. 3:15pm, October 2nd 1996), many different classes of "less precise" temporal relations can be found in legal texts. The following are some examples of indefinite relations: "... before or at the same time than ...", "... during ...", "... contains or overlaps ...", "... immediately precedes ...", "... in few days ...", "... between 2 or 3 days ...", "... either 2 or 3 days if ... or between 1 and 2 weeks if ...". They are *indefinite* in the sense that they represent a set (interpreted as a disjunction) of possible times. When the set is not convex we talk about *non-convex* or *disjunctive* relations.

Indefinite relations are also present in the description of legal cases (e.g. "...few days later the message was dispatched", "the transaction took a couple of weeks", "between 9:00 and 10:00 the suspect was seen at ...").

Several Temporal Levels Some legal applications require distinguishing among different levels of temporal information [43]. A common distinction (often made in database systems [45]) is *real time* (in databases called *valid time*) vs. *belief time* (i.e. *transaction time*).

Modularity Since legal domains usually involve knowledge related to various notions such as evidence, belief, intention, obligation, permission, uncertainty, modularity is a central issue. A desirable feature of a temporal representation is that it allows to orthogonally combine time with all these different knowledge modalities.

2.2 Computational Efficiency

The ability to efficiently encode and process temporal relations may have a high impact on the performance of the overall procedure. The *size* of the temporal representation will be polynomial in the number of temporal propositions and the number of possible temporal relations which, in turn, depends on the model of time adopted (bounded, dense/discrete, etc.).

The *time* performance of answering queries can be strongly influenced by the class of temporal relations. The worst-case time complexity of checking consistency of a set of temporal constraints can at best be linear in the number of relations, but if the indefiniteness of temporal relations is *non-convex* it is unlikely that the problem will be tractable [55, 15]. Fortunately, in most of legal scenarios the ratio *number of temporal relations vs. number of temporal propositions* is relatively low and the non-convex indefiniteness is small. However, some cases are found in specific domains (such as in some criminal cases) or some tasks (e.g. *legal planning*) when multiple temporal possibilities need to be taken into account.

In both, easy and hard cases, the capability of efficiently answering queries about temporal relations is important. In the easy case because some legal scenarios may involve many temporal propositions. In the hard case because of the potential dramatic performance degradation due to the combinatorial nature of non-convex relations.

2.3 We do not Address

Periodic Occurrences Although not very common, some legal norms and cases require the expression of periodic events such as "pay X once every month" or "get a supply twice a week from 1/1/95 to 1/1/96". This is an issue of current reseach [35, 57] that we do not address here.

The Time of Law Law changes over time. New norms are introduced and some existing ones are derogated over time. A proper account of these changes is obviously important to correctly interpret the law [10, 11]. This is a fairly open issue in automated legal reasoning. It could be handled by means of a temporal representation that associates time with objects more complex than propositions such as rules or contexts. Our investigation here is restricted to time associated to atomic propositions.

Non-monotonic Temporal Reasoning Rescinding agreements, withdrawing decisions, handling retro-active provisions[4], ... all require non-monotonic reasoning capabilities. This issue is related to time in that non-monotonic assumptions and inference rules will be formulated using the underlying temporal language. Moreover, there is a non-monotonic reasoning specificly temporal: it deals with assumptions about temporal relations. For instance, we may want to assume that a fluent over time as long as it is consistent with the rest of the information. This matter is out of the scope of this paper.

3 Temporal Representation: Background

3.1 Features

A temporal reasoning approach is defined by a set of features that we survey in this section. They are graphically presented in figure 2[5].

Time Ontology The most basic feature is *ontology*, namely the set of *primitive temporal units* and *primitive temporal relations.* The two classical approaches are *instants* (or time points) and *periods* (or time intervals). As in-

[4]Retro-active effects are related with the issue of law change.

[5]Labels in bold indicate framework components and the ones in *italics* are either a set of axioms or a set of algorithms based on a set of axioms. Imbricated blocks denote a strong dependency whereas arrows represent a loose dependency.

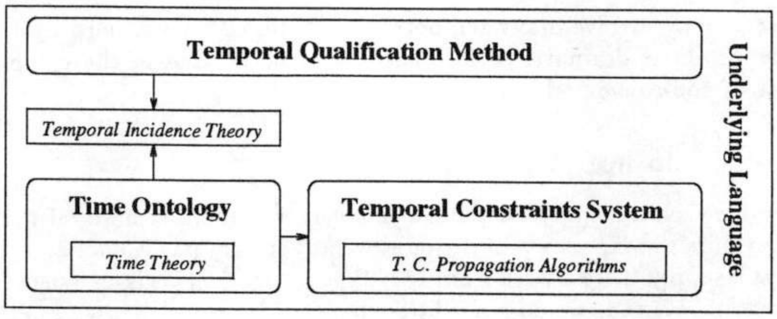

Temporal Qualification Method
Temporal Incidence Theory
Time Ontology
Time Theory
Temporal Constraints System
T. C. Propagation Algorithms
Underlying Language

Figure 2: The features of a temporal representation framework.

stant primitive relations, for example, one can take the three simple qualitative relations between two points in a line: $<$, $=$ and $>$.

When temporal relations involve numeric information, an additional ontological unit is needed: the *duration*. Durations represent distances between times[6].

A related ontological issue is *granularity*. From a semantical point of view, granularity is defined as the primitive unit of "real time"[7] over which the primitives of our time ontology are interpreted. From a practical point of view, the granularity is determined by the smaller unit used to specify durations in a given context. Different contexts may require different granularities and systems dealing with different contexts may require a mechanism to switch from one granularity to another.

The intuitions about the structure of time (such as the type of ordering, bound/unbound, discrete/dense, ...) are specified by a set of axioms called the **theory of time**. A lot of work has been done on the study of theories based on instants [48] and periods [56, 23, 36, 3], on deriving one primitive from the other, and on defining ontologies that combine them [48, 46, 4, 8, 18, 52].

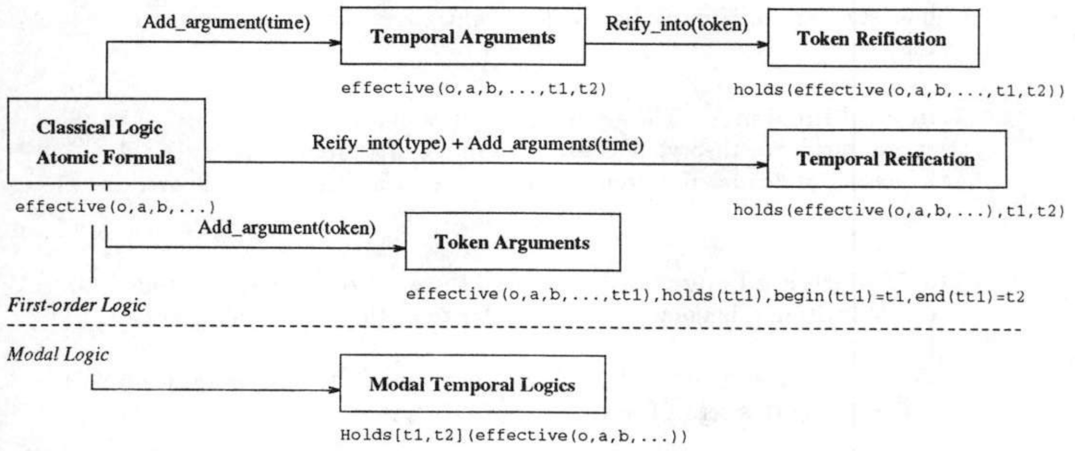
Temporal Constraints The primitive temporal relations and (logical) combinations of them are naturally regarded as constraints. For example, "the period p is before or after the period p'" is a constraint that restricts the set of possible values for the relative temporal distance between p and p'. When the set is non-convex we talk about non-convex constraints. This together with the temporal units and the allowed temporal constraints determine the *temporal constraint class*. For instance, the constraint in the above example is a *non-convex qualitative interval* constraint. A temporal constraint formalism must be provided with a set of specialized **temporal constraint satisfaction**

[6]Note that instant-to-instant numeric relations, period lengths and absolute times (such as dates) can all be regarded as durations.

[7]"Real time" here means time that can be measured by an existing device.

algorithms [47, 21, 40].

Temporal Qualification A central feature is the method employed to abscribe time to temporal propositions. It usually involves a number of newly defined predicates such as Allen's *Holds* or Shoham's *True* which express that a given proposition is true at a certain time(s). These are called *temporal incidence predicates* (TIP). Figure 3 presents an scheme of the various temporal qualification methods proposed in the literature.



Add_argument(time)
Temporal Arguments
Reify_into(token)
Token Reification
effective(o,a,b,...,t1,t2)
holds(effective(o,a,b,...,t1,t2))
Classical Logic
Atomic Formula
Reify_into(type) + Add_arguments(time)
Temporal Reification
effective(o,a,b,...)
holds(effective(o,a,b,...),t1,t2)
Add_argument(token)
Token Arguments
effective(o,a,b,...,tt1),holds(tt1),begin(tt1)=t1,end(tt1)=t2
First-order Logic
Modal Logic
Modal Temporal Logics
Holds[t1,t2](effective(o,a,b,...))

Figure 3: Temporal qualification methods in AI.

The most straight forward approach, called *temporal arguments* [24, 5][8], proposes introducing time as an additional argument(s) (e.g. **effective(o,a,b,...,t1,t2)**). A variation called *token arguments* [14, 19] uses a third element, the *temporal token* or *token*, to link propositions with their times (e.g. **effective(o,a,b,...,tt1), begin(tt1)=t1**). A token represents a particular temporal instance of a given temporal proposition.

The *temporal reification* approach [33, 2] models temporal propositions as logical terms called *propositional terms*. A propositional term is associated with its relevant times by making them all arguments of a TIP (e.g. **Holds(effective(o,a,b,...),[t1,t2])**). A variant called *token reification* [51], proposes first adding time as argument and then reifying (e.g. **holds(effective(o,a,b,...,t1,t2))**). In this case the propositional term denotes a temporal token.

Finally modal approaches introduce a number of temporal modal operators that qualify propositions. Classically temporal modal operators are *relative*. For instance, given a proposition Φ, **F**Φ means Φ is true in some future, **G**Φ means Φ is true in every future time, **N**Φ means Φ is true at

[8]This is the approach classically used in databases.

next time. *Absolute* operators are formed by using time as an index (e.g. `Holds[t1,t2](effective(o,a,b,...))`).

Modal approaches are attractive for their expressiveness, notational compactness and modularity. Although it is an appealing choice, in this paper we only consider methods based on first order logic since it is a more standard and widely used language. They turn out to be expressive enough for our requirements.

The trade-off among the various first order approaches is increaseda expressive power (which is limited in *temporal arguments*) vs. keeping the language simple, standard and ontologically clear (which are common objections to reification).

Temporal Incidence The general properties of the TIPs are specified by the temporal incidence theory. A classical example of temporal incidence axiom is *homogeneity* of `Holds`: if a proposition holds over a period it holds over any of its subtimes.

The Underlying Language Finally all these various temporal elements are integrated within a language which we refer to as the underlying language.

As an example, figure 4 shows how Allen's influential temporal logic [2] is described using this set of features.

Allen's Interval-based Temporal Logic	
Time Ontology	Units: Interval Relations: { 13 Qualitative Interval Relations }
Time Theory	Interval Existence Interval Relations Exclusivity Interval Transitivity Axioms
Temporal Constraints	Formalism: Interval Algebra (**IA**) Algorithm: **IA** Path-Consistency
Temporal Qualification	Temporal Reification
Temporal Incidence Theory	TIPs: {`holds,occurs,occurring`} Axioms: fluents homogeneity, events solidness
Underlying language	First order logic

Figure 4: Description of Allen's temporal logic.

3.2 Related Work

In many legal reasoning systems time is represented as any other attribute. Some systems are provided with an *ad hoc* temporal representation which may range from few built-in functions to a whole temporal subsystem. For example, Gardner [20] proposes a system for analysis of contract formation which includes a temporal component. The ontology is composed of time points and time intervals. A distinction is made between events and states (i.e. fluents). Time is treated as another argument. Since all the arguments are expressed through a proposition identifier, time among them, the temporal qualification method is similar to token arguments method. Some relevant features, however, are less developed due to the bias towards the specific application: the time unit is fixed to days, only few point-to-point relations are considered (some temporal relations such as "follows" or "immediately" are mentioned but not supported), and issues such as temporal constraints and temporal incidence are not considered at all.

KRIP-2 [37] is a system for legal management and reasoning in patent law whose language supports temporal representation. The ontology is also based on instants and periods, and includes both convex metric and qualitative interval temporal constraints. Events are qualified with time by using the form

event(Id, class, conditions, time)

Although *Id* looks like a token symbol, it is not used for temporal qualification since *time* is also an argument.

These temporal representation approaches turn out to be adequate for the purposes of the system they are defined in. However, as a general approach to temporal representation in law they lack of some of the following: (i) an explicit identification of requirements from legal domains, (ii) a consideration of the results in temporal reasoning in AI, and (iii) a rational decision on each of the issues involved in a temporal representation framework. In this paper we already went over (i) and (ii). In next section we go over (iii), but before that we analyze two pieces of work that include these three ingredients.

The first is the *event calculus* (EC) [28], a temporal database management framework specified in PROLOG. Although not specifically intended for legal reasoning, EC has been used in several legal formalizations [42, 6]. According to the above features, EC is described as follows:

Event Calculus	
Time Ontology	Units: Instant, period Relations: $\{<, =, >\}$
Time Theory	Not defined
Temporal Constraints	Not defined
Temporal Qualification	For fluents: *Temporal reification* For events: *Token arguments*
Temporal Incidence Theory	TIPs: {`holds`,`holds_at`} Axioms: holds homogeneity
Underlying language	PROLOG

The second is presented in the context of the *Chomexpert* system [30, 38], an application on the *Canadian Unemployment Insurance Law*. The features of the temporal representation language, called EXPERT/T, are summarized as follows:

EXPERT/T		
Time Ontology	Units:	Instant, Period
	Relations:	Qualitative point, qualitative interval, Qualitative point-interval, absolute dates
Time Theory	Not defined	
Temporal Constraints	Point and Interval Algebras Unary metric (absolute dates)	
Temporal Qualification	*Temporal reification*	
Temporal Incidence Theory	TIPs:	{holds_on, occurs_at}
	Axioms:	Not defined
Underlying language	PROLOG	

Although both works start from an analysis of temporal representation requirements, none identifies ***repeated temporal references, multiple time levels*** and ***modularity*** as relevant issues to address. This is the reason why either some of the choices they take are not the most well-suited for legal domains or are merely not defined. Both proposals (in EC only for fluents) use temporal reification as temporal qualification method. In next section we give a number of reasons to prefer the token arguments approach. Also both use PROLOG as underlying language. A shortcoming of languages purely based on logic (logic programming among them) is their inefficiency in handling constraints. Proof-driven inference procedures turn out to perform poorly in constraint processing. The integration of a constraint specialist seems the natural way to overcome this problem. EC does not provide any "machinery" for processing temporal constraints. Although the period primitive is part of the time ontology, period relations and interval algebra constraints (*a la* Allen) are not supported. EXPERT/T processes qualitative constraints using Allen's path-consistency propagation algorithm [38], but no type of metric constraints is supported.

Constraint Logic Programming (CLP) looks like the natural formalism to integrate temporal constraints in logic programming. Temporal CLP have been studied in a number of works [25, 9, 17, 41]. The temporal CLP language proposed by Schwalb, Vila and Dechter [41] has several similarities with our proposal here. The one presented here is more general in the sense that is not developed upon any specific underlying language, and it is more specific in the temporal qualification method to better fit the requirements of the legal domain.

4 Legal Temporal Representation

In this section we present our proposal called LTR. We analyze each of section 3.1 features: for each feature we propose the choice that best fits the requirements in section 2.

4.1 Time Ontology: Instants, Periods and Durations as Dates

Primitive Units Most temporal expressions in legal domains are associated with a period of time (e.g. "an offer being effective" in example 1, or the "qualifying" and "benefit" periods in example 2). Moreover, these expressions are often related by period relations such as "a period of validity of an offer happens during its period of existence" or "the qualifying period immediately precedes the benefit period". Hence, it is natural to include the period as a time primitive. Do we also need instants? A brief analysis of legal texts yields several cases where the notion of instant is involved:

1. The endpoints of the periods above are naturally associated with instants such as the moment where "the offer becomes effective" or the time as of which "the contract is no longer valid".

2. Some *events* such as "the offer reaches the offeree" are viewed as instantaneous. These are called *instantaneous events.*

3. Norms often involve conditions about the state of a certain fluent at a certain instant. For example, "If ...and the offer is not withdrawn at the moment when it reaches the offeree and ... then ...". Notice that, even if the "reach" event is modeled as durable, the condition may still refer to the instant at the end of that period.

4. Whenever metric temporal relations are involved, they are often stated as constraints between instants, (e.g. "a document sent by mail reaches its destination between 3 and 5 days later").

Besides instants and periods, the *duration* unit is needed as well since legal domains involve numeric relations.

In practice, time in legal domains is expressed in *clock/calendar units.* Accordingly we define our instant, period and duration constants in terms of *dates*, where a *date* is defined as an indexed sequence of values for clock/calendar units:

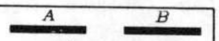
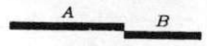
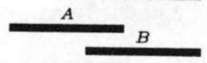
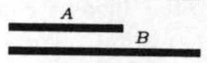
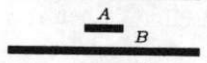
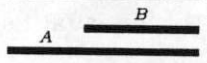
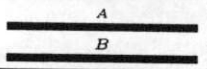
date ::= [*second*''][*minute*'][*hour*h][*day*d][*week*w][*month*m][*year*y]

For example, `00''15'21h2d10m96y`, `00''15'21h`, `21h2d10m96y`, `10w96y`, `96y` are well-formed dates. Some convenient shorthands are clock times (e.g.

00:15:21) and calendar dates (e.g. 2/10/96). Dates are used as both instant and duration constants. Period constants are defined as ordered pairs of dates. We use the conventional notation ()/[] to specify open/closed intervals. In addition, a set of indexed symbolic constants (i1,i2, ...p1,p2, ...) is included for each unit to express times not associated to any specific temporal proposition.

Granularity The adequate time granularity may vary from one legal context to another, yet the basic structure of time and the properties of temporal constraints do not change. We address this issue by allowing the user to select the appropriate granularity. Date constants will be interpreted as either an instant or a period according to what is specified by the directive `Granularity()` which takes a clock/calendar unit as its only argument. The issues of combining different granularities or dynamically changing of granularity are not addressed.

Primitive Relations Our proposal is based on the following primitive temporal relations: the 3 qualitative point relations $\prec$, $=$ and $\succ$, the 5 qualitative point-interval relations **Before**, **Begin**, $\in$, **End**, **After**, the 13 qualitative interval relations,

A Before B	B After A	
A Meets B	B Met_by A	
A Overlaps B	B Overlapped_by A	
A Starts B	B Started_by A	
A During B	B Contains A	
A Finishes B	B Finished_by A	
A Equal B	B Equal A	

and the duration relations $=$ and $\in$ used to express unary constraints only[9] (e.g. `duration(tt1)=52w`, `begin(tt2)-end(tt1)∈[3w,4w]`). Binary duration constraints are an issue of current research.

Primitive Functions We define a set of logical functions between temporal units. Some of them are just the functional version of a temporal relation above:

Begin, End :	*period*	$\mapsto$ *instant*
[],(),[),(] :	*instant* $\times$ *instant*	$\mapsto$ *period*
Duration :	*period*	$\mapsto$ *duration*

[9]Although the relations are binary, only one of the arguments will be a duration variable.

Besides, a set of *interpreted*[10] temporal functions is required in practice. These functions are not involved in the term unification but they are computed at inference time. This set includes functions such as the following:

- *Date arithmetics*, e.g. `+` : $date \times date \mapsto date$
- *Date predicates*, e.g. **is_holiday** : $date \mapsto \{t/f\}$
- *Date operations*, e.g. **next_holiday** : $date \mapsto day$
- *Date transformations*, e.g. **week_of** : $date \mapsto week$
- *Date set operations*, e.g. **nth,latest,shorter_of** : $date\text{-}set \mapsto date$

A list of them is given in [54].

Time Theory Provided with the set of dates as our underlying model of time, the only structural property of time that demands a specific discussion here is the dense/discrete one. Dense models are required in domains where *continuous change* needs to be modelled such as *qualitative physics*. This is not the case of legal domains where the relevant changes are (viewed as) discrete (e.g. "signing a contract", "receiving an offer", "interruption of earnings", ...) and the dates set has a basic, indivisible granularity. Therefore we adopt a discrete model of time which has two consequences. At the ontological level, we add two instant relations that are exclusive of discrete models: **Previous, Next**: $instant \times instant$[11]. At the axiomatics level, we take a discrete time theory. It is based on $\mathcal{IP}$ [49], a simple instant-period theory that accepts both discrete and dense models, plus few *discreteness* axioms. Both sets of axioms are given in appendix A.

The "Immediate" Relation Immediate is a difficult temporal term to characterize because its meaning may vary from one context to another. It may mean "in few seconds" or "in few hours". Even in a fixed context, it has not a precise interpretation. Our proposal is based on regarding immediate as a qualitative relation somewhere between `Previous(Next)` and $\prec(\succ)$. This loose connection is formally specified by the following axioms over instants:

$$\begin{array}{ll}
\mathbf{Im_1} & i \ \texttt{ImmediateAfter}\ i' \Rightarrow i' \prec i \\
\mathbf{Im_2} & i \ \texttt{ImmediateBefore}\ i' \Rightarrow i \prec i' \\
\mathbf{Im_3} & i \ \texttt{Previous}\ i' \Rightarrow i \ \texttt{ImmediateBefore}\ i' \\
\mathbf{Im_4} & i \ \texttt{Next}\ i' \Rightarrow i \ \texttt{ImmediateAfter}\ i'
\end{array}$$

When `Immediate` is adjoined to period relations, it is interpreted as one of the following two:

[10] Interpreted functions are also referred as *built-in* functions or *operators*.

[11] These relations will also be used in their functional form as time operators (e.g. `begin(tt1)=Next(end(tt2))`).

1. The period relation **Meets(Met_by)**.
2. The first (last) of the *set* of periods that follow (precede) the current period.

The appropriate choice will depend on the context and is left to the responsibility of the language user. We formalize some instances of immediate relations in the examples below.

4.2 Temporal Constraints

Given the indefiniteness of temporal relations in some legal domains[12] and the fact that existing temporal constraint algorithms scale down well in general, our framework includes almost all kinds temporal constraints:

- Qualitative constraints between instants (e.g. **begin(tt1)** $\preceq$ **begin(tt2)**)
- Metric constraints over instants (e.g. **begin(tt2)-begin(tt1)** $\in$ {**[2d,3d] [1w,2w]**})
- Qualitative constraints between periods (e.g. **period(tt3) Contains Overlaps period(tt2)**)
- Qualitative constraints between an instant and a period (e.g. **instant(tt2)** $\in$ **1/Oct/95**)
- Unary metric constraints over durations (e.g. **duration(P1)=52w**)

Besides representing indefinite temporal relations, temporal constraints can be used to maintain a partial representation over time. Consider, for instance, a fluent **f** that is holding now. Unless we have specific information, it may cease holding any time as the current time. It can be expressed by a constraint similar to **end(f)**$\in$ **[now,**$+$**inf]**.

Temporal constraints are either unary or binary and in both cases the syntax has the form

time-term temporal-relation time-term

where the types of the time terms agree with the signature of the temporal relation. In unary constraints, one of the timetemrs is alwasys ground. The formal syntax of the constraints is given in [54].

[12]Although in most legal applications only some specific classes of temporal constraints are involved, different applications require different types of constraints. Moreover, some few domains (such as labor law) where the temporal issue is paramount and data may be imprecise, involve all kinds of temporal constraints.

Temporal constraints processing is done by representing them in a constraint network and applying available efficient techniques for processing different classes of constraints: qualitative point [22, 47, 21, 16], qualitative interval [47] and metric point [15, 40]. Also some progress has been achieved in combining metric-point and interval algebra constraints [34, 26]. This currently is an area of active research and forthcoming results can be straighforwardly integrated within our framework.

4.3 Temporal Qualification: Token Arguments

Since *repeated temporal references* are pervasive in legal domains, temporal qualification methods based on tokens are more adequate. Among the two token-based methods proposed in the literature, *token arguments* is better suited to our needs here as we shall see in a moment. In *token arguments*, something like an offer of the contract `c` from `a` to `b` is formalized as `offer(c,a,b,...,tt1)` where `tt1` is a constant symbol of the new *token* sort[13]. We call these atomic formula *token atoms.* To improve *readability* we emphasize the role of the token argument with some syntactic sugar: instead of `offer(c,a,b,...,tt1)` (where `tt1` is a token term) we shall write

```
tt1 : offer(c,a,b,...)
```

A set of functions, called *token temporal* fuctions[14], that map tokens to their relevant times is defined. For example, `begin(tt1)` denotes the initial instant of the token denoted by `tt1` and `period(tt1)` its period. TIPs are used to express that the temporal proposition is true at its associated time(s) as discussed below in section 4.4.

The token arguments method has several advantages:

1. Token symbols can be directly used as an argument of other predicates. In the above example, `tt1` can be used in `dispatch(tt1,a,b,...)` to express that the offer `tt1` is dispatched from `a` to `b`.

2. Different levels of time are supported by diversifying the token temporal functions. For instance, we may have `begin_v(tt1)` to refer to *valid time* and `begin_t(tt1)` to refer to *transaction time.* At the implementation level, a different temporal constraint network instance is maintained for each time level.

3. Token symbols can be used as the link to other knowledge modalities. For instance, in a multiple agents domain, the degree of belief of a proposition

[13]The idea behind token arguments is similar to the *Compound Predicate Formula* approach [58] when applied to temporal pieces of information.

[14]To be distiguished from the temporal functions in section 4.1 with similar names but different signature.

`p(...)` by an agent `a` can be represented by `belief(a,tt1)` where `tt1` is a token from `tt1:p(...)`. *Deontic modalities* can be represented by predicates (such as `O` for obligation and `P` for permission) that take a token as an argument. Furthermore, we can distinguish between the time where the deontic relation holds and the time of the object in the relation. For example, consider that a legal person `a` is obligated to offer a contract `c` to `b`. We represent the offer by `tt1:offer(c,a,b,...)`, its relevant instants by `begin(tt1)` and `end(tt1)`, the obligation by `tt2:O(a,tt1)` and the beginning and end instants of the obligation by `begin(tt2)` and `end(tt2)`.

To increase notation *compactness* we define syntactic sugar that allows omitting token symbols whenever they are not strictly necessary (i.e. whenever there is no reference to them). There are two cases. In the first case two or more token atoms are collapsed into one. For instance, the facts

```
tt1:  offer(c,a,b,...)
tt2:  withdrawal(tt1)
tt3:  reach(tt2,b)
```

in a rule that does not contain other references to `tt2`, can be rewritted as

```
tt1:  offer(c,a,b,...)
tt3:  reach(withdrawal(tt1),b)
```

The second case is related with temporal incidence expressions and is explained in next subsection.

4.4 Temporal Incidence

We introduce the TIP `holds` to express holding of fluents (e.g. `holds(tt1)`) and `occurs` to express occurrence of events. We call these atomic formula *incidence atoms.*

Holds Incidence There is a common agreement in the literature about the *homogeneity* of holding of fluents [33, 2, 44]. Since our ontology includes both intants and periods, the holding of a fluent over a period should not constrain its holding at the period endpoints to avoid the *divided instant problem* [52]. These properties are captured by a simple axiom which, expressed in temporal reification form, is as follows:

$$\forall f : \text{fluent}, p : \text{period } \texttt{holds\_on}(f,p) \Longrightarrow (\forall i : \text{instant } \texttt{Within}(i,p) \Rightarrow \texttt{holds\_at}(f,i))$$

An important convention we make at this point is what we call *token holds maximality*:

> A fluent token denotes a maximal piece of time where that fluent is true.

A consequence of this convention is the following *Event Calculus* axiom:

> "Any two periods associated with the same fluent are either identical or disjoint."

In pratice, there is need for talking about a certain fluent being true at a certain time, according to what is entailed by the current token database. To this purpose we define the following TIPs:

```
holds_on(fluent, period)
holds_at(fluent, instant)
```

Notice that these are neither syntactic sugar of the above nor temporal reification TIPs, but they are new TIPs with the following existential meaning. Given a fluent *f*, a period *p* and an instant *i*:

$$\begin{aligned}
&\texttt{holds\_on}(f, p) \equiv \\
&\quad \exists TT\ TT{:}f \wedge \texttt{holds\_on}(TT) \wedge \\
&\quad p\ \texttt{During Starts Finishes Equal period}(TT) \\
&\texttt{holds\_at}(f, i) \equiv \\
&\quad \exists TT\ TT{:}f \wedge (\texttt{holds\_on}(TT) \wedge i\ \texttt{Within period}(TT) \vee \\
&\quad \texttt{holds\_at}(TT) \wedge i = \texttt{instant}(TT))
\end{aligned}$$

where TT is a variable of the *fluent token* sort.

Occurs Incidence There is no common agreement on the characterization of the occurrence of events [2, 44, 19]. As a matter of fact, no evidence on the need for any specific theory of events is found in practice. However, we keep `occurs` TIP to express the actual occurrence of an event and, thus, to allow describing events whose occurrence is unknown (e.g. to express the possibility or the obligation for that event to occur).

Some syntactic sugar for incidence expressions is defined to omit token symbols. The expression

```
TT:become-effective(...)
Occurs(TT)
instant(TT)=I
```

will be written as

```
Occurs(become-effective(...),I)
```

The formal syntax for incidence atoms is given in [54].

4.5 Underlying Language

Our proposal is independent of the underlying language. as long as it is a many-sorted language. Temporal and token sorts will be included.

In this section we address few additional relevant features:

Negation Negation of token and incidence atoms will be handled by the mechanism that the underlying language is provided with. Negation of temporal constraints is less problematic since temporal constraints exhibit the following nice property:

Proposition 1 *In a constraint language that does not restrict non-convex constraints, any negated constraint can be expressed as an equivalent non-negated constraint.*

For example $\neg(t \leq t') \equiv t > t'$, or $t - t' \in \{[3,5]\} \equiv t - t' \in \{[-\infty, 3), (5, +\infty]\}$. Hence negated constraints will be asserted and queried by regular constraint propagation and entailment.

Token Sets Some applications require dealing with sets of temporal elements[15]. For instance, let us consider the following text from example 2:

> ...(b) the period that begins on the commencement date of an immediately preceding **benefit period** and ends with the end of the week preceding the commencement of a benefit period under subsection 9(1).

Since for a given person there might be several *benefit periods*, a possible interpretation for "immediately preceding benefit period ..." is, as noted in section 4.1, "the last of all benefit periods before ...". Thus, we need to refer to the set of all those "benefit period" tokens that are `Before` ... Coping with the notion of set requires higher order expressiveness. Some research has been done extending first order languages in this direction [31, 29, 1, 12, 27, 13]. We restrict the development here to the context of a token-based approach where the notion of set applies to specifying sets of temporal tokens that satisfy certain conditions. The syntax we propose is as follows[16]:

`token_set(`[*temporal atom*]$^+$`)`

[15]This issue is not included in the requirements list (section 2) because the notion of set is not strictly a temporal representation feature, although it is directly related as we discuss in this section.

[16]We are not particularly happy with this syntax since is not in accordance with a pure declarative style, although it is adequate in practice.

where *temporal atom* can be either a token atom, an incidence atom or a temporal constraint. It has the form of an atomic formula but it is not. Instead it is an operator that binds the token variables appearing in the token atoms (e.g. the variable `TT3` in `TT3: benefit-period(TT1)`) to all those tokens of that relation that satisfy all the conditions inside the form. For instance, the example above is formalized as

```
token_set( TT3:  benefit-period(TT1)
          period(TT3) Before Meets period(TT2) )
```

We define a number of practical operators on sets of tokens. For instance, `latest` denotes the last token of that set according to the temporal ordering. These operators can be applied on token set variables (e.g. `latest(TT3)`). Some of these operators admit an alternative first order formulation by splitting the conditions into different rules and using negation, however this approach is clearly impractical[17].

Token Attributes The token arguments method allows to detach time from its temporal proposition. The same can be done for the remaining attributes of the propostion to enhance language flexibility. For example, we can refer to the offeror of `tt1: offer(c,a,b,...)` by `offeror(tt1)`. Now attribute names are represented explicitly. It requires (i) *declaring* the attributes for each predicate,

```
Attribute(what,offer)          for what we shall use the shorthand
Attribute(offeror,offer)
Attribute(offeree,offer)
...

Attributes(offer,{what,offeror,offeree,...})
```

and (ii) referring to the attributes of a particular token. Our `tt1: offer(c,a,b,...)` can be regarded as a shorthand[18] for

```
what(tt1)=c
offeror(tt1)=a
offeree(tt1)=b
...
```

[17]As an exercise, you may try to you this approach to specify the operator **4th** which selects the 4th token that satisfies certain conditions.

[18]The translation will take the order of the attributes from an explicit declaration supported by the undelaying language.

Summary The set of choices that defines our proposal is summarized in the following table:

LTR	
Time Ontology	Units: Instants, periods, durations with clock/calendar forms as constants. Relations: $\{\prec$, `begin`, `end`, `Next`, `Previous`, `ImmediateBefore`, `ImmediateAfter`$\}$
Time Theory	$\mathcal{IP}$ axioms + discreteness axioms + $\mathbf{Im}_{1\div4}$ axioms (The axioms are given in appendices A.1 and A.2)
Temporal Constraints	Combined (metric) Point – Interval Constraints
Temporal Qualification	*Token arguments*
Temporal Incidence Theory	TIPs: {`holds`,`occurs`,`holds_at`,`holds_on`} Axioms: `holds` and `holds_on` homogeneity

5 Examples

In this section we illustrate the application of our approach as we revisit the two examples introduced in section 1. We take a rule-based language as underlying language without making any assumption about the inference regime. A set of facts in both the body and the head of a rule is interpreted as a conjunction. The marks [[...]] indicate pieces of text that have not been formalized because either their meaning is not clear, their main emphasis is not temporal or they are merely redundant. The mark `% Implicit` indicates pieces of formal knowledge that are not directly derived from the legal text. Ontological elements resulting from a conceptualization process are emphasized in **bold**. Temporal relations are <u>underlined</u>.

5.1 Formalizing the CISG Example

The CISG is intended to provide a normative frame for international commerce. Part II of the law is devoted to the formation of contracts. For instance, it is used to determine when a contract is concluded. Queries like this can be answered in the LTR formalization we present next.

The predicate attributes used in the example are:

```
Attributes(contract,{offeror,offeree,class,type,qp-provision})
Attributes(offer,{what,offeror,offeree,is-irrevocable,offer-begin,offer-end})
Attributes(acceptance,{what})

Attributes(effective,{what})
Attributes(concluded,{what})
Attributes(withdrawn,{what})
Attributes(accepted,{what})

Attributes(become-effective,{what})
Attributes(become-concluded,{what})

Attributes(reach,{what,whom})
Attributes(dispatch,{what,whom,to-whom,type,stamped-date})
```

A granularity of days might seem fine enough for this example, however some occurrences of the "immediate" relation require moving to a finer granularity:

```
Granularity(second)
```

A law article is formalized as (a number of) rules that express the relations between occurrence of events under certain conditions and their effects in terms of the holding of derived fluents. For instance, in example 1, "**Article 15(1)** An **offer** becomes **effective** <u>when</u> it **reaches** the **offeree**." is formalized as

```
If      TT1:  offer(C,OR,OE,...)
        TT2:  reach(TT1,OE)
        Occurs(TT2)
        ¬Holds_at(withdrawn(TT1),instant(TT2))              % Implicit
then    Occurs(become-effective(TT1),instant(TT2))


If      TT2:  become-effective(TT1) % Implicit
        Occurs(TT2)
then    Holds(effective(TT1),(instant(TT2),_))
```

Next we include few additional interesting articles also from CISG part II.

Article 18(2) An **acceptance** of an **offer** becomes **effective** <u>at the moment</u> the **indication of assent reaches** the **offeror**. An **acceptance** is **not effective** if the **indication of assent** does not **reach** the **offeror** <u>within the time</u> he has fixed [[or, if no time is fixed, within a reasonable time, due account being taken of the circumstances of the transaction, including the rapidity of the means of communication employed by the offeror.]]

```
If      TT1:  offer(_,OR,OE,_,OBegin,OEnd)
        TT2:  acceptance(TT1)
        TT3:  reach(TT2,OR)
        Occurs(TT3)
        instant(TT3) ∈ [OBegin,OEnd]
        Holds_at(accepted(TT1),instant(TT3))                   % Implicit
then    Occurs(become-effective(TT1),instant(TT3))
```

Implicit from Article 18(2) When an acceptance of an offer of a contract becomes effective the contract becomes concluded.

```
If      TT2:  become-effective(acceptance(offer(TT1,...)))
        Occurs(TT2)
then    Occurs(become-concluded(TT1), instant(TT2))

If      TT2:  become-concluded(TT1) % Implicit
        Occurs(TT2)
then    Holds(concluded(TT1),(instant(TT2),_))
```

Article 18(2) (cont) An **oral offer** must be[19] **accepted** immediately [[unless the circumstances indicate otherwise.]]

```
If      TT1:  offer(_,OR,OE,...)
        TT2:  dispatch(TT1,oral)
        Occurs(TT2)
then    offer-begin(TT1)← instant(TT2)
        offer-end(TT1)← ImmediateAfter(instant(TT2))
```

Article 20(2) Official **holidays** or non-business days occurring during the **period for acceptance** are included in calculating the **period**. However, if a notice of **acceptance** cannot be **delivered** at the address of the **offeror** on the last day of the period because that day falls on an official holiday or a non-business day at the place of business of the **offeror**, the **period** is extended until the first business day which follows.

[19]Notice that "must be" here does not denote obligation but a temporal constraint.

```
If      TT2:  offer(...)
        Is_holiday(offer-end(TT2))
then    offer-end(TT2)← next_holiday(offer-end(TT2))
```

The complete formalization of part II of the CISG can be found in [53].

Temporal database projection[20] would be sufficient to answer the intended queries. The bottom-up inference procedure would make an intensive use of the specialized modules for (i) constraint processing and (ii) token management. The result will be a temporal map composed of instants and periods for the instances of events and fluents, together with the temporal constraints holding among them. For example, given the input formalized by the following facts

```
tt1:  contract(a,b,sale,machine,_)
tt2:  offer(tt1,a,b,_,[−∞,+∞],[−∞,+∞]), instant(tt2)∈1/Oct/95
tt4:  reach(tt2,b), instant(tt4)∈8/Oct/95
tt5:  withdrawal(tt2)
tt6:  dispatch(tt5,a), instant(tt6)∈7/Oct/95
tt7:  reach(tt5,b), instant(tt7)∈11/Oct/95
tt8:  acceptance(tt2)
tt9:  dispatch(tt8,b), instant(tt8)∈10/Oct/95
tt10:  reach(tt8,a), instant(tt10)∈12/Oct/95
```

the time map shown by figure 5 would be generated. The query "Is the contract concluded" will be affirmatively answered by `YES, as of October 12 '95`. The sequence of rules involved in deriving token `tt1.2:  concluded(tt2)` can be easily recorded and returned as justification.

5.2 Formalizing the Canadian Unemployment Insurance Law Example

A key section of the *Canadian Unemployment Insurance Law* [38] is intended to determine whether a person is eligible for benefits or not. It involves determining a *qualifying period* (the period during which the person has been employed) and a *benefit period* (the period during which the person should receive benefits).

The following predicate attributes need to be declared:

[20]As in the *TMM* system [14, 39] for example.

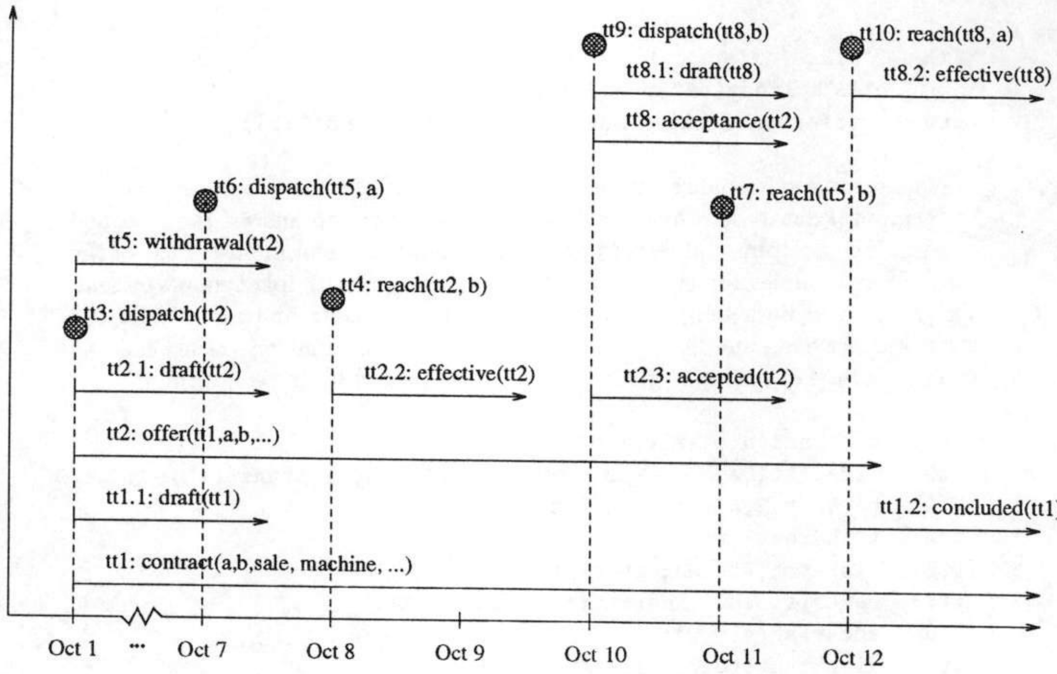


Figure 5: CISG example.

```
Attributes(insured-person,{...})
Attributes(benefit-period,{whom})
Attributes(qualifying-period,{whom})

Attributes(interruption-of-earnings,{what})
Attributes(initial-claim,{what})
```

For a proper formalization of the temporal aspects of this act, a granularity of days is fine enough.

```
Granularity(day)
```

Next we show the sections that address the assesment of the benefit and qualifying periods and their formalization in **LTR**:

Section 7(1) [...] the **qualifying period** of an **insured person** is the <u>shorter of</u>: (a) the period of <u>fifty-two weeks</u> that <u>immediately precedes</u> the <u>commencement of</u> a **benefit period** under subsection 9(1), and
(b) the period that <u>begins</u> on the <u>commencement date</u> of an <u>immediately preceding</u> **benefit**

period and ends with the end of the week preceding the commencement of a **benefit period** under subsection 9(1).

```
If      TT1:  insured-person()
        TT2:  benefit-period(TT1)
        duration(P1)=52w
        P1 Meets period(TT1)
        token_set( TT3:  benefit-period(TT1)
                   period(TT3) Before Meets period(TT2) )
        begin(P2)=begin(latest(TT3))
        end(P2)←end_of_week(week_before(week_of(begin(TT2))))
then    TT5:  qualifying-period(TT1)
        period(TT5)←shorter_of({P1,P2})
```

Section 9(1) [...] A **benefit period** begins on the Sunday of the week in which
(a) the **interruption of earnings** occurs, or
(b) the **initial claim** for benefit is made,
whichever the later.

```
If      TT1:  insured-person()
        TT2:  interruption-of-earnings(TT1)
        Occurs(TT2)
        TT3:  initial-claim(TT1)
        Occurs(TT3)
then    TT4:  benefit-period(TT1)
        begin(TT4)←sunday_of(week_of(latest_of(instant(TT2),instant(TT3))))
```

6 Conclusions

We explored the representation of time and temporal information in legal domains within the tradition of using logic to formalize law. We propose a temporal representation framework, called **LTR**, described by the following choices on the temporal reasoing features:

LTR		
Time Ontology	Units:	Instants, periods, durations with clock/calendar forms as constants.
	Relations:	{$\prec$, `begin`, `end`, `Next`, `Previous`, `ImmediateBefore`, `ImmediateAfter`}
Time Theory	$\mathcal{IP}$ axioms + discreteness axioms + $\mathbf{Im}_{1\div4}$ axioms (The axioms are given in appendices A.1 and A.2)	
Temporal Constraints	Combined (metric) Point - Interval Constraints	
Temporal Qualification	*Token arguments*	
Temporal Incidence Theory	TIPs:	{`holds`,`occurs`,`holds_at`,`holds_on`}
	Axioms:	`holds` and `holds_on` homogeneity

Our approach is independent of the underlying representation language and the specific legal reasoning application. We discussed its adequacy wrt. the requirements identified in legal domains. **LTR** is currently being used within a rule-based language in the formalization of the *Convention for International Sale of Goods.*

In this work we did not address the issues of (i) representing periodic occurrences, (ii) temporal non-monotonic reasoning, and (iii) handling time of legal statutes. For instance, tasks that involve meta-reasoning about the validity of statutes and laws over time are out the scope of our approach. It is a focus of our current research.

References

[1] S. Abiteboul and S. Grumbach. A logic-based language for complex objects. In *Proc. of the Intl. Conf. on Extending Database Technology*, 1988.

[2] J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[3] J. Allen and P. Hayes. A common-sense theory of time. In *Proc. IJCAI'85*, pages 528–531, 1985.

[4] J. Allen and P. Hayes. Moments and points in an interval-based temporal logic. *Computational Intelligence*, 5:225–238, 1989.

[5] F. Bacchus, J. Tenenberg, and J. Koomen. A non-reified temporal logic. *Artificial Intelligence*, 52:87–108, 1991.

[6] T. Bench-Capon, G. Robinson, T. Routen, and M. Sergot. Logic programming for large scale applications in law: A formalization of supplementary benefit legislation. In Hayes, Michie, and Richards, editors, *Machine Intelligence*, pages 209–260. Oxford Univ. Press, 1988.

[7] G. Blumsohn. *Three Essays on Law and Information in the Law of Damages.* Dissertation Information Service. UMI, 1991.

[8] A. Bochman. Concerted instant-interval temporal semantics i: Temporal ontologies. *Notre Dame Journal of Formal Logic*, 31(3):403–414, 1990.

[9] C. Brzoska. Temporal logic programming with metric and past operators based on constraint logic programming. In *Proc. of IJCAI'93 Workshop on Executable Modal and Temporal Logics*, pages 67–81, 1993.

[10] E. Bulygin. Time and validity. In A. Martino, editor, *Deontic Logic, Computational Liguistics and Information Systems*, chapter Vol. II, pages 65–81. North-Holland, 1982.

[11] M. Chemilieu-Gendrau. *Le Role du Temps dans la Formation du Droit International.* Droit International. Editions Pedone, 1987.

[12] W. Chen and D. Warren. C-logic of complex objects. In *Proc. of 8th ACM-SIGACT-SIGMOD-SIGART Symposium of Principles of Database Systems*, 1989.

[13] D. Chimenti. The IDL system prototype. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):78–90, 1990.

[14] T. Dean and D. McDermott. Temporal data base management. *Artificial Intelligence*, 32:1987, 1987.

[15] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[16] J. Delgrande and A. Gupta. A representation for efficient temporal reasoning. In *Proc. AAAI'96*, pages 381–388, Menlo Park, 1996. AAAI Press/The MIT Press.

[17] T. Frühwirth. Temporal annotated constraint logic programming. *Journal of Symbolic Computation*, (to appear), 1996.

[18] A. Galton. A critical examination of Allen's theory of action and time. *Artificial Intelligence*, 42:159–188, 1990.

[19] A. Galton. Reified temporal theories and how to unreify them. In *Proc. IJCAI'91*, pages 1177–1182, 1991.

[20] A. Gardner. *An Artificial Intelligence Approach to Legal Reasoning.* MIT Press, 1987.

[21] A. Gerevini and L. Schubert. Efficient algorithms for qualitative reasoning about time. *Artificial Intelligence*, 74(3):207–248, 1995.

[22] M. Ghallab and A. Mounir Alaoui. Managing efficiently temporal relations through indexed spanning trees. In *Proc. IJCAI'89*, pages 1297–1303, 1989.

[23] C. Hamblin. Instants and intervals. In J. Fraser, editor, *The Study of Time*, pages 325–331. Springer-Verlag, 1972.

[24] B. A. Haugh. Non-standard semantics for the method of temporal arguments. In *Proc. IJCAI'87*, pages 449–454, 1987.

[25] T. Hrycej. A temporal extension of prolog. *Journal of Logic Programming*, 15:113–145, 1993.

[26] H. Kautz and P. Ladkin. Integrating metric and qualitative temporal reasoning. In *Proc. AAAI'91*, pages 241–246, 1991.

[27] M. Kifer and G. Lausen. F-logic: A higher-order language for reasoning about objects. In *Proc. of 8th ACM-SIGACT-SIGMOD-SIGART Symposium of Principles of Database Systems*, 1989.

[28] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 3, 1986.

[29] G. Kuper. Logic programming with sets. In *Proc. of 6th ACM-SIGACT-SIGMOD-SIGART Symposium of Principles of Database Systems*, 1987.

[30] E. Mackaay, D. Poulin, J. Fremont, P. Bratley, and C. Deniger. The logic of time in law and legal expert systems. *Ratio Juris*, 3(2):254–271, 1990.

[31] D. Maier. A logic for objects. In *Proc. of the Workshop on Foundations of Deductive Database and Logic Programming*, 1986.

[32] L. T. McCarty. Some requirements on an action language for legal discourse (position paper). In *Spring Symposium Series'95: Extending Theories of Action*, pages 136–138. AAAI, 1995.

[33] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

[34] I. Meiri. Combining qualitative and quantitative constraints in temporal reasoning. In *Proc. AAAI'91*, 1991.

[35] R. Morris, W. Shoaf, and L. Khatib. Domain independent temporal reasoning with recurring events. *Computational Intelligence*, 1994.

[36] W. Newton-Smith. *The Structure of Time*. Routledge & Heagan Paul, 1980.

[37] K. Nitta, J. Nagao, and M. Tetsuya. A knowledge representation and inference system for procedural law. *New Generation Computing*, 5:319–359, 1988.

[38] D. Poulin, E. Mackaay, P. Bratley, and J. Fremont. Time server - a legal time specialist. In A. Martino, editor, *Expert Systems in Law*, pages 295–312, 1992.

[39] B. Schrag and M. Boddy. β-tmm functional description. Technical report, Honeywell SRC, 1991.

[40] E. Schwalb and R. Dechter. Processing temporal constraint networks. *Artificial Intelligence*, (to appear), 1997.

[41] E. Schwalb, L. Vila, and R. Dechter. Temporal constraint logic programming. (in preparation), UC Irvine, 1996.

[42] M. Sergot. Representing legislation as logic programs. In Hayes, Michie, and Richards, editors, *Machine Intelligence*, pages 209–260. Oxford Univ. Press, 1988.

[43] M. Sergot. Tutorial notes: Using logic for knowledge representation in legal knolwedge based system. 5th International Conference on Artificial Intelligence in Law, Maryland, May 1995.

[44] Y. Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, 33:89–104, 1987.

[45] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases*. Benjamin/Cummings Publishing, 1993.

[46] E. Tsang. Time structures for AI. In *Proc. IJCAI'87*, pages 456–461, 1987.

[47] P. van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326, 1992.

[48] J. van Benthem. *The Logic of Time*. Kluwer Academic, Dordrecht, 1983.

[49] L. Vila. *IP*: An instant-period–based theory of time. In R. Rodriguez, editor, *Proc. ECAI'94 Workshop on Spatial and Temporal Reasoning*, 1994.

[50] L. Vila. A survey on temporal reasoning in artificial intelligence. *AI Communications*, 7(1):4–28, March 1994.

[51] L. Vila and H. Reichgelt. The token reification approach to temporal reasoning. *Artificial Intelligence*, 83(1):59–74, May 1996.

[52] L. Vila and E. Schwalb. A theory of time and temporal incidence based on instants and periods. In *Proc. of the Intl. Workshop on Temporal Representation and Reasoning (TIME'96)*, pages 21–28, 1996.

[53] L. Vila and H. Yoshino. Formalizing time in the united nations convention for internation sale of goods. Technical Report 96-55, UC Irvine, 1996.

[54] L. Vila and H. Yoshino. Time in automated legal reasoning (the long report). Technical Report 96-57, UC Irvine, 1996.

[55] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In D. S. Weld and J. de Kleer, editors, *Readings on Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kauffman, 1989.

[56] A. Walker. Durées et instants. *Revue Scientifique*, 85:131–134, 1948.

[57] R. Wetprasit, A. Sattar, and L. Khatib. Reasoning with sequences of events (an extended abstract). In *Proc. of the Intl. Workshop on Temporal Representation and Reasoning (TIME'96)*, pages 36–38, 1996.

[58] H. Yoshino. Representation of legal knowledge by compound predicate formula. In D. T. C. Biagioli, G. Sartor, editor, *Proc. of the Workshop on Legal Application of Logic Programming*, pages 128–137. ICPL'94, 1994.

[59] H. Yoshino. Representation of legal knowledge by legal flowchart and compound predicate formula. Technical Report TM-1298, ICOT, 1994.

A A Discrete Theory of Time

A.1 IP Theory

$\mathcal{IP}$ is defined upon a structure composed of two sorts of symbols, instants ($\mathcal{I}$) and periods ($\mathcal{P}$) which are formed by two infinite disjoint sets of symbols, and three primitive binary relation symbols $\prec: \mathcal{I} \times \mathcal{I}$ and $\mathsf{begin}, \mathsf{end} : \mathcal{I} \times \mathcal{P}$.

The first order axiomatization of $\mathcal{IP}$ is as follows:

$\mathbf{IP}_1$	$\neg(i \prec i)$	$\mathbf{IP}_{7.1}$	$\exists i\ \mathsf{begin}(i,p)$
$\mathbf{IP}_2$	$i \prec i' \Rightarrow \neg(i' \prec i)$	$\mathbf{IP}_{7.2}$	$\exists i\ \mathsf{end}(i,p)$
$\mathbf{IP}_3$	$i \prec i' \wedge i' \prec i'' \Rightarrow i \prec i''$	$\mathbf{IP}_{8.1}$	$\mathsf{begin}(i,p) \wedge \mathsf{begin}(i',p) \Rightarrow i = i'$
$\mathbf{IP}_4$	$i \prec i' \vee i \prec i' \vee i = i'$	$\mathbf{IP}_{8.2}$	$\mathsf{end}(i,p) \wedge \mathsf{end}(i',p) \Rightarrow i = i'$
$\mathbf{IP}_{5.1}$	$\exists i'\ (i' \prec i)$	$\mathbf{IP}_9$	$i \prec i' \Rightarrow \exists\ p\ (\mathsf{begin}(i,p) \wedge \mathsf{end}(i',p))$
$\mathbf{IP}_{5.2}$	$\exists i'\ (i \prec i')$	$\mathbf{IP}_{10}$	$\mathsf{begin}(i,p) \wedge \mathsf{end}(i',p) \wedge$
$\mathbf{IP}_6$	$\mathsf{begin}(i,p) \wedge \mathsf{end}(i',p) \Rightarrow i \prec i'$		$\wedge\ \mathsf{begin}(i,p') \wedge \mathsf{end}(i',p') \Rightarrow p = p'$

$\mathbf{IP}_1 \div \mathbf{IP}_4$ are the conditions for $\prec$ to be a *strict linear order* –namely irreflexive, asymmetric, transitive and linear– relation over the instants[21]. $\mathbf{IP}_5$ imposes unboundness on this ordered set. $\mathbf{IP}_6$ orders the extremes of a period. This axiom rules out durationless periods which are not necessary since we have instants as a primitive. The pairs of axioms $\mathbf{IP}_{7.\_}$ and $\mathbf{IP}_{8.\_}$ formalize the intuition that the beginning and end instants of a period always *exist* and are *unique* respectively. Conversely, axioms $\mathbf{IP}_9$ and $\mathbf{IP}_{10}$ close the connection between instants and periods by ensuring the *existence* and *uniqueness* of a period for a given ordered pair of instants.

See [49] for a characterization of the models and relation with other time theories.

A.2 Discreteness Axioms

The discreteness axioms under an unbounded time are as follows:

$\mathbf{IP}_{\mathrm{di1}}$	$i\ \mathtt{Previous}\ i' \Leftrightarrow i'\ \mathtt{Next}\ i$
$\mathbf{IP}_{\mathrm{di2}}$	$i\ \mathtt{Previous}\ i' \Rightarrow i \prec i'$
$\mathbf{IP}_{\mathrm{di3}}$	$\exists i'\ i\ \mathtt{Previous}\ i'$
$\mathbf{IP}_{\mathrm{di3'}}$	$\exists i'\ i\ \mathtt{Next}\ i'$
$\mathbf{IP}_{\mathrm{di4}}$	$i\ \mathtt{Previous}\ i' \Rightarrow \neg\exists i''\ (i \prec i'' \prec i')$

B LTR Syntax

Sorts

- Temporal sorts = {instants, periods, durations}
- Token sorts = {fluent token, event token}

[21]Notice that $\mathbf{IP}_1$ is actually redundant since it can be derived from $\mathbf{IP}_2$. We include it for clarity.

B.1 Constants

Clock/Calendar Constants

date	::=	[*second*''][*minute*'][*hour*h][*day*d][*week*w][*month*m][*year*y]
	|	*second*: *minute*: *hour*
	|	*day*/ *month*/ *year*

Instant and Duration Constants

instant-constant	::=	*date* | `i1` | `i2` | ...
duration-constant	::=	*date* | `d1` | `d2` | ...

The dates allowed as instant and duration constants are dynamically determined in accordance with the granularity declared in the application.

Period Constants

period-constant	::=	*left-bracket date, date right-bracket* | `p1` | `p2` | ...
left-bracket	::=	`(` | `[`
right-bracket	::=	`)` | `]`

The dates allowed as period constants are dynamically determined in accordance with the granularity declared in the application.

Token constants

token-constant ::= `tt1` | `tt2` ...

B.2 Temporal Operators

The following is a representative, non-complete list of temporal operators:

Date arithmetics

`+,-` : *date* $\times$ *date* $\mapsto$ *date*

Date predicates

`Is_holiday` : *date* $\mapsto$ {*t*/*f*}

Date operations

previous_holiday :	*date*	$\mapsto$	*day*
next_holiday :	*date*	$\mapsto$	*day*

Date transformations

minute_of :	*date*	$\mapsto$	*minute*
hour_of :	*date*	$\mapsto$	*hour*
day_of :	*date*	$\mapsto$	*day*
week_of :	*date*	$\mapsto$	*week*
month_of :	*date*	$\mapsto$	*month*
year_of :	*date*	$\mapsto$	*year*

Date sets operations

first,latest,shortest :	*date-set*	$\mapsto$	*date*
nth :	*natural-number times date-set*	$\mapsto$	*date*

For the sake of syntax definition, the terms resulting from the application of the above operators is regarded as a date constant.

B.3 Temporal Functions

Begin, End :	*period*	$\mapsto$	*instant*
[],(),[),(] :	*instant* $\times$ *instant*	$\mapsto$	*period*
Duration :	*period*	$\mapsto$	*duration*
- :	*instant* $\times$ *instant*	$\mapsto$	*duration*

B.4 Token Temporal Functions

Begin,End,instant:	*token*	$\mapsto$	*instant*
period:	*token*	$\mapsto$	*period*
duration:	*token*	$\mapsto$	*duration*

B.5 Temporal Relations

qualitative-point-relation	::=	< | = | >
qualitative-point-interval-relation	::=	< | begin | $\in$ | end >
qualitative-interval-relation	::=	Before | Meets | Equal | Met_by | After | During | Contains | Overlaps | Overlapped_by | Starts | Started_by | Finishes | Finished_by

B.6 Temporal Terms

instant-term	::=	*instant-constant* | {Begin| End}(*period-term*) | {begin| end}(*token-term*)
period-term	::=	*left-bracket instant-term, instant-term right-bracket* | period(*token-term*)
duration-term	::=	duration(*period-term*) | *instant-term*−*instant-term*

B.7 Token Terms

token-term	::=	*fluent-token-term* | *event-token-term*
fluent-token-term	::=	*token-constant* | *fluent-token-function*(...)
event-token-term	::=	*token-constant* | *event-token-function*(...)

B.8 Temporal Constraints

temporal-constraint	::=	*q-instant-constraint*
	|	*m-instant-constraint*
	|	*period-constraint*
	|	*instant-period-constraint*
	|	*unary-duration-constraint*
q-instant-constraint	::=	*instant-term point-algebra-rel instant-term*
point-algebra-rel	::=	$<, =, >, \leq, \geq, \neq$
m-instant-constraint	::=	*instant-term* $\in$ { [[*duration-constant*, *duration-constant*]]$^+$ }
period-constraint	::=	*period-term interval-algebra-rel period-term*
interval-algebra-rel	::=	$\mathcal{P}$*(qualitative-interval-relation)*
instant-period-constraint	::=	*instant-term point-interval-algebra-rel period-term*
point-interval-algebra-rel	::=	$\mathcal{P}$(Before,Begin,$\in$,End,After})
unary-duration-constraint	::=	*duration-term* $\in$ { [[*duration-constant*, *duration-constant*]]$^+$ }

$\mathcal{P}$ denotes a disjunction formed with the elements of the power set of its arguments.

B.9 Token Atoms

token-atom ::= *token-term* : *token-type*

where a *token-type* is of the form *relation* (att_1, ..., att_n) where *att_i* is either a *token-term*, a *token-type* or a *non-temporal term*.

B.10 Incidence Atoms

```
incidence-atom  ::=  holds(fluent-token-term)
                  |  occurs(event-token-term)
                  |  holds_on(token-type, period-term)
                  |  holds_at(token-type, instant-term)
```