Provenance and Temporally Annotated Logic Programming

Anastasia Analyti
Institute of Computer Science, FORTH-ICS,
Heraklion, Greece

Ioannis Pachoulakis
Dept. of Applied Informatics & Multimedia, TEI of Crete,
Heraklion, Greece

***Abstract*— In this paper, we consider provenance and temporally annotated logic rules (pt-logic rules, for short), which are definite logic programming rules associated with the name of the source that they originate and the temporal interval during which they are valid. A collection of pt-logic rules form a provenance and temporally annotated logic program P, called pt-logic program, for short. We develop a model theory for P and define its maximally temporal entailments of the form A:<S, ti>, indicating that atom A is derived from a set of sources S and holds at a maximal temporal interval ti, according to S. We define a consequence operator that derives exactly the maximally temporal entailments of P for a set of sources. We show that the complexity of the considered entailment is EXPTIME-complete.**

Keywords- Annotated logic programming; provenance and temporal information; model theory; consequence operator.

I. INTRODUCTION

Definite logic programming rules traditionally are not associated with the name of the source (provenance information) from which they originate and the temporal interval during which are valid. However, logic programming rules are usually derived from different sources that interact. Additionally logic programming rules may not always be valid, but be valid only for a specific temporal interval.

A temporal interval has the form $[t,t']$, where t,t' are time points and $t \leq t'$. In this paper, time points are years. However, this assumption can be generalized and we may assume any set of time points that can be mapped one-to-one to the set natural numbers. We consider definite logic programming rules associated with a source name and a validity temporal interval, called *provenance and temporally annotated logic rules*, or *pt-logic rules* for short.

We assume that *pt*-logic rules are applied similarly to definite logic programming rules but at each application new provenance and temporal information is derived for the derived atom *A*. In particular, derived atoms (*pt*-atoms) have the form $A:\langle S, ti\rangle$, where *ti* is the temporal interval at which *A* is valid, as derived from a set of sources *S*.

Obviously, if $A:\langle S, ti\rangle$ is true then $A:\langle S', ti\rangle$ is true, where $S \subseteq S'$ and S' is a subset of a set of considered source names. Additionally, if $A:\langle S, ti\rangle$ and $A:\langle S', ti'\rangle$ are true, where temporal intervals ti, ti' are overlapping or consecutive then $A:\langle S \cup S', ti''\rangle$ is true, where ti'' is the combination of ti and ti'.

A collection of *pt*-logic rules form a *provenance and temporally annotated logic program*, called *pt*-logic program, for short. The models of *pt*-logic programs *P* are defined, as well as, the simple entailments and temporally maximal entailments of *P*. We show that *P* has a minimal model containing exactly the temporally maximal entailments of *P*.

A set of three operators are defined which are applied on *pt*-atoms such that the closure of their composition derives the minimal model of *P*. We define a query language that consists of simple and composite queries, querying provenance and temporal information of derived *pt*-atoms based on certain conditions.

We show that the complexity of simple entailment of a *pt*-atom or temporally maximal entailment of a *pt*-atom from *P* is EXPTIME-complete.

The rest of the paper is organized as follows: In Section II, we define *pt*-logic rules, *pt*-logic programs, *pt*-atoms, and the instantiation of a *pt*-logic program. In Section III, we provide a model theory for *pt*-logic programs *P* and the minimal model of *P* is defined.

In Section IV, we define a consequence operator deriving the minimal model of *P*. In Section V, a query language for *pt*-logic programs is defined. Section VI contains related work. Finally, Section VII contains directions for future work.

II. PROVENANCE AND TEMPORALLY ANNOTATED LOGIC PROGRAMS

In this Section, we define provenance and temporally annotated logic programs *P*. Additionally, we define the rules based on which the models of *P* are defined. We consider a set of variables *Var*, all preceded by the question mark symbol "?".

Definition 1. A *provenance and temporally annotated logic* rule, called *pt-logic rule* for short, is a definite logic programming rule *r* without function symbols, associated with a source name *nam* and a temporal interval *ti*. In particular, it has the form $\langle nam, ti\rangle: r$.

Definition 2. A provenance and temporally annotated logic program P, called pt-logic program for short, is a set of pt-logic rules.

Example 1. The following is a *pt*-logic program *P*.

<*Person*, [1990,1994]>: *has_job*(*Mary*,*Hairdresser*).
<*Person*, [1995,2002]>: *has_job(Mary*,*Secretaty*).
<*Person*, [2006,2009]>: *has_job*(*Mary*,*Hairdresser*).
<*Person*,[1999,2002]> : *extra_vacation*(*Mary*).
<*Person*, [2001,2003]>: *has_job*(*Peter*, *Garbage_collector*).
<*Person*, [2005,2008]>: *has_job*(*Peter*,*Bulider*).
<*Job*, [1980,1992]>: *heavy_job*(*Hairdresser*).
<*Job*, [1980, 2001]>: *heavy_job(Garbage_collector*).
<*Job*, [2006, 2012]>: *heavy_job(Builder*).
<*Vacation*, [1988,2012]: *extra_vacation*(?*x*) ← *has_job*(?*x*,?*y*), *heavy_job*(?*y*).

Predicate *has_job* indicates the job of a person. Predicate *heavy_job* indicates that a job is heavy and unhealthy. Finally, predicate *extra_vacation* indicates that a person gets for some reason extra vacation days.

C*onvention*: In the following by *P*, we will denote a *pt*-logic program.

We define $Names_P$ to be the set of source names appearing in *P*. Additionally, we define *definite*(*P*) to be the definite logic program derived from *P* after ignoring the provenance and temporal annotations of *P*. Further, we denote by $t_P^{\min}$ the minimum temporal point appearing in *P* and by $t_P^{\max}$ the maximum temporal point appearing in *P*.

Definition 3. A *pt-atom* of *P* is an atom *A* built using predicates and constants appearing in *definite*(*P*) and associated with (i) a set of source names that is a subset of $Names_P$ or a provenance variable *S* and (ii) a temporal interval within $[t_P^{\min}, t_P^{\max}]$ or a temporal variable *ti*. In particular, it has the form *A*: <*S*, *ti*>.

Below, we define the set of rules [*P*], built from ground *pt*-atoms of *P*, based on which the models of *P* are defined.

Definition 4. Let $r \in P$ be a *pt*-rule <*nam*, *ti*>: $A_0 \leftarrow A_1, \ldots, A_n$. We define $[r]_P$ to be the rule B_0:<S_0, ti_0> ← B_1:<S_1, ti_1>, …, B_n :<S_n, ti_n>, where (i) B_0 is derived from A_i by replacing all the variables in A_i by constants in *definite*(*P*), (ii) $S_i \subseteq Names_P$ s.t. S_0 is the union of the sets $S_1, \ldots, S_n$ and {*nam*}, and (iii) ti_i is a temporal interval within $[t_P^{\min}, t_P^{\max}]$ s.t. i_0 is the intersection of the temporal intervals $ti_1, \ldots, ti_n$ and *ti*. We define the *instantiation* of *P*, as follows:

$$[P] = \bigcup_{r \in P} [r]_P .$$

III. Model Theory of pt-Logic Programs

In this Section, we present the model theory of *pt*-logic programs and entailment of *pt*-atoms.

First, we present three auxiliary definitions. We say that two temporal intervals *ti* and *ti'* are *overlapping* if they have at least one common time point. We say that two temporal intervals $[t_1,t_2]$ and $[t_3,t_4]$ are *consecutive* if $t_3=t_2+1$. We say that a temporal interval $[t_1,t_2]$ is *included* in a temporal interval $[t_3,t_4]$ if $t_3 \leq t_1$ and $t_2 \leq t_4$.

Definition 4. An *interpretation I* of *P* is a set of ground *pt*-atoms of *P* s.t. (i) if *A*:<*S*, *ti*>, *A*:<*S*, *ti'*> $\in I$ then the temporal intervals *ti*, *ti'* are neither overlapping nor consecutive and (ii) if *A*:<*S* ,*ti*> $\in I$ then, for all *S'* s.t. $S \subseteq S' \subseteq Names_P$, there exists *A*:<*S'*, *ti'*> $\in I$ s.t. *ti* is included in *ti'*.

Below, we define entailment of a *pt*-atom for an interpretation *I* of *P*.

Definition 5. Let *I* be an interpretation of *P* and let *A*:<*S*, *ti*> be a ground *pt*-atom of *P*. We say that *I* (*simply*) *entails* *A*:<*S*, *ti*>, denoted *by I* |= *A*:<*S*, *ti*>, if there exists an *A*:<*S*, *ti'*> $\in I$ s.t. *ti* is included in *ti'*.

Additionally, we define an ordering on the interpretations of *P*. Let *I*, *I'* be interpretations of *P*. We say that $I \leq I'$ iff for each *A*:<*S*, *ti*> $\in I$ there exists an *A*:<*S*, *ti'*> $\in I$ s.t. *ti* is included in *ti'*. It is easy to see that if $I \leq I'$ and $I' \leq I$ then $I = I'$.

Below, we define the models if a *pt*-logic program *P*,

Definition 6. Let *M* be an interpretation of *P*. We say that *M* is a *model* of *P* if for each *r*= A_0:<S_0, ti_0> ← A_1:<S_1, ti_1>, …, A_n:<S_n, ti_n> $\in [P]$, it holds that: if *M* |= A_i:<S_i, ti_i>, for all *i=1,...,n*, then *M* |= A_0:<S_0, ti_0>. We denote the set of models of *P* by $\mathcal{M}_P$.

Let *ti* = [*t*,*t'*] be a temporal interval. We define *start*(*ti*)=*t* and *end*(*ti*)=*t'*.

Below, we define simple entailment of a ground *pt*-atom from a *pt*-logic program *P*, as well as maximally temporal entailment.

Definition 7. We say that P (simply) entails a ground pt-atom A:<S, ti>, denoted by P |= A:<S, ti>, iff for each M $\in \mathcal{M}_P$, M |= A:<S, ti>. We say that P maximally temporally entails a ground pt-atom A:<S, ti>, denoted by P $|=^{\max}$ A:<S, ti> iff (i) P |= A:<S, ti>, (ii) there exists M $\in \mathcal{M}_P$ s.t. M does not entail A:<S, [end(ti)+1, end(ti)+1]>, and (iii) there exists M $\in \mathcal{M}_P$ s.t. M does not entail A:<S, [start(ti)-1, start(ti)-1]>.

Below, we define the minimal model of a *pt*-logic program *P* w.r.t. ≤.

Definition 8. Let $M_1,..,M_n$ be all the models of *P*, we define the interpretation M_{min}={ *A*:<*S*, *ti*> | there exist *A*:<*S*, ti_i> $\in M_i$, for all *i=1,…,n,* and *ti* is the intersection of $ti_1,\ldots, ti_n$}.

Below, we show that M_{min} is an interpretation of *P*.

Proposition 1. M_{min} is an interpretation of *P*.

Proof: Obviously, M_{min} is a set of *pt*-atoms of *P*. Let $M_1,..,M_n$ be all the models of *P*. If *A*:<*S*, *ti*> $\in M_{min}$ then there exists *A*:<*S*, ti_i> $\in M_i$, for all *i=1,…,n,* and *ti* is the intersection of $ti_1, \ldots, ti_n$. Since $M_1,..,M_n$ interpretations of *P*, it holds that if *A*:<*S* ,*ti*> $\in M_{min}$ then, for all *S'* s.t. $S \subseteq S' \subseteq Names_P$, there exists *A*:<*S'*, *ti'*> $\in M_{min}$ s.t. *ti* is included in *ti'*. Additionally, since $M_1,..,M_n$ interpretations of *P*, there are no *A*:<*S*, *ti*>, *A*:<*S*, *ti'*> $\in M_{min}$ s.t. temporal intervals *ti, ti'* are overlapping or consecutive.

Obviously, $M_{min} \leq M$, for all M $\in \mathcal{M}_P$.

Below, we show that M_{min} is a minimal model of *P*.

Proposition 2. M_{min} is a minimal model of P.

Proof: Let $M_1,..,M_k$ be all the models of P and let r= A_0:<S_0, ti_0> ← A_1:<S_1, ti_1>, …, A_n:<S_n, ti_n>∈[P]. Assume that M_{min} |= A_i:<S_i, ti_i>, for all i=1,..n. Then, M_j |= A_i:<S_i, ti_i>, for all j=1,..,k and i=1,…,n. Thus, M_j |= A_o:<S_0, ti_0>, for all j=1,…,k. Therefore, M_{min} |= A_o:<S_0, ti_0>.

In Proposition 3, below, we show that P and M_{min} have the same simple entailments.

Proposition 3. Let A:<S, ti> be a ground *pt*-atom. It holds that: P |= A:<S, ti> iff M_{min} |= A:<S, ti>.

Proof:

=>) Let P |= A:<S, ti>. Then, for each $M \in \mathcal{M}_P$, M |= A:<S, ti>. This means that for each $M \in \mathcal{M}_P$, there exists A:<S, ti'>∈M s.t. ti is included in ti'. Thus, by the definition of M_{min}, there exists A:<S, ti''>∈ M_{min} s.t. ti is included in ti''. Thus, M_{min} |= A:<S, ti>.

<=) Assume that M_{min} |= A:<S, ti>. Then, there exists A:<S, ti'>∈ M_{min} s.t. ti is included in ti'. Thus, by the definition of M_{min}, for all $M \in \mathcal{M}_P$, there is A:<S, ti''>∈M s.t. ti is included in ti''. Therefore, P |= A:<S, ti>.

In Proposition 4, below, we show that M_{min} contains exactly the maximally temporal entailments of P.

Proposition 4. Let A:<S, ti> be a ground *pt*-atom. It holds that: P |=^{max} A:<S, ti> iff A:<S, ti>∈M_{min}.

Proof:

=>) Let P |=^{max} A:<S, ti>. This means that for each $M \in \mathcal{M}_P$, there exists A:<S, ti'>∈M s.t. ti is included in ti'. Additionally, there exists $M \in \mathcal{M}_P$, that does not contain A:<S, ti'>∈M s.t. time point $end(ti)$+1 is contained in ti' and there exists $M \in \mathcal{M}_P$, that does not contain A:<S, ti'>∈M s.t. time point $start(ti)$-1 is contained in ti'. Therefore, A:<S, ti>∈ M_{min}.

<=) Assume that A:<S, ti>∈M_{min}. This means that for each $M \in \mathcal{M}_P$, there exists A:<S, ti'>∈M s.t. ti is included in ti'. Additionally, there exists $M \in \mathcal{M}_P$, that does not contain A:<S, ti'>∈M s.t. time point $end(ti)$+1 is contained in ti' and there exists $M \in M_P$, that does not contain A:<S, ti'>∈M s.t. time point $start(ti)$-1 is contained in ti'. Therefore, P |=^{max} A:<S, ti>.

IV. Computing the Minimal Model of a pt-program

In Section, we provide three operators s.t. the closure of their composition provides the minimal model of a *pt*-logic program P.

We denote by Q_P the ground *pt*-atoms of P. We define the operator W_P from the powerset of Q_P to the powerset Q_P, as follows:

$$W_P(Q)=\{\ A:<S, ti> \mid \text{it exists } A:<S, ti> \leftarrow A_1:<S_1, ti_1>, \ldots, A_n:<S_n, ti_n> \in [P] \text{ s.t. } <S_i, ti_i> \in Q, \text{ for all } i=1,\ldots,n\}$$

We now define the operator Z_P from the powerset of Q_P to the powerset of Q_P, as follows:

$$Z_P(Q)=\{\ A:<S, ti> \mid \text{it exists } A:<S', ti> \in Q \text{ and } S' \subseteq S \subseteq Names_P\}$$

Before, we define the third operator, we provide a few definitions. Let $Q \subseteq Q_P$. We define $intervals(Q,S,A)=\{ti \mid$ it exists A:<S, ti> $\in Q\}$. Let TI be a set of temporal intervals. The *maximal subset* of TI w.r.t. a temporal interval $ti \in TI$ is a set B that (i) contains ti and (ii) if $ti' \in B$ and there exists a $ti'' \in TI$ s.t. ti' and ti'' are overlapping or consecutive then $ti'' \in B$. The *maximal interval* of TI w.r.t. a temporal interval $ti \in TI$, denoted by $max\_interval(TI,ti)$, is the temporal interval $[t,t']$ formed by the minimum time point t and maximum time point t' appearing in a maximal subset of TI w.r.t. ti.

We are now ready to define the operator R_P from the powerset of Q_P to the powerset of Q_P as follows:

$$R_P(Q)=\{\ A:<S, ti> \mid \text{it exists } A:<S, ti'> \in Q \text{ and } ti = max\_interval(intervals(Q,S,A), ti')\ \}$$

Finally, we define the consequence operator T_P from the powerset of Q_P to the powerset of Q_P as the composition of W_P, Z_P, and R_P. In particular, $T_P(Q)= R_P(Z_P(W_P(Q)))$, for $Q \subseteq Q_P$.

It can be easily seen that the consequence operator T_P is monotonic with respect to ≤. That is, if $Q,Q' \subseteq Q_P$ s.t. $Q \subseteq Q'$ then $T_P(Q) \leq T_P(Q')$. We will show that the closure of operator T_P coincides with M_{min}.

Proposition 4. Let M be an interpretation of P. It holds that M is a model of P iff $T_P(M) \leq M$.

Proof:

=>) Let M be a model of P. Consider the rules A:<S, ti> ← A_1:<S_1, ti_1>, …, A_n:<S_n, ti_n>∈[P] s.t. A_i:<S_i, ti_i>∈M, for all $i=1,\ldots,n$. Then, M |= A:<S, ti>. Since M is an interpretation of P, M satisfies each *pt*-atom in $T_P(M)$. Thus, $T_P(M) \leq M$.

<=) Assume that $T_P(M) \leq M$. We need to show that if A:<S, ti> ← A_1:<S_1, ti_1>, …, A_n:<S_n, ti_n>∈[P] s.t. M |=A_i:<S_i, ti_i>, for all i=1,…,n, then M |=A:<S, ti>. Note that there exists A:<S, ti'> ← A_1:<S_1, ti'$_1$>, …, A_n:<S_n, ti'$_n$>∈[P] s.t. A_i:<S_i, ti'$_i$>∈M, for all i=1,…,n.. Note that $T_P(M)$ is an interpretation of P and that $T_P(M)$ |= A:<S, ti'>. Since $T_P(M) \leq M$, it follows that M |= A:<S, ti'>. Now, since ti is included in ti', it follows that M |= A:<S, ti>. Thus, M is a model of P.

Below, we show that M_{min} can be computed as the closure of the T_P operator.

Proposition 5. $M_{min}=T_P^{\uparrow\omega}(\{\})$.

Proof: From Proposition 4, it follows that $M_{min}= minimal_{\leq}(\{M \mid M$ is an interpretation of P and $T_P(M) \leq M\})$. Since M_{min} is a model of P, it follows from Proposition 4 that $T_P(M_{min}) \leq M_{min}$. Further note that since the operator T_P is monotonic, it follows that $T_P(T_P(M_{min})) \leq T_P(M_{min})$. Additionally, $T_P(M_{min})$ is an interpretation of P. Thus, $M_{min} \leq T_P(M_{min})$. Therefore, $T_P(M_{min})=M_{min}$. Thus, $M_{min}=T_P^{\uparrow\omega}(\{\})$. Note that $T_P^{\uparrow\omega} \subseteq Q_P$.

Proposition 6. Simple and maximally temporal entailment of a ground *pt*-atom from a *pt*-logic program P is EXPTIME-complete w.r.t. the size of P.

Proof: *Membership*) Note that the computation of $M_{min}=T_P^{\uparrow\omega}(\{\})$, involves the application of at most an exponential number of rules applied at most exponential number of times. Obviously, the computation of *max_interval*(*intervals*(*Q,S,A*),*ti'*) is polynomial w.r.t. the size of *Q*. Therefore, the computation of M_{min} is in EXPTIME.

Hardness) It follows directly by the fact that datalog is program-complete for EXPTIME [1].

V. A Query Language for pt-logic Programs

Below, we define the queries that can be applied to a *pt*-logic program *P*.

A simple *pt*-query of type 1 has the form *SQ*=*A*:<?*S*, *ti*>, where *A*:<?*S*, *ti*> is a *pt*-atom of *P*, *ti* is a temporal interval, and ?*S* is a provenance variable. The answers of *SQ* w.r.t. *P*, denoted by $Ans_P(SQ)$, is the set of mappings *v* from the variables of *A* to the constants of *definite*(*P*) and from ?*S* to a subset of $Names_P$ s.t. $P \models v(A:<?S, ti>)$.

A simple *pt*-query of type 2 has the form *SQ*=*A*:<?*S*, ?*ti*>, where *A*:<?*S*, ?*ti* > is a *pt*-atom of *P* (note that ?*S* is a provenance variable and ?*ti* is a temporal variable). The answers of *SQ* w.r.t. *P*, denoted by $Ans_P(SQ)$, is the set of mappings *v* from the variables of *A* to the constants of *definite*(*P*), from ?*S* to a subset of $Names_P$, and from ?*ti* to a temporal interval s.t. $P \models^{\max} v(A:<?S, ?ti>)$.

A simple *pt*-query of type 3 has the form $SQ=A_1:<?S_1, ?ti> \wedge \ldots \wedge A_n:<?S_n, ?ti>$ where each $A_i:<?S_i, ?ti >$ is a *pt*-atom of *P*. The answers of *SQ* w.r.t. *P*, denoted by $Ans_P(SQ)$, is the set of mappings *v* s.t. if $u_i \in Ans_P(A_i:<?S_i, ?ti>)$ s.t. $u_1,\ldots,u_n$ coincide on the common variables of A_i and $?S_i$ then *v* coincides with u_i on the variables of A_i and $?S_i$ and *v*(?*ti*) is the intersection of $u_1(?ti),\ldots, u_n(?ti)$, if such intersection exists.

A simple *pt*-query of type 4 has the form $SQ=A_1<?S_1, ?ti> \wedge \ldots \wedge A_n:<?S_n, ?ti> \wedge included(?ti, [t,t'])$, where each $A_i:<?S_i, ?ti >$ is a *pt*-atom of *P* and [*t,t'*] is a temporal interval. The answers of *SQ* w.r.t. *P*, denoted by $Ans_P(SQ)$, is the set of mappings *v* s.t. if $u_i \in Ans_P(A_i:<?S_i, ?ti>)$ s.t. $u_1,\ldots,u_n$ coincide on the common variables of A_i and $?S_i$ then *v* coincides with u_i on the variables of A_i and $?S_i$ and *v*(?*ti*) is the intersection of $u_1(?ti),\ldots, u_n(?ti)$ and [*t,t'*], if such intersection exists.

A complex *pt*-query has the form $CQ=SQ_1 \wedge \ldots \wedge SQ_n \wedge$ *filter*, where SQ_i are simple *pt*-queries, each having a different temporal variable, and *filter* is an expression of the following EBNF grammar:

`term`:=*duration*(?*ti*) | *start*(?*ti*) | *end*(?*ti*) | *c*, where ?*ti* is a temporal variable and *c* is a decimal.

`complex_term`:= `term` | `complex_term` (+|-|*|/) `complex_term`

`temp_comparison`:= `complex_term` (< | > | = | ≤ | ≥ | ≠) `complex_term`

`prov_comparison`:= (?*S* (⊂|⊃|⊆|⊇|=) ?*S'*) | (?*S* (⊂|⊃|⊆|⊇|=) *S''*) , where ?*S*, ?*S'* are provenance variables and $S'' \subseteq Names_P$.

`filter`:= `temp_comparison` | `prov_comparison` | *filter* (∧|∨) `filter`,

such that each provenance and temporal variable appearing in *filter* appears in $SQ_1 \wedge \ldots \wedge SQ_n$.

The answers of *CQ* w.r.t. *P*, denoted by $Ans_P(CQ)$, is the set of mappings *v* s.t. (i) if $u_i \in Ans_P(SQ_i)$ s.t. $u_1,\ldots,u_n$ coincide on the common variables of SQ_i then *v* coincides with u_i on the variables of SQ_i and (ii) *v*(*filter*) holds.

Consider a simple or complex query *CQ*. Let $v,u \in Ans_P(CQ)$. We say that answer *v* is *more informative* than answer *u*, denoted by $v \leq u$, if (i) *v*, *u* coincide on their no provenance variable mappings, and (ii) for each provenance variable $?S \in domain(v)$, it holds that $v(?S) \subseteq u(?S)$. We define $Answer_P(CQ)=minimal_{\leq}(Ans_P(CQ))$ and we consider that these are the *desired answers*. This is because if $P \models A:<S, ti>$ then $P \models A:<S', ti>$, for each $S \subseteq S' \subseteq Names_P$.

Example 2. Consider the simple query *SQ* = *extra_vacation*(*Peter*): <?*S*, ?*ti*>. Then, the answers to this query is (i) the mapping *v*, where *v*(?*S*)={*Person, Job, Vacation*} and *v*(?*ti*)= [2001,2001], and (ii) the mapping *u*, where *u*(?*S*)={*Person, Job, Vacation*} and *u*(?*ti*)=[2006,2008].

Consider now the simple query *SQ* = *extra_vacation*(*Mary*):<?*S*, ?*ti*>. Then, the answers to this query is (i) the mapping *v*, where *v*(?*S*)={*Person*} and *v*(?*ti*)= [1999,2002], and (ii) the mapping *u*, where *u*(?*S*)={*Person, Job, Vacation*} and *u*(?*ti*)=[1900,1992].

Consider now the simple query *SQ* = *extra_vacation*(*Mary*):<?*S*, ?*ti*> ∧ *extra_vacation*(*Peter*):<?*S'*,?*ti*>, requesting the common temporal intervals that Mary and Peter get extra vacation days, independently of the set of sources that this information is derived. Then, the answer to this query is the mapping *v*, where *v*(?*ti*)=[2001,2001], *v*(?*S*)={*Person*} and *v*(?*S'*)={*Person, Job, Vacation*}.

Consider now the simple query *SQ* = *extra_vacation*(*Mary*):<?*S*, ?*ti*> ∧ *extra_vacation*(*Peter*):<?*S*, ?*ti*>, requesting the common temporal intervals that Mary and Peter get extra vacation days, as derived from the same set of sources. Then, the answer to this query is the mapping *v*, where *v*(?*ti*)=[2001,2001] and *v*(?*S*)={*Person, Job, Vacation*}.

Consider now the complex query *CQ*= *has_job*(*Mary*, ?*x*):<?*S*, ?*ti*> ∧ *included*(?*ti*, [1993,2008]) ∧ ?*S*={*Person*} , requesting the temporal intervals that Mary has a job from 1993 to 2008, as derived from the source *Person*. The answers to this query is (i) the mapping *v*, where *v*(?*ti*)=[1993, 2002] and *v*(?*S*)={*Person*} and (ii) the mapping *u*, where *u*(?*ti*)=[2006,2008] and *u*(?*S*)={*Person*}.

Consider now the complex query CQ = extra_vacation(Mary): <?S, ?ti> ∧ extra_vacation(Peter):<?S, ?ti'> ∧ start(?ti) ≤ end(?ti') ∧ start(?ti') ≤ end(?ti') , requesting the temporal intervals that Mary and Peter get extra vacation days, as long as these overlap, and as derived from the same set of sources.

Then, the answer to this query is the mapping v, where v(?ti)=[2001,2001], v(?ti')=[1999,2002], and v(?S)={Person, Job, Vacation}.

VI. Related Work

In this Section, we discuss related work.

Flouris et al. [2] add provenance information to RDF theory [3], [4]. In particular, they extend RDF triples to RDF quadruples, where the fourth element is the set of graph names that participated in the derivation of the RDF triple through a limited subset of the RDFS entailment rules. They also discuss atomic update operations (i.e. inserts and deletes) of the RDF quadruples. Comparing to this paper, [2] does not present a model theory and does not consider the temporal domain. Note that our approach can also be applied to RDFS, as RDFS inference rules can be expressed through definite logic programming rules [5].

In [6], we extend extended logic programming rules [6] with their validity temporal intervals. We consider derivations based on temporal time points and Answer Set Programming [7], and provide an algorithm that returns the maximal temporal intervals that a literal is true. Present work has a different model theory and implementation than [6], as we consider only definite logic programming rules and on the other hand we also consider the provenance domain. Yet, the query language presented here is an extension of the query language presented in [6].

In [8], the authors present a general framework for representing, reasoning, and querying with RDFS annotated data on the Semantic Web. They show that their formalism can be instantiated on the temporal, fuzzy, and provenance domain. The authors can associate RDF triples with their validity temporal intervals and supportive sources and apply the RDFS inference rules (which are always valid). Yet, [8] does not support simple queries of type 4. Moreover, our query answering is more efficient, since during query answering, we directly work on maximal temporal intervals. In [8], all entailed temporal intervals returned by the query are considered and then the maximal ones are returned. Further, our semantics is different than [8]. For example, consider the annotated RDF triples $p(a,b)$:<n, [1990, 2000]> and $q(c,d)$: <n, [1995, 2010]>. Then, according to [8], the answer to query $p(a,b)$:<?S, ?ti> $\wedge$ $q(c,d)$: <?S, ?ti'> $\wedge$ $end(?ti)$ < $start(?ti')$ will provide the mapping v s.t. v(?ti)=[1990,1994] and $v(?ti')$=[1995, 2010]. In our case, we will provide no answers, since 2000 > 1995.

In [9], the authors present a framework to incorporate temporal reasoning into RDFS. The authors associate RDF triples with their validity temporal interval and apply the RDFS inference rules (which are always valid). Unlike our work, their semantics is based on time points and not on temporal intervals. Additionally, [9] does not consider the provenance domain. Further, it does not support simple queries of type 3 and type 4 and the filter condition is limited.

In [10], the authors extend RDF graphs with temporal information, by associating RDF triples with their validity interval. They consider any entailment regime that can be expressed through definite rules $A_0 \leftarrow A_1, \ldots, A_n$, where A_i is an RDF triple. Each such rule is replaced by the temporal rule A_0:[$max(t_1,\ldots, t_n)$, $min(t'_1,\ldots, t'_n)$] $\leftarrow$ A_1:[t_1, t'_1],..., A_n:[t_n, t'_n]. These rules are applied recursively, until a fixpoint is reach. Then, maximal validity temporal intervals for each derived RDF triple are produced. Yet, this work does not present a model theory based on temporal intervals and does not consider the provenance domain. Additionally, it does not support simple temporal queries of type 4 and the filter condition is left unspecified.

Work in [11] provides a framework to support spatial and temporal analysis over RDFS data. With respect to the temporal component, [11] is similar to [10], as it also computes the maximal validity temporal intervals of derived RDF triples, using the RDFS entailment rules. Yet, [11] does not consider the provenance domain.

Finally, we would like to note that our theory cannot be considered as a special case of annotated logic programming [12], as the model theory and the operational semantics are different there.

VII. Conclusion

In this paper, we have presented a model theory and the operational semantics of a *pt-logic* program P, that is a set of definite logic programming rules, annotated with the source that have been derived and their validity temporal interval. We have defined the simple and maximally temporal entailments of P, showing that there exists a minimal model M_{min} that contains exactly these maximal entailments. Additionally, we defined a consequence operator whose closure coincides with M_{min}. Further, we showed that simple and maximally temporal entailment from a *pt*-logic program P is EXPTIME-complete w.r.t. the size of P. A query language for our framework is proposed.

As future work, we plan to extend our theory to extended logic programs. Further, we plan to add additional parameters to definite logic programming rules such that space and trust.

References

[1] Evgeny Dantsin, T. Eiter, G. Gottlob, and A. Voronkov: "Complexity and expressive power of logic programming". ACM Comput. Surv. 33(3),2001,pp. 374-425.

[2] G. Flouris, I. Fundulaki, P. Pediaditis, Y. Theoharis , and V. Christophides}, "Coloring RDF Triples to Capture Provenance", 8th International Semantic Web Conference (ISWC-2009), 2009, pp. 196-212.

[3] G. Klyne and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax", W3C Recommendation, 10 February 2004, available at http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[4] Patrick Hayes, "RDF Semantics", W3C Recommendation, 10 February 2004, available at http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

[5] G. Ianni, A. Martello, C. Panetta, and G. Terracina, "Efficiently Querying RDF(S) Ontologies with Answer Set Programming", Journal of Logic and Computation, 19(4), 2009, pp. 671-695.

[6] A. Analyti and I. Pachoulakis, "Temporally Annotated Extended Logic Programs", accepted to International Journal of Advanced Research in Artificial Intelligence (IJARAI), 2012.

[7] M. Gelfond and V. Lifschitz, "Classical Negation in Logic programs and Disjunctive Databases", New Generation Computing, 9, 1991, pp. 365-385.

[8] A. Zimmermann, N. Lopes, A. Polleres, and U. Straccia, "A General Framework for Representing, Reasoning and Querying with Annotated Semantic Web Data", Journal of Web Semantics , 11, 2012, pp. 72–95.

[9] C. Gutierrez, C. A. Hurtado, and A. A. Vaisman, "Introducing Time into RDF", IEEE Transactions on Knowledge and Data Engineering, 19(2), 2007, 207-218.

[10] B. Motik, "Representing and Querying Validity Time in RDF and OWL: A Logic-Based Approach", 9th International Semantic Web Conference (ISWC-2010), 2010, pp. 550-565.

[11] M. Perry, A. P. Sheth, F. Hakimpour, and P. Jain, "Supporting Complex Thematic, Spatial and Temporal Queries over Semantic Web Data", 2nd International Conference on GeoSpatial Semantics (GeoS-2007), 2007, pp. 228-246.

[12] Michael Kifer and V. S. Subrahmanian, "Theory of Generalized Annotated Logic Programming and its Applications", Journal of Logic Programming, 12(3&4), 1992, pp. 335-367.

AUTHORS PROFILE

Anastasia Analyti earned a B.Sc. degree in Mathematics from University of Athens, Greece and a M.Sc. and Ph.D. degree in Computer Science from Michigan State University, USA. She worked as a visiting professor at the Department of Computer Science, University of Crete, and at the Department of Electronic and Computer Engineering, Technical University of Crete. Since 1995, she is a principal researcher at the Information Systems Laboratory of the Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH-ICS). Her current interests include reasoning on the Semantic Web, modular web rule bases, non-monotonic-reasoning, faceted metadata and semantics, conceptual modelling, contextual organization of information, information integration and retrieval systems for the web, interoperability of heterogeneous and distributed information bases, and biomedical information systems. She has participated in several research projects and has published over 55 papers in refereed scientific journals and conferences.

Ioannis Pachoulakis received a B.Sc. in Physics (1988) at the University of Crete, Greece, and a Ph.D. in Astrophysics (1996) and an M.Sc. in Engineering (1998), both from the University of Pennsylvania in the U.S.A. Since 2001, he serves as an Assistant Professor at the Department of Applied Informatics and Multimedia at TEI of Crete with mainstream interests in realistic multimedia applications, virtual reality and multimedia applications for science.