Stable Semantics of Temporal Deductive Databases

Heinrich Herre Gerd Wagner
{herre,gw}@informatik.uni-leipzig.de
Institut für Informatik, Univ. Leipzig,
Augustusplatz 10-11, 04109 Leipzig, Germany, Fax: (+49 341) 973 22 09.

Abstract

We define a preferential semantics based on *stable generated* models for a very general class of temporal deductive databases. We allow two kinds of temporal information to be represented and queried: timepoint and timestamp formulas, and show how each of them can be translated into the other. Because of their generality, our formalism and our semantics can serve as a basis for comparing and extending other temporal deductive database frameworks.

1 Introduction

A deductive database consists of facts and deduction rules. Logically, facts correspond to sentences of a suitably restricted language, and rules may be viewed as Gentzen sequents. A set of facts can be viewed as a database whose semantics is, according to the Closed-World Assumption, determined by its minimal models. For a set of facts and rules, however, minimal models are not adequate because they interpret rules in a non-directed way (according to the symmetric contraposition principle), and are therefore not able to capture the directedness of rules with negation-as-failure.

As an alternative to minimal models, *stable* models in the form of certain fixpoints have been proposed by Gelfond and Lifschitz [GL88] as the intended models of deductive databases (resp. normal logic programs). If we consider reasoning with temporal information the situation is similar: while minimal temporal models are adequate for temporal databases we need again a more refined criterion to select the intended models of temporal deductive databases. We show that our definition of *stable generated* models provides such a preference criterion. Stable generated models are generated by a *stable chain*, i.e. a layered sequence of rule applications where all applied rules remain applicable throughout the model generation sequence.

The paper has the following structure. After introducing some basic notation in section 2, we describe the first order language of temporally qualified formulas and explain the meaning of some temporal null values in section 3. In section 4, then, temporal Herbrand models are defined and their relationship to temporal databases is briefly discussed. Temporal deductive databases are introduced in section 5, and finally in section 6, the stable semantics for them is defined.

2 Preliminaries

A *signature* $\sigma = \langle Rel, Const, Fun \rangle$ consists of a set of relation symbols, a set of constant symbols, and a set of function symbols. $Term(\sigma)$, resp. $Term^0(\sigma)$, denotes the set of all terms, resp. ground terms, of σ. For a tuple $t_1, \ldots, t_n$ we will also write $\vec{t}$ when its length is of no relevance. The logical functors are $\neg, \wedge, \vee, \supset, \forall, \exists$; $\supset$ is called *material implication*. $L(\sigma)$ is the smallest set containing the atomic formulas of σ, and being closed with respect to the following conditions: if $F, G \in L(\sigma)$, then $\{\neg F,\ F \wedge G,\ F \vee G,\ F \supset G,\ \exists x F,\ \forall x F\} \subseteq L(\sigma)$.

For sublanguages of $L(\sigma)$ formed by means of a subset $\mathcal{F}$ of the logical functors, we write $L(\sigma; \mathcal{F})$. With respect to a signature σ we define the following sublanguages: $\mathrm{At}(\sigma) = L(\sigma; \emptyset)$, the set of all atomic formulas (also called *atoms*); $\mathrm{Lit}(\sigma) = L(\sigma; \neg)$, the set of all *literals*. We introduce the following conventions. When $L \subseteq L(\sigma)$ is some sublanguage, L^0 denotes the corresponding set of sentences (closed formulas). If the signature σ does not matter, we omit it and write, e.g., L instead of $L(\sigma)$. Whenever appropriate we write $F(x)$ instead of F in order to indicate that x occurs free in F. The set of free variables of a formula F is denoted by $\mathrm{Free}(F)$.

If Y is a partially ordered set, then $\mathrm{Min}(Y)$ denotes the set of all minimal elements of Y, i.e. $\mathrm{Min}(Y) = \{X \in Y \mid \neg\exists X' \in Y : X' < X\}$.

3 The Logic of Temporally Qualified Information

We consider two types of temporal formulas corresponding to two types of temporal qualification. In order to say that Charles and Diana have been married from 1981 to 1996 we can use a collection of *timepoint* sentences of the form

$$m(C,D)@1981,\ m(C,D)@1982,\ \ldots,\ m(C,D)@1996$$

Alternatively, we can state this fact more compactly using the *timestamp* sentence

$$m(C,D)@[1981..1996]$$

Timepoint formulas are related to the *snapshot* view of temporal information, where the semantic value of a relation symbol is a function that maps each timepoint to a relation over the object domain. In the timestamp view, the semantic value of a relation symbol is a function that assigns a timestamp, representing valid time, to each object tuple. Both languages can be formalized in the framework of two-sorted predicate logic over an *object signature* and a *temporal signature*.

3.1 Syntax

Let σ be an object signature associated with an object domain, and let τ be a temporal signature associated with a temporal domain, such that σ and τ have no symbols in common. We assume that the temporal signature τ contains a unary function symbol s, a constant 0 and a binary relation symbol $<$. Timepoints will be represented by the terms 0, $s(0)$, $s(s(0))$, $s^3(0)$, ..., corresponding to natural numbers.[1] We describe a language

[1] Throughout the paper, we assume a discrete temporal domain isomorphic to the order type of natural numbers. There is a widespread agreement in the database literature that this is the proper model of time for most kinds of temporal databases.

$TP(\sigma,\tau)$ for *timepoint formulas*, and a language $TS(\sigma,\tau)$ for *timestamp formulas*. The vocabulary of both languages contains the following symbols:
- object variables: $OVar = \{x_1, x_2, \ldots\}$, we also use $y, z, \ldots$;
- object constants: $OConst = \{c_1, c_2, \ldots\}$;
- variables for timepoints: $TVar = \{t_1, t_2, \ldots\}$, we also use $t', s, s', s_1, \ldots$;
- timepoint constants: $TConst = \{i_1, i_2, \ldots\}$, we also use $j, j_1, \ldots$.

We assume that $OVar \cap TVar = \emptyset$. Timepoint terms are denoted by $p, p', p_1, p_2, \ldots$.

$L(\sigma)$ is the set of formulas expressing properties of objects, such as $\exists y(m(x,y))$ expresses the fact that x is married to someone. $L(\tau)$ is the set of formulas expressing properties of timepoints, such as $0 < t \wedge \exists t'(t < t')$. The interpretation of symbols from the temporal signature τ, e.g. of $<$, is fixed (see 3.2), while the interpretation of symbols from the object signature σ varies in the usual way (determined by the actual database state).

Definition 1 (Elementary Timepoint Formulas) *An expression of the form $F@p$, where $F \in L(\sigma)$, and $p \in Term(\tau)$, is called an* elementary timepoint formula *over the signature (σ,τ); it is called an* atomic *timepoint formula if F is an atomic formula. $TP_{el}(\sigma,\tau)$ denotes the set of all elementary timepoint formulas over (σ,τ), and $\mathrm{At}(\sigma,\tau)$ denotes the set of all atomic timepoint formulas.*

Definition 2 (Timepoint Formulas) *The set $TP(\sigma,\tau)$ of* timepoint formulas *over (σ,τ) is the smallest set containing $TP_{el}(\sigma,\tau) \cup L(\tau)$, and being closed with respect to the connectives $\wedge, \vee, \neg, \supset$ and the following condition: if $F(x), G(t) \in TP(\sigma,\tau)$, then the formulas $\forall x F(x)$, $\exists x F(x)$, $\forall t G(t)$, and $\exists t G(t)$ belong to $TP(\sigma,\tau)$.*

Example 1 *The query* who married the same person again *can be expressed by the timepoint formula*

$$\exists t_1 \exists t_2 \exists t_3 (t_1 < t_2 \wedge t_2 < t_3 \wedge m(x,y)@t_1 \wedge \neg m(x,y)@t_2 \wedge m(x,y)@t_3)$$

Definition 3 (Timestamp Formulas) *The set $TS(\sigma,\tau)$ of* timestamp formulas *over (σ,τ) is the smallest set containing $\{F@\lambda t G(t) : F \in L(\sigma), G(t) \in L(\tau)\}$, and being closed with respect to $\wedge, \vee, \neg, \supset$ and $\forall x, \forall t, \exists x, \exists t$.*

The expression $\lambda t G(t)$ stands for the set of all timepoints defined by $G(t)$, see the semantic definition below. We define the set of *temporal formulas* as $L(\sigma,\tau) = TP(\sigma,\tau) \cup TS(\sigma,\tau)$. Elements of $\mathrm{At}^0(\sigma,\tau)$, i.e. atomic timepoint sentences, are called *temporal atoms*.

Example 2 *The following expressions are timestamp formulas:*

(1) $R(x)@\lambda t(0 < t \wedge t < s^5(0) \vee s^6(0) < t \wedge t < s^9(0))$
(2) $Q(x)@\lambda t(0 < t)$
(3) $Q(x)@\lambda t(0 < t \wedge \exists t'(t < t'))$
(4) $\exists t_1 \exists t_2 [R(x)@\lambda t(t_1 < t \wedge t < t_2) \wedge Q_1(x)@\lambda t(t_1 < t) \wedge Q_2(x)@\lambda t(t < t_2)]$

The timestamp of (1) corresponds to a finite union of disjoint intervals, i.e. the form of timestamps proposed in [Gad88]. In the sequel, we also use the notation

(1′) $R(x)@[1..4, 7..8]$

instead of (1), and call such timestamps *normal*. (2) expresses an eternal valid time, that is it corresponds to the null-valued timestamp formula

$$(2') \quad Q(x)@[1..\infty]$$

where ∞ stands for the null value *ad eternum*. (3) expresses a timestamp with unknown end time, that is it corresponds to the null-valued timestamp formula

$$(3') \quad Q(x)@[1..\iota]$$

where ι stands for the null value *unknown timepoint*. There are further *indexical* temporal null values like *now* and *until changed* which are discussed below in section 7. All of (1′), (2′) and (3′) can be viewed as a resp. shorthand notation for (1), (2) and (3).

3.2 Semantics

In the sequel we assume that $\tau = \langle 0, s, < \rangle$, and we write $\mathcal{N} = \langle T, s, 0, < \rangle$ for the standard system of natural numbers. A $\langle \sigma, \tau \rangle$-interpretation $\mathcal{I}$ interprets constant and function symbols from σ in the standard way as elements of, resp. functions over, the object domain $U_\mathcal{I}$ independent of the time. $\mathcal{I}$ assigns a function $R^\mathcal{I} : T \to 2^{U_\mathcal{I}^n}$ to every n-place relation symbol R from σ, such that $R^\mathcal{I}(t) \subseteq U_\mathcal{I}^n$ is the set of those object tuples for which R holds at timepoint t. The σ-interpretation $\mathcal{I}(t)$ assigning the relation $R^\mathcal{I}(t)$ to a relation symbol R is called the *snapshot of* $\mathcal{I}$ at timepoint t.

An $\mathcal{I}$-instantiation μ is a function $\mu : OVar \cup TVar \to U_\mathcal{I} \cup T$ assigning objects to object variables and timepoints to timepoint variables. We write μ_o for the restriction of μ to $OVar$, and μ_t for the restriction of μ to $TVar$. If an instantiation μ is changed at the argument x (or t) by assigning c (or i) to x (or t), we simply write μ_c^x (or μ_i^t) for the changed instantiation.

Definition 4 (Satisfaction of Timepoint Formulas)

1. *For elementary timepoint formulas:* $\mathcal{I}, \mu \models F@p$ *iff* $\mathcal{I}(\mu_t(p)), \mu_o \models F$.
2. *For temporal constraints* $F \in L(\tau)$*:* $\mathcal{I}, \mu \models F$ *iff* $\mathcal{N}, \mu_t \models F$.
3. $\mathcal{I}, \mu \models F \wedge G$ *iff* $\mathcal{I}, \mu \models F$ *and* $\mathcal{I}, \mu \models G$ *(similarly for* $F \vee G$, $F \supset G$*).*
4. $\mathcal{I}, \mu \models \neg F$ *iff* $\mathcal{I}, \mu \not\models F$.
5. $\mathcal{I}, \mu \models \exists x F(x)$ *iff there is an object* c*, such that* $\mathcal{I}, \mu_c^x \models F(x)$.
   $\mathcal{I}, \mu \models \exists t F(t)$ *iff there is a timepoint* i*, such that* $\mathcal{I}, \mu_i^t \models F(t)$.
6. $\mathcal{I}, \mu \models \forall x F(x)$ *iff for all objects* c*:* $\mathcal{I}, \mu_c^x \models F(x)$.
   $\mathcal{I}, \mu \models \forall t F(t)$ *iff for all timepoints* i*:* $\mathcal{I}, \mu_i^t \models F(t)$.

Definition 5 (Satisfaction of Timestamp Formulas)

1. *An instantiation* μ *maps a timestamp into a set of timepoints:*
   $\mu(\lambda s F(s)) := \{i \in T : \mathcal{N}, (\mu_t)_i^s \models F(s)\}$.
2. $\mathcal{I}, \mu \models F@\lambda t G(t)$ *iff for all* $i \in \mu(\lambda t G(t))$*:* $\mathcal{I}(i), \mu_o \models F$.
3. *All other cases are defined as for timepoint formulas.*

We need some further definitions. Let $F \in L(\sigma,\tau)$ be a temporal formula, and $\mathcal{I}$ be a (σ,τ)-interpretation. Then $\mathrm{Sat}(\mathcal{I},F) = \{\mu : \mathcal{I},\mu \models F\}$, and $\mathcal{I} \models F$ iff for all μ, $\mathcal{I},\mu \models F$. For $X \subseteq L(\sigma,\tau)$ we define $\mathrm{Mod}(X) := \{\mathcal{I} : \mathcal{I} \models F \text{ for every } F \in X\}$, and the standard entailment relation $X \models F$ iff $\mathrm{Mod}(X) \subseteq \mathrm{Mod}(\{F\})$. Two temporal formulas F and G are *logically equivalent*, symbolically $F \equiv G$, iff for every (σ,τ)-interpretation $\mathcal{I}$ it holds that $\mathrm{Sat}(\mathcal{I},F) = \mathrm{Sat}(\mathcal{I},G)$.

3.3 From Timepoints to Timestamps and Vice Versa

We compare the expressive power of the languages $TP(\sigma,\tau)$ and $TS(\sigma,\tau)$, and show that both languages have the same expressive power.

Theorem 1 *There exist algorithmic functions $f : TP^0(\sigma,\tau) \to TS^0(\sigma,\tau)$ and $g : TS^0(\sigma,\tau) \to TP^0(\sigma,\tau)$, such that for every (σ,τ)-interpretation $\mathcal{I}$ and all sentences $F \in TP^0(\sigma,\tau)$, $G \in TS^0(\sigma,\tau)$ the following equivalences hold:*

$$(1) \quad \mathcal{I} \models F \quad \textit{iff} \quad \mathcal{I} \models f(F)$$
$$(2) \quad \mathcal{I} \models G \quad \textit{iff} \quad \mathcal{I} \models g(G)$$

Proof: obtained as a corollary of the following lemmas.

Lemma 1 *There is an algorithmic function $g : TS(\sigma,\tau) \to TP(\sigma,\tau)$, such that for all $F \in TS(\sigma,\tau)$ it holds that $\mathrm{Free}(F) = \mathrm{Free}(g(F))$, and for every interpretation $\mathcal{I}$ and instantiation μ:*

$$\mathcal{I},\mu \models F \quad \textit{iff} \quad \mathcal{I},\mu \models g(F)$$

Proof (inductively on the complexity of F): We sketch only the initial step. Let $F = G@\lambda t H(t)$, then $g(F) := \forall t(H(t) \supset G@t)$. □

Lemma 2 *There is an algorithmic function $h : TP(\sigma,\tau) \to TS(\sigma,\tau)$, such that for all $F \in TP(\sigma,\tau)$ it holds that $\mathrm{Free}(F) = \mathrm{Free}(h(F))$, and for every interpretation $\mathcal{I}$ and instantiation μ*

$$\mathcal{I},\mu \models F \quad \textit{iff} \quad \mathcal{I},\mu \models h(F)$$

Proof: (inductively on the complexity of F). We sketch only the initial steps. Let $F = G@p$ be an elementary timepoint formula, then $h(F) := G@\lambda t(t = p)$. Next, let $F = t < t'$ be an atomic temporal constraint, then

$$h(F) := H(t,t') = \exists x(x \neq x)@\lambda s(s = t' \wedge t' \leq t)$$

Assume $\mathcal{I},\mu \models t < t'$, then $\mu(t) < \mu(t')$ in $\mathcal{N}$, implying that the set $\{i \in T : i = \mu(t') \wedge \mu(t') \leq \mu(t)\}$ is empty, and therefore trivialy, $\mathcal{I},\mu \models H(t,t')$. Conversely, if $\mathcal{I},\mu \models H(t,t')$, then, since $\exists x(x \neq x)$ is not satisfiable, the set $\mu(\lambda s(s = t' \wedge t' \leq t))$ has to be empty, implying that $\mathcal{I},\mu \models t < t'$. □

4 Minimal Herbrand Models of Temporal Databases

A *Herbrand interpretation* $\mathcal{I}$ of the language $L(\sigma,\tau)$ is an interpretation whose object domain is the set of all ground terms of σ, and which interprets constants and function symbols canonically (i.e. identically). Notice that $\mathcal{I}$ is a temporal Herbrand interpretation iff for all $t \in T$, $\mathcal{I}(t)$ is a classical Herbrand interpretation. While a classcial Herbrand interpretation can be identified with a set of ground atoms (atomic sentences), a temporal Herbrand interpretation $\mathcal{I}$ can be identified with a set of temporal ground atoms

$$I = \{a@p \in \mathrm{At}^0(\sigma,\tau) : \mathcal{I} \models a@p\}$$

The class of all Herbrand (σ,τ)-interpretations is denoted by $\boldsymbol{I}_H(\sigma,\tau)$. In the sequel we simply say 'interpretation', resp. 'model', instead of 'temporal Herbrand interpretation', resp. 'temporal Herbrand model'. We write I instead of $\mathcal{I}$ when we want to indicate the use of sets of temporal atoms as interpretations. For a set of temporal sentences $X \subseteq L^0(\sigma,\tau)$, we define $\mathrm{Mod}_H(X) = \{\mathcal{I} \in \boldsymbol{I}_H(\sigma,\tau) : \mathcal{I} \models F \text{ for all } F \in X\}$.

Definition 6 (Informational Extension of an Interpretation)
Let $I, J \in \boldsymbol{I}_H(\sigma,\tau)$ be two interpretations. We say that J extends *I, resp. J is* informationally greater than or equal to *I, symbolically $I \leq J$, if for all $t \in T$: $I(t) \subseteq J(t)$.*

Obviously, $I \leq J$ iff $I \subseteq J$, and $\langle \boldsymbol{I}_H, \leq \rangle$ is a complete lattice.

Definition 7 (Minimal Model) *For $F \in L^0(\sigma,\tau) \supseteq X$, we define*

$$\mathrm{Mod}_m(X) = \mathrm{Min}(\mathrm{Mod}_H(X))$$

and the minimal entailment *relation $X \models_m F$ iff $\mathrm{Mod}_m(X) \subseteq \mathrm{Mod}_H(F)$.*

It is well-known that a relational database Δ over a language $L(\sigma)$ can be viewed as a finite set of atomic sentences $X_\Delta \subset \mathrm{At}^0(\sigma)$ which corresponds to a finite Herbrand interpretation $\mathcal{I}_\Delta$, such that $\mathcal{I}_\Delta$ is the unique minimal model of X_Δ (provided that Δ does not contain any null values). An if-query $F \in L^0(\sigma)$ is confirmed by Δ, if and only if $\mathcal{I}_\Delta \models F$, or equivalently, $X_\Delta \models_m F$. This consideration shows that query answering in relational databases is based on minimal entailment. The same holds for temporal databases.

An temporal database Δ without null values has a propositional representation as a set of temporal atoms $X_\Delta \subseteq \mathrm{At}^0(\sigma,\tau)$. For a more efficient representation, X_Δ may be 'compressed' into its *timestamp normal form* which contains for each atom a occuring in X_Δ exactly one timestamp atom of the form $a@[b_1..e_1,\ldots,b_n..e_n]$, where the timestamp consists of a finite sequence of disjoint intervals with begin time b_i and end time e_i. Both the timepoint and the timestamp representation of Δ have the same unique minimal model $\mathcal{I}_\Delta$, and thus for both propositional representations of Δ query answering has to be defined in accordance with minimal entailment.

Recall that a database schema Σ essentially consists of a finite set of predicates $\Sigma = \{r_1,\ldots,r_m\}$ which determine the temporal language $L(\Sigma,\tau)$.

Definition 8 (Temporal Database) *A* timestamp tuple *is an object tuple concatenated with a normal timestamp. A* temporal table *is a finite set of (homogeneous) timestamp tuples. A* temporal database *Δ over a schema $\Sigma = \{r_1,\ldots,r_m\}$ is a finite set of temporal tables: $\Delta = \{R_1,\ldots,R_m\}$, such that the table R_k represents the temporal extension of the predicate r_k.*

Example 3 *The following temporal table represents the salary history of employees:*

$$Sal = \begin{array}{|ll|l|} \hline Tom & 3600 & 1994..1995 \\ Tom & 4100 & 1996 \\ Tom & 3900 & 1997 \\ Pit & 4400 & 1993..1994, 1997 \\ Pit & 4600 & 1995..1996 \\ \hline \end{array}$$

Notice that in practice one does usually not know the end timepoint of the currently valid sentences. In order to represent this adequately, one needs the null value *until changed* which is dicussed in section 7.

Observation 1 (Propositional Representation) *A temporal database Δ over the schema $\Sigma = \{r_1, \ldots, r_m\}$ can be propositionally represented as*

$$X_\Delta = \bigcup_{k=1}^{m} \{r_k(\vec{c})@T : \langle \vec{c}, T \rangle \in R_k\}$$

If the timestamps T in X_Δ do not contain null values (except ∞), the timestamp atoms of X_Δ can be flattened to a set of temporal atoms:

$$\mathrm{At}_\Delta = \bigcup \{a@i : a@T \in X_\Delta \ \ \& \ \ i \in T\}$$

Only in the rather idealized case without any disjunctive or existential null values can a database Δ be viewed as an interpretation. In general, it should be viewed as a set of sentences X_Δ, most of which are atomic.

Observation 2 (Induced Interpretation) *A temporal database Δ without null values (except ∞) induces the interpretation $I_\Delta = \mathrm{At}_\Delta$, being the unique minimal model of X_Δ, such that for an if-query F,*

$$\mathcal{I}_\Delta \models F \ \Leftrightarrow \ X_\Delta \models_m F$$

Definition 9 (Natural Inference) *Let Δ be a temporal database over Σ, and $F \in L^0(\Sigma, \tau)$ be a temporal if-query. The natural inference relation $\vdash$, which is the logical basis of query answering, is defined by means of minimal entailment:*

$$\Delta \vdash F \ :\Leftrightarrow \ X_\Delta \models_m F$$

Example 4 *Let $\Delta_3 = \{Sal\}$. Then,*

$$\begin{array}{rcl} X_{\Delta_3} & = & \{sal(Tom, 3600)@[1994..1995],\ sal(Tom, 4100)@1996, \ldots\} \\ I_{\Delta_3} & = & \{sal(Tom, 3600)@1994,\ sal(Tom, 3600)@1995, \ldots\} \\ \Delta_3 & \vdash & \neg\exists x, y : sal(x, y)@1996 \wedge y < 4000 \end{array}$$

Obviously, a set of atomic sentences such as represented by a temporal database (without null values) has a minimal model. But how about more general sets of formulas ?

Proposition 3 *Let $X \subseteq L(\sigma, \tau)$ be a set of universal sentences. If X has a model then X has a minimal Herbrand model.*

Example 5 *The sentence* $\forall t \exists s(t < s \wedge R(\vec{c})@s)$ *does not have a minimal model.*

Since temporal databases are an extension of relational databases, an important question is *how to embed the latter in the former*. Formally, the answer to this question is straightforward: choose an arbitrary timepoint, say t', and consider a relational database Δ as a snapshot of the corresponding temporal database Δ' at this timepoint: $\mathcal{I}_\Delta = \mathcal{I}_{\Delta'}(t')$. The corresponding temporal database Δ' gives the same answers (at t') as the original database does: $\mathsf{Ans}(\Delta', F@t') = \mathsf{Ans}(\Delta, F)$. In this sense, relational databases can be faithfully embedded in temporal databases.[2]

5 Temporal Deductive Databases

We obtain a temporal deductive database by adding deduction rules to a temporal database. The general form of a temporal deduction rule is defined as follows.

Definition 10 (Temporal Deduction Rule) *A temporal deduction rule r is an expression of the form*

$$F_1, \ldots, F_m \leftarrow G_1, \ldots, G_n$$

where $F_i, G_j \in L(\sigma, \tau)$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The body *of r, denoted by Br, is given by $\{G_1, \ldots, G_m\}$, and the* head *of r, denoted by Hr, is given by $\{F_1, \ldots, F_n\}$. $R(\sigma, \tau)$ denotes the class of all rules r such that $Hr, Br \subseteq L(\sigma, \tau)$. For a given set of rules $R \subseteq R(\sigma, \tau)$, $[R]$ denotes the* ground instantiation *of R, i.e. the set of all ground instances of rules from R.*

As with Gentzen sequents, the head of a rule represents a disjunction, while the body represents a conjunction.

Definition 11 (Satisfaction of Rules) *Let $I \in \boldsymbol{I}_H(\sigma, \tau)$. Then,*

$$I \models F_1, \ldots, F_m \leftarrow G_1, \ldots, G_n \quad \textit{iff} \quad \bigcap_{i \leq n} \mathrm{Sat}(I, G_i) \subseteq \bigcup_{j \leq m} \mathrm{Sat}(I, F_j)$$

Note that a rule $F \leftarrow$ with an empty body corresponds to a fact $F \in L(\sigma, \tau)$. The question arises, whether the rule arrow $\leftarrow$ is really needed since we already have the material implication connective $\supset$ in our language. The following proposition seems to suggest that there is no need for the new connective $\leftarrow$.

Observation 3 (Rules and Material Implication) *Let $H \leftarrow B$ be a temporal deduction rule, and I be an interpretation. Then,*

(1) $I \models H \leftarrow B$ *iff* $I \models \bigwedge B \supset \bigvee H$.
(2) I *is a minimal model of* $H \leftarrow B$ *iff* I *is a minimal model of* $\bigwedge B \supset \bigvee H$.

[2] Yet, this formal consideration does not reflect the implicit temporality and dynamics of a relational database. Normally, the facts entered into a relational database are valid from the time being entered until they are deleted, and thus a relational database contains implicit temporal information which cannot be queried, however, and can also not be reflected in the embedding.

We will see below, however, that the intended models of a rule are not its minimal but its *stable* models, and the stable models of a rule do not coincide with the stable models of the corresponding material implication (see ex. 7).

For a rule set $R \subseteq R(\sigma, \tau)$, we define the model operators

$$\begin{array}{lcl} \mathrm{Mod}_H(R) & = & \{I \in \boldsymbol{I}_H(\sigma,\tau) : I \models r, \text{ for all } r \in R\} \\ \mathrm{Mod}_m(R) & = & \mathrm{Min}(\mathrm{Mod}_H(R)) \end{array}$$

We now define temporal deductive databases (TDDBs) in a very general way. This does not mean that we propose such general rule formats for practical systems, it only reflects the semantical perspective of our paper. From a semantical point of view it is desirable to define the most general class of structures under consideration. It is a different enterprise, then, to identify the theoretically and practically relevant subclasses.[3]

Definition 12 (TDDB) *A* temporal deductive database *is a finite set of quantifier-free temporal deduction rules.*

Notice that this definition allows disjunctive rules as a specific form of incompleteness, whereas it excludes null values representing existential information, such as ι, or *uc*. For instance, a rule describing Russian roulette, is expressible as

$$dead@s(t) \vee dead@s^2(t) \vee \ldots \vee dead@s^6(t) \leftarrow loaded@t$$

Also, negation is allowed both in the body and the head of a rule.

We have chosen quantifier-free rules because they can be viewed as a representation of the set of ground rules obtained by instantiation.

Proposition 4 *Let R be a TDDB. Then,* $\mathrm{Mod}_H(R) = \mathrm{Mod}_H([R])$*, where* $[R]$ *is the ground instantiation of* R*.*

A preferential semantics for rules is given by a preferred model operator $\Phi : 2^{R(\sigma,\tau)} \rightarrow 2^{\mathbf{I}_H(\sigma,\tau)}$, satisfying the condition $\Phi(R) \subseteq \mathrm{Mod}_H(R)$, and defining the preferential entailment relation

$$R \models_\Phi F \text{ iff } \Phi(R) \subseteq \mathrm{Mod}_H(F)$$

The question is: which preferred model operator captures the intended models of temporal deductive databases ?

Our intuitive understanding of rules suggests that they are able to generate additional information when applied to given information. A model I of a rule set R is considered to be *intended* if the information of I can be generated by an exhaustive iterated application of the rules $r \in [R]$. In other words, the intended models of R should be *grounded* in R. Furthermore, we want to have the property that every piece of information $a \in I$ is *supported* by a rule $r \in [R]$ in the sense that a occurs positively in the head of r, and I satisfies the body of r. Below, we will make these ideas precise.

[3] Usually, the language of databases is restricted to function-free signatures (e.g. the *schema* of a relational database does not allow function symbols). Another important restriction, which guarantees finite and 'sensible' query evaluation, is the requirement that rules in databases have to be *range-restricted*, i.e. the head of a rule must not contain other free variables than those occuring in the body, and the body must be *evaluable*, resp. *domain-independent* in the sense of [vGT91]. We will not investigate the definition of evaluable formulas of $L(\sigma, \tau)$ in this paper.

If a TDDB contains rules with negated premise formulas, it may have minimal models which are not grounded in the sense sketched above. Consequently, preferential semantics based on minimal models does, in general, not guarantee groundedness. This is illustrated by the following example.

Example 6 *We assume that a car insurance company refunds money to its customers if they did not have an accident in the previous year, expressed by the rule*

$$r_1 : refund(x)@s(t) \leftarrow customer(x)@t, \neg accident(x)@t$$

Let R_6 consist of this rule together with the fact that Wagner was a customer in 1995: $R_6 = \{customer(W)@1995, r_1\}$. Obviously, R_6 has two minimal models:

$$M_1 = \{\, customer(W)@1995, refund(W)@1996 \,\}$$
$$M_2 = \{\, customer(W)@1995, accident(W)@1995 \,\}$$

Only M_1 is intended, and Wagner should get the refund. Note that in the case of M_2, the atom accident$(W)@1995$ *cannot be generated by applying rules from $[R_6]$ because it does not appear in the head of any rule. M_2 is therefore neither grounded nor supported.*

In the next section, we define a more refined preference criterion for selecting the intended models of a TDDB: *stable generated models* are both grounded and supported.

6 Stable Semantics for Temporal Deductive Databases

We need some further notation. With respect to a class of interpretations $\boldsymbol{K}$, we write $\boldsymbol{K} \models F$ iff $I \models F$ for all $I \in \boldsymbol{K}$. We denote the set of all rules from a rule set R which are applicable in $\boldsymbol{K}$ by

$$R_{\mathbf{K}} = \{r \in [R] : \boldsymbol{K} \models Br\}$$

If $\boldsymbol{K}$ is a singleton, we omit brackets.

Definition 13 (Interpretation Interval) *Let $I_1, I_2 \in \boldsymbol{I}_H(\sigma, \tau)$. Then, $[I_1, I_2] = \{I \in \boldsymbol{I}_H : I_1 \leq I \leq I_2\}$.*

Recall the fact that every satisfiable set of quantifier-free formulas has a minimal Herbrand model which is the basis for the following definition.

Definition 14 (Stable Generated Model) *Let R be a TDDB. A model M of R is called* stable generated, *symbolically $M \in \mathrm{Mod}_s(R)$, if there is a chain of Herbrand interpretations $I_0 \leq \ldots \leq I_\kappa$ such that $M = I_\kappa$, and*

1. *$I_0 = \emptyset$.*
2. *For successor ordinals α with $0 < \alpha \leq \kappa$, I_α is a minimal extension of $I_{\alpha-1}$ satisfying all rules which are applicable in $[I_{\alpha-1}, M]$, i.e.*

$$I_\alpha \in \mathrm{Min}\{I \in \boldsymbol{I}_H(\sigma) : I \geq I_{\alpha-1}, \text{ and } I \models r, \text{ for all } r \in R_{[I_{\alpha-1}, M]}\}$$

3. *For limit ordinals $\lambda \leq \kappa$, $I_\lambda = \sup_{\alpha<\lambda} I_\alpha$.*

We also say that M is generated *by the* R-stable chain $I_0 \leq \ldots \leq I_\kappa$.

Not every satisfiable TDDB has a stable generated model. A simple example of an *unstable* TDDB is $\{r(c)@0 \leftarrow \neg r(c)@0\}$ whose unique minimal model $\{r(c)@0\}$ cannot be generated by a stable chain. This problem is, however, not related to the temporal qualification, and we encounter it also in the case of normal logic programs.

In the sequel, we simply say 'stable' instead of 'stable generated' model. The stable entailment relation is defined as follows:

$$R \models_s F \quad \text{iff} \quad \mathrm{Mod}_s(R) \subseteq \mathrm{Mod}_H(F)$$

With respect to stable models, rules differ from material implications.

Example 7 *Let $a, b \in \mathrm{At}^0(\sigma, \tau)$ be temporal ground atoms. Then,*

$$\mathrm{Mod}_s(\neg a \supset b) = \mathrm{Mod}_s(a \vee b) = \{\{a\}, \{b\}\} \neq \mathrm{Mod}_s(b \leftarrow \neg a) = \{\{b\}\}$$

And indeed, using stable entailment we obtain the right answer in the case of our car insurance example, i.e. Wagner gets the refund:

$$R_6 \models_s \mathit{refund}(W)@1996$$

since $\mathrm{Mod}_s(R_6) = \{M_1\}$.

Theorem 2 *A deductive database Δ under the stable model semantics of [GL88] can be faithfully embedded (as a snapshot) in a temporal deductive database Δ' under our stable semantics such that there is a one-to-one correspondence between the stable models of Δ and the stable generated models of Δ'.*

Proof: follows directly from the proof that the stable models of a normal logic program in the sense of [GL88] are exactly the models generated by a stable chain in [HW96].

7 Indexical Temporal Null Values

In this section, we briefly discuss the meaning of two important temporal null values. Notice, however, that we did not consider these special expressions in our treatment of TDDBs above. Their inclusion would require certain extensions of our framework which we left as an issue for future work.

It is natural to use the expression *now* in temporal queries like, for instance, *Is Tom's salary now higher than 4000 ?* Such indexical queries, however, cannot be evaluated against a temporal database X alone. Their evaluation requires in addition to X a reference timepoint t which determines the interpretation of *now*. Let X_1 consist of the table of ex. 3. Then, we get

$$\begin{array}{lll} X_1, 1996 & \vdash & \mathit{sal}(\mathit{Tom}, x)@\mathit{now} \wedge x > 4000 \\ X_1, 1997 & \nvdash & \mathit{sal}(\mathit{Tom}, x)@\mathit{now} \wedge x > 4000 \end{array}$$

since in the example database X_1, Tom's salaray is decreased in 1997 from 4100 to 3800 dollars.

The satisfaction relation for the extended language with *now* has to be extended accordingly. A model, then, consists of a an interpretation $\mathcal{I}$, an instantiation μ, and an evaluation instant i. E.g.,

$$\mathcal{I}, \mu, i \models F@now \quad \text{iff} \quad \mathcal{I}, \mu \models F@i$$

Formally, *now* is added to the temporal signature τ as a special constant which may be used in temporal terms, and which is replaced by the evaluation instant when interpreted.

Another important indexical null value is *until changed (uc)* which is used when storing new information with an open-ended valid time. Notice that it seems to be the rule, and not an exception, that new temporally dependent information (such as, e.g., the new salary of an employee) does not come with a definite end time. In that case, the null value *uc* must be used to store the new information in the database: $sal(Tom, 3900)@[1997, uc]$ expresses the fact that from 1997 until changed Tom's salaray is 3900. Formally, this means that the unknown end timepoint is greater than or equal to *now*, i.e.

$$[1997, uc] = \lambda t(1997 \leq t \wedge \exists s(now \leq s \wedge t \leq s))$$

The difference between *now* and *uc* comes only into play when statements about the future are concerned. This is the case if the database is used for planning or for recording commitments. For instance,

$$\begin{array}{rcl}
\{sal(Tom, 3900)@[1997, uc]\}, 1998 & \vdash & sal(Tom, 3900)@1998 \\
\{sal(Tom, 3900)@[1997, now]\}, 1998 & \vdash & sal(Tom, 3900)@1998 \\
\{sal(Tom, 3900)@[1997, now]\}, 1996 & \nvdash & sal(Tom, 3900)@1997 \\
\{sal(Tom, 3900)@[1997, uc]\}, 1996 & \nvdash & sal(Tom, 3900)@1997
\end{array}$$

The differences between ∞, *now* and *uc* are informally discussed in [WJ91]. Many authors seem to overlook the fact that *now* and *uc* are indexical expressions requiring a special semantical treatment, and should not be confused with ∞.

8 Related Work

The distinction between the snapshot and the timestamp view of temporal information is discussed at length in [Cho94], where timestamps, however, are more restricted than in our treatment (we allow for free variables in timestamps). The null values *uc* and ∞, which we have distinguished as certain timestamps, have been only informally discussed in [WJ91].

Negation in temporal deduction rules has been only allowed in *stratified* TDDBs by many authors (e.g. [Cho90]) whereas our semantics is also defined for non-stratified TDDBS, and it even allows negation and disjunction in the head of rules. Our definition of stable generated models of TDDBs is based on the corresponding definition for normal logic programs in [HW96].

9 Conclusion

We have shown that the notion of a stable model, originally proposed for normal logic programs, can also be defined for a very general class of temporal deductive databases. Our semantics can serve as a basis for investigating further specific subclasses of TDDBs which extend the temporal versions of Datalog by adding negation and disjunction. Such extensions are interesting for information (and knowledge representation) systems if they prove to be more expressive at an acceptable price.

References

[Cho94] J. Chomicki. Temporal Query Languages: a Survey. in: D. Gabbay and H. J. Ohlbach (eds.) Temporal Logic, LNAI vol. 827 (1994), 506–534

[Cho90] J. Chomicki: Polynomial-Time Computable Queries in Temporal Deductive Databases, *Proc. of PODS'90*, 1990.

[Gad88] S.K. Gadia: A Homogeneous Relational Model and Query Languages for Temporal Databases, *ACM Transactions on Database Systems* **13**:4 (1988), 418–448.

[vGT91] A. van Gelder and R.W. Topor: Safety and Translation of Relational Calculus Queries, *ACM Transactions on Database Systems* **16**:2 (1991), 235–278.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, editors, *Proc. of ICLP*, pages 1070–1080. MIT Press, 1988.

[HW96] H. Herre and G. Wagner: Stable Models are Generated by a Stable Chain, Technical Report, Univ. Leipzig, 1996, to appear in the *J. of Logic Programming.*

[WJ91] G. Wiederhold and S. Jajodia: Dealing with Granularity of Time in Temporal Databases, *Proc. of 3rd Int. Conf. CAiSE'91*, LNCS 498, Springer-Verlag, 1991.

A Weiteres Material

Timepoint formulas can be normalized in the following way.

Proposition 5 *Let $G(p)$ denote the elementary timepoint formula $F@p$, then there is a timepoint formula $H(p)$ such that:*

1. *$G(p) \equiv H(p)$, and*
2. *every elementary timepoint subformula of $H(p)$ has the form: $A@p$, where A is atomic and* $\mathrm{Free}(A) \subseteq \mathrm{Free}(G)$.

Definition 15 (Prenex Formula) *$F \in TP(\sigma,\tau)$ is a* prenex formula *if it has the form: $Q_1 v_1 Q_2 v_2 \ldots Q_m v_m G$, where $Q_i \in \{\forall, \exists\}$ and $v_i \in TVar \cup OVar$, $i = 1, 2, \ldots, m$, and G is a quantifier free formula from $TP(\sigma,\tau)$. A prenex formula is called* universal *if all quantifiers are universal.*

Proposition 6 *For every $F \in TP(\sigma,\tau)$ there is a prenex formula $G \in TP(\sigma,\tau)$, such that $F \equiv G$.*

Definition 16 (Persistent Formula) *A formula $F \in L(\sigma,\tau)$ is called* persistent *if for arbitrary Herbrand interpretations $I, J \in \boldsymbol{I}_H(\sigma,\tau)$ satisfying $I \leq J$, and every valuation $\nu : Var \to U_\sigma$ the condition $I \models F\nu$ implies $J \models F\nu$.*

Observation 4 *Every formula $F \in L(\sigma; \wedge, \vee, \exists, \forall)$ is persistent.*

Proposition 7 *Let $X \subseteq L(\sigma,\tau)$ be a set of universal sentences. If X has a model then X has a minimal Herbrand model.*

Theorem 3 *Let $X \subseteq TP(\sigma,\tau)$ be a set of universal sentences. Then every Herbrand model of X is an extension of a minimal Herbrand model of X, and can be extended to a maximal Herbrand model of X.*

Proof sketch: We assume that every sentence in X is a prenex formula, and that every elementary timepoint subformula in $F \in X$ has the form $R(\vec{x})@p(\vec{t})$. W.l.o.g. we assume that every formula $F \in X$ is a conjunctive normal form over formulas of the form $R(\vec{x})@p(\vec{t})$, $p(\vec{t}) < q(\vec{s})$, or $p(\vec{t}) = q(\vec{s})$. Let $I \in \mathrm{Mod}(X)$, and let $I_0 \leq I_1 \leq \ldots \leq I_n \leq \ldots$ be a sequence of models of X with $I = I_0$. We show that the interpretation J defined by $J(t) = \bigcup I_n(t)$, $t \in T$, is a model of X. By Zorn's lemma this implies the existence of a maximal model of X extending I. For any $F \in X$, we may assume that F has the following form:

$$p_1(\vec{t}) \vee \ldots \vee p_m(\vec{t}) \vee \neg q_1(\vec{t}) \vee \ldots \neg q_n(\vec{t}) \vee$$
$$A_1(\vec{x})@\alpha_1(\vec{t}) \vee \ldots \vee A_k(\vec{x})@\alpha_k(\vec{t}) \vee \neg B_1(\vec{x})@\beta_1(\vec{t}) \vee \ldots \vee \neg B_l(\vec{x})@\beta_l(\vec{t})$$

We show that $J \models \forall\vec{x}\forall\vec{t}F(\vec{x},\vec{t})$ by proving that the assumption $J \not\models \forall\vec{x}\forall\vec{t}F(\vec{x},\vec{t})$ leads to a contradiction.

Proposition 8 *Let R be a TDDB. Then,* $\mathrm{Mod}_H(R) = \mathrm{Mod}_H([R])$, *where $[R]$ is the ground instantiation of R.*

Question 1 *For which superclasses of quantifier-free rules does this property still hold ?*

Definition 17 (Query Languages) *We consider following query languages for generalized temporal deductive databases. $L^0_Q(\sigma,\tau)$ is the set of all quantifier-free sentences from TP(σ,τ), $L^1_Q(\sigma,\tau)$ is the set of all closed formulas from TP(σ,τ) not containing object quantifiers. $L^2_Q(\sigma,\tau) = L^1_Q(\sigma,\tau) = \{\exists(F) : \exists(F)$ is the existential closure of F and F is a formula from $TP(\sigma,\tau)$ without free time variables and without object quantifiers $\}$.*