A non-standard temporal deductive database system

Jean-Raymond Gagné and John Plaice
Département d’informatique, Université Laval
Ste-Foy (Québec) Canada G1K 7P4
{gagne,plaice}@ift.ulaval.ca
1995

Abstract

A new temporal deductive database system supporting a non-standard model of time is introduced. It consists of Non-standard Temporal Datalog (nstl) and Non-standard Temporal Relational Algebra (nstra). The time-line consists of non-standard reals that are of the form \((r, i)\), where \(r \in \mathbb{R}\) and \(i \in \mathbb{Z}\), using the natural order. Each real \(r\) determines a macro-instant, and each pair \((r, i)\) defines a micro-instant. The set of macro-instants forms a dense order, thereby allowing different relations to be valid at different moments, with independent rates of evolution. The micro-instants ensure that all intervals are closed, thereby simplifying the semantics. At the same time, it becomes possible to define a discrete memory operator.

The nstl language is an extension of Datalog in which the fact base is augmented with interval timestamps and in which rules are an extension of generalized Horn clauses that allow a memory operator “;” and allow timestamped atoms in the body.

The nstra language is a point-wise extension of the relational algebra over the time-line. To do this, three temporal operators are added to the relational algebra.

1 Introduction

This paper presents a non-standard temporal database system in which the time-line consists of non-standard reals \([?, ?]\), which are, in this case, pairs \((r, i)\), where \(r \in \mathbb{R}\) and \(i \in \mathbb{Z}\). Choosing such a time-line ensures that one can have a dense set of macro-instants and, within each macro-instant, a discrete set of micro-instants.

This vision of time is consistent with that taken in synchronous programming languages, such as Esterel [2], Lustre [3] and Signal [1]. Micro-instants can be perceived as the “time” associated with different computations within a single instant. It is also consistent with current work in hybrid systems.

When there are several relations in a database, it is normal that the times of validity for different relations may have little to do with one another. It does not seem appropriate to force them all onto a discrete time line. Rather, a dense time line seems more appropriate.

However, if several objects are placed at different places on the time line, then means for synchronizing unsynchronized relations are necessary. These means already exist for synchronous languages. In particular, the authors are developing a language called Blizzard to furnish the semantics of reactive systems [?].

Blizzard is a dataflow language, where operators apply to operands in a pointwise manner, along with an additional memory operator, called before. These operators are used below to extend the temporal algebra into the nstra.

To develop a deductive database system, we use Orgun’s technique of combining an extension of Datalog, nstl, with an extension of the temporal algebra, nstra. The basic extension in both formalisms is the addition of a memory operator, before in nstra and “;” in nstl.
This paper presents both NSTL and NSTRA, their properties and their interaction. The presentation begins with a description of the underlying time domain. This is followed by the syntax and semantics of NSTL programs. Perfect-model semantics are used, as NSTL programs use negation, both implicitly and explicitly.

Since NSTL programs are written in an extension of non-standard temporal generalized Horn clauses we give some properties and theoretical results about the computability of NSTL programs, along with a revised version of a new operator \( T_P \), used to find the least non-standard temporal generalized Herbrand model of a NSTL program.

The new operator \( ; \) implies the use of negation over the time axis. As a result, we do not allow change of the past, nor do we allow querying the future (as a transaction logic) although we are working on valid time database.

The NSTL presentation is followed by a discussion on NSTRA. The semantics is defined in terms of the minimal perfect Herbrand model of NSTL.

## 2 Time domain

**Definition 1.** The time domain is the set \( T = \mathbb{R} \times \mathbb{Z} \), endowed with the natural lexicographic order. Any pair of the form \( (r, z) \), where \( r \in \mathbb{R} \) and \( z \in \mathbb{Z} \), is called a micro-instant. The two components are reached with projection functions \( \pi_R \) and \( \pi_Z \), such that \( \pi_R(r, z) = r \) (the real part) and \( \pi_Z(r, z) = z \) (the integer part). A macro-instant \( r \) consists of all micro-instants whose real part is \( r \).

**Definition 2.** Let \( t, t_0 \) and \( t_1 \) be timestamps in \( T \). Then \([t_0, t_1]\) defines an interval over \( T \): \( \forall t \in T, t_0 \leq t \leq t_1 \rightarrow t \in [t_0, t_1] \). Any micro-instant \( t \) may also be written \([t, t]\). Intervals are normally written as \( \Delta \).

**Definition 3.** The names start and now refer, respectively, to micro-instants that denote the beginning of time and the present time. The interval \( U_T = [\text{start, now}] \), called the universal set of instants, contains all instants of interest.

**Definition 4.** A temporal element \( I \) is a finite union of intervals in \( T \). This set of intervals is closed under finite applications of union, intersection, and complementation; with \( U_T \) as its maximum element, and the empty set as its minimum element, it forms a Boolean algebra. An interval \( \Delta \in I \) is maximal if there does not exist another \( \Delta' \in I \) such that \( \Delta \cup \Delta' \) is an interval.

**Definition 5.** The relevant micro-instants of an interval \([t_0, t_1]\) are simply \( t_0 \) and \( t_1 \). The set of relevant micro-instants of a temporal element \( I \) is the set

\[
\Lambda(I) = \{ t \mid t \text{ is a relevant micro-instant of a maximal interval of } I \}.
\]

Let \( I \subseteq U_T \). Then \( I \) is a temporal element if and only if \( \Lambda(I) \) is finite. A proof can be found in [?].

## 3 Syntax, formulas and entailment

Logic programs without any function symbols, i.e. DATALOG programs [?], are regarded as deductive databases. All predicates in a DATALOG program are guaranteed to represent finite relations, since DATALOG is function-free and so the domain of any Herbrand interpretation of the program is finite.

To the standard signature of Datalog, we add a new logical connector \( ; \), called a serial conjunction, along with the delimiters \( \langle \), \( \rangle \), \( [ \) and \( ] \) (used to specify intervals).

**Definition 6.** Let Delta be an interval, \( r \) be a predicate symbol, \( \mathcal{V} \) be a countably-infinite set of variables, \( \Sigma \) be the NSTL signature, \( T_\Sigma \) be the set of terms for the \( \Sigma \) signature, \( At_\Sigma[\mathcal{V}] \) be the set of atoms of \( \Sigma \) and \( At_\Sigma \) be the set of closed-atoms of \( \Sigma \). Then
\[ r(x_1, ..., x_n) \] and \[ \Delta r(x_1, ..., x_n) \] are, respectively, an atomic formula and a timestamped atomic formula, where \( n \) denotes the arity of the predicate symbol \( r \) and \( \forall i \leq n, x_i \in T_\Sigma \). In the rest of the paper, \( r(\vec{x}) \) is understood as equivalent to \( r(x_1, ..., x_n) \).

- \( \varphi \) is an NSTL formula iff \( \varphi \) is one of:
  - an atomic-formula in \( At_\Sigma[\mathcal{V}] \),
  - a timestamped atomic-formula in \( At_\Sigma[\mathcal{V}] \),
  - one of the following expressions (where \( \phi \) and \( \psi \) are formulas):
    * \( \Delta \phi \), where \( \Delta \) is an time interval in \( \mathcal{T} \).
    * \( \phi \lor \psi, \phi \land \psi, \phi; \psi \), and \( \neg \phi \).
    * \( (\forall X)\phi \) and \( (\exists X)\phi \), where \( X \in \mathcal{V} \) is a variable.

The other logical connectors are composed as usual (“\( \phi \leftarrow \psi \)” is equivalent to “\( \phi \lor \neg \psi \)”). Intuitively, “\( \phi; \psi \)” means, “\( \phi \) is true at some time strictly before \( \psi \) is true”.

4 Perfect-model semantics

The NSTL uses negation, both implicitly and explicitly. The NSTRA “⊕” operator (see section 7), when translated to the “;” operator, is a kind of serialization involving implicit negation. All of this negation may be interpreted as negation as failure. As is standard, the semantics is given using perfect-model semantics [?].

We extend the usual generalized Horn clause with serial conjunction (“;”) and timestamped atomic formulas and introduce the notion of the actual part of a body of a Generalized Horn rule.

Definition 7. A generalized Horn rule is a formula of the form \( p \leftarrow \phi \), where \( p \) is an atomic-formula (maybe timestamped) and \( \phi \) are generalized conjunctions.

Definition 8. Generalized conjunctions are defined recursively as follows:

1. if \( q \) is an atomic-formula and \( \Delta \) is an interval, then \( q, \Delta q, \neg q \) and \( \Delta \neg q \) are generalized conjunctions.
2. if \( \psi_1, ..., \psi_n \) are generalized conjunctions, then so are \( \psi_1; ..., \psi_n \) and \( \psi_1 \land ... \land \psi_n \).

Definition 9. The actual part of a generalized conjunction, \( \varphi \), is \( \Upsilon(\varphi) \), defined recursively as follows:

\[
\begin{align*}
\Upsilon(\varphi) &= \{ \varphi \}, \quad \varphi \text{ is an atomic formula} \\
\Upsilon(\Delta \varphi) &= \Upsilon(\varphi) \\
\Upsilon(\neg \varphi) &= \Upsilon(\varphi) \\
\Upsilon(\varphi \land \psi) &= \Upsilon(\varphi) \cup \Upsilon(\psi) \\
\Upsilon(\varphi; \psi) &= \Upsilon(\psi) 
\end{align*}
\]

Definition 10. The actual part of a body of a Generalized Horn rule \( \phi \leftarrow \psi \) is the value \( \Upsilon(\psi) \).

Definition 11. A generalized Horn temporal base is a finite set of generalized Horn rules.

Definition 12. An NSTL program is a generalized Horn temporal base.

It is common knowledge that the semantics of negation as failure can be problematic when recursion occurs through negation. To avoid this problem, only locally stratified temporal base are considered [?, ?].
Owing to the implied negation of the new serial conjunction operator (“;”), the stratification process must take place over the entire time line. To ensure computability, no change of the past and no querying of the future is allowed. The past is done and the future has not yet taken place (as in transaction time).

The stratification only takes place over a micro-instant, so only the actual part of bodies of generalized Horn rules are examined.

Following [?], to define local stratification, let \( P \) be a temporal base. Then \( P^* \) denotes the ground instantiation of \( P \) (the set of all ground instance of rules in \( P \)). Then we construct a directed graph, \( D(P) \), whose nodes are atoms in \( P^* \). The arcs in the graph are defined as follows:

- There is an arc from \( p \) to \( q \) if and only if \( p \) occurs in the actual part of a body of a rule in \( P^* \) that has \( q \) in the head.
- An arc from \( p \) to \( q \) is negative if and only if \( \neg p \) occurs in the actual part of a body of a rule in \( P^* \) that has \( q \) in the head.

If we release the limitation to the actual part, that would have implied a stratification from start to now. We prefer to state that the past cannot be modified, so we just have the actual part of the body to take care of.

Two partial orders, \( \succeq \) and \( \succ \), are defined over this graph. We write \( q \succeq p \) if there is a directed path from \( p \) to \( q \), and \( q \succ p \) if there is a negative path from \( p \) to \( q \). A path is negative if it has at least one negative arc on it. Finally, a non-standard temporal base, \( P \), is locally stratified if and only if the relation “\( \succ \)” is well-founded, i.e. it has no infinite sequence of atoms such that \( p_1 \succ p_2 \succ \ldots \).

The perfect models of a non-standard temporal base, \( P \), are defined in terms of a preference order, \( \ll \), over the models of \( P \).

**Definition 13.** The expression \( M \ll M' \) holds iff for any atom \( a \), if \( M \models a \) and \( M' \not\models a \), then \( \exists b \succ a, M \not\models b \) and \( M' \models b \).

**Definition 14.** Let \( P \) be an NSTL program. Then a perfect-model of \( P \) is any model that is minimal with respect to \( \ll \). It can be verified that every locally stratified non-standard temporal base has a perfect model. We write \( M(P) \) for the set of perfect models of \( P \).

Entailment, in the sequel, will include negation as failure.

**Definition 15.** Let \( P \) be an NSTL program, \( H \subseteq At_{\Sigma} \) be a Herbrand interpretation of \( P \), and \( t, t', t_0 \) and \( t_1 \) denote micro-instants. Then:

- \( H \models_t r(\vec{x}) \) iff \( \exists [t_0, t_1]r(\vec{x}) \in H \) such that \( t_0 \leq t \leq t_1 \),
- \( H \models_t [t_0, t_1]r(\vec{x}) \) iff \( \exists [t_0', t_1']r(\vec{x}) \in H \) and \( t_1 \leq t \) and \( [t_0, t_1] \subseteq [t_0', t_1'] \),
- \( H \models_t [t_0, t_1]\varphi \) iff \( \exists [t_0', t_1']r(\vec{x}) \in H \) and \( t_1 \leq t \) and \( \forall t' \in [t_0, t_1]H \models_{t'} \varphi \),
- \( H \models_t \neg \varphi \) iff not \( H \models_{t'} \varphi \),
- \( H \models_t (\varphi \land \psi) \) iff \( H \models_{t'} \varphi \) and \( I \models_{t'} \psi \),
- \( H \models_t (\varphi \lor \psi) \) iff \( H \models_{t'} \varphi \) or \( I \models_{t'} \psi \),
- \( H \models_t (\varphi ; \psi) \) iff \( H \models_{t'} \psi \) and \( \exists t' \prec \psi \),
- \( H \models_t \forall X \varphi \) iff for every \( x \in At_{\Sigma}, H \models_{t'} \varphi [x/x] \),
- \( H \models_t \exists X \varphi \) iff there is a \( x \in At_{\Sigma}, H \models_{t'} \varphi [x/x] \).
5 The computation process of a nstl program

The computation made by $T_P$ is split into the 3 steps:

- Fixpoint computation at a specific micro-instant.
- Fixpoint computation at a specific macro-instant.
- Fixpoint computation for all instants.

5.1 Fixpoint computation at a specific micro-instant

**Definition 16.** Let $t$ be a micro-instant, $P$ be a nstl program, $D(P)$ be the directed graph corresponding to the stratification of $P$, atom($\phi$) be the set of atoms of $\phi$, $H$ be a Herbrand base, $V$ be a countably infinite set of variables, $\Sigma$ be the nstl signature, $T_{\Sigma}$ be the set of terms for the $\Sigma$ signature, $Alt_{\Sigma}$ be the set of closed-atoms of $\Sigma$, and $\theta$ be a substitution. In addition, let $\mu$ be the fixpoint operator defined by

$$\mu F(X) = F(X)^{\omega} = \bigcup_{n \geq 0} F^n(X).$$

1. The expression $\Delta_P(n)$ stands for the set of rules in $P$ that have $n$ levels of negated predicates under them. It is defined by:

$$\Delta_P(n) = \{ \phi \leftarrow \psi \in P \mid \forall \alpha \in \text{atom}(\psi), \neg \exists \phi' \leftarrow \psi' \in P - \left( \bigcup_{i=0..n-1} \Delta_P(i) \right) \text{ such that } \exists \theta : V \rightarrow T_{\Sigma} \text{ and } \theta \alpha \succ \theta \phi' \in D(P) \}. $$

2. A time specific version of $T_P$, limited to a specific micro-instant $t$, is the function defined by:

$$T_{P,t}(H) = \{ \theta \phi \mid \phi \leftarrow \psi \in P \text{ and } \theta : V \rightarrow T_{\Sigma} \text{ and } H \models_t \theta \psi \}. $$

3. A fixpoint of a micro-instant is the value $\Gamma_{P,t}(H, \max\{n \mid \Delta_P(n) \neq \emptyset\})$ where $\Gamma$ is a function recursively defined by:

$$\Gamma_{P,t}(H,0) = \mu T_{\Delta_P(0),t}(H),$$
$$\Gamma_{P,t}(H,n) = \mu T_{\Delta_P(n),t}(\Gamma_{P,t}(H,n-1)) \text{ (} n > 0 \text{).}$$

The $\Gamma$ function computes a fixpoint at a specific micro-instant $t$ by recursive application of $T_P$ over all levels of stratification of $P$ when invoked with $\max\{n \mid \Delta_P(n) \neq \emptyset\}$.

5.2 Fixpoint computation at a specific macro-instant

**Definition 17.** Let $t, t_0, t_1$ be micro-instants, $P$ be a nstl program, $H$ be a Herbrand base, and $s = \max\{n \mid \Delta_P(n) \neq \emptyset\}$ be the number of levels of stratification of $P$. In addition, let $r$ be an atom in $H$.

1. The set of potentially relevant timestamps for an nstl program rule $\phi$ is the value $\Lambda_H(\phi)$ defined by:

$$\Lambda_H(r(\vec{x})) = \cup \{ \Lambda(\Delta) \mid \Delta r \theta \in H \},$$
$$\Lambda_H([t_0, t_1]|\varphi) = \{ t_0, t_1 \} \cup ([t_0, t_1] \cap \Lambda_H(\varphi)),$$
$$\Lambda_H(\neg \varphi) = \Lambda_H(\varphi),$$
$$\Lambda_H(\varphi \land \psi) = \Lambda_H(\varphi) \cup \Lambda_H(\psi),$$
$$\Lambda_H(\varphi; \psi) = \Lambda_H(\psi) \cup \{ t + 1 \mid t \in \Lambda_H(\varphi) \},$$
$$\Lambda_H(\varphi \leftarrow \psi) = \Lambda_H(\psi).$$
2. The set of relevant timestamps for an \( \text{nstl} \) program \( P \) for an Herbrand base \( H \) is the value \( \Lambda_P(H) \), where function \( \Lambda_P \) is defined by:

\[
\Lambda_P(H) = \bigcup_{\varphi \in P} (\Lambda_H(\varphi)).
\]

3. A fixpoint of a macro-instant corresponding to the micro-instant \( t \), namely \( \pi_R(t) \), is the value

\[
T_{P,t^+}(H) = \bigcup \{ \Gamma_{P,t+n}^n(H,s) \mid n \geq 0 \land \neg \exists t' \in \Lambda_P(\Gamma_{P,t+n}^n(H,s)) \text{ such that } t' > t + n \land \pi_R(t') = \pi_R(t+n) \}.
\]

\( T_{P,t^+} \) computes a fixpoint for a micro-instant with \( \Gamma_{P,t+n}^n(H,s) \) and then advances to the next micro-instant, where \( n \) varies from 0 to the last relevant timestamp, thus making \( t + n \) equivalent to a running now.

The intervals that have been seen up to now are all rigid, i.e. the two bounds are fixed. However, to resolve recursive rules such as:

\[ [0].A; \quad \text{stating that } A \text{ is true at time } 0 \]
\[ A :- A; \quad \text{stating that } A \text{ is true at } t \text{ iff } A \text{ was true before } t \]

virtual intervals are needed.

**Definition 18.** A virtual interval from \( t \) is the interval \([t, \text{now}]\). At each moment \( t' \) greater than \( t \), this expression defines a new rigid interval \([t, t']\).

If the changes within a macro-instant are finite (which is the case for correct \( \text{nstl} \) programs) then when there are no longer any relevant micro-instants for that macro-instant, we have that:

\[ \neg \exists t' \in \Lambda_P(\Gamma_{P,t+n}^n(H,s)) \text{ such that } t' > t + n \land \pi_R(t') = \pi_R(t+n), \]

and, eventually, all facts in the Herbrand base will become valid either in a closed temporal interval or in a virtual interval. A stable state will be reached, thereby implying a fixpoint.

### 5.3 Fixpoint computations for all instants

As a fixpoint is computable over an entire macro-instant, it is now time to deal with the interaction over macro-instants.

**Definition 19.** Let \( P \) be a \( \text{nstl} \) program.

1. The set of real parts of a set of timestamps \( S \) is the value \( \pi_R(S) \) defined by:

\[
\pi_R(S) = \bigcup_{t \in S} \pi_R(t).
\]

2. The fixpoint of an \( \text{nstl} \) program \( P \) is the least Herbrand perfect model, \( F(P) \), defined by:

\[
F(P) = \bigcup_{t \in M} T_{P,t^+}^n(\emptyset),
\]

where

\[
M = \{\text{start}\} \cup \{\text{start}, \text{now}\} \cap \{m \mid \forall m' \in \pi_R(M'), m = \min\{t \in M' \mid \pi_R(t) = m'\}\},
\]

\[
M' = \bigcup_{i=0..n} \Lambda_P(T_{P,(M_i)}(\emptyset)),
\]

\[
n = \{t'' \in M \mid t'' < t\}.
\]
The function $F$ computes a fixpoint for a macro-instant with $T_{P,t^+}(H)$ and then advances to the beginning of the next relevant macro-instant.

The value $n$ is the ordinal of $t$ in $M$. Since $M$ has an initial element $\text{start}$ the definition of $n$ is well-founded although recursive. The set $M'$ contains all potential relevant timestamps of $P$ and the set $M$ contains all minimal micro-instants (one for each macro-instant).

6 Results about nstl

The following results are standard, so we omit the proofs.

**Theorem 1.** For all $I \in \mathcal{M}(P)$, $I \models_{\text{perf}} P$, where $\models_{\text{perf}}$ is the stratified entailment defined in section 4.

**Theorem 2.** The set $\mathcal{M}(P)$ is closed under model-intersection and $\bigcap \mathcal{M}(P)$ exists.

**Theorem 3.** The set $\bigcap \mathcal{M}(P)$ is computable and equal to $F(P)$.

7 Non-standard temporal algebra

The basis for the work below are the primitives of BLIZZARD, a language invented to express the semantics of timestamped dataflow programming, for the purposes of reactive systems. In BLIZZARD, discrete events take place over a dense time line. Here, variables can be defined everywhere on the non-standard time line, but should only change a finite number of times in a given closed interval.

Let $V$ be any set of values. Let $T$ be the time domain as defined in section 2. A subset $T$ of $T$ is called a date set. A flow $X$ on $V$ dated by $T$ is a pair $(T_X, v_X)$, where $T_X$ is a date set and $v_X : T_X \to V$.

It is important to note that a flow can be finite, even empty. Furthermore, if $X = (T_X, v_X)$ and $t \not\in T_X$, then $X$ has no value at time $t$.

7.1 Operations on flows

Let $X = (T_X, v_X)$ be a flow on $V_X$, $Y = (T_Y, v_Y)$ be a flow on $V_Y$ and, for each $i$, $X_i = (T_{X_i}, v_{X_i})$ be a flow on $V_{X_i}$.

7.1.1 Basic operations

**Constants**

Let $k \in V$ be a constant. Then $k$ is a flow on $V$ equal to $([k, -\infty])$: the value $k$ is available “before the beginning of time”.

**Data operations**

Let $f : V_{X_1} \times \cdots \times V_{X_n} \to V_Z$ be a mapping. Then $Z = f(X_1, \ldots, X_n)$ is a flow on $V_Z$ defined by:

$$T_Z = \bigcap_{i=1}^n T_{X_i},$$

$$v_Z(t) = f(v_{X_1}(t), \ldots, v_{X_n}(t)).$$

At any instant, an operation is only applied to its operands if each of the operands is available exactly at that instant. Should one of the operands not be available, then no operation is made and the available operands disappear into cyberspace.

For the purposes of the relational algebra, the operations $\cap$, $\cup$, $\times$, $-$, $\pi_X$, $\sigma_F$, $\text{sum}_x$, $\text{avg}_x$, $\text{count}$, $\text{max}_x$ and $\text{min}_x$ are all considered mappings.
7.1.2 Reactive temporal operations

The following three operations are called reactive in the sense that they do not actually create any dates. All manipulated dates are provided by the operands.

**Merge**

$Z = X \oplus Y$ is the flow $(T_Z, v_Z)$ on $V_X \cup V_Y$ defined by:

$$
T_Z = T_X \triangle T_Y = (T_X - T_Y) \cup (T_Y - T_X),
$$

$$
v_Z(t) = \begin{cases} 
  v_X(t) & \text{if } t \in T_X, \\
  v_Y(t) & \text{if } t \in T_Y.
\end{cases}
$$

If only one value arrives at time $t$, then $Z$ produces this value; if two values arrive together, then $Z$ discards them.

**Memory**

$Z = X \text{ before } Y$ is the flow $(T_Z, v_Z)$ on $V_X$ defined by:

$$
T_Z = \begin{cases} 
  \emptyset & \text{if } T_X \neq \emptyset, \\
  \inf(T_X) < t & \text{otherwise},
\end{cases}
$$

$$
v_Z(t) = v_X(\sup\{t' \in T_X \mid t' < t\}).
$$

If $Y$ produces a value at time $t$, then $Z$ produces at time $t$ the last value produced by $X$ strictly before $t$. Should $X$ never have produced a value before $t$, there is no input.

**Time-stamp filtering**

$Z = X \mid Y$ is the flow $(T_Z, v_Z)$ on $V_X$ defined by:

$$
T_Z = T_X \cap T_Y,
$$

$$
v_Z(t) = v_X(t).
$$

The flow $Z$ produces those values of $X$ that are simultaneous with values of $Y$.

**Value filtering**

$Z = \varphi_F X$ is the flow $(T_Z, v_Z)$ on $V_X$ defined by:

$$
T_Z = \{t \in T_X \mid v_X(t) \in F\},
$$

$$
v_Z(t) = v_X(t).
$$

If a value arrives at time $t$, then $Z$ produces it only if it is true, otherwise it is discarded.

8 Interpreting queries

Let $A$ be an atomic formula of the form $p(e_1, \ldots, e_n)$, where $p$ is a predicate symbol and all the $e_i$ are terms. Formulas of the form $[t] \text{ present}_p$ and $[t] A$ are called canonical terms. The idea is that $[t] A$ states that $A$ is satisfied at time $t$. As for $[t] \text{ present}_p$, it states that the predicate $p$ is present at time $t$, even if there may be no terms $e_i$ for which $p(e_1, \ldots, e_n)$ is satisfied: present$_p$ is necessary to distinguish instants in which no information about $p$ is available and instants where $p$ is present but never satisfied.

The semantics of NSTL is based on the Herbrand universe. For a given program $P$, the Herbrand universe of $P$ is written $U_P$, which should be finite. A non-standard temporal interpretation $I$
of $P$ assigns each predicate symbol used in $P$ a partial mapping from the collection of instants to finite relations over $U_P$. Let $Pred$ be the set of all predicate symbols appearing in $P$. Then

$$I \in \left[ Pred \to \bigcup_{n \geq 0} \left[ T \to \mathcal{P}(U^n_P) \right] \right],$$

where $[X \to Y]$ means the set of all total functions from $X$ to $Y$; $[X \to Y]$ the set of all partial functions from $X$ to $Y$; $X^n$ the unfolded Cartesian product of $X$ and $\mathcal{P}(X)$ the powerset of $X$.

Given a nstl program $db$ (a non-standard temporal deductive database), a nstra expression contains only those predicate symbols used in $db$ and terms from the Herbrand universe of $db$. Let $E$ be a nstra relation: then $[E](db)$ is the denotation of $E$ with respect to $db$. Therefore,

$$[E] \in \left[ DB \to \bigcup_{n \geq 0} \left[ T \to \mathcal{P}(U^n_P) \right] \right],$$

where $DB$ is the set of nstl programs and $U$ is the set of ground terms of the non-standard temporal logic. The definitions of the denotations of each kind of nstra expression are as follows.

$$[p](db) = (\cap \mathcal{M}(db))(p) \text{ default } (\emptyset \text{ before } ((\cap \mathcal{M}(db))(\text{present}_p)))$$

$$[\nabla_1 E](db) = \nabla_1[E](db)$$

$$[E \nabla_2 E'](db) = [E](db) \nabla_2 [E'](db)$$

where $p$ is a predicate symbol; $\nabla_1$ is a unary nstra operator ($\varphi_F$, $\pi_X$, $\sigma_F$, $\sum_x$, $\text{count}$, $\text{max}_x$ and $\text{min}_x$); and $\nabla_2$ is a binary nstra operator (⊕, before, |, ∩, ∪, × and ($\sim$)).

Since nstra works at the relation level, and nstl works at the tuple level, there is no need for a direct equivalence between nstra expressions and nstl programs. Nevertheless, nstl has an expressive power so similar to nstra that the intensional definition of relations, i.e. nstl programs, may be elaborated with the following guideline in mind:

**NSTRA**

**NSTL Horn clauses**

<table>
<thead>
<tr>
<th>true($A$)</th>
<th>A(true)</th>
<th>A(false) may be needed elsewhere</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \mid B$</td>
<td>$A(\vec{x}), B$</td>
<td>the usual conjunction</td>
</tr>
<tr>
<td>$A \oplus B$</td>
<td>$C(\vec{x}) : A(\vec{x})$$, \neg B$.</td>
<td>either $A$ or</td>
</tr>
<tr>
<td></td>
<td>$C(\vec{x}) : \neg A, B(\vec{x})$.</td>
<td>$B$ is true</td>
</tr>
<tr>
<td>$A$ before $B$</td>
<td>$A(\vec{x}); B$</td>
<td>we are looking for instances of $A$</td>
</tr>
</tbody>
</table>

Since the “⊕” operator is an exclusive-or, it must be translated into an intermediate predicate $C$.

The “;” operator is a logical serial operator meaning that “$A; B$” is true at time $t$ iff $B$ is true at time $t$ and $A$ is true at a time $t' < t$.

The implicit handling of present$_p$ in nstra must be explicit in nstl so the programmer must be able to deal with this situation.

### 9 Discussion

We have defined a temporal database system that supposes a non-standard time line. It is composed of a logic, nstl, essentially datalog with an additional operator, “;”, and of nstra, the relational algebra with three temporal operators.

Queries are made in nstra, which is capable of applying aggregate operations on results provided by the deductive database written in nstl. The temporal operations can be used to make queries about relations with different rates of validity.

The limitation of the current system is that the deductive system cannot compute the future. To do this would require some sort of delay operator; this problem is currently being looked at.
10 Acknowledgements

We would like to thank André Arnold, who came up with the basic primitives in Blizzard.

References

