

Chronolog admits a complete proof procedure

MEHMET A. ORGUN

Department of Computing, Macquarie University

Sydney, NSW 2109, Australia

E-mail: mehmet@mpce.mq.edu.au

and

WILLIAM W. WADGE

Department of Computer Science, University of Victoria

Victoria, B.C. V8W 3P6, Canada

E-mail: wwadge@csr.uvic.ca

Abstract

Chronolog(\mathcal{Z}) is a logic programming language based on a linear-time temporal logic with unbounded past and future. It is suitable for applications involving the notion of dynamic change such as modeling periodical changes, non-terminating computations and temporal databases. The declarative semantics of Chronolog(\mathcal{Z}) programs are given in terms of temporal Herbrand models and the operational semantics in terms of a resolution-type proof procedure called TiSLD-resolution. TiSLD-resolution is based on the axioms and the rules of inference of the underlying temporal logic. It is shown that TiSLD-resolution is sound and complete. The equivalence of the declarative and the operational semantics of Chronolog(\mathcal{Z}) programs is also established.

Keywords: Logic Programming, Temporal Logic, Temporal SLD-Resolution, Semantics.

AMS Subject Classification: 68N17 (Logic Programming).

1 Introduction

Temporal logic has been widely used as a formalism in concurrent program specification and verification [24], modeling temporal databases [25] and various forms of temporal reasoning [33]. In temporal logic [31], the meanings of formulas vary depending on an implicit time parameter and elements from different moments in time can be combined through the use of temporal operators. Therefore temporal logic can model time-dependent and dynamic properties of certain problems in a natural and problem-oriented way. More recently, several researchers have suggested that temporal logic can be directly used as a programming language in applications involving the notion of dynamic change. A number of logic programming languages based on diverse temporal logics have been proposed: Tempura [27] and Tokio [3] are based on interval logic; Templog [2] and Chronolog [35] are based on linear-time temporal logic and Temporal Prolog [15] is based on linear and branching time temporal logics.

Just as the temporal logics these languages are based on, their execution mechanisms differ markedly in the way programs are interpreted and answers are obtained. In Tempura, programs are systematically transformed into a sequence of state descriptions over an interval that satisfies the original program [27, 18]. Templog [2], Temporal Prolog [15] and Chronolog [35] are extensions of logic programming in which temporal logic programs are executed to obtain answers by the use of resolution-type proof procedures. Tokio is based on the same interval logic as Tempura, but mixes both of those execution mechanisms [3, 21].

However, there are still unanswered questions regarding the semantics of temporal logic programming (TLP) languages. Since it is known that first-order temporal logic is incomplete [1], it is important to show that there are fragments of temporal logic which admit a sound and complete proof procedure, and to establish the equivalence of the declarative and operational semantics of specific languages based on such a fragment. We know that, in classical logic programming, these two semantics coincide [34, 4, 23]. Baudinet [6, 7] showed that Templog's proof procedure, which forms the basis of the operational semantics for the language, is sound and complete, and then established the equivalence of the declarative and operational semantics of Templog programs. Orgun and Wadge [30] provided the declarative semantics of Chronolog programs, but they did not consider any specific proof procedure for the language. Gabbay [16] defined a resolution procedure for Temporal Prolog and proved its soundness, but did not offer any completeness results.

In the following, we introduce the temporal language Chronolog(\mathcal{Z}), which is an extension of Chronolog [35], based on a linear-time temporal logic with unbounded past and future. Here the collection of moments in time is modeled by the set \mathcal{Z} of integers. The underlying logic of Chronolog(\mathcal{Z}) has a temporal operator to look into the past (denoted by `prev`) as well as the temporal operators `first` and `next` of Chronolog. The main goal of this paper is to show that Chronolog(\mathcal{Z}) admits a sound and complete proof procedure.

Chronolog(\mathcal{Z}) is suitable for specifying time-dependent properties in a natural way. For instance, the following Chronolog(\mathcal{Z}) program [32] specifies the simulation of a traffic light modeled by the time-varying light predicate.

```
first light(green).
next light(amber) <- light(green).
next light(red) <- light(amber).
next light(green) <- light(red).
```

Program clauses in Chronolog(\mathcal{Z}) programs are interpreted as assertions true at all moments in time. The temporal operator `first` refers to the initial moment in time and `next` the next moment in time. The program says that the traffic light starts with a green light, then on rotates from green to amber, amber to red, red to green and so on. Applications of Chronolog and temporal logic programming in general include: mitigating frame problem [32], modeling non-determinism [28], simulation [26] and temporal deductive databases [29].

In the following, we first outline the underlying temporal logic of Chronolog(\mathcal{Z}) and its formal properties. Section 3 introduces the proof procedure of the language called TiSLD-resolution, which is based on the axioms and rules of inference of the temporal logic. Orgun and Wadge [30] showed that Chronolog programs enjoy the minimum model semantics based on temporal Herbrand interpretations. In Section 4, we extend their results to Chronolog(\mathcal{Z})

programs. In Section 5, we prove the soundness and completeness of TiSLD-resolution by extending a similar result for SLD-resolution [4, 23]. Given the traffic light simulation program above and the goal `<- first next light(Color)`, we can prove by TiSLD-resolution that when `amber` is substituted for the variable `Color`, we obtain a correct answer to the goal. We in particular establish the equivalence of the declarative and the operational semantics of $\text{Chronolog}(\mathcal{Z})$ programs.

2 Temporal Logic of $\text{Chronolog}(\mathcal{Z})$

Temporal logic [31] is a kind of modal logic in which the set of possible contexts models a collection of moments in time (usually discrete, linearly ordered and without a last moment). In the linear-time temporal logic of $\text{Chronolog}(\mathcal{Z})$, the collection of moments in time is the set of integers \mathcal{Z} with its usual ordering relation $<$. The temporal logic offers three temporal operators: `first`, `prev` and `next`. Informally, the temporal operators refer to the initial moment, the previous moment and the next moment in time respectively. We choose a linear-time temporal logic with unbounded past and future as the underlying logic of $\text{Chronolog}(\mathcal{Z})$, because certain time-dependent properties are more natural and easier to express with past operators and the symmetry between past and future operators is best expressed in such a temporal logic. The usefulness of a temporal logic with past operators as well as future operators has been pointed out by many researchers [31, 9, 22, 14].

The syntax of the temporal logic of $\text{Chronolog}(\mathcal{Z})$ extends that of first-order logic with three new formation rules: if A is a formula, so are `first A` , `prev A` and `next A` . Note that the temporal operators are applied to formulas, not to terms of the language. We write `next n` for n successive applications of `next` and `prev n` for n successive applications of `prev`. In case $n = 0$, `next n` and `prev n` are the empty string. From here on, we refer to the underlying logic simply as TL .

2.1 Temporal Interpretations, Semantics

The semantics of formulas of TL are provided by temporal interpretations. A temporal interpretation of TL assigns meanings at all moments in time to all basic elements of the language such as function symbols, predicate symbols and variables. The interpretation is extended upward to all terms and formulas of TL by a satisfaction relation \models . The meaning of a formula of TL varies in time. However, we restrict the discussion to those temporal interpretations in which the values of variables and function symbols are *rigid*. The value of a rigid term is the same at all moments in time. The formal definition of a temporal interpretation is given as follows:

Definition A temporal interpretation I of TL comprises a non-empty set \mathbf{D} , called the domain of the interpretation, over which the variables range, together with for each variable, an element of \mathbf{D} ; for each n -ary function symbol, an element of $[\mathbf{D}^n \rightarrow \mathbf{D}]$; and for each n -ary predicate symbol, an element of $[\mathcal{Z} \rightarrow P(\mathbf{D}^n)]$.

We now give the definition of the satisfaction relation \models in terms of temporal interpretations. Let the notation $\models_{I,t} A$ denote the fact that a formula A is true at a moment t in time in some temporal interpretation I . Let $I(e)$ denote the value in \mathbf{D} that I gives each term e .

Definition The semantics of elements of TL are given inductively by the following, where I is a temporal interpretation of TL , $t \in \mathcal{Z}$, and A and B are formulas of TL .

- (i) If $f(e_0, \dots, e_{n-1})$ is a term, $I(f(e_0, \dots, e_{n-1})) = I(f)(I(e_0), \dots, I(e_{n-1})) \in \mathbf{D}$.
If v is a variable, $I(v) \in \mathbf{D}$.
- (ii) For any n -ary predicate p symbol and terms e_0, \dots, e_{n-1} , $\models_{I,t} p(e_0, \dots, e_{n-1})$
iff $\langle I(e_0), \dots, I(e_{n-1}) \rangle \in I(p)(t)$.
- (iii) $\models_{I,t} \neg A$ iff it is not the case that $\models_{I,t} A$.
- (iv) $\models_{I,t} A \wedge B$ iff $\models_{I,t} A$ and $\models_{I,t} B$.
- (v) $\models_{I,t} (\forall x)A$ iff $\models_{I[d/x],t} A$ for all $d \in \mathbf{D}$ where the interpretation $I[d/x]$ is just like I except that the variable x is assigned the value d in $I[d/x]$.
- (vi) $\models_{I,t} \text{first } A$ iff $\models_{I,0} A$.
- (vii) $\models_{I,t} \text{prev } A$ iff $\models_{I,t-1} A$.
- (viii) $\models_{I,t} \text{next } A$ iff $\models_{I,t+1} A$.

If a formula A is true in a temporal interpretation I at all moments in time, we say that A is true in I or I is a model of A and denote this fact as $\models_I A$. Moreover, $\models A$ denotes the fact that A is true in any temporal interpretation. We use the notation $\Gamma \models A$ to denote the fact that A is true in every model of Γ where Γ is a set of formulas. A temporal interpretation is a model of a set of formulas Γ if it is a model of every formula in Γ . We regard \neg , \wedge and \forall as primitives and assume the usual definitions of \vee , \rightarrow , \leftrightarrow and \exists in terms of these primitives.

2.2 Axioms and Rules of Inference

There is a minimal logic of the given semantics scheme, consisting of all those formulas which are valid in all temporal interpretations under the flow of time described above. Valid formulas of TL are its theorems. Let the notation $\vdash A$ denote the fact that A is a theorem of TL . The notion of deducibility can be characterized in terms of theoremhood: $\Gamma \vdash A$ means that a formula A is deducible from a set Γ of formulas in TL . The following axioms (theorems) state some of the important properties of the temporal operators. Read \leftrightarrow as “if and only if”. Let ∇ stand for any of first , prev and next .

(i) Temporal operator cancellation rules

- E1. $\nabla(\text{first } A) \leftrightarrow \text{first } A$.
- E2. $\text{next prev } A \leftrightarrow A$.
- E3. $\text{prev next } A \leftrightarrow A$.

The first axiom (E1) says that when applied to initial truths, all of `first`, `prev` and `next` are superfluous. In other words, initial truths persist. The axioms E2 and E3 capture the fact that `prev` and `next` are complete inverses. In a linear-time temporal logic with bounded future (past), E2 (E3) would not be valid.

(ii) Temporal operator distribution rules

D1. $\nabla(A \wedge B) \leftrightarrow (\nabla A) \wedge (\nabla B)$.

D2. $\nabla(\neg A) \leftrightarrow \neg(\nabla A)$.

These axioms state that the temporal operators commute with the Boolean operators \wedge and \neg . The temporal operators also naturally commute with the defined operators \vee , \rightarrow and \leftrightarrow . The axioms also capture the fact that the Boolean connectives work point-wise in time.

(iii) Rigidity of variables

V. $\nabla(\forall x)(A) \leftrightarrow (\forall x)(\nabla A)$.

This axiom stipulates that the values of individual variables range over extensions (data values), not intensions (time-varying values). It is an instance of the so-called Barcan formula combined with its converse [19].

(iv) Rules of inference In addition to substitution and Modus Ponens, we have the following temporal operator introduction rules.

R1. If $\vdash A$, then $\vdash \text{first } A$.

R2. If $\vdash A$, then $\vdash \text{prev } A$.

R3. If $\vdash A$, then $\vdash \text{next } A$.

We read the rules of inference as “given A as a theorem, infer $\text{first } A$, $\text{prev } A$ and $\text{next } A$ as theorems”. These rules are instances of the rule of generalization from temporal logic [31].

(v) Induction rule

I. If $\vdash \text{first } A$, $\vdash A \rightarrow \text{next } A$ and $\vdash A \rightarrow \text{prev } A$, then $\vdash A$.

Induction rule is a form of temporal operator elimination rule. An analogous induction rule for Lucid is discussed by Ashcroft and Wadge [5].

The presentation of an axiomatic system for TL begs the question whether it is complete with respect to the given semantic scheme, that is, it axiomatizes the semantics scheme and the flow of time pictured above. We do not attempt to answer this question in this paper. However, it is straightforward to show the correctness (soundness) of the axioms and the rules of inference.

Lemma 1 *All of the axioms and the rules of inference are valid with respect to the given semantics scheme for TL .*

Proof. By the definition of the satisfaction relation \models . For axioms, we need to show that $\vdash A$ implies $\models A$. For rules of inference of the form “if $\vdash A$, then $\vdash B$ ”, we need to show that “if $\models A$, then $\models B$ ”. We omit the details. ■

We assume that the rules of inference given above are extended to consider the notion of deducibility from a set of formulas. For instance, now R1 reads “if $\Gamma \vdash A$, then $\Gamma \vdash \text{first } A$ ”. The formal properties of the deducibility relation are not given in this paper. It should be kept in mind that, under the presence of the rules of inference for the temporal operators, the deduction theorem [10] does not hold for \vdash . If it did, given $A \vdash A$, we could derive $A \vdash \text{first } A$ by R1 and then by the deduction theorem, $\vdash A \rightarrow \text{first } A$. But it can be shown that $A \rightarrow \text{first } A$ is not a valid formula of *TL*.

To show a glimpse of what is involved in an axiomatic proof, we derive the answer to the goal $\leftarrow \text{first next light}(\text{Color})$ from the traffic light simulation program given earlier (call it *P*). This amounts to showing that

$$P \vdash (\exists \text{Color}) \text{first next light}(\text{Color}).$$

Note that \vdash is a reflexive relation, that is, given a set Γ of formulas, $\Gamma \vdash A$ for all $A \in \Gamma$. The steps in the derivation are outlined as follows:

1. $P \vdash \text{next light}(\text{amber}) \leftarrow \text{light}(\text{green})$ (Reflexivity)
2. $P \vdash \text{first}(\text{next light}(\text{amber}) \leftarrow \text{light}(\text{green}))$ (1,R1)
3. $P \vdash \text{first next light}(\text{amber}) \leftarrow \text{first light}(\text{green})$ (2, Axioms of *TL*)
4. $P \vdash \text{first light}(\text{green})$ (Reflexivity)
5. $P \vdash \text{first next light}(\text{amber})$ (3-4, Modus Ponens)
6. $P \vdash (\exists \text{Color}) \text{first next light}(\text{Color})$ (5, \exists -introduction)

In logic programming, we are interested in answer substitutions, rather than a plain “yes” or “no” answer. We have just shown that the atomic formula $\text{first next light}(\text{amber})$ is deducible from the program. Therefore, by substituting *amber* for the variable *Color*, a correct answer to the goal can be obtained.

3 TiSLD-Resolution

In the following, we outline the refutation procedure of $\text{Chronolog}(\mathcal{Z})$ called TiSLD-resolution. Temporal logic programs of $\text{Chronolog}(\mathcal{Z})$ look like ordinary logic programs with the only difference being that program clauses may contain applications of temporal operators to atomic formulas. We do not consider applications of temporal operators to whole program clauses in programs, since all temporal operators can be pushed inside until we reach atomic formulas. We call atomic formulas with a number (possibly 0) of applications of temporal operators as *temporal atoms*. All variables in program clauses are assumed to be universally quantified. For convenience, we use upper-case letters for variables and lower-case letters for function and predicate symbols. A temporal logic program of $\text{Chronolog}(\mathcal{Z})$ is the conjunction of a set of program clauses regarded as assertions true at all moments in time. In the following presentation, we assume familiarity with the concepts of substitution, unification and refutation procedures; see [23] for a detailed discussion.

We need the notion of a *canonical* formula. Persistence of initial truths suggests that the value of an initial truth is an invariant of time. By the temporal operator introduction rules, a sequence of initial truths can be obtained from any given formula. For instance, given a formula A of TL , by systematic applications of R1, R2 and R3, the following initial truths can be formed:

- first A (by R1),
- first prev A (by R2, R1),
- first next A (by R3, R1),
- first prev prev A (by R2, R2, R1),
- first next next A (by R3, R3, R1)

and so on. The value of any of these formulas is an invariant of time. Initial truths obtained from a formula A in the above fashion are called *canonical instances* of A .

The value of a given formula in a temporal interpretation can be expressed in terms of the values of its canonical instances. The intuitive idea is that, for any given moment in time, we can find a canonical instance of the formula fixed to that moment in time and then combine the values of the canonical instances.

Lemma 2 *Let A be a formula and I a temporal interpretation of TL . $\models_I A$ if and only if $\models_I A_t$ for all canonical instances A_t of A .*

Proof. We have that $\models_I A$ iff $\models_{I,t} A$ for all $t \in \mathcal{Z}$ iff $\models_{I,t} \text{first next}^t A$ for all $t \geq 0$ and $\models_{I,t} \text{first prev}^{|t|} A$ for all $t < 0$ iff $\models_I \text{first next}^t A$ for all $t \geq 0$ and $\models_I \text{first prev}^{|t|} A$ for all $t < 0$, because the value of the initial truths $\text{first next}^t A$ and $\text{first prev}^{|t|} A$ are invariants of time, iff $\models_I A_t$ for all canonical instances A_t of A , because the previous clause exhausts them all. ■

TiSLD-resolution (read as “a Timely SLD-resolution”) is an extension of the refutation procedure called SLD-resolution [4]. TiSLD-resolution is applied to a set of *canonical* program clauses and goal clauses. We *assume* that, in canonical instances of program clauses, all superfluous applications of temporal operators are eliminated by axioms E1-E3. And lemma 2 still holds. For a given temporal logic program, canonical instances of program clauses are naturally obtained by rules R1-R3 and axioms D1 and D2. By lemmas 1 and 2, $P \vdash C$ for every canonical instance C of any program clause in P . Goal clauses are assumed to be canonical. When the canonicity restriction is lifted for goal clauses, they can be open-ended. Open-ended goal clauses are used to initiate non-terminating computations [30].

Given a Chronolog(\mathcal{Z}) program P and a goal G , a TiSLD-derivation of $P \cup \{G\}$ consists of a sequence G_0, G_1, \dots of goal clauses where $G_0 = G$, a sequence C_0, C_1, \dots of variants (up to renaming) of canonical instances of program clauses in P and a sequence $\theta_0, \theta_1, \dots$ of substitutions. At every step of a TiSLD-derivation, some temporal atom from the current goal is selected and it is unified with the conclusion (head) of a canonical instance of a program clause after renaming of the variables in the canonical instance. A new goal is produced by replacing the selected temporal atom in the goal by the premise (body) of the canonical instance and then the substitution (mgu) obtained from the unification process is applied to the new goal.

Consider the traffic light simulation program given earlier and the following goal $\leftarrow \text{first next light}(\text{Color})$. The steps in a TiSLD-refutation of the program and the goal are given as follows. The only temporal atom in the goal G_0 is selected and then matched with the head of a canonical instance of the second program clause.

$$\begin{aligned} G_0 &= \leftarrow \text{first next light}(\text{Color}), \\ C_0 &= \text{first next light}(\text{amber}) \leftarrow \text{first light}(\text{green}) \text{ (by R1)}, \\ \theta_0 &= \{\text{Color}/\text{amber}\}. \end{aligned}$$

Then a new goal is produced after replacing the selected temporal atom G_0 by the body of C_0 . The substitution θ_0 is applied to the new goal. The only temporal atom in the new goal G_1 is matched with the first program clause which is already canonical.

$$\begin{aligned} G_1 &= \leftarrow \text{first light}(\text{green}) \theta_0, \\ C_1 &= \text{first light}(\text{green}) \text{ (program clause)}, \\ \theta_1 &= \{\}. \end{aligned}$$

When the selected temporal atom in G_1 is replaced by the body of C_1 , we have that $G_2 = \leftarrow$, meaning that the refutation is successful. A successful TiSLD-derivation is called a TiSLD-refutation. The composition of mgu's θ_0 and θ_1 is regarded as a computed answer substitution for the original goal; indeed, a correct one (see Section 5 for details). Therefore TiSLD-resolution is used not only as a proof procedure to show that the goal $(\exists \text{Color}) \text{first next light}(\text{Color})$ follows from the program, but also as a computational procedure to find an answer substitution for the variable Color .

4 Declarative Semantics

The declarative semantics of logic programs is defined in terms of the minimum Herbrand models [34]. The minimum Herbrand model of a logic program consists of all ground temporal atoms that are logical consequences of the program. Similarly, the declarative semantics of temporal logic programs of $\text{Chronolog}(\mathcal{Z})$ is defined in terms of the minimum temporal Herbrand models [30]. We show that the minimum temporal Herbrand model of a $\text{Chronolog}(\mathcal{Z})$ program consists of all ground canonical temporal atoms that are logical consequences of the program. Orgun and Wadge [30] provided the declarative semantics of Chronolog (with natural numbers as the collection of moments in time). In the following, we extend their results to $\text{Chronolog}(\mathcal{Z})$ programs and therefore no proofs are given for the analogous theorems.

Let P be a $\text{Chronolog}(\mathcal{Z})$ program. Since P is the conjunction of a set of program clauses, we say that P is true in a temporal interpretation I if and only if all program clauses in P are true in I . By lemma 2, a program clause is true in I if and only if all canonical instances of the clause are true in I . Therefore, as far as the declarative semantics is concerned, we can regard P as the set of all canonical instances of the program clauses in P . In the following, we study the declarative semantics of temporal logic programs in terms of a special class of interpretations which we call “temporal Herbrand interpretations”.

4.1 Temporal Herbrand Models

In logic programming theory [34, 23], Herbrand interpretations are models generated by constants and function and predicate symbols used in a given logic program. In temporal logic programming, since the meaning of a predicate symbol varies in time, we need a different approach to generate temporal Herbrand interpretations of a given temporal logic program. By lemma 2, we know that the value of a formula can be expressed in terms of the values of its canonical instances. Furthermore the value of a canonical instance of the formula is an invariant of time. We use this idea to define temporal Herbrand interpretations.

Let P be a $\text{Chronolog}(\mathcal{Z})$ program. The domain of a temporal Herbrand interpretation of P is its Herbrand universe, denoted as U_P , generated by constants and function symbols that appear in P . The temporal Herbrand base B_P of P consists of all those canonical temporal atoms generated by predicate symbols that appear in P with terms in U_P used as arguments. We regard subsets of B_P as temporal Herbrand interpretations of P . This is just a reformulation of temporal interpretations having U_P as their domain. In fact, the correspondence can be established as follows.

Let I be a temporal interpretation of P (or more strictly, of the underlying temporal language of P) with U_P as its domain. Then I is identified with a subset H of B_P by the following.

$$\begin{aligned} \langle e_0, \dots, e_{n-1} \rangle \in I(p)(t) & \text{ iff } \text{first next}^t p(e_0, \dots, e_{n-1}) \in H, \quad t \geq 0, \\ \langle e_0, \dots, e_{n-1} \rangle \in I(p)(t) & \text{ iff } \text{first prev}^{|t|} p(e_0, \dots, e_{n-1}) \in H, \quad t < 0. \end{aligned}$$

The notion of a model naturally extends to subsets of B_P . As with ordinary logic programs, (temporal) Herbrand interpretations are satisfactory for developing the declarative semantics of $\text{Chronolog}(\mathcal{Z})$ programs. In fact, it can be shown that, given a $\text{Chronolog}(\mathcal{Z})$ program and a goal G , $P \cup \{G\}$ is unsatisfiable if and only if $P \cup \{G\}$ has no temporal Herbrand models.

The following lemma says that the entire temporal Herbrand base B_P of a temporal logic program P is a model of the program.

Lemma 3 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program. Then $\models_{B_P} P$.*

As the following lemma shows, the set of temporal Herbrand models of a given $\text{Chronolog}(\mathcal{Z})$ program is also closed under intersection.

Lemma 4 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $M = \{I_\alpha\}_{\alpha \in S}$ be a non-empty set of temporal Herbrand models of P . Then $\cap M = \cap_{\alpha \in S} I_\alpha$ is a temporal Herbrand model of P .*

Since the set of temporal Herbrand models of every $\text{Chronolog}(\mathcal{Z})$ program is non-empty (it at least contains B_P by lemma 3) and the model-intersection property holds for the set by lemma 4, the intersection of all the models in the set is the minimum temporal Herbrand model of the program.

Theorem 5 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $M = \{I_\alpha \mid \models_{I_\alpha} P\}_{\alpha \in S}$ be the set of temporal Herbrand models of P . Then $\text{MMOD}(P) =_{\text{def}} \cap_{\alpha \in S} I_\alpha$ is the minimum temporal Herbrand model of P .*

The following theorem says that the minimum model of every $\text{Chronolog}(\mathcal{Z})$ program consists of all those ground canonical temporal atoms that are logical consequences of the program.

Theorem 6 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program. Then $\text{MMOD}(P) = \{A \in B_P \mid P \models A\}$.*

4.2 Fixed Point Semantics

An alternative approach to the declarative semantics of $\text{Chronolog}(\mathcal{Z})$ programs involves the fixed point theory. Orgun and Wadge [30] provided the fixed point semantics of Chronolog programs based on an extension of the one-step Modus Ponens function T_P originally defined by van Emden and Kowalski [34]. In the following, we extend their results to $\text{Chronolog}(\mathcal{Z})$ programs. The definition of the mapping T_P for $\text{Chronolog}(\mathcal{Z})$ programs is given as follows:

Definition Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $\text{INT}(P)$ denote the set of temporal Herbrand interpretations of P . Let $T_P \in [\text{INT}(P) \rightarrow \text{INT}(P)]$ where, for any $H \in \text{INT}(P)$,

$$T_P(H) = \{ A \mid (A \leftarrow B_0, \dots, B_{n-1}) \text{ is a canonical ground instance of a program clause in } P \text{ and } \{B_0, \dots, B_{n-1}\} \subseteq H \}$$

Let P be a $\text{Chronolog}(\mathcal{Z})$ program. We have that $\text{INT}(P)$ is a complete lattice under the partial order of set inclusion, denoted as $(\text{INT}(P), \subseteq)$. The fixed point semantics of $\text{Chronolog}(\mathcal{Z})$ programs depends on the fact that the temporal Herbrand models of P are characterized by the mapping T_P and T_P over the complete lattice of $\text{INT}(P)$ is continuous and hence monotonic. We say that T_P is continuous if for any ω -chain $\langle C_n \rangle_{n \in \omega}$ of temporal Herbrand interpretations of P ,

$$T_P\left(\bigcup_{n \in \omega} C_n\right) = \bigcup_{n \in \omega} T_P(C_n).$$

Monotonic mappings over complete lattices have fixed points; see [23] for more details on the fixed point theory. We state the following lemma without proof.

Lemma 7 Let P be a $\text{Chronolog}(\mathcal{Z})$ program. Then T_P is continuous and hence monotonic.

The following lemma gives the necessary and sufficient conditions, in terms of T_P , for a temporal Herbrand interpretation to be a model of P . The lemma basically says that the mapping T_P does not improve a given temporal Herbrand interpretation of P if it is already a model.

Lemma 8 Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $H \in \text{INT}(P)$. Then H is a model of P if and only if $T_P(H) \subseteq H$.

We now give the fixed point characterization of the minimum temporal Herbrand model of $\text{Chronolog}(\mathcal{Z})$ programs. The continuity of T_P together with lemma 8 implies the following theorem. First some notation: Let $T_P \uparrow \omega = \bigcup_{n \in \omega} T_P \uparrow n$ where $T_P \uparrow 0 = T_P(\emptyset)$ and $T_P \uparrow n = T_P(T_P \uparrow (n-1))$ for all $n > 0$. Here \emptyset is the minimum element in $(\text{INT}(P), \subseteq)$ and B_P the maximum.

Theorem 9 Let P be a $\text{Chronolog}(\mathcal{Z})$ program. Then $\text{MMOD}(P) = \text{lfp}(T_P) = T_P \uparrow \omega$.

5 Operational Semantics

The operational semantics of $\text{Chronolog}(\mathcal{Z})$ programs are formulated in terms of TiSLD-refutations. Recall that successful TiSLD-derivations are called TiSLD-refutations. Let the notation $P \Vdash A$ denote that there exists a TiSLD-refutation of A from program P via some computation rule R . A computation rule determines the selection of temporal atoms from goal clauses [23]. The success set $SSET(P)$ of a given $\text{Chronolog}(\mathcal{Z})$ program P is defined as the set of all ground canonical temporal atoms that are deducible from the program by TiSLD-resolution.

$$SSET(P) =_{def} \{A \in B_P \mid P \Vdash A\}.$$

The operational semantics of P is characterized by its success set. In this section, we show that $MMOD(P) = SSET(P)$, that is, the declarative semantics and the operational semantics of $\text{Chronolog}(\mathcal{Z})$ programs are equivalent. In doing so, we establish that TiSLD-resolution is a sound and complete proof procedure. Therefore an implementation based on TiSLD-resolution can construct the minimum temporal Herbrand model of a given $\text{Chronolog}(\mathcal{Z})$ program.

This is how the correctness of the implementation is defined.

5.1 Soundness of TiSLD-resolution

We show the soundness of TiSLD-resolution and establish that computed answer substitutions are correct. Let the sequence $E = \{ \langle G_0, C_0, \theta_0 \rangle, \langle G_1, C_1, \theta_1 \rangle, \dots \}$ be a TiSLD-derivation. The relation between any two consecutive elements in E , say E_i and E_{i+1} , is formulated by the following:

$$\begin{aligned} E_i &= \langle \langle \leftarrow A_0, \dots, A_s, \dots, A_{k-1} \rangle, (A \leftarrow B_0, \dots, B_{m-1}), \theta_i \rangle, \\ E_{i+1} &= \langle \langle \leftarrow A_0, \dots, B_0, \dots, B_{m-1}, \dots, A_{k-1} \rangle \theta_i, C_{i+1}, \theta_{i+1} \rangle \end{aligned}$$

where A_s is the selected temporal atom in the goal G_i via the computation rule R and $A_s \theta_i = A \theta_i$ with mgu θ_i .

For any sequence E associated with a successful TiSLD-derivation, we have that for some $n \geq 0$, $G_n = \leftarrow$, in which case the sequence E has length n with the last element $\langle G_{n-1}, C_{n-1}, \theta_{n-1} \rangle$. The composition of the substitutions (mgu's) $\theta_0, \dots, \theta_{n-1}$ is the computed answer substitution. Given a $\text{Chronolog}(\mathcal{Z})$ program and a goal $\leftarrow A_0, \dots, A_{k-1}$, an answer substitution θ is said to be correct for the given goal if $P \models (\forall)(A_0 \wedge \dots \wedge A_{k-1})\theta$. The reason for the universal closure (\forall) is that there might still be variables in $(A_0 \wedge \dots \wedge A_{k-1})\theta$ for which any term from the Herbrand universe U_P can be substituted.

The following theorem establishes the correctness of computed answer substitutions. Without the restriction of TiSLD-resolution to canonical goals, we cannot prove the theorem due to the infinitary nature of open-ended goals. Moreover, in order to prove an open-ended goal from a given program, we may have to use the induction rule and possibly the deduction theorem (for implication introduction). However, the deduction theorem [10] does not hold for \vdash when the rules of inference for temporal operators are assumed. Let $[A]$ denote the set of all ground instances of a given formula. The notation $P \Vdash_{E(n)} A$ means that there exists a TiSLD-refutation of A from P of length n via some computation rule R .

Theorem 10 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $\leftarrow A_0, \dots, A_{k-1}$ a goal. If it is the case that $P \Vdash_{E(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})$ with the sequence $\theta_0, \dots, \theta_{n-1}$ of mgu's, then for all temporal atoms A_i in the goal ($0 \leq i < k$), $[A_i\theta_0 \dots \theta_{n-1}] \subseteq T_P \uparrow n$.*

Proof. The proof of an analogous theorem for (ordinary) logic programming [4, 23] carries over to $\text{Chronolog}(\mathcal{Z})$ programs, once we show that $P \Vdash_{E(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})$ implies that $S \Vdash_{E(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})$ with the same sequence of mgu's for a finite set S of canonical instances of program clauses in P . Then, by induction on the length of the TiSLD-refutation, the conclusion of the theorem can be proved from the ground instances of the canonical clauses in S . We omit the details. ■

The soundness of TiSLD-resolution directly follows from theorem 10. By restricting the soundness result to ground canonical temporal atoms, the ground soundness result can be obtained as a corollary.

Theorem 11 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $\leftarrow A_0, \dots, A_{k-1}$ a goal. If it is the case that $P \Vdash (\exists)(A_0 \wedge \dots \wedge A_{k-1})$, then $P \cup \{\leftarrow A_0, \dots, A_{k-1}\}$ is unsatisfiable.*

Proof. Suppose that $P \Vdash (\exists)(A_0 \wedge \dots \wedge A_{k-1})$. Then theorem 10 implies that for all canonical temporal atoms A_i in $\leftarrow A_0, \dots, A_{k-1}$ ($0 \leq i < k$), there exists a substitution θ such that $A_i\theta \in T_P \uparrow \omega = \text{MMOD}(P)$. Thus $\leftarrow A_0, \dots, A_{k-1}\theta$ is not true in $\text{MMOD}(P)$ and hence not true in any temporal Herbrand model of P . This means that $P \cup \{\leftarrow A_0, \dots, A_{k-1}\}$ has no models. Therefore $P \cup \{\leftarrow A_0, \dots, A_{k-1}\}$ is unsatisfiable. ■

Corollary 12 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program. Then $\text{SSET}(P) \subseteq \text{MMOD}(P)$.*

The soundness of TiSLD-resolution can also be established by converting TiSLD-refutations into axiomatic proofs of TL . Then the correctness of axioms and rules of inference for temporal operators by lemma 1 together with the correctness of Modus Ponens and substitution would imply the soundness of TiSLD-resolution. However, the axiomatic system cannot be used as a natural computational procedure. In fact, theorem 10 reveals the computational nature of TiSLD-resolution.

5.2 Completeness of TiSLD-resolution

In logic programming theory [23, 4], the completeness of SLD-resolution is established by first showing that SLD-resolution is complete when restricted to ground instances of clauses and then by lifting the ground completeness result to predicate logic. Baudinet [6] followed the same strategy with the addition of temporal lifting to establish the soundness and completeness of Templog's proof procedure called TSLD-resolution. Temporal lifting is required for TSLD-resolution, because goal clauses of Templog are open-ended and Templog lacks an analogous operator for *first*. To show the completeness of TiSLD-resolution, we do not need to use temporal lifting, because goal clauses are assumed to be canonical. However, we make use of the equivalence between a given program P and the set of canonical instances of program clauses in P , established by lemma 2.

In the following, we first show that TiSLD-resolution is a complete proof procedure when restricted to canonical ground instances of program clauses. Then the following two lemmas

are used to lift a ground TiSLD-refutation to a TiSLD-refutation. The proofs of the analogous lemmas from logic programming [4, 23] can be adapted to temporal logic programming. The mgu lemma is used in the proofs of the lifting lemma and the ground completeness theorem.

Lemma 13 (mgu) *Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $\leftarrow A_0, \dots, A_{k-1}$ a goal. If it is the case that $P \Vdash_{F(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})$ without the restriction of substitutions to mgu's, then $P \Vdash_{E(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})$. Moreover, if $\theta_0, \dots, \theta_{n-1}$ are the substitutions from the unrestricted refutation $P \Vdash_{F(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})$ and $\sigma_0, \dots, \sigma_{n-1}$ are the mgu's from the refutation $P \Vdash_{E(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})$, then there exists a substitution γ such that $\theta_0 \dots \theta_{n-1} = \sigma_0 \dots \sigma_{n-1} \gamma$.*

Lemma 14 (lifting) *Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $\leftarrow A_0, \dots, A_{k-1}$ a goal. If it is the case that $P \Vdash_{F(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})\theta$ for a substitution θ , then $P \Vdash_{E(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})$. Moreover, if $\theta_0, \dots, \theta_{n-1}$ are the mgu's from the refutation $P \Vdash_{F(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})\theta$ and $\sigma_0, \dots, \sigma_{n-1}$ are the mgu's from the refutation $P \Vdash_{E(n)} (\exists)(A_0 \wedge \dots \wedge A_{k-1})$, then there exists a substitution γ such that $\theta\theta_0 \dots \theta_{n-1} = \sigma_0 \dots \sigma_{n-1} \gamma$.*

The following theorem is the converse of the ground soundness result (corollary 12). It says that, given a $\text{Chronolog}(\mathcal{Z})$ program P , if a ground canonical temporal atom A is a logical consequence of P , then there is a TiSLD-refutation of A from P .

Theorem 15 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program. Then $\text{MMOD}(P) \subseteq \text{SSET}(P)$.*

Proof. We sketch the proof. Suppose $A \in \text{MMOD}(P)$ for some $A \in B_P$. By theorem 9, $A \in T_P \uparrow n$ for some n . We can prove by induction on n that $A \in T_P \uparrow n$ implies that $S \Vdash A$ where S is a finite set of canonical ground instances of the program clauses in P . We also have that $P \vdash S_i$ for any $S_i \in S$ by lemma 1 and the correctness of substitution. This implies that $P \Vdash A$ without the restriction of substitutions to the mgu's in the TiSLD-refutation. By the mgu lemma, $P \Vdash A$. Hence $A \in \text{SSET}(P)$. ■

The following completeness result follows from the ground completeness theorem 15 and lifting lemma 14.

Theorem 16 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $\leftarrow A_0, \dots, A_{k-1}$ a goal. If it is the case that $P \cup \{\leftarrow A_0, \dots, A_{k-1}\}$ is unsatisfiable, then there exists a computation rule R such that $P \Vdash (\exists)(A_0 \wedge \dots \wedge A_{k-1})$.*

Proof. Suppose that $P \cup \{\leftarrow A_0, \dots, A_{k-1}\}$ is unsatisfiable. Then $\text{MMOD}(P)$ is not a model of the goal $\leftarrow A_0, \dots, A_{k-1}$. Hence some ground instance of $\leftarrow A_0, \dots, A_{k-1}$ is false in $\text{MMOD}(P)$. Let $\leftarrow B_0\theta, \dots, B_{k-1}\theta$ be such an instance. We have that all B_i 's are canonical temporal atoms. Then $\{B_0\theta, \dots, B_{k-1}\theta\} \subseteq \text{MMOD}(P)$. By the ground completeness theorem 15, $P \Vdash B_i\theta$ for all $0 \leq i < k$. Since all $B_i\theta$'s are variable-free, by combining TiSLD-refutations for all $B_i\theta$'s, we obtain $P \Vdash (\exists)(A_0 \wedge \dots \wedge A_{k-1})\theta$. Therefore, by lifting lemma 14, we have that $P \Vdash (\exists)(A_0 \wedge \dots \wedge A_{k-1})$. ■

Let P be a $\text{Chronolog}(\mathcal{Z})$ program. We have shown that for any $A \in B_P$, $A \in \text{MMOD}(P)$ implies that $P \Vdash A$ (theorem 15) and $A \in \text{SSET}(P)$ implies that $P \models A$ (corollary 12). Combining these results together, we conclude that the declarative and operational semantics of $\text{Chronolog}(\mathcal{Z})$ programs coincide.

Corollary 17 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program. Then $\text{MMOD}(P) = \text{SSET}(P)$.*

A stronger completeness result for logic programming [23, 4] attributed to Clark shows that a correct answer substitution is an instance of a computed answer substitution. It is not possible to show that every correct answer substitution can be computed by a TiSLD-refutation, because computed answer substitutions are obtained from mgu's in TiSLD-refutations. Just as the other results such as the soundness completeness of SLD-resolution, the strong completeness result can be extended to TiSLD-resolution. We state the following theorem without proof.

Theorem 18 *Let P be a $\text{Chronolog}(\mathcal{Z})$ program and $\leftarrow A_0, \dots, A_{k-1}$ a goal. For every correct answer substitution θ for $P \cup \{\leftarrow A_0, \dots, A_{k-1}\}$, there exists a computation rule R , a computed answer substitution σ for $P \cup \{\leftarrow A_0, \dots, A_{k-1}\}$ and a substitution γ such that $\theta = \sigma\gamma$.*

6 Conclusions

We have shown that $\text{Chronolog}(\mathcal{Z})$ admits a sound and complete proof procedure called TiSLD-resolution. The operational semantics of a given $\text{Chronolog}(\mathcal{Z})$ program is characterized by its success set consisting of all those ground canonical temporal atoms for which a TiSLD-refutation exists. We have established the equivalence of the declarative and operational semantics of $\text{Chronolog}(\mathcal{Z})$ programs. An implementation of $\text{Chronolog}(\mathcal{Z})$ based on TiSLD-resolution therefore can construct the minimum model of a given program.

$\text{Chronolog}(\mathcal{Z})$ is based on a simpler temporal logic than those of the temporal languages Templog [2] and Temporal Prolog [15], and consequently, it has a simple and smooth theory. However, Temporal Prolog's proof procedure described in detail by Gabbay [16] works for branching-time temporal logics as well as linear-time temporal logics. Baudinet [6, 7] showed the soundness and the completeness of the proof procedure of Templog [2] called TSLD-resolution. Templog is based on a linear-time temporal logic with the temporal modalities "henceforth" and "sometime", but with neither past operators nor an analogous operator for first. Thus TSLD-resolution cannot be directly applied to $\text{Chronolog}(\mathcal{Z})$.

Other applications of non-classical logics to logic programming include the interval languages Tokio [3] and Tempura [27] and the multi-dimensional language InTense [26] which has an arbitrary number of temporal and spatial dimensions. InTense restricted to a time-dimension is very similar to $\text{Chronolog}(\mathcal{Z})$. Molog [11] is a framework for modal logic programming in which a specific modal resolution method is required for each particular modal logic used. Jackson and Reichgelt [19] gives a more general proof method for modal predicate logic which can also be applied to Molog. Brzoska [8] showed that Templog can be regarded as an instance of the CLP scheme of Jaffar and Lassez [20] over a suitable algebra, which suggests that $\text{Chronolog}(\mathcal{Z})$ can also be regarded as an instance of the CLP. For more details on temporal and modal logic programming and their applications, we refer the reader to the literature, for example, see [17, 12].

Implementations of $\text{Chronolog}(\mathcal{Z})$ rely on TiSLD-resolution for correctness. For efficiency, we must combine features of logic programming implementations (unification, backtracking) with features of dataflow implementations (associative memory, tagging) such as those of [13]. Any computation initiated by a goal may require that some sub-goals be proved more than once during

the course of the computation due to the time-dependent nature of the definitions of predicates in programs. Time-dependencies may be backward or forward in time or a combination of both. Canonical temporal atoms that are proved at a certain time may be stored in an associate memory, labeled with corresponding time-tags, so that they can be retrieved from the memory whenever they need to be proved again. However, an effective memory management scheme is required to reduce the amount of space used without sacrificing much of the efficiency of the implementation.

References

- [1] M. Abadi. The power of temporal proofs. *Theoretical Computer Science*, 65(1989):35–83, 1989.
- [2] M. Abadi and Z. Manna. Temporal logic programming. In *Proceedings of the 1987 Symposium on Logic Programming*, pages 4–16, San Fransisco, Calif, 1987. IEEE Computer Society Press.
- [3] T. Aoyagi, M. Fujita, and T. Moto-oka. Temporal logic programming language Tokio. In E. Wada, editor, *Logic Programming'85*, pages 138–147. Springer-Verlag, 1986. LNCS 221.
- [4] K. R. Apt and M. H. van Emden. Contributions to the theory of logic programming. *Journal of the Association for Computing Machinery*, 29:841–862, July 1982.
- [5] E. A. Ashcroft and W. W. Wadge. Lucid – a formal system for writing and proving programs. *SIAM Journal on COMPUTING*, 5:336–54, September 1976.
- [6] M. Baudinet. Temporal logic programming is complete and expressive. In *Conference Record of the Sixteenth ACM Symposium on Principles of Programming Languages*, pages 267–280, Austin, Texas, January 1989. The Association for Computing Machinery.
- [7] M. Baudinet. A simple proof of the completeness of temporal logic programming. In L. Fariñas del Cerro and M. Penttonen, editors, *Intensional Logics for Programming*, pages 51–83. Oxford University Press, 1992.
- [8] C. Brzoska. Temporal logic programming and its relation to constraint logic programming. In V. Saraswat and K. Ueda, editors, *Proceedings of the 1991 International Logic Programming Symposium*, pages 661–677, San Diego, Calif, October 28-31 1991.
- [9] J. P. Burgess. Basic tense logic. In D. M. Gabbay and F. Guethner, editors, *Handbook of Philosophical Logic, Vol. II*, pages 89–134. D. Reidel Publishing Company, 1984.
- [10] B. F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.
- [11] L. Fariñas del Cerro. MOLOG: A system that extends PROLOG with modal logic. *New Generation Computing*, 4:35–50, 1986.
- [12] L. Fariñas del Cerro and M. Penttonen, editors. *Intensional Logics for Programming*. Oxford University Press, 1992.
- [13] A. A. Faustini and W. W. Wadge. An eductive interpreter for pLucid. Technical Report TR-006-86, Department of Computer Science, Arizona State University, 1986.
- [14] A. A. Faustini and W. W. Wadge. Intensional programming. Technical Report DCS-55-IR, Department of Computer Science, University of Victoria, Victoria, B.C., Canada, June 1986.
- [15] D. M. Gabbay. Modal and temporal logic programming. In A. Galton, editor, *Temporal Logics and Their Applications*, pages 197–237. Academic Press, 1987.
- [16] D. M. Gabbay. A temporal logic programming machine : Modal and temporal logic programming, Part 2. Department of Computing, Imperial College, November 1989.
- [17] A. Galton, editor. *Temporal Logics and Their Applications*. Academic Press, 1987.

- [18] R. Hale. Temporal logic programming. In A. Galton, editor, *Temporal Logics and Their Applications*, pages 91–119. Academic Press, 1987.
- [19] P. Jackson and H. Reichgelt. A general proof method for modal predicate logic. In P. Jackson, H. Reichgelt, and F. van Harmelen, editors, *Logic-Based Knowledge Representation*, pages 177–228. MIT Press, 1989.
- [20] J. Jaffar and J-L. Lassez. Constraint logic programming. In *Conference Record of the Fourteenth ACM Symposium on Principles of Programming Languages*, pages 111–119, Munich, Germany, 1987. The Association for Computing Machinery.
- [21] S. Kono, T. Aoyagi, M. Fujita, and H. Tanaka. Implementation of temporal logic programming language Tokio. In E. Wada, editor, *Logic Programming'85*, pages 138–147. Springer-Verlag, 1986. LNCS 221.
- [22] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In Rohit Parikh, editor, *Logics of Programs*, LNCS 193, pages 196–218. Springer-Verlag, 1985.
- [23] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1984.
- [24] Z. Manna and A. Pnueli. Verification of concurrent programs: the temporal framework. In Boyer and Moore, editors, *Correctness Problem in Computer Science*, pages 215–273. Academic Press, 1981.
- [25] E. McKenzie. Bibliography: Temporal databases. *SIGMOD Record*, 15:40–52, December 1986.
- [26] W. H. Mitchell and A. A. Faustini. The intensional logic language InTense. In *Proceedings of the 1989 International Symposium on Lucid and Intensional Programming*, pages 70–83, Arizona State University, May 8 1989.
- [27] B. Moszkowski. *Executing Temporal Logic Programs*. Cambridge University Press, 1986.
- [28] M. A. Orgun and W. W. Wadge. A formal treatment of non-deterministic dataflow streams in intensional logic programming. In *Proceedings of the 1989 International Symposium on Lucid and Intensional Programming*, pages 55–69, Arizona State University, May 8 1989.
- [29] M. A. Orgun and W. W. Wadge. A relational algebra as a query language for Temporal DATALOG. In A. M. Tjoa and I. Ramos, editors, *Proceedings of DEXA '92, The Third International Conference on Database and Expert Systems Applications*, pages 276–281, Valencia, Spain, September 2–4 1992. Springer-Verlag Wien.
- [30] M. A. Orgun and W. W. Wadge. Theory and practice of temporal logic programming. In L. Fariñas del Cerro and M. Penttonen, editors, *Intensional Logics for Programming*, pages 23–50. Oxford University Press, 1992.
- [31] N. Rescher and A. Urquhart. *Temporal Logic*. Springer-Verlag, 1971.
- [32] D. Rolston. Toward a tense-logic-based mitigation of the frame problem. In F. M. Brown, editor, *Proceedings of the 1987 Workshop on the Frame Problem in AI*, Lawrence, Kansas, April 1987. Morgan Kaufmann, Los Altos, Calif.
- [33] F. Sadri. Three approaches to temporal reasoning. In A. Galton, editor, *Temporal Logics and Their Applications*, pages 121–168. Academic Press, 1987.
- [34] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the Association for Computing Machinery*, 23:733–42, 1976.
- [35] W. W. Wadge. Tense logic programming: a respectable alternative. In *Proceedings of the 1988 International Symposium on Lucid and Intensional Programming*, pages 26–32, Sidney, B.C., April 7–8 1988.