May 1988

Report No. STAN-CS-884203

On the Semantics of Temporal Logic Programming (Preliminary Report)

by

Marianne Baudinet

Department of Computer Science

Stanford University

Stanford, California 94305

On the Semantics of Temporal Logic Programming (Preliminary Report)

Marianne Baudinet
Computer Science Department
Stanford University

May 1988

Abstract

Recently, several researchers have suggested directly exploiting in a programming language temporal logic's ability to describe changing worlds. The resulting languages are quite diverse. They are based on different subsets of temporal logic and use a variety of execution mechanisms. So far, little attention has been paid to the formal semantics of these languages. In this paper, we study the semantics of an instance of temporal logic programming, namely, the TEMPLOG language defined by Abadi and Manna. We first give declarative semantics for TEMPLOG, in model-theoretic and in fixpoint terms. Then, we study its operational semantics and prove soundness and completeness theorems for the temporal-resolution proof method underlying its execution mechanism.

This research was supported by the National Science Foundation under Grants DCR-84-13230 and DCR-86-11272, by the Defense Advanced Research Projects Agency under Contract N00039-84-C-0211, and by the United States Air Force Office of Scientific Research under Contract AFOSR-87-0149.

1 Introduction

Temporal logic is more and more widely acknowledged as a useful formalism for program specification and verification. It has been used quite extensively for concurrent programs and digital hardware, but it is also applicable whenever it is necessary to specify or describe a sequence of states or events, such as in robot planning or historical databases. Recently, the idea has emerged that one could more easily use the expressive power of temporal logic if it could be made directly executable, for instance as is done with first order logic in PROLOG. This has lead to the definition of a number of programming languages based on temporal logic ([FKTM86], [Mos86], [AM87], [Gab87], [Wad88], [OW88a]).

The earliest of these languages, the TEMPURA language of [Mos84,Mos86] is based on a subset of interval temporal logic whose formulas can be interpreted as traditional imperative programs. In logical terms, executing a TEMPURA formula (program) amounts to building a model for that formula. The TOKIO language of [FKTM86] is an extension of logic programming, but resembles TEMPURA in the way it treats its temporal constructs. The other temporal programming languages ([Aba87], [AM87], [Gab86,Gab87], [Wad85,Wad88], [OW88a]) are based on the logic programming paradigm and view an execution of a program as a type of refutation proof.

For this last class of languages, important questions are left unanswered. Among these is the relation between their operational and their logical semantics. Indeed, in classical logic programming, the operational and the logical semantics coincide because of the completeness of SLD-resolution ([Hil74], [Cla79], [AvE82]). Unfortunately, first-order temporal logic is inherently incomplete ([Aba87]). So, one could very well expect that the operational and the logical semantics of temporal programming languages do not and even cannot coincide.

In this paper, we examine this question for the TEMPLOG language of [AM87]. We capture both the operational and the declarative semantics of this language and prove that they coincide, hence proving that the fragment of temporal logic defined by TEMPLOG admits a complete proof system.

TEMPLOG extends classical Horn logic programming to allow specific use of the temporal operators 0 (next), •I (always), and 0 (eventually). Programs are sets of temporal clauses, and computations are proofs by refutation. The proof method used is a resolution met hod for temporal logic to which we refer as TSLD-resolution. We study the declarative (logical) semantics of TEMPLOG and define it both in model-theoretic terms and in fixpoint terms. For this, we define the notion of temporal Herbrand interpretation and of temporally ground formulas. We prove that the declarative semantics of a program is characterized by the minimal Herbrand model of the program. We then show how to associate with a TEMPLOG program a mapping whose least

fixpoint is the minimal Herbrand model. This provides a fixpoint characterization of the declarative semantics. Next, we examine the TSLD-resolution method that is the basis of the operational semantics of TEMPLOG. We establish a correspondence between membership in the fixpoint of the mapping associated with programs and existence of a temporally ground resolution proof, thereby obtaining a type of ground completeness theorem. From this result, we establish the completeness of TSLD-resolution using a temporal lifting lemma. Our proof techniques extend those that have been used for giving semantics to classical logic programming ([vEK76], [Cla79], [AvE82], [Llo84], [Apt87]).

2 The Temporal Logic

Temporal logic is a modal logic for reasoning about various time instants or states. There are several versions of temporal logic. We are interested in a version of the logic in which formulas are interpreted over linear sequences of discrete states (linear-time temporal logic). In the following we refer to this logic simply as temporal logic.

The language of first-order temporal logic is an extension of the language of first-order classical logic that allows the application of temporal operators to terms and formulas. In the general case, an interpretation for temporal logic formulas is a sequence of classical interpretations. So, temporal terms may have a different interpretation at each time instant and temporal formulas have a truth value that may also vary with time. In the temporal logic of interest here, the constant and function symbols are assumed to have an interpretation that does not depend on time. They are said to be *rigid*. Only predicate symbols may be *flexible*, that is, may have time-varying interpretations. The temporal operators allowed in the language are 0 (next), •I (always) and 0 (eventually). Let F be a formula; intuitively, 0 F means that F is true in the next state; ❑ F means that F is true in all the following states; and 0 F means that F is true in some following state. The O-operator is the dual of □, that is, 0 $F \equiv \neg \bullet I \neg F$.

Let us begin with the syntax of the logic. We consider a language L of temporal logic to be characterized by a set of constant, function, and predicate symbols, as well as a countable number of variables. The function and predicate symbols have an associated arity. The *terms* of the language L are defined as follows.

- Variables are terms.
- Constant symbols are terms.

- If f is a k-ary function symbol and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term.

The class of ***formulas*** of the language L is defined by the following.

- If p is an ℓ-ary predicate symbol and $t_1, \ldots, t_\ell$ are terms, then $p(t_1, \ldots, t_\ell)$ is a formula, also called an ***atom.***
- Let F and G be formulas, and X a variable. Then $\neg F$, *F A G, F* $\rightarrow$ *G, 0 F,* $\Box$ *F, 0 F,* (VX)F are formulas.

An atom preceded by any number of O's is referred to as a ***next-atom.*** If *A* is an atom, we denote by $\bigcirc^i A$ the next-atom obtained by applying i times the O-operator to *A.* If Xi,. . . , X_n are the free variables of a formula *F,* then we denote by $(\forall *)\, F$ the universal closure of *F,* that is, the closed sentence $(\forall X_1) \cdots (\forall X_n) F$.

A *temporal interpretation* $\mathcal{I}$ for a language L of temporal logic is a quadruple $\langle D, \Sigma, \alpha, J \rangle$ such that:

- *D* is a non-empty set, referred to as the ***domain;***
- Σ is a *sequence of states* $\sigma_0, \sigma_1, \sigma_2, \ldots$ that is isomorphic to w (σ_0 is the *initial state*);
- α is an ***assignment*** to variables, that is, a function mapping each variable of L to an element of D;
- J is an ***interpretation,*** that is, a function that maps
  - every constant symbol of L into an element of *D,*
  - every k-ary function symbol of L into a function from D^k to *D,*
  - every ℓ-ary predicate symbol of L into a relation on D^ℓ for every state σ_i ($i \in$ w).

We informally say that $\mathcal{I}$ is a temporal interpretation of a formula *F* if $\mathcal{I}$ is a temporal interpretation of the language of *F.* Let *d* be an element of *D.* We denote by $\mathcal{I} \cdot (\mathrm{X} \leftarrow d)$ the temporal interpretation obtained from $\mathcal{I}$ by modifying the assignment so that the variable X is mapped to *d.* If i is a natural number, $\mathcal{I}^{(i)}$ is the temporal interpretation obtained from $\mathcal{I}$ by taking the initial state to be σ_i. In other words, $\mathcal{I}^{(i)}$ is the interpretation $\mathcal{I}$ whose sequence of states has been truncated of its i first states; the sequence of states for $\mathcal{I}^{(i)}$ is indeed $\sigma_i, \sigma_{i+1}, \sigma_{i+2}, \ldots$

The meaning of L with respect to a temporal interpretation $\mathcal{I} = \langle D, \Sigma, \alpha, J \rangle$ is given by a function $\mathcal{T}_\mathcal{I}$ that provides the meaning of the terms, and the satisfaction relation $\models_\mathcal{I}$ for the formulas of L. They are defined inductively as follows. Let X be a variable of L, a a constant symbol, f a k-ary function symbol, and p an ℓ-ary predicate symbol.

- $\mathcal{T}_{\mathcal{I}}[X] = \alpha[X]$,
- $\mathcal{T}_{\mathcal{I}}[a] = J[a]$,
- $\mathcal{T}_{\mathcal{I}}[f(t_1, \ldots, t_k)] = J[f](\mathcal{T}_{\mathcal{I}}[t_1], \ldots, \mathcal{T}_{\mathcal{I}}[t_k])$, where $t_1, \ldots, t_k$ are terms.

The satisfaction relation $\models_{\mathcal{I}}$ is defined by:

- $\models_{\mathcal{I}} \mathit{true}$,
- $\not\models_{\mathcal{I}} \mathit{false}$,
- $\models_{\mathcal{I}} p(t_1, \ldots, t_\ell)$ iff $J[p][\sigma_0](\mathcal{T}_{\mathcal{I}}[t_1], \ldots, \mathcal{T}_{\mathcal{I}}[t_\ell])$, where $t_1, \ldots, t_\ell$ are terms,
- $\models_{\mathcal{I}} \neg F$ iff $\not\models_{\mathcal{I}} F$,
- $\models_{\mathcal{I}} F \wedge G$ iff $\models_{\mathcal{I}} F$ and $\models_{\mathcal{I}} G$,
- $\models_{\mathcal{I}} F \rightarrow$ G iff $\models_{\mathcal{I}}$ G whenever $\models_{\mathcal{I}} F$,
- $\models_{\mathcal{I}}$ (VX)F iff for every element d of D: $\models_{\mathcal{I}\cdot\langle X \leftarrow d\rangle} F$,
- $\models_{\mathcal{I}} \bigcirc F$ iff $\models_{\mathcal{I}^{(1)}} F$,
- $\models_{\mathcal{I}} \Box F$ iff for every i in w: $\models_{\mathcal{I}^{(i)}} F$,
- $\models_{\mathcal{I}} \Diamond F$ iff for some i in w: $\models_{\mathcal{I}^{(i)}} F$.

Notice that for any formula F and any i in w, we have: $\models_{\mathcal{I}^{(i)}} F$ iff $\models_{\mathcal{I}} \bigcirc^i F$. This equivalence follows directly from the definition of $\mathcal{I}^{(i)}$ and of the satisfaction relation $\models_{\mathcal{I}}$ for formulas of the form 0 G. We can use it to rephrase as follows the satisfaction condition for formulas of the form $\Box F$ and $\Diamond F$.

- $\models_{\mathcal{I}} \Box F$ iff for every i in w: $\models_{\mathcal{I}} \bigcirc^i F$,
- $\models_{\mathcal{I}} \Diamond F$ iff for some i in ω: $\models_{\mathcal{I}} \bigcirc^i F$.

A temporal interpretation that satisfies a formula is said to be a *model* of the formula. A formula is ***satisfiable*** if it has a model. It is *valid* if it is satisfied by all temporal interpretations. A formula G is a *logical consequence* of a formula F, denoted $F \models$ G, if G is satisfied by every model of F. The following are valid equivalence schemata:

- $\bigcirc(F \wedge G) \equiv (\bigcirc F \wedge \bigcirc G)$,

- $\bigcirc \Diamond F \equiv \Diamond \bigcirc F$,
- $\Diamond \Diamond F \equiv \Diamond F$,
- $\Diamond(\Diamond F \wedge \Diamond G) \equiv (\Diamond F \wedge \Diamond G)$.

3 The TEMPLOG Language

We now define the logic programming language TEMPLOG that is based on the temporal logic introduced in the previous section. We begin with its syntax. Then we describe the proof method underlying its execution.

3.1 Syntax

Let L be a language, that is, a collection of variables and of constant, function, and predicate symbols. The TEMPLOG terms and formulas of L are the terms and formulas of L as defined for the temporal logic described in Section 2. Terms are assumed to be rigid. Only predicate symbols are flexible. We use strings of letters with the first letter capitalized to denote variables (usually X, $Y, Z, \ldots$), and strings of lower case letters to denote constant, function, and predicate symbols. The syntax of TEMPLOG clauses can be given as follows (":::=" and "|" are symbols of the metalanguage). Let A denote an atom and N a next-atom.

- Body: B : := $\varepsilon \mid A \mid B_1, B_2 \mid 0\ B \mid OB$ where ε denotes the empty body.
- Initial clause: IC : := $N \leftarrow B \mid UN \leftarrow B$
- Permanent clause: PC : := $\Box$ $(N \leftarrow B)$
- Program clause: C : := $IC \mid PC$
- Goal clause: G : := $\leftarrow B$

Program and goal clauses are implicitly universally quantified. A TEMPLOG program consists of a set of program clauses, that is, a conjunction of program clauses. In a body, the comma stands for the conjunction operator (we will use "," and "A" interchangeably in the semantic development). A program clause that has an empty body is also called a *fact.* An empty body corresponds to "true". A goal clause can also be seen as an initial clause with an empty head, where the empty head corresponds to "false". Hence, a goal of the form $\leftarrow B$ with free variables $X_1, \ldots, X_n$

corresponds to the formula $(\forall X_1)\cdots(\forall X_n)\neg B$, that is, $\neg(\exists X_1)\cdots(\exists X_n)B$ (we use "$\leftarrow B$" and "$\neg B$" interchangeably in the semantic development).

Throughout this paper, we use the symbol A to denote an atom, N for a next-atom, B for a body (empty or not), C for a clause, P for a program, and G for a goal.

Example: The following simple program defines a predicate p such that $p(X)$ is true at time i for X = $s^{2i}(a)$.

$$p(a) \leftarrow$$
$$\Box\left(\bigcirc\, p(s(s(X))) \leftarrow P(X)\right) \blacksquare$$

Example: The following two clauses describe a relation *reach* that is true at time i of the nodes in a graph that can be reached by going through i arcs. We assume that a number of facts describe the configuration of the graph *(arc)* and the initial location in the graph (at).

$$reach(X) \leftarrow at(X)$$
$$\Box\left(\bigcirc\, reach(X) \leftarrow reach(Y),\ arc(Y, X)\right)$$

One can also define the predicate *reachable* that is true at time j of all the locations that can be reached at a time later than j.

$$\Box\left(reachable(X) \leftarrow \bigcirc\, reach(X)\right) \blacksquare$$

The class of non-empty TEMPLOG bodies has been defined as the smallest set containing all the atoms and closed under conjunction and application of 0 and 0. However, the valid equivalence schemata listed at the end of Section 2 show that syntactically distinct formulas may represent logically equivalent bodies. We thus introduce a canonical form for the bodies.

Definition (Canonical body) A *canonical body* is a body whose occurrences of 0 are pushed all the way inwards and in which every next-atom is in the scope of the least possible number of $\Diamond$'s.

Each body has a unique equivalent canonical form (up to commutativity and associativity of the conjunction). It can be obtained by performing the following replacements iteratively until no further replacement is possible:

- replace $\bigcirc(F \wedge G)$ with $(\bigcirc F \wedge \bigcirc G)$,
- replace $\bigcirc\bigcirc F$ with $\bigcirc F$,

- replace 0 0 F with 0 F,
- replace $\Diamond(\Diamond F$ A OG) with (0 F A

Example: The goal $\leftarrow \Diamond\Big(p(X,Y), \bigcirc\Diamond\Big(\Diamond q(Y), \Diamond r(X,Z)\Big), \Diamond\Diamond\Big(s(Y,Z),$ has as canonical form the goal $\leftarrow \Diamond\Big(p(X,Y), \Diamond\bigcirc q(Y), \Diamond\bigcirc r(X,Z), \Diamond\Big(s(Y,Z), t(Z)\Big)\Big)$. ■

Throughout this paper, when referring to a body, we mean to refer to its canonical form. Notice that in the canonical form of a body, each occurrence of 0 always has in its scope at least one next-atom that is in the scope of **no** other 0. If this was not the case, one could use $\Diamond(\Diamond F$ A 0 G) $\equiv$ 0 F A 0 G to further reduce it, and so it would not be canonical. This observation is used in the proof procedure for TEMPLOG.

We assume that all the predicate symbols are flexible. This is not restrictive as the time-independence of a predicate can easily and efficiently be expressed in TEMPLOG. The following clause is sufficient to state that the ℓ-ary predicate p is rigid:

❑ $p(X_1, \ldots, X_\ell) \rightarrow \Diamond p(X_1, \ldots, X_\ell)$.

3.2 Proof Method

As described previously, a TEMPLOG program is a set of statements in temporal logic. Given such a program, a computation will consist in trying to derive some information that logically follows from the program. These inferences rely on proofs by contradiction using temporal resolution rules. Let us consider the program P and the body B with free variables Xi,. . . , X,. A computation attempts to prove that B follows from P for a certain instantiation of $X_1, \ldots,$ X,, by considering P -together with the goal $\leftarrow B$ (that is, the negation of B) and trying to derive a contradiction. When a refutation is obtained, it is usually for a certain instantiation of the variables Xi,. . . , X_n of B, referred to as an *answer substitution.* An answer substitution is said to be *correct* if the universal closure of $B\theta$ is a logical consequence of P.

Note; We assume some familiarity with the notions of substitution and unification (see e.g. [Rob65], [MW81], [Ede85], [LMM86]). We informally represent a substitution θ by a set of replacement pairs, denoted $\{X_1 \leftarrow t_1, \ldots, X_n \leftarrow t_n\}$, where each X_i is a variable and each t_i is a term, $X_i \neq t_i$, and $X_i \neq X_j$ when $i \neq$ j. The composition of the substitution θ_1 with the substitution θ_2, denoted $\theta_1 \circ \theta_2$, is such that when applied to a term or a formula, θ_1 is applied before θ_2, that is, $t(\theta_1 \circ \theta_2) = (t\theta_1)\theta_2$.

A substitution θ is *more general* than a substitution ϕ, denoted $\theta \succeq \phi$, if there is a substitution λ such that $\theta \circ \lambda = \phi$.

We refer to the refutation procedure underlying TEMPLOG as *TSLD-resolution* (for Temporal Linear resolution for Definite clauses' with a Selection function) by analogy with the SLD-resolution procedure for classical logic programming ([AvE82]). Every step of a TSLD-derivation consists in resolving *a candidate next-atom* from the current goal with the head of a program clause, to produce a new goal. **A** next-atom occurring in a goal is said to be a candidate next-atom if it is in the scope of at most one 0 in the canonical form of the goal. There is at least one candidate next-atom in any non-empty goal. The candidate next-atom that is selected for resolution at every step of a derivation is determined by a given *selection function* or *computation rule.* We refer to it as the *selected next-atom.*

Example: The canonical goal $\leftarrow \Diamond\Big(p(X,Y), \Diamond\bigcirc q(Y), \Diamond\bigcirc r(X,Z), \Diamond\big(s(Y,Z), t(Z)\big)\Big)$ has $p(X, Y)$ as only candidate next-atom. ■

The resolution rules used in TSLD-derivations are given in Table 1. For each of these rules, the candidate next-atom selected from the goal is $\bigcirc^i A$, and $\theta =$ *mgu(A, A')* is the most-general unifier of *A* and *A'*. The resolvent is also referred to as the *derived goal.*

Let *P* be a TEMPLOG program, G a goal, and *R* a computation rule. A *TSLD-derivation* for *P* ∪ {G} via *R* consists of

- a sequence of goals $G_0, G_1, \ldots$ where $G_0 =$
- a sequence of candidate next-atoms $N_0, N_1, \ldots$ such that N_i is the next-atom selected from G_i by R;
- a sequence of program clauses $C_1, C_2, \ldots$ such that each C_i is a clause in P that has been renamed so that none of the variables in C_i also occurs in G or in $C_1, \ldots, C_{i-1}$;
- a sequence of substitutions $\theta_1, \theta_2, \ldots$;

and G_{i+1} is the goal obtained by applying one of the TSLD-resolution rules to C_{i+1} and G_i with selected next-atom N_i and most general unifier θ_{i+1}. A *TSLD-refutation* for $P \cup \{G\}$ via R is a finite TSLD-derivation whose last goal is empty. The length of a TSLD-refutation is the number of resolution steps necessary to derive the empty goal. The *R-computed answer substitution* associated

[1] **A definite clause is a Horn clause with exactly one atom in the head.**

	Cond.	Goal	Clause	Resolvent (derived goal)
1		$\leftarrow B_1, \bigcirc^i A, B_2$	$0'' A' \leftarrow B'$	$\leftarrow (B_1, B', B_2)\theta$
2	$i \geq j$	$\leftarrow B_1, \bigcirc^i A, B_2$	$\Box 0JA' \leftarrow B'$	$\leftarrow (B_1, B', B_2)\theta$
3	$i \geq j$	$\leftarrow B_1, \bigcirc^i A, B_2$	$\Box(\bigcirc^j A' \leftarrow B)$	$\leftarrow (B_1, \bigcirc^{i-j} B', B_2)\theta$
4	$j \geq i$	$\leftarrow B_1, \Diamond(B_2, \bigcirc^i A, B_3), B4$	$\bigcirc^j A' \leftarrow B'$	$\leftarrow (B_1, \bigcirc^{j-i} B_2, B', \bigcirc^{j-i} B_3, B_4)\theta$
5	$j \geq i$	$\leftarrow B_1, \Diamond(B_2, \bigcirc^i A, B_3), B_4$	$\Box \bigcirc^j \leftarrow B'$	$\leftarrow (B_1, B', \Diamond(\bigcirc^{j-i} B_2, \bigcirc^{j-i} B_3), B_4)\theta$
6	$i \geq j$	$\leftarrow B_1, \Diamond(B_2, \bigcirc^i A, B_3), B_4$	$\Box Oj A' \leftarrow B'$	$\leftarrow (B_1, B', \Diamond(B_2, B_3), B_4)\theta$
7	$j \geq i$	$t\ B_1, 0(B_2, 0'A, B_3), B_4$	[illegible] $\leftarrow B')$	$\leftarrow (B_1, \Diamond(\bigcirc^{j-i} B_2, B', \bigcirc^{j-i} B_3), B_4)\theta$
8	$i \geq j$	$\leftarrow B_1, \Diamond(B_2, \bigcirc^i A, B_3), B_4$	$\Box(\bigcirc^j A' \leftarrow B')$	$\leftarrow (B_1, \Diamond(B_2, \bigcirc^{i-j} B', B_3), B_4)\theta$

Table **1:** TSLD-Resolution Rult ; for **TEMPLOG**

with an n-step refutation of $P \cup \{G\}$ via R is the substitution obtained by restricting the substitution $(\theta_1 \circ \cdots \circ \theta_n)$ to the variables of G.

Remark: We have augmented the original definition of **TEMPLOG** given in [AM87] to allow function symbols in terms. Also, the proof method underlying the execution of programs was given in [AM87] for a fixed computation rule that consists in always selecting the leftmost candidate next-atom first as in **PROLOG** ([CM84]). We allow an arbitrary computation rule. Moreover, in the original definition of the language, a clause with an empty head is referred to as a ***call***, and when a program is queried with a call of the form $\leftarrow B$, the sequence of goals $\leftarrow B, \leftarrow 0\ B, \leftarrow \bigcirc^2 B, \ldots$ is evaluated. We do not follow this convention. Instead, we refer to a clause with an empty head as a goal and when a program is queried with a goal, only that goal is evaluated.

4 Declarative Semantics for TEMPLOG

The declarative meaning of a logic program is the set of bodies that are logical consequences of the program, that is, the set of bodies that are true in every model of the program. We begin by giving a model-theoretic characterization of this denotation of programs. Then we develop an equivalent fixpoint characterization.

4.1 Model-Theoretic Semantics

In this section, we first introduce the notions of temporal Herbrand model and of temporal groundedness. We then formulate a satisfaction criterion for TEMPLOG clauses based on the notion of temporal groundedness. We prove that if a program has a temporal model then it has a temporal Herbrand model and that the intersection of a collection of temporal Herbrand models of a program is a temporal model of the program. Using these two results we show that the minimal Herbrand model, that is, the intersection of all the temporal Herbrand models of a program, satisfies exactly the bodies that are logical consequences of the program, and hence provides a characterization of the denotation of a program.

The *Herbrund universe* U_L of a language L is the set of variable-free terms constructed from the constant and the function symbols in L. (If there are no constant symbols in L, an arbitrary one is introduced.) This notion coincides with the notion of Herbrand universe in classical logic ([Man74]), which is quite natural since the constant and function symbols in TEMPLOG are rigid. Variable-free terms and formulas are said to be *ground*. Given a term or a formula E and a substitution θ, if the instance $E\theta$ of E is ground then $E\theta$ said to be a *ground instance* of E.

Definition (Temporal Herbrand Base) The *temporal Herbrand base* B_L of a language L is the set of ground next-atoms constructed from the predicate symbols of L and the ground terms of the Herbrand universe U_L.

Definition (Temporal Herbrand Interpretation) A *temporal Herbrand interpretation* for a language L is a temporal interpretation that satisfies the following conditions:

- it has the Herbrand universe U_L as domain;
- its interpretation maps every constant symbol a in L to the term "a" in U_L;
- its interpretation maps every k-ary function symbol f in L to the k-ary function over U_L that maps the terms "t_1", ..., "t_k" of U_L into the term "$f(t_1, \ldots, t_k)$" of U_L.

In fact, a temporal Herbrand interpretation for closed formulas of a language L corresponds exactly to a subset of the temporal Herbrand base B_L, namely the set of ground next-atoms that are true under the interpretation at the initial time. In the following, we will treat a temporal Herbrand interpretation as such a subset of the Herbrand base. So a ground next-atom N is satisfied by a temporal Herbrand interpretation *I*, denoted $\models_I$ N, iff N is in *I*.

Remark: It would be equivalent to consider the Herbrand base B_L to be, as in classical logic, the set of ground atoms of I;. Then, a temporal Herbrand interpretation *I* could be defined as an w-sequence of subsets of B_L, or equivalently, a function I : $\omega \rightarrow 2^B$ that associates with every natural number i the set of ground atoms that are true at time i.

Example: Let L be a language consisting of the constant symbol a and the unary predicate symbol *p*. We consider again the following simple program (whose language is L) consisting of the following clauses.

$$p(a) \leftarrow$$
$$\Box\left(\bigcirc p(s(s(X))) \leftarrow p(X)\right)$$

The Herbrand universe U_L and the Herbrand base B_L of the language of *P* are as follows.

$$U_L = (a, s(a), s^2(a), \ldots\}$$

$$B_L = \{p(a), \bigcirc p(a), \bigcirc^2 p(a), \ldots, p(s(a)), \bigcirc p(s(a)), \bigcirc^2 p(s(a)), \ldots, p(s^2(a)), \ldots\}$$

The set $M = \{p(a), \bigcirc p(s^2(a)), \bigcirc^2 p(s^4(a)), \bigcirc^3 p(s^6(a))\ldots\}$ is a model of *P* as it satisfies every ground instance of every clause in *P*. ■

The satisfaction relation of ground TEMPLOG clauses has a simple reformulation when one introduces the notions of temporally ground formula and of temporally ground instance.

Definition (Temporally Ground) A formula is said to be *temporally ground* (TG) if 0 is the only temporal operator that appears in it.

So atoms, next-atoms, program clauses of the form $0" A \leftarrow \bigcirc^{i_1} A_1, \ldots, \bigcirc^{i_m} A_{,,}$ and goal clauses oftheform $\leftarrow \bigcirc^{i_1} A_1, \ldots, \bigcirc^{i_m} A,$ are temporally ground.

Definition (Temporally Ground Instance of a Body) Let *B* be a TEMPLOG body. A *temporally ground instance* (TGI) of *B* is a TG body obtained from *B* by replacing every occurrence of 0 by a finite number of ○'s.

The canonical form of a body's TGI can be obtained by pushing the occurrences of 0 all the way inwards using the equivalence schema $\bigcirc(F \text{ A } G) \equiv 0\ F \text{ A } 0\ G$ from left to right. It is a conjunction of next-atoms of the form $\bigcirc^{i_1} A_1 \wedge \ldots \wedge \bigcirc^{i_m} A$, where $A_1, \ldots, A$, are the atoms that appear in the body. For example, the canonical form of a TGI of the goal $\leftarrow \Diamond\left(\bigcirc^2 p(X,Y), \Diamond q(Y)\right)$ is of the form $+ \bigcirc^{i+2} p(X, Y), \bigcirc^{i+j} q(Y)$ where i, j $\in$ w.

Proposition 4.1 *Let $\mathcal{I}$ be a temporal interpretation of a ground* TEMPLOG *body B. $\mathcal{I}$ satisfies B iff it satisfies some temporally ground instance of B.*

PROOF: The proof proceeds by induction on the structure of bodies using the definition of the satisfaction relation $\models_{\mathcal{I}}$ (Section 2). The only interesting case is when B is of the form $0\ B'$. It involves the property: $\models_{\mathcal{I}(i)} F$ iff $\models_{\mathcal{I}} \bigcirc^i F$. ■

Definition (Temporally Ground Instance of a Program Clause) Let C be a TEMPLOG program clause.

1. If C is a clause of the form $\bigcirc^i A \leftarrow B$ and B^* is a TGI of B, then $\bigcirc^i A + B^*$ is a temporally ground instance (TGI) of C.
2. If C is a clause of the form $\Box\, \bigcirc^i A + B$ and B^* is a TGI of B, then for every $k \in w$, $\bigcirc^{i+k} A + B^*$ is a TGI of C.
3. If C is a clause of the form $\Box$ $(0'A \leftarrow B)$ and B^* is a TGI of B, then for every $k \in w$, $\bigcirc^{i+k} A \leftarrow \bigcirc^k B^*$ is a TGI of C.

Definition (Strictly Ground) A clause is *strictly ground* (SG) if it is both ground (variable-free) and temporally ground.

Definition (Strictly Ground Instance) A *strictly ground instance* (SGI) of a clause C is a ground instance of a temporally ground instance of C, or equivalently, a temporally ground instance of a ground instance of C.

Proposition 4.2 *Let $\mathcal{I}$ be a temporal interpretation of a ground* TEMPLOG ***clause*** *C. $\mathcal{I}$ **satisfies** C **iff** it satisfies every TGI **of** C.*

PROOF: The result is proved separately for each type of clause (initial, permanent, and goal clause) using the definition of the satisfaction relation $\models_{\mathcal{I}}$ and Proposition 4.1. For permanent clauses, one also needs the property: $\models_{\mathcal{I}} \Box F$ iff for every i in w: $\models_{\mathcal{I}} \bigcirc^i F$. ■

From Proposition 4.2, it follows that a temporal Herbrand interpretation for a program P satisfies P if and only if it satisfies every strictly ground instance of every clause in P.

Proposition 4.3 *Let S be a set of* TEMPLOG *clauses. If S has a temporal model, then S has a temporal Herbrand model.*

PROOF: Let L be the language of the clauses in S, and let $\mathcal{I}$ be a temporal model of S. We associate with $\mathcal{I}$ a temporal Herbrand interpretation $I = \{N \in B_L : \models_{\mathcal{I}} N\}$, and we try to prove that I is a model of S. Assuming, contrary to the desired conclusion, that there is a ground instance C of a clause in S that I does not satisfy, one then shows that $\mathcal{I}$ does not satisfy C either, a contradiction.

$\not\models_I C \Rightarrow$ there is a TGI $N \leftarrow N_1, \ldots, N_m$ of C s.t. $\not\models_I N + N_1, \ldots, N_m$ (by Prop. 4.2)
$\Rightarrow$ there is a TGI $N \leftarrow N_1, \ldots, N_m$ of C s.t. $\{N_1, \ldots, N_m\} \subseteq I$ and $N \notin I$
$\Rightarrow$ there is a TGI $N \leftarrow N_1, \ldots, N_m$ of C s.t. $\models_{\mathcal{I}} N_1 \wedge \ldots \wedge N_m$ and $\not\models_{\mathcal{I}} N$ (by def. of I)
$\Rightarrow \not\models_{\mathcal{I}} C$ (by Prop. 4.2)

Hence I is a model of S. ■

Therefore, if a set of clauses has no temporal Herbrand model, it is unsatisfiable. This is simply the contrapositive of Proposition 4.3.

Theorem 4.4 (Model Intersection Property) *Let P be a* TEMPLOG *program. The intersection of a collection of temporal Herbrund models of P is a temporal Herbrand model of P.*

PROOF: Let M_k, with k ranging over some index set W, be a collection of temporal Herbrand models of P, and let $M = \bigcap_{k \in W} M_k$. If there was a ground instance C of a clause in P that M did not satisfy, then by Propositions 4.2 and 4.1, there would have to be some $M_\ell (\ell \in W)$ that did not satisfy C either, a contradiction. So M is a model of P. ■

Knowing that the intersection of the temporal Herbrand models of a program P is also a model for P, we can now establish that this least Herbrand model, denoted M_P, provides a characterization of the declarative semantics of P.

Theorem 4.5 *Let P be a* TEMPLOG *program and B a ground body: $P \models B$ iff $\models_{M_P} B$.*

PROOF:

$[\Rightarrow]$ $P \models B \Rightarrow \models_{M_P} B$ by definition of logical consequence and since M_P is a model of P.

$[\Leftarrow]$ $\models_{M_P} B \Rightarrow$ there is a TGI $B^* \equiv N_1$ A... A N_m of B s.t. $\models_{M_P} B^*$ (by Proposition 4.1)
$\Rightarrow$ there is a TGI $B^* \equiv N_1$ A ... A N_m of B s.t. $\{N_1, \ldots, N,\} \subseteq Mp$
$\Rightarrow$ there is a TGI $B^* \equiv N_1$ A... A N_m of B s.t.
for every temporal Herbrand model M of P: $\{N_1, \ldots, N,\} \subseteq M$ (def. of M_P)

$\Rightarrow$ for every temporal Herbrand model M of P,
there is a TGI $B^* \equiv N_1 \wedge \ldots \wedge N_m$ of B s.t. $\{N_1, \ldots, N_m\} \subseteq M$
$\Rightarrow$ for every temporal Herbrand model M of P, there is a TGI B^* of B s.t. $\models_M B^*$
$\Rightarrow$ for every temporal Herbrand model M of P: $\models_M B$ (by Proposition 4.1)
$\Rightarrow P \cup \{\neg B\}$ has no temporal Herbrand model
$\Rightarrow P \cup \{\neg B\}$ is unsatisfiable (by Proposition 4.3)
$\Rightarrow P \models B$. ■

The first of the following two corollaries specifies the contents of M_P as a subset of the Herbrand base. The second corollary expresses a relationship between correctness of answer substitutions and satisfaction by temporal Herbrand models.

Corollary 4.6 *$Mp = \{\bigcirc^i A \in B_P : P \models \bigcirc^i A\}$.*

This is simply a restriction of Theorem 4.5 to the case of bodies that are single ground next-atoms.

Corollary 4.7 *Let P be a* TEMPLOG *program and G a goal of the form $\leftarrow B$. Let θ be an answer substitution for $P \cup \{G\}$ such that $B\theta$ is ground. Then the following three statements are equivalent:*

1. *θ is correct, that is, $P \models B\theta$.*
2. *For every temporal Herbrand model M of P, $\models_M B\theta$.*
3. *$\models_{M_P} B\theta$.*

PROOF: The equivalence of (1) and (3) is simply Theorem 4.5; (1) implies (2) by definition of logical consequence; and (2) implies (3) because Mp is a temporal Herbrand model of P. ■

4.2 Fixpoint Semantics

Next, we show how to associate with a TEMPLOG program P a function Tp on the domain of the temporal Herbrand interpretations for P. Intuitively, this mapping corresponds to one step of strictly ground inference from P. We prove that this mapping is continuous and that its least fixpoint is exactly the least Herbrand model of the program, thereby providing a fixpoint characterization of the declarative meaning of TEMPLOG programs.

Let P be a TEMPLOG program with language L. The partially ordered set $(2^{B_L}, \subseteq)$ is a complete lattice. The bottom element is the empty set $\emptyset$ and the top element is the temporal Herbrand base B_L. The least upper bound *(lub)* operation corresponds to set union and the greatest lower bound

(glb) operation to set intersection (e.g. [Sch86], [Llo84]). If X is a subset of 2^{B_L}, we denote by $lub(X)$, the lub of the elements of X. The mapping *Tp* associated with a program *P* is defined as follows.

Definition (Mapping *Tp)* Let I be a temporal Herbrand interpretation of *P*. We have:

1. Let *0' A* t *B* be a ground instance of an initial clause in P. If there is a TGI $N_1 \wedge \ldots \wedge N_m$ of *B* such that $\{N_1, \ldots, N,\} \subseteq I$, then $\bigcirc^i A \in T_P(I)$.
2. Let $\Box \bigcirc^j A + B$ be a ground instance of an initial clause in *P*. If there is a TGI N_1 A. . .A N_m of *B* such that $\{N_1, \ldots, N,\} \subseteq I$, then for every $k \in w$, $\bigcirc^{j+k} A \in T_P(I)$.
3. Let $\Box$ *(OjA* $\leftarrow$ *B)* be a ground instance of a permanent clause in *P*. For every $k \in w$, we have: if there is a TGI N_1 A . . . A N_m of *B* such that $\{\bigcirc^k N_1, \ldots, \bigcirc^k N,\} \subseteq$ I, then $\bigcirc^{j+k} A \in T_P(I)$.

Using the notion of temporally ground instance of a clause, this definition simply states that if N $\leftarrow N_1, \ldots, N_m$ is a temporally ground instance of a ground instance of a clause in *P* and if each of $N_1, \ldots) N_m$ is in *I*, then N is in $T_P(I)$. A temporally ground instance of a ground instance of a clause is a strictly ground instance of a clause. So, we can give the following more concise formulation of the definition of T_P.

Definition (Mapping *Tp)* Let *I* be a temporal Herbrand interpretation of *P*. We have:

$$T_P(I) = \{N \in B_L : N \leftarrow N_1, \ldots, N_m \text{ is a SGI of a clause in } P \text{ and } \{N_1, \ldots, N_m\} \subseteq I)$$

Notice that this definition of *Tp* is similar to the definition of the mapping associated with classical logic programs, except that in classical logic one deals with atoms and with ground instances of clauses where in temporal logic we deal with next-atoms and with strictly ground instances of clauses, respectively. As a result of this resemblance, the next three properties of *Tp* (Propositions 4.8, 4.9, and Theorem 4.10) admit proofs that are very similar to the proofs of the analogous results for classical logic programming ([vEK76], [AvE82], [Llo84], [Apt87]).

We denote by $lfp(T_P)$ the least fixpoint of *Tp*, by *Tp* $\uparrow$ n the n-th iteration of T_P applied to the empty set, that is, $T_P^n(\emptyset)$, and by *Tp* $\uparrow$ w the least upper bound of $\{\emptyset, T_P(\emptyset), T_P(T_P(\emptyset)), \ldots\}$.

Proposition *4.8 Tp is continuous on* $(2^{B_L}, \subseteq)$, *and hence,* $lfp(T_P) = Tp \uparrow w$.

PROOF: Let X be a directed subset of 2^{B_L}. We have to show that $T_P(lub(X)) = lub(Tp(X))$. We will use the following property: *Y* is a *finite* subset of $lub(X)$ iff Y $\subseteq$ S for some S $\in$ X ([Sto77]). Let us establish that for any $N \in B_L$, *we* have $N \in Tp(\ lub(X))$ iff N $\in$ *lub(Tp(X))*.

$N \in Tp(\ lub(X))$ iff there is a SGI N $\leftarrow N_1, \ldots, N_m$ of a clause in *P* s.t. $\{ N_1, \ldots,$ N,$\} \subseteq lub(X)$
(by definition of *Tp)*

iff there is a SGI N $\leftarrow N_1, \ldots, N_m$ of a clause in *P*, $\exists S \in X$ s.t.
$\{N_1, \ldots, N_m\} \subseteq S$ (by the property mentioned above)
iff there is a SGI N + $N_1, \ldots, N_m$ of a clause in *P*, $\exists S \in X$ s.t. N $\in T_P(S)$
(by definition of *Tp)*
iff N $\in$ *lub(Tp(X))* (since *lub* corresponds to set union)

This establishes the continuity of *Tp*. We can thus infer that $lfp(T_P) = Tp \uparrow$ w by a version of the fixpoint theorem (see e.g. [dB80], [Llo84]). ■

The next proposition gives a criterion for a temporal Herbrand interpretation to be a model of a program *P* as a condition on the mapping *Tp*. It will be used in estalishing the correspondence between the least Herbrand model *Mp* and the least fixpoint of *Tp* (Theorem 4.10).

Proposition 4.9 *Let I be a temporal Herbrand interpretation for the* TEMPLOG *program P. Then* $\models_I P$ *iff* $T_P(I) \subseteq I$.

PROOF:

$\models_I P$ iff for every SGI $N \leftarrow N_1, \ldots, N_m$ of every clause in *P*: $\models_I N \leftarrow N_1, \ldots, N_m$
(by Proposition 4.2)
iff for every SGI N $\leftarrow N_1, \ldots, N_m$ of every clause in *P:* if $\{ N_1, \ldots,$ N,$\} \subseteq I$ then N $\in I$
iff $T_P(I) \subseteq I$. ■

Theorem 4.10 *Mp = Tp* $\uparrow$ *w.*

PROOF: The least Herbrand model *Mp* is the intersection of the temporal Herbrand models of *P*. So in the complete lattice $(2^{B_L}, \subseteq)$:

$M_P = glb\{I \in 2^{B_L} : \models_I P\}$
$= glb\{I \in 2^{B_L} : T_P(I) \subseteq I)$ (by Proposition 4.9)

In other words, *Mp* is the greatest lower bound of the pre-fixpoints of *Tp,* which is $lfp(T_P)$ by a fixpoint theorem (see e.g. [dB80], [Llo84]). And so *Mp = Tp* $\uparrow$ w by Proposition 4.8. ■

5 Soundness and Completeness of TSLD-resolution

In this section, we establish the soundness and the completeness of the TSLD-resolution proof method underlying TEM PLOG'S execution.

5.1 Soundness

We begin with a lemma stating the soundness of each resolution rule. Then, we extend the result to the TSLD-resolution method.

Lemma 5.1 (Soundness of the Resolution Rules) *Let* $\leftarrow B'$ *be the resolvent* ***of*** *the goal* $\leftarrow B$ *and the* TEM PLOG *program clause* C *with most general unifier* θ. *Then,* $C \models (B\theta \leftarrow B'\theta)$.

PROOF: The proof has to be carried out separately for each of the eight TSLD-resolution rules of Table 1. We give the details for one of them, namely rule (5). The other cases admit a similar kind of proof.

Let us consider the case where $\leftarrow B$ is a goal of the form $\leftarrow B_1, 0(B_2, \bigcirc^i A, B_3), B_4$, the clause C is of the form $\bullet \mathrm{I}\ \bigcirc^j A'' \leftarrow B''$ with $i \leq$ j, and $\theta = mgu(A, A'')$. Then, the resolvent is the goal $\leftarrow (B_1, B'', \Diamond(\bigcirc^{j-i} B_2, \bigcirc^{j-i} B_3), B_4)\theta$. The statement to be proved is:

$$(\Box \bigcirc^j A'' \leftarrow B'') \models (B_1 \wedge \Diamond(B_2 \wedge 0' \wedge A \wedge B_3) \wedge B_4)\theta \leftarrow (B_1 \wedge B'' \wedge \Diamond(\bigcirc^{j-i} B_2 \wedge \bigcirc^{j-i} B_3) \wedge B_4)\theta.$$

Let $\mathcal{I}$ be a temporal interpretation such that $\models_{\mathcal{I}} \Box \bigcirc^j A'' \leftarrow B''$, ***and*** let $j = i + k$ $(k \geq 0)$. Then,

$$\models_{\mathcal{I}} (B'' \wedge \Diamond(\bigcirc^{j-i} B_2 \wedge \bigcirc^{j-i} B_3))\theta \Rightarrow \models_{\mathcal{I}} (\Box \bigcirc^j A'' \wedge \Diamond(\bigcirc^{j-i} B_2 \wedge \bigcirc^{j-i} B_3))\theta$$

(since $\models_{\mathcal{I}} \bullet \mathrm{I}\ \bigcirc^j A'' \leftarrow B''$)

$$\Rightarrow \models_{\mathcal{I}} (\Box \bigcirc^k \bigcirc^i A \wedge \Diamond \bigcirc^k (B_2 \wedge B_3))\theta$$

(since j = i + k and $A8 = A''\theta$)

$$\Rightarrow \models_{\mathcal{I}} (\Diamond(B_2 \wedge \bigcirc^i A'' \wedge B_3))\theta.$$

The last step is justified using the definition of the satisfaction relation for a temporal interpretation. Indeed, it is straightforward to show that $\bullet \mathrm{I}\ \bigcirc^k F \wedge 0\ 0'' \mathrm{G} \rightarrow \Diamond \bigcirc^k (F \wedge \mathrm{G})$ and $0\ \bigcirc^k (F \wedge \mathrm{G}) \rightarrow O(F \wedge \mathrm{G})$ are valid schemata. The desired result follows. ■

Theorem 5.2 (Correctness of Computed Answer Substitution) *Let* P *be a* TEMPLOG *program and* G *the goal* $\leftarrow B$. ***If*** $P \cup \{G\}$ *has a refutation with computed answer substitution* θ, *then* θ *is correct, that is,* $P \models (\forall *)B\theta$.

PROOF: The proof is by induction on the length of the refutation of $P \cup \{G\}$ using Lemma 5.1. It involves no particular difficulty. ■

The following corollary is an immediate consequence of this theorem.

Corollary 5.3 (Soundness of TSLD-Resolution) *Let P be a* TEMPLOG *program and G a goal. If* $P \cup \{G\}$ *has a TSLD-refutation then* $P \cup \{G\}$ *is unsatisfiable.*

5.2 Completeness

In classical logic, the proof of the completeness of resolution is based on two main lemmas: a lemma stating the completeness of ground resolution and a ***lifting*** *lemma* to "lift" the ground completeness result to the first-order completeness result ([Rob65], [AvE82], [Llo84], [Apt87]). In the case of temporal logic, our strategy is somewhat similar. We first establish the correspondence between membership in the fixpoint of the mapping *Tp* and existence of a refutation, thereby obtaining a completeness result for strictly ground formulas (Lemma 5.5). Then we introduce a *temporal lifting lemma* (Lemma 5.6) that allows us to "lift" this completeness result for both ground and temporally ground formulas to a completeness result for ground formulas (Lemma 5.7). Finally, combining this ground completeness lemma with a lifting lemma (Lemma 5.8) as in classical logic, we obtain the desired completeness theorem (Theorem 5.9). It is via the temporal lifting lemma that the notion of temporal groundedness plays its crucial role. In fact, we could have withheld the introduction of temporal groundedness up to this point. Indeed, the proofs of the propositions and theorems in Sections 4 and 5.1 can be carried out without this notion, although they become more tedious. At the end of this section, we prove a more general version of the completeness theorem that takes the computed answer substitutions into account.

Let us begin by defining the notion of temporally ground refutation.

Definition (Temporally Ground Refutation) A *temporally ground refutation* (TG-refutation) for a program *P* and a TG goal G is a TSLD-refutation for G that only uses TGI of the clauses in *P*. (and hence only uses the first TSLD-resolution rule of Table 1). There is no occurrence of 0 in the goals of a TG-refutation and no occurrence of □ or 0 in the clauses used in a TG-refutation.

A refutation is said to be *unrestricted* if when unifying a subgoal with the head of a program clause, it uses a unifier that is not necessarily most general. Lemma 5.4 establishes the correspondence between existence of an unrestricted refutation and of a TSLD-refutation. It will be used in the proof of the completeness for strictly ground formulas. We omit its proof. It is similar to the proof of the analogous lemma for classical logic that can be found in [Llo84].

Lemma 5.4 *Let P be a* TEMPLOG *progmm and G a goal. If* $P \cup \{G\}$ *has an unrestricted refutation with* unifiers $\theta_1, \ldots, \theta_n$, *then* $P \cup \{G\}$ has a *TSLD-refutation of the same length with mgu's* $\theta'_1, \ldots, \theta'_n$. *Moreover,* $(\theta'_1 \circ \ldots \circ \theta'_n) \succeq (\theta_1 \circ \cdots \circ \theta_n)$.

Lemma 5.5 (Strictly Ground Completeness) *Let P be a* TEMPLOG *program and N a ground next-atom. If* $N \in Mp$ *then* $P \cup \{\leftarrow N\}$ *has a TG-refutation.*

PROOF: Let $N \in Mp$. Since $Mp = Tp \uparrow \omega$ (Theorem 4.10), there must exist an $n \in \omega$ such that $N \in Tp \uparrow n$. Let us prove by induction on n that N has an unrestricted TG-refutation.

$n = 1$: $N \in Tp \uparrow 1$ means that $N \leftarrow$ is a SGI of a clause in *P*. So, $P \cup \{\leftarrow N\}$ has an unrestricted TG-refutation.

n > 1: Let us assume the induction hypothesis: $\forall N' \in T_P \uparrow (n-1)$: $P \cup \{\leftarrow N'\}$ has an unrestricted TG-refutation.

Let $N \in T_P \uparrow n = T_P(T_P \uparrow (n-1))$. By definition of *Tp*, there is a SGI $(N' \leftarrow N_1, \ldots, N_m)\theta$ of a clause in *P* such that $N'\theta = N$ and $\{N_1\theta, \ldots, N_m\theta\} \subseteq Tp \uparrow (n-1)$. Therefore, by the induction hypothesis, each of $P \cup \{\leftarrow N_1\theta\}, \ldots, P \cup \{\leftarrow N_m\theta\}$ has an unrestricted TG-refutation. Since the next-atoms $N_1\theta, \ldots, N_m\theta$ are strictly ground, their unrestricted TG-refutations can be combined into an unrestricted TG-refutation of $P \cup \{\leftarrow N_1\theta, \ldots, N_m\theta\}$. So $P \cup \{\leftarrow N\}$ has an unrestricted TG-refutation.

We have shown that $P \cup \{\leftarrow N\}$ has an unrestricted TG-refutation. Therefore, by Lemma 5.4, $P \cup \{\leftarrow N\}$ has a TG-refutation. ■

The next lemma is intended to permit the lifting of this completeness result for strictly ground formulas to a completeness result for ground formulas.

Lemma 5.6 (Temporal Lifting Lemma) *Let P be a* TEMPLOG *program, G a goal. Let n > 0.* $P \cup \{G\}$ *has a TSLD-refutation of length n iff there is a temporally ground instance* G^* *of G that has a TG-refutation of length n.*

PROOF: Each direction of the equivalence is proved by induction on the length *n* of the refutation. We give the proof in detail for one direction. Namely, we show that if G has a TGI G* such that $P \cup \{G^*\}$ has a TG-refutation of length n, then $P \cup \{G\}$ has a refutation of length *n*. The other direction is proved in an analogous way. Both the base case ($n = 1$) and the inductive step ($n > 1$) split in two cases: the case where the next-atom selected for the first resolution step of the TG-refutation for $P \cup \{G^*\}$ is not in the scope of any 0 in G and the case where it is in the scope of

one 0 in G; then for each of these cases, we have to study the subcases where the program clause used in the first resolution step of the TG-refutation for $P \cup \{G^*\}$ is already temporally ground in the program or is the TGI of a non-TG program clause. In studying all these cases, we exhaust the eight resolution rules (Table 1) that can be used in the first step of the refutation for $P \cup \{G\}$.

Let us assume that G* is a TGI of G such that $P \cup \{G^*\}$ has a TG-refutation of length *n*. We try to show that $P \cup \{G\}$ has a refutation of length *n*. We denote by

- C_1^* the TGI of a program clause that is used in the first resolution step of the TG-refutation for $P \cup \{G^*\}$,
- G_1^* the (temporally ground) goal obtained by resolving G* and C_1^*,
- C_1 the program clause of which C_1^* is a TGI (C_1 is used in the first resolution step of the refutation for $P \cup$ [illegible]
- G_1 the goal obtained by resolving G and Cr.

n = 1: G* must be of the form $\leftarrow O'A$.

The clause C_1^* is a fact of the form $O^i A' \leftarrow$, and $\theta = mgu(A, A')$.

1. *G is temporally ground, so* $G = G^*$.
   (a) C_1^* *is a clause of P, so* $C_1 = C_1^*$: the desired result holds trivially.
   (b) C_1^* *is a TGI of a non-TG clause* C_1 *of P:* C_1 is of the form $\bullet I\ O^j A' \leftarrow$ with $i \geq j$. So G and C_1 can be resolved according to one of the rules (2) and (3), and hence $P \cup \{G\}$ has a one-step refutation.
2. *G is not temporally ground:* it is of the form $\leftarrow \Diamond O^k A$ with $i \geq k$.
   (a) C_1^* *is a clause of P, so* $C_1 = C_1^*$: G and C_1 can be resolved according to rule (4), and so $P \cup \{G\}$ has a one-step refutation.
   (b) C_1^* *is a TGI of a non-TG clause* C_1 *of P:* C_1 is of the form $\bullet I\ O^j A' \leftarrow$ with $i \geq j$. Then G and C_1 can be resolved according to one of the rules (5), (6), (7) and (8).

n > 1: We show that whatever the form of G and of its TGI G*, and whatever TGI C_1^* of a program clause C_1 is used in the first resolution step of the TG-refutation for $P \cup \{G^*\}$ (to produce G_1^*), there is a corresponding resolution step for G and C_1 that produces the new goal G_1 such that G_1^* is a TGI of G_1 (see details below). We know that $P \cup \{G^*\}$ has a refutation of length *n*, therefore $P \cup \{G_1^*\}$ has a refutation of length $n - 1$. So by the induction hypothesis, $P \cup \{G_1\}$ has a refutation of length $n - 1$; and since G_1 is the resolvent of G and C_1, we can

infer that $P \cup \{G\}$ has a refutation of length n. Let us now get into the details of the case analysis.

1. *The next-atom selected in G^* for the first resolution step of its TG-refutation corresponds to a next-atom in G that is not in the scope of any 0.*

   So G* is of the form $\leftarrow B_1^*, \bigcirc^i A, B_2^*$ where $\bigcirc^i A$ is the next-atom selected for the first resolution step; G is $\leftarrow B_1, \bigcirc^i A, B_2$; and B_1^* and B_2^* are TGI's of B_1 and B_2, respectively.

   The clause C_1^* is of the form $0'A' \leftarrow B'$ where B' is temporally ground, and $\theta = mgu(A, A')$. Therefore, G_1^* is of the form $\leftarrow (B_1^*, B', B_2^*)\theta$.

   (a) *C_1^* is a clause of P, so $C_1 = C_1^*$:* G and C_1 can be resolved according to rule (1) to yield $G_1 : \leftarrow (B_1, B', B_2)\theta$, and G_1^* is a TGI of G_1.

   (b) *C_1^* is a TGI of a non- TG clause C_1 of P:*

   - C_1 is of the form $\Box \bigcirc^j A' \leftarrow B$ with $i \geq j$ and B' is a TGI of B. Then G and C_1 can be resolved according to rule (2) to yield $G_1 : \leftarrow (B_1, B, B_2)\theta$, and G_1^* is a TGI of G_1.
   - C_1 is of the form $\square$ (Oj $A' \leftarrow B$) where $i \geq j$, $B' \equiv \bigcirc^{i-j} B''$, and B'' is a TGI of B. Then G and C_1 can be resolved according to rule (3) to yield the goal $G_1 : \leftarrow (B_1, \bigcirc^{i-j} B, B_2)\theta$. 0 ne can see that G_1^* is a TGI of G_1.

2. *The next-atom selected in G^* for the first resolution step of its TG-refutation corresponds to a next-atom in G that is in the scope of one 0.*

   So G* is of the form $\leftarrow B_1^*, \bigcirc^\ell B_2^*, \bigcirc^{\ell+i} A, \bigcirc^\ell B_3^*, B_4^*$ where $\bigcirc^{\ell+i} A$ is the next-atom selected for the first resolution step; G is of the form $\leftarrow B_1, \Diamond(B_2, \bigcirc^i A, B_3), B_4$; and B_1^*, B_2^*, B_3^*, and B_4^* are the TGI's of B_1, B_2, B_3, and B_4, respectively.

   The clause C_1^* is of the form $\bigcirc^j A' \leftarrow B'$ where B' is temporally ground, $j = \ell + i$, and $\theta = mgu(A, A')$. Therefore, G_1^* is $\leftarrow (B_1^*, \bigcirc^\ell B_2^*, B', \bigcirc^\ell B_3^*, B_4^*)\theta$.

   (a) *C_1^* is a clause of P, so $C_1 = C_1^*$:* G and C_1 can be resolved according to rule *(4)* to yield $G_1 : \leftarrow (B_1, \bigcirc^\ell B_2, B', \bigcirc^\ell B_3, B_4)\theta$, and G_1^* is a TGI of G_1.

   (b) *C_1^* is a TGI of a non- TG clause C_1 of P:*

   - C_1 is of the form $\Box \bigcirc^k A' \leftarrow B$ with $j \geq k$ and B' is a TGI of B:

     – $i \leq k$: G and C_1 can be resolved according to rule (5) to yield the new goal $G_1 : \leftarrow (B_1, B, \Diamond(\bigcirc^{k-i} B_2, \bigcirc^{k-i} B_3), B_4)\theta$. Since $j \geq k$, we have $j - i \geq k - i$, that is, $\ell \geq k - i$; so G_1^* is a TGI of G_1.

- i $\geq k$: G and C_1 can be resolved according to rule (6) to yield the new goal G_1: $\leftarrow (B_1, B, \Diamond(B_2, B_3), B_4)\,\theta$, and G_1^* is a TGI of G_1.

- C_1 is of the form ❑ **(Ok A' ← B)** with j = $k + p$, $B' \equiv \bigcirc^p B''$ and B'' is a TGI of B.
  - i $\leq k$: G and C_1 can be resolved according to rule (7) to yield the new goal G_1 : $t\,(\,B_1, \Diamond(\bigcirc^{k-i} B_2, B, \bigcirc^{k-i} B_3), B_4)\,\theta$. Since j = ℓ + i = $k + p$, *we* have $k - i = \ell - p$, and so G_1^* is a TGI of G_1.
  - i $\geq k$: G and C_1 can be resolved according to rule (8) to yield the new goal G_1 : $\leftarrow (B_1, \Diamond(B_2, \bigcirc^{i-k} B, B_3), B_4)\,\theta$. Since j = ℓ + i = $k + p$, *we* have $i - k = p - \ell$, and so G_1^* is a TGI of G_1. ■

Lemma 5.7 (Ground Completeness) *Let P be a* TEMPLOG *program. For every ground body B, if* $\models_{M_P} B$ *then* $P \cup \{\mathrm{t}\ B\}$ *has a TSLD-refutation.*

PROOF:

$\models_{M_P} B$ ⇒ there is a TGI N_1 A.. . $\wedge N_m$ of B s.t. $\{N_1, \ldots, N_m\} \subseteq Mp$ (by Proposition 4.1)

⇒ there is a TGI N_1 A . . . $\wedge N_m$ of B s.t. Vi = 1, . . . , *m:* $P \cup \{\leftarrow N_i\}$ has a TG-refut. (by Lemma 5.5)

⇒ there is a TGI N_1 A . . . A N_m of B s.t. $P \cup \{\leftarrow N_1, \ldots, N_m\}$ has a TG-refutation (because the N_i's are ground and their refutations are temporally ground)

⇒ $P \cup \{\leftarrow B\}$ has a TSLD-refutation (by Lemma 5.6). ■

The following lifting lemma to be used together with the ground completeness lemma in the proof of the completeness theorem is similar to the lifting lemma for classical logic programming. We introduce it here without proof and we refer the reader to the proof given in the context of classical logic programming in [Llo84].

Lemma 5.8 (Lifting Lemma) *Let P be a* TEMPLOG *program, G a goal and* θ *a substitution. If there is a TSLD-refutation of* $P \cup \{G\theta\}$ *with mgu's* $\theta_1, \ldots, \theta_n$, *then there is a TSLD-refutation of* $P \cup \{G\}$ *of the same length with mgu's* $\theta'_1, \ldots, \theta'_n$. *Moreover,* $(\theta'_1 \circ \cdots \circ \theta'_n) \succeq (\theta \circ \theta_1 \circ \ldots \circ \theta_n)$.

The next theorem states the completeness of TSLD-resolution. It is the converse of Corollary 5.3.

Theorem 5.9 (Completeness of TSLD-Resolution) *Let P be a program, and G a goal. If* $P \cup \{G\}$ *is unsatisfiable, then there is a TSLD-refutation of* $P \cup \{G\}$.

PROOF: Let G be the goal $\leftarrow B$ such that $P \cup \{G\}$ is unsatisfiable. We have:

$P \cup \{\leftarrow B\}$ is unsat. $\Rightarrow$ for every temporal model $\mathcal{I}$ of P: $\not\models_{\mathcal{I}} \neg B$

$\Rightarrow \not\models_{M_P} \neg B$

$\Rightarrow$ there is a ground instance $B\theta$ of B s.t. $\not\models_{M_P} \neg B\theta$

$\Rightarrow$ there is a ground instance $B\theta$ of B s.t. $\models_{M_P} B\theta$

$\Rightarrow$ there is a ground instance $B\theta$ of B s.t. $P \cup \{\leftarrow B\theta\}$ has a TSLD-refut. (by Lemma 5.7)

$\Rightarrow P \cup \{\leftarrow B\}$ has a TSLD-refutation (by Lemma 5.8). ■

Next, we generalize this result to take the computed answer substitutions into account. We cannot prove the exact converse of Theorem 5.2, that is, we cannot show that any correct answer substitution can be computed by a refutation. Instead, we prove Theorem 5.11 which states that for any correct answer substitution, one can compute by a refutation an answer substitution that is more general than the correct answer substitution. Notice that the proofs of Lemma 5.10 and Theorem 5.11 do not use Theorem 5.9. So, one can also derive Theorem 5.9 as a corollary of Theorem 5.11.

Lemma 5.10 *Let P be a program and B a body. If $P \models (\forall *)B$, then there is a TSLD-refutation of $P \cup \{\leftarrow B\}$ with the empty substitution as computed answer substitution.*

PROOF: Let $X_1, \ldots, X_n$ be the variables occuring in B. Suppose that we augment the language of P with the new constants $a_1, \ldots, a_n$ and let $\theta = \{X_1 \leftarrow a_1, \ldots, X_n \leftarrow a_n\}$. Then $P \models B\theta$ and $B\theta$ is ground, so by Lemma 5.7, $P \cup \{\leftarrow B\theta\}$ has a TSLD-refutation (with empty computed answer substitution). Since the a_i's do not appear in P or in B, we can textually replace a_i by X_i for i = 1, ..., n, in the refutation of $P \cup \{\leftarrow B\theta\}$ thereby producing a refutation of $P \cup \{\leftarrow B\}$ with the empty substitution as answer substitution. ■

Theorem 5.11 (Computability of Correct Answer Substitution) *Let P be a program, G a goal, and θ a correct answer substitution for $P \cup \{G\}$. There is a computation rule R and an R-Eomputed answer substitution σ for $P \cup \{G\}$ such that $\sigma \succeq 8$.*

PROOF: Let G be the goal $\leftarrow B$. Since θ is a correct answer substitution for $P \cup \{\leftarrow B\}$, we have $P \models (\forall *)B\theta$. So by Lemma 5.10, $P \cup \{\leftarrow B\theta\}$ has a TSLD-refutation with the empty answer substitution. Let $\theta_1, \ldots, \theta_n$ be the sequence of mgu's of this refutation. By Lemma 5.8, $P \cup \{\leftarrow B\}$ has a refutation with mgu's $\theta'_1, \ldots, \theta'_n$, and $(\theta'_1 \circ \cdots \circ \theta'_n) \succeq (\theta \circ \theta_1 \circ \cdots \circ \theta_n)$. Let σ be the substitution $(\theta'_1 \circ \cdots \circ \theta'_n)$ restricted to the variables in B. The restriction of $(\theta \circ \theta_1 \circ \ldots \circ 9,)$ to the variables in B is simply θ. Therefore, $\sigma \succeq \theta$. ■

As in classical logic programming, the completeness results can further be strengthened to take the computation rule into account ([AvE82], [Hil74], [Llo84], [Apt87]). One can show that if the conjunction of a program and a goal is unsatisfiable, then it has a refutation via any computation rule. For this, one can introduce the notion of *n-refutability:* a program P and goal G are n-refutable if for any computation rule R, $P \cup \{G\}$ has a refutation via R of length at most n. The proof involves strengthening the lemmas used in the completeness proof to deal with n-refutability. It is also possible to show that for any correct answer substitution θ, one can compute an answer substitution more general than θ whatever the computation rule.

6 A fragment of TEMPLOG: TL1

In this section, we examine a fragment of **TEMPLOG** that we call TL1. In TL1, the body of a clause cannot contain any occurrence of 0 and initial clauses cannot have •I in their head. There are several reasons that make TL1 worth considering. First of all, it is one of the smallest extensions of Horn logic programming with temporal operators; it was introduced in [AM87] as a first step towards temporal logic programming. Moreover, it is one of the few subsets of **TEMPLOG** that is closed under the applicable TSLD-resolution rules; on the contrary, any proper subset of TEMPLOG that allows the use of 0 in the body of clauses is not closed under the TSLD-resolution rules. Finally, TL1 has the same expressiveness as the "pure" fragment of the THLP[2] language introduced by Wadge in [Wad88] and also referred to as **CHRONOLOG** in [OW88a]. However, the only interpretation method suggested for THLP consists in reducing the programs to classical Horn programs with explicit time parameters and interpreting them with classical logic programming methods. One problem with this approach is that the time parameter is treated as any other parameter by the logic programming interpreter, which can lead to unexpected behaviors.

The syntax of TL1 can be summarized as follows.

- Body: $B ::= \varepsilon \mid A \mid B_1, B_2 \mid OB$ where ε denotes the empty body.
- Initial clause: $IC ::= N \leftarrow B$
- Permanent clause: $PC ::= \Box (N \leftarrow B)$
- Program clause: C : := $IC \mid PC$
- Goal clause: G ::= $\leftarrow B$

[2] THLP stands for Temporal Horn Logic Programming.

A canonical TL1 body is simply a conjunction of next-atoms. The proof procedure for TL1 is based on the two TSLD-resolution rules (1) and (3) of Table 1.

Although not restrictive for TEMPLOG, the assumption that all the predicate symbols be flexible becomes more restrictive in the case of TL1. Indeed, as noted in [AM87], allowing to distinguish rigid predicate symbols from flexible predicate symbols requires introducing 0 in the language, but the use of 0 is not allowed in TL1. Although not efficient operationally and hardly realistic in practice, one can express in TL1 the time-independence of an ℓ-ary predicate p by adding the following clauses for p:

$$\Box\Big(\bigcirc p(X_1,\ldots,X_\ell) \leftarrow p(X_1,\ldots X_\ell)\Big)$$
$$\Box\Big(p(X_1,\ldots,X_\ell) \leftarrow \bigcirc p(X_1,\ldots,X_\ell)\Big)$$

It has also been argued ([0 W88a]) that the presence of 0 in the body of clauses can be simulated in a language like TL1 that does not have 0. Indeed, in theory, an attempt to refute the goal $\leftarrow 00'p(\bar{t})$ could be replaced by an attempt to refute the goal $\leftarrow q(\bar{t})$, where q is a new predicate symbol defined by the following program clauses:

$$\Box\Big(q(\bar{t}) \leftarrow \bigcirc^i p(\bar{t})\Big)$$
$$\Box\Big(q(\bar{t}) \leftarrow \bigcirc q(\bar{t})\Big)$$

This obviously affects the readability of the programs. Moreover, one must be aware of the practical inefficiency of such an approach. It increases the number of resolution steps of the refutations and the number of clauses in the program thereby augmenting the number of applicable clauses at every step of the computations. With implementations based on the backtracking mechanism, this can considerably slow down the computation. And of course, these problems get worse when simulating in this way arbitrary nestings of 0.

As for TEMPLOG, the declarative semantics of TL1 can be given in model-theoretic and in fixpoint terms. One can also establish the completeness of the TSLD-resolution procedure for TL1. We omit this development here as it is essentially superseded by the semantic development for TEMPLOG. However, it is interesting to note that it can be completely carried out – even for the completeness theorem – without introducing the notion of temporal groundedness.

7 Conclusion and Related work

We have developed the declarative (logical) semantics of TEMPLOG programs and expressed it in two equivalent ways: as a minimum temporal Herbrand model and as the least fixpoint of a mapping.

We proved a correspondence between the least fixpoint semantics and the existence of refutations, hence proving a completeness theorem for strictly ground formulas. From this theorem and lifting lemmas, we established the completeness of TSLD-resolution.

The work closest to ours is that of [OW88a] and [OW88b] which was developed independently. There, Orgun and Wadge study the declarative semantics of "intensional" (modal) extensions of Horn clause programs. One such extension that they consider is the THLP language we discussed in the previous section. They give declarative semantics similar to ours, but as they do not consider proof systems in conjunction with their language, they have no completeness results. Also, as far as temporal programming, their results are only given for a language equivalent to our TL1. In the conclusion of [OW88a] and [OW88b], it is mentioned that one of their results (the minimal model semantics) also holds for full **TEMPLOG**. However, in drawing this conclusion, they do not consider the fact that in TEMPLOG, clause bodies can contain arbitrary nestings of conjunctions and O-operators.

Gabbay has proposed an extension of classical logic programming distinct from **TEMPLOG** ([Gab87]). His **TEMPORAL PROLOG** is based on a different subset of temporal logic: $\Box$ can only be applied to entire clauses and the only operators allowed in the body and in the head of clauses are .O and the corresponding operator for the past. A proof method is given for this language, but it is unclear how it could be used as the basis of an execution mechanism and of operational semantics for the language. The only semantics defined for this language is its logical semantics.

For temporal languages like Moszkowski's **TEMPURA** ([Mos86]) and **TOKIO** ([FKTM86]), that view executing a program as constructing a model for the program, the semantic issues are completely different. In fact, in the case of **TEMPURA** that imperatively executes a temporal logic formula, the states of the computation are exactly the states of the model of the formula, and the operational semantics of a program corresponds to its logical semantics. **TOKIO** extends **PROLOG** with temporal constructs that are interpreted as control features. To give its formal semantics one would need to combine a semantics of temporal logic with a semantics of **PROLOG** that explicitly represents the execution mechanism. Such a semantics could, for instance, be based on that of [JM84], [DM88] or [Bau88].

Acknowledgements

I would like to thank Martin Abadi, Rajeev Alur, Tom Henzinger, Zohar Manna, Amir Pnueli and Pierre Wolper for critical reading of drafts of this paper and/or for related discussions.

References

[Aba87] Martin Abadi. *Temporal-Logic Theorem Proving.* PhD thesis, Computer Science Department, Stanford University, Stanford, CA, March 1987.

[AM87] Martín Abadi and Zohar Manna. Temporal logic programming. In *International Symposium on Logic Programming*, pages 4-16, IEEE, San Francisco, CA, September 1987.

[Apt87] Krzysztof R. Apt. *Introduction to Logic Programming.* Technical Report TR-87-35, Department of Computer Science, The University of Texas at Austin, Austin, Texas, September 1987.

[AvE82] Krzysztof R. Apt and M.H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM,* 29(3):841–862, July 1982.

[Bau88] Marianne Baudinet. Proving termination properties of PROLOG programs: a semantic approach. In *Symposium on Logic in Computer Science,* Edinburgh, Scotland, July 1988.

[Cla79] K.L. Clark. *Predicate Logic as* a Computational Formalism. Research Monograph 79/59 TOC, Department of Computing and Control, Imperial College, London, December 1979.

[CM84] W.F. Clocksin and C.S. Mellish. *Programming in Prolog.* Springer-Verlag, Berlin, second edition, 1984.

[dB80] Jaco de Bakker. *Mathematical Theory of Program Correctness.* Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

[DM88] Saumya K. Debray and Prateek Mishra. Denotational and operational semantics for Prolog. *Journal of Logic Programming,* 5(1):61–91, March 1988.

[Ede85] Elmar Eder. Properties of substitutions and unifications. *Journal of Symbolic Computation,* 1(1):31–46, March 1985.

[FKTM86] M. Fujita, S. Kono, H. Tanaka, and T. Moto-oka. Tokio: logic programming language based on temporal logic and its compilation to Prolog. In *Third International Conference on Logic Programming*, pages 695-709, LNCS 225, Springer-Verlag, London, July 1986.

[Gab86] Dov Gabbay. *Modal and Temporal Logic* Programming. Technical Report 86/15, Department of Computing, Imperial College of Science and Technology, London, June 1986.

[Gab87] Dov Gabbay. Modal and temporal logic programming. In Antony Galton, editor, *Temporal Logics and Their Applications,* chapter 6, pages 197-237, Academic Press, London, 1987.

[Hil74] R. Hill. *LUSH-Resolution and its Completeness.* DCL Memo 78, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland, 1974.

[JM84] Neil D. Jones and Alan Mycroft. Stepwise development of operational and denotational semantics for Prolog. In *International Symposium on Logic Programming,* pages 281-288, IEEE, Atlantic City, New Jersey, February 1984.

[Llo84] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, 1984.

[LMM86] Jean-Louis Lassez, Michael J. Maher, and Kimbal G. Marriot. *Unification Revisited.* Technical Report RC 12394 (#55630), IBM - T.J. Watson Research Center, Yorktown Heights, NY, December 1986.

[Man74] Zohar Manna. *Mathematical Theory of Computation. Computer Science Series,* McGraw-Hill, 1974.

[Mos84] Ben Moszkowski. *Executing Temporal Logic Programs.* Technical Report No. 55, Computer Laboratory, University of Cambridge, Cambridge, England, 1984.

[Mos86] Ben Moszkowski. *Executing Temporal Logic Programs.* Cambridge University Press, Cambridge, England, 1986.

[MW81] Zohar Manna and Richard Waldinger. Deductive synthesis of the unification algorithm. *Science of Computer Programming,* 1(1):5–48, 1981.

[OW88a] Mehmet A. Orgun and William W. Wadge. Chronolog: a temporal logic programming language and its formal semantics. Unpublished Manuscript, 1988.

[OW88b] Mehmet A. Orgun and William W. Wadge. A theoretical basis for intensional logic programming. In *Proceedings of the 1988 Symposium on Lucid and Intensional Programming,* pages 33-49, Sidney, B.C., Canada, April 1988.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[Sch86] David A. Schmidt. *Denotational Semantics - A methodology for Language Development.* Allyn and Bacon, 1986.

[Sto77] Joseph E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* Volume 1 of *The MIT Press Series in Computer Science*, MIT Press, third printing 1985 edition, 1977.

[vEK76] M.H. van Emden and Robert Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, October 1976.

[Wad85] William W. Wadge. Tense logic programming: a sane alternative. Unpublished Manuscript, 1985.

[Wad88] William W. Wadge. Tense logic programming: a respectable alternative. In *Proceedings of the 1988 Symposium on Lucid and Intensional Programming*, pages 26-32, Sidney, B.C., Canada, April 1988.