TOWARDS A UNIFIED THEORY OF INTENSIONAL LOGIC PROGRAMMING

MEHMET A. ORGUN AND WILLIAM W. WADGE

▷ Intensional Logic Programming is a new form of logic programming based on intensional logic and possible worlds semantics. Intensional logic allows us to use logic programming to specify nonterminating computations and to capture the dynamic aspects of certain problems in a natural and problem-oriented style. The meanings of formulas of an intensional first-order language are given according to intensional interpretations and to elements of a set of possible worlds. Neighborhood semantics is employed as an abstract formulation of the denotations of intensional operators. Then we investigate general properties of intensional operators such as universality, monotonicity, finitariness and conjunctivity. These properties are used as constraints on intensional logic programming systems. The model-theoretic and fixpoint semantics of intensional logic programs are developed in terms of least (minimum) intensional Herbrand models. We show in particular that our results apply to a number of intensional logic programming languages such as Chronolog proposed by Wadge and Templog by Abadi and Manna. We consider some elementary extensions to the theory and show that intensional logic program clauses can be used to define new intensional operators. Intensional logic programs with intensional operator definitions are regarded as metatheories. ◁

1. INTRODUCTION

Intensional logic programming (ILP) is a new form of logic programming based on intensional logic and possible world semantics. Intensional logic [20] allows us to describe context-dependent properties of certain problems in a natural and prob-

Address correspondence to M. A. Orgun and W. W. Wadge, Department of Computer Science, University of Victoria, P.O. Box 3055, Victoria, B.C., CANADA V8W 3P6. E-mail: {morgun, wwadge}@csr.uvic.ca.

Received September 1989; accepted October, 1990.

THE JOURNAL OF LOGIC PROGRAMMING

655 Avenue of the Americas, New York, NY 10010 0743-1066/92/$5.00

lem-oriented way. In intensional logic, the values of formulas depend on an implicit context parameter. The values from different contexts can be combined through the use of intensional operators that serve as context-changing operations.

Temporal and modal logics have been successfully used for specifying and verifying concurrent programs. Recently, several researchers have proposed extending logic programming with temporal logic, modal logic, and other forms of intensional logic in order to be able to model the notion of dynamic change. There are a number of modal and temporal logic programming languages: Tempura [21] and Tokio [2] are based on interval logic; Chronolog [37], Templog [1] and Temporal Prolog [13] are based on temporal logic; the Molog system [10] is based on user-elected modal logics; InTense [19] is a multidimensional language with temporal and spatial dimensions. This paper in particular discusses the temporal language Chronolog [37] and two other ILP languages, previously introduced in [23]. There are also other nonclassical extensions of logic programming including multiple-valued logic programming schemes of Blair et al. [7] and of Fitting [12].

However, there are few attempts to develop rigorous model-theoretic semantics for these nonclassical languages. For instance, Blair et al. [7] develop a logic programming semantics scheme for multiple-valued logic programming. Fitting [12] employs topological bilattices to treat the semantics of multiple-valued logic programming. However, these two approaches deal with nonclassical semantics for logic programming. Baudinet [5, 6] shows the completeness of Templog, and provides the declarative semantics of Templog programs. Orgun and Wadge [22, 23] develop the model-theoretic semantics of Chronolog and describe a general framework to deal with several ILP systems. Balbiani et al. [3] provide a tree-like semantics for a class of Molog programs. For other languages, some kind of extended operational semantics is usually provided. In this paper, we will build on the work of Orgun and Wadge [23] and provide a language-independent theory which can be applied to a variety of intensional logic programming languages. We will try to answer the question of how an ILP language can be enriched in terms of already available tools in the language.

In the following, we will first outline the semantics of intensional logic in terms of intensional interpretations and the satisfaction relation $\models$. Then we will introduce several intensional logic programming languages. We will discuss semantic properties of intensional operators, such as monotonicity, universality, conjunctivity and finitariness, based on an abstract formulation of intensional operators and the neighborhood semantics of Scott [28] and Montague [20]. Then we will develop a language-independent, model-theoretic semantics of intensional logic programs based on intensional Herbrand models in the style of van Emden-Kowalski [34]. We will show that this semantics can be applied to diverse ILP languages including Chronolog [37], Templog [1] and Molog [10]. Later we will show that intensional program clauses can be used to define new intensional operators. However, when recursive definitions are allowed, an infinitary logic, a version of $L_{\omega_1\omega}$ [15] must be employed.

2. INTENSIONAL LOGIC

Intensional logic [20] is the study of context-dependent properties. In intensional logic, the meaning of expressions depends on an implicit context, abstracted away

from the object language. Temporal logic [25] can be regarded as an instance of intensional logic where the collection of contexts models a collection of moments in time. Similarly, we regard modal logics [14] as instances of intensional logic. The collection of contexts is also called the *universe* or the set of *possible worlds*, and denoted by $\mathscr{U}$. An intensional logic is equipped with intensional operators through which elements from different contexts can be combined.

Throughout, we will only consider *unary* intensional operators. However, all the following results can be extended to cover n-ary intensional operators in a straightforward manner. The underlying language is obtained from a first-order language by extending it with formation rules for intensional operators.

Let IL denote the underlying intensional language of an intensional logic. From here on, we assume that $\mathscr{U}$ is countable. An intensional interpretation of IL basically assigns meanings to all elements of IL at all possible worlds in $\mathscr{U}$. An intensional interpretation can also be viewed as a collection of first-order interpretations (Tarskian structures), one for each possible world in $\mathscr{U}$. Here the denotations of variables and function symbols are *extensional* (a.k.a. *rigid*), that is, independent of the elements of $\mathscr{U}$. This is not generally so in intensional logic; but quite satisfactory for the theory of intensional logic programs. Let $P(A)$ denote the set of all subsets of the set A and $[A \to B]$ the set of functions from A to B. Then the formal definition of an intensional interpretation can be given as follows.

Definition 2.1. An intensional interpretation I of an intensional language IL comprises a nonempty 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 $[\mathbf{D}^n \to \mathbf{D}]$; and for each n-ary predicate symbol, an element of $[\mathscr{U} \to P(\mathbf{D}^n)]$.

The fact that a formula A is true at world w in some intensional interpretation I will be denoted as $\models_{I,w} A$. All formulas of IL are *intensional*, that is their meanings may vary depending on the elements of $\mathscr{U}$. The definition of the satisfaction relation $\models$ in terms of intensional interpretations is given in part as follows. Let $I(E)$ denote the value in $\mathbf{D}$ that I gives an IL term E.

Definition 2.2. The semantics of elements of IL are given inductively by the following, where I is an intensional intepretation of IL, $w \in \mathscr{U}$, and A and B are formulas of IL:

(a) If $f(e_0, \ldots, e_{n-1})$ is a term, then $I(f(e_0, \ldots, e_{n-1})) = I(f)(I(e_0), \ldots, I(e_{n-1})) \in \mathbf{D}$. If v is a variable, then $I(v) \in \mathbf{D}$.
(b) For any n-ary predicate p and terms $e_0, \ldots, e_{n-1}$, $\models_{I,w} p(e_0, \ldots, e_{n-1})$ iff $\langle I(e_0), \ldots, I(e_{n-1}) \rangle \in I(p)(w)$.
(c) $\models_{I,w} \neg A$ iff $\not\models_{I,w} A$.
(d) $\models_{I,w} A \wedge B$ iff $\models_{I,w} A$ and $\models_{I,w} B$.
(e) $\models_{I,w} (\forall x) A$ iff $\models_{I[d/x],w} A$ for all $d \in \mathbf{D}$.

Furthermore, $\models_I A$ means that A is true in I at all worlds, that is, I is a model of A, and $\models A$ means that A is true in any interpretation of IL.

This definition is incomplete. We must define the semantics of intensional operators available in the language. For instance, consider two classical modal operators $\Box$ (*necessary*) and $\Diamond$ (*possible*) [14].

In Kripke-style semantics for modal logic, the meanings of $\Box$ and $\Diamond$ are determined by an "accessibility" relation R. Informally, $\Box A$ is true at a world w iff A is true at all worlds accessible from w; and $\Diamond A$ is true at w iff A is true at some world accessible from w. More formally,

- $\models_{I,w} \Box A$ iff $\models_{I,v} A$ for all $\langle w, v \rangle \in R$
- $\models_{I,w} \Diamond A$ iff $\models_{I,v} A$ for some $\langle w, v \rangle \in R$

where I is an intensional interpretation, and $w \in \mathscr{U}$. Note that $\Diamond A$ and $\neg \Box \neg A$ are logically equivalent; $\Diamond$ is the dual of $\Box$.

If $R = \mathscr{U} \times \mathscr{U}$, this gives a Kripke-style semantics for the modal logic $S5$ [14]. The traditional Kripke approach is, however, too restrictive, because it limits us to a dual pair of intensional operators. We could extend it in the obvious way, by allowing a family of dual pairs, each with its own accessibility relation. This is better but still not truly general because, as Scott [28] and others have pointed out, there are many natural intensional operators that cannot be defined in terms of an accessibility relation alone. Since we would like to develop a language-independent theory for intensional logic programming languages, there is no reason why we should restrict ourselves to those logics for which a Kripke-style semantics is possible.

There are more general approaches to the semantics of intensional logic, including the "neighborhood" semantics of Scott [28] and Montague [20]. For a detailed exposition of more general approaches and their relative strengths, we refer the reader to the literature (see [39] and [8]). Neighborhood semantics provide us with an abstract characterisation of intensional operators which we can exploit to explore the properties of intensional logics under discussion. Later in this paper, we will essentially use neighborhood semantics as the basis of our theory, but we will also make use of Kripke-style of semantics for illustrative purposes.

3. INTENSIONAL LOGIC PROGRAMMING

We start by defining an intensional logic program as a set of intensional Horn clauses. The basic building blocks in an intensional logic program are *intensional units* defined inductively as follows:

- All atomic formulas are intensional units.
- If A is an intensional unit and ∇ is a unary intensional operator of IL, then ∇A is an intensional unit.

We adopt the clausal notation [16] for intensional logic programs. All variables in an intensional Horn clause are assumed to be universally quantified. For convenience, we will use upper-case letters for variables, and lower-case letters for function and predicate symbols.

Definition 3.1.

(a) An intensional program clause is the universal closure of a clause of the form $A \leftarrow B_0, \ldots, B_{n-1}$ $(n \geq 0)$, where each B_i and A are intensional units.
(b) An intensional goal clause is the universal closure of a clause of the form $\leftarrow B_0, \ldots, B_{n-1}$ $(n > 0)$, where each B_i is an intensional unit.
(c) An intensional Horn clause is either an intensional goal clause or an intensional program clause.

The informal semantics of an intensional program clause $A \leftarrow B_0, \ldots, B_{n-1}$ is defined as follows: at all worlds $w \in \mathscr{U}$, for each variable assignment, if all of $B_0, \ldots, B_{n-1}$ are true, then A is true. Then an intensional logic program consists of the conjunction of a set of intensional program clauses regarded as axioms true at all worlds in $\mathscr{U}$. Intensional goal clauses are also called queries.

3.1. Chronolog: A Temporal Logic Programming Language

Temporal logic [25] can be regarded as a special case of intensional logic where the set of possible worlds $\mathscr{U}$ models a collection of moments in time, usually discrete, linearly ordered, without a last moment. The temporal logic of the temporal logic programming language Chronolog [22, 26] has two temporal operators, `first` and `next`, which refer to the initial and the next moment in time, respectively. Here the set of possible worlds is the set ω of natural numbers.

A temporal interpretation I basically assigns meanings to all elements of the language at each moment of time in ω. The Kripke-style semantics of the temporal operators `first` and `next` are defined in terms of the satisfaction relation $\models$ as follows. Let R_f and R_n be the accessibility relations associated with `first` and `next`. Formally, the semantics of `first` and `next` are as follows:

- $\models_{I,t} \texttt{first}A$ iff $\models_{I,x} A$ for all $\langle t, x\rangle \in R_f$
- $\models_{I,t} \texttt{next}A$ iff $\models_{I,x} A$ for all $\langle t, x\rangle \in R_n$

where A is a formula, I is a temporal interpretation, and $t \in \omega$. It is not hard to see that `first` and `next` are the necessity operators corresponding to the accessibility relations $R_f = \{\langle t, 0\rangle | t \in \omega\}$ and $R_n = \{\langle t, t+1\rangle | t \in \omega\}$. Notice that these relations are single valued; they are functions, namely $\lambda t.0$ and $\lambda t.t+1$. Therefore, the temporal operators `first` and `next` are self-dual; for example, $\neg \texttt{first} \neg A \leftrightarrow \texttt{first}A$.

The following Chronolog program taken from [37] defines the predicate `fib` which is true of $t+1$th Fibonacci number at time t for all $t \in \omega$ and no other. Read all clauses as assertions true at all moments in time.

```
first fib(0) ←
first next fib(1) ←
next next fib(N) ← next fib(X), fib(Y), N is X+Y
```

The first two clauses define the first two Fibonacci numbers as 0 and 1; the last clause defines the current Fibonacci number as the sum of the previous two. For instance, the answer to the query `← first next next fib(X)` is a substitution instance of the query with `X` replaced by `1` (at any given moment in time).

Temporal logic programming has the potential for describing nonterminating computations naturally. For instance, a query like `← fib(X)` may trigger an attempt to prove `fib(X)` at all moments in time, since it is an open-ended query and actually stands for an infinite series of *closed* queries, `← first fib(X)`, `← first next fib(X)`, and so forth; a closed query being the conjunction of formulas of the form $\texttt{first next}^n A$ where A is an atom and $\texttt{next}^n$ represents n successive applications of `next`. The answers to the original query are those answers to closed queries obtained from it.

The following program adapted from [30] in Concurrent Prolog defines the predicate `fib` as true of the *infinite list* of all Fibonacci numbers.

```
fib([0,1|L]) ← fibtest([0,1|L])
fibtest([X,Y,Z|L]) ← Z is X+Y, fibtest([Y,Z|L])
```

As mentioned in [35], the least Herbrand model of such a program is empty, and the intended meaning of the program could be modeled by the greatest fixpoint semantics. On the other hand, the minimum temporal Herbrand model [22] of the corresponding Chronolog program is exactly what we wanted in the first place.

3.2. A Spatial Logic Programming Language

We now introduce another ILP language, which is based on a two-dimensional (spatial) logic. The underlying logic has $\mathcal{Z} \times \mathcal{Z}$ as the set of possible worlds $\mathcal{U}$, where $\mathcal{Z}$ is the set of integers, and six intensional operators. We regard $\mathcal{Z} \times \mathcal{Z}$ as a collection of (x, y)-coordinates of a plane (or grid) with an absolute reference point $\langle 0, 0 \rangle$ which is analogous of 0 in the temporal logic of Chronolog.

Intensional (spatial) operators are `side`, `edge`, `north`, `south`, `west` and `east`. The informal semantics of these operators are given as follows: let A be a formula, $\langle x, y \rangle \in \mathcal{U}$ and I be a spatial interpretation. A *spatial* interpretation assigns meanings to all elements of the language at all spatial points. Any formula of the form `side`A is true in I at $\langle x, y \rangle$ iff A is true in I at $\langle 0, y \rangle$. Any formula of the form `west`A is true in I at $\langle x, y \rangle$ iff A is true in I at $\langle x - 1, y \rangle$. Any formula of the form `east`A is true in I at $\langle x, y \rangle$ iff A is true in I at $\langle x + 1, y \rangle$. Similarly, the operators `edge`, `north` and `south` operate on the y coordinate of a given world.

We now give an example of spatial logic programming which defines the predicate `pascal`. Read all clauses as assertions true at all points in the two-dimensional space.

```
side edge pascal(1) ←
side pascal(X) ← side north pascal(X)
edge pascal(X) ← edge west pascal(X)
pascal(X) ← north pascal(Y), west pascal(Z), X is Y+Z
```

If these are the only axioms for the `pascal` predicate, Pascal's triangle is constructed on the south-east quadrant, whose apex is at $\langle 0, 0 \rangle$, i.e., at $\langle 0, 0 \rangle$, `pascal(1)` is true (from the first clause). Figure 1 shows an approximate graphic representation of Pascal's triangle as specified by the `pascal` predicate.

This spatial language is in fact more expressive (powerful) than the temporal language Chronolog, because any Chronolog program can be rewritten as a spatial program by replacing all temporal operators by their spatial counterparts over one of the spatial dimensions.

3.3. A Three-Dimensional Logic Programming Language

Let us combine the two languages we have introduced in the preceding sections to obtain a (three-dimensional) ILP language [23]. The underlying logic now incorporates $\mathcal{Z} \times \mathcal{Z} \times \omega$ as the set of possible worlds $\mathcal{U}$ and employs all the intensional operators defined previously. A triple $\langle x, y, z \rangle \in \mathcal{U}$ is interpreted as representing

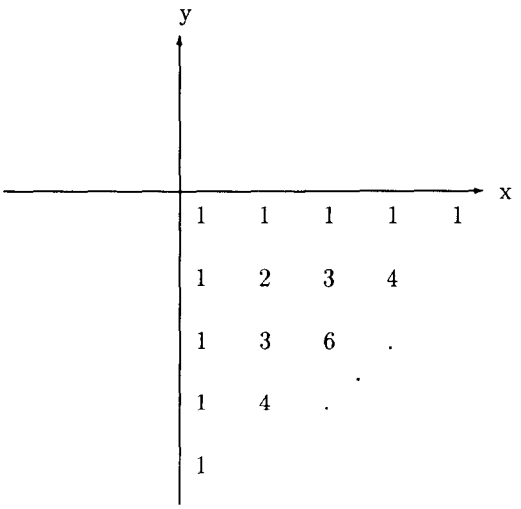

FIGURE 1. Pascal's triangle on the south-east quadrant.

the coordinates of some world, the first two of which refer to the location of the world on a plane and the last coordinate refers to a moment in time. Of course, all intensional operators work on their respective coordinates.

Now we will give an example of three-dimensional intensional logic programming from [23]. Perhaps Conway's game of life is one of the best examples which include relative references to the neighbors of a point in space at different moments in time. The game involves a (possibly infinite) plane divided into grids. Inside each grid (or cell) resides an organism that may become alive or dead depending on the status of its immediate neighbors in the surrounding cells on the plane. The game starts with an initial configuration on the plane in which some of the organisms are alive.

Supposing the initial configuration is defined elsewhere, the following program describes all relationships and state changes in the game.

```
next organism(alive) ← neighbors(L), count-alive(L,2)
next organism(alive) ←
     organism(alive), neighbors(L), count-alive(L,3)
next organism(dead) ← neighbors(L), lonely(L)
next organism(dead) ← neighbors(L), overcrowded(L)
next organism(dead) ←
     organism(dead), neighbors(L), count-alive(L,3)
neighbors([X1,X2,X3,X4,X5,X6,X7,X8]) ←
     north west organism(X1), north organism(X2),
     north east organism(X3), east organism(X4),
     south east organism(X5), south organism(X6),
     south west organism(X7), west organism(X8)
lonely(L) ← count-alive(L,X), X < 2
overcrowded(L) ← count-alive(L,X), X > 3
count-alive([],0) ←
count-alive([alive|L],N) ← count-alive(L,X), N is X+1
count-alive([dead|L],X) ← count-alive(L,X)
```

Read all clauses as assertions true at all worlds. We will briefly explain what the first six clauses mean. The first clause says that an organism will become alive at the next moment if exactly two of its neighbors are alive at the current moment no matter if the organism itself is alive or dead. This clause also covers the case where the birth of an organism will occur at the next moment if it is dead and exactly two of its neighbors are alive at the current moment. The second clause says that an alive organism will continue to live at the next moment if exactly three of its neighbors are alive at the current moment. The next two clauses state that an organism will become dead at the next moment if it is lonely (less than two neighbors are alive) or the surrounding area is overcrowded (more than three neighbors are alive). The fifth clause says that an organism will stay dead if it is already dead and has exactly three alive neighbors. The sixth clause simply bundles up the status of the neighbors of a given cell in a list for further use. The rest of the clauses define some auxiliary predicates. According to the program, note that at any world exactly one of the atoms `organism(alive)` and `organism(dead)` is true.

4. INTENSIONAL SEMANTICS

This section lays down the groundwork for a language-independent model-theoretical investigation of intensional logic programming. As mentioned before, we will adopt the more general semantics of intensional operators of Scott [28] and Montague [20] instead of Kripke's. This is because we would like our theory to be general enough to apply intensional logics for which a Kripke-style of semantics is not possible. Moreover, Scott-Montague semantics will provide us with an abstract characterisation of intensional logic, according to which intensional operators receive a denotation reflecting the mathematical properties of the intensional logic under discussion.

Let us call the intensional language under discussion IL. Let $\|A\|^I$ denote the meaning of a formula A of the language in some intensional intepretation I. As A may have different values at different possible worlds, $\|A\|^I$ is really a function, i.e., $\|A\|^I \in [\mathscr{U} \rightarrow 2]$ where $2 = \{0, 1\}$ and the set of all functions from $\mathscr{U}$ into 2 is denoted by $[\mathscr{U} \rightarrow 2]$. We write 0 for *false* and 1 for *true*. $\|A\|^I$ is also called an *intension* which, given an element $w \in \mathscr{U}$, returns the extension (0 or 1) of A at w. Note that $\|A\|^I$ can also be viewed as a subset of $\mathscr{U}$, whose elements are all the possible worlds at which A is true, i.e., $\|A\|^I = \{w \in \mathscr{U} \mid \models_{I,w} A\}$.

Note that $[\mathscr{U} \rightarrow 2]$, or equivalently $P(\mathscr{U})$ together with the usual set operations and a complementation operation relative to $P(\mathscr{U})$, is a complete Boolean algebra denoted by $\langle P(\mathscr{U}), \varnothing, \mathscr{U}, \neg, \cap, \cup \rangle$. Here $\neg$ is the complementation operation. We also have that $P(\mathscr{U})$ is a complete lattice denoted by $\langle P(\mathscr{U}), \subseteq \rangle$.

4.1. Semantics of Intensional Operators

Intensional operators take formulas as their arguments, and the denotations of formulas are intensions. If $\triangledown$ is a unary operator of IL, its denotation is a function in $[P(\mathscr{U}) \rightarrow P(\mathscr{U})]$ [28]. Then the definitions of the satisfaction relation $\models$ can be extended to assign meanings to formulas of the form $\triangledown A$ as follows.

Definition 4.1. Let ∇ be a unary intensional operator of *IL* and $\|\nabla\| = \Theta$ for some element $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. Then the semantics of the formulas of the form ∇A are given by the following:

$$\models_{I,w} \nabla A \text{ iff } w \in \|\nabla\|(\|A\|^I),$$

where I is an intensional interpretation and $w \in \mathscr{U}$.

It is easy to see that this approach allows us to formalize traditional modal operators. Suppose R is an accessibility relation and that we wish $\Box$ and $\Diamond$ to be the necessity and possibility operations associated with R. Let I be an intensional interpretation. If we set

$$\begin{aligned}\|\Diamond\|(\|A\|^I) &= \{w \in \mathscr{U} \mid \models_{I,w} \Diamond A\}\\ &= \{w \in \mathscr{U} \mid \models_{I,v} A \text{ for some } \langle w, v\rangle \in R\}\\ &= \{w \in \mathscr{U} \mid v \in \|A\|^I \text{ for some } \langle w, v\rangle \in R\}.\end{aligned}$$

We have that $\|A\|^I \in P(\mathscr{U})$. Therefore, the function $\|\Diamond\|$ can be obtained from the above expression by lambda abstraction:

$$\|\Diamond\| = \lambda X.\{w \in \mathscr{U} \mid v \in X \text{ for some } \langle w, v\rangle \in R\}.$$

The function $\|\Box\|$ can be formed in a similar fashion. We omit the details.

$$\|\Box\| = \lambda X.\{w \in \mathscr{U} \mid v \in X \text{ for all } \langle w, v\rangle \in R\}.$$

For modal logic *S*5 in which case $R = \mathscr{U} \times \mathscr{U}$, the definitions of $\|\Box\|$ and $\|\Diamond\|$ can be simplified further:

$$\|\Diamond\| = \lambda X.\begin{cases}\varnothing & \text{if } X = \varnothing\\ \mathscr{U} & \text{otherwise.}\end{cases} \qquad \|\Box\| = \lambda X.\begin{cases}\mathscr{U} & \text{if } X = \mathscr{U}\\ \varnothing & \text{otherwise.}\end{cases}$$

We will now construct the denotations of the temporal operators `first` and `next`. Consider the semantics of a formula of the form `first`A in a temporal interpretation I. Then, from the definition of the satisfaction relation $\models$, the function $\|\texttt{first}\|$ can be obtained as follows.

$$\begin{aligned}\|\texttt{first}\|(\|A\|^I) &= \{t \in \omega \mid \models_{I,t} \texttt{first}A\}\\ &= \{t \in \omega \mid \models_{I,x} A \text{ for all } \langle t, x\rangle \in R_f\}\\ &= \{t \in \omega \mid 0 \in \|A\|^I\}, \text{ since } R_f \text{ is functional and } x = 0.\end{aligned}$$

We can now obtain the function $\|\texttt{first}\|$ by lambda abstraction.

$$\|\texttt{first}\| = \lambda X.\{t \in \omega \mid 0 \in X\}.$$

Similarly, the function $\|\texttt{next}\|$ can be formed as $\|\texttt{next}\| = \lambda X.\{t \in \omega \mid t + 1 \in X\}$.

4.2. Neighborhood Semantics

Suppose ∇ is an intensional operator of *IL*. Then $\|\nabla\| = \Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. Associate with Θ an indexed family of subsets of $P(\mathscr{U})$ by the following: let $\Theta|_w = \{X \in P(\mathscr{U}) \mid w \in \Theta(X)\}$ for any $w \in \mathscr{U}$. In other words, $\Theta|_w$ consists of sets of "neighborhoods" of w with respect to Θ. Note that w is not necessarily a member of each of these neighborhoods. Let A be a formula of *IL*, I be an intensional

interpretation of IL, and $w \in \mathscr{U}$. Then the meaning of a formula of the form ∇A at w in I can be defined as follows:

$$\models_{I,w} \nabla A \text{ iff } \|A\|^I \in \Theta|_w.$$

The above statement says that ∇A is true at w in I just in case the set of worlds at which A is true in I is one of the neighborhoods of w with respect to Θ. This approach is called *neighborhood semantics* and it is attributed to Scott [28] and Montague [20].

For instance, consider the temporal operator `next`. Then the corresponding indexed family for $\|\texttt{next}\|$ can be defined as follows. For all $t \in \omega$,

$$\begin{aligned}(\|\texttt{next}\|)|_t &= \{X \in P(\omega) \mid t \in \|\texttt{next}\|(X)\} \\ &= \{X \in P(\omega) \mid t+1 \in X\}, \text{ by the definition of } \|\texttt{next}\|.\end{aligned}$$

Given an element $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$, we have described how to obtain the corresponding indexed family of neighborhoods. We can go in the opposite direction as well. Let $\{\mathscr{N}_w\}_{w \in \mathscr{U}}$ be an indexed family of neighborhoods where for all $w \in \mathscr{U}, \mathscr{N}_w \in P(P(\mathscr{U}))$. Then an element $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$ that corresponds to the family can be obtained as follows:

$$\Theta = \lambda X.\{w \in \mathscr{U} \mid X \in \mathscr{N}_w\}.$$

Therefore, both of the approaches lead to the same semantics.

In the examples given above, we employed accessibility relations and the satisfaction relation $\models$ to obtain the denotations of certain intensional operators in terms of neighborhood semantics. This in no way suggests that neighborhood semantics is equivalent to Kripke-style of semantics for intensional logic. In fact, as van Benthem [32] points out, Kripke-style of semantics based on accessibility relations may be cast as neighborhood semantics, but not conversely.

4.3. Properties of Intensional Operators

We will investigate several properties of intensional operators. These properties will be used later to impose restrictions on intensional logic programming languages. Note that some of the following results appeared in [23] and [24] where the foundations of a language-independent theory for intensional logic programming were originally outlined.

Our first requirement of an intensional operator of IL is that its denotation be monotonic. Monotonicity simply implies that if we know more information about the argument of a function, we shall know no less about the result.

Definition 4.2. Let $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. We say that Θ is monotonic iff for all X and $Y \in P(\mathscr{U})$, $X \subseteq Y$ implies $\Theta(X) \subseteq \Theta(Y)$.

The denotation of negation $\|\neg\|$ is not monotonic. Indeed, given an element X of $P(\mathscr{U})$, $\|\neg\|$ returns the complement of X with respect to $\mathscr{U}$. It can be verified that the denotations of temporal operators `first` and `next` are both monotonic, and so are those of modal $\Box$ and $\Diamond$.

Monotonicity has some consequences in the neighborhood semantics. It can be shown that, for a given unary monotonic function Θ, the neighborhoods of any

$w \in \mathscr{U}$ with respect to Θ, are closed under superset relation (also called "supplemented" in the terminology of Chellas [9]).

We also need a property to ensure that if ∇ is an operator of IL with the property, then any formula of the form ∇A is guaranteed to have a model. Monotonicity is not enough for this purpose. Suppose `start` is a unary temporal operator where $\models_{I,t} \texttt{start}A$ iff $t = 0$ and $0 \in \|A\|^I$. The function $\|\texttt{start}\| \in [P(\omega) \to P(\omega)]$ is $\lambda X.\{0 \mid 0 \in X\}$. It can be shown that $\|\texttt{start}\|$ is monotonic, but any formula of the form $\texttt{start}A$ does not have any temporal model.

Below is the formal definition of the property of *universality*.

Definition 4.3. Let $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. We say that Θ is universal iff for some $X \in P(\mathscr{U}), \Theta(X) = \mathscr{U}$.

Clearly $\|\texttt{start}\|$ is not universal. Since $\|\texttt{first}\|(\{0\}) = \omega$, $\|\texttt{first}\|$ is an example of universal functions.

Let $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. When Θ is both universal and monotonic, we can obtain a stronger condition which says Θ turns universal truth into universal truth.

Lemma 4.1. Let $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. If Θ is universal and monotonic, then $\Theta(\mathscr{U}) = \mathscr{U}$, i.e., $\mathscr{U} \in \Theta|_w$ for all $w \in \mathscr{U}$.

The following axiom related to the temporal operator `first` states that `first` can be distributed over conjunction, and initial truths can be conjoined:

$$\texttt{first}(A \wedge B) \leftrightarrow (\texttt{first}A \wedge \texttt{first}B).$$

We need to encapsulate a similar property but at a more general and semantic level. Van Benthem [32] has introduced a similar notion of conjunctivity related to the necessity operator $\Box$ to study the conditions under which a neighborhood semantics may be replaced by a Kripke semantics.

Below is the formal definition of the property of *conjunctivity*.

Definition 4.4. Let $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. We say that Θ is conjunctive iff for all $\{X_\alpha\}_{\alpha \in S} \in P(P(\mathscr{U}))$, $\Theta(\bigcap_{\alpha \in S} X_\alpha) = \bigcap_{\alpha \in S} \Theta(X_\alpha)$.

Conjunctivity captures the following property besides monotonicity. Recall that for any $t \in \omega$, $(\|\texttt{first}\|)|_t = \{X \in P(\omega) \mid 0 \in X\}$. Then $\{0\}$ is the least element in $(\|\texttt{first}\|)|_t$, and, in fact, is the intersection of all elements of $(\|\texttt{first}\|)|_t$. In other words, $(\|\texttt{first}\|)|_t$ is closed under intersection and therefore, it contains a least element. This intersective property is what we are after.

Lemma 4.2. Let $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. Θ is conjunctive iff Θ is monotonic and for all $w \in \mathscr{U}$, $\Theta|_w \neq \varnothing$ implies that $\Theta|_w$ is closed under intersection.

PROOF. Suppose Θ is conjunctive. The monotonicity of Θ is trivially implied by the conjunctivity of Θ. As for the intersective property, pick any subset S of a nonempty $\Theta|_w$. Then for any member S_α of the subset, we have that $w \in \Theta(S_\alpha)$, which implies that $w \in \bigcap_\alpha \Theta(S_\alpha)$. But then the conjunctivity of Θ implies that $w \in \Theta(\cap S)$, and hence $\cap S \in \Theta|_w$. Conversely, suppose Θ is monotonic and has the intersective property. The monotonicity of Θ implies one-half of the conjunctivity of Θ. The intersective property provides the other half. We omit the details. $\Box$

The denotation of the $S5$ possibility operator $\|\Diamond\|$ is monotonic but not conjunctive. Consider the indexed family of subsets of $P(\mathscr{U})$ associated with $\|\Diamond\|$, i.e., for all $w \in \mathscr{U}$,

$$(\|\Diamond\|)|_w = \{X \in P(\mathscr{U}) \mid X \neq \emptyset\}.$$

Then for any $w \in \mathscr{U}$, $(\|\Diamond\|)|_w$ is not closed under intersection. The denotation of the $S5$ necessity operator $\|\Box\|$ is conjunctive and, in fact, for any $w \in \mathscr{U}$, $(\|\Box\|)|_w = \{\mathscr{U}\}$.

Let ∇ be an intensional operator of IL. Then the value of a formula of the form ∇A at a given world w may depend on the extensions of A at a set of worlds, possibly including w. If this set happens to be infinite, any machinery to prove ∇A at w may fail to terminate because A needs to be proved at all worlds in the set. Therefore, for practical reasons, another property is necessary to filter out such kind of intensional operators. Yaghi [40] investigated a similar property for the operators of the dataflow language Lucid [38]; Fitting [11] introduced the same notion to modular logic programming.

Below is the formal definition of *finitariness* (called compactness in Fitting's work).

Definition 4.5. Let $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. Θ is finitary iff for all $X \in P(\mathscr{U})$ and for all $w \in \mathscr{U}, w \in \Theta(X)$ implies $w \in \Theta(S)$ for some finite $S \subseteq X$.

Intuitively, this definition means that we can verify that $w \in \Theta(X)$ by only referring to a finite subset of X. For instance, consider $\|\texttt{first}\|$ and $\|\texttt{next}\|$. Let $t \in \omega$ and $X \in P(\omega)$. If $t \in \|\texttt{first}\|(X)$, then it must be the case that $0 \in X$ and $t \in \|\texttt{first}\|(\{0\})$. It follows that $\|\texttt{first}\|$ is finitary. Similarly, we can show that $\|\texttt{next}\|$ is finitary as well.

As for $S5$ necessity $\Box$, we have that $w \in \|\Box\|(X)$ for any $w \in \mathscr{U}$ iff $X = \mathscr{U}$; thus $\|\Box\|$ is not finitary. But $\|\Diamond\|$ is finitary, since $w \in \|\Diamond\|(X)$ implies that for any $z \in X, w \in \|\Diamond\|(\{z\})$.

Monotonic and finitary functions are generally called *continuous* in the literature [11, 31]. In the programming language semantics, computable functions are identified with continuous functions. Here we will adopt the following definition of continuity.

Definition 4.6. Let $\Theta \in [P(\mathscr{U}) \to P(\mathscr{U})]$. We say that Θ is continuous iff for all chains $\langle X_n \rangle_{n \in \omega}$ over $P(\mathscr{U})$, $\bigcup_{n \in \omega} \Theta(X_n) = \Theta(\bigcup_{n \in \omega} X_n)$.

The following theorem establishes the connection between continuity and monotonicity combined with finitariness. We omit the proof of this theorem, because it is very similar to that of an analogous theorem in the programming language semantics. Yaghi [40] in fact proved a similar theorem for the operators of the dataflow language Lucid.

Theorem 4.1. Let $\Theta \in [P(\mathscr{U})^n \to P(\mathscr{U})]$. Then Θ is continuous iff it is monotonic and finitary.

Consider the $S5$ necessity operator $\Box$. Since $\|\Box\|$ is not finitary, it fails to be continuous. Indeed, $\|\Box\|$ does not satisfy the definition of continuity. Let $\langle C_n \rangle_{n \in \omega}$

$= \langle \emptyset, \{w_0\}, \{w_0, w_1\}, \{w_0, w_1, w_2\}, \ldots \rangle$ be an ω-chain of elements of $P(\mathscr{U})$ where $\bigcup_{n \in \omega} C_n = \mathscr{U}$. For any $n \in \omega$, $\|\Box\|(C_n) = \emptyset$. Then $\|\Box\|(\bigcup_{n \in \omega} C_n) = \mathscr{U}$, but $\bigcup_{n \in \omega} \|\Box\|(C_n) = \emptyset$.

4.4. Monotonic Formulas as Intensional Operators

It is a common practice in logic for new operators to be introduced in a language by considering formulas as the definitions of operators in terms of already available ones. For instance, disjunction $\vee$ can be defined as $A \vee B =_{def} \neg(\neg A \wedge \neg B)$. Similarly, the possibility operator $\Diamond$ can be defined as $\Diamond A =_{def} \neg \Box \neg A$. As an alternative, we can directly extend the underlying language with extra (intensional) operators. In the first case, the denotation of a defined intensional operator can be obtained from the denotations of those operators used in the definition. In the second case, we explicitly specify the denotations of the extra intensional operators as elements of $[P(\mathscr{U}) \rightarrow P(\mathscr{U})]$.

In particular, we can directly define the semantics of $\neg$ and $\wedge$ as follows:

- $\models_{I,w} \neg A$ iff $w \in \neg \|A\|^I$
- $\models_{I,w} A \wedge B$ iff $w \in \|A\|^I \cap \|B\|^I$

where I is an intensional interpretation and $w \in \mathscr{U}$. We will also regard disjunction as part of the language whose denotation corresponds to set union. We will use negation, conjunction and disjunction as primitives along with other intensional operators available in the language to define new intensional operators.

Consider the underlying temporal language of Chronolog and a formula of the form $A \wedge \mathtt{next} A \wedge \cdots \wedge \mathtt{next}^{n-1} A$ where A is any formula. We denote this formula as $\bigwedge_{i \in n} \mathtt{next}^i A$. Then a new unary temporal operator, say $[n]$, can be defined as follows

$$[n]A =_{def} \bigwedge_{i \in n} \mathtt{next}^i A,$$

where A is a propositional variable, i.e., a place-holder for formulas. Then any formula of the form $[n]B$ would be regarded as a formula obtained from the definition of $[n]$ by substituting B for A. Equivalently, we can specify the denotation of $[n]$ as an element of $\Theta \in [P(\mathscr{U}) \rightarrow P(\mathscr{U})]$ as follows:

$$\|[n]\|(X) = \bigcap_{i \in n} \|\mathtt{next}\|^i(X).$$

Here $\|\mathtt{next}\|^i$ is the i-fold composition of $\|\mathtt{next}\|$. Note that $\|\mathtt{next}\|^0 = \lambda X.X$.

We will consider, as defining formulas for new operators, *monotonic formulas* of one propositional variable. Monotonic formulas with one propositional variable correspond to unary monotonic functions, but not conversely. We can pick a monotonic formula of IL and treat it as the definition of a new intensional operator $\triangledown$, or, equivalently, we can compose a new function Θ in terms of unions, intersections, compositions of given functions, and enrich IL with a new symbol $\triangledown$ by letting $\|\triangledown\| = \Theta$.

The properties of functions involved in the definitions of new functions are preserved under certain restrictions. The following lemma shows that functions composed out of monotonic functions are monotonic as well.

Lemma 4.3. Any function Θ in $[P(\mathscr{U})\rightarrow P(\mathscr{U})]$, defined in terms of unions, intersections, monotonic functions and compositions of such functions, is monotonic.

The following lemma states that universality is preserved for functions composed out of unions, intersections, and universal and monotonic functions. Both monotonicity and universality are needed for this lemma. Suppose $\Theta = \lambda X.\mathscr{U}$; then $\|\neg\|\circ\Theta$ is not universal, even though Θ and $\|\neg\|$ are.

Lemma 4.4. Any function Θ in $[P(\mathscr{U})\rightarrow P(\mathscr{U})]$, defined in terms of unions, intersections, universal and monotonic functions, and compositions of such functions, is universal.

We also have the following two lemmas.

Lemma 4.5. Any function Θ in $[P(\mathscr{U})\rightarrow P(\mathscr{U})]$, defined in terms of unions, intersections, monotonic and finitary functions, and compositions of such functions, is finitary.

Lemma 4.6. Any function Θ in $[P(\mathscr{U})\rightarrow P(\mathscr{U})]$, defined in terms of intersections, conjunctive functions and compositions of such functions, is conjunctive.

In summary, the definition of the satisfaction relation $\models$ can be extended in at least two different ways: if we consider $[n]$ as part of the language, then $\|[n]\| = \Theta$ for a particular $\Theta \in [P(\omega)\rightarrow P(\omega)]$ and

$$\models_{I,t}[n]A \text{ iff } t \in \Theta(\|A\|^I),$$

where I is a temporal interpretation and $t \in \omega$. If we regard $[n]$ as a defined operator, the semantics of $[n]A$ somewhat reads differently.

$$\models_{I,t}[n]A \text{ iff } t \in \bigcap_{i\in n}\|\mathrm{next}\|^i(\|A\|^I).$$

Technically, both of these definitions lead to a logic with the same expressive power. However, in the first case, $[n]B$ is a formula of the language, where $[n]$ is applied to two arguments, whereas in the second case, $[n]B$ refers to the defining formula of $[n]$ with A substituted by B. But again, from a model-theoretical point of view, there is no difference. Segerberg [29] presents a thorough discussion on this topic.

5. MODELS OF INTENSIONAL LOGIC PROGRAMS

In this section, we will develop a language-independent, model-theoretic semantics of intensional logic programs along the lines of van Emden-Kowalski [34]. Van Emden and Kowalski showed that the family of Herbrand models of a given logic program is nonempty and closed under intersection; thus the least Herbrand model of the program exists and it consists of the ground atomic consequences of the program. We will extend their result to intensional logic programming. However, we will not focus on any particular ILP language. From here on, unless otherwise stated, we *assume* that all (intensional) operators that can be used in intensional logic programs have the properties formulated in the previous section. In other words, if $\triangledown$ is an operator of the language, its denotation is universal, monotonic,

conjunctive and finitary. Therefore, IL is not just an arbitrary intensional logic, rather a logic with these constraints. The following results rely on this assumption.

The basic building blocks in an intensional logic program are now monotonic formulas, which we call *intensional units* in this context. In the previous section, we have shown that monotonic formulas in one propositional variable can be considered as the definitions of new intensional operators. Then intensional units that appear in the programs may be regarded as the applications of such defined operators to some other intensional units.

5.1. Intensional Herbrand Interpretations

We understand an intensional logic program $\mathscr{P}$ in terms of intensional interpretations as follows: $\mathscr{P}$ is true in an intensional interpretation I iff all clauses in $\mathscr{P}$ are true in I. A clause is true in I iff it is true in I at all worlds in $\mathscr{U}$.

Definition 5.1. Let I be an intensional interpretation of IL. Then I is a model of $\mathscr{P}$ iff I is a model for each clause $C \in \mathscr{P}$, that is, $\models_I \mathscr{P}$ iff for all $C \in \mathscr{P}$, $\models_I C$.

We call ground atomic formulas *intensional ground atoms*. Let $\mathscr{P}$ be an intensional logic program. Then the intensional Herbrand universe $U_{\mathscr{P}}$ of $\mathscr{P}$ is the set of all ground terms that can be constructed from the function symbols and constants that appear in $\mathscr{P}$. The intensional Herbrand base $B_{\mathscr{P}}$ of $\mathscr{P}$ is the set of all intensional ground atoms that can be constructed from the predicate symbols that appear in $\mathscr{P}$ with ground terms from $U_{\mathscr{P}}$ as arguments.

Intensional Herbrand interpretations of $\mathscr{P}$ have $U_{\mathscr{P}}$ as their domain. If I is an intensional Herbrand interpretation, it satisfies the condition that for all $e \in U_{\mathscr{P}}$, $I(e) = e$. Then I can be identified with a function H which assigns to each intensional ground atom $p(e_0, \ldots, e_{n-1}) \in B_{\mathscr{P}}$ an element of $P(\mathscr{U})$ by the following.

$$\langle e_0, \ldots, e_{n-1} \rangle \in I(p)(w) \text{ iff } w \in \| p(e_0, \ldots, e_{n-1}) \|^H.$$

We say that H is a model of $\mathscr{P}$, in notation $\models_H \mathscr{P}$, iff $\models_I \mathscr{P}$ for any I corresponding to H.

From here on, we will use these two dual notions of an intensional Herbrand interpretation interchangeably. We also say that a clause is true in an intensional Herbrand interpretation I iff all of its ground instances are true in I at all possible worlds.

We now define an ordering relation on intensional Herbrand interpretations. Let $\mathscr{I}(\mathscr{P})$ denote the set of intensional Herbrand interpretations of an intensional logic program $\mathscr{P}$.

Definition 5.2. Let I and $J \in \mathscr{I}(\mathscr{P})$. Then $I \sqsubseteq J$ iff $\|A\|^I \subseteq \|A\|^J$ for all $A \in B_{\mathscr{P}}$.

Note that $\langle \mathscr{I}(\mathscr{P}), \sqsubseteq \rangle$ is a complete lattice induced by the complete lattice of $\langle P(\mathscr{U}), \subseteq \rangle$. We also have that $\mathscr{I}(\mathscr{P})$ is in fact a complete Boolean algebra induced by the complete Boolean algebra of $P(\mathscr{U})$.

The following two lemmas justify that intensional Herbrand interpretations are sufficient for proving the unsatisfiability of a set of intensional Horn clauses.

Lemma 5.1. Let S be a set of intensional Horn clauses and suppose S has an intensional model. Then S has an intensional Herbrand model.

Lemma 5.2. Let $\mathscr{P}$ be an intensional logic program and A be an intensional unit, where all the atomic formulas that appear in A are in $B_{\mathscr{P}}$. Then $\mathscr{P} \cup \{\neg A\}$ is unsatisfiable in any intensional model of $\mathscr{P}$ iff no intensional Herbrand model of $\mathscr{P}$ satisfies $\mathscr{P} \cup \{\neg A\}$.

5.2. Model-Theoretic Semantics

Intensional logic programs have models under monotonicity and universality constraints. Consider negation whose denotation is not monotonic: $\|\neg\| = \lambda X. \neg X$. But $\|\neg\|$ is universal, since $\|\neg\|(\emptyset) = \mathscr{U}$. If any such operator is used in the head of any clause of an intensional logic program $\mathscr{P}$, $\mathscr{P}$ in general may not even be consistent.

Lemma 5.3. Let $\mathscr{P}$ be an intensional logic program, and $H_{\mathscr{P}}$ denote $\sqcup\mathscr{F}(\mathscr{P})$. Then $H_{\mathscr{P}}$ is a model of $\mathscr{P}$, i.e., $\models_{H_{\mathscr{P}}} \mathscr{P}$.

PROOF. We have that for all $A \in B_{\mathscr{P}}$, $\|A\|^{H_{\mathscr{P}}} = \mathscr{U}$. Consider any ground instance of any clause in $\mathscr{P}$ and let $A \leftarrow B_1, \ldots, B_n$ be one such ground instance. Here A is of the form $\triangledown A$, where $\triangledown$ is a sequence of intensional operators. Let $\|\triangledown\| = \Theta$ for some element Θ of $[P(\mathscr{U}) \rightarrow P(\mathscr{U})]$. We have that Θ is universal and monotonic by Lemmas 4.3 and 4.4. We also know that for any $F \in B_{\mathscr{P}}$, $\|F\|^{H_{\mathscr{P}}} = \mathscr{U}$ by construction, therefore, for all $w \in \mathscr{U}$, $w \in \|\triangledown A\|^{H_{\mathscr{P}}}$ iff $w \in \Theta(\|A\|^{H_{\mathscr{P}}})$ iff $w \in \Theta(\mathscr{U})$. Then by Lemma 4.1, $\Theta(\mathscr{U}) = \mathscr{U}$, therefore the ground instance must be true at w. Thus $\models_{H_{\mathscr{P}}} A \leftarrow B_1, \ldots, B_n$ for any ground instance of any clause in $\mathscr{P}$, which implies that $\models_{H_{\mathscr{P}}} \mathscr{P}$. □

Suppose `start` were a temporal operator of Chronolog. The lemma given above would not be valid for the single line program $\mathscr{P} = \{$`startp` $\leftarrow\}$ as we know that $\|$`start`$\|$ is not universal. In fact, $\mathscr{P}$ has no temporal Herbrand models.

Conjunctivity is related to the model intersection property:

Lemma 5.4. Let $\mathscr{P}$ be an intensional logic program. Let $A \in B_{\mathscr{P}}$ and $\triangledown$ be an intensional operator where $\|\triangledown\| = \Theta$ is universal and conjunctive. Let $M = \{I_\alpha\}_{\alpha \in S}$ be a family of intensional Herbrand interpretations of $\mathscr{P}$, where for all $I_\alpha \in M$, $\models_{I_\alpha} \triangledown A$. Then $\models_{\sqcap M} \triangledown A$.

PROOF. Since Θ is universal and conjunctive, for all $w \in \mathscr{U}, \Theta|_w \neq \emptyset$ by Lemma 4.1 and $\cap \Theta|_w \in \Theta|_w$ by Lemma 4.2. Given for any $w \in \mathscr{U}$ and for any $I_\alpha \in M$, $\|A\|^{I_\alpha} \in \Theta|_w$, we have that $\|A\|^{\sqcap M} = \cap_{\alpha \in S} \|A\|^{I_\alpha} \in \Theta|_w$, because $\Theta|_w$ is closed under intersection by Lemma 4.2. But this means that $\sqcap M$ is a model as well. □

The family of models of a formula of the form $\Diamond A$ is not closed under $\sqcap$-intersection where $\Diamond$ is the $S5$ possibility operator. Indeed, take all models which assign a different singleton set to A, then the denotation of A in the $\sqcap$-intersection is the empty set.

The following lemma states that the model $\sqcap$-intersection property smoothly extends to a family of intensional Herbrand models of an intensional logic

program. The proof of the lemma is along the lines of that of classical logic programming, the difference being that we now have to use the properties of monotonicity and conjunctivity in the proof.

Lemma 5.5. Let $\mathscr{P}$ be an intensional logic program and $M = \{I_\alpha\}_{\alpha \in S}$ be a nonempty family of intensional Herbrand models of $\mathscr{P}$. Then $\sqcap M$ is an intensional Herbrand model of $\mathscr{P}$, i.e., $\models_{\sqcap M} \mathscr{P}$.

PROOF. Suppose $\sqcap M$ is not a model of $\mathscr{P}$. Then there is a ground instance of an intensional clause in $\mathscr{P}$ of the form $A \leftarrow B_0, \ldots, B_{n-1}$ which is false in $\sqcap M$ at some $w \in \mathscr{U}$. That means all B_i's are true, but A is false in $\sqcap M$ at w. Since all B_i's are true at w in $\sqcap M$ and all the intensional operators in B_i's have the monotonicity property, it must be the case that all B_i's are true at w in all $I_\alpha \in M$. This implies that A must be true at w in all $I_\alpha \in M$. Then A is true in $\sqcap M$ at w by Lemma 5.4, which is a contradiction to the assumption that $\sqcap M$ is not a model of $\mathscr{P}$. □

Let $\mathscr{P} = \{\Diamond \mathrm{p} \leftarrow\}$ be an intensional logic program where $\Diamond$ is the $S5$ possibility operator. It can be verified that the model $\sqcap$-intersection property does not hold for $\mathscr{P}$, since Lemma 5.4 no longer applies for $\Diamond \mathrm{p}$.

The following theorem states that there is a model of an intensional logic program called the minimum intensional Herbrand model, which as far as declarative semantics is concerned, is all we need to know about the program. The theorem follows from Lemmas 5.3 and 5.5.

Theorem 5.1. Every intensional logic program $\mathscr{P}$ has a $\sqsubseteq$-minimum intensional Herbrand model $M_{\mathscr{P}}$, which is the $\sqcap$-intersection of all intensional Herbrand models of $\mathscr{P}$.

The theorem given below characterizes logical consequences of intensional logic programs as formulas of the form $\triangledown A$. Note that $\| \triangledown \|$ is universal, monotonic, and finitary, but does not have to be conjunctive. The proof of the following theorem is quite similar to that of an analogous theorem for classical logic programming.

Theorem 5.2. Let $\mathscr{P}$ be an intensional logic program and $A \in B_{\mathscr{P}}$. Then $\triangledown A$ is a logical consequence of $\mathscr{P}$ iff $\mathscr{P} \models_{M_{\mathscr{P}}} \triangledown A$.

5.3. The Fixpoint Semantics of Intensional Logic Programs

The continuous mapping $T_{\mathscr{P}}$ originally given in [34] provides the basis for fixpoint semantics and therefore establishes the connection between the model-theoretic and operational semantics of logic programs. The major result of the fixpoint theory of logic programs is that the prefixpoints of the mapping $T_{\mathscr{P}}$ are models of a logic program $\mathscr{P}$ and $lfp(T_{\mathscr{P}}) = T_{\mathscr{P}} \uparrow \omega = M_{\mathscr{P}}$. We will generalise this result to intensional logic programming.

Let $\mathscr{P}$ be an intensional logic program and $H \in \mathscr{F}(\mathscr{P})$ be an intensional Herbrand interpretation. Let $T_{\mathscr{P}} \in [\mathscr{F}(\mathscr{P}) \rightarrow \mathscr{F}(\mathscr{P})]$ where we want $T_{\mathscr{P}}(H)$ to be an intensional Herbrand interpretation satisfying the model-theoretical condition given

below: for all $w \in \mathscr{U}$,

$$w \in \|A\|^{T_{\mathscr{P}}(H)} \text{ iff } w \in \|B_i\|^H \text{ for all } i \in n,$$

where $A \leftarrow B_0, \ldots, B_{n-1}$ is a ground instance of some clause in $\mathscr{P}$. However, this condition does not explicitly specify what the intensions assigned to ground intensional atoms in $B_{\mathscr{P}}$ should be.

The major problem in defining $T_{\mathscr{P}}$ is that intensional operators may appear in the heads of program clauses in $\mathscr{P}$. Here A may be an intensional unit of the form $\triangledown B$ where $B \in B_{\mathscr{P}}$. Therefore we must elaborate what $T_{\mathscr{P}}(H)$ assigns to B. In case $\triangledown$ is the empty sequence, we have that $\|\triangledown\| = \lambda X.X$. Let $\Theta = \|\triangledown\|$ for universal and conjunctive Θ. For any given conjunctive Θ and for any $w \in \mathscr{U}$, we call $\bigcap \Theta|_w$ the *cluster* of $\Theta|_w$, i.e., the least element in $\Theta|_w$. Then the following is the formal definition of the mapping $T_{\mathscr{P}}$.

Definition 5.3. Let $\mathscr{P}$ be an intensional logic program and H be an intensional Herbrand intepretation of $\mathscr{P}$. Then $T_{\mathscr{P}}(H)$ is an intensional Herbrand interpretation defined as follows: for all $A \in B_{\mathscr{P}}$,

$$\|A\|^{T_{\mathscr{P}}(H)} = \cup\big\{X \mid \triangledown A \leftarrow B_0, \ldots, B_{n-1} \text{ is a ground instance of some clause in } \mathscr{P},$$
$$\text{and for some } w \in \mathscr{U},\ X = \cap\|\triangledown\|\,|_w \text{ and } w \in \|B_i\|^H \text{ for all } i \in n\big\}.$$

We will illustrate the importance of constraints on intensional operators in the fixpoint theory of intensional logic programs. If $\|\triangledown\| = \Theta$ is not conjunctive, then the cluster of $\Theta|_w$ for some $w \in \mathscr{U}$ is not an element of $\Theta|_w$. Consider the following intensional logic program: $\mathscr{P} = \{\Diamond \mathrm{p} \leftarrow\}$. Let $H_\varnothing$ denote $\sqcap\mathscr{F}(\mathscr{P})$. We have that for all $A \in B_{\mathscr{P}}$, $\|A\|^{H_\varnothing} = \varnothing$ and $\cap\|\Diamond\|\,|_w = \varnothing$ for all $w \in \mathscr{U}$. Then $T_{\mathscr{P}}(H_\varnothing) = H_\varnothing$, which means that $H_\varnothing$ is a fixpoint of $T_{\mathscr{P}}$; indeed, the least one, and yet it is not a model of $\mathscr{P}$! Similar anomalies occur when $\|\triangledown\|$ is not universal.

The mapping $T_{\mathscr{P}}$ shares the properties of that of classical logical programs [17, 34] and that of multiple-valued logic programming [7, 12]. We will first show that $T_{\mathscr{P}}$ is continuous.

Lemma 5.6. Let $\mathscr{P}$ be an intensional logic program. Then $T_{\mathscr{P}}$ is continuous, that is, for any chain $C = \langle C_n \rangle_{n \in \omega}$ of intensional Herbrand interpretations of $\mathscr{P}$, $T_{\mathscr{P}}(\sqcup_{n \in \omega} C_n) = \sqcup_{n \in \omega} T_{\mathscr{P}}(C_n)$.

PROOF. We proceed as follows: for all $A \in B_{\mathscr{P}}$ and $w \in \mathscr{U}, w \in \|A\|^{T_{\mathscr{P}}(\sqcup C)}$

iff for some ground instance $\triangledown A \leftarrow B_0, \ldots, B_{n-1}$ of some clause in $\mathscr{P}$, $w \in \cap\|\triangledown\|\,|_v$ for some $v \in \mathscr{U}$, where $v \in \|B_i\|^{\sqcup C}$ for all $i \in n$

iff for some $C_\alpha \in C$, $v \in \|B_i\|^{C_\alpha}$ for all $i \in n$, since the denotations of all intensional operators that appear in $\mathscr{P}$ are continuous

iff $w \in \|A\|^{T_{\mathscr{P}}(C_\alpha)}$

iff $w \in \bigcup_{\alpha \in \omega} \|A\|^{T_{\mathscr{P}}(C_\alpha)}$

iff $w \in \|A\|^{\sqcup_{\alpha \in \omega} T_{\mathscr{P}}(C_\alpha)}$. $\square$

Lemma 5.6 fails for the following intensional logic program in which $\Box$ is the $S5$ necessity operator: $\mathscr{P} = \{\mathrm{p(a)} \leftarrow \Box\,\mathrm{q(a)}, \mathrm{q(X)} \leftarrow\}$. Pick an ω-chain $\langle C_n \rangle_{n \in \omega} = \langle C_0, C_1, \ldots \rangle$ of intensional Herbrand interpretations of $\mathscr{P}$ where each C_n is defined as follows: $\|\mathrm{p(a)}\|^{C_n} = \varnothing$ and $\|\mathrm{q(a)}\|^{C_n} = \{w_0, w_1, \ldots, w_{n-1}\} \subseteq \mathscr{U}$. Then $\|\mathrm{q(a)}\|^{\sqcup_{n \in \omega} C_n} = \mathscr{U}$, which implies that $\|\mathrm{p(a)}\|^{T_{\mathscr{P}}(\sqcup_{n \in \omega} C_n)} = \mathscr{U}$, but $\|\mathrm{p(a)}\|^{\sqcup_{n \in \omega} T_{\mathscr{P}}(C_n)} = \varnothing$. Therefore, $T_{\mathscr{P}}$ is not continuous. Similarly, the continuity of $T_{\mathscr{P}}$ fails if intensional operators whose denotations are nonmonotonic are used in intensional logic programs. This fact is also noted in [17] for logic programs with negation.

The following lemma characterises intensional Herbrand models of $\mathscr{P}$ in terms of $T_{\mathscr{P}}$. The lemma fails if conjunctivity, and universality constraints are not satisfied.

Lemma 5.7. Let $\mathscr{P}$ be an intensional logic program. Let I be an intensional Herbrand intepretation of $\mathscr{P}$. Then I is a model of $\mathscr{P}$ iff $T_{\mathscr{P}}(I) \sqsubseteq I$.

The following theorem gives the fixpoint characterisation of the minimum intensional Herbrand model of an intensional logic program $\mathscr{P}$.

Theorem 5.3. Let $\mathscr{P}$ be an intensional logic program. Then $lfp(T_{\mathscr{P}}) = T_{\mathscr{P}} \uparrow \omega = M_{\mathscr{P}}$ provided that all (intensional) operators have the properties, that is, the denotations of intensional operators are universal, monotonic, finitary and conjunctive.

PROOF. Suppose all (intensional) operators have the properties. Since $T_{\mathscr{P}}$ is continuous by Lemma 5.6, it follows that the closure ordinal of $T_{\mathscr{P}}$ is $\leq \omega$. Thus the least fixpoint of $T_{\mathscr{P}}$, $lfp(T_{\mathscr{P}}) = T_{\mathscr{P}} \uparrow \omega$. We also have that $M_{\mathscr{P}} = \sqcap\{I \mid T_{\mathscr{P}}(I) \sqsubseteq I\}$ by Lemma 5.7 and Theorem 5.1. By a version of Knaster-Tarski fixpoint theorem [17], $lfp(T_{\mathscr{P}}) = \sqcap\{I \mid T_{\mathscr{P}}(I) \sqsubseteq I\}$. Therefore, $lfp(T_{\mathscr{P}}) = T_{\mathscr{P}} \uparrow \omega = M_{\mathscr{P}}$. $\Box$

6. APPLYING THE THEORY

We have shown that each constraint on intensional operators has model-theoretical consequences on the semantics of ILP. However, it is possible to relax some of these constraints, because we do not need all of them to prove each theorem, except monotonicity. Finitariness is not needed for intensional operators used in the heads of intensional program clauses. Conjunctivity and universality are not needed for intensional operators used in the bodies of intensional program clauses. We can now formulate the following theorem:

Theorem 6.1. Let $\mathscr{P}$ be an intensional logic program. Then $\mathscr{P}$ has the minimum model property which can also be characterised by the least fixpoint of the mapping $T_{\mathscr{P}}$ provided that the denotations of all (intensional) operators that appear in the heads of the clauses in $\mathscr{P}$ are universal, monotonic and conjunctive, and the denotations of all (intensional) operators that appear in the bodies of the clauses in $\mathscr{P}$ are monotonic and finitary.

PROOF. We will outline an informal proof without any details. If all the intensional operators appearing in the heads have monotonicity and universality properties, the model existence lemma (Lemma 5.3) holds for $\mathscr{P}$. To prove that the model-intersection property (Lemma 5.5) holds for $\mathscr{P}$, we need to use monotonicity for all

intensional operators in $\mathscr{P}$, and conjunctivity for the intensional operators appearing in the heads. Then the minimum model semantics for $\mathscr{P}$ follows. As for the fixpoint semantics, we need to use monotonicity, universality, and conjunctivity for the intensional operators appearing in the heads, and monotonicity and finitariness, that is, continuity, for the intensional operators appearing in the bodies. □

In short, the precondition of the theorem is sufficient to prove individual theorems about intensional logic programs, as can be seen from the proofs of the results presented in the previous section. Theorem 6.1 can be applied to a variety of ILP languages including Chronolog. The temporal operators `first` and `next` have all the desired properties, therefore Chronolog programs enjoy Theorem 6.1. In [22], the declarative semantics of Chronolog programs are defined in terms of canonical atoms and temporal Herbrand models. We are also considering an extension of Chronolog with negative moments in time. Here $\mathscr{U}$ is the set of integers, $\mathscr{T}$, and the underlying language is extended with an extra unary operator `pre` to refer to the previous moment in time. The denotation of `pre` is given as follows:

$$\|\texttt{pre}\| = \lambda X.\{t \in \mathscr{T} \mid t-1 \in X\}.$$

It can be shown that $\|\texttt{pre}\|$ has all the desired properties, therefore, temporal logic programs in Chronolog with negative time enjoy Theorem 6.1 as well.

Theorem 6.1 also applies to the spatial and three-dimensional languages described previously, since all intensional operators of these languages have the desired properties. As for extensional operators like $\wedge$ and $\vee$, $\|\wedge\| = \cap$ is monotonic, universal, finitary and conjunctive; $\|\vee\| = \cup$ is monotonic, universal and finitary but not conjunctive, which means that $\vee$ cannot be used in the heads of intensional program clauses whereas $\wedge$ can be used anywhere in programs.

6.1. Temporal Logic Programming

Templog, originally proposed by Abadi and Manna [1], incorporates temporal modalities $\Diamond$ and $\Box$ as well as next time operator $\bigcirc$, same as `next` of Chronolog. Templog allows monotonic formulas in the bodies of program clauses. Moreover, there are two kinds of program clauses in Templog:

- Permanent clauses of the form $\Box(\bigcirc^k C \leftarrow B_0, \ldots, B_{n-1})$
- Initial clauses of the form $\bigcirc^k C \leftarrow B_0, \ldots, B_{n-1}$ or $\Box \bigcirc^k C \leftarrow B_0, \ldots, B_{n-1}$

where all of $B_0, \ldots, B_{n-1}$ are monotonic formulas and C is an atomic formula. Initial clauses are interpreted as assertions true at the initial moment in time, whereas permanent clauses are true at all moments in time. Therefore, the temporal operator `first` is implicitly available.

In our approach, the necessity operator $\Box$ is implicit; we regard all program clauses as true assertions at all moments in time. Initial clauses in Templog programs may be turned into equivalent permanent clauses by applying `first` to the whole clause by the following. Let $A \leftarrow B_0, \ldots, B_{n-1}$ be an initial clause. If A is

of the form $\Box \bigcirc^k C$ for an atom C, then the corresponding permanent clause is

$$\bigcirc^k C \leftarrow \texttt{first}\, B_0, \ldots, \texttt{first}\, B_{n-1};$$

otherwise; it is

$$\texttt{first}\, \bigcirc^k C \leftarrow \texttt{first}\, B_0, \ldots, \texttt{first}\, B_{n-1}.$$

It can be shown that this transformation is correct and preserves modelhood.

Then the next question is if Templog with `first` enjoys the minimum model semantics. The denotations of temporal $\Diamond$ and $\Box$ are given as:

$$\|\Box\| = \lambda X.\{t \in \omega \mid \text{for all } t \leq z, z \in X\}$$

$$\|\Diamond\| = \lambda X.\{t \in \omega \mid \text{for some } t \leq z, z \in X\}.$$

It can be shown that temporal $\|\Box\|$ and $\|\Diamond\|$ share the properties of the denotations of $S5$ modalities $\Box$ and $\Diamond$. Moreover, Abadi and Manna [1] restrict the use of $\Diamond$ to the bodies of temporal clauses, and $\Box$ to the heads of temporal clauses. Therefore, Theorem 6.1 applies to Templog programs. Baudinet [5, 6] independently developed the declarative semantics of Templog programs.

Temporal Prolog, introduced by Gabbay [13], allows in the heads of the program clauses temporal operators such as possible in the future and possible in the past, denoted by F and P, and temporal programs may include clauses whose heads contain conjunctions of other clauses. Even if we consider Horn clause subsets of this language, we have a negative result: Gabbay's system is not contained in our theory; one of the reasons being that $\|F\|$ and $\|P\|$ are nonconjunctive, which implies that Temporal Prolog does not enjoy Theorem 6.1.

6.2. *Modal Logic Programming*

Molog, proposed by Fariñas del Cerro [10], is a framework for modal logic programming. The user fixes the underlying modal logic. In [10], The Kripke-style of semantics of modal operators are defined in terms of the satisfaction relation $\models$, and the modal operators are grouped in two categories, universal and existential. Furthermore, no constraints are imposed on the use of modal operators. Note that $\Box$ is a universal operator, and $\Diamond$ is an existential operator. In [3], the declarative semantics of a particular instance of Molog is developed in terms of trees, and certain transformations on trees. We will summarize some features of this language and show that it is contained in the theory.

In [3], first, a language with modal operators $\Box$ and $\Diamond$ is described. With $\Diamond$ disallowed in the heads and $\Box$ in the bodies of modal clauses, the preconditions of Theorem 6.1 are satisfied. Then in order to skolemize the possibility operator, the language is enriched with a countable set of modal operators. If $\langle \alpha \rangle$ is such an operator, then $\models_{I,w} \langle \alpha \rangle A$ iff $\models_{I, f\alpha(w)} A$, where $f_\alpha \in [\mathscr{U} \to \mathscr{U}]$. The denotations of those operators can be obtained as follows:

$$\|\langle \alpha \rangle\| = \lambda X.\{w \in \mathscr{U} \mid f_\alpha(w) \in X\}.$$

It can be verified that each $\|\langle \alpha \rangle\|$ has all the desired properties, therefore, we conclude that Theorem 6.1 applies to this particular language.

InTense, proposed by Mitchell and Faustini [19], supports any finite number of temporal and spatial dimensions. Thus a possible world in InTense is a point in a

time-space hyperfield. For instance, if we have m temporal and n spatial dimensions, the set of possible worlds $\mathscr{U}$ is $\mathscr{Z}^{m+n}$. InTense provides users with unary intensional operators for each dimension: $\texttt{prior}_k$, $\texttt{initial}_k$ and $\texttt{rest}_k$ are associated with the kth spatial dimensions; and similarly, $\texttt{prev}_k$, $\texttt{initial}_k$ and $\texttt{rest}_k$ are associated with the kth temporal dimension.

The semantics of temporal operators $\texttt{prev}_k$, $\texttt{first}_k$, and $\texttt{next}_k$ are similar to those of Chronolog with negative time, the only difference being that they operate on the kth temporal dimension. The semantics of spatial operators $\texttt{prior}_k$, $\texttt{initial}_k$ and $\texttt{rest}_k$ are counterparts of temporal operators, over the kth spatial dimension. Therefore all of the intensional operators of InTense share the desired properties, which in turn implies that *pure* InTense programs enjoy the minimum model semantics.

6.3. Interval Logic Programming

As for interval type temporal logic programming languages such as Tokio [2] and Tempura [21], our theory does not directly apply. These languages have features which go beyond pure logic programming. In fact, Tempura is not based on the Horn-clause subset of logic. As for Tokio, Aoyagi et al. [2] do not clearly specify the syntax of Tokio, but intensional operators of Tokio are in fact those of Tempura. In Tokio, even the variables are intensional, i.e., the meanings of variables depend on moments in time, and there are even temporal operators that can be applied to the terms of the language. However, the fact that the minimum model semantics does not apply to Tempura and especially Tokio does not mean that it cannot be applied to interval logic programming at all. As a matter of fact, it is known that an interval logic can be embedded into a two-dimensional logic by transformation [33]. This result suggests that, if all the constraints are satisfied, interval logic programming enjoys the minimum model semantics.

An interval is a pair of natural numbers $[x, y]$, where $x \leq y$. Let $Sub([x, y]) =_{def} \{[m, n] | x \leq m \leq n \leq y\}$, i.e., $Sub([x, y])$ is the set of all subintervals of $[x, y]$. In interval logic, the satisfaction relation can be defined over intervals and temporal interpretations. For instance, the semantics of interval operators $\Box$ and $\Diamond$ of Tempura are defined as

- $\models_{I,[m,n]} \Box A$ iff $\models_{I,[x,n]} A$ for all $[x, n] \in Sub([m, n])$
- $\models_{I,[m,n]} \Diamond A$ iff $\models_{I,[x,n]} A$ for some $[x, n] \in Sub([m, n])$

where I is a temporal interpretation and $[m, n]$ is an interval. The semantics of atomic formulas can be defined as

- $\models_{I,[x,y]} p(e_0, \ldots, e_{n-1})$ iff $x \in \| p(e_0, \ldots, e_{n-1}) \|^I$

Note that there are surely other ways of defining the semantics of atomic formulas.

Consider the intensional logic IL where $\mathscr{U} = \{\langle x, y \rangle \in \omega \times \omega | x \leq y\}$, and the intensional operators of IL are $\{\odot, \Box, \Diamond, \ldots\}$. The semantics of intensional operators are defined as in interval logic, but in terms of pairs in $\mathscr{U}$. The extra operator $\odot$ projects any given world onto the main diagonal in $\mathscr{U}$, i.e., $\models_{I,\langle x,y \rangle} \odot A$ iff $\models_{I,\langle x,x \rangle} A$. Here the world $\langle x, x \rangle$ can be interpreted as a moment in time, that is, x. We need the extra operator $\odot$, because the semantics of atomic formulas of IL

are defined for all pairs in $\mathscr{U}$. Then the transformation procedure from interval logic to this two-dimensional logic adds the new operator $\odot$ in front of every atomic formula A. This way, the semantics of A in terms of intervals coincides with the semantics of $\odot A$ in terms of pairs in $\mathscr{U}$. Moreover, IL inherits all the properties of operators in the interval logic as well as the function *Sub* with intervals interpreted as possible worlds.

The next step is to transform a given interval logic program $\mathscr{P}$ into this two-dimensional logic. Then we just check whether the transformed program satisfies the preconditions of Theorem 6.1. If so, the minimum intensional Herbrand model of the interval logic program may be constructed from that of the transformed program in IL, by carrying everything from the worlds along the main diagonal in $\mathscr{U}$ over to a model of $\mathscr{P}$, with worlds of the form $\langle x, x\rangle$ interpreted as moments in time.

If we restrict Tokio to extensional variables, and strip Tokio off all of its structures which go beyond our framework, it can be shown that the resulting interval language enjoys the minimum model semantics. We have that the denotations of all interval operators of Tokio are universal, monotonic, and finitary (all intervals are of finite length). On the other hand, $\|\Diamond\|$ and the chop operator $\|\&\&\|$ are not conjunctive, which in turn implies that $\Diamond$ and && can not be used in the heads of program clauses.

7. DEFINING INTENSIONAL OPERATORS

Recall that, in temporal logic, the formula $\wedge_{i \in n} \mathrm{next}^i A$ states that A is true "now and during the next $n-1$ moments." We can move to a *definitional extension* [29] of the underlying logic and use this formula as the definition of a new temporal operator, say $[n]$. This is fine, but, in temporal logic programming, we can do a better job and define $[n]$ on the fly by an intensional program clause.

The definition of $[n]$ in the form of a temporal program clause can be given as:

$$[n]A \leftarrow \bigwedge_{i \in n} \mathrm{next}^i A,$$

where A is just a propositional variable. Let $\mathscr{P}$ be a temporal logic program in which $[n]$ is applied to some temporal atom B in the body of some clause. To prove $[n]B$, the definition can be invoked after substituting A by B. Moreover, the occurrence of $[n]$ is not a direct application of $[n]$ to A; it is there to establish the connection between the use and the definition of $[n]$. In other words, the definition works as a metarule.

There is one important restriction: the symbol $[n]$ cannot appear in the body of any clause in the definition of $[n]$ or in the definitions of other new temporal operators. This way, we avoid direct or indirect recursion.

Theorem 6.1 does not extend to intensional logic programs with metarules, because in the object language, the meaning of the symbol $[n]$ is unknown. But we can formulate a model-theoretical condition for the denotation of $[n]$ by the following: for all $X \in P(\omega)$,

$$\|[n]\|(X) \supseteq \bigcap_{i \in n} \|\mathrm{next}\|^i(X).$$

There may be many functions in $[P(\omega) \to P(\omega)]$ which satisfy the condition. However, it can be shown that $\lambda X. \bigcup_{i \in n} \|\mathtt{next}\|^i(X)$ is the least such function. Then the definition of $[n]$ is a monotonic formula in one propositional variable, which in fact corresponds to the least function obtained from the definition.

In general, a new intensional operator ∇ can be defined by an intensional program clause by the following:

$$\nabla A \leftarrow \bigvee_{\alpha \in S} \bigwedge_{k \in m_\alpha} B_{\alpha,k},$$

where $S \in \omega$ and each $B_{\alpha,k}$ is a monotonic formula in the propositional variable A. We assume that ∇ or other defined operators do not appear in the definition of ∇. Most importantly, the constraints on intensional operators still apply.

Let $\mathscr{P}$ be an intensional logic program with intensional operator definitions. Then $\mathscr{P}$ is a metatheory of the underlying logic. We will describe a transformation procedure to obtain an equivalent intensional logic program from $\mathscr{P}$. We first define a syntactic translation function τ from formulas of a metatheory to formulas of IL.

Definition 7.1. Let τ be a syntactic translation function from formulas of a metatheory to formulas of IL defined as follows:

- $\tau(\nabla B) = \tau(\bigvee_{\alpha \in S} \bigwedge_{k \in m_\alpha} B_{\alpha,k}\{A/B\})$, where ∇ is a new intensional operator defined as $(\nabla A \leftarrow \bigvee_{\alpha \in S} \bigwedge_{k \in m_\alpha} B_{\alpha,k}) \in \mathscr{P}$.
- $\tau(\nabla B) = \nabla(\tau(B))$ where ∇ is an intensional operator of the underlying logic.
- $\tau(\bigvee_{\alpha \in S} B_\alpha) = \bigwedge_{\alpha \in S} \tau(B_\alpha)$ (similarly for $\vee$).
- $\tau(B) = B$ where B is an atomic formula.

Let $\mathscr{P}_c$ denote the set of intensional program clauses in $\mathscr{P}$. Then the transformed program $\mathscr{P}^\tau$ is defined as follows:

$$\mathscr{P}^\tau = \{(A \leftarrow \tau(B_0), \ldots, \tau(B_{n-1})) \mid (A \leftarrow B_0, \ldots, B_{n-1}) \in \mathscr{P}_c\}$$

All the model-theoretical results from the previous sections apply to $\mathscr{P}^\tau$, since the preconditions of Theorem 6.1 are met.

7.1. Recursive Definitions

Let us focus on the temporal logic of Chronolog and call the underlying temporal language TL. We will now enrich TL with two new temporal operators, $\Box$ and $\Diamond$ as follows. Read $\Box$ as "always" and $\Diamond$ as "sometime." A formula of the form $\Box A$ is true at time t in a temporal interpretation I just in case A is true at all moments in time; a formula of the form $\Diamond A$ is true at time t in I just in case A is true at some moment in time. In other words, $\Box$ and $\Diamond$ are just like (temporal) necessity and possibility operators. Then it can be shown that the following theorems hold.

- $\models \Box A \leftrightarrow \mathtt{first} A \wedge \Box(\mathtt{next} A)$
- $\models \Box A \leftrightarrow \mathtt{first} A \vee \Diamond(\mathtt{next} A)$

The theorems given above suggest that both $\Box$ and $\Diamond$ could be recursively defined within *TL*. However, it is not hard to see that such definitions actually lead to infinitary formulas in the object language (formulas with countable conjunctions and disjunctions). If recursion is allowed in defining new intensional operators, both $\Box$ and $\Diamond$ can be defined from the above theorems as metarules within temporal Horn logic as follows: $\Box A \leftarrow \texttt{first}A \wedge \Box(\texttt{next}A)$; and $\Diamond A \leftarrow \texttt{first}A \vee \Diamond(\texttt{next}A)$.

The methods described above are not directly applicable to metatheories of intensional Horn logic in the presence of recursively defined intensional operators. Barringer [4] employs a temporal fixpoint calculus for recursively defined temporal operators in the context of program specification and proof theories. Baudinet [6] shows the connections between the fixpoint semantics of temporal logic programming and temporal logic μTL of Vardi [36]. Here we will sketch a fixpoint approach to recursive definitions in intensional logic programming based on $L_{\omega_1\omega}$ [15]. Note that the theory developed so far can be extended to infinitary intensional logic programs in a straightforward manner. Now intensional logic programs may contain countably many program clauses.

A functional Υ is an element of $[[P(\mathscr{U}) \rightarrow P(\mathscr{U})] \rightarrow [P(\mathscr{U}) \rightarrow P(\mathscr{U})]]$. Continuity and monotonicity of functionals can be defined as usual, e.g., see [18]. We have the following theorem.

Theorem 7.1. Any functional Υ, when defined by unions, intersections and compositions of monotonic and finitary functions and the function variable Θ, is continuous.

A continuous functional Υ has a least fixpoint denoted by $lfp(\Upsilon)$. In fact, $lfp(\Upsilon) = \Upsilon \uparrow \omega$. Note that $\Upsilon \uparrow 0 = \lambda X.\varnothing$. Then the functions implied by recursive definitions given for $\Box$ and $\Diamond$ of *TL* are the least fixpoints of the following functionals $\Upsilon_\Box$ and $\Upsilon_\Diamond$:

$$\Upsilon_\Box(\Theta) = \lambda X.\|\texttt{first}\|(X) \cap \Theta \circ \|\texttt{next}\|(X)$$

$$\Upsilon_\Diamond(\Theta) = \lambda X.\|\texttt{first}\|(X) \cup \Theta \circ \|\texttt{next}\|(X).$$

Since $\|\Box\|$ is not continuous, we do not expect to obtain a discontinuous function from the least fixpoint of $\Upsilon_\Box$; indeed, $lfp(\Upsilon_\Box) = \lambda X.\varnothing$. As $\|\Diamond\|$ is continuous, the least fixpoint of $\Upsilon_\Diamond$ is exactly what we wanted, i.e., the least function implied by the definition of $\Diamond$.

$$lfp(\Upsilon_\Diamond) = \lambda X. \bigcup_{n \in \omega} \|\texttt{first}\| \circ \|\texttt{next}\|^n(X).$$

The greatest fixpoint of a continuous functional Υ also exists and is defined as $gfp(\Upsilon) = \Upsilon \downarrow \omega$ where $\Upsilon \downarrow 0 = \lambda X.\mathscr{U}$. Then it can be verified that the function $\|\Box\|$ is actually the greatest fixpoint of $\Upsilon_\Box$.

$$gfp(\Upsilon_\Box) = \lambda X. \bigcap_{n \in \omega} \|\texttt{first}\| \circ \|\texttt{next}\|^n(X).$$

However, under continuity restriction, the greatest fixpoint construction in temporal logic [4] can not be adapted to ILP, because we consider the use of such defined intensional operators only in the bodies of intensional program clauses.

Now the syntactic definition of $\Diamond$ can be obtained from the least fixpoint of $\Upsilon_\Diamond$ by just using the syntactic counterparts of functions, compositions, intersections,

and unions. For set variables, we introduce propositional variables. The nonrecursive definition of $\Diamond$ obtained from $lfp(\Upsilon_\Diamond)$ is given below:

$$\Diamond A \leftarrow \bigvee_{n \in \omega} \texttt{first}\,\texttt{next}^n A,$$

where A is a propositional variable.

If we use the greatest fixpoint techniques, the nonrecursive definition of $\Box$ can be formed from the greatest fixpoint of $\Upsilon_\Box$ by the following:

$$\Box A \leftarrow \bigwedge_{n \in \omega} \texttt{first}\,\texttt{next}^n A.$$

The body of the definition of $\Box$ has a countable conjunction. Although the body of the definition of $\Box$ is a monotonic formula, it cannot be regarded as the definition of an intensional operator as we did earlier, since Lemma 4.5 does not hold for logics based on $L_{\omega_1\omega}$. Consequently the continuity of $T_{\mathscr{P}}$ is lost.

Let $\mathscr{P}$ be a metatheory with recursive intensional operator definitions. We first modify $\mathscr{P}$ by replacing the bodies of the definitions of new intensional operators by the nonrecursive definitions obtained from the least fixpoints of the corresponding functionals. The modified $\mathscr{P}$ is surely a metatheory, but now the transformation procedure induced by the function τ can be applied to it. From the modified $\mathscr{P}$, we can obtain an intensional logic program $\mathscr{P}^\tau$ in the underlying language; Theorem 6.1 applies to $\mathscr{P}^\tau$.

In case of mutually recursive definitions, the fixpoint theory of functionals can be extended to cover a set of recursive definitions in a usual way, e.g., see [18]. However, we do not explore this subject any further.

8. CONCLUSIONS

We will first give a summary of the main results of this paper. In Section 3, we showed that temporal logic programs can model nonterminating computations and the notion of dynamic change naturally and without employing infinitary objects. In Section 4, we developed an intensional semantics based on algebraic and neighborhood semantics [20, 28] and defined several constraints on intensional operators. We also showed how monotonic formulas can be regarded as the definitions of new intensional operators. The major result of Section 6, Theorem 6.1, combines all the results obtained in Section 5, and it can be applied to a variety of ILP paradigms including Chronolog [37], Templog [1], instances of Molog [10], restricted versions of Tokio [2] and InTense [19]. We also showed how new intensional operators can be defined within (infinitary) intensional Horn logic.

In short, the objective of a language-independent unified theory for intensional logic programming is twofold:

- We investigate whether some intensional logic programming language is contained in the theory and enjoys the properties outlined in this paper.
- We use the theory as a template to *design* a new intensional logic programming language with the desired properties.

We do not claim that the theory developed so far is the ultimate way to go about intensional logic programming, but it clarifies how important the role of each property is.

However, this approach to the semantics of ILP is not complete. It lacks rules of inferences and therefore, we do not have the connections between model-theoretical and proof-theoretical semantics. In general, in order to be able to describe a formal proof procedure for an ILP language, we need to know intensional operators available in the language and particular rules of inference for those intensional operators. We conjecture that for "acceptable" languages, a complete proof procedure with respect to the minimum intensional Herbrand model can be devised, since a finite proof of a given query is guaranteed by the continuity restriction. Baudinet [5, 6] showed the completeness of Templog's proof procedure called TSLD-resolution. Chronolog is simpler than Templog, and therefore, TSLD-resolution can be adapted to a complete proof procedure for Chronolog. Rolston [26] is exploring efficient implementation techniques for Chronolog. Rolston [27] also investigated the potentials of Chronolog to mitigate the frame problem.

REFERENCES

1. Abadi, M., and Manna, Z., Temporal Logic Programming, in: *Proceedings of the 1987 Symposium on Logic Programming*, San Francisco, Calif., 1987, pp. 4–16.
2. Aoyagi, T., Fujita, M., and Moto-Oka, T., Temporal Logic Programming Language Tokio, in: E. Wada (ed.), *Logic Programing '85*, Springer-Verlag, New York, 1986, pp. 138–147.
3. Balbiani, P., Fariñas del Cerro, Luis, and Herzig, A., Declarative Semantics for Modal Logic Programs, in: *Proceedings of the 1988 International Conference on Fifth Generation Computer Systems*, Tokyo, Japan, 1988, pp. 507–514.
4. Barringer, H., The Use of Temporal Logic in the Compositional Specification of Concurrent Systems, in: A. Galton (ed.), *Temporal Logics and Their Applications*, Academic Press, San Diego, Calif., 1987, pp. 53–90.
5. Baudinet, M., On the Semantics of Temporal Logic Programming, Technical Report STAN-CS-88-1203, Computer Science Department, Stanford University, Calif., 1988.
6. Baudinet, M., Temporal Logic Programming is Complete and Expressive, in: *Conference Record of the Sixteenth ACM Symposium on Principles of Programming Languages*, Austin, Tex., January 1989, pp. 267–280.
7. Blair, H. A., et al., A Logic Programming Semantics Scheme, Part I, Technical Report LPRG-TR-88-8, Logic Programming Research Group, Syracuse University, 1988.
8. Bull, R., and Segerberg, K., Basic Modal Logic, in: D. M. Gabbay and F. Guethner (ed.), *Handbook of Philosophical Logic*, Vol. II, D. Reidel Publishing Company, 1984.
9. Chellas, B.F., *Modal Logic: An Introduction*, Cambridge University Press, England, 1980.
10. Fariñas del Cerro, Luis, MOLOG: A System That Extends Prolog with Modal Logic, *New Generation Comput.*, 4:35–50 (1986).
11. Fitting, M., Enumeration Operators and Modular Logic Programming, *J. Logic Programming* 4:11–21 (1987).
12. Fitting, M., Logic Programming on a Topological Bilattice, *Fundamenta Informaticae* XI:209–18 (1988).
13. Gabbay, D. M., Modal and Temporal Logic Programming, in: A Galton (ed.), *Temporal Logics and Their Applications*, Academic Press, San Diego, Calif., 1987, pp. 197–237.
14. Hughes, G. E., and Creswell, M. J., *An Introduction to Modal Logic*, Methuen, London, 1968.

15. Keisler, H. J., *Model Theory for Infinitary Logic*, North-Holland, Amsterdam, 1971.
16. Kowalski, R. A., *Logic for Problem Solving*, North-Holland, Amsterdam, 1979.
17. Lloyd, J. W., *Foundations of Logic Programming*, Springer-Verlag, New York, 1984.
18. Manna, Z., *Mathematical Theory of Computation*, McGraw-Hill, New York, 1974.
19. Mitchell, W. H., and Faustini, A. A., The Intensional Logic Language InTense, in: *Proceedings of the 1989 International Symposium on Lucid and Intensional Programming*, Arizona State University, Tempe, 1989, pp. 70–83.
20. Montague, R., *Formal Philosophy, Selected Papers of Richard Montague*, edited by Richmond Thomason, Yale University Press, New Haven, Conn., 1974.
21. Moszkowski, B., *Executing Temporal Logic Programs*, Cambridge University Press, England, 1986.
22. Orgun, M. A., and Wadge, W. W., CHRONOLOG: A Temporal Logic Programming Language and Its Formal Semantics, Department of Computer Science, University of Victoria, Canada, 1988.
23. Orgun, M. A., and Wadge, W. W., A Theoretical Basis for Intensional Logic Programming, in: *Proceedings of the 1988 International Symposium on Lucid and Intensional Programming*, Sidney, Canada, 1988, pp. 33–49.
24. Orgun, M. A., and Wadge, W. W., Towards a Unified Theory of Intensional Logic Programming, Technical Report DCS-112-IR, Department of Computer Science, University of Victoria, Canada, 1989.
25. Rescher, N., and Urquhart, A., *Temporal Logic*, Springer-Verlag, New York, 1971.
26. Rolston, D., CHRONOLOG: A Pure Tense-Logic-Based Infinite-Object Programming Language for Parallel Symbolic Execution, Ph.D. Dissertation Proposal, Arizona State University, Tempe, 1986.
27. Rolston, D., Toward a Tense-Logic-Based Mitigation of the Frame Problem, in: *Proceedings of the AAAI Workshop on the Logical Frame Problem*, Lawrence, Kans., 1987.
28. Scott, D., Advice on Modal Logic, in: K. Lambert (ed.), *Philosophical Problems in Logic*, D. Reidel Publishing Company, 1970, pp. 143–173.
29. Segerberg, K., *Classical Propositional Operators*, Oxford University Press, U.K., 1982.
30. Shapiro, E., A Subset of Concurrent Prolog and Its Interpreter, in: E. Shapiro (ed.), *Concurrent Prolog: Collected Papers*, MIT Press, Cambridge, Mass., 1987, pp. 27–83.
31. Stoy, J. E., *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*, MIT Press, Cambridge, Mass., 1977.
32. van Benthem, J. F. A. K., Possible Worlds Semantics: A Research Program That Cannot Fail? *Studia Logica* XLIII(4):379–393 (1984).
33. van Benthem, J. F. A. K., *A Manual of Intensional Logic*, 2nd ed., Center for the Study of Language and Information, Stanford University, 1988.
34. van Emden, M. H., and Kowalski, R. A., The Semantics of Predicate Logic as a Programming Language, *J. ACM* 23:733–742 (1976).
35. van Emden, M. H., and Nait Abdallah, M. A., Top-Down Semantics of Fair Computations of Logic Programs, Technical Report CS-84-27, Department of Computer Science, University of Waterloo, Canada, 1984.
36. Vardi, M., A Temporal Fixpoint Calculus, in: *Conference Record of the Sixteenth ACM Symposium on Principles of Programming Languages*, San Diego, Calif., 1988, pp. 250–259.
37. Wadge, W. W., Tense Logic Programming: A Sane Alternative, Department of Computer Science, University of Victoria, Canada, 1985.
38. Wadge, W. W., and Ashcroft, E. A., *Lucid, the Dataflow Programming Language*, Academic Press, San Diego, Calif., 1985.
39. Wojcicki, R., *Theory of Logical Calculi*, Kluwer Academic Publishers, 1988.
40. Yaghi, A. A., *An Intensional Implementation Technique for Functional Languages*, Ph.D. Dissertation, Department of Computer Science, University of Warwick, Coventry, England, 1984.