

A Formal Treatment of Non-Deterministic Dataflow Streams in Intensional Logic Programming

Mehmet A. Orgun and William W. Wadge

Department of Computer Science

University of Victoria

P.O.Box 1700, Victoria, B.C.

CANADA V8W 2Y2

March, 1989

Abstract

Intensional logic programming is an alternative to concurrent logic programming languages which employ infinitary data structures such as streams to specify non-terminating computations. Wadge has proposed a logical extension to intensional logic programming, i.e., choice predicates, through which dataflow-style of computations can be naturally expressed. This paper discusses model-theoretical semantics of intensional logic programs with choice predicates, and introduces the notion of constructible minimal models which characterises the operational semantics of such programs.

1 Introduction

Intensional logic programming (ILP) is an alternative to concurrent logic programming languages which employ infinitary data structures such as streams to specify non-terminating computations. Intensional logic [Mon74] studies formulas/expressions whose meaning depend on an implicit context (time or other.) Temporal logic programming languages such as Chronolog [Wad85,Rol87,OW88a,Wad88] and Templog [AM87,Bau88a,Bau88b] can be considered as instances of ILP, whose underlying logic incorporates time as the implicit context.

In ILP, a stream can be represented by a predicate over a set of possible worlds, such as a collection of moments in time. But ILP languages are non-deterministic in the sense that each predicate defined in an intensional logic program does not necessarily specify a unique stream of ground terms, because programmers are not allowed to make arbitrary but definite choices. Then a problem emerges in connection with dataflow style of (stream-oriented) communication such as that of the dataflow language Lucid [WA85].

Let us first consider stream-oriented input/output, and designate two unary predicates, `input` and `output`, as playing a special role in the underlying intensional language. The predicate `input` is supplied by the implementation, and should not appear in the heads of any clause in an intensional logic program. At any possible world w , there is only one ground instance of `input(X)` which succeeds – namely that in which X is instantiated to the corresponding input item indexed by w in the input stream of ground terms. In this way the `input` predicate represents a stream of input values over the set of possible worlds in the underlying intensional logic.

On the other hand, the `output` predicate is axiomatised by the programmer. In the simplest case, the program should ensure that at any given world w , there is exactly one ground instance of `output(X)` which succeeds. Then the unique ground term which X instantiates to is produced by the implementation as the output item indexed by w in the output stream of ground terms.

The problems arise when at a given world the `output` predicate does not specify a unique ground term. And there is no reason that it should – perhaps the problem specification itself does not constrain the programmer to produce one particular output stream. In that case it would be completely against the spirit of logic programming to force the programmer to complicate the program by making it unnecessarily specific. Clearly, the implementation should choose one of the possible instantiations as the output.

This is fine – except that the programmer may need to know what was actually output. For example, suppose that we are required to produce an increasing stream of natural numbers starting at 0. The following temporal logic program seems to work, but does not.

```
first output(0) ←
next output(Y) ← output(X), X < Y
```

What this really says is that if X might be the (say) 10th output, and X is less than Y , then Y might be the 11th output. What the above program specifies is any arbitrary stream whose time t component is greater than that of $t - 1$. What we need to say is that the time 11 *possible* outputs are those numbers which are greater than the time 10 *actual* output. This is not possible in pure intensional logic programming.

The solution is to provide an extra predicate `#output` which succeeds only for the ground term that was actually produced. Then the temporal logic program should

be

```
first output(0) ←  
next output(Y) ← #output(X), X < Y
```

Here the $\#output$ predicate is called a “choice predicate”. Then the $\#output$ predicate represents a non-deterministic stream of output values over the set of possible worlds in the underlying intensional logic. Of course the problem is more than just one of stream-oriented I/O. In a dataflow-structured program some parts are producers and others are consumers; thus every predicate is potentially an “output stream”. This means that we must supply choice predicates for every predicate symbol in an intensional logic program. Of course, the programmer is free to use any number of them, or none at all.

The relation between `output` and $\#output$ can be axiomatised only in a first-order logic with equality. Therefore the choice predicates represent a very controlled extension of ILP beyond intensional Horn logic. Answers to queries are no longer logical consequences of the program, because choice predicates represent (non-deterministic) streams which depend on arbitrary choices made by the implementation. Essentially, the implementation is constructing a *minimal* model of the program plus the implicit non-Horn axioms which establish the connection between predicates and corresponding choice predicates.

In the following, we will first define the notion of extended program, which includes non-Horn axioms for choice predicates, and then investigate the general properties of intensional Herbrand models of such programs. The declarative semantics of extended programs will be developed in terms of minimal intensional Herbrand models. We will show that some of these minimal models are not computable from the program, i.e., no implementation can construct them due to the logical structure of the program. In other words, computable minimal models characterise the operational semantics of extended programs. We will also relate our work to concurrent logic programming languages, that is, committed choice languages.

2 Extended Intensional Logic Programs

We build on the work of Orgun and Wadge [OW88b] which describes a language-independent model theory for intensional logic programs. There, an intensional logic programming language is identified with its underlying intensional logic, a countable set of possible worlds over which the values of formulas vary, and intensional operators that satisfy some restrictions to preserve the minimum model semantics. Let \mathcal{P} be an intensional logic program written in some language with choice predicates. Then we extend the underlying intensional language, say IL , with equality. We denote the set of possible worlds by \mathcal{U} . Note that we assume that \mathcal{U} is countable and an

enumeration of \mathcal{U} is given, i.e., $\mathcal{U} = \{w_0, w_1, \dots\}$. Semantics of the elements of IL can be defined in the usual way.

Choice Formulas and Extended Programs

We differentiate the choice predicates from the others. Let $Pred$ denote the set of all predicate symbols of IL other than the choice predicates. Then the choice predicate related to any $p \in Pred$ is denoted by $\#p$ and can be used only in the bodies of program clauses in intensional logic programs and in queries.

We now define non-Horn axioms for establishing the relation between predicates and choice predicates. Any intensional logic program in which the use of some choice predicates appear in the bodies of clauses implicitly includes these axioms.

Definition 2.1 (Choice formulas.) *Let $p \in Pred$ be any n -ary predicate symbol. The following formula establishes the connection between p and $\#p$: let $\vec{X} =_{def} \langle X_0, \dots, X_{n-1} \rangle$ and $\vec{Y} =_{def} \langle Y_0, \dots, Y_{n-1} \rangle$.*

$$(\forall \vec{X})(\#p(\vec{X}) \leftrightarrow (p(\vec{X}) \wedge (\forall \vec{Y})(\#p(\vec{Y}) \rightarrow p(\vec{Y}) \wedge (\bigwedge_{i \in n} X_i \doteq Y_i))))$$

We use \doteq for equality symbol in the object language in order to distinguish it from the equality symbol $=$ in the meta-language. Read choice formulas as true assertions at all worlds in \mathcal{U} . Intuitively, the choice formula for p says that at any world w , whenever p is true of some ground term, $\#p$ is true of a unique ground term chosen from those terms for which p is true of at w ; and if p is not true of any ground term, neither is $\#p$.

Then an intensional logic program \mathcal{P} written in IL and extended with choice formulas given above is called an extended (intensional logic) program. The intensional Herbrand base $B_{\mathcal{P}}$ of an extended intensional logic program \mathcal{P} now includes all those ground atomic formulas obtained from choice predicates and ground terms in the Herbrand universe $U_{\mathcal{P}}$ of \mathcal{P} as well as atomic formulas obtained from non-choice predicates. An intensional Herbrand interpretation I of \mathcal{P} assigns a subset of the set of possible worlds \mathcal{U} to each ground atom A in $B_{\mathcal{P}}$, i.e., $\|A\|^I \in P(\mathcal{U})$ where $\|A\|^I$ is the denotation of A in I . Here $P(\mathcal{U})$ is the power set of \mathcal{U} , that is, the family of all subsets of \mathcal{U} .

We treat \doteq in a different way and assume that the denotation of \doteq in *any* intensional (Herbrand) interpretation is a reflexive binary relation over the Herbrand universe of a given program, i.e., $\|\doteq\| = \{\langle t, t \rangle \mid t \in U_{\mathcal{P}}\}$. Therefore, technically, there is no need to include those ground atoms related to the equality in the intensional Herbrand base of any extended program.

Properties of Models of Extended Programs

Here we will formulate some model-theoretical conditions for intensional Herbrand interpretations of an extended program to be a model. This way, we do not have to refer to the syntactic properties and structure of extended programs.

Lemma 2.1 *Let \mathcal{P} be an extended intensional logic program and \mathcal{P}_C be the set of choice formulas for \mathcal{P} . Suppose I is an intensional Herbrand interpretation of \mathcal{P} . Then the following are equivalent.*

- (a) $\models_I \mathcal{P}_C$ (\mathcal{P}_C is true in I at all worlds in \mathcal{U} .)
- (b) For all predicate symbols $p \in \text{Pred}$, let n be the arity of p ; then
 - C1: $\|p(\vec{e})\|^I \subseteq \|p(\vec{e})\|^I$ for all $\vec{e} \in (U_{\mathcal{P}})^n$,
 - C2: for all \vec{e} and $\vec{t} \in (U_{\mathcal{P}})^n$, $\vec{e} \neq \vec{t}$ implies $\|p(\vec{e})\|^I \cap \|p(\vec{t})\|^I = \emptyset$,
 - C3: $\bigcup_{\vec{e} \in (U_{\mathcal{P}})^n} \|p(\vec{e})\|^I = \bigcup_{\vec{e} \in (U_{\mathcal{P}})^n} \|p(\vec{e})\|^I$.

We will explain the intuitive meanings of these conditions. C1 states that choice predicates may be only true of those ground terms which the corresponding predicate is true of. C2 merely asserts the uniqueness of the ground term which any choice predicate is true of. C3 guarantees that whenever a predicate is true of some ground term at some world w , the corresponding choice predicate must be true of either the same ground term or a possibly different ground term at w . In short, C1, C2 and C3 are the necessary and sufficient conditions for an intensional Herbrand interpretation to be a model of the choice formulas.

The intensional Herbrand interpretation that corresponds to the entire Herbrand base of an ordinary logic program – namely the one that assigns \mathcal{U} to every atom in $B_{\mathcal{P}}$ – is in general not a model of \mathcal{P} , because it may fail to satisfy some of the choice formulas whenever a predicate is true of more than one ground term at a world. Then the question is whether an extended logic program \mathcal{P} is consistent or not. The following lemma gives an affirmative answer.

Lemma 2.2 *Let \mathcal{P} be an extended intensional logic program. Then \mathcal{P} has a model, that is, $\models_I \mathcal{P}$ for some intensional Herbrand interpretation I .*

Proof. Let $B_{\mathcal{P}}$ be the intensional Herbrand base of \mathcal{P} and t be an arbitrary term in the Herbrand universe of \mathcal{P} . We define I as follows: if $p \in \text{Pred}$ is an n -ary predicate symbol, then $\|p(\vec{e})\|^I = \mathcal{U}$ for all $p(\vec{e}) \in B_{\mathcal{P}}$, and $\|p(\vec{t})\|^I = \mathcal{U}$ where \vec{t} denotes an n -tuple of terms, each of whose components is t . The semantics of \models in I is defined as before. Then it can be verified that I is a model of the intensional Horn clauses in \mathcal{P} . In order to show that all choice formulas in \mathcal{P} are also true in I , it suffices to check whether I satisfies C1, C2, and C3. We omit the details. \dashv

3 Minimal Models of Extended Programs

The family of intensional Herbrand models of an extended intensional logic program \mathcal{P} is in fact a partially ordered set. The actual ordering relation is defined as follows: Let I and J be any two intensional Herbrand interpretation of \mathcal{P} . Then we say that $I \sqsubseteq J$ iff for all $A \in B_{\mathcal{P}}$, $\|A\|^I \subseteq \|A\|^J$. Note that here we ignore the denotation of \doteq , since it is the same reflexive relation in any intensional Herbrand interpretation of \mathcal{P} .

We now define the model \sqcap -intersection operation which is analogous to set intersection. Let \mathcal{P} be an extended program and $M = \{M_\alpha\}_{\alpha \in S}$ be a family of intensional Herbrand interpretations of \mathcal{P} . Then $\sqcap M =_{\text{def}} \sqcap_{\alpha \in S} M_\alpha$ is an intensional Herbrand interpretation of \mathcal{P} defined as follows: for all $A \in B_{\mathcal{P}}$, $\|A\|^{\sqcap M} = \bigcap_{\alpha \in S} \|A\|^{M_\alpha}$. Similarly, we can define the model \sqcup -union operation.

Because of the non-deterministic choices involved in constructing the denotations of choice predicates, extended programs do not enjoy the minimum model semantics and fixpoint semantics of intensional logic programs developed by Orgun and Wadge [OW88b]. Nevertheless, as far as the declarative semantics of extended programs are concerned, we can prove a weaker but important result which states that any downwards chain of intensional Herbrand models of an extended program is bounded below by the model \sqcap -intersection of all the models in the chain, and therefore the family of intensional Herbrand models of the program contains minimal elements.

Lemma 3.1 *Let \mathcal{P} be an extended intensional logic program and $M = \langle M_\alpha \rangle_{\alpha \in S}$ be a nonempty downwards chain of intensional Herbrand models of \mathcal{P} , ordered by \sqsubseteq . Then $\sqcap M =_{\text{def}} \sqcap_{\alpha \in S} M_\alpha$ is also a model of \mathcal{P} .*

Proof. We will give the outline. Orgun and Wadge [OW88b] has shown that the model \sqcap -intersection property holds for intensional Horn clauses. Then it suffices to show that, over a downwards chain of models of \mathcal{P} , model \sqcap -intersection preserves all of C1, C2, and C3, the necessary and sufficient conditions for an intensional Herbrand interpretation to be a model. We omit the details. \dashv

The following lemma is also needed.

Lemma 3.2 (The dual of Zorn's lemma.) *Let X be a nonempty family of sets which are closed under intersections of nonempty downwards chains; then X has a minimal element.*

The following theorem, which is the major result of our model-theory, is a consequence of lemmas 3.1 and 3.2.

Theorem 3.3 *Let \mathcal{P} be an extended intensional logic program and $\mathcal{M}(\mathcal{P})$ be the family of all intensional Herbrand models of \mathcal{P} . Then $\mathcal{M}(\mathcal{P})$ has a minimal element.*

Proof. The theorem follows from lemmas 3.1 and 3.2. \dashv

In summary, every downwards chain of models in the family of models of an extended program \mathcal{P} has a lower bound in the family with respect to \sqcap , and thus the family of models of \mathcal{P} has minimal elements by the dual of Zorn's lemma. These minimal elements (models) constitute the model-theoretical semantics of extended programs. As mentioned earlier, any implementation of intensional logic programming that supports choice predicates must construct one such minimal model of an extended program.

4 Constructible Minimal Models

The logical structure of an extended program dictates which minimal intensional Herbrand models an implementation can construct. We illustrate this by a naive example. Consider the following program: $\mathcal{P} = \{ p(a) \leftarrow, p(b) \leftarrow \#p(X) \}$ and the query “ $? - \#p(X)$ ”. There are many minimal models of \mathcal{P} (here we consider \mathcal{P} as extended with choice formulas, since no confusion may arise.) But it can be that any implementation can only construct exactly *one* and the same minimal model, i.e., the one that assigns \mathcal{U} to all of $p(a)$, $p(b)$, and $\#p(a)$; and \emptyset to $\#p(b)$. Then the answer to the query at any world is the ground atom $\#p(a)$. In the following, we will formalise the notion of constructible minimal models by modifying and extending the fixpoint semantics of intensional logic programs [OW88b] which generalises that of ordinary logic programming [vEK76].

The Mappings $T_{\mathcal{P}}$ and $C_{\mathcal{P}}$

Let \mathcal{P} be an extended intensional logic program. Then the family of all intensional Herbrand interpretations of \mathcal{P} , denoted by $\mathcal{F}(\mathcal{P})$, is a complete lattice ordered by \sqsubseteq . Let I_0 be an intensional Herbrand interpretation of \mathcal{P} defined as follows: for all $p \in \text{Pred}$ and for all $\vec{e} \in (U_{\mathcal{P}})^n$,

$$\|p(\vec{e})\|^{I_0} = \|\#p(\vec{e})\|^{I_0} = \emptyset \quad \text{and} \quad \|\dot{=}\|^{I_0} = \{ \langle e, e \rangle \mid e \in U_{\mathcal{P}} \}$$

Then I_0 is the greatest lower bound of the family of all intensional Herbrand interpretations of \mathcal{P} . It can be shown that I_0 is a model of choice formulas of \mathcal{P} .

We now define the mapping $T_{\mathcal{P}}$ which is the one step modus ponens operation for intensional clauses in an extended program.

Definition 4.1 *Let \mathcal{P} be an extended intensional logic program, and H be an intensional Herbrand interpretation of \mathcal{P} . Then the mapping $T_{\mathcal{P}} \in [\mathcal{F}(\mathcal{P}) \rightarrow \mathcal{F}(\mathcal{P})]$ is*

defined as follows. $T_{\mathcal{P}}(H)$ is an intensional Herbrand interpretation given below: for all $w \in \mathcal{U}$,

$$\begin{aligned} \|p(\vec{e})\|^{T_{\mathcal{P}}(H)} &= \|p(\vec{e})\|^H \text{ for all } \|p(\vec{e})\| \in B_p \text{ and} \\ w \in \|A\|^{T_{\mathcal{P}}(H)} &\text{ iff } w \in \|B_i\|^H \text{ for all } i \in n \end{aligned}$$

where $A \leftarrow B_0, \dots, B_{n-1}$ is a ground instance of an intensional clause in \mathcal{P} .

$T_{\mathcal{P}}$ does not cover the choice formulas of \mathcal{P} , therefore we define another mapping $C_{\mathcal{P}}$ as follows. Let H be an intensional Herbrand interpretation of \mathcal{P} . For all $p \in \text{Pred}$ and for all $w \in \mathcal{U}$, define $E_{p,w}(H)$ and $E_{\#p,w}(H)$ as:

$$E_{p,w}(H) = \{\vec{e} \mid w \in \|p(\vec{e})\|^H\}$$

$$E_{\#p,w}(H) = \{\vec{e} \mid w \in \|\#p(\vec{e})\|^H\}$$

In other words, $E_{p,w}(H)$ is the set of ground terms which p is true of at world w ; and similarly $E_{\#p,w}(H)$ is the set of ground terms which $\#p$ is true of at world w . Let $S = \{ \langle p, w \rangle \mid E_{p,w}(H) \neq \emptyset \text{ and } E_{\#p,w}(H) = \emptyset \}$, i.e., for any $\langle p, w \rangle \in S$, no choice has been made for p at world w . Let $Cset(H) = \prod_{\langle p, w \rangle \in S} E_{p,w}(H)$; then any member of $Cset(H)$ represents some arbitrary choice function, which, given any $\langle p, w \rangle \in S$, returns an element of $U_{\mathcal{P}}$ chosen from $E_{p,w}$. We now give the formal definition of $C_{\mathcal{P}}$ operation.

Definition 4.2 Let \mathcal{P} be an extended intensional logic program and H be an intensional Herbrand interpretation of \mathcal{P} . Then the mapping $C_{\mathcal{P}} \in [\mathcal{F}(\mathcal{P}) \rightarrow P(\mathcal{F}(\mathcal{P}))]$ is defined as follows: $H_{\alpha} \in C_{\mathcal{P}}(H)$ iff for all $p \in \text{Pred}$, and for all $\vec{e} \in (U_{\mathcal{P}})^n$,

$$\|p(\vec{e})\|^{H_{\alpha}} = \|p(\vec{e})\|^H \text{ and}$$

$$\|\#p(\vec{e})\|^{H_{\alpha}} = \|\#p(\vec{e})\|^H \cup \{w \mid \vec{e} = \alpha(p, w)\}$$

where $\alpha \in Cset(H)$.

The $C_{\mathcal{P}}$ operation returns all possible immediate extensions of a given intensional Herbrand interpretation of \mathcal{P} , determined by arbitrary choice functions. The interpretations in $C_{\mathcal{P}}(H)$ for any given H are indexed by α 's in $Cset(H)$.

Alternating Chains of Models

We first give the standard definition of the upward closure of $T_{\mathcal{P}}$ operation with respect to any intensional Herbrand interpretation of an extended program \mathcal{P} as follows: $T_{\mathcal{P}} \uparrow \omega (I) =_{def} \sqcup \{T_{\mathcal{P}}^n(I) \mid n \in \omega\}$ where I is an intensional Herbrand interpretation of \mathcal{P} .

$T_{\mathcal{P}}$ and $C_{\mathcal{P}}$ interact nicely: we start from I_0 and alternatively apply $T_{\mathcal{P}} \uparrow \omega$ and $C_{\mathcal{P}}$, which in turn results in an alternating chain of interpretations of \mathcal{P} . Clearly, there are many such chains because of the arbitrary choices made by $C_{\mathcal{P}}$ operations. We will show that the limits (upper bounds) of these alternating chains are in fact the constructible minimal models of \mathcal{P} .

We will formalise the notion of alternating chains of interpretations of an extended program \mathcal{P} originating from I_0 by the following.

Definition 4.3 *Let \mathcal{P} be an extended intensional logic program, and \mathcal{R}_ω be a binary relation between intensional Herbrand interpretations of \mathcal{P} defined below inductively.*

$$\begin{aligned}\mathcal{R}_0 &= \{ \langle I_0, T_{\mathcal{P}} \uparrow \omega (I_0) \rangle \} \\ \mathcal{R}_i &= \{ \langle I, T_{\mathcal{P}} \uparrow \omega (I) \rangle \mid \text{for some } J, \langle J, I \rangle \in \mathcal{R}_{i-1} \} \quad i > 0 \text{ and even} \\ \mathcal{R}_i &= \{ \langle I, I_\alpha \in C_{\mathcal{P}}(I) \rangle \mid \text{for some } J, \langle J, I \rangle \in \mathcal{R}_{i-1} \} \quad i > 0 \text{ and odd} \\ \mathcal{R}_\omega &= \bigcup_{i \in \omega} \mathcal{R}_i\end{aligned}$$

In fact, all Herbrand interpretations of \mathcal{P} in the domain of \mathcal{R}_ω , denoted by $|\mathcal{R}_\omega|$, constitute a partially ordered set denoted by $(|\mathcal{R}_\omega|, \sqsubseteq)$ whose smallest element is I_0 . There are alternating (upwards) ω -chains in $(|\mathcal{R}_\omega|, \sqsubseteq)$, all of which start from I_0 . We will show that the least upper bounds of these ω -chains characterise the constructible minimal models of \mathcal{P} . Let $\mathcal{C} = \langle I_\alpha \rangle_{\alpha \in S}$ be an ω -chain as such. Observe that for any adjacent pair of interpretations (I_α, I_β) in \mathcal{C} , either $I_\beta = T_{\mathcal{P}} \uparrow \omega (I_\alpha)$ or $I_\beta \in C_{\mathcal{P}}(I_\alpha)$; in other words, for some $i \in \omega$, $\langle I_\alpha, I_\beta \rangle \in \mathcal{R}_i$.

The following lemma shows that, when applied in an alternating ω -chain of interpretations starting from I_0 , $C_{\mathcal{P}}$ operation always returns a family of models of the choice formulas.

Lemma 4.1 *Let \mathcal{P} be an extended intensional logic program; and $\langle I_\alpha, I_\beta \rangle \in \mathcal{R}_\omega$ where $I_\beta \in C_{\mathcal{P}}(I_\alpha)$. Let \mathcal{P}_C be the set of choice formulas for all $p \in \text{Pred}$. Then $\models_{I_\beta} \mathcal{P}_C$.*

The following lemma shows that, when applied in an alternating ω -chain of interpretations starting from I_0 , the upward closure of $T_{\mathcal{P}}$ operation always returns a model of the program clauses in \mathcal{P} , which is an extension of the given interpretation.

Lemma 4.2 *Let \mathcal{P} be an extended intensional logic program, and \mathcal{P}_H be the set of intensional Horn clauses in \mathcal{P} . Let $\langle I_\alpha, T_{\mathcal{P}} \uparrow \omega (I_\alpha) \rangle \in \mathcal{R}_\omega$. Then $\models_{T_{\mathcal{P}} \uparrow \omega (I_\alpha)} \mathcal{P}_H$.*

Lemmas 4.1 and 4.2 fail when we consider arbitrary chains of intensional Herbrand interpretations of \mathcal{P} . Any ω -chain \mathcal{C} in $|\mathcal{R}_\omega|$ starts from I_0 and each element in the chain is obtained from the previous one by either $T_{\mathcal{P}} \uparrow \omega$ or $C_{\mathcal{P}}$ operation. Lemmas 4.1 and 4.2 ensure that \mathcal{C} is in fact an alternating chain of interpretations of \mathcal{P} whose

elements are either models of intensional program clauses (those obtained by $T_{\mathcal{P}} \uparrow \omega$ operations) or models of choice formulas (those obtained by $C_{\mathcal{P}}$ operations).

$T_{\mathcal{P}}$ operation has another property that upward closure of $T_{\mathcal{P}}$ ($T_{\mathcal{P}} \uparrow \omega$) produces an intensional Herbrand interpretation which satisfies both C1 and C2.

Lemma 4.3 *Let \mathcal{P} be an extended intensional logic program; $\langle I_{\alpha}, I_{\beta} \rangle \in \mathcal{R}_{\omega}$ and $I_{\beta} = T_{\mathcal{P}} \uparrow \omega (I_{\alpha})$. Then I_{β} satisfies C1 and C2.*

Below we give another major result of the theory of extended intensional logic programs, which asserts that the least upper bounds of ω -chains in the domain of \mathcal{R}_{ω} are models of extended programs. In other words, if we start from I_{\emptyset} , consecutive applications of $T_{\mathcal{P}} \uparrow \omega$ and $C_{\mathcal{P}}$ will eventually produce a model of \mathcal{P} , in fact a minimal one. Therefore we conclude that constructible minimal models characterise the operational semantics of extended programs.

Theorem 4.4 *Let \mathcal{P} be an extended intensional logic program; \mathcal{C} be an ω -chain in $(|\mathcal{R}_{\omega}|, \sqsubseteq)$ and $\sqcup \mathcal{C} = \sqcup_{\alpha \in S} I_{\alpha}$ be the least upper bound of \mathcal{C} . Then $\models_{\sqcup \mathcal{C}} \mathcal{P}$.*

Proof. $\sqcup \mathcal{C}$ is clearly an intensional Herbrand interpretation of \mathcal{P} . Suppose $\sqcup \mathcal{C}$ is not a model of intensional Horn clauses in \mathcal{P} . Then there exists a ground instance of some clause in \mathcal{P} which is false in $\sqcup \mathcal{C}$ at some $w \in \mathcal{U}$. Let $A \leftarrow B_0, \dots, B_{n-1}$ be such an instance. Then $w \in \|B_i\|^{\sqcup \mathcal{C}}$ for all $i \in n$, but $w \notin \|A\|^{\sqcup \mathcal{C}}$. This means for all $i \in n$, $w \in \|B_i\|^{I_{\alpha}}$ for some $I_{\alpha} \in \mathcal{C}$. Since \mathcal{C} is a chain of models, for some I_{β} , $w \in \|B_i\|^{I_{\beta}}$ for all $i \in n$. That I_{β} is not a model of intensional Horn clauses in \mathcal{P} implies $I_{\beta} \in \mathcal{C}_{\mathcal{P}}(I)$ for some I . Then $T_{\mathcal{P}} \uparrow \omega (I_{\beta}) \in \mathcal{C}$ and $T_{\mathcal{P}} \uparrow \omega (I_{\beta})$ is a model of intensional Horn clauses in \mathcal{P} by lemma 4.2. Hence $w \in \|A\|^{T_{\mathcal{P}} \uparrow \omega (I_{\beta})}$, which means $w \in \|A\|^{\sqcup \mathcal{C}}$, a clear contradiction.

To complete the proof, it suffices to show that $\sqcup \mathcal{C}$ is also a model of the choice formulas, i.e., $\sqcup \mathcal{C}$ satisfies all of C1, C2, and C3. By lemmas 4.1 and 4.3, C1 and C2 are already satisfied by all the interpretations in \mathcal{C} . We omit the details. \dashv

Corollary 4.5 *Let \mathcal{P} be an extended intensional logic program, \mathcal{C} be an ω -chain in $(|\mathcal{R}_{\omega}|, \sqsubseteq)$ and $\sqcup \mathcal{C} = \sqcup_{\alpha \in S} I_{\alpha}$ be the least upper bound of \mathcal{C} . Then $\sqcup \mathcal{C}$ is a minimal model of \mathcal{P} , that is, for any intensional Herbrand interpretation I of \mathcal{P} , $\models_I \mathcal{P}$ and $I \sqsubseteq \sqcup \mathcal{C}$ implies $I = \sqcup \mathcal{C}$.*

Proof. By theorem 4.4, $\sqcup \mathcal{C}$ is a model of \mathcal{P} . Suppose it is not minimal. Since there is no minimality condition attached to choice predicates, we can safely consider only those ground atoms related to non-choice predicates. In other words, for any $w \in \mathcal{U}$ and any $p(\vec{e}) \in B_{\mathcal{P}}$, $w \in \|p(\vec{e})\|^{I_{\alpha}}$ for some $I_{\alpha} \in \mathcal{C}$ implies $w \in \|p(\vec{e})\|^{I_{\beta}}$ for some $I_{\beta} \in \mathcal{C}_{\mathcal{P}}(I)$ where $\beta \in \alpha^I$ and $I \sqsubseteq I_{\beta} \sqsubseteq I_{\alpha} \in \mathcal{C}$. Suppose that $w \in \|p(\vec{e})\|^{\sqcup \mathcal{C}}$, but this is not implied by any ground instance of any clause in \mathcal{P} . It must be the case that

for some $I_\alpha \in \mathcal{C}$, $w \in \|p(\vec{e})\|^{I_\alpha}$ where $w \notin \|p(\vec{e})\|^{I_\beta}$ for all $I_\beta \sqsubset I_\alpha$ in \mathcal{C} . But then for some $I_\gamma, < I_\alpha, I_\gamma > \in \mathcal{C}$ and $I_\gamma = T_{\mathcal{P}} \uparrow \omega(I_\alpha)$, so that $w \notin \|p(\vec{e})\|^{I_\gamma}$. This leads to the contradiction that \mathcal{C} can not be a chain. Thus $\sqcup \mathcal{C}$ must be minimal. \dashv

Operationally Deterministic Programs

Some extended programs, such as the one given in the beginning of this section, are deterministic in the sense that they operationally specify a single minimal model. Then any implementation can only construct that minimal model.

Definition 4.4 *Let \mathcal{P} be an extended intensional logic program. We say that \mathcal{P} is operationally deterministic iff $\text{card}(M) = 1$ where $M = \{\sqcup \mathcal{C} \mid \mathcal{C} \text{ is an } \omega\text{-chain in } (\mathcal{R}_\omega, \sqsubseteq)\}$.*

The properties of operationally deterministic programs are worth investigating in the future. But we still can not recover the notion of logical consequence, since it is basically model-theoretical.

5 A Comparison With Committed-Choice Languages

At the first glance, the idea of choice predicates seems to be a variant of committed-choices introduced in concurrent logic programming languages such as Parlog [CG86], Concurrent Prolog [Sha87], and GHC [Ued85], but it is not so. Up to now, proposed approaches to declarative semantics of concurrent logic programming languages, such as the greatest fixpoint semantics [vEdLF82, vENA84, Mur88] etc., suffer in that they employ infinite data structures such as streams; they also fail to capture the idea of committed-choices, and furthermore lack the notion of logical consequence. Besides, we claim that the pure declarative reading of a concurrent logic program is not necessarily what the programmer intends to specify.

For instance, suppose we have only two alternative clauses for predicate `use` in the following innocent-looking concurrent logic program, which say that some non-shared resource `r` is used by either process `a` or process `b`,

```
...
use(a,r) ← ... | ...
use(b,r) ← ... | ...
```

which is fine, since the programmer *knows* that given a query like “?- use(`P`, `r`)” any implementation will only commit to at most one of these two clauses and produce the corresponding ground instance of the query. On the other hand, with respect to the greatest fixpoint semantics of the program, both of the ground instances `use(a,r)`

and $\text{use}(b, r)$ may be true. Furthermore, since committed-choice non-determinism can not guarantee the uniqueness of the ground term which the use predicate is true of, a query like “ $?- \text{use}(P1, r), \text{use}(P2, r)$ ” may result in ground instances like “ $\text{use}(a, r), \text{use}(b, r)$ ” which is certainly not what the programmer has in mind.

Intensional logic programming offers a logical alternative to infinitary versions of concurrent logic programming and has well-defined minimum model semantics. Intensional logic programming with choice predicates actually solves the mutual exclusion problem faced in the above program. We just use, in the queries, choice predicates provided for each predicate symbol. Then at any given world, the answer to the query “ $?- \# \text{use}(P, r)$ ” is exclusively either $\# \text{use}(a, r)$ or $\# \text{use}(b, r)$. Similarly, the answer to the query “ $?- \# \text{use}(P1, r), \# \text{use}(P2, r)$ ” in any implementation is either the ground instance “ $?- \# \text{use}(a, r), \# \text{use}(a, r)$ ” or the ground instance “ $?- \# \text{use}(b, r), \# \text{use}(b, r)$ ”, which is also reflected in the minimal model semantics. In summary, choice predicates represent what is actual whereas non-choice predicates specify what is possible.

Hewitt and Agha [HA88] argue that proof theory does not provide a good model for the semantics of committed-choice languages due to the arrival-order assumptions of the messages which are not implied by the logical reading of concurrent logic programs. Intensional logic programs with choice predicates can mitigate this problem, because semantics of such programs are defined in terms of minimal models where each choice predicate represents a non-deterministic stream in any minimal model of the program. Then it remains to show that proof theoretical semantics of intensional logic programs is equivalent to constructing one such minimal model, indeed, a constructible one. But classical proof theory does not directly apply to ILP languages with choice predicates and thus much work has to be done.

6 Concluding Remarks

We have developed a model-theory for intensional logic programming with choice predicates, which extends that of [OW88b], and shown that any implementation can only construct some minimal model of an extended intensional logic program because of arbitrary but definite choices involved. Choice predicates can be used to model non-deterministic behaviour through the use of non-deterministic dataflow streams and within an intensional logic programming language such as temporal languages Chronolog [Wad85, Rol87, Wad88, OW88a], and Templog [AM87, Bau88a, Bau88b], both of which enjoy the model-theoretical semantics provided by [OW88b]. Applicability of our (minimal) model-theory to other intensional logic programming paradigms, such as Tokio [AFMo86], Temporal Prolog [Gab87], and Molog [Far86] will be investigated in the future.

The relation between ILP with choice predicates and committed-choice languages should be investigated further. We believe that ILP is already a plausible alterna-

tive to infinitary logic programming which supports streams and other infinite data structures. ILP with choice predicates provides dataflow-style of (stream-oriented) communication within a more powerful logic, and without any non-logical features.

References

- [AFMo86] Aoyagi, T., Fujita, M. and Moto-oka, T. ; "Temporal Logic Programming Language Tokio," in *Logic Programming'85*, E. Wada (ed.), LNCS 221, pp.138-147, Springer-Verlag, 1986.
- [AM87] Abadi, M. and Manna, Z. ; "Temporal Logic Programming," *Proceedings of the 1987 Symposium on Logic Programming*, Aug.31-Sept.4 1987, San Fransisco, CA, pp.4-16.
- [Bau88a] Baudinet, M. ; "On the Semantics of Temporal Logic Programming," Technical Report STAN-CS-88-1203, Computer Science Dept., Stanford University, Stanford, CA, June 1988.
- [Bau88b] Baudinet, M. ; "Temporal Logic Programming is Complete and Expressive," in the *Conference Record of the Sixteenth ACM Symposium on Principles of Programming Languages*, Austin, Texas, January 1989.
- [CG86] Clark, K. and Gregory, S. ; "PARLOG: Parallel Programming in Logic," ACM Trans. on Programming Languages and Systems, Vol. 8, No. 1, January 1986, pp. 1-49.
- [Far86] Fariñas Del Cerro, L. ; "MOLOG: A System that Extends Prolog with Modal Logic," *New Generation Computing*, 4(1986):35-50.
- [Gab87] Gabbay, D. ; "Modal and Temporal Logic Programming," in *Temporal Logics and Their Applications*, A. Galton (ed.), Academic Press, 1987.
- [HA88] Hewitt, C. and Agha, G. ; "Guarded Horn Clause Languages: Are They Deductive and Logical?" in the *Proceedings of the International Conference on Fifth Generation Computer Systems 1988*, edited by ICOT.
- [Mon74] Montague, R. ; *Formal Philosophy*, Selected Papers of Richard Montague, Richmond Thomason (ed.), Yale University Press, 1974.
- [Mur88] Murakami, M. ; "A Declarative Semantics of Parallel Logic Programs with Perpetual Processes," in the *Proceedings of the International Conference on Fifth Generation Computer Systems 1988*, edited by ICOT.

- [OW88a] Orgun, M.A. and Wadge, W.W. ; "Chronolog : A Temporal Logic Programming Language and Its Formal Semantics," unpublished manuscript, January 1988.
- [OW88b] Orgun, M.A. and Wadge, W.W. ; "A Theoretical Basis for Intensional Logic Programming," in the *Proceedings of the 1988 International Symposium on Lucid and Intensional Programming*, pp. 33-49, Sidney, B.C., April 7-8, 1988.
- [Rol87] Rolston, D. ; "Toward A Tense-Logic-Based Mitigation of the Frame Problem," in the *Proceedings of the AAAI Workshop on the Logical Frame Problem*, Lawrence, Kansas, April 1987, Morgan & Kaufman, Palo Alto, CA.
- [Sha87] Shapiro, E. ; "A Subset of Concurrent Prolog and Its Interpreter," in *Concurrent Prolog: Collected Papers*, E. Shapiro (ed.), MIT Press, 1987.
- [Ued85] Ueda, K. ; "Guarded Horn Clauses," in *Logic Programming'85*, E. Wada (ed.), LNCS 221, pp. 168-179, Springer-Verlag, 1986.
- [vEdLF82] van Emden, M.H. and de Lucena Filho, G.J. ; "Predicate Logic as a Language for Parallel Programming," in *Logic Programming*, K.L. Clark and S.A. Tärnlund (eds.), Academic Press, 1982.
- [vEK76] van Emden, M.H. and Kowalski, R.A. ; "The Semantics of Predicate Logic as a Programming Language," *J.ACM*, 23(4):733-742, Oct. 1976.
- [vENA84] van Emden, M.H. and Nait Abdallah, M.A. ; "Top-down Semantics of Fair Computations of Logic Programs," Technical Report CS-84-27, Dept. of Computer Science, University of Waterloo, October 1984.
- [WA85] Wadge, W.W. and Ashcroft, E.A. ; *Lucid, the Dataflow Language*, Academic Press, 1985.
- [Wad85] Wadge, W.W. ; "Tense Logic Programming: a Sane Alternative," unpublished manuscript, Dept. of Computer Science, Univ. of Victoria, 1985.
- [Wad88] Wadge, W.W. ; "Tense Logic Programming: a Respectable Alternative," in the *Proceedings of the 1988 International Symposium on Lucid and Intensional Programming*, pp. 26-32, Sidney, B.C., April 7-8, 1988.