A Paraconsistent Semantics for Generalized Logic Programs

Heinrich Herre[1,2] and Axel Hummel[1,2]

[1] Department of Computer Science, Faculty of Mathematics and Computer Science, University of Leipzig, Johannisgasse 26, 04103 Leipzig, Germany, heinrich.herre@imise.uni-leipzig.de, hummel@informatik.uni-leipzig.de
[2] Research Group Ontologies in Medicine and Life Sciences, Institute of Medical Informatics, Statistics and Epidemiology, University of Leipzig, Härtelstrasse 16-18, 04107 Leipzig, Germany

Abstract. We propose a paraconsistent declarative semantics of possibly inconsistent generalized logic programs which allows for arbitrary formulas in the body and in the head of a rule (i.e. does not depend on the presence of any specific connective, such as negation(-as-failure), nor on any specific syntax of rules). For consistent generalized logic programs this semantics coincides with the stable generated models introduced in [HW97], and for normal logic programs it yields the stable models in the sense of [GL88].

Keywords: Paraconsistency, generalized logic programs, multi-valued logic

1 Introduction

Declarative semantics provides a mathematical precise definition of the meaning of a program in a way, which is independent of procedural considerations, easy to manipulate, and reason about. Logic programs and deductive databases should be as easy to write and comprehend as possible and as close to natural discourse as possible. Standard logic programs, in particular definite programs, seems to not be sufficiently expressive for a comprehensible representation of large knowledge bases and of informal descriptions. In recent years there has been an increasing interest in extensions of the classical logic programming paradigm beyond the class of normal logic programs. Generalized logic programs, introduced and studied in [HW97], admit more complex formulas in the rules and thus are more expressive and natural to use since they permit in many cases easier translation from natural language expressions and from informal specifications. The expressive power of generalized logic programs also simplifies the problem of translation of non-monotonic formalisms

into logic programs, [EH97], [EH99]. In many of the traditional logic programming semantics local inconsistency might spoil the whole program, because a contradictory statement $F \wedge \neg F$ implies every formula, i.e. the whole program would be trivialized. This is also the case for the semantics of stable generated models introduced and studied in [HW97].

In this paper we represent a declarative semantics of possibly inconsistent generalized logic programs. This paraconsistent semantics is an extension of the semantics of stable generated models, which agree on normal logic programs with the stable models of Gelfond and Lifschitz [GL88]. We assume the following leading principles for a well-behaved paraconsistent extension of logic programs.

1. The proposed syntax of rules in such programs resembles the syntax of normal logic programs but it applies to a significantly broader class of programs;
2. The proposed semantics of such programs constitutes an intuitively natural extension of the stable semantics of normal logic programs.
3. For consistent generalized logic programs the semantics coincides with the semantics of stable generated models.

The paper has the following structure. After introducing some basic notation in section 2, we introduce in section 3 several kinds of minimal models. From this we derive an adequate paraconsistent semantics for arbitrary theories in predicate logic. In section 4 we discuss the notion of a generalized program. In section 5, we define the concept of a paraconsistent stable generated model and show that the semantics satisfies the before mentioned principles 1.- 3. Section 6 contains the conclusion and a discussion of the related work.

2 Preliminaries

A *signature* $\sigma = \langle Rel, Const, Fun \rangle$ consists of a set of relation symbols, a set of constant symbols, and a set of function symbols. U_σ denotes the set of all ground terms of σ. For a tuple $t_1, \ldots, t_n$ we will also write $\overline{t}$ when its length is of no relevance. The logical functors are $\neg, \wedge, \vee, \rightarrow, \forall, \exists$. $L(\sigma)$ is the smallest set containing the atomic formulas of σ, and being closed with respect to the following conditions: if $F, G \in L(\sigma)$, then $\{\neg F,\ F \wedge G,\ F \vee G,\ F \rightarrow G,\ \exists x F,\ \forall x F\} \subseteq L(\sigma)$.

$L^0(\sigma)$ denotes the corresponding set of *sentences* (closed formulas). For sublanguages of $L(\sigma)$ formed by means of a subset $\mathcal{F}$ of the logical functors, we write $L(\sigma; \mathcal{F})$. Formulas from $L(\sigma; \{\neg, \wedge, \vee, \rightarrow\})$ are called

quantifier-free and a *quantifier-free theory* is a set of quantifier-free formulas. With respect to a signature σ we define $\mathrm{At}(\sigma) = L(\sigma;\emptyset)$, the set of all atomic formulas (also called *atoms*). For a set X of formulas let $\overline{X} = \{\neg F \mid F \in X\}$. Then the set of all *literals* is defined as $\mathrm{Lit}(\sigma) = \mathrm{At}(\sigma) \cup \overline{\mathrm{At}(\sigma)}$. We introduce the following conventions. When $L \subseteq L(\sigma)$ is some sublanguage, L^0 denotes the corresponding set of sentences. If the signature σ does not matter, we omit it and write, e.g., L instead of $L(\sigma)$. ω denotes the least infinite ordinal number, and $Pow(X)$ or 2^X denotes the set of all subsets of X.

A logic $\mathcal{L} = (L, C)$ over the language L can be understood as an operator $C : Pow(L) \to Pow(L)$ determining the consequences of a set $X \subseteq L$ of formulas. Cn denotes the closure operator of classical logic, i.e. $Cn(X)$ is the set of all classical logical consequences of X. Obviously, if X is classically inconsistent then $Cn(X) = L$. A logic (L, C) is said to be a *paraconsistent approximation* of (L, Cn) if the following conditions are satisfied.

1. $C(\{F, \neg F\}) \neq L$ for every formula $F \in L$ (Paraconsistency).
2. If $Cn(X) \neq L$ then $Cn(X) = C(X)$ (Conservativity).
3. $C(X) = C(C(X))$.

Definition 1 (Interpretation) *Let $\sigma = \langle Rel, Const, Fun\rangle$ be a signature. A Herbrand σ-interpretation is a set of literals $I \subseteq \mathrm{Lit}^0(\sigma)$ satisfying the condition $\{a, \neg a\} \cap I \neq \emptyset$ for every ground atom $a \in \mathrm{At}^0(\sigma)$ (interpretations with this property are also called total). Its* universe *is equal to the set of all ground terms U_σ; its canonical interpretation of ground terms is the identity mapping.*

The class of all Herbrand σ-interpretations is denoted by $\boldsymbol{I}(\sigma)$. In the sequel we shall also simply say 'interpretation' instead of 'Herbrand interpretation'. An interpretation I can be represented as a truth-value function from $At^0(\sigma)$ to $\{t, f, \top\}$ by the following stipulation: $I(a) = \top$ if $\{a, \neg a\} \subseteq I$, $I(a) = t$ if $a \in I \wedge \neg a \notin I$, and $I(a) = f$ if $\neg a \in I \wedge a \notin I$. Conversely, every truth-value function $I : At^0 \to \{t, f, \top\}$ can be understood as an interpretation. In the sequel we use the notion of an interpretation simultaneously as a set of literals and as a truth-value function.

A *valuation* over an interpretation I is a function ν from the set of all variables *Var* into the Herbrand universe U_σ, which can be naturally extended to arbitrary terms by $\nu(f(t_1, \ldots, t_n)) = f(\nu(t_1), \ldots, \nu(t_n))$. Analogously, a valuation ν can be canonically extended to arbitrary formulas F, where we write $F\nu$ instead of $\nu(F)$. Furthermore the truth-value function I can be extended to a function $\overline{I}$ which is defined for every formula

$F \in L$. In order to be in a position to give a formal definition of $\overline{I}$ we fix a linear ordering $f < \top < t$ and a unary function *neg* defined by $neg(t) = f, neg(f) = t, neg(\top) = \top$.

Definition 2 (Model Relation) *Let I be an interpretation of signature σ. The extension $\overline{I}$ of I is a function from the set of all sentences F from $L(\sigma)$ into the set $\{t, f, \top\}$, and it is defined inductively by the following conditions.*

1. $\overline{I}(F) = I(F)$ *for every* $F \in At^0(\sigma)$
2. $\overline{I}(\neg F) = neg(\overline{I}(F))$
3. $\overline{I}(F \wedge G) = min\{\overline{I}(F), \overline{I}(G)\}$
4. $\overline{I}(F \vee G) = max\{\overline{I}(F), \overline{I}(G)\}$
5. $\overline{I}(F \rightarrow G) = \overline{I}(\neg F \vee G)$
6. $\overline{I}(\exists x F(x)) = sup\{\overline{I}(F(x/t)) \ : \ t \in U(\sigma)\}$
7. $\overline{I}(\forall x F(x)) = inf\{\overline{I}(F(t)) \ : \ t \in U(\sigma)\}$

Let be $\{t, \top\}$ the set of designated truth values. We say that an interpretation I is a model of a set X of sentences, denoted by $I \models X$, if for every sentence $F \in X$ holds: $\overline{I}(F) \in \{t, \top\}$. The model relation between an interpretation $I \in \boldsymbol{I}(\sigma)$ and a formula $F \in L(\sigma)$ is defined by $I \models F$ iff $I \models F\nu$ for every valuation $\nu : Var \rightarrow U_\sigma$. $\mathrm{Mod}(X) = \{I \in \boldsymbol{I} : I \models X\}$ denotes the Herbrand model operator, and $\models$ denotes the corresponding consequence relation, i.e. $X \models F$ iff $\mathrm{Mod}(X) \subseteq \mathrm{Mod}(F)$. For a set $\mathbf{K} \subseteq \boldsymbol{I}(\sigma)$ and $F \in L(\sigma)$ define $\mathbf{K} \models F$ iff for all $I \in \mathbf{K}$ holds $I \models F$. The set $Th(\mathbf{K}) = \{F \mid \mathbf{K} \models F\}$ is called the theory of $\mathbf{K}$.

Proposition 1 *[We97] Let $C(X)$ be the operator defined by $C(X) = \{F \mid X \models F\}$. Then C satisfies paraconsistency, inclusion, idempotence and compactness, but not conservativity.*

An example which illustrates that the operator C violates the conservativity can be found in [We97, page 14].

Example 1 *Consider the set $X = \{a, a \rightarrow b\}$. Then the following interpretations are models of X: $I_1 = \{a, b\}$, $I_2 = \{a, b, \neg b\}$, $I_3 = \{a, \neg a, b\}$, $I_4 = \{a, \neg a, \neg b\}$, $I_5 = \{a, \neg a, b, \neg b\}$. Because of I_4 we obtain $b \notin C(X)$.*

3 Minimal Models

Our aim is to define a semantics for logic programs which defines for classical consistent programs the (two-valued) stable generated models,

but in case of inconsistent programs yields suitable three-valued models assuring paraconsistency. The class of models $\mathrm{Mod}(X)$ is not suitable because conservativity does not hold, i.e. there are consistent theories T such that $C(T) \neq Cn(T)$. An adequate solution of this problem uses different types of minimal models, one of them, minimizing inconsistency, was introduced in [Pr91] and studied in [We97]. Let $\leq$ be a transitive and reflexive relation on the set $\mathcal{K}$. An element $M \in \mathcal{K}$ is said to be $\leq$-minimal if and only if there is no $N \in \mathcal{K}$ such that $N \leq M$ and $N \neq M$. Let $\mathrm{Min}_{\leq}(\mathcal{K})$ be denote the set of $\leq$-minimal elements of $\mathcal{K}$. In this section we analyze several forms of minimal models of a paraconsistent theory. Let I be an interpretation, then $Pos(I) = I \cap At^0$ and $Neg(I) = I \cap \{\neg a : a \in At^0\}$. Let be $inc(I) = \{a : \{a, \neg a\} \subseteq I\}$.

Definition 3 *Let I, J be interpretations. Then*

1. $I \preceq J$ *if and only if* $Pos(I) \subseteq Pos(J)$ *and* $Neg(J) \subseteq Neg(I)$;
2. $I \sqsubseteq J$ *if and only if* $inc(I) \subseteq inc(J)$.

Using the relations $\preceq, \sqsubseteq, \subseteq$ we introduce the following forms of minimality.

Definition 4 *Let X be a set of formulas and I be an interpretation.*

1. *I is a t-minimal model of X iff* $I \in \mathrm{Min}_{\preceq}(\mathrm{Mod}(X))$.
2. *I is an inc-minimal model of X iff* $I \in \mathrm{Min}_{\sqsubseteq}(\mathrm{Mod}(X))$.
3. *I is an i-minimal model of X iff* $I \in \mathrm{Min}_{\subseteq}(\mathrm{Mod}(X))$.

We introduce following model operators: $\mathrm{Mod}_{incm}(X) = \{I \mid I$ is an inc-minimal model of $X\}$, $\mathrm{Mod}_{im}(X) = \{I \mid I$ is an i-minimal model of $X\}$, and $\mathrm{Mod}_{tm}(X) = \{I \mid I$ is a t-minimal model of $X\}$. Using these different notions of minimality we get following consequence operations, for $* \in \{incm, tm, im\}$: $X \models_* F$ iff $\mathrm{Mod}_*(X) \subseteq \mathrm{Mod}(\{F\})$.

Proposition 2 *Every inc-minimal model of a set X of formulas is i-minimal.*

Proof: Let I be an inc-minimal model of X and assume that I is not i-minimal. Then there is a model $J \subseteq I$, $J \models X$, and $J \neq I$. This implies the existence of a literal $l \in I - J$, assume that $l = a$ is a ground atom. Since J is a total interpretation this yields $\neg a \in J$, but then $\{a, \neg a\} \subseteq I$, which contradicts the inc-minimality of I. $\square$

There are t-minimal models not being inc-minimal and inc-minimal models not being t-minimal.

Example 2 *For clarification, we discuss the following examples:*

1. *Consider the program* $P_1 = \{\neg b \rightarrow a\}$. *Obviously,* $I = \{a, \neg a, \neg b\}$ *is a t-minimal model of* P, *but* I *is not inc-minimal because* $I = \{a, \neg b\}$ *is a model of* P *satisfying* $inc(J) = \emptyset$. *There are, trivially inc-minimal models not being t-minimal (note that every two-valued model is inc-minimal).*
2. *Let* $P_2 = \{b \rightarrow \neg a; \rightarrow a\}$. *Then* $I = \{\neg a, a, b\}$ *is an i-minimal model not being inc-minimal, since* $\{a, \neg b\}$ *is a model of* P.
3. *Every two-valued model is i-minimal (among the total models), but not, in general t-minimal. There are also t-minimal models not being i-minimal: the interpretation* $\{\neg a, a, \neg b\}$ *is a t-minimal model of* P_1, *but it is not i-minimal.*

We recall the following result from [We97].

Proposition 3 *1. Let* T *be a quantifier-free theory. Then the theory* T *has an inc-minimal model.*
2. *The consequence operation* $C_{incm}(X) = \{F \mid X \models_{incm} F\}$ *is a paraconsistent, non-monotonic approximation of* Cn.

The main result of this section is the following.

Proposition 4 *Let* T *be a quantifier-free theory and* I *an inc-minimal model of* T. *Then there exists a model* J *of* T *such that*

1. $inc(I) = inc(J)$
2. $J \preceq I$
3. *for all* $J_0 \preceq J$ *such that* $J_0 \neq J$ *either* $inc(J_0) \neq inc(I)$ *or* $J_0 \not\models T$.

Proof: We may assume that T is a set of clauses, these are formulas of the form $a_1 \vee \ldots \vee a_m \vee \neg b_1 \vee \ldots \vee \neg b_n$, where a_i, b_j are atomic formulas. Let be $\Delta(I) = \{J \mid J \preceq I$ and $J \models T$ and $inc(I) = inc(J)\}$. We consider decreasing sequences within the system $(\Delta(I), \preceq)$. Let be $J_0 \succeq \ldots \succeq J_n \succeq \ldots$ a decreasing sequence, $J_n \in \Delta(I), n < \omega$. Obviously, the sequence $\{J_n \mid n < \omega\}$ has a lower bound J^*, defined by $Pos(J^*) = \bigcap\{Pos(J_n) | n < \omega\}$, and $Neg(J^*) = \bigcup\{Neg(J_n) | n < \omega\}$. We show that $J^* \in \Delta(I)$. By *Zorn's lemma* this implies the result of the theorem. Since $inc(J^*) = inc(I)$, it remains to show that $J^* \models T$. Assume, this is not the case; then there is a formula $F(\overline{x}) = a_1(\overline{x}) \vee \ldots \vee a_m(\overline{x}) \vee \neg b_1(\overline{x}) \vee \ldots \vee \neg b_n(\overline{x})$ from T such that $J^* \not\models \forall \overline{x} F(\overline{x})$, which implies $J^*(\forall \overline{x} F(\overline{x})) \notin \{\top, t\}$, hence $J^*(\forall \overline{x} F(\overline{x})) = f$ which yields $J^*(\neg \forall \overline{x} F(\overline{x})) = J^*(\exists \overline{x} \neg F(\overline{x})) = t$. Hence, there are variable-free terms $\bar{t}$ such that $\{\neg a_1(\bar{t}), \ldots, \neg a_m(\bar{t}), b_1(\bar{t}), \ldots, b_n(\bar{t})\} \subseteq J^*$. This

condition implies $\{b_1(\bar{t}), \ldots, b_n(\bar{t})\} \subseteq J_n$ for every $n < \omega$. But, there must be also a number $k < \omega$ such that $\{\neg a_1(\bar{t}), \ldots, \neg a_m(\bar{t})\} \subseteq J_k$, and this implies $J_k(\bigwedge_{i \leq m} \neg a_i(\bar{t}) \wedge \bigwedge_{j \leq n} b_j(\bar{t})) = t$ (the value $\top$ is not possible, otherwise this would imply $inc(J_k) \neq inc(J^*)$). This gives a contradiction. □

Proposition 4 shows that for every model I of a quantifier-free theory T the set $\mathrm{Min}_{\preceq}(\{J \mid inc(J) = inc(I),\ J \in \mathrm{Mod}(T)\})$ is non-empty.

Corollary 5 *Let T be a universal theory, I a model of T and let J be a set of literals such that $J \preceq I$ and $inc(J) = inc(I)$. Then the set $\{K \mid J \preceq K \preceq I$ and $K \models T$ and $inc(K) = inc(I)\}$ contains a $\preceq$-minimal element.*

Proof: Follows immediately from proposition 4.

4 Sequents and Logic Programs

In the sequel we use Gentzen-sequents to represent rule knowledge as proposed in [HW97]. A sequent, then, is not a schematic but a concrete expression representing some piece of knowledge.[3]

Definition 5 (Sequent) *A* sequent *s is an expression of the form*

$$F_1, \ldots, F_m \ \Rightarrow\ G_1, \ldots, G_n$$

where $F_i, G_j \in L(\sigma)$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The body *of s, denoted by $B(s)$, is given by $\{F_1, \ldots, F_m\}$, and the* head *of s, denoted by $H(s)$, is given by $\{G_1, \ldots, G_n\}$.* $\mathrm{Seq}(\sigma)$ *denotes the class of all sequents s such that $H(s), B(s) \subseteq L(\sigma)$, and for a given set $S \subseteq \mathrm{Seq}(\sigma)$, $[S]$ denotes the set of all ground instances of sequences from S.*

The satisfaction set of a formula $F \in L(\sigma)$ with respect to an interpretation $I \in \boldsymbol{I}(\sigma)$ is defined as $\mathrm{Sat}_I(F) = \{\nu \in {U_I}^{Var} : I \models F\nu\}$

Definition 6 (Model of a Sequent) *Let $I \in \boldsymbol{I}$. Then,*

$$I \models F_1, \ldots, F_m \ \Rightarrow\ G_1, \ldots, G_n \quad \textit{iff} \quad \bigcap_{i \leq m} Sat_I(F_i) \subseteq \bigcup_{j \leq n} Sat_I(G_j)$$

[3] The use of sequents is mainly technico-methodological, the sequent-arrow $\Rightarrow$ should be distinguished from the implication connective $\rightarrow$.

Obviously for every sequent $B \Rightarrow H$ and $I \in \boldsymbol{I}$ we have $I \models B \Rightarrow H$ iff $I \models \bigwedge B \rightarrow \bigvee H$. Sometimes, we represent the latter formula by the expression $\bigvee H \leftarrow \bigwedge B$. We define the following classes of sequents corresponding normal, normal disjunctive and generalized logic programs, respectively.

1. $\mathrm{NLP}(\sigma) = \{s \in \mathrm{Seq}(\sigma) : H(s) \in \mathrm{At}(\sigma),\ B(s) \subseteq \mathrm{Lit}(\sigma)\}$.
2. $\mathrm{NDLP}(\sigma) = \{s \in \mathrm{Seq}(\sigma) : H(s) \subseteq \mathrm{At}(\sigma),\ B(s) \subseteq \mathrm{Lit}(\sigma), H(s) \neq \emptyset\}$.
3. $\mathrm{GLP}(\sigma) = \{s \in \mathrm{Seq}(\sigma) : H(s), B(s) \subseteq L(\sigma; \neg, \wedge, \vee, \rightarrow)\}$.

For $P \subseteq \mathrm{GLP}(\sigma)$, the model operators $\mathrm{Mod}_*(P), * \in \{inc, i, t\}$ are defined as in section 3. The associated entailment relations are defined by $P \models_* F$ iff $\mathrm{Mod}_*(P) \subseteq \mathrm{Mod}(F)$, where $* = inc, i, t$, and $F \in L(\sigma)$. The *ground instantiation* of a generalized logic program P is denoted by $[P]$, and defined by $[P] = \{\theta(r) \mid r \in P, \theta \text{ is a ground substitution}\}$. Obviously, if $P \subseteq \mathrm{GLP}$, then $\mathrm{Mod}(P) = \mathrm{Mod}([P])$.

A preferential semantics for sequents is given by a preferred model operator $\Phi : 2^{\mathrm{Seq}} \rightarrow 2^{\mathbf{I}}$, satisfying the condition $\Phi(P) \subseteq \mathrm{Mod}(P)$ and selecting suitable preferred models. Our intuitive understanding of rules suggests a meaning which interprets a sequent as a rule for generating information. We may consider a model I of a set P of sequents as intended if I can be generated bottom-up starting from a suitable least interpretation by an iterated application of the sequents $r \in [P]$. A model of P which can be generated in this way is said to be *grounded* in P. The following examples show that even the following strong form of minimality is not sufficient to satisfy this condition. A model I of P is said to be *inc-t-minimal* if I is *inc-minimal* and there is no model J of P satisfying the conditions $inc(J) = inc(I)$, $J \preceq I$, and $J \neq I$.

Example 3 *1. Let $P = \{\neg p(a) \Rightarrow q(a)\}$. Every intended model of P should contain $q(a)$. But $M = \{p(a), \neg q(a)\}$ is also an inc-t-minimal model of P.*

2. This observation is also valid if P has no two-valued model.
Let $P = \{\Rightarrow r(c);\ \Rightarrow \neg p(a);\ \Rightarrow \neg p(b);\ \Rightarrow p(a), p(b);\ \neg p(x) \Rightarrow q(x)\}$. Every intended model of P should contain $q(c)$. Assume $I \models P$ is inc-minimal, but $q(c) \notin I$. Then, $\neg q(c) \in I$, which implies $p(c) \in I$ (note that I is total). But $p(c)$ cannot be generated by applying the sequents from $[P]$ because $p(c)$ does not appear in the head of any rule $s \in [P]$. [4]

[4] Application of a rule r means, roughly, to make the body $B(r)$ true and then to detach the head $H(r)$.

But $M_1 = \{\neg p(a), \neg p(b), p(a), q(a), q(b), \neg q(c), p(c), r(c), \neg r(a), \neg r(b)\}$ *is an inc-t-minimal model of* P.

5 Paraconsistent Stable Generated Models

Definition 7 (Interpretation Interval) *Let* $I_1, I_2 \in \boldsymbol{I}$ *and* $inc(I_1) = inc(I_2)$. *Then,* $[I_1, I_2] = \{I \in \boldsymbol{I} : I_1 \preceq I \preceq I_2$ *and* $inc(I) = inc(I_1)\}$. *For a program* $P \subseteq$ GLP *let be* $P_{[I_1,I_2]} = \{r \mid r \in [P], [I_1, I_2] \models B(r)\}$.

The following definition of a *paraconsistent stable generated model* combines the construction of a generated model in [HW97] with the notion of inc-minimality.

Definition 8 (Paraconsistent Stable Generated Model) *Let* $P \subseteq$ GLP(σ). *An inc-minimal model* M *of* P *is called* paraconsistent stable generated, *symbolically* $M \in \mathrm{Mod}_{ps}(P)$, *if there is a chain of Herbrand interpretations* $I_0 \preceq \ldots \preceq I_\kappa$ *such that* $M = I_\kappa$, *and*

1. M *is inc-minimal*
2. $I_0 = inc(M) \cup \{\neg a \mid a \in At^0(\sigma)\}$.
3. *For successor ordinals* α *with* $0 < \alpha \leq \kappa$, I_α *is a* $\preceq$*-minimal extension of* $I_{\alpha-1}$ *satisfying the heads of all sequents whose bodies hold in* $[I_{\alpha-1}, M]$, *i.e.* $I_\alpha \in \mathrm{Min}_{\preceq}\{I \in \boldsymbol{I}(\sigma) : M \succeq I \succeq I_{\alpha-1}, inc(M) = inc(I), I \models \bigvee H(s)$, *for all* $s \in P_{[I_{\alpha-1},M]}\}$
4. *For limit ordinals* $\lambda \leq \kappa$, $I_\lambda = \sup_{\alpha<\lambda} I_\alpha$.

We also say that M *is* generated *by the* P-stable chain $I_0 \preceq \ldots \preceq I_\kappa$.

Intuitive, we define that an inc-minimal model M of a generalized logic program P is a paraconsistent stable generated model of P if M is created bottom-up by an iterative application of the rules of P starting with the state of no information (that means every atom is negated) and the inconsistency of M. In every step of the construction the model is extended in that way that the inconsistency is preserved and the head of every applicable sequent is satisfied.

Example 4 *Let* P *be the second logic program of Example 3. Because of the rules* $\{\Rightarrow \neg p(a); \Rightarrow \neg p(b); \Rightarrow p(a), p(b)\}$ *it is easy to see that* P *has no two-valued model. But there are two paraconsistent stable generated models:*
$M_2 = \{\neg r(a), \neg r(b), r(c), \neg p(a), \neg p(b), p(a), \neg p(c), q(a), q(b), q(c)\}$ *and*
$M_3 = \{\neg r(a), \neg r(b), r(c), \neg p(a), \neg p(b), p(b), \neg p(c), q(a), q(b), q(c)\}$.

The model M_2 is constructed by the chain $I_0^2 \preceq I_1^2 = M_2$.
In detail we obtain:
$I_0^2 = \{p(a)\} \cup \{\neg r(a), \neg r(b), \neg r(c), \neg p(a), \neg p(b), \neg p(c), \neg q(a), \neg q(b), \neg q(c)\}$.
So $P_{[I_0^2, M_2]} = \{\Rightarrow r(c);\ \Rightarrow \neg p(a);\ \Rightarrow \neg p(b);\ \Rightarrow p(a), p(b);\ \neg p(a) \Rightarrow q(a);\ \neg p(b) \Rightarrow q(b);\ \neg p(c) \Rightarrow q(c)\}$. *Therefore* $I_1^2 = M_2$ *with*
$I_1^2 = \{\neg r(a), \neg r(b), r(c), \neg p(a), \neg p(b), p(a), \neg p(c), q(a), q(b), q(c)\}$.
For M_3 we obtain:
$I_0^3 = \{p(b)\} \cup \{\neg r(a), \neg r(b), \neg r(c), \neg p(a), \neg p(b), \neg p(c), \neg q(a), \neg q(b), \neg q(c)\}$.
So $P_{[I_0^3, M_3]} = \{\Rightarrow r(c);\ \Rightarrow \neg p(a);\ \Rightarrow \neg p(b);\ \Rightarrow p(a), p(b);\ \neg p(a) \Rightarrow q(a);\ \neg p(b) \Rightarrow q(b);\ \neg p(c) \Rightarrow q(c)\}$ *and therefore* $I_1^3 = M_3$ *with*
$I_1^3 = \{\neg r(a), \neg r(b), r(c), \neg p(a), \neg p(b), p(b), \neg p(c), q(a), q(b), q(c)\}$.
Hence it follows: $P \models_{ps} q(c)$.

Remark: If we assume in definition 8 that the set $inc(M)$ is empty then we get the notion of a *stable generated model* as introduced and studied in [HW97].

Notice that the notion of stable generated models applies to programs admitting negation(-as-failure) in the head of a rule and nested negations, such as in $p(x) \wedge \neg(q(x) \wedge \neg r(x)) \Rightarrow s(x)$ which would be the result of folding $p(x) \wedge \neg ab(x) \Rightarrow s(x)$ and $q(x) \wedge \neg r(x) \Rightarrow ab(x)$.
It turns out that the length of the generated sequence of a stable generated model can be restricted by ω.

Proposition 6 *Let $P \subseteq$ GLP, and let M be a paraconsistent stable generated model of P generated by the sequence $M_0 \preceq \ldots \preceq M_\kappa$. Then there is an ordinal $\beta \leq \omega$ such that $M_\beta = M_\kappa$.*

Proof: We show that every sequence stabilizes at an ordinal $\beta \leq \omega$. Obviously, if $M_\alpha = M_{\alpha+1}$ then $M_\alpha = M_\gamma$ for all $\alpha < \gamma \leq \kappa$. It is sufficient to prove $M_\omega = M_{\omega+1}$. Analogously to [HW97], we proceed in two steps:

(1) First we show that if $s \in [P]$ and $[M_\omega, M] \models B(s)$ then there is a number $n < \omega$ such that $[M_n, M] \models B(s)$. Without loss of generality, we may assume that $B(s)$ is a set of clauses (disjunction of ground literals), i.e. $B(s) = \{C_1, \ldots, C_k\}$, $C_i = a_1^i \vee \ldots \vee a_{m_i}^i \vee \neg b_1^i \vee \ldots \vee \neg b_{n_i}^i$, $i \in \{1, 2, ..., k\}$. A clause C_i is said to be positive if the set $P(i) := M_\omega \cap \{a_1^i, \ldots, a_{m_i}^i\}$ is nonempty, otherwise it is called negative. Let $\{C_1, \ldots, C_s\}$ be the set of positive and $\{C_{s+1}, \ldots, C_k\}$ the set of negative clauses. Because the set $P := \bigcup_{1 \leq i \leq s} P(i)$ is finite, there is a number $j < \omega$ such that $P \subseteq M_j$. Then, trivially, $[M_j, M] \models C_1, \ldots, C_s$. It remains to show: if $M_j \preceq J \preceq M$ then $J \models C_{s+1}, \ldots, C_k$. We proof this

fact indirectly. Assume $J \not\models C_{s+1}, \ldots, C_k$. Then, there exists a number j, $s+1 \leq j \leq k$, such that $J \not\models C_j$ with the following form $C_j = a_1^j \vee \ldots \vee a_{m_j}^j \vee \neg b_1^j \vee \ldots \vee \neg b_{n_j}^j$. Because of $J(C_j) = f$, we obtain $neg(J(C_j)) = t$ and therefore $J \models \neg C_j$. So $J \models \neg a_1^j \wedge \ldots \wedge \neg a_{m_j}^j \wedge b_1^j \wedge \ldots \wedge b_{n_j}^j$ since De Morgan's laws are valid in our paraconsistent semantics. We may assume that the elements in the set $\{a_1^j, \ldots, a_{m_j}^j, b_1^j, \ldots, b_{n_j}^j\}$ are pairwise distinct. Now, we define $M_\omega^* = (M_\omega \setminus \{\neg b_1^j, \ldots, \neg b_{n_j}^j\}) \cup \{b_1^j, \ldots, b_{n_j}^j\}$. Then $Pos(M_\omega) \subseteq Pos(M_\omega^*)$ and $Neg(M_\omega^*) \subseteq Neg(M_\omega)$. So $M_\omega \preceq M_\omega^*$. Furthermore it holds $M_\omega^* \preceq M$ and therefore we obtain $M_\omega^* \in [M_\omega, M]$. Because of $M_\omega^* \cap \{a_1^j, \ldots, a_{m_j}^j, \neg b_1^j, \ldots, \neg b_{n_j}^j\} = \emptyset$ and $\{b_1^j, \ldots, b_{n_j}^j\} \subseteq M_\omega^*$ it follows $M_\omega^* \not\models C_j$. This is a contradiction to $[M_\omega, M] \models C_j$.

(2) Now, we show that $M_\omega = M_{\omega+1}$. It is sufficient to prove: if $s \in P_{[M_\omega, M]}$, then $M_\omega \models \bigvee H(s)$. By (1), the condition $s \in P_{[M_\omega, M]}$ implies that $s \in P_{[M_n, M]}$ for a certain number $n < \omega$, and hence for every $j > n$: $s \in P_{[M_j, M]}$. Hence, $M_j \models \bigvee H(s)$ for every j, $n < j < \omega$. Again, we may assume that $\bigvee H(s)$ is given as a set of clauses $\{C_1, \ldots, C_n\}$. We have to check that $M_\omega \models C_1, \ldots, C_n$. Assume, there is a j, $1 \leq j \leq n$, such that $M_\omega \not\models C_j$, then $M_\omega \models \neg C_j$, $C_j = a_1^j \vee \ldots \vee a_{m_j}^j \vee \neg b_1^j \vee \ldots \vee \neg b_{n_j}^j$, and $M_\omega \models \neg a_1^j \wedge \ldots \wedge \neg a_{m_j}^j \wedge b_1^j \wedge \ldots \wedge b_{n_j}^j$. It is easy to show that there exists a number $m < \omega$ such that $\{b_1^j, \ldots, b_{n_j}^j\} \subseteq M_m$, and from this follows $M_m \not\models C_j$, which is a contradiction. □

Corollary 7 *If M is a paraconsistent stable generated model of $P \subseteq$ GLP, then there is either a finite P-stable chain, or a P-stable chain of length ω, generating M.*

The following example shows that stable generated entailment is not cumulative, i.e. adding derivable formulas to programs may change their consequence set.

Example 5 (Observation 18, [HJW99]) *Let P be the following logic program: $P = \{\neg r(a) \Rightarrow q(a); \neg q(a) \Rightarrow r(a); \neg p(a) \Rightarrow p(a); \neg r(a) \Rightarrow p(a);\}$. Then $Mod_{ps}(P) = \{\{p(a), q(a)\}\}$. Therefore $P \models_{ps} p(a), q(a)$. But $Mod_{ps}(P \cup \{p(a)\}) = \{\{p(a), q(a)\}, \{p(a), r(a)\}\}$ and hence $P \cup p(a) \not\models q(a)$.*

The relation to consistent generalized programs is captured by the following proposition.

Proposition 8 *Let P be a generalized logic program, and assume P is consistent, i.e. has a two-valued classical interpretation. Then a model I of P is paraconsistent stable generated if and only if it is stable generated.*

Proof: By proposition 3 every inc-minimal model is two-valued. □

Corollary 9 *Let P be a normal logic program. Then a model I of P is paraconsistent stable generated if and only if it is stable (in the sense of [GL88]).*

Proof: P is always a two-valued model, since the negation $\neg$ does not appear in the heads of the rules. Now we may apply the preceding proposition and the result in [HW97] (stating that the stable generated models coincide with the stable models for normal logic programs). □

6 Conclusion and Related Work

A framework of paraconsistent logic programs was firstly developed by Blair and Subrahmanian in [BS89]; they employ Belnap's four-valued logic [Be77] as a theoretical basis, but this framework does not treat default negation in a program. Kifer and Lozinskii in [KL92] extend Blair's framework to theories possibly containing default negation. Sakama and Inoue are studying programs in [SI95] whose rules admit disjunction in the head and default negation in the bodies of the rules. Our approach gives a declarative semantics to logic programs whose rules admit arbitrary quantifier-free formulas in the heads and bodies containing negation that can be interpreted as default negation. This semantics coincides on normal logic programs with the stable models in [GL88]. Note, that stable models I satisfies the condition $I \cap \{a, \neg a\} \neq \emptyset$ for every ground atom, and that any adequate generalization of this notion to paraconsistent models should preserve this property. This is the reason, why we assume that the considered interpretations to be total. Our semantics uses the concept of minimal inconsistent interpretations as introduced by Priest in [Pr91], the results in [We97] and the notion of a stable generated model introduced and studied in [HW97].

By introducing a general definition of paraconsistent stable generated models, we have continued the foundation of a *stable model theory* for possibly inconsistent logic programs. It seems to be possible to analyze further extensions of normal logic programs within a similar framework, such as admitting quantifiers in the bodies and the heads of rules. As a consequence of the rapid growth of the Semantic Web, powerful ontology languages like Extended RDF [AADW08] were developed which use the logic programming paradigm. Therefore the application of the stable model theory to that family of languages is a beneficial challenge

for the near future. In [Hu09] a paraconsistent stable generated semantics for a four-valued logic is proposed which depends on the minimally inconsistent models too. Another interesting step is to develop a paraconsistent declarative semantics for generalized logic programs with two kinds of negation satisfying the coherency condition, i.e. $\sim F$ implies $\neg F$, where $\sim$ represents strong negation, and $\neg$ means weak negation which is assumed to be total.

Since the stable models in the sense of [GL88] correspond to the stable models of [FLL07] for normal logic programs, the stable models in the sense of [FLL07] agree also with the two-valued stable generated models if normal logic programs are considered. Because of the fact that the semantics of [FLL07] is also defined for generalized logic programs, a detailed characterization of the relationship between this semantics and the stable generated models belongs to our future plans.

Acknowledgment
Thanks due to the anonymous referees for their criticism and useful comments.

References

[AADW08] Analyti, A, Antoniou, G, Damsio, C. V. and Wagner, G.: Extended RDF as a Semantic Foundation of Rule Markup Languages, *Journal of Artificial Intelligence Research*, 32: 37-94, 2008

[Be77] Belnap, D.N.: A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern Uses of Many-valued Logic*, 8-37, Reidel, 1977

[BS89] Blair, H.A. and Subrahamanian, V.S.: Paraconsistent logic programming, *Theoretical Computer Science*. 68: 135-154, 1989

[EH97] Engelfriet, J. and Herre, H.: Generated Preferred Models and Extensions of Nonmonotonic Systems, *Logic Programming*, 85-99, Proc. of the 1997 International Symposium, ed. J. Maluszynski, The MIT Press, 1997

[EH99] Engelfriet, J. and Herre, H.: Stable Generated Models, Partial Temporal Logic and Disjunctive Defaults, *Journal of Logic and Algebraic Programming*, 41 (1): 1-25, 1999

[FLL07] Ferraris, P., Lee, J. and Lifschitz, V.: A New Perspective on Stable Models, 372-379, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007, 2007

[GL88] Gelfond, M. and Lifschitz, V.: The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, editors, *Proc. of ICLP*, 1070–1080. MIT Press, 1988

[Hu09] Hummel, A.: Untersuchungen von Ontologiesprachen mit modelltheoretischer Semantik, Diploma Thesis, University of Leipzig, 2009

[HW97] Herre, H. and Wagner, G.: Stable Models Are Generated by a Stable Chain, *Journal of Logic Programming*, 30 (2): 166-177, 1997

[HJW99] Herre, H., Jaspars, J. and Wagner, G.: Partial logics with two kinds of negation as a foundation for knowledge-based reasoning, in D. Gabbay and H. Wansing (Eds.), *What is negation ?*, pages 121-159. Kluwer Academic Publishers, 1999

[KL92] Kifer, M. and Lozinskii, E.: A logic for reasoning with inconsistency. *Journal of Automated Reasoning*, 8: 179-215, 1992

[Pr91] Priest, G: Minimally inconsistent LP. *Studia Logica*, 50 (2): 321-331, 1991

[SI95] Sakama, C. and Inoue, K.: Paraconsistent Stable Semantics for extended disjunctive programs; *Journal of Logic and Computation* 5: 265-285, 1995

[We97] Weber, S.: Investigations in Belnap's Logic of Inconsistent and Unknown Information, Dissertation, University of Leipzig, 1998