Well-Supported Models of Disjunctive Logic Programs

Martin Baláž balaz@ii.fmph.uniba.sk

Comenius University, Faculty of Mathematics, Physics and Informatics,
Mlynská dolina, 842 48 Bratislava, Slovakia

Abstract. The stable models semantics is nowadays one of the most accepted semantics. For logic programs without disjunction, there also exists the well-supported model semantics. It describes the same set of models, but in spite of the stable model semantics it uses level mappings which can be used to detect cyclic dependencies among literals.
We are interested in similar characterization of stable models of logic programs with disjunction. We will use AND/OR graphs to formalize the derivation of stable models and to detect cyclic dependencies among literals.

1 Introduction

Stable models [1, 2] are minimal models of a program reduct, which is obtained from the original program by evaluation of default literals. Stable models of logic programs without disjunction are supported [3], i.e. for every true atom there must exists a rule containing it in the head and having satisfied body. From the original definition of stable models, it is not easy to see that there must exists a supporting rule whose assumptions in addition do not strictly depend on the conclusion.

Example 1. The model $M = \{a, b\}$ of the logic program $P = \{a \leftarrow b; b \leftarrow a\}$ is supported. The atom a is supported by a rule $a \leftarrow b$ and the atom b is supported by the rule $b \leftarrow a$. But the truth value of a can not be derived from b, because to derive the truth value of b we already have to know the truth value of a. The model M of P is not stable.

For logic programs without disjunction, there also exists the well-supported model semantics [4]. A model is well-supported if every true atom is supported by a rule whose assumptions are derived before its conclusion. The level of derivation of an atom is expressed by level mapping. In general, there may exist more level mappings, each of them corresponds to different way of derivation. An assumption of a rule strictly depends on the conclusion of the rule if there does not exists a level mapping representing the derivation of the model such that the level of the conclusion is greater than level of the assumption. It has been shown that stable models coincide with well-supported models.

In the case of logic programs with disjunction, we need to distinguish between weak and strong support [5]. A rule weakly supports an atom in its head if it has satisfied body. If the supported atom is the only true atom in the head, it is strongly supported.

Example 2. The interpretation $M = \{a, b\}$ is a stable model of the logic program $P = \{a \vee b; a \leftarrow b; b \leftarrow a\}$. To incrementally derive M, we need to use not only strongly supporting rules but also weakly supporting rules. The rule $a \vee b$ is the only rule we can start with. It weakly supports both atoms a and b. If we derive the atom a, by using the rule $b \leftarrow a$ we later derive the atom b. Symmetrically, if we derive the atom b, by using the rule $a \leftarrow b$ we can derive the atom a. The only reason, why we need the weakly supporting rule, is that the atoms a and b occurring in the head of the same rule $a \vee b$ are cyclic dependent.

It is important that we use the weakly supporting rule $a \vee b$ to derive only one atom at once. M is also a model of the logic program $P' = \{a \vee b\}$, but it is not stable. If we use $a \vee b$ to derive a, there is no rule left to derive b. Symmetrically, if we use $a \vee b$ to derive b, there is no rule left to derive a.

Like in the case of logic programs without disjunction, the derivation procedure avoids cyclic dependencies between literals, but only if they do not occur in the heads of rules.

In the next sections we show how we can use the non-deterministic immediate consequence operator for defining one step of derivation. We use AND/OR graphs to formalize derivation procedure of minimal models. Then we extend the well-support for disjunctive logic programs and show how level mappings are related to derivations.

2 Preliminaries

A *(propositional) language* is a countable set $\mathcal{L}$ of *atoms*. A *literal* is either an atom A or its default negation $not\, A$. A *rule* is a formula r of the form $A_1 \vee \ldots \vee A_m \leftarrow L_{m+1} \wedge \ldots \wedge L_n$ where $1 \leq m \leq n$, and $A_i, 1 \leq i \leq m$, are atoms, $L_i, m < i \leq n$, are literals. The disjunction $A_1 \vee \ldots \vee A_m$ is called a *head of* the rule r (denoted by $head(r)$) and the conjunction $L_{m+1} \wedge \ldots \wedge L_n$ is called a *body of* the rule r (denoted by $body(r)$). For simplicity, if the body of a rule is empty, we will skip the symbol "$\leftarrow$". A *disjunctive logic program* (DLP) is a countable set of rules and a *normal logic program* (NLP) is a countable set of rules not containing disjunction. A *positive disjunctive logic program* (PDLP) is a DLP which does not contain default negation. A *positive logic program* (PLP) is a NLP which does not contain default negation.

A *dependency graph of* a DLP is a directed graph where nodes are atoms and an atom A is connected to an atom B iff there exists a rule containing A in the body and B in the head. A DLP is *head cycle free* iff it does not contain a rule with two different atoms in the head which appear in the same cycle in the dependency graph.

An *interpretation* is a set of atoms. I *satisfies* an atom A ($I \models A$) iff $A \in I$, and a literal *not* A ($I \models \mathit{not}\, A$) iff $A \notin I$. I *satisfies* a conjunction $L_1 \wedge \ldots \wedge L_n$, $n \geq 0$ ($I \models L_1 \wedge \ldots \wedge L_n$) iff it satisfies all literals L_i, $1 \leq i \leq n$, and I *satisfies* a disjunction $L_1 \vee \ldots \vee L_n$, $n \geq 0$ ($I \models L_1 \vee \ldots \vee L_n$) iff it satisfies at least one literal L_i, $1 \leq i \leq n$. I *satisfies* a rule r ($I \models r$) iff it satisfies the head of r whenever it satisfies the body of r. An interpretation I is a *model* of a DLP P ($I \models P$) iff it satisfies all rules r from P.

A model M of P is *minimal* iff does not exists a model N of P such that $N \subset M$. A model M of P is *least* iff for all models N of P holds $M \subseteq N$. By $MM(P)$ we will denote the set of all minimal models of P and by $LM(P)$ we will denote the least model of P.

An atom A in an interpretation I is *supported by* a rule r from a NLP P iff A is the head of r and I satisfies the body of r. I is supported by P iff every atom A in I is supported by a rule r from P. An atom A in an interpretation I is *weakly supported by* a rule r from a DLP P iff A is in the head of r and I satisfies the body of r. A is *strongly supported by* r if A is the only one true atom weakly supported by r. We say that I is *weakly (resp. strongly) supported* by P iff every atom A in I is weakly (resp. strongly) supported by a rule r from P.

A *reduct of* a DLP P with respect to an interpretation I is a PDLP P^I obtained from P by removing rules containing false default literal in the body and by removing all remaining default literals. We say that I is a stable model of P if I is a minimal model of P^I.

3 Minimal Models

The immediate consequence operator [6] computes all necessary consequences of applicable rules in given interpretation. Applying such operator can be viewed as one step of derivation.

Definition 1 (Immediate Consequence Operator). *Let I be an interpretation and P be a PLP. An* immediate consequence operator *is defined as follows:*

$$T_P(I) = \{\mathit{head}(r) \mid r \in P, I \models \mathit{body}(r)\}$$

*An α-*iteration of *an immediate consequence operator T_P is defined as follows:*

$$T_P \uparrow \alpha = \begin{cases} \emptyset & \text{if } \alpha = 0 \\ T_P(T_P \uparrow \beta) & \text{if } \alpha \text{ is a successor ordinal of } \beta \\ \bigcup_{\beta < \alpha} T_P \uparrow \beta & \text{if } \alpha \text{ is a limit ordinal} \end{cases}$$

Proposition 1. *An interpretation I is a minimal model of a PLP P iff I is a minimal fixpoint of T_P, i.e. I is a fixpoint of T_P ($T_P(I) = I$) and there does not exists a fixpoint J of T_P such that $J \subset I$. [6]*

The following proposition states that positive logic programs have always only one minimal model which is in addition the least model. It can be computed by iteration of the immediate consequence operator after at most ω steps.

Proposition 2. *Let P be a PLP. Then $LM(P) = T_P \uparrow \omega$. [6]*

The disjunction in the heads of rules causes non-determinism. In general, there exist more ways how to satisfy the head of a disjunctive rule. Therefore the result of the non-deterministic immediate consequence operator [7] is not only one interpretation, but a set of interpretations.

Definition 2 (Non-deterministic Immediate Consequence Operator). *Let I be an interpretation and P be a PDLP. A* non-deterministic immediate consequence operator *N_P is defined as follows:*

$$N_P(I) = MM(\{head(r) \mid r \in P, I \models body(r)\})$$

If we consider only positive logic programs, N_P is a generalization of T_P in the sense that $N_P(I) = \{T_P(I)\}$. Like in the case of positive logic programs, minimal models of positive disjunctive logic programs coincide with minimal fixpoints of N_P. However, minimal models can not be computed by straightforward iteration of N_P.

Proposition 3. *An interpretation I is a minimal model of a PDLP P iff I is a minimal fixpoint of N_P, i.e. I is a fixpoint N_P ($I \in N_P(I)$) and there does not exists a fixpoint of N_P such that $J \subset I$. [7]*

Example 3. The interpretation $M = \{a, b, c\}$ is a minimal model of $P = \{a \vee b \vee c; b \leftarrow a; c \leftarrow b; a \leftarrow c\}$. We have

$$\begin{aligned} N_P(\emptyset) &= \{\{a\}, \{b\}, \{c\}\} \\ N_P(\{a\}) &= \{\{b\}\} \\ N_P(\{b\}) &= \{\{c\}\} \\ N_P(\{c\}) &= \{\{a\}\} \end{aligned}$$

By iterating the non-deterministic immediate consequence operator N_P we get three possible computations:

$$\begin{gathered} \emptyset, \{a\}, \{b\}, \{c\}, \{a\}, \{b\}, \{c\}, \ldots \\ \emptyset, \{b\}, \{c\}, \{a\}, \{b\}, \{c\}, \{a\}, \ldots \\ \emptyset, \{c\}, \{a\}, \{b\}, \{c\}, \{a\}, \{b\}, \ldots \end{gathered}$$

Any computation does not terminate in M. All of them alternate over interpretations $\{a\}$, $\{b\}$, and $\{c\}$. The operator N_P is not even monotone.

If we add the newly derived atoms to the previous instead of replacing them, we get three monotone computations:

$$\begin{gathered} \emptyset, \{a\}, \{a, b\}, \{a, b, c\} \\ \emptyset, \{b\}, \{b, c\}, \{a, b, c\} \\ \emptyset, \{c\}, \{a, c\}, \{a, b, c\} \end{gathered}$$

All of them terminate in the same model M.

Now we use AND/OR graphs to define something like the iteration of the non-deterministic immediate consequence operator.

4 AND-OR Graph

An *AND-OR graph* is a directed acyclic graph $G = (V, E)$ with three distinct kinds of nodes – AND-nodes, OR-nodes and terminals. Every node represents a problem to solve and every edge represents the problem decomposition. To solve an AND-node, all of its subproblems must be solved. To solve an OR-node, it is sufficient to solve one of its subproblems. The solution for a terminal is already known. A *solution* of a start node s is a minimal subgraph $G' = (V', E')$ of G such that

– $s \in V'$
– if $v \in V'$ is an AND-node then $(v, v') \in E'$ for all $(v, v') \in E$
– if $v \in V'$ is an OR-node then $(v, v') \in E'$ for one $(v, v') \in E$

An OR-node is an interpretation $I \subset M$ such that I is not a model of P. To model all rules from P, we choose a set of rules $R \subseteq P$ not satisfied by I. In general, there are many possibilities how to choose R. I is an OR-node, because it is sufficient to choose only one set R to check the minimality of M.

An AND-node is a pair (I, R). The set $N_R(I)$ contains all minimal interpretations J we need to add to I to satisfy R. (I, R) is an AND-node, because to check if M is minimal we need test all interpretations $I \cup J$ such that $J \in N_R(I)$, $J \subseteq M$.

A terminal is an interpretation $I \subseteq M$ such that I is a model of P. If $I = M$, then all computations from I terminate in M. If $I \subset M$, then no computation from I terminates in M.

Definition 3 (AND-OR graph). *An* AND-OR graph for *a model M of a PDLP P contains*[1]

– AND-nodes $(I, R) \in \mathcal{P}(M) \times \mathcal{P}(P)$ *where* $R \subseteq \{r \in P \mid I \not\models r\}$,
– OR-nodes $I \in \mathcal{P}(M)$ *where* $I \not\models P$,
– *and* terminals $I \in \mathcal{P}(M)$ *where* $I \models P$.

There is an edge between

– *OR-node I and AND-node (I, R), and*
– *AND-node (I, R) and OR-node or terminal $I \cup J$ where $J \in N_R(I)$.*

A computation is an ordinal sequence of nodes. It is an extension of a path, because it can contain concatenation of infinite paths. It starts in an OR-node, following by alternating AND-nodes and OR-nodes and possibly terminates in a terminal node.

Definition 4 (Computation). *Let G be the AND-OR graph for a model M of a PDLP P. A* computation starting in *an interpretation $I \subseteq M$ is a (possibly infinite) sequence of nodes $I_0, (I_0, R_0), I_1, (I_1, R_1), \ldots$ from G such that I_α, $\alpha \geq 0$, are OR-nodes or terminals, (I_α, R_α), $\alpha \geq 0$, are AND-nodes, and*

[1] $\mathcal{P}(.)$ denotes the power set

– *$I_0 = I$*
– *if α is a successor ordinal of β, then (I_β, R_β) is connected to I_α*
– *if α is a limit ordinal, then $I_\alpha = \bigcup_{\beta<\alpha} I_\beta$*

We say that a computation $I_0, (I_0, R_0), \ldots, I_\gamma$ terminates in *M (*after at most δ steps*) iff $I_\gamma = M$ (and $\gamma \leq \delta$).*

A derivation is a subgraph. Like a computation extends a path, a derivation extends a solution. An OR-node is the source for just one edge, an AND-node is the source for all edges in the original AND-OR graph. Terminals are not sources for any edge.

Definition 5 (Derivation). *Let $G = (V, E)$ be the AND-OR graph for a model M of a DLP P. A* derivation starting in *an interpretation $I \subseteq M$ is a minimal subgraph $G' = (V', E')$ of G such that*

– *$I \in V'$*
– *if $v \in V'$ is an AND-node, then $(v, v') \in E'$ for all $(v, v') \in E$*
– *if $v \in V'$ is an OR-node, then $(v, v') \in E'$ for one $(v, v') \in E$*
– *V' is* closed, *i.e. if $I_0, (I_0, R_0), I_1, (I_1, R_1), \ldots$ is a computation in G', then $\bigcup_{\alpha\geq 0} I_\alpha \in V'$*

We say that a derivation terminates in *M (*after at most δ steps*) iff every maximal computation in G' terminates in M (after at most δ steps).*

Example 4. The following graph is the AND-OR graph for the model $M = \{a, b, c\}$ of the program $P = \{a \vee b \vee c; b \leftarrow a; c \leftarrow b; a \leftarrow c\}$.

$$\begin{tikzcd}
 & & & {\{a,b,c\}} & & \\
 & {(\{a,b\},\{c\leftarrow b\})} \arrow[urr] & & {(\{b,c\},\{a\leftarrow c\})} \arrow[u] & & {(\{a,c\},\{b\leftarrow a\})} \arrow[ull] \\
{(\{a,b\},\emptyset)} \arrow[r, shift left] & {\{a,b\}} \arrow[l, shift left] \arrow[u] & {(\{b,c\},\emptyset)} \arrow[r, shift left] & {\{b,c\}} \arrow[l, shift left] \arrow[u] & {(\{a,c\},\emptyset)} \arrow[r, shift left] & {\{a,c\}} \arrow[l, shift left] \arrow[u] \\
 & {(\{a\},\{b\leftarrow a\})} \arrow[u] & & {(\{b\},\{c\leftarrow b\})} \arrow[u] & & {(\{c\},\{a\leftarrow c\})} \arrow[u] \\
{(\{a\},\emptyset)} \arrow[r, shift left] & {\{a\}} \arrow[l, shift left] \arrow[u] & {(\{b\},\emptyset)} \arrow[r, shift left] & {\{b\}} \arrow[l, shift left] \arrow[u] & {(\{c\},\emptyset)} \arrow[r, shift left] & {\{c\}} \arrow[l, shift left] \arrow[u] \\
 & & & {(\emptyset,\{a\vee b\vee c\})} \arrow[ull] \arrow[u] \arrow[urr] & & \\
 & & {(\emptyset,\emptyset)} \arrow[r, shift left] & \emptyset \arrow[l, shift left] \arrow[u] & &
\end{tikzcd}$$

The following subgraph is a derivation starting in $\emptyset$ and terminating in M.

$$\begin{tikzcd}
 & \{a,b,c\} & \\
(\{a,b\},\{c \leftarrow b\}) \arrow[ur] & ((\{b,c\},\{a \leftarrow c\}) \arrow[u] & (\{a,c\},\{b \leftarrow a\}) \arrow[ul] \\
\{a,b\} \arrow[u] & \{b,c\} \arrow[u] & \{a,c\} \arrow[u] \\
(\{a\},\{b \leftarrow a\}) \arrow[u] & (\{b\},\{c \leftarrow b\}) \arrow[u] & (\{c\},\{a \leftarrow c\}) \arrow[u] \\
\{a\} \arrow[u] & \{b\} \arrow[u] & \{c\} \arrow[u] \\
 & (\emptyset,\{a \vee b \vee c\}) \arrow[ul] \arrow[u] \arrow[ur] & \\
 & \emptyset \arrow[u] &
\end{tikzcd}$$

In the previous derivation, we have three computations starting in $\emptyset$ and terminating in M:

$$\begin{array}{l}
\emptyset,\ (\emptyset,\{a \vee b \vee c\}),\ \{a\},\ (\{a\},\{b \leftarrow a\}),\ \{a,b\},\ (\{a,b\},\{c \leftarrow a\}),\ \{a,b,c\} \\
\emptyset,\ (\emptyset,\{a \vee b \vee c\}),\ \{b\},\ (\{b\},\{c \leftarrow b\}),\ \{b,c\},\ (\{b,c\},\{a \leftarrow b\}),\ \{a,b,c\} \\
\emptyset,\ (\emptyset,\{a \vee b \vee c\}),\ \{c\},\ (\{c\},\{a \leftarrow c\}),\ \{a,c\},\ (\{a,c\},\{b \leftarrow a\}),\ \{a,b,c\}
\end{array}$$

Now we show that the existence of a derivation starting in $\emptyset$ and terminating in a model M ensures the minimality of M.

Proposition 4. *Let G be the AND-OR graph for a model M of a PDLP P. Then M is a minimal model of P if there exists a derivation starting in $\emptyset$ and terminating in M.*

Proof. Let G' be a derivation starting in $\emptyset$ and terminating in M. Let $N \subseteq M$ be a model of P. We show that there exists a computation $I_0, (I_0, R_0), \ldots, I_\gamma$ in G' starting in $\emptyset$ and terminating in M such that $I_\alpha \subseteq N$, $0 \leq \alpha \leq \gamma$, i.e. $M = I_\gamma \subseteq N$.

For $\alpha = 0$ we have $I_\alpha = \emptyset \subseteq N$. Let α be a successor ordinal of β and $I_\beta \subseteq N$. Let (I_β, R_β) be the AND-node in G'. Because N is a model of P, N is also a model R_β. Therefore there exists an interpretation $J \in N_{R_\beta}(I_\beta)$ such that $J \subseteq N$ and $I_\alpha = I_\beta \cup J \subseteq N$. Let α be a limit ordinal and for all $\beta < \alpha$ holds $I_\beta \subseteq N$. Then $I_\alpha = \bigcup_{\beta<\alpha} I_\beta \subseteq N$. $\square$

Proposition 5. *Let G be an AND-OR graph for a minimal model M of a PDLP P. Then for all $I \subseteq M$ there exists a derivation in G starting in I and terminating in M after at most ω steps.*

Proof. Let $I \subseteq M$ and G' be a derivation starting in I such that $R = \{r \in P \mid J \not\models r\}$ for all AND-nodes (J, R). Because M is minimal, G' terminates in M. Now we show that G' terminates in M after at most ω steps.

Let $I_0, (I_1, R_1), \ldots, I_\gamma$ be a computation in G' starting in I and terminating in M. Let $\alpha \leq \gamma$ be a sucessor ordinal of β and $J_\beta \in N_{R_\beta}(I_\beta)$ be an interpretation such that $I_\alpha = I_\beta \cup J_\beta$. Then for all atoms $A \in J_\beta$ there exists a rule $r \in R_\beta$ such that r strongly supports A with respect to I_α. Let $Q_\beta = \{A \leftarrow \mathit{body}(r) \mid r \in R_\beta \text{ strongly supports } A \in J_\beta \text{ with respect to } I_\alpha\}$ and $Q = I_0 \cup \bigcup_{\beta<\gamma} Q_\beta$. It holds that $I_\alpha = T_Q(I_\beta)$. Because T_Q is a continuous operator [8] and $I_\gamma = M$, $\gamma \leq \omega$. $\square$

5 Well-Supported Models

The well-supported model semantics [4] uses level mappings. A level of an atom expresses the level of its derivation. Every rule well-supporting an atom also supports this atom. In addition, the assumptions of the rule must be derived before the conclusion of the rule. It has been shown that well-supported models of normal logic programs coincide with stable models.

Definition 6 (Well-Support). *Let I be an interpretation and P be a NLP. An atom $A \in I$ is* well-supported *by a rule $r \in P$ with respect to a level mapping ℓ iff A is supported by r and*

$$\forall L \in \mathit{body}(r)\colon \ell(A) > \ell(L)$$

We say that I is well-supported by P with respect to ℓ *iff every atom $A \in I$ is well-supported by a rule $r \in P$ with respect to ℓ. A model I of P is* well-supported *if there exists a level mapping ℓ such that I is well-supported by P with respect to ℓ.*

Proposition 6. *Let M be a model of a NLP P. Then M is stable iff M is well-supported. [4]*

A DLP P in a language $\mathcal{L}$ can be viewed as a PDLP P^* in the propositional language $\mathcal{L}^* = \mathcal{L} \cup \{\mathit{not}\, A \mid A \in \mathcal{L}\}$. For a given interpretation I in $\mathcal{L}$, by I^- we will denote the set of all true default literals, i.e. $I^- = \{\mathit{not}\, A \mid A \in \mathcal{L}, A \notin I\}$. The interpretation $I^* = I \cup I^-$ in $\mathcal{L}^*$ corresponds to the interpretation I in $\mathcal{L}$.

Definition 7 (Well-Support). *Let M be a model of a DLP P and G be the AND/OR graph for a model M^* of $P^* \cup M^-$. We say that M is well-supported iff there exists a derivation starting in $\emptyset$ and terminating in M^*.*

Proposition 7. *Let M be an interpretation and P be a DLP. Then M is a stable model of P iff M^* is a minimal model of $P^* \cup M^-$.*

Proof. Let M be a stable model of P. Then M is a minimal model of P^M. It is easy to see that M^* is a model of $P^* \cup M^-$. Let $N \subset M^*$ be a model of $P^* \cup M^-$.

Because $N \models M^-$, $N \setminus M^- \subset M$ models P^M. It is in the contradiction with the assumption that M is a minimal model of P^M.

Let M^* be a minimal model of $P^* \cup M^-$. It is easy to see that M is a model of P^M. Let $N \subset M$ be a model of P^M. Then $N \cup M^- \subset M^*$ is a model of $P^* \cup M^-$. It is in the contradiction with the assumption that M^* is a minimal model of $P^* \cup M^-$. Thus M is a stable model of P. $\square$

Corollary 1. *Let M be a model of a DLP P. Then M is stable iff M is well-supported.*

Weakly well-supporting rules may have more true atoms in the heads, but all of them are derived after assumptions. There is only one atom in the head which is derived first. It is the only atom which is weakly well-supported.

Definition 8 (Weak and Strong Well-Support). *Let I be an interpretation and P be a DLP. An atom $A \in I$ is* weakly (resp. strongly) well-supported *by a rule $r \in P$ with respect to a level mapping ℓ iff A is weakly (resp. strongly) supported by r and*

- $\forall L \in body(r)\colon \ell(A) > \ell(L)$
- $\forall B \in head(r)\colon I \models B, B \neq A \Rightarrow \ell(B) > \ell(A)$

We say that I is weakly (resp. strongly) well-supported by P with respect to ℓ *iff every atom $A \in I$ is well-supported by a rule $r \in P$ with respect to ℓ.*

Proposition 8. *Let M be a stable model of a DLP P and G be the AND/OR graph for $P^* \cup M^-$ and M^*. Let $J_0, (J_0, R_0), J_1, (J_1, R_1), \ldots, I_\gamma$ be a computation starting in $\emptyset$ and terminating in M^*. Let ℓ be a level mapping such that $\ell(L) = \min\{\alpha \mid L \in J_\alpha\}$ for all literals L. Then M is weakly well-supported by P with respect to ℓ.*

Proof. Let $A \in M$ and $\alpha = \ell(A)$ be a successor ordinal of β. Then there exists a rule $r \in I_\beta$ such that r strongly supports A with respect to I_α, i.e. r weakly well-supports A with respect to M and ℓ. So M is weakly well-supported by P with respect to ℓ. $\square$

6 Conclusions

We used non-deterministic immediate consequence operator to define one step of derivation of minimal models. We defined AND/OR graphs for models of positive disjunctive logic programs and showed that the model M is minimal iff there exists a derivation starting in $\emptyset$ and terminating in M after at most ω steps. We used also AND/OR graphs for defining well-supported models of disjunctive logic programs and showed that they coincide with stable models.

References

1. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the 5th International Conference on Logic Programming, ICLP'88, MIT Press (1988) 1070–1080
2. Przymusinski, T.C.: Stable semantics for disjunctive programs. New Generation Computing **9**(3/4) (1991) 401–424
3. Apt, K.R., Blair, H.A., Walker, A.: Towards a theory of declarative knowledge. In: Foundations of deductive databases and logic programming. Morgan Kaufmann Publishers Inc. (1988) 89–148
4. Fages, F.: A new fixpoint semantics for general logic programs compared with the well-founded and the stable model semantics. In: Proceedings of the 7th International Conference on Logic Programming, ICLP'90, MIT Press (1990) 442–458
5. Brass, S., Dix, J.: Characterizations of the stable semantics by partial evaluation. In: Proceedings of the 3rd International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR'95, Springer-Verlag (1995) 85–98
6. van Emden, M.H., Kowalski, R.A.: The semantics of predicate logic as a programming language. Journal of the ACM **23**(4) (1976) 733–742
7. Pelov, N., Truszczyński, M.: Semantics of disjunctive programs with monotone aggregates – an operator-based approach. In: 10th International Workshop on Non-Monotonic Reasoning, NMR 2004. (2004) 327–334
8. Lloyd, J.W.: Foundations of logic programming. Springer-Verlag New York, Inc. (1984)