TPLP **26** (1-2): 283–301, 2026. © The Author(s), 2026. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution-NoDerivatives licence (https://creativecommons.org/licenses/by-nd/4.0/), which permits re-use, distribution, and reproduction in any medium, provided that no alterations are made and the original article is properly cited.

doi:10.1017/S1471068426100660

Diamonds Are Forever: Stabilization Semantics for Unrestricted Aggregation and Recursion in Logica

EVGENY SKVORTSOV
Google LLC, USA
(*e-mail:* evgskv@google.com)

YILIN XIA
University of Illinois, Urbana-Champaign, USA
(*e-mail:* yilinx2@illinois.edu)

OJASWA GARG
Google LLC, USA
(*e-mail:* ojaswagarg@google.com)

SHAWN BOWERS
Gonzaga University, USA
(*e-mail:* bowers@gonzaga.edu)

BERTRAM LUDÄSCHER
University of Illinois, Urbana-Champaign, USA
(*e-mail:* ludaesch@illinois.edu)

Abstract

Logica is an open-source logic programming language that compiles to SQL and runs on platforms including DuckDB, SQLite, PostgreSQL, and BigQuery. Unlike classic Datalog, Logica permits the free combination of recursion and aggregation, enabling concise formulations of algorithms from shortest paths to PageRank. This expressiveness introduces fundamental semantic challenges: aggregate predicates are updated by replacement rather than accumulation, evaluation is sensitive to rule scheduling, and programs may converge to meaningful results without ever reaching a fixpoint, placing them outside the scope of traditional fixpoint semantics. We address these challenges with *Defendant-Opponent (DO) semantics*, a stabilization-based framework for nonmonotonic logic programs. Program evaluation is modeled as a rewrite system over derivation states. A ground atom is true if, from every reachable state, there exists a continuation after which the atom persists in all further derivations. This notion admits two equivalent characterizations: (1) game-theoretically, truth is what a Defendant can defend against any Opponent in a three-turn derivation game; and (2) modally, a formula t is a theorem precisely when the condition $\Box\Diamond\Box\, t$ holds in the derivation graph viewed as a Kripke structure. We show that DO semantics coincides with classic least fixpoint semantics for positive Datalog programs and is compatible with both the Well-Founded Semantics (the two never disagree on definite

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

answers) and the Stable Model Semantics (every stable model is a DO interpretation). For programs that converge without reaching a fixpoint, we introduce ω-limit interpretations, giving rigorous meaning to iterative computations such as PageRank. DO semantics thus offers a coherent framework that complements existing logic programming semantics while supporting recursive aggregation.

KEYWORDS: Logica, Datalog, nonmonotonic reasoning, aggregation, modal logic

1 Introduction

Logic programming has long served as a cornerstone of declarative computing. Datalog became particularly influential due to its elegant formal semantics grounded in monotonic least fixpoint computations (van Emden and Kowalski 1976; Ceri *et al.* 1990), enabling concise recursive queries such as transitive closure.

Extending Datalog with negation led to fundamental semantic challenges. Well-Founded Semantics (WFS) (Van Gelder *et al.* 1991; Van Gelder 1993) uses three-valued logic and an alternating fixpoint construction to handle nonmonotonic negation. Stable Model Semantics (SMS) (Gelfond and Lifschitz 1988) takes an alternative approach through self-supporting fixpoints, enabling multiple stable interpretations and forming the basis of Answer Set Programming (Gelfond 2008; Gebser *et al.* 2019).

Integrating aggregation into recursive logic programming brought further challenges (Mumick *et al.* 1990; Ross and Sagiv 1997). Work on stratified aggregates (Kemp and Stuckey 1991), recursive aggregation (Zaniolo *et al.* 2017), unified founded semantics (Liu and Stoller 2022), and recursive aggregates in ASP (Faber *et al.* 2004) has made significant progress, but has primarily focused on programs that reach fixpoints. The distinctive challenge lies with programs that *converge without reaching a fixpoint*.

PageRank (Brin and Page 1998) is a canonical example: iteration quickly approaches the eigenvector distribution, yet formally no fixpoint is attained. Such programs fall outside the scope of fixpoint-based semantics.

Logica and the semantic gap. Logica (Skvortsov *et al.* 2024a,b), an open-source logic programming language developed at Google, compiles to standard SQL and runs on platforms including DuckDB, SQLite, PostgreSQL, and BigQuery (Skvortsov 2023). Unlike classic Datalog, Logica allows unrestricted combinations of recursion and aggregation, addressing practical needs in modern data analytics. This exposes a semantic gap: programs like PageRank are expressible and useful, yet lack formal meaning under traditional fixpoint semantics.

DO semantics. We introduce *Defendant-Opponent (DO) semantics*, a framework that generalizes classical logic programming semantics to address these limitations. Inspired by Hilbert-style formal systems but extended for nonmonotonic programs, DO semantics defines truth through an adversarial interaction between a *Defendant* (asserting statements) and an *Opponent* (challenging assertions), giving meaning to programs with arbitrary recursion, aggregation, numerical computation, and non-stratified negation.

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

DO semantics also has a modal logic characterization. In monotonic systems, a formula t is a theorem if it is derivable from axioms A, that is $A \Vdash \Diamond t$. In nonmonotonic systems where derived facts can be retracted, mere derivability is insufficient. We show that DO-truth corresponds to the modal formula $A \Vdash \Box\Diamond\Box\, t$: from any state, it is *always possible* ($\Box\Diamond$) to reach a state where t holds, and once reached, t *necessarily persists* ($\Box$) under further derivation. This places nonmonotonic reasoning within S4 modal logic, where the seven distinct modalities provide a complete taxonomy of formula stability.

DO semantics bears a family resemblance to dialogue games for intuitionistic and classical logic (Lorenzen 1961), where truth emerges from a winning strategy in a two-player debate. However, the frameworks differ fundamentally. In Lorenzen games, players argue about the structure of a single formula through attack and defense moves on connectives. In DO semantics, the players jointly construct a *derivation* as a trajectory through the space of derivation states. The Opponent challenges not internal formula structure, but the *robustness* of derivability: can the Defendant reach a state containing the target atom that remains stable under any further moves? This shift from formula-structure to derivation-trajectory games is what allows DO semantics to handle nonmonotonic reasoning.

Contributions. The main contributions of this paper are:

1. A stabilization-based semantic framework (DO semantics) for nonmonotonic logic programs with unrestricted recursion and aggregation, with equivalent game-theoretic and modal ($\Box\Diamond\Box$) characterizations.
2. Backward compatibility: DO semantics coincides with classical least fixpoint semantics for positive Datalog (Corollary 1).
3. Compatibility with WFS (Theorem 5) and SMS (Theorem 6): the three semantics never disagree on definite answers, and every stable model is a DO interpretation.
4. ω-limit interpretations that give rigorous meaning to convergent programs such as PageRank (Theorems 3 and 4).

Outline. Section 2 introduces Logica through examples. Section 3 develops DO semantics, from monotonic formal systems to nonmonotonic rewriting rules, the Thesis Defense game, and modal logic. Section 4 introduces static, dramatic, and ω-limit interpretations. Section 5 establishes compatibility with WFS and SMS.

2 Logica

This section introduces the fragment of Logica relevant to this paper. Many features (named arguments, meta-programming, database integration) are practical conveniences that do not require the semantic treatment we develop here. A complete formal syntax of this fragment, together with a glossary of the key terms used throughout, is provided in the online supplementary material.

In Logica, predicates begin with capital letters, variables begin with lowercase letters, and rules end with semicolons.

```
Sibling(a,b) :- Parent(x,a), Parent(x,b), a!=b ;
Ancestor(a,b) :- Parent(a,b);
Ancestor(a,b) :- Ancestor(a,x), Ancestor(x,b);
```

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

Logica supports functional notation: $P(x) = y$ is syntactic sugar for the relation $P(x, y)$ where the last column is viewed as the output.

```
Square(x) = x * x;
Hypotenuse(a,b) = Sqrt(Square(a) + Square(b));
```

The central extension relevant to this paper is aggregation, which comes in two forms. *Aggregating expressions* appear within rule bodies (similar to ASP syntax Gelfond (2008); Gebser *et al.* (2019)):

```
HighDegree(x) :- E(x,y), Count{z :- E(x,z)} > 5;
```

Predicate-level aggregation resembles SQL's `GROUP BY` and aggregates over all derivations of a predicate's value. In this paper, we focus on Logica's predicate-level function value aggregation:

Definition 1.
A rule with predicate-level function value aggregation *has the form*

$$P(t_1, \ldots, t_n) \odot\!= e \texttt{ :- } Body(t_1, \ldots, t_n, t_{n+1}, \ldots, t_m, e)$$

where $\odot$ *is an aggregation operator that maps finite multisets to a single value; for each tuple* $(t_1, \ldots, t_n)$, $\odot$ *aggregates the values of* e *over all instantiations satisfying Body.*

In practice, efficient aggregation requires certain algebraic properties (namely, an Abelian semigroup operator), but this is irrelevant for the semantic treatment; we use `Sum`, `Min`, and `Max` as running examples.

As a simple example, the shortest distances between all pairs of vertices in a graph can be computed in Logica as follows:

```
D(a,b) Min= E(a,b);
D(a,b) Min= D(a,x) + D(x,b);
```

For each pair (a, b), the two rules define a multiset of candidate distances: the direct edge weight $E(a, b)$ if it exists, together with sums $D(a, x) + D(x, b)$ over all intermediate vertices x. The `Min=` operator returns the minimum of this multiset, which is assigned as the value of D. As we see in Section 3.4, the computation of the distance amounts to iterated application of these rules.

PageRank (Brin and Page 1998) provides a more challenging example:

```
PageRank(x) += ResetProb() / N() :- Page(x);
PageRank(y) += (1.0 - ResetProb()) * PageRank(x) / Degree(x) :- Link(x,y);
```

Iteration converges to the eigenvector distribution, yet formally never reaches a fixpoint: each step refines the approximation without stabilizing. Traditional fixpoint semantics cannot assign meaning to this program, which is precisely the gap that DO semantics fills.

Logica can also express numerical computations. The following approximates π using the Leibniz formula:

```
L() = 1 :- L = nil;
L() = L() + 1;
Pi() = 4 * Sum{(-1) ^ k / (2 * k + 1) :- k in Range(L())};
```

The `in` operator iterates over a collection. The following computes sine and cosine via Euler's method for differential equations.

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

```
Infinitesimal() = 1 / L();
C(0) = 1;
S(0) = 0;
S(t + dt) = S(t) + C(t) * dt :- dt = Infinitesimal();
C(t + dt) = C(t) - S(t) * dt :- dt = Infinitesimal();
```

DO semantics makes this precise through ω-limit interpretations (Section 4).

Finally, negation in Logica is encoded via aggregation: `not` $P(\vec{x})$ becomes `Count{1 :-` $P(\vec{x})$`}` $=$ `0`, which holds precisely when $P(\vec{x})$ has no derivation in the current state. This reduces negation to aggregation, requiring no additional semantic machinery.

3 Defendant-Opponent semantics

We define DO semantics by generalizing Hilbert-style formal systems to nonmonotonic logic. We begin by recalling the standard monotonic case.

3.1 Monotonic formal systems

Definition 2.
A formal system *is a triple* (H, A, R)*, where* H *is a set of* well-formed formulas*,* $A \subseteq H$ *is a set of axioms, and* R *is a set of inference rules. Each rule* $r \in R$ *is applied to a tuple* $(t_1, \ldots, t_n)$ *of well-formed formulas and returns a well-formed formula* $r(t_1, \ldots, t_n)$ *that follows from the rule, that is* $r : H^n \to H$*.*

In logic programming, H corresponds to the Herbrand base, that is the set of all potentially derivable ground atoms.

Definition 3.
For a formal system (H, A, R)*, a sequence* $T_0, \ldots, T_m$ *of sets of well-formed formulas* $T_i \subseteq H$ *is called a* derivation *if: (1)* $T_0 = A$*; and (2) for each* i *there exists some rule* $r \in R$ *of arity* n *and some* $\vec{t} = (t_1, \ldots, t_n)$ *with all* $t_j \in T_i$ *such that* $T_{i+1} = T_i \cup \{r(\vec{t})\}$*.*

Monotonic derivations are *inflationary* (Vianu 2021): each step only *adds* formulas, so the sets T_i form a growing chain converging toward a least fixpoint.

Definition 4.
The set of theorems T *of a formal system* (H, A, R) *is the set of all formulas that occur in some derivation:*

$$T = \bigcup_{\substack{T_0,\ldots,T_m \\ \text{is a derivation}}} \bigcup_{i=0}^{m} T_i.$$

It is well known that for finite systems, this set of theorems coincides with the least fixpoint of the immediate consequence operator (van Emden and Kowalski 1976; Abiteboul *et al.* 1995), providing the classic semantics for Datalog.

3.2 Nonmonotonic formal systems

In many practical situations, derived formulas must be *retracted*. For example, when computing shortest paths, we may derive distance k between two vertices, then find a shorter path of length $k' < k$ and want to replace the old value.

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

Various approaches model such retractions, for example via deletion rules (Vianu 2021). Here, we consider a *noninflationary* approach where rules rewrite the entire state at each step (similar to noninflationary *while* queries (Abiteboul *et al.* 1995)).

Definition 5.
A nonmonotonic formal system *is a triple* (H, A, W), *where* H *is a set of well-formed formulas,* $A \subseteq H$ *is a set of axioms, and* W *is a set of rewrite rules. Each rewrite rule* $w \in W$ *maps sets of formulas to sets of formulas:* $w : 2^H \to 2^H$.

Definition 6.
For a nonmonotonic formal system (H, A, W), *a sequence* $T_0, \ldots, T_m$ *with* $T_i \subseteq H$ *is called a* derivation *if* $T_0 = A$ *and* $T_{i+1} = w(T_i)$ *for some* $w \in W$.

It is not immediately clear what the theorems of such a system should be, since derived formulas may be introduced and subsequently retracted as computation proceeds. We define truth based on the *eventual behavior* of derivations.

3.3 Thesis and truth

Rather than limiting attention to programs with fixpoints, we consider a semantics based on a *thesis*: a claim about the eventual behavior of derivations.

Definition 7.
Given a nonmonotonic formal system (H, A, W), *a* thesis *is a set* $M \subseteq 2^H$ *of sets of formulas such that a set of formulas* $T \subseteq H$ *is* consistent *with* M *if* $T \in M$, *and otherwise is* inconsistent *with* M.

Intuitively, a thesis M represents a claim such as "a deep enough derivation will arrive at a state in M." To make this precise, we introduce the following graph-theoretic notions.

Definition 8.
Given a nonmonotonic formal system (H, A, W), *a set* $T \subseteq H$ *is a* state of derivation *if there exists a derivation of* T, *that is* $T_0, \ldots, T_{m-1}, T$.

Derivation states form a derivation graph.

Definition 9.
The graph of derivation $G(H, A, W)$ *is a directed graph whose vertices are states of derivation and whose edges are pairs* (T, T') *such that* $T' = w(T)$ *for some* $w \in W$.

A state T' is *reachable* from T if $T = T'$ or there is a finite sequence of edges in G from T to T'; that is, reachability is the reflexive–transitive closure of the edge relation. Truth is defined over this graph.

Definition 10.
A thesis M holds *(equivalently:* M *is* true, M *is a* truth*) in a nonmonotonic formal system* (H, A, W) *with derivation graph* G *if for any derivation state* T_i *there exists a derivation state* $T_j \in M$ *such that: (1)* T_j *is reachable from* T_i; *and (2) for every derivation state* T_k *reachable from* T_j, $T_k \in M$.

Thus M holds if no matter where you are in the derivation, you can steer toward a state in M, and once there, any further derivation stays within M. Truth permits *stabilization*:

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

derivation can always be continued to achieve and maintain consistency with the true thesis.

Definition 11.
A formula $t \in H$ is a theorem *in (H, A, W) if the thesis $\{T \subseteq H \mid t \in T\}$ holds. A formula t is an* anti-theorem *if the thesis $\{T \subseteq H \mid t \notin T\}$ holds. A formula that is either a theorem or an anti-theorem is called* definite.

Note that a formula may be neither a theorem nor an anti-theorem, that is its status may be genuinely undetermined. It also follows that no formula can be both a theorem and an anti-theorem: if both held, then by Lemma 1 the empty thesis $\{T \mid t \in T\} \cap \{T \mid t \notin T\} = \emptyset$ would be a truth, which is impossible since no derivation state lies in $\emptyset$.

3.4 Logica as a nonmonotonic formal system

We now define how a Logica program gives rise to a nonmonotonic formal system.

Definition 12.
Given a Logica program $\mathcal{P}$ over a database (EDB) D, the corresponding nonmonotonic formal system *(H, A, W) is defined as follows:*

- *H, the Herbrand base, is the set of all ground atoms of the form $P(v_1, \ldots, v_n)$ for each predicate P in $\mathcal{P}$ and all possible values. For functional predicates, we assume the last argument v_n is the value argument.*
- *$A = D$, the extensional database.*
- *$W = \{w_P \mid P$ is an intensional predicate in $\mathcal{P}\}$, where each w_P is a rewriting rule that recomputes all values of predicate P based on the current state. Formally, $w_P(T) = (T \setminus \{P(\vec{v}) \in T\}) \cup eval_P(T)$, where $eval_P(T)$ evaluates all rules defining P against T (snapshot semantics): body literals are looked up in T as-is, without recomputing other predicates. All non-P atoms, including EDB facts, are unchanged.*

A key observation is that W contains one rewriting rule per *predicate*, not per syntactic rule. Consider again the program to compute shortest path distances `D`.

```
D(a,b) Min= E(a,b);
D(a,b) Min= D(a,x) + D(x,b);
```

The two clauses in this case constitute a single rewriting rule w_D that computes the minimum over all derivable values, both direct edges and composite paths. This design reflects a fundamental feature of aggregation: when w_D is applied, it replaces the *entire* predicate D with newly computed values. If the current state contains `D("a","b") = 42` from an earlier derivation, but a shorter path of length 37 is now derivable, the new state will contain `D("a","b") = 37`, where the old value is *replaced*, not accumulated. This is precisely what makes the system nonmonotonic.

Each edge in the derivation graph (Definition 9) corresponds to applying one rewrite rule $w_P \in W$. At each step, we choose *which predicate* to recompute, introducing nondeterminism. For single-predicate programs like shortest paths, only one rule applies. For programs with multiple interdependent predicates, different choices lead to different derivation graph trajectories.

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

3.5 The thesis defense game

Truth admits a game-theoretic characterization that gives DO semantics its name.

Definition 13.
The Thesis Defense game *for a nonmonotonic formal system* (H, A, W) *and thesis* M *is a two-player game between a* Defendant *and an* Opponent. *The game constructs a derivation in three turns that begins with the Opponent:*

- ***Turn 1 (Opponent):*** *Starting from* $T_0 = A$, *the Opponent extends the derivation to some state* T_{m_1}.
- ***Turn 2 (Defendant):*** *The Defendant continues from* T_{m_1} *to some state* T_{m_2}.
- ***Turn 3 (Opponent):*** *The Opponent continues from* T_{m_2} *to a final state* T_{m_3}.

In a single turn a player may apply any finite number of rewrite rules, that is move to any state reachable from the current one; the available move sets may be infinite. The Defendant wins if $T_{m_3} \in M$; *the Opponent wins otherwise.*

Theorem 1.
For a nonmonotonic formal system (H, A, W) *and thesis* M, *the following are equivalent:*

1. *Thesis* M *holds in* (H, A, W).
2. *The Defendant has a winning strategy.*
3. *The Opponent has no winning strategy.*

Equivalence of (2) and (3) holds by determinacy (the number of turns is fixed at three and there are no draws; the move sets may be infinite). Equivalence with (1) captures the essence of DO semantics: *truth is what you can defend*. The O-D-O turn order is essential: reversing it or reducing to two turns would break the closure of truths under intersection.

Lemma 1.
If M_1 *and* M_2 *are truths, then* $M_1 \cap M_2$ *is also a truth.*

Proof (Sketch).
Since M_1 is true, the Defendant has a strategy to reach a state $T_j \in M_1$ that is stable within M_1. From T_j, since M_2 is also true, the Defendant can further reach a state $T_k \in M_2$ that is stable within M_2. Because T_k is reachable from T_j and T_j is stable within M_1, we have $T_k \in M_1$ as well. Thus $T_k \in M_1 \cap M_2$, and any state reachable from T_k remains in both M_1 and M_2. □

3.6 Truth as modality

DO truth has a natural modal logic interpretation. The derivation graph is a Kripke frame, with derivation states as worlds and reachability as the accessibility relation.

Definition 14.
Let (H, A, W) *be a nonmonotonic formal system. A* modal statement *is defined inductively:*

- *For any formula* $h \in H$ *statements* h *and* $\neg h$ *are modal statements.*

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

- *If t is a modal statement, then $\Box t$ (read: "necessarily t") is a modal statement.*
- *If t is a modal statement, then $\Diamond t$ (read: "possibly t") is a modal statement.*

Definition 15.
Whether a modal statement t holds at a state of derivation T, written $T \Vdash t$, is defined inductively, where any state T is by definition considered reachable from itself:

- *$T \Vdash h$ (for $h \in H$) if and only if $h \in T$.*
- *$T \Vdash \neg h$ (for $h \in H$) if and only if $h \notin T$.*
- *$T \Vdash \Box t$ if and only if t holds at all states reachable from T.*
- *$T \Vdash \Diamond t$ if and only if t holds at some state reachable from T.*

We can now express DO-truth using a single modal-logic formula.

Observation 1.
In a nonmonotonic formal system (H, A, W), a formula $t \in H$ is a theorem iff $A \Vdash \Box\Diamond\Box\, t$.

Unpacking: from A, *necessarily* (regardless of the Opponent's first move), it is *possible* (the Defendant finds a way) to reach a state where t *necessarily* holds (persists under any further moves). The three alternations correspond to the three turns of the Thesis Defense game. The central $\Diamond$ is the diamond that guarantees forever: it ensures a path exists to a region where t is permanently stable. In the framework of Alpern and Schneider (1985), this stabilization condition decomposes into liveness (a stable region is reachable) and safety (t persists once reached). For monotonic systems, $A \Vdash \Diamond t$ suffices: once derived, a formula persists, so $\Diamond t$ implies $\Box\Diamond\Box\, t$.

Definition 16.
A nonmonotonic formal system (H, A, W) is monotonic *if every rewrite rule is inflationary, that is $T \subseteq w(T)$ for all $w \in W$ and all states T (equivalently, no atom is ever retracted). The monotonic formal systems considered earlier are the special case $w_r(T) = T \cup \{r(\vec{t})\}$.*

Observation 2.
In a monotonic formal system (H, A, W), for any $t \in H$, $A \Vdash \Diamond t$ iff $A \Vdash \Box\Diamond\Box\, t$.

In positive Datalog, derivation is monotonic: derived facts can only grow. The least fixpoint consists of atoms satisfying $A \Vdash \Diamond t$, which are exactly those satisfying $A \Vdash \Box\Diamond\Box\, t$.

Corollary 1.
For positive Datalog programs (without negation or nonmonotonic aggregation), DO semantics coincides with classical least-fixpoint semantics: a ground atom is a DO theorem iff it belongs to the least fixpoint.

The DO accessibility relation is reflexive and transitive, implying the modal logic S4. It is well known that in S4, arbitrary sequences of $\Box$ and $\Diamond$ collapse to just seven distinct modalities (Chellas 1980), as shown on the left in Figure 1. Figure 1 (right) interprets these modalities: t is *derivable* ($\Diamond t$) if reachable from the axioms; *provable* ($\Diamond\Box t$) if some reachable state contains t permanently; and *always derivable* ($\Box\Diamond t$) if from any state, a state with t can be reached. DO-theoremhood ($\Box\Diamond\Box$) is strictly stronger than each (but weaker than $\Box t$).

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

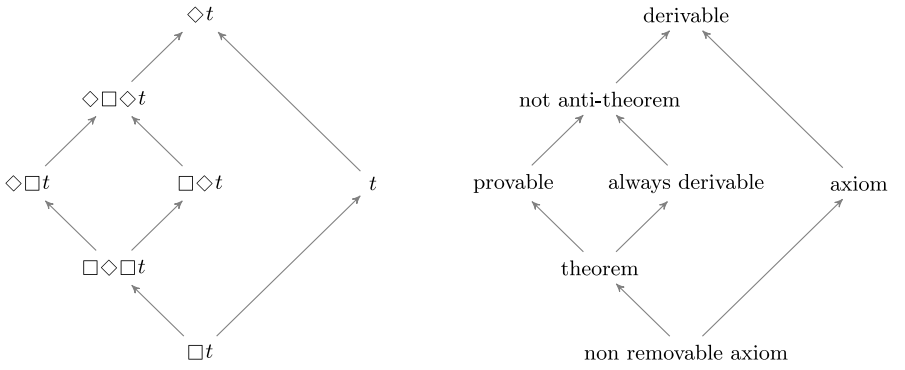


Fig. 1. The seven distinct S4 modalities (left, strongest $\Box t$ at bottom) with their DO-semantics interpretations (right). Here $\Box$ is "necessarily" and $\Diamond$ is "possibly" in the standard modal reading; the right side shows what each modality means for derivations. Arrows point upward in the direction of implication: a lower (stronger) modality entails a higher (weaker) one, for example, $\Box\Diamond\Box\, t$ (theorem) implies $\Box\Diamond\, t$ (always derivable). Note: "axiom" means t holds at the current state; "non-removable axiom" ($\Box\, t$) means t holds at every reachable state.

4 Interpretations in DO semantics

Having defined which theses hold, we turn to interpretation. In classic logic programming, an interpretation is a model consistent with all theorems. For nonmonotonic systems, we distinguish *static* and *dramatic* interpretations.

4.1 Static interpretations

Definition 17.
Let (H, A, W) be a nonmonotonic formal system and let $\mathcal{T}$ be the set of all truths, that is all theses that hold in it. If a state of derivation T belongs to the intersection of all truths $T \in \bigcap_{M\in\mathcal{T}} M$, then T is called a static interpretation *of the system.*

A static interpretation is a state of derivation consistent with *every* truth: a stable resting point respecting all defensible claims. For finite systems, such interpretations always exist.

Theorem 2.
Let (H, A, W) be a nonmonotonic formal system with a finite derivation graph. Then static interpretations are precisely the nodes of the derivation graph that belong to the terminal (sink) strongly connected components.

The finiteness assumption is essential: it ensures that every path in the derivation graph eventually reaches a terminal SCC. The proof follows from the observation that static interpretations are precisely the states T for which the Opponent has a strategy to end the game on T.

These terminal SCCs are the analog of *attractors* in Boolean-network theory; the related notion of a *trap space*, a region closed under the dynamics, captures the same persistence-under-derivation that DO expresses through the final $\Box$ of $\Box\Diamond\Box$. Trinh *et al.* (2025) establish this Boolean-network view of Datalog with negation (subset-minimal

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

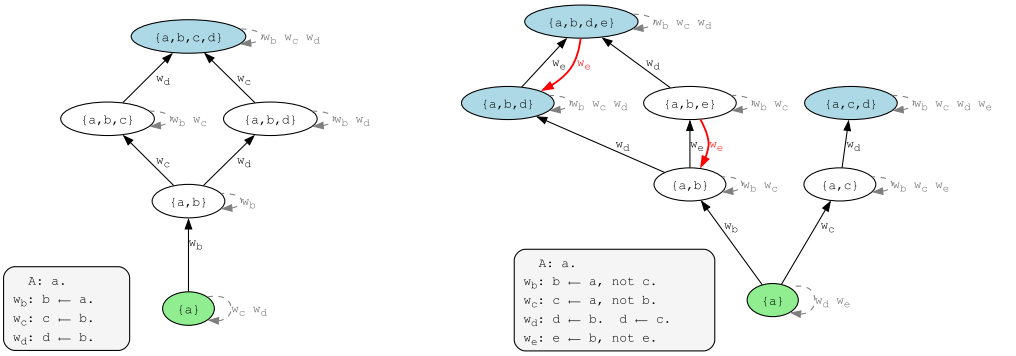


Fig. 2. Two logic programs and their derivation graphs. Monotonic program (left): derivation flows upward to the unique fixpoint $\{a, b, c, d\}$. Nonmonotonic program (right): $\mathtt{w_e}$ causes oscillation (red edges) between $\{a, b, d\}$ and $\{a, b, d, e\}$; $\{a, c, d\}$ is a separate fixpoint. Green nodes are axioms; blue nodes are static interpretations (sink SCC nodes). Edge labels identify rewrite rules; gray self-loops indicate rules that leave the state unchanged.

stable trap spaces correspond to regular models), while Trinh *et al.* (2026) develop a trap-space *semantics* for normal logic programs.

Definition 18.
A state of derivation $T \subseteq H$ is a fixpoint *if $w(T) = T$ for all rules $w \in W$.*

Observation 3.
Any fixpoint is a static interpretation.

While fixpoints are effectively sink states, static interpretations may not be fixpoints (but they are consistent with all truths). See Figure 2 for examples of static interpretations.

4.2 Dramatic interpretations

Static interpretations suffice when the system settles into a stable state. For programs like PageRank, where iteration converges *toward* a value without reaching it, we need a dynamic counterpart.

Definition 19.
A dramatic interpretation[1] *of (H, A, W) is an infinite sequence $T_1, T_2, \ldots$ of derivation states such that for any thesis M that holds, there is an n where $T_k \in M$ for all $k > n$.*

A dramatic interpretation eventually enters and stays within every true thesis. Note that consecutive states T_k, T_{k+1} need not be connected by a rewrite rule: a dramatic interpretation is any sequence of derivation states with this eventual-consistency property, not necessarily a path in the graph.

Observation 4.
If a dramatic interpretation $T_1, T_2, \ldots$ eventually stabilizes (i.e. $T_k = T_n$ for all $k > n$), then T_n is a static interpretation. Conversely, if T is a static interpretation, then the constant sequence $T, T, T, \ldots$ is a dramatic interpretation.

[1] "Dramatic" comes from the Ancient Greek word *δράω* referring to action, where a dramatic interpretation unfolds through action via the process of derivation itself.

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

Thus static interpretations are "finished" dramatic interpretations.

4.3 ω-limit interpretations

For real-valued computation (PageRank, differential equations), we want derivations to converge topologically. This requires equipping the state space with a topology. The nondeterministic scheduling in DO semantics is related to asynchronous iteration (Bertsekas and Tsitsiklis 1989), where convergence must hold regardless of update order.

Definition 20.
Let (H, A, W) be a nonmonotonic formal system and let $\mathcal{S} \subseteq 2^H$ be a set containing all states of derivation, equipped with a topology $\mathbb{T}$. A set $T \in \mathcal{S}$ is an ω-limit interpretation *if every open set containing T is a true thesis.*

For Logica programs with real-valued computation, we define a concrete topology.

Definition 21.
Given a Herbrand base H with arguments typed as strings, integers, or reals, we define a subset $\mathcal{S} \subseteq 2^H$ and a Hausdorff topology as follows. Let r be an injection from predicate symbols, strings, and integers into $\mathbb{N}$, extended to reals by the identity. Since argument positions do not mix types, r is injective across all values in any given position. For a ground atom $P(v_1, \ldots, v_n)$, let $\rho(P(v_1, \ldots, v_n)) = (r(P), r(v_1), \ldots, r(v_n), 0, \ldots, 0) \in \mathbb{R}^k$, padded to dimension k (the maximum predicate arity plus one). The space $\mathcal{S}$ consists of all compact *subsets of $\mathbb{R}^k$, equipped with the Hausdorff distance:*

$$d_H(T_1, T_2) = \max\left(\sup_{x \in T_1} \inf_{y \in T_2} d(\rho(x), \rho(y)), \sup_{y \in T_2} \inf_{x \in T_1} d(\rho(x), \rho(y))\right)$$

where d is the Euclidean distance. The Hausdorff topology *on $\mathcal{S}$ is the topology induced by this metric.*

Note that per Definition 21 the Herbrand base has arguments from $\mathbb{R}$ and is thus of the cardinality of the continuum, while numbers used in the definition of a logic program have only finitely many decimal digits. Thus, for instance, transcendental numbers (ones that are not roots of any polynomial with rational coefficients) never occur in states of derivation, because they cannot be computed in a finite number of steps. However, transcendental numbers can occur in the ω-limit, as we see in the program converging to π in Section 2.

All ω-limit interpretations in this paper use the Hausdorff topology. States of derivation are finite, hence compact, hence elements of $\mathcal{S}$. Since predicate symbols map to distinct natural numbers, facts from different predicates remain at distance ≥ 1, so each predicate's convergence can be analyzed independently.

Observation 5.
If an ω-limit interpretation exists, then it is unique and every dramatic interpretation converges to it. (This follows from the Hausdorff separation property.)

Theorem 3.
The PageRank program from Section 2 has an ω-limit interpretation, which assigns to each page its true PageRank value, that is the eigenvector of the link matrix.

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

Proof.
Applying $w_{PageRank}$ corresponds to multiplying the current PageRank vector by the transition matrix $M = (1-d)\frac{1}{n}\mathbf{1} + d \cdot L$, where d is the damping factor and L the column-normalized link matrix (assuming no dangling nodes, or with the standard dangling-mass correction, so that L is column-stochastic). Since M is stochastic with all positive entries, the Perron-Frobenius theorem (Meyer 2000) gives a unique eigenvector v^* with eigenvalue 1, and the power method converges to v^* from any starting distribution.

For any open neighborhood U of v^*, the Defendant can apply $w_{PageRank}$ sufficiently many times to enter U, and all subsequent iterations remain within U (they only get closer to v^*). Thus every neighborhood of v^* is a true thesis, making v^* an ω-limit interpretation. □

By a similar argument, the π-approximation program has an ω-limit interpretation with `Pi()` $= \pi$ (the Leibniz series converges, so the Defendant can increase L until the partial sum is within any ε of π).

Theorem 4.
The differential equation program for sine and cosine, restricted to $t \leq 2\pi$, has an ω-limit interpretation where `S` *and* `C` *equal the sine and cosine functions on* $[0, 2\pi]$.

Proof.
The program implements Euler's method for $S' = C$, $C' = -S$ with $S(0) = 0$, $C(0) = 1$, whose unique solution is $S = \sin$, $C = \cos$. Each rewriting rule is idempotent when its dependencies are unchanged, so no progress is lost by repeated application.

Let U be any open neighborhood of $(\sin, \cos)$ on $[0, 2\pi]$. Euler's method converges uniformly on compact sets (Atkinson *et al.* 2009): for any $\varepsilon > 0$, there exists L_0 such that step size $1/L_0$ stays within ε of the true solution.

The Defendant first applies w_L until $L = L_0$ (choosing ε so the ε-neighborhood lies inside U), then alternates w_S and w_C to compute the full solution on $\{0, 1/L_0, 2/L_0, \ldots\}$, reaching a state within U.

On the Opponent's final turn: applying w_S or w_C is idempotent (dependencies unchanged); applying w_L increments L to $L' > L_0$, which only *improves* accuracy, so the state remains within U. Thus every open neighborhood of $(\sin, \cos)$ is a true thesis. □

4.4 Irreducible truth

In general, static interpretations do not capture the full structure of truth in a nonmonotonic formal system. As discussed, a meaningful program (e.g. PageRank) may have no static interpretation. But even when a static interpretation exists, a distinct dramatic interpretation may exist alongside. The program below illustrates this (see Figure 3 for its derivation graph):

```
Left() :- not Right();
Right() :- not Left();
X() = 1 :- not X(), Right();
X() = -1 :- not X(), Left();
X() = X() + 1 :- X() > 0;
X() = X() :- X() < 0;
```

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

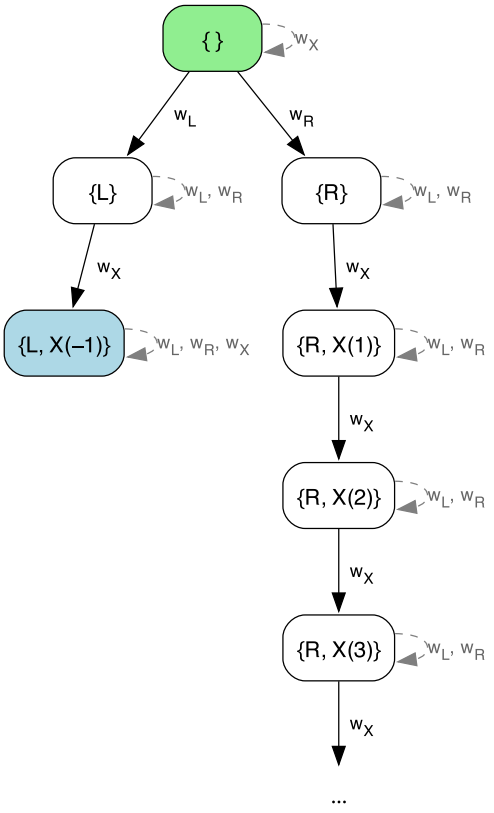


Fig. 3. Derivation graph for the Left/Right program (L and R abbreviate Left and Right). Green: axiom; blue: sole static interpretation (fixpoint). The left branch stabilizes; the right diverges (X grows without bound). Since the blue node is unreachable from the right branch, no irreducible truth exists. Dashed gray: self-loops.

If `Left` holds, then `X` stabilizes at -1, yielding a static interpretation $\{\texttt{Left}, \texttt{X}(-1)\}$. If `Right` holds, then `X` grows without bound $(1, 2, 3, \ldots)$ and no fixpoint is reached. Since derivation states in the `Right` branch cannot reach $\{\texttt{Left}, \texttt{X}(-1)\}$, the set of static interpretations is not reachable from every state and thus is not a truth.

For finite systems, however, static interpretations do characterize all truths. We formalize this via the notion of irreducible truth.

Definition 22.
Let (H, A, W) be a nonmonotonic formal system. If the set of all static interpretations is itself a truth, then it is called the irreducible truth *of the system.*

Observation 6.
If the irreducible truth exists, then a thesis holds if and only if it contains the irreducible truth, that is every static interpretation belongs to the thesis.

Proof.
Any thesis that contains a truth is itself a truth: if from every state one can reach and stay in M, then one can also reach and stay in any superset of M. Thus any thesis

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

containing the irreducible truth holds. Conversely, if a thesis holds, it must contain every static interpretation: a static interpretation belongs to every truth, hence to every thesis that holds. Therefore it contains the irreducible truth. □

By definition, whenever the irreducible truth exists, any static interpretation is an element of the irreducible truth. The following observation shows that dramatic interpretations are also characterized by the irreducible truth.

Observation 7.
If the irreducible truth exists, then dramatic interpretations are precisely sequences of derivation states $T_1, \ldots, T_k, \ldots$ *for which there exists* n *such that for any* $k > n$ *we have* T_k *being an element of the irreducible truth.*

Proof.
Consider a dramatic interpretation $T_1, \ldots, T_k, \ldots$. Since the irreducible truth is a truth, by definition of a dramatic interpretation there exists n such that states $T_{n+1}, \ldots$ belong to the irreducible truth. The converse also holds because a tail of elements of the irreducible truth is consistent with every truth. □

Observation 8.
If the derivation graph is finite, then the irreducible truth exists and equals the set of nodes in the terminal (sink) strongly connected components.

Proof.
In a finite derivation graph, every path eventually reaches a terminal SCC. Thus from any state, there exists a path to a node in a terminal SCC. Once in a terminal SCC, all further states remain within terminal SCCs, since there are no outgoing edges to non-terminal nodes. Therefore the set of terminal SCC nodes is a truth. By Theorem 2, this set is exactly the set of static interpretations, so it is the irreducible truth. □

5 Connections to classical nonmonotonic semantics

We now examine how DO semantics relates to WFS and SMS. The relationship is one of compatibility rather than equivalence: DO semantics partially aligns with WFS and SMS, but differs on programs that fall outside their intended scope. This asymmetry is by design. WFS and SMS are defined for programs that reach fixpoints, while DO semantics also assigns meaning to programs that converge without reaching one, such as PageRank, numerical approximation of π, and differential equations solved via Euler's method (Section 4.3).

The program and its derivation graph shown on the left in Figure 4 illustrate a key distinction of DO semantics. The rule $b \leftarrow a, \neg b$ is self-defeating: applying $\mathtt{w_b}$ alternately introduces and removes b. However, when b is present, $c \leftarrow b$ fires, and once derived, c persists via $c \leftarrow c$. WFS declares only a definite and leaves b and c undefined. SMS finds no stable model at all, since b's self-defeating nature prevents any self-consistent fixpoint. DO semantics, by contrast, identifies $\{a, c\}$ and $\{a, b, c\}$ as static interpretations and recognizes c as a theorem: from any state, derivation has a chance to reach the sink SCC where c is present in every state. This willingness to let rules fire and retain

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

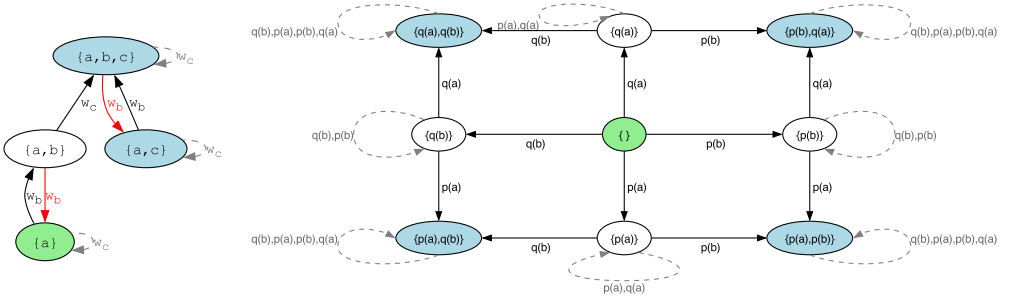


Fig. 4. Left: derivation graph for the program with rules $\mathtt{w_b}$: $\{\mathtt{b} \leftarrow \mathtt{a}, \neg\, \mathtt{b}\}$ and $\mathtt{w_c}$: $\{\mathtt{c} \leftarrow \mathtt{b};\ \mathtt{c} \leftarrow \mathtt{c}\}$. Right: Ground nonmonotonic system with rules $\mathtt{w_{p(a)}}$: $\{\mathtt{p(a)} \leftarrow \neg\mathtt{q(a)}\}$; $\mathtt{w_{p(b)}}$: $\{\mathtt{p(b)} \leftarrow \neg\mathtt{q(b)}\}$; $\mathtt{w_{q(a)}}$: $\{\mathtt{q(a)} \leftarrow \neg\mathtt{p(a)}\}$; and $\mathtt{w_{q(b)}}$: $\{\mathtt{q(b)} \leftarrow \neg\mathtt{p(b)}\}$. In both figures, green nodes represent axioms ($\{\mathtt{a}\}$ and $\emptyset$, respectively); blue nodes are static interpretations; dashed gray loops are rule applications that leave the state unchanged; red edges mark oscillation (caused by $\mathtt{w_b}$ on the left).

what stabilizes is precisely what allows DO semantics to give meaning to convergent computations like PageRank.

Logica's semantics (Definition 12) uses per-predicate rewrite rules, while established logic programming semantics are defined on ground programs. To connect the two, we define the ground nonmonotonic system of a logic program.

Definition 23.
Given a normal logic program P, the ground nonmonotonic system *of P is the nonmonotonic formal system (H, A, W_{gr}) where H is the Herbrand base of P, A is the set of facts of P, and W_{gr} contains one rewrite rule $w_{P(\vec{v})}$ per ground atom $P(\vec{v}) \in H$. Rule $w_{P(\vec{v})}$ ensures that $P(\vec{v})$ belongs to the resulting state iff some rule with head $P(\vec{v})$ has its body satisfied in the current state, leaving all other atoms unchanged.*

The ground nonmonotonic system is not a separate semantics: it instantiates the same nonmonotonic formal system with finer-grained, per-ground-atom rewrite rules, serving as a bridge to the ground structure assumed by WFS and SMS.

On the right of Figure 4 we show the derivation graph of a ground nonmonotonic system for a program with two predicates $\mathtt{p}$ and $\mathtt{q}$. The ground system has four static interpretations (blue nodes), including mixed states such as $\{p(a), q(b)\}$ that are unreachable under per-predicate rewrite rules.

5.1 Well-Founded Semantics

WFS (Van Gelder *et al.* 1991) assigns each ground atom to *true*, *false*, or *undefined* via an alternating fixpoint construction (Van Gelder 1993). The key observation is that the result of this construction can be reproduced as a derivation strategy within DO semantics, reaching a state T_{WFS} containing exactly the WFS-true atoms. From T_{WFS}, no further rule application can derive WFS-false atoms. In modal terms, T_{WFS} satisfies $\Diamond\Box\, t$ for every WFS-true atom t and $\Diamond\Box\, \neg f$ for every WFS-false atom f.

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

Example (Win-Move). Consider the Win-Move game (Van Gelder 1993):

```
Win(x) :- Move(x,y), not Lose(y);
Lose(x) :- Position(x), not Win(x);
```

Under Logica's DO semantics, derivation alternates between recomputing *Win* and *Lose* (each recomputation is idempotent). Since the derivation graph is finite, every infinite derivation enters a terminal SCC. Atoms (e.g. $Win(x)$) present in every state of every terminal SCC correspond to WFS-true atoms; atoms in none correspond to WFS-false; atoms that oscillate correspond to WFS-undefined.

Theorem 5.
Let P be a logic program and (H, A, W_{gr}) its ground nonmonotonic system. Let t be a ground atom.

- *If t is true under WFS, then t is not an anti-theorem in (H, A, W_{gr}).*
- *If t is false under WFS, then t is not a theorem in (H, A, W_{gr}).*

Proof (Sketch).
The alternating-fixpoint construction produces the WFS-true atoms as an increasing sequence $K_1 \subseteq K_2 \subseteq \cdots$: an atom enters K_{i+1} only when it is derivable using, positively, atoms already established true, and, negatively, only atoms already established *false* (atoms that can never become WFS-true). The Opponent's strategy is to fire the rewrite rule $w_{P(\vec{v})}$ for each WFS-true atom $P(\vec{v})$ in this order of first appearance (ties within a stage in any order). Negative conditions invoked at each stage refer only to already-falsified atoms, so the Opponent never relies on the absence of an atom that is in fact WFS-true. This reaches a state T_{WFS} containing exactly the WFS-true atoms. From T_{WFS} no rule application can introduce a WFS-false atom (its body would require a WFS-true atom to be absent or a WFS-false atom to be present, neither of which holds), and every WFS-true atom is self-supporting; hence every reachable state retains all WFS-true atoms and excludes all WFS-false atoms. Therefore no WFS-true atom is an anti-theorem and no WFS-false atom is a theorem. □

Theorem 5 does *not* claim WFS-true atoms are DO theorems. The relationship is compatibility: the two never contradict, but each may be definite where the other is not.

5.2 Stable Model Semantics

SMS (Gelfond and Lifschitz 1988) defines a stable model M as the least model of the reduct P^M and forms the basis of Answer Set Programming (Gelfond 2008).

Theorem 6.
Let P be a logic program and (H, A, W_{gr}) its ground nonmonotonic system. Every stable model of P is a static interpretation of (H, A, W_{gr}).

Proof (Sketch).
Let M be a stable model of P. We show that M is a fixpoint of (H, A, W_{gr}). For any ground atom $P(\vec{v})$, the rewrite rule $w_{P(\vec{v})}$ evaluates all rules with head $P(\vec{v})$ against M, with $not\,Q(\vec{x})$ true iff $Q(\vec{x}) \notin M$. This is exactly the evaluation under the reduct P^M. If $P(\vec{v}) \in M$, then since $M = lfp(T_{P^M})$, some rule body is satisfied, so $w_{P(\vec{v})}$ retains

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

$P(\vec{v})$. If $P(\vec{v}) \notin M$, then no rule body is satisfied, so $w_{P(\vec{v})}$ does not introduce $P(\vec{v})$. In both cases $w_{P(\vec{v})}(M) = M$, so M is a fixpoint and therefore a static interpretation by Observation 3. □

The converse does not hold, that is DO admits interpretations that are not stable models. For programs having stable models, DO agrees; but DO also assigns meaning to programs where stable models are absent or where convergent iteration matters (as in PageRank).

6 Conclusion

We introduced DO semantics, a stabilization-based framework for nonmonotonic logic programs with unrestricted recursion and aggregation. DO-truth admits three equivalent characterizations: graph-theoretic (stabilization in the derivation graph), game-theoretic (the Thesis Defense game), and modal (the $\Box\Diamond\Box$ condition in S4). DO semantics coincides with least fixpoint semantics for positive Datalog, is compatible with WFS and SMS, and extends to programs that converge without reaching a fixpoint. Through ω-limit interpretations, it bridges discrete symbolic reasoning with continuous mathematics, giving formal meaning to programs that compute real-valued limits.

Supplementary material

To view supplementary material for this article, please visit https://doi.org/10.1017/S1471068426100660.

References

Abiteboul, S., Hull, R. and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Alpern, B. and Schneider, F. B. 1985. Defining liveness. *Information Processing Letters* 21, 4, 181–185.

Atkinson, K., Han, W. and Stewart, D. E. 2009. *Numerical Solution of Ordinary Differential Equations*. Wiley.

Bertsekas, D. P. and Tsitsiklis, J. N. 1989. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall.

Brin, S. and Page, L. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks* 30, 1, 107–117.

Ceri, S., Gottlob, G. and Tanca, L. 1990. Logic programming and databases. In *Surveys in Computer Science*. Springer.

Chellas, B. F. 1980. *Modal Logic: An Introduction*. Cambridge University Press.

Faber, W., Leone, N. and Pfeifer, G. 2004. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *JELIA 2004, Lecture Notes in Computer Science*, Vol. 3229, Springer, 200–212.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming* 19, 1, 27–82.

Gelfond, M. 2008. Answer sets. In *Handbook of Knowledge Representation*, Vol. 3, Elsevier, 285–316.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP*, 1070–1080.

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press

Kemp, D. B. and Stuckey, P. J. 1991. Semantics of logic programs with aggregates. In *Proceedings of the 1991 International Symposium on Logic Programming (ISLP)*. MIT Press, 387–401.

Liu, Y. A. and Stoller, S. D. 2022. Recursive rules with aggregation: A simple unified semantics. *Journal of Logic and Computation* 32, 8, 1659–1693.

Lorenzen, P. 1961. Ein dialogisches Konstruktivitätskriterium. In *Infinitistic Methods*. Pergamon Press, 193–200.

Meyer, C. D. 2000. *Matrix Analysis and Applied Linear Algebra*. SIAM.

Mumick, I. S., Pirahesh, H. and Ramakrishnan, R. 1990. The magic of duplicates and aggregates. In *VLDB*, 264–277.

Ross, K. A. and Sagiv, Y. 1997. Monotonic aggregation in deductive databases. *Journal of Computer and System Sciences* 54, 1, 79–97.

Skvortsov, E. 2023. *Full Support of PostgreSQL Engine Comes to Logica*. Google Open Source Blog.

Skvortsov, E. S., Xia, Y., Bowers, S. and Ludäscher, B. 2024a. The Logica system: Elevating SQL Databases to declarative data science engines. In *Datalog-2.0*, Vol. 3801 of CEUR Workshop Proceedings, 69–73.

Skvortsov, E. S., Xia, Y. and Ludäscher, B. 2024b. Logica: declarative data science for mere mortals. In *EDBT*, 842–845.

Trinh, V.-G., Benhamou, B., Soliman, S. and Fages, F. 2025. On the Boolean network theory of Datalog$^{\neg}$. arXiv preprint arXiv:2504.15417.

Trinh, V.-G., Soliman, S., Fages, F. and Benhamou, B. 2026. On the trap space semantics of normal logic programs. In *Proceedings of the 41st International Conference on Logic Programming (ICLP 2025)*, Vol. 439 of Electronic Proceedings in Theoretical Computer Science, Open Publishing Association, 294–319.

van Emden, M. H. and Kowalski, R. A. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM* 23, 4, 733–742.

Van Gelder, A. 1993. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences* 47, 1, 185–221.

Van Gelder, A., Ross, K. A. and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM* 38, 3, 620–650.

Vianu, V. 2021. Datalog unchained. In *ACM PODS*, 57–69.

Zaniolo, C., Yang, M., Das, A., Shkapsky, A., Condie, T. and Interlandi, M. 2017. Fixpoint semantics and optimization of recursive Datalog programs with aggregates. *Theory and Practice of Logic Programming* 17, 5-6, 1048–1065.

https://doi.org/10.1017/S1471068426100660 Published online by Cambridge University Press