Graphical conditions for existence, unicity and multiplicity of non-trivial regular models

Van-Giang Trinh, Belaid Benhamou, Sylvain Soliman, François Fages

TPLP: Page 1–8. © The Author(s), 2021. Published by Cambridge University Press 2021
doi:10.1017/xxxxx

Graphical conditions for existence, unicity and multiplicity of non-trivial regular models

VAN-GIANG TRINH
LIRICA team, LIS, Aix-Marseille University, Marseille, France

BELAID BENHAMOU
LIRICA team, LIS, Aix-Marseille University, Marseille, France

SYLVAIN SOLIMAN
Inria Saclay, EP Lifeware, Palaiseau, France

FRANCOIS FAGES
Inria Saclay, EP Lifeware, Palaiseau, France

submitted xx xx xxxx; revised xx xx xxxx; accepted xx xx xxxx

Abstract

The regular models of a logic program are a particular type of partial (3-valued) models which correspond to stable partial models with minimal undefinedness. In this paper, we explore graphical conditions on the dependence graph of a normal logic program to analyze the existence, unicity and multiplicity of non-trivial regular models for the program. We show three main results: 1) a necessary condition for the existence of non-trivial regular models, 2) a sufficient condition for the unicity of regular models, and 3) two upper bounds for the number of regular models based on positive feedback vertex sets. The first two conditions generalize the existing results obtained by You and Yuan (1994) for well-founded stratification logic programs. The third result is new to the best of our knowledge. Key to our proofs is a connection that we establish between logic programs and Boolean network theory.

KEYWORDS: logic program, stable model, regular model, graphical condition, Boolean network

1 Introduction

Relating graphical representations of a logic program and its model-theoretic semantics is an interesting research direction in theory that also has many useful applications in practice (Fages 1994; Costantini 2006; Linke 2001). Historically, the first studies of this direction focused on the existence of a unique stable model in classes of logic programs with special graphical properties on (positive) dependence graphs, including positive programs (Gelfond and Lifschitz 1988), acyclic programs (Apt and Bezem 1991), and locally stratified programs (Gelfond and Lifschitz 1988). In 1994, Fages proved an important result showing that the set of stable models and the set of 2-valued models of the Clark's completion of a *tight* logic program are the same (Fages 1994). Being finer-represented but more computationally expensive than dependence graphs, several other graphical representations (e.g., cycle and extended dependence graphs, rule graphs, block graphs)

were introduced and several improved results were obtained (Costantini 2006; Costantini and Provetti 2011; Dimopoulos and Torres 1996; Linke 2001). There are some recent studies on dependence graphs (Fandinno and Lifschitz 2023; Trinh and Benhamou 2024), but they still focus only on stable models. In contrast, very few studies were made about regular models despite of their prominent importance (Janhunen et al. 2006). The work of Eiter et al. (1997) showed the unicity of regular and stable models in locally stratified logic programs. The work of You and Yuan (1994) showed two sufficient graphical conditions, one for the coincidence between stable and regular models, and another one for the unicity of regular models. However, these two conditions were only proved in the case of well-founded stratification programs, and the question if they are still valid for generic logic programs is still open to date.

The stable partial semantics is the 3-valued generalization of the (2-valued) stable model semantics (Przymusinski 1990). The regular model semantics not only inherits the advantages of the stable partial model semantics but also imposes two notable principles in non-monotonic reasoning: *minimal undefinedness* and *justifiability* (which is closely related to the concept of labeling-based justification in Doyle's truth maintenance system (Doyle 1979)), making it become one of the well-known semantics in logic programming (You and Yuan 1994; Janhunen et al. 2006). Furthermore, regular models in logic programs were proved to correspond to preferred extensions in Dung's frameworks (Wu et al. 2009), which are a central focus in abstract argumentation (Baroni et al. 2011).

Recently, we have proposed a new semantics for normal logic programs, called the *trap space semantics*, which establishes formal links between the model-theoretic and dynamical semantics of logic programs (Trinh et al. 2024b). It is built on two newly proposed concepts: *stable* and *supported trap spaces*, which are inspired by the concepts of *trap* (or its duality, *siphon*) in Petri net theory and *trap space* in Boolean network theory (Murata 1989; Klarner et al. 2015; Trinh et al. 2023; 2024a). We relate the new semantics to other widely-known semantics, in particular showing that subset-minimal stable trap spaces of a logic program coincide with its regular models. This result can open potential applications to graphical analysis for the number of regular models.

Motivated by the above elements, in this paper, we explore graphical conditions on the dependence graph of a normal logic program to analyze the existence of non-trivial (i.e., not 2-valued) regular models and the unicity and multiplicity of regular models for the program. More specifically, we show three main results: 1) the existence of negative cycles is a necessary condition for the existence of non-trivial regular models, 2) the absence of positive cycles is a sufficient condition for the unicity of regular models, and 3) $3^{|U^+|}$ (resp. $2^{|U^+|}$) is an upper bound (resp. a finer upper bound) for the number of regular models in generic (resp. tight) logic programs where U^+ is a positive feedback vertex set of the dependence graph. The first two conditions generalize the existing results obtained by You and Yuan (1994) for well-founded stratification logic programs. The third result is new to the best of our knowledge. Key to our proofs is a connection that we establish between logic programs and Boolean network theory based on the trap space semantics.

Boolean Networks (BNs) are a simple and efficient mathematical formalism that has been widely applied to many areas from science to engineering (Schwab et al. 2020). Originated in the early work of Thomas and d'Ari (1990), studying relationships between the dynamics of a BN and its influence graph has a rich history of research (Richard 2019). To date, this research direction is still growing with many prominent and deep

results (Richard 2019; Schwab et al. 2020). Hence, the established connection can bring a plenty of existing results in BNs to studying logic programs as well as provide a unified framework for exploring and proving more new theoretical results in logic programs.

The rest of this paper is organized as follows. In the next section, we recall preliminaries on normal logic programs, regular models, BNs, and related concepts. Section 3 presents the connection that we establish between logic programs and BNs. In Section 4, we present the main results on relationships between regular models and graphical conditions. Finally, Section 5 concludes the paper with some perspectives for future work.

2 Preliminaries

We assume that the reader is familiar with logic programs and the stable model semantics (Gelfond and Lifschitz 1988). Unless specifically stated, a logic program (or a program for short) means a *ground* normal logic program. In addition, $\mathbb{B} = \{\text{true}, \text{false}\} = \{1, 0\}$ is the Boolean domain and all Boolean operators used in this paper include $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), $\leftarrow$ (implication), and $\leftrightarrow$ (bi-implication).

2.1 Normal logic programs

A logic program P is a finite set of rules of the form $p \leftarrow p_1, \ldots, p_m, \sim p_{m+1}, \ldots, \sim p_k$ where p and p_i are variable-free atoms ($k \geq m \geq 0$), $\sim$ denotes the negation as failure and can be equivalent to $\neg$ in a Boolean formula. We use $\text{atom}(P)$ to denote the set of all atoms of P. For any rule r of this form, $\text{h}(r) = p$ is the *head* of r, $\text{b}^+(r) = \{p_1, \ldots, p_m\}$ is called the *positive body* of r, $\text{b}^-(r) = \{p_{m+1}, \ldots, p_k\}$ is called the *negative body* of r, and $\text{bf}(r) = p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \cdots \wedge \neg p_k$ is the *body formula* of r. If $\text{b}^+(r) = \text{b}^-(r) = \emptyset$, then r is called a *fact*. If $\text{b}^-(r) = \emptyset, \forall r \in P$, then P is called a *positive program*. If $\text{b}^+(r) = \emptyset, \forall r \in P$, then P is called a *quasi-interpretation* program.

We shall use the fixpoint semantics of logic programs (Dung and Kanchanasut 1989) to prove many new results in the next sections. To be self-contained, we briefly recall the definition of the *least fixpoint* of a logic program P as follows. Let r be the rule $p \leftarrow \sim p_1, \ldots, \sim p_k, q_1, \ldots, q_j$ and let r_i be rules $q_i \leftarrow \sim q_i^1, \ldots, \sim q_i^{l_i}$ where $1 \leq i \leq j$ and $l_i \geq 0$. Then $\sigma_r(\{r_1, \ldots, r_j\})$ is the following rule $p \leftarrow \sim p_1, \ldots, \sim p_k, \sim q_1^1, \ldots, \sim q_1^{l_1}, \ldots, \sim q_j^1, \ldots, \sim q_j^{l_j}$. σ_P is the transformation on quasi-interpretation programs: $\sigma_P(Q) = \{\sigma_r(\{r_1, \ldots, r_j\}) | r \in P, r_i \in Q, 1 \leq i \leq j\}$. Let $\text{lfp}_i = \sigma_P^i(\emptyset) = \sigma_P(\sigma_P(\ldots \sigma_P(\emptyset)))$, then $\text{lfp}(P) = \bigcup_{i \geq 1} \text{lfp}_i$ is the least fixpoint of P. $\text{lfp}(P)$ is finite and also a quasi-interpretation program (Dung and Kanchanasut 1989).

2.1.1 Stable and supported partial models

A *3-valued interpretation* I of a logic program P is a total function $I\colon \text{atom}(P) \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ that assigns one of the truth values true ($\mathbf{t}$), false ($\mathbf{f}$) or unknown ($\mathbf{u}$), to each atom of P. If $I(a) \neq \mathbf{u}, \forall a \in \text{atom}(P)$, then I is an *Herbrand (2-valued) interpretation* of P. Usually, a 2-valued interpretation is written as the set of atoms that are true in this interpretation. A 3-valued interpretation I characterizes the set of 2-valued interpretations denoted by $\gamma(I)$ as $\gamma(I) = \{J | J \in 2^{\text{atom}(P)}, \forall a \in \text{atom}(P), I(a) \neq \mathbf{u} \Rightarrow J(a) = I(a)\}$. For example, if $I = \{p = \mathbf{t}, q = \mathbf{f}, r = \mathbf{u}\}$, then $\gamma(I) = \{\{p\}, \{p, r\}\}$.

We consider two orders on 3-valued interpretations. The truth order $\leq_t$ is given by $\mathbf{f} <_t \mathbf{u} <_t \mathbf{t}$. Then, $I_1 \leq_t I_2$ iff $I_1(a) \leq_t I_2(a), \forall a \in \text{atom}(P)$. The subset order $\leq_s$ is given by $\mathbf{f} <_s \mathbf{u}$ and $\mathbf{t} <_s \mathbf{u}$. Then, $I_1 \leq_s I_2$ iff $I_1(a) \leq_s I_2(a), \forall a \in \text{atom}(P)$. In addition, $I_1 \leq_s I_2$ iff $\gamma(I_1) \subseteq \gamma(I_2)$, i.e., $\leq_s$ is identical to the subset partial order.

Let f be a propositional formula on $\text{atom}(P)$. Then the valuation of f under a 3-valued interpretation I (denoted by $I(f)$) is defined recursively as follows:

$$I(f) = \begin{cases} I(a) & \text{if } f = a, a \in \text{atom}(P) \\ \neg I(f_1) & \text{if } f = \neg f_1 \\ \min_{\leq_t}(I(f_1), I(f_2)) & \text{if } f = f_1 \wedge f_2 \\ \max_{\leq_t}(I(f_1), I(f_2)) & \text{if } f = f_1 \vee f_2 \end{cases}$$

where $\neg\mathbf{t} = \mathbf{f}, \neg\mathbf{f} = \mathbf{t}, \neg\mathbf{u} = \mathbf{u}$, and $\min_{\leq_t}$ (resp. $\max_{\leq_t}$) is the function to get the minimum (resp. maximum) value of two values w.r.t. the order $\leq_t$. We say 3-valued interpretation I is a *3-valued model* of a logic program P iff for each rule $r \in P$, $I(\text{bf}(r)) \leq_t I(\text{h}(r))$.

Definition 1
Let I be a 3-valued interpretation of P. We build the *reduct* P^I as follows.

- Remove any rule $a \leftarrow a_1, \ldots, a_m, \sim b_1, \ldots, \sim b_k \in P$ if $I(b_i) = \mathbf{t}$ for some $1 \leq i \leq k$.
- Afterwards, remove any occurrence of $\sim b_i$ from P such that $I(b_i) = \mathbf{f}$.
- Then, replace any occurrence of $\sim b_i$ left by a special atom $\mathbf{u}$ ($\mathbf{u} \notin \text{atom}(P)$).

P^I is positive and has a unique $\leq_t$-least 3-valued model. See Przymusinski (1990) for the method for computing this $\leq_t$-least 3-valued model. Then I is a *stable partial model* of P iff I is equal to the $\leq_t$-least 3-valued model of P^I. A stable partial model I is a regular model if it is $\leq_s$-minimal. A regular model is non-trivial if it is not 2-valued.

The *Clark's completion* of P (denoted by $\text{cf}(P)$) consists of the following sentences: for each $p \in \text{atom}(P)$, let $r_1, \ldots, r_k$ be all the rules of P having the same head p, then $p \leftrightarrow \text{bf}(r_1) \vee \cdots \vee \text{bf}(r_k)$ is in $\text{cf}(P)$. If there is no rule whose head is p, then the equivalence is $p \leftrightarrow \mathbf{f}$. Let $\text{rhs}(a)$ denote the right hand side of atom a in $\text{cf}(P)$. A 3-valued interpretation I is a 3-valued model of $\text{cf}(P)$ iff for every $a \in \text{atom}(P)$, $I(a) = I(\text{rhs}(a))$. We define a *supported partial model* of P as a 3-valued model of $\text{cf}(P)$. Note that 2-valued stable (resp. supported) partial models are stable (resp. supported) models.

2.1.2 Dependence and transition graphs

The Dependence Graph (DG) of a logic program P (denoted by $\text{dg}(P)$) is a signed directed graph (V, E) on the set of signs $\{\oplus, \ominus\}$ where $V = \text{atom}(P)$ and $(uv, \oplus) \in E$ (resp. $(uv, \ominus) \in E$) iff there is a rule $r \in P$ such that $v = \text{h}(r)$ and $u \in \text{b}^+(r)$ (resp. $u \in \text{b}^-(r)$). An arc $(uv, \oplus)$ is positive, whereas an arc $(uv, \ominus)$ is negative. A cycle of $\text{dg}(P)$ is positive (resp. negative) if it contains an even (resp. odd) number of negative arcs. A positive (resp. negative) feedback vertex set is a set of vertices that intersect all positive (resp. negative) cycles of $\text{dg}(P)$. The positive DG of P (denoted by $\text{dg}^+(P)$) is a sub-graph of $\text{dg}(P)$ that has the same set of vertices but contains only positive arcs. P is *locally stratified* if every cycle of $\text{dg}(P)$ contains no negative arc (Gelfond and Lifschitz 1988). P is *tight* if $\text{dg}^+(P)$ has no cycle (Fages 1994). P is *well-founded stratification* if there is a topological order on the set of Strongly Connected Components (SCCs) of

$\mathrm{dg}(P)$ and for every SCC B, there exists SCC $A \leq B$ and for any SCC C, if $C \leq A$ then there are only positive arcs from atoms in C to atoms in A (You and Yuan 1994). Herein, $A \leq B$ iff there is a path from some atom in A to some atom in B.

The *immediate consequence operator* (or the T_P *operator*) is defined as a mapping $T_P \colon 2^{\mathrm{atom}(P)} \to 2^{\mathrm{atom}(P)}$ such that $T_P(I)(a) = I(\mathrm{rhs}(a))$ where I is a 2-valued interpretation. If I is a 2-valued interpretation, then P^I is exactly the reduct defined in (Gelfond and Lifschitz 1988) and the unique $\leq_t$-least model of P^I is 2-valued. The *Gelfond-Lifschitz operator* (or the F_P *operator*) is defined as a mapping $F_P \colon 2^{\mathrm{atom}(P)} \to 2^{\mathrm{atom}(P)}$ such that $F_P(I)$ is the unique $\leq_t$-least model of P^I (Gelfond and Lifschitz 1988). The *stable* (resp. *supported*) *transition graph* of P is a directed graph (denoted by $\mathrm{tg}_{st}(P)$ (resp. $\mathrm{tg}_{sp}(P)$)) on the set of all possible 2-valued interpretations of P such that (I, J) is an arc of $\mathrm{tg}_{st}(P)$ (resp. $\mathrm{tg}_{sp}(P)$) iff $J = F_P(I)$ (resp. $J = T_P(I)$). A *trap domain* of a directed graph is a set of vertices having no out-going arcs.

2.1.3 Stable and supported trap spaces

In Trinh et al. (2024b), we introduce a new semantics for normal logic programs, called the *trap space semantics*. This semantics shall be used in this work as the bridge between logic programs and Boolean networks. To be self-contained, we briefly recall the definition and essential properties of this semantics.

A set S of 2-valued interpretations of a logic program P is called a *stable trap set* (resp. *supported trap set*) of P if $\{F_P(I) | I \in S\} \subseteq S$ (resp. $\{T_P(I) | I \in S\} \subseteq S$). A 3-valued interpretation I of a logic program P is called a *stable trap space* (resp. *supported trap space*) of P if $\gamma(I)$ is a stable (resp. supported) trap set of P. By definition, a stable (resp. supported) trap set of P is a trap domain of $\mathrm{tg}_{st}(P)$ (resp. $\mathrm{tg}_{sp}(P)$). Hence, we can deduce that a 3-valued interpretation I is a stable (resp. supported) trap space of P if $\gamma(I)$ is a trap domain of $\mathrm{tg}_{st}(P)$ (resp. $\mathrm{tg}_{sp}(P)$). We also show in Trinh et al. (2024b) that I is a supported trap space of P iff I is 3-valued model of $\overleftarrow{\mathrm{cf}}(P)$ w.r.t. to the order $\leq_s$ where $\overleftarrow{\mathrm{cf}}(P)$ is the $\leftarrow$ part of the Clark's completion of P, and a stable (resp. supported) partial model of P is also a stable (resp. supported) trap space of P.

Example 1

Consider logic program P_1 (taken from Inoue and Sakama (2012)) where $P_1 = \{p \leftarrow \sim q; q \leftarrow \sim p; r \leftarrow q\}$. Herein, we use ';' to separate program rules. Figures 1 (a), (b), and (c) show the dependence graph, the stable transition graph, and the supported transition graph of P_1, respectively. P_1 is tight, but not locally stratified or well-founded stratification. P_1 has five stable (also supported) trap spaces: $I_1 = \{p = \mathbf{t}, q = \mathbf{f}, r = \mathbf{u}\}$, $I_2 = \{p = \mathbf{f}, q = \mathbf{t}, r = \mathbf{u}\}$, $I_3 = \{p = \mathbf{u}, q = \mathbf{u}, r = \mathbf{u}\}$, $I_4 = \{p = \mathbf{t}, q = \mathbf{f}, r = \mathbf{f}\}$, and $I_5 = \{p = \mathbf{f}, q = \mathbf{t}, r = \mathbf{t}\}$. Among them, only I_3, I_4, and I_5 are stable (also supported) partial models of P_1. P_1 has two regular models (I_4 and I_5). The least fixpoint of P_1 is $\mathrm{lfp}(P_1) = \{p \leftarrow \sim q; q \leftarrow \sim p; r \leftarrow \sim p\}$.

2.2 Boolean networks

A Boolean Network (BN) f is a set of Boolean functions on a set of Boolean variables denoted by var_f. Each variable v is associated with a Boolean function $f_v \colon \mathbb{B}^{|\mathrm{var}_f|} \to \mathbb{B}$.

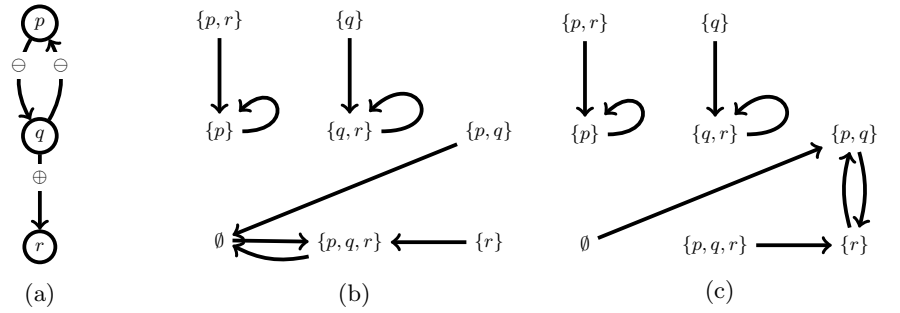

Fig. 1: (a) $\mathrm{dg}(P_1)$, (b) $\mathrm{tg}_{st}(P_1)$, and (c) $\mathrm{tg}_{sp}(P_1)$.

f_v is called *constant* if it is always either 0 or 1 regardless of its arguments. A state s of f is a mapping $s\colon \mathrm{var}_f \mapsto \mathbb{B}$ that assigns either 0 (inactive) or 1 (active) to each variable. We can write s_v instead of $s(v)$ for short.

Let x be a state of f. We use $x[v \leftarrow a]$ to denote the state y so that $y_v = a$ and $y_u = x_u, \forall u \in \mathrm{var}_f, u \neq v$ where $a \in \mathbb{B}$. The Influence Graph (IG) of f (denoted by $\mathrm{ig}(f)$) is a signed directed graph (V, E) on the set of signs $\{\oplus, \ominus\}$ where $V = \mathrm{var}_f$, $(uv, \oplus) \in E$ (i.e., u positively affects the value of f_v) iff there is a state x such that $f_v(x[u \leftarrow 0]) < f_v(x[u \leftarrow 1])$, and $(uv, \ominus) \in E$ (i.e., u negatively affects the value of f_v) iff there is a state x such that $f_v(x[u \leftarrow 0]) > f_v(x[u \leftarrow 1])$.

At each time step t, variable v can update its state to $s'(v) = f_v(s)$, where s (resp. s') is the state of f at time t (resp. $t+1$). An *update scheme* of a BN refers to how variables update their states over (discrete) time (Schwab et al. 2020). Various update schemes exist, but the primary types are *synchronous*, where all variables update simultaneously, and *fully asynchronous*, where a single variable is non-deterministically chosen for updating. By adhering to the update scheme, the BN transitions from one state to another, which may or may not be the same. This transition is referred to as the *state transition*. Then the dynamics of the BN is captured by a directed graph referred to as the *State Transition Graph* (STG). We use $\mathrm{sstg}(f)$ (resp. $\mathrm{astg}(f)$) to denote the synchronous (resp. asynchronous) STG of f.

A non-empty set of states is a *trap set* if it has no out-going arcs on the STG of f. An *attractor* is a subset-minimal trap set. An attractor of size 1 (resp. at least 2) is called a fixed point (resp. cyclic attractor). A *sub-space* m of a BN is a mapping $m\colon \mathrm{var}_f \mapsto \mathbb{B} \cup \{\star\}$. A sub-space m is equivalent to the set of all states s such that $s(v) = m(v), \forall v \in \mathrm{var}_f, m(v) \neq \star$. With abuse of notation, we use m and its equivalent set of states interchangeably. For example, $m = \{v_1 = \star, v_2 = 1, v_3 = 1\} = \{011, 111\}$ (for simplicity, we write states as a sequence of values). If a sub-space is also a trap set, it is a *trap space*. Unlike trap sets and attractors, trap spaces of a BN are independent of the update scheme (Klarner et al. 2015). Then a trap space m is minimal iff there is no other trap space m' such that $m' \subset m$. It is easy to derive that a minimal trap space contains at least one attractor of the BN regardless of the update scheme.

Example 2

Consider BN f_1 with $f_p = \neg q, f_q = \neg p, f_r = q$. Figures 2 (a), (b), and (c) show the influence graph, the synchronous STG, and the asynchronous STG of f_1. Attractor states are highlighted with boxes. $\mathrm{sstg}(f_1)$ has two fixed points and one cyclic attractor, whereas

astg(f_1) has only two fixed points. f_1 has five trap spaces: $m_1 = 10\star$, $m_2 = 01\star$, $m_3 = \star\star\star$, $m_4 = 100$, and $m_5 = 011$. Among them, m_4 and m_5 are minimal.

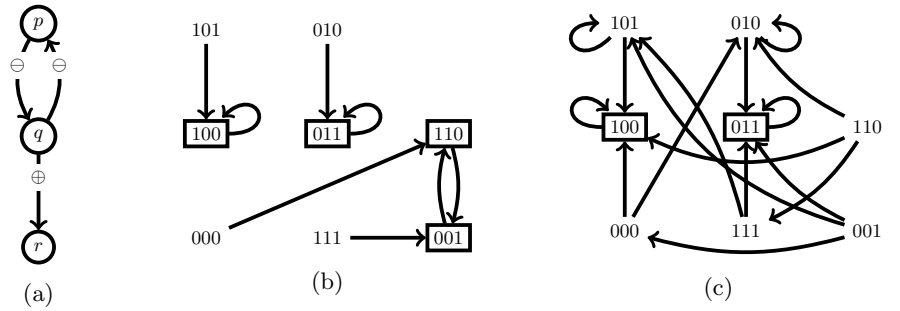

Fig. 2: (a) ig(f_1), (b) sstg(f_1), and (c) astg(f_1).

3 Logic programs and Boolean networks

We define a BN encoding for logic programs in Definition 2. Then, we show two relationships between a logic program and its encoded BN (see Theorems 1 and 3).

Definition 2
Let P be a logic program. We define a BN f encoding P as follows: $\mathrm{var}_f = \mathrm{atom}(P)$, $f_v = \bigvee_{r \in P, v = \mathrm{h}(r)} \mathrm{bf}(r), \forall v \in \mathrm{var}_f$. Conventionally, if there is no rule $r \in P$ such that $\mathrm{h}(r) = v$, then $f_v = 0$. By considering 1 (resp. 0) as **t** (resp. **f**), and $\star$ as **u**, sub-spaces (resp. states) of f are identical to 3-valued (resp. 2-valued) interpretations of P.

Theorem 1
Let P be a logic program and f be its encoded BN. Then $\mathrm{ig}(f) \subseteq \mathrm{dg}(P)$.

Proof
By construction, ig(f) and dg(P) have the same set of vertices. Let $\mathrm{in}_f^+(v)$ (resp. $\mathrm{in}_P^+(v)$) denote the set of vertices u such that $(uv, \oplus)$ is an arc of ig(f) (resp. dg(P)). We define $\mathrm{in}_f^-(v)$ (resp. $\mathrm{in}_P^-(v)$) similarly. We show that $\mathrm{in}_f^+(v) \subseteq \mathrm{in}_P^+(v)$ and $\mathrm{in}_f^-(v) \subseteq \mathrm{in}_P^-(v)$ for every $v \in \mathrm{atom}(P)$ (*). Consider atom u. The case that both u and $\sim u$ appear in rules whose heads are v is trivial. For the case that only u appears in rules whose heads are v, u is essential in f_v by construction, and it positively affects the value of f_v, leading to $u \in \mathrm{in}_f^+(v)$ and $u \notin \mathrm{in}_f^-(v)$. This implies that (*) still holds. The case that only $\sim u$ appears in rules whose heads are v is similar. By (*), we can conclude that $\mathrm{ig}(f) \subseteq \mathrm{dg}(P)$, i.e., ig($f$) is a sub-graph of dg($P$). In addition, if P is a quasi-interpretation program, then $\mathrm{ig}(f) = \mathrm{dg}(P)$. □

Lemma 2 (*derived from Theorem 4.5 of Inoue and Sakama (2012)*)
Let P be a logic program and f be its encoded BN. Then $\mathrm{tg}_{sp}(P) = \mathrm{sstg}(f)$.

Theorem 3
Let P be a logic program and f be its encoded BN. Then supported trap spaces of P coincide with trap spaces of f.

Proof
By Lemma 2, $\text{tg}_{sp}(P) = \text{sstg}(f)$. Note that trap spaces of f are the same under both the synchronous and asynchronous update schemes (Klarner et al. 2015). Hence, trap spaces of f coincide with trap spaces of $\text{sstg}(f)$. Since $\text{tg}_{sp}(P) = \text{sstg}(f)$, supported trap spaces of P coincide with trap spaces of f. □

For illustration, BN f_1 of Example 2 is the encoded BN of logic program P_1 of Example 1. $\text{tg}_{sp}(P_1)$ is identical to $\text{sstg}(f_1)$, and the five supported trap spaces of P_1 are identical to the five trap spaces of f_1. In addition, P_1 is tight and $\text{ig}(f_1) = \text{dg}(P_1)$.

4 Graphical analysis results

In this section, we present our new results on graphical conditions for several properties of regular models in logic programs by exploiting the connection established in Section 3.

4.1 Preparations

For convenience, we first recall several existing results in both logic programs and Boolean networks that shall used later.

Theorem 4 (*Inoue and Sakama (2012)*)
Let P be a quasi-interpretation program. Then $\text{tg}_{st}(P) = \text{tg}_{sp}(P)$, i.e., the stable and supported transition graphs of P are the same.

Theorem 5 (*Inoue and Sakama (2012)*)
Let P be a logic program and $\text{lfp}(P)$ denote its least fixpoint. Then P and $\text{lfp}(P)$ have the same stable transition graph.

Theorem 6 (*Theorem 6 of Trinh and Benhamou (2024)*)
Let P be a logic program and $\text{lfp}(P)$ denote its least fixpoint. If P is locally stratified, then $\text{dg}(\text{lfp}(P))$ has no cycle.

Lemma 7
Let P be a logic program and $\text{lfp}(P)$ denote its least fixpoint. If $\text{dg}(P)$ is has no negative cycle, then $\text{dg}(\text{lfp}(P))$ has no negative cycle.

Proof
It directly follows from Lemma 5.3 of Fages (1994). □

Proposition 8 (*Trinh et al. (2024b)*)
Let P be a logic program. Let $T(P)$ denote the set of all supported trap spaces of P. Let $C(P)$ denote the set of all 3-valued models of $\text{cf}(P)$ (i.e., the Clark's completion of P). For every supported trap space $I \in T(P)$, there is a model $I' \in C(P)$ such that $I' \leq_s I$.

Sketch of proof
Let I^j be an arbitrary supported trap space in $T(P)$. We construct a 3-valued interpretation I^{j+1} as follows: $\forall a \in \text{atom}(P), I^{j+1}(a) = I^j(\text{rhs}(a))$. We prove that I^{j+1} is also a supported trap space of P. For every supported trap space I in $T(P)$, we start with $I^j = I$ and repeat the above process by increasing j by 1, and finally reach the case that $I^{j+1} = I^j$ because $\gamma(I)$ is finite. By construction, $I^j(a) = I^j(\text{rhs}(a)), \forall a \in \text{atom}(P)$, and $I^j \leq_s I$. Hence, by setting $I' = I^j$, there is a model $I' \in C(P)$ such that $I' \leq_s I$. □

Theorem 9 (*Trinh et al. (2024b)*)
Let P be a logic program. Then a 3-valued interpretation I is a regular model of P iff I is a $\leq_s$-minimal stable trap space of P.

Sketch of proof
Let $\text{lfp}(P)$ be the least fixpoint of P. By Proposition 8, we can deduce that $\leq_s$-minimal supported trap spaces of $\text{lfp}(P)$ coincide with $\leq_s$-minimal supported (also stable) partial models spaces of $\text{lfp}(P)$. P and $\text{lfp}(P)$ have the same set of stable partial models (Aravindan and Dung 1995). By Theorem 5, P and $\text{lfp}(P)$ have the same stable transition graph, thus they have the same set of stable trap spaces. Since stable trap spaces of $\text{lfp}(P)$ coincide with its supported trap spaces, we can conclude the theorem. □

Theorem 10 (*Theorem 1 of Richard (2019)*)
Let f be a BN. If $\text{ig}(f)$ has no cycle, $\text{astg}(f)$ has a unique attractor that is also the unique fixed point of f.

Theorem 11 (*Theorem 12 of Richard (2019)*)
Let f be a BN. If $\text{ig}(f)$ has no negative cycle, then $\text{astg}(f)$ has no cyclic attractor.

4.2 Unicity of regular and stable models

To illustrate better applications of the connection between logic programs and Boolean networks, we start with providing a probably simpler proof for a well-known result on the unicity of regular and stable models in locally stratified logic programs.

Theorem 12 (*Eiter et al. (1997)*)
If P is a locally stratified logic program, then P has a unique regular model that is also the unique stable model of P.

New proof
Let $\text{lfp}(P)$ denote the least fixpoint of P. Let f be the encoded BN of $\text{lfp}(P)$. By Theorem 6, $\text{dg}(\text{lfp}(P))$ has no cycle. Since $\text{ig}(f)$ is a sub-graph of $\text{dg}(\text{lfp}(P))$ by Theorem 1, it also has no cycle. By Theorem 10, $\text{astg}(f)$ has a unique attractor that is also the unique fixed point of f. P and $\text{lfp}(P)$ have the same set of regular (also stable) models (Aravindan and Dung 1995). By Theorem 9, regular models of $\text{lfp}(P)$ are $\leq_s$-minimal stable trap spaces of $\text{lfp}(P)$. Since $\text{lfp}(P)$ is a quasi-interpretation program, its stable trap spaces coincide with its supported trap spaces. Supported trap spaces of $\text{lfp}(P)$ coincide with trap spaces of f by Theorem 3. Hence, regular models of P coincide with $\leq_s$-minimal trap spaces of f. Since the number of $\leq_s$-minimal trap spaces of f are a lower bound of the

number of attractors of astg(f) and f has at least one $\leq_s$-minimal trap space (Klarner et al. 2015), f has a unique $\leq_s$-minimal trap space that is also the unique fixed point of f. Hence, P has a unique regular model that is also the unique stable model of P. □

4.3 Existence of non-trivial regular models

Theorem 13 (*Theorem 5.3(i) of You and Yuan (1994)*)
Let P be a well-founded stratification program. If dg(P) has no negative cycle, then all the regular models of P are 2-valued.

Theorem 13 provides a sufficient (resp. necessary) condition on the dependence graph for the non-existence (resp. existence) of non-trivial regular models, but it is only limited to well-founded stratification programs. Note that the set of all well-founded stratification programs is only a small piece of the set of all possible programs (You and Yuan 1994). Moreover, the proof of this result is quite complicated, and to the best of our knowledge, the question if it is valid for generic logic programs (i.e., programs with no graphical constraints) is still open to date. We answer this question in Theorem 14.

Theorem 14
Let P be a generic logic program. If dg(P) has no negative cycle, then all the regular models of P are 2-valued.

Proof
Let lfp(P) be the least fixpoint of P. By Lemma 7, dg(lfp(P)) has no negative cycle. Let f be the encoded BN of lfp(P). Since ig(f) is a sub-graph of dg(lfp(P)) by Theorem 1, ig(f) also has no negative cycle. By Theorem 11, astg(f) (i.e., the asynchronous transition graph of f) has no cyclic attractor. This implies that all attractors of astg(f) are fixed points (*). Assume that f has a $\leq_s$-minimal trap space (say m) that is not a fixed point. Since every $\leq_s$-minimal trap space of f contains at least one attractor of astg(f) (Klarner et al. 2015), there is an attractor (say A) of astg(f) such that $A \subseteq \gamma(m)$. By (*), A is a fixed point, leading to $A <_s m$. This is a contradiction because m is $\leq_s$-minimal. Hence, all $\leq_s$-minimal trap spaces of f are fixed points.

By Theorem 3, trap spaces of f coincide with supported trap spaces of lfp(P). lfp(P) is a quasi-interpretation program, thus $\mathrm{tg}_{st}(\mathrm{lfp}(P)) = \mathrm{tg}_{sp}(\mathrm{lfp}(P))$. It follows that its supported trap spaces are also its stable trap spaces. Hence, $\leq_s$-minimal trap spaces of f are $\leq_s$-minimal stable trap spaces of lfp(P). This implies that all $\leq_s$-minimal stable trap spaces of lfp(P) are 2-valued. By Theorem 9, all regular models of lfp(P) are 2-valued. P and lfp(P) have the same set of regular models (Aravindan and Dung 1995). Hence, all regular models of P are 2-valued. □

Theorem 14 implies that the undefinedness is only needed if there is a negative cycle in the DG, i.e., the regular model and stable model semantics are the same under the absence of negative cycles. In addition, we can get from Theorem 14 a straightforward corollary: if the DG of a logic program has no negative cycle, then it has at least one stable model. The reason is because a logic program always has at least one regular model. This corollary is exactly the generalization of Theorem 5.7 of You and Yuan (1994) for well-founded stratification programs.

4.4 Unicity of regular models

The work of You and Yuan (1994) shows a sufficient condition for the unicity of regular models for well-founded stratification programs.

Theorem 15 (*Theorem 5.3(ii) of You and Yuan (1994)*)
Let P be a well-founded stratification program. If $\mathrm{dg}(P)$ has no positive cycle, P has a unique regular model.

Hereafter, we would like to show that Theorem 15 is also true for generic logic programs. Note however that the technique of using least fixpoint applied for negative cycles seems difficult to use for positive cycles because there is some program whose dependence graph has no positive cycle but the dependence graph of its least fixpoint can have positive cycle (e.g., $P = \{a \leftarrow c; b \leftarrow c; c \leftarrow \sim a, \sim b\}$). We here use another approach.

Theorem 16 (*Theorem 3.4 of Paulevé and Richard (2011)*)
Let f be a BN. If $\mathrm{ig}(f)$ has no positive cycle, then $\mathrm{astg}(f)$ has a unique attractor.

Theorem 17 (*Lemma 16 of Dietz et al. (2014)*)
Supported partial models of a tight logic program coincide with its stable partial models.

Lemma 18
Let P be a logic program and f be its encoded BN. If P is tight, then regular models of P coincide with $\leq_s$-minimal trap spaces of f.

Proof
Since P is tight, stable partial models of P coincide with supported partial models of P (i.e., 3-valued models of $\mathrm{cf}(P)$) by Theorem 17. Then regular models of P coincide with $\leq_s$-minimal supported partial models of P. We have that trap spaces of f coincide with supported trap spaces of P by Theorem 3. By Proposition 8, $\leq_s$-minimal supported partial models of P coincide with $\leq_s$-minimal supported trap spaces of P. Hence, regular models of P coincide with $\leq_s$-minimal trap spaces of f. □

Theorem 19
Let P be a logic program. If $\mathrm{dg}(P)$ has no positive cycle, then P has a unique regular model.

Proof
Since $\mathrm{dg}(P)$ has no positive cycle, $\mathrm{dg}^+(P)$ (i.e., the positive dependence graph of P) has no cycle, i.e., P is tight. Let f be the encoded BN of P. By Lemma 18, regular models of P coincide with $\leq_s$-minimal trap spaces of f. Since $\mathrm{ig}(f)$ is a sub-graph of $\mathrm{dg}(P)$, it also has no positive cycle. By Theorem 16, $\mathrm{astg}(f)$ has a unique attractor. Since every $\leq_s$-minimal trap space of f contains at least one attractor of $\mathrm{astg}(f)$ and f has at least one $\leq_s$-minimal trap space (Klarner et al. 2015), f has a unique $\leq_s$-minimal trap space. Hence, we can conclude that P has a unique regular model. □

Since a stable model is also a regular model, Theorem 19 implies that if $\mathrm{dg}(P)$ has no positive cycle, then P has at most one stable model. In addition, P may have no stable model because the unique regular model may be not 2-valued. This result seems to be

already known in the folklore of logic programming, but to the best of our knowledge, there is no existing formal proof for it except the one that we have directly proved recently in Trinh and Benhamou (2024).

4.5 Upper bound for number of regular models

To the best of our knowledge, there is no existing work connecting between regular models of a logic program and (positive/negative) feedback vertex sets of its dependence graph. In Trinh and Benhamou (2024), we have shown that $2^{|U^+|}$ is an upper bound for the number of stable models where U^+ is a positive feedback vertex set of the dependence graph. Since stable models are 2-valued regular models, we can naturally generalize this result for the case of regular models, i.e., $3^{|U^+|}$ is an upper bound for the number of regular models. The underlying intuition for the base of 3 is that in a regular model, an atom can be $\mathbf{t}$, $\mathbf{f}$, or $\mathbf{u}$.

Theorem 20
Let P be a logic program. Let U^+ be a positive feedback vertex set of $\mathrm{dg}(P)$. Then the number of regular models of P is at most $3^{|U^+|}$.

Sketch of proof
Since the full proof is long, we only show the sketch of proof due to the space limitation.

By Theorem 9, regular models of P coincide with $\leq_s$-minimal stable trap spaces of P. For any mapping $\widehat{I} : U^+ \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, we build a new logic program $\widehat{P}$ from P as follows. First, remove from P all the rules whose heads belong to U^+. Second, remove all the rules whose body formulas are false under the values of the atoms in U^+ and otherwise remove all the appearances of the atoms that are in U^+ and not assigned to $\mathbf{u}$ in $\widehat{I}$. Third, for any atom $a \in U^+$ such that $\widehat{I}(a) = \mathbf{u}$, add the rule $a \leftarrow \sim a$. We can see that the part of $\mathrm{tg}_{st}(P)$ induced by $\widehat{I}$ is isomorphic to $\mathrm{tg}_{st}(\widehat{P})$. Hence, $\leq_s$-minimal stable trap spaces of P induced by $\widehat{I}$ one-to-one correspond to those of $\widehat{P}$. U^+ intersects all positive cycles of $\mathrm{dg}(P)$. Every atom $a \in U^+$ such that $\widehat{I}(a) \neq \mathbf{u}$ is removed from $\mathrm{dg}(P)$. In the case that $a \in U^+$ and $\widehat{I}(a) = \mathbf{u}$, all the arcs ending at a are removed and an negative arc $(aa, \ominus)$ is added. It follows that $\mathrm{dg}(\widehat{P})$ has no positive cycle. By Theorem 19, $\widehat{P}$ has a unique $\leq_s$-minimal stable trap space. There are $3^{|U^+|}$ possible mappings $\widehat{I}$, thus we can conclude the theorem. □

Theorem 21 (Theorem 3.5 of Paulevé and Richard (2011))
Let f be a BN. Let U^+ be a positive feedback vertex set of $\mathrm{ig}(f)$. Then the number of attractors of $\mathrm{astg}(f)$ is at most $2^{|U^+|}$.

We observed that the bound of $3^{|U^+|}$ is too rough for many example programs in the literature. Then inspired by Theorem 21 for an upper bound for the number of attractors of an asynchronous BN, we obtain an interesting result for tight logic programs.

Theorem 22
Let P be a tight logic program. Let U^+ be a positive feedback vertex set of $\mathrm{dg}(P)$. Then the number of regular models of P is at most $2^{|U^+|}$.

Proof

Let f be the encoded BN of P. By Lemma 18, regular models of P coincide with $\leq_s$-minimal trap spaces of f. By definition, U^+ intersects all positive cycles of $\mathrm{dg}(P)$. Since $\mathrm{ig}(f)$ is a sub-graph of $\mathrm{dg}(P)$, every positive cycle of $\mathrm{ig}(f)$ is also a positive cycle of $\mathrm{dg}(P)$. Hence, U^+ is also a positive feedback vertex set of $\mathrm{ig}(f)$. By Theorem 21, the number of attractors of $\mathrm{astg}(f)$ is at most $2^{|U^+|}$. Since the number of $\leq_s$-minimal trap spaces of f is a lower bound of the number of attractors of $\mathrm{astg}(f)$ (Klarner et al. 2015), the number of regular models of P is at most $2^{|U^+|}$. □

Note however that the question if $2^{|U^+|}$ is also an upper bound for the number of regular models in any logic program remains open.

5 Conclusion and future work

In this work, we show three main results relating graphical conditions of a normal logic program and its regular models: 1) the existence of negative cycles is a necessary condition for the existence of non-trivial regular models, 2) the absence of positive cycles is a sufficient condition for the unicity of regular models, and 3) an upper bound (resp. a finer upper bound) for the number of regular models in generic (resp. tight) logic programs based on positive feedback vertex sets. The first two conditions generalize the existing results obtained by You and Yuan (1994) for well-founded stratification logic programs. The third result is new to the best of our knowledge. Key to our proofs is a connection that we establish between logic programs and Boolean network theory, bridged by the trap space semantics of logic programs. Furthermore, the established connection can provide a unified framework for exploring and proving more new graphical conditions for models in logic programs via exploiting a plenty of existing results in Boolean networks.

As for future work, we would like to explore more new results on relating the dependence graph and models of a logic program. The results presented in this paper use only information on either positive cycles or negative cycles. It is then natural to think that by using both kinds of cycles simultaneously, we can obtain improved results. In addition, we also conjecture that the upper bound for tight logic program is also valid for generic logic programs. However, it seems quite difficult to prove this conjecture. Finally, it would be important to find efficient methods for computing regular models.

Acknowledgments

This work was supported by Institut Carnot STAR, Marseille, France.

References

Apt, K. R. and Bezem, M. 1991. Acyclic programs. *New Gener. Comput.*, *9*, 335–363.

Aravindan, C. and Dung, P. M. 1995. On the correctness of unfold/fold transformation of normal and extended logic programs. *J. Log. Program.*, *24*, 3, 201–217.

Baroni, P., Caminada, M., and Giacomin, M. 2011. An introduction to argumentation semantics. *Knowl. Eng. Rev.*, *26*, 4, 365–410.

Costantini, S. 2006. On the existence of stable models of non-stratified logic programs. *Theory Pract. Log. Program.*, *6*, 1-2, 169–212.

Costantini, S. and Provetti, A. Conflict, consistency and truth-dependencies in graph representations of answer set logic programs. In *Second International Workshop on Graph Structures for Knowledge Representation and Reasoning* 2011, pp. 68–90. Springer.

Dietz, E., Hölldobler, S., and Wernhard, C. 2014. Modeling the suppression task under weak completion and well-founded semantics. *J. Appl. Non Class. Logics*, *24*, 1-2, 61–85.

Dimopoulos, Y. and Torres, A. 1996. Graph theoretical structures in logic programs and default theories. *Theor. Comput. Sci.*, *170*, 1-2, 209–244.

Doyle, J. 1979. A truth maintenance system. *Artif. Intell.*, *12*, 3, 231–272.

Dung, P. M. and Kanchanasut, K. A fixpoint approach to declarative semantics of logic programs. In *Proc. of NACLP* 1989, pp. 604–625. MIT Press.

Eiter, T., Leone, N., and Saccà, D. 1997. On the partial semantics for disjunctive deductive databases. *Ann. Math. Artif. Intell.*, *19*, 1-2, 59–96.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Methods Log. Comput. Sci.*, *1*, 1, 51–60.

Fandinno, J. and Lifschitz, V. 2023. Positive dependency graphs revisited. *Theory Pract. Log. Program.*, *23*, 5, 1128–1137.

Gelfond, M. and Lifschitz, V. The stable model semantics for logic programming. In *Proc. of ICLP* 1988, pp. 1070–1080. MIT Press.

Inoue, K. and Sakama, C. Oscillating behavior of logic programs. In *Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz* 2012, pp. 345–362. Springer.

Janhunen, T., Niemelä, I., Seipel, D., Simons, P., and You, J. 2006. Unfolding partiality and disjunctions in stable model semantics. *ACM Trans. Comput. Log.*, *7*, 1, 1–37.

Klarner, H., Bockmayr, A., and Siebert, H. 2015. Computing maximal and minimal trap spaces of Boolean networks. *Nat. Comput.*, *14*, 4, 535–544.

Linke, T. Graph theoretical characterization and computation of answer sets. In *Proc. of IJCAI* 2001, pp. 641–648. Morgan Kaufmann.

Murata, T. 1989. Petri nets: Properties, analysis and applications. *Proc. IEEE*, *77*, 4, 541–580.

Paulevé, L. and Richard, A. Static analysis of Boolean networks based on interaction graphs: A survey. In *Proc. of SASB* 2011, pp. 93–104. Elsevier.

Przymusinski, T. C. 1990. The well-founded semantics coincides with the three-valued stable semantics. *Fundam. Inform.*, *13*, 4, 445–463.

Richard, A. 2019. Positive and negative cycles in Boolean networks. *J. Theor. Biol.*, *463*, 67–76.

Schwab, J. D., Kühlwein, S. D., Ikonomi, N., Kühl, M., and Kestler, H. A. 2020. Concepts in Boolean network modeling: What do they all mean? *Comput. Struct. Biotechnol. J.*, *18*, 571–582.

Thomas, R. and d'Ari, R. 1990. *Biological feedback*. CRC press.

Trinh, V., Benhamou, B., and Paulevé, L. 2024a. mpbn: a simple tool for efficient edition and analysis of elementary properties of Boolean networks. *CoRR*, *abs/2403.06255*a.

Trinh, V.-G. and Benhamou, B. 2024. Static analysis of logic programs via Boolean networks. Submitted paper.

Trinh, V.-G., Benhamou, B., and Soliman, S. 2023. Trap spaces of Boolean networks are conflict-free siphons of their Petri net encoding. *Theor. Comput. Sci.*, *971*, 114073.

Trinh, V.-G., Benhamou, B., Soliman, S., and Fages, F. 2024b. On trap space semantics of logic programs. Submitted paper.

Wu, Y., Caminada, M., and Gabbay, D. M. 2009. Complete extensions in argumentation coincide with 3-valued stable models in logic programming. *Stud Logica*, *93*, 2-3, 383–403.

You, J. and Yuan, L. 1994. A three-valued semantics for deductive databases and logic programs. *J. Comput. Syst. Sci.*, *49*, 2, 334–361.