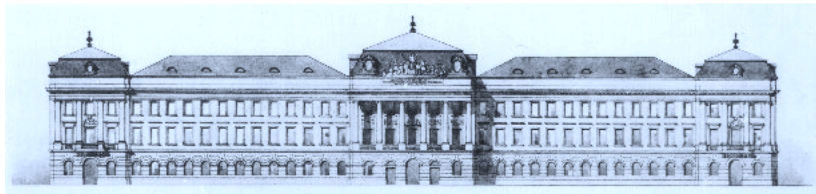
I N F S Y S

R E S E A R C H

R E P O R T



INSTITUT FÜR INFORMATIONSSYSTEME

ARBEITSBEREICH WISSENSBASIERTE SYSTEME

SEMI-EQUILIBRIUM MODELS FOR PARACOHERENT ANSWER SET PROGRAMS

GIOVANNI AMENDOLA   THOMAS EITER   MICHAEL FINK
NICOLA LEONE   JOÃO MOURA

INFSYS RESEARCH REPORT 14-02
DECEMBER 2014

Institut für Informationssysteme
AB Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 14-02, DECEMBER 2014

SEMI-EQUILIBRIUM MODELS FOR PARACOHERENT ANSWER SET PROGRAMS

Giovanni Amendola[1] Thomas Eiter[2] Michael Fink[2] Nicola Leone[1]
João Moura[3]

Abstract. The answer set semantics may assign a logic program no model, due to logical contradiction or unstable negation, which is caused by cyclic dependency of an atom from its negation. While logical contradictions can be handled with traditional techniques from paraconsistent reasoning, instability requires other methods. We consider resorting to a paracoherent semantics, in which 3-valued interpretations are used where a third truth value besides true and false expresses that an atom is believed true. This is at the basis of the semi-stable model semantics, which was defined using a program transformation. In this paper, we give a model-theoretic characterization of semi-stable models, which makes the semantics more accessible. Motivated by some anomalies of semi-stable model semantics with respect to basic epistemic properties, we propose an amendment that satisfies these properties. The latter has both a transformational and a model-theoretic characterization that reveals it as a relaxation of equilibrium logic, the logical reconstruction of answer set semantics, and is thus called the semi-equilibrium model semantics. We consider refinements of this semantics to respect modularity in the rules, based on splitting sets, the major tool for modularity in modeling and evaluating answer set programs. In that, we single out classes of canonical models that are amenable for customary bottom-up evaluation of answer set programs, with an option to switch to a paracoherent mode when lack of an answer set is detected. A complexity analysis of major reasoning tasks shows that semi-equilibrium models are harder than answer sets (i.e., equilibrium models), due to a global minimization step for keeping the gap between true and believed true atoms as small as possible. Our results contribute to the logical foundations of paracoherent answer set programming, which gains increasing importance in inconsistency management, and at the same time provide a basis for algorithm development and integration into answer set solvers.

[1]Dip. di Matematica e Informatica, Universitá della Calabria, Rende (CS), Italy; {amendola,leone}@mat.unical.it
[2]Institut für Informationssysteme, TU Wien, Vienna, Austria; {eiter,fink}@kr.tuwien.ac.at.
[3]CENTRIA, Universidade Nova de Lisboa, Portugal; joaomoura@yahoo.com

Acknowledgements: This work was partially supported by Regione Calabria under the EU Social Fund and project PIA KnowRex POR FESR 2007- 2013, by the Vienna Science and Technology Fund (WWTF) grant ICT 08-020, the Austrian Science Fund (FWF) grant P20841, and by the Italian Ministry of University and Research under PON project Ba2Know (Business Analytics to Know) S.I.-LAB n. PON03PE_0001. The work of J. Moura was supported by grant SFRH/BD/69006/2010 from Fundação para a Ciência e Tecnologia (FCT) from the Portuguese Ministério do Ensino e da Ciência.

Some of the results were presented in preliminary form at the 12th Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR 2010) and the 14th European Conference on Logics in AI (JELIA 2014)

Contents

1 Introduction

Answer Set Programming (ASP) is a premier formalism for nonmonotonic reasoning and knowledge representation, mainly because of the existence of efficient solvers and well-established relationships to common nonmonotonic logics. It is a declarative programming paradigm with a model-theoretic semantics, where problems are encoded into a logic program using rules, and its models, called answer sets (or stable models) [20], encode solutions; see [6, 10, 18].

As well-known, not every logic program has some answer set. This can be due to different reasons: (1) an emerging logical contradiction, as e.g. for the program

$$P = \{\, \mathit{locked}(\mathit{door}) \leftarrow \mathit{not}\ \mathit{open}(\mathit{door});\ -\mathit{locked}(\mathit{door}) \,\}$$

where "$-$" denotes strong (sometimes also called classical) negation and "not" denotes weak (or default negation); according to the first rule, a door is locked unless it is known to be open, and according to the second rule it is not locked. The problem here is a missing connection from $-\mathit{locked}(\mathit{door})$ to $\mathit{open}(\mathit{door})$.[1] (2) Due to cyclic dependencies which pass through negation, as e.g. in the following simplistic program.

Example 1 Russell's paradox is captured by the logic program

$$P = \{\mathit{shaves}(\mathit{joe}, \mathit{joe}) \leftarrow \mathit{not}\ \mathit{shaves}(\mathit{joe}, \mathit{joe})\}$$

(where joe is the barber), which informally states that Joe shaves himself if we can assume that he is not shaving himself. Under answer set semantics, P has no model; the problem is a lack of stability, as either assumption on whether $\mathit{shaves}(\mathit{joe}, \mathit{joe})$ is true or false can not be justified by the rule.

In general, the absence of an answer set may be well-accepted and indicate that the rules can not be satisfied under stable negation. There are nonetheless many cases when this is not intended and one might want to draw conclusions also from a program without answer sets, e.g., for debugging purposes, or in order to keep a system (partially) responsive in exceptional situations; in particular, if the contradiction or instability is not affecting parts of a system.

In order to deal with this, Inoue and Sakama [38] have introduced paraconsistent semantics for answer set programs. While dealing with logical contradictions can be achieved with similar methods as for (non-) classical logic (cf. also [8, 1, 27]), dealing with cyclic default negation turned out to be tricky. We concentrate in this article on the latter, in presence of constraints, and refer to it as *paracoherent reasoning*, in order to distinguish reasoning under logical contradictions from reasoning on programs without strong negation that lack stability in models.

With the idea that atoms may also be possibly true (i.e., believed true), Inoue and Sakama defined a semi-stable semantics which for the program in Example 1 has a model in which $\mathit{shaves}(\mathit{joe}, \mathit{joe})$ is believed true; this (arguably) is reasonable, as $\mathit{shaves}(\mathit{joe}, \mathit{joe})$ can not be false while satisfying the rule. Note however that believing $\mathit{shaves}(\mathit{joe}, \mathit{joe})$ is true does not provide a proof that this fact is true in reality; as a mere belief it is regarded to be weaker than if $\mathit{shaves}(\mathit{joe}, \mathit{joe})$ would be a fact or derived from a rule.

In fact, semi-stable semantics *approximates* answer set semantics and coincides with it whenever a program has some answer set; otherwise, under Occam's razor it yields models with a least set of atoms believed to be true. That is, the intrinsic *closed world assumption (CWA)* of logic programs is slightly relaxed for achieving stability of models.

In a similar vein, we can regard many semantics for non-monotonic logic programs that relax answer sets as *paracoherent semantics*, e.g. [4, 16, 28, 32, 33, 36, 37, 39, 42, 45]. Ideally, such a relaxation meets for a program P the following properties:

[1] Constraints (rules with empty head) may be considered as logical contradictions, if $\perp$ (falsum) is added to the head; however, also an instability view is possible, cf. Section 6.2.

(D1) Every (consistent) answer set of P corresponds to a model (*answer set coverage*).

(D2) If P has some (consistent) answer set, then its models correspond to answer sets (*congruence*).

(D3) If P has a classical model, then P has a model (*classical coherence*).

In particular, (D3) intuitively says that in the extremal case, a relaxation should renounce to the selection principles imposed by the semantics on classical models (in particular, if a single classical model exists).

Widely-known semantics, such as 3-valued stable models [36], L-stable models [16], revised stable models [32], regular models [45], and pstable models [28], satisfy only part of these requirements (see Section 8.2 for more details). Semi-stable models however, satisfy all three properties and thus have been the prevailing paracoherent semantics.

1.1 Application scenario: inconsistency-tolerant query answering

Paracoherent semantics may be fruitfully employed in different use cases of ASP, such as model building respectively scenario generation, but also traditional reasoning from the models of a logical theory.

The standard answer set semantics may be regarded as appropriate when a knowledge base, i.e., logic program, is properly specified adopting the CWA principle to deal with incomplete information. Query answering over a knowledge base then resorts usually to brave or cautious inference from the answer sets of a knowledge base; let us focus on the latter here. However, if (unexpected) incoherence arises, then we lose all information and query answers are trivial. This, however, may not be satisfactory, especially if one can not modify the knowledge base, which may be due to various reasons. Paracoherent semantics can be exploited to overcome this problem and to render query answering operational, without trivialization and inference explosion. In particular, semi-stable model semantics is attractive as it (1) brings in "unsupported" assumptions, (2) remains close to answer sets in model building, but distinguishes atoms that require such assumptions from atoms derivable without them, and (3) keeps the CWA/LP spirit of minimal assumptions.

Example 2 Consider a variant of the Russell's paradox, cf. [38]:

$$P = \{shaves(joe, X) \leftarrow not\, shaves(X, X);\ man(paul)\}.$$

While this program has no answer set, the semi-stable model semantics gives us the model $\{shaves(joe, paul)$, $man(paul)$, $K shaves(joe, joe)\}$, in which $shaves(joe, joe)$ is believed to be true (as expressed by the prefix 'K'); here the incoherent rule $shaves(joe, joe) \leftarrow not\, shaves(joe, joe)$, which is an instance of the rule in P for joe, is isolated from the rest of the program to avoid the absence of models;[2] this treatment allows us to derive, for instance, that $shaves(joe, paul)$ and $man(paul)$ are true; furthermore, we can infer that $shaves(joe, joe)$ can not be false. Such a capability seems to be very attractive in query answering.

The well-founded semantics (WFS) [42], which is the most prominent approximation of the answer set semantics, has similar capabilities, but takes intuitively a coarser view on the truth value of an atom, which can be either true, false, or undefined; in semi-stable semantics, however, undefinedness has a bias towards truth, expressed by "believed true"(or stronger, by "must be true"); in the example above, under WFS $shaves(joe, joe)$ would be undefined. Furthermore, undefinedness is cautiously propagated, which may prevent one from drawing expected conclusions.

Example 3 Consider the following extension of Russell's paraphrase:

[2] A similar intuition underlies the CWA inhibition rule in [31] that is used for contradiction removal in logic programs

$$P = \left\{ \begin{array}{l} shaves(joe, joe) \leftarrow not\, shaves(joe, joe); \\ visits\_barber(joe) \leftarrow not\, shaves(joe, joe) \end{array} \right\}.$$

Arguably one expects that $visits\_barber(joe)$ is concluded false from this program: to satisfy the first rule, $shaves(joe, joe)$ can not be false, and thus the second rule can not be applied; thus under CWA, $visits\_barber(joe)$ should be false. However, under well-founded semantics all atoms are undefined; in particular, the undefinedness of $shaves(joe, joe)$ is propagated to $visits\_barber(joe)$ by the second rule.

The single semi-stable model of P from its epistemic transformation is $\{K shaves(joe, joe)\}$, according to which $shaves(joe, joe)$ is believed true while $visits\_barber(joe)$ is false.

Furthermore, it is well-known that the well-founded semantics has problems with reasoning by cases.

Example 4 From the program

$$P = \left\{ \begin{array}{l} shaves(joe, joe) \leftarrow not\, shaves(joe, joe); \\ angry(joe) \leftarrow not\, happy(joe);\ happy(joe) \leftarrow not\, angry(joe); \\ smokes(joe) \leftarrow angry(joe);\ smokes(joe) \leftarrow happy(joe) \end{array} \right\},$$

which is incoherent, we can not conclude that $smokes(joe)$ is true under WFS, while we can do so under semi-stable semantics and its relatives.

We elucidate the relationship between paracoherent semantics and WFS in more detail in Section 8.

1.2 Contributions

Despite the model-theoretic nature of ASP, semi-stable models have been defined by means of a program transformation, called epistemic transformation. A semantic characterization in the style of equilibrium models for answer sets [30] was still missing. Such a characterization was desired because working with program transforms becomes cumbersome, if properties of semi-stable models should be assessed; and moreover, while the program transform is declarative and the intuition behind it is clear, the interaction of the rules does not make it easy to understand or see how the semantics works in particular cases.

Starting out from these observations, we have addressed the problem and make the following main contributions.

– We characterize semi-stable models by pairs of 2-valued interpretations of the original program, similar to so-called here-and-there (HT) models in equilibrium logic [29, 30]. In the course of this, we point out some anomalies of the semi-stable semantics with respect to basic rationality properties in modal logics (**K** and **N**) which essentially prohibit a 1-to-1 characterization[3] in terms of HT-models. Roughly speaking, the epistemic transformation misses some links between atoms encoding truth values of atoms, which may lead in some cases to unintuitive results.

– The anomalies lead us to propose an alternative paracoherent semantics, called *semi-equilibrium ($\mathcal{SEQ}$) model semantics*, which remedies the anomalies of the semi-stable model semantics. It satisfies the properties (D1)-(D3) from above and is fully characterized using HT-models. Informally, semi-equilibrium models are 3-valued interpretations in which atoms can be true, false or believed true; the gap between believed and (derivably) true atoms is globally minimized. Note that the semantic distinction between believed true and true atoms in models is important. Other approaches, e.g. CR-Prolog [4], make a syntactic distinction at the rule level which does not semantically discriminate believed atoms; this may lead to more models. Notably, $\mathcal{SEQ}$-models can be obtained by an extension of the epistemic transformation that adds further rules.

[3] By 1-to-1 we mean a one to one and onto (i.e., bijective) correspondence.

– Resorting to splitting sets [24], the major tool for modularity in modeling and evaluating answer set programs, we define *split $\mathcal{SEQ}$-models*, for which the program is evaluated in progressive layers according to a *splitting sequence* $S = (S_1, \ldots, S_n)$ of the atoms. This is motivated by the fact that answer set program evaluation typically proceeds from bottom to top modules, and that switching to a paracoherent mode on the encounter of incoherence is possible on the fly. E.g., the program

$$P = \{shaves(joe, joe) \leftarrow not\ woman(joe), adult(joe), not\ shaves(joe, joe);\ adult(joe)\}$$

has two $\mathcal{SEQ}$-models: in both $adult(joe)$ is true, and in one $shaves(joe, joe)$ is believed true and $woman(joe)$ is false, while in the other $woman(joe)$ is believed true and thus $shaves(joe, joe)$ is false. Of these two models, in lack of any further information the first is more appealing, as there is no rule from which $woman(joe)$ could be derived.

– In general, the resulting split $\mathcal{SEQ}$-models depend on the particular splitting sequence. We thus introduce *canonical splitting sequences*, with the property that the models are *independent* of any particular member from a class of splitting sequences, and thus yield canonical models (Section 6). This is analogous to the *perfect models* of a (disjunctive) stratified program, which are independent of a concrete stratification [3, 35]. For programs P with a benign form of constraints, the class derived from the strongly connected components (SCCs) of P warrants this property, as well as modularity properties. For arbitrary programs, independence is held by a similar class derived from the maximal joined components (MJCs) of P, which merge SCCs involved in malign constraints.

– We study major reasoning tasks for the semantics above and provide precise characterizations of their computational complexity for various classes of logic programs. Besides brave and cautious reasoning, deciding whether a program has a model, respectively recognizing models, is considered. Briefly, the results show that semi-stable and $\mathcal{SEQ}$-model semantics reside in the polynomial hierarchy one level above the answer set semantics, and is for brave and cautious reasoning from disjunctive programs Σ_3^p- respectively Π_3^p-complete; for normal programs, the problems are Σ_2^p- respectively Π_2^p-complete. Notably, split $\mathcal{SEQ}$- and canonical $\mathcal{SEQ}$-models have the same complexity as $\mathcal{SEQ}$-models for these problems, but the model existence problem (which is NP-complete for SEQ-models) is harder (Σ_3^p- resp. Σ_2^p-complete).

– We compare the $\mathcal{SEQ}$-model semantics to a number of related semantics in the literature. It turns out that it coincides with the evidential stable model semantics for disjunctive logic programs [39], which has been defined like the semi-stable model semantics in terms of a two stage program transformation, but using a rather different program. Thus our results provide as a byproduct also a semantic and computational characterization of the evidential stable model semantics.

Our results contribute to enhanced logical foundations of paracoherent answer set programming, which gains increasing importance in inconsistency management. They provide a model-theoretic characterization and an amendment of the semi-stable semantics, given by the semi-equilibrium semantics, linking it to the view of answer sets semantics in equilibrium logic; this provides the basis for immediate extensions to richer classes of logic programs (see Section 9.1). Furthermore, the split $\mathcal{SEQ}$-model semantics, and in particular the $\mathcal{SCC}$-models semantics, lends itself for a modular use and bottom up evaluation of programs. Cautious merging of components, as done for $\mathcal{MJC}$-models, aims at preserving independence of components and thus possible parallel evaluation. This makes the refined semantics attractive for incorporation into answer set solvers and evaluation frameworks, in order to offer paracoherent features.

1.2.1 Organization

The remainder of this article is organized as follows. In the next section, we review answer set programs, equilibrium logic and semi-stable model semantics. After that, we provide in Section 3 the semantic characterization of semi-stable models and point out some anomalies, which leads us to introduce semi-equilibrium models in Section 4. The refinement of the latter relative to splitting sets and arbitrary splitting sequences is considered in Section 5, while canonical semi-equilibrium models are introduced in Section 6. Section 7 is devoted to characterize the complexity of various semantics and to computational issues in this context. Related work is discussed in Section 8, followed by Section 9 that addresses possible extensions. Section 10 concludes the article with open issues and an outlook. In order not to disrupt the flow of reading, most proofs have been moved to the Appendix.

2 Preliminaries

In this paper, we consider a propositional setting of logic programs; extensions to the usual non-ground setting are straightforward. Since we are primarily interested in paracoherence, we also disregard aspects devoted to paraconsistency, i.e., logical contradictions; more specifically, we exclude strong negation. A discussion of how the work extends to non-ground programs and strong negation is given in Section 9.1)

We first recall the answer set semantics of disjunctive logic programs, and then its reconstruction as equilibrium logic based on a non-classical logic.

2.1 Answer Set Programs

Given a propositional signature, i.e., a set of propositional atoms Σ, a (*disjunctive*) rule r is of the form

$$a_1 \vee \cdots \vee a_l \ \leftarrow \ b_1, \ldots, b_m, not\, c_1, \ldots, not\, c_n, \qquad (1)$$

where $l + m + n > 0$, such that all a_i, b_j and c_k are atoms. As usual, "not" stands for *weak* or *default negation*. The *head* of r is the set $H(r) = \{a_1, \ldots, a_l\}$, and the *positive* respectively *negative body* is the set $B^+(r) = \{b_1, \ldots, b_m\}$ respectively $B^-(r) = \{c_1, \ldots, c_n\}$; the *body* of r is $B(r) = B^+(r) \cup not\, B^-(r)$, where for any set S of atoms, $not\, S = \{not\, a \mid a \in S\}$. Furthermore, $At(r) = H(r) \cup B^+(r) \cup B^-(r)$ denotes the set of all atoms occurring in r. In abuse of notation, we will denote r also by

$$H(r) \leftarrow B(r) \quad \text{ or } \quad H(r) \leftarrow B^+(r), not\, B^-(r).$$

A rule r is a *(disjunctive) fact*, if $B(r) = \emptyset$ (we then omit $\leftarrow$); a *constraint*, if $H(r) = \emptyset$; *normal*, if $|H(r)| \leq 1$; and *positive*, if $B^-(r) = \emptyset$.

A *(disjunctive logic) program* P is a finite set of disjunctive rules (over Σ). A program P is called *normal* (resp. *positive*) if each $r \in P$ is normal (resp. positive); P is *constraint-free*, if P contains no constraints. We let $At(P) = \bigcup_{r \in P} At(r)$ and by default $\Sigma = At(P)$.

An *interpretation* is any set $I \subseteq \Sigma$ of atoms. An interpretation I *satisfies* a rule r, denoted $I \models r$, if $I \cap H(r) \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$, and I is a *model* of a program P (denoted $I \models P$), if $I \models r$ for each rule $r \in P$. A model I of P is *minimal*, if no model $J \subset I$ of P exists; $MM(P)$ denotes the set of all minimal models of P.

An interpretation I is a *stable model* (or *answer set*) of P, if $I \in MM(P^I)$, where P^I is the well-known *Gelfond-Lifschitz (GL) reduct* [20] of P w.r.t. I, which is the positive program $P^I = \{H(r) \leftarrow B^+(r) \mid r \in P, B^-(r) \cap I \neq \emptyset\}$. We denote by $\mathcal{AS}(P)$ the set of all answer sets of P.

Example 5 Consider the program $P = \{b \vee c \leftarrow \mathit{not}\, a;\ d \leftarrow c, \mathit{not}\, b\}$. It has the minimal models $MM(P) = \{\{a\}, \{b\}, \{c, d\}\}$, the answer sets $\mathcal{AS}(P) = \{\{b\}, \{c, d\}\}$; note that $I = \{a\}$ is not an answer set as M is not a minimal model of $P^I = \{d \leftarrow c\}$.

Recall that the *dependency graph* of a program P is the directed graph $DG(P) = \langle V_{DG}, E_{DG}\rangle$ with nodes $V_{DG} = At(P)$ and edges $E_{DG} = \{(a, b) \mid a \in H(r), b \in B^+(r) \cup B^-(r) \cup (H(r) \setminus \{a\}), r \in P\}$. The *strongly connected components (SCCs)* of P, denoted $\mathcal{SCC}(P)$, are the SCCs of $DG(P)$, i.e. the maximal node sets $C \subseteq At(P)$ such that every pair of nodes $v, v' \in C$ is connected by some path in G with nodes only from C.

A program P is *stratified*, if for each $r \in P$ and $C \in \mathcal{SCC}(P)$ either $H(r) \cap C = \emptyset$ or $B^-(r) \cap C = \emptyset$; P is *headcycle-free (hcf)*, if $|H(r) \cap C| \leq 1$ for each $r \in P$ and $C \in \mathcal{SCC}(P')$, where $P' = \{a \leftarrow B^+(r) \mid r \in P,\ a \in H(r)\}$.

Example 6 (cont'd) The program $P = \{b \vee c \leftarrow \mathit{not}\, a;\ d \leftarrow c, \mathit{not}\, b\}$ is stratified and also headcycle-free.

2.1.1 Splitting sets and sequences

Splitting sets [24] allow one to divide a program P into a lower and a higher part which can be evaluated bottom up. More formally, a set $S \subseteq \Sigma$ is a *splitting set* of P, if for every rule r in P such that $H(r) \cap S \neq \emptyset$ we have that $At(r) \subseteq S$. We denote by $b_S(P) = \{r \in P \mid At(r) \subseteq S\}$ the *bottom* part of P, and by $t_S(P) = P \setminus b_S(P)$ the *top* part of P relative to S. Note that the union $S = S_1 \cup S_2$ of splitting sets S_1, S_2 of a program P is also a splitting set of P.

As shown in [24], it holds that (where as usual, "$\cup M$" means adding all atoms in M as facts)

$$\mathcal{AS}(P) = \bigcup_{M \in \mathcal{AS}(b_S(P))} \mathcal{AS}(t_S(P) \cup M). \tag{2}$$

Example 7 (cont'd) For the program $P = \{b \vee c \leftarrow \mathit{not}\, a;\ d \leftarrow c, \mathit{not}\, b\}$, the set $S = \{a, b, c\}$ is a splitting set, and we have $b_S(P) = \{b \vee c \leftarrow \mathit{not}\, a\}$ and $t_S(P) = \{d \leftarrow c, \mathit{not}\, b\}$; as $\mathcal{AS}(P) = \{\{b\}, \{c\}\}$, we get $\mathcal{AS}(P) = \mathcal{AS}(b_S(P) \cup \{b\}) \cup \mathcal{AS}(b_S(P) \cup \{c\}) = \{\{b\}, \{c, d\}\}$.

Splitting sets naturally lead to splitting sequences. A *splitting sequence* $S = (S_1, \ldots, S_n)$ of P is a sequence of splitting sets S_i of P such that $S_i \subseteq S_j$ for each $i < j$; note that usually $S_n \subset \Sigma$; the characterization in (2) can be extended accordingly. With an eye on practical implementation, we do not consider infinite splitting sequences here, but will comment on them at the end of Section 5.

Example 8 (cont'd) A splitting sequence for $P = \{b \vee c \leftarrow \mathit{not}\, a;\ d \leftarrow c, \mathit{not}\, b\}$ is $S = (S_1, S_2)$ where $S_1 = \{a\}$ and $S_2 = \{a, b, c\}$; $b_{S_1}(P) = \emptyset$, $b_{S_2}(P) = \{b \vee c \leftarrow \mathit{not}\, a\}$ and $t_{S_2}(P) = \{d \leftarrow c, \mathit{not}\, b\}$.

2.2 Equilibrium Logic

The *logic of here-and-there* (HT) [30] serves as a valuable basis for characterizing semantic properties of ASP. It is an intermediate logic between (full) intuitionistic and classical logic, and it coincides with 3-valued Gödel logic. As such, it considers a full language $\mathcal{L}_{\pm}$ of formulas built over a propositional signature Σ with the connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\perp$. We restrict our attention here to formulas of the form

$$b_1 \wedge \ldots \wedge b_m \wedge \neg b_{m+1} \wedge \ldots \wedge \neg b_n \rightarrow a_1 \vee \cdots \vee a_l, \tag{3}$$

which correspond in a natural way to rules of form (1); every program P corresponds then similarly to a theory (set of formulas) Γ_P. For example, the program $P = \{a \leftarrow b;\ b \leftarrow \mathit{not}\, c;\ c \leftarrow \mathit{not}\, a\}$, corresponds to the theory $\Gamma_P = \{b \rightarrow a;\ \neg c \rightarrow b;\ \neg a \rightarrow c\}$. In the rest of the article, we tacitly use this correspondence. We note, however, that the key notions extend to the full language, and in this way some of the results to extensions of the rule language that we consider (see Section 9.1).

As a restricted intuitionistic logic, HT can be semantically characterized by Kripke models, in particular using just two worlds, namely "*here*" and "*there*" (assuming that the *here* world is ordered before the *there* world). An *HT-interpretation* is a pair (X, Y) of interpretations $X, Y \subseteq \Sigma$ such that $X \subseteq Y$; it is *total*, if $X = Y$. Intuitively, atoms in X (the *here* part) are considered to be true, atoms not in Y (the *there* part) to be false, while the remaining atoms (from $Y \setminus X$) are undefined.

Assuming that $X \models F$ denotes satisfaction of a formula ϕ by an interpretation X in classical logic, satisfaction of F in HT-logic (an HT-model), denoted $(X, Y) \models \phi$, is defined recursively as follows:

1. $(X, Y) \models a$ if $a \in X$, for any atom a,
2. $(X, Y) \not\models \bot$,
3. $(X, Y) \models \neg\phi$ if $Y \not\models \phi$,[4]
4. $(X, Y) \models \phi \wedge \psi$ if $(X, Y) \models \phi$ and $(X, Y) \models \psi$,
5. $(X, Y) \models \phi \vee \psi$ if $(X, Y) \models \phi$ or $(X, Y) \models \psi$,
6. $(X, Y) \models \phi \rightarrow \psi$ if (i) $(X, Y) \not\models \phi$ or $(X, Y) \models \psi$, and (ii) $Y \models \phi \rightarrow \psi$.

Then, an HT-interpretation (X, Y) is a model of a theory Γ, denoted $(X, Y) \models \Gamma$, if $(X, Y) \models \phi$ for every formula $\phi \in \Gamma$.

In particular, $(X, Y) \models \neg a$ iff $a \notin Y$, and $(X, Y) \models r$ for a rule r of form (1) iff either $H(r) \cap X \neq \emptyset$, or $B^+(r) \not\subseteq Y$, or $B^-(r) \cap Y \neq \emptyset$; in terms of the GL-reduct, we have $(X, Y) \models P$ for a program P iff $Y \models P$ and $X \models P^Y$ [41].

A total HT-interpretation (Y, Y) is an *equilibrium model* of a theory Γ, if $(Y, Y) \models \Gamma$ and for every HT-interpretation (X, Y), such that $X \subset Y$, it holds that $(X, Y) \not\models \Gamma$. For further details see, e.g., [30].

Example 9 (cont'd) For the program $P = \{b \vee c \leftarrow \mathit{not}\, a;\ d \leftarrow c, \mathit{not}\, b\}$, the sets $(\emptyset, a)$, (a, a), (b, b), $(\emptyset, ab)$, (a, ab), (b, bc), (c, bc), (cd, cd) are some HT-models (X, Y) of the corresponding theory Γ_P.[5] The equilibrium models of P resp. Γ_P are (b, b) and (cd, cd).

In the previous example, the program P has the answer sets $I_1 = \{b\}$ and $I_2 = \{c, d\}$, which amount to the equilibrium models (b, b) and (cd, cd), respectively. In fact, the answer sets and equilibrium models of a program always coincide.

Proposition 1 ([29]) *For every program P and $M \subseteq At(P)$, it holds that $M \in \mathcal{AS}(P)$ iff (M, M) is an $\mathcal{EQ}$-model of Γ_P.*

In particular, as $\mathcal{AS}(P) = MM(P)$ for any positive program P, we have $\mathcal{EQ}(P) = \{(M, M) \mid M \in MM(P)\}$ in this case.

We call a logic program *incoherent*, if it lack answer sets due to cyclic dependency of atoms among each other by rules through negation; that is, no answer set (equivalently, no equilibrium model) exists even if all constraints are dismissed from the program.

[4]That is, Y satisfies $\neg\phi$ classically; equivalently, $(X, Y) \models \phi \rightarrow \bot$.

[5]We write (as common) sets $\{a_1, a_2, \ldots, a_n\}$ as juxtaposition $a_1 a_2 \cdots a_n$ of their elements.

Example 10 Reconsider Russell's paradox; the HT-models of the respective program are $(\emptyset, a)$ and (a, a), where a stands for $shaves(joe, joe)$; the single total HT-model is (a, a), which however is not an equilibrium model. Similarly, the program $P = \{a \leftarrow b;\ b \leftarrow not\, a\}$ has the HT-models $(\emptyset, a)$, $(\emptyset, ab)$, (a, a), (a, ab), and (ab, ab); likewise, the total HT-models (a, a) and (ab, ab) are not equilibrium models.

We next recall the semi-stable model semantics which deals with such incoherence.

2.3 Semi-Stable Models

Inoue and Sakama [38] introduced *semi-stable models* as an extension of paraconsistent answer set semantics (called PAS semantics, respectively p-stable models by them) for extended disjunctive logic programs. Their aim was to provide a framework which is paraconsistent for incoherence, i.e., in situations where stability fails due to cyclic dependencies of a literal from its default negation.

We consider an extended signature $\Sigma^\kappa = \Sigma \cup \{Ka \mid a \in \Sigma\}$. Intuitively, Ka can be read as a is believed to hold. Semantically, we resort to subsets of Σ^κ as interpretations I^κ and the truth values false $\bot$,[6] believed true $\mathbf{bt}$, and true $\mathbf{t}$, where $\bot \preceq \mathbf{bt} \preceq \mathbf{t}$. The truth value assigned by I^κ to a propositional variable a is defined by

$$I^\kappa(a) = \begin{cases} \mathbf{t} & \text{if } a \in I^\kappa, \\ \mathbf{bt} & \text{if } Ka \in I^\kappa \text{ and } a \notin I^\kappa, \\ \bot & \text{otherwise.} \end{cases}$$

The semi-stable models of a program P are obtained from its *epistemic transformation* P^κ.

Definition 1 (Epistemic Transformation P^κ [38]) *Let P be a disjunctive program. Then its epistemic transformation is defined as the positive disjunctive program P^κ obtained from P by replacing each rule r of the form (1) in P, such that $B^-(r) \neq \emptyset$, with:*

$$\lambda_{r,1} \vee \ldots \vee \lambda_{r,l} \vee Kc_1 \vee \ldots \vee Kc_n \leftarrow b_1, \ldots, b_m, \quad (4)$$
$$a_i \leftarrow \lambda_{r,i}, \quad (5)$$
$$\leftarrow \lambda_{r,i}, c_j, \quad (6)$$
$$\lambda_{r,i} \leftarrow a_i, \lambda_{r,k}, \quad (7)$$

for $1 \leq i, k \leq l$ and $1 \leq j \leq n$, where the $\lambda_{r,i}$, $\lambda_{r,k}$ are fresh atoms.

Note that for any program P, its epistemic transformation P^κ is positive. Models of P^κ are defined in terms of a fixpoint operator in [38], with the property that for positive programs, according to Theorem 2.9, minimal fixpoints coincide with minimal models of the program. Therefore, for any program P, minimal fixpoints of P^κ coincide with answer sets of P^κ.

Semi-stable models are then defined as *maximal canonical* interpretations among the minimal fixpoints (answer sets) of P^κ as follows. For every interpretation I^κ over $\Sigma' \supseteq \Sigma^\kappa$, let $gap(I^\kappa) = \{Ka \in I^\kappa \mid a \notin I^\kappa\}$ denote the atoms believed true but not assigned true.

Definition 2 (maximal canonical) *Given a set $\mathcal{S}$ of interpretations over Σ', an interpretation $I^\kappa \in \mathcal{S}$ is* maximal canonical in $\mathcal{S}$, *if no $J^\kappa \in \mathcal{S}$ exists such that $gap(I^\kappa) \supset gap(J^\kappa)$. By $mc(\mathcal{S})$ we denote the set of maximal canonical interpretations in $\mathcal{S}$.*

Then we can equivalently paraphrase the definition of semi-stable models in [38] as follows.

[6] In [38] $\bot$ is called 'undefined', as it should be if strong negation is considered as well.

Definition 3 (semi-stable models) *Let P be a program over Σ. An interpretation I^κ over Σ^κ is a* semi-stable model *of P, if $I^\kappa = S \cap \Sigma^\kappa$ for some maximal canonical answer set S of P^κ. The set of all semi-stable models of P is denoted by $\mathcal{SST}(P)$, i.e., $\mathcal{SST}(P) = \{S \cap \Sigma^\kappa \mid S \in mc(\mathcal{AS}(P^\kappa))\}$.*

Example 11 Reconsider $P = \{a \leftarrow \mathit{not}\, a\}$, where a stands for $\mathit{shaves}(\mathit{joe}, \mathit{joe})$. Then $P^\kappa = \{\lambda_1 \vee Ka \leftarrow;\ a \leftarrow \lambda_1;\ \leftarrow a, \lambda_1;\ \lambda_1 \leftarrow a, \lambda_1\}$, which has the single answer $M = \{Ka\}$; hence, $\{Ka\}$ is the single semi-stable model of P.

Example 12 Consider the simple stratified program $P = \{b \leftarrow \mathit{not}\, a\}$. Its epistemic transformation is $P^\kappa = \{\lambda_1 \vee Ka \leftarrow;\ b \leftarrow \lambda_1;\ \leftarrow a, \lambda_1;\ \lambda_1 \leftarrow b, \lambda_1\}$, which has the answers sets $M_1 = \{Ka\}$ and $M_2 = \{\lambda_1, b\}$; as $\mathit{gap}(M_1) = \{a\}$ and $\mathit{gap}(M_2) = \emptyset$, among them M_2 is maximal canonical, and hence $M_2 \cap \Sigma^\kappa = \{b\}$ is the single semi-stable model of P. This is in fact also the unique answer set of P.

For a study of the semi-stable model semantics, we refer to [38]; notably,

Proposition 2 ([38]) *The $\mathcal{SST}$-models semantics, given by $\mathcal{SST}(P)$ for arbitrary programs P, satisfies properties (D1)-(D3).*

3 Semantic Characterization of Semi-Stable Models

As opposed to its transformational definition, we aim at a model-theoretic characterization of semi-stable models in the line of model-theoretic characterizations of the answer set semantics by means of HT.

Example 13 Reconsider $P = \{a \leftarrow \mathit{not}\, a\}$ in Example 11. The HT-models of P are $(\emptyset, \{a\})$ and $(\{a\}, \{a\})$. One might aim characterizing the semi-stable model by $(\emptyset, \{a\})$.

However, resorting to HT-interpretations will not uniquely characterize semi-stable models as illustrated next.

Example 14 Consider the program

$$P = \{a;\ b;\ c;\ d \leftarrow \mathit{not}\, a, \mathit{not}\, b;\ d \leftarrow \mathit{not}\, b, \mathit{not}\, c\}.$$

It is coherent, with a single answer set $\{a, b, c\}$, while $\mathcal{SST}(P) = \{\{a, b, c, Kb\}, \{a, b, c, Ka, Kc\}\}$. Note that neither $(\{a, b, c\}, \{b\})$ nor $(\{a, b, c\}, \{a, c\})$ is a HT-interpretation.

Hence, for a 1-to-1 characterization we have to resort to different structures. Sticking to the requirement that, given a program P over Σ, pairs of two-valued interpretations over Σ should serve as the underlying semantic structures, we say that a bi-interpretation of a program P over Σ is any pair (I, J) of interpretations over Σ, and define:

Definition 4 (*bi*-model) *Let ϕ be a formula over Σ, and let (I, J) be a bi-interpretation over Σ. Then, (I, J) is a* bi-model *of ϕ, denoted $(I, J) \models_\beta \phi$, if*

1. *$(I, J) \models_\beta a$ if $a \in I$, for any atom a,*
2. *$(I, J) \not\models_\beta \bot$,*
3. *$(I, J) \models_\beta \neg\phi$ if $J \not\models \phi$,*
4. *$(I, J) \models_\beta \phi \wedge \psi$ if $(I, J) \models_\beta \phi$ and $(I, J) \models_\beta \psi$,*

5. $(I, J) \models_\beta \phi \vee \psi$ *if* $(I, J) \models_\beta \phi$ *or* $(I, J) \models_\beta \psi$,

6. $(I, J) \models_\beta \phi \rightarrow \psi$ *if (i)* $(I, J) \not\models_\beta \phi$, *or (ii)* $(I, J) \models_\beta \psi$ *and* $I \models \phi$.

Moreover, (I, J) *is a bi-model of a program* P, *if* $(I, J) \models_\beta \phi$, *for all* ϕ *of the form (3) corresponding to a rule* $r \in P$.

In case of programs, its bi-models can alternatively be characterized by the following condition on its rules.

Proposition 3 *Let* r *be a rule over* Σ, *and let* (I, J) *be a bi-interpretation over* Σ. *Then,* $(I, J) \models_\beta r$ *if and only if* $B^+(r) \subseteq I$ *and* $J \cap B^-(r) = \emptyset$ *implies that* $I \cap H(r) \neq \emptyset$ *and* $I \cap B^-(r) = \emptyset$.

To every bi-model of a program P, we associate a corresponding interpretation $(I, J)^\kappa$ over Σ^κ by $(I, J)^\kappa = I \cup \{Ka \mid a \in J\}$. Conversely, given an interpretation I^κ over Σ^κ its associated bi-interpretation $\beta(I^\kappa)$ is given by $(I^\kappa \cap \Sigma, \{a \mid Ka \in I^\kappa\})$.

In order to relate these constructions to models of the epistemic transformation, which builds on additional atoms of the form $\lambda_{r,i}$, we construct an interpretation $(I, J)^{\kappa,P}$ of P^κ from a given bi-interpretation (I, J) of P as:

$$(I, J)^{\kappa,P} = (I, J)^\kappa \cup \{\lambda_{r,i} \mid r \in P, B^-(r) \neq \emptyset, a_i \in I, I \models B(r), J \models B^-(r)\},$$

where r is of the form (1).

Proposition 4 *Let* P *be a program over* Σ. *Then,*

(1) if (I, J) *is a bi-model of* P, *then* $(I, J)^{\kappa,P} \models P^\kappa$;

(2) if $M \models P^\kappa$ *then* $\beta(M \cap \Sigma^\kappa)$ *is a bi-model of* P.

Based on bi-models, we obtain a 1-to-1 characterization of semi-stable models by imposing suitable minimality criteria.

Theorem 5 *Let* P *be a program over* Σ. *Then,*

(1) if (I, J) *is a bi-model of* P *such that (i)* $(I', J) \not\models_\beta P$, *for all* $I' \subset I$, *(ii)* $(I, J') \not\models_\beta P$, *for all* $J' \subset J$, *and (iii) there is no bi-model* (I', J') *of* P *that satisfies (i) and* $gap(I', J') \subset gap(I, J)$, *then* $(I, J)^\kappa \in \mathcal{SST}(P)$;

(2) if $I^\kappa \in \mathcal{SST}(P)$, *then* $\beta(I^\kappa)$ *is a bi-model of* P *that satisfies (i)-(iii).*

Intuitively, Conditions (i) and (ii) filter bi-models that uniquely correspond to (some but not all) answer sets of P^κ: due to minimality every answer set satisfies (i); there may be answer sets of P^κ that do not satisfy (ii), but they are certainly not maximal canonical. Eventually, Condition (iii) ensures that maximal canonical answer sets are selected. More formally, the proof of this theorem builds on the following relationship between bi-models of P and answer sets of P^κ.

Corollary 6 *Let* P *be a program over* Σ. *If* $M \in \mathcal{AS}(P^\kappa)$, *then* $\beta(M \cap \Sigma^\kappa)$ *satisfies (i). If* (I, J) *is a bi-model of* P *that satisfies (i) and (ii), then there exists* $M \in \mathcal{AS}(P^\kappa)$, *such that* $\beta(M \cap \Sigma^\kappa) = (I, J)$.

For illustration consider the following example.

Example 15 Let $P = \{a \leftarrow b;\ b \leftarrow not\, b\}$. Its bi-models are all pairs (I, J), where $I \in \{\emptyset, \{a\}, \{a, b\}\}$ and $J \in \{\{b\}, \{a, b\}\}$. Condition (i) of Theorem 5 holds for bi-models such that $I = \emptyset$, and Condition (ii) holds only-if $J = \{b\}$. Thus, $\{Kb\}$ is the unique semi-stable model of P.

The examples given so far also exhibit some anomalies of the semi-stable semantics with respect to basic rationality properties considered in epistemic logics. In particular, *knowledge generalization* (or *necessitation*, resp. modal axiom **N**) is a basic principle in respective modal logics. For a semi-stable model I^κ, it would require that

Property N: $a \in I^\kappa$ implies $Ka \in I^\kappa$, for all $a \in \Sigma$.

This property does not hold as witnessed by Example 14.

Another basic requirement is the *distribution axiom* (modal axiom **K**). Assuming that we belief the rules of a given program (which might also be seen as the consequence of adopting knowledge generalization) the distribution property can be paraphrased for a rule of the form (1) as follows:

Property K: If $I^\kappa \models Kb_1 \wedge \ldots \wedge Kb_m$ and $I^\kappa \not\models Kc_1 \vee \ldots \vee Kc_n$, then $I^\kappa \models Ka_1 \vee \ldots \vee Ka_l$.

Note that this does not hold for rule $a \leftarrow b$ in Example 15.

4 Semi-Equilibrium Models

In this section we define and characterize an alternative paracoherent semantics which we call semi-equilibrium semantics (for reasons which will become clear immediately). The aim for semi-equilibrium models is to enforce Properties **N** and **K** on them. Let us start considering bi-models of a program P, that satisfy these properties. It turns out that such structures are exactly given by HT-models.

Proposition 7 *Let P be a program over Σ. Then,*

(1) if (I, J) is a bi-model of P, such that $(I, J)^\kappa$ satisfies Property **N** *and Property* **K***, for all $r \in P$, then (I, J) is an HT-model of P;*

(2) if (H, T) is an HT-model of P, then $(H, T)^\kappa$ satisfies Property **N** *and Property* **K***, for all $r \in P$.*

In order to define semi-equilibrium models, we follow the basic idea of the semi-stable semantics and select subset minimal models that are maximal canonical. Let us define $HT^\kappa(P) = \{(H, T)^\kappa \mid (H, T) \models P\}$ and denote by $MM(HT^\kappa(P))$ its minimal elements with respect to subset inclusion.

Definition 5 (semi-equilibrium models) *Let P be a program over Σ. An interpretation I^κ over Σ^κ is a semi-equilibrium ($\mathcal{SEQ}$) model of P, if $I^\kappa \in mc(MM(HT^\kappa(P)))$. The set of semi-equilibrium models of P is denoted by $\mathcal{SEQ}(P)$.*

A model-theoretic characterization for this semantics is obtained as before, replacing bi-models by HT-models and dropping Condition (ii). Intuitively, Condition (ii) is not needed as it is subsumed by Condition (iii) (i.e., Condition (ii') below) if Property **N** and Condition (i) hold.

To formulate the result, we extend the notion of gap from Σ^κ-interpretations to HT-interpretations as follows. For any HT-interpretation (X, Y), let $gap(X, Y) = Y \setminus X$, i.e., $gap(X, Y) = gap(\beta((X, Y)))$.

Theorem 8 *Let P be a program over Σ. Then,* *(1) If (H,T) is an HT-model of P such that (i′) $(H',T) \not\models P$, for all $H' \subset H$, and (ii′) no HT-model (H',T') of P exists that satisfies (i') and $gap(H',T') \subset gap(H,T)$, then $(H,T)^\kappa \in \mathcal{SEQ}(P)$;*

(2) if $I^\kappa \in \mathcal{SEQ}(P)$, then $\beta(I^\kappa)$ is an HT-model of P that satisfies (i') and (ii').

We refer to the condition (i') as *h-minimality* and to the condition (i'') as *gap-minimality* of an HT-model of a program P.

Like semi-stable models, semi-equilibrium models may be computed as maximal canonical answer sets, i.e., equilibrium models, of an extension of the epistemic program transformation.

Definition 6 (P^{HT}) *Let P be a program over Σ. Then its epistemic HT-transformation P^{HT} is defined as the union of P^κ with the set of rules:*

$$Ka \leftarrow a,$$
$$Ka_1 \vee \ldots \vee Ka_l \vee Kc_1 \vee \ldots \vee Kc_n \leftarrow Kb_1, \ldots, Kb_m,$$

for $a \in \Sigma$, respectively for every rule $r \in P$ of the form (1).

The extensions of the transformation naturally ensure Properties **N** and **K** on its models and its maximal canonical answer sets coincide with semi-equilibrium models.

Theorem 9 *Let P be a program over Σ, and let I^κ be an interpretation over Σ^κ. Then, $I^\kappa \in \mathcal{SEQ}(P)$ if and only if $I^\kappa \in \{M \cap \Sigma^\kappa \mid M \in mc(\mathcal{AS}(P^{HT}))\}$.*

We note at this point that an alternative, less involving encoding of semi-equilibrium models can be found in Section 8.

The resulting semantics is classically coherent, i.e., fulfills property (D3) from the Introduction.

Proposition 10 *Let P be a program over Σ. If P has a model, then it has a semi-equilibrium model.*

Another simple property is a 1-to-1 correspondence between answer sets and semi-equilibrium models.

Proposition 11 *Let P be a coherent program over Σ. Then,*

(1) if $Y \in \mathcal{AS}(P)$, then $(Y,Y)^\kappa$ is a semi-equilibrium model of P;

(2) if I^κ is a semi-equilibrium model of P, then $\beta(I^\kappa)$ is an equilibrium model of P, i.e., $\beta(I^\kappa)$ is of the form (Y,Y) and $Y \in \mathcal{AS}(P)$.

From Propositions 10 and 11, we thus obtain that semi-equilibrium models behave similarly as semi-stable models with respect to the properties (D1)-(D3) in the Introduction.

Proposition 12 *The $\mathcal{SEQ}$-models semantics, given by $\mathcal{SEQ}(P)$ for arbitrary programs P, satisfies properties (D1)-(D3).*

Furthermore, an immediate consequence of Proposition 11 is the following property.

Corollary 13 *For every positive program P, $\mathcal{SEQ}(P) = \mathcal{EQ}(P) = \{(M,M) \mid M \in MM(P)\}$.*

For an illustration of the 1-to-1 relationship between answer sets and semi-equilibrium models, let us reconsider Example 14. Note that this example also gave evidence that semi-stable models do not satisfy Property **N**, which is the case for semi-equilibrium models, however.

Example 16 Consider the coherent program of Example 14. Its unique semi-equilibrium model is $\{a, b, c, Ka, Kb, Kc\}$, which corresponds to the single answer set $\{a, b, c\}$; viewed as HT-models, we have $\beta(\{a, b, c, Ka, Kb, Kc\}) = (abc, abc)$, which is the equilibrium-model corresponding to the answer set $\{a, b, c\}$.

As a consequence of Property **K**, semi-equilibrium semantics differs from semi-stable semantics not only with respect to believed consequences.

Example 17 Consider the program $P = \{a \leftarrow b;\ b \leftarrow not\, b;\ c \leftarrow not\, a\}$, which extends the program in Example 15 with the rule $c \leftarrow not\, a$. The single semi-stable model of P is $\{c, Kb\}$ (which corresponds to the bi-model (c, b)), while the single $\mathcal{SEQ}$-model is $\{Ka, Kb\}$ (which corresponds to the HT-model $(\emptyset, ab)$). Thus while c is true under $\mathcal{SST}$-model semantics, it is false under $\mathcal{SEQ}$-model semantics: due to lacking belief propagation, the CWA assigns a false in the $\mathcal{SST}$-model which in turn causes c to get true; in the $\mathcal{SEQ}$-model, as a is believed to be true the rule with c in the head is defeated. As there is no other way to derive c, the CWA assigns it false.

As each $\mathcal{SEQ}$-model I^κ of P is uniquely determined by the HT-model $\beta(I^\kappa)$, we shall in the rest of this article also identify these models and refer to the set $\{\beta(I^\kappa) \mid I^\kappa \in \mathcal{SEQ}(P)\}$ as the $\mathcal{SEQ}$-models of P (and denote it in abuse of notation by $\mathcal{SEQ}(P)$).

5 Split Semi-Equilibrium Semantics

While the $\mathcal{SEQ}$-semantics has nice properties, it may select models that do not respect modular structure in the rules. To illustrate this, consider the following example.

Example 18 Suppose we have a program that captures knowledge about friends of a person regarding visits to a party, where $go(X)$ informally means that X will go:

$$P = \left\{ \begin{array}{c} go(John) \leftarrow not\; go(Mark); \\ go(Peter) \leftarrow go(John), not\; go(Bill); \\ go(Bill) \leftarrow go(Peter) \end{array} \right\}$$

Then P has no answer set; its semi-equilibrium models are $M_1 = (\emptyset, \{go(Mark)\})$, and $M_2 = (\{go(John)\}, \{go(John), go(Bill)\})$. Informally, a key difference between M_1 and M_2 concerns the beliefs on Mark and John. In M_2 Mark does not go, and, consequently, John will go (moreover, Bill is believed to go, and Peter will not go). In M_1, instead, we believe Mark will go, thus John will not go (likewise Peter and Bill).

None of the two models provides a fully coherent view (on the other hand, the program is incoherent, having no answer set). Nevertheless, M_2 appears preferable over M_1, since, according with a layering (stratification) principle, which is widely agreed in LP, one should prefer $go(John)$ rather than $go(Mark)$, as there is no way to derive $go(Mark)$ (which does not appear in the head of any rule of the program).

Modularity via rule dependency as in the example above is widely used in problem modeling and logic programs evaluation; in fact, program decomposition is crucial for efficient answer set computation. For the program P above, advanced answer set solvers like DLV and clasp immediately set $go(Mark)$ to false, as $go(Mark)$ does not occur in any rule head. In a customary bottom up computation along program components, answer sets are gradually extended until the whole program is covered, or incoherence is detected at some component (in our example for the last two rules). But rather than to abort the computation, we would like to switch to a paracoherent mode and continue with building semi-equilibrium models, as an approximation of answer sets.

To overcome this limitation, we introduce a refined paracoherent semantics, called *split semi-equilibrium semantics*. It coincides with the answer sets semantics in case of coherent programs, and selects a subset of the $\mathcal{SEQ}$-models otherwise. The main results of this section are two model-theoretic characterizations which identify necessary and sufficient conditions for deciding whether a $\mathcal{SEQ}$-model is selected.

5.1 Split Semi-Equilibrium Models

We now introduce the notion of *$\mathcal{SEQ}$-models relative to a splitting set*. First given a splitting set S for a program P and an HT-interpretation (I, J) for $b_S(P)$, we let

$$P^S(I,J) = P \setminus b_S(P) \cup \{a \mid a \in I\} \cup \{\leftarrow not\, a \mid a \in J\} \cup \{\leftarrow a \mid a \in S \setminus J\}. \tag{8}$$

Informally, the bottom part of P w.r.t. S is replaced with rules and constraints which fix in any $\mathcal{EQ}$-model of the remainder (= $t_S(P)$) the values of the atoms in S to (I, J).

Definition 7 (Semi-equilibrium models relative to a splitting set) *Let S be a splitting set of a program P. Then the* semi-equilibrium models of P relative to S *are defined as*

$$\mathcal{SEQ}^S(P) = mc\Big(\bigcup_{(I,J)\in\mathcal{SEQ}(b_S(P))} \mathcal{SEQ}(P^S(I,J))\Big). \tag{9}$$

Example 19 Reconsider the program in Example 18, $P = \{b \leftarrow not\, a;\ d \leftarrow b, not\, c;\ c \leftarrow d\}$, where a, b, and c, d stand for $go(Mark)$, $go(John)$, $go(Bill)$, and $go(Peter)$, respectively. We have $\mathcal{SEQ}(P) = \{\, (\emptyset, a), (b, bc)\,\}$, where (b, bc) is more appealing than $(\emptyset, a)$ because a is not derivable, as no rule has a in the head. Moreover, intuitively, $P_1 = \{b \leftarrow not\, a\}$ is a lower (coherent) part feeding into the upper part $P_2 = \{d \leftarrow b, not\, c;\ c \leftarrow d\}$. This is formally captured by the splitting set $S = \{a, b\}$, which yields $b_S(P) = P_1$ and $\mathcal{SEQ}(b_S(P)) = \{(b, b)\}$. Hence, $P^S(b, b) = \{d \leftarrow b, not\, c;\ c \leftarrow d;\ b;\ \leftarrow a\}$ and $\mathcal{SEQ}^S(P) = \mathcal{SEQ}(P^S(b, b)) = \{(b, bc)\}$.

In what follows, we establish a semantic characterization of the $\mathcal{SEQ}$-models relative to a splitting set as those $\mathcal{SEQ}$-models of the program that extend $\mathcal{SEQ}$-models of the bottom part.
Notation. For any HT-model (X, Y) and set S of atoms, we define the *restriction of* (X, Y) *to* S as $(X, Y)|_S = (X \cap S, Y \cap S)$.

Proposition 14 *Let S be a splitting set of a program P. If $(X, Y) \in \mathcal{SEQ}^S(P)$, then $(X, Y)|_S \in \mathcal{SEQ}(b_S(P))$.*

The following result shows that each semi-equilibrium model relative to a given splitting set is always a semi-equilibrium model of the program.

Proposition 15 (Soundness) *Let S be a splitting set of a program P. If $(X, Y) \in \mathcal{SEQ}^S(P)$, then $(X, Y) \in \mathcal{SEQ}(P)$.*

This result is proven by establishing first that HT-models of the program $P^S(I, J)$ are HT-models of the program P, and then the h-minimality and gap-minimality of (X, Y). More precisely, the first step uses the following lemma:

Lemma 16 *Let S be a splitting set of a program P and let $(I, J) \in \mathcal{SEQ}(b_S(P))$. If (X, Y) is an HT-model of $P^S(I, J)$, then (X, Y) is an HT-model of P.*

However, the converse of Proposition 15 does not hold in general; in fact if we consider the program of Example 19 and the splitting set $S = \{a, b\}$ we have $\mathcal{SEQ}^S(P) = \{(b, bc)\}$, while $\mathcal{SEQ}(P) = \{(\emptyset, a), (b, bc)\}$. Clearly, $\mathcal{SEQ}^S(P)$ depends on the choice of S; in fact if we choose $S = \emptyset$, then $\mathcal{SEQ}^\emptyset(P) = \mathcal{SEQ}(P)$.

Moreover for Proposition 15 to hold, the selection of maximal canonical HT-models is necessary.

Example 20 For $P = \{a \leftarrow \mathit{not}\, b;\ b \leftarrow \mathit{not}\, a;\ c \leftarrow b, \mathit{not}\, c\}$ and the splitting set $S = \{a, b\}$, we have $\mathcal{SEQ}(b_S(P)) = \{(a, a), (b, b)\}$; hence $\mathcal{SEQ}(P^S(a, a)) \cup \mathcal{SEQ}(P^S(b, b)) = \{(a, a), (b, bc)\}$, while $\mathcal{SEQ}(P) = \{(a, a)\}$.

So far, we have presented two properties of an HT-model that are necessary conditions to qualify as a $\mathcal{SEQ}$-model relative to a given splitting set. The natural question is whether these conditions are also sufficient; this is indeed the case.

Proposition 17 (Completeness) *Let S be a splitting set of a program P. If $(X, Y) \in \mathcal{SEQ}(P)$ and $(X, Y)|_S \in \mathcal{SEQ}(b_S(P))$, then $(X, Y) \in \mathcal{SEQ}^S(P)$.*

Putting the results above together, we obtain the following semantic characterization of $\mathcal{SEQ}$-models relative to a splitting set.

Theorem 18 (SEQ-model characterization) *Let S be a splitting set of a program P. Then $(X, Y) \in \mathcal{SEQ}^S(P)$ iff $(X, Y) \in \mathcal{SEQ}(P)$ and $(X, Y)|_S \in \mathcal{SEQ}(b_S(P))$.*

Proof. The only-if direction follows from Propositions 14 and 15; the if direction holds by Proposition 17. □

Like the ordinary $\mathcal{SEQ}$-models, also the split $\mathcal{SEQ}$-models coincide with the answer sets of a program if some answer set exists.

Corollary 19 *Let P be a program such that $\mathcal{EQ}(P) \neq \emptyset$. Then for every splitting set S of P, $\mathcal{SEQ}^S(P) = \mathcal{EQ}(P)$; in particular, if P is positive, then $\mathcal{SEQ}^S(P) = \{(M, M) \mid M \in MM(P)\}$.*

We observe that a program which has some model does not necessarily have split semi-equilibrium models (but always semi-equilibrium models).

Example 21 Let us consider $P = \{\leftarrow b;\ \ b \leftarrow \mathit{not}\, a\}$ and the splitting set $S = \{a\}$. Then we obtain $\mathcal{SEQ}(b_S(P)) = \{(\emptyset, \emptyset)\}$ and so $\mathcal{SEQ}^S(P) = \emptyset$. However (a, a) and $(\emptyset, a)$ are HT-models of P.

Note that occurrence of a constraint in the previous example is not accidental; in fact,

Proposition 20 *For every constraint-free program P and splitting set S of P, it holds that $\mathcal{SEQ}(P^S) \neq \emptyset$.*

In summary, the split $\mathcal{SEQ}$-models have the following profile with respect to the properties (D1)-(D3).

Proposition 21 *The split $\mathcal{SEQ}$-models semantics of a program P relative to a splitting set S of P, given by $\mathcal{SEQ}^S(P)$, satisfies properties (D1)-(D2), and if P is constraint-free, also (D3).*

5.2 Split Sequence Semi-Equilibrium Models

Now we generalize the use of splitting sets to $\mathcal{SEQ}$-models of a program via splitting sequences. To this end, we naturally reduce a splitting sequence to its head and its remainder and apply splitting sets recursively.

Definition 8 (Semi-equilibrium models relative to a splitting sequence) *Let $S = (S_1, \dots, S_n)$, $n \geq 1$, be a splitting sequence for a program P. then the* semi-equilibrium models of P relative to S *are given by*

$$\mathcal{SEQ}^S(P) = mc\Big(\bigcup_{(I,J)\in\mathcal{SEQ}(b_{S_1}(P))} \mathcal{SEQ}^{S'}(P^{S_1}(I,J))\Big), \tag{10}$$

where $S' = (S_2, ..., S_n)$ and $\mathcal{SEQ}^{()}(P) = \mathcal{SEQ}(P)$.

The $\mathcal{SEQ}$-models relative to a splitting sequence can be characterized similarly as those relative to a splitting set, namely as $\mathcal{SEQ}$-models of the program that remain by filtering the $\mathcal{SEQ}$-models along the splitting sequence.

To ease presentation, for a given program P and splitting sequence $S = (S_1, ..., S_n)$, we let $P_0 = P$ and $P_k = (P_{k-1})^{S_k}(I_k, J_k)$, where $(I_k, J_k) \in \mathcal{SEQ}(b_{S_k}(P_{k-1}))$, $k = 1, ..., n$; that is, P_k is not uniquely defined but ranges over a set of programs.

The main result of this section is now as follows.

Theorem 22 *Let $S = (S_1, ..., S_n)$ be a splitting sequence of a program P. Then $(X, Y) \in \mathcal{SEQ}^S(P)$ iff $(X, Y) \in \mathcal{SEQ}(P)$ and $(X, Y)|_{S_k} \in \mathcal{SEQ}(b_{S_k}(P_{k-1}))$, for some P_k, for $k = 1, ..., n$.*

The proof proceeds by induction using Theorem 18. Corollary 19 off Theorem 18 also generalizes to splitting sequences.

Corollary 23 *Let P be a program such that $\mathcal{EQ}(P) \neq \emptyset$. Then for every splitting sequence S of P, $\mathcal{SEQ}^S(P) = \mathcal{EQ}(P)$; in particular, if P is positive, then $\mathcal{SEQ}^S(P) = \{(M, M) \mid M \in MM(P)\}$.*

Proof. [Sketch] Using Theorem 22, this can be shown by induction, using Corollaries 13 and 19. □

Another consequence of Theorem 22 is that, written in other form, the split sequence $\mathcal{SEQ}$-models of a program can be bottom up constructed, taking into account that at each stage only the respective rules (i.e., $b_{S_{j+1}}(P) \setminus b_{S_j}(P)$) need to be considered. More formally,

Corollary 24 *For every splitting sequence $S = (S_1, \dots, S_n)$ of a program P, it holds that $\mathcal{SEQ}^S(P) = \mathcal{S}_n$, where for $j = n, \dots, 1$ we have*

$$\mathcal{S}_j = mc(\textstyle\bigcup_{(X,Y)\in\mathcal{S}_{j-1}} \mathcal{SEQ}(Q^j(X, Y))),$$

where $Q^j = b_{S_{j+1}}(P) \setminus b_{S_j}(P)$ with $b_{S_{n+1}}(P) = P$ and $\mathcal{S}_0 = \mathcal{SEQ}(b_{S_1}(P))$.

This form is in fact a suitable starting point for computation; we refer to Section 6.1 for further discussion.

Regarding the existence of split sequence $\mathcal{SEQ}$-models, we obtain a generalization of Proposition 20.

Proposition 25 *For every splitting sequence S of a constraint-free program P, it holds that $\mathcal{SEQ}(P^S) \neq \emptyset$.*

Proof. [Sketch] This can be shown by an inductive argument, along the lines of the proof of Proposition 20, using Propositions 10 and 20. □

In particular, we obtain from this for stratified programs the following result.

Corollary 26 *For every splitting sequence S of a stratified program P that is constraint-free, it holds that $\mathcal{SEQ}^S(P) = \mathcal{EQ}(P)$.*

In conclusion, we obtain the following profile of split sequence $\mathcal{SEQ}$-models with respect to the properties (D1)-(D3).

Proposition 27 *The split sequence $\mathcal{SEQ}$-models semantics of a program P relative to a splitting sequence S of P, given by $\mathcal{SEQ}^S(P)$, satisfies properties (D1)-(D2), and if P is constraint-free, also (D3).*

5.2.1 Infinite splitting sequences

As mentioned earlier, we concentrate in this article on finite splitting sequences; however split $\mathcal{SEQ}$-models can be easily extended to infinite splitting sequences $S = (S_1, S_2, \ldots, S_i, \ldots)$. To this end, we can define the split-$\mathcal{SEQ}$ models of P relative to a splitting sequence S by $\mathcal{SEQ}^S(P) = \bigcap_{i\geq 1}^{\infty} \mathcal{SEQ}^{S[1..i]}(P)$, where $S[1..i] = (S_1, \ldots, S_i)$ is the initial segment of S of length i. Indeed, any extension of the finite sequence $S[1..i]$ by some S_{i+1} may lead to the loss of $\mathcal{SEQ}$-models; on the other hand, after passing S_i, no new model candidates relative to S_i will be encountered.

6 Canonical Semi-Equilibrium Models

The split semi-equilibrium semantics depends on the choice of the particular splitting sequence, which is not much desirable. We thus consider a way to obtain a refined split $\mathcal{SEQ}$-semantics that is independent of a particular splitting sequence, but imposes conditions on sequences that come naturally with the program and can be easily tested.

Attractive for this purpose are the *strongly connected components* (SCCs) of a given program, which are at the heart of bottom up evaluation algorithms in ASP systems. In absence of constraints, we get the desired independence of a particular splitting sequence, such that we can then talk about the *$\mathcal{SCC}$-models of a program*. Allowing for constraints will need a slight extension.

6.1 $\mathcal{SCC}$-split Sequences and Models

We start with recalling further notions. The supergraph of a program P is the graph $SG(P) = \langle V_{SG}, E_{SG}\rangle$, where $V_{SG} = \mathcal{SCC}(P)$ and $E_{SG} = \{(C, C') \mid C \neq C' \in \mathcal{SCC}(P), \exists a \in C, \exists b \in C', (a, b) \in E_{DG}\}$. Note that $SG(P)$ is a directed acyclic graph (dag); recall that a *topological ordering* of a dag $G = \langle V, E\rangle$ is an ordering $v_1, v_2, ..., v_n$ of its vertices, denoted $\leq$, such that for every $(v_i, v_j) \in E$ we have $i > j$. Such an ordering always exists, and the set $\mathcal{O}(G)$ of all topological orderings of G is nonempty. Any such ordering of $SG(P)$ naturally induces a splitting sequence as follows.

Definition 9 *Let P be a program and let $\leq = (C_1, ..., C_n)$ be a topological ordering of $SG(P)$. Then the* splitting sequence induced by $\leq$ *is $S_\leq = (S_1, ..., S_n)$, where $S_1 = C_1$ and $S_j = S_{j-1} \cup C_j$, for $j = 2, ..., n$.*

We call any such $S_\leq$ a *$\mathcal{SCC}$-splitting sequence*; note that $S_\leq$ is indeed a splitting sequence of P.

We now show that for constraint-free programs, the split $\mathcal{SEQ}$-models relative to $\mathcal{SCC}$-split sequence are independent of the concrete such sequence; in fact, we establish this result for programs in which certain constraints do not occur.

Definition 10 *A constraint r in P is a* cross-constraint*, if r intersects distinct SCCs C_i, C_j in $\mathcal{SCC}(P)$ that are incomparable in $SG(P)$, i.e., $C_i \cap At(r) \neq \emptyset$, $C_j \cap At(r) \neq \emptyset$, and $SG(P)$ has topological orderings of the forms $(\ldots, C_i, \ldots, C_j, \ldots)$ and $(\ldots, C_j, \ldots, C_i, \ldots)$.*

For example, the constraint $\leftarrow b$ in the program P of Example 21 is trivially not a cross-constraint, and likewise an additional constraint $\leftarrow a, b$. However, an additional constraint $\leftarrow b, c$ would be a cross-constraint. We obtain the following result.

Theorem 28 *Let P be a program without cross-constraints. Then for every $\leq, \leq' \in \mathcal{O}(SG(P))$, we have $\mathcal{SEQ}^{S_\leq}(P) = \mathcal{SEQ}^{S_{\leq'}}(P)$.*

Corollary 29 *For every constraint-free program P, the $\mathcal{SEQ}$-models of P relative to an $\mathcal{SCC}$-split sequence S are independent of the choice of S.*

The proof of Theorem 28 is technically involving as it needs to be shown that changes in the ordering of the SCCs do not matter in the end. It uses a series of lemmas which assert certain properties of semi-equilibrium models (I_k, J_k) of the programs P_k that emerge in the bottom up characterization of Theorem 22, and independence properties in certain cases; in particular, where for any sets $\mathcal{M}$ and $\mathcal{M}'$ of HT-models, their product is given by $\mathcal{M} \times \mathcal{M}' = \{(X \cup X', Y \cup Y') \mid (X,Y) \in \mathcal{M}, (X',Y') \in \mathcal{M}'\}$:

Proposition 30 *Let P be a program in which each constraint r fulfills either $At(r) \subseteq S$ or $At(r) \subseteq At(P) \setminus S$. If $S \subseteq At(P)$ is such that both S and $At(P) \setminus S$ are splitting sets of P, then*

$$\mathcal{SEQ}(P) = \mathcal{SEQ}(b_S(P)) \times \mathcal{SEQ}(t_S(P)).$$

Theorem 28 is an analog of the Stratification Theorem [3, 35] which states that the perfect (stratified) model of a logic program relative to a stratification is independent of the concrete stratification, and thus one can simply refer to the perfect model of a stratified program; similarly, we thus can define the strongly connected components models of a program as follows.

Definition 11 ($\mathcal{SCC}$-models) *For every program P without cross-constraints, the $\mathcal{SCC}$*-models *of P are given as $M^{\mathcal{SCC}}(P) = \mathcal{SEQ}^{S_\leq}(P)$ for an arbitrary topological ordering $\leq$ of $SG(P)$.*

Example 22 Consider the program

$$P = \left\{ \begin{array}{l} \leftarrow a, d;\ a \leftarrow c, not\, a;\ \ a \leftarrow not\, b;\ \ b \leftarrow not\, e;\ \ b \leftarrow f; \\ c \leftarrow not\, d;\ \ c \leftarrow g, not\, h;\ \ f \leftarrow b, not\, f;\ \ g \leftarrow h;\ \ h \leftarrow c, g \end{array} \right\}.$$

Its SCCs are $C_1 = \{a\}$, $C_2 = \{b,f\}$, $C_3 = \{c,g,h\}$, $C_4 = \{d\}$ and $C_5 = \{e\}$; as a depends on d, the single constraint $\leftarrow a, d$ is not a cross-constraint. For the ordering $\leq = (C_4, C_5, C_3, C_2, C_1)$, we obtain that

$$\begin{aligned} \mathcal{SEQ}^{S_\leq}(P) &= \mathcal{SEQ}^{(S_2,S_3,S_4,S_5)}(P^{S_1}(\emptyset,\emptyset)) = \mathcal{SEQ}^{(S_3,S_4,S_5)}(P_1^{S_2}(\emptyset,\emptyset)) \\ &= \mathcal{SEQ}^{(S_4,S_5)}(P_2^{S_3}(c,c)) = \mathcal{SEQ}^{(S_5)}(P_3^{S_4}(bc,bcf)) = \{(bc,abcf)\}; \end{aligned}$$

hence $M^{\mathcal{SCC}}(P) = \{(bc,abcf)\}$. For $\leq' = (C_5, C_2, C_4, C_3, C_1)$, we obtain $\mathcal{SEQ}^{S_{\leq'}}(P) = \{(bc,abcf)\}$, in line with Theorem 28. Note that $\mathcal{SEQ}(P) = \{(bc,abcf), (b,bdf), (ac,ace)\}$.

Regarding the properties (D1)-(D3) of a paracoherent semantics in the Introduction, we obtain from Proposition 27 immediately

Corollary 31 *The $\mathcal{SCC}$-models semantics, given by $M^{\mathcal{SCC}}(P)$ for programs P without cross-constraints, satisfies properties (D1)-(D2), and it satisfies (D3) for programs without constraints.*

As for the properties of $\mathcal{SCC}$-models, we focus here on a particular aspect that is important with respect to an envisaged exploitation for paracoherent answer set construction; computational aspects are considered in Section 7.

6.1.1 Modularity of $\mathcal{SCC}$-models

In the definition of split SEQ-models, we made use of splitting sets as a major tool for modular computation of equilibrium models (answer sets) of a logic program. Indeed, for any splitting set S of P, as follows from [24] we have that

$$\mathcal{EQ}(P) = \bigcup_{(X,X)\in\mathcal{EQ}(b_S(P))} \mathcal{EQ}(t_S(P) \cup \{a \mid a \in X\} \cup \{\leftarrow a \mid a \in S \setminus X\}). \tag{11}$$

Note the similarity to the equation in (9) which we used to *define* seq-models of a program relative to a splitting set; the major difference is that we use the $mc(\cdot)$ operator to single out smallest gaps at a global level. And, in general for different S we shall obtain different $\mathcal{SEQ}$-models from (9). However, if we confine to $\mathcal{SCC}$-models, then an analog to (11) and its generalization to splitting sequences holds.

That is, if we replace in Equation (10) $\mathcal{SEQ}$, $\mathcal{SEQ}^S$, and $\mathcal{SEQ}^{S'}$ all by $M^{\mathcal{SCC}}$, then the resulting equation hold.

Theorem 32 *Let S be a splitting set of a program P without cross-constraints. Then*

$$M^{\mathcal{SCC}}(P) = mc\Big(\bigcup_{(I,J)\in M^{\mathcal{SCC}}(b_S(P))} M^{\mathcal{SCC}}(P^S(I,J))\Big). \tag{12}$$

Thanks to this result, we can compute the $\mathcal{SCC}$-models of a given program modularly bottom up along an arbitrary splitting sequence (using always $M^{\mathcal{SCC}}$); in particular, if an algorithm has processed a bottom part $b_S(P)$ of a program P and found equilibrium models (answer sets) for it, and it encounters that an extension of these equilibrium models using (11) does not yield any answer set, then it can switch to a "paracoherent mode" and apply (32); as $M^{\mathcal{SCC}}(b_S(P)) = \mathcal{EQ}(b_S(P))$, we obtain the same result as if we would compute the $\mathcal{SCC}$-models of P from scratch. That is, no backtracking or restarting of the computation is necessary.

6.2 $\mathcal{MJC}$-split Sequences and Models

Unfortunately, Theorem 28 fails if we allow arbitrary constraints in P. This is shown by the following simple example.

Example 23 The program $P = \{\, b;\ \leftarrow b, not\, a \,\}$ has the SCCs $\{a\}$ and $\{b\}$; hence $\mathcal{O}(SG(P)) = \{(\{a\}, \{b\}), (\{b\}, \{a\})\}$. However, the respective semi-equilibrium models are different: $\mathcal{SEQ}^{(\{a\},\{a,b\})}(P) = \emptyset$ and $\mathcal{SEQ}^{(\{b\},\{a,b\})}(P) = \{(b, ba)\}$. As the constraint $\leftarrow b, not\, a$ in P is a cross-constraint that intersects both SCCs, the order in which these incomparable components appear in a splitting sequence matters.

To deal with this situation, different ways are possible. The first one is to exclude constraints (or less restrictive, cross-constraints), and resort instead to the usage of rules which create unstable negation; that is

$$\leftarrow Body \tag{13}$$

is replaced with

$$f \leftarrow Body, not f, \tag{14}$$

where f is a fresh atom. Indeed, on some (early) implementations of answer set solvers constraints have been provided in this way. The $\mathcal{SEQ}$-model semantics is able to distinguish between (13) and (14); this can be exploited to use (14) as a soft constraint that may intuitively be violated if needed to achieve an $\mathcal{EQ}$-model resp. answer set; indeed, this rule can always be satisfied by considering f as believed true.

Another possibility is to remedy situations in which constraints are not embedded in SCCs. To this end, we consider merging of SCCs in such a way that independence of concrete topological orderings is preserved and, furthermore, merging is performed conservatively, that is only if it is deemed necessary. This is embodied by the *maximal joinable components* of a program, which lead to so called $\mathcal{MJC}$-split sequences and models. Informally, relevant SCCs that are unordered (thus unproblematic in evaluation) are merged if they intersect with a constraint.

We start with introducing the notions of *related pairs* and *joinable pairs* of SCCs. We call a pair (K_1, K_2) of SCCs of P a *related pair*, if either $K_1 = K_2$ or some constraint $r \in P$ intersects both K_1 and K_2, i.e., $At(r) \cap K_1 \neq \emptyset$ and $At(r) \cap K_2 \neq \emptyset$. By $C_{(K_1,K_2)}(P)$ we denote the set of all such constraints r.

Definition 12 *A related pair (K_1, K_2) is a* joinable pair*, if $K_1 = K_2$ or some ordering $(C_1, \ldots, C_n)$ in $\mathcal{O}(SG(P))$ exists such that (i) $K_1 = C_s$ and $K_2 = C_{s+1}$ for some $1 \leq s < n$, (ii) $(K_2, K_1) \notin E_{SG}$ and (iii) some $r \in C_{(K_1,K_2)}(P)$ exists such that $At(r) \subseteq C_1 \cup ... \cup C_{s+1}$. By $JP(P)$ we denote the set of all* joinable pairs *of P.*

Intuitively item (i) states that in some topological ordering K_1 immediately precedes K_2; item (ii) states that no atom in K_2 directly depends on an atom from K_1. If this does not hold, joining K_1 and K_2 to achieve independence is not necessary as their ordering is fixed. Finally item (iii) requires that some constraint must access the two SCCs (which thus must be a cross-constraint) and appear in the evaluation in the bottom of the program computed so far.

Example 24 For $P = \{\leftarrow b, not\, a;\ \leftarrow b, not\, c;\ d \leftarrow not\, a;\ c \leftarrow not\, e;\ b \leftarrow c\}$, we have $\mathcal{SCC}(P) = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}\}$. We observe that $(\{c\}, \{b\})$ is a related, but not a joinable pair, because $(\{c\}, \{b\})$ satisfies conditions (i) and (iii), but not (ii). On the other hand, $(\{a\}, \{b\})$ is a joinable pair.

We now extend joinability from pairs to any number of SCCs.

Definition 13 *Let P be a program. Then $K_1, ..., K_m \in \mathcal{SCC}(P)$ are* joinable*, if $m = 2$ and some $K \in \mathcal{SCC}(P)$ exists such that $(K_1, K), (K, K_2) \in JP(P)$, or otherwise K_i, K_j are joinable for each $i, j = 1, ..., m$. We let $JC(P) = \{\bigcup_{i=1}^{m} K_i \mid K_1, ..., K_m \in \mathcal{SCC}(P)$ are joinable$\}$ and call*

$$\mathcal{MJC}(P) = \{J \in JC(P) \mid \forall J' \in JC(P) : J \not\subset J'\}$$

the set of all maximal joined components *($\mathcal{MJC}$s) of P.*

Note that $(K_1, K_2) \in JP(P)$ implies that K_1 and K_2 are joinable (choose $K = K_1$).

Example 25 (cont'd) In Example 24, $(\{a\}, \{b\})$ is the only nontrivial joinable pair; hence $\mathcal{MJC}(P) = \{\{a, b\}, \{c\}, \{d\}, \{e\}\}$.

As easily seen, $\mathcal{MJC}(P)$ is a partitioning of $At(P)$ that results from merging SCCs. We define the *$\mathcal{MJC}$ graph* of P as $JG(P) = \langle V_{JG}, E_{JG}\rangle$, where $V_{JG} = \mathcal{MJC}(P)$ and $E_{JG} = \{(J, J') \mid J \neq J' \in \mathcal{MJC}(P), \exists a \in J, \exists b \in J', (a, b) \in E_{DG}\}$. Note that $JG(P)$ is like $SG(P)$ a directed acyclic graph, and hence admits a topological ordering; we denote by $\mathcal{O}(JG(P))$ the set of all such orderings. We thus define

Definition 14 *Let P be a program and $\leq\ = (J_1, ..., J_m)$ be a topological ordering of $JG(P)$. Then the* splitting sequence induced by $\leq$ *is $S_\leq = (S_1, ..., S_m)$, where $S_1 = J_1$ and $S_k = S_{k-1} \cup J_k$, for $k = 2, \ldots, m$.*

The sequence $S_\leq$ is again indeed a splitting sequence, which we call a *$\mathcal{MJC}$-splitting sequence*. We obtain a result analogous to Theorem 28, but in presence of constraints.

Theorem 33 *Let P be a program. For every $\leq, \leq' \in \mathcal{O}(JG(P))$, we have $\mathcal{SEQ}^{S_{\leq}}(P) = \mathcal{SEQ}^{S_{\leq'}}(P)$.*

The proof of this result is similar to the one of Theorem 28, but uses different lemmas.

Similarly as $\mathcal{SCC}$-models, we thus can define the $\mathcal{MJC}$-models of a program.

Definition 15 ($\mathcal{MJC}$-models) *For any program P, the* $\mathcal{MJC}$-models *of P are given as $M^{\mathcal{MJC}}(P) = \mathcal{SEQ}^{S_{\leq}}(P)$ for an arbitrary topological ordering $\leq$ of $JG(P)$.*

Example 26 (cont'd) Reconsider P in Example 24. Then for the ordering $\leq = (\{a\}, \{d\}, \{e\}, \{c\}, \{b\})$ we obtain $\mathcal{SEQ}^{S_{\leq}}(P) = \emptyset$, while for $\leq' = (\{e\}, \{c\}, \{b\}, \{a\}, \{d\})$ we obtain $\mathcal{SEQ}^{S_{\leq'}}(P) = \{(bc, abc)\}$. On the other hand, $JG(P)$ has the single topological ordering $\leq = (\{e\}, \{c\}, \{a, b\}, \{d\})$, and $\mathcal{SEQ}^{S_{\leq}}(P) = \{(bc, abc)\}$; hence $M^{\mathcal{MJC}}(P) = \{(bc, abc)\}$. Note that $\mathcal{SEQ}(P) = \{(bc, abc), (d, de)\}$.

The problem in Section 6.2 disappears when we use the MJCs. The program $P = \{\leftarrow b, not\,a;\ b\}$ there has the single MJC $J = \{a, b\}$, since the two SCCs $\{a\}$ and $\{b\}$ are related through the constraint $\leftarrow b, not\,a$ and thus joinable. As desired, we get (b, ab) as the (single) $\mathcal{MJC}$-model of P.

Note that trivially, the $\mathcal{MJC}$- and the $\mathcal{SCC}$-semantics coincide for constraint-free programs (in fact, also in absence of cross-constraints). As for the properties (D1)–(D3), again from Proposition 27 we obtain:

Corollary 34 *The $\mathcal{MJC}$-models semantics, given by $M^{\mathcal{MJC}}(P)$ for any program P, satisfies (D1)-(D2), and if P is constraint-free, also (D3).*

Program coherence (D3) is not ensured by $\mathcal{MJC}$-models, due to lean component merging that fully preserves dependencies. To obtain a $\mathcal{SEQ}$-model, blurring strict dependencies can be necessary, where two aspects need to taken into account.

(A1) Inconsistency may still emerge from cross-constraints.

Example 27 Consider the program $P = \{\leftarrow b, not\,a;\ b;\ b \leftarrow a\}$. It has the SCCs $\{a\}$ and $\{b\}$; as they are not joinable, $\mathcal{MJC}(P) = \{\{b\}, \{a\}\}$. The single $\mathcal{MJC}$-splitting sequence is $(\{a\}, \{a, b\})$, which however does not admit a split $\mathcal{SEQ}$-model; consequently, P has no $\mathcal{MJC}$ model.

This can be remedied by suitably merging components that intersect the same constraint.

(A2) A second, orthogonal aspect is dependence.

Example 28 The program $P = \{\ \leftarrow b;\ b \leftarrow not\,a\ \}$ has no $\mathcal{MJC}$-model, as the $\mathcal{MJC}$-splitting sequence $S = (\{a\}, \{a, b\})$ admits no split $\mathcal{SEQ}$-model; the culprit is a, which does not occur in the constraint.

Clearly, the problem extends to dependence via an (arbitrarily long) chain of rules; e.g. change in Example 28 the rule $b \leftarrow not\,a$ to $b \leftarrow c_1$, $c_1 \leftarrow c_2$, $\ldots$, $c_{n-1} \leftarrow c_n$, $c_n \leftarrow not\,a$. Again, this can be remedied by merging components. Many merging policies to ensure (D3) are conceivable; however, such a policy should ideally not dismiss structure unless needed, and it should be efficiently computable; we defer a discussion to Section 8, as the complexity results in the next section will provide useful insight for it.

6.2.1 Modularity of $\mathcal{MJC}$-models

A naive generalization of the modularity property of $\mathcal{SCC}$-models in Theorem 32 fails, as it does not hold for arbitrary splitting sets. To wit, for $P = \{b;\ \leftarrow b, not\,a\}$ and the splitting set $S = \{a\}$, the modular computation (similar as in the right hand side of (12)) yields no models, while $M^{\mathcal{MJC}}(P) = \{(b, ba)\}$. However, if we properly restrict S, then the generalization holds.

Theorem 35 *Let S be a splitting set of a program P such that $S = \bigcup \mathcal{M}$ for some $\mathcal{M} \subseteq \mathcal{MJC}(P)$. Then*

$$M^{\mathcal{MJC}}(P) = mc\Big(\bigcup_{(I,J) \in M^{\mathcal{MJC}}(b_S(P))} M^{\mathcal{MJC}}(P^S(I,J))\Big). \tag{15}$$

Thus, the same evaluation strategy as for $\mathcal{SCC}$-models can be applied.

7 Complexity and Computation

In this section, we turn to the computational complexity of the paracoherent model semantics that we have considered in the previous sections. In this, we deal with the $\mathcal{SEQ}$-model and the split $\mathcal{SEQ}$-model semantics in detail, while we treat the $\mathcal{SST}$-model semantics more in passing; the reason is that the complexity of $\mathcal{SST}$-model semantics has been elucidated in more detail in [15], while the $\mathcal{SEQ}$-model semantics has been only briefly considered there.

Regarding $\mathcal{SEQ}$-model semantics, we study the following major reasoning tasks:

(MCH) Given a program P and an HT-interpretation (X,Y), decide whether $(X,Y) \models_{\mathcal{SEQ}} P$.

(INF) Given a program P, an atom a and $v \in \{\mathbf{t}, \mathbf{f}, \mathbf{bt}\}$, decide whether a is a brave [resp. cautious] *$\mathcal{SEQ}$-consequence* of P with value v, denoted $P \models^{b,v}_{\mathcal{SEQ}} a$ [resp. $P \models^{c,v}_{\mathcal{SEQ}} a$], i.e., a has in some (every) $(X,Y) \in \mathcal{SEQ}(P)$ value v.

(COH) Given a program P, decide whether $\mathcal{SEQ}(P) \neq \emptyset$.

The generalizations of these problems to split $\mathcal{SEQ}$-semantics, where in addition a split sequence S is part of the input and $\mathcal{SEQ}$ is replaced with $\mathcal{SEQ}^S$, are denoted with **MCH**-S, **INF**-S, and **COH**-S, respectively. We consider all problems for several classes of programs, viz. normal, disjunctive, stratified, and headcycle-free programs[7] and the split $\mathcal{SEQ}$-models problems also for $\mathcal{SCC}$- and $\mathcal{MJC}$-splitting sequences S,

The attentive reader might ask why positive programs are not considered here; they are of less interest, as the (split sequence) $\mathcal{SEQ}$-models coincide with the minimal models of P (see Corollaries 13 and 23). Furthermore, we note that hcf-programs are under $\mathcal{SEQ}$-semantics sensitive to body shifts; e.g., $P = \{a \vee b;\ \ a \leftarrow \mathit{not}\, a;\ \ b \leftarrow \mathit{not}\, b\}$ has the $\mathcal{SEQ}$-models (a, ab) and (b, ab), while its shift $P_{\rightarrow} = \{a \leftarrow \mathit{not}\, b;\ \ b \leftarrow \mathit{not}\, a;\ \ a \leftarrow \mathit{not}\, a;\ \ b \leftarrow \mathit{not}\, b\}$ has the single $\mathcal{SEQ}$-model $(\emptyset, ab)$. Thus results for hcf-programs do not immediately carry over to normal program.

7.1 Overview of complexity results

Our complexity results are summarized in Tables 1 and 2. They show that $\mathcal{SEQ}$-model semantics is with respect to model checking (MCH) and inference (INF) one level higher up in the polynomial hierarchy than the $\mathcal{EQ}$-model (i.e., answer set) semantics; this is not surprising as the characterization of a $\mathcal{SEQ}$-model in Theorem 8 involves besides h-minimality also gap-minimality, while the $\mathcal{EQ}$-model definition involves only h-minimality. As gap-minimality is a global property and has to be checked across all h-minimal HT-models of a program, intuitively an (additional) quantifier is needed to express that no h-minimal HT-model with smaller gap exists; in particular, this causes $\mathcal{SEQ}$-model checking for normal programs to become intractable.

[7] Note that [15] did not consider stratified and hcf-programs.

The additional quantifier is then also needed for brave and cautious reasoning, where we need to find a suitable $\mathcal{SEQ}$-model that establishes respectively refutes the query atom, with one exception (this will be discussed below). For the coherence problem, however, the complexity is different compared to the $\mathcal{EQ}$-models semantics as it resorts to classical coherence, and thus to SAT; for some programs it is lower (e.g., for programs without constraints, where $\mathcal{EQ}$-model existence is NP-complete resp. Σ_2^p-complete, while COH is polynomial), while for others it is higher (e.g., for normal stratified programs with constraints COH is NP-complete, while $\mathcal{EQ}$-model existence is polynomial).

The results in Table 2 show that split $\mathcal{SEQ}$-models have the same complexity as $\mathcal{SEQ}$-models (i.e., structural information does not affect complexity) except on Problem COH, which is harder. Problems MCH and INF do not become harder, as MCH reduces to polynomially many MCH instances without splitting; the hardness results for arbitrary splitting sequences are inherited from respective results without splitting.

The reason for the complexity increase of COH is that coherence (D3) no longer holds for split $\mathcal{SEQ}$-model semantics. In particular, this means that imposing a structural condition on building $\mathcal{SEQ}$-models along SCCs may eliminate such models. The increase in complexity has a further important implication. Namely, that under usual complexity hypotheses, no polynomial-time method μ exists that associates with P a splitting sequence $S = \mu(P)$, using a polynomial-time checkable criterion on P, such that (i) μ respects structure and does not become trivial, i.e., $\mu(P) \neq (At(P))$ if $\mathcal{SEQ}^S(P) \neq \emptyset$ for some $S \neq (At(P))$, and (ii) μ preserves coherence, i.e., $\mathcal{SEQ}(P) \neq \emptyset$ implies $\mathcal{SEQ}^S(P) \neq \emptyset$. This negative result holds even if the method μ is allowed to be nondeterministic, i.e., can for example "guess" a suitable splitting sequence S for P. In other words, the price for ensuring coherence of a splitting sequence with tractable (or NP) effort is to merge sometimes more components than necessary.

For $\mathcal{SCC}$ and $\mathcal{MJC}$ splitting sequences, we obtain analogous results; informally, the problems do not get easier as splitting (which is a purely syntactic notion) can be blocked by irrelevant rules.

7.1.1 Semi-stable models

For semi-stable models, similar results hold as for $\mathcal{SEQ}$-models in Table 1. The reason is that model checking for semi-stable models amounts, by the characterization of Theorem 5, to a test that is similar to the one for $\mathcal{SEQ}$-models according to Theorem 8: testing $(I, J) \models_\beta P$ is like testing $(I, J) \models P$ feasible in polynomial time, and the conditions (i) and (ii) are analog to the conditions (i') and (ii'). Similar arguments as for $\mathcal{SEQ}$-models establish then the membership results for $\mathcal{SST}$-models. The matching hardness results are derived, however, using different reductions, which can be found in [15]. Noticeably, the proofs there establish hardness also under the restrictions to hcf, stratified normal, and disjunctive stratified programs; for hcf-programs, membership of model checking in coNP follows from the fact that deciding item (i) in Theorem 5 is feasible in polynomial time: as easily seen, this test amounts to deciding whether $I \in MM(P^J)$; as P^J is hcf and minimal model checking for hcf programs is polynomial [7], the tractability follows.

7.2 Derivation of the results

In the following, we formally state and derive the results in Tables 1 and 2. Rather than going into tiring technical details, we shall confine in the membership parts to the essential points and describe in the hardness parts the constructed programs without proving the correcntess in each case, which is routine.

We exploit that in most cases the split-variant Π-S of a problem Π features its full complexity already for the trivial split sequence $S = (At(P))$; thus Π-S and Π have the same complexity.

Theorem 36 *Given a program P, a splitting sequence S and an HT-interpretation (X, Y) recognizing if $(X, Y) \in \mathcal{SEQ}^S(P)$ is*

Table 1: Complexity of $\mathcal{SEQ}$-models (completeness results). The same results hold for $\mathcal{SST}$ models.

Problem / Program P:		normal, strat. normal, headcycle-free	disj. stratified, disjunctive
(MCH) Model checking:	$(X,Y) \in \mathcal{SEQ}(P)$?	coNP-c	Π_2^p-c
(INF) Brave reasoning:	$P \models^{b,v}_{\mathcal{SEQ}} a$?	Σ_2^p-c	Σ_3^p-c
Cautious reasoning:	$P \models^{c,v}_{\mathcal{SEQ}} a$?	Π_2^p-c	Π_3^p-c
(COH) Existence:	$\mathcal{SEQ}(P) \neq \emptyset$?	NP-c	NP-c

Table 2: Complexity of split $\mathcal{SEQ}$-models (completeness results). The same results hold for canonical models ($\mathcal{SCC}$-, $\mathcal{MJC}$-split sequences S).

Problem / Program P:		normal, strat. normal, headcycle-free	disj. stratified, disjunctive
(MCH-S) Model checking:	$(X,Y) \in \mathcal{SEQ}^S(P)$?	coNP-c	Π_2^p-c
(INF-S) Brave reasoning:	$P \models^{b,v}_{\mathcal{SEQ}^S} a$?	Σ_2^p-c	Σ_3^p-c
Cautious reasoning:	$P \models^{c,v}_{\mathcal{SEQ}^S} a$?	Π_2^p-c	Π_3^p-c
(COH-S) Existence:	$\mathcal{SEQ}^S(P) \neq \emptyset$?	Σ_2^p-c	Σ_3^p-c

(i) coNP-*complete for each of normal, stratified, and headcyle free P, and*

(ii) Π_2^p-*complete for disjunctive and stratified disjunctive P.*

In all cases, coNP- *resp.* Π_2^p-*hardness holds for $S = (\Sigma)$, i.e., $\mathcal{SEQ}$-model semantics.*

Proof. The membership parts for MCH can be derived as follows. Given an HT-interpretation (X, Y) of a program P, we can verify by Theorem 8 whether it is a $\mathcal{SEQ}$-model of P by checking that $(X, Y) \models P$, which obviously is feasible in polynomial time, and proving h-minimality (item (i')) and gap-minimality (item (ii')) of (X, Y); as for (i'), a guess for a HT-model (X', Y) of P such that $X' \subset X$ can be verified in polynomial time; thus h-minimality can be tested in coNP. Condition (ii') on top can be decided using an oracle for Π_2^p that no h-minimal model (X', Y') with $gap(X', Y') \subset gap(X, Y)$ exists; this establishes membership in Π_2^p. In case that P is hcf or normal, deciding h-minimality is polynomial, since (i') amounts to $X \in MM(P^Y)$; if P is hcf then also P^Y is hcf, and minimal model checking for such programs is polynomial [7]; if P is normal, then P^Y is Horn and minimal model checking is well-known to be polynomial.

As for split $\mathcal{SEQ}$-models, by Theorem 22 deciding whether (X, Y) is a $\mathcal{SEQ}$-model of P w.r.t. $S = (S_1, ..., S_n)$ reduces to checking whether (X, Y) and all $(X, Y)|_{S_k}$ are $\mathcal{SEQ}$-models of P resp. $b_{S_k}(P_{k-1})$, for $k = 1, \ldots, n$. Each program $b_{S_k}(P_{k-1})$ is normal (stratified normal, hcf, stratified disjunctive) if P has this property. Hence the already established membership results for $\mathcal{SEQ}$-models generalize to the case of splitting sequences.

The matching hardness results for item (ii) and $\mathcal{SEQ}$-models are proved in Appendix C.1; for stratified normal programs, which covers also normal and hfc-programs, we give a simple reduction from minimal model checking of positive programs P (which is well-known to be coNP-complete, cf. [14]). For any rule r, let $cs(r)$ be its constraint rewriting, i.e., $cs(r) = \leftarrow B^+(r), not\, B^-(r), not\, H(r)$, and let $cs(P) = \{cs(r) \mid r \in P\}$. Then $M \in MM(P)$ iff $(\emptyset, M) \in \mathcal{SEQ}(cs(P))$. All hardness results trivially extend to arbitrary splitting sequences, which establishes the result. $\square$

Theorem 37 *Given a program P, a splitting sequence S, an atom a and a value $v \in \{\mathbf{t}, \mathbf{f}, \mathbf{bt}\}$, deciding whether*

(i) $P \models^{b,v}_{\mathcal{SEQ}^S} a$ *is Σ^p_2-complete for each of normal, stratified normal, and hcf P and Σ^p_3-complete for disjunctive and stratified disjunctive P;*

(ii) $P \models^{c,v}_{\mathcal{SEQ}^S} a$ *is Π^p_2-complete for each of normal, normal stratified, and hcf P and Π^p_3-complete for disjunctive and stratified disjunctive P.*

In all cases, Σ^p_2/Π^p_2- resp. Σ^p_3/Π^p_3-hardness holds for $S = (\Sigma)$, i.e., $\mathcal{SEQ}$-model semantics.

Proof. Membership of brave (resp. cautious) reasoning from $\mathcal{SEQ}$-models w.r.t. S in Σ^p_3 (resp. Π^p_3) for disjunctive programs follows from Theorem 36, and similarly membership for normal, normal stratified and hcf-programs in Σ^p_2 [resp. Π^p_2]. The Σ^p_3/Π^p_3-hardness for brave [resp. cautious] reasoning from $\mathcal{SEQ}$-models from stratified disjunctive programs is proven in Appendix C.1 resp. C.2. The Σ^p_2/Π^p_2-hardness for stratified normal programs (and thus for normal and hcf-programs) follows by a reduction from brave (resp. cautious) reasoning from positive disjunctive programs P, which is Σ^p_2- resp. Π^p_2-hard (see Appendix C.1). For every such P and atom a, we have that $a \in M$ for some $M \in MM(P)$ iff $cs(P) \models^{b,\mathbf{bt}}_{S} a$ (resp. $P \models^{\mathbf{f}}_{c} a$ iff $cs(P) \models^{c,\mathbf{f}}_{\mathcal{SEQ}} a$); indeed, the $\mathcal{SEQ}$-models of P and $cs(P)$ are the HT-models (M, M) resp. $(\emptyset, M)$, where $M \in MM(P)$. $\square$

Notably brave reasoning has the same complexity in all cases, if we fix the truth value v arbitrarily, already for $S = At(P)$ (i.e., for $\mathcal{SEQ}$-models). For cautious reasoning, this similarly holds, except that for $v = \mathbf{bt}$ and $S = At(P)$, the complexity drops to coNP resp. Π^P_2 (see Appendix C.2).

Theorem 38 *Given a program P and a splitting sequence S, deciding whether $\mathcal{SEQ}^S(P) \neq \emptyset$ is*

(i) *Σ^p_2-complete for each of normal, stratified normal, and hcf P; and*

(ii) *Σ^p_3-complete for stratified disjunctive and disjunctive P; and*

(iii) NP*-complete for all program classes considered, if $S = (\Sigma)$ (i.e., for $\mathcal{SEQ}$ in place of $\mathcal{SEQ}^S$).*

Proof. The membership parts of (i) and (ii) follow easily from the results for MCH in Theorem 36, as a candidate $\mathcal{SEQ}$-model of P w.r.t. S can be guessed and checked with an NP resp. Σ^p_2 oracle in polynomial time. The hardness parts of (i) and (ii) can be obtained via a reduction from brave reasoning $P \models^v_b a$ in Problem INF. The Σ^p_3-hard (resp. Σ^p_2-hard) instances are of a form such that $P \models^v_b a$ iff some $\mathcal{SEQ}$-model (X, Y) of P exists with $a \in Y$. Let b be a fresh atom and define then $P' = P \cup \{\leftarrow b;\ b \leftarrow not\, a\}$. Then P' has a $\mathcal{SEQ}$-model w.r.t. $S = (At(P), At(P'))$ iff $P \models^v_b a$; this proves the Σ^p_3- (resp. Σ^p_2-) hardness.

The result in (iii) is an immediate consequence of the NP-completeness of SAT (satisfiability of a clause set) in propositional logic and the classical coherence property (D3) of $\mathcal{SEQ}$-model semantics. $\square$

Canonical split $\mathcal{SEQ}$-semantics For $\mathcal{SCC}$- and $\mathcal{MJC}$-splitting sequences, we have

Theorem 39 *The results on Problems MCH, INF and COH in Table 2 continue to hold if S is restricted to $\mathcal{SCC}$- (resp. $\mathcal{MJC}$-) splitting sequences.*

Proof. Indeed, the respective hardness proofs are extended to this setting. For a program P, let p be a fresh atom and let $P_{cl} = P \cup \{a \leftarrow a, p;\ p \leftarrow p, a \mid a \in \Sigma\}$. Clearly, P and P_{cl} have the same $\mathcal{SEQ}$-models, and P_{cl} has the single $\mathcal{SCC}$ $\Sigma' = \Sigma \cup \{p\}$. Exploiting this, the programs for MCH and INF have the single splitting sequence $S = (\Sigma')$ and those for Problem COH have $S' = (\Sigma', \Sigma' \cup \{b\})$; these are $\mathcal{SCC}$- and $\mathcal{MJC}$-splitting sequences. Furthermore, from S' we conclude that no method μ as in Subsection 7.1 exists (under usual complexity hypotheses). □

7.3 Constructing and recognizing canonical splitting sequences

It is well-known that $\mathcal{SCC}(P)$ and $SG(P)$ are efficiently computable from P (using Tarjan's [40] algorithm even in linear time); hence, it is not hard to see that one can recognize a $\mathcal{SCC}$-splitting sequence S in polynomial time, and that every such S can be (nondeterministically) generated in polynomial time (in fact, in linear time). We obtain similar tractability results for $\mathcal{MJC}(P)$ and $\mathcal{MJC}$-splitting sequences. To this end, we first note the following useful proposition.

Proposition 40 *Let P be a program and let $K_1, K_2 \in \mathcal{SCC}(P)$. Then K_1 and K_2 satisfy (i) and (ii) of Definition 12 iff they are disconnected in $SG(P)$, i.e., no path from K_1 to K_2 and vice versa exists.*

Based on this proposition, we can characterize the joinable pairs that are witnessed by a constraint from r as follows. As usual, let us call a SCC C_i in a set $\mathcal{C} \subseteq \mathcal{SCC}(P)$ of SCCs maximal, if no C_j in $\mathcal{C}$ exists that is comparable to C_i in $SG(P)$ and ordered after C_i, i.e., every topological ordering of $SG(P)$ is of the form $(\ldots, C_j, \ldots, C_i, \ldots)$.

Corollary 41 *Given a constraint $r \in P$, let $C_1, \ldots, C_l$ be the maximal SCCs C of P in $SG(P)$ such that $At(r) \cap C \neq \emptyset$. Then (K_1, K_2) where $K_1 \neq K_2$ is a joinable pair of P witnessed by r (i.e., satisfies (iii) for r) iff $K_1, K_2 \in \{C_1, \ldots, C_l\}$.*

By exploiting this characterization, we can construct $\mathcal{MJC}(P)$ and furthermore $JG(P)$ by the following steps:

1. compute $DG(P)$, $\mathcal{SCC}(P)$ and $SG(P)$;
2. for every constraint $r \in P$, determine all maximal $C_1^r, \ldots, C_l^r$ in $\mathcal{SCC}(P)$ such that $C_i^r \cap At(r) \neq \emptyset$;
3. let $C^r = C_1^r \cup \cdots \cup C_l^r$, and set $MC := \{C^r \mid r \in P, H(r) = \emptyset\}$ and $NMI := \mathcal{SCC}(P) \setminus \{C_1^r, \ldots, C_l^r \mid r \in P, H(r) = \emptyset\}$;
4. merge $J_1, J_2 \in MC$ such that $J_2 \cap J_2 \neq \emptyset$ (i.e., set $MC := (MC \setminus \{J_1, J_2\}) \cup \{J_1 \cup J_2\}$) until no longer possible;
5. set $\mathcal{MJC}(P) := MC \cup NMI$ and $JG(P) = (V_{JG}, E_{JG})$ where $V_{JG} = \mathcal{MJC}(P)$ and $E_{JG} = \{(J_1, J_2) \mid J_1 \neq J_2 \in \mathcal{MJC}(P), \exists a \in J_1, \exists b \in J_2, (a, b) \in E_{DG}\}$.

Example 29 Reconsider the program P from Example 24, which contains the constraints $r_1\colon \leftarrow b, \mathit{not}\, a$ and $r_2\colon \leftarrow b, \mathit{not}\, c$. We recall that $\mathcal{SCC}(P) = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}\}$. In Step 2 of the procedure, the maximal SCCs of r_1 are $\{a\}, \{b\}$ and the single maximal one of r_2 is $\{b\}$; thus in Step 3, we have $MC = \{\{a, b\}, \{b\}\}$ and $NMI = \{\{c\}, \{d\}, \{e\}\}$. In Step 4, $\{a, b\}$ and $\{b\}$ are merged, resulting in $MC = \{\{a, b\}\}$. Finally, in Step 5 $\mathcal{MJC}(P)$ is assigned $MC \cup NMI = \{\{a, b\}, \{c\}, \{d\}, \{e\}\}$; this is the correct result.

The following result states the correctness of the procedure and that it can be implemented to run in bilinear time.

Theorem 42 *Given a program P, $\mathcal{MJC}(P)$ and $JG(P)$ are computable in time $\mathcal{O}(cs{\cdot}\|P\|)$, where $cs = |\{r \in P \mid H(r) = \emptyset\}|$ is the number of constraints in P and $\|P\|$ is the size of P.*

In particular, the algorithm runs in linear time if the number of constraints is bounded by a constant. It remains as an interesting open issue whether the same time bound is feasible without this constraint.

8 Related Work

In this section, we first review some general principles for logic programs with negation, and we then consider the relationship of semi-stable and semi-equilibrium semantics to other semantics of logic programs with negation. Finally, we address some possible extensions of our work.

8.1 General principles

In the context of logic programs with negation, several principles have been identified which a semantics desirably should satisfy. Among them are:

- the *principle of minimal undefinedness* [45], which says that a smallest set of atoms should be undefined (i.e., neither true nor false);
- the *principle of justifiability (or foundedness)* [45]: every atom which is true must be derived from the rules of the program, possibly using negative literals as additional axioms.
- the *principle of the closed world assumption (CWA)*, for models of disjunctive logic programs (Eiter et al. [16]): "If every rule with an atom a in the head has a false body, or its head contains a true atom distinct from a w.r.t. an acceptable model, then a must be false in that model."

It can be shown that both the semi-stable and the semi-equilibrium semantics satisfy the first two principles (properly rephrased and viewing $\mathbf{bt}$ as undefined), but not the CWA principle; this is shown by the simple program $P = \{a \leftarrow \mathit{not}\, a\}$ and the acceptable model $\{Ka\}$. However, this is due to the particular feature of making, as in this example, assumptions about the truth of atoms; if the CWA condition is restricted to atoms a that are not believed by assumption, i.e., $I^\kappa(a) \neq \mathbf{bt}$ in a semi-stable resp. semi-equilibrium model I^κ, then the CWA property holds.

We eventually remark that Property **N** can be enforced on semi-stable models by adding constraints $\leftarrow a, \mathit{not}\, a$ for all atoms a to the (original) program. However, enforcing Property **K** on semi-stable models is more involved and efficient encodings seem to require an extended signature.

8.2 Related semantics

In this section, we relate the semi-stable and semi-equilibrium semantics to several semantics in the literature that allow for models even if a no answer set of a program exists.

8.2.1 Evidential Stable Models

Motivated by the fact that a disjunctive deductive database (DDDB) may lack stable models or even P-stable models, Seipel [39] presented a paracoherent semantics, called the *evidential stable model (ESM) semantics*, which assigns some model to every DDDB (that is, to every disjunctive logic program), such that the properties (D1)-(D3) in the Introduction are satisfied. Similar to [38], but guided by slightly different intuition, he defined the evidential stable models of a program P in a two-step process. First P is transformed into a positive disjunctive program $P^{\mathcal{E}}$, called the *evidential transform* of P, whose answer sets, i.e., its minimal models are considered. Among them are in the second step those selected that are informally preferred according to the amount of reasoning by contradiction that they involve.

Formally, for a given Σ let $\Sigma^{\mathcal{E}} = \Sigma \cup \{\mathcal{E}a \mid a \in \Sigma\}$, where $\mathcal{E}a$ intuitively means that there is evidence that a is true. Given a program P, its evidential transformation $P^{\mathcal{E}}$ consists of the following rules:

1. $H(r) \cup \mathcal{E}B^{-}(r) \leftarrow\ B^{+}(r)$ and

2. $\mathcal{E}H(r) \cup \mathcal{E}B^{-}(r) \leftarrow\ \mathcal{E}B^{+}(r)$, for each rule $r \in P$ of form (1), and

3. $\mathcal{E}a \leftarrow\ a$, for each $a \in \Sigma$.

where for every set $S \subseteq \Sigma$ of atoms, $\mathcal{E}S = \{\mathcal{E}a \mid a \in S\}$. Intuitively, the rules in (2) and (3) correspond to the rules that are added to Sakama and Inoue's program P^{κ} in the epistemic transformation to ensure the Properties **N** and **K** (see Definition 6); the rules in (2), however, are quite different from P^{κ}. They intuitively infer evidence for the truth of some atom b_j under negation ($m < j \leq n$) from the violation of the positive part of the rule (i.e., if all b_j, $1 \leq j \leq m$ are true and no a_i, $1 \leq i \leq l$ is true).

An interpretation I over $\Sigma^{\mathcal{E}}$ is an *evidential stable model*, if (1) I is a minimal model of $P^{\mathcal{E}}$, and (2) I has among all minimal models of $P^{\mathcal{E}}$ a $\subseteq$-minimal $\mathcal{E}$-violation set $\mathcal{V}_{\mathcal{E}}(I)$, which is defined as $\mathcal{V}(I) = \{a \in \Sigma \mid \mathcal{E}a \in I, a \notin I\}$.

Now the following can be shown. For every bi-interpretation (X, Y) let $(X, Y)^{\mathcal{E}} = X \cup \mathcal{E}Y$, and for every $I \subseteq \Sigma^{\mathcal{E}}$, let $\beta(I)$ the inverse of $\cdot^{\mathcal{E}}$, i.e., $\beta(I) = (X, Y)$ such that $(X, Y)^{\mathcal{E}} = I$.

Theorem 43 *Let P be a coherent program over Σ. Then for every bi-interpretation (X, Y) over Σ, it holds that $(X, Y) \in \mathcal{SEQ}(P)$ iff $(X, Y)^{\mathcal{E}}$ is an evidential stable model of P.*

Thus the $\mathcal{SEQ}$-model semantics coincides with the evidential stable model semantics for disjunctive logic programs. The theorem above gives a characterization of evidential stable models in terms of HT-logic, and in turn we obtain with $P^{\mathcal{E}}$ a simpler program to describe the $\mathcal{SEQ}$-models than the epistemic transformation P^{κ} in Section 4. Note, however, that the program is not a straightforward encoding of the semantic characterization of $\mathcal{SEQ}$-models in Theorem 8; $P^{\mathcal{E}}$ does not contain all h-minimal HT-models of P, but sufficiently many to single out all the $\mathcal{SEQ}$-models by gap minimization.

8.2.2 CR-Prolog

In order to deal with inconsistency in answer set programs, Balduccini and Gelfond introduced CR-Prolog [4] as a declarative approach for inconsistency removal from program. Roughly speaking, each program P is equipped with a further set of rules CR of the form

$$r: \quad h_1 \text{ or } \ldots \text{ or } h_k \stackrel{+}{\leftarrow} l_1, \ldots, l_m, not\, l_{m+1}, \ldots, not\, l_n.$$

where intuitive reading is: if $l_1, \ldots, l_m$ are accepted beliefs while $l_{m+1}, \ldots, l_n$ are not, then one of $h_1, \ldots, h_k$ "may possibly" be believed. In addition, a preference relation on the rules may be provided.

Rules from this pool CR can be added to restore consistency of the program P if no answer set exists, applying Occam's razor. Informally, a subset-minimal set $R \subseteq CR$ of rules is chosen such that $P' = P \cup R'$ is coherent, where R' is R cast to the ASP syntax; the answer sets of P' are then accepted as CR-answer sets of P. Formally, P and CR are compiled into a single abductive logic program where an abducible atom $appl(r)$ is used for the each rule r from CR to control (and be aware of) its applicability; a minimal set of abducibles may be assumed to be true without further justification. For simplicity, however, we use the abstract description from above.

The CR-Prolog approach is different from semi-stable and $\mathcal{SEQ}$-model semantics in several respects. First, it provides a (syntactic) inconsistency management strategy, not a semantic treatment of incoherence at the semantic level of interpretations. Second, it remains with the user to ensure coverage of all cases of incoherence; this bears risk that some cases are overlooked. On the other hand, depending on the application it might be preferred that this is pointed out.

Even if CR consists of all atoms in P, CR-answer sets and $\mathcal{SEQ}$-models may disagree, as adding facts, as done in this case by CR-Prolog, is stronger than blocking negated atoms as in semi-stable and $\mathcal{SEQ}$-models semantics (which then admits more answer sets).

Example 30 Consider the program $P = \{a \leftarrow not\, a;\ c \leftarrow not\, b;\ b \vee c \leftarrow a\}$. This program has the unique $\mathcal{SEQ}$-model $(\{c\}, \{a, c\})$; i.e., c is true, b is false, and a is believed true.

Let $CR = \{r_a :\ a \stackrel{+}{\leftarrow};\ r_b :\ b \stackrel{+}{\leftarrow};\ r_c :\ c \stackrel{+}{\leftarrow}\}$ and assume that there are no preferences. Then $R' = \{r_a\}$ is the single minimal subset of CR such that $P' = P \cup R'$ is coherent, and $P' = \{a \leftarrow not\, a;\ c \leftarrow not\, b;\ b \vee c \leftarrow a; a \leftarrow\}$ has two answer sets, viz. $\{a, c\}$ and $\{a, b\}$, which are then both CR answer sets.

The program in the previous example shows that adding a as a fact is stronger than blocking the use of a under negation. We remark that this similarly applies to the generalised stable model semantics [23], in which abducible facts may be added to the program P in order to obtain a stable model.

8.2.3 Well-founded Semantics

The most prominent approximation of the stable semantics is the well-founded semantics (WFS) [42]. It assigns each normal logic program P, in our terminology, an HT-model $WF(P) = (I, J)$ (called the *well-founded model*) such that all atoms in I are regarded as being true and all atoms not in J being false; all the remaining atoms (i.e., those in $gap(WF(P))$ are regarded as undefined (rather than possibly true, as in HT logic). Intuitively, the false atoms are those which can never become true, regardless of how the undefined atoms will be assigned. Extending WFS to disjunctive program is non-trivial and many proposals have been made, but there is no general consensus on the "right" such extension (see [43, 11] for more recent proposals); we comment on the proposal of Cabalar et al. [11] in the subsection on partial stable models below.

The well-founded semantics has many different characterizations; among them is the well-known alternating fixpoint-characterization, cf. [19, 5]: for normal constraint-free programs P, the operator $\gamma_P(X) = LM(P^X)$, $X \subseteq \Sigma$ is anti-monotonic, where $LM(Q)$ denotes the unique minimal model of Q (which for $Q = P^X$ exists). We then have $WF(P) = (I, J)$ where I is the least fixpoint of the monotonic operator $\gamma_P^2(X) = \gamma_P(\gamma_P(X))$, and $J = \gamma_P(I)$. Furthermore, the well-founded model is the least partial stable model (see Section 8.2.4 below); it has been characterized in the logic HT^2 in terms of the partial equilibrium model that leaves the most atoms undefined [12].

With regard to Section 8.1, WFS does not satisfy minimal undefinedness, but justifiability and naturally the CWA principle. It does not satisfy answer set coverage (D1) nor congruence (D2) (even if a single answer set exists), but coherence (D3). Roughly speaking, the well-founded model remains agnostic about atoms that are involved in cycles through negation whose truth value can not be determined from other parts of the program, and it propagates undefinedness. This may effect that all atoms remain undefined; e.g., the program in Example 17 has this property.

It is well-known that the well-founded model $WF(P) = (I, J)$ approximates the answer sets of P in the sense that $I \subseteq M \subseteq J$ for each answer set M of P; it is thus geared towards approximating cautious inference of literals from all answer sets of P, rather than towards approximating individual answer sets. If no answer set exists, WFS avoids trivialization and still yields a model; however, the notion of undefinedness and the associated propagation may lead to less informative results, as shown in Example 3.

$\mathcal{SEQ}$-refinement of the WFS A closer look at the WFS reveals that the $\mathcal{SEQ}$-model semantics refines it in the following sense.

Notation. Let for HT-interpretations $M = (X, Y)$ and $M' = (X', Y')$ denote $M \sqsubseteq M'$ that $X' \subseteq X$ and $Y \subseteq Y'$, i.e., M is a refinement of M' that results by assigning atoms in $gap(M')$ either true of false.[8]

Recall that an HT-interpretation (X, Y) of a program P is h-minimal, if no HT-model (X', Y) exists such that $X' \subset X$; for normal P, this means that X is the least model of P^Y.

Proposition 44 *Let $M = (X, Y)$ be an h-minimal model of a (constraint-free) normal program P. If $gap(M) \subseteq gap(WF(P))$, then $M \sqsubseteq WF(P)$, i.e., M is a refinement of the well-founded model of P.*

Note that this proposition is not immediate as we just compare gaps, not models themselves. The result follows from some well-known properties of $WF(P)$ and its relationship to the answer set semantics.

First, as already mentioned above, WFS is an approximation of the stable semantics:

Lemma 45 *For every equilibrium (stable) model $M = (Y, Y)$ of P, it holds that $M \sqsubseteq WF(P)$.*

Furthermore, WF is such that by making yet unassigned atoms true, the values of the already assigned atoms are not affected. That is,

Lemma 46 *For every set $G \subseteq gap(WF(P))$, it holds that $WF(P \cup G) \sqsubseteq WF(P)$.*

Intuitively, this is because for each atom a outside $gap(WF(P))$, a rule already fires resp. all rules are definitely not applicable. Next, h-minimality allows for unsupported atoms (the gap). By making them facts, we get an answer set:

Lemma 47 *If $M = (X, Y)$ is a h-minimal model of P, then $M = (Y, Y)$ is an answer set of $P' = P \cup gap(M)$.*

Indeed, X is the least model of P^Y, so each atom in X can be derived from P^Y; by adding $gap(M) = Y - X$, all atoms of Y can be derived from $P^Y \cup gap(M) = P'^Y$, and clearly no proper subset can be derived.

Armed with these lemmas, we now prove the proposition.

Proof. [of Proposition 44] Let $M = (X, Y)$ be a h-minimal model of P such that $gap(M) \subseteq gap(WF(P))$, and let $WF(P) = (I, J)$. By Lemma 47, $N = (Y, Y)$ is an answer set of $P' = P \cup gap(M)$. By Lemma 45, $N \sqsubseteq WF(P')$, and by Lemma 46, $WF(P') \sqsubseteq WF(P)$. As refinement is transitive, we obtain $N \sqsubseteq WF(P)$; it follows that $Y \subseteq J$.

Regarding X, by the alternating fixpoint characterization of $WF(P)$ we have $I = LM(P^J)$, and thus $WF(P)$ is a h-minimal model of P; as M is a h-minimal model of P, we have $X = LM(P^Y)$. As $\gamma_P(I) = LM(P^I)$ is anti-monotonic and $Y \subseteq J$, it follows that $X \supseteq I$.

Thus, we get $M = (X, Y) \sqsubseteq (I, J) = WF(P)$. This proves the proposition. □

[8] That is, $M \sqsubseteq M'$ iff $M' \leq_p M$, where $\leq_p$ is the precision ordering.

From this proposition, we obtain a refinement result for arbitrary normal programs, i.e., programs that may contain constraints. For such a program P, we define its well-founded model as $WF(P) = WF(P')$, where P' is the constraint-free part of P, if $WF(P') \models P \setminus P'$; otherwise, $WF(P)$ does not exist. Note that each constraint r in P must have a false body in $WF(P)$, i.e., either some $b_i \in B^+(r)$ is false in $WF(P)$ or some $c_j \in B^-(r)$ is true in $WF(P)$ (this can be seen from the alternating fixpoint characterization).

Corollary 48 (of Proposition 44) *Every normal program P such that $WF(P)$ exists has a $\mathcal{SEQ}$-model M such that $M \sqsubseteq WF(P)$. In fact, every $\mathcal{SEQ}$-model M of P such that $gap(M) \subseteq gap(WF(P))$ satisfies $M \sqsubseteq WF(P)$.*

Proof. Indeed, $\mathcal{SEQ}$-models are special h-minimal models (global gap-minimization), so the result follows from Proposition 44 and the fact that $WF(P) = WF(P') = (I, J)$ is h-minimal (as $I = LM(P^J) = LM(P'^J)$), where P' is the constraint-free part of P. □

Note, however, that not every $\mathcal{SEQ}$-model refines the well-founded model. E.g., take $P = \{a \leftarrow not\, a, not\, b\}$. Then $WF(P) = (\emptyset, \{a\})$ but the $\mathcal{SEQ}$-models are $M_1 = (\emptyset, \{a\})$ and $M_2 = (\emptyset, \{b\})$, and M_2 has a gap outside the gap of $WF(P)$.

If desired, one can easily restrict the $\mathcal{SEQ}$-models of a program P to those which refine its well-founded model $WF(P) = (I, J)$, by replacing P with

$$P^{wf} = P \cup I \cup \{\leftarrow A \mid A \in \Sigma \setminus J\}.$$

Note that $WF(P^{wf})$ exists whenever $WF(P)$ exists. We then obtain the following result.

Proposition 49 *For every normal program P such that $WF(P)$ exists, $\mathcal{SEQ}(P^{wf}) = \{M \in \mathcal{SEQ}(P) \mid gap(M) \subseteq gap(WF(P))\}$.*

By combining Corollary 48 and Proposition 49, we get a paracoherent way to refine the well-founded semantics for query answering, which coincides with the answer set semantics for coherent programs and provides in general more informative results and reasoning by cases (see Examples 3 and 4).

8.2.4 Partial Stable Model Semantics

P-stable (partial stable) models, which coincide with the 3-valued stable models of [36], are one of the best known approximation of answer sets. Like the well-founded model, P-stable models can be characterized in several ways (cf. [16]); with respect to equilibrium logic, Cabalar et al. [12] semantically characterized P-stable models in the logic HT^2 in terms of partial equilibrium models. For the concerns of our discussion, we use here a characterization of P-stable models (X, Y) in terms of the multi-valued operator $\hat{\gamma}_P(X) = MM(P^X)$ as the HT-interpretations (X, Y) such that $Y \in \hat{\gamma}_P(X)$ and $X \in \hat{\gamma}_P(Y)$; this characterization is easily obtained from [16]. In particular, for normal programs $WF(P)$ is a P-stable model of P (and in fact the least refined such model w.r.t. $\sqsubseteq$), and every answer set M of P (as $M = LM(P^M)$) amounts to a P-stable model (M, M) of P; in this vein, according to Cabalar et al. [11, 12] the well-founded models of a disjunctive program P are the least refined P-stable models M of P (i.e., no P-stable model $M' \neq M$ of P exists such that $M \sqsubseteq M'$); however, no well-founded model might exist.

The P-stable models, while more fine-grained than the well-founded model, behave similarly with regard to the properties in Subsection 8.1 and the properties (D1)–(D3) in the Introduction. Among the refinements of P-stable models in [16], the one that is closest in spirit to semi-stable and $\mathcal{SEQ}$-models are the L-stable models, which are the P-stable models that leave a minimal set of atoms undefined.

In fact, L-stable models satisfy all properties in Subsection 8.1 and (D1)–(D3), with the exception that coherence (D3) fails for disjunctive programs, as such programs may lack a P-stable model, and thus also an L-stable model.

Example 31 The program

$$P = \{a \leftarrow not\, b;\ b \leftarrow not\, c;\ c \leftarrow not\, a;\ a \vee b \vee c\} \tag{16}$$

has no P-stable models, while it has multiple $\mathcal{SEQ}$-models, viz. (a, ac), (b, ab), and (c, bc), which coincide with the $\mathcal{SST}$-models. Intuitively, one of the atoms in the disjunctive fact $a \vee b \vee c \leftarrow$, say a, must be true; then c must be false and in turn b must be true. The resulting (total) interpretation $(\{a, b\}, \{a, b\})$, however, does not fulfill that $\{a, b\}$ is a minimal model of $P^{\{a,b\}} = \{b \leftarrow; a \vee b \vee c \leftarrow\}$. With a symmetric argument for b and c, we conclude that no P-stable model of P exists. However, by adopting in addition that c is believed true, we arrive at the $\mathcal{SEQ}$-model (a, ac).

The main difference between that L-stable semantics and semi-stable resp. semi-equilibrium semantics is that the former takes —like P-stable semantics—a neutral position on undefinedness, which in combination with negation may lead to weaker conclusions.

For example, the program P in Example 3 has a single P-stable model, and thus P has a single L-stable model which coincides with its well-founded model; thus we can not conclude under L-stable semantics from P that $visits\_barber(joe)$ is false.

Also the program in Example 17 has a single P-stable (and L-stable) model in which all atoms are undefined, while c is true under $\mathcal{SEQ}$-model semantics. Similarly, the program

$$P = \{a \leftarrow not\, b;\ b \leftarrow not\, c;\ c \leftarrow not\, a\} \tag{17}$$

has a single P-stable (and thus L-stable) model in which all atoms are undefined; if we add the rules $d \leftarrow a$, $d \leftarrow b$, and $d \leftarrow c$ to P, the new program cautiously entails under both semi-stable and $\mathcal{SEQ}$-model semantics that d is true, but not under L-stable semantics.

Possible $\mathcal{SEQ}$-refinement of the L-stable semantics As the $\mathcal{SEQ}$-semantics refines the WFS as shown in Section 8.2.3, the natural question is whether a similar refinement property holds for L-stable models. Unfortunately this is not the case, even for normal programs without constraints (which always possess L-stable models); this is witnessed e.g. by the following example.

Example 32 Consider the program

$$P = \left\{ \begin{array}{l} a \leftarrow not\, b, d;\ b \leftarrow not\, a, d;\ c \leftarrow not\, c \\ d \leftarrow not\, c;\ d \leftarrow not\, a, not\, e;\ d \leftarrow not\, b, not\, e \end{array} \right\} \cup \{e \leftarrow not\, a, not\, b.\}.$$

Intuitively, the rules with heads a and b make a guess a or b, if d is true; c must be undefined as there is no other way to derive c than from its negation; d is true if one of a and b is false but not both, i.e., we have a guess for a and b. Thus proper guessing on a and b makes the gap smallest.

Under WFS, all atoms must be undefined as each atom occurs in of P only on cycles with negation. Furthermore, $N_1 = (ad, acd)$ and $N_2 = (bd, bcd)$ are L-stable models, because they are partially stable and no smaller gap than $gap(N_1) = gap(N_2) = \{c\}$ is possible. There is no further L-stable model (d would need to be true in it, which means that e must be false and hence either a false or b false; thus we end up with N_1 or N_2), and actually also no other P-stable model.

As one can check, $M = (e, ec)$ is a h-minimal model of P, and $gap(M) = \{c\}$. Thus M is an "additional" h-minimal model of P, and M does neither refine N_1 nor N_2.

If we slightly extend P in (17) to

$$P' = P \cup \{c' \leftarrow not\, c, not\, c'\}, \tag{18}$$

then we get a similar situation. Again, as c only occurs in the head of the rule $c \leftarrow not\, c$, it must be undefined in each partial stable model, and hence the same follows also for c'. Thus we obtain that $N_1' = (ad, acc'd)$ and $N_2' = (bd, bcc'd)$ are the L-stable models of P', and they have $gap(N_1') = gap(N_2') = \{c, c'\}$. On the other hand, M is also an h-minimal model of P', and $gap(M) = \{c\}$; thus M is the unique $\mathcal{SEQ}$-model of P', and the models are unrelated.

Possible $\mathcal{SEQ}$-refinement of disjunctive P-stable models The previous example showed that $\mathcal{SEQ}$-models with smaller gaps than L-stable models do not necessarily refine them. However, as they refine always some P-stable model (the WFM) of a normal program, it does not rule out that the refine some P-stable model of a disjunctive program P, and in particular a well-founded model (i.e., a least refined (w.r.t. $\sqsubseteq$) P-stable model). It appears that this refinement property does not hold.

Example 33 Consider the following variant of the program on line (16) in Example 31:

$$P = \{a \leftarrow not\, b;\; b \leftarrow not\, c;\; c \leftarrow not\, a;\; a \vee b \vee c \leftarrow d;\; d \vee e;\; d \leftarrow e, not\, d\,\}.$$

By the disjunctive fact $d \vee e$, either d or e must be true in each h-minimal model (and thus in each P-stable resp. $\mathcal{SEQ}$-model of P). If d is true, then the clauses containing $a, b, c,$ do not admit a P-stable model; if e is true, the single P-stable model is $M = (e, abcde)$. On the other hand, the $\mathcal{SEQ}$-models of P are $M_1 = (ad, acd)$, $M_2 = (bd, abd)$, and $M_3 = (cd, bcd)$; note that each h-minimal model of P in which e is true must have d and some atom from a, b, c believed true but not true, and thus can not be gap-minimal. As each M_i has smaller gap than M but does not refine it, the refinement property does not hold.

Note that the example shows even more: different from normal programs, for disjunctive programs the $\mathcal{SEQ}$-models do not refine the intersection of all P-stable models (i.e., the HT-interpretation (X, Y) where X resp. $\Sigma \setminus Y$ is what is true resp. false in every P-stable model of P). Thus in conclusion, for disjunctive programs, P-stable and $\mathcal{SEQ}$-models are in general unrelated.

8.2.5 Further Semantics

The regular model semantics [45] is another 3-valued approximation of answer set semantics that satisfies least undefinedness and foundedness, but not the CWA principle. However, it is classically coherent (satisfies (D3)). For the odd loop program P in (17) the regular models coincide with the L-stable models; the program P' in (18) has the regular models $\{a\}$, $\{b\}$, and $\{c\}$. While regular models fulfill answer set coverage, they do not fulfill congruence. For more discussion of 3-valued stable and regular models as well as many other semantics coinciding with them, see [16].

Revised stable models [32] are a 2-valued approximation of answer sets; negated literals are assumed to be maximally true, where assumptions are revised if they would lead to self-incoherence through odd loops or infinite proof chains. For example, the odd-loop program P in (17) has three revised stable models, viz. $\{a, b\}$, $\{a, c\}$, and $\{b, c\}$. The semantics is only defined for normal logic programs, and fulfills answer set coverage (D1) but not congruence (D2), cf. [32]. Similarly, the so called pstable models in [28], which should not be confused with P-stable models, have a definition for disjunctive programs however, satisfy answer set coverage (D1) (but just for normal programs) and congruence (D2) fails. Moreover, every pstable model of a program is a minimal model of the program, but there are programs, e.g. P in (17) again, that have models but no pstable model, thus classical coherence does not hold.

8.3 Modularity

To our knowledge, modularity aspects of paracoherent semantics have not been studied extensively. A noticeable exception is [16], which studied the applicability of splitting sets for several partial models semantics, among them the P-stable and the L-stable semantics that were already considered above. While for P-stable models a splitting property similar to the one of answer sets holds, this is not the case for L-stable models, due to global gap-minimization however, an analogue to Theorem 32, with L-stable models in place of $\mathcal{SCC}$-models is expected to hold.

Huang et al. [21] showed that hybrid knowledge bases, which generalize logic programs, have modular paraconsistent semantics for stratified knowledge bases; however, the semantics aims at dealing with classical contradictions and not with incoherence in terms of instability through cyclic negation.

Pereira and Pinto [34], using a layering notion that is similar to SCC-split sequences, introduce *layered models* (LM) semantics which is an alternative semantics that extends the stable models semantics for normal logic programs. The layered models of a program P are a superset of its answer sets, and this inclusion can be strict even if P is coherent; thus, property (D2) does not hold. In a sense, the CWA is relaxed more than necessary in the model construction process.

Faber et al. [17] introduced a notion of modularity for answer set semantics, based on syntactic relevance, which has paracoherent features. However, this notion was geared towards query answering rather than model building, and did not incorporate gap minimization at a semantic level.

Finally, we look at models related to a splitting sequence. Not every $\mathcal{SEQ}$-model of P that is a refinement of $WF(P)$ is a $\mathcal{SCC}$-model of P; we might "lose" $\mathcal{SEQ}$-models by splitting. E.g.,

$$P = \{\, a \leftarrow not\, a;\ b \leftarrow not\, b, not\, a;\ c \leftarrow not\, b, not\, c\,\}$$

has the SCCs $C_1 = \{a\}$, $C_2 = \{b\}$ and $C_3 = \{c\}$, and $WF(P) = (\emptyset, abc)$; the single $\mathcal{SCC}$-model of P is $M = (\emptyset, ac)$, while P has a further $\mathcal{SEQ}$-model $M' = (\emptyset, ab)$; the latter is lost along the splitting sequence $S = (a, ab, abc)$, as restricted to C_1, M has smaller gap (viz. $\{a\}$) than M' (whose gap is $\{a, b\}$). However, we get an analogue to Corollary 48.

Proposition 50 *Let P be a normal program such that $WF(P)$ exists and let S be an arbitrary splitting sequence of P. Then P has some $\mathcal{SEQ}^S$-model M such that $M \sqsubseteq WF(P)$, and moreover every $\mathcal{SEQ}^S$-model M of P such that $gap(M) \subseteq gap(WF(P))$ satisfies $M \sqsubseteq WF(P)$.*

The reason is that the well-founded semantics satisfies modularity with respect to splitting sequences. This is a consequence of the following lemma.

Lemma 51 *For every splitting set S of a normal program P such that $WF(P)$ exists, it holds that*

1. *$WF(P)|_S$ is a partial stable model of $b_S(P)$ (recall that $|_S$ denotes restriction to S), and*

2. *$WF(P) = WF(t_S(P) \cup I \cup \{A \leftarrow not\, A \mid A \in J \setminus I\})$, where $WF(b_S(P)) = (I, J)$.*

This lemma in turn follows from Proposition 12 in [16], which states this property for partial stable models of constraint-free (even disjunctive) programs, and $WF(P)$ is the least partial stable model; note also that constraints in P merely determine the existence of $WF(P)$ but do not influence the truth valuation of atoms.

An immediate corollary to Proposition 50 is that normal programs P for which the well-founded model exists and the $\mathcal{SCC}$-model semantics is applicable have some $\mathcal{SCC}$-model that refines the well-founded model $WF(P)$, and moreover that every $\mathcal{SCC}$-model of P which adopts some the undefined atoms in $WF(P)$ as believed true refines $WF(P)$; the same holds for $\mathcal{MJC}$-models.

We finally note that we can, as in the case of all $\mathcal{SEQ}$-models of P, restrict the split $\mathcal{SEQ}$-models of P to those which refine $WF(P)$ by adding respective constraints; recall that $P^{wf} = P \cup I \cup \{\leftarrow A \mid A \notin J\}$ where $WF(P) = (I, J)$.

Proposition 52 *Let P be a normal program such that $WF(P)$ exists. Then for every splitting sequence S of P, it holds that $\mathcal{SEQ}^S(P^{wf}) = \{M \in \mathcal{SEQ}^S(P) \mid gap(M) \subseteq gap(WF(P))\}$.*

Proof. [Sketch] By Proposition 49, $\mathcal{SEQ}(P^{wf}) = \{M \in \mathcal{SEQ}(P) \mid gap(M) \subseteq gap(WF(P))\}$. The result can then be shown by induction along the split sequence S, using Theorem 22 and Lemma 51. □

As a consequence of Propositions 50 and 52, in particular the $\mathcal{SCC}$- and $\mathcal{MJC}$-models of a normal program can be easily restricted such that they refine its well-founded semantics in a paracoherent manner, as discussed at the end of Subsection 8.2.3.

9 Further Issues

9.1 Language extensions

As already mentioned, semi-stable semantics has originally been developed as an extension to p-minimal model semantics [38], a paraconsistent semantics for extended disjunctive logic programs, i.e., programs which besides default negation also allow for strong (classical) negation. A declarative characterization of p-minimal models by means of frames was given by Alcantara et al. [1], who coined the term *Paraconsistent Answer-set Semantics* (PAS) for it. This characterization has been further simplified and underpinned with a logical axiomatization in [27] by using Routley models, i.e., a simpler possible worlds model.

Our characterizations for both, semi-stable models and semi-equilibrium models, can be easily extended to this setting if they are applied to semantic structures which are given by quadruples of interpretations rather than bi-interpretations, respectively to Routley here-and-there models rather than HT-models. Intuitively, this again amounts to considering two 'worlds', each of which consists of a pair of interpretations: one for positive literals (atoms), and one for negative literals (strongly negated atoms). The respective epistemic transformations are unaffected except for the fact that literals are considered rather than atoms. One can also show for both semantics that there is a simple 1-to-1 correspondence to the semi-stable (resp. semi-equilibrium) models of a transformed logic program without strong negation: A given extended program P is translated into a program P' over $\Sigma \cup \{a' \mid a \in \Sigma\}$ without strong negation by replacing each negative literal of the form $-a$ by a'. If (I, J) is a semi-stable (semi-equilibrium) model of P', then

$$(I \cap \Sigma, \{-a \mid a' \in I\}, J \cap \Sigma, \{-a \mid a' \in J\})$$

is a semi-stable (semi-equilibrium) model of P. Note that semi-stable (semi-equilibrium) models of extended logic programs obtained in this way generalize the PAS semantics, which means that they are paraconsistent as well as paracoherent. Logically this amounts to distinguishing nine truth values rather than three, with the additional truth values *undefined*, *believed false*, *believed inconsistent*, *true with contradictory belief*, *false with contradictory belief*, and *inconsistent*. The computational complexity for extended programs is the same.

Compared to [38], we have confined here to propositional programs, as opposed to programs with variables (non-ground programs). However, respective semantics for non-ground programs via their grounding are straightforward. Alternatively, in case of semi-equilibrium models one can simply replace HT-models by Herbrand models of quantified equilibrium logic [30]. Similarly for the other semantics, replacing interpretations in the semantic structures by Herbrand interpretations over a given function-free first-order signature, yields a characterization of the respective models.

9.2 Parametric merging semantics

By the results of Section 7, tractable merging policies that ensure classical coherence (D3) will sometimes merge more components than necessary. To deal with the issues (1) and (2) in Section 6.2, i.e., with all cross-constraints and dependence, a parametric approach that gradually merges more SCCs seems attractive. We briefly outline here one possible such approach, which merges components within bounded distance.

Denote for every $C \in \mathcal{SCC}(P)$ by $D_k(C)$ the set of all descendants of C in $SG(P)$ within distance $k \geq 0$; then we may proceed as follows.

1. create a graph G_k with a node v_r for each constraint r in P, which is labeled with the set

$$\lambda(v_r) = cl_p\Big(\bigcup\{D_k(C_i) \mid C_i \in \mathcal{SCC}(P), C_i \cap At(r) \neq \emptyset\}\Big)$$

   of SCCs; that is, all SCCs within distance k to a SCC C_i that intersects with r are collected into one set, and on the resulting collection D of SCCs a function $cl_p(D)$ is applied. The latter closes D with respect to SCCs C that are on some path between members C_1 and C_2 of D in $SG(P)$.

2. Merge then nodes v_r and $v_{r'}$ (and their labels, using cl_p) such that $\lambda(v_r) \cap \lambda(v'_r) \neq \emptyset$ as long as possible.

3. After that, create a node v for each SCC C that does not occur in any label of the graph, and set $\lambda(v) = \{C\}$;

4. add an edge from v to v', if $v \neq v'$ and $SG(P)$ has some edge (C_i, C_j) where $C_i \in \lambda(v)$ and $C_j \in \lambda(v')$.

The resulting graph G_k is acyclic and distinct nodes have disjoint labels. Similar as for $JG(P)$, any topological ordering $\leq$ of G_k induces a splitting sequence $S_\leq$ (via the node labels $\lambda(v)$, which are taken as union $\bigcup\lambda(v)$ of the SCCs they contain); thanks to an analog of Theorem 33, one can define the $\mathcal{M}_k$-models of P as $\mathcal{M}_k(P) = \mathcal{SEQ}^{S_\leq}(P)$ for an arbitrary $\leq$.

For $k = 0$, we have $D_k(C) = \{C\}$ and thus the node v_r in the initial graph G_0 contains in its label $\lambda(v_r)$ the SCCs that intersect r; the final graph G_0 is such that each $Jx <\in \mathcal{MJC}(P)$ is included in some node label (i.e., $J \subseteq \lambda(v)$ for some node v). Hence, $M^{\mathcal{MJC}}(P) \subseteq \mathcal{M}_0(P)$ holds. As clearly $\mathcal{M}_k(P) \subseteq \mathcal{M}_{k+1}(P)$ holds for every $k \geq 0$, and $\mathcal{M}_k(P) = \mathcal{SEQ}(P)$ for large enough k; as holds, we have a hierarchy of models between $M^{\mathcal{MJC}}(P)$ and $\mathcal{SEQ}(P)$ which eventually establishes (D3); however, the results of Section 7 imply that predicting the least k such that $\mathcal{M}_k(P) \neq \emptyset$ is intractable.

Other relaxed notions of models (using different parameters for cross-constraints and direct dependency) are conceivable; we leave this for future study.

10 Conclusion

In this paper, we have studied paracoherent semantics for answer set programs, that is, semantics that ascribes models to (disjunctive) logic programs with non-monotonic negation even if no answer set (respectively stable model) exists, due to a lack of stability in models caused by cyclic dependency through negation, or due to constraints. Ideally, such a semantics approximates the answer set semantics faithfully and delivers models whenever possible, as expressed by the properties (D1)–(D3); this can be beneficially exploited in scenarios where unexpected inconsistency arises and one needs to stay operational, such as in inconsistency tolerant query answering. Among few well-known semantics which feature these properties are the semi-stable model semantics [38], and the novel semi-equilibrium model semantics, which amends the semi-stable model semantics by eliminating some anomalies. For both semantics, which are defined by program transformations, we have given model-theoretic characterizations in terms of bi-models and HT-models, respectively; in particular, semi-equilibrium models relax the notion of equilibrium models, which reconstruct answer sets in HT-logic, by allowing for minimal sets of unsupported assumptions. We have then refined the semi-equilibrium model semantics with regard to modular program structure, by defining models via splitting sets and splitting sequences; this constrains the set of semi-equilibrium models, in a way that is amenable to modular bottom up evaluation of programs. For that, we have presented canonical semi-equilibrium models for which, in analogy to the classical Stratification Theorem for logic programs, the particular evaluation order does not matter, and we have identified modularity properties for these semantics that allow for flexible rearrangement in evaluation.

Furthermore, we have characterized the complexity of major reasoning tasks of all these semantics, and we have compared semi-equilibrium semantics to related proposals for paracoherent semantics and approximations of answer sets in the literature. Notably, it appeared that semi-equilibrium models coincide with evidential stable models in [39]; our semantic and computational results thus carry over to them. Different from other formalisms such as CR-Prolog [4] or generalizes stable models, [23], unsupported assumptions in semi-stable and semi-equilibrium models serve to block rules but not to establish positive evidence for deriving atoms from rules. Furthermore, we have shown that the well-founded model of a normal logic program is refined by semi-equilibrium models, and that the program can be easily modified such that all semi-equilibrium models refine the well-founded model; the same holds also for canonical semi-equilibrium models. This provides a paracoherent way to refine the well-founded semantics for inconsistency-tolerant query answering, which coincides with the answer set semantics for coherent programs and is in general more informative than the well-founded semantics and supports reasoning by cases, being as close to answer sets as possible.

As for computation, an attractive feature is that canonical semi-equilibrium semantics allows for easy switching from a coherent (answer set) mode to a "paracoherent" evaluation mode in the bottom up evaluation of a program, if incoherence is encountered. And notably, this is possible also for disjunctive logic programs.

10.1 Open issues and outlook

Several issues remain for future work and investigations. A natural issue is to introduce paracoherence for further language extensions besides strong negation and non-ground programs. Fortunately, the generic framework of equilibrium logic makes it easy to define $\mathcal{SEQ}$-semantics for many such extensions, among them nested programs, programs with aggregates and external atoms, hybrid knowledge bases etc. It remains to consider modularity in these extensions and to define suitable refinements of $\mathcal{SEQ}$-models. Particularly interesting are modular logic programs [22, 13] where explicit (by module encapsulation) and implicit modularity (by splitting sets) occur at the same time. Related to the latter are multi-context systems [9], in which knowledge bases exchange beliefs via non-monotonic bridge rules; based on ideas and results of this paper, paracoherent semantics for certain classes of such multi-context systems may be devised.

Besides language extensions, another issue is generalizing the model selection. To this end, preference in gap minimization may be supported, especially if domain-specific information is available; subset-minimality is a natural instance of Occam's razor in lack of such information. Furthermore, preference of higher over lower program components may be considered; however, this intuitively requires more guessing and hinders bottom up evaluation.

On the computation side, developing efficient algorithms and their implementation remain to be done, as well integration into an answer set building framework. Currently, experimental prototypes for computing $\mathcal{SST}(P)$ and $\mathcal{SEQ}(P)$ based on the semantic characterizations are available. Another computation method is filtering the answer sets of the epistemic transformation P^{κ} resp. its extension P^{HT} or the evidential transform $P^{\mathcal{E}}$, which are computed with an ASP solver. However simple such postprocessing is not efficient in general; indeed, the Σ_3^P/Π_3^P-completeness of brave/cautious reasoning, respectively, calls for better methods. An interesting issue in this context is a polynomial transformation of the evaluation of normal and hcf-programs into disjunctive ASP, which by our results is feasible.

We have considered paracoherence based on program transformation, as introduced by Inoue and Sakama [38]. Other notions, like forward chaining construction and strong compatibility [44, 25] might be alternative candidates to deal with paracoherent reasoning in logic programs; this remains to be explored.

Finally, another issue is to investigate the use of paracoherent semantics in AI applications such as diagnosis, where assumptions may be exploited to generate candidate diagnoses, in the vein of the generalised stable model semantics [23], or in systems for planning and reasoning about actions based on ASP, where emerging incoherence should be meaningfully tolerated.

Acknowledgments

We would like to thank the anonymous reviewers of preliminary conference versions of parts of this paper for their comments, and José Alferes and Tomi Janhunen for interesting discussions and suggestions. We are grateful to Diemar Seipel for pointing us to his work on evidential stable models.

A Appendix: Proofs

A.1 Section 3

Proof of Proposition 3. Let r be a rule over Σ, and let (I, J) be a bi-interpretation over Σ.

($\Leftarrow$) Suppose that (I, J) satisfies (a), i.e., $B^+(r) \subseteq I$ and $J \cap B^-(r) = \emptyset$ implies $I \cap H(r) \neq \emptyset$ and $I \cap B^-(r) = \emptyset$. We prove that $(I, J) \models_\beta r$, considering three cases:

1) Assume that $B^+(r) \not\subseteq I$. Then $(I, J) \not\models_\beta a$, for some atom $a \in B^+(r)$, and thus $(I, J) \not\models_\beta B(r)$ which implies $(I, J) \models_\beta r$.

2) Assume that $J \cap B^-(r) \neq \emptyset$, Then $(I, J) \not\models_\beta \neg a$, for some atom $a \in B^-(r)$, and thus $(I, J) \not\models_\beta B(r)$ which implies $(I, J) \models_\beta r$.

3) Assume that $B^+(r) \subseteq I$ and $J \cap B^-(r) = \emptyset$. Then, since (I, J) satisfies (a), it also holds that $I \cap H(r) \neq \emptyset$ and $I \cap B^-(r) = \emptyset$. From $B^+(r) \subseteq I$ and $I \cap B^-(r) = \emptyset$, we conclude that $I \models B(r)$. Moreover, $I \cap H(r) \neq \emptyset$ implies $(I, J) \models_\beta H(r)$. Thus, $(I, J) \models_\beta r$.

By our assumption, one of these three cases applies for (I, J), proving the claim.

($\Rightarrow$) Suppose that $(I, J) \models_\beta r$. We prove that (I, J) satisfies (a), distinguishing two cases:

1) Assume that $(I, J) \not\models_\beta B(r)$. Then either $(I, J) \not\models_\beta a$, for some atom $a \in B^+(r)$, or $(I, J) \not\models_\beta \neg a$, for some atom $a \in B^-(r)$. Hence, $B^+(r) \not\subseteq I$ or $J \cap B^-(r) \neq \emptyset$, which implies that (I, J) satisfies (a).

2) Assume that $(I, J) \models_\beta H(r)$ and $I \models B(r)$. Then $I \cap H(r) \neq \emptyset$ and $I \cap B^-(r) = \emptyset$, and thus (I, J) satisfies (a).

By our assumption, one of the two cases applies for (I, J), which proves the claim. □

Proof of Proposition 4. Let P be a program over Σ. Part (1). First, let (I, J) be a bi-model of P. We prove that $(I, J)^{\kappa,P} \models P^\kappa$.

Towards a contradiction assume the contrary. Then there exists a rule r' in P^κ, such that $(I, J)^{\kappa,P} \not\models r'$. Suppose that r' is not transformed, i.e., $r' \in P$ and $B^-(r') = \emptyset$. Since $(I, J) \models_\beta r'$, by Proposition 3 we conclude that $B^+(r') \subseteq I$ implies $I \cap H(r') \neq \emptyset$ (recall that $B^-(r') = \emptyset$). By construction $(I, J)^{\kappa,P}$ restricted to Σ coincides with I. Therefore, $B^+(r') \subseteq (I, J)^{\kappa,P}$ implies $(I, J)^{\kappa,P} \cap H(r') \neq \emptyset$, i.e., $(I, J)^{\kappa,P} \models r'$, a contradiction.

Next, suppose that r' is obtained by the epistemic transformation of a corresponding rule $r \in P$ of the form (1), and consider the following cases:

– r' is of the form (3): then $\{b_1, \ldots, b_m\} \subseteq (I, J)^{\kappa,P}$, which implies $B^+(r) \subseteq I$. Moreover, $H(r') \cap (I, J)^{\kappa,P} = \emptyset$ by the assumption that $(I, J)^{\kappa,P} \not\models r'$. By construction of $(I, J)^{\kappa,P}$, this implies $J \cap B^-(r) = \emptyset$. Since $(I, J) \models_\beta r$, we also conclude that $I \cap H(r) \neq \emptyset$ and that $I \cap B^-(r) = \emptyset$. Consequently, $J \models B^-(r)$, $a_i \in I$ for some $a_i \in H(r)$, and $I \models B(r)$. Note also, that $B^-(r) \neq \emptyset$ by definition of the epistemic transformation. According to the construction of $(I, J)^{\kappa,P}$, it follows that $\lambda_{r,i} \in (I, J)^{\kappa,P}$, a contradiction to $H(r') \cap (I, J)^{\kappa,P} = \emptyset$.

– r' is of the form (4): in this case, $(I, J)^{\kappa,P} \not\models r'$ implies $\lambda_{r,i} \in (I, J)^{\kappa,P}$ and $a_i \notin (I, J)^{\kappa,P}$. However, by construction $\lambda_{r,i} \in (I, J)^{\kappa,P}$ implies $a_i \in I$; from the latter, again by construction, we conclude $a_i \in (I, J)^{\kappa,P}$, a contradiction.

– r' is of the form (5): in this case, $(I, J)^{\kappa,P} \not\models r'$ implies $\lambda_{r,i} \in (I, J)^{\kappa,P}$ and $b_j \in (I, J)^{\kappa,P}$. Note that $b_j \in (I, J)^{\kappa,P}$ iff $b_j \in I$. A consequence of the latter is that $I \not\models B(r)$, contradicting a requirement for $\lambda_{r,i} \in (I, J)^{\kappa,P}$ (cf. the construction of $(I, J)^{\kappa,P}$).

– r' is of the form (6): by the assumption that $(I, J)^{\kappa,P} \not\models r'$, it holds that $\lambda_{r,k} \in (I, J)^{\kappa,P}$ and $a_i \in (I, J)^{\kappa,P}$, but $\lambda_{r,i} \notin (I, J)^{\kappa,P}$. From the latter we conclude, by the construction of $(I, J)^{\kappa,P}$, that $a_i \notin I$, since all other requirements for the inclusion of $\lambda_{r,i}$ (i.e., $r \in P$, $B^-(r) \neq \emptyset$, $I \models B(r)$, and $J \models B^-(r)$) must be satisfied as witnessed by $\lambda_{r,k} \in (I, J)^{\kappa,P}$. However, if $a_i \notin I$, then $a_i \notin (I, J)^{\kappa,P}$ (again by construction), contradiction.

This concludes the proof of the fact that if (I, J) is a bi-model of P, then $(I, J)^{\kappa,P} \models P^\kappa$.

Part (2). Let M be a model of P^κ. We prove that $\beta(M \cap \Sigma^\kappa) = (I, J)$ is a bi-model of P. Note that by construction $I = M \cap \Sigma$ and $J = \{a \mid Ka \in M\}$. First, we consider any rule r in P such that $B^-(r) = \emptyset$. Then $r \in P^\kappa$, $J \cap B^-(r) = \emptyset$ and $I \cap B^-(r) = \emptyset$. Hence, by Proposition 3, we need to show that $B^+(r) \subseteq (M \cap \Sigma)$ implies $(M \cap \Sigma) \cap H(r) \neq \emptyset$. Since $r \in P^\kappa$, this follows from the assumption, i.e., $M \models P^\kappa$ implies $M \models r$, and therefore if $B^+(r) \subseteq M$, then $M \cap H(r) \neq \emptyset$. Since r is over Σ, this proves the claim for all $r \in P$ such that $B^-(r) = \emptyset$.

It remains to show that $(I, J) \models_\beta r$ for all $r \in P$ such that $B^-(r) \neq \emptyset$. Towards a contradiction assume that this is not the case, i.e., (i) $B^+(r) \subseteq (M \cap \Sigma)$, (ii) $J \cap B^-(r) = \emptyset$, and either (iii) $(M \cap \Sigma) \cap H(r) = \emptyset$ or (iv) $(M \cap \Sigma) \cap B^-(r) \neq \emptyset$ hold for some $r \in P$ of the form (1), such that $B^-(r) \neq \emptyset$. Conditions (i) and (ii), together with $M \models P^\kappa$, imply that $\lambda_{r,i}$ is in M, for some $1 \leq i \leq l$ (cf. the rule of the form (3) in the epistemic transformation of r). Consequently, a_i is in M (cf. the corresponding rule of the form (4) in the epistemic transformation of r), and hence $a_i \in (M \cap \Sigma)$. This rules out (iii), so (iv) must hold, i.e., $b_j \in (M \cap \Sigma)$, for some $m + 1 \leq j \leq n$. But then, M satisfies the body of a constraint in P^κ (cf. the corresponding rule of the form (5) in the epistemic transformation of r), contradicting $M \models P^\kappa$. This proves that there exists no $r \in P$ such that $B^-(r) \neq \emptyset$ and $(I, J) \not\models_\beta r$, and thus concludes our proof of $(I, J) \models_\beta r$. Since $r \in P$ was arbitrary, it follows that $\beta(M \cap \Sigma^\kappa)$ is a bi-model of P. □

Proof of Theorem 5. Let P be a program over Σ. The proof uses the following lemmas.

Lemma 53 *If $M \in \mathcal{AS}(P^\kappa)$, then $\beta(M \cap \Sigma^\kappa)$ satisfies (i).*

Proof. Towards a contradiction assume that $M \in \mathcal{AS}(P^\kappa)$ and $\beta(M \cap \Sigma^\kappa) = (I, J)$ does not satisfy (i). Then, there exists a bi-model (I', J) of P, such that $I' \subset I$. By Proposition 4, $(I', J)^{\kappa,P} \models P^\kappa$. Note that $(I', J)^\kappa \subset (M \cap \Sigma^\kappa)$. Let $S' = \{\lambda_{r,i} \mid \lambda_{r,i} \in (I', J)^{\kappa,P}\}$ and let $S = \{\lambda_{r,i} \mid \lambda_{r,i} \in M\}$. We show that $S' \subseteq S$. Suppose that this is not the case and assume that $\lambda_{r,i} \in S'$ and $\lambda_{r,i} \notin S$, for some $r \in P$ of the form (1) and $1 \leq i \leq l$. By the construction of $(I', J)^{\kappa,P}$, we conclude that $a_i \in I'$, $I' \models B(r)$, and $J \models B^-(r)$. Since $I' \subset I$, it also holds that $a_i \in I$ and that $I \models B^+(r)$. Consider the rule of the form (3) of the epistemic transformation of r. We conclude that $\{b_1, \ldots, b_m\} \subseteq M$ (due to $I \models B^+(r)$), and that $M \not\models Kc_1 \vee \ldots \vee Kc_n$ (due to $J \models B^-(r)$). But $M \models P^\kappa$, hence $\lambda_{r,k}$ is in M, for some $1 \leq k \leq l$. However, considering the corresponding rule of the form (6) of the epistemic transformation of r, we also conclude that $\lambda_{r,i} \in M$, a contradiction. Therefore $S' \subseteq S$ holds, and since $(I', J)^\kappa \subset (M \cap \Sigma^\kappa)$, we conclude that $(I', J)^{\kappa,P} \subset M$. The latter contradicts the assumption that M is an answer-set, i.e., a minimal model, of P^κ. This concludes the proof of the lemma. □

Lemma 54 *If (I, J) is a bi-model of P that satisfies (i) and (ii), then there exists some $M \in \mathcal{AS}(P^\kappa)$, such that $\beta(M \cap \Sigma^\kappa) = (I, J)$.*

Proof. Let (I, J) be a bi-model of P that satisfies (i) and (ii). If $(I, J)^{\kappa,P} \in \mathcal{AS}(P^\kappa)$, then (c) holds since $\beta((I, J)^{\kappa,P} \cap \Sigma^\kappa) = (I, J)$. If $(I, J)^{\kappa,P} \notin \mathcal{AS}(P^\kappa)$, then there exists a minimal model, i.e. an answer set, M' of P^κ, such that $M' \subset (I, J)^{\kappa,P}$. Let $(I', J') = \beta(M' \cap \Sigma^\kappa)$. Then $I' \subseteq I$ and $J' \subseteq J$ holds by construction and the fact that $M' \subset (I, J)^{\kappa,P}$. Towards a contradiction, assume that $I' \subset I$. We show that then (I', J) is a bi-model of P. Suppose that (I', J) is not a bi-model of P. Then, by Proposition 3, there exists $r \in P$, such that $B^+(r) \subseteq I'$, $J \cap B^-(r) = \emptyset$, and either $I' \cap H(r) = \emptyset$ or $I' \cap B^-(r) \neq \emptyset$. Note that $B^+(r) \subseteq I'$ implies $B^+(r) \subseteq I$, and since (I, J) is a bi-model of P, we conclude $I \cap H(r) \neq \emptyset$ and $I \cap B^-(r) = \emptyset$. The latter implies $I' \cap B^-(r) = \emptyset$, hence $I' \cap H(r) = \emptyset$ holds. If $B^-(r) = \emptyset$, then r is in P^κ and $M' \not\models r$, contradiction. Thus, $B^-(r) \neq \emptyset$. However, in this case the epistemic transformation of r is in P^κ. Since $J \cap B^-(r) = \emptyset$ and $J' \subseteq J$ together imply $J' \cap B^-(r) = \emptyset$, we conclude that for the rule of the form (3) of the epistemic transformation of r, it holds that $\{b_1, \ldots, b_m\} \subseteq M'$ (due to $B^+(r) \subseteq I'$), and that $M' \not\models Kc_1 \vee \ldots \vee Kc_n$ (due to $J' \cap B^-(r) = \emptyset$). Moreover $M' \models P^\kappa$, hence $\lambda_{r,i}$ is in M', for some $1 \leq i \leq l$. Considering the corresponding rule of the form (4) of the epistemic transformation of r, we also conclude that $a_i \in M'$, a contradiction to $I' \cap H(r) = \emptyset$. This proves that (I', J) is a bi-model of P, and thus contradicts the assumption that (I, J) satisfies (i). Consequently, $I' = I$. Now if $J' \subset J$, then we obtain a contradiction with the assumption that (I, J) satisfies (ii). Therefore also $J' = J$, which concludes the proof of the Lemma. □

The proof of Theorem 5 is then as follows.

Part (1). Let (I, J) be a bi-model of P that satisfies (i)-(iii). We prove that $(I, J)^\kappa \in \mathcal{SST}(P)$. By Lemma 54, we conclude that there exists some $M \in \mathcal{AS}(P^\kappa)$ such that $\beta(M \cap \Sigma^\kappa) = (I, J)$. It remains to show that M is maximal canonical. Towards a contradiction assume the contrary. Then, there exists $M' \in \mathcal{AS}(P^\kappa)$ such that $gap(M') \subset gap(M)$. Let $(I', J') = \beta(M' \cap \Sigma^\kappa)$. By Lemma 53, (I', J') satisfies (i), and by construction since $gap(M') \subset gap(M)$, it holds that $J' \setminus I' \subset J \setminus I$. However, this contradicts the assumption that (I, J) satisfies (iii). Therefore, M is maximal canonical, and hence $(I, J)^\kappa \in \mathcal{SST}(P)$.

Part (2). Let $I^\kappa \in \mathcal{SST}(P)$. We show that $\beta(I^\kappa)$ is a bi-model of P that satisfies (i)-(iii). Let $(I, J) = \beta(I^\kappa)$ and let M be a maximal canonical answer set of P^κ corresponding to I^κ. Then, $\beta(M \cap \Sigma^\kappa) = (I, J)$ by construction, and (I, J) satisfies (i) by Lemma 53.

Towards a contradiction first assume that (I, J) does not satisfy (iii). Then there exists a bi-model (I', J') of P such that (I', J') satisfies (i) and $J' \setminus I' \subset J \setminus I$. Let $M' = (I', J')^{\kappa,P}$ and note that if

$M' \in \mathcal{AS}(P^\kappa)$, we arrive at a contradiction to $M \in mc(\mathcal{AS}(P^\kappa))$, since $gap(M') \subset gap(M)$. Thus, there exists $M'' \in \mathcal{AS}(P^\kappa)$, such that $M'' \subset M'$. Let $(I'', J'') = \beta(M'' \cap \Sigma^\kappa)$. We show that (I'', J') is a bi-model of P, and thus by (i) it follows that $I'' = I'$. Towards a contradiction, suppose that (I'', J') is not a bi-model of P. Then, by Proposition 3, there exists $r \in P$, such that $B^+(r) \subseteq I''$, $J' \cap B^-(r) = \emptyset$, and either $I'' \cap H(r) = \emptyset$ or $I'' \cap B^-(r) \neq \emptyset$. Note that $B^+(r) \subseteq I''$ implies $B^+(r) \subseteq I'$, and since (I', J') is a bi-model of P, we conclude $I' \cap H(r) \neq \emptyset$ and $I' \cap B^-(r) = \emptyset$. The latter implies $I'' \cap B^-(r) = \emptyset$, hence $I'' \cap H(r) = \emptyset$ holds. If $B^-(r) = \emptyset$, then r is in P^κ and $M'' \not\models r$, contradiction. Thus, $B^-(r) \neq \emptyset$. However, in this case the epistemic transformation of r is in P^κ. Since $J' \cap B^-(r) = \emptyset$ and $J'' \subseteq J'$ together imply $J'' \cap B^-(r) = \emptyset$, we conclude that for the rule of the form (3) of the epistemic transformation of r, it holds that $\{b_1, \ldots, b_m\} \subseteq M''$ (due to $B^+(r) \subseteq I''$), and that $M'' \not\models Kc_1 \vee \ldots \vee Kc_n$ (due to $J'' \cap B^-(r) = \emptyset$). Moreover $M'' \models P^\kappa$, hence $\lambda_{r,i}$ is in M'', for some $1 \leq i \leq l$. Considering the corresponding rule of the form (4) of the epistemic transformation of r, we also conclude that $a_i \in M''$, a contradiction to $I'' \cap H(r) = \emptyset$. This proves that (I'', J') is a bi-model of P. From the assumption that (I', J') satisfies (i), it follows that $I'' = I'$. Therefore $gap(M'') \subseteq gap(M')$ holds, which implies $gap(M'') \subset gap(M)$, a contradiction to $M \in mc(\mathcal{AS}(P^\kappa))$. This proves (I, J) satisfies (iii).

Next assume that (I, J) does not satisfy (ii). Then, there exists a bi-model (I, J') of P, such that $J' \subset J$. We show that (I, J') satisfies (i). Otherwise, there exists a bi-model (I', J') of P, such that $I' \subset I$; but then also (I', J) is a bi-model of P. To see the latter, assume that there exists a rule $r \in P$, such that $B(r) \subseteq I'$, $J \cap B^-(r) = \emptyset$ and either $I' \cap H(r) = \emptyset$ or $I' \cap B^-(r) \neq \emptyset$. Since $J' \subset J$, it then also holds that $J' \cap B^-(r) = \emptyset$. This contradicts the assumption that (I', J') is a bi-model of P, hence $(I', J) \models_\beta P$. The latter is a contradiction to the assumption that (I, J) satisfies (i), proving that (I, J') satisfies (i). Since (I, J) satisfies (iii), we conclude that $J' \setminus I = J \setminus I$. Now let $S' = \{\lambda_{r,i} \mid \lambda_{r,i} \in (I, J')^{\kappa,P}\}$ and let $S = \{\lambda_{r,i} \mid \lambda_{r,i} \in M\}$. It holds that $S' \not\subseteq S$ (otherwise $(I, J')^{\kappa,P} \subset M$, a contradiction to $M \in \mathcal{AS}(P^\kappa)$), i.e., there exists $r \in P$ of the form (1) and $1 \leq i \leq l$, such that $\lambda_{r,i} \in S$ and $\lambda_{r,i} \notin S'$. From the former, since M is a minimal model of P^κ, we conclude that $I \models B^+(r)$, $a_i \in I$, and $J \cap B^-(r) = \emptyset$. Since $J' \subset J$, also $J' \cap B^-(r) = \emptyset$. This implies that $\lambda_{r,k} \in S'$, for some $1 \leq k \neq i \leq l$ (otherwise $(I, J')^{\kappa,P}$ does not satisfy the rule of form (3) corresponding to r in P^κ, a contradiction to $(I, J')^{\kappa,P} \models P^\kappa$). However, since $a_i \in I$, and thus $a_i \in (I, J')^{\kappa,P}$, and since $\lambda_{r,k} \in (I, J')^{\kappa,P}$, we conclude that $\lambda_{r,i} \in (I, J')^{\kappa,P}$ (cf. the respective rule of form (6) of the epistemic transformation of r). This contradicts $\lambda_{r,i} \notin S'$, and thus proves that (I, J) satisfies (ii). □

A.2 Section 4

Proof of Proposition 7. Let P be a program over Σ.

Part (1). Let (I, J) be a bi-model of P, such that $(I, J)^\kappa$ satisfies Property **N** and Property **K**, for all $r \in P$. We show that (I, J) is an HT-model of P. Since $(I, J)^\kappa$ satisfies Property **N**, it holds that $a \in I$ implies $a \in J$, therefore $I \subseteq J$, i.e., (I, J) is an HT-interpretation. For every rule $r \in P$, $(I, J) \models_\beta r$ implies $(I, J) \not\models_\beta B(r)$, or $(I, J) \models_\beta H(r)$ and $I \models B(r)$. First suppose that $(I, J) \not\models_\beta B(r)$. Then $(I, J) \not\models B(r)$ (note that for a conjunction of literals, such as $B(r)$, the satisfaction relations do not differ). Moreover, since $(I, J)^\kappa$ satisfies Property **K** for r, it holds that $J \models r$. To see the latter, let Kr denote the rule obtained from r by replacing every $a \in \Sigma$ occurring in r by Ka, and let KJ denote the set $\{Ka \in (I, J)^\kappa \mid a \in \Sigma\}$. Then, $(I, J)^\kappa$ satisfies Property **K** for r iff $KJ \models Kr$. Observing that $KJ = \{Ka \mid a \in J\}$, we conclude that $J \models r$. This proves $(I, J) \models r$, if $(I, J) \not\models_\beta B(r)$. Next assume that $(I, J) \models_\beta H(r)$ and $I \models B(r)$. We conclude that $(I, J) \models H(r)$ (the satisfaction relations also coincide for disjunctions of atoms). As $(I, J)^\kappa$ satisfies Property **K** for r, it follows $J \models r$. This proves $(I, J) \models r$, for every $r \in P$; in other words, (I, J) is an HT-model of P.

Part (2). Let (H,T) be an HT-model of P. We show that $(H,T)^\kappa$ satisfies Property **N** and Property **K**, for all $r \in P$. As a consequence of $H \subseteq T$, for every $a \in (H,T)^\kappa$ such that $a \in \Sigma$, it also holds that $Ka \in (H,T)^\kappa$, i.e., $(H,T)^\kappa$ satisfies Property **N**. Moreover, $(H,T) \models P$ implies $T \models r$, for all $r \in P$. Let $KT = \{Ka \mid a \in T\}$ and let Kr as above; $T \models r$ implies $KT \models Kr$, for all $r \in P$. By construction of $(H,T)^\kappa$ and definition of Property **K** for r, we conclude that $(H,T)^\kappa$ satisfies Property **K** for all $r \in P$. □

Proof of Theorem 8. Let P be a program over Σ.

Part (1). Let (H,T) be an HT-model of P that satisfies (i') and (ii'). We show that $(H,T)^\kappa \in \mathcal{SEQ}(P)$. Towards a contradiction, first assume that $(H,T)^\kappa \notin MM(HT^\kappa(P))$. Then, there exists an HT-model (H',T') of P, such that $H' \subseteq H$, $T' \subseteq T$, and at least one of the inclusions is strict. Suppose that $H' \subset H$. Then (H',T) is an HT-model of P (by a well-known property of HT), a contradiction to the assumption that (H,T) satisfies (i'). Hence, $H' = H$ and $T' \subset T$ must hold. Moreover, by the same argument (H',T') also satisfies (i'). But, since $T' \setminus H' \subset T \setminus H$, this is in contradiction to the assumption that (H,T) satisfies (ii'). Consequently, $(H,T)^\kappa \in MM(HT^\kappa(P))$. We continue the indirect proof assuming that $(H,T)^\kappa \notin mc(MM(HT^\kappa(P)))$, i.e., there exists an HT-model (H',T') of P, such that $T' \setminus H' \subset T \setminus H$ and $(H',T')^\kappa \in MM(HT^\kappa(P))$. The latter obviously implies that (H',T') satisfies (i'). Again, this contradicts that (H,T) satisfies (ii'), which proves that $(H,T)^\kappa \in \mathcal{SEQ}(P)$.

Part (2). Let $I^\kappa \in \mathcal{SEQ}(P)$. We show that $\beta(I^\kappa)$ is an HT-model of P that satisfies (i') and (ii'). Let $\beta(I^\kappa) = (H,T)$. Towards a contradiction first assume that (H,T) is not an HT-model of P. Then by the definition of $\mathcal{SEQ}(P)$, and the fact that I^κ uniquely corresponds to sets H and T, we conclude that $I^\kappa \notin mc(MM(HT^\kappa(P)))$, i.e., $I^\kappa \notin \mathcal{SEQ}(P)$; contradiction. Next, suppose that (H,T) does not satisfy (i'). Then, $I^\kappa \notin MM(HT^\kappa(P))$, as witnessed by $(H',T)^\kappa$ for an HT-model (H',T) such that $H' \subset H$, which exists if (H,T) does not satisfy (i'). Therefore, $I^\kappa \notin mc(MM(HT^\kappa(P)))$, i.e., $I^\kappa \notin \mathcal{SEQ}(P)$; contradiction. Eventually assume that (H,T) does not satisfy (ii'). Then, $I^\kappa \notin mc(MM(HT^\kappa(P)))$, as witnessed by $(H',T')^\kappa$ for an HT-model (H',T'), such that $T' \setminus H' \subset T \setminus H$ and (H',T') satisfies (i')—note that (H',T') exists if (H,T) does not satisfy (ii'). To see that $(H',T')^\kappa$ is a witness for $I^\kappa \notin mc(MM(HT^\kappa(P)))$, observe that either $(H',T')^\kappa \in MM(HT^\kappa(P))$ or there exists an HT-model (H',T''), such that $(H',T'')^\kappa \in MM(HT^\kappa(P))$ and $T'' \subset T'$ (which implies $T'' \setminus H' \subset T' \setminus H' \subset T \setminus H$). This proves that $I^\kappa \notin \mathcal{SEQ}(P)$, again a contradiction. This concludes the proof that $\beta(I^\kappa)$ is an HT-model of P that satisfies (i') and (ii'). □

Proof of Theorem 9. Let P be a program over Σ, and let I^κ be an interpretation over Σ^κ. The proof uses the following lemmas.

Lemma 55 *If $M \models P^{HT}$, then $\beta(M \cap \Sigma^\kappa)$ is an HT-model of P.*

Proof. Let $(I,J) = \beta(M \cap \Sigma^\kappa)$. Since $M \models P^\kappa$, (I,J) is a bi-model of P by Proposition 4. Moreover, $M \cap \Sigma^\kappa = (I,J)^\kappa$ and $(I,J)^\kappa$ satisfies Property **N**, otherwise there is an atom $a \in M$ such that $Ka \notin M$, a contradiction to $M \models Ka \leftarrow a$. Also, $(I,J)^\kappa$ satisfies Property **K** for all $r \in P$; otherwise, if Property **K** does not hold for some $r \in P$ of the form (1), then $M \models Kb_1 \wedge \ldots \wedge Kb_m$ and $M \not\models Ka_1 \vee \ldots \vee Ka_l \vee Kc_1 \vee \ldots \vee Kc_n$, i.e., $M \not\models P^{HT}$; contradiction. Hence by Proposition 7, (I,J) is a HT-model of P. □

Next, we prove:

Lemma 56 *If (H,T) is an HT-model of P, then $(H,T)^{\kappa,P} \models P^{HT}$.*

Proof. Note that every HT-model of P is a bi-model of P. Assume the contrary; then $(H,T) \models r$ and $(H,T) \not\models_\beta r$, for some $r \in P$. Then, $H \not\models B(r)$, while $(H,T) \models B(r)$, must hold. However,

$(H,T) \models B(r)$ implies $B^+(r) \subseteq H$ and $B^-(r) \cap H = \emptyset$, and therefore $H \models B(r)$; contradiction. This proves that (H,T) is a bi-model of P. Consequently, $(H,T)^{\kappa,P} \models P^\kappa$ by Proposition 4. Moreover, since (H,T) is an HT-model, $(H,T)^\kappa$ satisfies Property **N** (and Property **K** for all $r \in P$) by Proposition 7. Because $(H,T)^{\kappa,P} \cap \Sigma^\kappa = (H,T)^\kappa$, this implies that $(H,T)^{\kappa,P} \models r$, for all rules of the form $Ka \leftarrow a$ in $P^{HT} \setminus P^\kappa$ (this is an obvious consequence of Property **N**). For the remaining rules r in $P^{HT} \setminus P^\kappa$, $(H,T)^{\kappa,P} \models r$ is a simple consequence of $T \models P$. This proves $(H,T)^{\kappa,P} \models P^{HT}$. □

Lemma 57 *For every $M \in \mathcal{AS}(P^{HT})$, $\beta(M \cap \Sigma^\kappa)$ satisfies (i') in Theorem 8.*

Proof. Towards a contradiction assume the contrary. Then there exists an HT-model (H',T) of P such that $H' \subset H$. Note that $M \in \mathcal{AS}(P^{HT})$ implies $M = \beta(M \cap \Sigma^\kappa)^{\kappa,P}$. Since the latter is a model of P^{HT} by Lemma 56, M must be a subset thereof; however it obviously cannot be a strict subset on Σ^κ. By construction of $\beta(M \cap \Sigma^\kappa)^{\kappa,P}$ and the rules of form (6) of the epistemic transformation, we also conclude that $\lambda_{r,i} \in \beta(M \cap \Sigma^\kappa)^{\kappa,P}$ implies $\lambda_{r,i} \in M$, for any $r \in P$ of the form (1) and $1 \leq i \leq l$. This proves $M = \beta(M \cap \Sigma^\kappa)^{\kappa,P}$. Now consider $M' = (H',T)^{\kappa,P}$. Then, $M' \subset M$ by construction, and $M' \models P^{HT}$ by Lemma 56. This is a contradiction to the assumption that $M \in \mathcal{AS}(P^{HT})$, and thus proves that $\beta(M \cap \Sigma^\kappa)$ satisfies (i'). □

Lemma 58 *For every HT-model (H,T) of P that satisfies (i') of Theorem 8, there exists some $M \in \mathcal{AS}(P^{HT})$ such that $gap(M) \subseteq gap((H,T)^\kappa)$.*

Proof. Since $(H,T)^{\kappa,P} \models P^{HT}$ by Lemma 56, there exists $M \in \mathcal{AS}(P^{HT})$, such that $M \subseteq (H,T)^{\kappa,P}$. To prove the lemma, it suffices to show that $M \cap \Sigma = H$. Assume the contrary; then by (d) there exists an HT-model (H',T') of P, such that $H' \subset H$ and $T' \subseteq T$. However, then $(H',T) \models P$, which contradicts the assumption that (H,T) satisfies (i'). □

The proof of Theorem 9 is then as follows.

($\Leftarrow$) Suppose that $I^\kappa \in \{M \cap \Sigma^\kappa \mid M \in mc(\mathcal{AS}(P^{HT}))\}$. We prove $I^\kappa \in \mathcal{SEQ}(P)$ via Theorem 8. Let $M \in mc(\mathcal{AS}(P^{HT}))$, such that $I^\kappa = M \cap \Sigma^\kappa$, and let $(I,J) = \beta(M \cap \Sigma^\kappa)$. Then, (I,J) is an HT-model of P by Lemma 55 and (I,J) satisfies (i') in Theorem 8 by Lemma 57. We prove that (I,J) satisfies (ii') in Theorem 8. Towards a contradiction, assume that this is not the case, then there exists an HT-model (H,T) of P, such that $T \setminus H \subset J \setminus I$ and (H,T) satisfies (i'). According to Lemma 58, there exists $M' \in \mathcal{AS}(P^{HT})$, such that $gap(M') \subseteq gap((H,T)^\kappa)$, which implies $gap(M') \subset gap(M)$ due to $T \setminus H \subset J \setminus I$. This contradicts the assumption that $M \in mc(\mathcal{AS}(P^{HT}))$, and thus proves that (I,J) satisfies (ii') in Theorem 8. We conclude that $I^\kappa \in \mathcal{SEQ}(P)$.

($\Rightarrow$) Suppose that $I^\kappa \in \mathcal{SEQ}(P)$. We prove $I^\kappa \in \{M \cap \Sigma^\kappa \mid M \in mc(\mathcal{AS}(P^{HT}))\}$. Let $(H,T) = \beta(I^\kappa)$. By Theorem 8, (H,T) is an HT-model of P that satisfies (i') and (ii'). We show that there exists $M \in mc(\mathcal{AS}(P^{HT}))$ such that $\beta(M \cap \Sigma^\kappa) = (H,T)$. Since $(H,T)^{\kappa,P} \models P^{HT}$, there exists $M \in \mathcal{AS}(P^{HT})$ such that $M \subseteq (H,T)^{\kappa,P}$. Since (H,T) satisfies (i'), it holds that $M \cap \Sigma = H$. Moreover, $M \cap \Sigma^\kappa \subset (H,T)^\kappa$ contradicts the fact that (H,T) satisfies (ii'), because then $\beta(M \cap \Sigma^\kappa) = (H,T')$ is an HT-model of P, such that $T' \setminus H \subset T \setminus H$ and (H,T') satisfies (i') due to Lemma 57. Hence, $\beta(M \cap \Sigma^\kappa) = (H,T)$. It remains to show that $M \in mc(\mathcal{AS}(P^{HT}))$. If this is not the case, then some HT-model (H',T') of P exists such that $T' \setminus H' \subset T \setminus H$. Since $(H',T') = \beta(M' \cap \Sigma^\kappa)$ for some $M' \in \mathcal{AS}(P^{HT})$, we conclude by Lemma 57 that (H',T') satisfies (i'), which again leads to a contradiction of the fact that (H,T) satisfies (ii'). This proves that $M \in mc(\mathcal{AS}(P^{HT}))$. As $M \cap \Sigma^\kappa = I^\kappa$, we conclude that $I^\kappa \in \{M \cap \Sigma^\kappa \mid M \in mc(\mathcal{AS}(P^{HT}))\}$. □

Proof of Proposition 10. Let P be a program over Σ. If P has a model M, then (M, M) is an HT-model of P. Therefore $HT^{\kappa}(P) \neq \emptyset$, which implies $MM(HT^{\kappa}(P)) \neq \emptyset$, and thus $mc(MM(HT^{\kappa}(P))) \neq \emptyset$. We conclude that $\mathcal{SEQ}(P) \neq \emptyset$, i.e., P has a semi-equilibrium model. $\Box$

Proof of Proposition 11. Let P be a coherent program over Σ, and let $Y \in \mathcal{AS}(P)$. Then (Y, Y) is an HT-model of P that satisfies (i') in Theorem 8, since it is in equilibrium. Moreover, it trivially satisfies also (ii') because $Y \setminus Y = \emptyset$. Hence, $(Y, Y)^{\kappa} \in \mathcal{SEQ}(P)$.

As P is coherent, there exists $(T, T) \in HT(P)$ that satisfies (i') in Theorem 8 and (trivially) (ii'). Hence, $gap(I^{\kappa}) = \emptyset$ for all $I^{\kappa} \in \mathcal{SEQ}(P)$. Moreover, $\beta(I^{\kappa})$ is of the form (Y, Y), and $Y \in \mathcal{AS}(P)$. $\Box$

A.3 Section 5

Proof of Proposition 14. If $(X, Y) \in \mathcal{SEQ}^S(P)$, then there exists some $(I, J) \in \mathcal{SEQ}(b_S(P))$ such that $(X, Y) \in \mathcal{SEQ}(P^S(I, J))$. We will prove that $(I, J) = (X, Y)|_S$. Obviously $I \subseteq J \subseteq S$. Moreover because $(X, Y) \models a$ for each $a \in I$, we have $a \in X$ for all $a \in I$, so $I \subseteq X$; because $(X, Y) \models \{\leftarrow \mathit{not}\, a \mid a \in J\}$, then $a \in Y$ for all $a \in J$, so $J \subseteq Y$; and because $(X, Y) \models \{\leftarrow a \mid a \in S \setminus J\}$, then $a \notin Y$ for all $a \in S \setminus J$, so $(S \setminus J) \cap Y = \emptyset$. In particular we obtain that $I \subseteq X \cap S$ and $J \subseteq Y \cap S$. We know that $(X, Y) \models P^S(I, J)$. So if we consider $a \in X \cap S$, then $a \in H(r)$ for some rule $r \in P \setminus b_S(P) \cup \{a \mid a \in I\}$. But because $a \in S$, it follows that $r \notin P \setminus b_S(P)$, so $r \in \{a \mid a \in I\}$. Therefore $a \in I$, that is $I = X \cap S$. Moreover if we consider an atom $a \in Y \cap S$, then $a \in Y$ and $a \in S$, and because $(S \setminus J) \cap Y = \emptyset$, we obtain that $a \in J$, that is $J = Y \cap S$. In conclusion, we have that $(X \cap S, Y \cap S) = (I, J)$ is a semi-equilibrium model of $b_S(P)$. $\Box$

Proof of Lemma 16. Suppose that (X, Y) is an HT-model of $P^S(I, J)$. Hence, $(X, Y) \models P \setminus b_S(P)$. It remains to show that $(X, Y) \models r$ for every $r \in b_S(P)$. Suppose that r has the form (1). By assumption $(I, J) \in \mathcal{SEQ}(b_S(P))$, hence we conclude that $(I, J) \models b_S(P)$.

If $(I, J) \models a_i$ for some $a_i \in H(r)$, then $a_i \in I$ and because $(X, Y) \models P^S(I, J)$, we have $(X, Y) \models a_i$, i.e. $(X, Y) \models r$.

If we assume that $(I, J) \not\models b_1 \wedge ... \wedge b_m \wedge \neg c_1 \wedge ... \wedge \neg c_n$, then there exists some $b_j \in B^+(r)$ such that $(I, J) \not\models b_j$ or some $c_k \in B^-(r)$ such that $(I, J) \not\models \neg c_k$, that is, by definition of HT-satisfaction that $b_j \notin I$ respectively $c_k \in J$.

In the first case, b_j is not in the head of any other rule in $P \setminus b_S(P)$, for which $b_j \notin X$ and so $(X, Y) \models r$.

In the second case, we have in $P^S(I, J)$ the rule $\leftarrow \mathit{not}\, c_k$; this implies $c_k \in Y$, and therefore, also in this case, $(X, Y) \models r$. $\Box$

Proof of Proposition 15. Let $(X, Y) \in \mathcal{SEQ}^S(P)$. Then there exists $(I, J) \in \mathcal{SEQ}(b_S(P))$ such that $(X, Y) \in \mathcal{SEQ}(P^S(I, J))$. By Lemma 16, (X, Y) is an HT-model of P. So, by definition of semi-equilibrium model, remains to prove the h-minimality and the gap-minimality of (X, Y). Suppose by contradiction that there exists some $(X', Y) \models P$ with $X' \subset X$. So that $(X', Y) \models t_S(P)$ and $(X', Y) \models b_S(P)$. By this last sentence we also obtain that $(X' \cap S, Y \cap S) \models b_S(P)$, but by Proposition 14, $(X \cap S, Y \cap S) \in \mathcal{SEQ}(b_S(P))$. So by the h-minimality of the semi-equilibrium model $(X \cap S, Y \cap S)$ of the bottom of P, we have that $(X' \cap S) \not\subset (X \cap S)$. But because $X' \subset X$ implies that $(X' \cap S) \subseteq (X \cap S)$, then necessarily $X' \cap S = X \cap S$. So that $(X' \cap S, Y \cap S) = (X \cap S, Y \cap S) = (I, J)$. Therefore

$$(X' \cap S, Y \cap S) \models \{a \mid a \in I\} \cup \{\leftarrow \mathit{not}\, a \mid a \in J\} \cup \{\leftarrow a \mid a \in S \setminus J\}.$$

In particular $(X', Y) \models \{a \mid a \in I\} \cup \{\leftarrow \mathit{not}\, a \mid a \in J\} \cup \{\leftarrow a \mid a \in S \setminus J\}$. And because $(X', Y) \models t_S(P)$, we conclude that $(X', Y) \models P^S(I, J)$ against the h-minimality of (X, Y) respect to $P^S(I, J)$. Similarly, suppose by contradiction that there exists some $(X', Y') \models P$ and

(1) there is no $(X'', Y') \models P$ such that $X'' \subset X'$ and

(2) $Y' \setminus X' \subset Y \setminus X$.

Moreover, we suppose that

(3) $gap(X, Y)$ is minimal among the gaps of the HT-models that satisfy (1) and (2).

Because $(X', Y') \models P$, it holds that $(X', Y') \models t_S(P)$ and $(X', Y') \models b_S(P)$. From this we obtain that $(X' \cap S, Y' \cap S) \models b_S(P)$ and by condition (2) we obtain that

$$(Y' \cap S) \setminus (X' \cap S) = (Y' \setminus X') \cap S \subseteq (Y \setminus X) \cap S = (Y \cap S) \setminus (X \cap S).$$

Moreover $(X', Y')|_S$ satisfies the h-minimality with respect to $b_S(P)$. In fact if by contradiction there exists $(I', Y' \cap S) \models b_S(P)$, such that $I' \subset X' \cap S$, then $(I' \cup (X' \setminus S), Y') \models P$ and $I' \cup (X' \setminus S) \subset (X' \cap S) \cup (X' \setminus S) = X'$ against the condition (1). By Proposition 14, $(X \cap S, Y \cap S) \in \mathcal{SEQ}(b_S(P))$, so we have necessarily that $(Y' \cap S) \setminus (X' \cap S) = (Y \cap S) \setminus (X \cap S) = J \setminus I$. Otherwise $(X, Y)|_S$ could not be a semi-equilibrium model of $b_S(P)$, because $(X', Y')|_S$ contradicts the gap-minimality of $(X, Y)|_S$. Therefore $(X', Y')|_S \in \mathcal{SEQ}(b_S(P))$, because if there exists $(\hat{I}, \hat{J}) \models b_S(P)$, that satisfies the *h-minimality property* and $\hat{J} \setminus \hat{I} \subset (Y' \cap S) \setminus (X' \cap S)$, then $\hat{J} \setminus \hat{I} \subset (Y \cap S) \setminus (X \cap S)$, and therefore $(X, Y)|_S \notin \mathcal{SEQ}(b_S(P))$, contrary to what is assumed. Now we show that (X', Y') must be a semi-equilibrium model of $P^S(X' \cap S, Y' \cap S)$. First since $(X', Y') \models t_S(P)$ and $(X', Y')|_S \in \mathcal{SEQ}(b_S(P))$, it follows that $(X', Y') \models P^S(X' \cap S, Y' \cap S)$. We prove the h-minimality of (X', Y') with respect to $P^S(X' \cap S, Y' \cap S)$. If by contradiction there exists $(\hat{X}, Y') \models P^S(X' \cap S, Y' \cap S)$ with $\hat{X} \subset X'$, then, by Lemma 16, $(\hat{X}, Y') \models P$ against the hypothesis (1). Finally we prove the gap-minimality of (X', Y') respect to $P^S(X' \cap S, Y' \cap S)$. If by contradiction there exists $(\hat{X}, \hat{Y}) \models P^S(X' \cap S, Y' \cap S)$, that satisfies the *h-minimality property* and, moreover, $\hat{Y} \setminus \hat{X} \subset Y' \setminus X'$, then there exists $(\hat{X}, \hat{Y}) \models P$ (by Lemma 16) that satisfies the *h-minimality property* and $\hat{Y} \setminus \hat{X} \subset Y' \setminus X'$, against the hypothesis (3). In conclusion we have proved that $(X', Y') \in \mathcal{SEQ}(P^S(X' \cap S, Y' \cap S))$ and since hypothesis (2), $Y' \setminus X' \subset Y \setminus X$, it follows that (X, Y) would not be a semi-equilibrium model relative to S. And so we come to a contradiction, so a supposed (X', Y') can not exist. Therefore (X, Y) satisfies the *gap-minimality property* respect to P, so that $(X, Y) \in \mathcal{SEQ}(P)$. □

Proof of Proposition 17. Let $(X, Y) \in \mathcal{SEQ}(P)$ and $(X, Y)|_S \in \mathcal{SEQ}(b_S(P))$. To demonstrate that $(X, Y) \in \mathcal{SEQ}^S(P)$, first we will prove that (X, Y) is a semi-equilibrium model of $P^S(X \cap S, Y \cap S)$. Since $(X, Y) \in \mathcal{SEQ}(P)$, we obtain in particular that $(X, Y) \models t_S(P)$. Now because $X \cap S \subseteq X$ then $(X, Y) \models \{a \mid a \in X \cap S\}$, because $Y \cap S \subseteq Y$ then $(X, Y) \models \{\leftarrow not\, a \mid a \in Y \cap S\}$, and because $(S \setminus (Y \cap S)) \cap Y = \emptyset$ then $(X, Y) \models \{\leftarrow a \mid a \in S \setminus (Y \cap S)\}$. So that (X, Y) is an HT-model of $P^S(X \cap S, Y \cap S)$. So it remains to prove the h-minimality and the gap-minimality of (X, Y) as regards to $P^S(X \cap S, Y \cap S)$. If, by contradiction, we suppose that there exists X' such that $X' \subset X$ and $(X', Y) \models P^S(X \cap S, Y \cap S)$, then, by Lemma 16, $(X', Y) \models P$ and this contradicts the h-minimality of (X, Y) as regards to P. Similarly if, by contradiction, we assume that there exists $(X', Y') \models P^S(X \cap S, Y \cap S)$ that satisfies the h-minimality property and $Y' \setminus X' \subset Y \setminus X$, then by Lemma 16, we obtain that $(X', Y') \models P$ and this contradicts the gap-minimality of (X, Y) as regards to P. Finally, it must be shown that there is no $(\hat{X}, \hat{Y}) \in \mathcal{SEQ}(P^S(I, J))$ with $(I, J) \in \mathcal{SEQ}(b_S(P))$, such that $gap(\hat{X}, \hat{Y}) \subset gap(X, Y)$. In fact if, by contradiction, there exists such a $(\hat{X}, \hat{Y})$, then $(\hat{X}, \hat{Y}) \models P$ (by Lemma 16), $(\hat{X}, \hat{Y})$ satisfies the h-minimality property respect to P and $gap(\hat{X}, \hat{Y}) \subset gap(X, Y)$; i. e. (X, Y) does not satisfy the gap-minimality property respect to P, against the hypothesis. Therefore, in conclusion, $(X, Y) \in \mathcal{SEQ}^S(P)$. □

Proof of Corollary 19. By Theorem 18, $\mathcal{SEQ}^S(P) = \{(X, Y) \in \mathcal{SEQ}(P) \mid (X, Y)|_S \in \mathcal{SEQ}(b_S(P))\}$. As $\mathcal{SEQ}(P) \neq \emptyset$, by Proposition 11 $\mathcal{SEQ}(P) = \mathcal{EQ}(P)$, and $\mathcal{SEQ}(b_S(P)) = \mathcal{EQ}(b_S(P))$; by Proposition 1 and the identity (2) (i.e., by identity (11), it follows that $\mathcal{SEQ}^S(P)\{(X,Y) \in \mathcal{EQ}(P) \mid (X,Y)|_S \in \mathcal{EQ}(b_S(P))\} = \mathcal{EQ}(P)$. As for any positive program P, $\mathcal{EQ}(P) = \{(M,M) \mid M \in MM(P)\}$, the result follows. $\square$

Proof of Proposition 20. If P is constraint-free, then P has some model, hence also $b_S(P)$ ($\subseteq P$) has some model, and thus by Proposition 10, $\mathcal{SEQ}(b_S(P)) \neq \emptyset$. For any $(I,J) \in \mathcal{SEQ}(b_S(P))$, the program $P^S(I,J)$ also has a model, e.g. $J \cup (\Sigma \setminus S)$. Thus, $\mathcal{SEQ}(P^S(I,J)) \neq \emptyset$ by Proposition 10, and hence it follows $\mathcal{SEQ}(P^S) \neq \emptyset$. $\square$

Proof of Theorem 22. We proceed by induction on the length $n \geq 1$ of the splitting sequence. If $n = 1$, then we have $S = (S_1)$ and $S' = \emptyset$, so $\mathcal{SEQ}^S(P) = \mathcal{SEQ}^{S_1}(P)$ and, by Theorem 18, we obtain that $(X,Y) \in \mathcal{SEQ}^S(P)$ if and only if $(X,Y) \in \mathcal{SEQ}(P)$ and $(X,Y)|_S \in \mathcal{SEQ}(b_S(P))$, that is $(X,Y)|_{S_1} \in \mathcal{SEQ}(b_{S_1}(P))$. We assume that the statement is valid for a splitting sequence of length $n-1$ and consider a splitting sequence $S = (S_1, ..., S_n)$ of length n. As usual, we put $S' = (S_2, \ldots, S_n)$. Then $(X,Y) \in \mathcal{SEQ}^S(P)$ if and only if there exists $(I_1,J_1) \in \mathcal{SEQ}(b_{S_1}(P))$ such that $(X,Y) \in \mathcal{SEQ}^{S'}(P_1)$ and (X,Y) is a maximal canonical HT-interpretation. Applying the induction hypothesis to $(X,Y) \in \mathcal{SEQ}^{S'}(P_1)$, we know that $(X,Y) \in \mathcal{SEQ}(P_1)$ and $(X,Y)|_{S_k} \in \mathcal{SEQ}(b_{S_k}(P_{k-1}))$, for $k = 2, \ldots, n$. Now $(X,Y) \in \mathcal{SEQ}(P_1)$ with $(I_1,J_1) \in \mathcal{SEQ}(b_{S_1}(P))$ and (X,Y) is a maximal canonical HT-interpretation is equivalent, by definition, to $(X,Y) \in \mathcal{SEQ}^{S_1}(P)$. So that, by Theorem 18, $(X,Y) \in \mathcal{SEQ}(P)$ and $(X,Y)|_{S_1} \in \mathcal{SEQ}(b_{S_1}(P))$. In conclusion we have demonstrated that $(X,Y) \in \mathcal{SEQ}^S(P)$ if and only if $(X,Y) \in \mathcal{SEQ}(P)$ and $(X,Y)|_{S_k} \in \mathcal{SEQ}(b_{S_k}(P_{k-1}))$, for some P_{k-1}, for $k = 1, \ldots, n$. $\square$

Proof of Corollary 26. This is immediate from Proposition 25 and Corollary 23, given that as well-known $\mathcal{EQ}(P) \neq \emptyset$ for every stratified program. $\square$

B Section 6

Proof of Theorem 28. The proof of uses the following lemmas.

Lemma 59 *Let P be a program and let $S = (S_1, ..., S_n)$ be a splitting sequence of P. We let as above $P_0 = P$ and $P_k = (P_{k-1})^{S_k}(I_k, J_k)$, where $(I_k, J_k) \in \mathcal{SEQ}(b_{S_k}(P_{k-1}))$, with $k = 1, ..., n$. Furthermore, we let $A_k = \{a | a \in I_k\} \cup \{\leftarrow not\, a | a \in J_k\} \cup \{\leftarrow a | a \in S_k \setminus J_k\}$. Then*

$$P_k = P \setminus b_{S_k}(P) \cup A_k$$

for $k = 1, ..., n$.

Proof. We will prove this statement by induction on $k \geq 1$. If $k = 1$, we obtain by definition that

$$P_1 = (P_0)^{S_1}(I_1, J_1) = P_0 \setminus b_{S_1}(P_0) \cup A_1 = P \setminus b_{S_1}(P) \cup A_1.$$

We assume that the statement is true for $k = j-1$ and consider P_j. By definition we have that $P_j = (P_{j-1})^{S_j}(I_j, J_j) = P_{j-1} \setminus b_{S_j}(P_{j-1}) \cup A_j$. Now we can applying the inductive hypothesis on P_{j-1} and we obtain that

$$P_j = (P \setminus b_{S_{j-1}}(P) \cup A_{j-1}) \setminus b_{S_j}(P \setminus b_{S_{j-1}}(P) \cup A_{j-1}) \cup A_j.$$

Since $S_{j-1} \subseteq S_j$, we have that $b_{S_j}(A_{j-1}) = A_{j-1}$, and so

$$\begin{aligned} P_j &= (P \setminus b_{S_{j-1}}(P) \cup A_{j-1}) \setminus (b_{S_j}(P \setminus b_{S_{j-1}}(P)) \cup A_{j-1}) \cup A_j \\ &= (P \setminus b_{S_{j-1}}(P)) \setminus b_{S_j}(P \setminus b_{S_{j-1}}(P)) \cup A_j. \end{aligned}$$

Moreover since $b_{S_{j-1}}(P) \subseteq b_{S_j}(P)$, we can conclude that

$$P_j = (P \setminus b_{S_{j-1}}(P)) \setminus (b_{S_j}(P) \setminus b_{S_{j-1}}(P)) \cup A_j = P \setminus b_{S_j}(P) \cup A_j.$$

$\square$

Lemma 60 *Let P be a program. Let $S = (S_1, ..., S_n)$ be a splitting sequence of P. Let $P_0 = P$ and let P_k and (I_k, J_k) for $k = 1, ..., n-1$ be defined as above. If $(X, Y) \in \mathcal{SEQ}^{(S_{k+1},...,S_n)}(P_k)$, then $I_k \subseteq X$, $J_k \subseteq Y$ and $(S_k \setminus J_k) \cap Y = \emptyset$ for $k = 1, ..., n-1$.*

Proof. Let $(X, Y) \in \mathcal{SEQ}^{(S_{k+1},...,S_n)}(P_k)$. We remember that $P_k = (P_{k-1})^{S_k}(I_k, J_k)$, where $(I_k, J_k) \in \mathcal{SEQ}(b_{S_k}(P_{k-1}))$, for $k = 1, ..., n$ and $P_0 = P$. By Theorem 22 we have that $(X, Y) \in \mathcal{SEQ}(P_k)$ and by Lemma 59,

$$P_k = P \setminus b_{S_k}(P) \cup \{a \mid a \in I_k\} \cup \{\leftarrow not\, a \mid a \in J_k\} \cup \{\leftarrow a \mid a \in S_k \setminus J_k\}.$$

So that $I_k \subseteq X$, $J_k \subseteq Y$ and $(S_k \setminus J_k) \cap Y = \emptyset$. $\square$

Lemma 61 *Let P be a program. Let $S = (S_1, ..., S_n)$ be a splitting sequence of P such that $At(P) = S_n$. If $(X, Y) \in \mathcal{SEQ}^{(S_1,...,S_n)}(P)$, then there exists $(I_k, J_k) \in \mathcal{SEQ}(b_{S_k}(P_{k-1}))$ for $k = 1, ..., n$ such that*

$$(X, Y) = (I_1 \cup (I_2 \setminus I_1) \cup ... \cup (I_n \setminus I_{n-1}), J_1 \cup (J_2 \setminus J_1) \cup ... \cup (J_n \setminus J_{n-1}))$$

with $(I_k \setminus I_{k-1}) \subseteq (J_k \setminus J_{k-1}) \subseteq (S_k \setminus S_{k-1})$, for $k = 2, ..., n$.

Proof. We proceed by induction on the length $n \geq 1$ of the splitting sequence. If $n = 1$, then $At(P) = S_1$ and $(X, Y) \in \mathcal{SEQ}^{S_1}(P)$ imply that there exists some $(I_1, J_1) \in \mathcal{SEQ}(b_{S_1}(P))$ such that $(X, Y) \in \mathcal{SEQ}(P^{S_1}(I_1, J_1))$, but $P^{S_1}(I_1, J_1) = P \setminus b_{S_1}(P) \cup A_1 = A_1$, so that

$$\begin{aligned}\mathcal{SEQ}(P^{S_1}(I_1, J_1)) &= \mathcal{SEQ}(A_1) \\ &= \mathcal{SEQ}(\{a \mid a \in I_1\} \cup \{\leftarrow not\, a \mid a \in J_1\} \cup \{\leftarrow a \mid a \in S_1 \setminus J_1\}) = \{(I_1, J_1)\},\end{aligned}$$

that is $(X, Y) = (I_1, J_1)$.

Now we suppose that the statement is valid for splitting sequence of length $n-1$ and we consider $(X, Y) \in \mathcal{SEQ}^{(S_1,...,S_n)}(P)$. Then there exists $(I_1, J_1) \in \mathcal{SEQ}(b_{S_1}(P))$ such that $(X, Y) \in \mathcal{SEQ}^{(S_2,...,S_n)}(P_1)$ and $At(P_1) = S_n$, so by the inductive hypothesis there exists $(I_k, J_k) \in \mathcal{SEQ}(b_{S_k}(P_{k-1}))$ for $k = 2, ..., n$ such that $(X, Y) = (I_2 \cup (I_3 \setminus I_2) \cup ... \cup (I_n \setminus I_{n-1}), J_2 \cup (J_3 \setminus J_2) \cup ... \cup (J_n \setminus J_{n-1}))$ with $I_k \setminus I_{k-1} \subseteq J_k \setminus J_{k-1} \subseteq S_k \setminus S_{k-1}$, for $k = 3, ..., n$. Moreover, by Lemma 60, $I_1 \subseteq X$, $J_1 \subseteq Y$ and $(S_1 \setminus J_1) \cap Y = \emptyset$ and because $(I_2, J_2) \in \mathcal{SEQ}(b_{S_2}(P_1))$ we obtain that $I_1 \subseteq I_2$, $J_1 \subseteq J_2$ and $(S_1 \setminus J_1) \cap J_2 = \emptyset$. These last results imply that $I_2 \setminus I_1 \subseteq J_2 \setminus J_1 \subseteq S_2 \setminus S_1$. $\square$

Lemma 62 *Let P be a program and let $S \subseteq At(P)$ such that both S and $At(P) \setminus S$ are splitting sets of P. If for each constraint r, $At(r) \subseteq S$ or $At(r) \subseteq At(P) \setminus S$, then*

$$\mathcal{SEQ}(P) = \mathcal{SEQ}^S(P).$$

Proof. The inclusion $\mathcal{SEQ}^S(P) \subseteq \mathcal{SEQ}(P)$ follows from Proposition 15. So we have just to prove that $\mathcal{SEQ}(P) \subseteq \mathcal{SEQ}^S(P)$.

Let $(X, Y) \in \mathcal{SEQ}(P)$. We want to prove that $(X \cap S, Y \cap S) \in \mathcal{SEQ}(b_S(P))$.

We know that $(X, Y) \models b_S(P)$. As S is a splitting set of P, $At(b_S(P)) \subseteq S$ and so $(X \cap S, Y \cap S) \models b_S(P)$.

Now we prove the claim showing that $(X \cap S, Y \cap S)$ satisfies h-minimality and gap-minimality.

If by contradiction some $I \subset X \cap S$ exists such that $(I, Y \cap S) \models b_S(P)$, then $X' = I \cup (X \cap (At(P) \setminus S)) \subset X$ and $(X', Y) \models P$ which contradicts the h-minimality of (X, Y).

Similarly, if by contradiction, some $(I, J) \models b_S(P)$ exists such that (I, J) satisfies h-minimality and $J \setminus I \subset (Y \cap S) \setminus (X \cap S)$, then having set $X' = I \cup (X \cap (At(P) \setminus S))$ and $Y' = J \cup (Y \cap (At(P) \setminus S))$, we obtain that $(X', Y') \models P$, satisfies the h-minimality and $Y' \setminus X' \subset Y \setminus X$ in contradiction to the gap-minimality of (X, Y).

Therefore $(X \cap S, Y \cap S) \in \mathcal{SEQ}(b_S(P))$. Then, by Theorem 18, $(X, Y) \in \mathcal{SEQ}^S(P)$; hence $\mathcal{SEQ}(P) = \mathcal{SEQ}^S(P)$. □

For any sets $\mathcal{M}$ and $\mathcal{M}'$ of HT-models, define their product $\mathcal{M} \times \mathcal{M}'$ as the set of HT models given by $\mathcal{M} \times \mathcal{M}' = \{(X \cup X', Y \cup Y') \mid (X, Y) \in \mathcal{M}, (X', Y') \in \mathcal{M}'\}$.

Lemma 63 *Let P be a program in which each constraint r fulfills either $At(r) \subseteq S$ or $At(r) \subseteq At(P) \setminus S$. If both S and $At(P) \setminus S$ are splitting sets of P, then*

$$\mathcal{SEQ}^S(P) = \mathcal{SEQ}(b_S(P)) \times \mathcal{SEQ}(t_S(P)).$$

Proof. If $\mathcal{SEQ}(b_S(P)) = \emptyset$, then

$$\mathcal{SEQ}(b_S(P)) \times \mathcal{SEQ}(t_S(P)) = \emptyset$$

and

$$\mathcal{SEQ}^S(P) = mc\Big(\bigcup_{(I,J) \in \mathcal{SEQ}(b_S(P))} \mathcal{SEQ}(P^S(I, J)) \Big) = \emptyset.$$

Let $(I, J) \in \mathcal{SEQ}(b_S(P))$. For each rule $r \in b_S(P)$, no atom of r is in some rule of $t_S(P)$ and vice versa, that is $At(b_S(P)) \cap At(t_S(P)) = \emptyset$. Hence

$$\begin{aligned}
&\mathcal{SEQ}(t_S(P) \cup \{a \mid a \in I\} \cup \{\leftarrow not\, a \mid a \in J\} \cup \{\leftarrow a \mid a \in S \setminus J\}) \\
&\qquad = \{(X, Y) \mid X = X_1 \cup I, Y = Y_1 \cup J, (X_1, Y_1) \in \mathcal{SEQ}(t_S(P))\} \\
&\qquad = \mathcal{SEQ}(t_S(P)) \times \{(I, J)\}.
\end{aligned}$$

Then

$$\begin{aligned}
\mathcal{SEQ}^S(P) &= mc\Big(\bigcup_{(I,J) \in \mathcal{SEQ}(b_S(P))} \mathcal{SEQ}(t_S(P)) \times \{(I, J)\} \Big) \\
&= mc\, (\mathcal{SEQ}(b_S(P)) \times \mathcal{SEQ}(t_S(P))) \\
&= \mathcal{SEQ}(b_S(P)) \times \mathcal{SEQ}(t_S(P)).
\end{aligned}$$

□

Proof of Proposition 30. Follows immediately from Lemmas 62 and 63. □

Lemma 64 *Let P be a program without cross-constraints. Let $(C_1, ..., C_n)$ and $(C_1, ..., C_{i-1}, C_{i+1}, C_i, C_{i+2}, ..., C_n)$ be two topological orderings of $\mathcal{SCC}(P)$. If we put $S_k = C_1 \cup ... \cup C_k$ for $k = 1, ..., n$ and $S'_i = S_{i-1} \cup C_{i+1}$ then*

$$b_{S'_i}(P \setminus b_{S_{i-1}}(P)) = b_{S_{i+1}}(P \setminus b_{S_i}(P)).$$

Proof . In general we know that $b_{S_i}(P) \setminus b_{S_{i-1}}(P) = b_{S_i}(P \setminus b_{S_{i-1}}(P))$. Hence it is sufficient to prove that $b_{S_{i+1}}(P) \setminus b_{S_i}(P) = b_{S_i'}(P) \setminus b_{S_{i-1}}(P)$.

Let $r \in P$, and assume that $r \in b_{S_{i+1}}(P)$ and $r \notin b_{S_i}(P)$. If r is a constraint, then $At(r) \cap C_{i+1} \neq \emptyset$. As P has no cross-constraints, it follows that $At(r) \cap C_i = \emptyset$. If r is not a constraint, then there exists some $a \in H(r)$ such that $a \in C_{i+1}$. But because there is no edge between C_i and C_{i+1}, we obtain again that $At(r) \cap C_i = \emptyset$. Therefore $r \in b_{S_{i-1} \cup C_{i+1}}(P)$ and clearly $r \notin b_{S_{i-1}}(P)$.

Conversely, assume that $r \in b_{S_{i-1} \cup C_{i+1}}(P)$ and $r \notin b_{S_{i-1}}(P)$. Then $r \in b_{S_{i-1} \cup C_{i+1}}(P) \subseteq b_{S_{i+1}}(P)$. Moreover $r \in b_{S_{i-1} \cup C_{i+1}}(P)$ implies that $At(r) \cap C_i = \emptyset$, and because $r \notin b_{S_{i-1}}(P)$, it follows that $r \notin b_{S_i}(P)$. □

Lemma 65 *Let P be a program without cross-constraints. Let $(C_1, ...C_n)$ and $(C_1, ..., C_{i-1}, C_{i+1}, C_i, C_{i+2}, ..., C_n)$ be two topological orderings of $\mathcal{SCC}(P)$. If we put $S_k = C_1 \cup ... \cup C_k$ for $k = 1, ..., n$ and $S_i' = S_{i-1} \cup C_{i+1}$ then*

$$\mathcal{SEQ}^{(S_1,...,S_{i-1},S_i,S_{i+1},S_{i+2},...,S_n)}(P) = \mathcal{SEQ}^{(S_1,...,S_{i-1},S_i',S_{i+1},S_{i+2},...,S_n)}(P).$$

Proof . Let $(X, Y) \in \mathcal{SEQ}^{(S_1,...,S_{i-1},S_i,S_{i+1},S_{i+2},...,S_n)}(P)$. Since $At(P) = C_1 \cup ... \cup C_n = S_n$, by Lemma 61 we obtain that

$$(X, Y) = (I_1 \cup (I_2 \setminus I_1) \cup ... \cup (I_n \setminus I_{n-1}), J_1 \cup (J_2 \setminus J_1) \cup ... \cup (J_n \setminus J_{n-1}))$$

where $(I_k, J_k) \in \mathcal{SEQ}(b_{S_k}(P_{k-1}))$ for $k = 1, ..., n$, with

$$(I_k \setminus I_{k-1}) \subseteq (J_k \setminus J_{k-1}) \subseteq (S_k \setminus S_{k-1}) = C_k$$

for $k = 2, ..., n$.

First we show that

$$(X, Y)|_{S_i'} \in \mathcal{SEQ}(b_{S_i'}(P_{i-1})).$$

We know that

$$(X, Y)|_{S_i'} = (X, Y)|_{S_{i-1} \cup C_{j+1}} = (I_{i-1} \cup (I_{i+1} \setminus I_i), J_{i-1} \cup (J_{i+1} \setminus J_i)).$$

Moreover, using Lemma 64, we obtain

$$\begin{aligned} b_{S_i'}(P_{i-1}) &= b_{S_{i-1} \cup C_{j+1}}(P_{i-1}) = b_{S_{i-1} \cup C_{i+1}}(P \setminus b_{S_{i-1}}(P) \cup A_{i-1}) \\ &= b_{S_{i-1} \cup C_{i+1}}(P \setminus b_{S_{i-1}}(P)) \cup A_{i-1} \\ &= b_{S_{i+1}}(P \setminus b_{S_i}(P)) \cup A_{i-1}. \end{aligned}$$

And we note that

$$\begin{aligned} b_{S_{i+1}}(P_i) &= b_{S_{i+1}}(P \setminus b_{S_i}(P) \cup A_i) \\ &= b_{S_{i+1}}(P \setminus b_{S_i}(P)) \cup A_{i-1} \cup (A_i \setminus A_{i-1}). \end{aligned}$$

Now in the program $b_{S_{i+1}}(P_i)$ both $S_{i-1} \cup C_{i+1}$ and C_i are splitting sets and in particular

$$b_{S_{i-1} \cup C_{i+1}}(b_{S_{i+1}}(P_i)) = b_{S_{i+1}}(P \setminus b_{S_i}(P)) \cup A_{i-1}$$

and

$$b_{C_i}(b_{S_{i+1}}(P_i)) = A_i \setminus A_{i-1}.$$

Therefore by Proposition 30 we obtain that

$$\mathcal{SEQ}(b_{S_{i+1}}(P_i)) = \mathcal{SEQ}(b_{S_{i+1}}(P \setminus b_{S_i}(P)) \cup A_{i-1}) \times \mathcal{SEQ}(A_i \setminus A_{i-1}).$$

So we have that

$$\mathcal{SEQ}(b_{S_{i+1}}(P_i)) = \mathcal{SEQ}(b_{S_{i-1} \cup C_{j+1}}(P_{i-1})) \times \{(I_i \setminus I_{i-1}, J_i \setminus J_{i-1})\},$$

and since

$$(X,Y)|_{S_{i+1}} = (I_{i-1} \cup (I_i \setminus I_{i-1}) \cup (I_{i+1} \setminus I_i), J_{i-1} \cup (J_i \setminus J_{i-1}) \cup (J_{i+1} \setminus J_i)) \in \mathcal{SEQ}(b_{S_{i+1}}(P_i)),$$

it follows

$$(I_{i-1} \cup (I_{i+1} \setminus I_i), J_{i-1} \cup (J_{i+1} \setminus J_i)) \in \mathcal{SEQ}(b_{S_{i-1} \cup C_{j+1}}(P_{i-1})).$$

By Theorem 22, we know that if $(X,Y) \in \mathcal{SEQ}^{(S_1,\dots,S_{i-1},S_i,S_{i+1},S_{i+2},\dots,S_n)}(P)$, then

$$\begin{aligned}
&(X,Y) \in \mathcal{SEQ}(P), (X,Y)|_{S_1} \in \mathcal{SEQ}(b_{S_1}(P)), \dots, (X,Y)|_{S_{i-1}} \in \mathcal{SEQ}(b_{S_{i-1}}(P_{i-2})),\\
&(X,Y)|_{S_i} \in \mathcal{SEQ}(b_{S_i}(P_{i-1})), (X,Y)|_{S_{i+1}} \in \mathcal{SEQ}(b_{S_{i+1}}(P_i)),\\
&(X,Y)|_{S_{i+2}} \in \mathcal{SEQ}(b_{S_{i+2}}(P_{i+1})), \dots (X,Y)|_{S_n} \in \mathcal{SEQ}(b_{S_n}(P_{n-1})),
\end{aligned}$$

We want to prove that $(X,Y) \in \mathcal{SEQ}^{(S_1,\dots,S_{i-1},S'_i,S_{i+1},S_{i+2},\dots,S_n)}(P)$. That is, by Theorem 22:

$$\begin{aligned}
&(X,Y) \in \mathcal{SEQ}(P), (X,Y)|_{S_1} \in \mathcal{SEQ}(b_{S_1}(P)), \dots (X,Y)|_{S_{i-1}} \in \mathcal{SEQ}(b_{S_{i-1}}(P_{i-2})),\\
&(X,Y)|_{S'_i} \in \mathcal{SEQ}(b_{S'_i}(P_{i-1})), (X,Y)|_{S_{i+1}} \in \mathcal{SEQ}(b_{S_{i+1}}(P \setminus b_{S'_i}(P) \cup A_{i-1} \cup (A_{i+1} \setminus A_i))),\\
&(X,Y)|_{S_{i+2}} \in \mathcal{SEQ}(b_{S_{i+2}}(P_{i+1})), \dots, (X,Y)|_{S_n} \in \mathcal{SEQ}(b_{S_n}(P_{n-1})),
\end{aligned}$$

So it remains to prove that

$$(X,Y)|_{S_{i+1}} \in \mathcal{SEQ}(b_{S_{i+1}}(P \setminus b_{S'_i}(P_{i-1}) \cup A_{i-1} \cup (A_{i+1} \setminus A_i))).$$

We know that

$$\begin{aligned}
&b_{S_{i+1}}(P \setminus b_{S'_i}(P) \cup A_{i-1} \cup (A_{i+1} \setminus A_i))\\
&\qquad = b_{S_{i+1}}(P \setminus b_{S_{i-1} \cup C_{i+1}}(P)) \cup A_{i-1} \cup (A_{i+1} \setminus A_i)\\
&\qquad = b_{S_i}(P \setminus b_{S_{i-1}}(P)) \cup A_{i-1} \cup (A_{i+1} \setminus A_i)\\
&\qquad = b_{S_i}(P \setminus b_{S_{i-1}}(P)) \cup A_{i-1}) \cup (A_{i+1} \setminus A_i)\\
&\qquad = b_{S_i}(P_{i-1}) \cup (A_{i+1} \setminus A_i).
\end{aligned}$$

Now in this program both S_i and C_{i+1} are splitting sets and in particular

$$b_{S_i}(b_{S_i}(P_{i-1}) \cup (A_{i+1} \setminus A_i)) = b_{S_i}(P_{i-1})$$

and

$$b_{C_{i+1}}(b_{S_i}(P_{i-1}) \cup (A_{i+1} \setminus A_i)) = A_{i+1} \setminus A_i.$$

Therefore by Proposition 30 we obtain that

$$\begin{aligned}
&\mathcal{SEQ}(b_{S_{i+1}}(P \setminus b_{S'_i}(P_{i-1}) \cup A_{i-1} \cup (A_{i+1} \setminus A_i)))\\
&\qquad = \mathcal{SEQ}(b_{S_i}(P_{i-1})) \times \mathcal{SEQ}(A_{i+1} \setminus A_i)\\
&\qquad = \mathcal{SEQ}(b_{S_i}(P_{i-1})) \times \{(I_{i+1} \setminus I_i, J_{i+1} \setminus J_i)\}.
\end{aligned}$$

Now since $(I_i, J_i) \in \mathcal{SEQ}(b_{S_i}(P_{i-1}))$, we obtain that

$$(I_{i+1}, J_{i+1}) = (X, Y)|_{S_{i+1}} \in \mathcal{SEQ}(b_{S_{i+1}}(P \setminus b_{S'_i}(P_{i-1}) \cup A_{i-1} \cup (A_{i+1} \setminus A_i))).$$

In conclusion, we have proved that

$$\mathcal{SEQ}^{(S_1,\ldots,S_{i-1},S_i,S_{i+1},S_{i+2},\ldots,S_n)}(P) \subseteq \mathcal{SEQ}^{(S_1,\ldots,S_{i-1},S'_i,S_{i+1},S_{i+2},\ldots,S_n)}(P).$$

The proof of the reverse inclusion is similar. □

Theorem 28 is then proven as follows. Let $(C_{i_1}, \ldots, C_{i_n}) \in \mathcal{O}(SG(P))$. We define a function

$$t_{(C_{i_1},\ldots,C_{i_n})} : \mathcal{O}(SG(P)) \longrightarrow \mathcal{O}(SG(P)).$$

Let $(C_{j_1}, \ldots, C_{j_n}) \in \mathcal{O}(SG(P))$. If $C_{i_r} = C_{j_r}$ for $r = 1, \ldots, l$, $C_{i_{l+1}} \neq C_{j_{l+1}}$ and there exists $k+1 > l+1$ such that $C_{j_{k+1}} = C_{i_{l+1}}$, then

$$\begin{aligned} t_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1}, \ldots, C_{j_n}) &= t_{(C_{i_1},\ldots,C_{i_n})}(C_{i_1}, \ldots, C_{i_l}, C_{j_{l+1}}, \ldots, C_{j_{k-1}}, C_{j_k}, C_{i_{l+1}}, C_{j_{k+2}}, \ldots, C_{j_n}) \\ &= (C_{i_1}, \ldots, C_{i_l}, C_{j_{l+1}}, \ldots, C_{j_{k-1}}, C_{i_{l+1}}, C_{j_k}, C_{j_{k+2}}, \ldots, C_{j_n}), \end{aligned}$$

else $t_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1}, \ldots, C_{j_n}) = (C_{j_1}, \ldots, C_{j_n}) = (C_{i_1}, \ldots, C_{i_n})$. This function is well-defined because there are no edges from C_{i_m} to $C_{i_{l+1}}$ for $m = l+2, \ldots, n$. That is there are no edges from C_{j_k} to $C_{i_{l+1}}$, therefore $(C_{i_1}, \ldots, C_{i_l}, C_{j_{l+1}}, \ldots, C_{j_{k-1}}, C_{i_{l+1}}, C_{j_k}, C_{j_{k+2}}, \ldots, C_{j_n})$ is another topological ordering of $\mathcal{SCC}(P)$. Moreover for each $(C_{j_1}, \ldots, C_{j_n}) \in \mathcal{O}(SG(P))$, there exists some finite N such that

$$t^N_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1}, \ldots, C_{j_n}) = (C_{i_1}, \ldots, C_{i_n}).$$

During the proof, in order not to introduce additional symbols, we shall denote the splitting sequence S^i with $(C_{i_1}, \ldots, C_{i_n})$ and S^j with $(C_{j_1}, \ldots, C_{j_n})$.

Let N be such that $t^N_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1}, \ldots, C_{j_n}) = (C_{i_1}, \ldots, C_{i_n})$. We will prove the theorem using induction on N. If $N = 1$, then $t_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1}, \ldots, C_{j_n}) = (C_{i_1}, \ldots, C_{i_n})$, i.e. $(C_{j_1}, \ldots, C_{j_n})$ and $(C_{i_1}, \ldots, C_{i_n})$ differ at most by the exchange of two consecutive strongly connected components. Then, by Lemma 65, $\mathcal{SEQ}^{(C_{i_1},\ldots,C_{i_n})}(P) = \mathcal{SEQ}^{(C_{j_1},\ldots,C_{j_n})}(P)$. Now we suppose that the theorem is valid for topological orderings $(C_{s_1}, \ldots, C_{s_n})$ such that $t^{N-1}_{(C_{i_1},\ldots,C_{i_n})}(C_{s_1}, \ldots, C_{s_n}) = (C_{i_1}, \ldots, C_{i_n})$. We consider $(C_{j_1}, \ldots, C_{j_n})$ such that $t^N_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1}, \ldots, C_{j_n}) = (C_{i_1}, \ldots, C_{i_n})$. By definition of the function $t_{(C_{i_1},\ldots,C_{i_n})}$, we know that

$$t_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1}, \ldots, C_{j_n}) = (C_{i_1}, \ldots, C_{i_l}, C_{j_{l+1}}, \ldots, C_{j_{k-1}}, C_{i_{l+1}}, C_{j_k}, C_{j_{k+2}}, \ldots, C_{j_n}).$$

Therefore, by Lemma 65, we have that

$$\mathcal{SEQ}^{(C_{j_1},\ldots,C_{j_n})}(P) = \mathcal{SEQ}^{t_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1},\ldots,C_{j_n})}(P).$$

But now $t^{N-1}_{(C_{i_1},\ldots,C_{i_n})}(t_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1}, \ldots, C_{j_n})) = (C_{i_1}, \ldots, C_{i_n})$ such that, by the induction hypothesis, we obtain that

$$\mathcal{SEQ}^{t_{(C_{i_1},\ldots,C_{i_n})}(C_{j_1},\ldots,C_{j_n})}(P) = \mathcal{SEQ}^{(C_{i_1},\ldots,C_{i_n})}(P).$$

In conclusion, we have proved that $\mathcal{SEQ}^{(C_{j_1},\ldots,C_{j_n})}(P) = \mathcal{SEQ}^{(C_{i_1},\ldots,C_{i_n})}(P)$. □

Proof of Theorem 32. First we observe that for every splitting set S of a program P, we can always write S as the union of some SCCs of P. More in detail, if $\mathcal{SCC}(P) = \{C_1, \ldots, C_n\}$, then we can assume that

$S = C_1 \cup ... \cup C_k$, where C_1, ..., C_k are consecutive in some topological ordering $(C_1, ..., C_k, ..., C_n)$ of $\mathcal{SCC}(P)$.

By definition, we have that

$$M^{\mathcal{SCC}}(P) = \mathcal{SEQ}^{(S_1,...,S_n)}(P),$$

where $S_j = \cup_{i=1}^{j} C_i$, for $1 \le j \le n$; note that $S = S_k$.

If we explicate the computation of $\mathcal{SEQ}^{(S_1,...,S_n)}(P)$ up to k-th union, we obtain

$$M^{\mathcal{SCC}}(P) = mc\Big(\bigcup_{M_k \in \mathcal{M}_k} \mathcal{SEQ}^{(S_{k+1},...,S_n)}(P \setminus b_{S_k}(P) \cup M_k)\Big) \tag{19}$$

where $\mathcal{M}_k$ is last in a sequence $\mathcal{M}_i$, $1 \le i \le k$ of sets $\mathcal{M}_i$ of HT-models $M_i = (I_i, J_i)$, over S_i, such that $\mathcal{M}_1 = \mathcal{SEQ}(b_{S_1}(P))$ and $\mathcal{M}_{i+1} = mc(\bigcup_{M_i \in \mathcal{M}_i} \mathcal{SEQ}((b_{S_{i+1}}(P) \setminus b_{S_i}(P)) \cup M_i)$, $1 \le i < k$, where in abuse of notation "$\cup M_i$" stands for $\cup\{a \mid a \in I_i\} \cup \{\leftarrow \mathit{not}\, a \mid a \in J_i\} \cup \{\leftarrow a \mid a \in S_i \setminus J_i\}$. Note that all $M_i \neq M'_i \in \mathcal{M}_i$ have incomparable gaps, i.e., $gap(M_i) \not\subseteq gap(M'_i)$.

Now we show that the set $\mathcal{M}_k$ coincides with $M^{\mathcal{SCC}}(b_S(P))$. Indeed, by definition, we know that

$$M^{\mathcal{SCC}}(b_S(P)) = \mathcal{SEQ}^{(S_1,...,S_k)}(b_S(P)).$$

Therefore, applying k-times the definition of semi-equilibrium models relative to a splitting sequence, we obtain

$$\mathcal{SEQ}^{(S_1,...,S_k)}(b_S(P)) = mc\Big(\bigcup_{M'_k \in \mathcal{M}'_k} \mathcal{SEQ}(b_S(P) \setminus b_{S_k}(P) \cup M'_k)\Big) \tag{20}$$

where $\mathcal{M}'_k$ and M'_k are analogously defined to $\mathcal{M}_k$ and M_k using $b_S(P)$ instead of P, i.e., $\mathcal{M}'_1 = \mathcal{SEQ}(b_{S_1}(b_S(P)))$ and $\mathcal{M}'_{i+1} = mc(\bigcup_{M'_i \in \mathcal{M}'_i} \mathcal{SEQ}((b_{S_{i+1}}(b_S(P)) \setminus b_{S_i}(b_S(P))) \cup M'_i)$, $1 \le i < k$. As $b_{S_i}(b_S(P)) = b_{S_i}(P)$ for each i, the $\mathcal{M}_i$ and the $\mathcal{M}'_i$ coincide; as $b_S(P) = b_{S_k}(P)$, we thus obtain from (20)

$$\mathcal{SEQ}^{(S_1,...,S_k)}(b_S(P)) = mc\Big(\bigcup_{M_k \in \mathcal{M}_k} \mathcal{SEQ}(M_k)\Big) = \bigcup_{M_k \in \mathcal{M}_k} M_k = \mathcal{M}_k;$$

here we use that the M_k have incomparable gaps. This proves the claim that $\mathcal{M}_k = M^{\mathcal{SCC}}(b_S(P))$.

To prove the result, it remains by (19) to show that for each $M_k \in \mathcal{M}_k$,

$$\mathcal{SEQ}^{(S_{k+1},...,S_n)}(P \setminus b_S(P) \cup M_k) = M^{\mathcal{SCC}}(P \setminus b_S(P) \cup M_k).$$

We observe that the programs $Q = P \setminus b_S(P) \cup M_k$ and P have the same atoms but in general different SCCs. However it is easy to see that every atom in $a \in S_k$ induces a $\mathcal{SCC}$ $C_a = \{a\}$ w.r.t. Q, and thus $S_k = C_{a_1} \cup \cdots \cup C_{a_\ell}$ where $S_k = \{a_1, \ldots, a_\ell\}$. Furthermore, Q contains only constraints r such that either $At(Q) \subseteq S_k$ or $At(Q) \cap S_k = \emptyset$. As $(C_{a_1}, ..., C_{a_\ell}, C_{k+1}, ...C_n)$ is a topological ordering of $\mathcal{SCC}(Q)$, we obtain

$$M^{\mathcal{SCC}}(Q) = \mathcal{SEQ}^{(S_{a_1},...,S_{a_\ell},S_{k+1},...,S_n)}(Q) = \mathcal{SEQ}^{(S_{k+1},...,S_n)}(Q).$$

where $S_{a_i} = \bigcup_{j \le i} C_{a_j}$. The last equality can be seen by noting that, for each $j = 1, ..., \ell$, we have $\mathcal{SEQ}(b_{S_{a_j}}(Q)) = \{M_k|_{S_{a_j}}\}$ (where $M_k|_{S_{a_j}}$ denotes the restriction of M_k to S_{a_j}) and thus for each $(X_j, Y_j) \in \mathcal{SEQ}(b_{S_{a_j}}(Q))$,

$$Q \setminus b_{S_{a_j}}(Q) \cup (X_j, Y_j) = (Q \setminus M_k|_{S_{a_j}}) \cup (X_j, Y_j) = Q.$$

In conclusion, by replacing in Equation (19) the $\mathcal{SEQ}$-model $M_k \in \mathcal{M}_k$ with $(I, J) \in M^{\mathcal{SCC}}(b_{S_k}(P))$ and $\mathcal{SEQ}^{(S_{k+1},\dots,S_n)}(P \setminus b_{S_k}(P) \cup M_k)$ with $M^{\mathcal{SCC}}(P \setminus b_{S_k}(P) \cup (I, J))$ and reminding that $S_k = S$ and $P^S(I, J) = P \setminus b_{S_k}(P) \cup (I, J)$, we have proved that

$$M^{\mathcal{SCC}}(P) = mc\Big(\bigcup_{(I,J)\in M^{\mathcal{SCC}}(b_S(P))} M^{\mathcal{SCC}}(P \setminus b_S(P) \cup (I, J))\Big).$$

□

Proof of Theorem 33. For the proof of Theorem 33, we use the following lemmas.

Lemma 66 *Let P be a program. Let $MJC(P) = \{J_1, ..., J_m\}$. Let $(J_1, ..., J_{i-1}, J_i, J_{i+1}, J_{i+2}, ..., J_m)$ and $(J_1, ..., J_{i-1}, J_{i+1}, J_i, J_{i+2}, ..., J_m)$ be two topological orderings. If we put $S_k = J_1 \cup ... \cup J_k$ for $k = 1, ..., m$ and $S'_i = S_{i-1} \cup J_{i+1}$ then*

$$b_{S'_i}(P \setminus b_{S_{i-1}}(P)) = b_{S_{i+1}}(P \setminus b_{S_i}(P)).$$

Proof. In general we know that $b_{S_i}(P) \setminus b_{S_{i-1}}(P) = b_{S_i}(P \setminus b_{S_{i-1}}(P))$. So that is sufficient to prove that $b_{S_{i+1}}(P) \setminus b_{S_i}(P) = b_{S'_i}(P) \setminus b_{S_{i-1}}(P)$.

Let $r \in P$. We assume that $r \in b_{S_{i+1}}(P)$ and $r \notin b_{S_i}(P)$.

If r is not a constraint, then there exists some $a \in H(r)$ such that $a \in J_{i+1}$. But because there is no edge among J_i and J_{i+1}, we obtain that $At(r) \cap J_i = \emptyset$. Therefore $r \in b_{S_{i-1}\cup J_{i+1}}(P)$ and clearly $r \notin b_{S_{i-1}}(P)$.

If r is a constraint then there exists $a \in (B^+(r) \cup B^-(r)) \cap J_{i+1}$. If, by contradiction, we assume that there exists some $b \in (B^+(r) \cup B^-(r)) \cap J_i$, then there exist $K_i, K_{i+1} \in \mathcal{SCC}(P)$ such that $K_{i+1} \subseteq J_{i+1}$ and $K_i \subseteq J_i$ with $r \in C_{K_i,K_{i+1}}(P)$. But because there is no edge among J_i and J_{i+1}, then there exists a topological ordering of strongly connected components of P that are in J_i and J_{i+1}, such that K_i precedes K_{i+1}. So there exists $(C_1, ..., C_n) \in \mathcal{O}(P)$ in which $C_l = K_i$ and $C_{l+1} = K_{i+1}$ for some $l = 1, ..., n-1$ and moreover $At(r) \subseteq C_1 \cup ... \cup C_{l+1}$. Then (K_i, K_{i+1}) is a joinable pair and therefore K_i, K_{i+1} are joinable components, but this contradicts the maximality of J_i and J_{i+1}. So that $(B^+(r) \cup B^-(r)) \cap J_i = \emptyset$. That is $r \in b_{S_{i-1}\cup J_{i+1}}(P)$ and clearly $r \notin b_{S_{i-1}}(P)$.

Conversely we assume that $r \in b_{S_{i-1}\cup C_{i+1}}(P)$ and $r \notin b_{S_{i-1}}(P)$. Then $r \in b_{S_{i-1}\cup C_{i+1}}(P) \subseteq b_{S_{i+1}}(P)$. Moreover $r \in b_{S_{i-1}\cup C_{i+1}}(P)$ implies that $At(r) \cap C_i = \emptyset$, and because $r \notin b_{S_{i-1}}(P)$, then $r \notin b_{S_i}(P)$. □

Lemma 67 *Let P be a program. Let $\mathcal{MJC}(P) = \{J_1, ..., J_m\}$. Let $(J_1, ..., J_{i-1}, J_i, J_{i+1}, J_{i+2}, ..., J_m)$ and $(J_1, ..., J_{i-1}, J_{i+1}, J_i, J_{i+2}, ..., J_m)$ be two topological orderings. If we put $S_k = J_1 \cup ... \cup J_k$ for $k = 1, ..., m$ and $S'_i = S_{i-1} \cup J_{i+1}$ then*

$$\mathcal{SEQ}^{(S_1,\dots,S_{i-1},S_i,S_{i+1},S_{i+2},\dots,S_m)}(P) = \mathcal{SEQ}^{(S_1,\dots,S_{i-1},S'_i,S_{i+1},S_{i+2},\dots,S_m)}(P).$$

Proof. The proof is *mutatis mutandis* the same as that of Lemma 65, and one identifies $b_{S'_i}(P \setminus b_{S_{i-1}}(P))$ and $b_{S_{i+1}}(P \setminus b_{S_i}(P))$ using Lemma 66 instead of Lemma 62. □

The proof of Theorem 33 the same as that of Theorem 28, but uses Lemma 67 instead of Lemma 65. □

Proof of Theorem 35. The proof is very similar to the one of Theorem 32: under the premise, the MJCs which form S respectively the SCCs constituting them are in the initial segment of some topologic ordering, like the SCCs in the proof of Theorem 32. Thus the same line of argumentation applies. □

C Section 7

C.1 Hardness results for semi-equilibrium semantics

Several results about Problem MCH and INF for disjunctive program under semi-equilibrium model semantics ($S = (At(P))$) can be shown using a reduction from deciding the validity of a quantified Boolean formula (QBF) of the form

$$\Phi = \exists Z \forall Y \exists X. E(X, Y, Z)$$

where $X = \{x_1 \dots x_r\}$, $Y = \{y_1 \dots y_s\}$ and $Z = \{z_1 \dots z_t\}$ We may assume without loss of generality that $E(X, Y, Z) = \bigwedge_{i=1}^{m} (l_{i1} \vee l_{i2} \vee l_{i3})$ where each l_{ij} is a literal over $X \cup Y \cup Z$ (i.e., 3-CNF form). We define a program P_0 with the following rules:

1. $p \leftarrow l_{i1}^*, l_{i2}^*, l_{i3}^*$, where $l_{ij}^* = \begin{cases} \overline{v}, & \text{if } l_{ij} = v \\ v, & \text{if } l_{ij} = \neg v \end{cases}$ and $v \in X \cup Y \cup Z$;

2. $x \leftarrow p$ and $\overline{x} \leftarrow p$ for each $x \in X$;

3. $y \vee \overline{y}$ for each $y \in Y$;

4. $x \vee \overline{x}$ for each $x \in X$.

We assume for the moment that Z is void (i.e., $Z = \emptyset$); then one can show the following property [14]:

$$\text{Some } M \in MM(P_0) \text{ exists s.t. } p \in M \text{ iff } \neg(\forall Y \exists X. E(X, Y)) \text{ is true.} \tag{21}$$

As P_0 is positive, $\mathcal{SEQ}(P_0) = \{(M, M) \mid M \in MM(P_0)\}$; it follows from this that brave reasoning from the $\mathcal{SEQ}$-models of a positive disjunctive program, i.e., deciding $P \models_{\mathcal{SEQ}}^{b,\mathbf{t}} p$, is Σ_2^P-hard; furthermore, cautious reasoning $P \models_{\mathcal{SEQ}}^{c,\mathbf{f}} p$, is Π_2^P-hard.

Now we construct a new program P_1 that is obtained by adding a fresh atom q in each rule head of P_0 and the following rules:

5. $p' \leftarrow p$ and

6. $\leftarrow not\, p'$.

It is easy to see that $\{q\}$ is a minimal model of P_1. Now the following property holds:

$$(\{q\}\ , \{q, p'\}) \in \mathcal{SEQ}(P_1) \text{ if and only if } \forall Y \exists X. E(X, Y) \text{ is true.} \tag{22}$$

Clearly, the program is stratified; consequently, Problem MCH under $\mathcal{SEQ}$-semantics is Π_2^P-hard for disjunctive and stratified disjunctive programs, which proves the hardness part of item (ii) in Theorem 36.

Eventually, we consider the target case in which $Z \neq \emptyset$. We construct a final program P given by the union of P_1 with the following rules:

7. $z \vee \overline{z}$ for each $z \in Z$ and

8. $\leftarrow z, not\, b_z$ and $\leftarrow \overline{z}, not\, b_{\overline{z}}$ for each $z \in Z$ where b_z and $b_{\overline{z}}$ are fresh atoms.

Intuitively, the effect of these rules is that in each $\mathcal{SEQ}$-model (I, J), either b_z or $b_{\overline{z}}$ but not both must be contained in $gap(I, J)$, for each $z \in Z$; this serves to emulate quantification over Z. For each $Z' \subseteq Z$, the HT-interpretation $(I_Z, J_Z) = (\{b_z \mid z \in Z'\} \cup \{q\}, \{q, p'\} \cup \{b_{\overline{z}} \mid z \in Z \setminus Z'\})$ is a HT-model of P; it will be a $\mathcal{SEQ}$-model of P precisely if $\forall Y \exists X.E(X, Y, Z = Z')$ is true. Formally, one can show:

$$\text{Some } (I, J) \in \mathcal{SEQ}(P) \text{ exists s.t. } p' \in J \setminus I \text{ iff } \Phi = \exists Z \forall Y \exists X.E(X, Y, Z) \text{ is true.} \tag{23}$$

Note that the program P is stratified; it follows that brave reasoning under $\mathcal{SEQ}$-semantics is Σ_3^P-hard for disjunctive and stratified disjunctive programs; this proves the respective hardness parts of item (i) in Theorem 37. For cautious reasoning from disjunctive and stratified disjunctive programs under $\mathcal{SEQ}$-semantics, Π_3^P-hardness of item (ii) in Theorem 37 is shown by a slight extension of the reduction, which is carried out in Subsection C.2 to derive this result for fixed truth value v.

C.2 Hardness results for Problem INF with fixed truth value

C.2.1 Brave reasoning

The construction in Section 7.2 for normal, stratified normal and hcf programs uses $\mathbf{bt}$, but in no $\mathcal{SEQ}$-model any atom is true (all rules are constraints); thus we can add $b \leftarrow not\, a$ and ask for b about the truth value $\mathbf{f}$, and add further $c \leftarrow not\, b$ and ask for c about the truth value $\mathbf{t}$.

For disjunctive programs, we consider the Σ_3^P-hardness proof for brave reasoning under $\mathcal{SEQ}$-semantics in Section C.1. Then for the program P constructed from the QBF Φ and the particular atom q, we have that $P \models^{b,\mathbf{t}}_{\mathcal{SEQ}} q$ iff the QBF Φ evaluates to true, and $P \models^{b,\mathbf{t}}_{\mathcal{SEQ}} q$ is equivalent to $P \models^{b,\mathbf{bt}}_{\mathcal{SEQ}} p'$. Furthermore, q has never value $\mathbf{bt}$ in the $\mathcal{SEQ}$-models of the program P; if we let $P' = P \cup \{q' \leftarrow not\, q\}$, then $P' \models^{b,\mathbf{f}}_{\mathcal{SEQ}} q'$ iff $P \models^{b,\mathbf{t}}_{\mathcal{SEQ}} q$. So for each fixed value v, brave inference from the $\mathcal{SEQ}$-models of a (stratified) disjunctive program is Σ_3^P-hard; this trivially generalizes to $\mathcal{SEQ}$-models relative to arbitrary splitting sequences S.

C.2.2 Cautious reasoning

For fixed truth value $v = \mathbf{bt}$, the cautious inference problem is for $\mathcal{SEQ}$-models easier than for a truth value given in the input:

Proposition 68 *Given a program P and an atom a, deciding whether $P \models^{c,\mathbf{bt}}_{\mathcal{SEQ}} a$ is (i) in* coNP *for each of normal, normal stratified, and hcf P and (ii) in Π_2^p for disjunctive P.*

This holds because in this case, $P \not\models^{c,\mathbf{bt}}_{\mathcal{SEQ}} a$ iff some h-minimal HT-model (X, Y) of P exists such that $a \notin Y \setminus X$; such a h-minimal model can be guessed and verified in polynomial time in case (i) resp. in polynomial time with an NP oracle in case (ii).

For the other truth values, the construction in Section 7.2 for normal, stratified normal and hcf programs uses truth value $\mathbf{f}$ for cautious reasoning, and as in no $\mathcal{SEQ}$-model any atom is true, we can add $b \leftarrow not\, a$ and ask whether b has cautiously value $\mathbf{t}$; if we add another split layer with a rule $b \leftarrow not\, b, not\, a$ (such that $S = (S_1, S_2)$ and $b \in S_2 \setminus S_1$), then we can ask whether b has cautiously value $\mathbf{bt}$.

Regarding disjunctive programs, we had above in the programs P and P' for brave reasoning with fixed truth values $\mathbf{t}$ and $\mathbf{f}$ query atoms q resp. q' whose truth values are opposite in the $\mathcal{SEQ}$-models of P' and always true or false; so we immediately obtain the Π_3^P-hardness for cautious reasoning. If we add another split layer with $b \leftarrow not\, b, p$ similarly as above, then we can ask whether b has cautiously value $\mathbf{bt}$.

C.3 Constructing and recognizing canonical splitting sequences

Proof of Proposition 40. Let P be a program. First we prove that conditions (i) and (ii) in Definition 12 imply that there is no path from K_1 to K_2 and vice versa. By contradiction, first suppose that there is a path from K_1 to K_2, i.e. there exist $K'_1, \dots, K'_m \in \mathcal{SCC}(P)$ such that such that $K_1 = K'_1$, $K'_m = K_2$ and $(K'_i, K'_{i+1}) \in E_{SG}$ for $1 \leq i < m$. As in each topological ordering $(C_1, \dots, C_n) \in \mathcal{O}(SG(P))$ K'_{i+1} must precede K'_i, for $1 \leq i < m$, it follows that K_2 precedes K_1, which contradicts condition (i). Otherwise, suppose that there exists some path from K_2 to K_1. Let $K'_1, \dots, K'_m \in \mathcal{SCC}(P)$ be an arbitrary such path, i.e., $K'_1 = K_2$, $(K'_i, K'_{i+1}) \in E_{SG}$ for $1 \leq i < m$ and $K'_m = K_1$. By condition (ii) we know that $(K_2, K_1) \notin E_{SG}$. Hence $m > 2$ and $K'_{m-1} \neq K_1$, $K'_{m-1} \neq K_2$; thus in every topological ordering $(C_1, \dots, C_n) \in \mathcal{O}(SG(P))$, K_1 precedes K'_{m-i} and K'_{m-i} precedes K_2, which contradicts condition (i).

Now we prove that the disconnectedness hypothesis implies conditions (i) and (ii). As there is no path from K_2 to K_1, condition (ii) trivially holds. Moreover for each topological ordering of $\mathcal{SCC}(P)$ there exist maximal (possibly empty) sets $A_i \subseteq \mathcal{SCC}(P)$ such that for each $K'_i \in A_i$, K'_i precedes K_i, $i = 1, 2$. Because there is no path from K_1 to K_2, it follows that $K_2 \notin A_1$ and because there is no path from K_2 to K_1, it follows that $K_1 \notin A_2$. Therefore we can construct a topological ordering in which all strongly connected components in $A_1 \cup A_2$ precede K_1 (this is possible because if there exists some $K \in A_2$ such that K_1 precedes K, then K_1 precedes K and K precedes K_2; this contradicts the hypothesis that no path from K_2 to K_1 exists), and K_1 precedes immediately K_2, i.e., condition (i) holds (this is possible because there is no $K \in A_1$ such that K_2 precedes K). □

Proof of Corollary 41. ($\Rightarrow$) If (K_1, K_2) is a joinable pair witnessed by r, then by Proposition 40 K_1 and K_2 are disconnected in $SG(P)$; i.e., they are incomparable in the partial order on $\mathcal{SCC}(P)$ induced by $SG(P)$. By condition (iii), $At(r) \subseteq C_1 \cup \dots \cup C_{s+1}$ holds with $C_s = K_1$ and $C_{s+1} = K_2$; as every SCC $C \neq K_1, K_2$ such that $At(r) \cap C \neq \emptyset$ occurs in $C_1, \dots, C_{s-1}$, no path in $SG(P)$ from C can reach K_1 or K_2; consequently, K_1 and K_2 are maximal SCCs in $SG(P)$ such that $At(r) \cap C \neq \emptyset$

($\Leftarrow$) Suppose without loss of generality that $K_1 = C_1$ and $K_2 = C_2$. Then, K_1 and K_2 must be disconnected; hence by Proposition 40, K_1 and K_2 satisfy condition (i) and (ii) of a joinable pair. Furthermore, as all C_i, C_j, $1 \leq i \neq j \leq l$, must be pairwise disconnected, by extending the argument in the proof of Proposition 40, we can build from a topological ordering $\leq = (C_1, \dots, C_n)$ of $SG(P)$ another topological ordering of $SG(P)$ in which all SCCs in $A = \bigcup_{i=1}^{l} A_i \cup \{C_3, \dots, C_l\}$ precede K_1 and K_1 immediately precedes K_2, where $A_i = \{K \in \mathcal{SCC}(P) \mid K < C_i\}$; this is possible since no $K \in A$ exists such that K_2 precedes K. As $A \cup \{C_1, C_2\}$ must contain all SCCs C such that $At(r) \cap C \neq \emptyset$, it follows that condition (iii) holds; hence (K_1, K_2) is a joinable pair. □

Proof of Theorem 42. By Corollary 41, the joinable pairs (K_1, K_2), $K_1 \neq K_2$ witnessed by constraint r are given by all C^r_i, C^r_j from $C^r_1, \dots, C^r_l$ computed in Step 2, $1 \leq i \neq j \leq l$; hence, this collection is joinable, if $l > 1$; if $l = 1$, $K_1 = C^r_1$, $K_2 = C^r_1$ is trivially joinable. Thus, in Step 3 $C^r \in JC(P)$ holds. Furthermore, merging J_1 and J_2 in Step 4 results in a set $J_1 \cup J_2 \in JC(P)$: by an inductive argument, all $C^{r_i}_{j_i}$ that have been merged into J_i, $i = 1, 2$ are joinable; thus if $J_1 \cap J_2 \neq \emptyset$, then some $J \in J_1 \cap J_2$ exists such that all $(C^{r_1}_{j_1}, C)$ and $(C, C^{r_2}_{j_2})$ are joinable pairs; hence all C^r_j merged into $J_1 \cup J_2$ are joinable and $J_1 \cup J_2 \in JC(P)$. Finally, suppose that after Step 4 $\mathcal{MJC}(P) \neq MC \cup (\mathcal{SCC}(P) \setminus NMI)$; by construction of MC and the maximality condition on $\mathcal{MJC}(P)$, it follows that some $J' \in \mathcal{MJC}(P)$ and $J \in MC \cup (\mathcal{SCC}(P) \setminus NMI$ exist such that $J \subset J'$. From Corollary 41, it follows that all SCCs C merged into J' are joinable and that $J \in MC$ must hold; otherwise, J is a non-joinable SCC, which implies $J = J'$. Furthermore, some SCC C^r_j merged into J must be joinable to some SCC C merged into J' but not into J; as the joinable pair (C^r_j, C) is witnessed by some constraint r', C^r_j, C were merged into some $J'' in MC$; but this means $J \cap J'' \neq \emptyset$, and hence Step 4 for MC would not have been completed, a contradiction. Thus $\mathcal{MJC}(P) = MC \cup NMI$

holds. The correctness of the constructed $JG(P)$ is then obvious.

Regarding the time complexity, we note the following:

In Step 1, $DG(P)$, $\mathcal{SCC}(P)$ and $SG(P)$ are constructable in linear time;

We can compute the SCCs $C_1^r, \ldots, C_l^r$ efficiently, e.g. by using a stratified program P^r with the following rules:

1. $r_j \leftarrow$, for each $C_j \in V_{SG}$ such that $C_j \cap At(r) \neq \emptyset$;
2. $r_j \leftarrow r_i$ and $n\_max\_r_j \leftarrow r_i$, for each $(C_i, C_j) \in E_{SG}$;
3. $max\_r_j \leftarrow r_j,\ not\, n\_max\_r_j$, for each $C_i \in V_{SG}$.

Informally, the atom r_j encodes reachability of the component C_j in the $\mathcal{SCC}$-graph from a component that contains atoms from the constraint r; max_r_j and $n\_max\_r_j$ are used to single out the topmost (maximal) reached components using double negation. The single answer set of P_r yields then the desired maximal components $C_1^r, \ldots, C_l^r$; as P_r can be built and evaluated in linear time, Step 2 is feasible in linear time for each r.

Step 3 is clearly feasible in linear time; also Step 4 (iterative merging the J_1, J_2) is feasible (if properly done) in linear time, and similarly Step 5 given $\mathcal{MJC}(P)$ and $SG(P)$.

Thus in total, $\mathcal{MJC}(P)$ and $JG(P)$ are computable in time $\mathcal{O}(cs \cdot \|P\|)$, which proves the result. □

D Section 8

Proof of Theorem 43. The proof proceeds as follows. We first show that (1) the models of $P^{\mathcal{E}}$ correspond to the HT-models (X, Y) of P via $\cdot^{\mathcal{E}}$; next, we establish that (2) for every minimal model $P^{\mathcal{E}}$, the corresponding HT-model of P is h-minimal and (3) that every $\mathcal{SEQ}$-model of P is among the models in (2), i.e., $\{(X,Y)^{\mathcal{E}} \mid (X,Y) \in \mathcal{SEQ}(P)\} \subseteq MM(P^{\mathcal{E}})$. As the $\mathcal{E}$-violation set $\mathcal{V}(I)$ of any model $I = (X,Y)^{\mathcal{E}}$ of $P^{\mathcal{E}}$ corresponds to the gap of (X, Y) (precisely, $\mathcal{V}(I) = \mathcal{E}gap(X,Y)$), it follows that $I \in MM(P^{\mathcal{E}})$ has a $\subseteq$-minimal $\mathcal{E}$-violation set, i.e., is an evidential stable model of P, iff (X, Y) is a $\mathcal{SEQ}$-model of P.

As for (1), it is readily seen that for every HT-model (X, Y) of P, $I = (X,Y)^{\mathcal{E}} = X \cup \mathcal{E}Y$ is a model of $P^{\mathcal{E}}$: all rules (2) are satisfied as $Y \models P$, and all rules (3) as $X \subseteq Y$. Finally for the rules (1), as $(X, Y) \models r$, either $H(r) \cap X \neq \emptyset$, or $B^+(r) \not\subseteq Y$ (which implies $B^+(r) \not\subseteq X$), or $B^-(r) \cap Y \neq \emptyset$; hence I satisfies the rules (1). The proof of the converse, for every model I of $P^{\mathcal{E}}$, $\beta(I)$ is a HT-model of P, is similar.

Regarding (2), if $I \in MM(P^{\mathcal{E}})$, in particular no model $J \subset I$ of $P^{\mathcal{E}}$ exists such that $I \setminus \Sigma = J \setminus \Sigma$; thus if $\beta(I) = (X, Y)$, no HT-model (X', Y) of P exists such that $X' \subset X$.

As for (3), towards a contradiction assume that some $(X,Y) \in \mathcal{SEQ}(P)$ fulfills $I = (X,Y)^{\mathcal{E}} \notin MM(P^{\mathcal{E}})$. Hence, some $J = (X',Y')^{\mathcal{E}} \in MM(P^{\mathcal{E}})$ exists such that $J \subset I$. As $X' \subseteq X$, $Y' \subseteq Y$, and (X, Y) is h-minimal, it follows that $Y' \subset Y$. As $P^Y \subseteq P^{Y'}$ it follows that $X' \models P^Y$; since $X \in MM(P^Y)$ and $X' \subseteq X$, it follows $X' = X$. Therefore, $gap(X', Y') \subset gap(X, Y)$; as by (2) (X', Y') is h-minimal, $(X, Y) \notin \mathcal{SEQ}(P)$, which is a contradiction. This proves the result. □ □

Proof of Proposition 49. ($\subseteq$) If $M = (X, Y)$ is a $\mathcal{SEQ}$-model of P^{wf}, then M is a h-minimal model of P^{wf} and $gap(M) \subseteq gap(WF(P^{wf})) = gap(WF(P))$. Corollary 48 implies that $M \sqsubseteq WF(P^{wf})) = WF(P) = (I, J)$, and thus $Y \subseteq J$. By antimonotonicity of $\gamma_P(.)$, we have $\gamma_P(Y) \supseteq \gamma_P(J) = I$, and thus $\gamma_{P^{wf}}(Y) = \gamma_P(Y) \cup I = \gamma_P(Y) = X$. Thus M is also a h-minimal model of P. If M were not a $\mathcal{SEQ}$-model of P, then by Corollary 48 some refinement M' of $WF(P)$ with $gap(M') \subset gap(M)$ would be a $\mathcal{SEQ}$-model of P. But M' would then be a h-minimal model of P^{wf} and contradict that M is a $\mathcal{SEQ}$-model of P^{wf}. Thus M is a $\mathcal{SEQ}$-model of P.

($\supseteq$). Let M be a $\mathcal{SEQ}$-model of P such that $gap(M) \subseteq gap(WF(P))$. Then by Corollary 48 M refines $WF(P)$ and thus is clearly a model of P^{wf}, and moreover h-minimal. If M were not a $\mathcal{SEQ}$-model of P^{wf}, then some $\mathcal{SEQ}$-model M' of P^{wf} with smaller gap exists; we can then as in the case ($\subseteq$) infer that M' is also a h-minimal model of P, which contradicts that M is a $\mathcal{SEQ}$-model of P. $\Box$

Proof of Proposition 50. Consider any splitting sequence $S = (S_1, S_2, ...)$ of the program P and let $M = (X, Y)$ be any $\mathcal{SEQ}$-model of P such that $M \sqsubseteq WF(P)$ (by Corollary 48 such an M exists). Let $M_1 = M|_{S_1}$ and $P_1 = b_{S_1}(P)$.

Then, M_1 is a HT-model of P_1 and moreover h-minimal for P_1 (for otherwise, M would not be h-minimal for P: we could make X on S_1 smaller, as we can keep the same valuation for the atoms in $\Sigma \setminus S_1$; note that P^Y is positive and atoms from S_1 occur in $t_{S_1}(P)$ only in rule bodies). Furthermore, we have $M_1 \sqsubseteq WF(P)|_{S_1}$. Now some $\mathcal{SEQ}$-model $N_1 = (X_1, Y_1)$ of P_1 must exist such that $gap(N_1) \subseteq gap(M_1)$; as $gap(M_1) \subseteq gap(WF(P)|_{S_1})$, Corollary 48 and Lemma 51 imply that $N_1 \sqsubseteq WF(P_1)$ (observe that $WF(P)|_{S_1} = WF(P_1)$, which follows from items 1 and 2 of Lemma 51).

If we consider the program $P_2 = P^{S_1}(X_1, Y_1)$, then by an inductive argument on the length of the splitting sequence it has some $\mathcal{SCC}$-model N_2 w.r.t. $S' = (S_2, ..., S_n)$ such that $N_2 \sqsubseteq WF(P_2)$, provided $WF(P_2)$ exists; however, $P^{S_1}(X_1, Y_1)$ adds a constraint $\leftarrow not\, a$ for each $a \in Y_1 \setminus X_1$, and as a does not occur in any rule head of P_2, $WF(P_2)$ does not exist if $X_1 \subset Y_1$. To address this, we use in the argument a variant of the transformation $P^{S_1}(X_1, Y_1)$, denoted $\hat{P}^{S_1}(X_1, Y_1)$, that adds a rule $a \leftarrow not\, a$ for each $a \in Y_1$ to $P^{S_1}(X_1, Y_1)$; clearly, $P^{S_1}(X_1, Y_1)$ and $\hat{P}^{S_1}(X_1, Y_1)$ have the same splitting sets and the same $\mathcal{SEQ}$-models w.r.t. any splitting sequence; let $\hat{P}_2 = \hat{P}^{S_1}(X_1, Y_1)$, Then we claim that $WF(\hat{P}_2)$ exists and $WF(\hat{P}_2) \sqsubseteq WF(P)$ holds. Indeed, consider the constraint-free part P' of P; then $WF(P') = WF(P)$ and, if Q' denotes the (constraint-free) program for P' according to item 2 of Lemma 51, we have $WF(Q') = WF(P') = WF(P)$. If we add to Q' all constraints of P, then the resulting program Q fulfills $WF(Q) = WF(P)$. If we modify Q by (i) adding from $\hat{P}^{S_1}(X_1, Y_1)$ all facts $a \in X_1$ and all constraints $\{a \leftarrow not\, a \mid a \in Y_1\} \cup \{\leftarrow a \mid a \in S_1 \setminus Y_1\}$, and (ii) remove all rules $a \leftarrow not\, a$ such that $a \in S_1 \setminus Y_1$, the resulting program Q'' is such that $WF(Q'') \sqsubseteq WF(Q) = WF(P)$ if $WF(Q'')$ exists, as assigning any atoms in $gap(WF(P))$ true or false does not affect the already assigned atoms. But as every constraint r in P has some body literal that is false in $WF(P)$, this holds also for Q'', and thus $WF(Q'')$ exists. Now we note that $Q'' = \hat{P}_2$; this proves the claim.

Consequently, N_2 is an $\mathcal{SCC}$-model of $\hat{P}_2$ and $N_2 \sqsubseteq WF(\hat{P}_2) \sqsubseteq WF(P)$ holds. Now the $\mathcal{SEQ}^S$-models of P are, by definition,

$$\begin{aligned}
\mathcal{SEQ}^S(P) &= mc\Big(\bigcup_{(X,Y)\in\mathcal{SEQ}(b_{S_1}(P))} \mathcal{SEQ}^{S'}(P^{S_1}(X,Y)\Big) \\
&= mc\Big(\bigcup_{(X,Y)\in\mathcal{SEQ}(b_{S_1}(P))} \mathcal{SEQ}^{S'}(\hat{P}^{S_1}(X,Y)\Big).
\end{aligned}$$

If the model N_2 appears in this set, then it is an $\mathcal{SEQ}^S$-model of P that refines $WF(P)$ and proves the first claim of the proposition. Otherwise, some $\mathcal{SEQ}^S$-model N' of P must exist such that $gap(N') \subset gap(N_2)$; as N' is a $\mathcal{SEQ}$-model of P and $gap(N') \subseteq gap(WF(P))$, it follows from Corollary 48 that $N' \sqsubseteq WF(P)$, and also in this case an $\mathcal{SEQ}^S$-model of P that refines $WF(P)$ exists; this proves the first claim of the proposition. As for the second claim, by Corollary 48 every $\mathcal{SEQ}$-model M of P, and thus in particular every $\mathcal{SEQ}^S$-model M of P such that $gap(M) \subseteq gap(WF(P))$ satisfies $M \sqsubseteq WF(P)$; thus if we let M in the argument above be an arbitrary $\mathcal{SEQ}^S$-model of P, we arrive at $N_2 = M$ and thus the second claim holds. This proves the result. $\Box$

References

[1] Alcântara, J., Damásio, C.V., Pereira, L.M.: A declarative characterization of disjunctive paraconsistent answer sets. In: de Mántaras, R.L., Saitta, L. (eds.) Proc. 16th European Conference on Artificial Intelligence (ECAI'2004). pp. 951–952. IOS Press (2004)

[2] Amendola, G., Eiter, T., Leone, N.: Modular paracoherent answer sets. In: Fermé, E., Leite, J. (eds.) Proc. 14th European Conference on Logics in Artificial Intelligence (JELIA 2014). pp. 457–471. No. 8761 in LNCS/LNAI, Springer (2014)

[3] Apt, K., Blair, H., Walker, A.: Towards a theory of declarative knowledge. In: Minker [26], pp. 89–148

[4] Balduccini, M., Gelfond, M.: Logic programs with consistency-restoring rules. In: McCarthy, J., Williams, M.A. (eds.) International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series. pp. 9–18 (2003)

[5] Baral, C., Subrahmanian, V.S.: Dualities between alternative semantics for logic programming and nonmonotonic reasoning. J. Automated Reasoning 10(3), 399–420 (1993)

[6] Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, Cambridge (2003)

[7] Ben-Eliyahu, R., Dechter, R.: Propositional semantics for disjunctive logic programs. Annals of Mathematics and Artificial Intelligence 12, 53–87 (1994)

[8] Blair, H.A., Subrahmanian, V.S.: Paraconsistent logic programming. Theor. Comput. Sci. 68(2), 135–154 (1989)

[9] Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: Proc. 22nd Conference on Artificial Intelligence (AAAI '07), July 22-26, 2007, Vancouver. pp. 385–390. AAAI Press (2007)

[10] Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Communications of the ACM 54(12), 92–103 (2011)

[11] Cabalar, P., Odintsov, S.P., Pearce, D.: Logical foundations of well-founded semantics. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 25–35. AAAI Press (2006)

[12] Cabalar, P., Odintsov, S.P., Pearce, D., Valverde, A.: Partial equilibrium logic. Ann. Math. Artif. Intell. 50(3-4), 305–331 (2007)

[13] Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Modular nonmonotonic logic programming revisited. In: Hill, P., Warren, D. (eds.) Proc. 25th International Conference on Logic Programming (ICLP 2009). pp. 145–159. No. 5649 in Lecture Notes in Computer Science, Springer (2009)

[14] Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: Propositional case. Annals of Mathematics and Artificial Intelligence 15(3/4), 289–323 (1995)

[15] Eiter, T., Fink, M., Moura, J.: Paracoherent answer set programming. In: Proc. 12th International Conference on Principles on Knowledge Representation and Reasoning (KR 2010). pp. 486–496. AAAI Press (2010)

[16] Eiter, T., Leone, N., Saccà, D.: On the partial semantics for disjunctive deductive databases. Annals of Mathematics and Artificial Intelligence 19(1/2), 59–96 (1997)

[17] Faber, W., Greco, G., Leone, N.: Magic sets and their application to data integration. J. Comput. Syst. Sci. 73(4), 584–609 (2007)

[18] Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2012),

[19] Gelder, A.V.: The alternating fixpoint of logic programs with negation. J. Comput. Syst. Sci. 47(1), 185–221 (1993),

[20] Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385 (1991)

[21] Huang, S., Li, Q., Hitzler, P.: Reasoning with inconsistencies in hybrid MKNF knowledge bases. Logic Journal of the IGPL 21(2), 263–290 (2013)

[22] Janhunen, T., Oikarinen, E., Tompits, H., Woltran, S.: Modularity aspects of disjunctive stable models. J. Artif. Intell. Res. (JAIR) 35, 813–857 (2009)

[23] Kakas, A.C., Mancarella, P.: Generalized stable models: A semantics for abduction. In: Proc. 9th Eureopean Conference on Artificial Intelligence (ECAI 1990). pp. 385–391, IOS Press (1990)

[24] Lifschitz, V., Turner, H.: Splitting a logic program. In: Proc. International Conference on Logic Programming (ICLP-94). pp. 23–38. MIT-Press (1994)

[25] Marek, V.W., Nerode, A., Remmel, J.B.: Logic programs, well-orderings, and forward chaining. Annals of Pure and Applied Logic 96(1-3), 231–276 (1999)

[26] Minker, J. (ed.): Foundations of Deductive Databases and Logic Programming. Morgan Kaufman, Washington DC (1988)

[27] Odintsov, S.P., Pearce, D.: Routley semantics for answer sets. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) Proc. 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005). Lecture Notes in Computer Science, vol. 3662, pp. 343–355. Springer (2005)

[28] Osorio, M., Ramírez, J.R.A., Carballido, J.L.: Logical weak completions of paraconsistent logics. J. Log. Comput. 18(6), 913–940 (2008)

[29] Pearce, D.: Equilibrium logic. Annals of Mathematics and Artificial Intelligence 47(1-2), 3–41 (2006)

[30] Pearce, D., Valverde, A.: Quantified equilibrium logic and foundations for answer set programs. In: de la Banda, M.G., Pontelli, E. (eds.) Proc. 24th International Conference on Logic Programming (ICLP 2008). Lecture Notes in Computer Science, vol. 5366, pp. 546–560. Springer (2008)

[31] Pereira, L.M., Alferes, J.J., Aparício, J.N.: Contradiction removal semantics with explicit negation. In: Masuch, M., Pólos, L. (eds.) International Conference on Logic at Work: Knowledge Representation and Reasoning Under Uncertainty. Lecture Notes in Computer Science, vol. 808, pp. 91–105. Springer (1992)

[32] Pereira, L.M., Pinto, A.M.: Revised stable models - a semantics for logic programs. In: Bento, C., Cardoso, A., Dias, G. (eds.) Proc. 12th Portuguese Conference on Artificial Intelligence (EPIA 2005). Lecture Notes in Computer Science, vol. 3808, pp. 29–42. Springer (2005)

[33] Pereira, L.M., Pinto, A.M.: Approved models for normal logic programs. In: Dershowitz, N., Voronkov, A. (eds.) Proc. 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2007). Lecture Notes in Computer Science, vol. 4790, pp. 454–468. Springer (2007)

[34] Pereira, L.M., Pinto, A.M.: Layered models top-down querying of normal logic programs. In: Gill, A., Swift, T. (eds.) Proc. 11th International Symposium on Practical Aspects of Declarative Languages, (PADL 2009). Lecture Notes in Computer Science, vol. 5418, pp. 254–268. Springer (2009)

[35] Przymusinski, T.C.: On the declarative semantics of deductive databases and logic programs. In: Minker [26], pp. 193–216

[36] Przymusinski, T.C.: Stable semantics for disjunctive programs. New Generation Computing 9, 401–424 (1991)

[37] Saccà, D., Zaniolo, C.: Partial models and three-valued stable models in logic programs with negation. In: Subrahmanian, V. (ed.) Proc. First Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR 1991), pp. 87–101. MIT Press (1991)

[38] Sakama, C., Inoue, K.: Paraconsistent stable semantics for extended disjunctive programs. J. Log. Comput. 5(3), 265–285 (1995)

[39] Seipel, D.: Partial evidential stable models for disjunctive deductive databases. In: Dix, J., Pereira, L.M., Przymusinski, T.C. (eds.) Proc. Third International Workshop on Logic Programming and Knowledge Representation (LPKR '97), Selected Papers. Lecture Notes in Computer Science, vol. 1471, pp. 66–84. Springer (1997)

[40] Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. 1(2), 146–160 (1972)

[41] Turner, H.: Strong equivalence made easy: nested expressions and weight constraints. Theory and Practice of Logic Programming 3(4-5), 609–622 (2003),

[42] van Gelder, A., Ross, K., Schlipf, J.: The well-founded semantics for general logic programs. J. ACM 38(3), 620–650 (1991)

[43] Wang, K., Zhou, L.: Comparisons and computation of well-founded semantics for disjunctive logic programs. ACM Trans. Comput. Log. 6(2), 295–327 (2005)

[44] Wang, Y., Zhang, M., You, J.H.: Logic programs, compatibility and forward chaining construction. J. Comput. Sci. Technol. 24(6), 1125–1137 (2009)

[45] You, J.H., Yuan, L.: A three-valued semantics for deductive databases and logic programs. J. Computer and System Sciences 49, 334–361 (1994)