Disjunctive logic programs, answer sets, and the cut rule

Éric Martin¹

Received: 31 January 2018 / Accepted: 7 February 2022 / Published online: 4 March 2022
© The Author(s) 2022

Abstract

In Minker and Rajasekar (J Log Program 9(1):45–74, 1990), Minker proposed a semantics for negation-free disjunctive logic programs that offers a natural generalisation of the fixed point semantics for definite logic programs. We show that this semantics can be further generalised for disjunctive logic programs with classical negation, in a constructive modal-theoretic framework where rules are built from *claims* and *hypotheses*, namely, formulas of the form $\Box\varphi$ and $\Diamond\Box\varphi$ where φ is a literal, respectively, yielding a “base semantics” for general disjunctive logic programs. Model-theoretically, this base semantics is expressed in terms of a classical notion of logical consequence. It has a complete proof procedure based on a general form of the cut rule. Usually, alternative semantics of logic programs amount to a particular interpretation of nonclassical negation as “failure to derive.” The counterpart in our framework is to complement the original program with a set of hypotheses required to satisfy specific conditions, and apply the base semantics to the resulting set. We demonstrate the approach for the answer set semantics. The proposed framework is purely classical in mainly three ways. First, it uses classical negation as unique form of negation. Second, it advocates the computation of logical consequences rather than of particular models. Third, it makes no reference to a notion of preferred or minimal interpretation.

Keywords Disjunctive logic programs · Answer sets · Tableau proofs · Modalities · Cut · Negation

Mathematics Subject Classification 68N17 · 03B45

1 Introduction

1.1 Disjunctive logic programs, fixed points, and negation

As noted in [19], the field of disjunctive logic programming had its beginnings in 1982, but the first major semantics for disjunctive logic programs were proposed in 1990, in

✉ Éric Martin
eric.martin@unsw.edu.au

¹ School of Computer Science and Engineering, The University of New South Wales, UNSW, Sydney 2052, Australia

[17], which offered, in particular, a natural generalisation of the fixed point semantics for definite logic programs given in [27]. The key feature of that generalised semantics is that disjunctions of atoms are generated using a bottom-up approach applied to sets of rules, assumed to have (possibly empty) conjunctions of atoms as bodies and nonempty disjunctions of atoms as heads. The model-theoretic interpretation is easy and flexible, as the absence of negation in the rules allows one to interpret disjunction either constructively or not, and also to choose for intended interpretations either all structures, all standard (Herbrand) structures, or all minimal standard structures.

It has been observed, in [21] in particular, that the generation process at work in [17] is a form of application of the hyper-resolution rule, that involves a finite but arbitrary number of clauses, as opposed to classical resolution that uses precisely two clauses as premises. Resolution, like modus ponens, is a form of the cut rule. We will refer to the cut rule to motivate and describe a generalisation to the immediate consequence operator associated with the fixed point semantics studied in [17]. The target will be the more general class of disjunctive logic programs with possibly empty heads, and more crucially, with possible occurrences of negation in the bodies and in the heads of the rules. Before we clarify what we actually mean by “occurrences of negation,” let us recall that the field of logic programming has first focused on logic programs without negation, then logic programs with nonclassical negation in the bodies of the rules, and then logic programs with nonclassical negation in the bodies of the rules and classical negation in the bodies and the heads of the rules; see [2] for a survey. Some researchers have also proposed to consider more than two forms of negation [1].

Similarly to the situation with normal programs (sets of rules with no disjunction in the heads), different views on nonclassical negation have given rise to a large number of alternative semantics. “Failure to derive” is a notion that can be applied to sets of rules with no occurrence of (any form of) negation, and [17] already establishes a relationship between their fixed-point semantics and the generalised closed world assumption. When nonclassical negation is allowed to occur in the bodies of the rules, many possible interpretations become possible. But the various interpretations of nonclassical negation do not exhaust all issues raised by the presence of negation in logic programs: in [12], an argument is made that it is often desirable to be able to work with even more powerful disjunctive programs, in which classical negation can be used, and the answer set semantics is proposed as a natural alternative semantics able to deal with both classical and nonclassical negation. See [3], [7], [23] for other approaches and complexity results on disjunctive logic programs where both nonclassical and classical negation coexist.

Let us focus on classical negation first. What happens to the fixed point semantics in [17] when classical negation enters the stage? A technique to reduce programs with classical negation to programs without is proposed in both [12] and in [24] and discussed in [18]¹; essentially, for every predicate symbol $\wp$, a new predicate symbol $\wp^\neg$ is introduced, that allows one to get rid of $\neg$ by replacing the occurrences of $\neg\wp$ with $\wp^\neg$, and (taking $\wp$ to be nullary not to clutter the discussion) an integrity constraint $\leftarrow \wp \wedge \wp^\neg$ is added to express mutual exclusivity between $\wp$ and $\wp^\neg$. The immediate consequence operator in [17], slightly generalised to accept rules with

¹ The same technique has also been used to reduce programs with nonclassical negation (and no classical negation) to programs without: see for instance [8].

an empty head, can then be applied. For instance, the rules $p \leftarrow$ and $\neg p \vee \neg q \leftarrow$ are transformed into $p \leftarrow$ and $p^\neg \vee q^\neg \leftarrow$, which complemented with $\leftarrow p \wedge p^\neg$, allows the immediate consequence operator in [17] generalised to accept rules with an empty head to generate $q^\neg$; and if one added the rule $\neg q \leftarrow q^\neg$, then one would eventually generate $\neg q$, as required from a complete proof procedure. This solution is not fully satisfactory though, as one has obtained a complete proof procedure for an extension of the original set of rules, not for the set of rules itself. Our immediate consequence operator will be strongly related to the one in [17], but will deal gracefully with classical negation, without enriching the original language.

Now let us focus on nonclassical negation. In close relationship to stable autoepistemic expansions ([14, 20]), [26] lets the language of disjunctive logic programs include a modal operator $\mathcal{B}$ that is used to capture nonclassical negation in one of two ways: *not* $\wp$ is expressed as either $\mathcal{B}\neg\wp$ or $\neg\mathcal{B}\wp$, conveying that $\neg\wp$ is believed or that $\wp$ is not believed, respectively. So classical negation is used in combination with $\mathcal{B}$ in one of the two proposed translations of *not*. As for classical negation in the original program, it gives rise in [26] to a third form of negation, denoted $\sim$ and referred to as “strong” negation. Based on a relation of minimal model entailment, [5, 26] define a notion of static expansion that yields a fixed-point semantics for disjunctive logic programs; the associated notion of logical consequence is not standard as it requires selecting, amongst all possible interpretations, those that happen to be minimal according to an underlying notion of closed-world assumption. In this paper, we will show that we can further exploit the power of modal operators and work in a classical framework, in which classical negation is the only form of negation, a theme which has been developed in depth in [15]. Disjunctive logic programs will be modal disjunctive sets of rules, and the least fixed point of such a set of rules $\mathcal{F}$ will consist precisely of the disjunctions that are logical consequences of $\mathcal{F}$. In contrast to [26] in which the heads and bodies of rules are disjunctions and conjunctions of both formulas with no occurrence of $\mathcal{B}$ and formulas with occurrences of $\mathcal{B}$, our “building blocks” will be formulas of the form $\Box\varphi$ or $\Diamond\Box\varphi$ where φ is a literal. More specifically, given an atom $\wp$, we will map $\wp$ to $\Box\wp$, $\neg\wp$ to $\Box\neg\wp$, *not* $\wp$ to $\Diamond\Box\neg\wp$ and *not* $\neg\wp$ to $\Diamond\Box\wp$. This mapping is studied in depth in the context of nondisjunctive logic programs in [16]. One of the key properties of the proposed syntax of rules is that disjunction is constructive. In [26], it is claimed that “as pointed out by several researchers, the form of negation proposed by Gelfond and Lifschitz does not represent real classical negation $\neg F$ but rather its weaker form, denoted here by $\sim F$, which does not require the law of excluded middle $F \vee \sim F$.” This paper adopts a different view, namely, that only classical negation is necessary, and that there are other ways to circumvent excluded middle than to introduce a weaker form of negation; rather than expressing that either F or $\neg F$ holds, one can express that either F or $\neg F$ has been derived, that is, $\Box F \vee \Box\neg F$, in accordance with Gödel’s interpretation of intuitionistic logic in S4. Our representation of *not* allows disjunction to behave constructively with respect to both $\neg$ and *not*, as $\Box p \vee \Diamond\Box\neg p$ will not be valid. In contrast, $\Box p \vee \Diamond\neg p$ would be valid, so $\Diamond\Box$ cannot be replaced by $\Diamond$; our results will provide evidence that for our purposes, $\Diamond\Box$ is the right choice. There are many examples of frameworks where the “building blocks” are of the form $\Box\varphi$ and $\Diamond\varphi$, in particular within the setting of autoepistemic logic (e.g., [10]), but not only (e.g., [22]).

1.2 Answer sets and program transformation

Let us describe the answer set semantics with an example. Consider the following set of rules, $\mathcal{R}$.

$$\begin{array}{ll} p_2 \wedge p_3 \rightarrow p_1 & \text{not } \neg p_2 \vee \text{not } p_4 \rightarrow \neg p_1 \\ p_4 \rightarrow p_2 & \neg p_3 \rightarrow \neg p_2 \\ p_3 \rightarrow p_3 & \neg p_3 \wedge \text{not } \neg p_2 \rightarrow \neg p_3 \\ \text{not } p_3 \rightarrow p_4 & \end{array}$$

It has $X = \{\neg p_1, p_2, p_4\}$ as an answer set. Indeed, let us replace in (the bodies of the rules in) $\mathcal{R}$ all formulas of the form $\text{not } \psi$ by true if $\psi \notin X$, by false otherwise, and simplify. This is the Gelfond-Lifschitz transformation [11]. It results in the following set of rules, $\mathcal{R}'$:

$$\begin{array}{ll} p_2 \wedge p_3 \rightarrow p_1 & \rightarrow \neg p_1 \\ p_4 \rightarrow p_2 & \neg p_3 \rightarrow \neg p_2 \\ p_3 \rightarrow p_3 & \neg p_3 \rightarrow \neg p_3 \\ & \rightarrow p_4 \end{array}$$

For all atoms α , replace in $\mathcal{R}'$ all occurrences of $\neg\alpha$ by a new atom α^* , resulting in a new set of rules, $\mathcal{R}''$:

$$\begin{array}{ll} p_2 \wedge p_3 \rightarrow p_1 & \rightarrow p_1^* \\ p_4 \rightarrow p_2 & p_3^* \rightarrow p_2^* \\ p_3 \rightarrow p_3 & p_3^* \rightarrow p_3^* \\ & \rightarrow p_4 \end{array}$$

The logical consequences of $\mathcal{R}''$ is $\{p_1^*, p_2, p_4\}$, and that set consists precisely of the members of X where all negated atoms of the form $\neg\alpha$ have been replaced by α^* , completing the verification that X is an answer set for $\mathcal{R}$.

The answer set semantics imposes a condition that in our framework, translates into complementing the (modal version of the) logic program with a set of formulas of the form $\Diamond\Box\varphi$. More precisely: if $\Diamond\Box\varphi$ occurs in the (modal version of the) logic program, then that program should be complemented with $\Diamond\Box\varphi$ provided that the resulting set of formulas does not logically imply $\Box\neg\varphi$ (which will logically imply $\Diamond\Box\neg\varphi$, hence be logically inconsistent with $\Diamond\Box\varphi$). One can think of this condition as: any possible hypothesis ψ —formula of form $\Diamond\Box\varphi$ that occurs in the body of some rule—should be made, unless the resulting theory (the modal logic program complemented with the set of selected hypotheses) refutes ψ .² If we were dealing with normal programs,

² Variations on the same idea can be used to capture other semantics besides the answer set semantics, and in particular the well-founded semantics; the key condition that captures the well-founded semantics would be expressed not in terms of “nonrefutation of hypothesis,” but in terms of “confirmation of hypothesis,” when adding $\Diamond\Box\varphi$ (and possibly other hypotheses) to a modal logic program results in a theory that logically implies $\Box\varphi$. But these considerations go beyond the key purpose of this paper and are developed in depth in [16], hence we will not say anything more on the well-founded semantics.

then no other machinery would be needed to capture the answer set semantics. But as we are dealing with logic programs whose heads are disjunctions of (one or more) literals, it is necessary to introduce one more component. Indeed, the cut rule will not be applied to $\mathcal{F}$, but to a closure of $\mathcal{F}$ under an operation that creates more rules, while still preserving logical validity (the created rules will be logical consequences of those from which they originate). What is required is a form of program extension, a particular case of program transformation. Program transformation is a technique that has been widely used in logic programming in general, and for the particular purpose of defining semantics of disjunctive programs, for example in [4]. Essentially, the transformation we shall use will, from a rule of the form³

$$\psi \rightarrow \Box \wp_1 \vee \cdots \vee \Box \wp_n \vee \Box \wp_{n+1} \vee \cdots \vee \Box \wp_{n+k}$$

produce rules of the following form.

$$\psi \wedge \Diamond \Box \neg \wp_{n+1} \wedge \cdots \wedge \Diamond \Box \neg \wp_{n+k} \rightarrow \Box \wp_1 \vee \cdots \vee \Box \wp_n$$

In a different syntactic setting, this operation was studied in [6] to transform disjunctive programs into a normal program. In our framework, the original set of rules will not be replaced by, but complemented with, formulas obtained by left-shift. The answer set semantics lets literals assumed not to hold play a role in activating a rule by having them preceded with *not* in the body of the rule. Translated into our setting, the set of literals preceded with *not* in the bodies of some rules makes up a sufficient pool of hypotheses to cast the answer set semantics, provided the focus is on normal programs. But when dealing with disjunctive logic programs, this pool of literals is not sufficient, and the operation above, that left-shifts some formulas and turns their negations into hypotheses, makes it possible to usefully expand it as needed, no more, no less, and capture exactly the classical notion of an answer for a disjunctive logic program, as defined for instance in [10]. Interestingly, this operator is not only necessary in relation to the answer set semantics, but also for what we referred to above as the “base semantics,” that is, essentially, the fixed point semantics in [17] generalised to disjunctive logic programs with classical negation.

There are alternative approaches to cast the answer set semantics in a logical framework where the rules of a logic program are modified or complemented with other formulas. The framework can be classical; see for instance [13], where “loop formulas” are added, in a setting restricted to logic programs without classical negation, and more importantly, limited to finite sets of finite rules—two conditions that we do not impose. The framework can be nonclassical; see for instance [25], in a setting that sits between intuitionistic logic and classical logic. There is a vast literature that examines how to compute answer sets; see [9] for a good example of work along those lines. Our study sheds no light on how to compute answer sets. Our framework is not classical in the sense that classical logical inferences would make it possible to derive, from a fixed theory, all answer sets and nothing but answer sets. Rather, its key features are the following.

³ As we will see, in our framework $\rightarrow$ will be classical implication, which is why we write the body of a rule on the left hand side and the head on the right hand side here.

- It allows one to formalise a very general class of logic programs, with mechanical translations into the chosen logical language.
- It characterises semantics of interest in the form of some formulas being logical consequences of some theory T , possibly together with conditions imposing that certain formulas be or be not logical consequences of T .
- It is a unifying framework in that the characterisations all rely on a fixed set of underlying notions and a fixed classical notion of logical consequence (a claim that is partially supported by the work described in this paper, but more strongly so thanks to the additional work presented in other papers).

1.3 Plan of the paper

We proceed as follows. In Sect. 2, we motivate the notions to be introduced in the rest of the paper. In Sect. 3, we fix the logical background, and in particular, define the modal language from which the bodies and heads of a disjunctive logic program will be made up, together with its semantics. In Sect. 4, we formalise the left shift operation and establish the relationship between this framework and the answer set semantics; the rest of the paper can then be applied to the answer set semantics as a particular case. In Sect. 5, we introduce an intermediate proof system, in the spirit of tableau proofs, and establish its completeness with respect to the class of disjunctive logic programs under consideration. In Sect. 6, we let the cut rule be a counterpart to the immediate consequence operator described in [17]; we then establish the completeness of the system of proof based on applications of this rule. Essentially, we convert a tableau proof into a proof by cuts, a technique which is interesting in its own right.

2 Motivation

2.1 Objective

Consider the very simple sets of rules whose left hand sides are empty or positive propositional formulas (built from atoms using conjunction and disjunction) and whose right hand sides are atoms.⁴ Here is an example of such a set of rules $\mathcal{P}$:

$$\begin{array}{ll}
 \rightarrow p_1 & (p_3 \wedge p_5) \rightarrow p_8 \\
 \rightarrow p_2 & p_7 \rightarrow p_8 \\
 (p_1 \vee p_5) \wedge p_2 \rightarrow p_3 & p_5 \wedge p_7 \rightarrow p_9 \\
 (p_1 \wedge p_4) \vee p_6 \rightarrow p_4 & (p_1 \vee p_4) \wedge (p_5 \vee p_8) \rightarrow p_{10} \\
 p_3 \rightarrow p_5 &
 \end{array}$$

⁴ From now on we write the rules of a classical (nonmodal) logic program with the body on the left hand side and the head on the right hand side, “pretending” that the arrow is classical implication. This is for the sake of making the discussion easier to follow.

One can describe the set of literals $[\mathcal{P}]$ that are logical consequences of this set of implications, namely, $\{p_1, p_2, p_3, p_5, p_8, p_{10}\}$, as the $\subseteq$ -minimal fixed point of $\mathcal{P}$. One can also describe it as the result of

- first collecting the facts—the right hand sides of the rules with empty left hand sides—and
- then firing the rules that can be activated because the atoms generated so far validate their left hand sides, and collecting their right hand sides.

That yields successively $[\mathcal{P}]_0 = \{p_1, p_2\}$, $[\mathcal{P}]_1 = \{p_1, p_2, p_3\}$, $[\mathcal{P}]_2 = \{p_1, p_2, p_3, p_5\}$, and finally, $[\mathcal{P}]_3 = [\mathcal{P}]$. In a first-order setting where interpretations are standard and might have infinite domains and where rules accommodate existential and universal quantifiers, the iterative process of generating atoms could be transfinite, but there would still be a unique $\subseteq$ -minimal fixed point $[\mathcal{P}]$, equal to $[\mathcal{P}]_{\alpha \in \text{Ord}}$ where Ord denotes the class of ordinals; as Ord is a proper class, $[\mathcal{P}]_\alpha$ is guaranteed to be equal to $\bigcup_{\beta < \alpha} [\mathcal{P}]_\beta$ for some least ordinal α , marking the point at which the whole of $[\mathcal{P}]$ has been generated. What are the key features of this generation process?

- Rules are read from left to right, they operate from the left hand side to the right hand side; the contrapositives of the rules can be ignored.
- At any stage, rules can be considered individually, independently of any others, to determine what to generate at the next stage.
- At any stage, what is generated involves a rule whose left hand side is “validated” with what has been previously generated.

The theme of this paper is that properties (A)–(C) can be preserved for rules that are more interesting and general, and more particularly, for rules with disjunctions on the right hand side; moreover, an adapted form of the cut rule can operate on rules in a way that satisfies properties (A)–(C).

The most elegant presentation of the cut rule is in the sequent calculus, and takes the form

$$(\star) \quad \frac{\varphi_1, \dots, \varphi_n \vdash \psi_1, \dots, \psi_m, \xi \quad \xi, \varphi'_1, \dots, \varphi'_{n'} \vdash \psi'_1, \dots, \psi'_{m'}}{\varphi_1, \dots, \varphi_n, \varphi'_1, \dots, \varphi'_{n'} \vdash \psi_1, \dots, \psi_m, \psi'_1, \dots, \psi'_{m'}}$$

to express that

$$\varphi_1 \wedge \dots \wedge \varphi_n \wedge \varphi'_1 \wedge \dots \wedge \varphi'_{n'} \rightarrow \psi_1 \vee \dots \vee \psi_m \vee \psi'_1 \vee \dots \vee \psi'_{m'}$$

is a logical consequence of

$$\varphi_1 \wedge \dots \wedge \varphi_n \rightarrow \psi_1 \vee \dots \vee \psi_m \vee \xi$$

and

$$\xi \wedge \varphi'_1 \wedge \dots \wedge \varphi'_{n'} \rightarrow \psi'_1 \vee \dots \vee \psi'_{m'}.$$

We will go through intermediate sets of rules and intermediate adaptations of the cut rule till we reach the final form of the cut rule that can be satisfactorily applied to the

sets of rules of the kind that we want to eventually be able to work with. Let us first adapt the cut rule so that it can deal with sets of rules of the form described above, in a way that satisfies properties (A)–(C) above: we let it take the form

$$(\diamond) \quad \frac{\vdash \xi_0 \quad \dots \quad \vdash \xi_k \quad \varphi \vdash \psi}{\vdash \psi}$$

where $k \in \mathbb{N}$, $\xi_0, \dots, \xi_k, \psi$ are atoms, φ is a positive propositional formula with at least one occurrence of each of $\xi_0, \dots, \xi_k$, and $\varphi[\xi_0/\text{true}, \dots, \xi_k/\text{true}]$, that is, the result of making all occurrences of $\xi_0, \dots, \xi_k$ in φ true, is logically valid. This is a big modification of $(\star)$, and in some way also a big simplification, more similar to a generalised modus ponens than to a full cut, but further adaptations will bring us closer to $(\star)$ as we consider sets of rules with disjunction on the right hand side. As an example of an application of $(\diamond)$ for the set of rules $\mathcal{P}$ defined above, p_{10} is added to $[\mathcal{P}]$ by the following application of $(\diamond)$:

$$\frac{\vdash p_1 \quad \vdash p_5 \quad (p_1 \vee p_4) \wedge (p_5 \vee p_8) \vdash p_{10}}{\vdash p_{10}}$$

Let us emphasise the key differences between $(\star)$ and $(\diamond)$, that will be applicable to all further adaptations of the cut rule.

- In $(\star)$, the cut rule has two antecedents. In $(\diamond)$, at least two sequents, but possibly more, make up the antecedents.
- In $(\star)$, the antecedents of the cut rule are arbitrary sequents. In $(\diamond)$, one antecedent of the cut rule is an arbitrary sequent, but all other antecedents are sequents with an empty left hand side.
- In $(\star)$, the formula to which the cut is applied, namely ξ , is one of a number of formulas on the left hand side of a sequent that are implicitly conjuncted. In $(\diamond)$, the formulas $\xi_0, \dots, \xi_k$ to which the cut is simultaneously applied occur in a formula φ that is not necessarily the conjunction of $\xi_0, \dots, \xi_k$, but that is logically implied by that conjunction.
- In $(\star)$, the consequent is an arbitrary sequent. In $(\diamond)$, it is a sequent with an empty left hand side.

2.2 Negation

To see whether negation is problematic, let $\mathcal{P}$ now denote the extension of the set of rules defined above with the following rules.

$$\neg p_{11} \rightarrow \neg p_3 \quad p_{12} \rightarrow p_{13} \quad \neg p_{12} \rightarrow p_{13}$$

One might think that the contrapositive of the first extra rule should let p_{11} join $[\mathcal{P}]$, breaking down property (A) above, and the last two extra rules taken together should let p_{13} join $[\mathcal{P}]$, breaking down properties (B) and (C). But following standard practice in logic programming, we work in a paradigm where disjunction is constructive. For example, given literals $\varphi_1, \varphi_2, \varphi_3$ and φ , the intended meaning of the rule $\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \rightarrow \varphi$ is, in that paradigm: if φ_1 has been generated, and if at least one of φ_2 and φ_3 has been generated, then φ can be generated. A representation of

our set of rules more faithful to that intended meaning uses the modal operator $\Box$ to capture the notion “has been generated” or “has been proved.” And in accordance with the expected meaning of $\Box$, we will work in a logical setting where for all atoms φ , $\Box\varphi \wedge \Box\neg\varphi$ is inconsistent, while $\Box\varphi \vee \Box\neg\varphi$ is satisfiable but not valid. Call *claim* any formula of the form $\Box\varphi$ where φ is a literal, and *claiming condition*⁵ any formula obtained from the set of claims by arbitrary application of conjunction and disjunction. So we now consider sets of rules whose left hand sides are claiming conditions and whose right hand sides are claims.

Following on from our example, let $\mathcal{F}$ denote:⁶

$$\begin{array}{ll} \rightarrow \Box p_1 & \Box p_7 \rightarrow \Box p_8 \\ \rightarrow \Box p_2 & \Box p_5 \wedge \Box p_7 \rightarrow \Box p_9 \\ (\Box p_1 \vee \Box p_5) \wedge \Box p_2 \rightarrow \Box p_3 & (\Box p_1 \vee \Box p_4) \wedge (\Box p_5 \vee \Box p_8) \rightarrow \Box p_{10} \\ (\Box p_1 \wedge \Box p_4) \vee \Box p_6 \rightarrow \Box p_4 & \Box \neg p_{11} \rightarrow \Box \neg p_3 \\ \Box p_3 \rightarrow \Box p_5 & \Box p_{12} \rightarrow \Box p_{13} \\ (\Box p_3 \wedge \Box p_5) \rightarrow \Box p_8 & \Box \neg p_{12} \rightarrow \Box p_{13} \end{array}$$

We also define $[\mathcal{F}]$ as the set of claims that are logical consequences of $\mathcal{F}$. The contrapositive of the implication $\Box \neg p_{11} \rightarrow \Box \neg p_3$ is a formula that is logically equivalent to $\Diamond p_3 \rightarrow \Diamond p_{11}$, which allows one to generate $\Diamond p_{11}$ but not the stronger $\Box p_{11}$, and as $\Box p_{12} \vee \Box \neg p_{12}$ is not valid, $\Box p_{13}$ cannot be generated either. Hence

$$[\mathcal{F}] = \{\Box p_1, \Box p_2, \Box p_3, \Box p_5, \Box p_8, \Box p_{10}\}.$$

Though it uses classical negation and no other form of negation, this modal representation and associated interpretation of a set of rules is suitable to model negation as failure and the main semantics of logic programs. It is easy to see that properties (A)–(C) above are preserved for the kind of rules now under consideration, thanks to the form of the cut rule that has been described as $(\diamond)$ with the only difference that $\xi_0, \dots, \xi_k, \psi$ are claims rather than atoms, and φ is a claiming condition rather than a positive propositional formula.

2.3 Disjunction

Let us now allow disjunction on the right hand side of a rule. Call *alternative* any formula of the form $\varphi_1 \vee \dots \vee \varphi_n$ where $n \in \mathbb{N}$ and $\varphi_1, \dots, \varphi_n$ are pairwise distinct claims, with no a priori reference to any particular rule (when $n = 0$, the alternative is empty). So we now consider sets of rules whose left hand sides are claiming conditions and whose right hand sides are alternatives. Let us extend our running example so that $\mathcal{F}$ now denotes the set of rules above complemented with the following rules.

⁵ We do not say more simply *condition* because we keep that expression for a more general notion, to be introduced later.

⁶ Here the writing of the (modal) rules with bodies on the left hand side and heads on the right hand side is totally legitimate as $\rightarrow$ is classical implication.

$$\begin{array}{ll}
\rightarrow \Box \neg p_3 \vee \Box p_{15} & \rightarrow \Box p_{18} \vee \Box p_{19} \\
\Box p_{15} \rightarrow \Box \neg p_{16} & \Box p_{18} \rightarrow \Box p_{20} \\
\Box p_{10} \rightarrow \Box p_{16} \vee \Box p_{17} & \Box p_{19} \rightarrow \Box p_{20}
\end{array}$$

And now, we let $[\mathcal{F}]$ denote the set of alternatives that are logical consequences of $\mathcal{F}$. Clearly, we then have to add $\Box p_{15}$, $\Box \neg p_{16}$, $\Box p_{17}$ and $\Box p_{20}$ to $[\mathcal{F}]$. But the very idea of rules that read from left to right and fire individually seems to break down. Consider first the claims $\Box p_{15}$ and $\Box p_{17}$. Each of them is inferred from two generated alternatives ($\Box \neg p_3 \vee \Box p_{15}$ and $\Box p_3$, and $\Box p_{16} \vee \Box p_{17}$ and $\Box \neg p_{16}$, respectively), breaking down property (C) above. This is still easily fixed by closing $\mathcal{F}$ under a *left shift operation*, that moves claims from right to left as exemplified below, in a way that will preserve the logical validity of the original set of rules and capture well the reasoning behind the inference of $\Box p_{15}$ from $\Box \neg p_3 \vee \Box p_{15}$ and $\Box p_3$, and the inference of $\Box p_{17}$ from $\Box p_{16} \vee \Box p_{17}$, $\Box \neg p_{16}$ and $\Box p_{10}$:

$$\Box p_3 \rightarrow \Box p_{15} \quad \Box p_{10} \wedge \Box \neg p_{16} \rightarrow \Box p_{17}$$

When we define more formally the syntax of a rule, we will, for good reasons, take disjunction and conjunction as operators on sets. This is why we assumed that all claims that occur in an alternative are pairwise distinct. For an illustration, the fact that $\rightarrow \Box p \vee \Box p$ is not an admissible rule implies that $\Box \neg p \rightarrow \Box p$ is not admissible either (it would have to be obtained from the latter by left shift); that is good, as would $\Box \neg p \rightarrow \Box p$ be admissible, it could be demanded that it lets one derive $\Box p$, but $\Box p$ cannot be derived from $\Box \neg p \rightarrow \Box p$ in a way that satisfies properties (A)–(C) above. Also note that a left shift can move the whole right hand side of a rule to the left. That will allow one to deal with inconsistent sets of rules and reduce any particular contradiction, involving two claims of the form $\Box p$ and $\Box \neg p$, to the “generic” contradiction that emerges when the empty alternative (disjunction without disjunct) is derived, as is the case for instance for a set of rules that contains both $\rightarrow \Box p$ and $\rightarrow \Box \neg p$, with $\Box p \rightarrow$ produced from the latter by left shift.

Now consider the claim $\Box p_{20}$. Here it seems that in order to generate $\Box p_{20}$, it is necessary to consider, together with $\Box p_{18} \vee \Box p_{19}$, both rules $\Box p_{18} \rightarrow \Box p_{20}$ and $\Box p_{19} \rightarrow \Box p_{20}$, breaking down property (B) above. This is the point where the cut rule has to be generalised from $(\diamond)$ to a form that brings it closer to $(\star)$: we now let it take the form

$$(\dagger) \quad \frac{\vdash \xi_0^0, \dots, \xi_0^{n_0} \quad \dots \quad \vdash \xi_k^0, \dots, \xi_k^{n_k} \quad \varphi \vdash \psi_0, \dots, \psi_n}{\vdash \xi_1, \dots, \xi_m, \psi_0, \dots, \psi_n}$$

where

- $k, n_0, \dots, n_k, n, m \in \mathbb{N}$, $\xi_0^0, \dots, \xi_0^{n_0}$ are pairwise distinct claims, $\dots, \xi_k^0, \dots, \xi_k^{n_k}$ are pairwise distinct claims, $\psi_0, \dots, \psi_n$ are pairwise distinct claims,
- φ is a claiming condition with at least one occurrence of each of $\xi_0^{n_0}, \dots, \xi_k^{n_k}$,
- $\varphi[\xi_0^{n_0}/\text{true}, \dots, \xi_k^{n_k}/\text{true}]$ is logically valid, and
- $\xi_1, \dots, \xi_m$ are the pairwise distinct members of $\{\xi_i^j \mid i \leq k, j < n_i\}$ that do not belong to $\{\psi_0, \dots, \psi_n\}$.

For instance, we can add $\Box p_{20}$ to $[\mathcal{F}]$ thanks to two applications of $(\dagger)$:

$$\frac{\frac{\frac{\vdash \Box p_{18} \vee \Box p_{19} \quad \Box p_{18} \vdash \Box p_{20}}{\vdash \Box p_{19} \vee \Box p_{20}} \quad \Box p_{19} \vdash \Box p_{20}}{\vdash \Box p_{20}}$$

One can see that the logically strongest members of $[\mathcal{F}]$ can be obtained by successive applications of $(\dagger)$ on the closure of $\mathcal{F}$ under left shift, validating properties (A)–(C) above.

One might object that the order of the claims on the right hand side of a sequent has to be taken into account, and that it is necessary to add a rule that permutes the various elements of a sequence so that the claims to which the cut is applied can always be last on the right hand side of the corresponding sequents. But this will not be necessary again because disjunction will be treated as an operator over a set: the right hand side of a sequent in $(\dagger)$ is implicitly disjuncted as a set, and is therefore to be conceived of as some arbitrary enumeration of that set, the order of the enumeration being irrelevant. One could therefore write the right hand side of a sequent either as $\{\xi_1, \dots, \xi_n\}$ or as $\bigvee \{\xi_1, \dots, \xi_n\}$ rather than as $\xi_1, \dots, \xi_n$, depending on whether which of making the disjunction implicit or explicit would be preferred.

2.4 Answer sets

Can we consider positive formulas over a set of formulas that is not constrained to containing claims only? As we work in a constructive paradigm, and as for all literals φ , $\Box\varphi \vee \Diamond\neg\varphi$ will be valid, we have to keep formulas of the form $\Diamond\varphi$ out of both sides of the rules. But we will work in a logical framework where for all literals φ , $\Diamond\Box\varphi$ is a logical consequence of $\Box\varphi$, is consistent with $\Diamond\Box\neg\varphi$ and is inconsistent with $\Box\neg\varphi$, and where $\Diamond\Box\varphi \vee \Diamond\Box\neg\varphi$ is not valid. Call *hypothesis* any formula of the form $\Diamond\Box\varphi$ where φ is a literal (so a hypothesis is any formula of the form $\Diamond\varphi$ where φ is a claim). We now motivate why hypotheses are interesting and why it is worth allowing them to occur on the left hand side of the rules, motivated by relationships to the answer set semantics.

The usual presentation of the answer set semantics uses two kinds of negation: classical $\neg$ and nonclassical *not*. The former can only be applied to atoms, and the latter to atoms or classically negated atoms; moreover, *not* can only occur on the left hand side of a rule, and the right hand side of a rule can be either a literal or a disjunction. Such sets of rules are referred to as *extended-disjunctive programs*, and as *extended-normal* in case disjunction does not occur on the right hand side of any rule. One can transform an extended-disjunctive program $\mathcal{R}$ into a set of rules $\mathcal{R}^*$ that uses modalities but not *not*, proceeding as follows.

- Precede all occurrences of literals preceded with neither *not* nor $\neg$ with $\Box$.
- Replace every occurrence of *not* which is not followed by $\neg$ with $\Diamond\Box\neg$.
- Replace every occurrence of *not* $\neg$ with $\Diamond\Box$.

For instance, this technique transforms the following extended-normal program $\mathcal{R}$, already met in Sect. 1.2,

$$\begin{array}{ll}
p_2 \wedge p_3 \rightarrow p_1 & \text{not } \neg p_2 \vee \text{not } p_4 \rightarrow \neg p_1 \\
p_4 \rightarrow p_2 & \neg p_3 \rightarrow \neg p_2 \\
p_3 \rightarrow p_3 & \neg p_3 \wedge \text{not } \neg p_2 \rightarrow \neg p_3 \\
\text{not } p_3 \rightarrow p_4 &
\end{array}$$

into the following set of rules $\mathcal{R}^*$.

$$\begin{array}{ll}
\Box p_2 \wedge \Box p_3 \rightarrow \Box p_1 & \Diamond \Box p_2 \vee \Diamond \Box \neg p_4 \rightarrow \Box \neg p_1 \\
\Box p_4 \rightarrow \Box p_2 & \Box \neg p_3 \rightarrow \Box \neg p_2 \\
\Box p_3 \rightarrow \Box p_3 & \Box \neg p_3 \wedge \Diamond \Box p_2 \rightarrow \Box \neg p_3 \\
\Diamond \Box \neg p_3 \rightarrow \Box p_4 &
\end{array}$$

Call *condition*, with no a priori reference to any particular rule, any formula obtained from the set of claims and hypotheses by arbitrary application of conjunction and disjunction. We have now reached the final form of the rules we want to be able to work with: they have conditions as left hand sides, and alternatives as right hand sides—let us refer to them as *conditional alternatives*.

Given an extended-normal program $\mathcal{R}$, let us examine the relationship between $\mathcal{R}$ and the associated set of conditional alternatives $\mathcal{R}^*$. It is easy to see that if $\mathcal{R}$ is the extended-normal program defined above, then $\mathcal{R}$ has a unique answer set, namely, $\{\neg p_1, p_2, p_4\}$,⁷ and the set of logical consequences of $\mathcal{R}^* \cup \{\Diamond \Box p_2, \Diamond \Box \neg p_3\}$ is $\{\Box \neg p_1, \Box p_2, \Box p_4\}$. We will verify that more generally, given an extended-normal program $\mathcal{R}$ and a set of literals X , X is an answer set for $\mathcal{R}$ iff $\{\Box \varphi \mid \varphi \in X\}$ is the set of all claims that are logical consequences of $\mathcal{R}^* \cup H$ where H is a set of hypotheses with the following properties.

- $\mathcal{R}^* \cup H$ is consistent;
- for all literals ψ , $\Diamond \Box \psi$ belongs to H iff $\Diamond \Box \psi$ occurs in (the conditions of the conditional alternatives in) $\mathcal{R}^*$ and $\Box \neg \psi$ is not a logical consequence of $\mathcal{R}^* \cup H$.

Obviously, the previous relationship between $\mathcal{R}$ and $\mathcal{R}^*$ does not generalise to extended-disjunctive programs. For instance, the extended-disjunctive program consisting of only $\rightarrow p \vee q$ has $\{p\}$ and $\{q\}$ as answer sets, and neither $\{\Box p\}$ nor $\{\Box q\}$ is the set of claims that are logical consequences of $\{\Box p \vee \Box q\}$ complemented with some set of hypotheses. What we need is the left-shift operation described in the previous section, adapted to let claims that move from right to left become hypotheses rather than claims, which still preserves logical validity; let us talk about *hypothetical left shift* to refer to this form of the left shift operator. With $\rightarrow \Box p \vee \Box q$ as example, the hypothetical left-shift generates the three conditional alternatives that follow.

$$\Diamond \Box \neg p \rightarrow \Box q \quad \Diamond \Box \neg q \rightarrow \Box p \quad \Diamond \Box \neg p \wedge \Diamond \Box \neg q \rightarrow$$

The resulting set of conditional alternatives can obviously be complemented with $\{\Diamond \Box \neg p\}$ or $\{\Diamond \Box \neg q\}$ so that the set of claims that are logical consequences of the resulting theory is $\{\Box q\}$ or $\{\Box p\}$, respectively, which captures answer sets along the lines of what we described above in reference to an extended-normal program, but

⁷ That $\{\neg p_1, p_2, p_4\}$ is an answer set for $\mathcal{R}$ has been verified in Sect. 1.2.

where instead of considering $\mathcal{R}^*$, we consider the closure of $\mathcal{R}^*$ under hypothetical left shift. Of course, the conditional alternatives introduced by hypothetical left shift might not be of any use. For instance, the extended-disjunctive program

$$\rightarrow p \vee q \quad p \rightarrow q \quad q \rightarrow p$$

has $\{p, q\}$ as unique answer set, and $\{\Box p, \Box q\}$ is the set of claims that are logical consequences of $\{\Box p \vee \Box q, \Box p \rightarrow \Box q, \Box q \rightarrow \Box p\}$.

We can now formulate the key question that is the object of this paper in full generality.

Let $\mathcal{F}$ be a set of conditional alternatives and H a set of hypotheses. Let $[\mathcal{F}, H]$ denote the set of all alternatives that are logical consequences of $\mathcal{F} \cup H$. Is there a form of the cut rule that can be applied to a closure of $\mathcal{F}$ and H and generate $[\mathcal{F}, H]$, in such a way that properties (A)–(C) discussed at the beginning of the paper hold?

We have claimed that this question can be positively answered in case H is empty and no hypothesis occurs in $\mathcal{F}$, thanks to the first version of the left shift operator and the version of the cut rule given by ($\dagger$). We will see that there is also a positive answer in the general case. The closure of $\mathcal{F}$ and H will be obtained by hypothetical left shift and replacement of all occurrences of the members of H in the conditions of the conditional alternatives so obtained by *true*. The version of the cut rule to be used is what has been described as ($\dagger$), except that φ has to be assumed to be a condition rather than a claiming condition, and the requirement that $\varphi[\xi_0^{n_0}/\text{true}, \dots, \xi_k^{n_k}/\text{true}]$ be logically valid has to be replaced by the requirement that

$$\varphi[\Diamond \xi_0^{n_0}/\text{true}, \dots, \Diamond \xi_k^{n_k}/\text{true}][\xi_0^{n_0}/\text{true}, \dots, \xi_k^{n_k}/\text{true}]$$

be logically valid: we set to *true* in the condition of the conditional alternative that is the target of the cut all hypotheses and claims built from one of the $k + 1$ literals to which the cut applies.

2.5 Dealing properly with substitution and validity

Our last form of the cut rule still leaves to be desired: eliminating the left hand side of the selected conditional alternative by turning it into a valid formula thanks to substitution of claims and hypotheses by *true* is not a mechanical, syntactic, proof-theoretic operation. But we will proceed in a way that addresses this issue satisfactorily. Recall that we intend to take disjunction and conjunction as operators on (possibly empty) sets. We have mentioned already that this has the advantage of making duplicate disjuncted claims a non-issue. Note now that there is no need to introduce a propositional constant *true* as $\bigwedge \emptyset$ is logically valid, and that $\bigvee \emptyset$ is logically invalid. This will be useful to avoid empty left or right hand sides in a conditional alternative, which is formally sloppy. But the key point is that we can replace the requirement that

$$\varphi[\Diamond \xi_0^{n_0}/\text{true}, \dots, \Diamond \xi_k^{n_k}/\text{true}][\xi_0^{n_0}/\text{true}, \dots, \xi_k^{n_k}/\text{true}] \text{ is logically valid}$$

by the requirement that

$\bigwedge \emptyset$ is the result of substituting all occurrences of $\Diamond \xi_0^{n_0}, \dots, \Diamond \xi_k^{n_k}$ and all occurrences of $\xi_0^{n_0}, \dots, \xi_k^{n_k}$ not preceded by $\Diamond$ in φ by $\bigwedge \emptyset$, collapsing conjunctions, collapsing disjunctions, letting $\bigwedge \emptyset$ absorb enclosing disjunctions, and letting $\bigvee \emptyset$ absorb enclosing conjunctions.

For an example, let φ be

$$\bigwedge \left\{ \Box p_1, \bigwedge \left\{ \Box p_2, \bigvee \left\{ \Box p_3, \Box p_4, \Box p_5 \right\} \right\}, \bigvee \left\{ \Box p_6, \bigvee \left\{ \Box p_6, \Box p_7 \right\} \right\} \right\}.$$

When one replaces in φ the claims $\Box p_1, \Box p_2, \Box p_4$ and $\Box p_7$ by $\bigwedge \emptyset$ and successively applies the transformations described above, one obtains

$$\bigwedge \left\{ \bigwedge \emptyset, \bigvee \left\{ \Box p_3, \bigwedge \emptyset, \Box p_5 \right\}, \bigvee \left\{ \Box p_6, \bigwedge \emptyset \right\} \right\},$$

$\bigwedge \left\{ \bigwedge \emptyset \right\}$ and eventually $\bigwedge \emptyset$. What we have described is a mechanical, syntactic, proof-theoretic way guaranteed to derive $\bigwedge \emptyset$ from a condition in which the occurrences of some claims and hypotheses have been replaced by $\bigwedge \emptyset$ whenever the resulting formula logically follows from the set of those claims and hypotheses.

3 Logical background

3.1 Claims, hypotheses, and disjunctive programs

$\mathbb{N}$ denotes the set of natural numbers and Ord the class of ordinals.

Definition 3.1 A *vocabulary* is a countable set of nullary predicate symbols.

Notation 3.2 We denote by $\mathcal{V}$ a vocabulary.

Members of $\mathcal{V}$ are called *atoms* (over $\mathcal{V}$). Members of $\mathcal{V}$ and negations of members of $\mathcal{V}$ are called *literals* (over $\mathcal{V}$). Given a literal φ , we let $\sim \varphi$ denote $\neg \varphi$ if φ is an atom, and ψ if φ is of the form $\neg \psi$.

Definition 3.3 The set of *conditions* (over $\mathcal{V}$) is inductively defined as the smallest set that satisfies the following conditions.

- All expressions of the form $\Box \varphi$ with φ a literal over $\mathcal{V}$, are conditions.
- All expressions of the form $\Diamond \Box \varphi$ with φ a literal over $\mathcal{V}$, are conditions.
- All expressions of the form $\bigvee X$ with X a countable set of conditions over $\mathcal{V}$, are conditions.
- All expressions of the form $\bigwedge X$ with X a finite set of conditions over $\mathcal{V}$, are conditions.

Definition 3.4 We call *claim* (over $\mathcal{V}$) any condition over $\mathcal{V}$ of the form $\Box \varphi$.

We call *hypothesis* (over $\mathcal{V}$) any condition over $\mathcal{V}$ of the form $\Diamond \Box \varphi$.

We call *stance* (over $\mathcal{V}$) any claim or hypothesis over $\mathcal{V}$.

A few remarks about Definition 3.3 are in order. First, note that all conditions are in negation normal form: negation can be applied to atoms only. Second, note that disjunction and conjunction can be applied to the empty set, yielding a logically invalid and a logically valid formula, respectively. Third, note that contrary to conjunction, disjunction can be applied to an infinite set. The motivation is that it will be formally advantageous to group together all rules that have a common alternative: rather than considering a set of rules of the form $\{\Box p_i \rightarrow \Box q \mid i \in \mathbb{N}\}$, we will prefer the single rule $\bigvee\{\Box p_i \mid i \in \mathbb{N}\} \rightarrow \Box q$. Infinite vocabularies and infinite sets of rules are natural if one thinks of propositionalising a set of first order (modal) rules. For instance, the previous set of rules could be obtained by propositionalising the first-order rule $\exists x \Box p(x) \rightarrow \Box q(\bar{0})$ in a setting where all intended interpretations are standard and the set of closed terms is equal to the set of numerals $\{\bar{n} \mid n \in \mathbb{N}\}$; then one would map $p(\bar{n})$ to p_n for all $n \in \mathbb{N}$, $q(\bar{0})$ to q , and obtain the former set of rules as an alternative representation. If we worked in a first-order language with standard structures as intended interpretations, that language could sometimes be kept finite thanks to function symbols when infinitely many nullary predicate symbols are needed to perform the propositionalisation. As this would bring no significant difference in the results or in their proofs, we opt for the simpler formulation of an infinite propositional language. So infinite sets of rules are natural objects of study, and disjunctions that apply to infinite sets are natural tools. Moreover, disjunctions can be assumed to operate on countably infinite sets without affecting any of the formal developments. On the other hand, many results would break down if conjunction was allowed to operate on infinite sets.

Given $n \in \mathbb{N}$ and conditions $\varphi_1, \dots, \varphi_n$, we use $\varphi_1 \vee \dots \vee \varphi_n$ and $\varphi_1 \wedge \dots \wedge \varphi_n$ as abbreviations for $\bigvee\{\varphi_i \mid 1 \leq i \leq n\}$ and $\bigwedge\{\varphi_i \mid 1 \leq i \leq n\}$, respectively.

Notation 3.5 We denote by $\text{Alt}(\mathcal{V})$ the set of finite sets of literals over $\mathcal{V}$.

As we know from Sect. 2, we will consider rules whose right hand sides are of the form $\bigvee \Box D$ for some member D of $\text{Alt}(\mathcal{V})$, where $\Box D$ is defined next.

Notation 3.6 For all $D \in \text{Alt}(\mathcal{V})$, we let $\sim D$ denote $\{\sim \varphi \mid \varphi \in D\}$, $\Box D$ denote $\{\Box \varphi \mid \varphi \in D\}$, and $\Diamond \Box D$ denote $\{\Diamond \Box \varphi \mid \varphi \in D\}$.

If, as explained before, one chooses to disjunct the conditions of all rules that have a common alternative, one is then led to define the sets of rules that are our object of study as follows, using either an “implicit” representation (Definition 3.7, mapping to a unique condition the set of literals that once disjuncted, make up an alternative) or an “explicit” representation (Definition 3.9, putting the mapping in proper logical form).

Definition 3.7 We define a *disjunctive program* (over $\mathcal{V}$) as an $\text{Alt}(\mathcal{V})$ -family of conditions over $\mathcal{V}$.

Definition 3.8 We call *conditional alternative* (over $\mathcal{V}$) any expression which is of the form $\varphi \rightarrow \bigvee \Box D$ where φ is a condition over $\mathcal{V}$ and D a member of $\text{Alt}(\mathcal{V})$; we call φ the *condition of the conditional alternative* and $\bigvee \Box D$ the *alternative of the conditional alternative*.

Definition 3.9 Let a disjunctive program $\mathcal{F} = (\varphi_D)_{D \in \text{Alt}(\mathcal{V})}$ be given.

The *theory associated with $\mathcal{F}$* is defined as the set of conditional alternatives

$$\left\{ \varphi_D \rightarrow \bigvee \Box D \mid D \in \text{Alt}(\mathcal{V}) \right\}.$$

Notation 3.10 Given a disjunctive program $\mathcal{F}$, we let $\text{Th}(\mathcal{F})$ denote the theory associated with $\mathcal{F}$.

For instance, suppose that $\mathcal{V} = \{p_1, p_2, p_3, p_4\}$. Then $D = \{p_3, \neg p_4\}$ is a member of $\text{Alt}(\mathcal{V})$, and φ_D could be $(\Box p_1 \wedge \Box p_2) \vee \Diamond \Box \neg p_4$, in which case $\mathcal{F}$ would have as rule $(\Box p_1 \wedge \Box p_2) \vee \Diamond \Box \neg p_4 \rightarrow \Box p_3 \vee \Box \neg p_4$. That would formalise an extended-disjunctive program having as rules $p_1 \wedge p_2 \rightarrow p_3 \vee \neg p_4$ and *not* $p_4 \rightarrow p_3 \vee \neg p_4$, and no other rule with $p_3 \vee \neg p_4$ as head. All rules with the same head are grouped together (thanks to potentially infinite disjunctions) in $\mathcal{F}$. In practice, for most members D' of $\text{Alt}(\mathcal{V})$, $\varphi_{D'}$ would be $\bigvee \emptyset$ as $\mathcal{F}$ would model an extended-disjunctive program with no rule with the disjunctions of the members of D' as head (with $\mathcal{V} = \{p_1, p_2, p_3, p_4\}$, $\text{Alt}(\mathcal{V})$ is of cardinality 2^8).

Notation 3.11 Given a disjunctive program $\mathcal{F} = (\varphi_D)_{D \in \text{Alt}(\mathcal{V})}$, we let $\text{Hyp}(\mathcal{F})$ denote the set of hypotheses that occur in $\{\varphi_D \mid D \in \text{Alt}(\mathcal{V})\}$.

3.2 Semantics

The formulas that make up the theory associated with a disjunctive program are very specific: they are implications both sides of which do not contain occurrences of a formula of the form $\Diamond \varphi$ for some literal φ , where no modal operator is in the scope of another modal operator except for hypotheses, etc. This implies that we do not need to develop a complete semantics for a set of formulas closed under modal and boolean operators. We opt for keeping the concepts to a minimal, and in particular, avoid to resort to Kripke frames or similar semantic objects. Instead, we restrict the notion of logical consequence we will work with to that part of the language that we strictly need. But it is perfectly possible to embed the notions defined in this section into a full fledged semantics, and this is done in [16].

Definition 3.12 Let X be a set of stances. We say that X is *consistent* just in case for all literals φ , if $\Box \varphi \in X$ then neither $\Box \sim \varphi$ nor $\Diamond \Box \sim \varphi$ belongs to X ; otherwise we say that X is *inconsistent*.

Definition 3.13 A set X of stances is *closed* just in case it is consistent and for all literals φ , if $\Box \varphi \in X$ then $\Diamond \Box \varphi$ belongs to X .

Essentially, the notion of a closed set of stances provides a convenient, syntactic definition of logical consequence between a set of stances (used as antecedent) and a stance (used as consequent) which suffices for our purposes: we can think of a stance ψ to be a logical consequence of a set X of stances iff ψ belongs to the closure of X (that is, the $\subseteq$ -smallest set of stances which is closed and contains X).

The first part of next definition is motivated by the relationship this framework bears to the answer set semantics, as sketched in Sect. 2. The second part fulfils a different purpose: intuitively, a complete set of stances is the set of all stances that are true at a particular point in a suitable Kripke frame, which suffices to determine the truth of any condition or conditional alternative at that point, hence which suffices to define a notion of logical consequence restricted to the language of hypotheses and of the theory associated with a disjunctive program.

Definition 3.14 Let H be a set of hypotheses. A set X of stances is *H-complete* just in case it is consistent and for all literals φ with $\Diamond\Box\varphi \in H$,

$$\Diamond\Box\varphi \in X \text{ iff } \Box\sim\varphi \notin X.$$

A set X of stances is *complete* just in case, denoting by H the set of all hypotheses, X is *H-complete*.

Property 3.15 *A complete set of stances is closed.*

Notation 3.16 *We denote by $\mathcal{S}$ the set of all closed sets of stances (over $\mathcal{V}$).*

The next definition exploits the remark that precedes Definition 3.14. If we were not in a modal setting, we could think of a member of $\mathcal{S}$ as the atomic diagram of a standard structure that determines the truth value of any sentence in that structure. Here we can think of a member of $\mathcal{S}$ as the “stance diagram” of a point of a Kripke frame that determines the truth value of any condition or conditional alternative at that point.

Definition 3.17 Let a member $\mathfrak{S}$ of $\mathcal{S}$ be given.

For φ a condition, we say that $\mathfrak{S}$ *forces* φ and write $\mathfrak{S} \Vdash \varphi$ iff:

- when φ is a stance, $\varphi \in \mathfrak{S}$;
- when φ is of the form $\bigvee X$, $\mathfrak{S}$ forces some member of X ;
- when φ is of the form $\bigwedge X$, $\mathfrak{S}$ forces all members of X .

For all conditional alternatives φ , we say that $\mathfrak{S}$ *forces* φ , denoted $\mathfrak{S} \Vdash \varphi$, iff either $\mathfrak{S}$ does not force the condition of φ or $\mathfrak{S}$ forces the alternative of φ .

If $\mathfrak{S}$ does not force a condition or a conditional alternative φ then we write $\mathfrak{S} \nVdash \varphi$.

Given a set T of conditions or conditional alternatives, we write $\mathfrak{S} \Vdash T$ if $\mathfrak{S}$ forces all members of T , and $\mathfrak{S} \nVdash T$ otherwise.

Definition 3.18 Given two sets of conditions or conditional alternatives T and X , we say that X is a *logical $\mathcal{S}$ -consequence* of T , or that T *logically $\mathcal{S}$ -implies* X , and we write $T \models_{\mathcal{S}} X$, just in case every member of $\mathcal{S}$ that forces T forces X .

If a set of conditions or conditional alternatives T does not logically $\mathcal{S}$ -imply a set of conditions or conditional alternatives X then we write $T \not\models_{\mathcal{S}} X$.

The same terminology and notation applies if one or both sets of conditions are replaced by a condition or a conditional alternative.

Definition 3.19 Given two sets of conditions or conditional alternatives T_1 and T_2 , we say that T_1 and T_2 are *logically $\mathcal{S}$ -equivalent* just in case T_1 logically $\mathcal{S}$ -implies T_2 and T_2 logically $\mathcal{S}$ -implies T_1 .

Definition 3.20 Let a disjunctive program $\mathcal{F}$ and a set H of hypotheses. We say that $\mathcal{F}$ is $\mathcal{S}$ -consistent with H just in case some member of $\mathcal{S}$ forces $H \cup \text{Th}(\mathcal{F})$; otherwise we say that $\mathcal{F}$ is $\mathcal{S}$ -inconsistent with H .

Definition 3.21 A condition is said to be $\mathcal{S}$ -valid iff it is logically $\mathcal{S}$ -equivalent to $\bigwedge \emptyset$.

3.3 Remarks

Let a disjunctive program $\mathcal{F} = (\varphi_D)_{D \in \text{Alt}(\mathcal{V})}$ be given.

First, suppose that $\varphi_D = \bigvee \emptyset$ for all members D of $\text{Alt}(\mathcal{V})$ that contain at least one negated atom. Also suppose that for all members D of $\text{Alt}(\mathcal{V})$, φ_D contains no occurrence of a hypothesis and no occurrence of a claim of the form $\Box \neg p$. So $\mathcal{F}$ formalises a logic program whose rules are all of the form

$$p_{i_1} \wedge \cdots \wedge p_{i_n} \rightarrow p_{j_1} \vee \cdots \vee p_{j_m}$$

where $p_{i_1}, \dots, p_{i_n}, p_{j_1}, \dots, p_{j_m}$ are atoms. Such are the logic programs considered in [17]. They are given a semantics, in terms of the derivation of disjunctions of the form $p_{k_1} \vee \cdots \vee p_{k_p}$, that is precisely captured by the notion of a disjunction of the form $\Box p_{k_1} \vee \cdots \vee \Box p_{k_p}$ being a logical $\mathcal{S}$ -consequence of $\text{Th}(\mathcal{F})$. This is an immediate consequence of a very special case of Proposition 6.10, proved in Sect. 6: the proof system used to establish the validity of Proposition 6.10 corresponds to the procedure that defines the semantics presented in [17].

Second, suppose that for all members D of $\text{Alt}(\mathcal{V})$, φ_D contains no occurrence of a hypothesis. So $\mathcal{F}$ formalises a logic program whose rules are all of the form

$$l_{i_1} \wedge \cdots \wedge l_{i_n} \rightarrow l_{j_1} \vee \cdots \vee l_{j_m}$$

where $l_{i_1}, \dots, l_{i_n}, l_{j_1}, \dots, l_{j_m}$ are literals. Then the notion of a disjunction of the form $\Box l_{k_1} \vee \cdots \vee \Box l_{k_p}$ being a logical $\mathcal{S}$ -consequence of $\text{Th}(\mathcal{F})$ also captures the semantics described in [17], but generalised to the logic programs where classical negation is accepted in the (bodies and heads) of the rules. Moreover, for all sets H of hypotheses and for all disjunctions φ of the form $\Box l_{k_1} \vee \cdots \vee \Box l_{k_p}$, φ is a logical $\mathcal{S}$ -consequence of $\text{Th}(\mathcal{F})$ iff φ is a logical $\mathcal{S}$ -consequence of $H \cup \text{Th}(\mathcal{F})$: hypotheses do not increase logical power when dealing with disjunctive programs that formalise logic program with classical negation but without *not*.

Third, imposing no condition on $\mathcal{F}$, the following holds. Let $\mathcal{F}' = (\varphi'_D)_{D \in \text{Alt}(\mathcal{V})}$ be the disjunctive program such that for all members D of $\text{Alt}(\mathcal{V})$, φ'_D is φ_D with any occurrence of a hypothesis of the form $\Diamond \Box l$ being replaced by $\Box l$. Then for all disjunctions φ of the form $\Box l_{k_1} \vee \cdots \vee \Box l_{k_p}$, φ is a logical $\mathcal{S}$ -consequence of $\text{Th}(\mathcal{F})$ iff φ is a logical $\mathcal{S}$ -consequence of $\text{Th}(\mathcal{F}')$: disjunctive programs that formalise logic programs in which *not* can occur in the bodies of the rules would not add logical power, if hypotheses were not introduced too as “complementary axioms”. Proposition 6.10 will embody the full generality of the proposed framework.

4 Left shift completions and answer sets

The next definition captures the notion of hypothetical left shift discussed in Sect. 2, here defined up to logical $\mathcal{S}$ -equivalence, which suffices for our purposes. A left shift completion of a disjunctive program $\mathcal{F}$ should be thought of as the closure of $\mathcal{F}$ under hypothetical left shift.

Definition 4.1 Let a disjunctive program $\mathcal{F} = (\varphi_D)_{D \in \text{Alt}(\mathcal{V})}$ be given. A *left shift completion* of $\mathcal{F}$ is a disjunctive program $\mathcal{F}' = (\varphi'_D)_{D \in \text{Alt}(\mathcal{V})}$ such that for all $D \in \text{Alt}(\mathcal{V})$, φ'_D is logically $\mathcal{S}$ -equivalent to

$$\bigvee \left\{ \bigwedge \{ \varphi_{D'} \} \cup \Diamond \Box \sim (D' \setminus D) \mid D' \supseteq D \right\}$$

For instance, suppose that $\mathcal{V} \subseteq \{p_1, p_2, p_3, p_4\}$, and $\varphi_{\{\neg p_2, \neg p_3, p_4\}}$, $\varphi_{\{p_1, \neg p_2, p_4\}}$ and $\varphi_{\{p_1, \neg p_2, \neg p_3, p_4\}}$ are different to $\bigvee \emptyset$ because $\mathcal{F}$ models an extended-disjunctive program that has rules with $\neg p_2 \vee \neg p_3 \vee p_4$ as head, rules with $p_1 \vee \neg p_2 \vee p_4$ as head, and rules with $p_1 \vee \neg p_2 \vee \neg p_3 \vee p_4$ as head, and no other rule whose head has at least $\neg p_2$ and p_4 as disjuncts. Then the theory associated with a left shift completion of $\mathcal{F}$ would have as conditional alternative (a formula equivalent to):

$$\begin{aligned} & \varphi_{\{\neg p_2, p_4\}} \wedge \left((\varphi_{\{\neg p_2, \neg p_3, p_4\}} \wedge \Diamond \Box p_3) \wedge (\varphi_{\{p_1, \neg p_2, p_4\}} \wedge \Diamond \Box \neg p_1) \wedge \right. \\ & \left. (\varphi_{\{p_1, \neg p_2, \neg p_3, p_4\}} \wedge \Diamond \Box \neg p_1 \wedge \Diamond \Box p_3) \right) \rightarrow \Box \neg p_2 \vee \Box p_4 \end{aligned}$$

Proposition 4.2 Two disjunctive programs such that one is a left shift completion of the other are logically $\mathcal{S}$ -equivalent.

Proof Let disjunctive programs $\mathcal{F} = (\varphi_D)_{D \in \text{Alt}(\mathcal{V})}$ and $\mathcal{F}' = (\varphi'_D)_{D \in \text{Alt}(\mathcal{V})}$ be such that $\mathcal{F}'$ is a left shift completion of $\mathcal{F}$. Let $D \in \text{Alt}(\mathcal{V})$ be given. Since φ'_D is logically $\mathcal{S}$ -equivalent to a condition of the form $\bigvee \{ \varphi_D \} \cup X$, $\varphi'_D \rightarrow \bigvee \Box D$ logically $\mathcal{S}$ -implies $\varphi_D \rightarrow \bigvee \Box D$. Moreover, for all members D' of $\text{Alt}(\mathcal{V})$ with $D' \supset D$, $\{ \bigwedge \Diamond \Box \sim (D' \setminus D), \bigvee \Box (D' \setminus D) \}$ is inconsistent. Hence $\{ \varphi_{D'} \rightarrow \bigvee \Box D' \mid D' \supseteq D \}$ logically $\mathcal{S}$ -implies $\varphi'_D \rightarrow \bigvee \Box D$. We conclude that $\mathcal{F}$ and $\mathcal{F}'$ are logically $\mathcal{S}$ -equivalent. $\square$

Recall the illustrated definition of an answer set at the beginning of Sect. 1.2. The definition of an answer set for an extended-disjunctive program is exactly the same; the fact that rule heads can be disjunctions rather than simple literals making absolutely no difference. That is, given an extended-disjunctive program $\mathcal{R}$, an answer set for $\mathcal{R}$ is a consistent set X of literals with the following property. Replace in (the bodies of the rules in) $\mathcal{R}$ all formulas of the form *not* ψ by *true* if $\psi \notin X$, by *false* otherwise, and simplify, resulting in a set of rules, say $\mathcal{R}'$. Now for all atoms α , replace in $\mathcal{R}'$ all occurrences of $\neg \alpha$ by a new atom α^* , resulting in a new set of rules, say $\mathcal{R}''$. Then the set X^* of members of X where all negated atoms of the form $\neg \alpha$ have been replaced by α^* , is the set of logical consequences of $\mathcal{R}''$.

In Sect. 2.4, we saw how an answer set program $\mathcal{R}$ can be put in correspondence with a disjunctive program $\mathcal{R}^*$. Clearly, every disjunctive program is of the form $\mathcal{R}^*$ for some unique extended-disjunctive program $\mathcal{R}$ (generalised to allow countable disjunctions on the left hand sides of the rules). Hence answer sets can be defined on the basis of disjunctive programs rather than on the basis of extended-disjunctive programs, resulting in the definition that follows.

Definition 4.3 Let a disjunctive program $\mathcal{F}$ be given. An *answer set* for $\mathcal{F}$ is the set of claims in a member $\mathfrak{S}$ of $\mathcal{S}$ with the following properties.

- $\mathfrak{S}$ forces $\text{Th}(\mathcal{F})$ and is $\text{Hyp}(\mathcal{F})$ -complete.
- For all $\mathfrak{T} \in \mathcal{S}$, if $\mathfrak{T}$ forces $\text{Th}(\mathcal{F})$ and is $\text{Hyp}(\mathcal{F})$ -complete then the set of claims in $\mathfrak{T}$ is not strictly included in the set of claims in $\mathfrak{S}$.

It is easily verified that Definition 4.3 is equivalent to the standard definition, previously outlined, of an answer set. The verification amounts to rewriting an extended-disjunctive program $\mathcal{R}$ as a disjunctive program $\mathcal{F}$, essentially rewriting an occurrence in $\mathcal{R}$ of a literal α not preceded by *not* as $\Box\alpha$ in $\mathcal{F}$, and rewriting an occurrence in $\mathcal{R}$ of a formula of the form *not* α or *not* $\neg\alpha$, with α an atom, as $\Diamond\Box\neg\alpha$ or $\Diamond\Box\alpha$ in $\mathcal{F}$, respectively. In effect, Definition 4.3 is nothing but the standard definition of an answer set modulo the mechanical translation, the mechanical rewriting, of an extended-disjunctive program as a disjunctive program.

To illustrate the need for the second condition in Definition 4.3 with a trivial example (that does not even need disjunction), the logic program *not* $\neg p \rightarrow q$, which has $\{q\}$ as unique answer set, corresponds in our framework to a disjunctive program $\mathcal{F}$ with $\text{Th}(\mathcal{F}) = \{\Diamond\Box p \rightarrow \Box q\}$ and $\text{Hyp}(\mathcal{F}) = \{\Diamond\Box p\}$. Then $\mathfrak{S} = \{\Diamond\Box p, \Box q\}$ satisfies both conditions of the previous definition, whereas $\{\Diamond\Box p, \Box q, \Box p\}$ and $\{\Diamond\Box p, \Box q, \Box r\}$ satisfy only the first one (and have to be ruled out as $\{q, r\}$ and $\{q, p\}$ are not answer sets for *not* $\neg p \rightarrow q$). Note that the “standard” left shift completion of a disjunctive program moves all subsets of literals in the head of a given clause to the left. For instance, the logic program $\{p \rightarrow q, q \rightarrow p, p \vee q\}$ has as left shift completion:

$$\begin{aligned} (\Box p \wedge \Diamond\Box\neg q) \vee (\Box q \wedge \Diamond\Box\neg p) \vee (\Diamond\Box\neg p \wedge \Diamond\Box\neg q) &\rightarrow \bigvee \emptyset \\ \Box q \vee \Diamond\Box\neg q &\rightarrow \Box p \\ \Box p \vee \Diamond\Box\neg p &\rightarrow \Box q \\ \bigwedge \emptyset &\rightarrow \Box p \vee \Box q \end{aligned}$$

with $\{\Box p, \Box q\}$ as unique answer set.

The correspondence, discussed in Sect. 2, between answer sets and the sets of claims that are logical $\mathcal{S}$ -consequences of the theory associated with an associated disjunctive program complemented with a set of hypotheses constrained in a particular way, can now be fully formalised and established.

Definition 4.4 Let a disjunctive program $\mathcal{F}$ and a subset H of $\text{Hyp}(\mathcal{F})$ be given. We say that H is a *complete hypothetical extension* for $\mathcal{F}$ iff the set of stances φ such that $\text{Th}(\mathcal{F}) \cup H \models_{\mathcal{S}} \varphi$ is $\text{Hyp}(\mathcal{F})$ -complete.

Proposition 4.5 *Let a disjunctive program $\mathcal{F}$, a left shift completion $\mathcal{F}'$ of $\mathcal{F}$, and a set X of claims be given. The following conditions are equivalent.*

- X is an answer set for $\mathcal{F}$.
- There exists a complete hypothetical extension H for $\mathcal{F}'$ such that X is the set of claims that are logical $\mathcal{S}$ -consequences of $\text{Th}(\mathcal{F}') \cup \{H\}$.

Proof Suppose that X is an answer set for $\mathcal{F}$. Let $\mathfrak{S}$ be a member of $\mathcal{S}$ that satisfies both items in Definition 4.3 and such that X is the set of claims in $\mathfrak{S}$. Let H be the set of all hypotheses $\Diamond\Box\varphi$ in $\text{Hyp}(\mathcal{F}')$ such that $\mathfrak{S} \not\models \Box\sim\varphi$. Note that $H \cap \text{Hyp}(\mathcal{F})$ is equal to the set of hypotheses in $\mathfrak{S} \cap \text{Hyp}(\mathcal{F})$. Hence $\mathfrak{S} \cup H$ forces $\text{Th}(\mathcal{F}) \cup H$, and by Proposition 4.2, also forces $\text{Th}(\mathcal{F}') \cup H$. Moreover, for all $\mathfrak{T} \in \mathcal{S}$ that force $\text{Th}(\mathcal{F}') \cup H$, $\mathfrak{T}$ is $\text{Hyp}(\mathcal{F})$ -complete, hence $X \subseteq \mathfrak{T}$. Hence X is the set of claims that are logical $\mathcal{S}$ -consequences of $\text{Th}(\mathcal{F}') \cup H$, and the set of stances φ such that $\text{Th}(\mathcal{F}') \cup H \models_{\mathcal{S}} \varphi$ is trivially $\text{Hyp}(\mathcal{F}')$ -complete.

Conversely, let H be a complete hypothetical extension for $\mathcal{F}'$ such that X is the set of claims that are logical $\mathcal{S}$ -consequences of $\text{Th}(\mathcal{F}') \cup H$. Obviously, $X \cup H$ belongs to $\mathcal{S}$, forces $\text{Th}(\mathcal{F})$ by Proposition 4.2, and is $\text{Hyp}(\mathcal{F})$ -complete since $\text{Hyp}(\mathcal{F})$ is a subset of $\text{Hyp}(\mathcal{F}')$. Also, every member $\mathfrak{T}$ of $\mathcal{S}$ that forces $\text{Th}(\mathcal{F})$ and contains H is such that all members of X are claims in $\mathfrak{T}$. Hence X is an answer set for $\mathcal{F}$. $\square$

5 Tableau proofs

5.1 General strategy

We aim to show that given a disjunctive program $\mathcal{F}$ and a set H of hypotheses, a modification of the cut rule can be applied to H and any left shift completion of $\mathcal{F}$ and generate all alternatives that are logical $\mathcal{S}$ -consequences of $\text{Th}(\mathcal{F}) \cup H$, in such a way that properties (A)–(C) listed in Sect. 2 are satisfied. To this aim, we first introduce an intermediate proof system and demonstrate that it is complete; we refer to a proof in this system as a tableau proof. Then we will see how a tableau proof can be translated into a proof by cuts.

Tableau proofs are best represented as trees. Say that we try and derive an alternative of the form $\bigvee \Box D$, $D \in \text{Alt}(\mathcal{V})$, from a disjunctive program $\mathcal{F}$ and a set of hypotheses H . We build a tree T whose nodes are labeled with claims, except for the root and possibly some of the leaves. Let N be a node in T that has not been declared to be a leaf, and let X be the set of claims that label the nodes on the path from the root of T up to N . We try and select a conditional alternative R in the theory associated with a left shift completion of $\mathcal{F}$ whose condition is seen to be a logical $\mathcal{S}$ -consequence of $H \cup X$. If R 's alternative is the empty disjunction, then N is given a nonlabelled child that is declared to be a leaf. If R 's alternative is of the form $\bigvee \{\varphi_1, \dots, \varphi_n\}$ for some $n > 0$ and pairwise distinct claims $\varphi_1, \dots, \varphi_n$, then N is given n children labeled $\varphi_1, \dots, \varphi_n$ and every child that receives a member of D as label is declared to be a leaf. If the construction eventually stops and results in a finite tree not consisting of its root only (this is guaranteed in case $\mathcal{F}$ is finite), then all leaves are unlabelled or labelled with nothing but members of D . For the tree to represent a successful tableau proof,

D should be empty, in which case $\text{Th}(\mathcal{F})$ is $\mathcal{S}$ -inconsistent with H , or at least one leaf should be labelled with a member of D . We will verify that it is always possible to build such a tree whenever $\bigvee D$ is a logical consequence of $\text{Th}(\mathcal{F}) \cup H$, hence that the tableau proof procedure is complete.

5.2 Example

To illustrate both tableau proofs and proofs by cut, consider a disjunctive program $\mathcal{F}$ over the vocabulary $\mathcal{V} = \{p_0, \dots, p_{10}\}$ such that $\text{Th}(\mathcal{F})$ is logically $\mathcal{S}$ -equivalent to the set consisting of

$$(\Box p_3 \wedge (\Box p_5 \vee \Box \neg p_5)) \vee (\Box p_3 \wedge \Box p_6) \vee (\Box p_4 \wedge \Box \neg p_1) \vee \Box \neg p_4 \rightarrow \Box p_0$$

and all conditional alternatives that follow.

$$\begin{aligned} & \bigwedge \emptyset \rightarrow \Box p_1 \vee \Box \neg p_2 \vee \Box p_3 \vee \Box p_4 \vee \Box \neg p_4 \\ & \Box p_4 \vee \Box \neg p_4 \rightarrow \Box \neg p_1 \vee \Box p_9 \vee \Box p_{10} \\ & \bigwedge \emptyset \rightarrow \Box p_2 \vee \Box p_3 \\ & \Box p_1 \rightarrow \Box \neg p_2 \\ & \bigwedge \emptyset \rightarrow \Box \neg p_3 \vee \Box p_5 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7 \\ & \Box p_8 \rightarrow \Box \neg p_3 \vee \Box \neg p_7 \\ & \Box p_6 \vee \Box p_7 \rightarrow \Box \neg p_4 \vee \Box p_8 \\ & \Box p_4 \wedge \Box p_{10} \rightarrow \Box p_9 \\ & \bigwedge \emptyset \rightarrow \Box \neg p_9 \end{aligned}$$

It is easy to verify that $\Box p_0$ is a logical $\mathcal{S}$ -consequence of $\text{Th}(\mathcal{F})$ (it will be demonstrated in the detailed illustration of Sect. 6.2).

Let $(\varphi'_D)_{D \in \text{Alt}(\mathcal{V})}$ be a left shift completion of $\mathcal{F}$. It is clear that there is no point in left shifting a claim, say $\Box \varphi$, from the right hand side to the left hand side of one of $\text{Th}(\mathcal{F})$'s conditional alternatives if $\Box \sim \varphi$ does not occur on the right hand side of any other conditional alternative of $\text{Th}(\mathcal{F})$. To simplify the matter further, let us ignore all left shifts that are not involved in the derivation of $\Box p_0$ from $\mathcal{F}$ thanks to the tableau proof tree we are about to present. This allows us not to describe φ'_D for all $D \in \text{Alt}(\mathcal{V})$, but only provide, for some members D of $\text{Alt}(\mathcal{V})$, a condition that is logically $\mathcal{S}$ -equivalent to the disjunction of φ_D with the conditions of the form $\bigwedge \Diamond \Box \sim (D' \setminus D)$ for those strict supersets D' of D , if any, such that the conditional alternative $\varphi_{D'} \rightarrow \bigvee \Box (D' \setminus D)$ turns out to be useful. These considerations lead to writing down 9 relations of logical $\mathcal{S}$ -consequence:

$$(0) \quad (\Box p_3 \wedge (\Box p_5 \vee \Box \neg p_5)) \vee (\Box p_3 \wedge \Box p_6) \vee (\Box p_4 \wedge \Box \neg p_1) \vee \Box \neg p_4 \models_{\mathcal{S}} \varphi'_{\{p_0\}}$$

and

$$(\Box p_8 \wedge \Diamond \Box p_3 \wedge \Diamond \Box p_7) \vee (\Box p_4 \wedge \Box p_{10} \wedge \Diamond \Box \neg p_9) \models_{\mathcal{S}} \varphi'_{\emptyset} \quad (\text{I})$$

$$\bigwedge \emptyset \models_{\mathcal{S}} \varphi'_{\{p_1, \neg p_2, p_3, p_4, \neg p_4\}} \quad (\text{II})$$

$$(\Box p_4 \vee \Box \neg p_4) \wedge \Diamond \Box \neg p_9 \models_{\mathcal{S}} \varphi'_{\{\neg p_1, p_{10}\}} \quad (\text{III})$$

$$\Box p_1 \models_{\mathcal{S}} \varphi'_{\{\neg p_2\}} \quad (\text{IV})$$

$$\Diamond \Box \neg p_2 \models_{\mathcal{S}} \varphi'_{\{p_3\}} \quad (\text{V})$$

$$\Box p_6 \vee \Box p_7 \models_{\mathcal{S}} \varphi'_{\{\neg p_4, p_8\}} \quad (\text{VI})$$

$$\Diamond \Box p_3 \models_{\mathcal{S}} \varphi'_{\{p_5, \neg p_5, p_6, p_7\}} \quad (\text{VII})$$

$$\bigwedge \emptyset \models_{\mathcal{S}} \varphi'_{\{\neg p_9\}} \quad (\text{VIII})$$

The tree depicted next represents a tableau proof of $\Box p_0$. It differs slightly from the general description we sketched at the beginning of the section, in that leaves with no label are not represented and additional information on the nodes is provided in the form of a finite set of numbers, whose meaning will be explained, and that will be needed to convert the tree into a proof by cuts. Also, it is technically more convenient to work with literals rather than claims. Finally, we stop referring to node labels, and rather adopt the usual definition of a tree, to be reminded shortly, as a finite sequence of entities, here literals; what was referred to above as the label of a node is now the last member of the sequence that defines that node.

The first level of T is determined by (2), and expresses that one of $\Box p_1$, $\Box \neg p_2$, $\Box p_3$, $\Box p_4$ and $\Box \neg p_4$ holds. The node $(p_1, \neg p_2)$ is determined by (4); its member of index 0 is p_1 , and $\{\Box p_1\}$ is a $\subseteq$ -minimal set of claims that logically $\mathcal{S}$ -implies $\varphi'_{\{\neg p_2\}}$, from which $\Box \neg p_2$ can be generated. The node $(p_1, \neg p_2, p_3)$ is determined by (5); its member of index 1 is $\neg p_2$, and $\{\Box \neg p_2\}$ is a $\subseteq$ -minimal set of claims that logically $\mathcal{S}$ -implies $\varphi'_{\{p_3\}}$, from which $\Box p_3$ can be generated. The node $(p_1, \neg p_2, p_3)$ branches out into $(p_1, \neg p_2, p_3, p_5)$, $(p_1, \neg p_2, p_3, \neg p_5)$, $(p_1, \neg p_2, p_3, p_6)$ and $(p_1, \neg p_2, p_3, p_7)$ as determined by (7); p_3 is the element of index 2 of all those sequences, and $\{\Box p_3\}$ is a $\subseteq$ -minimal set of claims that logically $\mathcal{S}$ -implies $\varphi'_{\{p_5, \neg p_5, p_6, p_7\}}$, from which one of $\Box p_5$, $\Box \neg p_5$, $\Box p_6$ and $\Box p_7$ is known to hold. The nodes $(p_1, \neg p_2, p_3, p_5, p_0)$, $(p_1, \neg p_2, p_3, \neg p_5, p_0)$ and $(p_1, \neg p_2, p_3, p_6, p_0)$ are all determined by (0); $\{p_3, p_5\}$, $\{p_3, \neg p_5\}$ and $\{p_3, p_6\}$ are the sets of elements of index 2 and 3 of these sequences, respectively, and if X denotes any of these sets then $\Box X$ is a $\subseteq$ -minimal set of claims that logically $\mathcal{S}$ -implies $\varphi'_{\{p_0\}}$, from which $\Box p_0$ is known to hold. We skip a few nodes and move to $(p_1, \neg p_2, p_3, p_7, p_8)$; its members of index 2, 3 and 4 are p_3 , p_7 and p_8 , and $\{\Box p_3, \Box p_7, \Box p_8\}$ is a $\subseteq$ -minimal set of claims that logically $\mathcal{S}$ -implies $\varphi'_{\emptyset}$, indicating that $\{\Box p_1, \Box \neg p_2, \Box p_3, \Box p_7, \Box p_8\}$ cannot hold together and making $(p_1, \neg p_2, p_3, p_7, p_8)$ a leaf of T' . The subtrees of T rooted at $(\neg p_2)$, (p_3) and $(\neg p_4)$ duplicate the subtrees rooted at $(p_1, \neg p_2)$, $(p_1, \neg p_2, p_3)$ and $(p_1, \neg p_2, p_3, p_7, \neg p_4)$, respectively, and are not explicitly represented; if they were depicted then of course, the integers associated with the nodes would have to be appropriately adapted. The tree T represents a tableau proof of $\Box p_0$ because it has at least one leaf labeled with p_0 (actually, it has 17 such leaves) and the remaining 5 leaves have no label (they are

left unrepresented, and would extend four nodes ending in p_8 and one node ending in p_{10}).

5.3 Completeness of the system of tableau proofs

Let us introduce all the terminology and notation relative to sequences and trees that will be needed in the sequel.

Notation 5.1 Given a sequence σ , we denote by $\text{rng}(\sigma)$ —the range of σ —the set of members of σ , by $\text{lt}(\sigma)$ the length of σ , and in case σ is not the empty sequence, written $()$, by $\text{lst}(\sigma)$ the last element of σ . Given a sequence σ and an element x , we denote by $\sigma \star x$ the concatenation of σ with (x) . Given a sequence σ and $n \in \mathbb{N}$ with $n \leq \text{lt}(\sigma)$, we denote by $\sigma|_n$ the initial segment of σ of length n . Given a sequence σ and $n \in \mathbb{N}$ with $n < \text{lt}(\sigma)$, we denote by $\sigma(n)$ the $(n + 1)$ -st element of σ . Given a nonempty sequence σ , we write σ^- to denote σ truncated from its last element. Given two sequences σ and τ , we write $\sigma \subseteq \tau$ to express that σ is an initial segment of τ .

Definition 5.2 Let a set X be given.

A tree over X is a set of finite sequences of members of X that is closed under initial segments.

Let a tree T over X be given. Note that $()$ is the root of T . A *inner node* of T is a member of T that has a *child* in T , namely, a member of T of the form $\sigma \star x$ for some

$x \in X$. A *leaf* of T is a member of T that is not an inner node of T . A *branch* of T is a $\subseteq$ -maximal subset B of T with the property that any two members of B are such that one is an initial segment of the other.

Notation 5.3 Given a set X , a tree T over X , and a member σ of T , we let $\text{Succ}_T(\sigma)$ denote the set of all $x \in X$ with $\sigma \star x \in T$.

It is time to fix the notation for substitution of stances in a condition by $\bigwedge \emptyset$.

Notation 5.4 Let a condition φ and a set X of literals be given. We denote by $\varphi[X]$ the result of substituting in φ all occurrences of the stances of the form $\Box\xi$ or $\Diamond\xi$ with $\xi \in X$ by $\bigwedge \emptyset$. If X is a singleton $\{\psi\}$ then we write $\varphi[\psi]$ rather than $\varphi[X]$.

Let a condition φ and a set X of hypotheses and literals be given. Let Y be the set of literals in X . We denote by $\varphi[X]$ the condition $\psi[Y]$ where ψ is the result of substituting in φ all occurrences of all hypotheses in X by $\bigwedge \emptyset$.

We can now define tableau proofs in accordance with the semi-formal description given at in Sect. 5.1.

Definition 5.5 Let a disjunctive program $\mathcal{F} = (\varphi_D)_{D \in \text{Alt}(\mathcal{V})}$, a set H of hypotheses, and a member D of $\text{Alt}(\mathcal{V})$ be given. A *tableau proof of D from $\mathcal{F}$ and H* is a nonempty finite tree T over the set of literals with the following properties.

- For all branches B of T , $H \cup \bigcup \{\Box \text{rng}(\sigma) \mid \sigma \in B\}$ is consistent.
- No member of D occurs in any inner node of T .
- Let σ be a node of T . Then
 - either $\varphi_{\text{Succ}_T(\sigma)}[H \cup \text{rng}(\sigma)]$ is $\mathcal{S}$ -valid,
 - or $\text{Succ}_T(\sigma) = \emptyset$ and σ ends in a member of D .

The next proposition shows that the tableau proof procedure is sound and complete.

Proposition 5.6 Let a disjunctive program $\mathcal{F}$, a set H of hypotheses, a left shift completion $\mathcal{F}'$ of $\mathcal{F}$, and a member D of $\text{Alt}(\mathcal{V})$ be given. Then $H \cup \text{Th}(\mathcal{F}) \models_{\mathcal{S}} \bigvee \Box D$ iff there exists a tableau proof of D from $\mathcal{F}'$ and H .

Proof Let T be a tableau proof of D from $\mathcal{F}'$ and H . Let n be the number of inner nodes of T and leaves of T that do not end in a member of D (note that $n > 0$ whether or not T consists only of the empty sequence). Let $X_1, \dots, X_n$ be sets of nodes of T such that $X_1 = \text{Succ}_T(\emptyset)$ and for all nonzero $i < n$, there exists $\sigma \in X_i$ such that

- either σ is an inner node of T and $X_{i+1} = X_i \cup \{\sigma \star \xi \mid \xi \in \text{Succ}_T(\sigma)\} \setminus \{\sigma\}$,
- or σ is a leaf of T , σ does not end in a member of D , and $X_{i+1} = X_i \setminus \{\sigma\}$.

Note that X_n is the set of leaves of T that end in a member of D ; so in order to show that $H \cup \text{Th}(\mathcal{F}) \models_{\mathcal{S}} \bigvee \Box D$, it suffices to prove by Proposition 4.2 that $H \cup \text{Th}(\mathcal{F}')$ logically $\mathcal{S}$ -implies $\bigvee \{\bigwedge \Box \text{rng}(\sigma) \mid \sigma \in X_n\}$ (from which we can derive that $\mathcal{F}$ is $\mathcal{S}$ -inconsistent with H in case X_n is empty). We prove by induction that for all nonzero $i \leq n$, $H \cup \text{Th}(\mathcal{F}')$ logically $\mathcal{S}$ -implies $\bigvee \{\bigwedge \Box \text{rng}(\sigma) \mid \sigma \in X_i\}$. It is immediately verified that $H \cup \text{Th}(\mathcal{F}')$ logically $\mathcal{S}$ -implies $\bigvee \{\bigwedge \Box \text{rng}(\sigma) \mid \sigma \in X_1\}$ (including if

$X_1 = \emptyset$, which can only be the case if $\mathcal{F}$ is $\mathcal{S}$ -inconsistent with H). Let a nonzero $i < n$ be given, and assume that $H \cup \text{Th}(\mathcal{F}') \models_{\mathcal{S}} \bigvee \{ \bigwedge \Box \text{rng}(\sigma) \mid \sigma \in X_i \}$. Assume that $\mathcal{F}'$ is $\mathcal{S}$ -consistent with H , and let $\mathfrak{S} \in \mathcal{S}$ force $H \cup \text{Th}(\mathcal{F}')$. Suppose that X_{i+1} is of the form $X_i \cup \{ \tau \star \xi \mid \xi \in \text{Succ}_T(\tau) \} \setminus \{ \tau \}$ for some inner node τ of T . It is immediately verified that if $\mathfrak{S} \not\models \Box \text{rng}(\tau)$ then $\mathfrak{S} \models \bigvee \{ \bigwedge \Box \text{rng}(\sigma) \mid \sigma \in X_i \setminus \{ \tau \} \}$, and if $\mathfrak{S} \models \Box \text{rng}(\tau)$ then $\mathfrak{S} \models \bigvee \{ \bigwedge \Box \text{rng}(\tau \star \xi) \mid \xi \in \text{Succ}_T \tau \}$, hence $\mathfrak{S}$ forces $\bigvee \{ \bigwedge \Box \text{rng}(\sigma) \mid \sigma \in X_{i+1} \}$. Suppose that X_{i+1} is of the form $X_i \setminus \{ \tau \}$ for some leaf τ of T that does not end in a member of D . Since $\mathfrak{S}$ does not force $\bigwedge \emptyset \rightarrow \bigvee \emptyset$, we infer that $\mathfrak{S} \not\models \Box \text{rng}(\tau)$, hence again, $\mathfrak{S} \models \bigvee \{ \bigwedge \Box \text{rng}(\sigma) \mid \sigma \in X_{i+1} \}$, as wanted.

Conversely, assume that $H \cup \text{Th}(\mathcal{F}) \models_{\mathcal{S}} \bigvee \Box D$. Fix an enumeration $(D_i)_{i \in \mathbb{N}}$ of $\text{Alt}(\mathcal{V})$. Set $\mathcal{F}' = (\varphi_E)_{E \in \text{Alt}(\mathcal{V})}$. Define a sequence $(T_n)_{n \in \mathbb{N}}$ of trees over the set of literals as follows. Set $T_0 = \{ \emptyset \}$. Let $n \in \mathbb{N}$ be given, and assume that T_n has been defined. First we let $T_n \subseteq T_{n+1}$. Let σ be a leaf of T_n . If $\text{rng}(\sigma) \cap D \neq \emptyset$ then no strict extension of σ belongs to T_{n+1} . Suppose that $\text{rng}(\sigma) \cap D = \emptyset$. If there exists a least $i \in \mathbb{N}$ such that $H \cup \Box \text{rng}(\sigma) \cup \Box D_i$ is consistent, $\varphi_{D_i}[H \cup \text{rng}(\sigma)]$ is $\mathcal{S}$ -valid and there is no initial segment τ of σ such that $\{ \tau \star \xi \mid \xi \in D_i \} \subseteq T_n$, then the strict extensions of σ in T_{n+1} are precisely the sequences of the form $\sigma \star \xi$ with $\xi \in D_i$; otherwise no strict extension of σ belongs to T_{n+1} . This completes the definition of T_{n+1} . Set $T = \bigcup_{n \in \mathbb{N}} T_n$. We are done if we show that T is a tableau proof of D from $\mathcal{F}'$ and H . Let B be a branch of T . We first show the following.

- (1) B is finite.
- (2) Let σ be the leaf of T that B ends in. If $\text{rng}(\sigma) \cap D = \emptyset$ then $\varphi_{\emptyset}[H \cup \text{rng}(\sigma)]$ is $\mathcal{S}$ -valid.

Suppose for a contradiction that either B is infinite or B is finite but (2) does not hold. Let X be the set of literals that occur in B . It is immediately verified that $H \cup \Box X$ is consistent. Let $\mathfrak{S}$ be the $\subseteq$ -minimal member of $\mathcal{S}$ that contains $H \cup \Box X$. Note that X contains no member of D whether B is finite or not, and so $\mathfrak{S}$ does not force $\bigvee \Box D$, whether D is empty or not. Let $i \in \mathbb{N}$ be such that $D = D_i$, so $\mathfrak{S}$ forces φ_{D_i} . Let Y be the set of all literals ψ in D_i such that $H \cup \Box X \cup \{ \Box \psi \}$ is inconsistent. Let $j \in \mathbb{N}$ be such that $D_j = D_i \setminus Y$. Then by the choice of $\mathcal{F}'$, $\mathfrak{S}$ forces φ_{D_j} , and $H \cup \Box X \cup \Box D_j$ is obviously consistent. It is then easy to verify that by assumption on B and by construction of $(T_n)_{n \in \mathbb{N}}$, there exists a member τ of B with the property that:

- $\varphi_{D_j}[H \cup \text{rng}(\tau)]$ is $\mathcal{S}$ -valid;
- for all $k < j$ and strict initial segments τ' of τ , either $\varphi_{D_k}[H \cup \text{rng}(\tau')]$ is not $\mathcal{S}$ -valid or $\{ \tau' \star \xi \mid \xi \in D_k \} \subseteq T$.

If $D_j = \emptyset$ then B clearly ends in τ , which contradicts the assumption that (2) above does not hold. If $D_j \neq \emptyset$ then $\tau \star \xi$ belongs to B for some $\xi \in D_j$ and $\mathfrak{S}$ forces $\bigvee \Box D_j$. So if $\mathfrak{S}$ does not force $\varphi_{\emptyset}$ then we infer that $\mathfrak{S}$ forces $H \cup \text{Th}(\mathcal{F}')$, which contradicts the assumption that $H \cup \text{Th}(\mathcal{F}') \models_{\mathcal{S}} \bigvee \Box D$. So we have shown that B satisfies (A) and (B) above. In particular, we have shown that T contains no infinite branch, which by König's lemma, implies that T is finite. Finally, from the construction of $(T_n)_{n \in \mathbb{N}}$ and the properties of T 's branches demonstrated above, we conclude that T is a tableau proof of D from $\mathcal{F}$ and H . $\square$

The next corollary emphasises that tableau proofs are suitable for refutation.

Corollary 5.7 *Let a disjunctive program $\mathcal{F}$, a left shift completion $\mathcal{F}'$ of $\mathcal{F}$, and a set H of hypotheses be given. Then the following conditions are equivalent.*

- $\mathcal{F}$ is $\mathcal{S}$ -inconsistent with H .
- There exists a tableau proof of $\emptyset$ from $\mathcal{F}'$ and H .

It is well worth also to take note of both corollaries that follow, the second of which expresses the compactness of the tableau proof procedure.

Corollary 5.8 *Let a disjunctive program $\mathcal{F}$, a left shift completion $\mathcal{F}'$ of $\mathcal{F}$, and a set H of hypotheses be such that $\mathcal{F}$ is $\mathcal{S}$ -consistent with H . Let a member D of $\text{Alt}(\mathcal{V})$ be such that $H \cup \text{Th}(\mathcal{F}) \models_{\mathcal{S}} \bigvee \square D$. Then every tableau proof of D from $\mathcal{F}'$ and H has at least one branch that ends in a member of D .*

Corollary 5.9 *Let a disjunctive program $\mathcal{F}$, a set H of hypotheses, and a member D of $\text{Alt}(\mathcal{V})$ be such that $H \cup \text{Th}(\mathcal{F}) \models_{\mathcal{S}} \bigvee \square D$. Let $\mathcal{F}' = (\varphi_E)_{E \in \text{Alt}(\mathcal{V})}$ be a left shift completion of $\mathcal{F}$. There exists a finite subset X of $\text{Alt}(\mathcal{V})$ with the following property. Let $\mathcal{F}'' = (\varphi_E)_{E \in \text{Alt}(\mathcal{V})}$ be the disjunctive program such that for all members E of $\text{Alt}(\mathcal{V})$, either $E \in X$ and $\varphi'_E = \varphi_E$, or $E \notin X$ and $\varphi'_E = \bigvee \emptyset$. Then there exists a tableau proof of D from $\mathcal{F}''$ and H .*

6 Proofs by cuts

6.1 General strategy

Let us further exploit the example given in the previous section and explain, on the basis of that example, how a tableau proof can be converted into a proof by cuts. Let T be the tree depicted in the previous section, which, recall, represents a tableau proof of $\square p_0$. The strategy is to explore T depth first and label some nodes N in T with a member of $\text{Alt}(\mathcal{V})$ determined by the (possibly empty) set of N 's children and by the labels associated with $N_{|i_1+1}, \dots, N_{|i_k+k}$ where $\{i_1, \dots, i_k\}$ is the (possibly empty) set of numbers associated with N in T (those labels will necessarily exist). There might be subtrees of T that will be skipped during this exploration; the nodes of those subtrees will then not be labeled. Also, some nodes might receive various labels over time: when a leaf gets labeled, then the exploration of T proceeds by backtracking and the label assigned to that leaf replaces the label (guaranteed to exist) that had been previously assigned to the node we backtrack to. The fact that we will eventually obtain a proof of $\square p_0$ by cut will be captured by the fact that $\square p_0$ will be the label last assigned to a node. More generally, a proof by cut of an alternative of the form $\bigvee \square D$, $D \in \text{Alt}(\mathcal{V})$, will require that the label last assigned to a node be a subset of D .

Let us explain in a little more detail how labels are determined. Let N be a node in T that has not received any label yet but whose parent, if any, has received some label. If N ends in p_0 (more generally, if we try and prove $\bigvee \square D$ for some $D \in \text{Alt}(\mathcal{V})$ and N ends in a member of D) then N is necessarily a leaf, and it receives the label

last assigned to its parent. Suppose that we are not in that situation. If N has a parent and the label last assigned to it does not contain the literal N ends in, then the subtree of T rooted at N is skipped and none of its nodes receives any label. Suppose that we are not in that situation either. Let $k \in \mathbb{N}$ and literals $\xi_1, \dots, \xi_k$ be such that N has k children in T , those children being $N \star \xi_1, \dots, N \star \xi_k$. Let $n \in \mathbb{N}$ be the number of integers associated with N in T , and let $i_1, \dots, i_n$ be those integers. Let $\psi_1, \dots, \psi_n$ be the last elements of $N_{|i_1}, \dots, N_{|i_n}$, and let $D_1, \dots, D_n$ be the labels that have (necessarily) been assigned to $N_{|i_1}, \dots, N_{|i_n}$, respectively (less precisely, $\psi_1, \dots, \psi_n$ are the literals on the path from the root of T to N at positions $i_0, \dots, i_n$, and $D_1, \dots, D_n$ are the labels currently associated with those positions, respectively). Then $\{\Box\psi_1, \dots, \Box\psi_n\}$ is a $\subseteq$ -minimal set of claims that logically $\mathcal{S}$ -implies $\varphi_{\{\xi_1, \dots, \xi_k\}}$, ψ_1 belongs to $D_1, \dots$, and ψ_n belongs to D_n . Hence the cut rule can be applied to $\vdash \Box D_1, \dots, \vdash \Box D_n$ and $\varphi_{\{\xi_1, \dots, \xi_k\}} \vdash \Box\xi_1, \dots, \Box\xi_k$. If $\vdash \Box\chi_1, \dots, \Box\chi_m$ is the consequent of that application of the cut rule, then $\{\chi_1, \dots, \chi_m\}$ is the label we first (and possibly last) assign to N .

6.2 Example

We will prove the completeness of the cut proof technique based on a construction which will deviate slightly from what has been outlined in that we will not assign labels to nodes in T (the tree used as an example), but rather define a new tree T' from T ; the difference is purely technical. Following is a pictorial representation of T' , together with some indication of how T' is constructed as we explore T depth first and backtrack from leaves to inner nodes down the tree. To save space, we represent a negated atom φ as $\bar{\varphi}$.

Let us explain how T' is obtained. Set $\sigma_0 = \{p_1, \neg p_2, p_3, p_4, \neg p_4\}$. The unique child of the root of T' , namely (σ_0) , is determined by the successors of the root of T , similarly expressing that one of $\Box p_1, \Box \neg p_2, \Box p_3, \Box p_4$ and $\Box \neg p_4$ holds. Set $\sigma_1 = \{\neg p_2, p_3, p_4, \neg p_4\}$. The node (σ_0, σ_1) of T' is determined by the node (p_1) of T , and more precisely, the associated number (namely, 0) and the successor (namely, $\neg p_2$) of that node in T : σ_1 is obtained by applying the cut rule as follows to $\Box p_1 \vee \Box \neg p_2 \vee \Box p_3 \vee \Box p_4 \vee \Box \neg p_4$, which is currently associated with the element (σ_0, σ_1) of index 0, and to $\varphi'_{\{\neg p_2\}} \rightarrow \Box \neg p_2$.

$$\frac{\Box p_1 \vee \Box \neg p_2 \vee \Box p_3 \vee \Box p_4 \vee \Box \neg p_4 \quad \varphi'_{\{\neg p_2\}} \rightarrow \Box \neg p_2}{\Box \neg p_2 \vee \Box p_3 \vee \Box p_4 \vee \Box \neg p_4} p_1$$

Set $\sigma_2 = \{p_3, p_4, \neg p_4\}$. The node $(\sigma_0, \sigma_1, \sigma_2)$ of T' is determined by the node $(p_1, \neg p_2)$ of T , and more precisely, the associated number (namely, 1) and the successor (namely, p_3) of that node in T : σ_2 is obtained by applying the cut rule as follows to $\Box \neg p_2 \vee \Box p_3 \vee \Box p_4 \vee \Box \neg p_4$, which is currently associated with the element $(\sigma_0, \sigma_1, \sigma_2)$ of index 1, and to $\varphi'_{\{p_3\}} \rightarrow \Box p_3$.

$$\frac{\Box \neg p_2 \vee \Box p_3 \vee \Box p_4 \vee \Box \neg p_4 \quad \varphi'_{\{p_3\}} \rightarrow \Box p_3}{\Box p_3 \vee \Box p_4 \vee \Box \neg p_4} \neg p_2$$

- $\sigma_3 = \{p_5, \neg p_5, p_6, p_7, p_4, \neg p_4\}$,
- $\sigma_4 = \{p_0, p_4, \neg p_4, \neg p_5, p_6, p_7\}$,
- $\sigma_5 = \{p_0, p_4, \neg p_4, p_6, p_7\}$,
- $\sigma_6 = \{p_0, p_4, \neg p_4, p_7\}$,
- ...,
- $\sigma_{10} = \{\neg p_9\}$,
- $\sigma_{11} = \{p_0, \neg p_1, p_{10}, \neg p_4\}$,
- ...,
- $\sigma_{14} = \{p_0\}$.

$$\frac{\Box p_3 \vee \Box p_4 \vee \Box \neg p_4 \quad \varphi'_{\{p_5, \neg p_5, p_6, p_7\}} \rightarrow \Box p_5 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7}{\Box p_5 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7 \vee \Box p_4 \vee \Box \neg p_4} p_3$$
$$\frac{\begin{array}{c} \Box p_3 \vee \Box p_4 \vee \Box \neg p_4 \quad \Box p_5 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7 \vee \Box p_4 \vee \Box \neg p_4 \quad \varphi'_{\{p_0\}} \rightarrow \Box p_0 \\ \hline \Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7 \end{array}}{p_3, p_5}$$

As we backtrack one level up in T , we associate $\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7$ in place of $\Box p_5 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7 \vee \Box p_4 \vee \Box \neg p_4$ to $(\sigma_0, \sigma_1, \sigma_2, \sigma_3)$. The node $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_5)$ of T' is determined by the node $(p_1, \neg p_2, p_3, \neg p_5)$ of T , and more precisely, the associated numbers (namely, 2 and 3) and the successor (namely, p_0) of that node in T : σ_5 is obtained by applying the cut rule as follows, first to $\Box p_3 \vee \Box p_4 \vee \Box \neg p_4$, which is currently associated with the element $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of index 2, second to $\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7$, which is currently associated with the element $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of index 3, and to $\varphi'_{\{p_0\}} \rightarrow \Box p_0$.

$$\frac{\Box p_3 \vee \Box p_4 \vee \Box \neg p_4 \quad \Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7 \quad \varphi'_{\{p_0\}} \rightarrow \Box p_0}{\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box p_6 \vee \Box p_7} p_3, \neg p_5$$

As we backtrack one level up in T , we associate $\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box p_6 \vee \Box p_7$ in place of $\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box \neg p_5 \vee \Box p_6 \vee \Box p_7$ to $(\sigma_0, \sigma_1, \sigma_2, \sigma_3)$. Then σ_6 is obtained as follows.

$$\frac{\Box p_3 \vee \Box p_4 \vee \Box \neg p_4 \quad \Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box p_6 \vee \Box p_7 \quad \varphi'_{\{p_0\}} \rightarrow \Box p_0}{\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box p_7} p_3, p_6$$

Moving on, σ_7 and σ_8 are obtained as follows.

$$\frac{\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box p_7 \quad \varphi'_{\{\neg p_4, p_8\}} \rightarrow \Box p_4 \vee \Box p_8}{\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box p_8} p_7$$

$$\frac{\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box p_8 \quad \varphi'_{\{p_0\}} \rightarrow \Box p_0}{\Box p_0 \vee \Box p_4 \vee \Box p_8} \neg p_4$$

Node $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_7, \sigma_9)$ of T' is determined by node $(p_1, \neg p_2, p_3, p_7, p_8)$ of T , and more precisely, the associated numbers (namely, 2, 3 and 4) and the fact that that node has no successor in T : σ_9 is obtained by applying the cut rule as follows, first to $\Box p_3 \vee \Box p_4 \vee \Box \neg p_4$, which is currently associated with the element $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_7, \sigma_9)$ of index 2, second to $\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box p_7$, which is currently associated with the element $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_7, \sigma_9)$ of index 3, third to $\Box p_0 \vee \Box p_4 \vee \Box p_8$, which is currently associated with the element $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_7, \sigma_9)$ of index 4, and to $\varphi'_{\emptyset} \rightarrow \bigvee \emptyset$.

$$\frac{\Box p_3 \vee \Box p_4 \vee \Box \neg p_4 \quad \Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \vee \Box p_7 \quad \Box p_0 \vee \Box p_4 \vee \Box p_8 \quad \varphi'_{\emptyset} \rightarrow \bigvee \emptyset}{\Box p_0 \vee \Box p_4 \vee \Box \neg p_4} p_3, p_7, p_8$$

The node of T that comes after $(p_1, \neg p_2, p_3, p_7, p_8)$ in a depth-first exploration of T is $(\neg p_2)$, but as $\neg p_2$ does not occur in σ_9 , the subtree of T rooted at $(\neg p_2)$ is skipped over. The next node of T that is then reached in a depth-first exploration of T is (p_3) , that also does not occur in σ_9 , so the subtree of T rooted at (p_3) is skipped over. The next node of T to consider is then (p_4) , and as p_4 belongs to σ_9 , we associate $\Box p_0 \vee \Box p_4 \vee \Box \neg p_4$ in place of $\Box p_1 \vee \Box \neg p_2 \vee \Box p_3 \vee \Box p_4 \vee \Box \neg p_4$ to (σ_0) . As the condition of $\varphi'_{\neg p_9}$ is $\bigwedge \emptyset$, no application of the cut rule is necessary to determine σ_{10} . Then σ_{11}, σ_{12} and σ_{13} are determined as follows.

$$\begin{array}{c}
\frac{\frac{\frac{\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \quad \Box \neg p_9 \quad \varphi'_{\{\neg p_1, p_{10}\}} \rightarrow \Box \neg p_1 \vee \Box p_{10}}{\Box p_0 \vee \Box \neg p_1 \vee \Box p_{10} \vee \Box \neg p_4} \quad p_4, \neg p_9}{\frac{\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \quad \Box \neg p_9 \quad \Box p_0 \vee \Box \neg p_1 \vee \Box p_{10} \vee \Box \neg p_4 \quad \varphi'_{\emptyset} \rightarrow \bigvee \emptyset}{\Box p_0 \vee \Box \neg p_1 \vee \Box \neg p_4} \quad p_4, \neg p_9, p_{10}} \\
\frac{\Box p_0 \vee \Box p_4 \vee \Box \neg p_4 \quad \Box p_0 \vee \Box \neg p_1 \vee \Box \neg p_4 \quad \varphi'_{\{p_0\}} \rightarrow \Box p_0}{\Box p_0 \vee \Box \neg p_4} \quad p_4, \neg p_1
\end{array}$$

Finally, σ_{14} is found out to be equal to $\{p_0\}$ thanks to the following application of the cut rule, completing the proof by cuts that $\mathcal{F}$ logically $\mathcal{S}$ -implies $\Box p_0$.

$$\frac{\Box p_0 \vee \Box \neg p_4 \quad \varphi'_{\{p_0\}} \rightarrow \Box p_0}{\Box p_0} \neg p_4$$

6.3 Condition reduction

In Sect. 2.5, we discussed how the substitution of stances in a condition by $\bigwedge \emptyset$ could be mechanically processed to eventually result in $\bigwedge \emptyset$ precisely in case the condition with those substitutions performed was $\mathcal{S}$ -valid. This is a particular case of the reduction of a condition with stances substituted by $\bigwedge \emptyset$ into a simpler condition, in a way captured by the couple of definitions that follow.

Definition 6.1 Let a condition φ be given. We call *reduct* of φ any condition ψ such that one of the conditions that follow holds.

- ψ is φ .
- φ is of the form $\bigvee X \cup \{\bigvee Y\}$ and ψ is $\bigvee X \cup Y$.
- φ is of the form $\bigwedge X \cup \{\bigwedge Y\}$ and ψ is $\bigwedge X \cup Y$.
- φ is of the form $\bigvee X \cup \{\bigwedge \emptyset\}$ and ψ is $\bigwedge \emptyset$.
- φ is of the form $\bigwedge X \cup \{\bigvee \emptyset\}$ and ψ is $\bigvee \emptyset$.
- φ is of the form $\bigvee X \cup \{\xi\}$ and ψ is $\bigvee X \cup \{\chi\}$ for some reduct χ of ξ .
- φ is of the form $\bigwedge X \cup \{\xi\}$ and ψ is $\bigwedge X \cup \{\chi\}$ for some reduct χ of ξ .

Definition 6.2 A condition is said to be *in reduced form* iff it is its only reduct.

Property 6.3 Every condition has a unique reduct in reduced form.

Notation 6.4 Given a condition φ , we let $\rho(\varphi)$ denote the unique condition in reduced form that is a reduct of φ .

For an example, if φ is the condition defined at the beginning of the last paragraph of Sect. 2.5, then the discussion in that paragraph shows that $\rho(\varphi) = \bigwedge \emptyset$.

Property 6.5 For all conditions φ , $\rho(\varphi)$ is logically $\mathcal{S}$ -equivalent to φ .

The key feature that a $\mathcal{S}$ -valid condition can be reduced to $\bigwedge \emptyset$ is a corollary of the property that follows.

Property 6.6 Let a condition φ and a set X of hypotheses and literals be given. Let $\mathfrak{S}$ be the $\subseteq$ -minimal member of $\mathcal{S}$ that contains all hypotheses in X and all claims of the form $\Box \xi$ with $\xi \in X$. Then $\mathfrak{S}$ forces φ iff $\rho(\varphi[X])$ is equal to $\bigwedge \emptyset$.

Corollary 6.7 For all conditions φ , if φ is $\mathcal{S}$ -valid then $\rho(\varphi)$ is $\bigwedge \emptyset$.

6.4 Completeness of the system of proofs by cuts

What can be derived from a disjunctive program and a set of hypotheses by applying the cut rule iteratively is defined in the notation that follows. It formalises the notation $[\mathcal{F}, H]$ which had been informally introduced towards the end of Sect. 2.4, and essentially captures the notion of proof by application of a suitable form of the cut rule, identified in our discussion of Sect. 2.

Notation 6.8 Let a disjunctive program $\mathcal{F} = (\varphi_D)_{D \in \text{Alt}(\mathcal{V})}$ and a set H of hypotheses be given. We define $[\mathcal{F}, H]$ as the $\subseteq$ -minimal set of members of $\text{Alt}(\mathcal{V})$ of the form $D \cup (D_1 \setminus \{\psi_1\}) \cup \dots \cup (D_k \setminus \{\psi_k\})$ such that

- k belongs to $\mathbb{N}$, D to $\text{Alt}(\mathcal{V})$, each of $D_1, \dots, D_k$ to $[\mathcal{F}, H]$, and $\psi_1, \dots, \psi_k$ are literals in $D_1, \dots, D_k$, respectively,
- $\psi_1, \dots, \psi_k$ all occur in $\varphi_D[H]$, and
- $\rho(\varphi_D[H \cup \{\psi_1, \dots, \psi_k\}]) = \bigwedge \emptyset$.

The cut rule is valid:

Proposition 6.9 For all disjunctive programs $\mathcal{F}$, sets H of hypotheses and members D of $[\mathcal{F}, H]$, $\bigvee D$ is a logical $\mathcal{S}$ -consequence of $H \cup \text{Th}(\mathcal{F})$.

Proof Let a disjunctive program $\mathcal{F} = (\varphi_D)_{D \in \text{Alt}(\mathcal{V})}$ and a set H of hypotheses be given. Let $D \in \text{Alt}(\mathcal{V})$, $k \in \mathbb{N}$, members $D_1, \dots, D_k$ of $[\mathcal{F}, H]$, and literals $\psi_1, \dots, \psi_k$ in $D_1, \dots, D_k$, respectively, be such that $\psi_1, \dots, \psi_k$ all occur in $\varphi_D[H]$ and $\rho(\varphi_D[H \cup \{\psi_1, \dots, \psi_k\}]) = \bigwedge \emptyset$. Set $D' = D \cup (D_1 \setminus \{\psi_1\}) \cup \dots \cup (D_k \setminus \{\psi_k\})$. Let $\mathfrak{S} \in \mathcal{S}$ force $H \cup \text{Th}(\mathcal{F})$ and each of $\bigvee \square D_1, \dots, \bigvee \square D_k$. If $\mathfrak{S}$ does not force all of $\square \psi_1, \dots, \square \psi_k$, then $\mathfrak{S}$ forces $\bigvee D_i \setminus \{\psi_i\}$ for some nonzero $i \leq k$, hence $\bigvee \square D'$. Suppose that $\mathfrak{S}$ forces all of $\square \psi_1, \dots, \square \psi_k$. Since $\mathfrak{S} \Vdash H$ and $\rho(\varphi_D[H \cup \{\psi_1, \dots, \psi_k\}]) = \bigwedge \emptyset$, then $\mathfrak{S}$ forces φ_D . Since also $\mathfrak{S} \Vdash \text{Th}(\mathcal{F})$ then $\mathfrak{S}$ forces $\bigvee \square D$, hence $\bigvee \square D'$. $\square$

We can finally formulate and prove the key result of this paper.

Proposition 6.10 Let a disjunctive program $\mathcal{F}$, a left shift completion $\mathcal{F}'$ of $\mathcal{F}$, and a set H of hypotheses be given. Then for all $D \in \text{Alt}(\mathcal{V})$, $\bigvee \square D$ is a logical $\mathcal{S}$ -consequence of $H \cup \text{Th}(\mathcal{F})$ iff $[\mathcal{F}', H]$ contains a subset of D .

Proof Set $\mathcal{F}' = (\varphi_E)_{E \in \text{Alt}(\mathcal{V})}$. By Proposition 5.6, let T be a tableau proof of D from $\mathcal{F}'$ and H . Let T' be the set of members of T that contain no occurrence of any member of D . Let $N \in \mathbb{N}$ be the cardinality of T' , and let $(\sigma_0, \dots, \sigma_{N-1})$ be an enumeration of the members of T' such that $\sigma_0 = ()$ and for all $i < N$, σ_i is a child of a member of $\{\sigma_j \mid j < i\}$ in T' of maximal length (so $(\sigma_i)_{i < N}$ is a depth-first enumeration of T'). For all $n \in \mathbb{N}$ and for all members σ of T' of length n , we let $\text{spt}(\sigma)$ —the support of σ —denote a $\subseteq$ -minimal subset of $\{0, \dots, n-1\}$ such that $\varphi_{\text{Succ}_T(\sigma)}[H \cup \{\sigma(i) \mid i \in \text{spt}(\sigma)\}]$ is $\mathcal{S}$ -valid (it is immediately verified that such a set exists). We now inductively define for all $i < N$ a sequence $([\sigma_i])_{i < N}$ of members of $\text{Alt}(\mathcal{V})$ of length either 0 or $\text{lt}(\sigma_i) + 1$. Set $[\sigma_0] = (\text{Succ}_T(\sigma_0))$. Let a nonzero $i < N$ be least such that $[\sigma_i]$ has not been defined yet. Then by construction, $[\sigma_{i-1}] \neq ()$. We

now determine some integer j with $i \leq j < N$ and define $[\sigma_k]$ for all $k \in \{i, \dots, j\}$. Let j be the least integer such that either $j = N$, or $i \leq j < N$, $\text{lt}(\sigma_j) \leq \text{lt}(\sigma_i)$, and $\text{lst}(\sigma_j) \in \text{lst}([\sigma_{i-1}])$. Then for all $k \in \{i, \dots, j-1\}$, $[\sigma_k] = ()$. If $j = N$ then we are done with the construction, so suppose otherwise. Note that for all strict initial segments τ of σ_j , $[\tau]$ has been defined and is different to $()$.

- If $\text{lt}(\sigma_j) > 1$ then $[\sigma_j]_{\text{lt}(\sigma_j)-1}$ is equal to $[\sigma_{i-1}]_{\text{lt}(\sigma_j)-1}$.
- The penultimate element of $[\sigma_j]$ is equal to $\text{lst}([\sigma_{i-1}])$.
- $\text{lst}([\sigma_j])$ is equal to $\text{Succ}_T(\sigma_j) \cup \bigcup \{[\sigma_j](n) \setminus \sigma_j(n) \mid n \in \text{spt}(\sigma_j)\}$.

We prove by induction that the following holds for all $i < N$ with $[\sigma_i] \neq ()$.

- (1) For all $n < \text{lt}(\sigma_i)$, $[\sigma_i](n)$ is included in the union of $D \cup \{\sigma_i(n)\}$ with $\{\text{lst}(\sigma_j) \mid i < j < N, \text{lt}(\sigma_j) \leq n+1, \sigma_j^- \subseteq \sigma_i\}$.
- (2) $\text{lst}([\sigma_i]) \subseteq D \cup \{\text{lst}(\sigma_j) \mid i < j < N, \text{lt}(\sigma_j) \leq \text{lt}(\sigma_i) + 1, \sigma_j^- \subseteq \sigma_i\}$.

Verification of (1) and (2) is straightforward for $i = 0$. Let $k < N$ be such that $[\sigma_k] \neq ()$, and assume that for all $i < k$ with $[\sigma_i] \neq ()$, (1) and (2) hold. Let i be the maximal integer smaller than k such that $[\sigma_i] \neq ()$. If $\text{lt}(\sigma_k) > 1$ then, using part (1) of the inductive hypothesis, the fact that $(\sigma_j)_{j < N}$ is a depth-first enumeration of T' , and the definition of $[\sigma_k]_{\text{lt}(\sigma_k)-1}$, it is easy to verify the following.

($\dagger$) For all $n < \text{lt}(\sigma_k) - 1$, $[\sigma_k](n)$ is included in

$$D \cup \{\sigma_k(n)\} \cup \{\text{lst}(\sigma_j) \mid k < j < N, \text{lt}(\sigma_j) \leq n+1, \sigma_j^- \subseteq \sigma_k\}.$$

Using part (2) of the inductive hypothesis, the fact that $(\sigma_j)_{j < N}$ is a depth-first enumeration of T' , and the fact that the penultimate element of $[\sigma_k]$ is $\text{lst}([\sigma_i])$, it is easy to verify the following, whether $\text{lt}(\sigma_k) = \text{lt}(\sigma_i) + 1$ or whether $\text{lt}(\sigma_k) \leq \text{lt}(\sigma_i)$.

($\ddagger$) $[\sigma_k](\text{lt}(\sigma_k) - 1)$ is included in

$$D \cup \{\sigma_k(\text{lt}(\sigma_k) - 1)\} \cup \{\text{lst}(\sigma_j) \mid k < j < N, \text{lt}(\sigma_j) \leq \text{lt}(\sigma_k), \sigma_j^- \subseteq \sigma_k\}.$$

Finally, using ($\dagger$) and ($\ddagger$) and the definition of $\text{lst}([\sigma_k])$, it is easy to verify that

$$\text{lst}([\sigma_k]) \subseteq D \cup \{\text{lst}(\sigma_j) \mid k < j < N, \text{lt}(\sigma_j) \leq \text{lt}(\sigma_k) + 1, \sigma_j^- \subseteq \sigma_k\}.$$

So (1) and (2) hold for all $i < N$ with $[\sigma_i] \neq ()$. From (2) and the definition of $(\sigma_i)_{i < N}$, we then infer that for all $i < N$, if $\text{lst}([\sigma_i]) \not\subseteq D$ then there exists $j < N$ with $i < j$ and $[\sigma_j] \neq ()$. But this obviously implies the following.

There exists $i < N$ with $[\sigma_i] \neq ()$ and $\text{lst}([\sigma_i]) \subseteq D$.

To complete the proof of the proposition, it suffices to show that for all $i < N$ with $[\sigma_i] \neq ()$, $\text{lst}([\sigma_i])$ belongs to $[\mathcal{F}', H]$. Proof is by induction. Trivially, $\text{lst}([\sigma_0])$ is a member of $[\mathcal{F}', H]$. Let $i < N$ be such that $[\sigma_i] \neq ()$ and for all $j < i$ with $[\sigma_j] \neq ()$, $\text{lst}([\sigma_j]) \in [\mathcal{F}', H]$. Note that for all $n < \text{lt}(\sigma_i)$, there exists $j < i$ with $[\sigma_j] \neq ()$ and $[\sigma_i](n)$ is equal to $\text{lst}([\sigma_j])$. Let $k \in \mathbb{N}$ denote the cardinality of $\text{spt}(\sigma_i)$, and let $e_1, \dots, e_k$ enumerate its elements. By definition of $\text{spt}(\sigma_i)$ (and more

particularly, the $\subseteq$ -minimality condition), $\varphi_{\text{lst}(\sigma_i)}[H]$ contains at least one occurrence of each of $\Box\sigma_i(e_1), \dots, \Box\sigma_i(e_k)$. Also note that for all nonzero $j \leq k$, $\sigma_i(k)$ belongs to $[\sigma_i](k)$, and that $\varphi_{\text{lst}(\sigma_i)}[H \cup \{[\sigma_i](e_1), \dots, [\sigma_i](e_k)\}]$ is $\mathcal{S}$ -valid. We conclude from the previous observations and the definitions of $[\mathcal{F}', H]$ and $\text{lst}([\sigma_i])$ that $[\mathcal{F}', H]$ contains $\text{lst}([\sigma_i])$, completing the proof of the proposition. $\square$

7 Conclusion

We have presented a classical, modal approach to disjunctive logic programs. It is classical in three respects. First, in that only classical negation is used. Second, in that a classical proof technique, based on a generalisation of the cut rule, is complete. Third, in that the semantics can be defined in terms of logical consequence, rather than in terms of minimal or preferred models. The semantics is flexible enough to capture the well known semantics that have been proposed, by possibly expanding the set of rules with formulas referred to as hypotheses, requested to satisfy some special conditions. This has been demonstrated for the answer set semantics.

Acknowledgements The author expresses his many thanks to the reviewers whose well thought comments helped improve the presentation of the paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Alferes, J.J., Pereira, L.M., Przymusiński, T.C.: Strong and explicit negation in non-monotonic reasoning and logic programming. In: Logics in artificial intelligence Évora, volume 1126 of Lecture notes in computer science, pp. 143–163. Springer-Verlag (1996)
2. Apt, K.R., Bol, R.: Logic programming and negation: a survey. *J. Log. Program.* **19–20**(Supplement 1), 9–71 (1994)
3. Baral, C., Lobo, J., Minker, J.: Generalized disjunctive well-founded semantics for logic programs: procedural semantics. *Methodol. Intell. Syst.* **5**, 456–464 (1990)
4. Brass, S., Dix, J.: Semantics of (disjunctive) logic programs based on partial evaluation. *J. Logic Program.* **40**(1), 1–46 (1999)
5. Brass, S., Dix, J., Przymusiński, T.C.: Super logic programs. *ACM Trans. Comput. Log.* **5**(1), 129–176 (2004)
6. Dix, J., Gottlob, G., Marek, W.: Reducing disjunctive to non-disjunctive semantics by shift-operations. *Fund. Inf.* **28**(1–2), 87–100 (1996)
7. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: propositional case. *Ann. Math. Artif. Intell.* **15**, 289–323 (1995)
8. Eshghi, K., Kowalski, R.A.: Abduction compared with negation by failure. In: Proceedings of the 6th International Conference on Logic Programming, pp. 234–254 (1989)
9. Gebser, M., Schaub, T.: Generic tableaux for answer set programming. In: Proceedings of the 23rd International Conference on Logic Programming, pp. 119–133 (2007)

10. Gelfond, M., Leone, N.: Logic programming and knowledge representation—the a-prolog perspective. *Artif. Intell.* **138**(1–2), 3–38 (2002)
11. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, volume 2 of MIT Press Series in logic programming, pp. 1070–1080. MIT Press (1988)
12. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. *N. Gener. Comput.* **9**, 365–385 (1991)
13. Lin, F., Zhao, Y.: ASSAT: computing answer sets of a logic program by SAT solvers. *Artif. Intell.* **157**(1–2), 115–137 (2004)
14. Marek, W., Truszczyński, M.: Autoepistemic logic. *J. Assoc. Comput. Mach.* **38**(3), 587–618 (1991)
15. Martin, E.: Contextual hypotheses and semantics of logic programs. *Theory Pract. Logic Program.* **12**(6), 843–887 (2012)
16. Martin, E.: Logic programming as classical inference. *J. Appl. Logic Arch.* **13**(3), 316–369 (2015)
17. Minker, J., Rajasekar, A.: A fixpoint semantics for disjunctive logic programs. *J. Log. Program.* **9**(1), 45–74 (1990)
18. Minker, J., Ruiz, C.: Semantics for disjunctive logic programs with explicit and default negation. *Fund. Inform.* **20**(1–3), 145–192 (1994)
19. Minker, J., Seipel, D.: Disjunctive logic programming: a survey and assessment. In: *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, pp. 472–511. Springer-Verlag (2002)
20. Moore, R.C.: Semantical considerations on nonmonotonic logic. *Artif. Intell.* **25**(1), 75–94 (1985)
21. Linh Anh Nguyen and Rajeev Goré: Completeness of hyper-resolution via the semantics of disjunctive logic programs. *Inf. Process. Lett.* **95**(2), 363–369 (2005)
22. Palacios, H., Geffner, H.: Compiling uncertainty away: solving conformant planning problems using a classical planner (sometimes). In: *AAAI*, pp. 900–905 (2006)
23. Pearce, D., Tompits, H., Woltran, S.: Characterising equilibrium logic and nested logic programs: reductions and complexity 1,2. *Theory Pract. Logic Program.* **9**(5), 565–616 (2009)
24. Pearce, D., Wagner, G.: Logic programming with strong negation. In: *Proceedings of the International Workshop on Extensions of Logic Programming*, pp. 311–326. Springer-Verlag (1991)
25. Pearce, D.: A new logical characterization of stable models and answer sets. In: *Non-monotonic Extensions of Logic Programming*, pp. 57–70. Springer-Verlag (1997)
26. Przymusiński, T.C.: Static semantics for normal and disjunctive logic programs. *Ann. Math. Artif. Intell.* **14**(2–4), 323–357 (1995)
27. Van Emden, M.H., Kowalski, R.A.: The semantics of predicate logic as a programming language. *J. Assoc. Comput. Mach.* **23**(4), 733–742 (1976)