Disjunctive logic programs, answer sets, and the cut rule

Eric A. Martin

University of New South Wales, Australia
emartin@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-1011
March 2010

THE UNIVERSITY OF
NEW SOUTH WALES

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

Abstract

In [11], Minker proposed a semantics for negation-free disjunctive logic programs that offers a natural generalization of the fixed point semantics for definite logic programs. We show that this semantics can be further generalized for disjunctive logic programs with classical negation, in a constructive modal-theoretic framework where rules are built from *assertions* 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.

1 Introduction

1.1 Disjunctive logic programs, fixed points, and negation

As noted in [13], 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 [11], which offered, in particular, a natural generalization of the well-known fixed point semantics for definite logic programs given in [19]. The key feature of that generalized 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 [15] in particular, that the generation process at work in [11] 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 generalization to the immediate consequence operator associated with the fixed point semantics studied in [11], targeted at 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 [11] already establish a relationship between their fixed-point semantics and the generalized 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 the range of issues raised by the presence of negation in logic programs: in [9], 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, 16] 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 [11] when classical negation enters the stage? A technique to reduce

programs with classical negation to programs without is proposed in [9, 17] and discussed in [12]; 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 in order 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 [11], slightly generalised to accepting rules with 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 [11] generalized 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 [11], 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, 10], [18] 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 [18] to a third form of negation, denoted $\sim$ and referred to as "strong" negation. Based on a relation of minimal model entailment, [18, 5] defines 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. Disjunctive logic programs will be modal disjunctive sets of rules, and the least fixed point of such a set of rules $\mathcal{P}$ will consist precisely of the disjunctions that are logical consequences of $\mathcal{P}$. In contrast to [18] 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 $\Diamond\Box\wp$. One of the key properties of the proposed syntax of rules is that disjunction is constructive. In [18], 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 not only with respect to $\neg$, but also

with respect to *not*, as $\Box p \vee \Diamond\Box\neg p$ is not valid.

1.2 Answer sets and program transformation

To obtain an answer set from a logic program, the Gelfond-Lifschitz transformation [8] amounts to forcing $not\,\varphi$ to hold for some literals φ. In our framework, this corresponds to complementing the (modal version of the) logic program with a set of formulas of the form $\Diamond\Box\varphi$. More precisely, the answer set semantics imposes a condition that translates into: if $\Diamond\Box\varphi$ occurs in the (modal version of the) logic program, then that program should be complemented with $\Diamond\Box\varphi$ so that the resulting set of formulas does not logically imply $\Box\neg\varphi$ (which will logically imply $\Box\Diamond\neg\varphi$, 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. 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, hence we will not say anything more on the well-founded semantics. We can describe our plan here as:

- start from a disjunctive logic program $\mathcal{P}$ written in a modal language with classical negation as only form of negation, and possibly complement $\mathcal{P}$ with a set H of so-called hypotheses, that is, formulas of the form $\Diamond\Box\varphi$;
- apply a form of the cut rule to $\mathcal{P} \cup H$ to determine a least fixed point, or equivalently, a set of logical consequences of $\mathcal{P} \cup H$, that captures a particular semantics of $\mathcal{P}$ depending on the constraints imposed on H, the case $H = \varnothing$ corresponding to the "base semantics";
- consider the case where H is chosen in such a way that we obtain the answer set semantics.

This plan has to be slightly amended, in that the cut rule will not be applied to $\mathcal{P}$, but to a closure of $\mathcal{P}$ under an operation that creates more rules, while preserving logical validity (the created rules will be logical consequences of those from which they originate). So it 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, a rule of the form

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

will produce rules of the following form.

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

This operation has been studied in [6] to transform a disjunctive program into a normal program. In our framework, the original set of rules is complemented with formulas obtained by left-shift. When dealing with normal programs, the answer-set semantics imposes that the literals that can be assumed not to hold be those that are preceded with *not* in the bodies of some rules. 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, will expand this pool of literals as needed, no more, no less. 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 [11] generalized to disjunctive logic programs with classical negation.

We process as follows. In Section 2, we motivate the notions to be introduced in the rest of the paper. In Section 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 Section 4, we formalize the left shift operation and establish the relationships between this framework and the answer set semantics; the rest of the paper can then applied to the answer set semantics as a particular case. In Section 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 Section 6, we establish the completeness of the system of proof based on applications of the form of the cut rule we use as a counterpart to the immediate consequence operator described in [11]. 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. Here is an example of such a set of rules $\mathcal{P}$:

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

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 $\mathcal{P}$, or 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, yielding 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 $[\mathcal{P}]_{\alpha\mathrm{Ord}}$ 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?

(A) 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.

(B) At any stage, rules can be considered individually, independently of any others, to determine what to generate at the next stage.

(C) 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

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

to express that

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

is a logical consequence of

$$\varphi_1 \wedge \ldots \wedge \varphi_n \to \psi_1 \vee \ldots \vee \psi_m \vee \xi$$

and

$$\xi \wedge \varphi'_1 \wedge \ldots \wedge \varphi'_{n'} \to \psi'_1 \vee \ldots \vee \psi'_{m'}.$$

We will go trough 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

$$(*) \qquad \frac{\vdash \xi_0 \quad \ldots \quad \vdash \xi_k \qquad \varphi \vdash \psi}{\vdash \psi}$$

where $k \in \mathbb{N}$, $\xi_0, \ldots, \xi_k, \psi$ are atoms, φ is a positive propositional formula with at least one occurrence of each of $\xi_0, \ldots, \xi_k$, and $\varphi[\xi_0/\mathit{true}, \ldots, \xi_k/\mathit{true}]$, that is, the result of making all occurrences of $\xi_0, \ldots, \xi_k$ 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 $(*)$ for the set of rules $\mathcal{P}$ defined above, p_{10} is added to $[\mathcal{P}]$ by the following application of $(*)$:

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

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

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

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.

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

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 φ_1, φ_2, φ_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 *assertion* any

formula of the form $\Box\varphi$ where φ is a literal, and *assertive body*[1], with no a priori reference to any particular rule, any formula obtained from the set of assertions by arbitrary application of conjunction and disjunction. So we now consider sets of rules whose left hand sides are assertive bodies and whose right hand sides are assertions.

Following on from our example, we now let $\mathcal{P}$ denote:

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

We also redefine $[\mathcal{P}]$ as the set of assertions that are logical consequences of $\mathcal{P}$. The contrapositive of the implication $\Box\neg p_{11} \to \Box\neg p_3$ is a formula that is logically equivalent to $\Diamond p_3 \to \Diamond p_{11}$, 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

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

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 finite 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 $(*)$ with the only difference that $\xi_0, \ldots, \xi_k, \psi$ are assertions rather than atoms, and φ is an assertive body rather than a positive propositional formula.

2.3 Disjunction

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

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

[1] We do not say more simply *body* because we keep that expression for a more general notion, to be introduced later.

And now, we let $[\mathcal{P}]$ denote the set of heads that are logical consequences of $\mathcal{P}$. Clearly, we then have to add $\Box p_{15}$, $\Box\neg p_{16}$, $\Box p_{17}$ and $\Box p_{20}$ to $[\mathcal{P}]$. But the very idea of rules that read from left to right and fire individually seems to break down. Consider first the assertions $\Box p_{15}$ and $\Box p_{17}$. Each of them is inferred from two generated heads ($\Box\neg p_3 \vee \Box p_{15}$ 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 $\mathcal{P}$ under a *left shift operation*, that moves assertions from right to left as exemplified below, in a way that preserves the logical validity of the original set of rules and captures 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 $\Box p_{10}$:

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

When we define more formally the syntax of a rule, we will, for good reasons, take disjunction and conjunction as operators on sets, which is why we assumed that all assertions that occur in a head are pairwise distinct; so we do not have to be concerned that a rule such as $\Box\neg p \rightarrow \Box p$—which could be obtained from $\rightarrow \Box p \vee \Box p$ by left shift would the latter be an admissible rule—does not allow one to derive $\Box p$ 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 assertions of the form $\Box p$ and $\Box\neg p$, to the "generic" contradiction that emerges when the empty head (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 assertion $\Box p_{20}$. 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 generalized from $(*)$ to a form that brings it closer to $(\star)$: we now let it take the form

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

where

- $k, n_0, \ldots, n_k, n, m \in \mathbb{N}$, $\xi_0^0, \ldots, \xi_0^{n_0}$ are pairwise distinct assertions, $\ldots$, ξ_k^0, $\ldots$, $\xi_k^{n_k}$ are pairwise distinct assertions, ψ_0, $\ldots$, ψ_n are pairwise distinct assertions,
- φ is an assertive body with at least one occurrence of each of $\xi_0^{n_0}$, $\ldots$, $\xi_k^{n_k}$,
- $\varphi[\xi_0^{n_0}/\mathit{true}, \ldots, \xi_k^{n_k}/\mathit{true}]$ is logically valid, and
- $\xi_1, \ldots, \xi_m$ are the pairwise distinct members of $\{\xi_i^j \mid i \leq k,\, j < n_i\}$ that do not belong to $\{\psi_0, \ldots, \psi_n\}$.

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

$$\dfrac{\dfrac{\vdash \Box p_{18} \vee \Box p_{19} \qquad \Box p_{18} \vdash \Box p_{20}}{\vdash \Box p_{19} \vee \Box p_{20}} \qquad \Box p_{19} \vdash \Box p_{20}}{\vdash \Box p_{20}}$$

It is easy to see that the logically strongest members of $[\mathcal{P}]$ can be obtained by successive applications of (†) on the closure of $\mathcal{P}$ under left shift, validating properties (A)–(C) above.

One might object that the order of the assertions 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 assertions 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 (†) 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 $\{\xi_1, \ldots, \xi_n\}$ or as $\bigvee\{\xi_1, \ldots, \xi_n\}$ rather than as $\xi_1, \ldots, \xi_n$, 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 assertions only? As we work in a constructive paradigm, and as for all literals φ, $\Box\varphi \vee \Diamond\neg\varphi$ is 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 an assertion). 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 of no, two or more literals. 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 $\mathcal{R}^\star$ 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* not followed by $\neg$ with $\Diamond\Box\neg$.
- Replace every occurrence of *not* $\neg$ with $\Diamond\Box$.

For instance, this technique transforms the extended-normal program $\mathcal{R}$

$$
\begin{array}{rcl}
p_2 \wedge p_3 & \rightarrow & p_1 \\
p_4 & \rightarrow & p_2 \\
p_3 & \rightarrow & p_3 \\
not\, p_3 & \rightarrow & p_4
\end{array}
\qquad\qquad
\begin{array}{rcl}
not\, \neg p_2 \vee not\, p_4 & \rightarrow & \neg p_1 \\
\neg p_3 & \rightarrow & \neg p_2 \\
\neg p_3 \wedge not\, \neg p_2 & \rightarrow & \neg p_3
\end{array}
$$

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

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

Call *body*, with no a priori reference to any particular rule, any formula obtained from the set of assertions 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 bodies as left hand sides, and heads as right hand sides—let us refer to them as *general rules*.

Given an extended-normal program $\mathcal{R}$, let us examine the relationship between $\mathcal{R}$ and the associated general set of rules $\mathcal{R}^\star$. 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, $\{\neg p_1, p_2, p_4\}$, and the logical consequences of $\mathcal{R}^\star \cup \{\Diamond\Box p_2, \Diamond\Box \neg p_3\}$ are $\Box\neg p_1$, $\Box p_2$ and $\Box p_4$. 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 $\Box\varphi$, $\varphi \in X$, are the assertions that are logical consequences of $\mathcal{R}^\star \cup H$ where H is a set of hypotheses with the following properties.

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

Obviously, the previous relationship between $\mathcal{R}$ and $\mathcal{R}^\star$ does not generalize to extended-disjunctive programs. For instance, the extended-disjunctive program consisting of $\rightarrow p \vee q$ only has $\{p\}$ and $\{q\}$ as answer sets, and neither $\{\Box p\}$ nor $\{\Box q\}$ is the set of assertions 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 assertions that move from right to left become hypotheses rather than assertions, 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 general rules that follow.

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

The resulting set of general rules can obviously be complemented with $\{\Diamond\Box\neg p\}$ or $\{\Diamond\Box\neg q\}$ so that the set of assertions 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 where instead of considering $\mathcal{R}^\star$, we consider the closure of $\mathcal{R}^\star$ under hypothetical left shift. Of course, the rules introduced by hypothetical left shift might not be of any use. For instance, the extended-disjunctive program

$$\rightarrow\ p \vee q \qquad\qquad p \rightarrow q \qquad\qquad q \rightarrow p$$

has $\{p, q\}$ as unique answer set, and $\{\Box p, \Box q\}$ is the set of assertions 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 a set of general rules $\mathcal{P}$ and a set H of hypotheses be given. Let $[\mathcal{P}, H]$ denote the set of all heads that are logical consequences of $\mathcal{P} \cup H$. Is there a form of the cut rule that can be applied to a closure of $\mathcal{P}$ and H and generate $[\mathcal{P}, H]$, 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 $\mathcal{P}$, thanks to the first version of the left shift operator and the version of the cut rule given by (†). We will see that there is also a positive answer in the general case. The closure of $\mathcal{P}$ and H will be obtained by hypothetical left shift and replacement of all occurrences of the members of H in the bodies of the general rules so obtained by *true*. The version of the cut rule to be used is what has been described as (†), except that φ has to be assumed to be a body rather than an assertive body, and the requirement that $\varphi[\xi_0^{n_0}/\mathit{true}, \ldots, \xi_k^{n_k}/\mathit{true}]$ be logically valid has to be replaced by the requirement that

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

be logically valid: we set to *true* in the body of the general rule that is the target of the cut all hypotheses and assertions built from one of the k literals to which the cut applies.

2.5 Dealing properly with substitution and validity

The form of the cut rule we have eventually converged to somehow leaves to be desired: eliminating the left hand side of the selected general rule by turning it into a valid formula thanks to substitution of assertions 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 assertions a nonissue. Note now that there is no need to introduce a propositional constant *true* as $\bigwedge \varnothing$ is logically valid, and that $\bigvee \varnothing$ is logically invalid. This will be useful to avoid empty left or right hand sides in a general rule, which is formally sloppy. But the key point is that we can replace the requirement that

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

by the requirement that

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

For an example, let φ be

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

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

$$\bigwedge\big\{\bigwedge\varnothing, \bigvee\{\Box p_3, \bigwedge\varnothing, \Box p_5\}, \bigvee\{\Box p_6, \bigwedge\varnothing\big\},$$

$\bigwedge\big\{\bigwedge\varnothing\big\}$ and eventually $\bigwedge\varnothing$. What we have described is a mechanical, syntactic, proof-theoretic way guaranteed to derive $\bigwedge\varnothing$ from a body in which the occurrences of some assertions and hypotheses have been replaced by $\bigwedge\varnothing$ whenever the resulting formula is logically valid.

3 Logical background

3.1 Assertions, hypotheses, disjunctive logic programs

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

Definition 1. A *vocabulary* is a countable set of nullary predicate symbols.

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

- All *assertions* (*over* $\mathcal{V}$), namely, all expressions of the form $\Box\varphi$ with φ a literal over $\mathcal{V}$, are bodies.
- All *hypotheses* (*over* $\mathcal{V}$), namely, all expressions of the form $\Diamond\Box\varphi$ with φ a literal over $\mathcal{V}$, are bodies.
- All expressions of the form $\bigvee X$ with X a countable set of bodies over $\mathcal{V}$, are bodies.

- All expressions of the form $\bigwedge X$ with X a finite set of bodies over $\mathcal{V}$, are bodies.

A few remarks about Definition 3 are in order. First, note that all bodies 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 head: rather than considering a set of rules of the form $\{\Box p_i \to \Box q \mid i \in \mathbb{N}\}$, we will prefer the single rule $\bigvee\{\Box p_i \mid i \in \mathbb{N}\} \to \Box q$. Infinite vocabularies and infinite sets of rules are natural if one thinks of propositionalizing a set of first order (modal) rules. For instance, the previous set of rules could be obtained by propositionalizing the first-order rule $\exists x \Box p(x) \to \Box q(0)$ in a setting where all intended interpretations are standard and the set of closed terms is equal to the set of numerals $\{\overline{n} \mid n \in \mathbb{N}\}$; then one would map $p(\overline{n})$ to p_n for all $n \in \mathbb{N}$, $q(\overline{0})$ 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 propositionalization. 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 bodies $\varphi_1, \ldots, \varphi_n$, we use $\varphi_1 \vee \cdots \vee \varphi_n$ and $\varphi_1 \wedge \cdots \wedge \varphi_n$ as abbreviations for $\bigvee\{\varphi_i \mid 1 \leq i \leq n\}$ and $\bigwedge\{\varphi_i \mid 1 \leq i \leq n\}$, respectively.

Notation 4. We denote by $\mathrm{Alt}(\mathcal{V})$ the set of finite sets of literals over $\mathcal{V}$.

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

Notation 5. For all members D of $\mathrm{Alt}(\mathcal{V})$, 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 bodies of all rules that have a common head, then one is led to define the sets of rules that are our object of study as follows, using both an "implicit" representation and an "explicit" one.

Definition 6. We define a *disjunctive logic program* (*over* $\mathcal{V}$) as an $\mathrm{Alt}(\mathcal{V})$-family of bodies over $\mathcal{V}$.

Definition 7. We call *rule* (*over* $\mathcal{V}$) any expression of the form $\varphi \to \bigvee \Box D$ where φ is a body over $\mathcal{V}$ and D a member of $\mathrm{Alt}(\mathcal{V})$; we call φ the *body of the rule* and $\bigvee \Box D$ the *head of the rule*.

Definition 8. Let a disjunctive logic program $\mathcal{P} = (\varphi_D)_{D \in \mathrm{Alt}(\mathcal{V})}$ be given.

The *logical form of* $\mathcal{P}$ is defined as the set of rules

$$\Big\{\varphi_D \to \bigvee \Box D \;\Big|\; D \in \mathrm{Alt}(\mathcal{V})\Big\}.$$

Notation 9. Given a disjunctive logic program $\mathcal{P}$, we let $\mathrm{LF}(\mathcal{P})$ denote the logical form of $\mathcal{P}$.

Notation 10. Given a disjunctive logic program $\mathcal{P} = (\varphi_D)_{D\in\mathrm{Alt}(\mathcal{V})}$, we let $\mathrm{Hyp}(\mathcal{P})$ denote the set of hypotheses that occur in $\{\varphi_D \mid D \in \mathrm{Alt}(\mathcal{V})\}$.

3.2 Semantics

The formulas that make up the logical form of a disjunctive logic 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 of course, it would be perfectly possible to embed the notions defined in this section into a full fledged semantics.

Definition 11. Let X be a set of claims and hypotheses. 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 12. A set X of claims and hypotheses 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.

Basically, a consistent set of claims and hypotheses is closed if it is closed under logical consequence (w.r.t. the set of all claims and hypotheses).

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

Definition 13. Let H be a set of hypotheses. A set X of claims and hypotheses 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 claims and hypotheses is *complete* just in case, denoting by H the set of all hypotheses, X is H-complete.

Property 14. *A complete set of claims and hypotheses is closed.*

Notation 15. We denote by $\mathcal{W}$ the set of all closed sets of claims and hypotheses (over $\mathcal{V}$).

The next definition exploits the remark that precedes Definition 13. If we were not in a modal setting, we could think of a member of $\mathcal{W}$ 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 $\mathcal{W}$ as the "assertion-hypothesis diagram" of a point of a Kripke frame that determines the truth value of any body or rule at that point.

Definition 16. Let a member $\mathfrak{M}$ of $\mathcal{W}$ be given.

For all bodies φ, we inductively define the notion $\mathfrak{M}$ *forces* φ, denoted $\mathfrak{M} \Vdash \varphi$, as follows.

- For all claims and hypotheses φ, $\mathfrak{M} \Vdash \varphi$ iff $\varphi \in \mathfrak{M}$.
- For all countable sets X of bodies, $\mathfrak{M} \Vdash \bigvee X$ iff $\mathfrak{M}$ forces some body in X.
- For all finite sets X of bodies, $\mathfrak{M} \Vdash \bigwedge X$ iff $\mathfrak{M}$ forces all bodies in X.

For all rules φ, we say that $\mathfrak{M}$ *forces* φ, denoted $\mathfrak{M} \Vdash \varphi$, iff either $\mathfrak{M}$ does not force the body of φ or $\mathfrak{M}$ forces the head of φ.

If $\mathfrak{M}$ does not force a body or a rule φ then we write $\mathfrak{M} \nVdash \varphi$

Given a set T of bodies or rules, we write $\mathfrak{M} \Vdash T$ if $\mathfrak{M}$ forces all members of T, and $\mathfrak{M} \nVdash T$ otherwise.

Definition 17. Given two sets of bodies or rules T and X, we say that X *is a logical* $\mathcal{W}$*-consequence of* T, or that T *logically* $\mathcal{W}$*-implies* X, and we write $T \vDash_{\mathcal{W}} X$, just in case every member of $\mathcal{W}$ that forces T forces X.

If a set of bodies or rules T does not logically $\mathcal{W}$-imply a set of bodies or rules X then we write $T \nvDash_{\mathcal{W}} X$.

The same terminology and notation applies if one or both sets of bodies are replaced by a body or a rule.

Definition 18. Given two sets of bodies or rules T_1 and T_2, we say that T_1 *and* T_2 *are logically* $\mathcal{W}$*-equivalent in* just in case T_1 logically $\mathcal{W}$-implies T_2 and T_2 logically $\mathcal{W}$-implies T_1.

Definition 19. Let a disjunctive logic program $\mathcal{P}$ and a set H of hypotheses. We say that $\mathcal{P}$ is *is* $\mathcal{W}$*-consistent with* H just in case some member of $\mathcal{W}$ forces $H \cup \mathrm{LF}(\mathcal{P})$; otherwise we say that $\mathcal{P}$ is $\mathcal{W}$*-inconsistent with* H.

Definition 20. A body is said to be $\mathcal{W}$*-valid* iff it is logically $\mathcal{W}$-equivalent to $\bigwedge \varnothing$.

4 Left shift completions and answer sets

The next definition captures the notion of hypothetical left shift discussed in Section 2, here defined up to logical $\mathcal{W}$-equivalence, which suffices for our pur-

poses. A left shift completion of a disjunctive logic program $\mathcal{P}$ should be thought of as the closure of $\mathcal{P}$ under hypothetical left shift.

Definition 21. Let a disjunctive logic program $\mathcal{P} = (\varphi_D)_{D \in \mathrm{Alt}(\mathcal{V})}$ be given. A *left shift completion of* $\mathcal{P}$ is a disjunctive logic program $\mathcal{P}' = (\varphi'_D)_{D \in \mathrm{Alt}(\mathcal{V})}$ such that for all $D \in \mathrm{Alt}(\mathcal{V})$, φ'_D is logically $\mathcal{W}$-equivalent to

$$\bigvee \Big\{ \bigwedge \{\varphi_{D'}\} \cup \Diamond\Box{\sim}(D' \setminus D) \;\Big|\; D' \supseteq D \Big\}$$

Proposition 22. *Two disjunctive logic programs such that one is a left shift completion of the other are logically $\mathcal{W}$-equivalent.*

Proof. Let two disjunctive logic programs $\mathcal{P}$ and $\mathcal{P}'$ be such that $\mathcal{P}'$ is a left shift completion of $\mathcal{P}$. Write $\mathcal{P} = (\varphi_D)_{D \in \mathrm{Alt}(\mathcal{V})}$ and $\mathcal{P}' = (\varphi'_D)_{D \in \mathrm{Alt}(\mathcal{V})}$. Let $D \in \mathrm{Alt}(\mathcal{V})$ be given. Since φ'_D is logically $\mathcal{W}$-equivalent to a body of the form $\bigvee\{\varphi_D\} \cup X$, $\varphi'_D \rightarrow \bigvee \Box D$ logically $\mathcal{W}$-implies $\varphi_D \rightarrow \bigvee \Box D$. Moreover, for all $D' \in \mathrm{Alt}(\mathcal{V})$ 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 $\mathcal{W}$-implies $\varphi'_D \rightarrow \bigvee \Box D$. We conclude that $\mathcal{P}$ and $\mathcal{P}'$ are logically $\mathcal{W}$-equivalent. □

Recall the discussion in Section 2 on how an answer set program $\mathcal{R}$ can be put into correspondence with a disjunctive logic program $\mathcal{R}^\star$. Clearly, every disjunctive logic program is of the form $\mathcal{R}^\star$ 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 logic programs rather than on the basis of extended-disjunctive programs, resulting in the definition that follows.

Definition 23. Let a disjunctive logic program $\mathcal{P}$ be given. An *answer set for* $\mathcal{P}$ is the set of claims in a member $\mathfrak{M}$ of $\mathcal{W}$ with the following properties.

- $\mathfrak{M}$ forces $\mathrm{LF}(\mathcal{P})$ and is $\mathrm{Hyp}(\mathcal{P})$-complete.
- For all $\mathfrak{N} \in \mathcal{W}$, if $\mathfrak{N}$ forces $\mathrm{LF}(\mathcal{P})$ and is $\mathrm{Hyp}(\mathcal{P})$-complete then the set of assertions in $\mathfrak{N}$ is not strictly included in the set of assertions in $\mathfrak{M}$.

The correspondence, discussed in Section 2, between answer sets and the sets of assertions that are logical $\mathcal{W}$-consequences of the logical form of an associated disjunctive logic program complemented with a set of hypotheses constrained in a particular way, can now be fully formalized and established.

Definition 24. Let a disjunctive logic program $\mathcal{P}$ and a subset H of $\mathrm{Hyp}(\mathcal{P})$ be given. We say that H is a *complete hypothetical extension for* $\mathcal{P}$ iff the set of claims and hypotheses φ such that $\mathrm{LF}(\mathcal{P}) \cup H \models_{\mathcal{W}} \varphi$ is $\mathrm{Hyp}(\mathcal{P})$-complete.

Proposition 25. *Let a disjunctive logic program $\mathcal{P}$, a left shift completion $\mathcal{P}'$ of $\mathcal{P}$, and a set X of claims be given. Then both conditions that follow are equivalent.*

- *X is an answer set for $\mathcal{P}$.*

- *X is the set of claims that are logical $\mathcal{W}$-consequences of the union of* $\mathrm{LF}(\mathcal{P}')$ *with a complete hypothetical extension for* $\mathcal{P}'$.

Proof. Suppose that X is an answer set for $\mathcal{P}$. Let $\mathfrak{M}$ be a member of $\mathcal{W}$ that satisfies both items in Definition 23 and such that X is the set of claims in $\mathfrak{M}$. Let H be the set of all hypotheses $\Diamond\Box\varphi$ in $\mathrm{Hyp}(\mathcal{P}')$ such that $\mathfrak{M} \nVdash \Box{\sim}\varphi$. Note that $H \cap \mathrm{Hyp}(\mathcal{P})$ is equal to the set of hypotheses in $\mathfrak{M} \cap \mathrm{Hyp}(\mathcal{P})$. Hence $\mathfrak{M} \cup H$ forces $\mathrm{LF}(\mathcal{P}) \cup H$, and by Proposition 22, also forces $\mathrm{LF}(\mathcal{P}') \cup H$. Moreover, for all $\mathfrak{N} \in \mathcal{W}$ that force $\mathrm{LF}(\mathcal{P}') \cup H$, $\mathfrak{N}$ is $\mathrm{Hyp}(\mathcal{P})$-complete, hence $X \subseteq \mathfrak{N}$. Hence X is the set of claims that are logical $\mathcal{W}$-consequences of $\mathrm{LF}(\mathcal{P}') \cup H$, and the set of assertions and hypotheses φ such that $\mathrm{LF}(\mathcal{P}') \cup H \models_{\mathcal{W}} \varphi$ is trivially $\mathrm{Hyp}(\mathcal{P}')$-complete.

Conversely, let H be a complete hypothetical extension for $\mathcal{P}'$ such that X is the set of claims that are logical $\mathcal{W}$-consequences of $\mathrm{LF}(\mathcal{P}') \cup H$. Obviously, $X \cup H$ belongs to $\mathcal{W}$, forces $\mathrm{LF}(\mathcal{P})$ by Proposition 22, and is $\mathrm{Hyp}(\mathcal{P})$-complete since $\mathrm{Hyp}(\mathcal{P})$ is a subset of $\mathrm{Hyp}(\mathcal{P}')$. Also, every member $\mathfrak{N}$ of $\mathcal{W}$ that forces $\mathrm{LF}(\mathcal{P})$ and contains H is such that all members of X are assertions in $\mathfrak{N}$. Hence X is an answer set for $\mathcal{P}$. □

5 Tableau proofs

5.1 General strategy

We set to show that given a disjunctive logic program $\mathcal{P}$ and a set H of hypotheses, a modification of the cut rule can be applied to H and any left shift completion of $\mathcal{P}$ and generate all heads that are logical $\mathcal{W}$-consequences of $\mathrm{LF}(\mathcal{P}) \cup H$, in such a way that properties (A)–(C) listed in Section 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 a head of the form $\bigvee \Box D$, $D \in \mathrm{Alt}(\mathcal{V})$, from a disjunctive logic program $\mathcal{P}$ and a set of hypotheses H. We build a tree T whose nodes are labeled with assertions, except for the root and possibly some of the leaves. To every node N in T that has not been declared to be a leaf, we try and select a rule R in the logical form of a left shift completion of $\mathcal{P}$ whose body is seen to be $\mathcal{W}$-valid thanks to H and the assertions that label the nodes on the path from the root of T up to N. If R's head is the empty disjunction, then N is given a nonlabelled child that is declared to be a leaf. If R's head is of the form $\bigvee\{\varphi_1, \ldots, \varphi_n\}$ for some nonzero $n \in \mathbb{N}$ and pairwise distinct assertions φ_1, …, φ_n, then N is given n children labeled φ_1, …, φ_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 reduced to its root, 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 $\mathrm{LF}(\mathcal{P})$ is $\mathcal{W}$-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 $\mathrm{LF}(\mathcal{P}) \cup H$, hence that the tableau proof procedure is complete.

5.2 Example

To illustrate both tableau proofs and proofs by cut, consider a disjunctive logic program $\mathcal{P}$ over the vocabulary $\mathcal{V} = \{p_0, \ldots, p_{10}\}$ such that $\mathrm{LF}(\mathcal{P})$ is logically $\mathcal{W}$-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 rules that follow.

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

It is easy to verify that $\Box p_0$ is a logical $\mathcal{W}$-consequence of $\mathrm{LF}(\mathcal{P})$.

Let $(\varphi'_D)_{D \in \mathrm{Alt}(\mathcal{V})}$ be a left shift completion of $\mathcal{P}$. It is clear that there is no point in left shifting an assertion, say $\Box\varphi$, from the right hand side to the left hand side of one of $\mathrm{LF}(\mathcal{P})$'s rules if $\Box{\sim}\varphi$ does not occur on the right hand side of any other rule of $\mathrm{LF}(\mathcal{P})$. To simplify the matter further, let us ignore all left shifts we do not need to perform on $\mathrm{LF}(\mathcal{P})$'s rules in order to be able derive $\Box p_0$ from $\mathcal{P}$ thanks to the tableau proof tree we are about to present. This allows us not to describe φ'_D for all $D \in \mathrm{Alt}(\mathcal{V})$, but only provide, for some members D of $\mathrm{Alt}(\mathcal{V})$, a body that is logically $\mathcal{W}$-equivalent to the disjunction of φ_D with the bodies of the form $\bigwedge \Diamond\Box{\sim}(D' \setminus D)$ for those strict supersets D' of D, if any, such that the rule $\varphi_{D'} \rightarrow \bigvee \Box(D' \setminus D)$ turns out to be useful. These considerations lead to writing down 9 relations of logical $\mathcal{W}$-consequence:

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

and

$$(\Box p_8 \wedge \Diamond\Box p_3 \wedge \Diamond\Box p_7) \vee (\Box p_4 \wedge \Box p_{10} \wedge \Diamond\Box\neg p_9) \quad \vDash_{\mathcal{W}} \quad \varphi'_{\varnothing} \tag{5.1}$$

$$\bigwedge \varnothing \quad \vDash_{\mathcal{W}} \quad \varphi'_{\{p_1,\neg p_2,p_3,p_4,\neg p_4\}} \tag{5.2}$$

$$(\Box p_4 \vee \Box\neg p_4) \wedge \Diamond\Box\neg p_9 \quad \vDash_{\mathcal{W}} \quad \varphi'_{\{\neg p_1,p_{10}\}} \tag{5.3}$$

$$\Box p_1 \quad \vDash_{\mathcal{W}} \quad \varphi'_{\{\neg p_2\}} \tag{5.4}$$

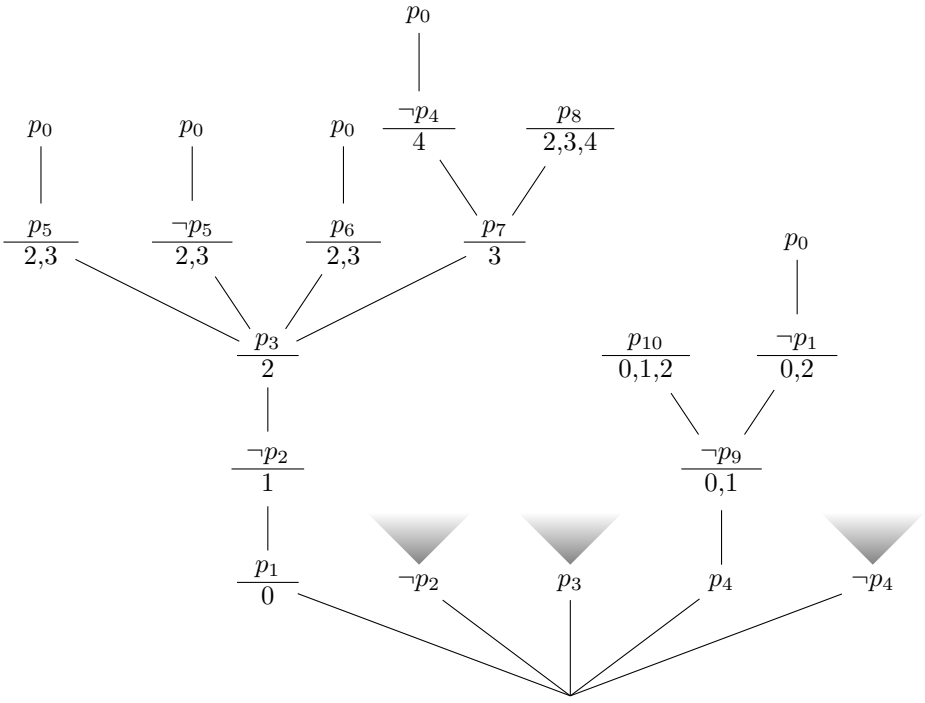
$$\Diamond\Box\neg p_2 \quad \vDash_{\mathcal{W}} \quad \varphi'_{\{p_3\}} \tag{5.5}$$

$$\Box p_6 \vee \Box p_7 \quad \vDash_{\mathcal{W}} \quad \varphi'_{\{\neg p_4,p_8\}} \tag{5.6}$$

$$\Diamond\Box p_3 \quad \vDash_{\mathcal{W}} \quad \varphi'_{\{p_5,\neg p_5,p_6,p_7\}} \tag{5.7}$$

$$\bigwedge \varnothing \quad \vDash_{\mathcal{W}} \quad \varphi'_{\{\neg p_9\}} \tag{5.8}$$

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 assertions. 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 mem-

ber of index 0 is p_1, and $\{\Box p_1\}$ is a $\subseteq$-minimal set of assertions that logically $\mathcal{W}$-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 assertions that logically $\mathcal{W}$-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 assertions that logically $\mathcal{W}$-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 $(p_1, \neg p_2, p_3, p_0)$ 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 assertions that logically $\mathcal{W}$-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 assertions that logically $\mathcal{W}$-implies $\varphi'_{\varnothing}$, 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.

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 26. Given a sequence σ, we denote by $\mathrm{rng}(\sigma)$ the set of members of σ, by $\mathrm{lt}(\sigma)$ the length of σ, and in case σ is not the empty sequence, written $()$, by $\mathrm{lst}(\sigma)$ 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 $n \leq \mathrm{lt}(\sigma)$, we denote by $\sigma_{|n}$ the initial segment of σ of length n. Given a sequence σ and $n \in \mathbb{N}$ with $n < \mathrm{lt}(\sigma)$, 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 27. 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. 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 28. Given a set X, a tree T over X, and a member σ of T, we let $\mathrm{Succ}_T(\sigma)$ denote the set of all $x \in X$ with $\sigma \star x \in T$.

We can now define tableau proofs in accordance with the semi-formal description given at the beginning of the section.

Definition 29. Let a disjunctive logic program $\mathcal{P} = (\varphi_D)_{D\in\mathrm{Alt}(\mathcal{V})}$, a set H of hypotheses, and a member D of $\mathrm{Alt}(\mathcal{V})$ be given. A *tableau proof of D from $\mathcal{P}$ and H* is a nonempty finite tree T over the set of literals with the following properties.

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

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

Proposition 30. *Let a disjunctive logic program $\mathcal{P}$, a set H of hypotheses, a left shift completion $\mathcal{P}'$ of $\mathcal{P}$, and $D \in \mathrm{Alt}(\mathcal{V})$ be given. Then $H \cup \mathrm{LF}(\mathcal{P}) \models_{\mathcal{W}} \bigvee \Box D$ iff there exists a tableau proof of D from $\mathcal{P}'$ and H.*

Proof. Let T be a tableau proof of D from $\mathcal{P}'$ 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 $X_1, \ldots, X_n$ be sets of nodes of T such that $X_1 = \mathrm{Succ}_T(())$ and for all nonzero $i < n$, there exists $\sigma \in X_i$ such that

- either σ is an inner node of T and $X_{i+1} = X_i \cup \{\sigma\star\xi \mid \xi \in \mathrm{Succ}_T(\sigma)\} \setminus \{\sigma\}$,
- 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 $H \cup \mathrm{LF}(\mathcal{P}) \models_{\mathcal{W}} \bigvee \Box D$, it suffices to prove by Proposition 22 that the set $H \cup \mathrm{LF}(\mathcal{P}')$ logically $\mathcal{W}$-implies $\bigvee\{\bigwedge \Box\mathrm{rng}(\sigma) \mid \sigma \in X_n\}$ (from which we can derive that $\mathcal{P}$ is $\mathcal{W}$-inconsistent with H in case X_n is empty). We prove by induction that for all nonzero $i \leq n$, $H \cup \mathrm{LF}(\mathcal{P}')$ logically $\mathcal{W}$-implies the disjunction $\bigvee\{\bigwedge \Box\mathrm{rng}(\sigma) \mid \sigma \in X_i\}$. It is immediately verified that $H \cup \mathrm{LF}(\mathcal{P}') \models_{\mathcal{W}} \bigvee\{\bigwedge \Box\mathrm{rng}(\sigma) \mid \sigma \in X_1\}$ (including if $X_1 = \varnothing$, which can only be the case if $\mathcal{P}$ is $\mathcal{W}$-inconsistent with H). Let a nonzero $i < n$ be given, and assume that $H \cup \mathrm{LF}(\mathcal{P}') \models_{\mathcal{W}} \bigvee\{\bigwedge \Box\mathrm{rng}(\sigma) \mid \sigma \in X_i\}$. Assume that $\mathcal{P}'$ is $\mathcal{W}$-consistent with H, and let $\mathfrak{M} \in \mathcal{W}$ force $H \cup \mathrm{LF}(\mathcal{P}')$. Suppose that X_{i+1} is of the form $X_i \cup \{\tau \star \xi \mid \xi \in \mathrm{Succ}_T(\tau)\} \setminus \{\tau\}$ for some inner node τ of T. It is immediately verified that if $\mathfrak{M} \nVdash \Box\mathrm{rng}(\tau)$ then $\mathfrak{M} \Vdash \bigvee\{\bigwedge \Box\mathrm{rng}(\sigma) \mid \sigma \in X_i \setminus \{\sigma\}\}$, and if $\mathfrak{M} \Vdash \Box\mathrm{rng}(\tau)$ then $\mathfrak{M} \Vdash \bigvee\{\bigwedge \Box\mathrm{rng}(\tau \star \xi) \mid \xi \in \mathrm{Succ}\,\tau\}$, hence $\mathfrak{M}$ forces $\bigvee\{\bigwedge \Box\mathrm{rng}(\sigma) \mid \sigma \in X_{i+1}\}$. 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. As $\mathfrak{M}$ does not force $\bigwedge \varnothing \to \bigvee \varnothing$, we infer that $\mathfrak{M} \nVdash \Box\mathrm{rng}(\tau)$, hence again, $\mathfrak{M} \Vdash \bigvee\{\bigwedge \Box\mathrm{rng}(\sigma) \mid \sigma \in X_{i+1}\}$, as wanted.

Conversely, assume that $H \cup \mathrm{LF}(\mathcal{P}) \models_{\mathcal{W}} \bigvee \Box D$. Fix an enumeration $(D_i)_{i\in\mathbb{N}}$ of $\mathrm{Alt}(\mathcal{V})$. Set $\mathcal{P}' = (\varphi_E)_{E\in\mathrm{Alt}(\mathcal{V})}$. Define a sequence $(T_n)_{n\in\mathbb{N}}$ of trees over the set of literals as follows. Set $T_0 = \{()\}$. 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 $\mathrm{rng}(\sigma) \cap D \neq \varnothing$ then no strict extension of σ belongs to T_{n+1}. Suppose that $\mathrm{rng}(\sigma) \cap D = \varnothing$. If there exists a least $i \in \mathbb{N}$ such that $H \cup \Box\mathrm{rng}(\sigma) \cup \Box D_i$ is consistent, $\varphi_{D_i}[H \cup \mathrm{rng}(\sigma)]$ is $\mathcal{W}$-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 $\mathcal{P}'$ 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 $\mathrm{rng}(\sigma) \cap D = \varnothing$ then $\varphi_{\varnothing}[H \cup \mathrm{rng}(\sigma)]$ is $\mathcal{W}$-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 $\mathfrak{M}$ be the $\subseteq$-minimal member of $\mathcal{W}$ that contains $H \cup \Box X$. Note that X contains no member of D whether B is finite or not, and so $\mathfrak{M}$ does not force $\bigvee \Box D$, whether D is empty or not. Let $i \in \mathbb{N}$ be such that $\mathfrak{M}$ 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 $\mathcal{P}'$, $\mathfrak{M}$ 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:

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

If $D_j = \varnothing$ then B clearly ends in τ, which contradicts the assumption that (2) above does not hold. If $D_j \neq \varnothing$ then $\tau \star \xi$ belongs to B for some $\xi \in D_j$ and $\mathfrak{M}$ forces $\bigvee \Box D_j$. So if $\mathfrak{M}$ does not force $\varphi_{\varnothing}$ then we infer that $\mathfrak{M}$ forces $H \cup \mathrm{LF}(\mathcal{P}')$, which contradicts the assumption that $H \cup \mathrm{LF}(\mathcal{P}') \models_{\mathcal{W}} \bigvee \Box D$. 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 $\mathcal{P}$ and H. □

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

Corollary 31. *Let a disjunctive logic program $\mathcal{P}$, a left shift completion $\mathcal{P}'$ of $\mathcal{P}$, and a set H of hypotheses be given. Then the following conditions are equivalent.*

- *$\mathcal{P}$ is $\mathcal{W}$-inconsistent with H.*

- *There exists a tableau proof of* $\varnothing$ *from* $\mathcal{P}'$ *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 32. *Let a disjunctive logic program* $\mathcal{P}$*, a left shift completion* $\mathcal{P}'$ *of* $\mathcal{P}$*, and a set* H *of hypotheses be such that* $\mathcal{P}$ *is* $\mathcal{W}$*-consistent with* H*. Let a member* D *of* $\mathrm{Alt}(\mathcal{V})$ *be such that* $H \cup \mathrm{LF}(\mathcal{P}) \vDash_{\mathcal{W}} \bigvee \Box D$*. Then every tableau proof of* D *from* $\mathcal{P}'$ *and* H *has at least one branch that ends in a member of* D*.*

Corollary 33. *Let a disjunctive logic program* $\mathcal{P}$*, a set* H *of hypotheses, and a member* D *of* $\mathrm{Alt}(\mathcal{V})$ *be such that* $H \cup \mathrm{LF}(\mathcal{P}) \vDash_{\mathcal{W}} \bigvee \Box D$*. Let a left shift completion* $\mathcal{P}' = (\varphi_E)_{E \in \mathrm{Alt}(\mathcal{V})}$ *be given. There there exists a finite subset* X *of* $\mathrm{Alt}(\mathcal{V})$ *with the following property. Let* $\mathcal{P}'' = (\varphi_E)_{E \in \mathrm{Alt}(\mathcal{V})}$ *be the disjunctive logic program such that for all* $E \in \mathrm{Alt}(\mathcal{V})$*, either* $E \in X$ *and* $\varphi'_E = \varphi_E$*, or* $E \notin X$ *and* $\varphi'_E = \bigvee \varnothing$*. Then there exists a tableau proof of* D *from* $\mathcal{P}''$ *and* H*.*

6 Proofs by cuts

6.1 General strategy

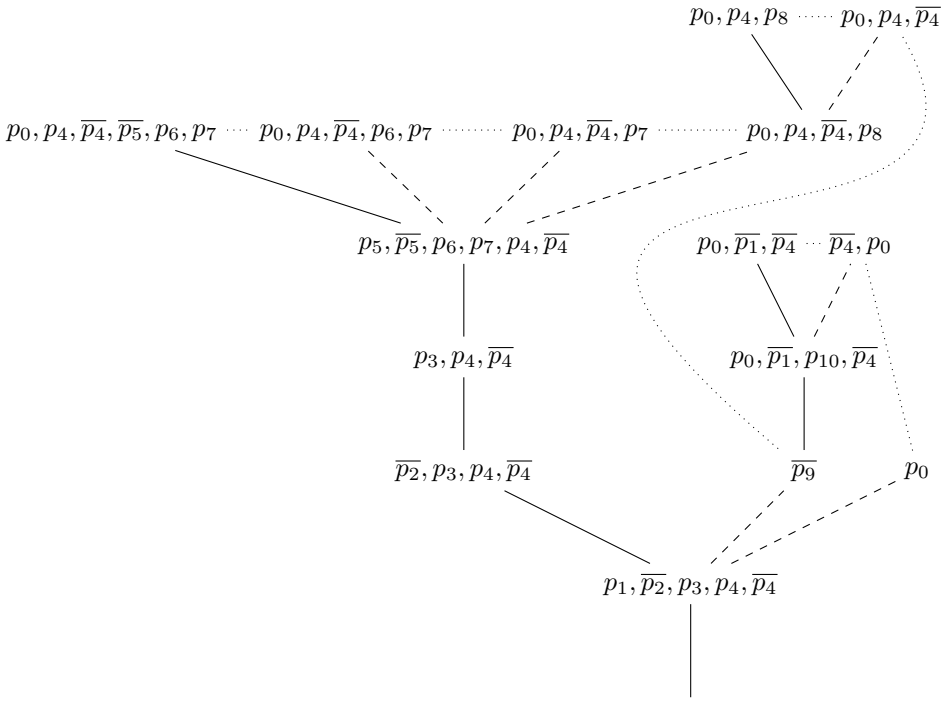
Let us further exploit the example given in the previous section and explain, on the basis of that example, how a tableau proof can be converted into a proof by cuts. Let T be the tree depicted in the previous section, which, recall, represents a tableau proof of $\Box p_0$. The strategy is to explore T depth first and label some nodes N in T with a member of $\mathrm{Alt}(\mathcal{V})$ determined by the (possibly empty) set of N's children and by the labels associated with $N_{|i_1+1}, \ldots, N_{|i_k+k}$ where $\{i_1, \ldots, i_k\}$ 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 exit) that had been previously assigned to the node we backtrack to. The fact that we will eventually obtain a proof of $\Box p_0$ by cut will be captured by the fact that $\Box p_0$ will be the label last assigned to a node. More generally, a proof by cut of a head of the form $\bigvee \Box D$, $D \in \mathrm{Alt}(\mathcal{V})$, 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 $\bigvee \Box D$ for some $D \in \mathrm{Alt}(\mathcal{V})$ 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 $\xi_1, \ldots, \xi_k$ be such that N has

k children in T, those children being $N \star \xi_1$, ..., $N \star \xi_k$. Let $n \in \mathbb{N}$ be the number of integers associated with N in T, and let $i_1, \ldots, i_n$ be those integers. Let $\psi_1, \ldots, \psi_n$ be the last elements of $N_{|i_1}, \ldots, N_{|i_n}$, and let $D_1, \ldots, D_n$ be the labels that have (necessarily) been assigned to $N_{|i_1}, \ldots, N_{|i_n}$, respectively (less precisely, $\psi_1, \ldots, \psi_n$ are the literals on the path from the root of T to N at positions $i_0, \ldots, i_n$, and $D_1, \ldots, D_n$ are the labels currently associated with those positions, respectively). Then $\{\Box\psi_1, \ldots, \Box\psi_n\}$ is a $\subseteq$-minimal set of assertions that logically $\mathcal{W}$-implies $\varphi_{\{\xi_1,\ldots,\xi_k\}}$, ψ_1 belongs to D_1, ..., and ψ_n belongs to D_n. Hence the cut rule can be applied to $\vdash \Box D_1, \ldots, \vdash \Box D_n$ and $\varphi_{\{\xi_1,\ldots,\xi_k\}} \vdash \Box\xi_1, \ldots, \Box\xi_k$. If $\vdash \Box\chi_1, \ldots, \Box\chi_m$ is the consequent of that application of the cut rule, then $\{\chi_1, \ldots, \chi_m\}$ 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 $\overline{\varphi}$.



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 $\varphi'_{\{\neg p_2\}} \to \Box\neg p_2$.

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

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 $\varphi'_{\{p_3\}} \to \Box p_3$.

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

In accordance with a depth-first exploration of T', set

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

Here is how σ_3 is obtained.

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

The node $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of T' is determined by the node $(p_1, \neg p_2, p_3, p_5)$ of T, and more precisely, the associated numbers (namely, 2 and 3) and the successor (namely, p_0) of that node in T: σ_4 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 $\Box p_5 \vee \Box\neg p_5 \vee \Box p_6 \vee \Box p_7 \vee \Box p_4 \vee \Box\neg p_4$, which is currently associated with the element $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of index 3, and to $\varphi'_{\{p_0\}} \to \Box p_0$.

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

Backtracking 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 $\varphi'_{\{p_0\}} \to \Box p_0$.

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

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.

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

Moving on, σ_7 and σ_8 are obtained as follows.

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

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

$(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_7, \sigma_9)$, node of T', is determined by the 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 $\varphi'_{\varnothing} \to \bigvee \varnothing$.

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

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 replacement 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 body of $\varphi'_{\neg p_9}$ is $\bigwedge \varnothing$, no application of the cut rule is necessary to determine σ_{10}. Then σ_{11}, σ_{12} and σ_{13} are determined as follows.

$$\frac{\Box p_0 \vee \Box p_4 \vee \Box\neg p_4 \quad \Box\neg p_9 \quad \varphi'_{\{\neg p_1, p_{10}\}} \to \Box\neg p_1 \vee \Box p_{10}}{\Box p_0 \vee \Box\neg p_1 \vee \Box p_{10} \vee \Box\neg p_4}\, p_4, \neg p_9$$

$$\frac{\Box p_0 \vee \Box p_4 \vee \Box\neg p_4 \quad \Box\neg p_9 \quad \Box p_0 \vee \Box\neg p_1 \vee \Box p_{10} \vee \Box\neg p_4 \quad \varphi'_{\varnothing} \to \bigvee \varnothing}{\Box p_0 \vee \Box\neg p_1 \vee \Box\neg p_4}\, p_4, \neg p_9, p_{10}$$

$$\frac{\Box p_0 \vee \Box p_4 \vee \Box\neg p_4 \quad \Box p_0 \vee \Box\neg p_1 \vee \Box\neg p_4 \quad \varphi'_{\{p_0\}} \to \Box p_0}{\Box p_0 \vee \Box\neg p_4}\, p_4, \neg p_1$$

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 $\mathcal{P}$ logically $\mathcal{W}$-implies $\Box p_0$.

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

6.3 Body reduction

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

Definition 34. Let a body φ be given. We call *reduct of* φ any body 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 $\bigvee X \cup \{\bigwedge \varnothing\}$ and ψ is $\bigwedge \varnothing$.
- φ is of the form $\bigwedge X \cup \{\bigvee \varnothing\}$ and ψ is $\bigvee \varnothing$.
- 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 35. A body is said to be *in reduced form* iff it is its only reduct.

Property 36. *Every body has a unique reduct in reduced form.*

Notation 37. Given a body φ, we let $\rho(\varphi)$ denote the unique body in reduced form that is a reduct of φ.

Property 38. *For all bodies φ, $\rho(\varphi)$ is logically $\mathcal{W}$-equivalent to φ.*

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

Property 39. *Let a body φ and a set X of hypotheses and literals be given. Let $\mathfrak{M}$ be the $\subseteq$-minimal member of $\mathcal{W}$ that contains all hypotheses in X and all assertions of the form $\Box\xi$ with $\xi \in X$. Then $\mathfrak{M}$ forces φ iff $\rho(\varphi[X])$ is equal to $\bigwedge \varnothing$.*

Corollary 40. *For all bodies φ, if φ is $\mathcal{W}$-valid then $\rho(\varphi)$ is $\bigwedge \varnothing$.*

It is time to fix the notation for substitution of assumptions or hypotheses in a body by $\bigwedge \varnothing$.

Notation 41. Let a body φ and a set X of literals by given. We denote by $\varphi[X]$ the result of substituting in φ all occurrences of the assertions and hypotheses of the form $\Box\xi$ or $\Diamond\Box\xi$ with $\xi \in X$ by $\bigwedge \varnothing$. If X is a singleton $\{\psi\}$ then we write $\varphi[\psi]$ rather than $\varphi[X]$.

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

6.4 Completeness of the system of proofs by cuts

What can be derived from a disjunctive logic program and a set of hypotheses by applying the cut rule iteratively is defined in the notation that follows.

Notation 42. Let a disjunctive logic program $\mathcal{P} = (\varphi_D)_{D \in \mathrm{Alt}(\mathcal{V})}$ and a set H of hypotheses be given. We define $[\mathcal{P}, H]$ as the $\subseteq$-minimal set of members of $\mathrm{Alt}(\mathcal{V})$ of the form $D \cup (D_1 \setminus \{\psi_1\}) \cup \cdots \cup (D_k \setminus \{\psi_k\})$ such that

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

The cut rule is valid:

Proposition 43. *For all disjunctive logic programs $\mathcal{P}$, sets H of hypotheses and $D \in [\mathcal{P}, H]$, $\bigvee D$ is a logical $\mathcal{W}$-consequence of $H \cup \mathrm{LF}(\mathcal{P})$.*

Proof. Let a disjunctive logic program $\mathcal{P} = (\varphi_D)_{D\in\mathrm{Alt}(\mathcal{V})}$ and a set H of hypotheses be given. Let $D \in \mathrm{Alt}(\mathcal{V})$, $k \in \mathbb{N}$, members $D_1, \ldots, D_k$ of $[\mathcal{P}, H]$, and literals $\psi_1, \ldots, \psi_k$ in $D_1, \ldots, D_k$, respectively, be such that $\psi_1, \ldots, \psi_k$ all occur in $\varphi_D[H]$ and $\rho\big(\varphi_D[H \cup \{\psi_1, \ldots, \psi_k\}]\big) = \bigwedge \varnothing$. Set $D' = D \cup (D_1 \setminus \{\psi_1\}) \cup \cdots \cup (D_k \setminus \{\psi_k\})$. Let $\mathfrak{M} \in \mathcal{W}$ force $H \cup \mathrm{LF}(\mathcal{P})$ and each of $\bigvee \Box D_1, \ldots, \bigvee \Box D_k$. If $\mathfrak{M}$ does not force all of $\Box\psi_1, \ldots, \Box\psi_k$, then $\mathfrak{M}$ forces $\bigvee D_i \setminus \{\psi_i\}$ for some nonzero $i \leq k$, hence $\bigvee \Box D'$. Suppose that $\mathfrak{M}$ forces all of $\Box\psi_1, \ldots, \Box\psi_k$. Since $\mathfrak{M} \Vdash H$ and $\rho\big(\varphi_D[H \cup \{\psi_1, \ldots, \psi_k\}]\big) = \bigwedge \varnothing$, then $\mathfrak{M}$ forces φ_D. Since also $\mathfrak{M} \Vdash \mathrm{LF}(\mathcal{P})$ then $\mathfrak{M}$ forces $\bigvee \Box D$, hence $\bigvee \Box D'$. □

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

Proposition 44. *Let a disjunctive logic program $\mathcal{P}$, a left shift completion $\mathcal{P}'$ of $\mathcal{P}$, and a set H of hypotheses be given. Then for all $D \in \mathrm{Alt}(\mathcal{V})$, $\bigvee \Box D$ is a logical $\mathcal{W}$-consequence of $H \cup \mathrm{LF}(\mathcal{P})$ iff $[\mathcal{P}', H]$ contains a subset of D.*

Proof. Set $\mathcal{P}' = (\varphi_E)_{E\in\mathrm{Alt}(\mathcal{V})}$. By Proposition 30, let T be a tableau proof of D from $\mathcal{P}'$ 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 $(\sigma_0, \ldots, \sigma_{N-1})$ 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 $\mathrm{spt}(\sigma)$ denote a $\subseteq$-minimal subset of $\{0, \ldots, n-1\}$ such that $\varphi_{\mathrm{Succ}_T(\sigma)}[H \cup \{\sigma(i) \mid i \in \mathrm{spt}(\sigma)\}]$ is $\mathcal{W}$-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 $\mathrm{Alt}(\mathcal{V})$ of length either 0 or $\mathrm{lt}(\sigma_i) + 1$. Set $[\sigma_0] = (\mathrm{Succ}_T(\sigma_0))$. 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 $k \in \{i, \ldots, j\}$. Let j be the least integer such that either $j = N$, or $i \leq j < N$, $\mathrm{lt}(\sigma_j) \leq \mathrm{lt}(\sigma_i)$, and $\mathrm{lst}(\sigma_j) \in \mathrm{lst}([\sigma_{i-1}])$. Then for all $k \in \{i, \ldots, j-1\}$, $[\sigma_k] = ()$. 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 $\mathrm{lt}(\sigma_j) > 1$ then $[\sigma_j]_{|\mathrm{lt}(\sigma_j)-1}$ is equal to $[\sigma_{i-1}]_{|\mathrm{lt}(\sigma_j)-1}$.
- The penultimate element of $[\sigma_j]$ is equal to $\mathrm{lst}([\sigma_{i-1}])$.
- $\mathrm{lst}([\sigma_j])$ is equal to $\mathrm{Succ}_T(\sigma_j) \cup \bigcup\big\{[\sigma_j](n) \setminus \sigma_j(n) \;\big|\; n \in \mathrm{spt}(\sigma_j)\big\}$.

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

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

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 $\mathrm{lt}(\sigma_k) > 1$ 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 $[\sigma_k]_{|\mathrm{lt}(\sigma_k)-1}$, it is easy to verify the following.

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

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

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 $\mathrm{lst}([\sigma_i])$, it is easy to verify the following, wether $\mathrm{lt}(\sigma_k) = \mathrm{lt}(\sigma_i) + 1$ or whether $\mathrm{lt}(\sigma_k) \leq \mathrm{lt}(\sigma_i)$.

(‡) $[\sigma_k](\mathrm{lt}(\sigma_k) - 1)$ is included in

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

Finally, using (†) and (‡) and the definition of $\mathrm{lst}([\sigma_k])$, it is easy to verify that

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

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 $\mathrm{lst}([\sigma_i]) \nsubseteq D$ 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 $\mathrm{lst}([\sigma_i]) \subseteq D$.

To complete the proof of the proposition, it suffices to show that for all $i < N$ with $[\sigma_i] \neq ()$, $\mathrm{lst}([\sigma_i])$ belongs to $[\mathcal{P}', H]$. Proof is by induction. Trivially, $\mathrm{lst}([\sigma_0])$ is a member of $[\mathcal{P}', H]$. Let $i < N$ be such that $[\sigma_i] \neq ()$ and for all $j < i$ with $[\sigma_j] \neq ()$, $\mathrm{lst}([\sigma_j]) \in [\mathcal{P}', H]$. Note that for all $n < \mathrm{lt}(\sigma_i)$, there exists $j < i$ with $[\sigma_j] \neq ()$ and $[\sigma_i](n)$ is equal to $\mathrm{lst}([\sigma_j])$. Let $k \in \mathbb{N}$ denote the cardinality of $\mathrm{spt}(\sigma_i)$, and let $e_1, \ldots, e_k$ enumerate its elements. By definition of $\mathrm{spt}(\sigma_i)$ (and more particularly, the $\subseteq$-minimality condition), $\varphi_{\mathrm{lst}(\sigma_i)}[H]$ contains at least one occurrence of each of $\Box\sigma_i(e_1), \ldots, \Box\sigma_i(e_k)$. Also note that for all nonzero $j \leq k$, $\sigma_i(k)$ belongs to $[\sigma_i](k)$, and that $\varphi_{\mathrm{lst}(\sigma_i)}[H \cup \{[\sigma_i](e_1), \ldots, [\sigma_i](e_k)\}]$ is $\mathcal{W}$-valid. We conclude from the previous observations and the definitions of $[\mathcal{P}', H]$ and $\mathrm{lst}([\sigma_i])$ that $[\mathcal{P}', H]$ contains $\mathrm{lst}([\sigma_i])$, completing the proof of the proposition. □

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 generalization 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.

Bibliography

[1] José Júlio Alferes, Luís Moniz Pereira, and Teodor C. Przymusinski. 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*, pages 143–163. Springer-Verlag, 1996.

[2] Krzysztof R. Apt and Roland Bol. Logic programming and negation: a survey. *Journal of Logic Programming*, 19-20(Supplement 1):9–71, 1994.

[3] Chitta Baral, Jorge Lobo, and Jack Minker. Generalized disjunctive well-founded semantics for logic programs: procedural semantics. *Methodologies for intelligent systems*, 5:456–464, 1990.

[4] Stefan Brass and Jürgen Dix. Semantics of (disjunctive) logic programs based on partial evaluation. *The Journal of Logic Programming*, 40(1):1–46, 1999.

[5] Stefan Brass, Jürgen Dix, and Teodor C. Przymusinski. Super logic programs. *ACM Transactions on Computational Logic*, 5(1):129–176, 2004.

[6] Jürgen Dix, Georg Gottlob, and Wiktor Marek. Reducing disjunctive to non-disjunctive semantics by shift-operations. *Fundamenta Informaticae*, 28(1-2):87–100, 1996.

[7] Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15:289–323, 1995.

[8] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Logic programming: proceedings of the fifth international conference and symposium*, volume 2 of *MIT Press series in logic programming*, pages 1070–1080. MIT Press, 1988.

[9] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[10] Wiktor Marek and Miroslaw Truszczyński. Autoepistemic logic. *Journal of the Association for Computing Machinery*, 38(3):587–618, 1991.

[11] Jack Minker and Arcot Rajasekar. A fixpoint semantics for disjunctive logic programs. *Journal of Logic Programming*, 9(1):45–74, 1990.

[12] Jack Minker and Carolina Ruiz. Semantics for disjunctive logic programs with explicit and default negation. *Fundamenta Informaticae*, 20(1-3):145–192, 1994.

[13] Jack Minker and Dietmar Seipel. Disjunctive logic programming: A survey and assessment. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, pages 472–511. Springer-Verlag, 2002.

[14] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[15] Linh Anh Nguyen and Rajeev Goré. Completeness of hyper-resolution via the semantics of disjunctive logic programs. *Information Processing Letters*, 95(2):363–369, 2005.

[16] David Pearce, Hans Tompits, and Stefan Woltran. Characterising equilibrium logic and nested logic programs: Reductions and complexity1,2. *Theory and Practice of Logic Programming*, 9(5):565–616, 2009.

[17] David Pearce and Gerd Wagner. Logic programming with strong negation. In *Proceedings of the international workshop on Extensions of logic programming*, pages 311–326. Springer-Verlag New York, Inc., 1991.

[18] Teodor C. Przymusinski. Static semantics for normal and disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):323–357, 1995.

[19] M. H. Van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the Association for Computing Machinery*, 23(4):733–742, 1976.