Default Reasoning by Deductive Planning

Michael Thielscher
FG Intellektik, TH Darmstadt
Alexanderstraße 10
D–64283 Darmstadt
Germany
mit@intellektik.informatik.th-darmstadt.de

Torsten Schaub
IRISA
Campus de Beaulieu
F–35042 Rennes Cedex
France
torsten@irisa.fr

Abstract

This paper deals with the automation of reasoning from incomplete information by means of Default Logics. We provide proof procedures for Default Logics' major reasoning modes, namely credulous and skeptical reasoning. We start by reformulating the task of credulous reasoning in Default Logics as deductive planning problems. This interpretation supplies us with several interesting and valuable insights into the proof theory of Default Logics. Foremost, it allows us to take advantage of the large number of available methods, algorithms, and implementations for solving deductive planning problems. As an example, we demonstrate how credulous reasoning in certain variants of Default Logic is implementable by means of a planning method based on equational logic programming. In addition, our interpretation allows us to transfer theoretical results, like complexity results, from the field of planning to that of Default Logics. In this way, we have isolated two yet unknown classes of default theories for which deciding credulous entailment is polynomial.

Our approach to skeptical reasoning relies on an arbitrary method for credulous reasoning. It does neither strictly require the inspection of all extensions nor the computation of entire extensions to decide whether a formula is skeptically entailed. Notably, our approach abstracts from an underlying credulous reasoner. In this way, it can be used to extend existing formalisms for credulous reasoning to skeptical reasoning.

Keywords. default logics, deductive planning, credulous and skeptical reasoning, logic programming

1 Introduction

The treatment of incomplete information constitutes one of the central problems for complex information systems. This issue has been isolated in the field of artificial intelligence by Minsky in [46], who prompted with it the creation of an area, known as *nonmonotonic reasoning*. This term stems from the observation that the addition of information to an incomplete knowledge base may change the set of drawable conclusions. So far, however, this phenomenon has been studied primarily from a theoretic point of view. This has led to numerous different nonmonotonic logics yet only a handful of resulting practical approaches, algorithms or even implementations.

In this paper, we turn to practical issues and consider one of the best known and most-widely used formalisms for nonmonotonic or, more specifically, default reasoning, namely Ray Reiter's *Default Logic* [51] along with its descendants. This logical system deals with incomplete information by providing general rules that allow for exceptional cases. These so-called *default rules* are in turn added to a standard first order logic.

From the very beginning, an important task was the development of proof theories in order to automate reasoning in Default Logic, preferably by adopting and extending methods known from classical automated deduction. However, Reiter himself observed that automating the reasoning

process in the entire framework is problematic because full-fledged Default Logic lacks the formal property of *semi-monotonicity*. This property, however, is indispensable for proving in a local fashion, since it allows us to restrict our attention to those parts of a given theory that are related to what shall be proved. For this purpose, Reiter defines in [51] a restricted class of default theories, called *normal* theories, that are provably semi-monotonic in general. Using this observation, he develops in [51] a first proof theory for this restricted class based on the resolution principle.

Nonetheless, it became apparent soon that many interesting problems cannot be encoded via normal default theories [53]. Moreover, it turned out that apart from semi-monotonicity, other desirable properties are not present in the original approach. This insight prompted several authors to develop modifications of the first approach to Default Logic, e.g. Łukaszewicz' *Justified Default Logic*[1] [39], Brewka's *Cumulative Default Logic* [8], or *Constrained Default Logic* [17]. These three variants turn out to be semi-monotonic even in case of arbitrary default theories. This is why they are of great interest, especially for automating default reasoning. In what follows, however, we will mainly focus on the finally mentioned dialect. The choice of Constrained Default Logic as our prime exemplar is of course not an arbitrary one. Constrained Default Logic enjoys several desirable computational properties needed for reasonable proof procedures. Moreover, it has recently been shown in [18] that in certain fragments of Constrained Default Logic reasoning is significantly easier than in Reiter's Default Logic. All this renders our exemplar a prime candidate for computational purposes. However, we show also how our results can be directly applied to Reiter's original definition in case of normal default theories. Moreover, we illustrate how similar results can be obtained for Łukaszewicz' variant, while we do not explicitly consider Cumulative Default Logic due to its tight relationship to Constrained Default Logic (see [58, 17] for details). An important characteristic feature of Constrained Default Logic, Łukaszewicz' variant, and classical Default Logic restricted to normal theories, is that extensions can be generated in a truly iterative way instead of using the usual fixpoint construction. This observation shall be the starting point of our analysis.

During the last decade, several calculi designed for classical logic have been applied to define proof theories for (variants of) Default Logic, e.g. the resolution principle as in [51, 4], the tableaux method [65] as in [61, 62], or the connection method [6] as in [55, 60]. The aim of this paper is *not* to provide just another specific implementation technique. Rather, we propose a new view on the reasoning task in Default Logics by regarding it as a problem solving task or, more specifically, as a *planning problem*. This view appears to be very natural and straightforward as soon as extensions can be generated truly iteratively. We claim that this interpretation reflects adequately the nature of what distinguishes Default Logics from classical logic, namely the additional expressive power provided by default rules. Any formalism for proving in default theories that employs methods known from classical automated deduction has to comply with three substantial differences between a default $\delta = \frac{\alpha : \beta}{\omega}$ (which allows to conclude ω *by default* if α holds and β can be consistently assumed) and its classical counterpart, viz. the implication $\alpha \rightarrow \omega$. First of all, the consistency requirement given by the so-called *justification* β may suppress the application of δ. Second, as δ is a *rule* instead of a formula, it is impossible to apply it the other way round, i.e. via contraposition. For instance, we are allowed to conclude $\neg a$ from $\neg b$ given $a \rightarrow b$ but not by using the default $\frac{a : b}{b}$. Third, the so-called *prerequisite* α of a default must be explicitly *derivable* which means that defaults cannot mutually satisfy their prerequisites. For example, from $a \rightarrow c$ and $\neg a \rightarrow c$ it is possible to conclude c. This cannot be obtained from the two defaults $\frac{a : c}{c}$ and $\frac{\neg a : c}{c}$.

[1] This variant was originally called *Modified Default Logic* [39].

These observations suggest a careful distinction between a default and a logical formula. To this end, we propose to interpret defaults as tools for acting in dynamically changing worlds. More precisely, starting from some given world knowledge W, which is assumed to be certain, we identify a particular set of actual beliefs with a particular *state* of (or *situation* in) a dynamical system. This set of beliefs is successively extended by applying defaults which are identified with *actions* that transform situations into situations. The task of finding default proofs, which means to find an appropriate set of defaults that provide such a proof, can then be identified with the task of finding an appropriate sequence of actions that transforms the initial set of beliefs into a situation which classically entails the formula under consideration. In other words, we intend to identify default proving with solving planning problems.

Although this view seems to be quite natural and shows some interesting and valuable consequences, it has not yet been formally established. Aside from the adequateness of the interpretation of default rules as actions operating on sets of formulas, some merits of this view are the following ones.

- There exists a variety of systems designed for planning problems which have been thoroughly investigated and improved for many years. All of them are candidates for competitive default reasoning systems, provided they are suitable for the particular problem class determined by Default Logics.

- Pure planning can be regarded as being only a part of a more extensive research field dealing with reasoning about actions and change in general. Other directions of research related to this field may suggest a variety of extensions as regards reasoning with Default Logics. For instance, *abductive planning* is concerned with planning problems where the system may generate and apply additional, i.e. not previously given facts that are necessary to achieve a goal. Adopting this concept, one may define abductive default reasoning where it is possible to abduce additional knowledge if it is necessary to obtain a default proof.

- An important task regarding both default reasoning as well as planning consists in fixing tractable problem classes. A variety of results concerning tractable planning problems have recently been developed. As will be illustrated later in this paper by two simple example, these results can often be directly applied to define subclasses of default theories where the task of finding default proofs shows the very same complexity. Our examples provide new classes of default theories in which deciding credulous entailment is polynomial.

We have not yet explicitly mentioned that the preceding discussion is only concerned with one kind of reasoning in Default Logics, called *credulous* reasoning. In fact, a Default Logic may induce one or several so-called *extensions* (i.e. distinct sets of default conclusions) of an underlying world description. Then, a formula is said to be credulously entailed if it is contained in at least one extension of the default theory at hand. Other extensions may make no statement at all about the formula, or even claim its contrary.[2] Indeed, previous work has mainly concentrated on this reasoning mode.

However, a second kind of reasoning in Default Logics deserves equal rights, namely *skeptical* reasoning. For this, a formula is required to be in all extensions of the default theory under consideration. Apart from the naïve way of checking this by simply generating and testing all extensions, only little effort has been put into automating skeptical reasoning up to this day. As a second major contribution of this paper, we formalize a more elaborated procedure to prove

[2] Curiously enough, a credulously entailed formula is often called a nonmonotonic *theorem* in the literature, despite the fact that it may happen that both a formula and its negation are credulously entailed.

membership in all extensions. Yet our approach does neither require the computation of all extensions nor the computation of a single entire extension. Notably, it relies on an (almost) arbitrary procedure providing credulous proofs. In fact, apart from the provision of certain complete, preferably minimal sets of credulous default proofs, there is no further restriction on the underlying credulous reasoner. In this way, our approach to skeptical reasoning abstracts from the underlying credulous reasoning method. In particular, our method can be founded but does not rely on the results presented in the first part of this paper. Rather it can be used to extend any formalism for automated credulous reasoning to skeptical reasoning. The basic ideas can be traced back to Poole's nonmonotonic THEORIST formalism [50]: First, it was applied to a restricted version of this theory [47, 48] and later extended to the entire framework [67].

The paper is organized as follows. We introduce in Section 2 Reiter's classical approach, Lukaszewicz' modification to it, and Constrained Default Logic. Furthermore, some fundamental observations and results concerning the iterative generation of extensions in Constrained Default Logic are recapitulated. In Section 3, we argue that these results lead in a rather straightforward way to a formalization of credulous reasoning in terms of deductive planning. We discuss the merits of this reformulation and illustrate by means of a simple example how complexity results known from the field of planning can be directly applied to formulate analogous results regarding Default Logic. This section is mainly concerned with Constrained Default Logic but we also discuss the applicability of our results to Łukaszewicz' Justified Default Logic. In Section 4, we exemplify how the aforementioned formalization leads to a concrete proof procedure for credulous reasoning. We use an approach designed for solving planning problems by appeal to logic programs augmented by an equational theory [28]. This method turns out to capture a certain class of default theories in a direct manner. In Section 5, we turn to the problem of skeptical reasoning and develop a procedure that does neither strictly require the inspection of all extensions nor the computation of entire extensions. Again, this section is mainly concerned with Constrained Default Logic but we also discuss the applicability of our results to Reiter's classical definition (in the case of normal default theories). Finally, our results are summarized in Section 6.

2 Default Logics

The following notions and notations are fundamental for all dialects of Default Logic. A *default* $\delta(\overline{x}) = \frac{\alpha(\overline{x}) : \beta(\overline{x})}{\omega(\overline{x})}$ consists of three sets of first-order formulas, where $\overline{x}$ denotes a sequence of free variables occurring in these formulas. A default $\delta(\overline{x})$ is interpreted as a representative of each instance $\delta(\overline{t})$ where $\overline{t}$ is a sequence of ground terms. As usual, we call $\alpha(\overline{t}) = \mathit{Prereq}(\delta(\overline{t}))$ the *prerequisite*, $\beta(\overline{t}) = \mathit{Justif}(\delta(\overline{t}))$ the *justification*, and $\omega(\overline{t}) = \mathit{Conseq}(\delta(\overline{t}))$ the *consequence* of $\delta(\overline{t})$. Furthermore, if D is a set of defaults then $\mathit{Prereq}(D) = \bigcup_{\delta \in D} \mathit{Prereq}(\delta)$, $\mathit{Justif}(D) = \bigcup_{\delta \in D} \mathit{Justif}(\delta)$, and $\mathit{Conseq}(D) = \bigcup_{\delta \in D} \mathit{Conseq}(\delta)$. A *default theory* $\Delta = (D, W)$ consists of a set of defaults D and a set of closed first-order formulas W, called *world knowledge* or *background knowledge*. A *closed* default theory does not contain defaults with free variables. For notational convenience, a closed default is simply written as $\delta = \frac{\alpha : \beta}{\omega}$. This paper mainly focuses on closed default theories. Handling defaults with free variables is discussed in e.g. [51, 37], and we will briefly raise this problem at the end of Section 3 and Section 4, respectively. If each member of D is of the form $\frac{\alpha : \omega}{\omega}$, i.e. if justification and consequence coincide, then Δ is called a *normal* default theory. A default theory (D, W) is said to be *inconsistent* iff W is inconsistent. If F is a set of first-order formulas then we denote by $\mathit{Th}(F)$ the *theory* determined by F, i.e. the set of all formulas which are classically entailed by F. Finally, the symbol $\top$ ($\bot$) denotes

a formula which is always true (false).

2.1 The classical approach

Drawing conclusions in default theories is based on the formation of *extensions*, each of them representing a possible, maximal set of beliefs. The different ways of constructing a set of extensions for a default theory characterize the three variants discussed in this section. Reiter's classical Default Logic uses the following definition which is based on a fixpoint construction [51]:

Definition 2.1 If (D, W) is a closed default theory and S a set of closed first-order formulas then let $\Gamma(S)$ be the smallest set of formulas such that the following conditions are satisfied:

1. $W \subseteq \Gamma(S)$,
2. $Th(\Gamma(S)) = \Gamma(S)$, and
3. for any $\frac{\alpha : \beta}{\omega} \in D$, if $\alpha \in \Gamma(S)$ and $S \cup \{\beta\} \not\models \bot$ then $\omega \in \Gamma(S)$.

A set of formulas E is an *extension* of (D, W) iff $\Gamma(E) = E$.

Based on the concept of extensions there are two different ways to determine what are the logical consequences of a default theory. Firstly, a formula is called *credulously* entailed iff it is contained in at least one extension. Secondly, a formula is called *skeptically* entailed iff it is contained in every extension. In what follows, we concentrate on credulous reasoning and turn our attention to skeptical reasoning in Section 5.

In Reiter's original paper [51], attention is also restricted to proving membership in one extension. There, it is pointed out that in view of the automation of proving in default theories, it is of great practical importance to avoid the generation of an entire extension when trying to prove the entailment of a particular formula. Rather, one prefers to prove in a *local* fashion which means to restrict ones attention to those defaults which are needed to obtain a derivation of the formula under consideration. The formal property reflecting this is called *semi-monotonicity* stating that the entailment relation which is defined for a default theory is monotonic regarding additional defaults. More formally, if E is an extension of a theory (D, W) and D' is obtained by adding elements to D then semi-monotonicity guarantees the existence of an extension E' of (D', W) such that $E' \supseteq E$. Hence, if we need defaults $D \subseteq D'$ to determine a proof of a formula given the entire theory (D', W) then semi-monotonicity allows us to conclude that the formula is indeed credulously entailed by the whole theory. Unfortunately, as pointed out in [51], semi-monotonicity does not hold in general. As an example, consider an arbitrary default theory $\Delta = (D, W)$ and a propositional constant a which does not occur elsewhere in Δ. Independently from whatever the extensions of Δ are, adding the default $\frac{:a}{\neg a}$ makes the whole theory collapse because $(D \cup \{\frac{:a}{\neg a}\}, W)$ admits no extensions at all. This illustrates that it is inevitably necessary to consider the entire set of defaults in general when reasoning credulously.

On the other hand, semi-monotonicity is guaranteed in case of normal default theories [51]. This observation enabled Reiter to develop a proof theory which admits local proofs according to the preceding discussion. Based on his work, several authors applied theorem proving methods to normal default theories during the last decade, e.g. [4, 20, 61, 62].

2.2 Łukaszewicz' Definition of Justified Extensions

Starting out from the problem of semi-monotonicity, Łukaszewicz proposed a redefinition of the original notion of extensions [39]. He observed that the lack of semi-monotonicity stems from

requiring the application of a default even if its consequence is inconsistent with its justification or the justification of other applied defaults. Łukaszewicz' addresses this problem via a slightly more complicated definition of extensions, where an additional set of formulas J is used to collect the justifications of applied defaults. This set can then be used to suppress the application of further defaults:

Definition 2.2 If (D, W) is a closed default theory and S, T are sets of closed first-order formulas then let $\Gamma(S, T)$ be the pair of smallest sets of formulas (S', T') such that the following conditions are satisfied:

1. $W \subseteq S'$,
2. $Th(S') = S'$, and
3. for any $\frac{\alpha : \beta}{\omega} \in D$, if $\alpha \in S'$ and for any $\gamma \in T \cup \{\beta\}$ we have that $S \cup \{\omega\} \cup \{\gamma\} \not\models \bot$, then $\omega \in S'$ and $\beta \in T'$.

A set of formulas E is a *justified extension* of (D, W) *wrt.* a set of formulas J iff $\Gamma(E, J) = (E, J)$.

As regards this definition, semi-monotonicity is guaranteed in case of arbitrary default theories [39]. For instance, the crucial default $\frac{: a}{\neg a}$ can never be applied because the contradiction between its consequence and its justification are detected in Łukaszewicz's variant (via condition 3 in the above definition). It is, however, noteworthy that the set of justifications J of applied defaults is neither required to be consistent with the entire extension nor to be consistent itself.

2.3 Constrained Default Logic

Constrained Default Logic [57, 58, 17] has recently been developed due to an unintuitive behavior of classical Default Logic, which is also not addressed in the modification described above. As first pointed out in [49], Reiter's approach is incapable of what is usually called *committing to assumptions*:

Example 2.1 Let Δ_1 denote the default theory $(\{\delta_1 = \frac{ho : \neg ra}{sw}, \delta_2 = \frac{ho : ra}{ba}\}, \{ho\})$.[3] According to Definition 2.1, Δ_1 has exactly one extension, viz. $Th(\{ho, sw, ba\})$. Analogously, according to Definition 2.2 this is the only justified extension of Δ_1 (wrt. $J = \{\neg ra, ra\}$).

Obtaining a single extension of Δ_1 appears to be counterintuitive since this extension can be said to be based on the two conflicting assumptions ra and $\neg ra$ (see e.g. [49, 8, 16, 17]). Rather one expects Δ_1 to admit two different extensions, one containing sw and the other one containing ba.

Constrained Default Logic was developed in view of guaranteeing both semi-monotonicity and commitment to assumptions. Informally, the idea is to extend the consistency requirement of Definition 2.2 such that the justifications of all applied defaults have to be consistent with the underlying extension. Example 2.1 — where $J = \{\neg ra, ra\}$ — illustrates that this is not required in Łukaszewicz' method. As for Definition 2.2, an additional set of formulas is used, but now both the justifications and consequences of applied defaults are collected. The formal definition is as follows:

[3] Read ho as "taking a holiday", ra as "it is raining", sw as "going for a swim", and ba as "joining a basketball match".

Definition 2.3 If (D, W) is a closed default theory and S, T are two sets of closed first-order formulas then let $\Gamma(T)$ be the pair of smallest sets of formulas (S', T') such that the following conditions are satisfied:

1. $W \subseteq S' \subseteq T'$,
2. $Th(S') = S'$, $Th(T') = T'$, and
3. for any $\frac{\alpha : \beta}{\omega} \in D$, if $\alpha \in S'$ and $T \cup \{\beta\} \cup \{\omega\} \not\models \bot$ then $\omega \in S'$ and $\beta \wedge \omega \in T'$.

A pair of sets of formulas (E, C) is a *constrained extension* of (D, W) iff $\Gamma(C) = (E, C)$.

The so-called *constraints* C form a superset of E by construction. C includes the justifications and consequences of all applied defaults and is required to be consistent — provided the underlying default theory is not inconsistent itself. As for Łukaszewicz' method, the property of being semi-monotonic is guaranteed in general wrt. constrained extensions [57, 17]. Moreover, our example concerning commitment of assumptions is treated satisfactorily:

Example 2.1 (continued) Our default theory Δ_1 admits two constrained extensions, namely $(Th(\{ho, sw\}), Th(\{ho, \neg ra \wedge sw\}))$ and $(Th(\{ho, ba\}), Th(\{ho, ra \wedge ba\}))$. The reason for not obtaining a single extension is that after having applied δ_1, say, $\neg ra$ is included in the corresponding set of constraints and, then, δ_2 cannot be applied due to $Th(\{ho, \neg ra \wedge sw\}) \cup \{ra\} \models \bot$.

On the analogy of a result concerning classical Default Logic, it is possible to give a more intuitive, pseudo-iterative[4] characterization of constrained extensions which is not based on a fixpoint construction [57, 17]:

Theorem 2.4 *If (D, W) is a closed default theory and E, C are two sets of closed first-order formulas then let $E_0 := W$, $C_0 := W$, and for each $i \geq 0$*

$$E_{i+1} := Th(E_i) \cup \{ \omega \mid \tfrac{\alpha : \beta}{\omega} \in D,\ \alpha \in E_i,\ C \cup \{\beta\} \cup \{\omega\} \not\models \bot \}$$
$$C_{i+1} := Th(C_i) \cup \{ \beta \wedge \omega \mid \tfrac{\alpha : \beta}{\omega} \in D,\ \alpha \in E_i,\ C \cup \{\beta\} \cup \{\omega\} \not\models \bot \}.$$

(E, C) is a constrained extension of (D, W) iff $(E, C) = (\bigcup_{i=0}^{\infty} E_i, \bigcup_{i=0}^{\infty} C_i)$.

Based on this observations, the authors of [55, 60] develop a third characterization of constrained extensions which enables one to describe the generation process in a truly iterative manner, i.e. without the necessity of using a fixpoint construction. Stating their result requires to define the notion of *groundedness*, which was first applied to defaults in [61]:

Definition 2.5 If (D, W) is a default theory then a set $D' \subseteq D$ is called *grounded in W* (or simply: *grounded*) iff there exists an enumeration $\langle \delta \rangle_{i \in I}$ of D' such that for each $i \in I$,

$$W \cup Conseq(\{\delta_1, \ldots, \delta_{i-1}\}) \models Prereq(\delta_i).$$

Based on this definition the following result has been proved in [55, 60]:

Theorem 2.6 *Let (D, W) be a default theory and E, C be two sets of formulas. (E, C) is a constrained extension of (D, W) iff there exists a maximal (wrt. set inclusion), grounded set D' such that $D' \subseteq D$, $W \cup Justif(D') \cup Conseq(D') \not\models \bot$ and the following holds:*

[4] Observe that the specification of a constrained extension in Theorem 2.4 is not truly iterative, since E_{i+1} and C_{i+1} make reference to the final extension C.

1. $E = Th(W \cup Conseq(D'))$

2. $C = Th(W \cup Justif(D') \cup Conseq(D'))$

Groundedness is for instance necessary to ensure that defaults do not mutually satisfy their prerequisites, e.g. that $(Th(\{a,b\}), Th(\{a,b\}))$ is not a constrained extension of $(\{\frac{b:a}{a}, \frac{a:b}{b}\}, \{\})$. In general, groundedness can be seen as the counterpart of the minimality requirement in the fixpoint definitions 2.1, 2.2, and 2.3.

Theorem 2.6 is fundamental for our results as regards both credulous as well as skeptical reasoning. To this end, we introduce some further useful notions regarding a proof theory in Constrained Default Logic.

Definition 2.7 Let $\Delta = (D,W)$ be a default theory. We call a subset $D' \subseteq D$ a *base of extensions* (or simply: *base*) of Δ iff D' is grounded and $W \cup Justif(D') \cup Conseq(D') \not\models \bot$. A *default proof* of a closed formula g from Δ is a base D' such that $W \cup Conseq(D') \models g$. We say that g is *provable by default* (or just *provable*) from Δ if we can find such a default proof.[5]

A trivial base is the empty set, which can be regarded as a default proof of any formula implied by the background knowledge. Each maximal base determines a single constrained extension according to Theorem 2.6. Hence, a set of defaults is a base if it can be extended to one or more constrained extensions in the spirit of Theorem 2.6.

Example 2.2 Recall our default theory Δ_1 (cf. Example 2.1) augmented by a third default, viz. $\Delta_2 = (\{\delta_1 = \frac{ho:\neg ra}{sw}, \delta_2 = \frac{ho:ra}{ba}, \delta_3 = \frac{sw:fun}{fun}\}, \{ho\})$. According to the previous definition, we find the four bases $\{\}$, $\{\delta_1\}$, $\{\delta_2\}$, and $\{\delta_1, \delta_3\}$, respectively. The last two are maximal and determine the two different constrained extension $(Th(\{ho, ba\}), Th(\{ho, ra \wedge ba\}))$ and $(Th(\{ho, sw, fun\}), Th(\{ho, \neg ra \wedge sw, fun\}))$ of Δ_2. Both $\{\delta_1\}$ and $\{\delta_1, \delta_3\}$ are default proofs of sw, and each of the four sets is a default proof of ho since $ho \in W$.

Following Theorem 2.6, it is easy to see that our notion of provability coincides with the definition of credulous reasoning:

Proposition 2.8 *Let $\Delta = (D,W)$ be a closed default theory and g be a closed formula. There exists a constrained extension (E,C) of Δ such that $g \in E$ iff g is provable from Δ.*

Proof:
"$\Rightarrow$": If (E,C) is a constrained extension then we can find a base D' such that $E = Th(W \cup Conseq(D'))$ according to Theorem 2.6. Due to $g \in E$, D' is a default proof of g.
"$\Leftarrow$": Assume D' to be some default proof of g from Δ, i.e. $W \cup Conseq(D') \models g$. Then we can find by semi-monotonicity at least one maximal (wrt. set inclusion) base D'' of Δ such that $D'' \supseteq D'$. According to Theorem 2.6 the pair (E,C) where

1. $E = Th(W \cup Conseq(D''))$

2. $C = Th(W \cup Justif(D'') \cup Conseq(D''))$

is a constrained extension of Δ. From $D' \subseteq D''$ and $W \cup Conseq(D') \models g$ we conclude by monotonicity that $W \cup Conseq(D'') \models g$, hence $g \in E$. ∎

Finally, on the analogy of a result already implicitly used in [51], we make use of the compactness property of first-order logic [19], which implies a certain finiteness criterion of default proofs. This is essential for our results concerning skeptical reasoning in Section 5.

[5] Note that an inconsistent default theory has no bases. To be consistent with the usual definition of provability, we therefore extend this definition and say that $\{\}$ is a default proof of any g from an inconsistent Δ.

Remark 2.9 If g is provable from a default theory (D, W) then we can find a finite base of (D, W) proving g.

Theorem 2.6 illustrates that extensions can be generated by successively applying defaults. Moreover, Definition 2.7 in conjunction with Proposition 2.8 ensure that default proofs can be found in the very same fashion by successively generating and extending bases. In the following section, we identify the application of a default with the execution of an *action* within a current set of beliefs (along with some constraints), referred to as a *situation*. Hence, searching for bases in order to provide a default proof can be formulated as a *planning problem*. The latter is solved as soon as the formula we want to prove is a logical consequence of the situation at hand.

3 Planning

Understanding and modeling the ability of humans to reason about dynamically changing worlds is one of the key issues in Artificial Intelligence and Cognitive Science (see e.g. [1]). The fundamental concepts to describe the behavior of dynamic systems are situations and actions. A *situation* is a snapshot of the world at a particular instant. *Actions* serve as descriptions of the dynamical aspect by defining how situations can be transformed. An important application of this kind of information processing is the field of *planning*, which describes the problem of searching for a sequence of actions such that its successive application transforms a given situation into a situation satisfying a certain criterion, given by the *goal* specification.

In case of *deductive planning*, situations are represented by means of logical formulas. The formal concept is as follows:

Definition 3.1 A *deductive planning problem* is a quadruple $(\Sigma, \Omega, \mathcal{I}, g)$ where Σ, the space of *situations*, is a set of sets of formulas, Ω is a set of partially defined functions of type $\Sigma \mapsto \Sigma$ (called *operations* or *actions*), $\mathcal{I} \in \Sigma$ (called *initial situation*), and g is an arbitrary formula (the *goal specification*).

An action $a \in \Omega$ is called *applicable* in a situation $\mathcal{S} \in \Sigma$ iff $a(\mathcal{S})$ is defined. In this case the *application* of a to $\mathcal{S}$ yields the new situation $a(\mathcal{S})$. A *solution* to a planning problem consists of a finite sequence $[a_1, \ldots, a_n]$ of actions, called a *plan*, such that there exist situations $\mathcal{S}_0, \ldots, \mathcal{S}_n \in \Sigma$ where $\mathcal{S}_0 \equiv \mathcal{I}$, $\mathcal{S}_n \models g$, and for each $i = 1 \ldots n$, a_i is applicable in $\mathcal{S}_{i-1}$ and its application yields $\mathcal{S}_i$.

A planning problem is called *solvable* if it has a solution.

In what follows, we use the concept of deductive planning problems to develop a proof theory for credulous reasoning in Constrained Default Logic. To this end, a particular set of beliefs along with a set of constraints is interpreted as a situation in the sense of Definition 3.1. To be more precise, we consider expressions of the form

$$\mathcal{S} \cup \{ \mathfrak{co}(\Phi) \} \tag{1}$$

as situations, where $\mathcal{S}$ is a set of closed first-order formulas (representing the set of beliefs $Th(\mathcal{S})$), $\mathfrak{co}$ is a special unique predicate, and Φ is a closed first-order formula (representing the constraints $Th(\Phi)$). Note that even if $\mathcal{S}$ is written as a conjunction of its elements, (1) is not a first-order formula. We will discuss the consequences of this fact later in this section.

Based on this representation, each default of a given default theory is interpreted as an action which, when executed in a situation, changes the current sets of beliefs and constraints, respectively, in the spirit of Theorem 2.6. We adopt the notion of applicability which is implicitly

determined by Theorem 2.6 to define applicability of actions in situations of the form (1). Altogether, the problem of finding a default proof within the context of Constrained Default Logic is interpreted in terms of deductive planning problems as follows:

Definition 3.2 If $\Delta = (D, W)$ is a closed default theory and g a formula then the corresponding planning problem $\mathcal{P}_{\Delta,g} = (\Sigma, \Omega, \mathcal{I}, g)$ is as follows:

1. The set of possible situations Σ contains each expression of the form (1) where $\mathcal{S}$ is any closed set of first-order formulas and Φ is a closed first-order formula.
2. Ω contains exactly one element δ' for each $\delta = \frac{\alpha:\beta}{\omega} \in D$. Such an action $\delta' \in \Omega$ is defined for a situation $\mathcal{S} \in \Sigma$ iff
   (a) $\mathcal{S} \models \alpha$ and
   (b) $\Phi \cup \{\beta\} \cup \{\omega\} \not\models \bot$ where $\mathfrak{co}(\Phi) \in \mathcal{S}$.

   If δ' is defined then $\delta'(\mathcal{S}) := (\mathcal{S} \setminus \{\mathfrak{co}(\Phi)\}) \cup \{\omega, \mathfrak{co}(\Phi \wedge \beta \wedge \omega)\}$.
3. $\mathcal{I} = W \cup \{\mathfrak{co}(W)\}$.

Note that in general the application of an action is nonmonotonic because a fact $\mathfrak{co}(\Phi)$ which holds in some situation $\mathcal{S}$ usually becomes false by replacing it by some $\mathfrak{co}(\Phi')$. Hence, as the applicability of an action depends on a fact of this form (via condition 2(b) above), the property of an action to be applicable in a certain situation might be lost in later situations. This is an important general characteristic of dynamical systems.

For the sake of simplicity, we will not distinguish between the name δ of a default and the corresponding element $\delta' \in \Omega$ in what follows. Before formally proving the adequateness of our formalization, let us illustrate it by the running example.

Example 2.2 (continued) Recall default theory Δ_2. If g is the atom *fun* then a solution to the corresponding planning problem $\mathcal{P}_{\Delta_2,g}$ is depicted in Figure 1. The resulting plan $[\delta_1, \delta_3]$ corresponds to the base $\{\delta_1, \delta_3\}$ which is a default proof of *fun* from Δ_2. Note that the action representing $\delta_2 = \frac{ho:ra}{ba}$ is not applicable in the resulting situation due to $\{ho \wedge \neg ra \wedge sw \wedge fun\} \cup \{ra\} \cup \{ba\} \models \bot$.

This example illustrates a one-to-one correspondence between the formation of bases and the generation of plans. As this observation holds in general, we are able to prove that credulous reasoning can be adequately modeled by solving deductive planning problems:

Theorem 3.3 *Let Δ be a default theory and g be a closed formula then g is provable from Δ iff $\mathcal{P}_{\Delta,g}$ is solvable.*

Proof: If Δ is inconsistent then W is inconsistent and there is exactly one constrained extension, viz. $(Th(\bot), Th(\bot))$ (see [57]), i.e. each g is provable from Δ (recall Footnote 5). Similarly, since W is part of the initial situation of $\mathcal{P}_{\Delta,g}$, $[\,]$ is a solution to g due to $g \in Th(W) = Th(\bot)$.

Now, assume $\Delta = (D, W)$ to be consistent.

"$\Rightarrow$": If g is provable from Δ then we can find a finite base $D' = \langle \delta_1, \ldots, \delta_n \rangle \subseteq D$ proving g according to Remark 2.9. By induction on n we show that there is a solution p to $\mathcal{P}_{\Delta,g} = (\Sigma, \Omega, \mathcal{I}, g)$ which satisfies the following conditions:

1. p contains exactly the elements of D' and

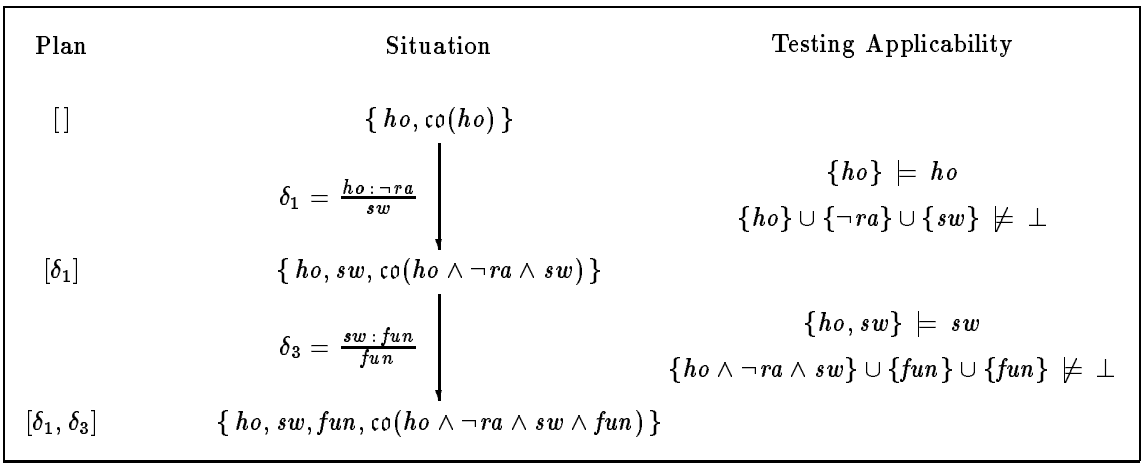

Figure 1: A solution to the planning problem $\mathcal{P}_{\Delta_2,fun}$ generating the plan $[\delta_1, \delta_3]$.

2. p transforms the initial situation $\mathcal{I}$ into a situation $\mathcal{S} \cup \{\mathfrak{co}(\Phi)\}$ such that $\mathcal{S} \equiv W \cup Conseq(D')$ and $\Phi \equiv W \cup Justif(D') \cup Conseq(D')$.

In case $n = 0$ we have $D' = \{\}$ and $W \models g$. Since $\mathcal{I} = W \cup \{\mathfrak{co}(W)\}$ due to Definition 3.2 we conclude that $\mathcal{I} \models g$ and, hence, the empty sequence $[\,]$ of actions solves $\mathcal{P}_{\Delta,g}$.

If $n > 0$ then let $D'_{n-1} = \langle \delta_1, \ldots, \delta_{n-1} \rangle$ and $\delta_n = \frac{\alpha_n : \beta_n}{\omega_n}$. Clearly, D'_{n-1} is grounded since $D' = \langle \delta_1, \ldots, \delta_n \rangle$ is grounded. Furthermore, D'_{n-1} is a default proof of α_n due to Definition 2.5. Thus the induction hypothesis implies that there is a solution p_{n-1} to the planning problem $(\Sigma, \Omega, \mathcal{I}, \alpha_n)$ such that p_{n-1} contains exactly the elements of D'_{n-1}, and applying p_{n-1} to $\mathcal{I}$ yields a situation $\mathcal{S}_{n-1} \cup \{\mathfrak{co}(\Phi_{n-1})\}$ such that $\mathcal{S}_{n-1} \models \alpha_n$ and

$$\mathcal{S}_{n-1} \equiv W \cup Conseq(D'_{n-1}) \tag{2}$$

$$\Phi_{n-1} \equiv W \cup Justif(D'_{n-1}) \cup Conseq(D'_{n-1}). \tag{3}$$

Following Definition 3.1 and Definition 3.2, δ_n is applicable in $\mathcal{S}_{n-1} \cup \{\mathfrak{co}(\Phi_{n-1})\}$ due to $\mathcal{S}_{n-1} \models \alpha_n$, $W \cup Justif(D') \cup Conseq(D') \not\models \bot$, and $\Phi_{n-1} \cup \{\beta_n\} \cup \{\omega_n\} \equiv W \cup Justif(D') \cup Conseq(D')$. Applying δ_n to $\mathcal{S}_{n-1} \cup \{\mathfrak{co}(\Phi_{n-1})\}$ yields $\mathcal{S}_{n-1} \cup \{\alpha_n\} \cup \{\mathfrak{co}(\Phi_{n-1} \wedge \beta_n \wedge \omega_n)\}$. From (2) it follows that $\mathcal{S}_{n-1} \cup \{\omega_n\} \equiv W \cup Conseq(D')$, hence applying p_{n-1} followed by δ_n to $\mathcal{I}$ solves the original planning problem $\mathcal{P}_{\Delta,g}$. Furthermore, from (3) we conclude $\Phi_{n-1} \wedge \beta_n \wedge \omega_n \equiv W \cup Justif(D') \cup Conseq(D')$.

"$\Leftarrow$": Let $p = [\delta_1, \ldots, \delta_n]$ be a solution to the planning problem $\mathcal{P}_{\Delta,g}$. By induction on n we show that $D' = \langle \delta_1, \ldots, \delta_n \rangle \subseteq D$ is a default proof of g. Furthermore, if $\mathcal{S}_n \cup \{\mathfrak{co}(\Phi)\}$ is the result of applying p to $\mathcal{I}$ then $\mathcal{S}_n \equiv W \cup Conseq(D')$ and $\Phi \equiv W \cup Justif(D') \cup Conseq(D')$.

As the base case $n = 0$ can be proved as above, we directly turn to the induction step and assume $n > 0$. Let $p_{n-1} = [\delta_1, \ldots, \delta_{n-1}]$, $\delta_n = \frac{\alpha_n : \beta_n}{\omega_n}$, and let $\mathcal{S}_{n-1} \cup \{\mathfrak{co}(\Phi_{n-1})\}$ be the result of applying p_{n-1} to $\mathcal{I}$. The induction hypothesis implies that $D'_{n-1} = \{\delta_1, \ldots, \delta_{n-1}\}$ is a base such that again (2) and (3) hold.

From Definition 3.1, Definition 3.2, and the fact that δ_n is applicable in $\mathcal{S}_{n-1} \cup \{\mathfrak{co}(\Phi_{n-1})\}$ we conclude that $\mathcal{S}_{n-1} \models \alpha_n$ and $\Phi_{n-1} \cup \{\beta_n\} \cup \{\omega_n\} \not\models \bot$. Hence, D' is a base. Furthermore, from (2) it follows that $\mathcal{S}_{n-1} \cup \{\omega_n\} \equiv W \cup Conseq(D')$ and, hence, D' is a default proof of g due to $\mathcal{S}_{n-1} \cup \{\omega_n\} \models g$. Finally, from (3) we can conclude that $\Phi_{n-1} \wedge \beta_n \wedge \omega_n \equiv W \cup Justif(D') \cup Conseq(D')$. ■

The above result illustrates that methods to solve deductive planning problems in the sense of Definition 3.1 can be applied to address the task of automatically performing credulous reasoning in Constrained Default Logic. The question naturally arises whether and how approaches known from the field of planning are applicable to our problem class, which is given by Definition 3.2. Today there exists a variety of deductive planning methods. Usually, each framework is characterized by the particular way it solves the so-called *frame problem* [43].[6] Many approaches are based on the *situation calculus* [43, 45] where each atomic expression contains an additional argument to characterize a particular situation to which this atom refers. The frame problem within the situation calculus was tackled by a collection of frame axioms in [25] or [34], by a nonmonotonic rule in [44], or by so-called successor state axioms in [52]. On the other hand, there are methods which are not based on the situation calculus and which do not require special axioms to solve the frame problem, e.g. STRIPS [21, 36], the Linear Connection Method [5], an approach [42] based on Linear Logic [23], and a method [28] based on logic programming with equational theories [31, 22, 27].

A principal difficulty stems from the fact that in our application situations are characterized by formulas of the form (1), i.e. particularly containing a subformula $\mathfrak{co}(\Phi)$ where Φ itself can be an arbitrary first-order formula. This restricts the number of approaches which are directly applicable, and methods which use the concept of *reification* appear to be the most suitable ones. This is so because in reified approaches predicates describing a situation are treated as terms and they are therefore more flexibly manipulable than in first-order logic. The equational logic programming based approach [28] mentioned above is such a method. To illustrate how the technique developed above can be used to obtain concrete proof systems, we will investigate the applicability of the aforementioned method to deductive planning problems defined in Definition 3.2 in the following section.

An interesting suggestion to weaken Definition 3.1 where a plan is a totally ordered set of actions, is provided by planning methods which create *partial* plans such as in [56, 35, 14]. If two or more actions can be executed in either order then it is not necessary to require the members which constitute a plan to be totally ordered. This resembles the proof theory developed by Reiter where a default proof actually consists in several sets of defaults which are ordered among themselves whereas each such set is unordered.

Aside from developing a large number of planning formalisms and systems, recent work focused on complexity analyses to fix restricted problem classes that are especially tractable. Correspondingly, some tractable subclasses of default theories have been fixed in e.g. [33]. The link provided by our formalization enables us to enrich this collection by adopting results known from planning. Although a detailed discussion of how to apply such results to Default Logic is beyond the scope of this paper, we want to illustrate this point by two examples taken from [3] and [12], respectively. Both examples can be directly adapted to form two classes of default theories for which credulous entailment is decidable in polynomial time:

Theorem 3.4 *Let $\Delta = (D, W)$ be a propositional, normal default theory such that*

1. *W is a conjunction of literals,*
2. *for each $\delta \in D$, $Prereq(\delta)$ is a conjunction of literals and $Justif(\delta) = Conseq(\delta)$ is a single literal, and*
3. *there are no $\delta_1, \delta_2 \in D$ such that $Prereq(\delta_1) \cup Prereq(\delta_2) \models \bot$.*

[6] The (technical) frame problem addresses the task to formalize the natural assumption that facts which hold in a situation and which are not affected by the action to be applied, continue to hold in the resulting situation.

If g is a conjunction of literals then determining the existence of a default proof of g from Δ is polynomial.

Proof: The result follows from Theorem 6.1 in [3] and Theorem 3.3. The former theorem implies that determining the existence of a solution to the corresponding planning problem $\mathcal{P}_{\Delta,g}$ is polynomial. ■

Theorem 3.5 *Let $\Delta = (D, W)$ be a propositional, normal default theory such that*

1. *W is a conjunction of literals and*
2. *for each $\delta \in D$, $Prereq(\delta)$ is a single literal and $Justif(\delta) = Conseq(\delta)$ is a conjunction of literals.*

If g is a literal then determining the existence of a default proof of g from Δ is polynomial.

Proof: The result follows from Theorem 3.8 in [12] and Theorem 3.3. The former theorem implies that determining the existence of a solution to the corresponding planning problem $\mathcal{P}_{\Delta,g}$ is polynomial. ■

Notably, the two previous results hold for *all* Default Logics mentioned in this paper. This is so because Constrained Default Logic coincides with Reiter's Default Logic (see Theorem 5.8) as well as the variants of Łukaszewicz [39] and Brewka [8] on the class of normal default theories (see [17] for details).

Likewise, many other complexity results found in the context of planning, e.g. [14, 11, 2, 3] can be applied to obtain classes of default theories for which credulous reasoning is simpler than in the general case (which is known to be Σ_2^p-complete [24, 66, 13]).

Finally, let us briefly illustrate how open defaults can be treated in our representation. Let $\delta(\overline{x}) = \frac{\alpha(\overline{x}) : \beta(\overline{x})}{\omega(\overline{x})}$ be such a default with free variables $\overline{x}$. The usual interpretation in default logics is to take these defaults as representatives for all their infinitely many ground instantiations. This resembles the usual distinction that is made between an operator and an action in the context of deductive planning: $\delta(\overline{x})$ is handled as a single *operator* while each instance $\delta(\overline{t})$ is called an *action*, where $\overline{t}$ is a sequence of ground terms. Based on this refinement, Definition 3.2 can be straightforwardly modified if the notions of applicability and application are defined wrt. instances of members of Ω. Consider, for example, the default theory

$$\left(\{\delta(x) = \frac{bird(x) : flies(x) \wedge \neg peng(x)}{flies(x)}\}\ ,\ \{bird(a), bird(b), peng(b)\} \right)$$

then the action $\delta(a)$ is an instance of the operator $\delta(x)$. This action is applicable in the corresponding initial situation, and its application yields

$$\{bird(a), bird(b), peng(b), flies(a)\} \ \cup\ \mathrm{co}\ (\ \{bird(a), bird(b), peng(b), \neg peng(a), flies(a)\}\ )$$

whereas $\delta(b)$ is not applicable due to $\neg peng(b)$ contradicting the background knowledge.

Łukaszewicz' Variant

Our claim is that the application of deductive planning to reasoning in Default Logics is not restricted to the particular variant we have discussed so far but can be useful in general derivatives, provided they are semi-monotonic. This shall be illustrated by showing how the main definition above can be modified to serve as an adequate proof mechanism for Łukaszewicz' variant of

classical Default Logic [39]. The major difference between his definition of justified extensions and constrained extensions is that the former employs a weaker consistency requirement. Fortunately, as has been shown by Risch [54], it is possible to characterize justified extensions in the sense of Definition 2.2 in a truly iterative fashion in analogy to Theorem 2.6:

Theorem 3.6 *Let (D, W) be a default theory and E, J be two sets of formulas. E wrt. J is a justified extension of (D, W) iff there exists a maximal (wrt. set inclusion), grounded set D' such that $D' \subseteq D$, $E \cup \{\beta\} \not\models \bot$ for each $\beta \in Justif(D')$, and the following holds:*

1. $E = Th(W \cup Conseq(D'))$
2. $J = Justif(D')$

Similar to the approach developed in Section 3, this theorem can straightforwardly be used to formalize the search for a credulous proof in Łukaszewicz' Default Logic as a deductive planning problem. To this end, let a situation description be of the form

$$\mathcal{S} \cup \{ju(\Phi)\} \tag{4}$$

where $\mathcal{S}$ is a set of closed first-order formulas and Φ is a first-order formula while ju is a special predicate symbol similar to co, i.e. here Φ is intended to contain the current set of justifications. On the analogy of Definition 3.2 we then define

Definition 3.7 If $\Delta = (D, W)$ is a default theory and g a formula then the corresponding planning problem $\mathcal{P}_{\Delta,g} = (\Sigma, \Omega, \mathcal{I}, g)$ wrt. Definition 2.2 is as follows:

1. The set of possible situations Σ contains each expression of the form (4) where both $\mathcal{S}$ and Φ are closed sets of first-order formulas.
2. Ω contains exactly one element δ' for each $\delta = \frac{\alpha : \beta}{\omega} \in D$. Such an action $\delta' \in \Omega$ is defined for a situation $\mathcal{S} \in \Sigma$ iff
   (a) $\mathcal{S} \models \alpha$ and
   (b) $\forall \gamma \in \Phi \cup \{\beta\}.\ \mathcal{S} \cup \{\omega\} \cup \{\gamma\} \not\models \bot$ where $ju(\Phi) \in \mathcal{S}$.

   If δ' is defined then $\delta'(\mathcal{S}) := (\mathcal{S} \setminus \{ju(\Phi)\}) \cup \{\omega, ju(\Phi \cup \{\beta\})\}$.
3. $\mathcal{I} = W \cup \{ju(\{\})\}$.

As in Section 3 it is possible to prove the adequateness of this definition as regards credulous entailment in Łukaszewicz' variant of Default Logic:

Theorem 3.8 *Let Δ be a closed default theory and g be a closed formula then g is provable from Δ iff $\mathcal{P}_\Delta$ (wrt. Definition 3.7) has a solution.*

4 The Equational Logic Programming Approach

To illustrate the applicability of the ideas developed in the previous section, we will use an approach based on logic programming with an underlying equational theory (ELP for short) to implement credulous reasoning for a subclass of default theories which we call *conjunctive default theories*: If each of the three components of $\delta(\overline{x}) = \frac{\alpha(\overline{x}) : \beta(\overline{x})}{\omega(\overline{x})}$ is a conjunction of literals then δ is called a *conjunctive* default, and $\Delta = (D, W)$ is a conjunctive default theory iff each

default in D is conjunctive and W is a set of conjunctions of literals as well.[7] Our class is similar to the most expressive class of default logics considered by Kautz and Selman [33] who call them *disjunction-free* theories, but who additionally require *semi-normal* defaults, i.e. each default has to be of the form $\frac{\alpha(\overline{x}):\beta(\overline{x})\wedge\omega(\overline{x})}{\omega(\overline{x})}$.

Let us first of all briefly summarize the notions and notations regarding logic programs with underlying equational theories as used, for instance, in [22, 27, 63]. A *normal logic program* [38] consists of a finite set of *clauses* $A \leftarrow L_1, \ldots, L_m$ ($m \geq 0$), where the *head* A is an atom and the elements of the *body* $L_1, \ldots, L_m$ are literals. A *normal goal* is a clause of the form $\leftarrow L_1, \ldots, L_m$ ($m \geq 0$) where $L_1, \ldots, L_m$ are again literals. If $m = 0$ then the so-called *empty goal* is denoted by $\Box$. In what follows, we adopt the usual practice and denote variables by uppercase letters such as $X, Y, \ldots$, and predicate and function symbols are written using lower case letters.

In a *normal equational logic program* (P, E), the program P itself is augmented by a special equational theory E which defines an equality relation on the terms.[8] Formally, such a theory E consists of a set of expressions $s = t$ which are implicitly assumed to be universally closed. Two terms s and t are called *E-equivalent* — written $s =_E t$ — if they are equal wrt. E. For instance, if $E_C = \{X \circ Y =_{E_C} Y \circ X\}$ describes the law of commutativity for the binary function $\circ$ then $a \circ X =_E X \circ a$. A substitution θ is called an *E-unifier* of two terms s and t iff $s\theta =_E t\theta$. If such a substitution exists then s and t are called *E-unifiable*. For instance, $a \circ X$ and $b \circ Y$ are not unifiable in general, but they are E_C-unifiable using the E_C-unifier $\{X \mapsto b, Y \mapsto a\}$. The notion of E-unifiers is extended to atoms in the obvious way.

An adequate computation procedure for normal equational programs is *SLDENF-Resolution*[9] [63, 69]: This resolution principle is based on the integration of the equational theory into the unification procedure [31, 22, 27]. Furthermore, negative subgoals $\neg A$ are treated by the *negation-as-failure* concept [15], i.e. by trying to prove that each derivation of the affirmative part A fails. More formally, let (P, E) be a normal equational logic program and $G = \leftarrow L_1, \ldots, L_k, \ldots, L_m$ a normal goal. An *SLDENF-derivation of* $(P, E) \cup \{G\}$ consists of a sequence of single SLDENF-steps which are successively applied to G wrt. (P, E). An *SLDENF-step* applied to G wrt. (P, E) consists of selecting a literal L_k of G ($1 \leq k \leq m$). If L_k is positive then let $A \leftarrow B_1, \ldots, B_l$ be a new variant of a program clause in P such that its head A and the selected literal L_k are E-unifiable with E-unifier θ. Then, the result of this SLDENF-step is the goal $\leftarrow (L_1, \ldots, L_{k-1}, B_1, \ldots, B_l, L_{k+1}, \ldots, L_m)\theta$. If, on the other hand, L_k is a negative literal $\neg A$ then we try to determine whether each SLDENF-derivation of $\leftarrow A$ fails: An SLDENF-derivation is *successful* if it ends up with the empty clause $\Box$, and it fails if the final goal is non-empty and no further SLDENF-derivation step is possible. If each SLDENF-derivation of $\leftarrow A$ fails then the negative literal $L_k \equiv \neg A$ has been solved and is removed from G. Otherwise, the original derivation of $(P, E) \cup \{G\}$ fails at this point. It is assumed that negative literals are only selected if they are ground [15]. A successful derivation is also called a *refutation*. The combination of all E-unifiers used during a refutation, restricted to the variables in the original goal, is called *computed answer substitution*. All these concepts will be illustrated by examples later in this section.

The most significant feature of the ELP based approach to planning is that a situation is completely reified by representing the various facts which hold in this situation as terms. These terms are connected via a binary function symbol denoted by $\circ$ [28, 26, 29, 30, 70]. This

[7] A conjunction of literals is also called a *1CNF-formula*, e.g. in [33, 13].

[8] As equality relations are intended to be defined by the additional theory E, no head of a program clause in P shall contain the equality predicate $=_E$.

[9] I.e. linear resolution with selection function on definite clauses with equality and negation-as-failure

representation technique can be applied to (sets of) conjunctions of literals: An atom $p(\bar{t})$ is represented by the identical term $p(\bar{t})$ but p is treated as a function symbol. A negative literal $\neg p(\bar{t})$ is represented by additionally employing a unary function which denotes the negation of its argument, illustratively written as $[p(\bar{t})]^-$. For instance, the formula $ho \wedge fun \wedge \neg ra$ can be represented by the term

$$(\, ho \circ fun \,) \circ ra^- \tag{5}$$

where the connective $\circ$ is written in infix notation. As a set of formulas can be adequately interpreted as a conjunction of its elements, a set of conjunctions of literals is treated in the very same way, i.e. (5) shall as well be a representation of $\{ho, fun \wedge \neg ra\}$ etc. The formal definition of how to represent conjunctions and sets of conjunctions is as follows:

Definition 4.1 The *representation* τ of a conjunction (resp. a set of conjunctions) of literals[10] is inductively defined by

1. $\tau_a = a$,
2. $\tau_{\neg a} = a^-$,
3. $\tau_{l_1 \wedge \ldots \wedge l_n} = \tau_{l_1} \circ \cdots \circ \tau_{l_n}$,
4. $\tau_{\{f_1,\ldots,f_n\}} = \tau_{f_1} \circ \cdots \circ \tau_{f_n}$, and
5. $\tau_\top = \tau_{\{\}} = \emptyset$

where a is an atom (treated as a term on the right hand side), $l_1, \ldots, l_n$ are literals, $f_1, \ldots, f_n$ are conjunctions of literals ($n \geq 1$), and $\emptyset$ is a special constant.

To ensure the adequateness of Definition 4.1, we have to introduce some properties of our connection function $\circ$. Obviously, the occurrence of parentheses and the order of the subterms should be irrelevant in so far as e.g. $fun \circ (ra^- \circ ho)$ should denote the very same situation as (5). We therefore employ an *equational theory* which captures the intention of our special function symbol. More precisely, $\circ$ is required to be associative, commutative, and to admit the constant $\emptyset$ (introduced in the previous definition) as its unit element defining the empty situation or, equivalently, the formula $\top$. Formally, the equational theory (AC1) defined by the three axioms

$$\begin{array}{rcl} X \circ (\, Y \circ Z \,) & = & (\, X \circ Y \,) \circ Z \\ X \circ Y & = & Y \circ X \\ X \circ \emptyset & = & X \end{array} \tag{AC1}$$

is fundamental for our approach, i.e. whenever terms are compared or have to be unified then comparison and unification is performed modulo (AC1). Due to the first axiom of associativity we are allowed to omit all parentheses at the level of $\circ$. Based on this equational theory, we can make the following observation:

Remark 4.2 Let Φ and Ψ be two sets of conjunctions of literals then

1. $\Phi \equiv \Psi$ iff $\tau_\Phi =_{\mathrm{AC1}} \tau_\Psi$,[11]
2. Φ is inconsistent iff $\tau_\Phi =_{\mathrm{AC1}} t \circ t^- \circ s$ for some terms s, t, and

[10] For the sake of simplicity, we assume here and in the sequel that no conjunction or set of conjunctions is redundant in so far as it contains a single literal more than once.

[11] Of course this is only true in case neither Φ nor Ψ contain some literal twice or more (recall Footnote 10) because $a \neq_{\mathrm{AC1}} a \circ a$, say. The reason for not requiring $\circ$ to be idempotent is explained below.

3. if Φ is consistent and f is a literal then $\Phi \models f$ iff $\tau_\Phi =_{\mathrm{AC1}} \tau_f \circ s$ for some term s,

where $\equiv$ denotes logical equivalence and $=_{\mathrm{AC1}}$ denotes equality wrt. the theory (AC1).

As our formalism is based on reification, it admits an elegant way to integrate the second order component, which occurs in situations of the form (1). If Φ is a conjunction of literals then the special formula $\mathfrak{co}(\Phi)$ can be represented by the term

$$\mathfrak{co}\,(\,\tau_\Phi\,)$$

where we treat $\mathfrak{co}$ as a unary function symbol and τ_Φ denotes the term representation of Φ as defined above. For instance, the resulting situation after having applied the first default δ_1 depicted in Figure 1 can be encoded via

$$ho \circ sw \circ \mathfrak{co}\,(\,ho \circ ra^- \circ sw\,). \tag{6}$$

Having discussed the formalization of situations, we now concentrate on the representation of action descriptions, which are used to transform situations. In the ELP based approach, an action description is determined by its name a and two terms $c_1 \circ \cdots \circ c_m$, called *conditions*, and $e_1 \circ \cdots \circ e_n$, called *effects*. Such an action description is introduced in the program via a unit clause based on the ternary predicate *action*, viz. by

$$action\,(\,c_1 \circ \cdots \circ c_m\,,\,a\,,\,e_1 \circ \cdots \circ e_n\,). \tag{7}$$

For instance, the fact

$$action\,(\,sw \circ \mathfrak{co}(X)\,,\,\delta_1\,,\,sw \circ fun \circ \mathfrak{co}(X \circ fun)\,) \tag{8}$$

will be used later to encode the action corresponding to the default $\delta_3 = \frac{sw\,:\,fun}{fun}$ in Example 2.2.

An action description is said to be *applicable* in a situation represented by the term s iff its conditions can be satisfied in s and the resulting situation is consistent. More precisely, conditions $c_1 \circ \cdots \circ c_m$ can be satisfied in s iff we can find a substitution θ for the variables occurring in the conditions such that each $c_i\theta$ is contained in s. In other words, we have to find a solution θ to the AC1-unification problem

$$(c_1 \circ \cdots \circ c_m \circ V)\theta \stackrel{?}{=}_{\mathrm{AC1}} s\theta \tag{9}$$

where V is a variable not occurring elsewhere. For instance, the conditions of (8) can be satisfied in (6) because the corresponding unification problem

$$(\,sw \circ \mathfrak{co}(X) \circ V\,)\,\theta \stackrel{?}{=}_{\mathrm{AC1}} (\,ho \circ sw \circ \mathfrak{co}(ho \circ ra^- \circ sw)\,)\,\theta \tag{10}$$

is solvable using the AC1-unifier $\theta = \{X \mapsto ho \circ ra^- \circ sw,\, V \mapsto ho\}$. Now, if an instance of an action description (7) is applied to a situation s then the resulting situation is computed via removing the conditions $(c_1 \circ \cdots \circ c_m)\theta$ from s and adding the effects $(e_1 \circ \cdots \circ e_n)\theta$ afterwards.[12] To formally perform these two operations, observe that a side effect of solving the unification problem (9) is that the variable V becomes bound to exactly those subterms which are in s

[12] Thus, planning in this approach is related to planning in STRIPS [21, 36] yet it is performed in a purely deductive context. In [70] or [7] it is illustrated that this fundamental difference allows for applying this approach to more general problem classes in the context of reasoning about actions and change, e.g. postdiction problems, nondeterministic actions, reasoning about hypothetical developments of a dynamical system [68], and concurrent actions.

but not amongst the conditions. Hence, the task left is to add the effects $(e_1 \circ \cdots \circ e_n)\theta$ to the term $V\theta$ yielding the new situation $(V \circ e_1 \circ \cdots \circ e_n)\theta$.[13] For instance, the application of (8) to (6) via $\theta = \{X \mapsto ho \circ ra^- \circ sw,\, V \mapsto ho\}$ results in the new situation term

$$ho \circ sw \circ fun \circ \mathfrak{co}(ho \circ ra^- \circ sw \circ fun).$$

The reasoning process described so far can be encoded via two program clauses defining the ternary predicate *causes* with the intended meaning that the instance $causes(i, [a_1, \ldots, a_n], g)$ is true if the middle argument is a sequence of action names whose successive application to the situation term i yields the situation term g:[14]

$$\begin{array}{l} causes(I, [\,], G) \leftarrow I =_{\mathrm{AC1}} G. \\ \\ causes(I, [A \mid P], G) \leftarrow \begin{array}{l} action(C, A, E), \\ C \circ V =_{\mathrm{AC1}} I, \\ \neg\, inconsistent(V \circ E), \\ causes(V \circ E, P, G). \end{array} \end{array} \tag{11}$$

Hence, if the empty sequence of actions is applied then i and g are required to be identical modulo (AC1). Otherwise we have to find an action description with name $A = a_1$, conditions C, and effects E such that an instance of C is contained in i and the resulting situation $V \circ E$ is consistent and used as the first argument of the recursive call which employs the remainder $P = [a_2, \ldots, a_n]$ of the sequence of actions.

Finally, we have to define the notion of inconsistency in view of our application. According to Definition 3.2, 2(b), a situation is defined to be inconsistent iff it contains a subterm $\mathfrak{co}(\tau_\Phi)$ such that Φ is (classically) inconsistent. The latter condition can be easily fixed via Remark 4.2 so that we need the clause

$$inconsistent(V \circ \mathfrak{co}(X \circ Y \circ Y^-)). \tag{12}$$

Example (8) already illustrates how defaults can be encoded via unit clauses representing the respective action descriptions. In general, the conditions of an action description of the form (7) consist of the prerequisite of the corresponding default — e.g. *sw* in (8) — along with the subterm $\mathfrak{co}(X)$ in order to be prepared for changing the current context of reasoning given by the set of constraints. As it is not intended to lose the prerequisite, it is included in the effects together with the consequence — e.g. *fun* in (8). Furthermore, the set of constraints is augmented by both justification and consequence of the default — which both are the atom *fun* in (8). Formally, given a finite set $\{\delta_1 = \frac{\alpha_1 : \beta_1}{\omega_1}, \ldots, \delta_m = \frac{\alpha_m : \beta_m}{\omega_m}\}$ of closed, conjunctive defaults, the following program clauses are generated:

$$\begin{array}{l} action(\tau_{\alpha_1} \circ \mathfrak{co}(X), \delta_1, \tau_{\alpha_1} \circ \tau_{\omega_1} \circ \mathfrak{co}(X \circ \tau_{\beta_1} \circ \tau_{\omega_1})). \\ \vdots \\ action(\tau_{\alpha_m} \circ \mathfrak{co}(X), \delta_m, \tau_{\alpha_m} \circ \tau_{\omega_m} \circ \mathfrak{co}(X \circ \tau_{\beta_m} \circ \tau_{\omega_m})). \end{array} \tag{13}$$

[13] Note that this method does not require any additional effort to solve the frame problem (c.f. Footnote 6) because each member of a situation term which is not amongst the conditions or effects, respectively, is obviously contained in the resulting situation term. For this solution to the frame problem it is important that the function $\circ$ is not required to be idempotent (i.e. $t \circ t \neq_{\mathrm{AC1}} t$) since otherwise (10) would have the second, unintended solution $\theta = \{X \mapsto ho \circ ra^- \circ sw, V \mapsto ho \circ sw \circ \mathfrak{co}(ho \circ ra^- \circ sw)\}$. This would be undesired as V contains the subterm $\mathfrak{co}(ho \circ ra^- \circ sw)$ which shall not be amongst the facts which continue to be true in the resulting situation (see e.g. [30]).

[14] In what follows we adopt the usual PROLOG notation $[h|t]$ to denote a sequence with head h and tail t.

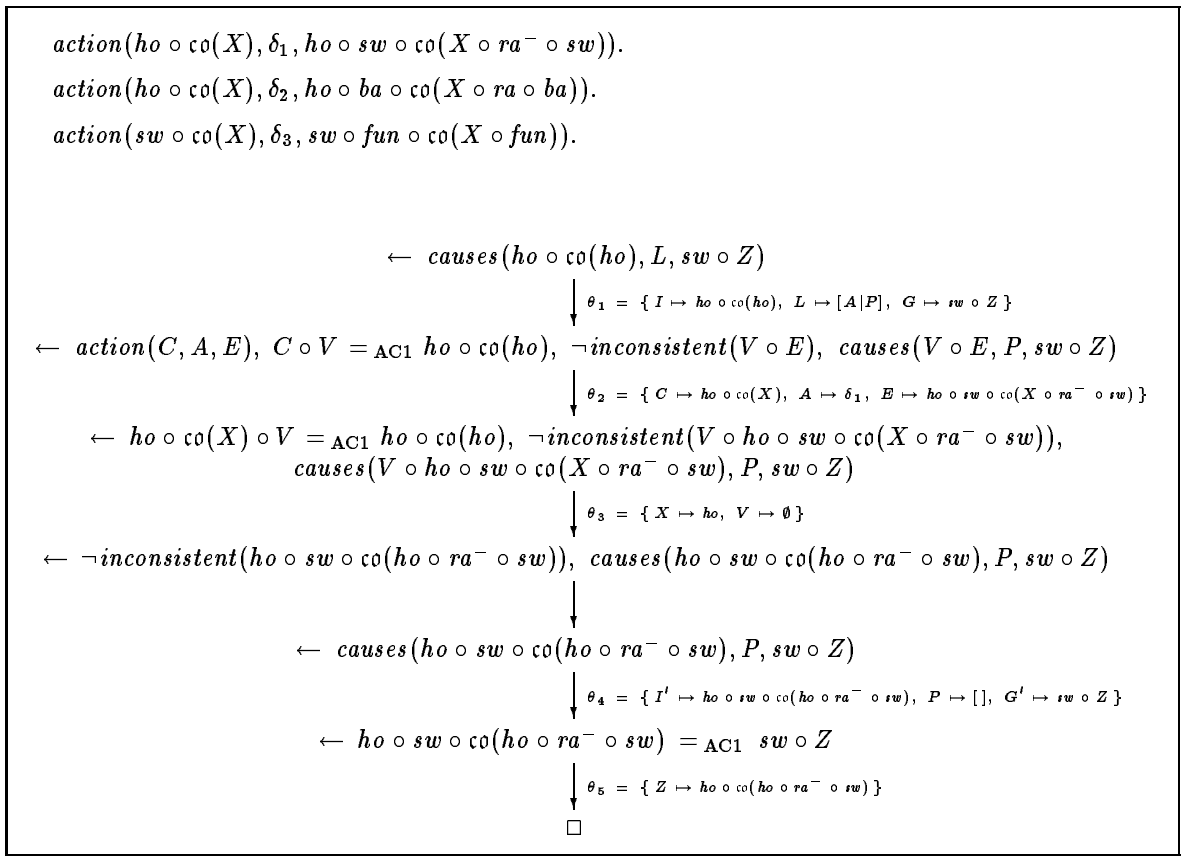

Figure 2: An SLDENF-refutation of $(P_{\Delta_2}, \mathrm{AC1}) \cup \{ \leftarrow causes(ho \circ co(ho), L, sw \circ Z) \}$, where P_{Δ_2} consists of the clauses (11) and (12) along with the facts depicted at the top.

To summarize, a closed, conjunctive default theory $\Delta = (D, W)$ is translated into the equational logic program $(P_\Delta, \mathrm{AC1})$ where P_Δ consist of the clauses (11), (12), and (13).

As an example, consider the default theory Δ_2 of Example 2.2 whose defaults are represented via the three clauses depicted at the top of Figure 2. Let P_{Δ_2} consist of the clauses (11) and (12) along with these three facts. As the background knowledge of Δ_2 is $\{ho\}$, the initial situation is represented by the AC1-term $ho \circ co(ho)$ according to Definition 3.2. Now, in order to test whether the planning problem $\mathcal{P}_{\Delta_2, sw}$, say, is solvable we try to derive the empty goal $\square$ given the goal $\leftarrow causes(ho \circ co(ho), L, sw \circ Z)$ via SLDENF-resolution and $(P_{\Delta_2}, \mathrm{AC1})$. In other words, we try to find a sequence of actions L such that its application to $ho \circ co(ho)$ yields the situation $sw \circ Z$ for some arbitrary Z. A corresponding SLDENF-refutation is depicted in Figure 2.[15] Observe that the fourth step (the one not labeled with a unifier) is justified by the fact that the goal $\leftarrow inconsistent(ho \circ sw \circ co(ho \circ ra^- \circ sw))$ fails as the two terms $ho \circ sw \circ co(ho \circ ra^- \circ sw)$ and $V \circ co(X \circ Y \circ Y^-)$ (cf. clause (12)) are not AC1-unifiable. The refutation yields the computed answer substitution $\{L \mapsto [\delta_1], Z \mapsto ho \circ co(ho \circ ra^- \circ sw)\}$. Hence, $[\delta_1]$ is a solution to our planning problem $\mathcal{P}_{\Delta_2, sw}$.

By investigating this refutation it becomes obvious that applying the second clause in (11) and solving the first three subgoals resembles the topmost planning step depicted in Figure 1.

[15] The variables I' and G' denote new copies of the variables I and G in the first program clause of (11).

This observation provably holds in general:

Lemma 4.3 *Let $\Delta = (D, W)$ be a closed, conjunctive default theory which determines a set of situations Σ along with a set of actions Ω according to Definition 3.2. Furthermore, let $\mathcal{S} \cup \{\mathfrak{co}(\Phi)\}$ be a situation where both the set of conjunctions $\mathcal{S}$ and the conjunction Φ are consistent. Then, an action δ is applicable in $\mathcal{S} \cup \{\mathfrak{co}(\Phi)\}$ and yields $\mathcal{S}' \cup \{\mathfrak{co}(\Phi')\}$ iff the goal $\leftarrow causes(\tau_{\mathcal{S}} \circ \mathfrak{co}(\tau_\Phi), [\delta|p], g)$ can be reduced to $\leftarrow causes(\tau_{\mathcal{S}'} \circ \mathfrak{co}(\tau_{\Phi'}), p, g)$ for arbitrary terms p, g via $(P_\Delta, \mathrm{AC1})$ and SLDENF-resolution.*

Proof: Let $\delta = \frac{\alpha : \beta}{\omega}$. The goal $\leftarrow causes(\tau_{\mathcal{S}} \circ \mathfrak{co}(\tau_\Phi), [\delta|p], g)$ can always be AC1-unified with the head of the second definition of *causes* in (11) but it is not unifiable with the head of any other clause, since $[\delta|p]$ and $[\,]$ are never AC1-unifiable. Using the AC1-unifier $\{I \mapsto \tau_{\mathcal{S}} \circ \mathfrak{co}(\tau_\Phi),\ A \mapsto \delta,\ P \mapsto p,\ G \mapsto g\}$, the resulting goal is

$$\begin{aligned} \leftarrow\ & action(C, \delta, E), \\ & C \circ V =_{\mathrm{AC1}} \tau_{\mathcal{S}} \circ \mathfrak{co}(\tau_\Phi), \\ & \neg inconsistent(V \circ E), \\ & causes(V \circ E, p, g). \end{aligned}$$

Since $\delta \in D$, the first literal in this goal is unifiable with the head of exactly one clause, viz. the particular clause in (13) which stems from the translation of δ. Using the AC1-unifier $\{C \mapsto \tau_\alpha \circ \mathfrak{co}(X),\ E \mapsto \tau_\alpha \circ \tau_\omega \circ \mathfrak{co}(X \circ \tau_\beta \circ \tau_\omega)\}$ the resulting goal is

$$\begin{aligned} \leftarrow\ & \tau_\alpha \circ \mathfrak{co}(X) \circ V =_{\mathrm{AC1}} \tau_{\mathcal{S}} \circ \mathfrak{co}(\tau_\Phi), \\ & \neg inconsistent(V \circ \tau_\alpha \circ \tau_\omega \circ \mathfrak{co}(X \circ \tau_\beta \circ \tau_\omega)), \\ & causes(V \circ \tau_\alpha \circ \tau_\omega \circ \mathfrak{co}(X \circ \tau_\beta \circ \tau_\omega), p, g). \end{aligned}$$

The unification problem corresponding to the first literal has a solution iff each element occurring in τ_α is also contained in $\tau_{\mathcal{S}}$, i.e. if and only if $\mathcal{S} \models \alpha$ according to Remark 4.2. In this case, there is a unique (modulo AC1) most-general unifier which substitutes X by τ_Φ and V by a term v such that $\tau_\alpha \circ v =_{\mathrm{AC1}} \tau_{\mathcal{S}}$. Hence, the above goal reduces to

$$\begin{aligned} \leftarrow\ & \neg inconsistent(\tau_{\mathcal{S}} \circ \tau_\omega \circ \mathfrak{co}(\tau_\Phi \circ \tau_\beta \circ \tau_\omega)), \\ & causes(\tau_{\mathcal{S}} \circ \tau_\omega \circ \mathfrak{co}(\tau_\Phi \circ \tau_\beta \circ \tau_\omega), p, g). \end{aligned}$$

Due to Clause (12) defining consistency and since there is exactly one subterm with leading function symbol $\mathfrak{co}$ in the argument of the first literal above, we find that $\leftarrow inconsistent(\tau_{\mathcal{S}} \circ \tau_\omega \circ \mathfrak{co}(\tau_\Phi \circ \tau_\beta \circ \tau_\omega))$ fails iff $\tau_\Phi \circ \tau_\beta \circ \tau_\omega$ does not contain two subterms of the form t and t^-. According to Remark 4.2, this is equivalent to $\Phi \cup \{\beta\} \cup \{\omega\} \not\models \bot$.

To summarize, we are left with the goal

$$\leftarrow\ causes(\tau_{\mathcal{S}} \circ \tau_\omega \circ \mathfrak{co}(\tau_\Phi \circ \tau_\beta \circ \tau_\omega), p, g) \tag{14}$$

iff the two requirements (a) and (b) of Definition 3.2 are satisfied, i.e. if δ is applicable in $\mathcal{S} \cup \{\mathfrak{co}(\Phi)\}$. Furthermore, $\mathcal{S}' \equiv \mathcal{S} \cup \{\omega\}$ and $\Phi' \equiv \Phi \wedge \beta \wedge \omega$ according to Definition 3.2. Relating this to our final goal (14) proves our claim. ∎

Having proved that the application of the second clause in (11) adequately models the performance of a single planning step, we are now prepared to apply the ELP based approach to

the problem class discussed in Section 3. Recall that plan p is a solution to a given planning problem $\mathcal{P}_{\Delta,g}$ iff g is classically entailed by the situation which is obtained after applying p. Following Remark 4.2, this can be easily checked in case of a conjunctive theory Δ and if g is a variable-free conjunction of literals as well, provided the resulting situation $\mathcal{S}_n$ is consistent. This is guaranteed whenever the original default theory is consistent.[16] Hence, to ensure that $\mathcal{S}_n \models g$, we simply require the corresponding term $\tau_{\mathcal{S}_n}$ to be of the form $\tau_g \circ z$ for some arbitrary term z (see also Figure 2):

Theorem 4.4 *Let $\Delta = (D, W)$ be a consistent closed and conjunctive default theory, g a variable-free conjunction, and p a finite sequence of elements of D. Then p is a solution to the corresponding planning problem $\mathcal{P}_{\Delta,g}$ iff there is an SLDENF-refutation for $\leftarrow \mathit{causes}(\tau_W \circ \mathfrak{co}(\tau_W), p, \tau_g \circ Z)$ wrt. $(P_\Delta, \mathrm{AC1})$.*

Proof: Let n be the length of $p = [\delta_1, \ldots, \delta_n]$.

In case $n = 0$, p solves $\mathcal{P}_\Delta$ iff $W \models g$. Correspondingly, the goal $\leftarrow \mathit{causes}(\tau_W \circ \mathfrak{co}(\tau_W), [\,], \tau_g \circ Z)$ can only be resolved with the first definition of *causes* in (11) since $[\,]$ and $[A|P]$ are not AC1-unifiable. Hence, we can find an SLDENF-refutation for the goal iff $\tau_W \circ \mathfrak{co}(\tau_W)$ and $\tau_g \circ Z$ are AC1-unifiable which is true iff each element occurring in τ_g is also contained in τ_W. This is equivalent to $W \models g$ according to Remark 4.2 since W is consistent by assumption.

If $n > 0$ then by successively applying Lemma 4.3 n-times we conclude that applying p to $W \cup \{\mathfrak{co}(W)\}$ yields a situation $\mathcal{S} \cup \{\mathfrak{co}(\Phi)\}$ if and only if the goal $\leftarrow \mathit{causes}(\tau_W \circ \mathfrak{co}(\tau_W), p, \tau_g \circ Z)$ can be reduced to some $\leftarrow \mathit{causes}(s \circ \mathfrak{co}(t), [\,], \tau_g \circ Z)$ such that $s =_{\mathrm{AC1}} \tau_{\mathcal{S}}$ and $t =_{\mathrm{AC1}} \tau_\Phi$. Now we can easily apply the argument above (where $n = 0$) to obtain the result. ■

In order to generate a plan, i.e. a default proof, the middle argument in a goal of the form $\leftarrow \mathit{causes}(i, p, g)$ can be left variable:

Corollary 4.5 *Let $\Delta = (D, W)$ be a consistent closed and conjunctive default theory and g a variable-free conjunction. Then g is provable from Δ iff $\leftarrow \mathit{causes}(\tau_W \circ \mathfrak{co}(\tau_W), P, \tau_g \circ Z)$ has an SLDENF-refutation wrt. $(P_\Delta, \mathrm{AC1})$.*

Each refutation of such a goal determines a binding for the variable P which then denotes a plan that solves the planning problem under consideration.

In case Δ is inconsistent, the empty plan solves $\mathcal{P}_{\Delta,g}$ for any g. Inconsistency of a default theory means inconsistency of the background knowledge W. This can be checked in conjunctive default theories wrt. clause (12) using the goal $\leftarrow \mathit{inconsistent}(\mathfrak{co}(\tau_W))$.

On the analogy of the discussion at the very end of the previous section, it is also possible to formalize open defaults in this approach. For instance, the default $\delta(x) = \frac{\mathit{bird}(x) : \mathit{flies}(x) \wedge \neg \mathit{peng}(x)}{\mathit{flies}(x)}$ can be encoded via the fact

$$\mathit{action}\ (\ \mathit{bird}(X) \circ \mathfrak{co}(Y)\ ,\ \delta(X)\ ,\ \mathit{bird}(X) \circ \mathit{flies}(X) \circ \mathfrak{co}(Y \circ \mathit{flies}(X) \circ \mathit{peng}(X)^-)\ ).$$

Now, for instance, let τ_W be an abbreviation of the term $\mathit{bird}(a) \circ \mathit{bird}(b) \circ \mathit{peng}(b)$ then it is easy to verify that the goal

$$\leftarrow\ \mathit{causes}\ (\ \tau_W \circ \mathfrak{co}(\tau_W)\ ,\ [\delta(a)]\ ,\ G)$$

can be reduced to

$$\leftarrow\ \mathit{causes}\ (\ \tau_W \circ \mathit{flies}(a) \circ \mathfrak{co}(\tau_W \circ \mathit{flies}(a) \circ \mathit{peng}(a)^-)\ ,\ [\,]\ ,\ G)$$

[16] This observation follows inductively from Lemma 4.3 and the fact that the application of a default to a consistent situation $\mathcal{S} \cup \mathfrak{co}(\Phi)$ yields again a consistent situation.

in the spirit of Lemma 4.3. In contrast, each SLDENF-derivation of

$$\leftarrow \; causes\,(\,\tau_W \circ \mathfrak{co}(\tau_W)\,,\,[\delta(b)]\,,\,G)$$

fails.

The reader should be aware of the fact that the approach presented in this section, as it stands, is just an straightforward implementation of the ideas developed in Section 3. To constitute a satisfactory and competitive system, improvements regarding efficiency are needed. For instance, AC1-unification is known to be hard in general [10, 32] but, as argued in [26], we are only concerned with very restricted unification problems. This boils down to testing sub-multiset relations, which can be computed much more efficiently, namely in polynomial time. Moreover, loop detecting mechanisms [64] should be applied to suppress the application of defaults that do not provide new information. These and other aspects have been discussed in e.g. [9].

5 Skeptical Reasoning

Now we turn to the problem of skeptical reasoning in our exemplary default logic, namely Constrained Default Logic. Recall that a formula is skeptically entailed by a default theory iff it is contained in all of its extensions.

Example 5.1 Let $\Delta_3 = (D_3, W_3)$ be our default theory Δ_2 (cf. Example 2.2) augmented by yet another default, viz. $\delta_4 = \frac{ba\,:\,fun}{fun}$. We obtain $D_3 = \{\delta_1 = \frac{ho\,:\,\neg ra}{sw}, \delta_2 = \frac{ho\,:\,ra}{ba}, \delta_3 = \frac{sw\,:\,fun}{fun}, \delta_4 = \frac{ba\,:\,fun}{fun}\}$ and, as before, $W_3 = \{ho\}$. Δ_3 has two constrained extensions, viz. $(Th(\{ho, sw, fun\}), Th(\{ho, \neg ra \wedge sw, fun\}))$ and $(Th(\{ho, ba, fun\}), Th(\{ho, ra \wedge ba, fun\}))$. Both of them contain fun, hence this atom is skeptically entailed by Δ_3. On the other hand, neither sw nor ba are in all extensions while $ho \in W_3$ trivially is skeptically entailed as well.

Note that in order to prove skeptical entailment of a formula g, it does not suffice to show that there is no default proof of $\neg g$, i.e. no extension containing $\neg g$, because there might be extensions which make no statement about g at all.

A naïve way to guarantee membership in every extension is to check all of them. From a practical point of view, this is obviously not satisfactory, because it does not reflect the idea of locality whose important rôle has already been elaborated in the previous sections. Consider for instance a default $\delta = \frac{:\,a}{a}$ and assume that the atom a does not occur elsewhere. Certainly, a is skeptically entailed whatever the concrete extensions are, since δ can be regarded as a kind of universal or unassailable proof of a. Hence there is no need to check all extensions for a. Moreover, it is of course difficult to check exponentially many extensions (in the worst case), or even to create and investigate entire extensions built up from defaults totally unrelated to the considered query.

Here, we follow a more promising approach whose underlying principle was originally applied by Poole [50] to a restricted version of his nonmonotonic THEORIST formalism [47, 48], and which was extended to the entire THEORIST framework by the first author [67]. Informally, the approach is based on the notion of a discourse in which two protagonists alternately raise arguments and counterarguments. Specifically, to prove skeptical entailment the first protagonist tries to find a single default proof of the formula under consideration, then his antagonist replies by giving a counterargument which "annuls" this proof (this shall be formalized below). Afterwards, it is again the first protagonist who searches for another default proof in view of the restriction determined by the preceding counterargument, and so on. This procedure ends if it is impossible to find either a default proof or a counterargument at some state. In the former

case, the formula is not skeptically entailed while it is in the latter. Note that this method takes into account locality: Take, for instance, the default $\delta = \frac{:a}{a}$ which is a default proof of a that cannot be refuted if a does not occur elsewhere in the theory. In this case, the above procedure terminates with success after just a single step.

In the sequel, we develop a formal approach along with an algorithm based on the above description while following the line of [67]. We start with the formal definition of skeptical entailment in Constrained Default Logic. For this purpose, we have to formalize the aforementioned notion of discourse. This involves in particular the formalization of default proofs taking into account the restrictions imposed by the antagonist's counterarguments. This is accomplished by a simple yet powerful extension to Constrained Default Logic, introduced in [59]. The basic idea is to supplement the set of constraints found in a constrained extension with some sort of initial consistency constraints.[17] The purpose of these constraints is to direct the reasoning process by enforcing their consistency. This is a well-known technique, also used in THEORIST [47]. These constraints allow us to capture the aforementioned restrictions on default proofs. Formally, a default theory becomes a triple (D, W, C_P),[18] where D and W are as before and C_P is some set of formulas. Then, a (pre-)constrained extension is specified as in Theorem 2.4 with the exception that $C_0 := W \cup C_P$. This results in the following counterpart to Theorem 2.6:

Theorem 5.1 *Let (D, W) be a default theory and E, C be two sets of formulas. (E, C) is a constrained extension of (D, W) iff there exists a maximal (wrt. set inclusion), grounded set D' such that $D' \subseteq D$, $W \cup C_P \cup Justif(D') \cup Conseq(D') \not\models \bot$ and the following holds:*

1. $E = Th(W \cup Conseq(D'))$
2. $C = Th(W \cup Justif(D') \cup Conseq(D'))$

Proof: Analogous to the one in [60]. ∎

The only difference between the previous specification and the one given in Theorem 2.6 is that C_P enters the consistency criterion in Theorem 5.1.

The notion of a base is also extended in the obvious way: A *base* of a default theory (D, W, C_P), is a subset $D' \subseteq D$ such that D' is grounded and $W \cup C_P \cup Justif(D') \cup Conseq(D') \not\models \bot$. All other notions, like that of a default proof, remain the same. Hence, the notion of skeptical entailment in Constrained Default Logic is formally defined as follows:

Definition 5.2 A closed formula g is *skeptically provable* from a default theory $\Delta = (D, W, C_P)$ iff for each constrained extension (E, C) of Δ, we have $g \in E$.

Another central rôle is played by the notion of credulous proving *from certain bases*. This complements the proof-theoretic counterpart of finding a default proof under certain restrictions. Formally, we say that a closed formula g is provable *from a base* D' of some $\Delta = (D, W, C_P)$ iff there exists a default proof D'' of g from Δ such that $D'' \supseteq D'$. Recall Example 5.1: Although sw is provable from Δ_3, it is not provable from base $\{\delta_2\}$ since δ_1 is inapplicable once δ_2 has been applied. This leads us to the following characterization of skeptical entailment based on the concept of credulous proofs from given bases:

Lemma 5.3 *Let $\Delta = (D, W, C_P)$ be a default theory. A closed formula g is skeptically provable from Δ iff it is provable from every base of Δ.*

[17] Called *pre-constraints* in [59].

[18] Such a default theory is called a *pre-constrained default theory* in [59].

Proof:

"$\Rightarrow$": Assume that g is skeptically provable from Δ. For each base D' of Δ we can find some maximal $D'' \supseteq D'$ such that (E'', C'') is a constrained extension of Δ, where

$$\begin{aligned} E'' &= Th(W \cup Conseq(D'')) \\ C'' &= Th(W \cup C_P \cup Justif(D'') \cup Conseq(D'')) . \end{aligned}$$

From the fact that g is skeptically provable, we conclude that $g \in E''$. Hence, $W \cup Conseq(D'') \models g$. In other words, g is provable from D' (by using D'').

"$\Leftarrow$": Assume that g is provable from every base. For each constrained extension (E', C') of $\Delta = (D, W, C_P)$ we can find a corresponding maximal base $D' \subseteq D$ generating this extension according to Theorem 5.1.

Let $D'' \supseteq D'$ be a default proof of g from D', i.e. $W \cup Conseq(D'') \models g$. From D' being maximal we conclude that $D'' = D'$ and, hence, $g \in E' = Th(W \cup Conseq(D'))$. ■

A second crucial point is the formalization of the concept of a counterargument against a proof. Similar to the standard definition, we introduce the notion of *orthogonality*[19] of bases:

Definition 5.4 Let (D, W, C_P) be a default theory. Two bases D' and D'' of (D, W, C_P) are called *C_P-orthogonal* iff

$$W \cup C_P \cup Justif(D') \cup Conseq(D') \cup Justif(D'') \cup Conseq(D'') \models \bot .$$

We simply say orthogonal whenever the given set of constraints is empty, i.e. $C_P = \{\}$. For instance, in Example 5.1 $\{\delta_1\}$ and $\{\delta_2\}$ are orthogonal due to $\neg ra \in Justif(\{\delta_1\})$ and $ra \in Justif(\{\delta_2\})$.

As orthogonality is the formal characterization of what we call "annulling" a counterargument, we are now prepared to formalize the discourse-based approach described at the beginning. The following theorem claims that in order to prove skeptical entailment (i.e. provability from every base according to Lemma 5.3), it suffices to find a default proof and, furthermore, to ensure that the formula under consideration is skeptically provable *from every counterargument* (i.e. from every orthogonal base).

Theorem 5.5 *Let $\Delta = (D, W, C_P)$ be a default theory and g be a closed formula. Then, g is provable from every base of Δ iff there exists a default proof D' of g from Δ such that the following holds: For all bases D'' of Δ which are C_P-orthogonal to D', g is provable from D''.*

Proof:

"$\Rightarrow$": If g is provable from every base then it is in particular provable from the empty base $\{\}$. Hence, there is a default proof of g from Δ. Furthermore, by assumption, g is clearly also provable from every C_P-orthogonal (wrt. D') base.

"$\Leftarrow$": We have to show that g is provable from every base of Δ. By assumption, g is provable from every base which is C_P-orthogonal to D'. Now, if D'' is a base which is not C_P-orthogonal to D' then the combination $D' \cup D''$ is also a base of Δ because it is obviously grounded and we also have

$$W \cup C_P \cup Justif(D') \cup Conseq(D') \cup Justif(D'') \cup Conseq(D'') \not\models \bot .$$

$D' \cup D''$ is a default proof of g from D'' since just D' itself proves g. Hence, g is also provable from every non-C_P-orthogonal (wrt. D') base, which proves our claim. ■

[19] The term *orthogonality* was used in [51] to express that two extensions are mutually contradictory to each other.

Example 5.1 (continued) A default proof of *fun* from Δ_3 [20] is $D' = \{\delta_1, \delta_3\}$. There are two bases orthogonal ($\{\}$-orthogonal, to be precise) to D', namely $D''_1 = \{\delta_2\}$ and $D''_2 = \{\delta_2, \delta_4\}$. Since $\{\delta_2, \delta_4\}$ is also a default proof of *fun*, we conclude that *fun* is provable from D''_1 as well as D''_2. Hence, *fun* is skeptically entailed. On the other hand, although D' is also a proof of *sw*, this atom is neither provable from D''_1 nor from D''_2, thus it is not skeptically provable. Furthermore, $\{\}$ is a proof of $ho \in W_3$. Since $\{\}$ is not orthogonal to any base, *ho* is therefore skeptically entailed as well.

The above theorem suggests to investigate all orthogonal bases. This is likely to be a heavy task and appears to be no improvement compared to the naïve way of checking every extension from the start. Fortunately it suffices to investigate only *minimal* (wrt. set inclusion) orthogonal bases if we perform skeptical instead of credulous reasoning from these bases. This is a consequence of the following observation.

Lemma 5.6 *Let $\Delta = (D, W, C_P)$ be a default theory and g be a closed formula. Furthermore, let D' be a base of Δ. Then, g is provable from every base $D'' \supseteq D'$ of Δ iff g is skeptically provable from $\Delta_{D'} := (D, W \cup Conseq(D'), C_P \cup Justif(D'))$.*

Proof: If D'' is a base of Δ such that $D'' \supseteq D'$ then D'' is also a base of $\Delta_{D'}$. Conversely, a base D'' of $\Delta_{D'}$ is also a base of Δ and, moreover, D'' is equivalent to $D'' \cup D'$ in any case.[21] Hence, the set of bases of $\Delta_{D'}$ equals the set of bases of Δ which are supersets of D'. Thus, the claim follows immediately from Lemma 5.3. ■

Example 5.2 Consider a slight modification of our default theory Δ_3 (cf. Example 5.1): Let $\Delta_4 = (D_4, W_4, \{\})$ where $D_4 = \{\delta_0 = \frac{:\neg work}{ho}\} \cup D_3$ and $W_4 = \{\}$. Furthermore, let $D' = \{\delta_0\}$ be a base of Δ_4. From the preceding discussion of Example 5.1, we conclude that *fun* is provable from every base $D'' \supseteq D'$ of Δ_4. Correspondingly, *fun* is skeptically provable from $\Delta_{D'} = (D_4, \{ho\}, \{\neg work\})$. [22]

For skeptical query-answering, we can thus start with a single credulous proof. Then, we have to ensure the provability of our query from all orthogonal bases — in the spirit of Theorem 5.5. For this, we take advantage of Lemma 5.6 and we simply investigate the minimal elements among all orthogonal bases. Then, we create the respective modified default theories (denoted by $\Delta_{D'}$ above) and use each such default theory for a recursive call of the overall algorithm.

Finally, there is an important task left, namely how are we to determine the set of all (minimal) orthogonal bases given a particular default proof? The key idea is again to map this onto credulous reasoning. Recall Definition 5.4 of orthogonality where not only the consequences of the involved defaults have to be considered but also the justifications. For this purpose, we introduce the notion of *normalizing* a default theory by adding the justifications of each default to the respective consequences. Formally, if D is a set of defaults then its *normalized* variant $\overline{D}$ is defined by[23]

$$\overline{D} = \{ \tfrac{\alpha : \beta}{\beta \wedge \omega} \mid \tfrac{\alpha : \beta}{\omega} \in D \}.$$

Based on this concept, the following theorem induces an elegant way of generating orthogonal bases. For notational convenience, let $JC(D)$ be an abbreviation for $Justif(D) \cup Conseq(D)$ where D is a set of defaults.

[20] Whenever there is no mention of any initial constraints, we assume that there are no such constraints. For Δ_3, we thus have $(D_3, W_3, \{\})$.

[21] We call two bases to be equivalent iff they determine identical sets of constrained extensions.

[22] Observe that this is the first place where we deal with a non-empty set of initial constraints.

[23] In what follows, we assume for simplicity that $\overline{D}$ provides a one-to-one mapping between default rules and their normalized format. An easy way of achieving this is to label defaults.

Theorem 5.7 *Let $\Delta = (D, W, C_P)$ be a default theory and g be a closed formula. Furthermore, let $D' = \{\delta_1, \ldots, \delta_n\}$ be a finite base and D'' be an arbitrary base of Δ. Then, D' and D'' are C_P-orthogonal iff there exists some $i \in \{1, \ldots, n\}$ such that $\overline{D''}$ is a default proof of $\neg(Justif(\delta_i) \wedge Conseq(\delta_i))$ from the default theory $\overline{\Delta}_{i-1} = (\overline{D}, W \cup JC(\{\delta_1, \ldots, \delta_{i-1}\}), C_P)$.*

Proof:
"$\Rightarrow$": Since D' and D'' are C_P-orthogonal, we have

$$W \cup C_P \cup JC(\{\delta_1, \ldots, \delta_n\}) \cup JC(D'') \models \bot$$

according to Definition 5.4. Following the deduction theorem [19] there must be some $i \in \{1, \ldots, n\}$ such that

$$W \cup C_P \cup JC(\{\delta_1, \ldots, \delta_{i-1}\}) \cup JC(D'') \models \neg(Justif(\delta_i) \wedge Conseq(\delta_i)) \qquad (15)$$

and the left hand side of (15) is consistent.[24] Now, let $\overline{D''} \subseteq \overline{D}$ denote the set of defaults which correspond to the elements in $D'' \subseteq D$. Then, $\overline{D''}$ is a base of $\overline{\Delta}_{i-1}$ as D'' is grounded and the left hand side of (15) is consistent. Furthermore, (15) and $Conseq(\overline{D''}) \equiv Justif(D'') \cup Conseq(D'')$ imply that that $\overline{D''}$ is a proof of $\neg(Justif(\delta_i) \wedge Conseq(\delta_i))$ from $\overline{\Delta}_{i-1}$.
"$\Leftarrow$": If $\overline{D''}$ is a default proof of $\neg(Justif(\delta_i) \wedge Conseq(\delta_i))$ from $\overline{\Delta}_{i-1}$ then, we have

$$W \cup C_P \cup JC(\{\delta_1, \ldots, \delta_{i-1}\}) \cup JC(\overline{D''}) \models \neg(Justif(\delta_i) \wedge Conseq(\delta_i)).$$

That is, $W \cup C_P \cup JC(\{\delta_1, \ldots, \delta_i\}) \cup JC(\overline{D''})$ and then also $W \cup C_P \cup JC(\{\delta_1, \ldots, \delta_i\}) \cup JC(D'')$ are inconsistent. Hence, $D' \supseteq \{\delta_1, \ldots, \delta_i\}$ and D'' are C_P-orthogonal. ■

Example 5.1 (continued) We have already observed earlier that Δ_3 has two bases orthogonal to $\{\delta_1\}$, viz. $D_1'' = \{\delta_2\}$ and $D_2'' = \{\delta_2, \delta_4\}$. Correspondingly, let $\overline{\Delta} = (\overline{D_3}, W_3, \{\})$, where $\overline{D_3} = \{\overline{\delta_1} = \frac{ho : \neg ra}{\neg ra \wedge sw}, \overline{\delta_2} = \frac{ho : ra}{ra \wedge ba}, \overline{\delta_3} = \frac{sw : fun}{fun}, \overline{\delta_4} = \frac{ba : fun}{fun}\}$, then both $\overline{D_1''}$ and $\overline{D_2''}$ are proofs of $\neg(Justif(\delta_1) \wedge Conseq(\delta_1)) \equiv sw \vee ra$ since $Conseq(\overline{\delta_2}) \models ra$.

Now, we are ready to formulate our algorithm for skeptical reasoning in our exemplary system, Constrained Default Logic. In fact, Theorem 5.5 (in conjunction with Lemma 5.6) and Theorem 5.7 provide the formal foundations for this undertaking. The resulting algorithm is depicted in Figure 3. It starts with searching for a credulous default proof D' of the given formula g (Step 1). Notably, the choice of the (single) credulous default proof is a *"don't care"*-choice. That is, the result is independent of what credulous default proof is taken. If there is no such proof (Step 2) then g cannot be contained in any extension (see Lemma 5.3) of the default theory at hand. Otherwise, in Step 3 along with Step 3.a, all minimal orthogonal bases wrt. the proof D' are generated according to Theorem 5.7. Note that we employ a function *credulous_default_proof*(Δ', g'), which is assumed to return *minimal* (wrt. set inclusion) proofs of some g' from some Δ'. Generating only minimal proofs is sufficient according to Lemma 5.6. Finally, a recursive call is performed in Step 3.a.1 following the line of Theorem 5.5 and Lemma 5.6. That is, while the consequences of defaults in D'' are added to W, the justifications of the same defaults are added to the initial constraints of theory Δ''. In this way, the recursive call focuses on those extensions compatible with D''. If at least one of these calls returns a negative answer then g is not skeptically entailed. Otherwise, a positive answer is given via Step 4.

[24] Note that at least in case $i = 0$ the left hand side is consistent as D'' is a base.

Algorithm *skeptically_provable*

Input $\Delta = (D, W, C_P)$: default theory
g : closed first-order formula

Step 1 $D' := \mathit{credulous\_default\_proof}\,(\,\Delta\,,\,g\,)$.

Step 2 If there is no such D' then return *no* and stop.

Step 3 Otherwise let $D' = \{\delta_1, \ldots, \delta_n\}$. For all $i = 1, \ldots, n$ do the following

Step 3.a Let $\overline{\Delta}_{i-1} := (\overline{D}, W \cup JC(\{\delta_1, \ldots, \delta_{i-1}\}), C_P)$.
For all $\overline{D}'' := \mathit{credulous\_default\_proof}\,(\,\overline{\Delta}_{i-1}\,,\,\neg(\mathit{Justif}(\delta_i) \wedge \mathit{Conseq}(\delta_i))\,)$
do the following

Step 3.a.1 Let $\Delta'' = (D, W \cup \mathit{Conseq}(D''), C_P \cup \mathit{Justif}(D''))$.
If $\mathit{skeptically\_provable}\,(\,\Delta''\,,\,g\,) = \mathit{no}$ then return *no* and stop.

Step 4 Return *yes*.

Figure 3: This algorithm determines whether a closed formula g is skeptically provable from a default theory Δ . It is assumed that $\mathit{credulous\_default\_proof}(\Delta', g')$ is a function returning minimal default proofs of a closed formula g' from a default theory Δ' .

Example 5.1 (continued) Let us illustrate our algorithm by applying it to $\Delta_3 = (D_3, W_3, \{\}) = (\{\delta_1 = \frac{ho : \neg ra}{sw}, \delta_2 = \frac{ho : ra}{ba}, \delta_3 = \frac{sw : fun}{fun}, \delta_4 = \frac{ba : fun}{fun}\}\,,\,\{ho\}\,,\,\{\})$ and $g = fun$.

Step 1: Assume that $\mathit{credulous\_default\_proof}(\Delta_3, fun)$ yields $D' = \{\delta_1, \delta_3\}$.

Step 3.a: $\overline{\Delta}_0 = (\overline{D_3}, \{ho\}, \{\})$.
$\mathit{credulous\_default\_proof}\,(\overline{\Delta}_0, \neg(\neg ra \wedge sw))$ yields the minimal proof $\overline{D}'' = \{\overline{\delta}_2\}$.

Step 3.a.1: $\mathit{skeptically\_provable}((D_3, \{ho, ba\}, \{ra\}), fun)$ is called:

Step 1: $\mathit{credulous\_default\_proof}((D_3, \{ho, ba\}, \{ra\}), fun)$ yields $D' = \{\delta_4\}$.

Step 3.a: $\overline{\Delta}_0 = (\overline{D_3}, \{ho, ba\}, \{ra\})$.
$\mathit{credulous\_default\_proof}\,(\overline{\Delta}_0, \neg fun)$ fails to find an orthogonal proof.

Step 4: returns *yes*.

Step 3.a: $\overline{\Delta}_1 = (\overline{D_3}, \{ho, \neg ra, sw\}, \{\})$.
$\mathit{credulous\_default\_proof}\,(\overline{\Delta}_1, \neg fun)$ fails to find an orthogonal proof.

Step 4: returns *yes*.

Hence, *fun* is skeptically entailed from Δ_3 .

It is interesting to observe that, apart from the original default theory Δ_3 and its normalized form $\overline{\Delta}_0$, all other theories are extended by formulas imposing restrictions on the

corresponding extensions. Observe, for instance, that the transformation of the original default theory $\Delta_3 = (D_3, \{ho\}, \{\})$ into the theory $(D_3, \{ho, ba\}, \{ra\})$ used in Step 3.a.1 adds the restrictions imposed by the counterargument $\overline{D}'' = \{\overline{\delta}_2\}$ determined in Step 3.a. In this way, we are able to focus on the second constrained extension of default theory Δ_3 (cf. Example 5.1), viz. $(Th(\{ho, ba, fun\}), Th(\{ho, ra \wedge ba, fun\}))$, while the first extension, $(Th(\{ho, sw, fun\}), Th(\{ho, \neg ra \wedge sw, fun\}))$ has been eliminated. That is, the former one is the only constrained extension of default theory $(D_3, \{ho, ba\}, \{ra\}), fun)$. In fact, each recursive call of our algorithm *skeptically_provable* results in a default theory having less extensions than the default theory at hand. These extensions form a subset of the extensions of the previously considered default theory.

Reiter's Normal Default Theories

So far, we focused on Constrained Default Logic defining the underlying notion of extensions. At this point, we want to discuss how our results can be applied to Reiter's original definition in the case of normal default theories. The extension to Łukaszewicz' variant of Default Logic is analogous and for brevity omitted here. To this end, we illustrate how the algorithm depicted in Figure 3 can be simplified in case of normal default theories within the context of classical Default Logic.

For the sake of clarity, we call an extension in the sense of Definition 2.1 a *classical* extension. It has been shown that in case of normal default theories, Reiter's definition and the definition of constrained extensions coincide [58, 17]:

Theorem 5.8 *Let (D, W) be a normal default theory and E be a set of formulas. Then, E is a classical extension of (D, W) iff (E, E) is a constrained extension of (D, W).*

Based on this observation, the method developed above for skeptical reasoning in Constrained Default Logic can be directly applied to classical Default Logic as soon as only normal defaults are involved. To this end, we do not have to distinguish between justifications and consequents anymore. In this way, the treatment of additional constraints along with the extended notion of a default theory becomes obsolete. Now, a *classical base* of a normal default theory (D, W) is defined as a grounded set $D' \subseteq D$ such that $W \cup Conseq(D') \not\models \bot$. Two classical bases D' and D'' are then called *orthogonal* iff $W \cup Conseq(D) \cup Conseq(D') \models \bot$. Following Theorem 5.7 we obtain a similar result which allows for computing orthogonal, classical bases via credulously reasoning:

Corollary 5.9 *Let $\Delta = (D, W)$ be a closed, normal default theory and g be a closed formula. Furthermore, let $D' = \{\delta_1, \ldots, \delta_n\}$ be a finite, classical base and D'' be an arbitrary classical base of Δ. Then, D' and D'' are orthogonal iff there exists some $i \in \{1, \ldots, n\}$ such that D'' is a default proof of $\neg Conseq(\delta_i)$ from the normal default theory $(D, W \cup Conseq(\{\delta_1, \ldots, \delta_{i-1}\}))$.*

Hence, the algorithm depicted in Figure 3 can be simplified in case of normal default theories as follows. Steps 1–3 as well as Step 4 remain unchanged (except that there are no constraints anymore) while Step 3.a and Step 3.a.1 are replaced by

Step 3.a Let $\Delta_{i-1} := (D, W \cup Conseq(\{\delta_1, \ldots, \delta_{i-1}\}))$.
For all $D'' := credulous\_default\_proof(\Delta_{i-1}, \neg Conseq(\delta_i))$
do the following

Step.3.a.1 If $skeptically\_provable((D, W \cup Conseq(D'')), g) = no$ then return *no* and stop.

Thus, using these two modifications we obtain an algorithm to perform skeptical reasoning in Reiter's original approach, provided that attention is restricted to normal defaults. Again, this algorithm is based on a procedure *credulous_default_proof*(Δ, g) which is assumed to provide minimal default proofs of g from the normal default theory Δ. Note that our algorithm does not require a special procedure as regards this problem, so that any known implementation of credulous reasoning can be extended to skeptical reasoning in this way.

6 Conclusion

In the first part of this paper, we have developed a method which identifies credulous reasoning in Default Logics with planning problems. We have illustrated that finding a default proof can straightforwardly be modeled by finding a solution to a deductive planning problem — provided that the generation of extensions in the considered (fragment of) Default Logic is characterizable in a truly iterative fashion. Based on our formalization of this concept, we have discussed some interesting and valuable implications. For instance, our approach should provide a large number of new proof procedures for Default Logics by applying well-known methods designed for solving planning problems. As an example, we have investigated a method based on equational logic programming, which is suitable for a certain subclass of default theories. With this approach to deductive planning, we have presented a straightforward encoding of credulous reasoning in Constrained Default Logic. We have argued that treating defaults as actions modifying particular sets of beliefs (including constraints, or justifications as in Łukaszewicz' variant) captures the difference between a default as a rule and a formula. Furthermore, two particular complexity results known from the field of planning were used to define classes of default theories for which credulous entailment can be efficiently computed. From a more general point of view, we have bridged the gap between the field of planning and default reasoning. As a consequence, there is reason for hope that numerous other complexity results become transferable from the area of planning to that of Default Logic by means of our bridging results. Among other benefits, this should lead us to even more undiscovered tractable subclasses of Default Logic.

A comparison between our methodology and existing proof theories is difficult because there is principally a whole variety of different implementations of our approach depending on the chosen deductive planning method. Interpreting reasoning in Default Logics as deductive planning problems cannot be, for instance, said to be either goal-oriented like [51, 55], or bottom up like [4] because this depends of course on the specific planning system being applied. However, two main features are characteristic for our approach:

First of all, we follow Reiter's philosophy in so far as our method does not require to generate and test entire extensions. This distinguishes our approach from implementations like [20] where extensions are approximately computed, or the tableaux based formalisms [61, 62], or the methods described in [40, 41].

Second, both the consistency check as well as the groundedness requirement are directly associated with the application of a default. As regards the consistency requirement, this is reflected in most of the existing approaches, except Reiter's original proof theory where consistency is checked at the very end of the procedure, or the algorithm presented in [55] where consistency is guaranteed via computing a pre-compilation step. On the other hand, there are many different ways of testing groundedness. For instance, Reiter's method does not explicitly take groundedness into account which may lead, as argued in [55], to infinite loops. Checking groundedness separately at the end is performed in the tableaux-based approaches [61, 62]. [60] complies with our approach in describing an incremental verification of both groundedness and consistency.

In the second part of this paper, we have developed an approach to skeptical reasoning in our

exemplar, Constrained Default Logic. As illustrated at the end, the resulting algorithm is also applicable, in a simplified form, to classical Default Logic in the case of normal default theories. In general, this approach is applicable to any semi-monotonic (fragment of) Default Logic, since then queries are answerable in a local way. We have adopted an idea originally applied to the THEORIST system [47, 48, 67]. This method does not require the investigation of all extensions to decide membership in all extensions. Moreover, it is based on an arbitrary procedure providing credulous default proofs. Hence, any known algorithm which is either designed for credulous reasoning in Constrained Default Logic, or at least for classical Default Logic and normal default theories, can be extended to skeptical reasoning in this way.

Acknowledgments. The authors would like to thank Stefan Brüning and two referees for helpful comments on earlier versions of this paper. The second author was a visiting professor at the University of Darmstadt while parts of this work were being carried out. The second author also acknowledges support from the Commission of the European Communities under grant no. ERB4001GT922433.

References

[1] J. Allen, J. Hendler, and A. Tate, editors. *Readings in Planning.* Morgan Kaufmann, 1990.

[2] C. Bäckström and I. Klein. Planning in polynomial time: The SAS-PUBS class. *Computational Intelligence*, 7(3):181–197, 1991.

[3] C. Bäckström and B. Nebel. Complexity Results for SAS$^+$ Planning. In R. Bajcsy, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1430–1435, Chambéry, France, Aug. 1993. Morgan Kaufmann.

[4] P. Besnard, R. Quiniou, and P. Quinton. A Theorem-Prover for a Decidable Subset of Default Logic. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 27–30, 1983.

[5] W. Bibel. A Deductive Solution for Plan Generation. *New Generation Computing*, 4:115–132, 1986.

[6] W. Bibel. *Automated Theorem Proving.* Vieweg, second, revised edition 1987.

[7] S.-E. Bornscheuer and M. Thielscher. Representing Concurrent Actions and Solving Conflicts. In B. Nebel and L. Dreschler-Fischer, editors, *Proceedings of the German Annual Conference on Artificial Intelligence (KI)*, volume 861 of *LNAI*, pages 16–27, Saarbrücken, Germany, Sept. 1994. Springer.

[8] G. Brewka. Cumulative Default Logic. *Artificial Intelligence Journal*, 50(2):183–205, 1991.

[9] S. Brüning. Towards Efficient Calculi for Resource-Oriented Deductive Planning. In F. Pfenning, editor, *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR)*, volume 822 of *LNAI*, pages 174–188, Kiev, Ukraine, 1994. Springer.

[10] H.-J. Bürckert, A. Herold, D. Kapur, J. H. Siekmann, M. E. Stickel, M. Tepp, and H. Zhang. Opening the AC-Unification Race. *Journal of Automated Reasoning*, 4:465–474, 1988.

[11] T. Bylander. Complexity results for planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 274–279, Sydney, 1991.

[12] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence Journal*, 69:165–204, 1994.

[13] M. Cardoli and M. Schaerf. A Survey of Complexity Results for Non-Monotonic Logics. *Journal of Logic Programming*, 17:127–160, 1993.

[14] D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence Journal*, 32:333–377, 1987.

[15] K. L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Workshop Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

[16] J. P. Delgrande and W. K. Jackson. Default logic revisited. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 118–127, Cambridge, MA, 1991.

[17] J. P. Delgrande, T. Schaub, and W. K. Jackson. Alternative Approaches to Default Logic. *Artificial Intelligence Journal*, 70:167–237, 1994.

[18] Y. Dimopoulos. The Computational Value of Joint Consistency. In L. Pereira and D. Pearce, editors, *Proceedings of the European Workshop on Logics in AI (JELIA)*, volume 838 of *LNAI*, pages 50–65. Springer, 1994.

[19] H. B. Enderton, editor. *A Mathematical Introduction to Logic.* Academic Press, Orlando, 1972.

[20] D. W. Etherington. *Reasoning with Incomplete Information.* Research Notes in Artificial Intelligence. Pitman / Morgan Kaufmann, London, 1988.

[21] R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence Journal*, 2:189–208, 1971.

[22] J. H. Gallier and S. Raatz. Extending SLD-Resolution to Equational Horn Clauses Using E-Unification. *Journal of Logic Programming*, 6:3–44, 1989.

[23] J.-Y. Girard. Linear Logic. *Journal of Theoretical Computer Science*, 50(1):1–102, 1987.

[24] G. Gottlob. Complexity Results for Non-monotonic Logics. *Journal of Logic and Computation*, 2:397–425, 1992.

[25] C. Green. Application of theorem proving to problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 219–239, Los Altos, CA, 1969. Morgan Kaufmann.

[26] G. Große, S. Hölldobler, J. Schneeberger, U. Sigmund, and M. Thielscher. Equational Logic Programming, Actions, and Change. In K. Apt, editor, *Proceedings of the International Joint Conference and Symposium on Logic Programming (IJCSLP)*, pages 177–191, Washington, 1992. MIT Press.

[27] S. Hölldobler. *Foundations of Equational Logic Programming*, volume 353 of *LNAI.* Springer, 1989.

[28] S. Hölldobler and J. Schneeberger. A New Deductive Approach to Planning. *New Generation Computing*, 8:225–244, 1990.

[29] S. Hölldobler and M. Thielscher. Actions and Specificity. In D. Miller, editor, *Proceedings of the International Logic Programming Symposium (ILPS)*, pages 164–180, Vancouver, Oct. 1993. MIT Press.

[30] S. Hölldobler and M. Thielscher. Computing Change and Specificity with Equational Logic Programs. *Annals of Mathematics and Artificial Intelligence*, special issue on Processing of Declarative Knowledge, 1995.

[31] J. Jaffar, J.-L. Lassez, and M. J. Maher. A theory of complete logic programs with equality. *Journal of Logic Programming*, 1(3):211–223, 1984.

[32] D. Kapur and P. Narendran. NP-Completeness of the Set Unification and Matching-Problems. In J. H. Siekmann, editor, *Proceedings of the International Conference on Automated Deduction (CADE)*, volume 230 of *LNCS*, pages 489–495, Oxford, July 1986. Springer.

[33] H. Kautz and B. Selman. Hard Problems for Simple Default Logics. *Artificial Intelligence Journal*, 49:243–279, 1991.

[34] R. Kowalski. *Logic for Problem Solving*, volume 7 of *Artificial Intelligence Series*. Elsevier, 1979.

[35] R. Kowalski and M. Sergot. A logic based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[36] V. Lifschitz. On the Semantics of STRIPS. In M. P. Georgeff and A. L. Lansky, editors, *Proceedings of the Workshop on Reasoning about Actions & Plans*. Morgan Kaufmann, 1986.

[37] V. Lifschitz. On Open Defaults. In J. W. Lloyd, editor, *Computational Logic*, pages 80–95. Springer, Berlin, 1990.

[38] J. W. Lloyd. *Foundations of Logic Programming*. Series Symbolic Computation. Springer, second, extended edition, 1987.

[39] W. Łukaszewicz. Considerations on Default Logic — An Alternative Approach. *Computational Intelligence*, 4:1–16, 1988.

[40] W. Marek and M. Truszczyński. *Nonmonotonic Logics: Context-Dependent Reasoning*. Springer, 1993.

[41] W. Marek and M. Truszczyński. Normal form results for default logics. In G. Brewka, K. P. Jantke, and P. H. Schmitt, editors, *Nonmonotonic and Inductive Logic (NIL)*, pages 153–174. Springer, volume 659 of LNAI, 1993.

[42] M. Masseron, C. Tollu, and J. Vauzielles. Generating Plans in Linear Logic. In *Foundations of Software Technology and Theoretical Computer Science*, volume 472 of *LNCS*, pages 63–75. Springer, 1990.

[43] J. McCarthy. Situations and Actions and Causal Laws. Stanford Artificial Intelligence Project, Memo 2, 1963.

[44] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence Journal*, 28:89–116, 1986.

[45] J. McCarthy and P. J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1969.

[46] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw–Hill, New York, 1975.

[47] D. Poole. A Logical Framework for Default Reasoning. *Artificial Intelligence Journal*, 36(1):27–47, 1988.

[48] D. Poole. Explanation and prediction: an architecture for default and abductive reasoning. *Computational Intelligence*, 5:97–110, 1989.

[49] D. Poole. What the Lottery Paradox tells us about Default Reasoning. In R. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 333–340, Toronto, 1989. Morgan Kaufmann.

[50] D. Poole. Compiling a Default Reasoning System into Prolog. *New Generation Computing*, 9(1):3–38, 1991.

[51] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence Journal*, 13:81–132, 1980.

[52] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380. Academic Press, 1991.

[53] R. Reiter and G. Criscuolo. On interacting defaults. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1270–276, Vancouver, 1981.

[54] V. Risch. *Les Tableaux Analytiques au Service des Logiques de Defauts.* PhD thesis, Universite Aix-Marseille II, G.I.A., Parc Scientifique et Technologique de Luminy, Apr. 1993. (In French).

[55] A. Rothschild and T. Schaub. A computational approach to default logics. In G. Brewka and C. Witteveen, editors, *Dutch/German Workshop on Non-Monotonic Reasoning Techniques and Their Applications*, volume 2, pages 1–14, 1993.

[56] E. D. Sacerdoti. *A structure for plans and behavior.* Elsevier, 1977.

[57] T. Schaub. *Considerations on Default Logics.* PhD thesis, FG Intellektik, TH Darmstadt, Germany, Nov. 1992.

[58] T. Schaub. On Constrained Default Theories. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 1992.

[59] T. Schaub. Variations of constrained default logic. In M. Clarke, R. Kruse, and S. Moral, editors, *Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU)*, pages 312–317. Springer, 1993.

[60] T. Schaub. A new methodology for query-answering in default logics via structure-oriented theorem proving. Technical report, IRISA, Campus de Beaulieu, F–35042 Rennes Cedex, France, Jan. 1994.

[61] C. B. Schwind. A Tableau-Based Theorem Prover for a Decidable Subset of Default Logic. In M. E. Stickel, editor, *Proceedings of the International Conference on Automated Deduction (CADE)*, volume 449 of *LNAI*, Kaiserslautern, Germany, 1990. Springer.

[62] C. B. Schwind and V. Risch. A Tableau-Based Characterization for Default Logic. In R. Kruse, editor, *Proceedings of European Conference on Symbolic and Quantitative Approaches to Uncertainty*, pages 310–317. Springer, 1991.

[63] J. C. Shepherdson. SLDNF-Resolution with Equality. *Journal of Automated Reasoning*, 8:297–306, 1992.

[64] D. E. Smith, M. R. Genesereth, and M. L. Ginsberg. Controlling recursive inference. *Artificial Intelligence Journal*, 30:343–389, 1986.

[65] R. M. Smullyan. *First-Order Logic*. Springer, Berlin, Heidelberg, New York, 1971.

[66] J. Stillman. The Complexity of Propositional Default Logics. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 794–799, 1992.

[67] M. Thielscher. On prediction in Theorist. *Artificial Intelligence Journal*, 60(2):283–292, 1993.

[68] M. Thielscher. An Analysis of Systematic Approaches to Reasoning about Actions and Change. In P. Jorrand, editor, *International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA)*, Sofia, Bulgaria, Sept. 1994. World Scientific Publishing Co. Singapore.

[69] M. Thielscher. On the Completeness of SLDENF–Resolution. Technical Report AIDA–94–08, FG Intellektik, TH Darmstadt, Oct. 1994. (Submitted). Available by anonymous ftp from `130.83.26.1` in `/pub/AIDA/Tech-Reports/1994`.

[70] M. Thielscher. Representing Actions in Equational Logic Programming. In P. V. Hentenryck, editor, *Proceedings of the International Conference on Logic Programming (ICLP)*, pages 207–224, Santa Margherita Ligure, Italy, 1994. MIT Press.