Reasoning with Prioritized Defaults

Michael Gelfond and **Tran Cao Son**
Computer Science Department
University of Texas at El Paso
El Paso, Texas 79968
mgelfond,tson@cs.utep.edu

Abstract

The purpose of this paper is to investigate the methodology of reasoning with prioritized defaults in the language of logic programs under the answer set semantics. We present a domain independent system of axioms, written as an extended logic program, which defines reasoning with prioritized defaults. These axioms are used in conjunction with a description of a particular domain encoded in a simple language allowing representation of defaults and their priorities. Such domain descriptions are of course domain dependent and should be specified by the users. We give sufficient conditions for consistency of domain descriptions and illustrate the use of our system by formalizing various examples from the literature. Unlike many other approaches to formalizing reasoning with priorities ours does not require development of the new semantics of the language. Instead, the meaning of statements in the domain description is given by the system of (domain independent) axioms. We believe that in many cases this leads to simpler and more intuitive formalization of reasoning examples. We also present some discussion of differences between various formalizations.

1 Introduction

The purpose of this paper is to investigate the methodology of reasoning with prioritized defaults in the language of logic programs under the answer set semantics. Information about relative strengths of defaults can be commonly found in natural language descriptions of various domains. For instance, in legal reasoning it is often used to state preference of some laws over others, e.g., federal laws in the U.S. can, in some cases, override the laws of a particular state. Preferences are also used in reasoning with expert's knowledge where they are assigned in accordance with the degree of our confidence in different experts. Sometimes preferences in the natural language description of the domain are given implicitly, e.g., a conflict between two contradictory defaults can be resolved by selecting the one which is based on more specific information. All these examples suggest that it may be useful to consider knowledge representation languages capable of describing defaults and preferences between them. There is a sizeable body of literature devoted to design and investigation of such languages [1, 4, 5, 6, 10, 19, 26, 28, 29, 32]. The work is too diverse and our knowledge of it is not sufficient to allow a good classification but we will try to mention several important differences in approaches taken by the different authors. To shorten the discussion we limit our attention to approaches based on logic programming and default logics.

Many differences in design seem to be caused by ambiguity of the very notion of default. Sometimes defaults are understood as statements of natural language, of the form "Elements of a class C *normally* (regularly, as a rule) satisfy property P". Sometimes this understanding is broadened to include all statements with defeasible conclusions. The following example is meant to illustrate the difference.

Suppose we are given a list t of people and want to define the class of people not listed in t. This, of course, can be done by the rule

r1. $unlisted(X) \leftarrow not\ t(X)$.

The conclusion of this statement can be defeated by expanding the table t but cannot be defeated by adding a fact of the form $\neg unlisted(x)$ where $x \notin t$. The attempt to do the latter will (justifiably) lead to contradiction. The statement $r1$ is not a default according to the first, narrow view. It is rather a universally true statement which does not allow exceptions and can not be defeated by other (preferred) statements; of course, according to the second view, $r1$ is a default. Notice, that the statement "Table *unlisted* normally contains all the people not contained in t" is a default according to the both views. Its logic programming representation can have a form

r2. $unlisted(X) \leftarrow not\ t(X), not\ \neg unlisted(X)$.

This time the addition of $\neg unlisted(x)$ where $x \notin t$ cause no contradiction.

This (and similar) differences in understanding of defaults seems to sometimes determine the syntax of the corresponding "default" languages. The first view seems to lead to introducing special syntax for defaults while the second uses standard logic programming syntax augmented by the preference relation among the rules. According to the second view it seems to be also more natural to consider static preference relation, i.e., to prohibit occurrence of the preference relation in the rules of the program.

Even more important differences can be found on the semantic level. Let us accept a narrow view of defaults and consider the theory consisting of three defaults:

$d1$. "Normally a";

$d2$. "Normally b"

$d1$. "Normally c"

and three rules

$r1$. "b's are always $\neg a$'s";

$r2$. "b's are always d's";

$r3$. "a's are always d's";

There seems to be at least three equally reasonable ways to deal with this theory. We can assume that it is inconsistent and entail everything (or nothing); We can be cautious and refuse to apply defaults $d1$ and $d2$. In this case the only conclusion is c. We can be less cautious and reason by cases entailing d supported by two different arguments. With preference relation the situation will become even less clear since we will have an additional difficult question of defining what we mean by a conflict between defaults.

Different choices made by the authors of default languages are expressed in their semantics given by defining the entailment and/or the derivability relation for the language. The corresponding new logics can often be viewed as "prioritized" versions of the existing general purpose non-monotonic formalisms [1, 4, 5, 6, 28, 24] with new level of complexity added in fixpoint (or other) constructions defining the semantics. The viability of new logics is normally demonstrated by using it for formalization of some examples of default reasoning aimed to illustrate special features of the logic and the inadequacy of other formalisms. This process, even though useful and necessary, is often complicated by our collective lack of experience in representing knowledge about defaults and their preferences. It is often unclear for instance, if unintuitive answers to queries given by various formalisms can be blamed on the formalism itself or on the inadequate representation of the original problem. Moreover, it is often unclear what is the "common-sense", natural language description of the original problem of which the corresponding formal theory claims to be a representation. This, together with technical complexity of definitions, lack of the developed mathematical theories for new logics and the absence of clearly understood parameters which determine the choice of the semantics make their use for knowledge representation a rather difficult task.

This paper is the result of the authors attempts to understand some of the issues discussed above. We wanted to design a simple language, $\mathcal{L}$, capable of expressing and reasoning with prioritized defaults satisfying (among others) the following requirements:

• Understand defaults in a narrow sense as statements of the form a's are normally b's.

• Allow dynamic priorities, i.e., defaults and rules about the preference relation.

• Give semantics of $\mathcal{L}$ without developing new general purpose nonmonotonic formalism.

• Make sure that changes in informal parameters of the language such as properties of the preference relation, the definitions of conflicting defaults, cautiousness or bravery in reasoning are reflected by comparatively simple changes in the formalism.

• Make sure that some inference mechanism is available to reason with theories of $\mathcal{L}$ and some mathematical theory is available to prove properties of these theories.

We achieve these goals by mapping theories of $\mathcal{L}$ (also called domain descriptions) into a class of extended logic programs under the answer sets semantics [18]. This is done by presenting a logic program, $\mathcal{P}$ consisting of (domain independent) axioms defining the use of prioritized defaults; viewing domain descriptions of $\mathcal{L}$ as collections of atoms; and defining the notion of entailment between query q and a domain description $\mathcal{D}$ in $\mathcal{L}$ via answer set entailment in logic programming. In other words, we say that a domain description $\mathcal{D}$ entails a query q if q is entailed by the logic program $\mathcal{P} \cup \mathcal{D}$.

This approach appears to be similar in principle to the one suggested recently in [10] (which was not yet published when this work was completed). The resulting formalisms however are quite different technically. The precise relationship between the two is not yet fully investigated.

The use of the language will be illustrated by various examples from the literature. All the examples were run using the SLG inference engine [8, 9]. We believe that the study of the class of logic programs described by $\mathcal{P}$ and its variants can complement the existing work and help to understand reasoning with prioritized defaults.

The paper is organized as follows. In the next section, we introduce the language of prioritized defaults $\mathcal{L}_0$ and present a collection of axioms $\mathcal{P}$. Afterwards, we show examples of domain descriptions in , and discuss the use of the axioms in $\mathcal{P}$ in the Section 3. In the Section 4, we briefly discuss possible extensions of $\mathcal{D}_0$. We then define the class of hierarchical domain descriptions and discuss important properties of hierarchical domain descriptions in the Section 5. In the Section 6, we prove the equivalent of Brewka's approach to prioritized logic program and our approach.

2 The Language of Prioritized Defaults

We start with describing the class $\mathcal{L}_0(\sigma)$ of languages used for representing various domains of discourse. $\mathcal{L}_0(\sigma)$ is parameterized by a multi-sorted signature σ containing names for objects, functions and relations of the user's domain. We assume that σ contains two special collections of terms used to name defaults and strict (non-defeasible) rules of the language. We normally use terms formed by predicate symbols d_i and r_i to denote defaults and rules of $\mathcal{L}_0$ respectively. By $lit(\sigma)$ and $atom(\sigma)$ we denote the set of all (ground) literals and atoms of σ. Domain knowledge in $\mathcal{L}_0(\sigma)$ will be described by a collection of literals of σ (called σ-literals) together with statements describing strict rules, defaults, and preferences between defaults. The syntax of such descriptions is given by the following definitions:

Definition 2.1

- σ-literals are literals of $\mathcal{L}_0(\sigma)$;
- if d, d_1, d_2 are defaults names, $l_0, \ldots, l_n$ are literals of $\mathcal{L}_0(\sigma)$ and [ ] is the list operator of Prolog then

$$rule(r, l_0, [l_1, \ldots, l_m]); \tag{1}$$

$$default(d, l_0, [l_1, \ldots, l_m]); \tag{2}$$

$$conflict(d_1, d_2); \tag{3}$$

$$prefer(d_1, d_2); \tag{4}$$

  are literals of $\mathcal{L}_0(\sigma)$.

A set D of ground literals of $\mathcal{L}_0(\sigma)$ will be called *domain description* (with underlying signature σ).

Domain descriptions from our examples will contain non-ground literals, which will be viewed as shorthands for their ground instantiations. The relations $default$, $rule$, $conflict$ and $prefer$ will be called *domain independent*. Statements (1) and (2) will be called *definitions* of rule r and default d respectively. Intuitively, the statement (1) is a definition of the rule r which says that if literals $l_1, \ldots, l_m$ are true in a domain description $\mathcal{D}$ then so is the literal l_0. It can be viewed as a counterpart of the logic programming rule

$l_0 \leftarrow l_1, \ldots, l_m.$

Literals l_0 and $l_1, \ldots, l_m$ are called the head and the body of r and are denoted by $head(r)$ and $body(r)$ respectively.

The statement (2) is a definition of the default d which says that *normally*, if $l_1, \ldots, l_m$ are true in $\mathcal{D}$ then l_0 is true in $\mathcal{D}$. The logic programming counterpart of d is the rule

$l_0 \leftarrow l_1, \ldots, l_m, not\ \neg l_0.$

As before we refer to l_0 as the head of d ($head(d)$) and to $l_1, \ldots, l_m$ as its body ($body(d)$).

The statement (3) indicates that d_1 and d_2 are conflicting defaults. In many interesting cases $conflict(d_1, d_2)$ will be true iff heads of defaults d_1 and d_2 are contrary literals, but other defaults can also be declared as conflicting by the designer of the domain description. Finally, the statement (4) stops the application of default d_2 if defaults d_1 and d_2 are in conflict with each other and the default d_1 is applicable.

This informal explanation of the meaning of domain independent relations of $\mathcal{L}_0(\sigma)$ will be replaced by the precise definition in the next section. But first we will attempt to clarify this meaning with the following examples.

Example 2.1 Let us assume that we are given complete lists of students enrolled in various university departments, and that we know that in general, students can not write computer programs and that computer science students do it regularly. Let us represent this information by the domain description $\mathcal{D}_0$.

The underlying signature σ of $\mathcal{D}_0$ contains student names, *mary, mike, sam, ...*, department names *cs, cis, art, ...*, appropriately typed predicate symbols $is\_in(S, D)$ and $can\_progr(S)$ read as "Student S is in department D" and "Student S can program", and default names of the form $d_1(S)$, $d_2(S)$, and $d_3(S, D)$.

The defaults from our informal description can be represented by statements

$$default(d1(S),\ \neg can\_progr(S),\ [student(S)]).$$
$$default(d2(S),\ can\_progr(S),\ [student(S), is\_in(S, cs)]).$$

Finally, the lists of students mentioned in the informal description will be represented by the collection F of facts:

$student(mary).$	$dept(cs).$	$is\_in(mary, cs).$
$student(mike).$	$dept(art).$	$is\_in(mike, art).$
$student(sam).$	$dept(cis).$	$is\_in(sam, cis).$
$\ldots$	$\ldots$	$\ldots$

and the closed world assumption [30] for is_in, written as the default

$$default(d3(S,D), \neg is\_in(S,D), [\,]).$$

Relations *student* and *dept* are, of course, not necessary. They are playing the role of types and will later allow us to avoid floundering when applying the SLG inference engine to this example.

We will assume that our domain descriptions contain statements of the form $conflict(d_1, d_2)$ for any two defaults with contrary heads and that the relation *conflict* is symmetric. This will guarantee that $\mathcal{D}_0$ will contain $conflict(d_1(X), d_2(X))$ and $conflict(d_2(X), d_1(X))$. (The assumption will be of course enforced later by the corresponding axioms).

Informally, the domain description $\mathcal{D}_0$ should allow us to conclude that Mike and Sam do not know how to program, while we should remain undecided about programming skills of Mary. This is the case only as long as we do not assume that the second default overrides the first one, due to the specificity principle. We can use the relation $prefer$ from our language to record this preference by stating

$$prefer(d2(X), d1(X)).$$

From the new domain description $\mathcal{D}_1$ we should be able to conclude that Mary can write programs.

The next example is meant to illustrate the behavior of conflicting defaults in the presence of strict rules.

Example 2.2 Consider the domain description $\mathcal{D}_2$ consisting of two defaults

$default(d_1, p, [])$

$default(d_2, q, [r]),$

the rules

$rule(r_1, \neg p, [q])$

$rule(r_2, \neg q, [p])$

and the fact

$r.$

(Intuitively, the logic programming counterpart of $\mathcal{D}_2$ consists of the rules

$p \leftarrow not\ \neg p$
$q \leftarrow r, not\ \neg q$
$\neg p \leftarrow q$
$\neg q \leftarrow p$

Notice that the last two rules can be viewed as a translation into the logic programming language of the conditional q's are always not p's.)

The intended meaning of $\mathcal{D}_2$ should sanction two alternative sets of conclusions: one, containing p and $\neg q$, and another containing q and $\neg p$. If we expand $\mathcal{D}_2$ by

$conflict(d_2, d_1)$

$prefer(d_2, d_1)$

the application of d_1 should be blocked and the new domain description $\mathcal{D}_3$ should entail q and $\neg p$. Notice, that if $conflict(d_2, d_1)$ were not added to the domain description then addition of $prefer(d_2, d_1)$ would not alter the conclusions of $\mathcal{D}_2$. This is because preference only influences application of conflicting defaults.

More examples of the use of the language $\mathcal{L}_0$ for describing various domains will be found in the following sections. In the next section we give a precise definition of entailment from domain descriptions which captures the intended meaning of statements of $\mathcal{L}_0$.

2.1 Axioms of $\mathcal{P}$

In this section we present a collection $\mathcal{P}_\sigma$ of axioms defining the meaning of the domain independent relations of $\mathcal{L}_0(\sigma)$. The axioms are stated in the language of logic programs under the answer set semantics. They are intended to be used in conjunction with domain descriptions of $\mathcal{L}_0(\sigma)$ and to define the collection of statements which (strictly and/or defeasibly) follow from the given domain description $\mathcal{D}$. More precisely, we consider two basic relations $holds(l)$ and $holds\_by\_default(l)$ defined on literals of $\mathcal{L}_0(\sigma)$ which stand for "strictly holds" and "defeasibly holds", respectively. The query language associated with domain descriptions of $\mathcal{L}_0(\sigma)$ will consist of ground atoms of the form $holds\_by\_default(l)$, $holds(l)$, and their negations. In what follows, by $laws(\mathcal{D})$ we denote the set of statements of the forms (1) and (2) from definition 2.1 which belong to $\mathcal{D}$; $fact(\mathcal{D}) = \mathcal{D} \setminus laws(\mathcal{D})$.

Definition 2.2 We say that a domain description $\mathcal{D}$ entails a query q ($\mathcal{D} \models q$) if q belongs to every answer set of the program $\mathcal{P}_\sigma(\mathcal{D}) = \mathcal{P}_\sigma \cup \{holds(l) \mid l \in fact(\mathcal{D})\} \cup laws(\mathcal{D})$.

Program $\mathcal{P}$[1] consists of the following rules:

Non-defeasible Inference:

$$holds(L) \leftarrow rule(R, L, Body), \; hold(Body). \tag{1}$$

$$hold([\,]). \tag{2}$$

$$hold([H|T]) \leftarrow holds(H), \; hold(T). \tag{3}$$

[1] In what follows we assume that σ is fixed and omit reference to it whenever possible.

The first axiom defines the relation *holds* which is satisfied by a $\mathcal{L}_0(\sigma)$ literal l iff l is non-defeasibly true in the domain description $\mathcal{D}$. The next two axioms define similar relation on the lists of literals in $\mathcal{L}_0(\sigma)$, i.e., $hold([l_1, \ldots, l_n])$ iff all the l's from the list are true in $\mathcal{D}$.

Defeasible Inference:

$$holds\_by\_default(L) \leftarrow holds(L). \tag{4}$$

$$\begin{aligned} holds\_by\_default(L) \leftarrow\ & rule(R, L, Body), \\ & hold\_by\_default(Body). \end{aligned} \tag{5}$$

$$\begin{aligned} holds\_by\_default(L) \leftarrow\ & default(D, L, Body), \\ & hold\_by\_default(Body), \\ & not\ defeated(D), \\ & not\ holds\_by\_default(\neg L). \end{aligned} \tag{6}$$

$$hold\_by\_default([\,]). \tag{7}$$

$$\begin{aligned} hold\_by\_default([H|T]) \leftarrow\ & holds\_by\_default(H), \\ & hold\_by\_default(T). \end{aligned} \tag{8}$$

The first axiom in this group ensures that strictly true statements are also true by default. The next one allows application of rules for defeasible inference. The third axiom states that defaults with proven premises imply their conclusions unless they are defeated by other rules and defaults of the domain description. The condition $not\ holds\_by\_default(\neg L)$ is used when the domain contains two undefeated defaults d_1 and d_2 with conflicting conclusions. In this case $\mathcal{P}(\mathcal{D})$ will have multiple answer sets, one containing the conclusion of d_1 and the other containing the conclusion of d_2. The alternative solution here is to stop applications of both defaults, but we believe that in some circumstances (like those described by the extended *"Nixon Diamond"*) our solution is preferable.

The last two rules from this group define relation $hold\_by\_default(List)$ which holds if all literals from the list hold by default.

Defeating defaults:

$$defeated(D) \leftarrow default(D, L, Body), \quad holds(\neg L). \tag{9}$$

$$\begin{aligned} defeated(D) \leftarrow\ & default(D, L, Body), \\ & default(D_1, L_1, Body_1), \\ & holds(conflict(D_1, D)), \\ & holds\_by\_default(prefer(D1, D)), \\ & hold\_by\_default(Body_1), \\ & not\ defeated(D_1). \end{aligned} \tag{10}$$

These axioms describe two possible ways to defeat a default d. The first axiom describes a stronger type of defeat when the conclusion of the default is proven to be false by non-defeasible means. The axiom (10) allows defeating of d by conflicting undefeated defaults of higher priority. They represents the "bravery" approach in the application of defaults. In the next section, we show how our axioms can be expanded or changed to allow other ways of defeating defaults.

Now we are left with the task of defining conflicts between defaults. There are several interesting ways to define this notion. Different definitions will lead to different default theories. This is, however, beyond the limits of this paper and therefore will not be discussed further here. The following three axioms constitute the minimal requirement for this relation.

$$holds(conflict(d_1, d_2)) \tag{11}$$

for any two defaults with contrary literals in their heads and for any two defaults whose heads are of the form $prefer(d_i, d_j)$ and $prefer(d_j, d_i)$ respectively.

$$\neg holds(conflict(D, D)) \tag{12}$$

$$holds(conflict(D_1, D_2)) \leftarrow holds(conflict(D_2, D_1)) \tag{13}$$

Finally, we include axioms stating asymmetry of the preference relation:

$$\neg holds(prefer(D_1, D_2)) \leftarrow holds(prefer(D_2, D_1)) \quad D_1 \neq D_2 \tag{14}$$

$$\neg holds\_by\_default(prefer(D_1, D_2)) \leftarrow holds\_by\_default(prefer(D_2, D_1)) \quad D_1 \neq D_2 \tag{15}$$

Without the loss of generality we can view these axioms as schemas where D_1 and D_2 stand for defaults present in $\mathcal{D}$. Notice, that our minimal requirements on the preference relation does not include transitivity. On the discussion of nontransitive preference relations see [15], [21].

Uniqueness of names for defaults and rules:

These three axioms guarantee uniqness of names for defaults and rules used in the domain description.

$$\neg rule(R, F_1, B_1) \leftarrow default(R, F_2, B_2) \tag{16}$$

$$\neg rule(R, F_1, B_1) \leftarrow rule(R, F_2, B_2), \quad rule(R, F_1, B_1) \neq rule(R, F_2, B_2) \tag{17}$$

$$\neg default(D, F_1, B_1) \leftarrow default(D, F_2, B_2), \quad default(D, F_1, B_1) \neq default(D, F_2, B_2) \tag{18}$$

Addition of these axioms is needed only to make domain descriptions containing defaults of the form $default(d, l_1, \Gamma_1)$ and $default(d, l_2, \Gamma_2)$, etc, inconsistent.

Auxiliary

Finally we have the axioms

$$\neg holds(L) \leftarrow holds(\neg L). \tag{19}$$

$$\neg holds\_by\_default(L) \leftarrow holds\_by\_default(\neg L). \tag{20}$$

which meaning is self-explanatory.

We believe that $\mathcal{P}(\mathcal{D})$ captures a substantial part of our intuition about reasoning with prioritized defaults and therefore deserves some study.

3 Using the Axioms

In this section we illustrate the use of our approach by formalizing several examples of reasoning with priorities. In what follows we will refer to running our programs using SLG inference engine. Since the syntax of SLG does not allow "$\neg$" we treat it as a new function symbol and consider only those stable models of $\mathcal{P}(\mathcal{D})$ which do not contain literals of the form a and $neg(a)$.

Example 3.1 *(Example 2.1 revisited)* It is easy to check that the program $\mathcal{P}(\mathcal{D}_0)$ (where $\mathcal{D}_0$ is the domain description from Example 2.1) has two answer sets, containing

$$\{\neg hd(can\_progr(mary)), \neg hd(can\_progr(mike)), \neg hd(can\_progr(sam))\}$$

and

$$\{hd(can\_progr(mary)), \neg hd(can\_progr(mike)), \neg hd(can\_progr(sam))\},$$

respectively, where hd is a shorthand for $holds\_by\_default$. Hence, we can conclude that Mike and Sam do not know how to program but we have to stay undecided on the same question about Mary.

If we expand the domain by adding the statement $prefer(d_2, d_1)$ to it then the first answer set will disappear which of course corresponds exactly to our intention. It may be instructive to expand our domain by the following information: "Bad students never know how to program. Bob is a bad computer science student". This can be represented by facts

$$student(bob).$$
$$bad(bob).$$
$$is\_in(bob, cs).$$

and the rule

$$rule(r_2(S),\ \neg can\_progr(S),\ [student(S), bad(S)]).$$

The new domain description $\mathcal{D}_4$ will correctly entail that Bob does not know how to program. Notice, that if the above rule were changed to the default

$$default(d_3(S),\ \neg can\_progr(S),\ [student(S), bad(S)])$$

we would again get two answer sets with contradictory conclusions about Bob, and that again the conflict could be resolved by adding, say,

$$prefer(d_3(S), d_2(S)).$$

□

The previous example had an introductory character and could have been nicely formalized without using the preference relation. The next example (from [4], which attributes it to [20]) is more sophisticated: Not only does it require the ability to apply preferences to resolve conflicts between defaults, but also the ability of using defaults to *reason about such preferences.* Brewka in [4] argues that the ability to reason about preferences between defaults in the same language in which defaults are stated is important for various applications. In legal reasoning similar arguments were made by Gordon, Prakken, and Sartor [20, 28]. On the other hand, many formalisms developed for reasoning with prioritized defaults treat preferences as something statically given and specified separately from the corresponding default theory.

Example 3.2 *(Legal Reasoning [4])* Assume that a person wants to find out if her security interest in a certain ship is perfected. She currently has possession of the ship. According to the Uniform Commercial Code a security interest in goods may be perfected by taking possession of the collateral. However, there is a federal law called Ship Mortgage Act (SMA) according to which a security interest in a ship may only be perfected by filing a financing statement. Such a statement has not been filed. Now, the question is whether the UCC or the SMA takes precedence in this case. There are two known legal principles for resolving conflicts of this kind. The principle of *Lex Posterior* gives preference to newer law. In our case the UCC is newer than the SMA. On the other hand, the principle of *Lex Superior* gives precedence to laws supported by the higher authority. In our case the SMA has higher authority since it is federal law.

Let us build the domain description $\mathcal{D}_5$ which represents the above information. We will follow the formalization from [4] which uses symbols $possession$ for "ship is a possession of the lady from the above story", $perfected$ for "the ownership of the ship is perfected", and $filed$ for "financial statement about possession of the ship is filed". The domain also contains symbols $state(D), federal(D)$, and $more\_recent(D_1, D_2)$ representing properties and relations between legal laws.

The UCC and SMA defaults of $\mathcal{D}_5$ can be represented by

$$default(d_1, perfected, [possession]).$$
$$default(d_2, \neg perfected, [\neg filed]).$$

The two legal principles for resolving conflicts are represented by the next two defaults:

$$default(d_3(D_1, D_2), prefer(D_1, D_2), [more\_recent(D_1, D_2)]).$$
$$default(d_4(D_1, D_2), prefer(D_1, D_2), [federal(D_1), state(D_2)]).$$

The next defaults will express the closed world assumptions for relations $more\_recent, federal$ and $state$. Presumably, a reasoning legal agent must have compete knowledge about the laws. The following defaults are added to $\mathcal{D}_5$ to represent this CWA assumption.

$$default(d_5(D_1, D_2), \neg more\_recent(D_1, D_2), [\,]).$$
$$default(d_6(D), \neg federal(D), [\,]).$$
$$default(d_7(D), \neg state(D), [\,]$$

To complete our formalization we need the following facts:

$$\neg filed.$$
$$possession.$$
$$more\_recent(d_1, d_2).$$
$$federal(d_2).$$
$$state(d_1).$$

It is not difficult to check (using SLG if necessary) that the program $\mathcal{P}(\mathcal{D}_5)$ has two answer sets where

(i) $holds\_by\_default(perfected)$

belongs to one answer set and

(ii) $\neg holds\_by\_default(perfected)$

belongs to the other. This is because we have two defaults d_1 and d_2: the former supports the first conclusion, the latter - the second one, and preference between them cannot be resolved using defaults d_3 and d_4. Thus, neither (i) nor (ii) is entailed by $\mathcal{P}(\mathcal{D}_5)$. This is also Brewka's result in [4].

However, if we know that d_4 has a preference over d_3 the situation changes; To see that, let us expand our domain description by

$$prefer(d4(D_1, D_2), d3(D_2, D_1)).$$

and denote the new domain description by $\mathcal{D}_6$; as a result, program $\mathcal{P}(\mathcal{D}_6)$ has then only one answer set, which contains (ii). This is again the desired behavior, according to [4]. It may be worth noticing that the closed world assumptions d_5, d_6 and d_7 have no role in the above arguments and could be removed from the domain description. They are important, however, for its general correctness. The example can be substantially expanded by introducing more realistic representation of the story and by using more complex strategies of assigning preferences to conflicting defaults. We found that the corresponding domain descriptions remain natural and correct. □

Example 3.3 *(Simple Inheritance Hierarchy)* Now let us consider a simple inheritance hierarchy of the form depicted in fig (1).

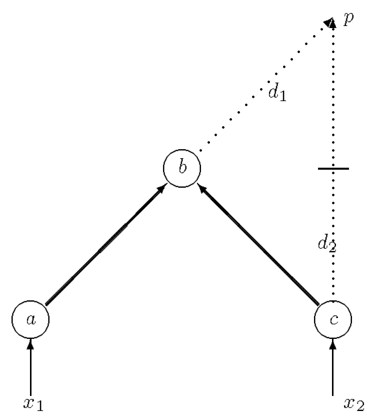

Figure 1. The Inheritance Hierarchy of $\mathcal{D}_7$

A simple hierarchy consists of two parts: an acyclic graph representing the proper subclass relation between classes of objects and a collection of positive and negative defaults from these subclasses to properties of objects. In fig (1) we have three class nodes, a, b, and c. The strict link between the class nodes, say, a and b can be read as "a is a proper subclass of b".

Dotted lines from b and c to property p represent positive and negative defaults respectively. The simple hierarchy is used in conjunction with a collection of statements $is\_in(x, c)$ read as "x is an elements of a class c". For simplicity we assume completeness of information about relations $subclass$ and is_in. (For discussion of hierarchies with incomplete information, see [17]).

The encoding of simple hierarchies will consists of two parts: the first representing a particular graph and the second containing general properties of a hierarchy together with the inheritance principle. Notice, that the second part is common to all simple hierarchies.

In our case, the domain description $\mathcal{D}_7$ encoding the hierarchy from fig (1) consists of domain dependent axioms

$$
\begin{array}{l}
subclass(a, b). \\
subclass(c, b). \\
is\_in(x_1, a) \\
is\_in(x_2, c) \\
default(d_1(X), has(X, p), [is\_in(X, b)]) \\
default(d_2(X), \neg has(X, p), [is\_in(X, c)])
\end{array}
$$

(where $has(X, P)$ stands for "element X has a property P") and the domain independent axioms

$$
\begin{array}{ll}
rule(r_1(C_0, C_2), subclass(C_0, C_2), [subclass(C_0, C_1), subclass(C_1, C_2)]). & \\
rule(r_2(X, C_1), is\_in(X, C_1), [subclass(C_0, C_1), is\_in(X, C_0)]). & \\
rule(r_3(D_1(X), D_2(X)), prefer(D_1(X), D_2(X)), & [d(D_1(X), \_, [is(X, A)]), \\
 & d(D_2(X), \_, [is(X, B)]), \\
 & subclass(A, B)]). \\
default(d_3(X), \neg is\_in(X), []). & \\
default(d_4, \neg subclass(A, B), []). &
\end{array}
$$

(where d stands for $default$ and _ is used where names are not important). The first two rules represent general properties of $subclass$ and is_in. The next rule is an encoding of the inheritance principle. The last two defaults express the closed world assumptions for simple hierarchies.

It is easy to check that $\mathcal{D}_7$ is consistent and that the logic program $\mathcal{P}(\mathcal{D}_7)$ has the unique answer set containing $holds\_by\_default(has(x_1, p))$ and $holds\_by\_default(\neg has(x_2, p))$. Consistency result can be easily expanded to rule-consistent domains representing simple hierarchies.

We use the next example from Brewka [6] to illustrate differences between our theory and several other formalisms dealing with prioritized defaults.

Example 3.4 *(Gray Area)* Brewka considers the following defaults:

1. "Penguins normally do not fly;",
2. "Birds normally fly;", and

3. “Birds that can swim are normally penguins;”,

under the assumption that default (1) is preferred over (2), and (2) is preferred over (3). (Notice, that Brewka assumes transitivity of the preference relation).

These defaults are represented in his formalism by a program

$bird.$
$swims.$
$(d_1)\ \neg flies \leftarrow not\ flies, penguin.$
$(d_2)\ flies \leftarrow not\ \neg flies, bird.$
$(d_3)\ penguin \leftarrow not\ \neg penguin, bird, swims.$

According to Brewka, the prioritized default theories from [1, 4, 24] are applicable to this case and produce single extension $E_1 = \{swims,\ bird,\ flies,\ penguin\}$ which seems contrary to intuition. According to the semantics from [6] the corresponding program has one prioritized answer set, $E_2 = \{swims, bird, penguin, \neg flies\}$ which is a more intuitive result. The information above is naturally encoded in the domain description $\mathcal{D}_8$ by the following statements

$bird.$
$swims.$

$default(d_1, \neg flies, [penguin]).$
$default(d_2, flies, [bird]).$
$default(d_3, penguin, [bird, swims]).$

$prefer(d_1, d_2).$
$prefer(d_2, d_3).$
$prefer(d_1, d_3).$

The program $\mathcal{P}(\mathcal{D}_8)$ has only one answer set which contains
$S_1 = \{holds\_by\_default(bird), holds\_by\_default(swim),$
$holds\_by\_default(penguin), \neg holds\_by\_default(flies)\}.$
which coincides with the approach from [6]. This happens because the default d_3 is in conflict with neither d_1 nor d_2 and therefore its application is not influenced by the preference relation. If we expand the domain description $\mathcal{D}_8$ by a statement

$conflict(d_2, d_3)$

the situation changes. Now we will have the second answer set,

$$S_2 = \{holds\_by\_default(bird), holds\_by\_default(swim), holds\_by\_default(flies)\}.$$

which corresponds to the following line of reasoning: We are initially confronted with “ready to fire” defaults (d_2) and (d_3). Since (d_2) has a higher priority and d_2 and d_3 are conflicting defaults, d_2 wins and we conclude $flies$. Now, (d_1) is not applicable and hence we stop.

To obtain S_1, we can apply defaults (d_1) and (d_3). Since (d_2) is then defeated by (d_1) it will not block (d_3).

We realize of course that this example belongs to the *gray area* and can be viewed differently. The main lesson from this observation is that in the process of expressing ourself (while programming or otherwise) we should try to avoid making unclear statements. Of course, we hope that further work on semantics will help to clarify some statements which so far remain unclear. We also hope that the reader is not left with the impression that we claim success in following our own advice.

4 Extending $\mathcal{L}_0(\sigma)$

In this section we briefly outline and discuss several extensions of the language $\mathcal{L}_0(\sigma)$. We show how to extend the language and the corresponding collection of axioms to allow the representation of more powerful defaults and default defeaters.

4.1 Beyond normal defaults

So far, our language $\mathcal{L}_0$ allowed only the representation of normal defaults [31]. In this section we expand the language and the corresponding system of axioms to make it capable of representing more general types of defaults. To this end we replace the definition of default description in $\mathcal{L}_0$ (see 2 in the Definition 2.1) by the more powerful construct

$$default(d, l_0, [l_1, \ldots, l_m], [l_{m+1}, \ldots, l_n]) \tag{2'}$$

The intuitive meaning of this statement is that *normally*, if $l_1, \ldots, l_m$ are true in $\mathcal{D}$ and there is no reason to believe that $l_{m+1}, \ldots, l_n$ are true in $\mathcal{D}$ then l_0 is true in $\mathcal{D}$. In other words, the statement (2') corresponds to the logic programming rule

$$l_0 \leftarrow l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n, not\ \neg l_0.$$

Literals $l_1, \ldots, l_m$ and $l_{m+1}, \ldots, l_n$ are called positive and negative preconditions of d respectively. Both sets of preconditions will be sometimes referred to as the body of statement (2').

Our set of axioms $\mathcal{P}$ will be modified as follows: axioms (6) and (10) will be replaced by axioms

$$\begin{aligned} holds\_by\_default(L) \leftarrow\ & holds(default(D, L, Positive, Negative)), \\ & hold\_by\_default(Positive), \\ & fail\_by\_default(Negative), \\ & not\ defeated(D), \\ & not\ \neg holds\_by\_default(L). \end{aligned} \tag{6'}$$

$$\begin{aligned} defeated(D) \leftarrow\ & holds(default(D1, L, Positive, Negative)), \\ & holds\_by\_default(prefer(D1, D)), \\ & hold\_by\_default(Positive), \\ & fail\_by\_default(Negative), \\ & not\ defeated(D_1). \end{aligned} \tag{10'}$$

where $fail\_by\_default$ is defined as follows:

$$fail\_by\_default([\,]). \tag{21}$$

$$\begin{array}{rcl} fail\_by\_default([H|T]) & \leftarrow & not\ holds\_by\_default(H), \\ & & fail\_by\_default(T). \end{array} \tag{22}$$

We hope that the modification is self-explanatory.

The following example, taken from [28], illustrates the use of the new language.

Example 4.1 *[29]* Consider the following two legal default rules from [29]:

1. Normally, a person who cannot be shown to be a minor has the capacity to perform legal acts.

2. In order to exercise the right to vote the person has to demonstrate that he is not a minor.

The first default can be represented as

$default(d_1(x), has\_legal\_capacity(x), [\,], [minor(x)])$

which requires a negative precondition. The second default has the form

$default(d_2(x), has\_right\_to\_vote(x), [\neg minor(x)], [\,])$.

These defaults, used in conjunction with statement

$\neg minor(jim)$,

entail that Jim has legal capacity and can vote. If the system is asked the same questions about Mary whose legal age is not known it will conclude that Mary has legal capacity but will remain in the dark about Mary's right to vote. If we expand our domain description by the closed world assumption for $has\_right\_to\_vote$

$default(d_3(x), has\_right\_to\_vote(x), [\,], [\,])$

then the answer to the last question will be *no*.

4.2 Weak Exceptions to Defaults

So far our language allowed only strong exceptions to defaults, i.e., a default d could be defeated by rules and by defaults conflicting with d. Many authors argued for a need for so called weak exceptions - statements of the form "do not apply default d to objects satisfying property p". (For the discussion of the difference between weak and strong exceptions see, for instance, [2].) Weak exceptions of this type can be easily incorporated in our language. First we expand the language by allowing literals of the form

$$exception(d, [l_1, \ldots, l_n], [l_{n+1}, \ldots, l_{n+m}]) \tag{23}$$

read as "default d is not applicable to an object x which satisfies $l_1, \ldots, l_n$ and $not\ l_{n+1}, \ldots, not\ l_{n+m}$". The formal meaning of this statement is defined by an axiom

$$\begin{aligned} defeated(D) \ \leftarrow \ & exception(D, Positive, Negative), \\ & hold\_by\_default(Positive), \\ & fail\_by\_default(Negative). \end{aligned} \tag{24}$$

added to $\mathcal{P}$.

Consider a domain description $\mathcal{D}_9$.

$default(d(X), p(X), [q(X)], [\,]).$
$exception(d(X), [r(X)], [\,]).$
$q(x_1).$
$q(x_2).$
$r(x_2).$

It is easy to check, that the corresponding program $\mathcal{P}(\mathcal{D}_9)$ (and hence $\mathcal{D}_9$) entails $p(x_1)$ but remains undecided about $p(x_2)$. Notice, that we were able to entail $p(x_1)$ even though x_1 may satisfy property r, i.e. $\mathcal{D}_9 \not\models \neg r(x_1)$. In some cases we need to be able to say something like "do not apply d to x if x may satisfy property r". This can be achieved by replacing the exception clause in $\mathcal{D}_9$ by

$exception(d(X), [\,], [\neg r(X)]).$

The new domain description entails neither $p(x_1)$ nor $p(x_2)$.

4.3 Changing the mode of reasoning

In our theory $\mathcal{P}$ we formalized a " brave" mode of applying defaults. In this section we briefly mention how the axioms can be changed to allow for cautious reasoning. This, of course, can be achieved by adding to $\mathcal{P}$ the axiom

$$\begin{aligned} defeated(D) \ \leftarrow \ & default(D, L, Body), \\ & default(D_1, L_1, Body_1), \\ & holds(conflict(D_1, D)), \\ & not\ holds\_by\_default(prefer(D1, D)), \\ & not\ holds\_by\_default(prefer(D, D1)), \\ & hold\_by\_default(Body), \\ & hold\_by\_default(Body_1) \end{aligned} \tag{25}$$

Let us denote the resulting program by $\mathcal{P}_c$. Now let us consider the domain description $\mathcal{D}_{10}$ consisting of defaults and conditionals mentioned in the introduction

$default(d_1, a, [\,])$.
$default(d_2, b, [\,])$.
$default(d_3, c, [\,])$.
$conflict(d_1, d_2)$.

$rule(r_1, \neg a, [b])$. $\quad$ $rule(r_1', \neg b, [a])$.
$rule(r_2, d, [b])$. $\quad$ $rule(r_2', \neg b, [\neg d])$.
$rule(r_3, d, [a])$. $\quad$ $rule(r_3', \neg a, [\neg d])$.

It is easy to check that $\mathcal{P}(\mathcal{D}_{10})$ has two answer sets containing $\{c, a, d, \neg b\}$ and $\{c, b, d, \neg a\}$ and therefore entails d and c. In contrast $\mathcal{P}_c(\mathcal{D}_{10})$ has one answer set not containing d.

It is worth to mention that it may be possible in this framework to introduce two types of defaults - those requiring brave and cautious reasoning and add the above axiom for the latter.

5 Hierarchical Domain Descriptions

Definition 5.1 We will say that a domain description $\mathcal{D}$ is *consistent* if $\mathcal{P}(\mathcal{D})$ is consistent, i.e., has a consistent answer set.

Obviously, not all domain descriptions are consistent. $\mathcal{D} = \{p, \neg p, q\}$, for instance, is not.

(Notice that is the intended meaning. We believe that the question of drawing conclusions in the presence of inconsistency is somewhat orthogonal to the problem we address in this paper and should be studied separately.)

In the next example inconsistency is slightly less obvious.

Example 5.1 The domain description $\mathcal{D}_{11}$ consists of the three literals:

$default(d, a, [\,])$.
$rule(r_1, \neg c, [a])$.
c.

It is easy to see that $\mathcal{P}(\mathcal{D}_{11})$ does not have a consistent answer set. Notice, that addition of the rule

$rule(r_2, \neg a, [c])$.

will restore consistency. $\square$

In this section we give a simple condition guaranteeing consistency of domain descriptions of $\mathcal{L}_0$. The condition can be expanded to domain descriptions of $\mathcal{L}$ but we will not do it here.

We will need the following definitions.

Definition 5.2 The domain description $\mathcal{D}$ is said to be *rule-consistent* if the non-defeasible part of $\mathcal{P}(\mathcal{D})$ has a consistent answer set. (By the non-defeasible part of $\mathcal{P}(\mathcal{D})$ we mean the program $\mathcal{P}_s(\mathcal{D})$ consisting of the set $\{holds(l) \mid l \in fact(\mathcal{D})\} \cup laws(\mathcal{D})$ and nondefeasible rules (rules 1-3, 9, 12-14, and 16-19 or $\mathcal{P}(\mathcal{D})$).

Definition 5.3 A domain description $\mathcal{D}$ over signature σ will be called *hierarchical* if it satisfies the following conditions:

1. $\mathcal{D}$ is rule-consistent;
2. for any $d_1, d_2 \in \mathcal{D}$, $conflict(d_1, d_2) \in \mathcal{D}$ iff heads of d_1 and d_2 are contrary literals of σ or have the form $prefer(d_i, d_j)$ and $prefer(d_j, d_i)$ where $i \neq j$;
3. heads of defaults in $\mathcal{D}$ are σ-literals or literals of the form $prefer(d_1, d_2)$;
4. no literal from the head of a default in $\mathcal{D}$ belongs to the body of a rule in $\mathcal{D}$;
5. there is a function *rank* from the set $heads(\mathcal{D})$ of literals belonging to the heads of defaults in $\mathcal{D}$ to the set of ordinals such that
   (a) $rank(l) = rank(\neg l)$;
   (b) $rank(prefer(d_1, d_2)) = rank(prefer(d_2, d_1))$
   (c) if $default(d, l, [l_1, \ldots, l_n]) \in \mathcal{D}$ and $l_i \in heads(\mathcal{D})$ then $rank(l) > rank(l_i)$;
   (d) if $prefer(d_1, d_2) \in heads(\mathcal{D})$ and $d_1, d_2 \in \mathcal{D}$ then $rank(head(d_i)) > rank(prefer(d_1, d_2))$ for $i = 1, 2$;

It is easy to check that domain descriptions $\mathcal{D}_0$, $\mathcal{D}_1$, $\mathcal{D}_4$, and $\mathcal{D}_6$ are hierarchical while $\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_7$ are not. In $\mathcal{D}_2$ and $\mathcal{D}_7$, the condition (4) is violated while (2) is not true in $\mathcal{D}_3$. Domain description $\mathcal{D}_5$ is also hierarchical. The rank function for $\mathcal{D}_5$ can be given by $rank(l) = 1$ for $l \notin \{perfected, \neg perfected\}$, $rank(perfected) = rank(\neg perfected) = 4$, and $rank(prefer(d1(X), d2(X))) = rank(prefer(d2(X), d1(X))) = 2$.

Theorem 5.1 Hierarchical domain descriptions are consistent.

To prove the theorem we need the following lemmas.

Lemma 5.1 [2]Let $\mathcal{P}$ be a logic program and

$q \leftarrow \Gamma_1$
$q \leftarrow \Gamma_2$
$\ldots$

be the collection of all rules of $\mathcal{P}$ with the head q. Then the program $\mathcal{Q}$ obtained from $\mathcal{P}$ by replacing rules of the form

$p \leftarrow \Delta_1, q, \Delta_2$

by the set of rules

$p \leftarrow \Delta_1, \Gamma_1, \Delta_2$
$p \leftarrow \Delta_1, \Gamma_2, \Delta_2$
$\ldots$

is equivalent to $\mathcal{P}$, i.e., $\mathcal{P}$ and $\mathcal{Q}$ have the same consistent answer sets.

[2]This is a well-know property of logic programs called "partial evaluation" in [3]. We were, however, unable to find a proof of it for an infinite P.

Proof. Let us denote the set of all rules removed from $\mathcal{P}$ by $\mathcal{S}$ and let

$\mathcal{R} = \mathcal{Q} \cup \mathcal{S}$.

$\mathcal{R}$ can be viewed as a union of $\mathcal{P}$ and the set of new rules obtain from $\mathcal{P}$ by the application of the cut inference rule. Since the cut is sound w.r.t. constructive logic N_2 [27] which is an extension of the logic N from [25], $\mathcal{P}$ and $\mathcal{R}$ are equivalent in N_2. As shown in [27], programs equivalent in N_2 have the same consistent answer sets, i.e.,

(a) programs $\mathcal{P}$ and $\mathcal{R}$ are equivalent.

This means that to prove our lemma it suffices to show equivalence of $\mathcal{R}$ and $\mathcal{Q}$.

Let $\mathcal{Q}^A$ and $\mathcal{S}^A$ be reducts of $\mathcal{Q}$ and $\mathcal{S}$ w.r.t. set A of literals (as in the definition of answer sets). We show that A is the minimal set closed under $\mathcal{Q}^A$ iff it is the minimal set closed under $\mathcal{Q}^A \cup \mathcal{S}^A$.

(b) Let A be the minimal set closed under $\mathcal{Q}^A$. We show that it is closed under $\mathcal{S}^A$.

Consider a rule

$p \leftarrow \Delta_1^A, q, \Delta_2^A \ \in S^A$

s.t. $\{\Delta_1^A, q, \Delta_2^A\} \subseteq A$. (Here by Δ_i^A we denote the result of removing from Δ_i all the occurrences of *not* l s.t. $l \notin A$. Obviously, Δ_i^A's above do not contain *not* .) From the assumption of (b) and the fact that $q \in A$ we have that there is i s.t. a rule

$q \leftarrow \Gamma_i^A \ \in \mathcal{Q}^A$

with $\Gamma_i^A \subseteq A$. This implies that there is a rule

$p \leftarrow \Delta_1^A, \Gamma_i^A, \Delta_2^A \ \in \mathcal{Q}^A$

whose body is satisfied by A, and therefore $p \in A$. This implies that A is the minimal set closed under $\mathcal{Q}^A \cup \mathcal{S}^A$.

(c) Let A be the minimal set closed under $\mathcal{Q}^A \cup \mathcal{S}^A$. We will show that it is the minimal set closed under $\mathcal{Q}^A$.

A is obviously closed under $\mathcal{Q}^A$. Suppose that there is $B \subset A$ closed under $\mathcal{Q}^A$. As was shown above it would be also closed under $\mathcal{S}^A$ which contradicts our assumption.

From (b), (c) and the definition of answer set we have that $\mathcal{R}$ and $\mathcal{Q}$ are equivalent, which, together with (a), proves the lemma. $\Box$

To formulate the next lemma we need the following notation: Let $\mathcal{P}$ be a ground logic program not containing negative literals $\neg l$ and let p be a unary predicate symbol from the language of $\mathcal{P}$. By $\mathcal{P}^*$ we denote the result of replacing all occurrence of atoms of the form $p(t)$ in $\mathcal{P}$ by t. Notice, that $\mathcal{P}^*$ can be viewed as a propositional logic program with different terms viewed as different propositional letters.

Lemma 5.2 Let $\mathcal{P}$ be a logic program not containing negative literals $\neg l$, and let p be a unary predicate symbol from the language of $\mathcal{P}$. Then A is an answer set of $\mathcal{P}$ iff A^* is an answer set of $\mathcal{P}^*$.

Proof. If $\mathcal{P}$ does not contain *not* the lemma is obvious. (We assume of course that terms of the language of $\mathcal{P}$ cannot be used as atoms in this language). Otherwise, notice that by definition of answer set, A is an answer set of $\mathcal{P}$ iff it is an answer set of $\mathcal{P}^A$. Since $\mathcal{P}^A$ does not contain *not* this happens iff A^* is the answer set of $(\mathcal{P}^A)^*$. To complete the proof it remains to notice that $(\mathcal{P}^A)^* = (\mathcal{P}^*)^{A^*}$. □

Lemma 5.3 Let $\mathcal{D}$ be a domain description. By $\mathcal{P}_1(\mathcal{D})$ we denote the program obtained from $\mathcal{P}(\mathcal{D})$ by

- replacing all occurrences of literals $hold([l_1, \ldots, l_n])$ and $hold\_by\_default([l_1, \ldots, l_n])$ in the bodies of the rules from $\mathcal{P}(\mathcal{D})$ by $holds(l_1), \ldots, holds(l_n)$ and $holds\_by\_default(l_1), \ldots, holds\_by\_default(l_n)$ respectively (we denote the resulting program by $\mathcal{P}_0(\mathcal{D})$);
- Dropping the rules with heads formed by literals $hold$ and $hold\_by\_default$.

Then

(a) if A is an answer set of $\mathcal{P}(\mathcal{D})$ then $A \setminus lit(\{hold, hold\_by\_default\})$ is an answer set of $\mathcal{P}_1(\mathcal{D})$;

(b) if A is an answer set of $\mathcal{P}_1(\mathcal{D})$ then
$A \cup \{hold([l_1, \ldots, l_n]) : holds(l_1), \ldots, holds(l_n) \in A\} \cup$
$\{hold\_by\_default([l_1, \ldots, l_n]) : holds\_by\_default(l_1), \ldots, holds\_by\_default(l_n) \in A\}$
is an answer set of $\mathcal{P}(\mathcal{D})$.

Proof. First notice that by Lemma 5.1, programs $\mathcal{P}(\mathcal{D})$ and $\mathcal{P}_0(\mathcal{D})$ are equivalent. Then observe that atoms formed by predicate symbols $hold$ and $hold\_by\_default$ form the complement of a splitting set of program $\mathcal{P}_0$. The conclusion of the lemma follows immediately from the splitting set theorem ([23]) and the fact that rules defining $hold$ and $hold\_by\_default$ contains neither *not* nor $\neg$. □

Lemma 5.4 Let $\mathcal{D}$ be a hierarchical domain description over signature σ and
$H = \{holds(l) : \mathcal{P}_s(\mathcal{D}) \models holds(l)\} \cup \{defeated(d) : \mathcal{P}_s(\mathcal{D}) \models defeated(d)\}$.

By $\mathcal{P}_2(\mathcal{D})$ we denote the program consisting of the following rules

$$holds\_by\_default(l). \qquad \text{if } holds(l) \in H \tag{1}$$

$$\begin{aligned} holds\_by\_default(l) \leftarrow \; & holds\_by\_default(l_1), \\ & \vdots \\ & holds\_by\_default(l_n), \end{aligned} \tag{2}$$

$$
\begin{array}{rcll}
 & & not\ defeated(d), & \\
 & & not\ holds\_by\_default(\neg l). & \\
 & & & \\
 & & \text{if } default(d, l, [l_1, \ldots, l_n]) \in \mathcal{D} & \\
 & & \text{and } holds(l) \notin H, & \\
 & & \text{and } holds(\neg l) \notin H & \\
 & & & \\
defeated(d_2) & \leftarrow & holds\_by\_default(l_1), & (3) \\
 & & \vdots & \\
 & & holds\_by\_default(l_n), & \\
 & & holds\_by\_default(prefer(d_1, d_2)) & \\
 & & not\ defeated(d_1). & \\
 & & & \\
 & & \text{if } d_2 \in \mathcal{D}, & \\
 & & default(d_1, l, [l_1, \ldots, l_n]) \in \mathcal{D}, & \\
 & & holds(conflict(d_1, d_2)) \in H & \\
 & & \text{and } holds(l) \notin H & \\
 & & \text{and } holds(\neg l) \notin H & \\
 & & & \\
holds\_by\_default(\neg prefer(d_1, d_2)) & \leftarrow & holds\_by\_default(prefer(d_2, d_1)) & (4) \\
 & & & \\
 & & \text{if } holds(prefer(d_1, d_2)) \notin H & \\
 & & \text{and } holds(prefer(d_2, d_1)) \notin H & \\
 & & \text{and } d_1, d_2 \in \mathcal{D} & \\
 & & & \\
\neg holds\_by\_default(l) & \leftarrow & holds\_by\_default(\neg l). & (5)
\end{array}
$$

Then, A is an answer set of $\mathcal{P}_1(\mathcal{D})$ iff $A = laws(\mathcal{D}) \cup H \cup B$ where B is an answer set of $\mathcal{P}_2(\mathcal{D})$.

Proof. Let U_0 be the set of literals formed by predicate symbols *holds*, *rule* and *default*. U_0 is a splitting set of program $\mathcal{P}_1(\mathcal{D})$ and hence A is an answer set of $\mathcal{P}_1(\mathcal{D})$ iff $A = A_0 \cup A_1$ where A_0 is the answer set of program $b_{U_0}(\mathcal{P}(\mathcal{D}))$ consisting of rules of $\mathcal{P}_1(\mathcal{D})$ whose heads belong to U_0 and A_1 is an answer set of the partial evaluation, $\mathcal{R} = e_{U_0}(t_{U_0}(\mathcal{P}(\mathcal{D})), A_0)$, of the rest of the program w.r.t. U_0 and A_0. It is easy to see that the program $\mathcal{R}$ consists of the rules of $\mathcal{P}_2(\mathcal{D})$ and

(a) rules of the type (2) where $holds(l)$ or $holds(\neg l)$ is in H;

(b) rules of the type (3) where $holds(l) \in H$ or $holds(\neg l) \in H$;

(c) rules of the type

$$
\begin{array}{lll}
holds\_by\_default(l) & \leftarrow & holds\_by\_default(l_1), \\
 & & \ldots \\
 & & holds\_by\_default(l_n),
\end{array}
$$

for each rule $rule(r, l, [l_1, \ldots, l_n]) \in \mathcal{D}$;

(d) rules of the type (4) where $holds(prefer(d_1, d_2)) \in H$ or $holds(prefer(d_2, d_1)) \in H$;

(e) facts of the type $defeated(d)$ where d is a default in $\mathcal{D}$ with the head l s.t. $holds(\neg l) \in H$.

From the rule (9) of program $\mathcal{P}$ we have that facts of the type (e) belong to H and hence to prove the lemma it is enough to show that the rules of the type (a)-(d) can be eliminated from $\mathcal{R}$ without changing its answer sets. To do that let us first make the following simple observation. Consider a program $\mathcal{Q}_1$ containing a rule $p \leftarrow \Gamma$ and the fact p and let $\mathcal{Q}_2$ be obtained from $\mathcal{Q}_1$ by removing the rule. $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are obviously equivalent in the logic N_2 and hence have the same answer sets. Similarly, we can show that a rule whose body contradicts a fact of the program can be removed from the program.

(1a) Consider a rule r of $\mathcal{R}$ of the type (a).
If $holds(l) \in H$ then, from rule (4) of $\mathcal{P}$ we have that $holds\_by\_default(l) \in \mathcal{R}$. Hence, by the above observation, r can be removed from $\mathcal{R}$ without changing its answer sets.
If $holds(\neg l) \in H$ then from rule (4) of $\mathcal{P}$ we have that $holds\_by\_default(\neg l) \in \mathcal{R}$ which contradicts the body of r. Hence r is useless and can be safely removed.

(1b) Now consider a rule r of the type (b). We will show that its head, $defeated(d_2)$, is a fact of $\mathcal{R}$.

First notice that, if $holds(\neg l) \in H$ then $\mathcal{P}_s(\mathcal{D}) \models defeated(d_1)$. Therefore, $defeated(d_1) \in A_0$ and hence, in this case, $r \notin \mathcal{R}$.

Suppose now that $holds(l) \in H$. Consider two cases:

(i) The head l of d_2 is σ literal.
By definition of rules of type (b) we have that $holds(conflict(d_1, d_2)) \in H$. From condition (2) of definition 5.3 of hierarchical domain description we conclude that the head of default d_1 is literal $\neg l$. Since r is of type (b), this means that $holds(\neg l) \in H$ and, from the rule (9) of $\mathcal{P}$ we have that $defeated(d_2) \in H$.

(ii) The head l of d_2 is of the form $prefer(d_i, d_j)$.
From conditions (2), (3) of definition 5.3 we have that the head of d_1 is $prefer(d_j, d_i)$. From rule (14) of $\mathcal{P}$ we have that $\neg prefer(d_i, d_j) \in H$. Finally, from rule (9) of $\mathcal{P}$ we have that $defeated(d_2) \in H$.
This demonstrates that the rules of the type (b) can be removed from $\mathcal{R}$ without changing its answer sets.

(1c) It is easy to check that by the condition (4) of definition 5.3 the body of a rule of the type (d) is satisfied iff $holds(l_1), \ldots, holds(l_n) \in H$ and hence the head of such a rule is in H or the rule is useless.

(1d) Similar argument can be used for the rules of the type (c). The conclusion of the lemma follows now from the observation above and the splitting set theorem. $\square$

Let us consider a logic program $\mathcal{Q}(\mathcal{D})$ obtained from program $\mathcal{P}_2(\mathcal{D})$ by

(a) removing rules of the type (5);

(b) replacing literals of the form $holds\_by\_default(l)$ and $defeated(d)$ by l and d respectively.

The program $\mathcal{Q}(\mathcal{D})$ has a form

$$
\mathcal{Q}(\mathcal{D}) = \left\{
\begin{array}{lcll}
l. & & \text{if } holds(l) \in H & (1) \\
 & & & \\
l & \leftarrow & l_1, \ldots, l_n, & (2) \\
 & & not\ d, & \\
 & & not\ \neg l. & \\
 & & & \\
 & & \text{if } default(d, l, [l_1, \ldots, l_n]) \in \mathcal{D} & \\
 & & \text{and } holds(l) \notin H, & \\
 & & \text{and } holds(\neg l) \notin H & \\
 & & & \\
d_2 & \leftarrow & l_1, \ldots, l_n, & (3) \\
 & & prefer(d_1, d_2) & \\
 & & not\ d_1. & \\
 & & & \\
 & & \text{if } d_2 \in \mathcal{D}, & \\
 & & default(d_1, l, [l_1, \ldots, l_n]) \in \mathcal{D}, & \\
 & & holds(conflict(d_1, d_2)) \in H & \\
 & & \text{and } holds(l) \notin H & \\
 & & \text{and } holds(\neg l) \notin H & \\
 & & & \\
\neg prefer(d_1, d_2) & \leftarrow & prefer(d_2, d_1). & (4) \\
 & & & \\
 & & \text{if } holds(prefer(d_1, d_2)) \notin H & \\
 & & \text{and } holds(prefer(d_2, d_1)) \notin H & \\
 & & \text{and } d_1, d_2 \in \mathcal{D} &
\end{array}
\right.
$$

Lemma 5.5 Let $\mathcal{D}$ be a hierarchical domain description over signature σ and let H be the set of literals defined as in Lemma 5.4. Then the program $\mathcal{Q}(\mathcal{D})$ is consistent.

Proof. First let us notice that the set F of facts of the form (1) from the program $\mathcal{Q}(\mathcal{D})$ form a splitting set of this program. Since $\mathcal{D}$ is rule consistent so is F. This implies that $\mathcal{Q}(\mathcal{D})$ is consistent iff the result $\mathcal{Q}_0$ of partial evaluation of $\mathcal{Q}(\mathcal{D})$ with respect to F is consistent. Let $\mathcal{Q}_1$ be the result of removal from $\mathcal{Q}_0$ all the rules whose bodies contain literals not belonging to the heads of rules from $\mathcal{Q}_0$. Obviously, $\mathcal{Q}(\mathcal{D})$ is equivalent to $\mathcal{Q}_1$.

To prove consistency of $\mathcal{Q}_1$ we construct its splitting sequence and use the splitting sequence theorem from [23].

Since $\mathcal{D}$ is hierarchical it has a rank function $rank$. Let μ be the smallest ordinal number such that $rank(l) < \mu$ for every l from the domain of $rank$. Let $heads(\mathcal{Q}_1)$ be the set of literals from the heads of rules in $\mathcal{Q}_1$ and

$$\begin{aligned} U_\alpha = \ & \{l : l \in lit(\sigma) \cap heads(\mathcal{Q}_1) \text{ s.t. } rank(l) < \alpha\} \cup \\ & \{d \in heads(\mathcal{Q}_1) : rank(head(d)) < \alpha\} \cup \\ & \{prefer(d_1, d_2) \in heads(\mathcal{Q}_1) : rank(prefer(d_1, d_2)) < \alpha\} \cup \\ & \{\neg prefer(d_1, d_2) : prefer(d_2, d_1) \in heads(\mathcal{Q}_1), rank(prefer(d_2, d_1)) < \alpha\} \end{aligned}$$

The sequence $U = \langle U_\alpha \rangle_{\alpha<\mu}$ is monotone ($U_\alpha \subset U_\beta$ whenever $\alpha < \beta$) and continuous (for each limit ordinal $\alpha < \mu$, $U_\alpha = \bigcup_{\beta<\alpha} U_\beta$). Using the property of the $rank$ function from the definition of hierarchical domain description it is not difficult to check that for each $\alpha < \mu$, U_α is a splitting set of $\mathcal{Q}_1$ and that $\bigcup_{\alpha<\mu} U_\alpha$ is equal to the set of all literals occurring in $\mathcal{Q}_1$. Hence, U is a splitting sequence of $\mathcal{Q}_1$. By the splitting sequence theorem existence of an answer set of $\mathcal{Q}_1$ follows from existence of a solution to $\mathcal{Q}_1$ (with respect to U). Let T_α be a collection of all the rules from $\mathcal{Q}_1$ whose heads belong to U_α. To show existence of such a solution it suffices to

(i) assume that for α such that $\alpha + 1 < \mu$ the program T_α has a consistent answer set A_α;

(ii) use this assumption to show that $T_{\alpha+1}$ also has a consistent answer set;

(iii) show that $\bigcup_{\alpha<\mu} A_\alpha$ is consistent.

Let us show (ii) and (iii). Let T be the result of partial evaluation of the program $T_{\alpha+1}$ with respect to the set A_α. T can be divided into three parts consisting of rules of the form

(a) $d_2 \leftarrow not\ d_1$.

and

(b) $l \leftarrow not\ d, not\ \neg l$ where l is a σ-literal

and

(c1) $prefer(d_i, d_j) \leftarrow not\ d, not\ \neg prefer(d_i, d_j)$

(c2) $\neg prefer(d_m, d_n) \leftarrow prefer(d_n, d_m)$.

respectively.

To show consistency of the program $T(a)$ consisting of rules (a) we first observe that, by construction, if a rule r of type (a) is in T then d_1, d_2 are conflicting defaults and hence, by condition 2 of definition 5.3 their heads are either contrary σ-literals or of the form $prefer(d_i, d_j)$ and $prefer(d_j, d_i)$ where $i \neq j$. Consider the dependency graph D of S_1. D obviously does not contain cycles with positive edges. We will show that it does not contain odd cycles with negative edges. (Programs with this property are called call-consistent). Suppose that $d_1, \ldots, d_{2n+1}, d_1$ is such a cycle. Since d_i and d_{i+1} ($i = 1, \ldots, 2n$) are conflicting defaults we have that d_1 and d_{2n+1} have the same heads (clause (2) of the definition). Since d_1 and d_{2n+1} are conflicting their heads must be different. Hence our program has no odd cycles. As was shown by Fages [14] (see also [12]), call consistent programs with dependency graphs without positive cycles have an answer set.

To show consistency of the program $T(a,b)$ consisting of rules (a) and (b) of T it suffices to take an arbitrary answer set of program $T(a)$ and use the splitting set theorem. The corresponding reduct R will consist of rules of the form $l \leftarrow not\ \neg l$. Let X_0 be the set of all positive literals from the heads of R and X_1 be the set of negative literals of the form $\neg l$ from the heads of R such that $l \notin X_0$. It is easy to see that the set $X_0 \cup X_1$ is a consistent answer set of R.

Now we need to show consistency of the partial evaluation T_r of T with respect to some answer set of $T(a,b)$. T_r consists of rules

$prefer(d_i, d_j) \leftarrow not\ \neg prefer(d_i, d_j)$

and

$\neg prefer(d_m, d_n) \leftarrow prefer(d_n, d_m)$.

Let $heads(T_r)$ be the set of the heads of the rules of T_r and let us assume that each default is associated with a unique index i. Consider a set X_0

$$\begin{array}{l} X_0 = \{prefer(d_i,d_j) : prefer(d_i,d_j) \in heads(T_r), prefer(d_j,d_i) \notin heads(T_r)\} \cup \\ \qquad \{prefer(d_i,d_j) : i < j \text{ if } prefer(d_i,d_j) \in heads(T_r) \text{ and } prefer(d_j,d_i) \in heads(T_r)\} \end{array}$$

Now let

$X = X_0 \cup \{\neg prefer(d_n, d_m) : prefer(d_m, d_n) \in X_0\}$

Obviously, X is consistent. To show that it is a consistent answer set of T_r let us construct T_r^X and show that

$prefer(d_i, d_j) \in T_r^X$ iff $prefer(d_i, d_j) \in X$.

Let

$prefer(d_i, d_j) \in X$.

Then, by construction of X,

$prefer(d_j, d_i) \notin X$, hence

$\neg prefer(d_i, d_j) \notin X$, i.e.

$prefer(d_i, d_j) \in T_r^X$.

Similar argument demonstrates equivalence in the opposite direction. This implies that X is a consistent answer set of T_r. By the splitting set theorem we conclude consistency of T and $T_{\alpha+1}$. Statement (iii) follows immediately from the above construction of answer set of $T_{\alpha+1}$ and hence, from the splitting sequence theorem we have that $\mathcal{Q}(\mathcal{D})$ is consistent. $\square$

By the splitting sequence theorem we conclude that $\mathcal{Q}_1$ is consistent which immediately implies consistency of $\mathcal{P}_2^1(\mathcal{D})$. To prove consistency of $\mathcal{P}_2^2(\mathcal{D})$ it is enough to notice that since answer set of $\mathcal{Q}_1$ is consistent, the corresponding answer set of $\mathcal{P}_2^1(\mathcal{D})$ never contains two literals of the form $holds\_by\_default(l)$ and $holds\_by\_default(\neg l)$. $\square$

Lemma 5.6 Let $\mathcal{D}$ be a hierarchical domain description over signature σ and let $\mathcal{Q}(\mathcal{D})$ be the program defined as in Lemma 5.5. Then for any literal l of $\mathcal{L}_0(\sigma)$

$\mathcal{D} \models holds\_by\_default(l)$ iff $\mathcal{Q}(\mathcal{D}) \models l$.

Proof. By definition,

1. $\mathcal{D} \models holds\_by\_default(l)$ iff $\mathcal{P}(\mathcal{D}) \models holds\_by\_default(l)$.

From (1) and lemma 5.3 we have that

2. $\mathcal{D} \models holds\_by\_default(l)$ iff $\mathcal{P}_1(\mathcal{D}) \models holds\_by\_default(l)$.

From (2) and lemma 5.4 we have that

3. $\mathcal{D} \models holds\_by\_default(l)$ iff $\mathcal{P}_2(\mathcal{D}) \models holds\_by\_default(l)$.

Now let us consider the program $\mathcal{Q}_p(\mathcal{D})$ obtained from $\mathcal{Q}(\mathcal{D})$ by replacing every negative literal $l = \neg p(t)$ by the atom $\overline{l} = \bar{p}(t)$ where $\bar{p}$ is a new predicate symbol.

From (3) and lemma 5.2 we have that

4. $\mathcal{P}_2(\mathcal{D}) \models holds\_by\_default(l)$ iff $\mathcal{Q}_p(\mathcal{D}) \models l$.

As was shown in [18] answer sets of $\mathcal{Q}(\mathcal{D})$ coincide with answer sets (stable models) of $\mathcal{Q}_p(\mathcal{D})$ which do not contain pairs of atoms of the form l, $\overline{l}$. Let us show that no answer set A of $\mathcal{Q}_p(\mathcal{D})$ contains such literals. Consider two cases:

(i) l is a σ-literal. Suppose that $l \in A$. Obviously there is no rule of the type (2) in $\mathcal{Q}_p(\mathcal{D})$ whose head is $\overline{l}$ and whose body is satisfied by A. Since $\mathcal{D}$ is rule-consistent $\overline{l} \notin \mathcal{Q}_p(\mathcal{D})$ and hence $\overline{l} \notin A$.

(ii) $l = prefer(d_i, d_j)$. There are free types of rules in $\mathcal{Q}_p(\mathcal{D})$ which contain literals formed by $prefer$ in the heads:

(a). $prefer(d_i, d_j)$.

from rule (1) of $\mathcal{Q}(\mathcal{D})$

(b). $prefer(d_i, d_j) \leftarrow \Gamma, not\ \overline{prefer}(d_i, d_j)$.

from rule (2) of $\mathcal{Q}(\mathcal{D})$ and

(c). $\overline{prefer}(d_i, d_j) \leftarrow prefer(d_j, d_i)$.

from rule (2) of $\mathcal{Q}(\mathcal{D})$.

Suppose that $prefer(d_i, d_j) \in A$. Then, from the rule consistency of $\mathcal{D}$ we have that $prefer(d_j, d_i)$ does not belong to (a). Since, by rule (c) we have that $\overline{prefer}(d_j, d_i) \in A$ and hence $prefer(d_j, d_i) \notin A$. This implies that $\overline{prefer}(d_i, d_j) \notin A$.

Hence, we have that

5. $\mathcal{Q}_p(\mathcal{D}) \models l$ iff $\mathcal{Q}(\mathcal{D}) \models l$,

which, together with (4) proves the lemma. $\square$

The proof of the theorem 5.1 follows immediately from Lemmas 5.5 and 5.6.

As lemma 5.6 points out the defeasible entailment in $\mathcal{Q}(\mathcal{D})$ and $\mathcal{D}$ are equivalent. Therefore, in what follows, we will refer to the program $\mathcal{Q}(\mathcal{D})$ as the *defeasible counterpart* of $\mathcal{D}$.

6 Relationship With Prioritized Logic Programs

In this section we discuss the relationship between our theory of prioritized defaults and the prioritized logic programs recently introduced by G. Brewka [6]. In Brewka's approach, a domain description is represented by a prioritized logic program $(P, <)$ where P is a logic program with the answer set semantics representing the domain without preferences and $<$ is a preference relation among rules of P. The semantics of $(P, <)$ is defined by answer sets of P satisfying some conditions determined by $<$.

We will recall the notion of preferred answer sets and show that for a restricted class of hierarchical domain descriptions Brewka's approach and our approach are equivalent. In what follows, we will use the following terminology.

A binary relation R on a set S is called *strict partial order* (or *order*) if R is irreflexive and transitive. An order R is *total* if for every pair $a, b \in S$, either $(a, b) \in R$ or $(b, a) \in R$ and is *well-founded* if every set $X \subseteq S$ has a minimal element. R is said to be *well-ordered* if R is total and well-founded.

A prioritized logic program $(P, <)$ where P is a collection of rules of the form

$$r: \qquad l_0 \leftarrow l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n$$

where l_i's are ground literals, and $<$ is an order on P. $l_1, \ldots, l_m$ are called the *prerequisites* of r. If $m = 0$ then r is said to be a *prerequisite free* rule. r is *defeated* by a literal l if $l = l_i$ for some $i \in \{m+1, \ldots, n\}$. r is defeated by a set of literal X if X contains a literal that defeats r. A program P is *prerequisite free* if every rule in P is prerequisite free.

For a program P and a set of literals X, the *reduct of a program P with respect to X*, denoted by ${}^X P$, is the program obtained by

- deleting all rules with prerequisite l such that $l \notin X$; and
- deleting all prerequisites of the remaining rules.

Definition 6.1 *(Brewka [6])*

- Let $(P, <)$ be a prioritized logic program where P is a prerequisite free program and $<$ is a total order among rules of P, we define $C_<(P) = \bigcup_{i=1}^{\infty} S_i$ as follows.

  $$S_0 = \emptyset$$

  $$S_n = \begin{cases} S_{n-1} \text{ if } r_n \text{ is defeated by } S_{n-1} \\ S_{n-1} \cup \{head(r_n)\} \text{ otherwise} \end{cases}$$

  where r_n is the n^{th} rule in the order $<$.

- Let $(P, <)$ be a prioritized logic program where P is a prerequisite free program and $<$ is a total order among rules of P. An answer set A of P is called a *preferred answer set*[3], of $(P, <)$ if $A = C_<(P)$.

[3] Strong preferred answer set in Brewka's terminology.

- For an arbitrary prioritized logic program $(P,<)$, a set of literals A is a preferred answer set of $(P,<)$ if it is an answer set of P and $A = C_{<'}({}^{A}P)$ for some total order $<'$ that extends $<$.

- A prioritized program $(P,<)$ entails a query q, denoted by $(P,<) \mid\!\sim q$, if for every preferred answer set A of $(P,<)$, $A \models q$.

Notice that in our approach, up to this point, we mainly discuss domain descriptions containing normal defaults. We do, however, allow dynamical priorities whereas Brewka does not. Furthermore, Brewka requires the transitivity of the preference relationship. Therefore, in this section, we will restrict ourself on domain descriptions where priorities are static, i.e., there exists no default with the head of the form $prefer(d_1, d_2)$ and the binary relationship defined by $prefer$ is transitive. This leads to the following definition of static domain descriptions.

Definition 6.2 A domain description $\mathcal{D}$ is said to be *static* if it satisfies the following conditions:

- laws of $\mathcal{D}$ do not contain occurrences of the predicate symbol $prefer$;

- the transitive closure of the preference relation $\{\langle d_1, d_2\rangle : d_1, d_2$ are defaults in $\mathcal{D}$ such that $prefer(d_1, d_2) \in \mathcal{D}\}$, denoted by $prefer^*_{\mathcal{D}}$, is an order on defaults of $\mathcal{D}$.

For a static domain description $\mathcal{D}$, the prioritized logic program $\Pi(\mathcal{D}) = (\mathcal{B}(\mathcal{D}), <_{\mathcal{D}})$ of $\mathcal{D}$ is defined as follows.

$$\Pi(\mathcal{D}) = \begin{cases} \mathcal{B}(\mathcal{D}) = \begin{cases} l. & & & \text{if } l \text{ is a } \sigma\text{-literal in } \mathcal{D} & (1) \\ l & \leftarrow & l_1, \ldots, l_n. & \text{if } rule(r, l, [l_1, \ldots, l_n]) \in \mathcal{D} & (2) \\ d: \quad l & \leftarrow & l_1, \ldots, l_n, not\ \neg l. & \text{if } default(d, l, [l_1, \ldots, l_n]) \in \mathcal{D} & (3) \end{cases} \\ d_1 <_{\mathcal{D}} d_2 & & & \text{if } \langle d_1, d_2\rangle \in prefer^*_{\mathcal{D}} \quad (4) \end{cases}$$

We say that a domain description $\mathcal{D}$ entails a σ-literal l in the sense of Brewka (and write $\mathcal{D} \models_{\mathcal{B}} q$) if $\Pi(\mathcal{D}) \mid\!\sim q$.

To continue with our discussion, we need the following notation.

A domain description $\mathcal{D}$ will be called a domain description with a basic preference relation if for every literal of the form $prefer(d_1, d_2) \in \mathcal{D}$, $head(d_1)$ and $head(d_2)$ are contrary literals.

The following theorem shows that for a static and hierarchical domain description with a basic preference relation, Brewka's approach coincides with ours.

Theorem 6.1 For every hierarchical and static domain description $\mathcal{D}$ with a basic preference relation and every σ-literal l,

$$\mathcal{D} \models holds\_by\_default(l) \qquad \text{if and only if} \qquad \mathcal{D} \models_{\mathcal{B}} l.$$

Recall that, by Lemma 5.6,

$\mathcal{D} \models holds\_by\_default(l)$ iff $\mathcal{Q}(\mathcal{D}) \models l$

where $\mathcal{Q}(\mathcal{D})$ is the program defined in Lemma 5.5. Hence, to prove the theorem, it suffices to show that

$\mathcal{Q}(\mathcal{D}) \models l$ iff $\Pi(\mathcal{D}) \mid\!\sim l$.

Let us introduce some useful terminology and notation. Let $\mathcal{D}$ be a hierarchical domain description and

$$U(\mathcal{D}) = \{l \in \mathcal{L}_0(\sigma) : \mathcal{P}_s(\mathcal{D}) \models holds(l)\}$$

where $\mathcal{P}_s(\mathcal{D})$ is the non-defeasible part of $\mathcal{P}(\mathcal{D})$.

Let $\mathcal{D}_N$ be the domain description obtained from $\mathcal{D}$ by

(i) removing all rules and σ-literals from $\mathcal{D}$;

(ii) removing all defaults $d \in \mathcal{D}$ such that $head(d) \in U(\mathcal{D})$ or $\neg head(d) \in U(\mathcal{D})$;

(iii) removing every occurrence of σ-literal $l \in U(\mathcal{D})$ from the bodies of the remaining defaults of $\mathcal{D}$;

(iv) for each rank r:

- removing all defaults $d \in \mathcal{D}$ such that $rank(head(d)) = r$ and $body(d)$ contains a literal not belonging to the head of any of the remaining defaults in $\mathcal{D}$;
- removing all literals of the form $prefer(d_1, d_2)$ containing default d_i of the rank r not belonging the remaining part of $\mathcal{D}$.

The domain description $\mathcal{D}_N$ will be called *the normalization of* $\mathcal{D}$.

A hierarchical domain description $\mathcal{D}$ is said to be *normalized* if $\mathcal{D} = \mathcal{D}_N$.

Let $\mathcal{Q}(\mathcal{D})$ be the defeasible counterpart of a static domain description $\mathcal{D}$ and let $\mathcal{R}(\mathcal{D})$ be obtained from $\mathcal{Q}(\mathcal{D})$ by

(a) removing rules of the type (4);

(b) performing partial evaluation of the resulting program with respect to $U(\mathcal{D})$.

This construction, together with the following simple lemma, will be frequently used in our proof.

Lemma 6.1 For any static and hierarchical domain description $\mathcal{D}$ and σ-literal $l \notin U(\mathcal{D})$, $\mathcal{Q}(\mathcal{D}) \models l$ iff $\mathcal{R}(\mathcal{D}) \models l$.

Proof. First notice that since $\mathcal{D}$ is static $\neg prefer(d_1, d_2) \in U(\mathcal{D})$ or $prefer(d_2, d_1) \notin U(\mathcal{D})$. Hence the program $\mathcal{Q}_a(\mathcal{D})$ obtained from $\mathcal{Q}(\mathcal{D})$ by step (a) has the same answer sets as $\mathcal{Q}(\mathcal{D})$.

Now notice that since $\mathcal{D}$ is static the heads of rules of the type (2) in $\mathcal{Q}(\mathcal{D})$ belong to $lit(\sigma)$. By construction of $\mathcal{Q}(\mathcal{D})$ these heads do not belong to $U(\mathcal{D})$. Therefore, $U(\mathcal{D})$ is a splitting set of $\mathcal{Q}_a(\mathcal{D})$ and conclusion of the lemma follows from the splitting set theorem. $\square$

The proof of the theorem 6.1 will be based on the following lemmas.

Lemma 6.2 Let $\mathcal{D}_N$ be the normalization of a static and hierarchical domain description $\mathcal{D}$. Then, for every σ-literal l such that $l \notin U(\mathcal{D})$

$$\mathcal{D} \models holds\_by\_default(l) \quad \text{iff} \quad \mathcal{D}_N \models holds\_by\_default(l).$$

Proof. Let l be a σ-literal such that $l \notin U(\mathcal{D})$. Since $\mathcal{D}$ is hierarchical we have that by Lemma 5.6 it suffices to show that

a. $\mathcal{Q}(\mathcal{D}) \models l$ iff $\mathcal{Q}(\mathcal{D}_N) \models l$.

Domain descriptions $\mathcal{D}$ and $\mathcal{D}_N$ are static and hierarchical and hence, by Lemma 6.1 we have that (a) is true iff

b. $\mathcal{R}(\mathcal{D}) \models l$ iff $\mathcal{R}(\mathcal{D}_N) \models l$.

Let $\mathcal{D}^*$ be the domain description obtained from $\mathcal{D}$ by performing the steps (i), (ii), and (iii) in the construction of $\mathcal{D}_N$. Obviously, $\mathcal{D}_N \subseteq \mathcal{D}^*$. We first prove that

c. $\mathcal{R}(\mathcal{D})$ and $\mathcal{R}(\mathcal{D}^*)$ are identical.

Let

c1. $r \in \mathcal{R}(\mathcal{D})$

We consider two cases:

(i) $head(r) \in lit(\sigma)$, i.e.,

r is of the form $l_0 \leftarrow \Gamma, not\ d, not\ \neg l_0$

where Γ consists of σ-literals not belonging to $U(\mathcal{D})$. By construction of $\mathcal{R}(\mathcal{D})$ and $\mathcal{Q}(\mathcal{D})$ this is possible iff

c2. neither l_0 nor $\neg l_0$ is in $U(\mathcal{D})$ and there is a set of literals $\Delta \subseteq U(\mathcal{D})$ such that $default(d, l_0, [\Delta, \Gamma]) \in \mathcal{D}$.

From definition of $\mathcal{D}_N$ we have that (c2) holds iff

c3. $default(d, l_0, [\Gamma]) \in \mathcal{D}^*$.

Notice also that, by the same definition, $U(\mathcal{D}^*)$ consists of literals formed by $prefer$ and $conflict$ and hence do not contain σ-literals. This implies that (c3) holds iff c4. $r \in \mathcal{Q}(\mathcal{D}^*)$.

Since $\mathcal{D}$ is static, literals from $U(\mathcal{D}^*)$ do not belong to rules (2) of $\mathcal{Q}(\mathcal{D}^*)$. This implies that (c4) holds iff

c5. $r \in \mathcal{R}(\mathcal{D}^*)$.

(ii) $head(r) \notin lit(\sigma)$, i.e.

r is of the form $d_2 \leftarrow \Gamma, not\ d_1$

where Γ consists of σ-literals not belonging to $U(\mathcal{D})$.

By construction of $\mathcal{R}(\mathcal{D})$ this is possible iff

c6. $default(d_1, l_0, [\Delta, \Gamma]), default(d_2, \neg l_0, [\Delta_1, \Gamma_1]) \in \mathcal{D}$

for some $\Delta \subseteq U(\mathcal{D})$, $\Delta_1 \subseteq U(\mathcal{D})$ and Γ_1 consisting of σ-literals not belonging to $U(\mathcal{D})$; $l_0, \neg l_0 \notin U(\mathcal{D})$, and $prefer(d_1, d_2) \in \mathcal{D}$.

It follows from definition of $\mathcal{D}^*$ that (c6) holds iff

c7. $default(d_1, l_0, [\Gamma]) \in \mathcal{D}^*$, $default(d_2, \neg l_0, [\Gamma_1]) \in \mathcal{D}^*$ and $prefer(d_1, d_2) \in \mathcal{D}^*$.

which holds iff

c8. $r \in \mathcal{R}(\mathcal{D}^*)$.

From (c1), (c5) and (c8) we have that $\mathcal{R}(\mathcal{D})$ and $\mathcal{R}(\mathcal{D}^*)$ are identical. Therefore, to prove (b) we will show

d. $\mathcal{R}(\mathcal{D}^*) \models l$ iff $\mathcal{R}(\mathcal{D}_N) \models l$.

Let

e. A be an answer set of $\mathcal{R}(\mathcal{D}^*)$.

Let

f. $B = A \setminus \{d : d \in \mathcal{D}^* \setminus \mathcal{D}_N\}$.

We will prove that

d1. B is an answer set of $\mathcal{R}(\mathcal{D}_N)$.

By construction of $\mathcal{R}(\mathcal{D}^*)$ and $\mathcal{R}(\mathcal{D}_N)$ it is easy to see that

d2. $(\mathcal{R}(\mathcal{D}_N))^B \subseteq (\mathcal{R}(\mathcal{D}))^A$.

Hence,

d3. B is closed under the rules of $(\mathcal{R}(\mathcal{D}_N))^B$.

Assume that there exists a set of literals $C \subset B$, which is closed under the rules of $(\mathcal{R}(\mathcal{D}_N))^B$. Let d4. $D = (C \cap lit(\sigma)) \cup (A \setminus lit(\sigma))$.

We will prove that

d5. D is closed under the rules of $(\mathcal{R}(\mathcal{D}^*))^A$.

By construction of D,

d6. D is closed under the rules of $(\mathcal{R}(\mathcal{D}^*))^A$ whose heads do not belong to $lit(\sigma)$.

Consider a rule

e0. $l_0 \leftarrow \Gamma \in (\mathcal{R}(\mathcal{D}^*))^A$ such that

e1. $\Gamma \subseteq B$.

By construction of $(\mathcal{R}(\mathcal{D}^*))^A$, this is possible if there exists a default d,

e2. $default(d, l_0, [\Gamma]) \in \mathcal{D}^*$,

e3. $\neg l_0 \notin B$, $d \notin A$,

From (e2) and the fact that C is closed under the rules of $(\mathcal{R}(\mathcal{D}_N))^B$, by construction of $\mathcal{D}_N$, we conclude that

e4. $default(d, l_0, [\Gamma]) \in \mathcal{D}_N$.

which, together with (e3), implies that

e5. $l_0 \leftarrow \Gamma \in (\mathcal{R}(\mathcal{D}_N))^B$

Since C is closed under the rules of $(\mathcal{R}(\mathcal{D}_N))^B$, (e5) together with (e1), implies that $l_0 \in C$. This proves that D is closed under the rules of $(\mathcal{R}(\mathcal{D}^*))^A$ with σ-literals in their heads. This, together with (d6), implies (d5), and hence, implies that, A is not an answer set of $\mathcal{R}(\mathcal{D}^*)$. This contradiction proves (d1).

Now, let

f1. A be an answer set of $\mathcal{R}(\mathcal{D}_N)$,

and

f2. $B = A \cup \{d : d \in \mathcal{D}^* \setminus \mathcal{D}_N,$
$\exists d' \in \mathcal{D}_N,\ prefer(d', d) \in \mathcal{D}^*,\ body(d') \subseteq A\}$.

We will prove that B is an answer set of $\mathcal{R}(\mathcal{D}^*)$ by showing that B is a minimal set of literals closed under the rules of $(\mathcal{R}(\mathcal{D}^*))^B$.

Since A is an answer set of $\mathcal{R}(\mathcal{D}_N)$ we can conclude that

f3. for any $d \in \mathcal{D}^* \setminus \mathcal{D}_N$, $body(d)$ is not satisfied by A.

This, together with the construction of B and the fact that every rule of $(\mathcal{R}(\mathcal{D}^*))^B$ is of the form $l \leftarrow \Gamma$ or $d \leftarrow \Gamma$ where Γ is the body of some default in $\mathcal{D}^*$, implies that

f4. B is closed under the rules of $(\mathcal{R}(\mathcal{D}^*))^B$.

We need to prove the minimality of B. Assume the contrary, there exists a set of literals $C \subset B$ that is closed under the rules of $(\mathcal{R}(\mathcal{D}^*))^B$. Let f5. $D = C \setminus (B \setminus A)$.

Obviously, $D \subset A$. Since $(\mathcal{R}(\mathcal{D}_N))^A \subseteq (\mathcal{R}(\mathcal{D}^*))^B)$, it is easy to check that D is closed under the rules of $(\mathcal{R}(\mathcal{D}_N))^A$ which contradicts the fact that A is an answer set of $\mathcal{R}(\mathcal{D}_N)$, i.e., we have proved that

f6. B is an answer set of $\mathcal{R}(\mathcal{D}^*)$.

From (e), (d1), (f1), and (f6) we can conclude (d). which, together with (a), (b), and (c) proves the lemma. □

The next lemma shows that for a static and hierarchical domain description, the program $\mathcal{B}(\mathcal{D})$ can also be simplified.

Lemma 6.3 Let $\mathcal{D}$ be a static and hierarchical domain description and $\mathcal{D}_N$ be its normalization. Then, for each σ-literal l such that $l \notin U(\mathcal{D})$,

$$\Pi(\mathcal{D}) \mathrel{|\!\!\sim} l \qquad \text{if and only if} \qquad \Pi(\mathcal{D}_N) \mathrel{|\!\!\sim} l.$$

Proof. First, observe the following for prioritized programs.

Let $(Q, <)$ be a prioritized program where Q is a defeasible program without facts, i.e., each rule in Q contains at least a negation-as-failure literal. Let P be a strict program, i.e., no rule in P contains a negation-as-failure literal. Let $head(Q)$ be the set of literals belonging to the heads of Q and $body(P)$ be the set of literals belonging to the body of rules of P. Assume that $head(Q) \cap body(P) = \emptyset$. Then, we have that

(i) A is a preferred answer set of $(P \cup Q, <)$ iff $A = A_P \cup A_Q$ where A_P is the answer set of P and A_Q is a preferred answer set of $(Q_P, <)$ where Q_P is the partial evaluation of Q with respect to A_P.

(ii) Let P' be a strict program equivalent to P. Then, $(P \cup Q, <)$ and $(P' \cup Q, <)$ are equivalent.

(iii) Let R be the set of rules in Q such that for every $r \in R$, $P \models head(r)$ or $P \models \neg head(r)$. Then, $(P \cup Q, <)$ and $(P \cup Q \setminus R, <)$ are equivalent.

Let us denote the program consisting of rules (3) of $\mathcal{B}(\mathcal{D})$ by Q and $P = \mathcal{B}(\mathcal{D}) \setminus Q$. Obviously,

a. Q is a defeasible logic program without facts and P is a strict program.

Since $\mathcal{D}$ is hierarchical, we have that

b. $head(Q) \cap body(P) = \emptyset$.

Let $U_0(\mathcal{D})$ be the set of σ-literals belonging to $U(\mathcal{D})$. It is easy to see that $U_0(\mathcal{D})$ is the unique answer set of P, i.e., $U_0(\mathcal{D})$ and P are equivalent. Therefore, together with (a) and (b), by (ii) we can conclude that

c. $\Pi(\mathcal{D}) \mathrel{|\!\!\sim} l$ iff $(U_0(\mathcal{D}) \cup Q, <_{\mathcal{D}}) \mathrel{|\!\!\sim} l$.

Let R be the set of rules in Q such that for every $r \in R$, $head(r) \in U_0(\mathcal{D})$ or $\neg head(r) \in U_0(\mathcal{D})$, then by (iii) we know that

d. $(U_0(\mathcal{D}) \cup Q, <_{\mathcal{D}}) \mid\!\sim l$ iff $(U_0(\mathcal{D}) \cup Q \setminus R, <_{\mathcal{D}}) \mid\!\sim l$.

It is easy to see that $U_0(\mathcal{D})$ is a splitting set of $U_0(\mathcal{D}) \cup Q \setminus R$. Let S be the reduct of $U_0(\mathcal{D}) \cup Q \setminus R$ with respect to $U_0(\mathcal{D})$.

As in the previous proof, let $\mathcal{D}^*$ be the domain description obtained from $\mathcal{D}$ by performing the steps (i), (ii), and (iii) in the construction of $\mathcal{D}_N$. We will prove that S is identical to $\mathcal{B}(\mathcal{D}^*)$. Let

e1. $r \in S$.

It means that r has the form

e2. $l \leftarrow \Gamma, not\ \neg l$.

where Γ is a set of σ-literals containing no literals from $U_0(\mathcal{D})$. By construction of S, (e2) holds iff

e3. $l \notin U_0(\mathcal{D})$, $\neg l \notin U_0(\mathcal{D})$, and there exists a set of literals $\Delta \subseteq U_0(\mathcal{D})$ such that $default(d, l, [\Gamma, \Delta]) \in \mathcal{D}$.

From the definition of $\mathcal{D}^*$, (e3) holds iff

e4. $default(d, l, [\Gamma]) \in \mathcal{D}^*$

By definition of $\mathcal{B}(\mathcal{D}^*)$ and the definition of $\mathcal{D}^*$, (e4) holds iff

e5. r is a rule in $\mathcal{B}(\mathcal{D}^*)$.

From (e1) and (e5) we can conclude that

e. S is identical to $\mathcal{B}(\mathcal{D}^*)$.

From (e), (i), (c), and (d), and the splitting set theorem, we have that

f. $\Pi(\mathcal{D}) \mid\!\sim l$ iff $l \in U(\mathcal{D})$ or $\Pi(\mathcal{D}^*) \mid\!\sim l$.

This, implies that to prove the lemma, it suffices to show that

g. $\Pi(\mathcal{D}^*) \mid\!\sim l$ iff $\Pi(\mathcal{D}_N) \mid\!\sim l$.

To prove (g) we first prove that

g1. $\mathcal{B}(\mathcal{D}^*)$ and $\mathcal{B}(\mathcal{D}_N)$ are equivalent.

Let

g2. A be an answer set of $\mathcal{B}(\mathcal{D}^*)$.

Since $\mathcal{D}_N \subseteq \mathcal{D}^*$, we have that

g3. $(\mathcal{B}(\mathcal{D}_N))^A \subseteq (\mathcal{B}(\mathcal{D}^*))^A$

which immediately implies that

g4. A is closed under the rules of $(\mathcal{B}(\mathcal{D}_N))^A$.

Furthermore, it is easy to prove that if $B \subset A$ is closed under the rules of $(\mathcal{B}(\mathcal{D}_N))^A$ then B is closed under the rules of $(\mathcal{B}(\mathcal{D}^*))^A$. This, together with (g4), implies that

g5. A is an answer set of $\mathcal{B}(\mathcal{D}_N)$.

Now, let

g6. A be an answer set of $\mathcal{B}(\mathcal{D}_N)$.

Since for any rule

g7. $l \leftarrow \Gamma \in (\mathcal{B}(\mathcal{D}^*))^A \setminus (\mathcal{B}(\mathcal{D}_N))^A$

there exists a default d such that

g8. $default(d, l, [\Gamma]) \in \mathcal{D}^* \setminus \mathcal{D}_N$.

Hence, we can conclude that

g9. if r is a logic programming rule in $(\mathcal{B}(\mathcal{D}^*))^A \setminus (\mathcal{B}(\mathcal{D}_N))^A$ then $body(r)$ is not satisfied by A.

This, together with (g6) and the fact that $(\mathcal{B}(\mathcal{D}_N))^A \subseteq (\mathcal{B}(\mathcal{D}^*))^A$, implies that

g10. A is an answer set of $\mathcal{B}(\mathcal{D}^*)$.

From (g2), (g5), (g6), and (g10) we can conclude (g1).

The conclusion (g) follows from (g1) and the fact that ${}^A(\mathcal{B}(\mathcal{D}^*)$ is identical to ${}^A(\mathcal{B}(\mathcal{D}_N)$. □

The above two lemmas show that for any static and hierarchical domain description $\mathcal{D}$ and σ-literal $l \notin U(\mathcal{D})$

(i) $\mathcal{Q}(\mathcal{D}) \models l$ iff $\mathcal{R}(\mathcal{D}_N) \models l$ and

(ii) $\Pi(\mathcal{D}) \mid\!\sim l$ iff $\Pi(\mathcal{D}_N) \mid\!\sim l$.

where $\mathcal{D}_N$ is the normalization of $\mathcal{D}$.

Furthermore, for $l \in U(\mathcal{D})$, $\mathcal{Q}(\mathcal{D}) \models l$ and $\Pi(\mathcal{D}) \mid\!\sim l$.

Therefore, to prove the theorem 6.1, we will show that for $l \notin U(\mathcal{D})$,
$\mathcal{R}(\mathcal{D}_N) \models l$ iff $\Pi(\mathcal{D}_N) \mid\!\sim l$.

The above observation shows that in proving theorem 6.1 we can limit ourself to static and normalized domain descriptions with a basic reference relation. Since for a static and normalized domain description $\mathcal{D}$, the programs $\mathcal{R}(\mathcal{D})$ and $\Pi(\mathcal{D})$ are simpler than for general cases, for future references, we define these programs before continuing with the proof of theorem 6.1.

For a static and normalized domain description $\mathcal{D}$, the program $\mathcal{R}(\mathcal{D})$ consists of the following rules

$$\mathcal{R}(\mathcal{D}) \begin{cases} l \quad \leftarrow \quad l_1, \ldots, l_n, not\ d, not\ \neg l. & (1) \\ \qquad\qquad \text{if } default(d, l, [l_1, \ldots, l_n]) \in \mathcal{D} & \\ d_2 \quad \leftarrow \quad l_1, \ldots, l_n, not\ d_1. & (2) \\ \qquad\qquad \text{if } d_2 \in \mathcal{D}, default(d_1, l, [l_1, \ldots, l_n]) \in \mathcal{D},\ prefer(d_1, d_2) \in \mathcal{D}, & \\ \qquad\qquad \text{and } head(d_2) = \neg l & \end{cases}$$

and the program $\mathcal{B}(\mathcal{D})$ of $\Pi(\mathcal{D})$ consists of the following rules:

$$\mathcal{B}(\mathcal{D}) \begin{cases} l \quad \leftarrow \quad l_1, \ldots, l_n, not\ \neg l. & (1) \\ \qquad\qquad \text{if } default(d, l, [l_1, \ldots, l_n]) \in \mathcal{D} & \end{cases}$$

To continue with the proof we need the following definitions.

Definition 6.3 Let $\mathcal{D}$ be a static domain description with the preference relation P_0. Let P_1 be a well-ordered order defined on defaults in $\mathcal{D}$ which extends P_0. The domain description $\tilde{\mathcal{D}} = \mathcal{D} \cup \{prefer(d_1, d_2) : \langle d_1, d_2 \rangle \in P_1\}$ is called a *completion* of $\mathcal{D}$.

We will need the following technical observations.

Lemma 6.4 Let $\mathcal{D}$ be a static and normalized domain description. Let A be an answer set of $\mathcal{R}(\mathcal{D})$ and $default(d, l, [\Gamma])$ be a default in $\mathcal{D}$ such that $l \notin A$ and $\Gamma \subseteq A$. Then, $\neg l \in A$.

Proof. First notice that, since $\mathcal{D}$ is normalized, it is hierarchical. Therefore, in virtue of theorem 5.1, $\mathcal{D}$ is consistent. By Lemmas 5.6 and 6.1 this implies that $\mathcal{R}(\mathcal{D})$ is consistent. As was shown in [18] every answer set of consistent program is consistent which implies consistency of A.

Since $l \leftarrow \Gamma, not\ d, not\ \neg l$ is a rule in $\mathcal{R}(\mathcal{D})$, $\Gamma \subseteq A$, $l \notin A$, and A is a consistent answer set of $\mathcal{R}(\mathcal{D})$, we have two cases:

(i) $\neg l \in A$; or

(ii) $d \in A$.

Consider the second case: $d \in A$. Then there exists a rule (2) of $\mathcal{R}(\mathcal{D})$ with the head d whose body is satisfied by A. From construction of $\mathcal{R}$ this implies that there exists a default

1. $default(d_1, \neg l, [\Delta]) \in \mathcal{D}$

such that

2. $\Delta \subseteq A$ and $d_1 \notin A$.

From (1) and construction of $\mathcal{R}$ we can conclude that $\mathcal{R}$ contains the rule

3. $\neg l \leftarrow \Delta, not\ d_1, not\ l.$

Recall, that, by condition of the lemma, $l \notin A$. This, together with (2), implies that the body of the rule (3) is satisfied by A. Therefore, $\neg l \in A$. $\square$

Let X be a set of literals in the language of $\mathcal{R}(\mathcal{D})$. By $X|_l$ we denote $X \cap lit(\sigma)$.

Lemma 6.5 Let $\mathcal{D}$ be a static and normalized domain description and $\tilde{\mathcal{D}}$ be one of its completions. Then, for every answer set $\tilde{A}$ of $\mathcal{R}(\tilde{\mathcal{D}})$ there exists an answer set A of $\mathcal{R}(\mathcal{D})$ such that $\tilde{A}|_l = A|_l$.

Proof. Since the preference relation in $\tilde{\mathcal{D}}$ is a well-ordered order among defaults, we can enumerate the set of defaults in $\mathcal{D}$ by the sequence $d_0, d_1, \ldots, d_n, \ldots$.[4]

Let $\tilde{A}$ be an answer set of $\mathcal{R}(\tilde{\mathcal{D}})$. It is easy to see that, since $\mathcal{D}$ is normalized, $\tilde{A}$ is consistent.

We define a sequence of sets of literals $A_{i=0}^{\infty}$ in the language of $\mathcal{R}(\mathcal{D})$ as follows:

$$A_0 = \tilde{A}|_l$$

$$A_{n+1} = \begin{cases} A_n \cup \{d_{n+1}\} & \text{if there exists } d_i \text{ s.t.} \\ & \text{(0a) } default(d_i, \neg head(d_{n+1}), [\Gamma]) \in \mathcal{D}, \\ & \text{(0b) } prefer(d_i, d_{n+1}) \in \mathcal{D}, \\ & \text{(0c) } \Gamma \subseteq A_n, \text{ and} \\ & \text{(0d) } d_i \notin A_n. \\ & \\ A_n & \text{otherwise} \end{cases}$$

Let $A = \cup_{i=0}^{\infty} A_i$. Obviously, A is consistent. We will prove that A is an answer set of $\mathcal{R}(\mathcal{D})$ and $A|_l = \tilde{A}|_l$.

By the construction of A, we have that $A|_l = \tilde{A}|_l$. Hence, to prove the lemma we need to prove that A is an answer set of $\mathcal{R}(\mathcal{D})$. To do that, we will show that A is a minimal set of literals which is closed under the rules of $(\mathcal{R}(\mathcal{D}))^A$.

[4]For simplicity, here and in the following lemmas we assume that the set of defaults in $\mathcal{D}$ has the cardinality less than or equal to the ordinal number ω. However, the proofs presented in this paper can be expanded to the general case.

Since $\mathcal{D}$ is a normalized domain description, $(\mathcal{R}(\mathcal{D}))^A$ consists of the following rules:

$$
(\mathcal{R}(\mathcal{D}))^A = \left\{
\begin{array}{lll}
l & \leftarrow & \Gamma. \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (1) \\
 & & \\
 & & \text{if there is } d \text{ s.t.} \\
 & & \text{(1a) } default(d, l, [\Gamma]) \in \mathcal{D}, \\
 & & \text{(1b) } d \notin A, \text{ and } \neg l \notin A \\
 & & \\
d_2 & \leftarrow & \Gamma. \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (2) \\
 & & \\
 & & \text{if there is } d_1 \text{ s.t.} \\
 & & \text{(2a) } default(d_1, \neg head(d_2), [\Gamma]) \in \mathcal{D}, \\
 & & \text{(2b) } prefer(d_1, d_2) \in \mathcal{D}, \text{ and} \\
 & & \text{(2c) } d_1 \notin A.
\end{array}
\right.
$$

Let r be a rule of $(\mathcal{R}(\mathcal{D}))^A$ whose body is satisfied by A, i.e.,

a. $\Gamma \subseteq A$.

We consider two cases:

(i) r is of the form (1).

Since $A|_l = \tilde{A}|_l$, from (1b) and (a) we conclude that

b. $\neg l \notin \tilde{A}$ and $\Gamma \subseteq \tilde{A}$.

By Lemma 6.4, this, together with (1a) implies that $l \in \tilde{A}$ and hence $l \in A$, i.e.,

c. A is closed under the rules of type (1) of $(\mathcal{R}(\mathcal{D}))^A$.

(ii) r is of the form (2). From (2a)-(2c) and (a), by the construction of A, we conclude that $d_2 \in A$, i.e.,

d. A is closed under the rules of type (2) of $(\mathcal{R}(\mathcal{D}))^A$.

From (c) and (d) we can conclude that

e. A is closed under the rules of $(\mathcal{R}(\mathcal{D}))^A$.

We now prove the minimality of A.

Assume that there exists a set $B \subset A$ which is closed under the rules of $(\mathcal{R}(\mathcal{D}))^A$. We consider two cases:

(i) $A|_l \setminus B|_l \neq \emptyset$.

Since $\mathcal{D}$ is hierarchical, there exists a rank function $rank$ of $\mathcal{D}$ that satisfies the conditions of Definition 5.3.

Let $l \in A|_l \setminus B|_l$ such that

f. $rank(l) = \min\{rank(p) : p \in A|_l \setminus B|_l\}$.

Since $l \in A$ and $A|_l = \tilde{A}|_l$, we have that $l \in \tilde{A}$. Let

f1. $\Delta_l^+ = \{d : default(d, l, [\Gamma]) \in \mathcal{D},\ \Gamma \subseteq \tilde{A}\}$.

Since $\tilde{A}$ is an answer set of $\mathcal{R}(\tilde{\mathcal{D}})$, we have that

f2. $\Delta_l^+ \neq \emptyset$.

Since the preference relation in $\tilde{\mathcal{D}}$ is well-ordered, there exists a minimal element d_j of Δ_l^+ such that

f3. $prefer(d_j, d_k) \in \tilde{\mathcal{D}}$ for $d_k \in \Delta_l^+ \setminus \{d_j\}$.

We will prove that

g. $d_j \not\in \tilde{A}$.

Assume the contrary, $d_j \in \tilde{A}$. By construction of $\mathcal{R}(\tilde{\mathcal{D}})$, we conclude that there exists a default d_n such that

g1. $default(d_n, \neg l, [\Lambda]) \in \mathcal{D}$,

g2. $\Lambda \subseteq \tilde{A}$, and

g3. $prefer(d_n, d_j) \in \tilde{\mathcal{D}}$.

It follows from (f3) and (g3) and the fact that the preference order in $\tilde{\mathcal{D}}$ is well-ordered that

g3. $prefer(d_n, d) \in \tilde{\mathcal{D}}$ for $d \in \Delta_l^+$.

This, together with (g1) and (g2), implies that

g4. $d \in \tilde{A}$ for $d \in \Delta_l^+$.

which, in turn, implies that there exists no rule with the head l in $\mathcal{R}(\tilde{\mathcal{D}})$ whose body is satisfied by $\tilde{A}$, i.e., $l \not\in \tilde{A}$. This contradiction proves (g).

We now prove that

h. $d_j \not\in A$.

Assume that (h) does not hold, i.e.,

h1. $d_j \in A$.

Using the definition of A and the fact that A and $\tilde{A}$ coincide on σ-literals we can easily check that there is d_i such that

h2. $default(d_i, \neg l, [\Gamma]) \in \tilde{\mathcal{D}}$

h3. $prefer(d_i, d_j) \in \tilde{\mathcal{D}}$

h4. $\Gamma \subseteq \tilde{A}$

From construction of $\mathcal{R}(\tilde{\mathcal{D}})$ and conditions (h2), (h3) we have that

h5. $d_j \leftarrow \Gamma, not\ d_i \in \mathcal{R}(\tilde{\mathcal{D}})$

First assume that

h6: $d_i \notin \tilde{A}$

Then, from (h4), (h5), and the fact that $\tilde{A}$ is an answer set of $\mathcal{R}(\tilde{\mathcal{D}})$ we conclude that $d_j \in \tilde{A}$ which contradicts (g). Therefore,

h7. $d_i \in \tilde{A}$

This implies that there is a default d_k of the form $default(d_k, l, [\Delta]) \in \tilde{\mathcal{D}}$ such that

h8. $\Delta \subseteq \tilde{A}$

h9. $prefer(d_k, d_i) \in \tilde{\mathcal{D}}$

Since the preference relation in $\tilde{\mathcal{D}}$ is total from (h3) and (h9) we conclude that

h10. $prefer(d_k, d_j) \in \tilde{\mathcal{D}}$

which contradicts d_j being the minimal element of Δ_l^+. This contradiction proves (h).

Recall that $head(d_j) = l$ and let Θ be its body. Since d_j is best for l in A we have that

k. $\Theta \subseteq A$

Since $l \in A$ and A is consistent, $\neg l \notin A$. This, together with (h), implies that

l. $l \leftarrow \Theta \in (\mathcal{R}(\mathcal{D}))^A$.

Since $l \notin B$ and B is closed under the rules of $(\mathcal{R}(\mathcal{D}))^A$, from (l) we can conclude that there exists a literal $l' \in \Theta$ such that $l' \notin B$. This, together with (k), implies that

m. $l' \in A \setminus B$.

Since $\mathcal{D}$ is normalized and hence hierarchical, from condition 5 of Definition 5.3 we have that $rank(l') < rank(l)$. This, together with (m), contradicts with (f) which implies that $A|_l \setminus B|_l = \emptyset$.

(ii) $A|_l = B|_l$. Since $B \subset A$, there exists $d_j \in A \setminus B$. By the construction of A,

n. there exists a default $d_i \in \mathcal{D}$ of the form $default(d_i, \neg head(d_j), [\Gamma])$ such that

n1. $prefer(d_i, d_j) \in \mathcal{D}$, $d_i \notin A$ and

n2. $\Gamma \subseteq A$.

(n1), together with the definition of $(\mathcal{R}(\mathcal{D}))^A$ implies that

n3. $d_j \leftarrow \Gamma \in (\mathcal{R}(\mathcal{D}))^A$.

This, together with the assumption that B is closed under the rules of $(\mathcal{R}(\mathcal{D}))^A$ and $B|_l = A|_l$, implies that $d_j \in B$ which contradicts the selection of d_j.

We showed that no proper subset B of A is closed under the rules of $(\mathcal{R}(\mathcal{D}))^A$ and hence A is an answer set of $\mathcal{R}(\mathcal{D})$. □

The next lemma is the reverse of Lemma 6.5.

Lemma 6.6 Let $\mathcal{D}$ be a static and normalized domain description with a basic preference relation and A be an answer set of $\mathcal{R}(\mathcal{D})$. Then, there exists a completion $\tilde{\mathcal{D}}$ of $\mathcal{D}$ and an answer set $\tilde{A}$ of $\mathcal{R}(\tilde{\mathcal{D}})$ such that $\tilde{A}|_l = A|_l$.

Proof. We start with introducing some notation. Let P be a binary relation. By P^* we denote the transitive closure of P. For a σ-literal l, we define,

$\Delta_l^+ = \{d : default(d, l, [\Gamma]) \in \mathcal{D},\ \Gamma \subseteq A\}$,

$\Delta_l^- = \{d : default(d, \neg l, [\Gamma]) \in \mathcal{D},\ \Gamma \subseteq A\}$,

$\Delta_l = \Delta_l^+ \cup \Delta_l^-$, and

$\Delta^l = \{d \in \mathcal{D} : head(d) \in \{l, \neg l\}\}$

By $<_l$ we denote the order induced on Δ_l by the preference relation of $\mathcal{D}$.

In our further discussion we need the following well known result:

(*) if P is a well-founded strict partial order then there exists a well-founded total order containing P.

Now we start our construction of $\tilde{\mathcal{D}}$. Notice that if $l \in A$ then, since $<_l$ is well-founded, it is easy to prove that there exists a default $d \in \Delta_l^+$ which is a minimal element in Δ_l. Let us denote such a default by $d(l)$.

Let

$X_1(l) = \{prefer(d(l), d) : d \in \Delta_l^-\}$.

$X_2(l) = \{prefer(d_1, d_2) : prefer(d_1, d_2) \in \mathcal{D}, d_, d_2 \in \Delta^l\}$.

For every atom $p \in lit(\sigma)$ we define the set X_p as follows:

$$X_p \quad = \begin{cases} (X_1(p) \cup X_2(p))^* & \text{if } p \in A \\ \\ (X_1(\neg p) \cup X_2(p))^* & \text{if } \neg p \in A \\ \\ X_2(p) & \text{otherwise} \end{cases}$$

It is easy to see that X_p is a well-founded, strict partial order on Δ^p. Let Y_p be a well-founded, total order on Δ^p which extends X_p (existence of Y_p is ensured by (*)). Obviously, $\bigcup_{p \in atom(\sigma)} Y_p$ is a well-founded, strict partial order on the set of defaults of $\mathcal{D}$ which extends the preference relation in $\mathcal{D}$.

Let Y be a well-founded, total order on the set of defaults of $\mathcal{D}$ which extends

$\bigcup_{p \in atom(\sigma)} Y_p$.

Let

$\tilde{\mathcal{D}} = \mathcal{D} \cup Y$.

It is easy to see that $\tilde{\mathcal{D}}$ is a consistent completion of $\mathcal{D}$.

Now we will construct an answer set $\tilde{A}$ of $\mathcal{R}(\tilde{\mathcal{D}})$ such that $\tilde{A}|_l = A|_l$.

$$\begin{aligned} U_i = \quad & \{l : l \in lit(\sigma) \cap heads(\mathcal{R}(\tilde{\mathcal{D}})) \text{ s.t. } rank(l) < i\} \cup \\ & \{d \in heads(\mathcal{R}(\tilde{\mathcal{D}})) : rank(head(d)) < i\}. \end{aligned}$$

The sequence $U = U_0, U_1, \ldots$ is monotone and continuous. Using the property of the *rank* function from the definition of hierarchical domain description it is not difficult to check that each U_i is a splitting set of $\mathcal{R}(\tilde{\mathcal{D}})$ and that $\bigcup U_i$ is equal to the set of all literals occurring in $\mathcal{R}(\tilde{\mathcal{D}})$. Hence, U is a splitting sequence of $\mathcal{R}(\tilde{\mathcal{D}})$.

Let T_i be a collection of all the rules from $\mathcal{R}(\tilde{\mathcal{D}})$ whose heads belong to U_i and let $A_i = A \cap U_i$.

We define a sequence $\tilde{A}_0, \tilde{A}_1, \ldots$ such that

1a. $\tilde{A}_i$ is an answer set of T_i.

1b. $\tilde{A}_i|_l = A_i|_l$

(i) Let $\tilde{A}_0 = A_0$

Since both sets are empty conditions (1a) and (1b) are satisfied.

(ii) assume that conditions (1a) and (1b) are satisfied by the already constructed set $\tilde{A}_i$ Let T be the result of partial evaluation of the program T_{i+1} with respect to the set $\tilde{A}_i$.

T will consists of the rules

(r2) $l \leftarrow not\ d, not\ \neg l$ where l is a σ-literal.

and

(r1) $d_2 \leftarrow not\ d_1$.

Using the argument from Lemma 5.6 we can show that the program consisting of the rules of T of the form (r1) contains no negative odd cycles and therefore is consistent. Let S_0 be an answer set of this program and $S_1 = (A_{i+1} \setminus A_i)|_l$. We will show that

2. $S = S_0 \cup S_1$

is an answer set of T. By the splitting set theorem it suffices to show that S_1 is an answer set of the partial evaluation of rules of the type (r2) from T with respect to S_0. We denote this partial evaluation by π. This, in turn, is true iff

3. $S_1 = \pi^{S_1}$.

To prove (3) let us first assume that

4. $l \in S_1$.

This implies that $l \in A$ and hence $\Delta_l \neq \emptyset$. Consider $d \in \Delta_l$ which is minimal with respect to well-ordering induced on Δ_l by the preference relation from $\tilde{\mathcal{D}}$. It is easy to check that, since $l \in A$, $head(d) = l$ and $body(d) \subseteq A$. Since $\mathcal{D}$ is hierarchical we have that $body(d) \subseteq A_i$, and hence, by inductive hypothesis,

4a. $body(d) \subseteq \tilde{A}_i$.

Since d is minimal, by construction of $\tilde{\mathcal{D}}$ we have that there is no rule in T with d in the head. Hence,

4b. $d \notin S_0$.

By construction of $\mathcal{R}(\tilde{\mathcal{D}})$ and conditions (4a) and (4b) we have that

4c. $l \leftarrow not\ \neg l \ \in \pi$.

Since $l \in A$ and A is consistent we conclude that $\neg l \notin A_{i+1}$. Therefore, $\neg l \notin S_1$. Hence,

4d. $l \in \pi^{S_1}$

Suppose now that

5. $l \in \pi^{S_1}$.

This implies that there is d and $\Gamma \subseteq A$ such that

$default(d, l, \Gamma) \in \mathcal{D}$.

From (4d) we have that $\neg l \notin A$ and hence, by Lemma 6.4 we conclude that $l \in A$. Therefore $l \in S_1$ which concludes the proof of (3).

By the splitting set theorem, $\tilde{A}_{i+1} = \tilde{A}_i \cup S$ is an answer set of T_{i+1}. Obviously, $\tilde{A}_{i+1}$ also satisfies condition (1b). Now let

$\tilde{A} = \bigcup \tilde{A}_i$.

From construction we have that $\tilde{A}|_l = A|_l$. Using the splitting sequence theorem it is easy to check that $\tilde{A}$ is an answer set of $\mathcal{R}(\tilde{\mathcal{D}})$. □

Lemma 6.7 Let $\mathcal{D}$ be a static and normalized domain description and A be an answer set of $\mathcal{R}(\mathcal{D})$. Then, $A|_l$ is an answer set of $\mathcal{B}(\mathcal{D})$.

Proof. Since $\mathcal{D}$ is normalized, A is consistent, it suffices to prove that $A|_l$ is a minimal set of literals closed under the rules of $\mathcal{B}(\mathcal{D})^{A|_l}$.

Let

a. $l \leftarrow \Gamma \in \mathcal{B}(\mathcal{D})^{A|_l}$

and

b. $\Gamma \subseteq A|_l$.

By construction of $\mathcal{B}(\mathcal{D})$ and of $\mathcal{B}(\mathcal{D})^{A|_l}$, (a) implies that there exists a default $d \in \mathcal{D}$ such that

c. $default(d, l, [\Gamma]) \in \mathcal{D}$ and $\neg l \notin A|_l$.

Since A is an answer set of $\mathcal{R}(\mathcal{D})$, from (c), (b), and Lemma 6.4, we can conclude that $l \in A$ and hence $l \in A|_l$ which proves that

d. $A|_l$ is closed under the rules of $\mathcal{B}(\mathcal{D})^{A|_l}$.

We now prove the minimality of $A|_l$.

Assume that there exists a set $B \subset A|_l$ which is closed under the rules of $\mathcal{B}(\mathcal{D})^{A|_l}$. We will prove that the set of literals

$$C = B \cup \{d_i : d_i \in A\}$$

is closed under the rules of $(\mathcal{R}(\mathcal{D}))^A$.

Since C contains every d_i in A, $C \subset A$, and A is an answer set of $(\mathcal{R}(\mathcal{D}))^A$, we have that

e. C is closed under the rules of the form (2) of $(\mathcal{R}(\mathcal{D}))^A$.

Let r be a rule of the form (1) of $(\mathcal{R}(\mathcal{D}))^A$ whose body is satisfied by C, i.e.,

f1. $l \leftarrow \Gamma \in (\mathcal{R}(\mathcal{D}))^A$ and

f2. $\Gamma \subseteq C$.

By construction of $(\mathcal{R}(\mathcal{D}))^A$, (f1) implies that there exists a default d such that

g1. $default(d, l, [\Gamma]) \in \mathcal{D}$, and

g2. $\neg l \notin A$.

By definition of $\mathcal{B}(\mathcal{D})$ and $\mathcal{B}(\mathcal{D})^{A|_l}$, and from (g1) and (g2) we conclude that

h. $l \leftarrow \Gamma$ is a rule of $\mathcal{B}(\mathcal{D})^{A|_l}$.

which, together with (f2) and the assumption that B is closed under rules of $(\mathcal{B}(\mathcal{D}))^{A|_l}$ implies that $l \in B$ and hence $l \in C$ which, in turn, implies that

j. C is closed under the rule of the form (1) of $(\mathcal{R}(\mathcal{D}))^A$.

From (e) and (j) we can conclude that C is closed under the rules of $(\mathcal{R}(\mathcal{D}))^A$ which together with $C \subset A$ contradicts the fact that A is an answer set of $\mathcal{R}(\mathcal{D})$. This, together with (d), implies that $A|_l$ is an answer set of $\mathcal{B}(\mathcal{D})$. □

Lemma 6.8 Let $\mathcal{D}$ be a static and normalized domain description with a well-ordered preference order P and let A be an answer set of $\mathcal{R}(\mathcal{D})$. Then, $A|_l$ is a preferred answer set of $\Pi(\mathcal{D})$.

Proof. Lemma 6.7 shows that $A|_l$ is an answer set of $\mathcal{B}(\mathcal{D})$. We need to show that $A|_l = Z$ where $Z = C_{<_\mathcal{D}}(\mathcal{B}(\mathcal{D}))$ and $C_{<_\mathcal{D}}(\mathcal{B}(\mathcal{D}))$ is defined as in Definition 6.1.

Let $d_0, d_1, \ldots$ be the sequence of defaults in $\mathcal{D}$, ordered by P.

Notice that

$l \leftarrow not\ \neg l \in^{A|_l} \mathcal{B}(\mathcal{D})$

iff there exists a default d such that

0a. $default(d, l, [\Gamma]) \in \mathcal{D}$, and

0b. $\Gamma \subseteq A|_l$.

(i) We first prove that $Z \subseteq A|_l$. Let

a. $l \in Z$.

This implies that there exists a default $d_i \in \mathcal{D}$ such that

b1. d_i satisfies (0a) and (0b), and

b2. the rule $l \leftarrow not\ l$ is not defeated by S_{i-1}. (see Definition 6.1).

Let i be the minimal integer such that

c. d_i satisfies (b1) and (b2).

From (c) and (b2) and the definition of Z, we can conclude that

d. there exists no $j < i$ and $\Delta \subseteq A|_l$ such that $default(d_j, \neg l, [\Delta]) \in \mathcal{D}$.

By construction of $\mathcal{R}(\mathcal{D})$ and (d), we conclude that there exists no rule of $\mathcal{R}(\mathcal{D})$ with the head d_i whose body is satisfied by A, which implies that

e. $d_i \notin A$.

Furthermore, for every default d_k such that $i < k$ and $default(d_k, \neg l, [\Delta]) \in \mathcal{D}$, it follows from (b1), (e), and the construction of $(\mathcal{R}(\mathcal{D}))^A$ that

f. $d_k \in A$.

This implies that

g. there exists no rule of $(\mathcal{R}(\mathcal{D}))^A$ with the head $\neg l$ whose body is satisfied by A.

This implies that

h. $\neg l \notin A$.

From (h), (b1), and Lemma 6.4, we can conclude that $l \in A$ and hence $l \in A|_l$ which, together with (a) proves that

j. $Z \subseteq A|_l$.

(ii) We now prove that $A|_l \subseteq Z$. Let

k. $l \in A|_l$.

Since A is an answer set of $\mathcal{R}(\mathcal{D})$, there exists a default d such that

l. $default(d, l, [\Gamma]) \in \mathcal{D}$,

m. $\Gamma \subseteq A$, and $\neg l \notin A$.

which implies that $l \leftarrow not\ \neg l$ is a rule of ${}^{A|_l}\mathcal{B}(\mathcal{D})$. This indicates that

n1. $l \in Z$ or

n2. $\neg l \in Z$.

If (n2) holds, then, by (j), $\neg l \in A|_l$, which, together with $l \in A$, contradicts the fact that $A|_l$ is consistent. Hence, (n1) holds, i.e., $l \in Z$ which, together with (k) entails

o. $A|_l \subseteq Z$.

The lemma is proved by (o) and (j). □

We now prove the reverse of Lemma 6.8.

Lemma 6.9 Let $\mathcal{D}$ be a static and normalized domain description with a well-ordered preference order P. Let A be a preferred answer set of $\Pi(\mathcal{D})$. Then, there exists an answer set B of $\mathcal{R}(\mathcal{D})$ such that $B|_l = A$.

Proof. First, notice that since $\mathcal{D}$ is normalized, $\mathcal{R}(\mathcal{D})$ is consistent and therefore, by Lemma 6.8, $\mathcal{B}(\mathcal{D})$ is consistent. Thus, A is consistent.

Let $d_0, d_1, \ldots$ be the sequence of defaults in $\mathcal{D}$, ordered by P. We define a sequence of sets of literals $A_{i=1}^{\infty}$ as follows.

$$B_0 = B$$

$$B_{n+1} = \begin{cases} B_n \cup \{d_{n+1}\} & \text{if there exists } i \leq n \text{ s.t.} \\ & \text{(0a) } default(d_i, \neg head(d_{n+1}), [\Gamma]) \in \mathcal{D}, \\ & \text{(0b) } \Gamma \subseteq A_n, \text{ and} \\ & \text{(0c) } d_i \notin A_n. \\ & \\ B_n & \text{otherwise} \end{cases}$$

Let $B = \cup_{i=0}^{\infty} B_i$. Obviously B is consistent and $B|_l = A$. We prove that B is an answer set of $\mathcal{R}(\mathcal{D})$, i.e., B is a minimal set of literals closed under the rules of $(\mathcal{R}(\mathcal{D}))^B$. By definition, $(\mathcal{R}(\mathcal{D}))^B$ consists of the following rules:

$$
(\mathcal{R}(\mathcal{D}))^B = \begin{cases}
l \quad \leftarrow \quad \Gamma. & (1) \\
\\
\qquad\quad \text{if there is } d \text{ s.t.} \\
\qquad\quad \text{(1a) } default(d, l, [\Gamma]) \in \mathcal{D}, \\
\qquad\quad \text{(1b) } d \notin B, \text{ and } \neg l \notin B \\
\\
d_2 \quad \leftarrow \quad \Gamma. & (2) \\
\\
\qquad\quad \text{if there is } d_1 \text{ s.t.} \\
\qquad\quad \text{(2a) } default(d_1, l, [\Gamma]) \in \mathcal{D}, \\
\qquad\quad \text{(2b) } prefer(d_1, d_2) \in \mathcal{D}, \\
\qquad\quad \text{(2c) } head(d_2) = \neg l, \text{ and} \\
\qquad\quad \text{(2d) } d_1 \notin B.
\end{cases}
$$

Let r be a rule of $(\mathcal{R}(\mathcal{D}))^B$ whose body is satisfied by B, i.e.,

a. $\Gamma \subseteq B$.

We consider two cases:

(i) r is of the form (1).

By the construction of $\mathcal{B}(\mathcal{D})$ we have that

b. $l \leftarrow \Gamma, not\ \neg l \in \mathcal{B}(\mathcal{D})$.

From $B|_l = A$, (a), and (1b), we conclude that

c. $\Gamma \subseteq A$ and $\neg l \notin A$.

Since A is an answer set of $\mathcal{B}(\mathcal{D})$, from (b) and (c) we conclude that $l \in A$ and hence, $l \in B$, which proves that

d. B is closed under the rules of the form (1) of $(\mathcal{R}(\mathcal{D}))^B$.

(ii) r is a rule of form (2).

By construction of B and from (a) and (2a)-(2d), we can conclude that $d_2 \in B$ which implies that

e. B is closed under the rules of the form (2) of $(\mathcal{R}(\mathcal{D}))^B$.

It follows from (e) and (d) that

f. B is closed under the rules of $(\mathcal{R}(\mathcal{D}))^B$.

We now prove the minimality of B.

Assume that there exists a set of literals $C \subset B$ and C is closed under the rules of $(\mathcal{R}(\mathcal{D}))^B$. We will prove that

g. $C|_l$ is closed under the rules of $\mathcal{B}(\mathcal{D})^A$.

Let r be a rule of $\mathcal{B}(\mathcal{D})^A$ whose body is satisfied by $C|_l$, i.e., r is of the form

h1. $l \leftarrow \Gamma \in (\mathcal{B}(\mathcal{D}))^A$, and

h2. $\Gamma \subseteq C|_l$.

By construction of $\mathcal{B}(\mathcal{D})^A$, we conclude that there exists a default d_i in $\mathcal{D}$ such that

j1. $default(d_i, l, [\Gamma]) \in \mathcal{D}$, and

j2. $\neg l \notin A$.

(j1) and (h2) imply that the rule $l \leftarrow not\ \neg l$ belongs to ${}^A\mathcal{B}(\mathcal{D})$ which, together with (j2) and the assumption that A is a preferred answer set of $\Pi(\mathcal{D})$, implies that $l \in A$.

We will prove that

l. $d_i \notin B$.

Assume the contrary, i.e.,

m. $d_i \in B$.

By the construction of B, there exists $j < i$ such that

n1. $default(d_j, \neg l, [\Delta]) \in \mathcal{D}$,

n2. $\Delta \subseteq B$, and

n3. $d_j \notin B$.

From (n1) and (n2) and the construction of ${}^A\mathcal{B}(\mathcal{D})$, we can conclude that

p. $\neg l \leftarrow not\ l$ is a rule of ${}^A\mathcal{B}(\mathcal{D})$.

From $l \in A$, the fact that A is a preferred answer set of $\Pi(\mathcal{D})$, and (p), we can conclude that there exists a $k < j$ such that

q1. $default(d_k, l, [\Theta]) \in \mathcal{D}$,

q2. $\Theta \subseteq A$, and

q3. for every o, $o < k$, if $default(d_o, \neg l, [\Lambda]) \in \mathcal{D}$, then $\Lambda \not\subseteq A$.

From (q3) and the definition of $\mathcal{R}(\mathcal{D})^B$ we have that

r. $d_k \notin B$.

From (r), (q1), (q2), and the construction of B we have that

s. $d_j \in A_{j+1} \subseteq B$

which contradicts with (n3), i.e., we have proved (l).

It follows from (j1), (j2), and (l) that $l \leftarrow \Gamma \in (\mathcal{R}(\mathcal{D}))^B$ which, together with the assumption that C is closed under the rules of $(\mathcal{R}(\mathcal{D}))^B$ and $\Gamma \subseteq C$, implies $l \in C$, and hence, $l \in C|_l$ which proves (g).

Since A is an answer set of $\mathcal{B}(\mathcal{D})$, from (g) we can conclude that $C|_l = A$, which, together with the assumption that $C \subset B$, implies that there exists some $d_i \in \mathcal{D}$ such that

t. $d_i \in B \setminus C$.

By the construction of B, (t) implies that there exists a $j < i$ such that

u1. $default(d_j, \neg l, [\Delta]) \in \mathcal{D}$,

u2. $\Delta \subseteq B$, and

u3. $d_j \notin B$.

Since $j < i$, by the ordering P, we conclude that $prefer(d_j, d_i) \in \mathcal{D}$. This, together with (u1) and (u3), implies that

v. $d_i \leftarrow \Delta$ is a rule of $(\mathcal{R}(\mathcal{D}))^B$.

It follows from (u2), (v), and the assumption that C is closed under the rule of $(\mathcal{R}(\mathcal{D}))^B$ that $d_i \in C$ which contradicts with (t). In other words, B is a minimal set of literals which is closed under $(\mathcal{R}(\mathcal{D}))^B$, i.e., B is an answer set of $\mathcal{R}(\mathcal{D})$. □

We are now ready to prove the Theorem 6.1.

Proof of Theorem 6.1. Let $\mathcal{D}_N$ be the normalization of a static domain description $\mathcal{D}$. By Lemma 6.2, $\mathcal{D} \models holds\_by\_default(l)$ iff

a. $l \in U(\mathcal{D})$ or $\mathcal{R}(\mathcal{D}_N) \models l$,

and by Lemma 6.3, $\Pi(\mathcal{D}) \mid\!\sim l$ iff

b. $l \in U(\mathcal{D})$ or $\Pi(\mathcal{D}_N) \mid\!\sim l$.

By Lemmas 6.6-6.8, we have that

c. $\mathcal{R}(\mathcal{D}_N) \models l$ iff $\Pi(\mathcal{D}_N) \mid\!\sim l$.

The conclusion of theorem 6.1 follows immediately from (a), (b), and (c). □

The theorem 6.1 can be used to better understand properties of both formalizations. It implies, for instance, that queries to Brewka's prioritized programs corresponding to domain descriptions of $\mathcal{L}_0$ can be answered by the SLG inference engine. It can also be used for a simple proof of the fact that static, hierarchical domain descriptions with basic preference relations are monotonic with respect to $prefer$, i.e. for any such $\mathcal{D}_1$ and $\mathcal{D}_2$ with preference relations P_1 and P_2 such that $P_1 \subseteq P_2$, if $\mathcal{D}_1 \models l$ then $\mathcal{D}_2 \models l$.

7 Conclusions

In this paper we

• introduced a language $\mathcal{L}(\sigma)$ capable of expressing strict rules, defaults with exceptions, and the preference relation between defaults;

• gave a collection of axioms, $\mathcal{P}$, defining the entailment relation between domain descriptions of $\mathcal{L}(\sigma)$ and queries of the form $holds(l)$ and $holds\_by\_default(l)$;

• demonstrated, by way of examples, that the language and the entailment relation is capable of expressing rather complex forms of reasoning with prioritized defaults;

• gave sufficient conditions for consistency of domain descriptions;

• described a class of domain descriptions for which our treatment of prioritized defaults coincides with that suggested by G. Brewka in [6].

This work can be extended in several directions. First, the results presented in the paper can be generalized to much broader classes of theories of $\mathcal{L}$. We also plan a more systematic study of the class of logic programs defined by $\mathcal{P}$ (i.e., programs of the form $\mathcal{P} \cup \mathcal{D}$). It may be interesting and useful to check if cautious monotony [16] or other general properties of defeasible inference ([22, 11]) hold for this class of programs. Another interesting class of questions is related to investigating the relationship between various versions of $\mathcal{P}$. Under what conditions on $\mathcal{D}$, for instance, we can guarantee that addition of various axioms from section 4 do not change the set of conclusions. When such additions preserve conclusions of the revised theory? Finally, we want to see if a better language can be obtained by removing from it the notion of $conflict$. In the current language the statement $prefer(d_1, d_2)$ stops the application of default d_2 if defaults d_1 and d_2 are in conflict with each other and the default d_1 is applicable. It may be more convenient to make $prefer(d_1, d_2)$ simply mean that d_2 is stoped if d_1 is applicable. More experience with both languages is needed to make a justified design decision. We hope that answers to these and similar questions will shed new light on representation and reasoning with prioritized defaults.

Acknowledgment

We are grateful to Gerhard Brewka for an illuminating discussion on reasoning with prioritized defaults. We also would like to thank Alfredo Gabaldon for useful discussions and help with running our examples on SLG. Special thanks also to Vladik Kreinovich who helped us to better understand the use of priorities in the utility theory and whose remarks helped us to better understand our own work.

References

[1] Baader, F. and Hollunder, B,: Priorities on Defaults with Prerequisite and their Application in Treating Specificity in Terminological Default Logic, Journal of Automated Reasoning, 15:41–68, 1995.

[2] Baral, C. and Gelfond M.: Logic Programming and Knowledge Representation, Journal of Logic Programming, 19,20: 73–148, 1994.

[3] Brass, S. and Dix, J.: A Characterization of the Stable Semantics by Partial Evaluation, Proc. of the 10th Workshop on Logic Programming, Zuerich, Oct. 1994, 1994.

[4] Brewka, G.: Reasoning about Priorities in Default Logic, Proc. AAAI-94, Seattle, 1994

[5] Brewka, G.: Adding Priorities and Specificity to Default Logic, Proc. JELIA 94, Springer LNAI 838, 247–260, 1994

[6] Brewka, G.: Preferred Answer Sets, Proc. ILPS'97 Postconference Workshop, 76–88, 1997.

[7] Covington M.A., Nute D., and Vellino A.: Prolog Programming in Depth, Prentice Hall, NJ, 1997.

[8] Chen, W. and Warren, D.S.: Query Evaluation under the Well-Founded Semantics, The Twelfth ACM Symposium on Principles of Database System, 1993.

[9] Chen, W.: Extending Prolog with Nonmonotonic Reasoning, Journal of LP, 169–183, 1996.

[10] Delgrande , J.P., Schaub, T.H.: Compiling Reasoning with and about Preferences into Default Logic, IJCAi'97, (1997).

[11] Dix, J.: Classifying Semantics of Logics Programs. In Proc. of the International Workshop in Logic Programming and Nonmonotonic Reasoning, 166–180, Washington, DC, 1991.

[12] Dung, P.M.: On the Relations Between Stable and Well-Founded Semantics of Logic Programming, Theoretical Computer Science 105:7-25 (1992).

[13] Dung, P.M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning and Logic Programming and N-person game. AI (77) 2:321–357 (1995).

[14] Fages, F.: Consistency of Clark's Completion and Existence of Stable Models, Technical Report 90-15, Ecole Normale Superieure, 1990.

[15] Fishburn, P.C.: Nonlinear Preference and Utility Theory (Johns Hopkins University Press, Baltimore, 1988).

[16] Gabbay, D.: Theoretical Foundation for Nonmonotonic Reasoning in Experts System. In K. Apt, editor, Logics and models of Concurrent Systems, 439–457, Springer Verlag, NY, 1985.

[17] Gelfond, M., Gabaldon, A.: From Functional Specifications to Logic Programs, 355–370, Proc. of ILPS'97, 1997.

[18] Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases, New Generation of Computing 365–387, 1991.

[19] Grosof, B.N: Prioritized Conflict Handling for Logic Programs, 197–212, Proc. of ILPS'97, 1997.

[20] Gordon, T.: The Pleadings Game: An Artificial Intelligence Model of Procedural Justice. Ph.D. Dissertation, TU Darmstadt.

[21] Kosheleva, O.M. and Kreinovich, V.Ya.: Algorithm Problems of Nontransitive (SSB) Utilities, Mathematical Social Sciences 21 (1991) 95–100.

[22] Lehnmann, D., Kraus, S., and Magidor, M.: Nonmonotonic Reasoning, Preferential Models and Cumulative Logics, AI (44) 1: 167–207, 1990.

[23] Lifschitz, V., Turner, H.: Splitting a Logic Program, Proc. of ICLP, MIT Press, 1994.

[24] Marek, W. and Truszczynski, M.: Nonmonotonic Logic: Context-Dependent Reasoning, Springer, 1993.

[25] Nelson, D.: Constructible Falsity, JSL 14(1949), 16-26.

[26] Nute, D.: A Decidable Quantified Defeasible Logic. In Prawitz, D., Skyrms, B., and Westerstahl, D. (eds): Logic, Methodology and Philosophy of Science IX. Elsevier Science B.V., 263–284, 1994.

[27] Pearce, D.: A New Logical Characterization of Stable Models and Answer Sets, NMELP'96, Springer, 57–70, 1997.

[28] Prakken, H. and Sartor, G,: On the relation between legal language and legal argument: assumptions, applicability and dynamic priorities. Proc. of the Fifth International Conference on AI and Law, Maryland, College Park, MD USA, 1–10, 1995.

[29] Prakken, H. and Sartor, G.: Argument-based extended logic programming with defeasible priorities. Journal of applied non-classical logics, 1,2 (7), 25–77, 1997.

[30] Reiter, R.: On closed world data bases. In H. Gallaire and J. Minker, editors, Logic and data bases, 55–76, 1978.

[31] Reiter R.: A Logic for Default Reasoning in Readings in Nonmonotonic Reasoning, Edited by M. L. Ginsberg, Morgan Kaufmann Publishers, Inc., Los Altos, California (1987) 68–93

[32] Zhang, Y. and Foo , N.Y.: Answer Sets for Prioritized Logic Programs, 69–84, Proc. of ILPS'97, 1997.