STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

SEQUENT CALCULUS IN COMPUTING DEFAULT EXTENSIONS

MIHAIELA LUPEA

Abstract. Justified and constrained default logics are the versions of default logic that have the property of semi-monotonicity. Based on this property, in this paper we present an iterative approach of the problem of computing the justified and constrained extensions of a propositional default theory. The sequent calculus and its complementary system, the antisequent calculus, are used to check the cautious applicability condition for defaults.

Keywords: default logic, nonmonotonic reasoning, theorem proving.

1. Introduction

The nonmonotonic reasoning is an important part of human reasoning and represents the process of drawing conclusions from incomplete information. Adding new facts may later invalidate these conclusions which are only plausible, not necessarily true. Default logics (classical, justified, constrained, rational) formalize *default reasoning*, a special case of nonmonotonic reasoning. These logical systems overcome the lack of information by making default assumption about a situation. The *defaults* are nonmonotonic inference rules used to model laws which are true with a few exceptions.

A *default theory* ([8]) $\Delta = (\mathcal{D}, \mathcal{W})$ consists of a set $\mathcal{W}$ (the *facts*) of consistent formulas of first order logic and a set $\mathcal{D}$ of *default rules*. $\mathcal{W}$ represents absolute (sometimes incomplete) knowledge about the world while $\mathcal{D}$ represents defeasible knowledge.

A *default* has the form $d = \frac{\alpha:\beta_1,\ldots,\beta_m}{\gamma}$, where: α is called *prerequisite*, $\beta_1,\ldots,\beta_m$ are called *justifications* and γ is called *consequent*.

A default $d = \frac{\alpha:\beta_1,\ldots,\beta_m}{\gamma}$ can be applied and thus derive γ if α is deductible (derivable) and it is consistent to assume $\beta_1,\ldots,\beta_m$ (meaning that $\neg\beta_1,\ldots,\neg\beta_m$ can not be derived).

Default extensions contain all the formulas obtained from the set of facts using the classical inference rules and the defaults. The elements of extensions are called

Received by the editors: November 10, 2006.

2000 *Mathematics Subject Classification.* 03B79, 68T15, 68T27, 68T20.

1998 *CR Categories and Descriptors.* I.2[**Artificial Intelligence**]: Logic in artificial intelligence – *default logics, nonmonotonic reasoning, theorem proving*;

nonmonotonic theorems (*beliefs*). The set of defaults used in the construction of an extension is called the *generating default set* for the considered extension.

The versions (classical([8]), justified ([4]), constrained([9]), rational([7]) of default logic, by different applicability conditions, try to provide an appropriate definition of consistency condition for the justifications of the defaults, and thus to obtain many interesting and useful properties for these logics: existence of extensions, semi-monotonicity, commitment to assumptions, cumulativity, regularity.

In the literature there were developed several methods to solve the problem of computing extensions of the versions of default logic, using different approaches.

In the paper [3], a relaxed stratification of a default theory is the primary search-space pruning technique for computing the classical extensions. The semantic tableaux method is adapted to be used as a general or local prover.

Exten([1]) is a system that computes classical, justified and constrained extensions, based on an operational approach and uses pruning techniques for search tree.

Semantic tableaux method is used in [11] to compute classical extensions for a decidable subset of default logic. An uniform approach, based on a modified version of propositional semantic tableaux method, of computing constrained and rational default is presented in [5].

In paper [10], default reasoning is integrated into existing model elimination in order to solve the query-answering problem for constrained and cumulative default logics.

Based on the semi-monotonicity property of justified and constrained default logics, and their relationship, in this paper we propose an iterative approach for generating these two types of default extensions. The sequent calculus and antisequent calculus, as proof systems, are used to check the cautious applicability condition for defaults.

The paper is structured as follows. In section 2 two complementary proof systems for propositional logic, sequent calculus and anti-sequent calculus, are described. Section 3 presents the main aspects of *justified* and *constrained* default logics. Section 4 explains our approach, introducing the theoretical model for computing justified and constrained extensions. Conclusions and future work are outlined in Section 5.

2. Sequent calculus and anti-sequent calculus in propositional logic

This section presents two complementary systems: the sequent calculus and the anti-sequent calculus, used to check the derivability and non-derivability in propositional logic.

The *sequent calculus* method, as an improvement of Gentzen natural deduction system, is a direct and syntactic proof method.

A *sequent* has the form: $U \Rightarrow V$, where U and V are finite sets of propositional formulas. U is called *antecedent* and V is called *succedent.*

A *basic sequent* contains the same formula, A, in both antecedent and succedent: $U, A \Rightarrow V, A$;

Semantics: The sequent $U \Rightarrow V$ is *true* if each model of U is also a model for at least one of the formulas of V. All basic sequents are true, therefore they are the *axioms* of sequent calculus.

The inference rules of sequent calculus are presented in TABLE 1.

This proof method consists in reducing an initial sequent, by successive applications of sequent inference rules, in order to obtain basic sequents.

TABLE 1. Sequent rules

connective	Introduction into antecedent	Introduction into succedent
$\neg$	$(\neg_l)\ \frac{U\Rightarrow V,A}{U,\neg A\Rightarrow V}$	$(\neg_r)\ \frac{U,A\Rightarrow V}{U\Rightarrow V,\neg A}$
$\wedge$	$(\wedge_l)\ \frac{U,A,B\Rightarrow V}{U,A\wedge B,\Rightarrow V}$	$(\wedge_r)\ \frac{U\Rightarrow A,V \quad U\Rightarrow B,V}{U\Rightarrow A\wedge B,V}$
$\vee$	$(\vee_l)\ \frac{U,A\Rightarrow V \quad U,B\Rightarrow V}{U,A\vee B,\Rightarrow V}$	$(\vee_r)\ \frac{U\Rightarrow A,B,V}{U\Rightarrow A\vee B,V}$
$\rightarrow$	$(\rightarrow_l)\ \frac{U\Rightarrow A,V \quad U,B\Rightarrow V}{U,A\rightarrow B\Rightarrow V}$	$(\rightarrow_r)\ \frac{U,A\Rightarrow B,V}{U\Rightarrow A\rightarrow B,V}$

The *derivability* from propositional logic is expressed in sequent calculus as follows: $U1, U2, \ldots, Un \longmapsto V1 \vee V2 \vee \ldots \vee Vm$ if and only if the sequent $U1, U2, \ldots, Un \Rightarrow V1, V2, \ldots, Vm$ is *true*, meaning that from the conjunction of hypothesis at least one of the formulas from succedent can be proved.

The *anti-sequent calculus* for propositional logic was introduced in [2] as the complementary system of sequent calculus.

An *anti-sequent* has the form $U \not\Rightarrow V$, where U, V are finite sets of propositional formulas.

Semantics: $U \not\Rightarrow V$ is *true* if there is a model M of U in which all the formulas of V are false, and M is an *anti-model* for this anti-sequent.

An anti-sequent $U \not\Rightarrow V$ is called a *basic anti-sequent* if all the formulas of U and V are atomic formulas and $U \cap V = \emptyset$. The basic anti-sequents are true and represent the axioms of this system.

The *non-derivability* from propositional logic is expressed in anti-sequent calculus as follows: $U1, U2, \ldots, Un \not\longmapsto V1 \wedge V2 \wedge \ldots \wedge Vm$ if and only if the anti-sequent $U1, U2, \ldots, Un \Rightarrow V1, V2, \ldots, Vm$ is *true*, meaning that from the conjunction of hypothesis none of the formulas from succedent can be proved.

TABLE 2 contains the inference rules of anti-sequent calculus, used to reduce an initial anti-sequent to an axiom that represents a partial anti-model for the initial anti-sequent.

TABLE 2. Anti-sequent rules

Introduction into antecedent	Introduction into consequent
$(\neg^c{}_l)\ \frac{U \not\Rightarrow V, A}{U, \neg A \not\Rightarrow V}$	$(\neg^c{}_r)\ \frac{U, A \not\Rightarrow V}{U \not\Rightarrow V, \neg A}$
$(\wedge^c{}_l)\ \frac{U, A, B \not\Rightarrow V}{U, A \wedge B, \not\Rightarrow V}$	$(\wedge^c{}_{r1})\ \frac{U \not\Rightarrow A, V}{U \not\Rightarrow A \wedge B, V}$ $\mid$ $(\wedge^c{}_{r2})\ \frac{U \not\Rightarrow B, V}{U \not\Rightarrow A \wedge B, V}$
$(\vee^c{}_{l1})\ \frac{U, A \not\Rightarrow V}{U, A \vee B \not\Rightarrow V}$ $\mid$ $(\vee^c{}_{l2})\ \frac{U, B \not\Rightarrow V}{U, A \vee B \not\Rightarrow V}$	$(\vee^c{}_r)\ \frac{U \not\Rightarrow A, B, V}{U \not\Rightarrow A \vee B, V}$
$(\rightarrow^c{}_{l1})\ \frac{U \not\Rightarrow A, V}{U, A \rightarrow B \not\Rightarrow V}$ $\mid$ $(\rightarrow^c{}_{l2})\ \frac{U, B \not\Rightarrow V}{U, A \rightarrow B \not\Rightarrow V}$	$(\rightarrow^c{}_r)\ \frac{U, A \not\Rightarrow B, V}{U \not\Rightarrow A \rightarrow B, V}$

We remark that the difference between TABLE 1 and TABLE 2 consists in splitting the rules with two premisses from sequent calculus into pairs of rules in anti-sequent calculus. Thus the exhaustive search in sequent calculus becomes nondeterminism in anti-sequent calculus and the reduction process is a linear one.

The following theorem shows the complementarity of these two proof systems:

Theorem 2.1 ([2])
The anti-sequent $U \not\Rightarrow V$ is true if and only if the sequent $U \Rightarrow V$ is not true.

3. Justified and Constrained default logics

Justified default logic was introduced by Lukaszewicz ([4]). This version of default logic solves the problem of inconsistencies consequents-justifications using a *support set*, but the inconsistencies justifications-justifications are still not detected. *The existence of justified extensions is guaranted and the property of semi-monotonicity is satisfied.*

In *constrained default logic* ([9]), the assumptions (stored in a set of constraints) from the reasoning process are used to express a global consistency condition for justifications. This logic is strongly regular, *semi-monotonic, commits to assumptions and guarantees the existence of constrained extensions.*

The results from [6] show that default theories can be represented by unitary theories (all the defaults have only one justification, $d = \frac{\alpha:\beta}{\gamma}$) in such a way that extensions (classical, justified, constrained, rational) are preserved. In this paper we will use only unitary default theories and the following notations:

$Prereq(d) = \alpha$, $Justif(d) = \beta$, $Conseq(d) = \gamma$, $Prereq(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} Prereq(d)$,
$Justif(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} Justif(d)$, $Conseq(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} Conseq(d)$,
$Th(X) = \{A | X \vdash A\}$ the classical deductive closure of the set X of formulas.

All versions of default logics were introduced using fixed-point operators. The definitions below are the original ones for justified and constrained default logics.

Definition 3.1([4])
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory. For any pair (S, U) of sets of formulas, let $\Gamma_1(S, U)$ and $\Gamma_2(S, U)$ be the smallest sets of formulas which satisfy:
a) $\mathcal{W} \subseteq \Gamma_1(S, U)$;
b) $\Gamma_1(S, U) = Th(\Gamma_1(S, U))$;
c) For any $\frac{\alpha:\beta}{\gamma}$, if $\alpha \in \Gamma_1(S, U)$ and $\forall \eta \in U \cup \{\beta\}$, $S \cup \{\gamma\} \not\longmapsto \neg\eta$ then $\gamma \in \Gamma_1(S, U)$ and $\beta \in \Gamma_2(S, U)$.
A pair (E, J) of sets of formulas is a *justified extension* of Δ if and only if $E = \Gamma_1(E, J)$ and $J = \Gamma_2(E, J)$.

E is the *actual extension* and J is the *support set*. The applicatibility condition c) permits the detection of inconsistencies consequents-justifications, but the support set may be inconsistent, meaning that defaults with contradictory justifications were applied.

Definition 3.2([9])
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory. For any set T of formulas, let $\Upsilon(T)$ be the pair of the smallest sets (S', T') of formulas which satisfy:
a) $\mathcal{W} \subseteq S' \subseteq T'$;
b) $S' = Th(S')$ and $T' = Th(T')$;
c) For any $\frac{\alpha:\beta}{\gamma}$, if $\alpha \in S'$ and $T \cup \{\gamma\} \not\longmapsto \neg\beta$ then $\gamma \in S'$ and $\beta, \gamma \in T'$.
A pair (E, C) of sets of formulas is a *constrained extension* of Δ if and only if $\Upsilon(C) = (E, C)$.

The *actual extension* (E) is embedded in a consistent *context* (C) containing the facts, the consequents and all the justifications assumed to be true in the construction of E.
These definitions are difficult to be used in the process of constructing extensions and thus equivalent characterizations of extensions were proposed.
The *semi-monotonicity* property is defined as follows:

Definition 3.3
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory and $\mathcal{D}'$ be a set of defaults such that $\mathcal{D} \subseteq \mathcal{D}'$. If (E, C) is a default extension of Δ, then there is a default extension (E', C') of the theory $(\mathcal{D}', \mathcal{W})$, with $E \subseteq E'$ and $C \subseteq C'$.

Justified and constrained default logics are both semi-monotonic ([9]), meaning that new defaults can augment but never destroy previous extensions. The nonmonotonicity in this two versions of default logic is caused by addition of new facts which can invalidate formulas already derived.

The semi-monotonicity property guarantees the existence of justified and constrained extensions. In the worst case $(Th(\mathcal{W}), \emptyset)$ is the only justified extension and $(Th(\mathcal{W}), Th(\mathcal{W}))$ is the only constrained extension of the default theory $(\mathcal{D}, \mathcal{W})$.

Another advantage of this property is from a computational point of view. These two types of default extensions are constructible in a truly iterative way by applying one applicable default rule after another.

The relationship between justified and constrained logics is expressed by the following results from ([5]):
- A constrained extension is a justified extension, satisfying the property that all justifications of the applied defaults are consistent with the actual extension.
- Each constrained extension is included in at least one justified extension.

4. Computing justified and constrained default extensions

This section presents an uniform operational approach for computing justified and constrained extensions. It is inspired from the operational semantics ([5]) which characterizes these two types of extensions. Due to the smi-monotonicity we define the problem of computing the justified and constrained extension as an iterative process, applying the applicable defaults one by one.

Definition 4.1
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory.

- A triple of the form $\langle E_p, J_p, D_p \rangle$ is called a *j-structure* if $D_p \subseteq \mathcal{D}$, $E_p = \mathcal{W} \cup Conseq(D_p)$, and $J_p = Justif(D_p)$.
- A triple of the form $\langle E_p, C_p, D_p \rangle$ is called a *c-structure* if $D_p \subseteq \mathcal{D}$, $E_p = \mathcal{W} \cup Conseq(D_p)$, and $C_p = \mathcal{W} \cup Conseq(D_p) \cup Justif(D_p)$.

The applicability conditions for defaults in justified and constrained default logics can be expressed by the following definition:

Definition 4.2

- The default $d = \frac{\alpha:\beta}{\gamma}$ is *j-applicable with respect to* $\langle E_p, J_p, D_p \rangle$ if:
  - the sequent $E_p \Rightarrow \alpha$ is true;
  - $\forall \eta \in J_p \cup \{\beta\}$ the anti-sequent $E_p, \gamma \not\Rightarrow \neg\eta$ is true.
- The default d is *c-applicable with respect to* $\langle E_p, C_p, D_p \rangle$ if:
  - the sequent $E_p \Rightarrow \alpha$ is true;
  - the anti-sequent $C_p, \gamma \not\Rightarrow \neg\beta$ is true.

The sequent calculus is used to express the derivability of the prerequisite, while the fact that the justification is believed (its negation is not derivable) is expressed using the anti-sequent calculus.

Definition 4.3
To a default $d = \frac{\alpha:\beta}{\gamma}$ we assign a mapping d^j from the set of j-structures into the set of j-structures as follows:

- $d^j(\langle E_p, J_p, D_p\rangle) = \langle E_p \cup \{\gamma\}, J_p \cup \{\beta\}, D_p \cup \{d\}\rangle$ if d is j-applicable wrt $\langle E_p, J_p, D_p\rangle$;
- $d^j(\langle E_p, J_p, D_p\rangle) = \langle E_p, J_p, D_p\rangle$ otherwise.

To a default $d = \frac{\alpha:\beta}{\gamma}$ we assign a mapping d^c from the set of c-structures into the set of c-structures as follows:

- $d^j(\langle E_p, C_p, D_p\rangle) = \langle E_p \cup \{\gamma\}, C_p \cup \{\beta, \gamma\}, D_p \cup \{d\}\rangle$ if d is c-applicable wrt $\langle E_p, J_p, D_p\rangle$;
- $d^j(\langle E_p, C_p, D_p\rangle) = \langle E_p, C_p, D_p\rangle$ otherwise.

These mappings model a *caution application* of the defaults, meaning that once a default is applied it can not lead to inconsistency further in the process of building an extension.

Definition 4.4
Let $\langle E_p, J_p, D_p\rangle$ be a j-structure, $\langle E_p, C_p, D_p\rangle$ be a c-structure and D be a set of defaults.

- $\langle E_p, J_p, D_p\rangle$ is *j-stable* wrt D if $d^j(\langle E_p, J_p, D_p\rangle) = \langle E_p, J_p, D_p\rangle, \forall d \in D$
- $\langle E_p, C_p, D_p\rangle$ is *c-stable* wrt D if $d^c(\langle E_p, C_p, D_p\rangle) = \langle E_p, C_p, D_p\rangle, \forall d \in D$

A stable structure characterizes the end of the reasoning process in which were used all the applicable defaults.

Definition 4.5
A j-structure $\langle E_p^n, J_p^n, D_p^n\rangle$ is *j-accesible* from the j-structure $\langle E_p^0, J_p^0, D_p^0\rangle$ if there is a sequence of defaults $(d_1, d_2, \ldots, d_n)$ and a sequence o j-structures $(\langle E_p^0, J_p^0, D_p^0\rangle, \langle E_p^1, J_p^1, D_p^1\rangle, \ldots, \langle E_p^n, J_p^n, D_p^n\rangle)$ such that $d_i^j(\langle E_p^{i-1}, J_p^{i-1}, D_p^{i-1}\rangle)$ $= \langle E_p^i, J_p^i, D_p^i\rangle$, for $i = 1, \ldots, n$.

Definition 4.6
A c-structure $\langle E_p^n, C_p^n, D_p^n\rangle$ is *c-accesible* from the c-structure $\langle E_p^0, C_p^0, D_p^0\rangle$ if there is a sequence of defaults $(d_1, d_2, \ldots, d_n)$ and a sequence o c-structures $(\langle E_p^0, C_p^0, D_p^0\rangle, \langle E_p^1, C_p^1, D_p^1\rangle, \ldots, \langle E_p^n, C_p^n, D_p^n\rangle)$ such that $d_i^j(\langle E_p^{i-1}, C_p^{i-1}, D_p^{i-1}\rangle)$ $= \langle E_p^i, C_p^i, D_p^i\rangle$, for $i = 1, \ldots, n$.

If a j-structure $\langle E_p, J_p, D_p\rangle$ is j-accessible from $(\mathcal{W}, \emptyset, \emptyset)$ then it corresponds to a partial justified extension. If a c-structure $\langle E_p, C_p, D_p\rangle$ is c-accessible from $(\mathcal{W}, \mathcal{W}, \emptyset)$ then it corresponds to a partial constrained extension.

Theorem 4.1
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory. The j-structure $\langle E_p, J_p, D_p\rangle$ corresponds to the justified extension $(Th(E_p), J_p)$ of Δ, with D_p as generating default set if and only if $\langle E_p, J_p, D_p\rangle$ is j-stable wrt $\mathcal{D}$ and $\langle E_p, J_p, D_p\rangle$ is j-accesible from $(\mathcal{W}, \emptyset, \emptyset)$.

Theorem 4.2
Let $\Delta = (\mathcal{D}, \mathcal{W})$ be a default theory. The c-structure $\langle E_p, C_p, D_p \rangle$ corresponds to the constrained extension $(Th(E_p), Th(C_p))$ of Δ, with D_p as generating default set if and only if $\langle E_p, C_p, D_p \rangle$ is c-stable wrt $\mathcal{D}$ and $\langle E_p, C_p, D_p \rangle$ is c-accesible from $(\mathcal{W}, \mathcal{W}, \emptyset)$.

For lack of space we will not give the proofs of the above theorems.

5. Conclusions and further work

In this paper we defined the problem of computing the justified and constrained extension as an iterative process. Due to the semi-monotonicity property of these two versions of default logics, the applicable defaults have been applied one by one in order to build an extension. The sequent calculus and anti-sequent calculus proof systems were used to check the applicability conditions for defaults.

As further work we will implement an algorithm based on this approach, using a top-down techinque and pruning for efficiency, in order to generate all justified and constrained extensions of a propositional default theory.

References

[1] Antoniou, G., Courtney, A.P., Ernst, J., Williams, M.A., "A System for Computing Constrained Default logic Extensions", Logics in Artificial Intelligence, Lecture Notes in Artificial Intelligence, Vol. 1126, 1996, pp. 237–250.

[2] Bonatti, P., Olivetti, N., "Sequent Calculi for Propositional Nonmonotonic Logics", ACM Trans. Comput. Log., 2002, pp. 226–278.

[3] Cholewinski, P., Marek, W., Truszczynski, M., "Default reasoning system DeReS", Proceedings of KR-96, Morgan Kaufmann, 1996, pp. 518–528.

[4] Lukasiewicz, W., "Considerations on default logic - an alternative approach", Computational Intelligence **4**, 1988, pp. 1–16.

[5] Lupea, M. "Nonmonotonic reasoning using default logics", Ph.D. Thesis, "Babes-Bolyai" University, Cluj-Napoca, 2002.

[6] Marek, W., Truszczynski, M., "Normal form results for default logics", Non-monotonic and Inductive logic, LNAI Vol. 659, Springer Verlag, 1993, pp. 153–174.

[7] Mikitiuk, A., Truszczynski, M., "Rational default logic and disjunctive logic programming", Logic programming and non-monotonic reasoning, MIT Press, 1993, pp. 283–299.

[8] Reiter, R., "A Logic for Default reasoning", Artificial Intelligence **13**, 1980, pp. 81–132.

[9] Schaub, T.H., "Considerations on default logics", Ph.D. Thesis, Technischen Hochschule Darmstadt, Germany, 1992.

[10] Schaub, T.H., "XRay system: An implementation platform for local query-answering in default logics", Applications of Uncertainty Formalisms, Lecture Notes in Computer Science, Vol. 1455, Springer Verlag, 1998, pp. 254–378.

[11] Schwind, C., "A tableaux-based theorem prover for a decidable subset of default logic", Proceedings CADE, Springer Verlag, 1990.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

E-mail address: `lupea@cs.ubbcluj.ro`