STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVII**, Number 1, 2002

DARR – A THEOREM PROVER FOR CONSTRAINED AND RATIONAL DEFAULT LOGICS

MIHAIELA LUPEA

Abstract. Default logics represent an important class of the nonmonotonic formalisms. Using simple by powerful inference rules, called *defaults*, these logic systems model reasoning patterns of the form "in the absence of information to the contrary of...", and thus formalize the default reasoning, a special type of nonmonotonic reasoning. In this paper we propose an automated system, called *DARR*, with two components: a propositional theorem prover and a theorem prover for constrained and rational propositional default logics. A modified version of semantic tableaux method is used to implement the propositional prover. Also, this theorem proving method is adapted for computing extensions because one of its purpose is to produce models, and extensions are models of the world described by default theories.

1. Introduction

One of the first formalizations of nonmonotonic reasoning was *classical default logic*, proposed by Reiter [6]. This logic system is based on first-order logic and introduces a new kind of inference rules called *defaults*. Defaults are used to draw conclusions by making implicit assumptions in the absence of information. Default logic is nonmonotonic because conclusions derived can be later invalidated by adding new information.

A *default theory* (D,W) consists of W, which is a set of consistent formulas of first-order logic (the facts) and a set of default rules D. The formulas of W are the axioms of the theory and a default rule has the form[1]:$d = \frac{\alpha:\beta}{\gamma}$, where $\alpha, \beta, \gamma$are formulas of first order logic,α is the *prerequisite* (Precond(d)) of the default d, β is the *justification* (Justif(d)) of the default d and γ is the *consequent* (Conseq(d)) of the default d.

In the paper we will use the notations: Justif(D)=$\bigcup_{d\in D}$ Justif(d), Prereq(D)= $\bigcup_{d\in D}$ Prereq(d), Concl(D)= $\bigcup_{d\in D}$ Concl(d).

1991 *Mathematics Subject Classification.* 03B70, 68T27, 68T37.

1998 *CR Categories and Descriptors.* I2 [**Artificial Intelligence**] Deduction and Theorem Proving – nonmonotonic reasoning and belief revision.

[1]Due to the (semi) representability results for these versions of default logic, we use in this paper only defaults with at most one justification (unitary default theories).

Informally, an extension for a default theory is a set of formulas derived from W using the standard inference rules of classical logic and the defaults. Formulas belonging to an extension are called *nonmonotonic theorems*, that means default conclusions of the default theory, which are not necessarily true, only plausible. A default theory may have zero, one or more classical extensions. The set of defaults used in the construction of an extension is called the *set of generating defaults* for the considered extension.

A default $d = \frac{\alpha:\beta}{\gamma}$ can be applied and thus derive γ if α is believed and *it is consistent to assumed* β.

Different variants (justified, constrained, rational) of default logic try to provide an appropriate definition of consistency condition for the justifications of the defaults, and thus to obtain many interesting and useful formal properties for these logic systems.

There are three computational problems specific to default logics:

Search problem: finding the extensions of a default theory.
Decision problems:
(1) deciding whether a formula belongs to at least one extension of a default theory (credulous perspective of the default reasoning);
(2) deciding whether a formula belongs to all extensions of a default theory (skeptical perspective of the default reasoning).

Automated theorem proving for default logics has began with solving the decision and searching problems for particular default theories: normal [6], ordered seminormal, and then was extended to general theories. The well known classical theorem proving methods: resolution, semantic tableaux method, connection method, were incorporated and adapted in the automated systems for default logics to solve specific tasks.

We will enumerate some of the automated reasoning system for default logics:

- **DeReS** [2] computes classical extensions for stratified default theories, using a semantic tableaux propositional prover.
- **Exten** [1] is based on an operational approach for computing classical, justified and constrained extensions.
- **GADEL** [5] uses the principles of genetic algorithms for computing classical extensions.
- **Xray** [9] represents an approach of the query-answering problem in constrained and cumulative default logics.

The aim of this paper is to introduce theoretical aspects regarding theorem proving in constrained and rational default logics and to describe an automated system implemented in C++, called *DARR*, for these variants of default logic.

2. Constrained and rational default logics

Constrained default logic was introduced by Schaub [7]. The consistency condition is a global one and it is based on the observation that in commonsense reasoning we assume facts, we memorize our assumptions and we verify that they do not contradict each other. The *actual extension* is embedded in a consistent *context* where are retained all the assumptions (justifications) used in the reasoning process.

Due to the global consistency condition, the constrained logic is strong regular, semi-monotonic, strongly commits to assumptions and guarantees the existence of extensions.

Theorem 2.1 [8]: Let (D,W) be a default theory and let E, C be sets of formulas. (E=actual extension,C=context) is a *constrained extension* of (D,W) if and only if:

E=Th(W $\cup$ Conseq(D')) and C=Th(W $\cup$Justif(D') $\cup$ Conseq(D'))
for a maximal set D'$\subseteq$D such that D' is grounded in W and W$\cup$Justif(D')$\cup$Conseq(D') is consistent.

This theorem states that the reasoning process formalized by constrained default logic is guided by a consistent context generated by a strong regular set of defaults. We observe that a default theory has always a constrained extension because D'=$\emptyset$ is grounded in W, W is consistent and thus $\emptyset$ can be a set of generating defaults.

Rational default logic was developed in [4] as a version of classical default logic for solve the problem of handling disjunctive information. The property of rational default logic is that defaults with mutually inconsistent justifications are never used together in constructing an extension of a default theory.

This logic system is strongly regular but does not guarantee the existence of extensions, is not semi-monotonic and does not commit to assumptions.

Theorem 2.2: Let (D,W) be a default theory and let E and C be sets of formulas. (E=actual extension, C=context) is a *rational extension* of (D,W) if and only if:

E=Th(W $\cup$ Conseq(D')) and C=Th(W $\cup$ Justif(D') $\cup$ Conseq(D'))
for a maximal D'$\subseteq$D such that D' is grounded in W and are satisfied the following conditions:

(i) W $\cup$ Concl(D') $\cup$ Justif(D') is consistent
(ii)$\forall d \in$ D$\setminus$D' we have: W $\cup$ Concl(D') $\cup$ $\{\neg$Precond$(d)\}$ is consistent **or**
W $\cup$ Concl(D') $\cup$ Justif(D' $\cup\{d\}$) is inconsistent

This theorem provides a necessary and sufficient criteria for the existence of a set of generating defaults of a rational extension. If condition (i) is satisfied by a set D', but condition (ii) is not satisfied, D' cannot be a set of generating defaults for a rational extension.

Proof: For proving theorem 2.2 we will use the original definition of a rational extension.

Definition 2.1 [4]: Let (D,W) be a default theory, let X be a subset of the set D of defaults and let S be a set of formulas.

1. We define $X_S = \left\{ \frac{\alpha}{\gamma} | \frac{\alpha:\beta_1,\ldots,\beta_n}{\gamma} \in X,\ S \cup \{\neg\beta_i\} \text{ is inconsistent, } 1 \leqslant i \leqslant n \right\}$.

2. A set X of defaults is *active* with respect to W and S if it satisfies the conditions:

(i) Justif(X)=$\emptyset$ or Justif(X)$\cup$ S is consistent;

(ii) Prereq(X)$\subseteq Th^{X_S}$(W),

where Th^{X_S}(W) is the deductive closure of W using classical inference rules and the monotonic rules fromX_S.

We denote by A(D,W,S) the set of all subsets of the defaults in D which are active with respect to W and S. $\emptyset \subseteq$ A(D,W,S). MA(D,W,S) is defined as the set of all maximal elements in A(D,W,S).

The set E of formulas is a *rational extension* for the theory (D,W) if E=Th^{X_E}(W), where X$\in$ MA(D,W,E).

We observe that X is the *set of generating defaults* in this original definition of rational extensions.

The proof of this theorem consists in showing the following:

- Condition (i) from definition 2.1 and condition (i) from theorem 2.2 are equivalent, with the meaning: the reasoning context is consistent.
- Condition (ii) from definition 2.1 is equivalent with the condition of groundness for the set of generating defaults.
- Condition (ii) from theorem 2.2 is equivalent with the necessity to be maximal-active (from definition 2.1) for the set of generating defaults.

The proofs of these equivalencies are immediate.

The set D' from the theorems above is the *set of generating defaults* for the extension (E,C). Thus, both types of extensions are deductive closures of the set W (explicit content) and the consequents of D'.

The relationships between constrained and rational extensions are as follows:

- the set of rational extensions coincide with the set of constrained extensions for the class of seminormal theories (all defaults have the form $d = \frac{\alpha:\beta\wedge\gamma}{\gamma}$);
- every rational extension is a constrained extension of the same theory.

3. A theorem prover for propositional logic, based on a modified semantic tableaux method

The aim of the proposed propositional theorem prover is to verify the consistency/inconsistency of a propositional formula/set of formulas and to provide a model in the case of consistency. We will use at implementation level the symbols for the logical operations: $\sim (\neg)$, $\&(\wedge)$, $|(\vee)$, $> (\rightarrow)$, $-(\leftrightarrow)$.

This theorem prover is based on a modified version of the semantic tableaux method. We will use the same representation for a tableau, like the function TP [Schw90], as a set of sets of literals, but the construction of the tableau is different.

The semantic tableau $\cup_{i=1}^{n}\{\cup_{k=1}^{n_i}\{a_{ik}\}\}$ has n branches and corresponds to the disjunction of its branches. The i-branch of the tableau is a set of literals: $\cup_{k=1}^{n_i}\{a_{ik}\}$and represents the conjunction of its literals. The tableau corresponds to a formula with the disjunctive normal form $\vee_{i=1}^{n}\wedge_{k=1}^{n_i} a_{ik}$. If a branch contains a literal and its negation, we say that the *branch* is *closed,* otherwise the *branch* is *open.* If all the branches of a tableau are closed, the *tableau* is *closed*, otherwise the *tableau* is *open.*

All the open subtableaux of a semantic tableau T are called the *openings* of T.

The new idea is to construct the semantic tableau of a formula from its postfix form. Traversing the postfix form from left to right, using a stack mechanism to memorize partial semantic tableaux (corresponding to the subformulas of the formula), and applying operations to the tableaux, the construction of the semantic tableau is very simple and efficient.

Definition 3.1: Let denote by Tsem(F) the semantic tableau attached to formula F. We compute Tsem(F) as follows:

Tsem(a) = {{a}}, where 'a' is a propositional literal;

Tsem(∼F) = ∼Tsem(F), '∼' is negation

Tsem(F & G) = Tsem(F) & Tsem(G), '&' is conjunction

Tsem(F | G) = Tsem(F) | Tsem(G), '|' is disjunction

Tsem(F > G) = Tsem(F) > Tsem(G), '>' is logical implication

Tsem(F − G) = Tsem(F) - Tsem(G), '-' is logical equivalence

Definition 3.1 can be extended for computing the semantic tableau of a set of formulas as follows: Tsem({F1,F2,. . . ,Fn}) = Tsem(F1)*Tsem(F2)*. . . *Tsem(Fn), where T1*T2 =T1&T2.

Definition 3.2: Let T1 = $\cup_{i=1}^{n}\{\cup_{k=1}^{n_i}\{a_{ik}\}\}$and T2 = $\cup_{j=1}^{m}\{\cup_{k=1}^{m_j}\{b_{jk}\}\}$ be two semantic tableaux. We define the operations ∼, &, |, >, - for semantic tableaux as follows:

∼T1=$\{\{\sim x_1, ..., \sim x_n\} \mid x_i \in \cup_{k=1}^{n_i}\{a_{ik}\}, i = 1, ..., n\}$

T1 | T2 =$\{\cup_{k=1}^{n_i}\{a_{ik}\} \mid i=1,\ldots,n\} \cup \{\cup_{k=1}^{m_i}\{b_{ik}\} \mid i=1,\ldots,m\}$

T1 & T2=$\{\cup_{k=1}^{n_i}\{a_{ik}\} \cup \cup_{k=1}^{m_j}\{b_{jk}\} \mid i=1,2,\ldots,n,\ j=1,\ldots,m\}$

T1 > T2 = ∼T1 | T2 and T1 − T2 = (T1 > T2) & (T2 > T1)

Example 3.1: Formula F=∼(a&b)|c&∼d has the postfix form ab&∼cd∼&|. Its semantic tableau, Tsem(F), is calculated step by step traversing the postfix form from left to right:

symbol	*partial semantic tableaux*	*stack*
'a':	T1=Tsem(a)={{a}},	st_tab=(T1)
'b':	T2=Tsem(b)={{b}},	st_tab=(T2,T1)
'&':	T3=T1 & T2={{a}}&{{b}}={{a, b}},	st_tab=(T3)
'∼':	T4=∼T3=∼{a&b}}={{∼a },{∼b}},	st_tab=(T4)
'c':	T5=Tsem(c)={{c}},	st_tab=(T5,T4)
'd':	T6=Tsem(d)={{d}},	st_tab=(T6,T5,T4)
'∼':	T7=∼T6=∼{{d}}={{∼d}},	st_tab=(T7,T5,T4)
'&':	T8=T5 & T7={{c}}&{{∼d}}={{c,∼d}},	st_tab=(T8,T4)
'|':	T9=T4 |T8={{∼a},{∼b}} | {{c,∼d}}= = {{∼a},{∼b},{c,∼d}},	st_tab=(T9)

Tsem(F)=T9={{∼a},{∼b},{c,∼d}}

The semantic tableaux method is a refutation method:

- formula F is valid (tautology) $\iff$ formula $\sim F$ is inconsistent $\iff$ Tsem($\sim F$) is a closed tableau;
- formula F is consistent $\iff$ Tsem(F) is an open tableau;
- formula G is deductible from the set $\{F_1,\dots,F_n\}$ $\iff$ $\{F_1,\dots,F_n,\sim G\}$ is inconsistent $\iff$ Tsem(F_1)*... * Tsem(F_n) * Tsem($\sim G$) is a closed tableau.

The main data structures used to implement the concepts: formula, set of formulas, semantic tableau, branch of a tableau are: *stiva, lista, formula, mult_formula, ramura* and *tabela.*

4. Implementation of *DARR* – a theorem prover for propositional constrained and rational default logic

We will consider in this paper only the case of propositional language as the underlying language for the default theories.

For easy access to the connection between the literals and the default where they belong, in the construction of a tableau all the literals are indexed as follows:

- the superior index is: **f** (literal from W) or **j** (literal from justifications) or **c** (literal from consequents).
- the inferior index is the number of the default where it belongs or is 0 if the literal belongs to the set of facts.

Adding indices to the literals from a semantic tableau we obtain an *indexed semantic tableau*, and we denote it by Tsem_ind.

The basic idea in computing constrained/rational extensions is to consider a maximal set X of formulas, $X = W \cup \text{Concl}(D) \cup \text{Justif}(D)$, that characterize the reasoning process (the facts, the consequents of the defaults and the justifications of the defaults) and then to supress the literals from defaults responsible for contradictions.

The candidates for the sets of generating defaults for extensions correspond to the openings in W of the indexed semantic tableau Tsem_ind(X).

All variants of default logic have in common the following property: the sets of generating defaults for extensions are grounded in the set of facts.

Definition 4.1: Let W be a set of formulas and let D be a set of closed defaults. We define the sequence of sets $(R_i)_{i \geqslant 0}$ as follows: $R_0 = \emptyset$ and

$$R_{i+1} = R_i \cup \left\{ d = \frac{\alpha : \beta}{\gamma} | d \in \text{D and W} \cup \text{Concl}(R_i) | = \alpha \right\}, i \geqslant 0.$$

The set D is *grounded in* W, if and only if D $= \bigcup_{i=0}^{\infty} R_i$.

D_baza $= \bigcup_{i=0}^{\infty} R_i$ is the maximal subset of D, grounded in W and can be calculated using algorithm *Submult_max_baza*(W,D,D_baza), which implements the above definition.

Using theorems 2.1 and 2.2 we can develop the following algorithm for computing all constrained and rational extensions of the default theory (D,W).

Algorithm 4.1:

Calcul_ext_restrictii_rationale(D,W)

begin
 We construct the semantic indexed tableau:
  T= Tsem_ind(W) *Tsem_ind(Justif(D))* Tsem_ind(Concl(D)).
 We compute all the subsets $S_1, ..., S_n$ of D, such that the semantic tableaux:
  Tsem_ind(W)* Tsem_ind(Concl(S_i))*Tsem_ind(Justif(S_i)), *i=1,. . . , n*
 are open.
 We eliminate from $S_1, ..., S_n$ the sets that are not maximal and we obtain
 the sets $R_1, ..., R_{n'}$ of defaults.
 for i=1,. . . ,n' do
  Submult_max_baza(W,R_i,R'_i)
 endfor
 print "(Th(W∪Concl(R'_i)),Th(W∪Concl(R'_i)∪Justif($'R_i$))) *i=1,. . . ,n'* are
 all constrained extensions, R'_i are the set of generating defaults"
 if the theory (D,W) is semi-normal
  then print "(Th(W∪Concl(R'_i)),Th(W∪Concl(R'_i)∪Justif(R'_i))) *i=1,. . . ,n'*
   are all rational extensions, R'_i are the set of generating defaults"
 else
  for i=1,. . . ,n' do
 // we verify if R'_i is maximal active with respect to W and Th(W∪Concl(R'_i))
   ind=0
   while (ind==0 and not all$d \in D \backslash R'_i$ are chosen) do
    We chose a new $d \in D \backslash R'_i$
    if (Tsem(W) *Tsem(Concl(R'_i))*Tsem(Justif(R'_i∪{d}) is open

and Tsem(W) *Tsem(Concl(R'_i))*Tsem({∼Precond(d)) is closed)
then ind=1
endif
endwhile
if ind==1
then print "R'_i cannot generate a rational extension "
else print "(Th(W∪Concl(R'_i)),Th(W∪Concl(D'_i)∪Justif(R'_i))) is
a *rational extension* and R'_i is its set of generating defaults".
endif
endfor
endif
end

Accepting alternative possibilities for extending a default theory characterizes the *credulous reasoning.* The commonsense reasoning is the human model of reasoning, by making default assumptions for overcoming the lack of information. This type of reasoning belongs to the credulous perspective of the reasoning.

Skeptical reasoning is imposed in prediction problems because the nonmonotonic consequences cannot be later modified, which means that derived formulas does not depend on the alternative assumptions made during the reasoning process. It is considered irrational to have the possibility to chose one belief or another one if they are contradictory.

The specific of the problem will decide the appropriate perspective for the nonmonotonic reasoning used to solve the problem.

According to theorem 3.1 from [3] the skeptical nonmonotonic theorems of the theory (D,W) belong to the set: $Th^n_{D,\cap}$(W)=Th(W∪{$\vee_{i=1}^{k} \wedge_{j=1}^{n_i} c^i_j$}) where Concl($R_i$)={$c^i_1, c^i_2, ..., c^i_{n_i}$}, *i=1,. . . ,k*, and $R_1, ..., R_k$ are all the sets of generating defaults for the extensions of type *n=res (constrained) or n=rat (rational).*

If all extensions are calculated, the problem of membership to all extensions is reduced to a derivability problem in classical logic.

Algorithm 4.2:

Verif_consec_sceptica (f, D, W, mult_reg_gen)

begin
// mult_reg_gen={$R_1, ..., R_k$} from algorithm 4.1, Concl(R_i)={$c^i_1, c^i_2, ..., c^i_{n_i}$},
$i = 1, .., k$
if Tsem(W) *Tsem({$\vee_{i=1}^{k} \wedge_{j=1}^{n_i} c^i_j$})* Tsem({∼f}) is closed
then print "f is a skeptical nonmonotonic consequence of the theory
(D,W)"
else print "f is not a skeptical nonmonotonic consequence of the theory
(D,W)"
endif
end

The theorem prover for constrained and rational default logics is obtained by implementing the concepts: *indexed tableau, default, default theory* and the algorithms proposed above.

5. Conclusions

In this paper we have proposed a theorem for global characterization of rational extensions using the set of generating defaults and we have developed algorithms for solving the theorem proving problems specific for constrained and rational default logics.

The tight relationship between constrained and rational default logics was the reason to implement an automated theorem prover, called *DARR,* for both of these logic systems. The theorem prover proposed for propositional logic creates and manipulates in a very efficient and elegant way the semantic tableaux, using operators. A modified version of semantic tableaux method was adapted for computing constrained and rational extensions of a default theory.

References

[1] G.Antoniou, A.P.Courtney, J.Ernst, MA.Williams: *A System for Computing Constrained Default logic Extensions.* Logics in Artificial Intelligence, JELIA'96, Lecture Notes in Artificial Intelligence, 1126, 1996, pp. 237–250.

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

[3] M.Lupea: *Nonmonotonic inference operations for default logics.* ECIT 2002, Iasi, Romania, Symposium on Knowledge-based Systems and Expert Systems, pp. 1–12.

[4] A.Mikitiuk, M.Truszczynski: *Rational default logic and disjunctive logic programming*, în A. Nerode, L.Pereira, Logic programming and non-monotonic reasoning, MIT Press, 1993, pp. 283–299.

[5] P.Nicolas, F.Saubion, I.Stephan: *Genetic Algorithms for Extension Search in Default Logic*, 8-th International Workshop on Non-Monotonic Reasoning (NMR2000).

[6] R.Reiter: *A logic for default reasoning.* Journal of Artificial Intelligence, 13, 1980, pp. 81–132.

[7] T.H.Schaub: *Considerations on default logics.* Ph.D. Thesis, Technischen Hochschule Darmstadt, Germany, 1992.

[8] T.H.Schaub: *The Automation of Reasoning with Incomplete Information.* Springer-Verlag Berlin, 1997.

[9] T.H.Schaub: *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.

[10] C.B.Schwind: *A tableaux-based theorem prover for a decidable subset of default logic.* 10-th International Conference on Automated Deduction. Lecture Notes in A.I. 449, 1990.

Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: lupea@cs.ubbcluj.ro