CREATIVE MATH. & INF.
16 (2007), 99 - 107

Dedicated to Professor Ioan A. RUS on the occasion of his 70^{th} anniversary

Computing constrained default extensions - a constraint satisfaction problem

GABRIELA ŞERBAN and MIHAIELA LUPEA

ABSTRACT. Constrained default logic belongs to the class of default logics which formalize default reasoning. This type of nonmonotonic reasoning is modelled by *defaults* which permit inferring conclusions in the absence of complete information, using default assumptions. Using the classical inference rules and the defaults, the set of initial facts is extended with formulas, called *nonmonotonic theorems* (*beliefs*), obtaining *extensions*.

This paper presents a new approach in computing *constrained extensions*. We define the problem of computing the generating default sets of extensions as a *constraint satisfaction problem* and we introduce *BTCE* algorithm for solving this problem. The proposed algorithm is based on a top-down approach and uses pruning for an efficient search.

1. INTRODUCTION

Default logics formalize the reasoning with incomplete information, overcoming the lack of information by making default assumption about a situation. This type of reasoning is nonmonotonic, that means: derived conclusions (only plausible, not necessarily true) may be later invalidated by new information. These logical systems are based on first-order logic and use a new kind of inference rules, the *defaults*, that model laws which are true with a few exceptions.

A *default theory* ([11]) $\Delta = (\mathcal{D}, \mathcal{W})$ consists of a set $\mathcal{W}$ of facts, as consistent formulas of first order logic and a set $\mathcal{D}$ of *default rules*. A *default* has the form $d = \dfrac{\alpha : \beta_1, \ldots, \beta_m}{\gamma}$, where: α is called *prerequisite*, $\beta_1, \ldots, \beta_m$ are called *justifications* and γ is called *consequent*.

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

The differences among different versions (classical, justified, constrained, rational) of default logic are caused by the semantics (applicability condition) of the defaults.

An *extension* of a default theory is a maximal set of conclusions (*beliefs*) derived from the facts of $\mathcal{W}$ using classical derivation and the defaults as inference rules.

The set of defaults used in the construction of an extension is called the *generating default set* of the considered extension.

Received: 19.09.2006. In revised form: 19.02.2007.

2000 *Mathematics Subject Classification.* 03B70, 68T15, 68T27, 68T20.

Key words and phrases. *Nonmonotonic reasoning, default logic, constraint satisfaction problem, backtracking.*

Classical default logic was proposed by Reiter ([11]) and *justified default logic* was introduced by Lukaszewicz ([4]). These two versions of default logic have the disadvantage that the consistency condition for justifications is an individual one, and thus some inconsistencies consequents-justifications (in classical version) and justifications-justifications (in justified version) are not detected.

Two new versions, *constrained default logic* ([12]) and *rational default logic* ([8]) were developed, solving this drawback by keeping track of the implicit assumptions and verifying that they do not contradict each other. Therefore the consistency condition for the justifications is a global one.

Due to its very high level of theoretical complexity ($\sum_2^P = NP^{NP}$), caused by the great power of the inferential process, the problem of finding all extensions of a general default theory, can be solved in an efficient manner only for particular classes of default theories.

Related Work

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 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.

Semantic tableaux method is used in [15] to compute classical extensions for a decidable subset of default logic and is adapted in a top-down approach ([2]) for building extensions of terminological default theories.

An uniform approach, based on a modified version of propositional semantic tableaux method, for computing constrained and rational extensions, is presented in paper [6].

Default reasoning is integrated into existing model elimination based provers using the well-known Prolog Technology Theorem Proving Techniques ([14]) in order to solve the query-answering problem for constrained and cumulative default logics.

Based on an operational approach and using pruning techniques for the search tree ([1]), classical, justified and constrained extensions are computed.

In this paper we propose a constraint satisfaction based approach for computing all generating default sets of constrained extensions for general default theories. It is a top-down technique combined with pruning, that increases its efficiency. Its main idea, based on the global characterization of constrained extensions, is to consider the largest grounded subset of defaults and then to remove one by one the defaults in order to generate all the consistent contexts.

We focus in this paper only on constrained default logic, but our approach can be easily adapted for rational default logic, also.

The paper is structured as follows. In Section 2 we introduce the main theoretical aspects of constrained default logic. Section 3 explains our approach, presenting the theoretical model for computing the generating default sets of constrained

extensions (Subsection 3.1) and *BTCE* algorithm for this computation (Subsection 3.2). Conclusions and future work are outlined in Section 4.

2. CONSTRAINED DEFAULT LOGIC

Constrained default logic was introduced by Schaub ([12]). The consistency condition is a global one and it is based on the observation that in common-sense reasoning we assume things, we keep track of our assumptions and we verify that they do not contradict each other. A constrained extension is defined as a pair (*E*,*C*).The actual extension *E* is embedded in a consistent context *C* where are retained all the justifications of the generating defaults.

The results from [7] 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:
$Prereq(d) = \alpha$, $Justif(d) = \beta$, $Conseq(d) = \gamma$, $Prereq(\mathcal{D}) = \bigcup_{d\in\mathcal{D}} Prereq(d)$, $Justif(\mathcal{D}) = \bigcup_{d\in\mathcal{D}} Justif(d)$, $Conseq(\mathcal{D}) = \bigcup_{d\in\mathcal{D}} Conseq(d)$
and $Th(X) = \{A | X \vdash A\}$ the classical deductive closure of the set X of formulas. In the Definition 2.1 we will introduce the property of *groundness* that will be used in the definitions of default extensions.

Definition 2.1. ([2]) Let $\mathcal{W}$ be a set of facts and $\mathcal{D}$ be a set of defaults. We define $\mathcal{D}_0 = \emptyset$ and, for $i \geq 0$,

$$\mathcal{D}_{i+1} = \mathcal{D}_i \cup \Big\{ d = \frac{\alpha : \beta}{\gamma} \mid d \in \mathcal{D} \text{ and } \mathcal{W} \cup Conseq(\mathcal{D}_i) \vdash \alpha \Big\}.$$

$\mathcal{D}$ is called **grounded** in $\mathcal{W}$ if $\mathcal{D} = \bigcup_{i=0}^{\infty} \mathcal{D}_i$. If $\mathcal{D}$ is not grounded in $\mathcal{W}$, then $\bigcup_{i=0}^{\infty} \mathcal{D}_i$ is the **largest subset of $\mathcal{D}$ that is grounded in $\mathcal{W}$**.

Definition 2.1 is equivalent to the one given in [15] and its advantage is that can be used as a decision procedure for checking the groundness of a set of defaults and also to compute the largest grounded subset of a set of defaults.

All versions of default logics were introduced using fixed-point operators for the definition of extensions. These definitions are difficult to be used in the process of constructing extensions and thus equivalent global characterizations of extensions were proposed. Theorem 2.1 provides a global characterization of constrained extensions using the generating default sets.

Theorem 2.1. *(*[12]*) Let ($\mathcal{D}$,$\mathcal{W}$) be a default theory, and let E, C be sets of formulas. (E, C) is a* **constrained extension** *of ($\mathcal{D}$,$\mathcal{W}$) iff $E = Th(\mathcal{W} \cup Conseq(\mathcal{D}_g))$ and $C = Th(\mathcal{W} \cup Conseq(\mathcal{D}_g) \cup Justif(\mathcal{D}_g))$ for a maximal set $\mathcal{D}_g \subseteq \mathcal{D}$ such that $\mathcal{D}_g$ is grounded in $\mathcal{W}$ and condition (i) is satisfied:*

(i) *the set $\mathcal{W} \cup Conseq(\mathcal{D}_g) \cup Justif(\mathcal{D}_g)$ is consistent.*

Theorem 2.1 states that the reasoning process formalized by constrained default logic is guided by a consistent context. Each constrained extension is generated by a set $\mathcal{D}_g$ of defaults whose justifications and consequents are together consistent, and at the same time consistent with the set of facts.

We remark that a default theory has always a constrained extension because $\mathcal{D}_g = \emptyset$ is grounded in $\mathcal{W}$, $\mathcal{W}$ is consistent and thus, in the worst case, $\emptyset$ is the only generating default set.

3. Computing constrained default extensions - a constraint satisfaction problem

In this section we present the problem of computing the generating default sets of constrained extensions as a constraint satisfaction problem ([17]).

Constraint based reasoning is a simple, but powerful paradigm in which many interesting problems can be formulated. Informally, a *constraint satisfaction problem (CSP)* is a problem stated in the form of a set of constraints. The general *CSP* is *NP-complete*.

Theorem 2.1 from Section 2 shows that the problem of computing all default extensions can be reduced to the problem of finding the generating default sets for those extensions.

This approach is a top-down constraint satisfaction-based approach that uses pruning for an efficient search. Its basic idea is to consider the largest grounded subset $\mathcal{D}'$ of $\mathcal{D}$ and then remove one by one the defaults in order to generate all the consistent contexts.

In order to better explain our approach, in Subsection 3.1 we define the problem of computing the generating default sets of constrained extensions as a constraint satisfaction problem. In Subsection 3.2 we give the *BTCE* algorithm for computing these sets.

3.1. Theoretical model.

In this section we define the problem of computing the generating default sets of constrained extensions as a constraint satisfaction problem.

Let $(\mathcal{W}, \mathcal{D})$ be a default theory and $\mathcal{D}'$ be the largest grounded subset of $\mathcal{D}$ in $\mathcal{W}$. Let us assume that

$$\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$$

and

$$\mathcal{D}' = \{d_{i_1}, d_{i_2}, \ldots, d_{i_n}\},$$

where $n \leq m,\ 1 \leq i_j \leq m,\ \forall j = 1, \ldots, n$ and $i_j \neq i_l,\ \forall j, l \in \{1, \ldots, n\},\ j \neq l$.

Definition 3.2. A set of defaults $M \in 2^{\mathcal{D}'}$ is called a **candidate generating default set of a constrained extension** (we will refer it as a ***candidate***). We denote by $\mathcal{G}$ the set of all candidates which generate consistent contexts and is defined as follows:

$$\mathcal{G} = \{M \,|M \in 2^{\mathcal{D}'}\ , \mathcal{W} \cup Conseq(M) \cup Justif(M)\, is\ a\ consistent\ set\}.$$

We mention that we will refer an element from $\mathcal{G}$ as a **consistent candidate**.

In our approach, the candidates will be chosen as elements from $2^{\mathcal{D}'}$, instead of $2^{\mathcal{D}}$, because only grounded defaults can belong to generating default sets.

Definition 3.3. A candidate $M \in 2^{\mathcal{D}'}$ is called **maximal** in a set S of candidates, if M has no superset in S, i.e., $\nexists M' \in S,\ M \subset M'$.

Definition 3.4. We define the problem of computing the generating default sets of constrained extensions as a Constraint Satisfaction Problem (*CSP*), expressed as a triple $< X, D, C >$, where:

- $X = \{x_1, x_2, \ldots, x_k\}$ is the set of variables, where $k \leq n$.
- $D = \{D_1, D_2, \ldots, D_k\}$ is the set of domains, $D_j = \{0, 1\},\ \forall j = 1, \ldots, k$. The significance of a variable value is as follows:
  if $x_j = 1$ then the default d_{i_j} is eliminated from $\mathcal{D}'$,
  if $x_j = 0$ then the default d_{i_j} is not eliminated from $\mathcal{D}'$.
- C is the set of constraints, expressed in equations below:
  (i) The set $\mathcal{D}' \setminus \bigcup_{j=1,x_j=1}^{k} d_{i_j}$ is a **maximal consistent candidate**.

  (ii) $\forall l = 1, ..., k-1$, the set $\mathcal{D}' \setminus \bigcup_{j=1,x_j=1}^{l} d_{i_j}$ is **not** a **consistent candidate** or is a **consistent candidate, but not a maximal one**.

Definition 3.5. Let *CSP* be the constraint satisfaction problem corresponding to a default theory and x^l=$(x_1, x_2, \ldots, x_l)$, $l \leq k$ be a partial instantiation of the variables. We will denote by M_{x^l} the candidate corresponding to the instantiation x^l, $M_{x^l} = \mathcal{D}' \setminus \bigcup_{j=1,x_j=1}^{l} d_{i_j}$. We will call $M_{x^l} = \mathcal{D}' \setminus \bigcup_{j=1,x_j=1}^{l} d_{i_j}$ the **candidate generating default set corresponding to the instantiation** x^l. If M_{x^l} is a consistent candidate, then x^l is called a **valid instantiation**.

Because of our top-down approach, in our view, a value of an instantiation x^l=$(x_1, x_2, \ldots, x_l)$ of the problems' variables is obtained by eliminating from $\mathcal{D}'$ the defaults d_{i_j}, iff $x_j = 1$, $1 \leq j \leq l$.

Definition 3.6. Let *CSP* be the constraint satisfaction problem defined above. A **consistent candidate** $M \in \mathcal{G}$ is called a **generating default set of a constrained extension** if M is **grounded** in $\mathcal{W}$ and M is **maximal** in $\mathcal{G}$. We will denote by $\mathcal{GDSE}^C$ the set of all **Generating Default Sets of Constrained Extensions**,

$$\mathcal{GDSE}^C = \{M \,|M \in \mathcal{G},\ M \ is\ grounded\ in\ \mathcal{W}\ and\ is\ maximal\ in\ \mathcal{G}\}.$$

3.2. **The Algorithm for Computing the generating default sets of constrained Extensions (*BTCE*).** Let us consider the problem of computing the generating default sets of extensions, as formulated in Subsection 3.1. In this section we present the algorithm for computing these sets, *BTCE (Chronological **B**ack**T**racking for **C**omputing Default **E**xtensions*).

The algorithm performs a *depth-first* search of the space of potential solutions of the *CSP* defined in Subsection 3.1, combined with pruning techniques.

BTCE algorithm restricts the search to those regions of the search space where is possible that maximal consistent contexts to be generated. It prunes out all the regions of non-maximal consistent candidates.

We have to mention that we will use in the algorithm description the notions of **maximality** and **groundness** as they were introduced in Definition 3.3 and Definition 2.1, respectively.

The main idea of the algorithm is the following:

- A depth-first search is performed in the solution space in order to determine a solution set $\mathcal{S}$ containing all the maximal consistent candidates. The search will be **pruned** on the branches that will not lead to a solution.
- The variables of the *CSP* defined in Section 3.1 are instantiated sequentially, in order to obtain a valid instantiation (Definition 3.5).
- Backtracking takes place when an instantiation that is not valid is reached.
- **Pruning case 1**

  A **pruning** of the search is made when a candidate M that is not **maximal** in $\mathcal{S}$ is reached. The pruning is made because a superset of M was already detected. This case is applied if $M \in \mathcal{G}$, but without checking whether M is a consistent candidate or not (the maximality condition implies that M is a consistent candidate). The efficiency of the pruning case 1 from the algorithm derives from using the condition for a state not to be maximal in the partial set of solutions. This condition is more powerful, but computationally less expensive, than checking the property of being a **consistent candidate**.

- **Pruning case 2**

  If a consistent candidate $M \in \mathcal{G}$ that is **maximal** in $\mathcal{S}$ is reached during the search, M is retained in the solution set $\mathcal{S}$. According to Theorem 2.1 that a constrained extension is generated by a maximal consistent candidate, the corresponding partial instantiation will not be extended any more.
- After the solution set $\mathcal{S}$ was determined (all the valid paths in the search space were searched), a filtering step (for checking the groundness of a consistent candidate in the set of facts) is applied, in order to construct from $\mathcal{S}$ the *generating default sets* $\mathcal{GDSE}^C$ (Definition 3.6).

We give next *BTCE* algorithm. An auxiliary set (the solution set $\mathcal{S}$) that contains all maximal consistent candidates is used.

Algorithm ***BTCE*** **is**
Input: - the default theory ($\mathcal{W}$, $\mathcal{D}$)
Precondition: - the set of facts $\mathcal{W}$ is a consistent set
Output: - $\mathcal{GDSE}^C$ all the generating default sets
Begin
  $\mathcal{S} \leftarrow \emptyset$ //the solution set is initially empty
  @Compute $\mathcal{D}' \leftarrow \{d_{i_1}, d_{i_2}, \ldots, d_{i_n}\}$ the largest grounded subset of $\mathcal{D}$
  Call BT($\mathcal{W}, \mathcal{D}', x, 1, \mathcal{S}$) // the backtracking procedure is called
  $\mathcal{GDSE}^C \leftarrow \emptyset$ //the output set is initially empty
  For each $M \in \mathcal{S}$ do //the solution set $\mathcal{S}$ is filtered by checking

```
                    //the groundness property of M
      M' ← the largest grounded subset of M
      If M' ≠ ∅ then
        GDSE^C ← GDSE^C ∪ {M'}
      EndIf
  EndFor
End.
```

```
Procedure BT (W, D', x, l, S) is
Input: - the default theory (W, D)
       - the partial instantiation x = (x_1, x_2, ..., x_{l-1}) of the CSP, if l > 1
       - the solution set S
Precondition: 1 ≤ l ≤ n
Output: - the solution set S
        - the partial instantiation x = (x_1, x_2, ..., x_l) of the CSP
Begin
  If l ≤ n then
   For j ← 0, 1 do //we take each possible value for variable x_j
     x_l ← j
     @Compute the set M ← D' \ ⋃_{j=1, x_j=1}^{l} d_{i_j}
     If (M is maximal in S) then
        If (M ∈ G) then //M is a maximal consistent candidate
          S ← S ∪ {M}
          Exit// *** Pruning case 2 ***
         else
           Call BT (D, D', x, l + 1, S) //the backtracking procedure is called
        EndIf
      else //*** Pruning case 1 ***
     EndIf
   EndFor
  EndIf
End.
```

The top-down exploration method and the pruning conditions proposed above provide all the generating default sets of extensions, assuring *BTCE* completeness.

Based on the outputs of *BTCE* algorithm, the constrained extensions are obtained as follows:

- If $\mathcal{GDSE}^C = \{\emptyset\}$, then the default theory has only one constrained extension, $(Th(\mathcal{W}), Th(\mathcal{W}))$, with $\emptyset$ as a generating default set.
- If $\mathcal{GDSE}^C = \{g_1, \ldots, g_m\}$, the default theory has m constrained extensions, $(Th(\mathcal{W} \cup Conseq(g_i)), Th(\mathcal{W} \cup Conseq(g_i) \cup Justif(g_i)))$, $i = 1, \ldots, m$.

Example 3.1. Let us consider $(\mathcal{D}, \mathcal{W})$ a default theory with $\mathcal{W} = \{F \vee C, K\}$ and

$$\mathcal{D} = \{d1 = \frac{: \neg B \wedge \neg C}{E},\ d2 = \frac{K : \neg A}{C},\ d3 = \frac{C : \neg B \wedge \neg F}{G},$$
$$d4 = \frac{: F}{B},\ d5 = \frac{F : A}{B}\}$$

The largest grounded subset of $\mathcal{D}$ in $\mathcal{W}$ is $\mathcal{D}' = \{d1, d2, d3, d4\}$.

Using *BTCE* algorithm, during the depth-first exploration of the search space, all pruning cases are encountered. Three constrained extensions are computed: $(E1, C1)$, $(E2, C2)$, $(E3, C3)$ with $D1, D2, D3$ as generating default sets.

- valid instantiation $(0, 1, 1, 1)$ corresponding to $D1 = \{d1\}$

  $E1 = Th(\{F \vee C, K, E\})$, $C1 = Th(\{F \vee C, K, E, \neg B \wedge \neg C\})$.

- valid instantiation $(1, 0, 0, 1)$ corresponding to $D2 = \{d2, d3\}$

  $E2 = (Th(\{F \vee C, K, C, G\})$, $C2 = Th(\{F \vee C, K, C, G, \neg A, \neg B \wedge \neg F\})$.

- valid instantiation $(1, 0, 1)$ corresponding to $D3 = \{d2, d4\}$

  $E3 = Th(\{F \vee C, K, C, B\})$, $C3 = Th(\{F \vee C, K, C, B, \neg A, F\}))$.

4. Conclusions and further work

In this paper a constraint satisfaction based approach for computing the generating default sets of constrained extensions is presented. This problem is defined as a *constraint satisfaction problem*.

We have also introduced *BTCE* algorithm for computing the generating default sets of constrained extensions, based on the idea of searching the solution space of the associated *CSP* and pruning the paths that lead to failure.

The main disadvantage of *BTCE*, as all the other similar approaches from the literature for general default theories, is the fact that it is computational expensive (exponential as time and space).

Further work can be done in the following directions:

- To improve our approach using heuristic search procedures.
- To study the possibility to use other search techniques, as *intelligent backtracking* techniques ([16]): backjumping, backmarking, etc.
- To use heuristics for variables' instantiation in the associated *CSP*.

- To extend this approach for computing constrained default extensions towards a distributed one, using an asynchronous version of the *BTCE* algorithm.
- To use splitting techniques ([3]) for default theories and local search procedures for the strata (clusters) of the theory.

REFERENCES

[1] Antoniou G., Courtney A.P., Ernst J. and 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] Baader F. and Hollunder B., *Computing Extensions of Terminological Default Theories*, Foundantions of knowledge representation and reasoning, Lecture Notes in Artificial Intelligence, 1994, pp. 30–51

[3] Cholewinski P., Marek W. and 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 Mihaiela *Nonmonotonic reasoning using default logics*, Ph.D. Thesis, "Babes-Bolyai" University, Cluj-Napoca, 2002

[6] Lupea Mihaiela *DARR - A theorem prover for constrained and rational default logics*, Studia Universitatis "Babes-Bolyai", Informatica, Vol. XLVII, **1** (2002), pp. 45–52

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

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

[9] Mikitiuk A. and Truszczynsky M., *Constrained and rational default logics*, Proceedings of IJCAI-95, Morgan Kaufman, 1995, pp. 1509–1515

[10] Ruttkay Z., *Constraint satisfaction - a Survey*, Volume 11 (2–3), 1998, pp. 163–214

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

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

[13] Schaub T.H., *The automation of reasoning with incomplete information*, Springer-Verlag, Berlin, 1997

[14] 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

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

[16] Gabriela Şerban and Pop H.F., *Artificial Intelligence techniques - agent based approaches*, Ed. Mediamira, Cluj-Napoca, 2004

[17] Weiss G. (Ed.), *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999

BABEŞ-BOLYAI UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
M. KOGALNICEANU 1
400084 CLUJ-NAPOCA, ROMANIA
E-mail address: gabis@cs.ubbcluj.ro
E-mail address: lupea@cs.ubbcluj.ro