# Default Reasoning with a Constraint Resolution Principle

Sven Eric Panitz

Künstliche Intelligenz/Softwaretechnologie
Fachbereich 20
Johann Wolfgang Goethe-Universität
Postfach 11 19 32
60054 Frankfurt am Main
*Germany*
e-mail: panitz@informatik.uni-frankfurt.de

**Abstract.** Common non-monotonic deduction systems such as Reiters default logic or inheritance networks must handle the multiple extension problem, i.e. generally they have distinct consistent sets of conclusions (extensions) which can be drawn from a theory. Most of these systems do not provide any control structure that helps to determine in which of these extensions inferences are made. Default proof theories are in most cases only able to prove if a sentence is in at least one extension.
We create a resolution based system for default reasoning where an explicit control set can be built during reasoning and where default assumptions made in a deduction are remembered as constraints.
The resulting system is a cumulative default logic which assures an even stricter cumulativity than Brewka's Cumulative Default Logic and is strong enough to express Poole's Framework for default reasoning. It turns out that the created system ranges from the logic with restricted quantifiers of H.J.Bürckert via Reiter's default logic to the logic of theory change of Gärdenfors.

## 1 Introduction

It is commonly understood that various forms of default reasoning arise in artificial intelligence, therefore, intensive research has been made in this area during the last decade. Most of these works provide no way to remember which assumptions had to be made to derive a certain conclusion, (e.g. Reiter [9] defines in his default logic the notion of extensions as a maximal consistent set of sentences which can be drawn from a default theory) or provide further expensive checks for assuring consistency of the assumptions made (e.g. Brewka [3], who demands that assumptions be both consistent with each other and consistent with the drawn conclusions).
In an extension, the information indicating how these conclusions were reached is mostly lost; sentences derived with default assumptions are not distinguished from sentences derived without them.
The other extreme is to have no plain facts as conclusions but complicated formulae which explicitly include the made assumptions. But somehow this contradicts

the philosophy of having assumptions made implicitly as long as it is possible. It is true that most common sense reasoners make assumptions and in addition are usually not aware of these assumptions while handling normal cases, but in the moment of crisis, i.e. when inconsistent knowledge has been derived, a common sense reasoner will surely become aware of his unconsciously made assumptions.
As an example, let us consider the everyday routine of getting a bus ticket from a vending machine. Our knowledge about a ticket machine is something like the following:

$$\text{money-inserted}(x) \wedge \text{chosen-direction}(x,y) \\ \rightarrow \text{print-ticket}(x,y) || \text{out-of-order}(x)$$

stating, assumed a machine $x$ is not out of order, it will print a ticket to the chosen destination if money has been inserted.
Given an individual machine A, as a naive user we will just choose a direction, insert money and see that no ticket is printed. It is at this moment and only at this moment that we will remember the assumption we made about machine A and we will in future exclude making the assumption that machine A is not out of order.

Thus, our proposed system has to be able to tell which assumptions had to be made to infer a certain sentence. The system furthermore provides a control set, which restricts the possible assumptions. This control set also plays the role of a selector among different extensions, i.e. selects which consistent view of the world we will adopt.
In essence then, we demand our system to consist of three parts:

- **W**: the world description, a set of strict sentences
- **D**: a set of default formulae, i.e. sentences which are true under certain assumptions
- **C**: the control set, which restricts the assumptions that may be made.

A control set of a certain system will be considered useful if it controls deduction in such a way that no contradictions can be derived as long as the world description is consistent; thus a useful control set will not only resolve inconsistencies, but also control which of the initially possible extensions has been chosen. We cannot expect that a useful control set is provided by a user; therefore, we have to compute the control set.

## 2 The controlled default resolution

With three basic definitions, we will define, in a declarative way, how our system shall look. This will serve as a semantic. We will relate our nomenclature closely to that of Reiter's default logic.

**Def.:** (constraint clause theory)
A constraint clause theory (cct) is a pair (**W**,**D**), where **W** is a finite set of clauses and **D** a finite set of constraint clauses of the form: F||B, where F is a clause and B a non-empty set of clauses. F is called the kernel and B the constraint of F||B.

**Def.:** (control set)
A control set **C** for a constraint clause theory (**W**,**D**) is a set of literals so that for every L∈**C** there exists a constraint clause F||B and a substitution $\sigma$, such that there exists a literal L' somewhere in **B** with: $\sigma$(**L'**)=¬L.

By this definition we only allow predicate symbols in the control set, which occur in some constraint of **D**. Note that we do not restrict **C** to be a consistent set. This will cause us some trouble in formulating further definitions but will also ensure that we can always draw conclusions from a cct.
The next definition will show in which way a control set of a cct will control the deductions to be made:

**Def.:** (expansion)
An expansion **E** of a cct (**W**,**D**) is a maximal set of formulae, such that there exists a control set **C** with:
i) **E** is consistent iff **W** is consistent
ii) for all A∈**E** holds:
there exists a set $\{F_1||B_1,...,F_n||B_n\}$ ⊆**D** and substitutions $\sigma_{1_1},...,\sigma_{1_m},...,\sigma_{n_1},...,\sigma_{n_m}$, such that
**W**$\cup\{\sigma_{1_1}(F_1),...,\sigma_{1_m}(F_1),...,\sigma_{n_1}(F_n),...,\sigma_{n_m}(F_n)\}\models A$ and for all consistent subsets **C'** of **C** and all $\sigma_{i_j}$
$\sigma_{i_j}(B_i)\cup$**C'** is consistent.
The set of all expansions of (**W**,**D**) is denoted as **X**(**W**,**D**).
A control set is called useful for a cct, if it defines an expansion in the above definition.

With these three definitions we get directly the following theorem:

**Theorem**
A cct has always an expansion.
**Proof:**
If **W** is inconsistent this is trivially the case.
If **W** is consistent we know that Th(**W**) is a subset of all expansions (this is only possible, because we did not restrict **C** to be consistent). If there exist larger sets that fulfill the above conditions, the maximal of these will be the expansions, else Th(**W**) will be the single expansion.

By proving this we also have shown:

**Corollary**
For all $\mathbf{E}\in\mathbf{X}(\mathbf{W},\mathbf{D})$ the following holds:
$\mathrm{Th}(W)\subseteq\mathbf{E}$.

The above definitions create the three components we were demanding:

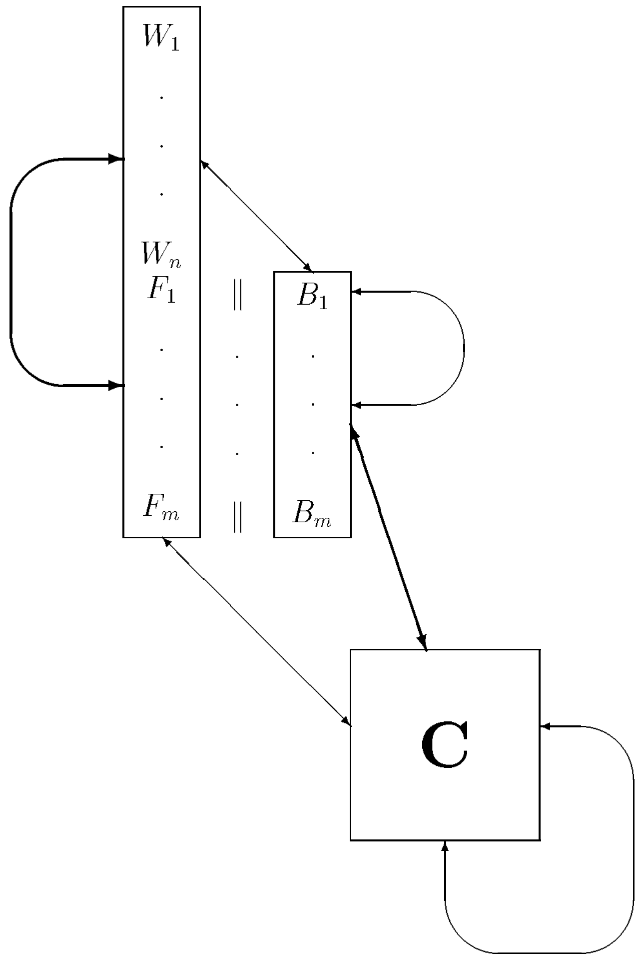
1. Clauses from which we can draw conclusions. These are the clauses of **W** together with the kernels of the restricted clauses.
2. Constraints which have to be assumed for special clauses.
3. A control set which restricts the assumptions that may be made.

A deduction in our system can be made in the following way:
We try to prove a formula indirectly by a constraint resolution principle and derive a constraint empty clause $\Box\|B$. If a useful control set is provided, our proof has been successful iff $B\cup\mathbf{C}$ is consistent. (this is an a bit simplified explanation which does not handle the cases of an inconsistent **C** or B)
One can imagine two different ways to use such a system:

1. the safe way:
We compute for a given cct a useful control set and afterwards we start refutations. The disadvantadge of such a system is obvious: It is no trivial task to compute a control set, because it requires the computation of all refutations in **W** joined with the kernels of **D**. This will in general not be decidable and there will be infinitly many refutations. As a matter of fact, nearly all non-monotonic deduction systems suffer from this computational problems.
2. the revising way:
We start with the empty control set (everything may be assumed) and infer new constraint clauses. In case we infer an empty clause, we will enlarge the control set in order to make the inference of this empty clause impossible. Then we have to withdraw all the conclusions whose constraints are not consistent with the new control set. This seems to be a rather unsophisticated way to reason. It will lead to beliefs, the negations of which could be proved, if we computed a bit longer. Such a system would sometimes believe in propositions and the system could prove them to be false. Nevertheless, the system can give its reasons for believing the assumptions.

Let us have a schematical view of the way the three parts interact:

There are six possible types of interaction among the three components, when reasoning among formulae. These are indicated with arrows in the picture. In the general case, we only consider two of these (the bold arrows).




– resolution between clauses and kernels of clauses
– the test between constraints of derived clauses and the control set.

We do not consider the remaining four cases:

– there is no test between derived clauses and the constraints of these clauses. They may be contradicting each other.
– there is no test between constraints which occur during derivation in one single expansion, i.e. we can assume contradicting formulae.
– there is also no test between derived clauses and the control set, i.e. we may derive clauses, which are not allowed to be assumed.
– finally there can appear the rather awkward case that we have contradicting literals in the control set.

As a matter of fact, we can have two extreme cases of ccts where at least two of the above connections are realised implicitly:

1. The signatures of the constraints and the kernels could be disjoint. By that we have modelled the resolution with restricted quantifiers of H.J.Bürckert [4] The only difference is that we interpret his constraint theory in a different, non-monotonic, way as our control set. He infers an empty clause, which has to have a constraint that is valid in the constraint theory, whereas we only demand the satisfiability of this constraint.
   By that we are able to deduce more than H.J.Bürckert, which goes hand in hand with our non-monotonic philosophy:
   infer as long as it is consistent with your view of the world
2. The second extreme arises when in all constrained clauses kernel and constraint are identical. In that case, we get some sort of belief revision system. All formulae represent beliefs, i.e. the expansions of **D** become belief set. We demand consistency of these sets. If a new constraint clause is inserted, we will have to contract our belief set in order to avoid inconsistencies which would arise by expanding our belief set with the new clause. It can be seen that in this special case our system is a base belief revision system as formalized by Gärdenfors [5].

In the first of the two cases, above, a test between deduced clauses and assumptions made or a test between deduced clauses and the control set is unnecessary, because they have no common predicate symbols. In the second extreme these tests are established automatically.
More interesting will be the cases between the two extremes, where kernels and constraints have partly the same predicates but are not identical. This especially will be ccts, which correspond to normal and semi-normal default theories.

## 3 A calculus for useful control sets

In the following we will consider how a useful control set of a cct could be computed. This will be done by infering empty clauses from **W** together with the kernels of **D** and shift the constraints of these empty clauses to the control set. If all of the empty clauses are handled, we should have an useful control set.
So these are the inference rules, which will be provided:

1) **resolution** on (W,D):

$$\frac{\begin{array}{c} D_1, K \| B_1 \\ D_2, L \| B_2 \end{array}}{\sigma(D_1 \cup D_2) \| \sigma(B_1 \cup B_2)}$$

where $\sigma$ is a *mgu* of K and $\neg L$.

2) **factorization:**
factorize on the kernel of the constrained clause

$$\frac{L, L', K_1, .., K_n \| B}{\sigma(L, K_1, ..., K_n) \| \sigma(B)} \quad \text{and } \sigma(L) = \sigma(L')$$

3) **empty clause shift:**

**if** a constrained empty clause $\square \| \{B_1, ..., B_n\}$ has been derived

**and** for every consistent subset **C'** of **C** and every $1 \leq i \leq n$: $B_i \cup \mathbf{C}'$ is consistent

**then** choose an i with 1 $\leq$i$\leq$n and enlarge **C**:
$\mathbf{C}$:=$\mathbf{C} \cup \{\neg B_i\}$

4) **deletion:**
**if C** is consistent
**then** delete constrained clauses F||B with the following property:
$\mathbf{C} \models \neg B$
{i.e. every instantiation of B is inconsistent with **C**}

This calculus can be used to compute useful control sets. If no empty clause with constraint satisfiable in **C** can be computed, we can no longer draw any contradicting conclusions from our theory. It is of course in most cases not decidable whether all empty clauses with satisfiable constraints have been computed, so that the calculus will rather be used to derive conclusions and to withdraw some of these in case of contradiction.
In the rule of *empty clause shift* one has to choose between different constraints, which have to be satisfied for the empty clause. The decision made at this moment is the decision between different expansions.
In the rule of *deletion*, only constrained clauses are deleted, which have not no instantiation that can be satisfied in the control set. This can be interpreted in a way that constrained clauses are wished to be true in as many instantiations as possible and are considered useless only if there exists no possible instantiation for it. The constrained clauses are clauses, where the exceptions are stored in the control set.

# 4 Comparison to default logic

## 4.1 Extensions and Expansions

One might be tempted to identify Reiters extensions with our expansions. It may appear that for default theories which can be expressed as constrained clause theories, expansions and extensions correspond to each other. In the non-normal case, this cannot possibly be, because as we already know, there always exists an expansions but there are default theories, which do not have any extension.
A famous example for such a default theory (as given e.g. by Besnard [2]) is the following:

$\mathbf{W}$={P}
$\mathbf{D} = \{\frac{P:R}{Q}, \frac{P:S}{\neg Q}\}$

The corresponding cct is of the form:

$\mathbf{W}$={{P}}
$\mathbf{D}=\{\{\neg P, Q \| R\}, \{\neg P, \neg Q \| S\}\}$

With our calculus we can infer the constrained empty clause:

$\Box \| R \wedge S$

There are two different control sets which can be created:

$\mathbf{C_1} = \{\neg R\}$
$\mathbf{C_2} = \{\neg S\}$

This effect is due to our separation of the theory in which we reason and the theory which controls our reasoning.
We see that our system is especially effective for theories where formulae and constraints are of different signatures, as in this example. This is not to our surprise: we developed the system out of the logic with restricted quantifiers, where this separation is demanded.

As for the normal case, we can show by a simple example that the desired correspondence between expansion and extension does not hold.

Consider the following default theory:

$\mathbf{W}=\emptyset$, $\mathbf{D}=\{\frac{A(a):B(a)}{B(a)}\}$

This theory has just one single extension, the set of all tautologies: Th($\emptyset$). There is no extension for this theory which entails $A(a) \rightarrow B(a)$.

Now consider this theory as constrained clause theory:

$\mathbf{W} = \emptyset$, $\mathbf{D} = \{\{\neg A(a), B(a) \| B(a)\}\}$

A useful control set for this theory is the empty set, eventually we get the expansion Th($\{\neg A(a) \vee B(a)\}$), so that $A(a) \rightarrow B(a)$ is an element of this expansion.

It can be seen that our approach in contrast to Reiter's integrates the defaults more closely in the theory and considers them more as formulae rather than as deduction rules like Reiter does.
This can be seen even more characteristically in the following example:

$\mathbf{W} = \{\neg B\}, \mathbf{D} = \{\frac{A:B}{B}\}$

The only extension of this default theory is: Th$(\neg B)$.
In the corresponding cct $\mathbf{W}$=$\{\neg B\}, \mathbf{D} = \{\neg A, B \| B\}$ with the empty control set we get the expansion Th$(\{\neg A, \neg B\})$, so we do have $\neg A$ in this expansion.
Again, the deduction rule of Reiter's default logic is interpreted as an implicational formula so that its contraposition is also valid and allows us to infer.
As a matter of fact, we are able to simulate default rules where the contrapositive is not valid.

In the above examples, we have always seen extensions that were subsets of expansions. The question arises as to whether this is generally the case. This indeed can be shown:

**Theorem:**
Every extension of a default theory is a subset of an expansion of the corresponding cct.
**Proof:**
The extension of a default theory is always the set of all consequences of **W** together with the consequences of some generating defaults:

$\mathbf{E}$=Th$(\mathbf{W} \cup CONS(GD(\mathbf{E}))$ (Th 2.5 Reiter)

Construct a control set **C** so that for every $\delta \in GD(\mathbf{E})$ it holds: $\beta_1 \wedge ... \wedge \beta_n$ is consistent with **C**.

### 4.2 Cumulativity

The following example shows that Reiter's formalism does not check the assumptions he has to make for the conclusions in one extension for consistency with each other:

$\mathbf{W}$=$\{$A$\}$
$\mathbf{D} = \{\frac{A:B}{C}, \frac{C:\neg B}{D}\}$

This default theory has a single extension:
$\mathbf{E}$=Th$(\{$A,C,D$\})$

This is the fact, although it had to be assumed for this deduction that both B and ¬B hold. Brewka' s answer to this was to introduce a check between assumptions. Our system shows in such cases exactly the same behaviour as Reiter's does; nevertheless we also are able to prove cumulativity, which was denoted by Makinson [6] as:

If $A \mid\!\sim y$ *then* $A \mid\!\sim x$ *iff* $A \cup \{y\} \mid\!\sim x$.

In our notation this gets:

**Theorem**
*if* there is a $\mathbf{E} \in \mathbf{X(W,D)}$ with $A \in \mathbf{E}$
*then:*

$$\mathbf{F} \in \mathbf{X}(\mathbf{W},\mathbf{D}) \ \textit{and}\ A \in \mathbf{F}$$
$$\textit{iff}$$
$$\mathbf{F} \in \mathbf{X}(\mathbf{W} \cup \{A\}, \mathbf{D}).$$

It can be seen that this is an even stronger kind of cumulativity than that of Brewka. We do not consider the made assumptions of a derived proposition in our definition of cumulativity. A proposition A derived via the constrained clause $\{A||B\}$ allows us not only to enlarge our theory with this constraint clause but also with the unconstraint clause $\{A\}$.
This fulfills our aim of having the constraints only internally for controlling the deductions but appearing externally only in the moment of crisis. Evidently this makes the fundamental difference to Brewka's approach.

### 4.3 Skolemization

Up to now we considered only formulae in clause form, that means that existential quantified variables are already skolemised. The skolemizations are known to cause some trouble in default theories, as they introduce new constant symbols and the context these new constants are introduced in, is lost. In the following we will consider two examples that give us an idea of how the different cases of skolem constants occur in a cct.

The first example is the standard birds example. We give the following cct.

$$\begin{aligned}\mathbf{W}=\{&\{\neg P(x), B(x)\},\\ &\{\neg \mathrm{P}(x), \neg \mathrm{Fl}(x)\},\\ &\{\mathrm{P(tweety)}\},\\ &\{\mathrm{B(fred)}\}\}\\ \mathbf{D}=\{&\{\neg B(x), \mathrm{Fl}(x) || \mathrm{Fl}(x)\}\}\end{aligned}$$

The only empty clause that can be derived in this theory is:
$\Box || \mathrm{Fl(tweety)}$

Therefore, we create the control set: $\mathbf{C}=\{\neg\mathrm{Fl(tweety)}\}$

Now we want to prove that birds can fly in this theory. So we try to prove this indirectly with resolution. The negation of our sentence is:
$\exists x B(x) \wedge \neg \mathrm{Fl}(x)$

By skolemization we get two clauses:

$\{B(a)\}$, $\{Fl(a)\}$

Trying to find a refutation with this two new clauses we get:

$\Box || Fl(a)$

The constraint Fl(a) is consistent with C, iff **Not** a=tweety. We can derive that all birds can fly if they are not our special friend Tweety.
Furthermore we would like to stress that it can be derived that fred can fly and tweety cannot fly.

The second example due to Baader& Hollunder [1] will again lead to a constrained empty clause where a Skolem constant has to satisfy a certain condition.

$\forall x[\mathrm{Ad}(x) \to \exists y(\mathrm{Spouse}(x,y) \wedge \mathrm{Woman}(y)) \vee \mathrm{Bachelor}(x)]$
Spouse(tom,mary)
Woman(mary)
Adult(tom)
$\mathrm{Woman}(x) || \mathrm{Woman}(x)$

From the first sentence we get two clauses:

$\neg\mathrm{Adult}(x)$, $\mathrm{Spouse}(x,\mathrm{s}(x))$,$\mathrm{Bachelor}(x)$
$\neg\mathrm{Adult}(x)$,$\mathrm{Woman}(\mathrm{s}(x))$,$\mathrm{Bachelor}(x)$

with the new Skolem function s.
For this theory we create the control set:

**C**={Woman(mary)}

If we now want to prove if tom is a Bachelor we will try to find a refutation with ¬Bachelor(tom). We will compute the empty clause:

$\Box || \mathrm{Woman}(\mathrm{s}(\mathrm{tom}))$

The constraint of this empty clause is only satisfiable with our control set if **Not** s(tom)=mary. If we interpret the Skolem function s correctly, i.e. with its intended meaning we will get no proof for tom being a bachelor. The advantage of our approach is that we explicitly record the conditions, which have to be satisfied for a proof completion.

## 5 Comparison to Poole's Theorist

In Poole's system [7] the user provides two sets of formulae: **F** the set *facts*, i.e. undefeasible formulae and $\Delta$ the set of *possible hypotheses*, i.e. defeasible formulae. An extension of **F**, $\Delta$ is the set of logical consequences of a maximal consistent set **F** joined with ground instances of $\Delta$.
We can rewrite his system by having **W** identified with the clause form of **F** and **D** identified with

$\{A||A$ :where A is in the clause form of $\Delta\}$.

With this reformulation it is possible to simulate the examples of Pooles Theorist.

## References


1. Baader, F. and B. Hollunder (1992) *Embedding Defaults into Terminological Knowledge Representation Formalisms*, in *Principles of Knowledge Representation and Reasoning - Proceedings of the 3rd International Conference*, B. Nebel, C. Rich and W. Swartout, eds.
2. Besnard, Ph. (1989) *An Introduction to Default Logic.* (Springer)
3. Brewka, G. (1991) Cumulative Default Logik: in defense of nonmonotonic inference rules, *Artificial Intelligence* **50**: 183-205.
4. Bürckert, H.-J. (1991) *A Resolution Principle for a Logic with Restricted Quantifiers*, Lecture Notes in Artificial Intelligence **568** (Springer).
5. Gärdenfors, P. (1988) *Knowledge in flux: modelling the dynamics of epistemic states*. Cambridge (MIT Press).
6. Makinson, D. (1989) *General Theory of Cumulative Inference* in *Proceedings Second International Workshop on Non-Monotonic Reasoning*, Grassau, Germany, Lecture Notes in Artificial Intelligence **346** (Springer).
7. Poole, D. (1988) A Logical Framework for Default Reasoning, *Artificial Intelligence* **36**: 27-47.
8. Poole, D., R.G. Goebel and R. Aleliunas (1987) *Theorist: A logical reasoning system for default and diagnosis* in N. Cerone and G. McCalla (Eds.), *the Knowledge Frontier: Essays in the Representation of Knowledge*:331-352 (Springer)
9. Reiter, R. (1980) A logic for default reasoning, *Artificial Intelligence* **13**: 81-132.


This article was processed using the LaTeX macro package with LLNCS style