Genetic Algorithms for Extension Search in Default Logic

Pascal Nicolas and Frédéric Saubion and Igor Stéphan
LERIA, Université d’Angers
2 Bd Lavoisier
F-49045 Angers Cedex 01
{Pascal.Nicolas,Frederic.Saubion,Igor.Steplan}@univ-angers.fr

Abstract
A default theory can be characterized by its sets of plausible conclusions, called its extensions. But, due to the theoretical complexity of Default Logic ($\Sigma^p_2$ - complete), the problem of finding such an extension is very difficult if one wants to deal with non trivial knowledge bases. Based on the principle of natural selection, Genetic Algorithms have been quite successfully applied to combinatorial problems and seem useful for problems with huge search spaces and when no tractable algorithm is available. The purpose of this paper is to show that techniques issued from Genetic Algorithms can be used in order to build an efficient default reasoning system. After providing a formal description of the components required for an extension search based on Genetic Algorithms principles, we exhibit some experimental results.

Introduction
Default Logic has been introduced by Reiter in order to formalize common sense reasoning from incomplete information, and is now recognized as one of the most appropriate framework for non monotonic reasoning. In this formalism, knowledge is represented by a default theory from which one tries to build some extensions, that is a set of plausible conclusions. But, due to the level of theoretical complexity of Default Logic, the computation of these extensions becomes a great challenge.

Technical Background
Default Logic is a non monotonic logic since the sets of conclusions (theorems) does not necessary grow when the set of premises (axioms) does, as it is always the case in classical logic. In Default Logic, such a maximal set of conclusions is called an extension of the given default theory ($W, D$) where $W$ is a set of first order formulas representing the sure knowledge, and $D$ a set of default rules (or defaults). A default $\delta = \gamma$ is an inference rule providing conclusions relying upon given, as well as absent information meaning “if the prerequisite $\alpha$ is proved, and if for all $i = 1, \ldots, n$ each justification $\beta_i$ is individually consistent (in other words if nothing proves its negation) then one concludes the consequent $\gamma$”. For a default rule $\delta$, $\text{Prereq}(\delta)$, $\text{Justif}(\delta)$ and $\text{Conseq}(\delta)$ respectively denotes the prerequisite, the set of justifications and the consequent of $\delta$. These definitions will be also extended to sets
theory and Defaults that occur in the generating default set are said to be applied and every generating default set is grounded.

Definition 4 \cite{Schwind1994} Given a default theory \((W, D)\), a set of default \(\Delta \subseteq D\) is grounded if \(\Delta\) can be ordered as the following sequence \(< \delta_1, \ldots, \delta_n>\) satisfying the property:

\[
\forall i = 1, \ldots, n, W \cup \text{Conseq}\{\delta_1, \ldots, \delta_{i-1}\} \vdash \text{Prereq}(\delta_i)
\]

Now, we briefly recall the Genetic Algorithms concepts we use. We have to adapt some basic techniques and modify some definitions to fit our context but we refer the reader to \cite{Michalewicz1996} for a survey.

Since Genetic Algorithms are based on the principle of natural selection, vocabulary issued from natural genetics will be used in the Genetic Algorithms framework. We first consider a population of individuals which are represented by their chromosome. Each chromosome represents a potential solution to the given problem. The semantics of a chromosome (called its phenotype) has to be defined externally by the user. Then, an evaluation process and genetic operators determine the evolution of the population in order to get better and better individuals.

A genetic algorithm consists of the following components:

- a representation of the potential solutions: in most cases, chromosomes will be strings of bits representing its genes,
- a way to create an initial population,
- an evaluation function \(\text{eval}\): the evaluation function rates each potential solution w.r.t. the given problem,
- genetic operators that define the composition of the children: two different operators will be considered: Crossover allows to generate two new chromosomes (the offspring) by crossing two chromosomes of the current population (the parents), Mutation arbitrarily alters one or more genes of a selected chromosome,
- parameters: population size \(p_{\text{size}}\) and probabilities of crossover \(p_c\) and mutation \(p_m\).

We now present the general mechanism. Chromosomes, denoted \(G_i\), are strings of bits of length \(n\). The initial population is created by generating \(p_{\text{size}}\) chromosomes randomly. Starting from this initial population, we have to define a selection process for the next population and how to apply genetic operators.

The selection process presented here is based on an ordering of the individuals w.r.t. their evaluation. This process slightly differs from the initial definition of selection in \cite{Michalewicz1996} which is based on the construction of a roulette wheel by scaling.

- for each chromosome \((G_i), i \in \{1..p_{\text{size}}\}\), calculate \(\text{eval}(G_i)\),
• order the population according to evaluation rates; note that identical individuals occur only once in this classification.

Then, an intermediate population is constructed by selecting chromosomes according to the following method:
• consider the ordered list of the different chromosomes,
• a decreasing number of occurrences of each chromosome is put in the selected population w.r.t. the place of the chromosome in this ordered list. For instance the best rated chromosome will be represented \( N \times \) times in this selected population, while next chromosome will occur \( N-1 \) times and so on...
• this repartition in this population is user-defined but should satisfy that its size is equal to \( p_{\text{size}} \).

This principle is illustrated on the example of Figure 1 where the evaluation corresponds to the number of 1 in the chromosome. Furthermore, the best chromosome is duplicated 4 times in the selected population, the second 3 times, the third 2 times and the fourth only once. Due to the extension of the order, one can remark that, even if their rating is the same, the chromosome (10010) is selected once while (01001) is selected twice. This is due to the fact that (01001) is greater than (10010) in the ordering. This example only shows how individuals are selected from a population to be involved in reproduction and mutation.

Therefore genetic operators will be now apply on this selected population. Crossover is performed in the following way:
• select randomly two chromosomes in the selected population
• generate randomly a number \( r \in [0,1] \)
• if \( r > p_c \) then the crossover is possible;
  – select a random position \( p \in \{1, \ldots, n-1\} \)
  – the two chromosomes \( (a_1, \ldots, a_p, a_{p+1}, \ldots, a_n) \) and \( (b_1, \ldots, b_p, b_{p+1}, \ldots, b_n) \) are replaced by the two new chromosomes \( (a_1, \ldots, a_p, b_{p+1}, \ldots, b_n) \) and \( (b_1, \ldots, b_p, a_{p+1}, \ldots, a_n) \) as shown in Figure 2.
• if the crossover does not occur then the two chromosomes are put back in the selected population.

The mutation is defined as:
• For each chromosome \( G_i, i \in \{1, p_{\text{size}}\} \) and for each bit \( b_j \) in \( G_i \), generate a random number \( r \in [0,1] \),
• if \( r > p_m \) then mutate the bit \( b_j \) (i.e. flip the bit).

This full process is repeated to generate successive populations and one has to define the number of populations to be explored. The best chromosome of each

Remark that the evaluation function provides a partial order on chromosomes which is arbitrarily extended to any total order.
population w.r.t. the evaluation function represents the current best solution to the problem.

Clearly, the main difficulty of defining a Genetic Algorithms based search lies in the choice of the population's representation and in the definition of the evaluation process. A lot of work has also to be done in order to get a fine tuning of the different parameters $p_{\text{size}}, p_c, p_m$. Concerning our particular problem, these steps will be fully detailed in the next section.

**Formal Description of the System**

Our purpose is to construct an extension of a given default theory $(W, D)$ w.r.t. Definition [1]. We call candidate extensions the possible solutions to our problem. According to the principles of Genetic Algorithms, we now consider a population of individuals representing candidate extensions.

A naive approach could consist in considering the underlying set of atomic propositions induced by the signature of the default theory. Thus, the chromosomes would represent a kind of truth table:

**Example 2** With the signature $a, b, c, d$ an individual $G$,

$$G = (1 0 1 1 0 0 0 0)$$

represents the candidate extension $\text{Th}([a, c, d])$.

It is clear that due to the basic definition of Default Logic for a default $\frac{a \cdot b}{c}$ either $b$ and $\neg b$ has to be represented in the chromosome since in Definition [2] one has to check that $\neg b \not\in S$ but this is not equivalent to $b \in S$. Consider the following default theory $(W, D)$ with $W = \{a\}$ and $D = \{\frac{a \cdot b}{c}, \frac{\neg a \cdot \neg c}{\neg d}\}$. It has only one extension $\text{Th}([a, c, d])$ which does not contain $b$ neither $\neg b$. This representation will produce a lot of inconsistent candidate extensions because both $b$ and $\neg b$ can be marked as potentially valid as it is specified in $G$.

Therefore, it seems impossible to insure the efficiency and the convergence of the mechanism. One solution could be to introduce a three-valued logic representation but, in this case chromosomes cannot be strings of bits and require a more complicated encoding.

To avoid these drawbacks, another approach consists in focusing on the defaults more than on their consequences (according to Definition [3]). Moreover, this approach seems to be natural since an extension is completely determined by its generating default set. The following definitions set out a common formal framework which consists of a representation scheme and of an evaluation process.

**Representation**

A representation consists of the following elements:

- a chromosome language $G$ defined by a chosen size $n$,
- an interpretation mapping to translate chromosomes in term of possibly applied defaults, which provides the semantics of the chromosomes.

In this context, the chromosome language $G$ is the regular language $(0 + 1)^n$ (i.e. strings of $n$ bits). Given a chromosome $G \in \mathcal{G}$, $G_i$ denotes the value of $G$ at occurrence $i$.

The mapping can be formally defined as:

**Definition 5** Given a default theory $(W, D)$ and chromosome language $\mathcal{G}$, an interpretation mapping is defined as:

$$\phi: \mathcal{G} \times D \rightarrow \{\text{true, false}\}$$

A candidate extension $CE(W, D, G)$ is associated to each chromosome and can also be characterized by its candidate generating default set $CGD(W, D, G)$ (see Definition [3]). These two sets are easily defined w.r.t. the interpretation mapping.

**Definition 6** Given a default theory $(W, D)$, a chromosome $G \in \mathcal{G}$, the candidate extension associated to $G$ is:

$$CGD(W, D, G) = \{\delta_i | \phi(G, \delta_i) = \text{true}\}$$

**Definition 7** Given a default theory $(W, D)$, a chromosome $G \in \mathcal{G}$, the candidate extension associated to $G$ is:

$$CE(W, D, G) = \text{Th}(W \cup \{\text{Conseq}(\delta), \delta \in CGD(W, D, G)\})$$

$CE(W, D, G)$ and $CGD(W, D, G)$ will be simply denoted $CE(G)$ and $CGD(G)$ when it is clear from the context. Remark that since we have to compute the set of logical consequences, a theorem prover will be needed in our system. We now comment two different possible representations according to the previous definitions.

- Given a set of defaults $D = \{\delta_1, \ldots, \delta_n\}$ we can choose to encode in the chromosome the fact that the default is applicable. In this case the size of the chromosome corresponds to the cardinality of $D$ (i.e. $n$) and the interpretation function is defined as:

  $$\forall \delta_i \in D, \phi(\delta_i) = \begin{cases} \text{true} & \text{if } G_i = 1 \\ \text{false} & \text{if } G_i = 0 \end{cases}$$

  The main problem with this representation is its sensitiveness to mutation and crossover since a bit flipping in the chromosome induces a great change in the candidate extension. To refine this, we suggest another solution.

- For each default $\frac{\alpha_1, \ldots, \alpha_k}{\beta_1, \ldots, \beta_m}$ we encode in the chromosome the prerequisite $\alpha$ and all justifications $\beta_1, \ldots, \beta_m$ conjointly. Given a set of defaults $D = \{\delta_1, \ldots, \delta_n\}$ the size of the chromosome will be $2n$ and its semantics is given by the interpretation mapping:

  $$\forall \delta_i \in D, \phi(\delta_i) = \begin{cases} \text{true} & \text{if } G_i^{\psi_{2i-1}} = 1 \text{ and } G_i^{\psi_{2i}} = 0 \\ \text{false} & \text{in other cases} \end{cases}$$

  Intuitively, for a default $\delta_i$, if $G_i^{\psi_{2i-1}} = 1$ then its prerequisite is considered to be in the candidate extension and if $G_i^{\psi_{2i}} = 0$ no negation of its justifications is assumed to belong to the candidate extension. This representation is chosen for the remaining of this paper.
Example 3 Let consider a default theory $(W, D)$ where $D = \{ \frac{a \cdot b}{c}, \frac{a \cdot c}{\lnot b} \}$ and $W = \{ a \}$. We get $CGD(100011) = \{ \frac{a \cdot b}{c} \}$ and then $CE(100011) = Th(\{a, c\})$ which is really an extension but also $CGD(101011) = \{ \frac{a \cdot c}{\lnot b} \}$ and $CE(101011) = Th(\{a, c, \lnot b\})$ which is not an extension (negations of the justification of the two defaults are in the set).

Once the representation has been settled, one has to describe the evaluation process and then to run the genetic algorithm principles over the population of chromosomes.

Evaluation

An evaluation can be defined as:

Definition 8 Given a chromosome language $\mathcal{G}$, an evaluation function is a mapping $eval: \mathcal{G} \rightarrow A$, where $A$ is any set such that there exists a total ordering $\prec$ on it (to achieve the selection process).

Here, the evaluation function is mainly based on the definition of the extension. Different problems can be identified providing different evaluation criteria.

For a default $d_i = \frac{\alpha_i: \beta_1, ..., \beta_n}{\gamma_i}$, an intermediate evaluation function $f$ is defined in Table 1. Given the two positions $G_{|i-1}$ and $G_i$ in the chromosome associated to the default $d_i$, the first point is to determine w.r.t. these values if this default is supposed to be involved in the construction of the candidate extension (i.e. its conclusion has to be added to the candidate extension or not). Then, we check if this application is relevant.

| Case | $G_{|i-1}$ | $G_i$ | $CG(G) \vdash \alpha_i$ | $\exists j. CE(G) \vdash \beta_j$ | $H$ |
|------|-----------|------|-----------------|----------------|----|
| 1    | 1         | 1    | true            | false          | n  |
| 2    | 1         | 0    | true            | true           | y  |
| 3    | 1         | 0    | false           | true           | y  |
| 4    | 1         | 0    | false           | false          | y  |
| 5    | 1         | 1    | true            | true           | n  |
| 6    | 1         | 1    | true            | true           | n  |
| 7    | 1         | 1    | false           | true           | n  |
| 8    | 1         | 1    | false           | false          | n  |
| 9    | 0         | 1    | true            | false          | y  |
| 10   | 0         | 1    | true            | true           | n  |
| 11   | 0         | 1    | false           | true           | n  |
| 12   | 0         | 0    | false           | false          | n  |
| 13   | 0         | 0    | true            | false          | y  |
| 14   | 0         | 0    | true            | true           | n  |
| 15   | 0         | 0    | false           | true           | n  |
| 16   | 0         | 0    | false           | false          | n  |

Table 1: Evaluation

A $y$ in the penalty column $H$ means that a positive value is assigned to $f(G_{|i-1}, G_i)$. Note that only cases 1 to 4 correspond to default considered to be applied (i.e. such that $\phi(\delta) = true$).

Comments on penalties

- Cases 2,3,4:
  The consequence $\gamma_i$ is in the candidate extension (because $G_{|2i-1} = 1$ and $G_{|2i} = 0$) while the default should not have been applied (because either $CE(G) \not\vdash \alpha_i$ or $\exists j, CE(G) \vdash \beta_j$).

- Cases 5,9,13:
  The consequence of the default is not in $CE(G)$ while it should since the prerequisite of the default is in the extension and no negation of justifications is deducible from it.

- Other cases:
  Even if the chromosome value does not agree with the generated candidate extension, these cases can be ignored since they do not affect the extension.

At last, due to the minimality condition in the extension Definition 1 we have also to take into account the cardinality of $CGD(G)$ (noted $card(CGD(G))$). Thus, we can define the evaluation function as:

$eval: \mathcal{G} \rightarrow \mathbb{N} \times \mathbb{N}$

$eval(G) = (\sum_{i \in \{1..n\}} f(G_{|2i-1}, G_{|2i}), card(CGD(G)))$

where $n=card(D)$

The ordering for the selection process is the lexicographic extension ($<, <$) of the natural ordering $<$ on $\mathbb{N}$.

Correctness of the Evaluation

We examine now what we have to do when the evaluation function attributes a value $(0,.$) to a chromosome $G$. First, let us remark that every candidate extension $E = CE(W, D, G)$ is based on the generating default set $CGD(W, D, G)$. Since $eval(G) = (0,.)$, we can easily conclude that for every default $\frac{\alpha: \beta_1, ..., \beta_n}{\gamma}$ in $CGD(W, D, G)$ we have $\alpha \in E$ and $\lnot \beta_i \notin E, \forall i = 1, \ldots, n$. But it is not sufficient to prove that $E$ is truly an extension of the default theory $(W, D)$ as shown in the following counter-example.

Example 4 Let $(W, D) = (\emptyset, \{ \frac{a \cdot c}{b}, \frac{c \cdot \lnot b}{a} \})$ be a default theory and $G = (1010)$. Then, the candidate extension is $E = CE(W, D, G) = Th(\{a, b\})$ and $eval(G) = (0,.)$. But, it is obvious that $E$ is not an extension of $(W, D)$ that has only one extension $: Th(\emptyset)$.

In fact, the counter-example illustrates that our evaluation function does not capture the groundedness (see Definition 1) of the generating default set of a candidate extension. So, when the evaluation function gives a chromosome with a null value, we have to check if the corresponding generating default set is grounded. If it is the case our following result ensures that we have found an extension. If not, the algorithm continues to search a new candidate.

Theorem 1 Let $(W, D)$ be a default theory, $G$ a chromosome and a candidate generating default set $\Delta = CGD(W, D, G)$.
Some particular types of defaults can be treated apart to improve the system.

- A default $(\frac{\gamma i}{\beta i})$ has not to be specifically encoded in the chromosome language and can be removed from the initial set of defaults. Since as soon as this default can be applied it blocks itself. One has only to check that for each candidate extension $CE(G)$ either $\alpha \notin CE(G)$ or $\neg \beta \notin CE(G)$. Moreover, we focus on this kind of defaults because they are very interesting in certain cases. For instance, a default $(\frac{\gamma i}{\beta})$ “keeps” only extensions that contain $\neg \beta$. This property is often used in the graph problem encoding described in [Cholewiński et al. (1999)].

- $\delta i = \frac{\alpha_i, \beta_i, \ldots, \gamma_i}{\gamma_i}$ with $W \vdash \alpha_i$ : then for every chromosome $G$ we impose $G|2i = 1$.

- $\delta i = \frac{\alpha_i, \beta_i, \ldots, \gamma_i}{\gamma_i}$ with $W \vdash \neg \beta_i$ : for some $j$ then for every chromosome $G$ we impose $G|2j = 1$.

### Experimental Results: the GADEL System

Our whole system GADEL (Genetic Algorithms for DEfault Logic) can be schematized by the Figure 3.

![Figure 3: System](image)

It is implemented in Sicstus Prolog and it is described with more details in [Stephan, Saubion, & Nicolas (2000)].

Basically, DeRes [Cholewiński et al. (1999)] and our system GADEL use a common approach in their search for an extension of a default theory $(W, D)$ : they both use a generate and test procedure. They explore the search space $2^D$ and check if a subset $DG \subset D$ can be the generating default set of an extension of $(W, D)$. But, DeRes explores the search space with an ad-hoc backtracking procedure while GADEL uses the Genetic Algorithms principles in order to reach as quickly as possible some “good” candidates. [Cholewiński et al. (1999)] describes the very good performances of DeRes on some kind of default theories : the stratified ones. But it is also noticed that for a non stratified default theory, as for the Hamiltonian cycle problem, the performance of DeRes is not enough to deal with a non very few number of defaults.

| problem       | GADEL | $T_G$ | $T_D$ | DeRes |
|---------------|-------|-------|-------|-------|
| boy           | 3.3   | 15.4  | 3600  |
| girl          | 3.4   | 15.6  | 3600  |
| man           | 5.3   | 22.5  | 3600  |
| woman         | 3.0   | 14.6  | 3600  |
| man $\land$ student | 186.7 | 467.5 | 3600 |
| woman $\land$ student | 271.6 | 704.4 | 3600 |
| ham.b.3,2,0,0,1,0,0 | 1.8   | 5.6   | 0.5   |
| ham.b.4,2,0,0,1,0,0 | -     | 3600  | 19.4  |
| ham.b.5,2,0,0,1,0,0 | -     | 3600  | 566.4 |
| ham.b.6,2,0,0,1,0,0 | -     | 3600  | 3600  |

Table 2: Experimental results

In Table the first column gives the used default theories. For the first lines it shows the formula $f$ added to the theory people (the whole description of this example is given in appendix) and for the last ones it shows which Hamiltonian cycle problem we have used (the encoding of the problem is furnished by Theory-Base [Cholewiński et al. 1999]). The second and third columns respectively give average number of generations $NG$, and average time $T_G$ in seconds to obtain one extension of $(W \cup \{f\}, D)$ by GADEL (the parameters of the genetic algorithm are $p_c = 0.8$, $p_m = 0.1$, $p_{size} = 325$ for people problems, and $p_{size} = 465$ for the Hamiltonian problems, the number of tests is 100). The fourth column gives the time $T_D$ spent by DeRes to solve the problem with the full prover option. Note that all these problems are not stratified.

We give in [Stephan, Saubion, & Nicolas 2000] a finer analysis of our experiments but results given in this table shows that DeRes has a lot of difficulties with our taxonomic example People (even if we use the local prover). Conversely the number of generations are quite small for GADEL (even if the time is not so good: all the implementation is written in Prolog). But, on its turn, GADEL has poor performances on Hamiltonian problems. We think that it is because we do not take into account the groundedness into our evaluation func-
tion. As a matter of fact, in the Hamiltonian problem, a solution is exactly one “chain” of defaults, but, there is a lot of potential solutions (whose evaluation is null) based on two, or more, chains of defaults. The only criterion to discard these candidate generating default sets is the groundedness property that they do not satisfy. Conversely, in people example, a solution is a set of non conflicting defaults, but at most four defaults are chained together, and so the groundedness property is less important to reach a solution.

Conclusion

The general method described in this paper provides a new framework in order to search for extensions of a Default Logic theory, by using Genetic Algorithms techniques. This new approach allows us to quickly generate good candidate extensions and experimental results are promising w.r.t. other systems. Moreover, the validity of our method is ensured by a theoretical correctness result.

Now, a first point to examine is to integrate the groundedness property in the evaluation function, but we have to take care to not much increase the computation time. The efficiency could be improved by combining other search techniques like local search heuristics. An another important feature of our approach is its ability to be parallelized. In fact, the evaluation of the whole population and its genetic manipulations can be distributed on several processors without fundamental difficulties. These points will be explored in a future work.

Appendices

Proof of the theorem

**Theorem 1** Let $(W, D)$ be a default theory, $G$ a chromosome and a candidate generating default set $\Delta = CGD(W, D, G)$.

$\text{eval}(G) = 0$ and $\Delta$ is groundediff

$(W, D)$ has an extension $E = \text{Th}(W \cup \text{Conseq}(\Delta))$ of which $\Delta$ is the generating default set.

**Proof**

$\leftrightarrow$: Let $E = \text{Th}(W \cup \text{Conseq}(\Delta))$ be an extension of $(W, D)$. Since $\Delta$ is the generating default set of $E$, it is obviously grounded. Let us suppose that $\text{eval}(G) > 0$. Then, according to the definition of our evaluation function (see Table 1), it means that there exists a default $\delta = \alpha \rightarrow \beta_1, \ldots, \beta_n \in D$ for which a penalty has been assigned. Let us examine the two possible cases:

• $\delta \in \Delta$: penalties can arise from cases 2, 3 or 4, but no one of them is possible since $E \not\vdash \alpha$ and $E \not\models \beta_i, \forall i = 1, \ldots, n$ by definition of a generating default set.

• $\delta \notin \Delta$: penalties can arise from cases 5, 9, or 13, but no one of them is possible since it would indicate that $\delta$ should be a generating default of $E$.

Thus $\text{eval}(G) = 0$.

$\rightarrow$: Let $\Delta = CGD(W, D, G)$ and $E = \text{Th}(W \cup \text{Conseq}(\Delta))$.

Since $\Delta$ is grounded, we can order it like $\Delta = \langle \delta_1, \ldots, \delta_p \rangle$ and we have the property

$$\forall i = 1, \ldots, p,$$

$$W \cup \text{Conseq}(\delta_1, \ldots, \delta_{i-1}) \vdash \text{Prereq}(\delta_i)$$

that is equivalent to

$$\forall i = 1, \ldots, p,$$

$$\text{Prereq}(\delta_i) \in \text{Th}(W \cup \text{Conseq}(\delta_1, \ldots, \delta_{i-1}))$$

from which we can build the sequence

$$E_0 = W$$

$$E_{i+1} = \text{Th}(E_i) \cup \{ \text{Conseq}(\delta_i) \}, \forall i = 0, \ldots, p-1$$

Because of the groundedness of $\Delta$, there is no difficulty to transform the previous sequence in the following way.

$$E_0 = W$$

$$(*)E_{i+1} = \text{Th}(E_i) \cup \{ \text{Conseq}(\delta_i) | \text{Prereq}(\delta_i) \in E_i \}, \forall i = 0, \ldots, p-1$$

Since $\text{eval}(G) = 0$, we can deduce :

$$\forall i, \exists \beta_i \in \text{Justif}(\delta_i), \neg \beta_i \notin E$$

and then we can reformulate $(*)$ like that

$$E_0 = W$$

$$(**)E_{i+1} = \text{Th}(E_i) \cup \{ \text{Conseq}(\delta_i) | \text{Prereq}(\delta_i) \in E_i, \beta_i \in \text{Justif}(\delta_i), \beta_i \notin E \},$$

$$\forall i = 0, \ldots, n-1$$

From $\text{eval}(G) = 0$, we can also deduce that for all other defaults $\alpha_1, \alpha_2, \ldots, \alpha_n \in D \setminus \Delta$, we have either $\alpha \notin E$, either $\exists j, \neg \beta_j \in E$. So, in $(**)$ we can delete the explicit reference to $i$ in the defaults and we can extend the sequence for all positive integer. So we have

$$E_0 = W$$

$$E_{k+1} = \text{Th}(E_k) \cup \{ \text{Conseq}(\delta) | \text{Prereq}(\delta) \in E_k, j \in \text{Justif}(\delta), j \notin E \}, \forall k = 0$$

Finally, let us remark that by construction $E$ is exactly the set $\bigcup_{k=0}^{\infty} E_k$. Thus we have obtain here the pseudo iterative characterization of an extension given in Definition 2, and we can conclude that $E$ is an extension of $(W, D)$. \qed
People example

This is the description of our examples.

Mikitiuk, A.; and Truszczynski, M. 1999. Computing with default logic. Proceedings of the International Joint Conference on Artificial Intelligence, 312–318. Morgan Kaufmann Publishers.

Stephan, I.; Saubion, F.; and Nicolas, P. 2000. Description of godel. In Proceedings of the special session System Description at Workshop on NonMonotonic Reasoning, Breckenridge, USA.

References

Antoniou 1997] Antoniou, G. 1997. Nonmonotonic Reasoning. MIT Press.

Besnard 1989] Besnard, P. 1989. An Introduction to Default Logic. Symbolic Computation — Artificial Intelligence. Springer Verlag.

Cholewiński et al. 1999] Cholewiński, P.; Marek, V.; Mikitiuk, A.; and Truszczynski, M. 1999. Computing with default logic. Artificial Intelligence 112:105–146.

Gottlob 1992] Gottlob, G. 1992. Complexity results for nonmonotonic logics. Journal of Logic and Computation 2(3):397–425.

Michalewicz 1996] Michalewicz, M. 1996. Genetic Algorithms + Data Structures = Evolution Programs. Springer Verlag.

Niemelä 1995] Niemelä, I. 1995. Towards efficient default reasoning. In Mellish, C., ed., Proceedings of the International Joint Conference on Artificial Intelligence, 312–318. Morgan Kaufmann Publishers.

Reiter 1980] Reiter, R. 1980. A logic for default reasoning. Artificial Intelligence 13(1-2):81–132.