Abduction via C -tableaux and δ -resolution

**Fernando Soler-Toscano^{*} — Ángel Nepomuceno-Fernández^{*}
Atocha Aliseda-Llera^{**}**

^{*}*Depto. Filosofía, Lógica y Filosofía de la Ciencia.
Universidad de Sevilla (Spain)
{fsoler, nepomuce}@us.es*

^{**}*Instituto de Investigaciones Filosóficas.
Universidad Nacional Autónoma de México (Mexico)
atocha@filosoficas.unam.mx*

ABSTRACT. *The formalization of abductive reasoning has received increasing attention from logicians. However, few work is found beyond abduction in propositional logic, given that in a first order formalism, the undecidability problem naturally appears, and therefore an abductive problem cannot even be appropriately formulated. Still, many applications in artificial intelligence allow finite domains to work with, and this gives an opportunity to apply abduction in first order logic with restricted domains. In this paper, we present an approach to abductive reasoning in C -structures, first order structures with a finite domain in which each of its elements has a corresponding constant representing its interpretation. By using semantic tableaux with bounded depth, that is, C -tableaux (a modification of N -tableaux, introduced by the authors in previous papers) and δ -resolution calculus, we build an effective procedure for the searching of minimal abductive solutions within the proposed semantics.*

KEYWORDS: *abductive reasoning, resolution, semantic tableaux, first order logic, finite models.*

DOI:10.3166/JANCL.19.211–225 © 2009 Lavoisier, Paris

1. Introduction

Explanatory reasoning, or abduction, is usually characterised as the inference that goes from a *conclusion* to a set of *premises*. Given a surprising (unexplained) fact $\mathcal{C}$, if we know that it could be obtained from $\mathcal{A}$ (with respect to a background theory), we may suppose $\mathcal{A}$ as a *plausible hypothesis*:

$$\frac{\mathcal{C}, \mathcal{A} \rightarrow \mathcal{C}}{\mathcal{A}} \quad (1)$$

Rule (1), unsound in classical logic (it is a well known fallacy), shows the *retroductive* character of abduction (abduction is commonly described as backwards deduction). Artificial Intelligence is full of areas in which abduction is applied, such as planning, diagnosis, language understanding and knowledge assimilation. Its formalization has received increasing attention from logicians. However, few work is found beyond abduction in propositional logic, given that in a first order formalism, the undecidability problem naturally appears, and therefore an abductive problem cannot even be appropriately formulated, much less solved. In general, it is not possible to determine, for a given *abductive problem*, whether a proposed hypothesis is an appropriate *abductive solution*.

Still, many applications in artificial intelligence require finite domains to work with, such as in the field of *Answer Set Programming* (ASP), in which it is frequent to make use of the reduction of certain satisfiability problems in first order logic to domains with finite cardinality. It is very usual that a theory is formulated using a first-order formalism, in order to achieve a certain level of generality, but its applications can be done through finite reductions. For example, a theory for the Blocks World can be formulated in first order logic, but when applying it to a planning problem with only four blocks it can be reduced to a finite domain with only four elements. We follow a similar strategy in this paper but within an abductive framework. The resulting abductive solutions may not be valid for the original first order version of the abductive problem, but they are surely valid for its corresponding finite version. The added novelty is that we combine two calculi used previously, but separately, to solve abductive problems. These calculi are based on semantic tableaux and resolution (Reyes-Cabello *et al.*, 2006; Soler-Toscano *et al.*, 2006b). Our approach is specially useful and hopefully interesting for abductive applications in contexts with finite domains.

The way to combine these calculi is that semantic tableaux are used to reduce abductive problems to finite domains and then, in a propositional frame, we use δ -resolution calculus (dual to standard resolution), to search for abductive solutions. What we do is to apply to abduction an original idea from (Avron, 1993) where tableaux and resolution are combined in a deductive framework in order to get a better efficiency. Tableaux are used to obtain a clausal forms, in a more efficient way than constructing prenex normal forms, and then resolution is applied. We have translated these ideas to abduction and the result is an abductive process which obtains all the *explanatory* and *minimal* abductive solutions (in the sense of (Aliseda, 2006)) for a given abductive problem. The approach we are presenting was implemented in the system *tarfa* (Soler-Toscano *et al.*, 2006a), and shows that, as in the case of deduction, the combination of tableaux and resolution also produces a better efficiency than the use of tableaux or resolution separately. But, for space limitations, we did not provide there the logical foundation of the procedures that *tarfa* follows. We devote this paper to that issue.

In section 2, we start by a series of basic definitions (δ -clause, δ -clausal form, C -structure and its corresponding notions for C -satisfiability, C -validity, C -logical con-

sequence and C -equivalence, together with definitions of an abductive problem and its solutions), which set up the framework to introduce the proposed C -tableaux calculus in section 3. This calculus is based on a proposed modification of our N -tableaux (introduced by us in (Reyes-Cabello *et al.*, 2006)), which in turn are a modification of the well-known classical semantic tableaux, this time restricted to a set of constants, what characterizes a class of C -structures. A set of lemmas and a corollary give place to our *Fundamental Theorem of C -tableaux*, which states that a δ -clausal form of a C -tableau $\mathcal{T}(\Theta)_C$, denoted by $\mathcal{C}\delta(\mathcal{T}(\Theta)_C)$, and Θ are C -equivalent. These results are then used in section 4, in which we introduce the δ -resolution calculus and prove that it is indeed δ -sound, δ -complete, and δ -abductive-complete (theorems 22, 24 and 26). These results establish an equivalence between the δ -resolution calculus and our semantic notion of C -validity. We then introduce a series of definitions (subsumption, saturation by δ -resolution) and corollaries which pave the way for our *Fundamental Theorem of δ -resolution*, our result concerning the calculation of abductive solutions given an abductive problem in our setting. All previous results are then essential for section 5, in which we describe the promised abductive procedure combining C -tableaux and δ -resolution, and show it is a correct procedure. We end by offering some conclusions of this work in final section 6.

2. The semantics of C -structures

By $\mathcal{L}$ we denote a first order language without identity and functors, defined as usual. Concerning the semantics, we restrict the interpretation to sentences, so we do not consider assignments to free variables. Given a sequence $t_1, \dots, t_n$ of terms (constants and variables) and the n -ary predicate P , we say that $Pt_1, \dots, t_n$ and $\neg Pt_1, \dots, t_n$ are complementary literals (positive and negative, respectively). Given any literal λ , its complementary is represented by $\bar{\lambda}$. We will use lower-case Greek letters to represent formulae and upper case Greek letters for sets of these.

DEFINITION 1 (C -STRUCTURE). — Given $C = \{c_1, \dots, c_n\}$, a non-empty finite set of constants of $\mathcal{L}$, we define a C -structure as a structure $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ (with universe of discourse $\mathcal{D}$ and interpretation function $\mathcal{I}$) such that, besides the usual requirements, verifies the following:

- $|\mathcal{D}| = n$, and
- If $i \neq j$, $1 \leq i, j \leq n$, then $\mathcal{I}(c_i) \neq \mathcal{I}(c_j)$.

DEFINITION 2. — Given a set of constants C and $\alpha, \beta \in \mathcal{L}$, if C contains all the constants of α and β , we say that:

- α is C -satisfiable iff there is a C -structure $\mathcal{M}$ such that $\mathcal{M} \models \alpha$.
- α is C -valid iff every C -structure $\mathcal{M}$ verifies $\mathcal{M} \models \alpha$; in symbols $\models_C \alpha$.
- β is a C -logical consequence of α iff for every C -structure $\mathcal{M}$ if $\mathcal{M} \models \alpha$ then $\mathcal{M} \models \beta$; in symbols $\alpha \models_C \beta$.
- α and β are C -equivalent iff $\alpha \models_C \beta$ and $\beta \models_C \alpha$.

Note that the C -validity plays the role of the *universal validity* in the semantics of C -structures. These notions can be generalized to sets of formulae, and also to δ -clauses and δ -clausal forms, given the following definitions.

DEFINITION 3 (δ -CLAUSE). — A δ -clause $\Sigma = \{\lambda_1, \dots, \lambda_n\}$ is a finite set of literals of $\mathcal{L}$ such that all of its terms are constants. Given a structure $\mathcal{M}$, $\mathcal{M} \models \Sigma$ iff $\mathcal{M} \models \lambda_i$ for every literal λ_i , $1 \leq i \leq n$. $\diamond$ denotes the empty δ -clause, which is universally valid.

DEFINITION 4 (δ -CLAUSAL FORM). — A δ -clausal form $A = \{\Sigma_1, \dots, \Sigma_n\}$ is a finite set of δ -clauses. Given a structure $\mathcal{M}$, $\mathcal{M} \models A$ iff $\mathcal{M} \models \Sigma_i$ for at least one δ -clause Σ_i , $1 \leq i \leq n$. The empty δ -clausal form is not satisfiable.

These definitions are dual to those of clause and clausal form in standard resolution calculus. A δ -clause is equivalent to an elementary conjunction (as opposed to a disjunction) of literals, and a δ -clausal form to a disjunctive (as opposed to a conjunctive) normal form. This duality with standard resolution will produce some interesting results like theorem 22, a soundness theorem (to be presented in section 4), in which a δ -clause Σ obtained by the δ -resolution rule from a δ -clausal form A , makes A a C -logical consequence from Σ .

DEFINITION 5 (ABDUCTIVE PROBLEM). — Given a finite $\Theta \subset \mathcal{L}$ and $\varphi \in \mathcal{L}$, (Θ, φ) is an abductive problem iff $\Theta \not\models \varphi$ and $\Theta \not\models \neg\varphi$.

DEFINITION 6 (ABDUCTIVE SOLUTION). — Given an abductive problem (Θ, φ) , a δ -clause Σ is a C -abductive solution for it iff:

- 1) $\Theta \cup \Sigma \models_C \varphi$.
- 2) $\Theta \cup \Sigma$ is C -satisfiable.
- 3) $\Sigma \not\models_C \varphi$.
- 4) There is no δ -clause $\Sigma' \subset \Sigma$ such that $\Theta \cup \Sigma' \models_C \varphi$.

$\text{Abd}(\Theta, \varphi)_C$ denotes the set of C -abductive solutions for the abductive problem (Θ, φ) .

3. The C -tableaux calculus

C -tableaux calculus derives from a modification of semantic tableaux (Beth, 1955) by (Boolos, 1984) and (Díaz Estévez, 1993). It has been used for abduction in (Nepomuceno, 2002; Reyes-Cabello *et al.*, 2006), as an extension of the abductive approach originally presented in (Cialdea Mayer *et al.*, 1993; Aliseda, 1997). In this paper, we define C -tableaux with respect to the set C of constants which defines a class of C -structures, rather than the cardinality of the searched models.

DEFINITION 7 (C -TABLEAU). — Given a finite set $\Theta \subset \mathcal{L}$ (the root of the C -tableau) and a non-empty finite set of constants $C = \{c_1, \dots, c_n\}$ consisting of all the

Table 1. Classification of formulae of types α , β , γ and δ , and their components

α	α_1	α_2	β	β_1	β_2	γ	$\gamma(c)$
$\eta \wedge \lambda$	η	λ	$\eta \vee \lambda$	η	λ	$\forall x\eta$	$\eta(x/c)$
$\neg(\eta \vee \lambda)$	$\neg\eta$	$\neg\lambda$	$\neg(\eta \wedge \lambda)$	$\neg\eta$	$\neg\lambda$	$\neg\exists x\eta$	$\neg\eta(x/c)$
$\neg(\eta \rightarrow \lambda)$	η	$\neg\lambda$	$\eta \rightarrow \lambda$	$\neg\eta$	λ		
			δ	$\delta(c)$			
			$\exists x\eta$	$\eta(x/c)$			
			$\neg\forall x\eta$	$\neg\eta(x/c)$			

constants of Θ , a C -tableau of Θ , denoted by $\mathcal{T}(\Theta)_C$, is a semantic tableau¹ which differs from a Beth tableau only on rules γ and δ (see below). The tableau construction starts from an initial tree consisting of one branch with one formula of Θ at each node. A branch is closed when it contains two complementary literals λ and $\bar{\lambda}$. A non-closed branch is an open one. The construction of the tableaux finishes (in which case the tableau is finished) when all its branches are finished; that is, when all of its branches are closed (in which case it is a closed tableau) or else when one of the following rules has been applied to every non-literal formula of its open branches:

1) α rule. When this rule is applied to a formula of type α (see table 1), both components α_1 and α_2 are added to the same branch of the α formula:

$$\frac{\alpha}{\alpha_1, \alpha_2}$$

2) β rule. The application of this rule splits the branch which contains the β formula into two new branches. One of the subformulae β_1 , β_2 is added to each of the branches (in a graphical representation of the tableau, β_1 and β_2 become the two children of the last formula of the original branch):

$$\frac{\beta}{\beta_1 \mid \beta_2}$$

3) γ rule. When this rule is applied to a formula of type γ , one instance of $\gamma(x/c_i)$ is introduced in the same branch of γ , for each constant c_i of C :

$$\frac{\gamma}{\gamma(c_1), \dots, \gamma(c_n)}$$

4) δ rule. This rule splits the current branch in a similar way that the β rule does. But in this case, n new branches are produced, one for each constant c_i of C , and one instance of $\delta(x/c_i)$ is added to each of these:

$$\frac{\delta}{\delta(c_1) \mid \dots \mid \delta(c_n)}$$

1. Given $\Theta = \{\theta_1, \dots, \theta_n\} \subset \mathcal{L}$, both $\mathcal{T}(\Theta)_C$ and $\mathcal{T}(\{\theta_1, \dots, \theta_n\})_C$ denote the same C -tableau.

5) *Double negation rule.* This rule can be applied to any formula $\neg\neg\eta$, so η is added to the same branch:

$$\frac{\neg\neg\eta}{\eta}$$

DEFINITION 8. — Given a C -tableau $\mathcal{T}(\Theta)_C$, if it contains an open finished branch consisting of literals $\lambda_1, \dots, \lambda_n$, we denote $\Sigma = \{\lambda_1, \dots, \lambda_n\}$ the δ -clause of such branch.

DEFINITION 9. — Given a finished C -tableau $\mathcal{T}(\Theta)_C$, we define its δ -clausal form $C\delta(\mathcal{T}(\Theta)_C)$ as that containing the δ -clauses of all open branches of $\mathcal{T}(\Theta)_C$.

LEMMA 10. — Given a δ -clause Σ , if it does not contain complementary literals, then Σ is C -satisfiable for each C containing all constants of the literals in Σ .

PROOF 11. — Let Σ be a set of literals with the mentioned properties. As it does not contain complementary literals, then there is a C -structure $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ satisfying all of its literals. Take $\mathcal{D} = C$. Moreover, let $\mathcal{I}(c) = c$ for every $c \in C$, and for every predicate P of $\mathcal{L}$, $\mathcal{I}(P)$ is the smallest set such that for every positive literal $P(c_1, \dots, c_n) \in \Sigma$, $(c_1, \dots, c_n) \in \mathcal{I}(P)$. It is then easy to check that $\mathcal{M} \models \Sigma$. ■

COROLLARY 12. — A δ -clause Σ of any open branch of a C -tableau is C -satisfiable.

PROOF 13. — Let Σ be the set of literals of an open branch of a C -tableaux. By definition 7, all terms of Σ are constants of C , and Σ does not contain complementary literals. Then, by lemma 10, Σ is C -satisfiable. ■

LEMMA 14. — If Σ is the δ -clause of an open branch of a C -tableau $\mathcal{T}(\{\theta_1, \dots, \theta_m\})_C$, then

$$\Sigma \models_C \theta_1 \wedge \dots \wedge \theta_m$$

PROOF 15. — Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ be a C -structure that satisfies Σ , the set of literals of an open branch of $\mathcal{T}(\{\theta_1, \dots, \theta_m\})_C$. Take $C = \{c_1, \dots, c_n\}$. We prove that $\mathcal{M}$ satisfies all formulae of the open branch, by induction over their logical degree². In the base case, formulae with logical degree equal to 0 are literals, which are satisfied by $\mathcal{M}$, as we have defined it. Now, suppose that $\mathcal{M}$ satisfies all formulae of the branch with logical degree less or equal to i . Let λ be a formula of the branch with logical degree $i + 1$. Then λ is one of the following:

- A negative literal (just in case that $i = 0$). But $\mathcal{M}$ has been constructed so that it satisfies all literals in the branch, so it satisfies λ too.
- A double negation $\neg\neg\epsilon$. Then, to complete the construction of the C -tableau, the double negation rule was applied to λ , so ϵ is in the branch and its logical degree is less or equal to i . So $\mathcal{M} \models \epsilon$, and by evaluation of $\neg$, $\mathcal{M} \models \lambda$.

2. The *logical degree* of a formula is a function $\sigma : \mathcal{L} \rightarrow \mathbb{N}$ such that: $\sigma(\lambda) = 0$, for every positive literal λ , and for any α and β in $\mathcal{L}$, $\sigma(\neg\alpha) = \sigma(\forall x\alpha) = \sigma(\exists x\alpha) = 1 + \sigma(\alpha)$ and $\sigma(\alpha \wedge \beta) = \sigma(\alpha \vee \beta) = \sigma(\alpha \rightarrow \beta) = 1 + \sigma(\alpha) + \sigma(\beta)$.

- A formula of type α . Then, its two components should be in the branch, and both have a logical degree less or equal to i , so both are satisfied by $\mathcal{M}$. By evaluation of the connectives of the formulae of type α , we have that $\mathcal{M} \models \lambda$.
- A formula of type β . Then, one of its components is in the branch, and $\mathcal{M}$ satisfies it as its logical degree is less or equal to i . By evaluation of the connectives of formulae of type β , $\mathcal{M} \models \lambda$.
- A formula of type γ , as $\forall x\varphi$ (when λ is $\neg\exists x\varphi$ the proof is analogous). Then, as the γ rule was applied during the construction of the C -tableau, all the formulae of type $\varphi(x/c_j)$, $1 \leq j \leq n$, are in the branch and their logical degree is less or equal to i , so they are satisfied by $\mathcal{M}$. Also, as for every element $d \in \mathcal{D}$ there is a constant c_j such that $\mathcal{I}(c_j) = d$, by evaluation of $\forall$ it is verified that $\mathcal{M} \models \lambda$.
- A formula of type δ , as $\exists x\varphi$ (the proof is analogous when λ is $\neg\forall x\varphi$). Then, the application of the δ rule put in the branch a formula $\varphi(x/c_j)$, $1 \leq j \leq n$, with logical degree less or equal to i , so satisfied by $\mathcal{M}$. Then, by evaluation of $\exists$, $\mathcal{M} \models \lambda$.

Hence, $\mathcal{M}$ satisfies all formulae in the branch, so also $\mathcal{M} \models \theta_k$, for every k between 1 and m . By evaluation of $\wedge$, $\mathcal{M} \models \theta_1 \wedge \dots \wedge \theta_m$. ■

LEMMA 16. — *Every C -satisfiable finite set $\Theta \subset \mathcal{L}$ verifies the following:*

- 1) *Any C -structure which satisfies Θ , satisfies all formulae of at least one branch of $\mathcal{T}(\Theta)_C$.*
- 2) *$\mathcal{T}(\Theta)_C$ is an open tableau.*

PROOF 17. — Let Θ be a C -satisfiable set of formulae, and $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ a C -structure which satisfies Θ . We prove that $\mathcal{M}$ satisfies all formulae of at least one branch of $\mathcal{T}(\Theta)_C$, by induction over the number of applications of tableau rules. In the base case, with 0 applications, the only branch of the C -tableau contains only the formulae of Θ , satisfied by $\mathcal{M}$. Now, suppose that $\mathcal{M}$ satisfies all the formulae of a certain branch until the i^{th} rule application. Let us consider the $(i+1)^{\text{th}}$ application of a rule to a certain formula λ in the construction of the tableau. When the rule does not affect the branch under analysis, it is trivial that $\mathcal{M}$ satisfies all of its formulae after the application. Else, we show what happens, according to the type of the applied rule:

- Double negation rule. λ is $\neg\neg\lambda'$, and λ' is added to the branch. By hypothesis, $\mathcal{M} \models \lambda$ and then, by evaluation of $\neg$, $\mathcal{M} \models \lambda'$, so $\mathcal{M}$ satisfies all formulae in the branch after the application of the $(i+1)^{\text{th}}$ rule.
- The α rule. λ is $\lambda_1 \wedge \lambda_2$ (or $\neg(\lambda_1 \vee \lambda_2)$, etc.), and consequently λ_1 and λ_2 (or $\neg\lambda_1$ and $\neg\lambda_2$, etc) are added to the branch. In any case, by hypothesis $\mathcal{M}$ satisfies λ , so by evaluation of logical connectives $\mathcal{M}$ satisfies all formulae in the branch.
- The β rule. λ is $\lambda_1 \vee \lambda_2$, (or $\neg(\lambda_1 \wedge \lambda_2)$, etc.), and then the branch splits into two, and each one of these is extended with one of λ_1 and λ_2 (or $\neg\lambda_1$ and $\neg\lambda_2$, etc). By hypothesis $\mathcal{M} \models \lambda$, so by evaluation of connectives, $\mathcal{M}$ satisfies at least one of the new formulae, so $\mathcal{M}$ satisfies at least one of the two new branches.
- The γ rule. λ is $\forall x\varphi$ (if λ is $\neg\exists x\varphi$ the proof is analogous). So, when the rule γ is applied, all formulae $\varphi(x/c_j)$, $1 \leq j \leq n$, are added to the branch. By inductive

hypothesis, $\mathcal{M} \models \forall x\varphi$ so, by evaluation of $\forall$, $\mathcal{M} \models \varphi(x/c_j)$ for each constant c_j , $1 \leq j \leq n$. So, $\mathcal{M}$ satisfies the branch after the application of the γ rule.

– The δ rule. λ is $\exists x\varphi$ (if λ is $\neg\forall x\varphi$ the proof is analogous), and by hypothesis $\mathcal{M} \models \lambda$, that is, $\mathcal{M} \models \exists x\varphi$. As $\mathcal{M}$ assigns to each constant of $\mathcal{T}(\{\eta\})_C$ a different element of $\mathcal{D}$, then, by evaluation of $\exists$, for at least one constant c_j , $1 \leq j \leq n$, $\mathcal{M} \models \varphi(x/c_j)$. But one of the new branches that appeared after the application of the δ rule contains all formulae of the original branch plus $\varphi(x/c_j)$, so there is one branch such that all of its formulae are satisfied by $\mathcal{M}$.

When the construction of the C -tableau finishes, there is at least one branch such that all of its formulae are satisfied by $\mathcal{M}$, so the first part of lemma 16 is proved. Moreover, among the formulae of the branch which is satisfied by $\mathcal{M}$ there cannot be complementary literals. Therefore, the C -tableau is open, what proves the second part of the lemma. ■

THEOREM 18 (FUNDAMENTAL THEOREM OF C -TABLEAUX). — *Given $\eta \in \mathcal{L}$ and $\mathcal{T}(\{\eta\})_C$, a C -tableau of $\{\eta\}$,*

$$\mathcal{C}\delta(\mathcal{T}(\{\eta\})_C)$$

is C -equivalent to η .

PROOF 19. — Let $\mathcal{M}$ be a C -structure which satisfies η . Then, by lemma 16, $\mathcal{M}$ satisfies all formulae of at least one branch of $\mathcal{T}(\{\eta\})_C$. Then there cannot be complementary literals in such branch, so it is an open branch, and $\mathcal{M}$ satisfies its δ -clause Σ (definition 8). By definition 9, $\Sigma \in \mathcal{C}\delta(\mathcal{T}(\{\eta\})_C)$. By definitions 3 and 4 we have that $\mathcal{M} \models \mathcal{C}\delta(\mathcal{T}(\{\eta\})_C)$. Therefore, $\eta \models_C \mathcal{C}\delta(\mathcal{T}(\{\eta\})_C)$.

Now, let $\mathcal{M}$ be a C -structure which satisfies $\mathcal{C}\delta(\mathcal{T}(\{\eta\})_C)$. By definitions 3 and 4, $\mathcal{M}$ satisfies all literals of at least one open branch of $\mathcal{T}(\{\eta\})_C$. By lemma 14, $\mathcal{M} \models \eta$. So, $\mathcal{C}\delta(\mathcal{T}(\{\eta\})_C) \models_C \eta$. ■

4. Search through δ -resolution

Resolution calculus (Robinson, 1965) is perhaps the logical procedure most applied in automatic theorem proving. In this section we extend to the semantics of C -structures the δ -resolution calculus, introduced in (Soler-Toscano, 2005) and later developed in (Soler-Toscano *et al.*, 2006b) for propositional logic. We will use δ -resolution to obtain explanatory hypotheses for a certain set of formulae. We start by presenting the δ -resolution rule which, though syntactically identical with that of resolution, is semantically dual to the classical resolution rule, as δ -clauses are equivalent to the conjunction of their literals. It should be reminded that as we will work with δ -clausal forms without variables (after the reduction to finite models that will be done with C -tableaux), we do not need to introduce dual notions for variable renaming, unification, and the like.

DEFINITION 20 (δ -RESOLUTION RULE). — *Given two δ -clauses $\Sigma_1 \cup \{\lambda\}$ and $\Sigma_2 \cup \{\neg\lambda\}$, the δ -resolution rule produces their δ -resolvent $\Sigma_1 \cup \Sigma_2$:*

$$\frac{\Sigma_1 \cup \{\lambda\} \quad \Sigma_2 \cup \{\neg\lambda\}}{\Sigma_1 \cup \Sigma_2}$$

This rule is presented in the standard forward resolution rule format, but differs from it, for we are now working with δ -clauses. As it will be proved shortly in the δ -Soundness theorem 22, any δ -clausal form which contains $\Sigma_1 \cup \{\lambda\}$ and $\Sigma_2 \cup \{\neg\lambda\}$, is a logical consequence of $\Sigma_1 \cup \Sigma_2$. This gives to the δ -resolution calculus its abductive backward character, as it does not infer conclusions from premises but rather the reverse. In this case, every δ -clause Σ we obtain by applications of the δ -resolution rule from a given δ -clausal form A is not a logical consequence of A , but rather A is a logical consequence of Σ , that is, $\Sigma \models A$. So the δ -resolution calculus produces a premise Σ for a given consequence A .

DEFINITION 21 (PROOF BY δ -RESOLUTION). — *A δ -clause Σ is provable by the δ -resolution rule from a δ -clausal form A , denoted by $A \vdash_\delta \Sigma$, iff there exists a sequence of δ -clauses such that:*

- *Every δ -clause of the sequence belongs to A or is a δ -resolvent of previous δ -clauses.*
- *The last δ -clause is Σ .*

THEOREM 22 (δ -SOUNDNESS). — *Given the δ -clausal form A and the δ -clause Σ , if $A \vdash_\delta \Sigma$ then $\Sigma \models A$.*

PROOF 23. — The proof is by induction over the number of applications of the δ -resolution rule. In the base case, with 0 applications, $\Sigma \in A$, and by definition 4, $\Sigma \models A$. Suppose that until the n^{th} application, A is a logical consequence of each resulting δ -clause. Let $\Sigma = \Sigma_1 \cup \Sigma_2$ be the $(n+1)^{\text{th}}$ resulting δ -clause, a δ -resolvent of $\Sigma_1 \cup \{\lambda\}$ and $\Sigma_2 \cup \{\neg\lambda\}$. Then, each structure $\mathcal{M}$ which satisfies Σ satisfies also one of the previous δ -clauses $\Sigma_1 \cup \{\lambda\}$ or $\Sigma_2 \cup \{\neg\lambda\}$, as $\mathcal{M}$ satisfies each literal of $\Sigma_1 \cup \Sigma_2$ and $\mathcal{M} \models \lambda$ or $\mathcal{M} \models \neg\lambda$. Finally, by the induction hypothesis, $\mathcal{M}$ satisfies A . Hence, $\Sigma \models A$. ■

THEOREM 24 (δ -COMPLETENESS). — *For any δ -clausal form A , if $\models_C A$, then $A \vdash_\delta \Diamond$.*

PROOF 25. — Let A be a C -valid δ -clausal form, $s = |A|$ and t the sum of the cardinalities of all the δ -clauses of A . We take $k = t - s$ and proceed by induction over the value of k , considering that $\Diamond \notin A$ (to avoid the trivial case, where it is obvious that $A \vdash_\delta \Diamond$). If $k = 0$, A is a set of unitary δ -clauses (with only one literal). Then, as for any C -structure $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ the value of $\mathcal{I}$ is different for each constant of A , if A is C -valid then there are in A two δ -clauses $\{\lambda\}$ and $\{\neg\lambda\}$. With only one application of the δ -resolution rule we get $\Diamond$.

Suppose that the theorem is verified for each $k \leq n$. We prove it for $k = n + 1$. Now, A contains a δ -clause $\Sigma = \{\lambda_1, \lambda_2, \dots, \lambda_m\}$, with $m \geq 2$. Then, we define

$\Sigma' = \{\lambda_1\}$ and $\Sigma'' = \{\lambda_2, \dots, \lambda_m\}$. But if A is C -valid, $(A - \Sigma) \cup \{\Sigma'\}$ and $(A - \Sigma) \cup \{\Sigma''\}$ are valid as well (this is a direct conclusion from definitions 3 and 4), and for such sets, $k \leq n$, so we can obtain $\Diamond$ from them, through the proofs (sequences of δ -clauses) $\mathcal{D}em'$ and $\mathcal{D}em''$, respectively. Let $\mathcal{D}em$ be a proof for A , built like $\mathcal{D}em''$, but each time that Σ'' is used in $\mathcal{D}em''$, Σ is used in $\mathcal{D}em$. As the only difference between Σ'' and Σ is that Σ contains λ_1 , the last δ -clause in $\mathcal{D}em$ is either $\Diamond$ (in this case $A \vdash_\delta \Diamond$, and the proof finishes) or $\{\lambda_1\}$. In the latter case, the proof contains all the δ -clauses of $(A - \Sigma) \cup \{\Sigma'\}$ (because Σ' is $\{\lambda_1\}$), and we can complete it as $\mathcal{D}em'$, in order to obtain $\Diamond$. ■

THEOREM 26 (δ -ABDUCTIVE COMPLETENESS). — *Let A be a δ -clausal form C -equivalent to α . Then, $A \vdash_\delta \Sigma$ for each δ -clause Σ such that:*

- 1) Σ is C -satisfiable.
- 2) $\Sigma \models_C \alpha$.
- 3) There is no δ -clause $\Sigma' \subset \Sigma$ such that $\Sigma' \models_C \alpha$.

PROOF 27. — Given $\alpha \in \mathcal{L}$, let A be a δ -clausal form C -equivalent to α which only contains constants of C and $\Sigma = \{\lambda_1, \dots, \lambda_n\}$ a δ -clause with the properties indicated above. We prove that $A \vdash_\delta \Sigma$. As $\Sigma \models_C \alpha$, we have that $\models_C \lambda_1 \wedge \dots \wedge \lambda_n \rightarrow \alpha$, because every C -structure either does not satisfy Σ or satisfies α (because $\Sigma \models_C \alpha$). By evaluation of $\rightarrow$ and $\neg$, we have that $\models_C \overline{\lambda_1} \vee \dots \vee \overline{\lambda_n} \vee \alpha$. Then, as $A \cup \{\{\overline{\lambda_1}\}, \dots, \{\overline{\lambda_n}\}\}$ is a δ -clausal form C -equivalent to $\overline{\lambda_1} \vee \dots \vee \overline{\lambda_n} \vee \alpha$, it is possible to obtain $\Diamond$ from it (theorem 24), through a proof that we call $\mathcal{D}em$. In a similar way that in the proof of theorem 24, we build a proof parallel to $\mathcal{D}em$, that we call $\mathcal{D}em'$, which only uses δ -clauses of A . Now, when a δ -clause of type $\{\overline{\lambda_i}\}$, $1 \leq i \leq n$, is used in $\mathcal{D}em$, nothing is done in $\mathcal{D}em'$. It is easy to observe that the last δ -clause of $\mathcal{D}em'$ is a $\Sigma' \subseteq \Sigma$, because each δ -clause $\{\overline{\lambda_i}\}$ can only eliminate the literal λ_i in the proof. But if $\Sigma' \subset \Sigma$, by theorem 22, we have that $\Sigma' \models A$, then $\Sigma' \models_C A$, and then $\Sigma' \models_C \alpha$, what is contrary to what we know of Σ . Then, $\Sigma' = \Sigma$, so $A \vdash_\delta \Sigma$. ■

DEFINITION 28 (SATURATION BY δ -RESOLUTION). — *Given the δ -clausal form A , the saturation set by δ -resolution of A , denoted by A^δ , is the smallest set containing every δ -clause Σ such that:*

- $A \vdash_\delta \Sigma$.
- Σ is satisfiable.
- There is no $\Sigma' \subset \Sigma$ such that $A \vdash_\delta \Sigma'$.

COROLLARY 29. — *Let A be a δ -clausal form C -equivalent to α , for a given set C of constants containing all terms of A and α . Then, A^δ is the set of δ -clauses such that $\Sigma \in A^\delta$ iff:*

- $\Sigma \models_C \alpha$.
- Σ is C -satisfiable.
- For every $\Sigma' \subset \Sigma$, $\Sigma' \not\models_C \alpha$.

PROOF 30. — By Theorem 26 we can be sure that every δ -clause Σ with the corresponding properties is provable by δ -resolution from A . So, by definition 28, Σ belongs to A^δ , given that if Σ is C -satisfiable then it is also satisfiable and there is no $\Sigma' \subset \Sigma$ such that $A \vdash_\delta \Sigma'$ (else, by theorem 22, $\Sigma' \models A$, and also $\Sigma' \models_C A$, what contradicts the properties of Σ). So, every δ -clause with the properties of Σ belongs to A^δ . Next let us prove that every δ -clause Σ of A^δ verifies the required properties of the corollary. First, given that $A \vdash_\delta \Sigma$, we have $\Sigma \models A$ (theorem 22) and so $\Sigma \models_C A$ and $\Sigma \models_C \alpha$. Moreover, by definition 28, Σ is satisfiable, what implies that it does not contain complementary literals. Furthermore, all terms in Σ are constants of C , so by lemma 10, we know that Σ is C -satisfiable. Next, suppose that there is a δ -clause Σ' , $\Sigma' \subset \Sigma$, such that $\Sigma' \models_C \alpha$; then there is a δ -clause Σ'' , $\Sigma'' \subset \Sigma$, such that $\Sigma'' \models_C \alpha$ and for all $\Sigma^* \subset \Sigma''$, $\Sigma^* \not\models_C \alpha$. But then, by theorem 26, $A \vdash_\delta \Sigma''$, what contradicts definition 28 with respect to Σ . Therefore, Σ verifies all properties required by the corollary. ■

THEOREM 31 (FUNDAMENTAL THEOREM OF δ -RESOLUTION). — *Given an abductive problem $(\{\theta_1, \dots, \theta_n\}, \varphi)$, if N_Θ and O are δ -clausal forms C -equivalent to $\neg(\theta_1 \wedge \dots \wedge \theta_n)$ and φ , respectively, and C contains all terms of N_Θ , O , φ and each θ_i , $1 \leq i \leq n$, then:*

$$Abd(\Theta, \varphi)_C = (N_\Theta^\delta \cup O^\delta)^\delta - (N_\Theta^\delta \cup O^\delta)$$

PROOF 32. — Let $\Theta = \{\theta_1, \dots, \theta_n\}$. Then, according to definition 6, $\Sigma \in Abd(\Theta, \varphi)_C$ is equivalent to the following items:

1) $\Theta, \Sigma \models_C \varphi$.

Note that this is equivalent to $\Sigma \models_C \neg(\theta_1 \wedge \dots \wedge \theta_n) \vee \varphi$ and to $\Sigma \models_C N_\Theta \cup O$, given that N_Θ and O are C -equivalent to $\neg(\theta_1 \wedge \dots \wedge \theta_n)$ and φ , respectively.

2) $\Theta \cup \Sigma$ is C -satisfiable.

This is equivalent to $\Sigma \not\models_C \neg(\theta_1 \wedge \dots \wedge \theta_n)$. This result allows us, in turn, to infer that Σ is C -satisfiable.

3) $\Sigma \not\models_C \varphi$.

4) There is no $\Sigma' \subset \Sigma$ such that $\Theta, \Sigma' \models_C \varphi$.

This is equivalent to the following: for every $\Sigma' \subset \Sigma$, $\Sigma' \not\models_C \neg(\theta_1 \wedge \dots \wedge \theta_n) \vee \varphi$.

First, let us consider that $\Sigma \in Abd(\Theta, \varphi)_C$. Then:

– From 1, 2 and 4 (these numbers correspond to the above enumeration), taking into account that $N_\Theta \cup O$ is C -equivalent to $\neg(\theta_1 \wedge \dots \wedge \theta_n) \vee \varphi$, we get, by corollary 29, $\Sigma \in (N_\Theta \cup O)^\delta$. Then, $\Sigma \in (N_\Theta^\delta \cup O^\delta)^\delta$ (the order of applications of the δ -resolution rule in a set A does not change the final A^δ).

– From 2 and 3 we get, by corollary 29, that $\Sigma \notin N_\Theta^\delta$ and $\Sigma \notin O^\delta$. Therefore, $\Sigma \notin (N_\Theta^\delta \cup O^\delta)$.

Hence, by these previous results, we conclude that

$$\Sigma \in (N_\Theta^\delta \cup O^\delta)^\delta - (N_\Theta^\delta \cup O^\delta) \quad (2)$$

Next, in order to prove $\Sigma \in \mathcal{A}bd(\Theta, \varphi)_C$, we assume (2), and we have the following:

- $\Sigma \in (N_\Theta^\delta \cup O^\delta)^\delta$. It is easy to see that this is equivalent to $\Sigma \in (N_\Theta \cup O)^\delta$, and given the C -equivalence between $N_\Theta \cup O$ and $\neg(\theta_1 \wedge \dots \wedge \theta_n) \vee \varphi$, and by corollary 29, we have $\Sigma \models_C \neg(\theta_1 \wedge \dots \wedge \theta_n) \vee \varphi$ (item 1 above) and for every $\Sigma' \subset \Sigma$, $\Sigma' \not\models_C \neg(\theta_1 \wedge \dots \wedge \theta_n) \vee \varphi$ (item 4).
- $\Sigma \notin N_\Theta^\delta$. Suppose that $\Sigma \models_C \neg(\theta_1 \wedge \dots \wedge \theta_n)$. Then, there is a $\Sigma' \subseteq \Sigma$ such that $\Sigma' \models_C \neg(\theta_1 \wedge \dots \wedge \theta_n)$ and for every $\Sigma^* \subset \Sigma'$, $\Sigma^* \not\models_C \neg(\theta_1 \wedge \dots \wedge \theta_n)$. But then, as Σ is C -satisfiable (lemma 10), and by corollary 29 we have that $\Sigma' \in N_\Theta^\delta$. So it is not possible that $\Sigma' = \Sigma$, but it is also impossible that $\Sigma' \subset \Sigma$, because this contradicts that $\Sigma \in (N_\Theta^\delta \cup O^\delta)^\delta$, as Σ would be subsumed by Σ' . So, $\Sigma \not\models_C \neg(\theta_1 \wedge \dots \wedge \theta_n)$ (item 2).
- $\Sigma \notin O^\delta$. In this case, by means of a *reductio ad absurdum* similar to the previous one, we conclude that $\Sigma \not\models_C \varphi$ (item 3).

These results prove items 1–4 above, those equivalent to $\Sigma \in \mathcal{A}bd(\Theta, \varphi)$. ■

COROLLARY 33. — *Let $\alpha, \beta \in \mathcal{L}$ be C -equivalent to δ -clausal forms A and B , respectively, so that each term of α , β , A and B belongs to C . Then $\alpha \models_C \beta$ iff for each $\Sigma \in A^\delta$ there is a $\Sigma' \in B^\delta$ such that $\Sigma' \subseteq \Sigma$.*

PROOF 34. — Suppose that $\alpha \models_C \beta$ and $\Sigma \in A^\delta$. Then, by corollary 29, Σ is C -satisfiable and $\Sigma \models_C \alpha$. But $\alpha \models_C \beta$, and so $\Sigma \models_C \beta$. So, there is a δ -clause $\Sigma' \subseteq \Sigma$ such that $\Sigma' \models_C \beta$, Σ' is satisfiable and for every $\Sigma^* \subset \Sigma'$, $\Sigma^* \not\models_C \beta$. Then, by corollary 29, $\Sigma' \in B^\delta$.

Now, suppose that for each $\Sigma \in A^\delta$ there is a $\Sigma' \in B^\delta$ such that $\Sigma' \subseteq \Sigma$. Let us prove that $\alpha \models_C \beta$. Let $\mathcal{M}$ be a C -structure such that $\mathcal{M} \models \alpha$. Then, by lemma 16, $\mathcal{T}(\{\alpha\})_C$ has at least an open branch with literals satisfied by $\mathcal{M}$. Let Σ^* be such δ -clause. By lemma 14, $\Sigma^* \models_C \alpha$. Then there is a δ -clause $\Sigma \subseteq \Sigma^*$ such that $\Sigma \models_C \alpha$ and for every $\Sigma'' \subset \Sigma$, $\Sigma'' \not\models_C \alpha$. Hence, by corollary 29, $\Sigma \in A^\delta$. Then, there is a $\Sigma' \subseteq \Sigma$ such that $\Sigma' \in B^\delta$. By corollary 29, $\Sigma' \models_C \beta$. Then, as $\mathcal{M} \models \Sigma'$ (because $\Sigma' \subseteq \Sigma \subseteq \Sigma^*$ and $\mathcal{M} \models \Sigma^*$), $\mathcal{M} \models \beta$. Hence, $\alpha \models_C \beta$. ■

5. Abductive procedure via C -tableaux and δ -resolution

The following procedure constitutes an abductive process which calculates abductive solutions within an strategy. The procedure analyzes the theory and the observation in order to evaluate whether $\mathcal{A}bd(\Theta, \varphi)_C$ is a genuine abductive problem, in which case it calculates its abductive solutions. All this is done via δ -resolution operations. The procedure is correct, as proved by corollary 36.

ALGORITHM 35. — *The searching process of C -abductive solutions for a proposed abductive problem (Θ, φ) , with $\Theta = \{\theta_1, \dots, \theta_n\}$, where $\Theta \subset \mathcal{L}$, $\varphi \in \mathcal{L}$, and all terms of Θ and φ belong to the set of constants C is done by the following steps:*

Step 1: Theory Analysis. Let N_Θ be $\mathcal{C}\delta(\mathcal{T}(\{\neg(\theta_1 \wedge \dots \wedge \theta_n)\})_C)$:

- If $N_\Theta = \emptyset$, then Θ is C -valid.
- Else, N_Θ^δ is obtained. If $\Diamond \in N_\Theta^\delta$, then Θ is not C -satisfiable.

Else,

Step 2: Observation Analysis. Let O be $\mathcal{C}\delta(\mathcal{T}(\{\varphi\})_C)$. Then:

- If $O = \emptyset$, then φ is not C -satisfiable.
- Else, O^δ is obtained. If $\Diamond \in O^\delta$, then φ is C -valid.

Else,

Step 3: Refutation search. If for every δ -clause $\Sigma \in O^\delta$ there is a $\Sigma' \in N_\Theta^\delta$ such that $\Sigma' \subseteq \Sigma$, then $\Theta \models_C \neg\varphi$.

Else,

Step 4: Explanations search. From N_Θ^δ and O^δ , the sets $(N_\Theta^\delta \cup O^\delta)$ and then $(N_\Theta^\delta \cup O^\delta)^\delta$ are obtained. Then, if $\Diamond \in (N_\Theta^\delta \cup O^\delta)^\delta$, $\Theta \models_C \varphi$.

Else, (Θ, φ) is an abductive problem, and the process returns the following:

$$Abd(\Theta, \varphi)_C = (N_\Theta^\delta \cup O^\delta)^\delta - (N_\Theta^\delta \cup O^\delta)$$

COROLLARY 36. — For every $\Theta \subset \mathcal{L}$ and $\varphi \in \mathcal{L}$, the searching process for obtaining C -abductive solutions (given a set of constants C including all terms of Θ and φ) which follows algorithm 35 given (Θ, φ) , is correct.

PROOF 37. — Given a formula $\alpha \in \mathcal{L}$ C -equivalent to the δ -clausal form A , $\Diamond \in A^\delta$ iff $\models_C \alpha$ (by theorems 22 and 24). Moreover, α is not C -satisfiable iff $\mathcal{C}\delta(\mathcal{T}(\{\alpha\})_C)$ is not satisfiable (by theorem 18). That is, iff $\mathcal{C}\delta(\mathcal{T}(\{\alpha\})_C) = \emptyset$ (by corollary 10 and lemma 14, $\mathcal{T}(\{\alpha\})_C$ does not have open branches). Given these observations, the proof of correction for steps 1 and 2 of algorithm 35 is straightforward. Step 3 is a consequence of corollary 33. For step 4, if $\Diamond \in (N_\Theta^\delta \cup O^\delta)^\delta$, then $N_\Theta^\delta \cup O^\delta$ is C -valid (theorem 22). But this δ -clausal form is C -equivalent to $\neg(\theta_1 \wedge \dots \wedge \theta_n) \vee \varphi$. Therefore, $\models_C \neg(\theta_1 \wedge \dots \wedge \theta_n) \vee \varphi$, which is equivalent in turn to $\Theta \models_C \varphi$. In the remaining case, the way in which step 4 constructs $Abd(\Theta, \varphi)_C$ is correct, by theorem 31. ■

6. Conclusions

Although first order abduction is, in general, undecidable, it makes sense to search for abductive solutions in structures with finite cardinality, for a significant amount of applications of abduction in Artificial Intelligence are given in closed domains, with well-known elements.

In this paper we proposed to tackle the problem of abduction within C -structures, first order structures with a finite domain in which each of its elements has a corresponding constant representing its interpretation. Moreover, we proposed a δ -resolution calculus for this semantics, and proved that it is δ -sound, δ -complete, and

δ -abductive-complete. Finally, we proposed a way to compute abductive solutions, combining C -tableaux and δ -resolution calculus. The former is used to reduce abductive problems to finite domains and then, in a propositional frame, we use δ -resolution calculus to search for abductive solutions. However, this kind of language has an advantage with respect to a simple propositional one, namely the logical structure of certain theories can be better represented by a first order language than by a propositional one. Indeed, first order formal languages with a semantics in terms of the finite model theory are commonly used in artificial intelligence. The proposed algorithm 35 returns, given an abductive problem (Θ, φ) , all C -abductive solutions which are *explanatory* and *minimal* abductive solutions, verifying those conditions of definition 6 (and in accord with a full description of abduction, as defined in (Aliseda, 2006)). These may seem to be quite strong conditions, but it is possible to weaken them without any sacrifice of the proposed framework.

Acknowledgements

Research for this paper was supported by subventions from the Spanish Ministry of Science and Innovation (Project HUM2007-65053) and from the Andalusian Council (Project P06-HUM-01538)

7. References

- Aliseda A., Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence, PhD thesis, Stanford University, 1997.
- Aliseda A., *Abductive Reasoning: Logical Investigations into Discovery and Explanation*, vol. 330 of *Synthese Library*, Springer, 2006.
- Avron A., “Gentzen-Type Systems, Resolution and Tableaux”, *Journal of Automated Reasoning*, vol. 10, num. 2, pp. 265-281, 1993.
- Beth E. W., “Semantic Entailment and Formal Derivability”, *Koninklijke Nederlandse Akademie van Wetenschappen, Proceedings of the Section of Sciences*, vol. 18, pp. 309–342, 1955.
- Boolos G., “Trees and finite satisfiability”, *Notre Dame Journal of Formal Logic*, vol. 25, pp. 193-197, 1984.
- Cialdea Mayer M., Pirri F., “First order abduction via tableau and sequent calculi”, *Bulletin of the IGPL*, vol. 1, pp. 99-117, 1993.
- Díaz Estévez E., “Árboles semánticos y modelos mínimos”, *Actas del I Congreso de la Sociedad de Lógica, Metodología y Filosofía de la Ciencia en España*, Universidad Complutense de Madrid, 1993.
- Nepomuceno A., “Scientific Explanation and Modified Semantic Tableaux”, in L. Magnani, N. Nersessian, C. Pizzi (eds), *Logical and Computational Aspects of Model-Based Reasoning*, Applied Logic Series, Kluwer Academic Publishers, pp. 181-198, 2002.

- Reyes-Cabello L., Aliseda-Llera A., Nepomuceno-Fernández A., “Abductive reasoning in first order logic”, *Logic Journal of the IGPL*, 2006.
- Robinson J., “A Machine-Oriented Logic Based on the Resolution Principle”, *Journal of the ACM*, vol. 12, pp. 23-41, 1965.
- Soler-Toscano F., Modelos Formales de Explicación en Lógica e Inteligencia Artificial, PhD thesis, Universidad de Sevilla, 2005.
- Soler-Toscano F., Nepomuceno-Fernández A., “`tarfa`: tableaux and resolution for finite abduction”, *Logics in Artificial Intelligence. 10th European Conference, JELIA 2006, Liverpool, UK, September 13-15, 2006, Proceedings*, vol. 4160 of *Lecture Notes in Computer Science*, Springer, 2006a.
- Soler-Toscano F., Nepomuceno-Fernández A., Aliseda-Llera A., “Model-Based Abduction via Dual Resolution”, *Logic Journal of the IGPL*, 2006b.