Natural Deduction System in Paraconsistent Setting: proof search for PCont.

Alexander Bolotov* and Vasilyi Shangin***

* School of Electronics and Computer Science
University of Westminster, UK.
A.Bolotov@wmin.ac.uk
http://www2.wmin.ac.uk/bolotoa/index.html
** Department of Logic, Moscow State University, Moscow, 119991, Russia.
shangin@philos.msu.ru

Abstract. This paper continues a systematic approach to build natural deduction calculi and corresponding proof procedures for non-classical logics. Our attention is now paid to the framework of paraconsistent logics. These logics are used, in particular, for reasoning about systems where paradoxes do not lead to the 'deductive explosion', i.e. where formulae of the type **false** $\Rightarrow A$, for any A, are not valid. We formulate the natural deduction system, explain its main concepts, define proof searching techniques and illustrate them by examples.

1 Introduction

When we speak about the reasoning tools related to modern computer systems we must take into account that these systems are complex, dynamic and heterogenous. Consider, for example, the problem of formation of heterogeneous resources into networks or into clouds. Conflicts of various types are inevitable here and very often a system functions quite well despite their present. This leads us to the necessity of equipping the system with reasoning techniques capable of coping with such conflicts. It is natural to think of a conflict as of an anomaly, some kind of a paradox, or simply of a contradiction. Classical reasoning is not appropriate here as it validates *ex falso quodlibet* the famous principle of deriving anything from a contradiction. If we obtain a specification, S, of a system with conflicts, and reason classically then S becomes trivial. Therefore, there is a need to develop deductive methods which makes it possible to reason about about paradoxical statements correctly, but at the same time without turning the S into trivial. We will enable then the system to identify, localise conflicts and to 'live with them' not violating its essential functionalities.

Classical reasoning is based on the assumptions that possible worlds cannot contain contradictions and are complete. If $Prop$ stands for the set of propositions and W for the set of possible worlds, then the former means that for every possible world $w \in W$, and any $\alpha \in Prop$, it is not possible that $\alpha \in w$

* The second author is supported by Russian Foundation for Humanities, project 10-03-00570a

and $\neg\alpha \in w$, while the latter principle suggests that for every $w \in W$, and any $\alpha \in Prop$, we require $\alpha \in w$ or $\neg\alpha \in w$. When the first principle, of bivalence, is not required we are led to the framework of *paraconsistency*.

In this paper we concentrate on paraconsistent logic PCont, [3], [2], [1] and [12]. In our presentation of an ND formulation of PCont we directly follow the notation of the latter.

The particular approach to build an ND-calculus we are interested in is described in detail in [5]. It is a modification of Quine's representation of subordinate proof [11] developed for classical propositional and first-order logic. Recall that natural deduction calculi (abbreviated in this paper by 'ND') of this type were originally developed by Jaskowski [9]. Jaskowski-style natural deduction was improved by Fitch [8] and simplified by Quine [11].

The ND technique initially defined for classical propositional logic was extended to first-order logic [5, 6] and subsequently to the non-classical framework of propositional intuitionistic logic [10]. In [4] it was further extended to capture propositional linear-time temporal logic PLTL and in [7] the ND system was proposed for the computation tree logic CTL.

to be corrected The paper is organized as follows. In §2 we describe PCont reviewing its axiomatics and semantics. In §3 we formulate the natural deduction calculus and give an example of the construction of the proof. Subsequently, in §4, we introduce the main proof-searching procedures. Finally, in §5, we provide concluding remarks and identify future work.

2 Paraconsistent Logic PCont

Fixing a set $Prop$ of propositions, we export the following axiomatics of PCont from [12].

PCont Axiomatics

1. $(A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$
2. $A \Rightarrow (A \vee B)$
3. $A \Rightarrow (B \vee A)$
4. $(A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow ((A \vee B) \Rightarrow C))$
5. $(A \wedge B) \Rightarrow A$
6. $(A \wedge B) \Rightarrow B$
7. $(C \Rightarrow A) \Rightarrow ((C \Rightarrow B) \Rightarrow (C \Rightarrow (A \wedge B)))$
8. $A \Rightarrow (B \Rightarrow A)$
9. $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$
10. $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$
11. $\neg(A \vee B) \Rightarrow (\neg A \wedge \neg B)$
12. $(\neg A \wedge \neg B) \Rightarrow \neg(A \vee B)$
13. $\neg(A \wedge B) \Rightarrow (\neg A \vee \neg B)$
14. $(\neg A \vee \neg B) \Rightarrow \neg A \wedge \neg B)$
15. $\neg(A \Rightarrow B) \Rightarrow (A \wedge \neg B)$
16. $(A \wedge \neg B) \Rightarrow \neg(A \Rightarrow B)$

17. $\neg\neg A \Rightarrow A$
18. $A \Rightarrow \neg\neg A$
19. $A \vee \neg A$.

Rule of inference: From A and $A \Rightarrow B$ infer B.

Semantics

The axioms of PCont are adequate to the following matrix semantics with three values 1, t, 0 and two designated values 1, t.

$\Rightarrow$	1	t	0
1	1	t	0
t	1	t	0
0	1	1	1

p	$\neg p$
1	o
t	t
0	1

$\vee$	1	t	0
1	1	1	1
t	1	t	t
0	1	t	0

$\wedge$	1	t	0
1	1	t	0
t	t	t	0
0	0	0	0

3 Natural deduction system N*PCont*

Notation

- By a *literal* we understand a proposition or its negation.
- We will use the symbols '$\vdash$' and '$\models$' as follows. By writing $\Gamma \vdash B$ we mean a task to establish a natural deduction derivation of a formula B from a set of assumptions Γ. If Γ, in $\Gamma \vdash B$, is empty then the task is to prove that B is a theorem, and in this case we will simply write $\vdash B$. The abbreviation $\Gamma \models B$ stands for establishing that B is a logical consequence of a set of assumptions Γ. If Γ, in $\Gamma \models B$, is empty then the task is to show that B is a valid formula and in this case we will simply write $\models B$.

Therefore, we might be given either of the following tasks: to find an ND derivation $\Gamma \vdash B$ or to find an ND proof $\vdash B$.

Specifically for an ND calculus, in constructing an ND derivation, we are allowed to introduce arbitrary formulae as new assumptions. Consequently, any formula in a derivation is either an assumption or a formula which is obtained as a result of the application of one of the inference rules.

Further, the set of rules is divided into the two classes: *elimination* and *introduction* rules. Rules of the first group allow us to simplify formulae to which they are applied.

These are rules for the 'elimination' of logical constants. Rules of the second group are aimed at 'building' formulae, introducing new logical constants. In Figure 1 we define sets of elimination and introduction rules, where prefixes '*el*' and '*in*' abbreviate an elimination and an introduction rule, respectively.

Definition 1 (Inference). *An* inference *in the system NPCont is a finite non-empty sequence of formulae with the following conditions:*

- *each formula is an assumption or is derived from the previous ones via a NPCont-rule;*

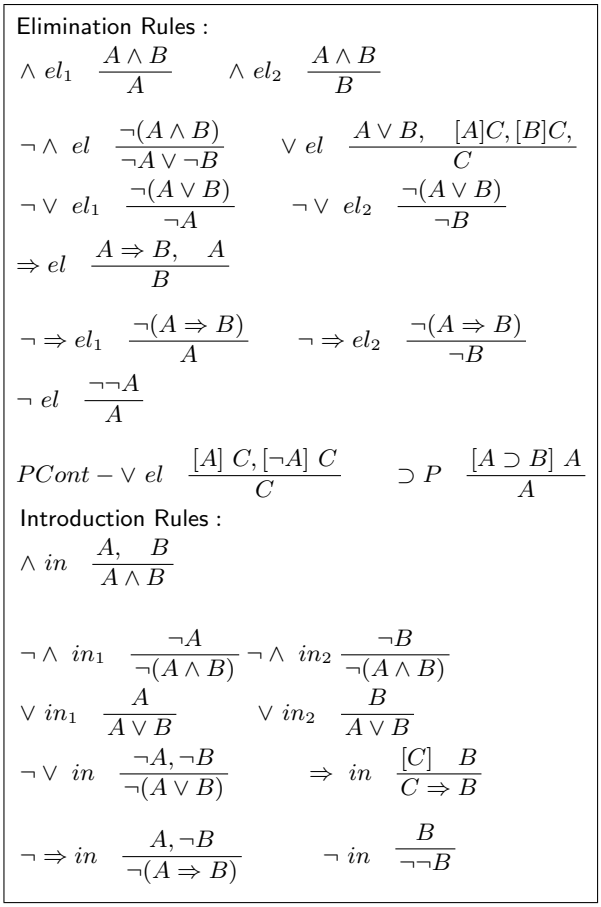

Fig. 1. NPCont-rules

- *by applying* $\Rightarrow_{in}$ *in each formula starting from the last alive assumption until the result of the application of this rule, inclusively, is discarded from the inference;*
- *by applying* $\vee_{el}$ *each formula starting from assumption* A *until formula* C*, inclusively, as well as each formula starting from assumption* B *until formula* C*, inclusively, is discarded from the inference;*
- *by applying* $PCont - \vee_{el}$ *each formula starting from assumption* A *until formula* C*, inclusively, as well as each formula starting from assumption* A *until formula* C*, inclusively, is discarded from the inference.*

Definition 2 (Proof). *A proof in the system NPCont is an inference from the empty set of assumptions.*

Two rules deserve attention. First, it is PCont$\vee_{el}$ rule which is specific for this logic and is one of variants of the disjunction elimination rule for some non-classical logics. The $\supset\ P$ rule is analoque to axiom 10 which represents Pierce law.

As an example of the ND proof let us consider the proof for axiom 7.

list_proof	annotation
1. $A \supset C$	assumption
2. $B \supset C$	assumption
3. $A \vee B$	assumption
4. A	assumption
5. C	$\Rightarrow el$, 1,4
6. B	assumption
7. C	$\Rightarrow el$, 2,6
8. C	$\vee el$, 3,4,6, [4-5],[6-7]
9. $(A \vee B) \supset C$	$\Rightarrow in$, 8, [3-7]
10. $(B \supset C) \supset ((A \vee B) \supset C)$	$\Rightarrow in$, 9, [2-9]
11. $(A \supset C) \supset ((B \supset C) \supset ((A \vee B) \supset C))$	$\Rightarrow in$, 10, [1-10]

N*PCont* has been shown to be sound and complete, i.e. the following theorem holds:

Theorem 1. $\Gamma \vdash_{\boldsymbol{N}PCont} A \Leftrightarrow \Gamma \models A$. *[13]*

4 Proof Searching Techniques in N*PCont*

We preserve the *goal-directed* nature of the proof searching strategy creating two sequences of formulae: *list_proof* and *list_goals*. The first sequence represents formulae which forms a proof. In the second sequence we keep track of the list of goals. An *algo-derivation*, $\mathrm{ND}_{\mathrm{alg}}$, as previously, is a pair (*list_proof*, *list_goals*) whose construction is determined by the searching procedure outlined below. On each step of constructing an $\mathrm{ND}_{\mathrm{alg}}$, a specific goal is chosen, which should be reached at the *current stage*, we call this goal a *current_goal*. The first goal of *list_goals* is extracted from the given task, we will refer to this goal as to the *initial goal*.

A current goal, G_n, occurring in *list_goals*$= \langle G_1, G_2, \ldots, G_n \rangle$, is reached if there is a formula in *list_proof* identical with G_n.

When we construct a derivation, we check whether the *current_goal* has been reached. If it has been reached then we apply the appropriate introduction rule, and this is *the only reason* for the application of introduction rules. Alternatively, (if the *current_goal* has not been reached), we continue searching how to update *list_proof* and *list_goals*.

Proof-Searching Procedures

Procedure 1. This procedure updates a sequence *list_proof* by searching for an applicable elimination ND-rule and updates *list_proof* by the conclusion of the corresponding rule.

The new rule $\vee_{el}$ is now associated with the new strategy. If *list_proof*contains a disjunctive formula $A \vee B$ then the rule is applicable if only the same formula C is derivable from both disjuncts. However, what is this formula C? It makes sense to restrict our search space when we try to apply this rule by considering C as a goal from *list_goals*, when procedure 2 has been applied, more precisely, when Procedure 2.5 has been applied.

The relevant strategy is now straightforward. Given that Procedure 2 has been applied, the current goal is some formula C, the following strategy is invoked:

$\Gamma, A \vee B \vdash \Delta, C \longrightarrow \Gamma, A \vdash \Delta, C, \quad \Gamma, B \vdash \Delta, C$

Procedure 2. Here a new goal is synthesized in a backward chaining style. This procedure applies when Procedure 1 terminates, i.e. when no elimination ND-rule can be applied, and the current goal, G_n, is not reached. The type of G_n determines *how* the sequences *list_proof* and *list_goals* must be updated. Procedures dealing with conjunctive, disjunctive and implication type goals are preserved:

(2.1) $\Gamma \vdash \Delta, A \wedge B \longrightarrow \Gamma \vdash \Delta, A \wedge B, B, A$
(2.2) $\Gamma \vdash \Delta, A \vee B \longrightarrow \Gamma \vdash \Delta, A \vee B, A$
(2.3) $\Gamma \vdash \Delta, A \vee B \longrightarrow \Gamma \vdash \Delta, A \vee B, B$
(2.4) $\Gamma \vdash \Delta, A \supset B \longrightarrow \Gamma, A \vdash \Delta, A \supset B, B$

The new procedure now is the one dealing with an unreachable goal, F, which is either a literal or a negative formula.

(2.5) $\Gamma \vdash \Delta, F \longrightarrow \Gamma, \neg F \vdash \Delta, F$

Let us explain the idea behind this last strategy. When we cannot current goal, F, and procedures 2.1-2.4 are not applicable, we follow similar to the classical refutation. However, now, in the setting of paraconsistent logic, we deal with this situation differently. Namely, once we assumed $\neg F$ we aim at achieving the goal F. If this can be done then we can always add to *list_proof* a proof of F from F. These two would give us the required basis to apply $P - Cont\vee_{el}$ rule, namely, $[\neg F], F$ and $[F], F$ which would enable us to derive the desired F.

Interestingly, the discovery of this strategy prompted us to consider narrowing the $P - Cont\vee_{el}$ rule to just

$$\frac{[C]\ C, [\neg C]\ C}{C}$$

However, this would have restricted the manual theorem proving in the system thus making some proofs longer.

Procedure 3. This procedure checks the reachability of the current goal in the sequence *list_goals*. If the current goal G_n is reached then the sequence *list_goals* is updated by deleting G_n and setting G_{n-1} as the current goal.

Procedure 4. Procedure 4 indicates that one of the introduction ND-rules, i.e. a rule which introduces a logical connective must be applied.

The proof searching algorithm is essentially preserved from the classical setting. Surely, the exception is now the technique to cope with disjunction, i.e with $\vee_{el}$ and PCont specific rules.

Examples

Now we will give two examples of ND proofs. The first example is a successful proof of axiom 7 following the searching technique. The second example shows how the procedure terminates without finding a proof.

We will see where an automated proof differs from the manual one given in previous section. We set the initial goal as axiom 7. As its main symbol is $\Rightarrow$ and the goal is not reached we apply Procedure 2.4 which gives us a new assumption $A \supset C$ (step 1) and a new goal $(B \supset C) \supset ((A \vee B) \supset C)$. Again, the current goal is not reached and analysing its structure we apply Procedure 2.4 to give us a new assumption $B \supset C$ (step 2) and a new goal $(A \vee B) \supset C$. Similar reasoning gives us step 3. Now, the current goal is C, it is not reachable, hence we apply Procedure 2.5 to give us new assumption $\neg C$ at step 4. This latter technique is the one which distinguishes automated proof from the manual one given in previous section. In the analysis, in *list_goals* we explicitly show that C must be obtained from the assumption $[\neg C]$ which should later be discarded.

list_proof	annotation	*list_goals*
		$G_0 = (A \supset C) \supset ((B \supset C) \supset ((A \vee B) \supset C))$
1. $A \supset C$	assumption	$G_0, G_1 = (B \supset C) \supset ((A \vee B) \supset C)$
2. $B \supset C$	assumption	$G_0, G_1, G_2 = (A \vee B) \supset C$
3. $A \vee B$	assumption	G_0, G_1, G_2, C
4. $\neg C$	assumption	$[\neg C]C$

At this stage we fire Procedure 1 once again, after Procedure 2.5 has been applied, and our target now is the disjunctive formula $A \vee B$ at step 3. Thus, we have to create two new subproofs, commencing with the disjuncts of this disjunctive formula, targeting elimination of disjunction. Now we know that the formula we want to derive from both disjuncts is the current goal, C. Hence steps 5 and 7 are new assumptions, disjuncts of $A \vee B$ and the notation in *list_goals* is chosen to explicitly show that the goal C should be obtained from the assumptions [A] and [B] such that both assumptions should be discharged later. In both cases we derive C, at steps 6 and 8 by modus ponens. Now we have grounds to eliminate disjunction from step 3 applying $\vee_{el}$ and deriving step 9. Note that we must discharge assumptions 5 and 7 and discard all formulae that are dependant on these assumptions, which is shown by the notation [5-6], [7-8].

list_proof	annotation	*list_goals*
5. A	assumption	$[A]C$
6. C	$\Rightarrow el$, 1,5	
7. B	assumption	$[B]C$
8. C	$\Rightarrow el$, 2,7	
9. C	$\vee el$, 3,5,7, [5-6],[7-8]	$[A]C, [B], C$

The notation at step 9 shows that C has been reached and the corresponding assumptions have been discharged. As we have completed the algo-proof task to derive C from $\neg C$ we now need to derive C from C which is trivial, step 10. At this stage we have all grounds to apply PCont$\vee_{el}$ rule to derive C at step 11 This gives us the desired goal C and by applying consecutively $\Rightarrow_{in}$ rule we complete the proof.

list_proof	annotation	*list_goals*
10. C	assumption	
11. C	PCont$\vee_{el}$:10,9, [10],[4-9]	
12. $(A \vee B) \supset C$	$\Rightarrow in$, 11, [3-10]	G_2 *reached*
13. $(B \supset C) \supset ((A \vee B) \supset C)$	$\Rightarrow in$, 12, [2-11]	G_1 *reached*
14. $(A \supset C) \supset$ $((B \supset C)) \supset ((A \vee B) \supset C)$	$\Rightarrow in$, 13, [1-13]	G_0 *reached*

Finally we will give an example of a formula for which proof is not found, $(A \Rightarrow \neg B) \Rightarrow (B \Rightarrow \neg A)$.

list_proof		*list_goals*
1. $A \Rightarrow \neg B$	assumption	$B \Rightarrow \neg A$
2. B	assumption	$\neg A$
3. A	assumption	$[A]\ \neg A, [\neg A]\ \neg A$
4. $\neg B$	$\Rightarrow el$, 1, 3	

Goal $[A]\neg A$ cannot be reached.

5 Conclusion and future work

6 Discussion

We have presented a proof search technique in natural deduction system for paraconsistent logic PCont. To the best of our knowledge, there is no other similar works.

Our proof-searching technique preserves many of the strategies developed earlier but also introduces new techniques that are specific for PCont. Showing the correctness of the presented proof search technique is our next task. The development of a proof-search procedure will enable the implementation of the method.

References

1. F. Asenjo and J. Tamburino. Logic of antinomies. *Notre Dame Journal of Formal Logic.*, 16:17–44, 1975.
2. A. Avron. On an implicational connective of rm. *Notre Dame Journal of Formal Logic.*, 27:201–209, 1986.
3. D. Batens. Paraconsistent extensional propositional logics. *Logique et Analyse.*, 23:127–139, 1980.
4. A. Bolotov, A. Basukoski, O. Grigoriev, and V. Shangin. Natural deduction calculus for linear-time temporal logic. In *Joint European Conference on Artificial Intelligence (JELIA-2006)*, pages 56–68, 2006.
5. A. Bolotov, V. Bocharov, A. Gorchakov, V. Makarov, and V. Shangin. *Let Computer Prove It.* Logic and Computer. Nauka, Moscow, 2004. (In Russian), Implementation of the proof search technique for classical propositional logic available on-line at `http://prover.philos.msu.ru`.
6. A. Bolotov, V. Bocharov, A. Gorchakov, and V. Shangin. Automated first order natural deduction. In *Proceedings of IICAI*, pages 1292–1311, 2005.
7. A. Bolotov, O. Grigoriev, and V. Shangin. Natural deduction calculus for computation tree logic. In *IEEE John Vincent Atanasoff Symposium on Modern Computing*, pages 175–183, 2006.
8. F. Fitch. *Symbolic Logic.* NY: Roland Press, 1952.
9. S. Jaskowski. On the rules of suppositions in formal logic. In *Polish Logic 1920-1939*, pages 232–258. Oxford Univ. Press, 1967.
10. V. Makarov. Automatic theorem-proving in intuitionistic propositional logic. In *Modern Logic: Theory, History and Applications. Proceedings of the 5th Russian Conference*, StPetersburg, 1998. (In Russian).
11. W. Quine. On natural deduction. *Journal of Symbolic Logic*, 15:93–102, 1950.
12. L. Rozonoer. On finding contradictions in formal theories. *Automatica and telemekhanica*, (6), 1983. (in Russian).
13. V. Shangin. Natural deduction systems for logics Par, PCont and PComp. Rostov-on-Don, 2010. in Russian.