Annales Univ. Sci. Budapest., Sect. Comp. 37 (2012) 229–238

PRESS-READY DEDUCTION TREES IN CLASSICAL LOGIC USING POINT-PLUS-EXPRESSIONS

Tamás Kádek (Debrecen, Hungary)

Communicated by Zoltán Horváth

(Received January 15, 2012; revised February 6, 2012;
accepted February 11, 2012)

Abstract. The LaTeX system is widely used in scientific journals with syntactically rich mathematical expressions. It is suitable not only for high-quality press-ready production, but it can also be applied for the development of dynamic presentations and learning materials containing many nice formulas. The need to construct numerous derivations using Gentzen style sequent calculus, gives the idea to automatize the process (which was implemented in the GenTreeCad application). The point-plus-expressions show how to use an existing derivation to produce different solutions by different theorem proving methods.
In this artice, we try to demonstrate the capability of point-plus-expressions meanwhile we are using them for automatic theorem proving. We want to show – using propositional logic – how we could take the advantages of the connection between the resolution and it is dual one, then the connection between sequent calculus and the tableau method as well.

1. Motivation

The basic idea of the born of the GenTreeCad application [8] was the need of the press-ready examples in the education of the classical logic. The stu-

Key words and phrases: Mathematical logic, mechanical theorem proving.
2010 Mathematics Subject Classification: 03F07.
1998 CR Categories and Descriptors: F.4.1.
The work/publication is supported by the TÁMOP-4.2.2/B-10/1-2010-0024 project. The project is co-financed by the European Union and the European Social Fund.

dents at the university require numerous examples to clearly understand the adequate strategy during the creation of a proof using a Gentzen style sequent calculus. The simple, plain text coding of LaTeX formulas allows the use of the LaTeX source text/code as input for further processing. The above mentioned benefits give us the idea to develop an application which can provide the ability to build up several derivations using different proving systems based on classical logic, such as Gentzen style sequent calculus, semantic tableaux method, and the resolution calculus for educational purposes. The present solution is still restricted to the propositional logic.

The additional goal was to develop a process which is able to produce a general derivation. This general derivation generates the correspondent derivation instances over the mentioned proving systems (instead of implementing and using them separately). Based on an original idea introduced by Dragálin [1] and investigated later deeply by Pásztor Varga and Várterész [5, 6], it makes a unified treatment of the relevant calculi using the point-plus meta-language possible for us.

2. Point-plus-expressions

Because we use classical propositional logic, we suppose that we have an infinite list of propositional letters X, Y, Z and so on (usually denoted by upper case letters). We will use $\neg$ for negation (one-place logical connective), $\wedge$ for conjunction, $\vee$ for disjunction and $\supset$ for implication (binary logical connectives). We are able to build up propositional formulas from this set of symbols with the usual syntax. To avoid misunderstanding, we will use fully parenthesized expressions (omitting only the outmost parentheses).

Definition 2.1. If A is a formula, then tA and fA are labelled formulas as well.

Definition 2.2. The set of point-plus-expressions[1] $\mathcal{P}$ (or pp-expressions for short) is the smallest set that contains:

- labelled formulas,
- $(P_1 \circ P_2 \circ \cdots \circ P_n)$ $(n \geq 2)$, if $P_i \in \mathcal{P}$ for all $i = 1, \cdots, n$,
- $(P_1 + P_2 + \cdots + P_n)$ $(n \geq 2)$, if $P_i \in \mathcal{P}$ for all $i = 1, \cdots, n$.

[1]The pp-expression was originally developed by Dragálin[1]

Definition 2.3. Elementary expressions from the set of point-plus-expressions:

- the elementary point-chain is a list of one or more labelled propositional letters, separated by $\circ$,
- the elementary plus-chain is a list of one or more labelled propositional letters, separated by $+$.

We use the following normal forms:

Definition 2.4. Normal forms in $\mathcal{P}$:

- The point-chain is a list of one or more labelled formulas, separated by $\circ$,
- the plus-chain is a list of one or more labelled formulas, separated by $+$,
- the point normal form is a plus-chain, or a list of $\circ$ separated plus-chains,
- the plus normal form is a point-chain, or a list of $+$ separated point-chains.

The order of the elements in the lists above is irrelevant. We do not distinguish between $tX \circ tY$ and $tY \circ tX$.

Note that a point-chain and a plus-chain is also a pp-expression.

3. Automatic proof generation

3.1. Pp-resolution

Now we want to show that a resolution like calculus can be defined over the language of $\mathcal{P}$.

Definition 3.1. Rewriting algorithm:

Let C be a formula. We define the pp-expression form of C, using the following rules on fC (which is a labelled formula). We continue to apply the rules until there exists a labelled formula in the pp-expression, where $\wedge$ or $\vee$ or $\neg$ is the main logical connective. Let A and B denote formulas.

1. $fA \wedge B \Rightarrow fA + fB$,
2. $fA \vee B \Rightarrow fA \circ fB$,
3. $f\neg A \Rightarrow tA$.

If C was a conjunctive normal form, then the output of the algorithm is a pp-expression without any logical connectives. Informally, we just substitute the conjunctions, disjunctions and negations with $+$ and $\circ$, and t symbols. In this case, the above algorithm generates a pp-expression in plus normal form, and furthermore, the plus normal form consists of only elementary point-chains.

Now let us see an example of how to use the *rewriting algorithm*.

$$\begin{aligned} fZ \wedge (X \vee \neg Y) \wedge (Y \vee \neg Z) \wedge (\neg X \vee \neg Z) &\Rightarrow \\ fZ + f(X \vee \neg Y) + f(Y \vee \neg Z) + f(\neg X \vee \neg Z) &\Rightarrow \\ fZ + (fX \circ f\neg Y) + (fY \circ f\neg Z) + (f\neg X \circ f\neg Z) &\Rightarrow \\ fZ + (fX \circ tY) + (fY \circ tZ) + (tX \circ tZ) & \end{aligned}$$

If we adopt the resolution idea, a resolution like calculus can be defined in the subset of $\mathcal{P}$, which contains elementary point-chain based plus normal forms (*elementary plus normal forms*). The above algorithm suggests thinking of about an elementary point-chain as a clause.

Definition 3.2. Axiom schemes of the pp-resolution:

$$fX + tX \quad \text{and} \quad \Gamma + fX + tX$$

where Γ is an elementary plus normal form, and X is a propositional letter.

Definition 3.3. Pp-resolution expansion rules:

$$\frac{\Delta_1 + \Delta_2 + \Delta}{\Delta_1 + \Delta_2} \quad \text{and} \quad \frac{\Gamma + \Delta_1 + \Delta_2 + \Delta}{\Gamma + \Delta_1 + \Delta_2}$$

where Γ is an elementary plus normal form. Δ_1 and Δ_2 are elementary point chains, where both of them contain exactly one propositional letter with different labels in Δ_1 and Δ_2. Let X be that propositional letter. In this case, Δ is an elementary point-chain with the labelled elements of Δ_1 and Δ_2, excluding tX and fX.

We are able to construct a proof for the example above, because it was an elementary plus normal form.

$$\begin{aligned} fZ + \underline{(fX \circ tY)} + \underline{(fY \circ tZ)} + (tX \circ tZ) &\Rightarrow \\ \underline{fZ} + (fX \circ tY) + (fY \circ tZ) + (tX \circ tZ) + \underline{(fX \circ tZ)} &\Rightarrow \\ \underline{fZ} + (fX \circ tY) + (fY \circ tZ) + \underline{(tX \circ tZ)} + (fX \circ tZ) + fX &\Rightarrow \\ fZ + (fX \circ tY) + (fY \circ tZ) + (tX \circ tZ) + (fX \circ tZ) + fX + tX & \end{aligned}$$

Here, the last pp-expression is an axiom.

Now we need to show that there is a conversion between a proof by the pp-resolution and a proof by the standard resolution.

3.2. Interpretations

Now we can look at the pp-expressions as formal languages, where the $+$, $\circ$ and the labels are non-terminal symbols. The only thing we need is a formal grammar. First, we try to create the opposite of the rewriting algorithm:

Definition 3.4. The formal grammar of the tn-interpretation[2]:

1. $(fA_1 \circ fA_2 \circ \cdots \circ fA_k) \Rightarrow f(A_1 \vee A_2 \vee \cdots \vee A_k)$, where $k \geq 2$,
2. $(fA_1 + fA_2 + \cdots + fA_k) \Rightarrow f(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$, where $k \geq 2$,
3. $tA \Rightarrow f\neg A$, where A is a formula.

If we apply the steps above until it is possible, then an fA formula appears, where A is the formula generated by the tn-interpretation.

If we apply the tn-interpretation on the pp-resolution expansion rule, we recognize the resolution rule. Let Γ' be the tn-interpretation of Γ, and let Δ'_i be the tn-interpretation of Δ_i, where $i \in \{1, 2\}$.

$$\frac{\Gamma' \wedge (X \vee \Delta'_1) \wedge (\neg X \vee \Delta'_2) \wedge (\Delta'_1 \vee \Delta'_2)}{\Gamma' \wedge (X \vee \Delta'_1) \wedge (\neg X \vee \Delta'_2)}$$

When we just enumerate the elementary point-chain components of the plus normal form, in the same order as they appear in the pp-expression, we get back the steps for the resolution. In the first case, the elementary point-chains represent elementary disjunctions (in other words, propositional clauses).

1.	fZ	1.	Z	
2.	$fX \circ tY$	2.	$X \vee \neg Y$	
3.	$fY \circ tZ$	3.	$Y \vee \neg Z$	
4.	$tX \circ tZ$	4.	$\neg X \vee \neg Z$	
5.	$fX \circ tZ$	5.	$X \vee \neg Z$	from 2 and 3
6.	fX	6.	X	from 1 and 5
7.	tX	7.	$\neg X$	from 1 and 4
		8.	empty clause	from 6 and 7

However, the tn-interpretation is not the only way which translates the pp-expressions into the language of propositional logic. This is the point where the benefits of the pp-expressions appear.

[2]Informally, in the tn-interpretation, the t label represents negation.

Definition 3.5. The formal grammar of the fn-interpretation:

1. $fA \Rightarrow t\neg A$ where A is a formula,
2. $(tA_1 \circ tA_2 \circ \cdots \circ tA_k) \Rightarrow t(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$, where $k \geq 2$,
3. $(tA_1 + tA_2 + \cdots + tA_k) \Rightarrow t(A_1 \vee A_2 \vee \cdots \vee A_k)$, where $k \geq 2$.

If we apply the steps above until it is possible, then a tA formula appears, where A is the formula generated by the fn-interpretation.

The table below shows the result of the fn-interpretation. In this second interpretation, we get the steps of the dual resolution. In this way, we can derive two different proofs from just one pp-resolution.

1.	fZ	1.	$\neg Z$	
2.	$fX \circ tY$	2.	$\neg X \wedge Y$	
3.	$fY \circ tZ$	3.	$\neg Y \wedge Z$	
4.	$tX \circ tZ$	4.	$X \wedge Z$	
5.	$fX \circ tZ$	5.	$\neg X \wedge Z$	from 2 and 3
6.	fX	6.	$\neg X$	from 1 and 5
7.	tX	7.	X	from 1 and 4
		8.	empty dual clause	from 6 and 7

A dual clause is an elementary conjunction, and the dual resolution rule is the following (Γ' is the fn-interpretation of Γ, ...):

$$\frac{\neg(\Gamma' \vee (\neg X \wedge \Delta_1') \vee (X \wedge \Delta_2') \vee (\Delta_1' \wedge \Delta_2'))}{\neg(\Gamma' \vee (\neg X \wedge \Delta_1') \vee (\neg X \wedge \Delta_2'))}$$

3.3. Pp-sequent

Based on the idea of pp-resolution, the language $\mathcal{P}$ is suitable to describe the Gentzen style sequent calculus. Now we want to build up the axioms and rules of the Gentzen style sequent calculus using pp-expressions.

The following sequent and formula are equal ($n \geq 0$ and $m \geq 0$ and $n + m \geq 1$).

$$A_1, A_2, \cdots, A_n \to B_1, B_2, \cdots, B_m \quad \sim$$
$$\neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_n \vee B_1 \vee B_2 \vee \cdots \vee B_m$$

If we remove the first rule[3] from the *rewriting algorithm*, then the pp-expression form of the formula above will be a point-chain. (For simplicity, we assume that B_i $(i = 1, \cdots, m)$ is not a disjunction.)

$$f\neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_n \vee B_1 \vee B_2 \vee \cdots \vee B_m \quad \Rightarrow$$
$$tA_1 \circ tA_2 \circ \cdots \circ tA_n \circ fB_1 \circ fB_2 \circ \cdots \circ fB_m$$

Of course, the tn-interpretation of the pp-expression above will be the formula form of the sequent.

While we construct a proof, maybe we have to work out more than one sequent. In the rules below, we collect the sequents into a list where we separate them by $+$. Note that the lists are still pp-expressions.

Definition 3.6. Axiom scheme of the pp-sequent is a plus-chain which contains only point-chains in the form:

$$fA \circ tA \quad \text{or} \quad \gamma \circ fA \circ tA$$

where γ is a point-chain, and A is a formula.

Let us note that the tn-implementation of any axiom pp-sequent is a tautology, because they are conjunction chains, where all elements of the chain are tautologies.

Definition 3.7. Rules of the pp-sequent calculus:

$$\frac{(fA \circ \gamma) + (tB \circ \gamma) + \gamma_1 + \cdots + \gamma_k}{(tA \supset B \circ \gamma) + \gamma_1 + \cdots + \gamma_k} \qquad \frac{(tA \circ fB \circ \gamma) + \gamma_1 + \cdots + \gamma_k}{(fA \supset B \circ \gamma) + \gamma_1 + \cdots + \gamma_k}$$

$$\frac{(fA \circ \gamma) + \gamma_1 + \cdots + \gamma_k}{(t\neg A \circ \gamma) + \gamma_1 + \cdots + \gamma_k} \qquad \frac{(tA \circ \gamma) + \gamma_1 + \cdots + \gamma_k}{(f\neg A \circ \gamma) + \gamma_1 + \cdots + \gamma_k}$$

$$\frac{(tA \circ tB \circ \gamma) + \gamma_1 + \cdots + \gamma_k}{(tA \wedge B \circ \gamma) + \gamma_1 + \cdots + \gamma_k} \qquad \frac{(fA \circ \gamma) + (fB \circ \gamma) + \gamma_1 + \cdots + \gamma_k}{(fA \wedge B \circ \gamma) + \gamma_1 + \cdots + \gamma_k}$$

$$\frac{(tA \circ \gamma) + (tB \circ \gamma) + \gamma_1 + \cdots + \gamma_k}{(tA \vee B \circ \gamma) + \gamma_1 + \cdots + \gamma_k} \qquad \frac{(fA \circ fB \circ \gamma) + \gamma_1 + \cdots + \gamma_k}{(fA \vee B \circ \gamma) + \gamma_1 + \cdots + \gamma_k}$$

where γ_i $(i = 1, \cdots, k)$ is a point-chain, and $k \geq 0$, and γ is a point-chain or missing.

The tn-interpretation of the rules generates the rules of the Gentzen style sequent calculus, so the pp-sequent calculus works correctly. As an example, we create the proof of the tautology $(\neg X \supset \neg Y) \supset (Y \supset X)$.

[3]We keep the labelled conjunctions.

1. $f(\neg X \supset \neg Y) \supset (Y \supset X)$
2. $t\neg X \supset \neg Y \circ fY \supset X$
3. $f\neg X \circ fY \supset X + t\neg Y \circ fY \supset X$
4. $tX \circ fY \supset X + t\neg Y \circ fY \supset X$
5. $tX \circ fY \supset X + fY \circ fY \supset X$
6. $tX \circ tY \circ fX + fY \circ fY \supset X$
7. $tX \circ tY \circ fX + fY \circ tY \circ fX$

Instead of the + symbols we can choose the binary tree representation. Simultaneously, we can show the sequents generated from the pp-expressions.

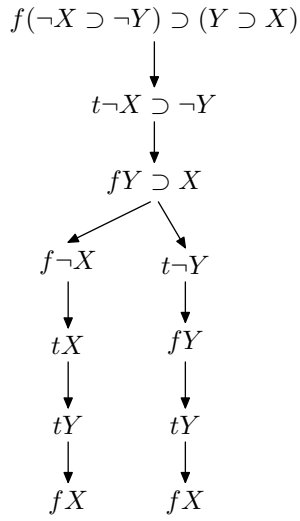
$$\dfrac{\dfrac{\dfrac{tX \circ tY \circ fX}{tX \circ fY \supset X}}{f\neg X \circ tY \supset X} \qquad \dfrac{\dfrac{fY \circ tY \circ fX}{fY \circ fY \supset X}}{t\neg Y \circ fY \supset X}}{\dfrac{t\neg X \supset \neg Y \circ fY \supset X}{f(\neg X \supset \neg Y) \supset (Y \supset X)}}$$

$$\dfrac{\dfrac{\dfrac{X, Y \to X}{X \to Y \supset X}}{\to \neg X, Y \supset X} \qquad \dfrac{\dfrac{Y \to Y, X}{\to Y, Y \supset X}}{\neg Y \to Y \supset X}}{\dfrac{\neg X \supset \neg Y \to Y \supset X}{\to (\neg X \supset \neg Y) \supset (Y \supset X)}}$$

So the reduced rewriting rule and the tn-interpretaion give us the possibility to use sequent calculus on a pp-expression. The pp-resolution shows that the fn-interpretation generates the dual calculus. Now we could recognise the labelled tableaux method.

Each step of the pp-sequent calculus collects the unsolved labelled formulas of the tableaux. The branches are separated by + and the unsolved formulas in a branch are separated by $\circ$.

In the example above, we start the tableaux with the f labelled original formula. That is the root of the tree. In the last (7th) step, we have two branches, where tX, tY, fX are unsolved, and in the other side, fY and tY and fX are unsolved.



The background of the parallelism based on the rules of the pp-sequent calculus, because they generate the rules of the labelled tableaux method. (+ divide the branch into two parts, ∘ appends the current branch, and the labels are identical.) For example:

$$\frac{fA + tB}{tA \supset B} \quad \Rightarrow \quad \frac{fA;\ tB}{tA \supset B} \qquad \frac{tA \circ fB}{fA \supset B} \quad \Rightarrow \quad \frac{\begin{array}{c} tA \\ fB \end{array}}{fA \supset B}$$

3.4. Point-plus-expressions in use

The GenTreeCad application supports currently LaTeX input and MetaPost or PDF output to create press-ready images. The deduction trees in this article were also generated by the GenTreeCad application. During the implementation process of the Gentzen calculus, the data structure shows some similarity with the pp-expressions. The implementation is matching with the rules of the pp-sequent calculus, in which way the application is already contorlled by point-plus-expressions. We are convinced to be able to enrich it with the labelled tableaux method in the near future.

4. Summary

Currently the GanTreeCad application generates only Gentzen style deduction trees, but extension is planned to support the tableaux method. With pp-expressions, the pp-sequent calculus is able to produce both of the deduction trees, with just one single run. Furthermore, the pp-expressions explore a

way for a unified discussion of different calculi in logic. Of course, we have to anticipate the need to extend the discussion to the predicate logic, simultaneously with the original work of Dragálin [1].

References

[1] **Dragálin, A.G.**, *On a Self-dual Notation in Automated Reasoning*, Technical Report No 96/16, Debrecen, 1996.

[2] **Toelstra, A.S. and H. Schwichtenberg,** *Basic Proof Theory*, Cambridge University Press, Cambridge, 1996.

[3] **Fitting, M.,** *First-order Logic and Automated Theorem Proving*, Springer, 1996.

[4] **Stachniak, Z.,** *Resolution Proof Systems: An Algebraic Theory*, Kluwer Academic Publishers, York University, North York, Canada, 1996.

[5] **Pásztor Varga, K. and M. Várterész,** A generalized approach to the theorem proving methods, in: *Proc. of 5th International Conference on Applied Informatics*, pp. 191–200, Eger, 2001.

[6] **Pásztor Varga, K. and M. Várterész,** Comparison and usability of two rewriting systems for theorem proving, *Pure Mathematics and Applications*, **13(1-2)**, pp. 293–302, Budapest-Siena, 2002.

[7] **Wettl, F., Gy. Mayer and P. Szabó,** *LaTeX kézikönyv*, Panem, Budapest, 2004.

[8] **Kádek, T.,** Gentzen-levezetések generálása oktatási célú számítógépes támogatással, in: *SzámOkt 2010 20th International Conference on Computers and Education*, pp. 136–139, Satu Mare, 2010.

T. Kádek
University of Debrecen
Faculty of Informatics
H-4028 Debrecen, Kassai st. 26.
Hungary
kadek.tamas@inf.unideb.hu