Proofs as Texts

Dynamic Proof Theory for Intuitionistic Propositional Logic

C.F.M. Vermeulen

October 20, 1994

Abstract

In this paper we develop a view on proofs as texts. Motivation for such an enterprise stems from current developments in formal semantics, where a shift of interest from the meaning of sentences to the meaning of texts is taking place. Such a shift will influence methodological issues in formal semantics, but, as will be shown, it also can lead to a new perspective on proofs. In the paper we consider some of the consequences of this new perspective.

1 Introduction

This paper presents a new perspective on proof theory, the *proofs as texts* perspective. The proofs as texts perspective is interesting for two reasons. First because of its motivation from dynamic semantics that we will discuss shortly. But also because it gives rise to the introduction of new techniques in proof theory, techniques that until now were only used in the theory of grammar. In the proofs as texts approach to proof theory we will see an intriguing interplay between syntactic structure and proof theoretical properties. The techniques from grammar that we use will be introduced below and we will show how they work in a fragment of intuitionistic propositional calculus. It is not our aim here to obtain new results about this fragment: we will just show how various familiar results — such as the subformula property and a decidability for the fragment — can be reproduced in our set up. Thereby we will see how the new techniques lead to a new view on familiar issues.
Besides giving us yet another way of obtaining familiar results, such a new view could also prove useful in the search of new results. It is well known that changing the design of a proofs system can be of great use in this respect. For example, when developing algorithms for automatic proof search, a sequent presentation of the logic is usually the most suitable, considering the conclusion of a sequent style proof as the root of the search space for such a proof. But in other branches of proof theory the use of the terms of a typed lambda calculus to represent proofs — following the Curry-Howard isomorphism — has been very successful for establishing normalisation results. Thus choosing a certain presentation of the logic can be of great help when one is looking for a particular kind of result.
As was said above, we do not yet have any (new) results that follow especially naturally when one chooses a proofs as texts presentation of the logic. But, since in the proofs as texts perspective the structural properties of a formula play a leading role, it is not unreasonable to expect that results for which the structure of a formula is particularly crucial will be easier to prove in this new set up. Here we think especially of results that depend on the

nesting of implications in a formula.

The origins of the proofs as texts perspective lie in our attempts to come to grips with the proof theory of the formalisms that are currently used in dynamic semantics. Dynamic semantics originated as an attempt to find an elegant and natural representation of phenomena of text coherence, such as anaphora. Whereas traditionally in formal semantics one tries to represent the meaning of isolated sentences, the dynamic semanticist is also concerned with the representation of the interaction between the meanings of different sentences. The attention to such interaction has serious consequences for the way in which the semantics should be set up.
As an example, consider this small text:

A dog barked. It was lonely.

Here the pronoun *It* in the second sentence has to be linked to the expression *A dog* before it can get the right denotation: the dog that barked is also the dog that was lonely. If we want to be able to make this link in the semantics, we cannot simply represent the first sentence as the closed first order formula:

$$\exists x\ (\textsc{dog}(x) \wedge \textsc{barked}(x))$$

Although this formula has the same truth conditions as the first sentence of our text, this representation would make it very hard to add the extra information that the dog, x, is also lonely. This effect can clearly not be obtained simply by adding the formula $\textsc{lonely}(x)$ as a conjunct. For this would give us

$$\exists x\ (\textsc{dog}(x) \wedge \textsc{barked}(x)) \wedge \textsc{lonely}(x)$$

but now the last occurrence of x is not bound by the quantifier, so we do not get the right truth conditions for the text as a whole. Therefore formalisms have been proposed in which the representation of the first sentence and the representation of the second sentence can be linked in a natural way with the required effect.
These investigations lead to a different look on the formal objects that are used in the semantics to represent the meanings of natural language expressions. The formulas used to be treated as if they were sentences, but in dynamic semantics it is more appropriate to treat formulas as if they were (small) texts. This view of *formulas as texts* has had several interesting consequences for the architecture of formal semantics.[1] We will not go into the details of these developments, but we will try to give a feel for the issues that arise if we try to give a proof theory for such formalisms.

First we note that the kind of anaphoric links that we saw in our example can also occur between the assumptions and the conclusions of some argument. For example in:

Assume a man owns a house. Then he also owns a garden.

the man in the conclusion is definitely supposed to be the same man as the man in the assumption. It seems natural to require that a proofs system for dynamic semantics should take such possibilities into account. This means for example that one of the theorems of dynamic logic should be:

[1] Here the main contributions were made by [?], [?] and [?], but also see [?], [?], [?].

If a man owns a house, then he owns a house.

Therefore, when we try to decide in our formalism whether $\phi \vdash \psi$, we should allow for anaphoric links between ϕ and ψ to occur *as if* ϕ *occurred before* ψ in some text.
These anaphoric phenomena give rise to unexpected complications in the logic. For example, a logic that can handle anaphoric links should be able to explain what happens in the following example:[2]

If a man owns a house, he owns a garden.

If he owns a garden, he waters it.

Therefore: if a man owns a house, he waters it.

It is quite clear, intuitively, what goes wrong here: the anaphoric link between *a house* and *it* that we find in the conclusion is not justified by the assumptions. The other link, between *a man* and *he*, is perfectly all right. It turns out to be rather difficult to find a formulation of the logic that explains these things in a natural and elegant way.[3]

These complications have led us to believe that a different design of the deduction system could be of great use in improving our understanding of the logic of these formalisms. For example, as was already pointed out, in $\phi \vdash \psi$ one should treat ϕ as if it occurred *before* ψ in a text. Therefore a presentation of the logic in which this order is reflected is to be preferred in a deduction system for dynamic logic.
So we see that anaphoric links do not only cause changes in the design of the *semantic* machinery, but also in the set up of the *deduction systems* for dynamic logic. At the very least these deduction systems should respect the order of the formulas in some way or other.
In this paper we go one step further: we will not only consider the expression $\phi \vdash \psi$ *as if* ϕ occurred before ψ in some text, we will in fact treat the whole expression as (the representation of) a text. For example, we read the expression $\phi \vdash \psi$ as:

Let's assume that ϕ holds. Then we may conclude that ψ.

This is the main point of the *proofs as texts* perspective. If we do this, the anaphoric links between ϕ and ψ will be represented correctly and later also other phenomena of text coherence that we may want to represent will automatically fall into place.
Note that it is not at all unreasonable to think of a proof as a text. For usually proofs are presented to us as texts. They are a special kind of text, of course: they are the kind of text that has been written in such a way that every step that is made in the text is valid. It is only natural to try and use these proof-texts in a deduction system for a formalism in which the formulas themselves are also (representations of) texts.
So these are the two main reasons for choosing a proofs as texts presentation of a dynamic logic: many proofs in real life are indeed presented to us as a special kind of text and by treating proofs as texts we will automatically be in a situation where we can discuss all phenomena of text coherence that we may want to treat in our logic.
In this paper we concentrate on developing this idea as such and we will apply it to a very simple example. We will not try to give deduction rules that can cope with anaphoric links:

[2] Due to Johan van Benthem.
[3] Cf. [?], [?], [?], [?].

our language will be too poor to represent such links. Here we will present a prototype of the kind of deduction system that we will try to use later to get at the logic of anaphora. So we intend to develop a satisfactory account of the logic of texts in several steps: we first give a rudimentary proofs system and then we intend to refine this later to include phenomena in the machinery that are especially important in texts. Anaphora is the most salient example of such a phenomenon.
An aspect of texts that we *will* be concerned with from the start is the structural coherence of a text. We will see that the structural organisation of a text is the main tool in the formulation of a logic in proofs as texts style.

The choice for the proofs as texts perspective has far reaching consequences. We are now in a situation where we have a formalism in which we represent texts, while at the same time the proofs for this formalism are also texts. So the proofs will form a special subset of the set of all formulas and the usual convenient distinction between the level of the formulas and the meta-level of the proofs over these formulas is no longer available. We represent the proofs at the same level as the formulas.
This situation could lead to a proliferation of connectives in the representation language: besides the usual connectives that occur in a sentence we now also have to represent connectives between sentences, i.e. ways in which sentences can be combined into texts. But fortunately the two ways of combining sentences into texts that we are interested in here have a natural counterpart on sentence level with more or less the same meaning. Therefore we can use just one connective on both levels. An example of this situation is conjunction: this is represented within one sentence by expressions such as *and* and *but*. But also different sentences can be presented in such a way that it is clear that they should be interpreted in a conjunctive manner. In fact this seems to be the default option if two sentences occur after each other. So one symbol for conjunction is enough.
Another example of such a situation is provided by implication and valid inference. Within one sentence implications occur in the guise of if-then constructions, but there are several constructions that do the same thing between different sentences. Think of constructions such as

> Assume that ... . Then it is clear that ... .

Also here we can use just one symbol for both situations: both if-then sentences and the entailment relation between different sentences will be represented with the same symbol. The technical justification for this move is the fact that we have the deduction theorem for our logic. If this were not the case, we might be forced to work in a language with two implications, one for ordinary implication and one for the entailment relation. Conjunction and implication are the only two connectives that we will use in this paper.
So we see that we get a representation on the level of formulas of all sorts of operations that we used to think of as operations on the level of proofs. For example, we now have a connective to represent the notion of valid inference in our formulas, namely the connective that we also use for implication. But we also find that notions that we used to think of a sentence level notions now apply to proofs. For example, now that a proof is 'just another formula' it makes sense to talk about the meaning of a proof in the same sense as about the meaning of any other text. This is an interesting observation that we will not follow up here, but the reader may wish to consult [?], [?] for more details on the issue of the meaning of proof texts.

There is one more aspect of texts that we will try to reflect in our deduction systems: the incremental nature of texts. Usually texts are written incrementally or, to be more precise, usually we expect texts to be written in such a way that they can be read incrementally. Elsewhere ([?], [?]) we have argued that this means that texts should allow for an incremental interpretation: the intuition that texts have this incremental quality seems too strong to ignore. Therefore we will also try to incorporate it in the design of our proofs system: the texts that are the proofs of our deduction system will be built up incrementally. Thus we get a picture in which proofs are texts that are built up step by step, taking care that validity is preserved along the way: theorem proving is modeled as text construction.

The rest of this paper will be devoted to the development of an example of a proofs as texts deduction system. We will apply the ideas discussed above to the $\{\wedge, \rightarrow, \bot\}$-fragment of intuitionistic propositional logic. First we will define the language of this fragment in a particularly suitable way and we will discuss the structural notions that will play an important role throughout the paper. Then we will go on and give a characterisation of derivability in terms of these structural notions. We will prove that our calculus does indeed give us precisely the theorems of intuitionistic logic. We will use the calculus to present an incremental proofs system for the logic. We have included an appendix with some familiar proof theoretic results about the fragment of intuitionistic propositional logic that we discus.

2 The language of proof texts

In this section we define the language that we will use for the representation of proofs. As was announced above, we will restrict ourselves to a propositional language and we will identify sentence level and text level operations. We have chosen to keep the sentence level notation. Hence our formulas will contain $\rightarrow$ as a sign both for implication and for entailment. Similarly $\wedge$ will be used for conjunction within a sentence as well as for concatenation of sentences. We will ignore all other connectives.
The objects of our proofs system will be (special) formulas of our language. We want to formulate the rules of our proofs system as conditions on the construction of these formulas. The conditions will have to guarantee that the formulas we build are exactly the valid formulas of the language.
These construction conditions will depend heavily on the structure of the formulas. The information about the structure of a formula tells us how the connectives in the formula are nested. In particular it will tell us how the implications in the formula are nested and thereby we will be able to tell which part of the text contains the assumptions that we can use at some point in the construction. We will use a representation of formulas as trees to characterise the structural information that we will need. Thus we will not only have a left to right order in the proof texts, but also a hierarchical order. These two ways to impose order on a text together will allow us to formulate proof rules as rules for building valid texts.
So the language that we will use is well known: it is simply the $\{\wedge, \rightarrow, \bot\}$-fragment of propositional logic. But instead of the usual inductive definition of this language, we define the formulas to be a special kind of *ordered labelled trees*. Of course there is an implicit tree structure in the formulas as we usually define them: their *construction* or *parsing* tree. Here we choose to make this structure explicit and identify the formulas with their parsing trees. We make this move for simplicity only: since there is a one-one correspondence between

formulas in the usual sense and the (parsing) trees that we will use as formulas, none of our results depend on this move. But as it is the structure of the parsing trees that we use for our characterisation of validity (in section 3), we can save some work by making the identification of formulas with their parsing trees from the start.

Definition 2.1

- ▷ *A tree is a pair* $\langle N, \leq \rangle$*, where* N *is a set, the set of nodes of the tree, and* $\leq$ *is an ordering on* N *such that:*
  - ○ *there is a maximum element* $r \in N$*, called the root of the tree, i.e.*
    $\exists r \in N \forall n \in N : n \leq r$.
  - ○ *for each* $n \in N$ *the set of nodes above* n*,* $\uparrow(n) = \{m \in N : n \leq m\}$*, is linearly ordered by* $\leq$*. So:*
    $\forall n \in N : \uparrow(n) \times \uparrow(n) \cap \leq$ *is a linear order on* $\uparrow(n)$.
- ▷ *Given a set* LAB *of labels we can define the notion of a tree with labels in* LAB*, or, for short, a labelled tree, as follows.*

  A labelled tree is a triple $\langle N, \leq, l \rangle$*, such that* $\langle N, \leq \rangle$ *is a tree and* l *is a mapping* $l : N \to LAB$.
- ▷ *An ordered labelled tree is a quadruple* $\phi = \langle N, \leq, l, \{\prec_n : n \in N\} \rangle$*, where* $\langle N, \leq, l \rangle$ *is a labelled tree and* $\{\prec_n : n \in N\}$ *is a family of relations such that each* $\prec_n$ *is a linear order on the daughters of* n.

Now we focus on a special set of labelled trees: the formulas of our propositional language. In the definition we use $D(n)$ for $\{m \in N : m \leq n \ \& \ \neg\exists k : m \leq k \leq n\}$, the set of daughters of n.

Definition 2.2 *Let* $ALPH = \{\wedge, \rightarrow, \bot, p_0, p_1, \ldots\}$ *be the set of labels. Now we define* $\mathcal{L}$ *to be a subset of the ordered trees with labels in* $ALPH$*:*
an ordered labelled tree $\langle N, \leq, l, \{\prec_n : n \in N\} \rangle \in \mathcal{L}$ *iff the following conditions are satisfied:*

- ▷ *If* $D(n) = \emptyset$*, then* $l(n) \notin \{\wedge, \rightarrow\}$.
- ▷ *If* $D(n) \neq \emptyset$*, then* $D(n)$ *contains exactly two elements and* $l(n) \in \{\wedge, \rightarrow\}$.

In the definition of $\mathcal{L}$ we have ensured that each tree in $\mathcal{L}$ is binary branching and that the labelling is such that the leaves of the tree are labelled with atomic propositions and all the internal nodes are labelled with a connective. This way we make sure that all the trees in $\mathcal{L}$ look like the parsing trees of the formulas in the usual sense.[4] So although we have chosen to define the language as a special set of labelled trees, it is clear that a formula in the usual sense can be obtained from a labelled tree by reading off the labels of the nodes.
Also the usual notions defined on formulas can easily be reformulated in tree terminology. For example, a subformula of a formula, ϕ say, is a tree, ψ say, that is itself a formula and that occurs as a subtree of ϕ. If there is more than one way to embed ψ in ϕ, then each way corresponds to a different occurrence of the same subformula. So when ϕ is given, the nodes

[4] Note that this way of defining construction trees works for every context free language. For more details on the correspondence of context free grammars and labelled trees we refer to [**?**].

of ϕ give us all the information about the subformulas of ϕ and their occurrences in ϕ. A formula ψ is a subformula of ϕ iff there is a node in ϕ such that ψ is (isomorphic to) the tree below this node. Each node that has this property determines a different occurrence of ψ in ϕ. The root of the tree corresponds to the only occurrence of ϕ as a subformula of ϕ.
In the rest of this paper we will use the lower case Greek letters ϕ, ψ, χ, ... as variables ranging over nodes. Thereby, strictly speaking, they correspond to *occurrences* of subformulas. But we will allow ourselves to use the same variables for subformulas if we feel that no confusion can arise.
Note that the tree ordering $\leq$ automatically gives us an ordering on the subformula occurrences of ϕ, the suboccurrence relation. We will also use $\leq$ for the subformula relation that can easily be obtained from the suboccurrence relation. In our notation we will use $=$ for the equality of subformulas and $\equiv$ for the equality of subformula occurrences, i.e. of nodes. $\stackrel{\text{def}}{=}$ is used in defining equations. Another important notation convention is that we will write down the formulas in the 'usual' way. Although officially formulas are trees we will use the traditional linear notation. So $(p \rightarrow q)$ stands for the tree with only three nodes: a root node and two daughter nodes, where the left daughter is labelled p, the right daughter is labelled q and the root has label $\rightarrow$. We will not spell out in detail how the traditional notation can be obtained for an arbitrary tree in $\mathcal{L}$: we trust that the reader can see how this is done.

Recall that we will not consider the formulas of this language as sentences, but more generally, as texts. A formula such as $(p \rightarrow q)$, for example, can stand for a text fragment such as:

> Let's assume p. Then it is safe to conclude q.

Note that in the texts that we use in real life to represent proofs all sorts of things are allowed that cannot be represented in our simple formal language. For example, in a proof text it can happen that the assumptions for some claim are given after the claim itself, as in:

> Then n^2 will be an even number if n itself is even.

We cannot represent such a situation in $\mathcal{L}$.
Another thing that is quite common in proof texts, but that will not be considered here, is the use of intermediate conclusions. In a proof text we usually find constructions such as:

> Assume that n is even. Then n^2 is even as well. So $n^2 + 1$ is odd. Therefore $\frac{1}{2}(n^2 + 1)$ is not an integer.

Here the second and the third sentence are intermediate conclusions that lead up to the final conclusion that $\frac{1}{2}(n^2 + 1)$ is not an integer. Such intermediate conclusions do not exist in $\mathcal{L}$. In the implications of $\mathcal{L}$ we only have room for the assumptions and the final conclusion. If we want to represent the intermediate conclusions we have to use a trick: for example, we could form a conjunction of both final and intermediate conclusions.
We choose to work within these limitations because we want to keep our representation language $\mathcal{L}$ as simple as possible. Thus we will be able to concentrate on the main point of the *proofs as texts* perspective: everything is a text. As was explained above, in this approach we will represent theorem proving as text construction. And while we are building

such a text we will have no other information at our disposal than what we find in the formula itself. In fact all that is left for us to use in formulating proof rules is the structure of the formulas.
In the next sections we will see that this is really all we need: the interaction of the linear and the hierarchical structure of texts together gives all the information we need in our deduction system.

In what follows we develop a deduction system for intuitionistic propositional logic in the *proofs as texts* style. We will do this in three steps: first we will discuss some important notions concerning the structure of formulas. Then we will show how these structural notions allow us to formulate criteria for the validity of the formulas of $\mathcal{L}$ (section 3). We will prove soundness and completeness theorems for these criteria in section 4.
Next we will actually define the deduction system in proofs as texts style (section 5): we will formulate rules for building formulas in such a way that validity is preserved. The rules will rely heavily on the criteria of section 3. Finally we will consider an alternative presentation of the deduction system that is reminiscent of the linear notation for natural deduction proofs.

3 The structure of formulas

In this section we define some structural notions on formulas. We will use these structural notions later on to give a structural characterisation of the valid formulas. The main idea of this characterisation can be illustrated with a few examples.

- ▷ For a formula of the form $(\phi \to \psi)$ to be valid, the information that we find in ϕ has to justify the information in ψ. For example in $(p \to p)$, the first occurrence of p justifies the second occurrence of p, but in $(q \to p)$ there is no information — only q — to justify the occurrence of p.
- ▷ In $(\phi \to (\psi \to \chi))$, $(\psi \to \chi)$ as a whole has support ϕ. We saw above that in $(\psi \to \chi)$ ψ is support for χ. Hence both ϕ and ψ may serve as evidence to justify the occurrence of χ. For example, both in $(p \to (q \to p))$ and in $(q \to (p \to p))$, the second occurrence of p is justified by the first occurrence of p.
- ▷ In $((\phi \to \psi) \to \chi)$, $(\phi \to \psi)$ is the support for χ. This implies that we can use ψ to support χ, provided we have sufficient support for ϕ. For example in $((p \to (q \to r)) \to (q \to (p \to r)))$ the final occurrence of r has as support p, q and $(p \to (q \to r))$. From $(p \to (q \to r))$ we may conclude $(q \to r)$ since we have support for p. But then we may conclude r, since we have support for q. Hence there is sufficient support for r. Thus the second occurrence of r in this formula is justified.

This kind of reasoning suggests that in a formula we can systematically locate the subformulas that can be used to justify the occurrence of other subformulas. In $(\phi \to \psi)$, ϕ is there to justify ψ. But this observation generalises to a structural characterisation that will allow us to recognise all valid formulas. In section 5 we will use this structural characterisation to give instructions for *building* valid formulas. Since we regard these formulas as texts, such a system of instructions is in fact a proofs system: it tells us how to build texts with preservation of validity.

3.1 Command relations on trees

Recall that the set of subformula occurrences of ϕ is ordered by the relation $\leq$. Each node in the tree represents an occurrence of a subformula of ϕ. Thus we can describe the structure of formulas in terms of structural notions on trees. Such notions are well known from the literature. We will use two kinds of relations on trees that are familiar from the literature on natural language syntax, *command* relations and *precedence* relations. For a general discussion we refer to [?], [?], here we will restrict ourselves to the relevant examples of these relations.

We will use these relations to characterise the support relation on subformulas: when is it the case that one occurrence of a subformula can be used to justify another occurrence of a(nother) subformula? To do this we need *precedence relations* and *command relations*. We need precedence relations for the following reason: a subformula occurrence ψ of ϕ can only support a subformula occurrence χ if the two are separated by an $\rightarrow$ sign. ψ has to be in the antecedent—i.e. left subtree—of this implication and χ has to be on the conclusion side, the right subtree. This will be what the $\rightarrow$-precedence relation gives us. Note that here the linear structure of texts is important: a subformula can support another subformula only if it occurs to the left of that formula, separated by an implication sign.

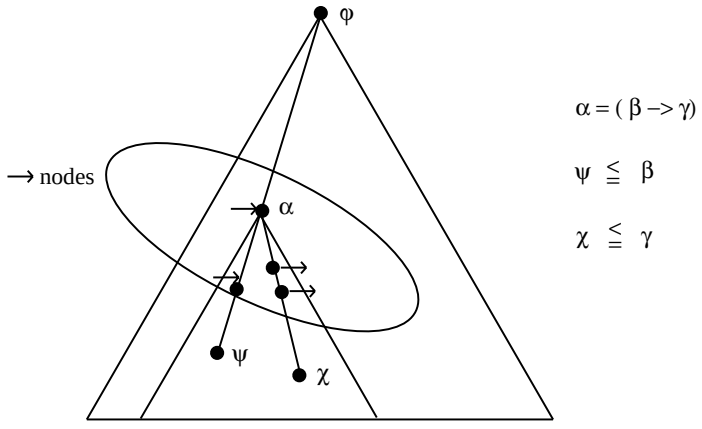
Let us call the set of nodes in ϕ that have as label $\rightarrow$, $Sub_{\rightarrow}(\phi)$. Then we can define the relation $Prec^{\rightarrow}_{\phi}$ as follows:

Definition 3.1

Let a formula ϕ be given and let ψ, $\chi \leq \phi$. Then $Prec^{\rightarrow}_{\phi}$ is the smallest solution of the following equation:[5]

$$\psi Prec^{\rightarrow}_{\phi} \chi \stackrel{\text{def}}{=}$$
$$\exists(\beta \rightarrow \gamma) \in Sub_{\rightarrow}(\phi): \quad \psi \leq \beta \ \ \& \ \ \chi \leq \gamma.$$

(Here the notation $\exists(\beta \rightarrow \gamma) \in Sub_{\rightarrow}(\phi)$ is used as shorthand for $\exists\alpha \in Sub_{\rightarrow}(\phi) : \exists\beta,\gamma : D(\alpha) = \{\beta,\gamma\} \ \wedge \ \beta \prec_{\alpha} \gamma$.) The following picture illustrates the situation.



[5] In the rest of the paper we will also only be interested in the smallest solution of such equations, even if we don't mention this explicitly.

The other structural relation that is important for the characterisation of the support relation is the command relation. We know that for ψ to support χ they have to be on alternate sides of an $\rightarrow$ and this is what the precedence relation gives us. But this is not enough: for example in $((p \rightarrow q) \rightarrow p)$, p and p are on alternate sides of an implication sign, but clearly p does not provide support for p in this situation. p can only support the material that occurs after the *first* implication sign that follows p. This means in the tree terminology that when ψ supports χ, the first $\rightarrow$-node above ψ will be above χ as well. This is known as a *command* relation. The command relation keeps an eye on the nesting of implications in a formula, i.e. it codes the hierarchical structure of a text.
We give the following definition.

Definition 3.2 *Fix $\phi \in \mathcal{L}$ and consider an arbitrary $\psi \leq \phi$.*

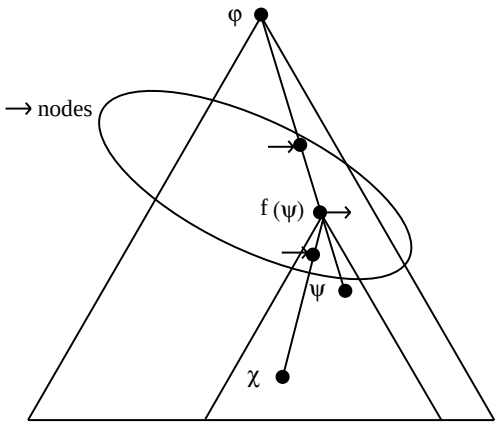
- $\triangleright$ *$f_{\rightarrow}(\psi)$ is the smallest subformula of ϕ that is larger than ψ and of which $\rightarrow$ is the main connective;*

$$
\begin{array}{l}
f_{\rightarrow}(\psi) \equiv \chi \stackrel{\text{def}}{=} \\
\exists \chi_1, \chi_2 \, (\chi \equiv (\chi_1 \rightarrow \chi_2) \,\&\, \psi \leq \chi \leq \phi \,\& \\
\quad \forall \chi', \chi_1', \chi_2' \, ((\chi' \equiv (\chi_1' \rightarrow \chi_2') \,\&\, \psi \leq \chi' \leq \phi) \Rightarrow \chi \leq \chi') \\
\vee \\
(\chi \equiv \phi \,\&\, \neg \exists \chi', \chi_1', \chi_2' \, (\chi' \equiv (\chi_1' \rightarrow \chi_2') \,\&\, \psi \leq \chi' \leq \phi))
\end{array}
$$

- $\triangleright$ *$\psi \; Com_{\phi}^{\rightarrow} \; \chi$, we say ψ $\rightarrow$-commands χ in ϕ;*

$$\psi \; Com_{\phi}^{\rightarrow} \; \chi \stackrel{\text{def}}{=} \chi \leq f_{\rightarrow}(\psi)$$

The following picture illustrates the situation.



We have argued that for ψ to support χ, ψ has to $\rightarrow$-precede and $\rightarrow$-command χ: the supporting subformula has to occur before the supported subformula and it should be nested into the implicational structure of the formula in the right way. Therefore we define the $\rightarrow$-precede-and-command relation.

Definition 3.3 *Let ϕ, ψ, $\chi \in \mathcal{L}$ be given.* $\psi\ PC^{\rightarrow}_{\phi}\ \chi$

$$\psi\ PC^{\rightarrow}_{\phi}\ \chi \stackrel{\text{def}}{=} \psi\ Com^{\rightarrow}_{\phi}\ \chi\ \&\ \psi\ Prec^{\rightarrow}_{\phi}\ \chi.$$

So, for example, in $\phi = ((p \rightarrow q) \rightarrow p)$ we see a situation where $p\ Prec^{\rightarrow}_{\phi} p$, but not $p\ Com^{\rightarrow}_{\phi} p$. Therefore p does not precede and command p in this formula. In $(p \rightarrow (q \rightarrow p))$ on the other hand we see that both $p\ Prec^{\rightarrow}_{\phi} p$ and $p\ Com^{\rightarrow}_{\phi} p$, so in this formula the first occurrence of p does precede and command the second occurrence of p. Note that the first occurrence is not preceded and commanded by the second occurrence in either case.
It is important to notice that in the definition of the command relation we do not find an existence condition: we take ϕ itself as a default value for $f_{\rightarrow}(\psi)$. But the precedence relation, as we defined it, demands that there is at least one subformula in $Sub_{\rightarrow}$ of which both ψ and χ are subformulas. Hence in $\psi \wedge \chi$, ψ will $\rightarrow$-command χ but it does not $\rightarrow$-precede χ. Hence ψ will not count as support for χ in this case.
We have defined precedence and command relations with respect to the set of $\rightarrow$-nodes in ϕ. But the notions of precedence and command make sense for any set of nodes S in ϕ. Hence the definitions generalise to definitions of S-precedence, written $Prec^{S}_{\phi}$, and S-command, written Com^{S}_{ϕ}, if we replace $Sub_{\rightarrow}(\phi)$ by S in the definitions. In what follows we will treat $S = Sub_{\rightarrow}(\phi)$ as default. We will allow ourselves to omit the superscripts unless $S \neq Sub_{\rightarrow}(\phi)$.[6]

3.2 Support relations on subformulas

In this subsection we apply the structural notions $\rightarrow$-precedence and $\rightarrow$-command to define the support relation $\psi \sqsubset_{\phi} \chi$ on subformula occurrences of ϕ. We define $\sqsubset_{\phi}$ in such a way that $\psi \sqsubset_{\phi} \chi$ implies that the subformula χ actually needs support. For example in $(p \rightarrow q)$ it is q, not p, that needs support. p is just an assumption an it is always allowed to make extra assumptions. But for q the situation is different: there we have to be careful if we want to preserve validity. We will call subformula occurrences that need support *claims*. For each ϕ the set of claims of ϕ can be defined as follows.

Definition 3.4 *Let ϕ, $\psi \in \mathcal{L}$ be given. We define $Cl_{\phi}(\psi)$, to express that ψ is claimed in the proof text ϕ;*

$$Cl_{\phi}(\phi)\ \&$$

$$Cl_{\phi}((\psi_1 \wedge \psi_2)) \Rightarrow Cl_{\phi}(\psi_1)\ \&\ Cl_{\phi}(\psi_2)\ \&$$

$$Cl_{\phi}((\psi_1 \rightarrow \psi_2)) \Rightarrow Cl_{\phi}(\psi_2)$$

Examples are:

- $p \wedge q$: $Cl_{p \wedge q} = \{\, p,\ q,\ p \wedge q \}$
- $(p \rightarrow q)$: $Cl_{p \rightarrow q} = \{\, q,\ p \rightarrow q \}$

Now we define the support relation as follows.

Definition 3.5 *Let $\psi, \chi \leq \phi$ be given. Then*

[6] Cf. [?], [?] for more on command relations in general.

$$\psi \sqsubset_\phi \chi \quad \stackrel{\text{def}}{=} \quad Cl_\phi(\chi) \ \& \ \psi \, PC_\phi \chi$$

So we say that a subformula occurrence supports a claim, precisely if it precedes and commands the claim. We can illustrate this with the following example: in the previous subsection we saw that in $(p \to (q \to p))$ the first occurrence of p precedes and commands the second occurrence of p. Since the second occurrence is in a claim position we may conclude that p supports p in this case. We can also see that the first occurrence of p precedes and commands q. But since q is not a claim of the formula p does not *support* q.
It is on this notion of support that our characterisation of valid formulas will be based. Basically we will say that a formula is valid if all its claims can be justified. The notion of justification will then be defined recursively: either a claim is justified directly because the formula contains an occurrence of the claim that supports it — as in $(p \to p)$ — or the claim is supported indirectly, as will be explained shortly.
The connection of this notion of support with our interest in texts that represent proofs is as follows: in reading a proof text we might at some point want to check a claim by the author. At such a point we have to know where we can find the standing assumptions: does the author still assume that we are living in a world where all men have three feet and all women five arms or is this assumption already cancelled? To check this we have to look at the place in the text where this assumption was made and see whether it was cancelled between that point and the point where we are know. This is exactly what our support relation does for us: given two places in the text, it tells us whether the assumption that we find in one place is still in force at the other place. Whenever we are reading a proof text we implicitly use the support relation to keep track of the standing assumptions.

3.3 Justifying claims

In this subsection we define a predicate $Just(\psi, \phi)$ that will express that the claim ψ of the text ϕ is justified. Our way of checking this was indicated at the beginning of this section: for a claim ψ we check whether there is sufficient support. This means that we search among the formulas that support the claim – as defined above – for material that actually justifies the occurrence of the claim. For example, in $(p \to p)$, the supporting material p clearly justifies the occurrence of the claim p. We will have several proof rules that determine which conclusions can be drawn from some set of supporting material.
In the examples that we have described we see that the justification of a claim consists of a trip through the formula searching for support. Thus a successful attempt at justification can be represented by the subformulas that we meet on such a trip. We could include a representation of these trips. That way we can have some kind of annotation of proofs in our system. Then we would have to define a three place predicate $Just(\xi, \psi, \phi)$, that expresses that ξ represents a justification of the claim ψ that occurs in ϕ.
Here we do not follow this direction. Right now we are content to ignore to possibility to add annotation in order to keep things simple. We have already made a similar decision when we chose propositional logic as a representation language. Thereby our language of proofs has become too poor to actually *contain* any kind of annotation. Adding the annotation now would be a rather artificial attempt to make up for this limitation. Instead we stick to our strategy of keeping things simple.
So we define the *two*-place predicate $Just(\psi, \phi)$ to express that ψ is a justified claim of ϕ. If ϕ is fixed by the context we will write $Just_\phi(\psi)$ or even simply $Just(\psi)$ instead. We will adopt a similar convention for Cl_ϕ. Also recall that the variables ϕ, ψ, χ, ξ ... range over

nodes, i.e. subformula occurrences. If we use $=$ this means that we are only interested in the subformulas below the nodes.

Definition 3.6 *Let $\phi \in \mathcal{L}$.*

$Just(\psi, \phi) \stackrel{\text{def}}{=}$

(direct proof)
$\exists \xi : \xi \sqsubset_\phi \psi \ \& \ (\xi = \psi \ \vee \ \xi = \bot)$

(conjunction of proofs)
$\vee \quad \exists \psi_1 \psi_2 : \psi \equiv (\psi_1 \wedge \psi_2) \ \&$
$Just(\psi_1, \phi[\psi := \psi_1]) \ \&$
$Just(\psi_2, \phi[\psi := \psi_2])$

(proof of an implication)
$\vee \quad \exists \psi_1 \psi_2 : \psi \equiv (\psi_1 \rightarrow \psi_2) \ \&$
$Just(\psi_2, \phi)$

(generalised modus ponens)
$\vee \quad \exists \xi : (\ Cl_\xi(\psi) \ \&$
$Just(\bigwedge\{\chi : \chi \sqsubset_\xi \psi\}, \phi[\psi := \bigwedge\{\chi : \chi \sqsubset_\xi \psi\}]) \ \&$
$Just(\xi, \phi[\psi := \xi]) \ )$
$\vee \ (\ Cl_\xi(\bot) \ \&$
$Just(\bigwedge\{\chi : \chi \sqsubset_\xi \bot\}, \phi[\psi := \bigwedge\{\chi : \chi \sqsubset_\xi \bot\}]) \ \&$
$Just(\xi, \phi[\psi := \xi]) \ )$

(Here $\phi[\psi := \chi]$ stands for the result of replacing the occurrence ψ in ϕ by χ. We write $\bigwedge A$ for the conjunction of all the formulas in the set A. We can either rely on the fact that specific bracketings and orderings of conjunctions are irrelevant or else use some fixed bracketing and ordering strategy.)

The definition looks rather threatening, but the examples later on will surely help to get a feel for the system.
The definition of $Just$ distinguishes four cases. The first case is the case of direct justification. Here the claim that has to be justified is one of the subformulas that support it or else the claim is supported by $\bot$. Prototypical examples are

$Just(p, (p \rightarrow p))$ and
$Just(p, (\bot \rightarrow p))$ and
$Just(p, ((p \wedge q) \rightarrow p))$.

The second case says that we can justify a claim of the form $(\psi_1 \wedge \psi_2)$ by justifying both ψ_1 and ψ_2. An example of this is

$Just((p \wedge q), (p \rightarrow (q \rightarrow (p \wedge q))))$.

The third case treats claims of the form $(\phi \rightarrow \psi)$. In these cases it suffices to justify the claim of this claim, i.e. ψ. Note that case two and three are similar in that they both say that to justify a complex claim it suffices to justify the sub-claims of this complex claim.

In case of a conjunction this means that we have to justify both conjuncts, in case of an implication we have to justify the consequent of the implication. Thereby one justification strategy that one could follow, is to look for justification of the atomic claims of a formula only. According to cases two and three this will suffice to justify all claims.[7] But this is not always the most efficient strategy. For example in

$$((p \to q) \to (p \to q))$$

it suffices to justify the second occurrence of q, the only atomic claim of this formula, and in fact this can be done, using generalised modus ponens. But it is easier to justify $(p \to q)$ as a whole, using the direct proof rule.

Case four can be seen as a generalised version of modus ponens. It can be applied in case the claim that has to be justified occurs as a claim of some other formula that would itself be a justified claim at this point. So if we have no direct evidence for a claim, ψ say, but we do have evidence for ξ, which has ψ as a claim, then we can try to apply generalised modus ponens (gmp) with this ξ. In this case we call ξ the *major premise* of the application of generalised modus ponens. For the rule to be applicable it must be the case that all the formulas that support ψ in ξ, $\{\chi : \chi \sqsubset_\xi \psi\}$, are now justifiable.

Note that we also have the case where not ψ, but $\bot$ occurs as a claim of ξ. So in general we are looking for a ξ that has a claim that gives a *direct proof* of our current claim.

A prototypical case is of an application of gmp is

$$((p \wedge (p \to q)) \to q).$$

Here the second occurrence of q is a claim that has to be justified. There is no direct support for q, but one of its supporting formulas contains q as a claim, namely $(p \to q)$. Now the direct proof rule tells us that this supporting formula would be a justified claim at this point. According to the generalised modus ponens rule it suffices to justify all the subformulas of this alternative claim that support the claim. In our example this means that it suffices to justify p, which is the only support for q in $(p \to q)$. Since p is in fact one of the formulas that supports our claim q we are done.

The example shows that the rule has modus ponens as a special case. But the rule allows for more than that. Basically the rule is a generalisation of modus ponens in three ways. First note that in a nested implication the rule allows us to eliminate several assumptions in one sweep. To justify

$$(((p \wedge q) \wedge (p \to (q \to r))) \to r)$$

we only need one application of generalised modus ponens. In a natural deduction formulation of the logic we would have to apply modus ponens twice: once to eliminate p and once to eliminate q from $(p \to (q \to r))$. But here both assumptions are eliminated at once.

The second kind of generalisation that is built in is that there can be some 'implicit conjunction elimination' going on in generalised modus ponens: our version of modus ponens will also work if the required conclusion is only one of several claims of the major premise. For example in

$$((p \wedge (p \to (r \wedge q))) \to q)$$

[7] More on this topic can be found in the appendix where we prove an *Inversion Lemma* that makes this statement precise.

q can be justified with one application of generalised modus ponens.
These first two generalisations of modus ponens are an advantage of our system: they allow us to ignore parts of the structural organisation of a text that are not relevant now. We only have to know which formulas support the current claim and not the precise way in which these formulas occur in the organisation of the text.

The third generalisation that is built into the generalised modus ponens rule is a kind of recursion. If we apply generalised modus ponens with some major premise ξ, we have to justify both ξ itself and all the formulas in ξ that support our claim ψ. Each of these justifications can in principle be a justification by generalised modus ponens. We are then in a situation where we have to apply generalised modus ponens with one major premise in order to be able to apply modus ponens on another major premise. The simplest example of such a situation is

$$(((p \to q) \wedge (q \to r)) \to (p \to r))$$

Here the justification of the claim r might proceed by modus ponens on $(q \to r)$. This major premise itself is easily justified (by direct proof), but we also have to justify q. In order to justify q we can then again apply modus ponens, with major premise $(p \to q)$ this time. In this second application the major premise itself is easily justified (by direct proof) and this time also the justification of the supporting material p is easy (also direct proof). So we see that in the application of generalised modus ponens on $(q \to r)$ another application of generalised modus ponens is invoked.
This third generalisation is not so attractive. It is because of this kind of implicit recursion that it is quite hard at times to recognise a text in our system as a proof. It would be nice if we could make some of this recursion explicit in the text, but as it stands we do not have this option: we have chosen to work with a poor representation language in which such information cannot be represented. So we will have to accept this kind of implicit recursion for now. But, clearly, we can improve on this at some point in the future.

Another way of viewing the situation is as follows: at this point we have a proofs system in which we can check whether validity is preserved throughout the text. This certainly is an important issue that any proofs system will have to take care of in some way. But if we think of a proof as a text, then we would like to be able to make a distinction between a proof and an arbitrary valid text. Certainly a proof is more than a text in which the validity of each step is guaranteed: it is a text in which the validity of each step is *obvious*. In a proof all the ideas have been elaborated on to the point where only very simple deduction steps remain. The validity of each proof step by itself should be obvious. Since we are using a simple representation language we have a system with proof texts of which the semantics is obvious, but which are not so obvious *qua* proofs since the annotation is missing.
Probably the best choice in the end will be to include some of the annotation in the texts. For example, in an application of generalised modus ponens the major premise has the flavour of an intermediate conclusion. We already said that we may want to include intermediate conclusions in the proof texts at some point, so this is a good candidate for annotation that we may want to include in the texts. But for now we stick to the strategy of keeping all the annotation out of the text.

So now we are at a point where we have defined a system by which we can check whether a certain claim in a text is justified. In the following subsection we give some more examples of the way in which this system works.
Our goal in the end is to *construct* texts which are themselves justified, i.e. justified texts. This is what we do in section 5. For now we conclude with the following definition of justified texts:

Definition 3.7

$$Jf(\phi) \stackrel{\text{def}}{=} Just(\phi, \phi)$$

expressing that ϕ is a justified claim of ϕ, i.e. ϕ is a justified text.

3.4 Examples

In this section we show how justifications of formulas can be obtained. We give two extensive examples of formulas and their justification. The reader who wants to know what happens to formulas that do not have a justification is referred to the appendix, where a decision procedure is discussed and it is shown that *Peirce's Law* cannot be justified in our system.

In these examples we use the inductive character of the definition: $Just$ is the smallest set that is closed under the inductive clauses. So we can show that some pair (ψ, ϕ) is justified by producing a sequence $Just(\psi_1, \phi_1) \ldots Just(\psi_n, \phi_n)$ where $\psi_n = \psi$, $\phi_n = \phi$ and each component of the sequence is either an instance of direct proof or follows from previous components in the sequence by an application of one of the inductive clauses. We will construct these sequences in the reverse order: starting from our goal $Just(\psi, \phi)$ we look for subgoals $Just(\psi_i, \phi_i)$ until we get instances of direct proof.

Our first example is the law of contraposition: $\gamma = ((\phi \to \psi) \to ((\psi \to \bot) \to (\phi \to \bot)))$.[8] Our aim is to show that $Jf(\gamma)$. To do this we show that:

$$Just(\gamma, \gamma).$$

We start by justifying the smallest claim of γ, i.e. $\bot$. This can be done as follows.

We try to use generalised modus ponens. We need a ξ such that:

- $Cl_\xi(\bot)$
- $Just(\xi, \gamma[\bot := \xi])$
- $Just(\bigwedge\{\chi : \chi \sqsubset_\xi \bot\}, \gamma[\bot := \bigwedge\{\chi : \chi \sqsubset_\xi \bot\}])$

We take $\xi = (\psi \to \bot)$, so we now need:

- $Just((\psi \to \bot), \ ((\phi \to \psi) \to ((\psi \to \bot) \to (\phi \to (\psi \to \bot)))))$
- $Just(\psi, \ ((\phi \to \psi) \to ((\psi \to \bot) \to (\phi \to \psi))))$

[8] Actually this is not a formula, but a formula scheme. As a consequence we will not find a justification, but a scheme for justification. Note also that in intuitionistic logic this formula is true but the implication cannot be reversed: we do not have $(((\psi \to \bot) \to (\phi \to \bot)) \to (\phi \to \psi))$.

We see that the first statement follows by direct proof. For the second statement we perform another application of generalised modus ponens. This time we will use $(\phi \to \psi)$ as major premise:

- $\triangleright$ $\xi_1 = (\phi \to \psi)$
- $\triangleright$ $Just(\xi_1, ((\phi \to \psi) \to ((\psi \to \bot) \to (\phi \to \xi_1))))$
- $\triangleright$ $Just(\phi, ((\phi \to \psi) \to ((\psi \to \bot) \to (\phi \to \phi))))$

Now we see that in both cases we can use *direct proof*.

Now we have justified the smallest claim of $\gamma, \bot$. It follows (by several applications of *proof of an implication*) that γ itself is justified.

The second example we present is one of the axioms for intuitionistic logic that we will see again in the next section. We consider the axiom $\alpha = ((\phi \to (\psi \to \chi)) \to (\phi \to \psi) \to (\phi \to \chi))$ and try to show $Jf(\alpha)$. So we have to show that:

$Just(\alpha, \alpha)$

Again we start by justifying the smallest claim first:

$Just(\chi, ((\phi \to (\psi \to \chi)) \to (\phi \to \psi) \to (\phi \to \chi)))$.

This can be done as follows:

We apply generalised modus ponens:

- $\triangleright$ $\xi = (\phi \to (\psi \to \chi))$,
- $\triangleright$ $Just(\xi, ((\phi \to (\psi \to \chi)) \to (\phi \to \psi) \to (\phi \to \xi)))$,
- $\triangleright$ $Just(\phi \wedge \psi, ((\phi \to (\psi \to \chi)) \to (\phi \to \psi) \to (\phi \to (\phi \wedge \psi))))$.

These two justifications are obtained as follows:

- $\triangleright$ the first statement holds by direct proof.
- $\triangleright$ the second statement follows by conjunction of proofs, as follows:
  - $\circ$ $Just(\phi, ((\phi \to (\psi \to \chi)) \to (\phi \to \psi) \to (\phi \to \phi)))$ and
  - $\circ$ $Just(\psi, ((\phi \to (\psi \to \chi)) \to (\phi \to \psi) \to (\phi \to \psi)))$

again the first statement follows by direct proof.

for the second statement we need another application of generalised modus ponens, with $\xi_1 = (\phi \to \psi)$ as major premise. Now both

- $\triangleright$ $Just(\xi_1, ((\phi \to (\psi \to \chi)) \to (\phi \to \psi) \to (\phi \to \xi_1)))$ and
- $\triangleright$ $Just(\xi_1, ((\phi \to (\psi \to \chi)) \to (\phi \to \psi) \to (\phi \to \phi)))$

follow by direct proof.

From this justification of the claim χ in α we get a justification for α itself by several applications of the rule for the proof of an implication. This way we get:

$Just(\alpha, \alpha)$

as required.[9]

[9] Perhaps the reader will find it useful to check which routes through the tree/text are made in these examples of justifications.

3.5 Other logics

So far we have concentrated on intuitionistic logic, and this is also what we will do in the remaining part of the paper. But the choice for intuitionistic logic over some other logic was completely arbitrary: the proofs as texts approach makes sense for any logic. So now that we have given the definitions for intuitionistic logic it may be helpful to think about using a similar system for other logics, just to make clear that our choice really was made for convenience only.
For example, if we want to obtain a system for classical propositional logic, we will have to include a rule for double negation one way or the other. The easiest way is probably to first add a direct proof rule that says that if $((\psi \to \bot) \to \bot) \sqsubset_\phi \psi$, then ψ is justified. Then we can extend the rule for generalised modus ponens: at present we search for ξ such that either $Cl_\xi(\psi)$ or $Cl_\xi(\bot)$ whenever we want to prove a claim ψ with generalised modus ponens. If we extend the rule to allow for ξ with $Cl_\xi(((\psi \to \bot) \to \bot))$, then we will obtain a system that is equivalent to classical logic.[10]
So we can obtain a stronger logic by strengthening the rules. Similarly a weaker logic could be obtained by weakening certain rules. For example, we could make the system resource sensitive by checking exactly which part of the supporting material of a claim is actually used in its justification. Then appropriate restrictions on the use and re-use of this material can be formulated. We will not go into details here. Suffice it to say that in the case of substructural logic the need for annotation of proofs becomes even more urgent, but once suitable annotation is added nothing seems to prevent us from formulating so-called substructural logics in proofs as texts style.

4 Soundness and Completeness

In this section we prove that our system for justification of claims in formulas gives us exactly the (intuitionistic) tautologies. In particular we will show that:

$\vdash_{IL} \psi \;\Rightarrow\; Jf(\psi)$
and
$Jf(\phi) \;\Rightarrow\; \vdash_{IL} \phi.$

In other words, ψ is a theorem of intuitionistic logic if and only if there is a justification for ψ.
To prove this we first fix a version of IL. We will use a formulation of the calculus in Hilbert style. The details of the formulation are not essential, but it will help us fix our thoughts.

Definition 4.1 *We define the calculus IL as follows:*

- *IL is based on the following axiom schemata:*
  - $((\phi \wedge \psi) \to \phi)$
  - $((\phi \wedge \psi) \to \psi)$
  - $(\phi \to (\psi \to (\phi \wedge \psi)))$

[10] This can be shown with a straightforward extension of the soundness and completeness proofs in the following section.

- $(\phi \to (\psi \to \phi))$
- $((\phi \to (\psi \to \chi)) \to ((\phi \to \psi) \to (\phi \to \chi)))$
- $(\bot \to \phi)$

▷ *The only rule of IL is modus ponens:*

$$\vdash_{IL} (\phi \to \psi) \ \& \ \vdash_{IL} \phi \ \Rightarrow \vdash_{IL} \psi$$

▷ *ϕ is a theorem of IL, $\vdash_{IL} \phi$, iff ϕ is an instance of one of the axiom schemata or ϕ is derived from such instances by a finite number of applications of modus ponens.*

Since the calculus IL is sound and complete for the intended interpretation, we can confuse it with any other such calculus or with the (semantic) entailment relation of intuitionistic logic whenever this is convenient.

4.1 Soundness

Now we can prove the following proposition.

Proposition 4.2 (Soundness) *Let $\phi \in \mathcal{L}$ be given. Then:*

$$Jf(\phi) \ \Rightarrow \models_{IL} \phi$$

Since the Hilbert system presented above is complete for the intended interpretation, the proposition does in fact imply that what we can justify in our system can also be proved in the Hilbert system. In other words from the proposition we get:

Corollary 4.3

$$Jf(\phi) \ \Rightarrow \vdash_{IL} \phi$$

The proposition is a direct consequence of the following lemma.

Lemma 4.4 *Let ϕ, ψ, $\xi \in \mathcal{L}$ be given. Then*

$$Just(\psi, \phi) \ \Rightarrow \ \{\chi : \chi \sqsubset_\phi \psi\} \models_{IL} \psi$$

Proof:
Note that $Just(\psi, \phi)$ is defined inductively. Thereby there is for each ψ, ϕ such that $Just(\psi, \phi)$ a sequence

$$Just(\psi_1, \phi_1) \ldots Just(\psi_n, \phi_n)$$

such that $\psi_n = \psi$, $\phi_n = \phi$ and each component of the sequence is either an instance of direct proof or can be obtained by one of the other proof rules from components that precede it in the sequence.
The proof is by induction on the length of these sequences. In the proof we will omit the subscript in $\models_{IL}$.
In some steps we will make use of the following lemma:

Lemma 4.5
$\chi \sqsubset_\phi \psi \quad \textit{iff } \chi \sqsubset_{\phi[\psi:=\alpha]} \alpha$

This lemma says that $\sqsubset$ only depends on the structure of the tree, not on the material that is stored there. The proof of the lemma is omitted.

Now the inductive proof proceeds as follows:

- ▷ Basic cases:
  - ∘ $\bot \sqsubset_\phi \psi$.
    Since $\bot \models \psi$, clearly also $\{\chi : \chi \sqsubset_\phi \psi\} \models \psi$.
  - ∘ $\psi \sqsubset_\phi \psi$.
    Since $\psi \models \psi$, clearly also $\{\chi : \chi \sqsubset_\phi \psi\} \models \psi$.
- ▷ Conjunction of proofs:
  Now $\psi = (\psi_1 \wedge \psi_2)$ and $\mathit{Just}(\psi_i, \phi[\psi := \psi_i])$.
  By the lemma we know that:

  $$\{\chi : \chi \sqsubset_\phi \psi\} \;=\; \{\chi : \chi \sqsubset_{\phi[\psi:=\psi_1]} \psi_1\} \;=\; \{\chi : \chi \sqsubset_{\phi[\psi:=\psi_2]} \psi_2\}.$$

  Now the induction hypothesis gives us (for $i = 1, 2$):

  $$\{\chi : \chi \sqsubset_{\phi[\psi:=\psi_i]} \psi_i\} \;\models \psi_i.$$

  Hence, by the lemma:

  $$\{\chi : \chi \sqsubset_\phi \psi\} \;\models \psi_i.$$

  From which we may conclude

  $$\{\chi : \chi \sqsubset_\phi \psi\} \;\models (\psi_1 \wedge \psi_2)$$

  as required.
- ▷ Proof of implication:
  We know that $\mathit{Just}(\psi_2, \phi)$.
  The key observation is that

  $$\{\chi : \chi \sqsubset_\phi \psi_2\} \text{ and } \{\psi_1\} \cup \{\chi : \chi \sqsubset_\phi \psi\}$$

  are logically equivalent. (This can be checked easily.)
  The induction hypothesis guarantees:

  $$\{\chi : \chi \sqsubset_\phi \psi_2\} \;\models \psi_2$$

  By the key observation this reads as

  $$\{\chi : \chi \sqsubset_\phi \psi\} \cup \{\psi_1\} \;\models \psi_2$$

  and thereby

$$\{\chi : \chi \sqsubset_\phi \psi\} \models (\psi_1 \to \psi_2)$$

as required.

▷ Generalised modus ponens:

Now a major premise ξ is given and either $Cl_\xi(\psi)$ or $Cl_\xi(\bot)$. We consider the case $Cl_\xi(\psi)$: the other case is similar.

$Just(\bigwedge\{\chi;\ \chi \sqsubset_\xi \psi\},\ \phi[\psi := \bigwedge\{\chi;\ \chi \sqsubset_\xi \psi\}])$ and $Just(\xi, \phi[\psi := \xi])$.

If we apply the lemma to the induction hypothesis we get first:

$$\{\chi : \chi \sqsubset_\phi \psi\} \models \bigwedge\{\chi : \chi \sqsubset_\xi \psi\}$$

and second:

$$\{\chi : \chi \sqsubset_\phi \psi\} \models \xi.$$

Since $Cl_\xi(\psi)$, we know that ξ is of the form

$$(\ldots \wedge (\alpha_1 \to (\ldots(\alpha_n \to (\ldots \wedge \psi \wedge \ldots))\ldots)) \wedge \ldots),$$

where each α_i is a χ such that $\chi \sqsubset_\xi \psi$.

Our first conclusion from the induction hypothesis guarantees that

$$\{\chi : \chi \sqsubset_\phi \psi\} \models \alpha_i.$$

for $1 \leq i \leq n$.

Now we can apply modus ponens n times to the second conclusion from the induction hypothesis to obtain

$$\{\chi : \chi \sqsubset_\phi \psi\} \models (\ldots \wedge \psi \wedge \ldots)$$

From this we obtain

$$\{\chi : \chi \sqsubset_\phi \psi\} \models \psi$$

by several eliminating the superfluous conjuncts, as required.

□

4.2 Completeness

Now we go on to prove the following proposition:

Proposition 4.6 (Completeness) *Let $\phi \in \mathcal{L}$ be given. Then:*

$$\vdash_{IL} \phi \ \Rightarrow\ Jf(\phi).$$

Thereby the justification system of the previous section inherits the completeness of IL.
The proposition can be proved as follows:

Proof:
The proof is an induction on the length of the derivation of $\vdash_{IL} \phi$. In the proof we will omit the subscript of $\vdash_{IL}$.

- ▷ First we justify the central claims of the axiom schemata.
  - ◦ $Just(\phi,\ (\phi \wedge \psi) \rightarrow \phi)$;
  - ◦ $Just(\psi,\ (\phi \wedge \psi) \rightarrow \psi)$;
  - ◦ $Just(\phi \wedge \psi,\ \phi \rightarrow (\psi \rightarrow (\phi \wedge \psi)))$;
  - ◦ $Just(\phi,\ \phi \rightarrow (\psi \rightarrow \phi))$;
  - ◦ $Just(\chi,\ ((\phi \rightarrow (\psi \rightarrow \chi)) \rightarrow (\phi \rightarrow \psi) \rightarrow (\phi \rightarrow \chi)))$;
    This has been done in section 3.4.
  - ◦ $Just(\phi,\ (\bot \rightarrow \phi))$;

  Now the justifications for the axiom schemata themselves can be obtained by several applications of the 'proof of an implication' rule. For example, since $Just(\phi,\ (\bot \rightarrow \phi))$, we have $Just((\bot \rightarrow \phi),\ (\bot \rightarrow \phi))$, as required.

- ▷ Next we consider an application of modus ponens in IL.

  Suppose $\vdash\ (\phi \rightarrow \psi)$ and $\vdash\ \phi$.

  Then (by induction hypothesis):

  $Just((\phi \rightarrow \psi),\ (\phi \rightarrow \psi))$ and

  $Just(\phi,\ \phi)$.

  Now we can use generalised modus ponens to conclude that

  $Just(\psi,\ \psi)$, as required.

□

We can conclude that the predicate $Jf(.)$ that we have defined in the previous section picks out exactly the theorems of intuitionistic logic.[11] This shows that our enterprise of regarding proofs as (special) formulas is not without a chance: we have shown that, if we regard our formulas as texts, we will actually be able to check for each text whether it is the kind of text that represents a proof. We are able to do this by using the structure of the text. The structure of the text tells us which assumptions are in force at each point in the text. Once we know this, things are easy: we just check whether the assumptions warrant the conclusion, either directly or indirectly via some applications of modus ponens.
The next step will be to use the criteria that we have developed to *check* whether some given text is valid as criteria for *building* texts that are valid. This is the topic of the next section.

[11] In the appendix we prove some interesting properties of our system. The properties we discuss are familiar from (intuitionistic) proof theory. In the appendix we will see how they look in the new set up.

5 Writing proof texts

In this section we will use the structural characterisation of validity to give rules for the incremental construction of valid formulas. Since the formulas are considered as representations of texts, this will in fact give us a deduction system for our logic: we will haves rule for building up proof texts.
At the end of this section we will give a presentation of the deduction system that is reminiscent of linear presentations of natural deduction.

5.1 Proofs

As we are constructing valid formulas of $\mathcal{L}$, we will sometimes have to work with constructs that are not strictly speaking in $\mathcal{L}$. As we are building up the proof step by step we will go through stages in which what we have cannot be represented as a formula of $\mathcal{L}$. Consider for example a situation where we have just made a few assumptions and have not yet drawn any conclusions from these assumptions. Then we are in a situation which cannot be described properly in $\mathcal{L}$: we are building up an implication, $(\phi \to \psi)$, and have just completed the antecedent ϕ of the implication, but we have not yet drawn any conclusion. So our construct looks something like $(\phi \to \ldots$.
In [?], [?] we have developed special machinery for describing these incomplete texts. The machinery could very well be adapted to the present situation, but this would lead us into issues that need not concern us now. Instead we use the following trick to represent incomplete texts: we introduce a new propositional constant $\star$ to represent incompleteness. The resulting language will be called $\mathcal{L}^\star$. $\mathcal{L}^\star$ inherits all structural notions, such as the definitions of precede and command relations, from $\mathcal{L}$. In $\mathcal{L}^\star$ we can distinguish the complete texts from the incomplete text by testing whether $\star \leq \phi$.
Note that this way we do not only get unfinished formulas in $\mathcal{L}^\star$, but also formulas with holes in the middle or even at the start. We do not only get

$$(\phi \to (\psi \to \chi) \wedge \star),$$

but also

$$(\phi \to (\star \to \chi) \wedge \xi),$$
$$(\star \to (\psi \to \chi) \wedge \xi), \text{ etc.}$$

For the proofs system we will not need formulas with holes: we will build up the proof texts step by step from left to right, so only unfinished formulas can occur. We will call such formulas prooflike.
The following definition summarises this discussion:

Definition 5.1

- ▷ *$\mathcal{L}^\star$ is defined as $\mathcal{L}$, but we replace $ALPH$ by $ALPH^\star = ALPH \cup \{\star\}$.*
- ▷ *We call $\phi \in \mathcal{L}^\star$* **incomplete** *iff $\star \leq \phi$. Else we call ϕ* **complete**.
- ▷ *We call $\phi \in \mathcal{L}^\star$* **prooflike** *iff for all $\psi \leq \phi$:*

$$\star \, Prec^{\to}_{\phi} \, \psi \;\Rightarrow\; l(\psi) \notin ALPH \text{ and}$$
$$\star \, Prec^{\wedge}_{\phi} \, \psi \;\Rightarrow\; l(\psi) \notin ALPH.$$

We see that prooflike formulas are such that wherever a $\star$ occurs everything to the right of this is also a $\star$. This way no holes can occur in a prooflike formula.

In our proofs system we will be building up these prooflike formulas. Here the act of building up will be represented by substituting material for occurrences of $\star$. By substituting only for the leftmost occurrence of $\star$ we can make sure that the substitution of prooflike formulas in prooflike formulas results in a prooflike formula. Note that the leftmost $\star$ indicates the point where we are in the construction process.
There are two kinds of construction steps to which two kinds of substitution will correspond: first there will be the substitution of a complete formula for a $\star$, which actually makes the construct 'more complete'. But we will also need to substitute of more $\star$s for a $\star$ sometimes. These substitutions will represent decisions about the structure of the proof.
For example, each proof will start from scratch: we start with just one $\star$. Then we may decide that we want to make some assumptions. This will lead to substituting $(\star \to \star)$ for $\star$ to get $(\star \to \star)$. Now the first $\star$ indicates the point where we actually are and we see that this is indeed a point where assumptions can be made. If we now decide that in fact the assumption we want to make is a conjunction, we will replace the first $\star$ by $(\star \wedge \star)$, after which we can start filling in the first conjunct of the assumption. Filling in the first conjunct of our assumption, p say, will then be represented by $\star := p$, a substitution of the first kind to get $((p \wedge \star) \to \star)$.
Now defining the proofs system will simply amount to restricting the substitutions of the first kind in case the leftmost $\star$ is in a claim position in the prooflike formula. Of course we do not want any restrictions on the substitutions if we are not in a claim position, because we want to be able to make any kind of assumption. It is only the conclusions, which will be the in claim positions, that we have to be careful with. There we will want to check whether we can justify the claim, before we actually make it. And for this purpose we can simply use the techniques that were developed in the previous section.
The following definition makes this description of the proofs system precise. We will define 'being-a-proof' as a property of formulas in $\mathcal{L}^\star$, but the recursive definition of this property can be read as a set of instructions for building proofs.
In what follows we will use the notation $\langle \phi := \psi \rangle$ for a substitution of ψ for the leftmost occurrence of ϕ. If ϕ does not occur, the operation $\langle \phi := \psi \rangle$ has no effect. We will say $claim(\phi)$ if the leftmost $\star$ in ϕ is a claim of ϕ.

Definition 5.2 *We define Proof, the set of proofs, as the smallest set such that:*

1	*(start)*			$\star \in Proof$
2	*(construct* $\to$*)*	$\phi \in Proof$	$\Rightarrow$	$\phi\langle \star := (\star \to \star)\rangle \in Proof$
3	*(construct* $\wedge$*)*	$\phi \in Proof$	$\Rightarrow$	$\phi\langle \star := (\star \wedge \star)\rangle \in Proof$
4	*(assume)*	$\phi \in Proof$ & $\neg claim(\phi)$ *for any formula* $\psi \in \mathcal{L}$	$\Rightarrow$	$\phi\langle \star := \psi\rangle \in Proof$
5	*(direct proof)*	$\phi \in Proof$ & $claim(\phi)$ *if* $\psi \sqsubset_\phi \star$ *or* $\bot \sqsubset_\phi \star$	$\Rightarrow$	$\phi\langle \star := \psi\rangle \in Proof$

6 *(modus ponens)* $\phi \in \mathit{Proof}$ & $\mathit{claim}(\phi) \quad \Rightarrow \quad \phi\langle\star := \psi\rangle \in \mathit{Proof}$
if there is a ξ *with* $\phi\langle\star := \xi\rangle \in \mathit{Proof}$
and either:
$Cl_\xi(\psi)$ *and* $\phi\langle\star := \bigwedge\{\chi : \chi \sqsubset_\xi \psi\}\rangle \in \mathit{Proof}$
or:
$Cl_\xi(\bot)$ *and* $\phi\langle\star := \bigwedge\{\chi : \chi \sqsubset_\xi \bot\}\rangle \in \mathit{Proof}$

We see that we have 6 rules for building proofs. The first rule gets us started. Each construction starts with an application of that rule. It represents the stage where we have an empty page before us and decide that we want to build a proof.[12] The second and third rule are construction rules. They allow us to construct implications and conjunctions in any part of the proof. Rule 4 tells us how to make assumptions in a proof: we can fill in any assumption ψ at a place that is not a claim of the formula. The rules 5 and 6 are the real proof rules: they tell us which conclusions we may draw. By convention an occurrence of a claim $\star$ is always justified. So, given the discussion in the previous section, we simply could have replaced 5 and 6 by:

(conclude) $\phi \in \mathit{Proof}$ & $\mathit{claim}(\phi) \quad \Rightarrow \quad \phi\langle\star := \psi\rangle \in \mathit{Proof}$
and $\mathit{Just}(\psi, \phi\langle\star := \psi\rangle)$.

It is easy to check that this would give an equivalent proofs system. We have chosen for our formulation, because it makes the proof steps more explicit.

Note that in steps 4, 5 and 6 we allow ourselves to substitute complex formulas for some $\star$. This is not necessary for obtaining the right proof strength: if we only allow the substitution of atomic material in 4, 5 and 6, then the other rules will enable us to build up complex assumptions and conclusions step by step.[13]

There are two ways of making a comparison between the system of the previous section and this deduction system. One is by simply restricting the attention to the complete formulas in $\mathcal{L}^\star$. Then we can compare:

$\mathit{Jf}(\phi)$ and $\mathit{Proof}(\phi)$

for $\phi \in \mathcal{L} \cap \mathcal{L}^\star$.
But we can also extend the comparison to incomplete formulas. Then the comparison works by defining a completion operation on formulas of $\mathcal{L}^\star$. For example, let for each $\phi \in \mathcal{L}^\star$ the completion of ϕ, ϕ^c be defined by:

$$\phi^c = \phi\langle\star := \top\rangle^n.$$

Here $\top$ is some fixed tautology and n is the number of $\star$s in ϕ. So $\phi^c \in \mathcal{L}$. Then we can go on and compare:

[12] Usually in such a situation we also have a theorem in mind that we want to prove, but we cannot write this theorem down in our system. We always write the conclusions *after* the assumptions, so in our representation we will not see the theorem that we want to prove until the proof is completed.

[13] Here the Inversion Lemma (proved in the appendix) is essential. It guarantees that all formulas that are justified can be justified via their *atomic* claims.

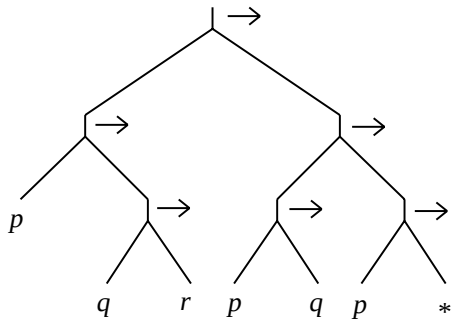
$Jf(\phi^c)$ and $Proof(\phi^c)$

for $\phi \in \mathcal{L}^\star$. Both approaches are essentially the same and in both cases it will be clear that the predicates *Proof* and *Jf* coincide. Therefore the soundness and completeness of this system is simply inherited from the soundness and completeness of $Jf(\phi)$.

5.2 Natural deduction

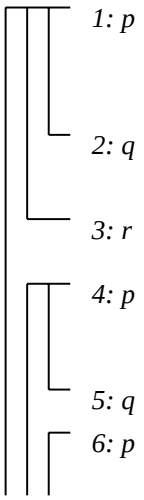
In this subsection we show how the building up of prooflike formulas can be represented in a linear style natural deduction system.[14] In such a system a proof looks like a column of formulas with vertical lines to its left—they indicate the scope of the assumptions—and annotation is added to the right of the column to indicate which rules have been applied.
Such a presentation for our proofs system can be obtained in several steps from the prooflike trees. The first step is to turn the tree in such a way that left to right order turns into top to bottom order. Now the atoms at the leafs of the tree appear in a (vertical) column and the atoms that used to occur to the left of some atom now can be found above the atom in the column.
The second step is to ignore the conjunction structure: the conjuncts are simply listed on consecutive rows of the column.
The third step is to replace the triangular shapes of the branches of the tree by squared braces, just as in linear natural deduction systems.
Then finally we delete the part of the tree where only $\star$s occur.

We give an example to illustrate the idea. First consider the proof $((p \to (q \to r)) \to ((p \to q) \to (p \to \star)))$. This is part of the proof of one of the axiom schemata from our Hilbert system for IL.

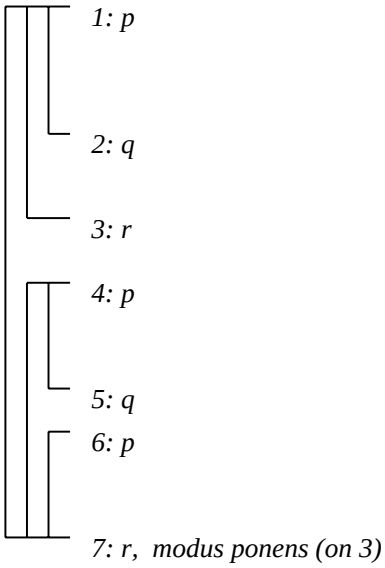


When we turn this around in the appropriate way we obtain:

[14] Cf. [?].



Let 's agree to write comments to the right of the claims to indicate which rule is used to derive it. Here there are no claims yet: all atoms have been obtained by the *assume*-rule, that allows us to add material in positions that are not claims. (In the picture we are not explicit about the *construct* rules that have been involved in the proof.)
We could now complete the proof of the axiom by applying modus ponens. Annotation is added to indicate which formulas are involved in the application of the modus ponens rule.



Of course the proof rules on the trees can be converted into proof rules on these natural deduction-like objects. For example, the basic structural notion of one formula preceding another, can be replaced by the notion of one formula being above another.
Thus we obtain a more familiar format for the proofs-as-texts system. Of course, from the

proofs as texts point of view, the horizontal, tree-like representation is preferable for those who tend to write their texts from left to right.

6 Conclusion

We can conclude that it is possible to give content to the slogan *proofs as texts*. Indeed in this paper we already have given an indication of what the slogan can lead to.
Our starting point has been the assumption that the formulas of our language represent texts. Since proofs can also be represented as texts, we concluded that in our situation both proofs and sentences should be represented in the same language. Once the distinction between an object language (for representing sentences) and a meta formalism (for representing proofs) is removed, we find that the only thing left to rely on is the structure of the formulas. In fact we are able to use the structure of the formula to formulate rules for building texts in such a way that validity is preserved, i.e. for building proofs. This is one of the points of the proofs as texts perspective: we model *theorem proving as text construction.*
So the main consequence of the *proofs as texts* perspective that we have been able to illustrate is how the information about the structure of a text comes to rescue once the distinction between an object level and a meta level is lost. We have also made an attempt to implement the fact that texts are built up incrementally. The way in which this has been done here — by introducing a new propositional constant $\star$ — is not very elegant. We have proposed another strategy to deal with incomplete formulas elsewhere ([**?**] [**?**]), but have refrained from implementing this strategy here, since this would take up too much space.
Other properties that are typical of texts have not yet been implemented. In particular we have postponed the discussion of the use of anaphora in proofs by restricting ourselves to a propositional language. So the obvious next step in our programme is the extension of the approach to predicate logic. Here we could either try to give a system for (static) predicate logic, or we could aim to give a proof theory for *Dynamic Predicate Logic*, a variant of predicate logic developed especially to deal with the semantics of anaphora. In both cases we will have extra labels for the quantifiers. The idea that we would like to pursue is that nodes that carry these labels systematically regulate the use of the material that we find at the nodes in their environment. For example, if a node is labelled with $\exists x$, then it will not allow us to use all information with a variable x freely. The nodes labelled with $\exists x$ tell us which nodes that have information about x can be used at some point in the text. This will depend on the relative positions of the $\exists x$ node and the node with information information about x. The difference between ordinary predicate logic and dynamic predicate logic should then turn out to be a difference in the way in which the quantifier nodes regulate the use of information at other nodes. We realise that this gives only a very rough indication of how the extension of the system to predicate logic should proceed, but it will have to do for now. We hope to discuss it in more detail in another paper.
Other points that have not been dealt with here are the treatment of other connectives, such as disjunction, the proper treatment of intermediate conclusions, situations where conclusions precede assumptions, etc. For some of these issues it will be easy to find a treatment in our system, others will require more work. For example, if we include new connectives in our language, then this can seriously complicate the structure of the formulas. As a consequence the definition of the support relation will become more complicated.
But as it stands there is no reason to suspect that any of the topics mentioned will lead to problems that cannot be solved within the *proofs as texts* perspective.

Acknowledgements

The motivation for the *proofs as texts* perspective was developed in collaboration with Albert Visser. Earlier attempts to give a deduction system in this style were made in [?] and by Lysbeth Zeinstra [?]. The main improvement in comparison with those attempts lies in the use of the techniques from [?] and [?], for which we thank Marcus Kracht. Developments in the area of labelled deductive systems [?] were also an encouragement for our own attempts to develop a new way of looking at proofs.
I would like to thank Jan van Eijck for comments on an earlier draft, which led to a simplification of some of the crucial definitions.

A Appendix: a little proof theory

In this appendix we give some properties of the system that was defined in section 3. The properties will discuss have natural analogues in intuitionistic proof theory and will lead up to a decidability result:[15] we will be able to give a decision procedure to find out whether $Jf(\phi)$ for any $\phi \in \mathcal{L}$.
At this point the development of the proof theory of our system is in a stage where we reproduce familiar results about intuitionistic logic in our setting. But we hope that in time our new approach to the proof theory of IL will lead to new theorems about IL or will lead to considerable improvements of the proofs of some familiar theorems.

A.1 Inversion Lemma

In this section we show that two of our proof rules, conjunction of proofs and proof of an implication, can be inverted. As it stands these rules allow us to build up justifications of complex claims from justifications of smaller claims. Here we show the converse: if we have a justification for some claim, then we can construct from it a justification for all its subclaims.

Lemma A.1 (Inversion Lemma) *Let $\psi_i \leq (\psi_1 \circ \psi_2) \leq \phi$ be given ($i = 1, 2$, $\circ = \wedge, \rightarrow$). Then:*

- $Just((\psi_1 \wedge \psi_2), \phi)$ *iff* $Jble(\psi_i, \phi)$ $(i = 1, 2)$
- $Just((\psi_1 \rightarrow \psi_2), \phi)$ *iff* $Jble(\psi_2, \phi)$

Proof:
$\Leftarrow$: This is given by the rules conjunction of proofs and proof of an implication.
$\Rightarrow$: Assume that $Just((\psi_1 \wedge \psi_2), \phi)$ has been established. We distinguish the following cases according to the last rule that has been applied in this justification:

- (direct proof) Either $(\psi_1 \wedge \psi_2) \sqsubset_\phi (\psi_1 \wedge \psi_2)$ or $\bot \sqsubset_\phi (\psi_1 \wedge \psi_2)$. In the first case we see that $\psi_i \sqsubset_\phi \psi_i$, so we have a direct proof of each ψ_i. In the second case we see that $\bot \sqsubset_\phi \psi_i$, so again we have a direct proof of ψ_i.
- (conjunction of proofs) Then we knew before the last step in the justification that: $Just(\psi_i, \phi)$. So we are done.

[15] Cf. for example, [?] for discussion of such results.

- ▷ (proof of an implication) Does not apply.
- ▷ (generalised modus ponens) Let ξ be the major premise. There are two cases: $Cl_\xi(\psi_1 \wedge \psi_2)$ or $Cl_\xi(\bot)$. We see that ξ also allows for the conclusion of each of the ψ_i (since either $Cl_\xi(\psi_i)$ or $Cl_\xi(\bot)$), so the same application of generalised modus ponens also justifies the ψ_i.

Next assume $Just((\psi_1 \to \psi_2), \phi)$. We distinguish the following cases:

- ▷ (direct proof) Either $(\psi_1 \to \psi_2) \sqsubset_\psi (\psi_1 \to \psi_2)$ or $\bot \sqsubset_\psi (\psi_1 \to \psi_2)$. In the first case we apply generalised modus ponens with major premise $(\psi_1 \to \psi_2)$. We know (via the hypothesis) that this major premise is justified and we see that ψ_1 is justified by direct proof. So this gives a justification of ψ_2.

  In the second case we see that $\bot \sqsubset_\psi \psi_2$, so in this case we have a direct proof of ψ_i.
- ▷ (conjunction of proofs) Does not apply.
- ▷ (proof of an implication) Now we know that $Just(\psi_2, \phi)$. So we are done.
- ▷ (generalised modus ponens) As above.

□

The Inversion Lemma is interesting by its own right, because it shows that there is a kind of symmetry in our system. But the main reason why we are interested in the Inversion Lemma is that it gives rise to the following corollary.

Corollary A.2 *A formula can be justified iff all its atomic claims can be justified.*

This is the first step on our way to a decision procedure: we now know that to decide whether a claim can be justified it suffices to check whether all its atomic subclaims can be justified.

A.2 Subformula Property

In this subsection we prove the *Subformula Property* for our system. The subformula property guarantees that a justification of some claim in a formula can be built up from justifications of other parts of the same formula. So we do not have to leave a formula/text to justify it.
For most proof rules this is clear: the rule for direct proof does not require other justifications at all and the rules for conjunction of proofs and proof of an implication typically rely on justifications of subclaims of the current claim. The crucial rule is generalised modus ponens: here we use justifications for the major premise and for some subformulas of the major premise, but there is no restriction on which formulas we can use as major premise. So for the proof of the subformula property will have to show that applications of generalised modus ponens where the major premise is a subformula suffice to justify all justifiable claims. In fact we will prove something slightly stronger: we will show that the ξ_3 that we use in generalised modus ponens can always be chosen from the formulas that support the current claim. Justifications in which only such applications of generalised modus ponens occur will be called *non-alien*. Other justifications are called *alien*.

In the proof we will frequently use a slightly stronger version of the Inversion Lemma: we will use that a non-alien justification of a claim exists iff non-alien justifications of its subclaims exist. This follows immediately from the proof of the Inversion Lemma as we have given it. So we state without further proof:

Lemma A.3 (Strong Inversion Lemma)

- ▷ *There exists a non-alien justification of* $Just((\psi_1 \wedge \psi_2), \phi)$ *iff there exist non-alien justifications of* $Just(\psi_i, \phi)$ $(i = 1, 2)$
- ▷ *There exists a non-alien justification of* $Just((\psi_1 \to \psi_2), \phi)$ *iff there exist non-alien justifications of* $Just(\psi_i, \phi)$ $(i = 1, 2)$

Proposition A.4 (Subformula Property) *If* $Jble(\psi, \phi)$*, then there exists a non-alien justification of* ψ *in* ϕ*.*

Proof:
We prove the statement by showing that the deepest alien application of gmp (generalised modus ponens) can be eliminated. Since there can only be finitely many alien applications in a justification, this shows that all alien justifications can be eliminated.
So we are in a situation where $Just(\psi, \phi)$ has been established by alien generalised modus ponens, but no other, deeper alien application of modus ponens is required. By the Strong Inversion Lemma we may assume $\psi = p$. Then we have a major premise ξ such that:

- ▷ $Cl_\xi(p)$ (or $Cl_\xi(\bot)$),
- ▷ $Just(\xi, \phi[p := \xi])$,
- ▷ $Just(\bigwedge\{\chi : \chi \sqsubset_\xi p\}, \phi[p := \bigwedge\{\chi;\ \chi \sqsubset_\xi p\}])$
  (or $Just(\xi_1, \bigwedge\{\chi : \chi \sqsubset_\xi \bot\}, \phi[p := \bigwedge\{\chi : \chi \sqsubset_\xi \bot\}])$).

Here the last two statements can be verified without alien generalised modus ponens.
In what follows we will ignore the case $Cl_\xi(\bot)$, but it is easy to extend the proof to include this case.
Since $Cl_\xi(p)$, also $Cl_{\phi[p:=\xi]}(p)$. The strong inversion lemma (applied to $Just(\xi, \phi[p := \xi])$) gives a justification of this claim: $Just(p, \phi[p := \xi])$.
Now we can consider this justification of p: how does it work?

1. It was obtained by direct proof. So $p \sqsubset_{\phi[p:=\xi]} p$. We distinguish two cases: the supporting occurrence of p occurs in ϕ or the supporting occurrence occurs in ξ. In the first case we find an alternative justification for p in ϕ (namely by direct proof) that does not uses alien rule applications.

   In the second case we also find such an alternative. For then $p \in \{\chi : \chi \sqsubset_\xi p\}$ and by assumption there is a justification of p in ϕ that does not use alien rule applications.

2. Conjunction of proofs or proof of an implication do not apply, since p is atomic.

3. We used non-alien generalised modus ponens.

We see that we can eliminate the alien gmp unless it is followed by a non-alien application of gmp. So it suffices to show that such non-alien gmps can be eliminated. We will do this by replacing the two gmps by one big gmp. We see to it that this big gmp will again be the deepest alien gmp in the resulting justification. This will prove that case 3 can be avoided. So only case 1 remains and we can eliminate the alien gmp.
Since we are past the deepest alien gmp, the next application of generalised modus ponens must be non-alien. So the major premise ζ is such that:

- $\triangleright$ $Cl_\zeta(p)$
- $\triangleright$ $\zeta \sqsubset_{\phi[p:=\xi]} p$,
- $\triangleright$ $Just(\chi_1 \wedge \ldots \wedge \chi_n, (\phi[p := \xi])[p := \chi_1 \wedge \ldots \wedge \chi_n])$
  where $\chi_1 \wedge \ldots \wedge \chi_n = \bigwedge\{\chi : \chi \sqsubset_\zeta p\}$

Now we construct a big gmp that does on its own what ξ and ζ do together.

By the strong inversion lemma we get a non-alien justification of each of the $\xi[p := \chi_i]$ in $\phi[p := \xi[p := \chi_i]]$. We also have the non-alien justification of $\bigwedge\{\chi : \chi \sqsubset_\xi p\}$ in $\phi[p := \bigwedge\{\chi : \chi \sqsubset_\xi p\}]$ in the big gmp. From these we can construct a non-alien justification of $\xi[p := \chi_1] \wedge \ldots \wedge \xi[p := \chi_n] \wedge \bigwedge\{\chi : \chi \sqsubset_\xi p\}$ in:

$$\phi[p := (\xi[p := \chi_1] \wedge \ldots \wedge \xi[p := \chi_n] \wedge \bigwedge\{\chi : \chi \sqsubset_\xi p\})].$$

By hypothesis we also get a non-alien justification of $\xi[p := \zeta]$ in $\phi[p := \xi[p := \zeta]]$. Now if we replace each of the χ_i in this claim by $\xi[p := \chi_i]$, we will be able to make a non-alien justification of this claim too. This construction is similar to the construction of a justification of $(\alpha \to ((\alpha \to \beta) \to \gamma))$ from a justification of $(\alpha \to (\beta \to \gamma))$.
So we get a non-alien justification of the claim

$$\xi[p := (\zeta[\chi_1 := \xi[p := \chi_1]] \ldots [\chi_n := \xi[p := \chi_n]])]$$

in $\phi[p := \xi[p := (\zeta[\chi_1 := \xi[p := \chi_1]] \ldots [\chi_n := \xi[p := \chi_n]])]]$.
This is the major premise of the big (alien) gmp that we are looking for. Note that we have made sure that this alien gmp is again a deepest alien gmp in the resulting justification.
So we can eliminate each deepest alien gmp. We may conclude that for all claims ψ a justification without applications of alien generalised modus ponens is available.
□

The proof of the *Subformula Property* is the second big step on our way to finding a decision procedure. From the Inversion Lemma we know that we only have to consider atomic claims. Now we know that in the justification of these claims no alien applications of modus ponens have to be considered.

A.3 Decidability

At this point we can see that the following three kinds of steps will suffice to find a justification of a claim, if there is one.

1. If the claim we aim to justify is complex, we may break it down into atomic claims.

2. Then we check whether there is a direct justification for these atomic claims.

3. If not, then we try to apply generalised modus ponens with some supporting formula that has the same atomic claim, or that has $\perp$ as a claim.

The three steps give a search procedure for justifications, but as it stands it is not guaranteed that this procedure will terminate. If we are not careful we might end up trying to apply generalised modus ponens infinitely many times.
Consider, for example, $\phi = (((p \to q) \to q) \to (p \to q))$. If we start looking for a justification for this formula, we will, after some steps in the procedure, have the following goal:

$$Just(q, \phi).$$

At this point the only option we have is to use generalised modus ponens to justify q, i.e. we perform step 3. Then our new goal is:

$$Just((p \to q), (((p \to q) \to q) \to (p \to (p \to q)))).$$

Then, again after some steps in the procedure, we have to justify:

$$Just(q, (((p \to q) \to q) \to (p \to (p \to q)))).$$

And again there is no other option then generalised modus ponens. If we do not check for loops now, we can go on like this for ever, always substituting $(p \to q)$ for q. The following lemma will help us to prevent such infinite searches.

Lemma A.5 (Finiteness Lemma) *If in a branch of the search procedure where we try to justify some claim ψ with supporting material $\{\chi_1, \ldots, \chi_n\}$ we arrive at a position where we have to justify ψ again with supporting material $\{\chi_1, \ldots, \chi_n\}$, then we may close this branch of the search tree.*

Proof:
At each point in the search procedure the search after that point is completely determined by the claim that we are trying to justify at that point and the supporting material that is available at that point. So the search tree after the first point where we try to justify ψ from $\{\chi_1, \ldots, \chi_n\}$ is isomorphic to the search tree after the second point where we try to justify ψ from $\{\chi_1, \ldots, \chi_n\}$. Therefore if we are looking for a finite branch in the search tree we need not look beyond the second point where we try to justify ψ from $\{\chi_1, \ldots, \chi_n\}$: any finite branch that exists there can also be found in the isomorphic tree after the first point where we try to justify ψ from $\{\chi_1, \ldots, \chi_n\}$. $\Box$

This lemma will give us the possibility of cutting of all infinite branches of the search tree. To be precise:

Corollary A.6 *If we are on an infinite branch in the search procedure, then this branch can be dismissed after finitely many steps.*

Proof:
Suppose that we find an infinite branch in the tree. Since we only have finitely many (atomic) subformulas in the original formula and each claim that we meet in the search is such a subformula, we can be sure that the infinite branch contains infinitely many attempts to prove the same claim, ψ say. Since all the supporting material that we find during the search will consist of subformulas of the original formula, the set of supporting material can only be one of finitely many sets. Therefore it is clear that at two points in the infinite branch we will try to prove ψ from the same supporting material, $\{\chi_1, \ldots, \chi_n\}$ say. At this point we can dismiss this branch according to the previous lemma. □

So the condition of the lemma allows us to eliminate infinite search paths. From this we may conclude:

Proposition A.7 *$Jf(\phi)$ is decidable.* □

Note that this does indeed eliminate the counterexample that we considered above.
As a last illustration we show how the decision procedure works for Peirce's Law.

Example:
We apply the procedure to $(((p \to q) \to p) \to p)$, an instance of Peirce's Law. By step one of the procedure we try to find a justification for the atomic claim p. Since this claim has no direct proof (step 2), we have to apply generalised modus ponens to some supporting formula with claim p or $\bot$. There is one such formula, $((p \to q) \to p)$. We have to justify all the formulas that support p in this formula, i.e $(p \to q)$. So we now have as a goal to justify: $(((p \to q) \to p) \to (p \to q))$.
Now we apply step one again and try to find a justification for the atomic claim q in this formula. Since there is no direct proof for q (step 2), we try to perform step 3 again. But now we see that the procedure ends, since there is no supporting formula that has q or $\bot$ as a claim. We may conclude that Peirce's Law is not a theorem of intuitionistic logic.