Hiproofs

Ewen Denney[1], John Power[2], and Konstantinos Tourlas[2*]

[1] NASA Ames Research Center, QSS Group Inc, USA
~~[2] Division of Informatics, The University of Edinburgh, SCOTLAND~~

Abstract. We introduce a hierarchical notion of formal proof, useful in the implementation of theorem provers, which we call *hiproofs.* Two alternative definitions are given, motivated by existing notations used in theorem proving research. We define transformations between these two forms of hiproof, develop notions of underlying proof, and give a suitable definition of refinement in order to model incremental proof development. We show that our transformations preserve both underlying proofs and refinement. The relationship of our theory to existing theorem proving systems is discussed, as is its future extension.

Keywords. tactic-based theorem proving, proof representation, hierarchy, proof planning, diagrams, higher dimensional graphs, hiproofs

1 Introduction

The activity of *tactical theorem proving* is, inherently, a hierarchically organised one. Its hierarchical structure is inherent both in the way composite tactics are defined to consist of other tactics, and in the way that tactics are then sequentially applied to successions of goals.

Accordingly, a number of proof assistants offer direct support for some notion of hierarchy in their data structure and language of tactics; see, for example, [CS00,RSG98,KNM94]. What has been lacking, however, is a sufficiently conceptual and abstract characterisation of hierarchy in mechanised proofs which, by being independent of implementation details, would in the long run underpin a larger theory to support the art of tool building. In particular, we believe this to be useful for developing interfaces: both graphical interfaces for individual theorem provers, and interfaces between theorem provers.

A first approach at such a characterisation, which is both mathematically precise and lends itself to diagrammatic visualisation, is the aim of this paper.

Consider, for example, how Fig. 1(a) could be understood as depicting a proof by induction. In this representation we emphasise the tactics rather the original goal and subsequent proof states, and so the nodes in the diagram are labelled by tactic identifiers. Inclusion of one node in another indicates a subtactic relationship, and the arrow indicates sequential composition: the tactic at the

* This work has been supported by EPSRC grants nos. GR/M45030 (Ewen Denney) M566333 (John Power) and GR/N12480 (Konstantinos Tourlas). The first author did most of this work while in Edinburgh.

source of the arrow is invoked first, then the tactic at the target. The diagram is then read as saying that, at the most abstract level, the proof is obtained by invoking an induction tactic, `Induction`. This consists of applying an induction rule, `Ind-Rule`, which then generates two subgoals. The first subgoal is handled by the `Base` tactic, the second by the `Step` tactic. In turn, `Step` is defined as first applying the `Rewrite` tactic, and then the `Use-Hyp` tactic. For the purposes of this example, we regard `Base`, `Rewrite` and `Use-Hyp` as primitive.

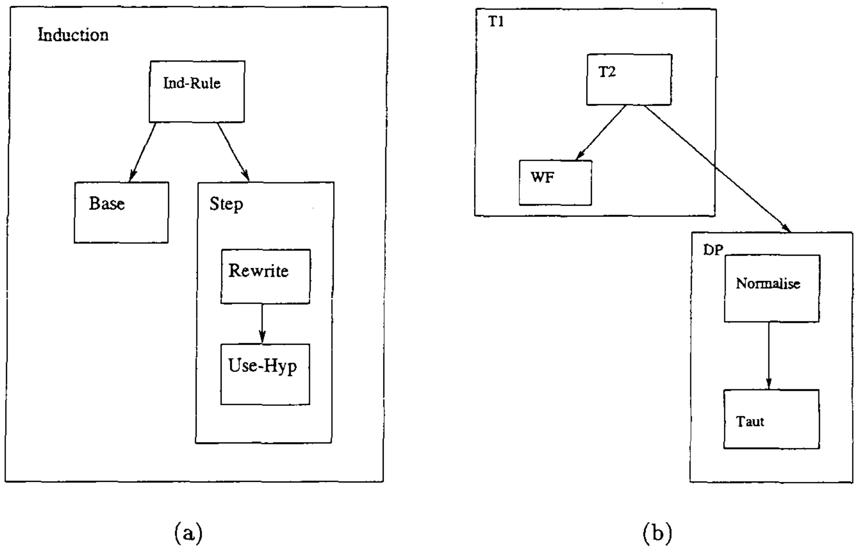

Fig. 1. Two Hierarchical Proofs

A slightly more complex example is given in Fig. 1(b). At the most abstract level, the proof consists of applying `T1`, and then `DP`. The tactic `T1` first applies `T2`, generating two subgoals, the first of which is handled by `WF`. The second is handled by `DP`, which applies `Normalise` and then `Taut`.

We will refer to these hierarchical representations of proofs as *hiproofs*. Obvious questions to ask are: what are the acceptable well-formed hiproofs? what are their mathematical properties? what operations can we perform on these structures? what is the relationship between such a hiproof and the underlying logic? Indeed, such diagrams are not the only possible hierarchical representations for proofs, and we should ask if there are any substantial differences between the different approaches. We will address most of these questions in this paper, giving formal definitions for two different notions of hiproof, and proving an equivalence theorem.

Before doing so, however, it is instructive to take one closer look at the relationship between hiproofs, tactics and standard notions of formal proof, such as proofs in natural deduction style.

Example 1. Consider a natural deduction proof of $A \Rightarrow A \wedge (x = x)$, as in Fig. 2. The obvious (backwards) proof is implication introduction, followed by conjunction introduction, and then applying axiom and reflexivity to the two subgoals. The essential information of the proof is the sequence of inference rules

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\texttt{Ax} \qquad \dfrac{}{A \vdash x = x}\,\texttt{Refl}}{A \vdash A \wedge (x = x)}\,\texttt{And-I}}{A \Rightarrow A \wedge (x = x)}\,\texttt{Imp-I}$$

Fig. 2. A simple natural deduction proof.

which are applied, and we can represent the order of these rules as a tree, like in Fig. 3(a). Typically, however, theorem provers allow the use of higher-level tactics

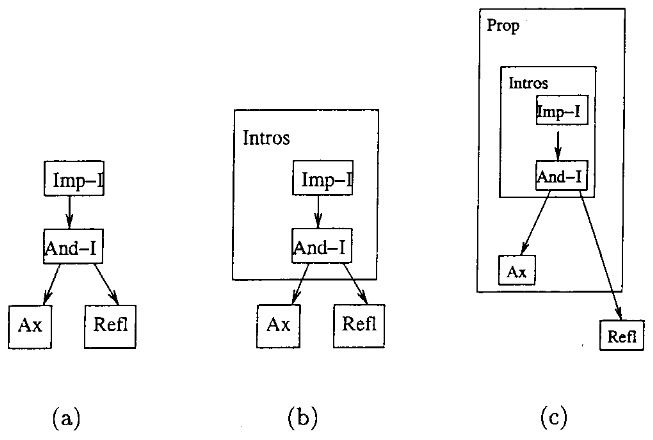

Fig. 3. Introducing hierarchy in proof diagrams by grouping.

which group together the application of a number of low-level inferences. For example, it is common to have an `Intros` command, which performs all possible introduction rules. We can indicate this on the proof diagram by grouping `Imp-I` and `And-I` together, as in Fig. 3(b). We could go further and define a tactic, `Prop`, which first calls `Intros`, and then tries to use axioms wherever possible. This gives the hierarchical structure shown in Fig. 3(c). □

Compared to the hierarchical structures which are implemented in state-of-the-art theorem provers, our hiproofs abstract away from some of the more concrete and operational aspects of those structures. These key abstractions, made necessary so as to arrive at a unifying and implementation-independent notion, are the following:

- ~~In hiproofs, we only model tactics, not goals. This is firstly because a notion~~ of hierarchy, *per se*, should ideally be independent of the underlying logic. Secondly, tactics themselves possess a rich structure, and it is our intention to find the simplest possible framework in which we can study this. This framework may then be used to assist the development of complex tactics in a top-down, hierarchical fashion.
- In our framework, a tactic is thought of as a black box, and so none of its implementation details are listed, except to record which other tactics it may be defined in terms of. In general, we also regard inference rules and axioms as primitive tactics.
- Hiproofs model static structure, not dynamics. Thus our diagrams represent only the sequence of tactic applications which led to a successful proof, and do not indicate the tactic definitions themselves or give any information about proof search.
- Hiproofs are structures on proofs, not on particular implementations of proofs. Thus, we have begun by regarding the underlying proofs to be trees, as is standard in the study of formal logic, whereas some systems implement proofs as dags rather than trees, and some allow and/or trees to model proof search. Nothing in our treatment is specific to the particular notion of basic proof we have taken, let alone specific to any implementation technology for proofs.

We now proceed to explain how the the key features of hiproofs can be intuitively understood in terms of the features of the proof diagrams in Figs. 1(a) and 1(b). (Although we will be motivated by the diagrams, we make a decision to take non-graphical definitions as basic, and so, abstract away from various graphical representations to the underlying mathematical structure.) To this end, we make the following observations:

- First, there is no requirement for uniqueness of tactic identifiers, since a tactic can certainly be applied repeatedly in a proof. However, we will informally refer to proof nodes by their tactic identifier where there is no ambiguity.
- There are only two relationships which can hold between nodes. In the diagrams, *inclusion* indicates the unfolding of a tactic into its definition, while the arrows indicate sequential *composition*. For example, in Fig. 1(b), the decision procedure, `DP`, unfolds to give the composition of `Normalise` with `Taut`.
- Like Figs. 1(a) and 1(b), hiproofs are essentially tree-like, that is, subgoals are *independent*: a tactic acts on a single subgoal[3]. Normally, tactics are

[3] Of course, in general, this is not the case, and such extensions will be the subject of future work.

thought of as returning a *list* of subgoals. Here, though, we abstract away from the order and do not place any order on child tactics.

A hiproof, therefore, consists of a finite collection of tactic-labelled nodes, related by inclusion and composition. Although the diagrams represent abstract versions of *full* proofs, we will be interested in how such proofs are built up, so ~~also consider intuitively *partial* proofs to be well-formed.~~

Related Work The use of diagrams in logic is not new, and is discussed, in general, by [BH96]. An important point, here, is that hierarchical structure can be put on a basic proof in different ways. Two hiproofs can be quite different, yet have the same underlying proof. This is in contrast to Fitch-style boxed natural deduction proofs, where there is only one way to draw the boxes on a given proof.

Hierarchical proofs, such as we consider here, are popular with the proof planning [Bun96] community. For example, [RB99] uses two different representations. There has not been much analysis of the algebraic features of proofs, however. An exception is [RS01], which concentrates on the dynamics of a particular representation language. Our concern, here, is with static structural properties.

Bearing the above general differences in mind, we now take a look at how hiproofs generalise the structures found in some of the most popular systems. In general, any system (such as Coq, HOL and PVS) in which tactics may be defined from other tactics leads naturally to the kind of hierarchical proof which is central to our theory.

More specifically, the notion of hierarchy which we have formalised here is similar to that underlying the Lambda Clam family of proof planners [RSG98]. Yet it is more general, since Lambda Clam does not (currently) allow tactics to leave open goals, as we do in Fig. 1(b), for example. The Tecton system [KNM94] also supports hierarchical proof, though its hierarchy is only 'two level' and not arbitrarily nested like here. The PDS data structure [CS00], on the other hand, implemented in Omega [BCF$^+$97] and other systems, is of similar generality to our hiproofs, but is less abstract. A PDS consists of *names*, *sequents*, and elements called *justifications* and *reasons*. Of those only named nodes and justification elements have counterparts in hiproofs. Sequents and reasons implement goals and back-tracking, and so have no counterparts in hiproofs. Otherwise, a PDS (one which, moreover, happens to be a tree at the lowest level) corresponds very closely to the second variant of hiproof we develop.

The paper is organised as follows. The following section develops two definitions of hiproof, assuming the reader to be broadly familiar with trees and forests. (The necessary graph and order theory is summarised in an appendix along with some detailed technical proofs.) Section 3 defines transformations between our two notions of hiproof, and shows their equivalence. We also introduce a notion of *underlying proof* which this equivalence respects. Section 4 defines refinement of hiproofs and proves that the transformations are refinement preserving. Finally, Section 5 concludes and discusses how this theory can be developed further.

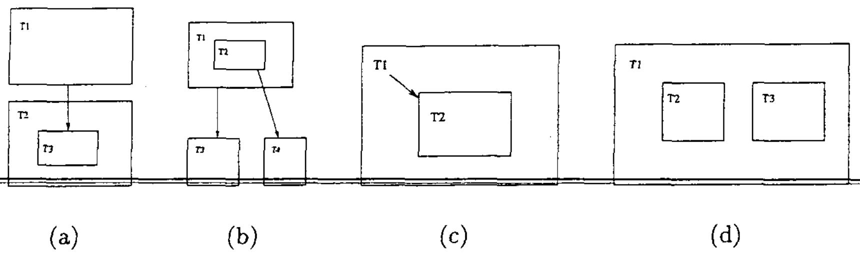

Fig. 4. Four non-proofs

2 Two definitions of hiproof structure

In this section two definitions of hiproof structure are presented, each corresponding to a different kind of existing graphical representation. In the next section, they will be shown to be equivalent to each other. To enhance the conceptual clarity of our definitions we will conveniently regard certain trees and forests (i.e. sets of trees) as partially ordered sets (posets) and vice versa. The formal details on which this interchange depends are, for reasons of space and conciseness of exposition, included in an appendix.

First, we fix a non-empty set Λ of *tactic identifiers* (or *method identifiers*). Based on the intuition for tactics outlined in the previous section, we now give an analysis of why the diagrams of Fig. 4 are not examples of valid hiproofs.

Operationally, one tactic is followed by another, which unfolds to give another tactic, and so on. Thus we might say that tactics are invoked 'at the most abstract level'. The first diagram is invalid under this interpretation, then, because if T1 is followed by T3 and T2 unfolds to T3 then the more abstract T2 should follow T1. Another way of putting it is that it would be permissible for T3 to follow T1, but then the fact that T2 is an abstraction of T3 is irrelevant to this proof and should not be added after the composition of T1 and T3.

Conversely, when a tactic finishes executing, control flows from the most recently executed tactic, i.e. the innermost, outwards, but the second diagram does not follow this convention.

The third diagram is also invalid: either T1 unfolds to T2 or T2 follows sequentially, but not both.

Finally, the fourth example fails as well, for the reason that tactic T1 must unfold to give a unique subsequent tactic to execute, not two. To make this precise we say that T2 and T3 are *siblings* under T1, a notion which we generalise from trees to forests (i.e. sets of trees):

Definition 1. *Let F be a forest, considered equivalently as a graph $\langle V, \rightarrow\rangle$ or set $\{T_1, \ldots, T_n\}$ of trees. The predicate $isroot_F(v)$ (also written $isroot_\rightarrow$ when the context of $\rightarrow$ is understood to be F) is defined to hold if and only if there is tree $T_j = \langle V_j, \rightarrow_j, r_j\rangle$ in F such that $v = r_j$; or, equivalently, when no v_0*

exists such that $v_0 \dashrightarrow v$. Further, we say v and v' are siblings in F, and write $siblings_F(v, v')$ (or $siblings_{\dashrightarrow}(v, v')$ when the context of $\dashrightarrow$ is understood to be F), if v and v' have the same parent (i.e. $\exists v''.\ v'' \dashrightarrow v$ and $v'' \dashrightarrow v'$), or are both roots. □

Notice, in particular, that every node v is trivially a sibling of itself.

~~Our discussion now leads to the following definition which, in accordance~~ with pictorial intuition, emphasises the reflexive, antisymmetric and transitive nature of spatial inclusion as a partial order, while using a graph structure for sequential composition:

Definition 2. *A* hiproof of type 1 *consists of a forest* qua poset $i = \langle V, \leq_{\mathrm{i}} \rangle$ *and a forest $s = \langle V, \dashrightarrow_{\mathrm{s}} \rangle$, together with a function $t : V \to \Lambda$ which labels the nodes in V with tactic identifiers in Λ. (The relations $\leq_{\mathrm{i}}$ and $\dashrightarrow_{\mathrm{s}}$ correspond to the* inclusion *and* sequentiality *relations among tactics.) These data are subject to the following additional conditions:*

1. *Inclusion and sequence are mutually exclusive: whenever $v \leq_{\mathrm{i}} w$ and ($v \dashrightarrow^{\star}_{\mathrm{s}} w$ or $w \dashrightarrow^{\star}_{\mathrm{s}} v$) then $v = w$.*
2. *Arrows always target outer nodes: whenever $v \dashrightarrow_{\mathrm{s}} w_1$ and $w_1 <_{\mathrm{i}} w_2$ then $v <_{\mathrm{i}} w_2$; and*
3. *Arrows always emanate from inner nodes: whenever $w_1 \leq_{\mathrm{i}} v$ and $v \dashrightarrow_{\mathrm{s}} w_2$ then $v = w_1$; and*
4. *Given any two nodes v and v' which both lie at the top inclusion level, or are both immediately included in the same node, then at most one of v, v' has no incoming $\dashrightarrow_{\mathrm{s}}$ edge:*

$$\forall v, v' \in V.\ siblings_i(v, v') \wedge isroot_s(v) \wedge isroot_s(v') \implies v = v' \quad \square$$

The first condition, equivalently written $v \leq_{\mathrm{i}} w \wedge v \neq w \implies v \not\dashrightarrow^{\star} w \wedge w \not\dashrightarrow^{\star}_{\mathrm{s}} v$, precludes non-proof examples such as that of Fig. 4(c). On the other hand, the fourth condition is designed to preclude the situation of Fig 4(d), as well as the similar non-proof example obtained from Fig. 4(d) after removing the enclosing node labelled `T1`. The second condition prohibits that the inclusion hierarchy is 'downwards' transcended by sequentiality (e.g. as in Fig. 4(a)), but still permits the situation of Fig. 1(b), where the arrow from the node labelled `T2` to the one labelled `DP` 'upwards' transcends the inclusion hierarchy. A further interesting consequence, proved in the appendix, is the following:

Lemma 1. *Let $\langle V, \leq_{\mathrm{i}}, \dashrightarrow_{\mathrm{s}}, t \rangle$ be a hiproof of type 1. Then: (1) There is a unique node $v_0 \in V$ which is both* maximal *wrt. $\leq_{\mathrm{i}}$ and has no incoming $\dashrightarrow_{\mathrm{s}}$ edge; i.e. there is a unique v_0 such that $isroot_{\leq_{\mathrm{i}}}(v_0)$ and $isroot_{\dashrightarrow_{\mathrm{s}}}(v_0)$; and (2) $v \dashrightarrow_{\mathrm{s}} w_1$ and $w_1 \in cover_{\leq_{\mathrm{i}}}(w_2) \implies v \in cover_{\leq_{\mathrm{i}}}(w_2)$ is equivalent to condition 2 in Def. 2.* □

Let $\mathbf{Hiproof}_1$ denote the set of hiproof structures of type 1 arising from Definition 2 above.

Example 2. The example from Fig. 1(b) forms a hiproof of type 1 like so:

$$\begin{tikzcd}
\mathrm{T1} \arrow[d, "i"'] \arrow[dr, "i"] & & \mathrm{DP} \arrow[d, "i"'] \arrow[dr, "i"] & & \mathrm{T2} \arrow[d, "s"'] \arrow[dr, "s"] & & \mathrm{Normalise} \arrow[d, "s"'] \\
\mathrm{T2} & \mathrm{WF} & \mathrm{Normalise} & \mathrm{Taut} & \mathrm{WF} & \mathrm{DP} & \mathrm{Taut}
\end{tikzcd}$$

~~The labels on the arrows distinguish between the inclusion and sequential com-~~ position forests. □

Proposition 1. *No 'composite cycles' exist in a hiproof of type 1: writing $v <_{\mathrm{i}}^{1} v'$ whenever $v \in \mathit{cover}_{\leq_{\mathrm{i}}}(v')$, then for all $v, v' \in V$, whenever $v\,(>_{\mathrm{i}}^{1} \cup \dashrightarrow_{\mathrm{s}})^{+}\,v'$ one has $v \neq v'$ (here, R^{+} denotes the transitive closure of a binary relation).*

Proof. (Sketch) *$v\,(>_{\mathrm{i}} \cup \dashrightarrow_{\mathrm{s}})^{+}\,v'$ means there exists non-empty sequence $v_0 = v, v_1, \ldots, v_n = v'$ such that $v_i >_{\mathrm{i}} v_{i+1}$, or $v_i \dashrightarrow_{\mathrm{s}} v_{i+1}$. The proof is by an easy induction on the length n of the sequence: in the the base case $n = 1$, $v = v_0 \neq v_1 = v'$ trivially follows from the inequality and also in the case of $v \dashrightarrow_{\mathrm{s}} v'$ as $\langle V, \dashrightarrow_{\mathrm{s}}\rangle$ is a forest (and therefore contains no $\dashrightarrow_{\mathrm{s}}$ cycles). The inductive hypothesis yields $v_0 \neq v_{n-1}$, in addition to which $v_{n-1} \neq v_n$ is easily obtained by an argument similar to that of the base case.* □

The definition of hiproofs of type 1 consists of two posets. Because of Prop. 1, it is possible to consolidate this in one single tree. As a first attempt at such a definition, we might try to represent Fig. 1(b) like so:

$$\begin{tikzcd}
& \mathrm{T1} \arrow[d, dashed] & \\
& \mathrm{T2} \arrow[dl] \arrow[dr] & \\
\mathrm{WF} & & \mathrm{DP} \arrow[d, dashed] \\
& & \mathrm{Normalise} \arrow[d] \\
& & \mathrm{Taut}
\end{tikzcd}$$

A similar notation is used in [RB99]. The solid lines denote composition and the dashed lines inclusion. The problem with this representation is that it does not distinguish the similar proof where `DP` is actually a subtactic of `T1`. This can be made explicit by pairing tactics with their level in the inclusion hierarchy; hence, `T1` and `DP` have level 0, `T2` has level 1, and so on. Then we do not need to distinguish two kinds of arrow, since this information is implicit from the respective levels of adjacent nodes.

Definition 3. *A* hiproof of type 2 *is a tree $\langle V, \dashrightarrow, r\rangle$ together with two functions: $t : V \dashrightarrow \Lambda$, which labels nodes with tactic identifiers, and $l : V \dashrightarrow \mathbb{N}$, which assigns to each node a natural number, called its* inclusion level. *(Informally, however, we shall often identify a node v with a pair $\langle \lambda, l\rangle$, thereby implicitly asserting that $t(v) = \lambda$ and $l(v) = l$.) These data are subject to the following conditions:*

1. *$l(r) = 0$, i.e. the root of the tree lies at inclusion level 0;*
2. *whenever $v \to v'$ one has $l(v') \leq l(v) + 1$; and*
3. *whenever $v \to v_1$, $v \to v_2$ and, moreover, $l(v_1) = l(v) + 1$, then $v_1 = v_2$.* □

The second condition states that nodes are only (directly) connected to those nodes which they *directly* include or are composed with. In the latter case, the node can 'escape' to an arbitrarily lower inclusion level. Let $\mathbf{Hiproof}_2$ denote the set of hiproof structures of type 2, arising from Definition 3 above.

Example 3. The example from Fig. 1(b) forms a hiproof of type 2 like so

$$\begin{tikzcd}
 & (T1,0) \arrow[d] & \\
 & (T2,1) \arrow[dl] \arrow[dr] & \\
(WF,1) & & (DP,0) \arrow[d] \\
 & & (Normalise,1) \arrow[d] \\
 & & (Taut,1)
\end{tikzcd}$$

where nodes are informally represented by their tactic identifiers and inclusion levels. □

In Def. 3, both sequentiality and inclusion depth are implicit in the structure of the nodes and thus, in terms of their cognitive properties, the diagrams arising from hiproofs of type 2 may be less suited to human users than the diagrams arising from hiproofs of type 1. In the latter, two distinct visual relations — spatial containment and edge connectivity — are used to represent tactic inclusion and sequentiality (recall Fig. 1(b)), thus eliminating any potential for confusion. Owing to their economy, however, type 2 hiproofs may have distinct advantages as internal, machine-oriented representations.

3 Hiproof mappings and equivalence

Contrasting Fig. 1(b) and Example 3 seems to suggest that any given proof may be 'equivalently' represented in either type of hiproof. Indeed, we show here that the two definitions of hiproof introduced in the previous section can be related by two, mutually inverse functions between the two sets of hiproofs. Since $\mathbf{Hiproof}_1$ and $\mathbf{Hiproof}_2$ are countable sets, this is not very interesting in itself, unless the mappings effected by the two functions in question are intuitive, meaningful and practically useful. This is indeed so in our case, where the two functions we consider intuitively map a hiproof of one type to the alternative but equivalent representation of the *same underlying proof* as a hiproof of the other type; and vice-versa.

Definition 4. *Define* μ_{12} : $\mathbf{Hiproof}_1 \to \mathbf{Hiproof}_2$ *as the function sending each hiproof* $\langle V, \leq_{\mathrm{i}}, \rightarrow_{\mathrm{s}}, t\rangle$ *of type 1 (Def. 2) to the hiproof* $\langle V, \rightarrow, r, t, l\rangle$ *of type 2 given by the following data:*

- $l(v)$ *is defined to equal 0 whenever* $\mathit{isroot}_{\leq_{\mathrm{i}}}(v)$ *(i.e. v is a root of the forest* qua poset $\langle V, \leq_{\mathrm{i}}\rangle$*) and, recursively, to equal* $l(\mathit{parent}_{\leq_{\mathrm{i}}}(v)) + 1$ *otherwise;* ~~*(explicitly in forest qua poset notation: for each* $v' \neq \top$, $\mathit{parent}_{\leq_{\mathrm{i}}}(v')$ *is the*~~ *unique v such that* $v' \in \mathit{cover}_{\leq_{\mathrm{i}}}(v)$*);*
- $v \rightarrow v'$ *whenever* $v \rightarrow_{\mathrm{s}} v'$ *or,* $v' \in \mathit{cover}_{\leq_{\mathrm{i}}}(v)$ *and* $\mathit{isroot}_{\rightarrow_{\mathrm{s}}}(v')$*; and*
- r *is the unique node asserted by Lemma 1.* □

Definition 5. *Define* μ_{21} : $\mathbf{Hiproof}_2 \to \mathbf{Hiproof}_1$ *as the function sending each hiproof* $\langle V, \rightarrow, r, t, l\rangle$ *of type 2 (Def. 3) to the hiproof* $\langle V, \leq_{\mathrm{i}}, \rightarrow_{\mathrm{s}}, t\rangle$ *of type 1 given by the following data:*

- $v \rightarrow_{\mathrm{s}} v'$ *whenever* $v \rightarrow v'$ *and* $l(v') \leq l(v)$
- $\leq_{\mathrm{i}}$ *is the reflexive and transitive closure of* $<_{\mathrm{i}}^{1}$*, the latter being defined thus:* $v <_{\mathrm{i}}^{1} v'$ *whenever a (non-empty) path* $v' = v_0 \rightarrow \ldots \rightarrow v_n \rightarrow v_{n+1} = v$ *exists such that* $l(v_1) = l(v_0) + 1$ *and* $l(v_i) = l(v_{i+1})$ *for* $1 \leq i \leq n$. □

Having established both μ_{12} and μ_{21} are well-defined (Propositions 5 and 6 in the Appendix), we now proceed to show also that they are mutually inverse.

Theorem 1. *The functions* μ_{12} : $\mathbf{Hiproof}_1 \to \mathbf{Hiproof}_2$ *and* μ_{21} : $\mathbf{Hiproof}_2 \to \mathbf{Hiproof}_1$ *are mutually inverse.*

Proof. We only sketch one direction of the argument, as the other is similar. Let $h_1 = \langle V, \leq_{\mathrm{i}}, \rightarrow_{\mathrm{s}}, t\rangle$, $h_2 = \mu_{12}(h_1) = \langle V_2, \rightarrow, r, t_2\rangle$ *and* $h_1' = \mu_{21}(h_2) = \langle V', \leq_{\mathrm{i}}', \rightarrow_{\mathrm{s}}', t'\rangle$*. From the definitions, it immediately follows that* $V = V' = V_2$ *and* $t = t_2 = t'$*. Here we present only the proof of* $\leq_{\mathrm{i}}' \subseteq \leq_{\mathrm{i}}$ *(required to show* $\leq_{\mathrm{i}}' = \leq_{\mathrm{i}}$*) as a representative case.*

Assume $v \leq_{\mathrm{i}}' v'$ *and proceed by induction on the length of the path linking* v' *to* v *in the forest* $\langle V, \leq_{\mathrm{i}}'\rangle$*. The base case is trivial. For the inductive case, assume* $v \leq_{\mathrm{i}} v''$ *by the induction hypothesis and* $v'' \in \mathit{cover}_{\leq_{\mathrm{i}}'}(v')$*. From the latter and Def. 5, a path* $v' = v_0 \rightarrow \ldots \rightarrow v_n \rightarrow v_{n+1} = v''$ *must exist in* h_2 *such that* $l(v_1) = l(v_0) + 1$ *and* $l(v_{i+1}) = l(v_i)$ *for* $1 \leq i \leq n$*. It follows from Def. 4 that* $v' = v_0 = \mathit{parent}_{\leq_{\mathrm{i}}}(v_1)$ *or, equivalently,* $v_1 \in \mathit{cover}_{\leq_{\mathrm{i}}}(v')$*. From* $v_i \rightarrow v_{i+1}$ *and* $l(v_i) = l(v_{i+1})$*, in turn, follows that* v_i *and* v_{i+1}*, for i ranging as above, also share* $v' = v_0$ *as parent in* $\leq_{\mathrm{i}}$*. Hence,* $v'' \in \mathit{cover}_{\leq_{\mathrm{i}}}(v')$*. Together with the inductive hypothesis* $v'' \leq_{\mathrm{i}} v'$*, this proves* $v \leq_{\mathrm{i}} v'$*.*

Similarly, $\leq_{\mathrm{i}} \subset \leq_{\mathrm{i}}'$*. The other equality* $\rightarrow_{\mathrm{s}} = \rightarrow_{\mathrm{s}}'$ *can be shown in a similar fashion using inductive arguments.* □

We have now shown that the maps are mutually inverse. The next step is to introduce a notion of 'underlying proof' for hiproofs, and show that the maps preserve this.

We will define the *skeleton* of a hiproof to be the tree of its atomic tactics. If we think of atomic tactics as actually being inferences and axioms then this gives us a standard non-hierarchical proof.

Definition 6. *Let $h_1 = \langle V, \leq_i, \rightarrow_s, t\rangle$ be a type 1 hiproof. We define the* skeleton *of h_1 to be the Λ-labelled tree $\langle V_T, \rightarrow_T, r\rangle$, corresponding (via Proposition 4) to the finite poset $T = \langle V_T, \leq_T\rangle$, where V_T are the leaves of $\leq_i$, and $v_1 \leq v_2$ iff there exists a $v \in V$ such that $v_2 \leq_i v$ and $v_1 \rightarrow_s v$.* □

It is easily seen that Proposition 4 applies (since the poset has the non-sharing ~~condition) and we can construct a tree.~~

We denote the skeleton of a type 1 hiproof, h_1, by $\mathsf{sk}_1(h_1)$. For example, the skeleton of the hierarchical proof at the end of Example 1 is the original basic proof.

Definition 7. *Let $h_2 = \langle V, \rightarrow, r, t, l\rangle$ be a type 2 hiproof. Define an* inclusion node *to be one which has a child with greater inclusion level. We define the* skeleton *of h_2 to be the Λ-labelled tree $\langle V_T, \rightarrow_T, r\rangle$ where V_T are the non-inclusion nodes, and $v \rightarrow_T v'$ iff $v \rightarrow v_1 \rightarrow \cdots \rightarrow v_n \rightarrow v'$, where $v_1, \ldots, v_n$ are inclusion nodes, r is the maximum non-inclusion node, and labelling is given by the restriction of the labelling function to V_T.* □

We denote the skeleton of a type 2 hiproof, h_2, by $\mathsf{sk}_2(h_2)$. This too is easily seen to be a well-formed tree. And so, we can now formally justify that equivalent type 1 and type 2 hiproof equivalence denote the same underlying proof:

Theorem 2. *Let h_1 and h_2 be hiproofs of type 1 and 2, respectively. Then,*

1. $\mathsf{sk}_1(h_1) = \mathsf{sk}_2(\mu_{12}(h_1))$
2. $\mathsf{sk}_2(h_2) = \mathsf{sk}_1(\mu_{21}(h_2))$.

Proof. By well-founded induction on the hiproofs. At each stage we extend the hiproof by one leaf, and show that $\mathsf{sk}\_$ *and* $\mu\_$ *respect this appropriately.* □

4 Hiproof refinement

We would like to introduce structure between hiproofs. The obvious notion of structure on a set of hiproofs is one of *refinement.* Intuitively, this corresponds to proof development. Hence h_1 refines to h_2 when h_2 extends the proof in h_1.

Proofs can grow in two ways: either by unfolding a tactic, or by applying a tactic to a subgoal. These correspond, respectively, to inclusion of and sequential composition with a tactic. Since we want to formalise a *semantic*, rather than operational, notion of refinement, refinement amounts to allowing trees to grow arbitrarily 'at the bottom' and, in the case of forests, adding additional trees. The definitions in this section make this precise. For each notion of hiproof introduced in the previous section, a relevant notion of refinement is presented, and we now ask that the maps preserve hiproof refinement.

We begin by defining preliminary notions of refinement for trees and forests, the simple structures from which hiproofs are constructed. A *rooted subtree* of a given tree is essentially a non-empty subset of the tree's nodes which is upwards closed wrt. $\rightarrow$. Formally:

Definition 8. *A rooted subtree T' of a given tree $T = \langle V, \rightarrow, r\rangle$ is a tree $\langle V', \rightarrow', r\rangle$ where*

- *V' is a non-empty subset of V which is upwards closed: whenever $v' \in V'$ then for all v'' such that $v'' \rightarrow v'$ one also has $v'' \in V$; and thus also $r \in V'$*
- *$\rightarrow'$ is the restriction of $\rightarrow$ to $V' \times V'$.* □

Henceforth, we refer to 'rooted subtrees' simply as 'subtrees'. Intuitively, refining a tree amounts to the addition, at possibly any level below the root, of any (finite) number of new nodes. Thus the original tree is always a subtree of any tree which refines it:

Definition 9. *Tree $\langle V_1, \rightarrow_1, r_1\rangle$ refines to tree $\langle V_2, \rightarrow_2, r_2\rangle$, written*

$$\langle V_1, \rightarrow_1, r_1\rangle \sqsubseteq_{\mathrm{T}} \langle V_2, \rightarrow_2, r_2\rangle \ ,$$

iff the former is a subtree of the latter. □

Definition 10. *Forest F_1 refines to forest F_2, written $F_1 \sqsubseteq_{\mathrm{F}} F_2$, if there exists an injective function $\iota : F_1 \rightarrow F_2$ such that for all trees $T \in F_1$, $T \sqsubseteq_{\mathrm{T}} \iota(T)$* □

In practice, it is easier to use the following characterisations of forest refinement, which benefit from regarding a forest as a graph or poset:

Proposition 2. *Given forests $F_1 = \langle V_1, \rightarrow_1\rangle$ and $F_2 = \langle V_2, \rightarrow_2\rangle$, $F_1 \sqsubseteq_{\mathrm{F}} F_2$ iff $V_1 \subseteq V_2$ and for all $v, v' \in V_1$, $isroot_{F_1}(v) \iff isroot_{F_2}(v)$ and $v \rightarrow_1 v' \iff v \rightarrow_2 v'$.* □

Proposition 3. *Given forests $F_1 = \langle V_1, \leq_1\rangle$ and $F_2 = \langle V_2, \leq_2\rangle$, $F_1 \sqsubseteq_{\mathrm{F}} F_2$, iff for all $v, v' \in V_1$, $isroot_{F_1}(v) \iff isroot_{F_2}(v)$ and $v \in cover_{F_1}(v') \implies v \in cover_{F_2}(v')$.* □

We are now in a position to formally present our notion of hiproof refinement.

Definition 11. *A hiproof $h = \langle V, \leq_{\mathrm{i}}, \rightarrow_{\mathrm{s}}, t\rangle$ of type 1 refines to a hiproof $h' = \langle V', \leq'_{\mathrm{i}}, \rightarrow'_{\mathrm{s}}, t'\rangle$ of the same type, written $h \sqsubseteq_1 h'$, iff $\langle V, \leq_{\mathrm{i}}\rangle \sqsubseteq_{\mathrm{F}} \langle V', \leq'_{\mathrm{i}}\rangle$, $\langle V, \rightarrow_{\mathrm{s}}\rangle \sqsubseteq_{\mathrm{F}} \langle V', \rightarrow'_{\mathrm{s}}\rangle$ and, moreover, labels are preserved: i.e. $t \subseteq t'$ (regarding here each of t and t' as a finite set of pairs, e.g. $\langle v, t(v)\rangle$).* □

Definition 12. *A hiproof $h = \langle V, \rightarrow, r, t, l\rangle$ of type 2 refines to a hiproof $h' = \langle V', \rightarrow, r', t', l'\rangle$ of the same type, written $h \sqsubseteq_2 h'$, iff $\langle V, \rightarrow, r\rangle \sqsubseteq_{\mathrm{T}} \langle V', \rightarrow', r'\rangle$ and, moreover, $t \subseteq t'$ and $l \subseteq l'$.* □

Theorem 3. *Let h_1 and h'_1 be hiproofs of type 1, and h_2 and h'_2 be hiproofs of type 2. Then,*

1. *if $h_1 \sqsubseteq_1 h'_1$ then $\mu_{12}(h_1) \sqsubseteq_2 \mu_{12}(h'_1)$, and*
2. *if $h_2 \sqsubseteq_2 h'_2$ then $\mu_{21}(h_2) \sqsubseteq_1 \mu_{21}(h'_2)$.* □

The proof is straightforward. The interesting point is that this is evidence that we have a natural definition of refinement.

5 Conclusion

This paper has made a first study of some structures used to represent hierarchy in formal proofs. We have described two different types of hiproof and shown their equivalence. There are other possible definitions of hiproof, not considered here, but a truly unifying treatment is likely to necessitate a more abstract approach, such as a category-theoretic one. Hiproof refinement already introduces categorical structure.

While the subtlety of our two definitions of hiproof clearly reflects the difficulty of capturing graphical intuitions mathematically, we do believe the underlying concepts to be robust and insightful. The feedback which we have received so far seems to support this belief. Making what is graphically implicit mathematically explicit is a necessary first step for reasoning about these diagrams.

We readily acknowledge that, in practice, tactics possess a structure far more complex than we have studied here. Although it may seem that we have abstracted too far we believe, nevertheless, that our definitions are at an appropriate level. Firstly, the particular axioms on hiproofs are motivated specifically by considering proofs and tactics. Secondly, our aim here has been to study *hierarchy* and it makes sense to first study this in a minimal setting. Thirdly, a simple graphical interface would, we believe, represent tactics at this level (for example, [RSG98]).

The main result of the paper is not the equivalence of two concrete definitions of hiproof, but the fact that we have found a suitable level of abstraction at which such equivalences exist. We are currently developing a more axiomatic definition, which captures the essential features of these, and other, concrete instances. We see the equivalence of two independent notions of hierarchy as evidence that we have chosen the correct level of abstraction.

Another important task is to the define operations on hiproofs, supported in the various proof assistants, such as various abstraction operations. Such 'zooming' operations have been considered for higraphs and statecharts [Har88,APT02], and there are natural operations to consider here.

We are not claiming that our theory will lead to new, or better written tactics in existing systems. The advantage, though, of deciding on the general properties and operations of hiproofs would be to give a principled way of designing an interface which does not depend on the specifics of hiproof representation. Moreover, such a theory would allow us to reason about the correctness of an implementation.

We also plan to characterise the relationship between our semantic structures and the underlying logic, introducing a notion of stepwise refinement. Finally, as we have acknowledged in this paper, our theory only attempts to describe a simple notion of tactic and proof. Modern theorem provers make use of many rich structures. This work is just a first step towards providing a semantic foundation for this area.

References

[APT02] Stuart Anderson, John Power, and Konstantinos Tourlas. Zooming out of higraph-based diagrams: syntactic and semantic issues. In *Proceedings of CATS 2002, the Australasian Symposium on Theory of Computing*, volume 61 of *Electronic Notes in Computer Science (ENTCS)*. Elsevier, 2002.

[BCF+97] Christoph Benzmüller, Lassaad Cheikhrouhou, Detlef Fehrer, Armin Fiedler, Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Karsten Konrad, Andreas Meier, Erica Melis, Wolf Schaarschmidt, Jörg Siekmann, and Volker Sorge. Omega: Towards a mathematical assistant. In *Proceedings of CADE-14*, volume 1249 of *LNAI*. Springer, 1997.

[BH96] Jon Barwise and Eric Hammer. Diagrams and the concept of logical system. In G. Allwein and J. Barwise, editors, *Logical Reasoning with Diagrams*, pages 49–78. Oxford University Press, 1996.

[Bun96] A. Bundy. Proof planning. In B. Drabble, editor, *Proceedings of the 3rd International Conference on AI Planning Systems, (AIPS) 1996*, pages 261–267, 1996. Also available as DAI Research Report 886.

[CS00] L. Cheikhrouhou and V. Sorge. PDS — A Three-Dimensional Data Structure for Proof Plans, 2000.

[Har88] David Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988.

[KNM94] D. Kapur, X. Nie, and D. R. Musser. An overview of the Tecton proof system. *Theoretical Computer Science*, 133(2):307–340, 1994.

[RB99] Julian Richardson and Alan Bundy. Proof Planning Methods as Schemas. DAI Technical Report, Division of Informatics, University of Edinburgh, 1999. Also available at `http://www.dai.ed.ac.uk/~julianr/proof_planning.ps.gz`.

[RS01] J. D. C. Richardson and A. Smaill. Continuations of proof strategies. In *International Joint Conference on Automated Reasoning, IJCAR - 2001 — Short Papers*, June 2001. Technical Report DII 11/01, Dipartimento di Ingegneria dell'Informazione, Università di Siena, Italy.

[RSG98] J. D. C Richardson, A. Smaill, and I. Green. System description: proof planning in higher-order logic with Lambda-Clam. In *15th International Conference on Automated Deduction*, pages 129–133, 1998.

Appendix: Graphs, trees and forests

In this section we present the basic order theory together with some novel definitions needed for our definitions of hiproof.

Definition 13. *A* partially ordered set *(or* poset *for short)* $\langle X, \leq \rangle$*, is a set* X *together with a reflexive, antisymmetric and transitive relation* $\leq$ *on* X*. For each* $x \in X$ *the set* $cover(x)$ *is defined to be*

$$\{x' \in X \mid x' < x \text{ and for all } x'' \in X \text{ such that } x'' < x \text{ one has } x'' \leq x'\}$$

and will be referred to as the cover *of* x*.* □

As usual, we write $x < y$ to mean $x \leq y$ but not $x = y$. Thus, an element y is in the cover of x iff y is strictly less than x and also $y' \leq y$ for all other y' such that $y' < x$.

The graphs we consider here are directed, consisting as usual of a pair $\langle V, E \rangle$, where V a set (the *vertices*), and $E \subseteq V \times V$ a binary relation on V (the *edges*). Thus, writing $E^\star$ for the reflexive and transitive closure of the relation E, a path from v to v' exists if and only if $\langle v, v' \rangle \in E^\star$. An important restriction on the graphs we consider is that they contain no cycles (i.e. non-empty paths from any vertex to itself).

Definition 14. *A* tree $T = \langle V, \rightarrow, r \rangle$ *is a finite dag* $\langle V, \rightarrow \rangle$ *together with a distinguished vertex* $r \in V$, *called the* root, *such that there is exactly one path from* r *to every other vertex* $v \neq r$. *For every edge* $\langle v, v' \rangle \in \rightarrow$, *which we shall conventionally write as* $v \rightarrow v'$, *one says that* v' *is a* child *of* v *or, equivalently, that* v' *has* parent v. *The vertices* V *in a tree are conventionally also called* nodes. □

Given any two nodes v and v' in a tree $T = \langle V, \rightarrow, r \rangle$ we say that v' is a *descendent* of v, and write $v' \leq v$, to mean that there is a path in the tree from v to v'. Thus, formally, $v' \leq v$ holds iff $v \rightarrow^\star v'$. As $\leq$ turns out to be a partial order, it is often most convenient to interchange the graph-theoretic and order-theoretic views of trees:

Proposition 4. *To give a tree (in the sense of Def. 14) is equivalent to giving a finite poset* $T = \langle V, \leq \rangle$ *which satisfies the 'non-sharing' condition*

$$\forall\, x, y, z \in F.\ \ x \leq y \ \text{ and } \ x \leq z \ \text{ implies } \ y \leq z \text{ or } z \leq y\ ,$$

and has a top element $\top$. *The resulting tree is* $\langle V, \rightarrow, \top \rangle$, *where* $v \rightarrow v'$ *whenever* $v' \in \mathit{cover}_{\leq}(v)$.

□

Definition 15. *A* forest F *is a finite set* $\{T_1, \ldots, T_n\}$ *of trees* $T_j = \langle V_j, \rightarrow_j, r_j \rangle$. *We shall write* $v \rightarrow_F v'$ *(or just* $v \rightarrow v'$ *when* F *understood) to mean that there exists tree* T_j *in* F *such that* $v, v' \in V_j$ *and* $v \rightarrow_j v'$. *Consequently we shall often also write the forest* F *as* $\langle V, \rightarrow_F \rangle$ *where* V *is the disjoint union of all* V_j. □

Corollary 1. *To give a forest in the sense of Def. 15 is equivalent to giving a* finite *poset* $\langle V, \leq \rangle$ *subject only to the 'non-sharing' condition of Prop. 4:* $\forall\, x, y, z \in V.\ \ x \leq y$ *and* $x \leq z$ *implies* $y \leq z$ *or* $z \leq y$. □

Appendix: Technical proofs

Proof of Lemma 1

Proof. Using contraposition and the shunting equivalence $(p \wedge q \implies r) \iff (p \implies (q \implies r))$, the third condition in Def. 2 may equivalently be rewritten thus: $\forall v, v'.\ siblings_i(v, v') \wedge v \neq v' \wedge isroot_s(v) \implies \neg isroot_s(v')$. Taking v to be v_0 with $isroot_s(v_0)$, i.e. a v_0 with no incoming $\rightarrow_s$ edges, it follows that every sibling v' of v_0 which is not v_0 itself cannot be root wrt. $\rightarrow_s$, i.e. v' must have an incoming $\rightarrow_s$ edge. Assuming also v_0 to be $\leq_i$-maximal, its siblings (excluding itself) are precisely all the other $\leq_i$-maximal nodes, and the result follows.

Defining $v <_i^1 v'$ as $v \in cover_{\leq_i}(v')$ and $<_i^n \stackrel{\text{def}}{=} (<_i^1)^n$ one proves $v \rightarrow_s w_1 \wedge w_1 <_i^n w_2 \implies v <_i^n w_2$, from which the claimed equivalence is easily obtained. □

Proposition 5. *μ_{12} above is indeed well-defined, i.e. each $\mu_{12}(\langle V, \leq_i, \rightarrow_s, t\rangle)$ conforms to Def. 3.*

Proof. *Firstly, by recalling Prop. 1 and observing that $\rightarrow^+ \subseteq (>_i^1 \cup \rightarrow_s)^+$, it follows that $\langle V, \rightarrow\rangle$ is an acyclic graph. Moreover, whenever $v_1 \rightarrow v$ and $v_2 \rightarrow v$ one has $v_1 = v_2$ (for the only possible cases are $v_1 \rightarrow_s v$ and $v_1 \rightarrow_s v$, or, $v \in cover_{\leq_i}(v_1)$ and $v \in cover_{\leq_i}(v_2)$; $v_1 = v_2$ immediately follows from $\langle V, \rightarrow_s\rangle$ and $\langle V, \leq_i\rangle$ being forests). Thus, whenever a path $v_0 \rightarrow^* v$ exists it must be unique. We show that indeed $r \rightarrow_s^* v$ for all $v \in V$ by induction on $d(v)$, the 'depth' of v wrt. $\rightarrow_s$, which is defined thus: $d(v) = d(v') + 1$ whenever $v' \rightarrow v$ and $d(v) = 0$ otherwise. When $d(v) = 0$ then clearly $isroot_{\rightarrow_s}(v)$ and $siblings_{\leq_i}(v, r)$ (in the sense that $isroot_{\leq_i}(v)$). Now $r = v$ by Lemma. 1, and $r \rightarrow^* v$ holds trivially. In the inductive case assume true for v' and $v' \rightarrow v$. Then the induction hypothesis yields $r \rightarrow v'$ and so, transitively, also $r \rightarrow^* v$.*

Showing that $l(v') \leq l(v) + 1$ whenever $v \rightarrow v'$ proceeds by case analysis. Case $v' \in cover_{\leq_i}(v)$ and $isroot_{\rightarrow_s}(v')$ is immediate. When $v \rightarrow_s v'$ one examines whether $isroot_{\leq_i}(v')$ or not. When so, $l(v') = 0 \leq l(v) + 1$. When $v' \in cover_{\leq_i}(v'')$ for some v'', condition 2 yields (via Lemma 1) $v' \in cover_{\leq_i}(v'')$, from which $l(v) = l(v'') + 1 = l(v')$ follows.

Assume $v \rightarrow v_1$ and $v \rightarrow v_2$ such that $l(v_1) = l(v) + 1$. Then one must have $v = parent_{\leq_i}(v_1)$ and hence $v_1 \leq_i v$ in the type 1 hiproof. Further, we distinguish two cases: firstly, in the case $l(v_2) = l(v_1)$ one has $siblings_{\leq_i}(v_1, v_2)$ while also $isroot_{\rightarrow_s}(v_1) \wedge isroot_{\rightarrow_s}(v_2)$. Then condition 4 of Def. 2 establishes $v_1 = v_2$, as required. Similarly the case $l(v_2) \leq l(v)$ means $v \rightarrow_s v_2$ while $v_1 \leq_i v$, hence $v_1 = v_2$ by the third condition in the definition of type 1 hiproofs. Finally, that $l(r) = 0$ is obvious. □

Lemma 2. *In the context of Def. 5, $isroot_{\rightarrow_s}(v)$ is equivalent to $\forall v_0 \in V.\ (v_0 \rightarrow v \implies l(v_0) + 1 \leq l(v))$.*

Proof. *Using the definition of $\rightarrow_s$ and the tautology $(p \implies q) \iff (\neg p \vee q)$:*

$$\begin{aligned} isroot_{\rightarrow_s}(v) &\iff \nexists v_0.\ (v_0 \rightarrow v \ \wedge\ l(v) \leq l(v_0)) \\ &\iff \forall v_0.\ (v_0 \not\rightarrow v \ \vee\ l(v) > l(v_0)) \\ &\iff \forall v_0.\ (v_0 \not\rightarrow v \ \vee\ l(v) \geq l(v_0) + 1) \\ &\iff \forall v_0.\ (v_0 \rightarrow v \implies l(v_0) + 1 \leq l(v)) \ \square \end{aligned}$$

Proposition 6. *μ_{21} above is well-defined, i.e. each $\mu_{21}(\langle V, \rightarrow, r, t, l\rangle)$ is a hiproof of type 1.*

Proof. (Sketch) $<_\mathrm{i}$ *is manifestly irreflexive and transitive. On the other hand,* $\leq_\mathrm{i}$ *is clearly antisymmetric, as follows from observing that* $\leq_\mathrm{i} \subseteq (\rightarrow^\star)^{-1}$ *while* $\langle V, \rightarrow, r\rangle$ *is a tree. Thus, the definition of* $\leq_\mathrm{i}$ *as the reflexive closure of* $<^1_\mathrm{i}$ *makes* $\langle V, \leq_\mathrm{i}\rangle$ *a poset. The non-sharing condition, needed by Corol. 1 to show* $\langle V, \leq_\mathrm{i}\rangle$ *a forest, as required, also follows from* $\langle V, \rightarrow, r\rangle$ *being a tree and* $\leq_\mathrm{i} \subseteq (\rightarrow^\star)^{-1}$*: to assume* $v \leq_\mathrm{i} w_1$ *and* $v \leq_\mathrm{i} w_2$ *while* $w_1 \neq w_2$ *would mean the existence of two distinct paths in* $\langle V, \rightarrow, r\rangle$ *from the root* r *to* v*, one via* w_1 *and the other via* w_2*, thus contradicting the fact of* $\langle V, \rightarrow, r\rangle$ *being a tree. One must therefore admit that, whenever* $v \leq_\mathrm{i} w_1$ *and* $v \leq_\mathrm{i} w_2$*,* w_1 *must equal* w_2*.*

To show $\langle V, \rightarrow_\mathrm{s}\rangle$ *a forest, as required, we shall show that* $\langle V, (\rightarrow^\star_\mathrm{s})^{-1}\rangle$ *is a forest-qua-poset and appeal to Corol. 1. The poset structure of* $\langle V, (\rightarrow^\star_\mathrm{s})^{-1}\rangle$ *is immediate as* $\rightarrow$ *is clearly antisymmetric. Again, observing that* $\rightarrow^\star_\mathrm{s} \subseteq \rightarrow^\star$*, the 'non-sharing' condition required by Corol. 1 follows, as above, from* $\langle V, \rightarrow, r\rangle$ *being a tree.*

To show that $\leq_\mathrm{i}$ *and* $\rightarrow_\mathrm{s}$ *are mutually exclusive in the sense of condition (1) of Def. 2, consider first the case of* $v \leq_\mathrm{i} w$ *and* $v \rightarrow^\star_\mathrm{s} w$*: as the former implies* $w \rightarrow^\star v$ *and the latter implies* $v \rightarrow^\star_\mathrm{s} w$*,* $v = w$ *follows easily from the acyclicity of the tree* $\langle V, \rightarrow, r\rangle$*. In the case of* $v \leq_\mathrm{i} w$ *(hence also* $l(w) \leq l(v)$*) while* $w \rightarrow^\star_\mathrm{s} v$ *(and hence* $l(v) \leq l(w)$*), one must have* $l(v) = l(w)$ *and so, according to the definition of* $\leq_\mathrm{i}$*,* $v = w$*.*

To establish condition 2 we appeal to Lemma 1. First, observe that $w_1 \in cover_{\leq_\mathrm{i}}(w_2)$ *means the existence of a non-empty path* $w_2 \rightarrow v_1 \rightarrow \ldots \rightarrow v_n \rightarrow w_1$ *in the tree* $\langle V, \rightarrow, r\rangle$ *such that* $l(v_1) = l(w_2) + 1$ *and* $l(v_1) = \ldots = l(v_n) = l(w_1)$*. Assuming also that* $v \rightarrow_\mathrm{s} w_1$*, i.e. also* $v \rightarrow w_1$*, forces* $v = v_n$*, for* $\langle V, \rightarrow, r\rangle$ *is a tree. That* $v \in cover_{\leq_\mathrm{i}}(w_2)$ *now follows immediately from Def. 5.*

For condition 3, suppose that $w_1 \leq_i v$ *and* $v \rightarrow_s w_2$*. We must show that* $v = w_1$*. By the definition of* μ_{21}*, we have that* $w_1\,(<^1_i)^*\,v$ *and* $v \rightarrow w_2$*, with* $l(w_2) \leq l(v)$*. Suppose that the path from* w_1 *to* v *is non-empty, i.e.* $w_1 <^1_i w'_0\,(<^1_i)^*\,v$*, for some* w'_0*. Then by the Definition of* $<^1_i$ *there exists a* w_0 *such that* $v \rightarrow w_0$ *and* $l(w_0) = l(v) + 1$*. Now, by condition 3 of Def. 3, we have that* $w_2 = w_0$*, but this is impossible because they have different levels. Therefore the path from* w_1 *to* v *must be be empty, and so* $w_1 = v$*.*

For showing condition 4 assume first that $siblings_{\leq_\mathrm{i}}(v, v')$*. Then either* $v = v'$*, or else (by unfolding the definition of* $cover_{\leq_\mathrm{i}}$*) there must exist* v_0 *such that, at least,* $v_0 \rightarrow v$*, and* $v_0 \rightarrow v'$*. Thus, by condition 2 of Def. 3,* $l(v) \leq l(v_0) + 1$ *and* $l(v') \leq l(v_0) + 1$ *also hold. Assuming further that* $isroot_{\rightarrow_\mathrm{s}}(v) \wedge isroot_{\rightarrow_\mathrm{s}}(v')$*,* $v_0 \rightarrow v$ *and* $v_0 \rightarrow v'$ *additionally yield, by Lemma 2, that* $l(v_0) + 1 \leq l(v)$ *and* $l(v_0) + 1 \leq l(v')$*. Hence,* $l(v) = l(v') = l(v_0) + 1$ *and, by condition (3) of Def. 3, it now follows that* $v_1 = v_2$*, as required.* □