Full First-Order Free Variable Sequents and Tableaux in Implicit Induction

Claus-Peter Wirth

FR Informatik, Saarland Univ., D–66041 Saarbrücken, Germany
cp@ags.uni-sb.de

Abstract. We show how to integrate implicit inductive theorem proving into free variable sequent and tableau calculi and compare the appropriateness of tableau calculi for this integration with that of sequent calculi.

When first-order validity is introduced to students it comes with some complete calculus. If this calculus happens to be an analytic calculus augmented with a Cut rule like a sequent or tableau calculus the students can compare the formal proofs with the informal ones they are hopefully acquainted with. This is because these calculi can mirror the human proof search process better than others. While knowing a complete calculus does not mean to know much about first-order theorem proving, the interrelation of a human-oriented calculus and the informal proof search of the students will turn out to be fruitful for their later mathematical work. It is a pity that—while nearly all proofs of a working mathematician include induction—nothing comparable for *inductive* first-order validity is offered to the students. Some may argue that this is generally impossible because not even the theory of the Peano algebra of natural numbers is recursively enumerable, cf. e.g. Enderton (1973). Nevertheless, there really is some general way a working mathematician searches for an informal proof, may it be inductive or not. The inductive version of this proof search method goes back to the ancient Greeks and was rediscovered under the name "descente infinie" by Pierre de Fermat (1601-1665). If you want to prove a conjecture, this method requires that you show, for each assumed counterexample of the conjecture, the existence of another counterexample of the conjecture that is strictly smaller in some wellfounded ordering. The working mathematician applies it in the following fashion. He (who may be female!) starts with the conjecture and simplifies it in case analyses which can be described as steps in a sequent or tableau calculus with Cut. When he realizes that the goals become similar to a different instance of the conjecture, he applies the conjecture just like a lemma, but keeps in mind that he actually has applied some induction hypothesis. Finally, he searches for some wellfounded ordering in which all the instances of the conjecture that he has applied as induction hypotheses are smaller than the original conjecture itself. Looking for a formal inductive calculus for mirroring this style of human inductive theorem proving (*ITP*), the "implicit induction" of Bachmair (1988) was a starting point because it included hypothesis application, although it was restricted to universally quantified pure equations and was not human-oriented. In Wirth (1997) we have presented a human-oriented inductive calculus for universally quantified clausal logic. In Kühler (1999)—implemented as the QuodLibet system—this calculus is extended with some necessary and important concretion for reasoning on the induction ordering and with a tactic-based concept for proof guidance that is intended to partially automate the construction of proofs. Extending this approach to full first-order logic turned out to be more difficult than expected (cf., however, Padawitz (1996) for the extension to another interesting sub-class): The state-of-the-art free variable analytic first-order calculi were not suited for the integration of "implicit induction" because they confused the Herbrand universes with their Skolem functions and did not preserve solutions (i.e. closing substitutions) (like Prolog does), thereby destroying the wellfoundedness of "descente infinie". In Wirth (1998) we have developed sequent and tableau calculi for full first-order formulas that do not Skolemize but do preserve solutions. These new calculi come in two versions. The *weak* version is simple, but cannot model liberalized versions of the δ-rule, which the *strong* version can. Since the strong version is much more complicated, the space is limited here, and many researchers today are quite unacquainted with "descente infinie", in this paper we will use the weak version only and concentrate on the inductive aspects (i.e. induction hypothesis application) and not on the deductive ones (i.e. α-, β-, γ-, and δ-steps, cf. Smullyan (1968)). Even for the experts in *implicit* ITP each of the following aspects will be new: Tableau presentation, full first-order formulas, and free variables.

We use '$\uplus$' for the union of disjoint classes and 'id' for the identity function. For a class R we define *domain*, *range*, *restriction to*, *image* and *reverse-image of a class* A by $\mathrm{dom}(R) := \{\, a \mid \exists b.\ (a,b) \in R \,\}$; $\mathrm{ran}(R) := \{\, b \mid \exists a.\ (a,b) \in R \,\}$; ${}_{A}{\upharpoonright}R := \{\, (a,b) \in R \mid a \in A \}$; $\langle A \rangle R := \{\, b \mid \exists a \in A.\ (a,b) \in R \}$; $R\langle B \rangle := \{\, a \mid \exists b \in B.\ (a,b) \in R \}$. '$\mathbf{N}$' denotes the set of and '$\prec$' the ordering on natural numbers. We use '$\emptyset$' to denote the empty set as well as the empty function or empty word. A *quasi-ordering '$\lesssim$' on A* is an A-reflexive and transitive (binary) relation on A. As with all our asymmetric relation symbols we define $a \gtrsim b$ if $b \lesssim a$. By an (irreflexive) *ordering* '$<$' (on A) we mean an irreflexive and transitive relation (on A). The *ordering* $<$ *of a quasi-ordering* $\lesssim$ is $\lesssim \setminus \gtrsim$. A quasi-ordering $\lesssim$ is called *total on* C if $C \times C \subseteq \lesssim \cup \gtrsim$. A $\lesssim$*-chain* is some subclass $C \subseteq A$ such that $\lesssim$ is total on C. $\lesssim$ is called *wellfounded* if each $\lesssim$-chain C has a least element, i.e. $\exists a \in C.\ \forall b \in C.\ a \lesssim b$. The *class of total functions from A to B* is denoted with $A \rightarrow B$. The *class of (possibly) partial functions from A to B* is denoted with $A \rightsquigarrow B$.

We define a *sequent* to be a list of formulas. The *conjugate* of a formula A (written: $\overline{A}$) is the formula B if A is of the form $\neg B$, and the formula $\neg A$ otherwise. In the tradition of Gentzen (1935) we assume the symbols for *free existential variables* (i.e. the free variables of Fitting (1996)), *free universal variables* (i.e. nullary parameters), *bound variables* (i.e. variables for quantified use only), and the *constants* (i.e. the function (and predicate) symbols from the signature) to come from four disjoint sets $\mathrm{V}_{\exists}$, $\mathrm{V}_{\forall}$, $\mathrm{V}_{\mathrm{bound}}$, and Σ. We assume each of $\mathrm{V}_{\exists}$, $\mathrm{V}_{\forall}$, $\mathrm{V}_{\mathrm{bound}}$ to be infinite (for each sort) and set $\mathrm{V}_{\mathrm{free}} := \mathrm{V}_{\exists} \uplus \mathrm{V}_{\forall}$. For a term, formula, sequent Γ &c., '$\mathcal{V}_{\exists}(\Gamma)$', '$\mathcal{V}_{\forall}(\Gamma)$', '$\mathcal{V}_{\mathrm{bound}}(\Gamma)$', '$\mathcal{V}_{\mathrm{free}}(\Gamma)$' denote the sets of variables from $\mathrm{V}_{\exists}$, $\mathrm{V}_{\forall}$, $\mathrm{V}_{\mathrm{bound}}$, $\mathrm{V}_{\mathrm{free}}$ occurring in Γ, resp.. For a substitution σ we denote with '$\Gamma\sigma$' the result of replacing in Γ each variable x in $\mathrm{dom}(\sigma)$ with $\sigma(x)$. We tacitly assume that each substitution σ satisfies $\mathcal{V}_{\mathrm{bound}}(\mathrm{dom}(\sigma) \cup \mathrm{ran}(\sigma))$

$= \emptyset$, such that no bound variables can be replaced and no additional variables become bound (i.e. captured) when applying σ.

A *variable-condition* R is a subset of $\mathrm{V}_\exists \times \mathrm{V}_\forall$. Roughly speaking, $(x^\exists, y^\forall) \in R$ says that $x^\exists$ is older than $y^\forall$, so that we must not instantiate the free existential variable $x^\exists$ with a term containing $y^\forall$.

Validity is expected to be given with respect to some Σ-structure (Σ-algebra) $\mathcal{A}$, assigning a universe (to each sort) and an appropriate function to each symbol in Σ. For $\mathrm{X} \subseteq \mathrm{V}_{\mathrm{free}}$ we denote the set of total $\mathcal{A}$-valuations of X (i.e. functions mapping free variables to objects of the universe of $\mathcal{A}$ (respecting sorts)) with $\mathrm{X} \to \mathcal{A}$ and the set of (possibly) partial $\mathcal{A}$-valuations of X with $\mathrm{X} \leadsto \mathcal{A}$. For $\pi \in \mathrm{X} \to \mathcal{A}$ we denote with '$\mathcal{A} \uplus \pi$' the extension of $\mathcal{A}$ to the variables of X which are then treated as nullary constants. More precisely, we assume the existence of some evaluation function 'eval' such that $\mathrm{eval}(\mathcal{A} \uplus \pi)$ maps any term over $\Sigma \uplus \mathrm{X}$ into the universe of $\mathcal{A}$ (respecting sorts) such that for all $x \in \mathrm{X}$: $\mathrm{eval}(\mathcal{A} \uplus \pi)(x) = \pi(x)$. Moreover, $\mathrm{eval}(\mathcal{A} \uplus \pi)$ maps any formula B over $\Sigma \uplus \mathrm{X}$ to TRUE or FALSE, such that B is valid in $\mathcal{A} \uplus \pi$ iff $\mathrm{eval}(\mathcal{A} \uplus \pi)(B) = \mathsf{TRUE}$. We assume that the *Substitution-Lemma* holds in the sense that, for any substitution σ, Σ-structure $\mathcal{A}$, valuation $\pi \in \mathrm{V}_{\mathrm{free}} \to \mathcal{A}$, and term or formula B:

$$\mathrm{eval}(\mathcal{A} \uplus \pi)(B\sigma) = \mathrm{eval}\big(\mathcal{A} \uplus \big((\sigma \uplus {}_{\mathrm{V}_{\mathrm{free}} \setminus \mathrm{dom}(\sigma)} \!\upharpoonright\! \mathrm{id}) \circ \mathrm{eval}(\mathcal{A} \uplus \pi)\big)\big)(B).$$

Finally, we assume that the value of the evaluation function on a term or formula B does not depend on the free variables that do not occur in B:

$$\mathrm{eval}(\mathcal{A} \uplus \pi)(B) = \mathrm{eval}(\mathcal{A} \uplus {}_{\mathcal{V}_{\mathrm{free}}(B)} \!\upharpoonright\! \pi)(B).$$

Further properties of validity or evaluation are definitely not needed.

We are now going to briefly recapitulate the notions from the weak version of Wirth (1998) which we need in what follows. Several binary relations on free variables will be introduced. The overall idea is that when (x, y) occurs in such a relation this means something like "x is older than y" or "the value of y depends on or is described in terms of x".

Definition 0.1 (E_σ, U_σ, Existential R-Substitution, σ-Update)
For a substitution σ with $\mathrm{dom}(\sigma) = \mathrm{V}_\exists$ we define the *existential relation* to be
$E_\sigma := \{ (x', x) \mid x' \in \mathcal{V}_\exists(\sigma(x)) \wedge x \in \mathrm{V}_\exists \}$ and the *universal relation* to be
$U_\sigma := \{ (y, x) \mid y \in \mathcal{V}_\forall(\sigma(x)) \wedge x \in \mathrm{V}_\exists \}$.
Let R be a variable-condition. σ is an *existential R-substitution* if σ is a substitution with $\mathrm{dom}(\sigma) = \mathrm{V}_\exists$ for which $U_\sigma \circ R$ is irreflexive.
Let σ be an existential R-substitution. The *σ-update of R* is $E_\sigma \circ R$.

Note that, regarding syntax, $(x^\exists, y^\forall) \in R$ is intended to mean that an existential R-substitution σ may not replace $x^\exists$ with a term in which $y^\forall$ occurs, i.e. $(y^\forall, x^\exists) \in U_\sigma$ must be disallowed, i.e. $U_\sigma \circ R$ must be irreflexive.

After application of an existential R-substitution σ, in case of $(x^\exists, y^\forall) \in R$, we have to ensure that $x^\exists$ is not replaced with $y^\forall$ via a future application of another existential R-substitution that replaces a free existential variable $u^\exists$ occurring in $\sigma(x^\exists)$ with $y^\forall$. In this case, the new variable-condition has to contain $(u^\exists, y^\forall)$. This means that $E_\sigma \circ R$ must be a subset of the updated variable-condition.

Let $\mathcal{A}$ be some Σ-structure. We now define a semantic counterpart of our existential R-substitutions, which we will call "existential $(\mathcal{A}, R)$-valuation". Suppose that e maps each free existential variable not directly to an object of $\mathcal{A}$ (of the same sort), but can additionally read the values of some free universal variables under an $\mathcal{A}$-valuation $\pi \in \mathrm{V}_\forall \to \mathcal{A}$, i.e. e gets some $\pi' \in \mathrm{V}_\forall \rightsquigarrow \mathcal{A}$ with $\pi' \subseteq \pi$ as a second argument; short: $e : \mathrm{V}_\exists \to ((\mathrm{V}_\forall \rightsquigarrow \mathcal{A}) \to \mathcal{A})$. Moreover, for each free existential variable x, we require the set of read free universal variables (i.e. $\mathrm{dom}(\pi')$) to be identical for all π; i.e. there has to be some "semantic relation" $S_e \subseteq \mathrm{V}_\forall \times \mathrm{V}_\exists$ such that for all $x \in \mathrm{V}_\exists$: $e(x) : (S_e\langle\!\{x\}\!\rangle \to \mathcal{A}) \to \mathcal{A}$. Note that, for each e, at most one semantic relation exists, namely

$$S_e \ := \ \{ \ (y, x) \mid \ y \in \mathrm{dom}(\bigcup(\mathrm{dom}(e(x)))) \wedge x \in \mathrm{V}_\exists \ \}.$$

Definition 0.2 (S_e, Existential $(\mathcal{A}, R)$-Valuation, ϵ)
Let R be a variable-condition, $\mathcal{A}$ a Σ-structure, and $e : \mathrm{V}_\exists \to ((\mathrm{V}_\forall \rightsquigarrow \mathcal{A}) \to \mathcal{A})$.
The *semantic relation of* e is $S_e := \{ \ (y, x) \mid y \in \mathrm{dom}(\bigcup(\mathrm{dom}(e(x)))) \wedge x \in \mathrm{V}_\exists \ \}$.
e is an *existential* $(\mathcal{A}, R)$*-valuation* if $S_e \circ R$ is irreflexive and, for all $x \in \mathrm{V}_\exists$,

$$e(x) : (S_e\langle\!\{x\}\!\rangle \to \mathcal{A}) \to \mathcal{A}.$$

Finally, for applying existential $(\mathcal{A}, R)$-valuations in a uniform manner, we define the function $\epsilon : \ (\mathrm{V}_\exists \to ((\mathrm{V}_\forall \rightsquigarrow \mathcal{A}) \to \mathcal{A})) \ \to \ ((\mathrm{V}_\forall \to \mathcal{A}) \to (\mathrm{V}_\exists \to \mathcal{A}))$ by $(\ e \in \mathrm{V}_\exists \to ((\mathrm{V}_\forall \rightsquigarrow \mathcal{A}) \to \mathcal{A}), \ \ \pi \in \mathrm{V}_\forall \to \mathcal{A}, \ \ x \in \mathrm{V}_\exists \ )$

$$\epsilon(e)(\pi)(x) := e(x)({}_{S_e\langle\!\{x\}\!\rangle}\!\upharpoonright\!\pi).$$

Lemma 0.3 *Let R be a variable-condition.*

1. *Let R' be a variable-condition with $R \subseteq R'$.*
   For each existential $(\mathcal{A}, R')$-valuation e' there is some existential $(\mathcal{A}, R)$-valuation e such that $\epsilon(e) = \epsilon(e')$.
2. *Let σ be an existential R-substitution and R' the σ-update of R.*
   For each existential $(\mathcal{A}, R')$-valuation e' there is some existential $(\mathcal{A}, R)$-valuation e such that for all $\pi \in \mathrm{V}_\forall \to \mathcal{A}$:
   $$\epsilon(e)(\pi) \quad = \quad \sigma \ \circ \ \mathrm{eval}(\mathcal{A} \uplus \epsilon(e')(\pi) \uplus \pi).$$

We now define R-validity of a set of sequents with free variables, in terms of validity of a formula (where the free variables are treated as nullary constants).

Definition 0.4 (Validity)
Let R be a variable-condition, $\mathcal{A}$ a Σ-structure, and G a set of sequents.
G is *R-valid in $\mathcal{A}$* if there is an existential $(\mathcal{A}, R)$-valuation e such that G is $(e, \mathcal{A})$-valid.
G is *$(e, \mathcal{A})$-valid* if G is $(\pi, e, \mathcal{A})$-valid for all $\pi \in \mathrm{V}_\forall \to \mathcal{A}$.
G is *$(\pi, e, \mathcal{A})$-valid* if G is valid in $\mathcal{A} \uplus \epsilon(e)(\pi) \uplus \pi$.
G *is valid in $\mathcal{A}$* if for all $\Gamma \in G$: Γ is valid in $\mathcal{A}$.
A *sequent Γ is valid in $\mathcal{A}$* if there is some formula listed in Γ that is valid in $\mathcal{A}$.
Validity in a class of Σ-structures is understood as validity in each of the Σ-structures of that class. If we omit the reference to a special Σ-structure we mean validity in some fixed class K of Σ-structures, e.g. the class of all Σ-structures (Σ-algebras) or the class of Herbrand Σ-structures (term-generated Σ-algebras), cf. Wirth (1997), Wirth & Gramlich (1994) for more interesting classes for establishing inductive validities.

Lemma 0.5 (Anti-Monotonicity of Validity in R)
Let G be a set of sequents and R and R' variable-conditions with $R{\subseteq}R'$. Now: If G is R'-valid in $\mathcal{A}$, then G is R-valid in $\mathcal{A}$.

Example 0.6 (Validity)
For $x^{\exists} \in \mathrm{V}_{\exists}$, $y^{\forall} \in \mathrm{V}_{\forall}$, the sequent $x^{\exists}{=}y^{\forall}$ is $\emptyset$-valid in any $\mathcal{A}$ because we can choose $S_e := \mathrm{V}_{\forall}{\times}\mathrm{V}_{\exists}$ and $e(x^{\exists})(\pi) := \pi(y^{\forall})$ resulting in $\epsilon(e)(\pi)(x^{\exists}) = e(x^{\exists})({}_{S_e\langle\!\{x^{\exists}\}\!\rangle}{\upharpoonleft}\pi) = e(x^{\exists})({}_{\mathrm{V}_{\forall}}{\upharpoonleft}\pi) = \pi(y^{\forall})$. This means that $\emptyset$-validity of $x^{\exists}{=}y^{\forall}$ is the same as validity of $\forall y.\ \exists x.\ x{=}y$. Moreover, note that $\epsilon(e)(\pi)$ has access to the π-value of $y^{\forall}$ just as a raising function f for x in the raised (i.e. dually Skolemized) version $f(y^{\forall}){=}y^{\forall}$ of $\forall y.\ \exists x.\ x{=}y$.

Contrary to this, for $R := \mathrm{V}_{\exists}{\times}\mathrm{V}_{\forall}$, the same formula $x^{\exists}{=}y^{\forall}$ is not R-valid in general because then the required irreflexivity of $S_e{\circ}R$ implies $S_e{=}\emptyset$ and $e(x^{\exists})({}_{S_e\langle\!\{x^{\exists}\}\!\rangle}{\upharpoonleft}\pi) = e(x^{\exists})({}_{\emptyset}{\upharpoonleft}\pi) = e(x^{\exists})(\emptyset)$ cannot depend on $\pi(y^{\forall})$ anymore. This means that $(\mathrm{V}_{\exists}{\times}\mathrm{V}_{\forall})$-validity of $x^{\exists}{=}y^{\forall}$ is the same as validity of $\exists x.\ \forall y.\ x{=}y$. Moreover, note that $\epsilon(e)(\pi)$ has no access to the π-value of $y^{\forall}$ just as a raising function c for x in the raised version $c{=}y^{\forall}$ of $\exists x.\ \forall y.\ x{=}y$.

For a more general example let $G = \{\ A_{i,0} \ldots A_{i,n_i-1} \mid i{\in}I\ \}$, where for $j{\prec}n_i$ and $i \in I$ the $A_{i,j}$ are formulas with free existential variables from $\vec{x}$ and free universal variables from $\vec{y}$. Then $(\mathrm{V}_{\exists}{\times}\mathrm{V}_{\forall})$-validity of G means validity of $\exists\vec{x}.\ \forall\vec{y}.\ \forall i{\in}I.\ \exists j{\prec}n_i.\ A_{i,j}$; whereas $\emptyset$-validity of G means validity of $\forall\vec{y}.\ \exists\vec{x}.\ \forall i{\in}I.\ \exists j{\prec}n_i.\ A_{i,j}$.

1 Weights, Syntactic Constructs, and Counterexamples

A proposition Γ can be proved by induction as follows:

> *Show that for each counterexample of Γ there is another counterexample of Γ that is strictly smaller in a quasi-ordering $\lesssim$ in that each $\lesssim$-chain [of counterexamples] has a least element!*

Now by the Principle of Dependent Choice (cf. Rubin & Rubin (1985)) a class without minimal elements contains a chain without a least element. Thus, if we can show the above, we know that Γ cannot have any counterexamples at all and must be valid.

This paradigm of ITP was already used by the ancient Greeks, rediscovered by Pierre de Fermat under the name "descente infinie", and is in our time sometimes called "implicit induction", cf. Wirth & Becker (1995). Moreover note that theoretically it is also possible to use the strictly stronger Axiom of Choice instead but require only that each $\lesssim$-chain [of counterexamples] has a lower bound [being a counterexample], cf. Geser (1995).

In order to measure our sequents in our *induction ordering* (i.e. the above quasi-ordering for avoiding non-termination in the inductive argumentation), we supply them with weights from some set 'Weight'. Why these explicit weights are so important in implicit induction is explained in Wirth & Becker (1995) and more detailed in Wirth (1997), Sect. 12. A weight $\aleph$ together with a sequent Γ forms a *syntactic construct* $(\Gamma, \aleph)$. For practical purposes a weight initially is a term $w(y_0, \ldots, y_{n-1})$ where the y_i are the free universal variables of $\mathit{\Gamma}$ and w is similar to a global (rigid) free existential variable in that it can be chosen during the induction proof appropriately, e.g. when the goal is $w(t_1, t_2) < w(t_2, \mathsf{s}(t_1))$ for natural number terms t_i a good idea might be to choose w to be the addition on natural numbers, or when in another proof we have the goals $w(t_1, (t_1 + t_2)) < w(\mathsf{s}(t_1), t_1)$ and $w(t_1, t_2) < w(t_1, \mathsf{s}(t_2))$ a good idea might be to choose w to be the lexicographic combination of length up to 2. While in principle also the induction ordering could be chosen for each proof differently, in practice it has shown to be sufficient and adequate to use a fixed wellfounded quasi-ordering (depending on the Σ-structures). E.g. in QuodLibet, a tactic-based ITP system for clausal logic, we essentially use the size of a uniquely denoting constructor ground term in the standard ordering on natural numbers, cf. Kühler (1999). Therefore, we assume that for each Σ-structure $\mathcal{A} \in \mathrm{K}$ there is some wellfounded quasi-ordering $\lesssim_{\mathcal{A}}$. Furthermore, we assume that Σ contains the binary constant predicate symbol '$<$' which is interpreted by $\mathcal{A}$ as $<_{\mathcal{A}}$, i.e. the ordering of $\lesssim_{\mathcal{A}}$. Furthermore, $\mathcal{A}$ should be able to interpret the functions constructing the weight terms (lexicographic combination &c.) as functions into $\mathrm{dom}(\lesssim_{\mathcal{A}})$.

Syntactic constructs are the basic data structure for ITP, just like sequents or formulas are for the deductive case. The set of all syntactic constructs is denoted by 'SynCons'. The function 'logic' extracts the *logic part* (here: the sequents) of a set G of syntactic constructs: $\mathrm{logic}(G) := \mathrm{dom}(G)$.

For powerful ITP we have to be able to restrict the test of whether the weight of a hypothesis is smaller than the weight of a goal (which has to be satisfied for the permission to apply the hypothesis to the goal) to the special case semantically described by their logic parts. This can be achieved by considering only such instances of their weights that result from those valuations that describe invalid instances of their logic parts. A syntactic construct augmented with such a valuation providing extra information on the invalidity of its logic part in some Σ-structure $\mathcal{A}$ is called a "counterexample". More precisely, for an existential $(\mathcal{A}, R)$-valuation e we define: (S, π) is an $(e, \mathcal{A})$-*counterexample (for S)* if S is a syntactic construct, $\pi \in \mathrm{V}_{\forall} \to \mathcal{A}$, and $\mathrm{logic}(\{S\})$ is not $(\pi, e, \mathcal{A})$-valid. Thus, the logic part of a syntactic construct S is $(e, \mathcal{A})$-valid iff S has no $(e, \mathcal{A})$-counterexamples. Furthermore, our induction ordering is not simply a wellfounded quasi-ordering on 'Weight' but actually the function mapping each $(e, \mathcal{A})$, where $\mathcal{A}$ is a Σ-structure from K and e an existential $(\mathcal{A}, R)$-valuation, to the wellfounded quasi-ordering on $\mathrm{Weight} \times (\mathrm{V}_{\forall} \to \mathcal{A})$ given by $(\aleph, \pi) \lesssim_{(e,\mathcal{A})} (\beth, \pi')$ if $\mathrm{eval}(\mathcal{A} \uplus \epsilon(e)(\pi) \uplus \pi)(\aleph) \lesssim_{\mathcal{A}} \mathrm{eval}(\mathcal{A} \uplus \epsilon(e)(\pi') \uplus \pi')(\beth)$. Finally, we extend the induction ordering to $\mathrm{SynCons} \times (\mathrm{V}_{\forall} \to \mathcal{A})$ by defining

$$((\mathit{\Gamma}, \aleph), \pi) \lesssim_{(e,\mathcal{A})} ((\mathit{\Delta}, \beth), \pi') \quad \text{if} \quad (\aleph, \pi) \lesssim_{(e,\mathcal{A})} (\beth, \pi').$$

Note that our induction ordering $\lesssim_{(e,\mathcal{A})}$ is semantic in the sense of Definition 13.7 of Wirth (1997) because it cannot depend on the syntactic term structure of a weight $\aleph$ but only on the value of $\aleph$ under the evaluation function. In Wirth (1997) we have rigorously investigated the price one has to pay for the possibility to have induction orderings also depending on the syntax of weights. For powerful concrete inference systems this price is surprisingly high. Furthermore, after improving the ordering information in implicit induction by our introduction of explicit weights, the former necessity of sophisticated induction orderings that exploit the term structure does not seem to exist anymore.

Definition 1.1 (Foundedness) *(Cf. Wirth & Becker (1995))*
Let R be a variable-condition. Let '$\searrow\!/\!\curvearrowright_R$' be a symbol for a single relation. Let $G_0, G_1, H \subseteq \text{SynCons}$. Now G_0 is *R-strict/quasi-founded* on (H, G_1) (denoted by $G_0 \searrow\!/\!\curvearrowright_R (H, G_1)$) if
$\forall \mathcal{A} \in \mathrm{K}.\ \forall e$ existential $(\mathcal{A}, R)$-valuation. $\forall S \in G_0.\ \forall \pi.$

$$\left(\begin{array}{l} ((S,\pi) \text{ is an } (e,\mathcal{A})\text{-counterexample}) \quad \Rightarrow \\ \exists S'.\ \exists \pi'. \left(\begin{array}{l} \qquad ((S',\pi') \text{ is an } (e,\mathcal{A})\text{-counterexample}) \\ \wedge \left(\begin{array}{l} \left(\begin{array}{l} S' \in H \\ \wedge\ (S',\pi') <_{(e,\mathcal{A})} (S,\pi) \end{array}\right) \\ \vee \left(\begin{array}{l} S' \in G_1 \\ \wedge\ (S',\pi') \precsim_{(e,\mathcal{A})} (S,\pi) \end{array}\right) \end{array}\right) \end{array}\right) \end{array}\right)$$

G_0 is *strictly R-founded on H* (denoted by $G_0 \searrow_R H$) if $G_0 \searrow\!/\!\curvearrowright_R (H, \emptyset)$.
G_0 is (quasi-) *R-founded on G_1* (denoted by $G_0 \curvearrowright_R G_1$) if $G_0 \searrow\!/\!\curvearrowright_R (\emptyset, G_1)$.

Note that the expressive power of $\searrow\!/\!\curvearrowright_R$ is higher than that of $\searrow_R$ and $\curvearrowright_R$ together: $(\{S\} \searrow_R H \vee \{S\} \curvearrowright_R G_1)$ implies $\{S\} \searrow\!/\!\curvearrowright_R (H, G_1)$ for $S \in$ SynCons, but the converse does not hold in general. For an informal but quite imaginative introduction to foundedness cf. Wirth (1997).

Lemma 1.2
Let R, R' be variable-conditions; $G_0, G_1, G_2, G_3, H_1, H_2, H_3 \subseteq \text{SynCons}$.

1. *If* $G_0 \curvearrowright_R G_1$ *and* $\text{logic}(G_1)$ *is R-valid, then* $\text{logic}(G_0)$ *is R-valid, too.*
2. *If* $G_0 \subseteq G_1$, *then* $G_0 \curvearrowright_R G_1$.
3. *If* $G_0 \curvearrowright_R G_1 \searrow\!/\!\curvearrowright_R (H_2, G_2)$, *then* $G_0 \searrow\!/\!\curvearrowright_R (H_2, G_2)$.
4. *If* $G_0 \searrow\!/\!\curvearrowright_R (H_2, G_2)$ *and* $G_1 \searrow\!/\!\curvearrowright_R (H_3, G_3)$,
   then $G_0 \cup G_1 \searrow\!/\!\curvearrowright_R (H_2 \cup H_3, G_2 \cup G_3)$.
5. *In case of* $R \subseteq R'$: *If* $G_0 \curvearrowright_R G_1$, *then* $G_0 \curvearrowright_{R'} G_1$.
6. *In case of R' being the σ-update of R for an existential R-substitution σ:*
   If $G_0 \curvearrowright_R G_1$, *then* $G_0\sigma \curvearrowright_{R'} G_1\sigma$.
7. *If* $H_1 \searrow\!/\!\curvearrowright_R (H_1, G_1)$, *then* $H_1 \curvearrowright_R G_1$.

Proof of Lemma 1.2: (1), (2), (3), and (4) are trivial. (5) follows from part (1) of Lemma 0.3, (6) follows from part (2) of Lemma 0.3. (7) relies on the well-foundedness of $\precsim_{(e,\mathcal{A})}$. □

2 Abstract Inductive Calculi

Now we are going to abstractly describe inductive sequent and tableau calculi. In Wirth (1998) we have shown that the usual deductive first-order calculi are instances of the deductive version of our abstract calculi. This will still be the case for the abstract calculi below, but in this paper we have to concentrate on the inductive part. The main difference to the deductive case is that the sequents are replaced with syntactic constructs, i.e. to each sequent a weight is added for controlling the loops in ITP. The benefit of the abstract version is that every instance is automatically sound. Due to the small number of inference rules in deductive first-order calculi and the locality of soundness, this abstract version is not really necessary for deductive calculi. For inductive calculi, however, due to a bigger number of inference rules (which usually have to be improved now and then) and the globality of soundness, such an abstract version is very helpful, cf. Wirth & Becker (1995), Wirth (1997).

Definition 2.1 (Inductive Proof Forest)
An *inductive proof forest in a sequent* (or else: *tableau*) *calculus* is a pair (F, R) where R is a variable-condition and F is a set of pairs (S, t), where S is a syntactic construct and t is a tree whose nodes are labeled with syntactic constructs (or else: whose root is labeled with a weight and whose other nodes are labeled with formulas).

Note that the tree t is intended to represent a proof attempt for S. In case of a tableau calculus the nodes of t are labeled with formulas (the root, however, with a weight). In case of a sequent calculus the nodes are labeled with syntactic constructs. While the syntactic constructs at the nodes of a tree in a sequent calculus stand for themselves, in a tableau calculus all the ancestors have to be included to make up a syntactic construct and, moreover, the formulas at the labels are in negated form:

Definition 2.2 (Goals(), $\mathcal{AX}$, Closedness)
'Goals(T)' denotes the set of syntactic constructs labeling the leaves of the trees in the set T (or else: the set of syntactic constructs $(\Delta, \sqsupset)$ with Δ resulting from listing the conjugates of the formulas labeling a branch from a leaf to the root (exclusively) in a tree t in T and $\sqsupset$ being the label of the root of the tree t).
In what follows, we assume $\mathcal{AX}$ to be some set of *axioms*. By this we mean that $\mathcal{AX}$ is $\mathrm{V}_\exists \times \mathrm{V}_\forall$-valid. (Cf. the last sentence in Def. 0.4.)
The tree t is *closed* if $\mathrm{logic}(\mathrm{Goals}(\{t\})) \subseteq \mathcal{AX}$.

The readers may ask themselves why we consider a forest instead of a single tree only. If we have two trees (S, t), $(S', t') \in F$ we can apply S as a lemma or induction hypothesis in the tree t' of S', provided that the lemma application relation is acyclic and the application of the hypothesis produces a goal that expresses that the instance of the hypothesis is smaller than S' in the induction ordering.

Definition 2.3 (Inductive Invariant Condition)
The *inductive invariant condition* on (F, R) is $H \curvearrowright_R \mathrm{Goals}(\mathrm{ran}(F))$ for the set of hypotheses $H := \mathrm{dom}(F)$.

From Lemma 0.5 and Lemma 1.2(1) we immediately get:

Theorem 2.4 *Let the inductive proof forest (F, R) satisfy the above inductive invariant condition and set $H := \mathrm{dom}(F)$.*
If all trees in $\mathrm{ran}(F)$ are closed, then $\mathrm{logic}(H)$ is R-valid.

Note that, contrary to the deductive case, local argumentation on a single tree is not possible: If $(S,t) \in F$, (F,R) satisfies the inductive invariant condition, and t is closed, we do not know that logic$(\{S\})$ is R-valid because we may have applied some induction hypothesis $S' \in H \backslash \{S\}$ when constructing the proof tree t of S and the proof tree of S' is not closed. In other words, all trees in the forest must be closed before we know that any hypothesis in logic(H) is R-valid.

Theorem 2.5 *The inductive invariant condition is always satisfied when we start with an empty inductive proof forest $(F,R) := (\emptyset,\emptyset)$ and then iterate only the following kinds of modifications of (F,R) (resulting in (F',R')):*

Hypothesizing: *Let R' be a variable-condition with $R \subseteq R'$. Let Let $(\Gamma,\aleph)$ be a syntactic construct. Let t be the tree with a single node only, which is labeled with $(\Gamma,\aleph)$ (or else: with a single branch only, such that Γ is the list of the conjugates of the formulas labeling the branch from the leaf to the root (exclusively) and $\aleph$ is the label of the root). Then we may set $F' := F \cup \{((\Gamma,\aleph),t)\}$.*

Expansion: *Let R' be a variable-condition with $R \subseteq R'$. Let $(S,t) \in F$ and $H := \mathrm{dom}(F)$. Let l be a leaf in t. Let $(\Delta,\beth)$ be the label of l (or else: Δ result from listing the conjugates of the formulas labeling the branch from l to the root (exclusively) and $\beth$ be the label of the root of t). Let G be a finite set of syntactic constructs (or else: let M be a finite set of sequents and set $G := \{\ (\Pi\Delta,\beth) \mid \Pi \in M\ \}$). Now if $\{(\Delta,\beth)\} \searrow\!/\!\curvearrowright_{R'} (H,G)$ then we may set $F' := (F \backslash \{(S,t)\}) \cup \{(S,t')\}$ where t' results from t by adding to the former leaf l, exactly for each syntactic construct S' in G, a new child node labeled with S' (or else: exactly for each sequent Π in M a new child branch such that Π is the list of the conjugates of the formulas labeling the branch from the leaf to the new child node of l).*

Instantiation: *Let σ be an existential R-substitution. Let R' be the σ-update of R. Then we may set $F' := F\sigma$.*

Proof of Theor. 2.5: The empty proof forest satisfies the inductive invariant condition by Lemma 1.2(2). <u>Hypothesizing:</u> From $H \curvearrowright_R \mathrm{Goals}(\mathrm{ran}(F))$ we get $H \curvearrowright_{R'} \mathrm{Goals}(\mathrm{ran}(F))$ by Lemma 1.2(5) for $H := \mathrm{dom}(F)$. Thus, from $\{(\Gamma,\aleph)\} \curvearrowright_{R'} \{(\Gamma,\aleph)\}$ (by Lemma 1.2(2)) we get $H \cup \{(\Gamma,\aleph)\} \curvearrowright_{R'} \mathrm{Goals}(\mathrm{ran}(F)) \cup \{(\Gamma,\aleph)\}$ by Lemma 1.2(4), i.e. $H' \curvearrowright_{R'} \mathrm{Goals}(\mathrm{ran}(F'))$ for $H' := \mathrm{dom}(F')$. <u>Expansion:</u> $\{(\Delta,\beth)\} \searrow\!/\!\curvearrowright_{R'} (H,G)$ and $(\mathrm{Goals}(\mathrm{ran}(F))) \setminus \{(\Delta,\beth)\} \curvearrowright_{R'} (\mathrm{Goals}(\mathrm{ran}(F))) \setminus \{(\Delta,\beth)\}$ (by Lemma 1.2(2)) give $\mathrm{Goals}(\mathrm{ran}(F)) \searrow\!/\!\curvearrowright_{R'} (H,G')$ for $G' := (\mathrm{Goals}(\mathrm{ran}(F)) \backslash \{(\Delta,\beth)\}) \cup G$ by Lemma 1.2(4). From the old invariant condition $H \curvearrowright_R \mathrm{Goals}(\mathrm{ran}(F))$ we get $H \curvearrowright_{R'} \mathrm{Goals}(\mathrm{ran}(F))$ by Lemma 1.2(5), and then by Lemma 1.2(3) conclude $H \searrow\!/\!\curvearrowright_{R'} (H,G')$. Thus, by Lemma 1.2(7) we have $H \curvearrowright_{R'} G'$, and then from $G' \curvearrowright_{R'} \mathrm{Goals}(\mathrm{ran}(F'))$ (due to Lemma 1.2(2)) we get $H \curvearrowright_{R'} \mathrm{Goals}(\mathrm{ran}(F'))$ due to Lemma 1.2(3), which is the new invariant condition on (F',R') due to $H = \mathrm{dom}(F) = \mathrm{dom}(F')$. <u>Instantiation:</u> From the old invariant condition $H \curvearrowright_R \mathrm{Goals}(\mathrm{ran}(F))$ for $H := \mathrm{dom}(F)$ we get $H\sigma \curvearrowright_{R'} \mathrm{Goals}(\mathrm{ran}(F\sigma))$ by Lemma 1.2(6), which is the new invariant condition on (F',R') due to $\mathrm{dom}(F') = \mathrm{dom}(F)\sigma = H\sigma$. $\square$

Now the crucial step of implicit induction (i.e. the application of an induction hypothesis $(\Gamma, \aleph)$ instantiated with a substitution ϱ to expand a goal $(\Delta, \beth)$) can be formulated as an Expansion step in the tableau calculus (sequent calculus analogously) as follows.

Theorem 2.6 *Let (F, R) be an inductive proof forest in a tableau calculus. We want to apply $(\Gamma, \aleph) \in H := \mathrm{dom}(F)$ as an induction hypothesis to expand the goal $(\Delta, \beth)$ that results from listing the conjugates of the formulas and the weight labeling the branch from a leaf l to the root of a tree $t \in \mathrm{ran}(F)$. Set $\mathrm{X} := \mathcal{V}_\exists((\Gamma, \aleph))$ and $\mathrm{Y} := \{\, y \in \mathcal{V}_\forall((\Gamma, \aleph)) \mid \mathrm{X} \times \{y\} \subseteq R \,\}$. Let $\varrho \in \mathrm{Y} \to (\mathrm{V}_\exists \setminus \mathcal{V}_\exists(F))$ be an injective substitution. Now the following instantiation of R' and M describes a sub-rule of the Expansion rule of the tableau calculus of Theor. 2.5: Set $R' := R$ and set M to be the set containing the sequents $\aleph\varrho {<} \beth$ and $\overline{B\varrho}$ for each formula B listed in the sequent Γ.*

Note that Y contains exactly those free universal variables of $(\Gamma, \aleph)$ that have no free existential variables in their scopes when imagining any list of quantifiers for all free variables of $(\Gamma, \aleph)$ that represents (a superset of) R. The variables in Y are those on which no solution for the free existential variables in X depends. Therefore, the variables in Y are those which we can instantiate when applying the induction hypothesis $(\Gamma, \aleph)$. Although it does not seem impossible to use more variables for induction, this does not seem to be necessary; especially because we can extend R with $\mathrm{X} \times \{y\}$ in order to instantiate y when applying the induction hypothesis. Moreover, I do not known any more general approach in the literature. E.g., in Baaz &al. (1997), the inductive part of theorem proving is triggered by application of a δ-rule and the variable y of the quantifier removed by the δ-rule becomes the induction variable. In our approach, the δ-rule application would replace y with a new free universal variable $y^\forall$ and extend the variable-condition with $\mathrm{X} \times \{y^\forall\}$ such that $y^\forall \in \mathrm{Y}$ would hold.

Proof of Theor. 2.6: According to the definition of an Expansion step in the tableau calculus of Theor. 2.5 we have to show $\{(\Delta, \beth)\} \searrow\!/\!\curvearrowright_R (H, \{\, (A\Delta, \beth) \mid A \in M \,\})$. Thus, for $\mathcal{A} \in \mathrm{K}$, e an existential $(\mathcal{A}, R)$-valuation, $\pi \in \mathrm{V}_\forall \to \mathcal{A}$, assume $((\Delta, \beth), \pi)$ to be an $(e, \mathcal{A})$-counterexample. If some formula $A \in M$ is not $(\pi, e, \mathcal{A})$-valid then also $((A\Delta, \beth), \pi)$ is an $(e, \mathcal{A})$-counterexample with $((A\Delta, \beth), \pi) \lesssim_{(e,\mathcal{A})} ((\Delta, \beth), \pi)$. Otherwise, $\aleph\varrho {<} \beth$ is but $\Gamma\varrho$ is not $(\pi, e, \mathcal{A})$-valid. Define $\pi' \in \mathrm{V}_\forall \to \mathcal{A}$ by

$$\pi'(y) := \epsilon(e)(\pi)(\varrho(y)) \text{ for } y \in \mathrm{Y} \qquad \text{and} \qquad \pi'(y) := \pi(y) \text{ for } y \in \mathrm{V}_\forall \setminus \mathrm{Y}.$$

Claim 1: For $x^\exists \in \mathcal{V}_\exists((\Gamma, \aleph))$ we have $\epsilon(e)(\pi)(x^\exists) = \epsilon(e)(\pi')(x^\exists)$.

Proof of Claim 1: Otherwise there must be some $y^\forall \in \mathrm{Y}$ with $y^\forall\ S_e\ x^\exists$. Since $x^\exists \in \mathrm{X}$ we have $x^\exists\ R\ y^\forall$ by definition of Y. But then $S_e \circ R$ is not irreflexive, which contradicts e being an existential $(R, \mathcal{A})$-valuation. Q.e.d. (Claim 1)

Hence, the values of Γ and $\aleph$ under $\mathrm{eval}(\mathcal{A} \uplus \epsilon(e)(\pi') \uplus \pi')$ are the same as the values of Γ and $\aleph$ under $\mathrm{eval}(\mathcal{A} \uplus \epsilon(e)(\pi) \uplus \pi')$ which again are the same as the values of $\Gamma\varrho$ and $\aleph\varrho$ under $\mathrm{eval}(\mathcal{A} \uplus \epsilon(e)(\pi) \uplus \pi)$ by the Substitution-Lemma. Thus, on the one hand from the $(\pi, e, \mathcal{A})$-validity of $\aleph\varrho {<} \beth$ we get $((\Gamma, \aleph), \pi') <_{(e,\mathcal{A})} ((\Delta, \beth), \pi)$, and on the other hand from $\Gamma\varrho$ not being $(\pi, e, \mathcal{A})$-valid we know that $((\Gamma, \aleph), \pi')$ is an $(e, \mathcal{A})$-counterexample. $\square$

3 An Example

Due to limited space we are not able to show the usefulness of our integration of free existential variables and full first-order formulas into implicit induction with a sophisticated example. Instead, we will sketch a simplified toy example with mutual induction that will give the reader a concrete idea on how proofs look like. Note, however, that (due to mutual induction and non-trivial weights) even this toy example has no straightforward proofs in the ITP calculus of Baaz &al. (1997) or any known ITP system with the exception of QuodLibet, cf. Kühler (1999).

In order not to require even more prerequisites, we do not explicitly refer to our inductive specification techniques described in Kühler & Wirth (1996), but use a standard (order-sorted) first-order specification style.

Signature: Sorts: $\mathsf{nat}\subseteq\mathsf{ORD}$. Here nat is the sort of natural numbers and ORD the sort for the induction ordering. We use zero $\mathsf{0} :\to \mathsf{nat}$ and successor $\mathsf{s} : \mathsf{nat} \to \mathsf{nat}$ as constructors for the sort nat. Moreover, P. nat and Q. $\mathsf{nat},\mathsf{nat}$ are two defined predicates on the natural numbers. Furthermore, $<$. $\mathsf{ORD},\mathsf{ORD}$ is the induction ordering and $\mathsf{lex} : \mathsf{nat},\mathsf{nat},\mathsf{nat} \to \mathsf{ORD}$ is the lexicographic combination of length 0, 1, or 2 as indicated by the first argument, e.g. $\mathsf{lex}(\mathsf{s}(\mathsf{0}), x, y)$ models the 1-tuple (x) while $\mathsf{lex}(\mathsf{0}, x, y)$ models the 0-tuple or empty word (). We use x, y, z for variables of the sort nat where superscripts like $x^{\exists}$, $x^{\forall}$ indicate free existential and free universal variables. Axioms:

$$\begin{array}{lrl}
(\mathsf{lex0}) & \forall x_1, y_0, y_1, z_0, z_1. & \mathsf{lex}(\mathsf{0}, y_0, z_0) < \mathsf{lex}(\mathsf{s}(x_1), y_1, z_1) \\
(\mathsf{lex1}) & \forall x_0, x_1, y_0, y_1, z_0, z_1. & \big(\mathsf{lex}(\mathsf{s}(x_0), y_0, z_0) < \mathsf{lex}(\mathsf{s}(x_1), y_1, z_1) \Leftarrow y_0 < y_1\big) \\
(\mathsf{lex2}) & \forall x_0, x_1, y, z_0, z_1. & \left(\begin{array}{l}\mathsf{lex}(\mathsf{s}(x_0), y, z_0) < \mathsf{lex}(\mathsf{s}(x_1), y, z_1) \\ \quad \Leftarrow \mathsf{lex}(x_0, z_0, \mathsf{0}) < \mathsf{lex}(x_1, z_1, \mathsf{0})\end{array}\right) \\
(\mathsf{nat0}) & \forall x. & \big(x = \mathsf{0} \vee \exists y.\ x = \mathsf{s}(y)\big)
\end{array}$$

(lex0) says that the empty tuple is the smallest, (lex1) implements a comparison of the first tuple elements, and (lex2) discards identical first tuple elements. (nat0) says that any natural number is zero or the successor of another natural number. The following axioms define the special predicates of our example.

$$\begin{array}{lrl}
(\mathsf{P0}) & & \mathsf{P}(\mathsf{0}) \\
(\mathsf{P1}) & \forall x. & \big(\mathsf{P}(\mathsf{s}(x)) \Leftarrow \big(\mathsf{P}(x) \wedge \mathsf{Q}(x, \mathsf{s}(x))\big)\big) \\
(\mathsf{Q0}) & \forall x. & \mathsf{Q}(x, \mathsf{0}) \\
(\mathsf{Q1}) & \forall x, y. & \big(\mathsf{Q}(x, \mathsf{s}(y)) \Leftarrow \big(\mathsf{Q}(x, y) \wedge \mathsf{P}(x)\big)\big)
\end{array}$$

We want to show $\forall x.\ \mathsf{P}(x)$ and $\forall y, z.\ \mathsf{Q}(y, z)$. We first do a tableau calculus proof. We start with the empty forest. Two Hypothesizing steps provide us with the hypotheses $(\mathsf{P}(x_0^{\forall}); w_0^{\exists}(x_0^{\forall}))$ with single-branch tree (1) $w_0^{\exists}(x_0^{\forall})$, (1.1) $\neg\mathsf{P}(x_0^{\forall})$; and $(\mathsf{Q}(y_0^{\forall}, z_0^{\forall}); w_1^{\exists}(y_0^{\forall}, z_0^{\forall}))$ with single-branch tree (2) $w_1^{\exists}(y_0^{\forall}, z_0^{\forall})$, (2.1) $\neg\mathsf{Q}(y_0^{\forall}, z_0^{\forall})$. Note that the first number in the preceding list is the number of the proof tree (indicating its root node) and a suffix "$.i$" denotes the step to the i^{th} child node. Since the formulas of the specification are implicitly on all branches, we can use (nat0) to add to (1.1) the children $x_0^{\exists} = \mathsf{0}$ and $x_0^{\exists} = \mathsf{s}(x_1^{\forall})$ in an Expansion step with variable restriction $(x_0^{\exists}, x_1^{\forall})$. An Instantiation step with $\{x_0^{\exists} \mapsto x_0^{\forall}\}$ (which a concrete inference system should do immediately together with the preceding Expansion step) gives us the new tree (1.1.1) $x_0^{\forall} = \mathsf{0}$, (1.1.2) $x_0^{\forall} = \mathsf{s}(x_1^{\forall})$. Rewriting copies of (1.1) with these children in two Expansion steps yields (1.1.1.1) $\neg\mathsf{P}(\mathsf{0})$, (1.1.2.1) $\neg\mathsf{P}(\mathsf{s}(x_1^{\forall}))$. By (P0) we can close the first branch with (1.1.1.1.1) $\mathsf{P}(\mathsf{0})$. By (P1) we can add (1.1.2.1.1) $\mathsf{P}(\mathsf{s}(x_1^{\forall}))$ (closed), (1.1.2.1.2) $\neg\mathsf{P}(x_1^{\forall})$, (1.1.2.1.3) $\neg\mathsf{Q}(x_1^{\forall}, \mathsf{s}(x_1^{\forall}))$. Now, after these standard first-order tableau steps we do an induction hypothesis application step as described in the previous section. We apply the hypothesis $(\mathsf{P}(x_0^{\forall}); w_0^{\exists}(x_0^{\forall}))$ with substitution $\varrho := \{x_0^{\forall} \mapsto x_1^{\exists}\}$ to (1.1.2.1.2), resulting in the new children $\mathsf{P}(x_1^{\exists})$ and $w_0^{\exists}(x_1^{\exists}) \not< w_0^{\exists}(x_0^{\forall})$, where $w_0^{\exists}(x_0^{\forall})$ comes from the root (1) and $\not<$ is the negation of $<$. After instantiating $\{x_1^{\exists} \mapsto x_1^{\forall}\}$ we get (1.1.2.1.2.1) $\mathsf{P}(x_1^{\forall})$ (closed) and (1.1.2.1.2.2) $w_0^{\exists}(x_1^{\forall}) \not< w_0^{\exists}(x_0^{\forall})$. Note that the only difference to an Extension step in Model Elimination tableaux (cf. Baumgartner &al. (1997)) lies with the additional child (1.1.2.1.2.2), which asks us to show that the instance of the hypothesis is smaller than the weight of our proof tree. Indeed: Hypothesis application differs from the standard lemma (or axiom) application only in producing an additional $\not<$-goal. This makes hypothesis application a little more expensive than lemma application. Similarly, we apply the induction hypothesis $(\mathsf{Q}(y_0^{\forall}, z_0^{\forall}); w_1^{\exists}(y_0^{\forall}, z_0^{\forall}))$ with substitution $\varrho := \{y_0^{\forall} \mapsto y_0^{\exists}, z_0^{\forall} \mapsto z_0^{\exists}\}$ to (1.1.2.1.3), which after instantiation with $\{y_0^{\exists} \mapsto x_1^{\forall}, z_0^{\exists} \mapsto \mathsf{s}(x_1^{\forall})\}$ results in (1.1.2.1.3.1) $\mathsf{Q}(x_1^{\forall}, \mathsf{s}(x_1^{\forall}))$ (closed) and (1.1.2.1.3.2) $w_1^{\exists}(x_1^{\forall}, \mathsf{s}(x_1^{\forall})) \not< w_0^{\exists}(x_0^{\forall})$. Rewriting the leaves of the open branches in place with (1.1.2) we get (1.1.2.1.2.2) $w_0^{\exists}(x_1^{\forall}) \not< w_0^{\exists}(\mathsf{s}(x_1^{\forall}))$ and (1.1.2.1.3.2) $w_1^{\exists}(x_1^{\forall}, \mathsf{s}(x_1^{\forall})) \not< w_0^{\exists}(\mathsf{s}(x_1^{\forall}))$. Expanding the tree (2) analogously to the tree (1) we get as leaves of the open branches (2.1.2.1.2.2) $w_1^{\exists}(y_0^{\forall}, z_1^{\forall}) \not< w_1^{\exists}(y_0^{\forall}, \mathsf{s}(z_1^{\forall}))$ and (2.1.2.1.3.2) $w_0^{\exists}(y_0^{\forall}) \not< w_1^{\exists}(y_0^{\forall}, \mathsf{s}(z_1^{\forall}))$. Now both hypotheses have been applied in both trees. Next, we choose our weight functions in such a way that we can close both trees: $w_0^{\exists}(x) := \mathsf{lex}(\mathsf{s}(\mathsf{0}), x, \mathsf{0})$ and $w_1^{\exists}(y, z) := \mathsf{lex}(\mathsf{s}(\mathsf{s}(\mathsf{0})), y, z)$. Applying (lex0), (lex1), (lex2) to the resulting leaves in the standard fashion results in $x_1^{\forall} \not< \mathsf{s}(x_1^{\forall})$ and $z_1^{\forall} \not< \mathsf{s}(z_1^{\forall})$ as the leaves of the only open branches. Finally, these branches are closed by comparing the size of a uniquely denoting constructor ground term: A branch containing a literal of the form $t_0 \not< t_1$ is closed if the number of occurrences of each free variable in t_0 is not bigger than the number of occurrences of that variable in t_1, the size of t_0 is strictly smaller than the size of t_1, and t_0, t_1 are pure constructor terms. Cf. Kühler & Wirth (1996) for the notion of "constructor term" and for the models where our induction ordering is wellfounded indeed.

We may ask: Which steps in this proof were typical for ITP in the sense that their soundness relies on notions of inductive validity instead of the stronger notion of deductive (first-order) validity? Besides the four induction hypothesis applications, the final branch closure rules are typical for induction because they require that, in all models in K, the successor of each natural number is different from that natural number and each natural number is built-up from zero by a finite number of successor steps (i.e. there are neither cycles nor **Z**-chains in the models, cf. Enderton (1973)). Other steps typical for induction but not applied in this example are narrowing steps to solve equality literals. Their soundness relies on the freeness of the models in K. (Note that narrowing in ITP relies on confluence but *not* on termination of the reduction relation of the specification, cf. Wirth (1997).) Moreover, ITP often is only successful when one tries to show theorems that are more general than the ones one initially intended to show. This is because an inductive theorem is not only a task (as goal) but also a tool (as induction hypothesis) for ITP. This generalization is *unsafe* in the sense that it may transform a valid hypothesis into an invalid one (*over-generalization*). Therefore, generalization should not be modeled in Expansion steps within a tree. Instead, the generalized sequent should start a new tree (Hypothesizing step) and be later applied to the original tree as an induction hypothesis or lemma. Since a valid input theorem may result in an invalid goal due to over-generalization, the ability of an ITP system to detect invalid goals is important under a practical aspect. When all Expansion and Instantiation steps in a tree are known to be safe, the detection of an invalid goal in the tree implies invalidity of the hypothesis of this tree, which then should be completely removed from the proof forest.

Now we are going to compare the above tableau calculus proof with a corresponding sequent calculus proof of the same hypotheses. Of course, we could simply transform the tableau trees into sequent trees by bottom-up replacing the label of each node with the syntactic construct listing the conjugates of the formulas and the weight labeling the (partial) branch from this node to the root, and finally removing the root part of the tree where the nodes are ancestors of a node of the initial Hypothesizing steps (here: removing the root nodes). This, however, would mean to pay the price for sequent calculi (i.e. multiplying the number of formulas labeling each proof tree with at most nearly the depth of that tree) without using the advantages of sequent calculi. Thus, let us start again with the hypotheses $(\mathsf{P}(x_0^\forall); w_0^\exists(x_0^\forall))$ and $(\mathsf{Q}(y_0^\forall, z_0^\forall); w_1^\exists(y_0^\forall, z_0^\forall))$. The single-node tree for the former is (1) $\mathsf{P}(x_0^\forall);\ w_0^\exists(x_0^\forall)$. Note that the goal in the tree is identical to the hypothesis, contrary to the tableau version where the two differ in duality and locality. While this is not a hindrance for completely automatic ITP systems, it poses considerable practical problems in systems where user-guidance is possible: The primitive process of switching duality is a typical source of errors for human beings (or me at least). Perfectly analogous to the tableau proof we get the children $x_0^\forall \neq \mathsf{0},\ \mathsf{P}(x_0^\forall);\ w_0^\exists(x_0^\forall)$ and $x_0^\forall \neq \mathsf{s}(x_1^\forall),\ \mathsf{P}(x_0^\forall);\ w_0^\exists(x_0^\forall)$. Contrary to the tableau proof we are now able to rewrite the literal inherited from the parent node in place without copying it. Note that in tableau proofs an equality literal can be used to rewrite formulas of its offspring in place, whereas it must copy ancestor formulas beforehand down to its offspring because the ancestor is also part of other branches that do not include the equality literal. Moreover, the weight term can be rewritten as well, which again is not possible in the tableau version where the weight is at the root node. After rewriting we get $x_0^\forall \neq \mathsf{0},\ \mathsf{P}(\mathsf{0});\ w_0^\exists(\mathsf{0})$ and $x_0^\forall \neq \mathsf{s}(x_1^\forall),\ \mathsf{P}(\mathsf{s}(x_1^\forall));\ w_0^\exists(\mathsf{s}(x_1^\forall))$. Since the equality literals are in solved form for the variable $x_0^\forall$ that does not occur elsewhere in the syntactic constructs, we know that validity cannot rely on this literal. This means that we can safely remove both equality literals resulting in (1.1) $\mathsf{P}(\mathsf{0});\ w_0^\exists(\mathsf{0})$ and (1.2) $\mathsf{P}(\mathsf{s}(x_1^\forall));\ w_0^\exists(\mathsf{s}(x_1^\forall))$. Removing redundant formulas is the most important simplification step besides contextual rewriting. It seems to be impossible in tableau trees unless the redundancy of the formula is due to the ancestor nodes only, which only is the case for useless formulas that should not have been added at all. In Wirth (1997) and in QuodLibet (cf. Kühler (1999)) the Expansion from (1) into (1.1) and (1.2) is done by a single inference step applying a so-called "covering set of substitutions". Note that the present state of the sequent proof is much simpler than the corresponding state of the tableau proof. The former consists of the nodes (1.1) and (1.2) and has two formulas and one variable. The latter consists of a six node tree with five formulas and two variables. This is of practical importance because tactics for proof search are more easily confused with less concise proof state representations. The rest of the whole sequent proof is analogous to the tableau proof with the exception that all rewrite steps are omitted since there are no equality literals to rewrite with and the terms are already in normal form.

Another possibility restricted to sequent calculi is that each syntactic construct labeling a node in the trees can be applied as an induction hypothesis. We do not see a real advantage in this because splitting the tree in two above such an induction hypothesis results in a better structure of the proof forest and in more successful proofs because we can adjust the syntactic construct appropriately: Suppose we had not started a new proof tree for the hypothesis for Q but instead kept the hypothesis for Q down in the tree (1) at position (1.2.3). Several unsafe generalization steps would have been necessary before $\mathsf{Q}(x_1^\forall, \mathsf{s}(x_1^\forall)),\ \mathsf{P}(\mathsf{s}(x_1^\forall));\ w_0^\exists(\mathsf{s}(x_1^\forall))$ would have become useful as an induction hypothesis, namely removing the second formula, generalizing $\mathsf{s}(x_1^\forall)$ to a new variable, and switching to a weight that measures also this new variable. Moreover, in practice one should not apply the hypothesis for Q in the tree for P before it is obvious that the tree for Q mutually needs the hypothesis for P: Most of the time a proof for Q can be completed in a proof forest not containing the tree for P. In this case, not only the number of trees in the proof forest for Q gets smaller, but also the tree for P because $\forall y, z.\ \mathsf{Q}(y, z)$ can then be applied as a lemma and not as an induction hypothesis, which cuts off the $\not<$-branch of the proof tree of P.

4 Conclusion

We have shown how to integrate implicit ITP into first-order sequent and tableau calculi. The following aspects are novel compared to the concrete implicit induction calculus of Wirth (1997): The tableau presentation, the possibility to use full first-order formulas instead of literals only, and the important addition of free existential variables, i.e. the "dummies" of Prawitz (1960), making the major difference between the free variable calculi of Fitting (1996) and the calculi of Smullyan (1968). Contrary to Baaz &al. (1997) we really integrate *implicit* induction: When we start an inductive proof we do not restrict the applicable induction hypotheses. We can do mutual induction and invent completely new induction hypotheses, which can be full first-order sequents instead of literals only. Moreover, we can also generate induction hypotheses eagerly in the style of explicit induction, which enables goal-directedness w.r.t. induction hypotheses. All this is not possible in the calculus of Baaz &al. (1997).

Furthermore, we exemplified that although tableau calculi may save repetition of formulas, sequent calculi have substantial advantages: Rewriting of formulas in place is always possible, and we can remove formulas that are redundant w.r.t. the other formulas in a sequent. Note that formulas like (nat0) make equality omnipresent in induction and that these simplification steps are even more important in inductive than in deductive theorem proving: Not only do they play a role in the generation of appropriate induction hypotheses; they are an essential part of the failure detection process that has to compensate for over-generalization of induction hypotheses in addition to the detection of invalid input theorems. Finally, the presence of two dual versions of each hypothesis in inductive tableau calculi makes proof guidance by human users more difficult.

References

Matthias Baaz, Uwe Egly, Christian G. Fermüller (1997). *Lean Induction Principles for Tableaux.* 6^{th} TABLEAU 1997, LNAI 1227, pp. 62-75, Springer.

Leo Bachmair (1988). *Proof By Consistency in Equational Theories.* 3^{rd} IEEE symposium on Logic In Computer Sci., pp. 228-233, IEEE Press.

Peter Baumgartner, Ulrich Furbach, Frieder Stolzenburg (1997). *Computing Answers with Model Elimination.* Artificial Intelligence **90**, pp. 135-176.

Herbert B. Enderton (1973). *A Mathematical Introduction to Logic.* Academic Press.

Melvin C. Fitting (1996). *First-Order Logic and Automated Theorem Proving.* 2^{nd} extd. ed., Springer.

Gerhard Gentzen (1935). *Untersuchungen über das logische Schließen.* Mathematische Zeitschrift **39**, pp. 176-210, 405-431.

Alfons Geser (1995). *A Principle of Non-Wellfounded Induction.* In: Tiziana Margaria (ed.): Kolloquium Programmiersprachen und Grundlagen der Programmierung, MIP-9519, pp. 117-124, Univ. Passau.

Ulrich Kühler (1999). *A Tactic-Based Inductive Theorem Prover for Data Types with Partial Operations.* Dissertation (Ph.D. thesis), Univ. Kaiserslautern, to appear.

Ulrich Kühler, Claus-Peter Wirth (1996). *Conditional Equational Specifications of Data Types with Partial Operations for Inductive Theorem Proving.* SEKI-Report SR–96–11, Univ. Kaiserslautern. Short version in: 8^{th} RTA 1997, LNCS 1232, pp. 38-52, Springer.

Peter Padawitz (1996). *Inductive Theorem Proving for Design Specifications.* J. Symbolic Computation **21**, pp. 41-99, Academic Press.

Dag Prawitz (1960). *An Improved Proof Procedure.* In: Siekmann & Wrightson (1983), Vol. 1, pp. 159-199.

Herman Rubin, Jean E. Rubin (1985). *Equivalents of the Axiom of Choice.* Elsevier.

Jörg H. Siekmann, G. Wrightson (eds.) (1983). *Automation of Reasoning.* Springer.

Raymond M. Smullyan (1968). *First-Order Logic.* Springer.

Claus-Peter Wirth (1997). *Positive/Negative-Conditional Equations: A Constructor-Based Framework for Specification and Inductive Theorem Proving.* Dissertation (Ph.D. thesis), Verlag Dr. Kovač, Hamburg.

Claus-Peter Wirth (1998). *Full First-Order Sequent and Tableau Calculi With Preservation of Solutions and the Liberalized δ-Rule but Without Skolemization.* Report 698/1998, FB Informatik, Univ. Dortmund. Short version in: Gernot Salzer, Ricardo Caferra (eds.). Proc. 2^{nd} Int. Workshop on First-Order Theorem Proving (FTP), pp. 244-255, Vienna, 1998. Also accepted for LNCS version, Springer, 1999.

Claus-Peter Wirth, Klaus Becker (1995). *Abstract Notions and Inference Systems for Proofs by Mathematical Induction.* 4^{th} CTRS 1994, LNCS 968, pp. 353-373, Springer.

Claus-Peter Wirth, Bernhard Gramlich (1994). *On Notions of Inductive Validity for First-Order Equational Clauses.* 12^{th} CADE 1994, LNAI 814, pp. 162-176, Springer.

WWW-home-page: http://ags.uni-sb.de/~cp/welcome.html

Acknowledgements: I would like to thank Ulrich Furbach and his whole group for all they taught me on tableau calculi and Paul Howard for a short private E-mail communication on the Principle of Dependent Choice that was very helpful to me.