Induction on Failing Derivations
Technical Report PL-Sep13
September 2013, with addenda from Spring 2016

Jay Ligatti
Department of Computer Science and Engineering
University of South Florida

Abstract
A proof technique, called induction on failing derivations, is introduced. We wish to prove properties of judgments in deductive systems. Standard techniques exist for proving such properties on valid judgments; this note defines a technique for proving such properties on invalid (underivable) judg-
ments.

The Technique
Standard induction on derivations is a form of tree induction; to prove that some property \( P \) holds on all roots of derivation trees (i.e., valid judgments), one may prove that \( P \) holds on all leaves of the trees (base cases) and on the internal nodes (inductive cases). When proving that \( P \) holds on the internal nodes, one may inductively assume that the property holds on all the children (i.e., subtrees) of that node, each of which must also be a valid derivation tree.

Dually, let’s consider failing derivation trees, which are finite trees in which a derivation “gets stuck” at one or more points. These points of failure are leaves in failing derivation trees. For example, trying to derive the invalid judgment

\[
(real \rightarrow real) \rightarrow (nat \rightarrow real) \leq (real \rightarrow nat) \rightarrow (real \rightarrow real)
\]

using the standard inference rules for subtyping produces the following failing derivation.

\[
\begin{align*}
\text{real} \leq \text{real} & \quad \text{nat} \leq \text{real} & \quad \text{real} \leq \text{nat} & \quad \text{real} \leq \text{real} \\
\downarrow \text{ Derivation fails here } \downarrow & \\
\text{real} \rightarrow \text{nat} \leq \text{real} \rightarrow \text{real} & \quad \text{real} \leq \text{nat} & \quad \text{nat} \rightarrow \text{real} \leq \text{real} \rightarrow \text{real} \\
\hline
\text{(real} \rightarrow \text{real}) \rightarrow \text{(nat} \rightarrow \text{real}) \leq \text{(real} \rightarrow \text{nat}) \rightarrow \text{(real} \rightarrow \text{real})
\end{align*}
\]

In the standard subtyping system for types \( \tau \) that may be \text{nat}, \text{real}, or function types \( \tau_1 \rightarrow \tau_2 \), every underivable \( \tau \leq \tau' \) judgment has a finite failing derivation,
in which at least one leaf judgment in the derivation tree fails (because it can’t be
the conclusion of any inference rule).

Notice that all judgments between a failing leaf and the root of the deriv-
ation tree must also be failing. Proofs by induction on failing derivations trace the
failure along this path, showing that some property holds on every (underivable)
judgment along the way. More specifically, proofs by induction on failing deriv-
a tions show as base cases that the property of interest holds on all possible failing
leaf judgments and then, while inductively assuming that the property holds on
the failing premise(s) of a failing internal judgment $J$, show that the property
holds on $J$ as well.

Notice also that the inductive hypothesis with regular proofs by induction
on derivations applies \textit{universally} to all premises of the internal judgment being
considered, because all its premises must be derivable. In contrast, the inductive
hypothesis with proofs by induction on failing derivations applies \textit{existentially}
to one or more premises of the internal judgment being considered, because at least
one of its premises must be underivable.

As an example, let’s consider proving a property $P$ on underivable $\tau \leq \tau'$ judg-
ments by induction on failing derivations (where again, types $\tau$ may be $\texttt{nat}$, $\texttt{real}$,
or $\tau_1 \to \tau_2$). The leaf nodes in a failing $\tau \leq \tau'$ derivation can only occur when
$\tau = \texttt{real}$ and $\tau' = \texttt{nat}$, or when exactly one of $\tau$ and $\tau'$ is a function type; hence,
the base cases of the proof must show that $P$ holds on all such judgments. The
inductive case occurs when subtyping function types; all internal nodes in failing
derivation trees must be attempts to subtype function types. Hence, the proof
must show, while inductively assuming that $P$ holds on the rule’s failing premise
(i.e., subtyping the argument or return type), that $P$ also holds on the failing
conclusion.

\textbf{Handling Nondeterministic Rules (added in February 2016)}

The above description holds for standard inference rules of the form

$$
\frac{J_1 \ldots J_n}{J},
$$

which can be written

$$
\frac{J_1 \land J_2 \ldots \land J_n}{J}.
$$

With this form of rule, standard proofs by induction on derivations may assume
the inductive hypothesis holds universally on all the premises (e.g., for this rule
to be used to derive $J$, all of $J_1$ to $J_n$ must also be derivable), while proofs
by induction on failing derivations (i.e., refutations) may assume the inductive
hypothesis holds existentially on at least one of the premises (e.g., for this rule
to be used to refute $J$, at least one of $J_1$ to $J_n$ must also be refutable). An
induction on failing derivations would, for this rule, need to show $n$ cases, the $i^{th}$
case assuming $J_i$ is underivable.
Now consider rules of the form

\[ \frac{J_1 \lor J_2 \ldots \lor J_n}{J} \]

With this form of rule, standard proofs by induction on derivations may assume the inductive hypothesis holds existentially on at least one of the premises (e.g., for this rule to be used to derive \( J \), at least one of \( J_1 \) to \( J_n \) must also be derivable), while proofs by induction on failing derivations may assume the inductive hypothesis holds universally on all the premises (e.g., for this rule to be used to refute \( J \), all of \( J_1 \) to \( J_n \) must also be refutable). An induction on failing derivations would, for this rule, need to show only one case, in which all the \( J_i \) premises are assumed underivable.

Rules of the form

\[ \frac{J_1 \lor J_2 \ldots \lor J_n}{J} \]

are often written as the nondeterministic rules

\[ \frac{J_1}{J} \frac{J_2}{J} \ldots \frac{J_n}{J} \]

That is, there are \( n \) ways to derive \( J \) in this case. However, the proof techniques must treat these \( n \) rules the same as if they were written as a single rule with disjuncted premises. Inductive hypotheses—and whether they apply universally or existentially—are the same regardless of whether rules with disjuncted premises are separated into nondeterministic rules.

For example, suppose there are two rules for deriving \( J \):

\[ \frac{J_1 \ J_2}{J} \frac{J_3 \ J_4}{J} \]

The proof techniques in this case must work the same as if the rules were written

\[ \frac{(J_1 \land J_2) \lor (J_3 \land J_4)}{J} \]

Standard proofs by induction on derivations would prove that some property \( P \) holds on derivable \( J \) by inductively assuming that \( P \) holds on both \( J_1 \) and \( J_2 \), or that \( P \) holds on both \( J_3 \) and \( J_4 \). That is, two cases need to be proved, one case assuming \( P \) holds on \( J_1 \) and \( J_2 \), and another assuming \( P \) holds on both \( J_3 \) and \( J_4 \). On the other hand, proofs by induction on failing derivations would prove that some property \( P \) holds on underivable \( J \) by inductively assuming that \( P \) holds on \( J_1 \) or \( J_2 \), and that \( P \) holds on \( J_3 \) or \( J_4 \). That is, four cases need to be proved, one case assuming \( P \) holds on \( J_1 \) and \( J_2 \), another assuming \( P \) holds on \( J_1 \) and \( J_4 \), another assuming \( P \) holds on \( J_2 \) and \( J_3 \), and another assuming \( P \) holds on \( J_2 \) and \( J_4 \). Some of these cases may be combined; for example, proving that \( P \) holds on \( J \) assuming only that \( P \) holds on \( J_1 \) would satisfy two of the four cases.

In summary, proofs by induction on failing derivations may assume that underivable judgments \( J \) are underivable using any rules, and the inductive hypothesis applies to all rules for concluding \( J \).
A Formalization of the Technique (added in March 2016)

Let’s call a deductive system navigable when its inference rules satisfy two properties:

1. Every rule’s conclusion completely determines its premises. Formally, there exists a total, computable function $f$—which we call a rule function—from any judgment $J$ to a disjunctive normal form (DNF) of judgments, such that

   $$ f(J) = (J_1^1 \land \ldots \land J_{n_1}^1) \lor \ldots \lor (J_1^m \land \ldots \land J_{n_m}^m) \quad (m, n_1, \ldots, n_m \in \mathbb{N}) $$

   means $J$ is derivable iff

   - $J_1^1 \ldots J_{n_1}^1$ are all derivable (using a rule $J_1^1 \ldots J_{n_1}^1$), or
   - $J_2^2 \ldots J_{n_2}^2$ are all derivable (using a different rule $J_2^2 \ldots J_{n_2}^2$), or
   - etc.

   Empty conjunctive clauses in $f(J)$ (i.e., when $n_i=0$) are written as () and specify that $J$ may be concluded using a premiseless rule. An empty disjunctive clause for $f(J)$ (i.e., when $m=0$) is written as $\varepsilon$ and specifies that no rule concludes $J$.

2. Infinite descent into premises is impossible; attempts to derive judgments always terminate. Formally, the relation $\{(J_p, J_c) \mid J_p \in f(J_c)\}$, which relates premise judgments to conclusion judgments, is well-founded.

The set of derivable judgments in a navigable system is plainly decidable.

For example, the standard subtyping system for the lambda calculus used in the example above is navigable:

1. On inputs $\text{nat} \leq \text{real}$, $\text{nat} \leq \text{nat}$, and $\text{real} \leq \text{real}$, rule function $f$ returns $()$, a trivially satisfiable DNF; on inputs of the form $\tau_1 \rightarrow \tau_2 \leq \tau_1' \rightarrow \tau_2'$, $f$ returns $(\tau_1' \leq \tau_1 \land \tau_2 \leq \tau_2')$; on all other inputs (i.e., on $\text{real} \leq \text{nat}$ and all inputs $\tau \leq \tau'$ such that exactly one of $\tau$ and $\tau'$ is a function type), $f$ returns $\varepsilon$, the trivially unsatisfiable DNF.

2. The relation of premise judgments to conclusion judgments, that is,

   $$ \bigcup_{\tau_1, \tau_2, \tau_1', \tau_2'} \{(\tau_1' \leq \tau_1, \tau_1 \rightarrow \tau_2 \leq \tau_1' \rightarrow \tau_2') \}, \quad (\tau_2 \leq \tau_2', \tau_1 \rightarrow \tau_2 \leq \tau_1' \rightarrow \tau_2')\}, $$

is well-founded. Attempts to derive $\tau \leq \tau'$ always terminate because premises decrease the sizes of types being considered.

However, the standard typing system for the lambda calculus with subtyping is nonnavigable:
1. No rule function $f$ exists. Even ignoring type subsumption, the unbounded nondeterminism in the typing rule for function applications would require $f$, on any input of the form $\Gamma \vdash e_1(e_2):\tau'$, to return the infinite DNF

$$\bigvee_{\tau} (\Gamma \vdash e_1:\tau \rightarrow \tau' \land \Gamma \vdash e_2:\tau).$$

2. Due to the subsumptive typing rule, the relation of premises to conclusions would contain elements of the form $(\Gamma \vdash e:\tau', \Gamma \vdash e:\tau)$ and therefore not be well-founded.

**Failing Derivations**  Given a navigable system with rule function $f$, a derivation of $J$ is a tree of judgments having root $J$, internal judgments $J_i$ such that $J_i$’s children are all the members of a conjunctive clause in $f(J_i)$, and leaves $J_l$ such that $f(J_l)$ contains the empty conjunctive clause (i.e., $f(J_l)$ is trivially satisfiable).

Complementarily, a failing derivation of $J$ is a tree of judgments having root $J$, internal judgments $J_i$ such that $J_i$’s children contain exactly one member of each conjunctive clause in $f(J_i)$, and leaves $J_l$ such that $f(J_l)=\varepsilon$ (i.e., $f(J_l)$ is trivially unsatisfiable).

(This definition of failing derivations differs from the examples shown in preceding sections: the irrelevant portions of the trees have been cut out, i.e., we only require one member of each conjunctive clause in $f(J_i)$, because additional children of $J_i$ are unnecessary.)

For navigable systems, $J$ is underivable iff there exists a failing derivation of $J$. All the leaves $J_l$ of failing derivations are underivable (because $f(J_l)=\varepsilon$), so inductively all the internal $J_i$ must be underivable as well (because every conjunctive clause in $f(J_l)$ contains an underivable judgment). Hence, the existence of a failing derivation of $J$ implies that $J$ is underivable. Conversely, if $J$ is underivable then by the definition of $f$, $f(J)$ must either be $\varepsilon$ (in which case the failing derivation of $J$ is just the leaf $J$) or have an underivable judgment $J_u$ in each conjunctive clause (in which case the failing derivation of $J$ gets built inductively by making $J$ an internal judgment having those $J_u$ as children).

For example, the judgment

$$J_0 = (\text{real} \rightarrow \text{real}) \rightarrow (\text{nat} \rightarrow \text{real}) \leq (\text{real} \rightarrow \text{nat}) \rightarrow (\text{real} \rightarrow \text{real})$$

is underivable using a standard subtyping system for lambda calculi, so there exists a failing derivation rooted at $J_0$. This failing derivation is a linear tree, with $J_0$’s only child being

$$J_1 = \text{nat} \rightarrow \text{real} \leq \text{real} \rightarrow \text{real},$$

and $J_1$’s only child being the leaf

$$J_2 = \text{real} \leq \text{nat}.$$

As required:
Every internal judgment $J_i$ has one child from each conjunctive clause in $f(J_i)$.

- $f(J_0) = (\text{real} \rightarrow \text{nat} \leq \text{real} \rightarrow \text{real} \land J_1)$
- $f(J_1) = (J_2 \land \text{real} \leq \text{real})$

- $f$ returns $\varepsilon$ on the leaf judgment $J_2 = \text{real} \leq \text{nat}$.

The underivability of $J_0$ can thus be traced from underivable $J_2$ to underivable $J_1$ to underivable $J_0$.

Derivations are sometimes guaranteed even in nonnavigable systems, e.g., those that are only nonnavigable due to inference rules having premises of the form $\exists x : (J_1(x) .. J_n(x))$, where $x$’s domain is infinite (such a rule prevents $f$ from returning a finite DNF in all cases). For example, the standard rules for function application and subsumption, in a simply typed lambda calculus with subtyping, have such premises, yet finite derivations exist, and induction can proceed on those derivations.

In the same way, failing derivations are sometimes guaranteed even in nonnavigable systems, e.g., those that are only nonnavigable due to inference rules having premises of the form $\forall x : (J_1(x) .. J_n(x))$, where $x$’s domain is infinite (such a rule again prevents $f$ from returning a finite DNF in all cases). Because all underivable judgments have finite failing derivations in such systems, induction may proceed on failing derivations.

Moreover, derivations and failing derivations may exist even without well-foundedness, e.g., derivations of $\Gamma \vdash e : \tau$ in a simply-typed lambda calculus having a standard rule of subsumption. Essentially, derivations exist when the set of derivable judgments is recursively enumerable (RE), and failing derivations exist when the set of derivable judgments is co-RE.

**Induction on Failing Derivations** Because judgments in navigable systems are underivable if they root failing derivation trees, one may establish that some property $P$ holds on all underivable $J$ by induction on the failing derivation of $J$.

As an example, let’s consider proving a property $P$ on underivable $\tau \leq \tau'$ judgments in the same lambda calculus with subtyping, by induction on the failing derivation of $\tau \leq \tau'$. Leaf judgments in such trees can only be of the form $\text{real} \leq \text{nat}$ or $\tau \leq \tau'$ such that exactly one of $\tau$ and $\tau'$ is a function type; the base cases of the proof must show that $P$ holds on all such judgments. The inductive case occurs when subtyping function types—all internal judgments in failing derivations must be of the form $J_i = \tau_1 \rightarrow \tau_2 \leq \tau'_1 \rightarrow \tau'_2$, such that $J_i$ has one child, which can be either an underivable $\tau'_1 \leq \tau_1$ or an underivable $\tau_2 \leq \tau'_2$. Hence, the proof must show both cases, i.e., when inductively assuming that $P$ holds on $\tau'_1 \leq \tau_1$, that $P$ holds on $J_i$, and when inductively assuming that $P$ holds on $\tau_2 \leq \tau'_2$, that $P$ holds on $J_i$.

**Comparison with Standard Induction on Derivations** Assuming a navigable system, an alternative to induction on failing derivations would be to create,
for every judgment form $F$, the judgment form $F \text{nd}$, where $\text{nd}$ means “not derivable”. Then inductive rules could be defined for deriving $J \text{nd}$ (for all judgments $J$), proof could be provided that $J \text{xor} J \text{nd}$ (for all judgments $J$), and every “induction on the failing derivation of $J$” could be replaced by “induction on the derivation of $J \text{nd}$”.

Induction on failing derivations has advantages over this alternative approach. First, the alternative approach requires defining new rules for deriving $J \text{nd}$, which is wasteful because the rules for deriving $J$ already implicitly define the rules for $J \text{nd}$. To make an analogy, when reasoning about a boolean $b_1$ and its negation, we don’t normally define a new boolean $b_2$, restrict $b_2$ such that $b_1 \text{xor} b_2$, and then use $b_2$ to refer to the negation of $b_1$; we instead just use $b_1$ and $\neg b_1$ because the value of $b_1$ already implicitly defines the value of its negation.

Second, induction on failing derivations avoids having to prove, for all judgments $J$, that $J \text{xor} J \text{nd}$. If the rules for deriving $J \text{nd}$ have been defined as specified earlier in this section—that is, every derivation of $J \text{nd}$ matches a failing derivation of $J$ and vice versa—then such a proof could repeat the arguments also provided earlier. The inference rules and xor-proof would then be an instantiation, on the particular system being analyzed, of the more general definitions and arguments provided earlier in this section.

Hence:

- Induction on failing derivations is simpler than the alternative approach. Induction on failing derivations avoids having to add judgment forms, define inference rules for all the new “$\text{nd}$” judgments, and prove an “xor-lemma” for every judgment form whose negation needs to be reasoned about.

- Induction on failing derivations provides a more general framework than the alternative approach. Each use of the alternative approach is an instantiation of the more general definitions and arguments provided earlier in this section.

Please also consider that an alternative to standard induction on derivations in navigable systems would be to define $J \text{nd}$ rules, prove that $J \text{xor} J \text{nd}$ (for all $J$), and then replace any “induction on the derivation of $J$” with “induction on the failing derivation of $J \text{nd}$”.

In this way, all inductions on derivations could be replaced by inductions on failing derivations. Consequently, any argument that induction on failing derivations is unnecessary (in light of standard induction on derivations) could be flipped into an argument that standard induction on derivations is unnecessary (in light of induction on failing derivations). In a sense, neither technique is “necessary”, as both are specializations of tree induction. The goal here is rather to provide a convenient, widely applicable technique for establishing properties of refutable judgments, like how standard induction on derivations is convenient and widely applicable for establishing properties of provable judgments.

**History** After defining what it means for subtyping relations to be complete with respect to type safety, I created this technique in Spring 2011 to help with proofs of completeness. Since then, I’ve also found the technique useful for proving
other properties, such as reflexivity and transitivity of subtyping (for details, and a more complex example of a navigable system, please see http://www.cse.usf.edu/~ligatti/papers/SubtypingTechReport.pdf). I hope the technique finds use in other domains as well.