Axioms in Definitional Calculi

Per Kreuger

piak@sics.se
Swedish Institute of Computer Science,
Box 1263, S-164 28 Kista, Sweden

Abstract. This paper introduces a restricted form of the axiom rule in calculi of Partial Inductive Definitions (PID). The paper argues that in calculi of PIDs the distinction between atomic and non-atomic formulae is not as clear as in traditional sequent calculi. Therefore the common restriction of the axiom rule to the atomic case is not adequate for this type of calculi.

A novel proviso for the axiom rule and corresponding provisos for the left and right definition rules are introduced with an accompanying discussion and suggestions for possible applications in the domain of declarative specification of operational behaviour of logic programs.

1 Background and motivation

In sequent calculi of Partial Inductive Definitions (PID) (see e.g. [1, 2, 3]) the axiom (Initial Sequent) rule has been modelled on the "*Axiom schema*" of the traditional (logical) sequent calculus (see e.g. [4]). However, in calculi of PIDs defined atoms are in many respects more properly understood as complex expressions. We will try to clarify this point with an example but first let us recall some basic notions of the theory of partial inductive definitions.

Let the *definiens* of an atom a with respect to a set of definitional clauses $\mathcal{D}$ be the following set of conditions:

$$\mathcal{D}(a) = \{A \mid (a \Leftarrow A) \in \mathcal{D}\} \qquad \text{(Eq. 1)}$$

and define the following two rules of inference in a sequent calculus of PIDs:

$$\frac{\Gamma \vdash C}{\Gamma \vdash c} \text{ for some } C \in \mathcal{D}(c), \qquad \text{(Rule 1)}$$

i.e if *any* condition in the definiens of an atom is derivable then the atom is itself derivable (c.f. resolution);

$$\frac{\Gamma, A \vdash C \text{ for all } A \in \mathcal{D}(a)}{\Gamma, a \vdash C}, \qquad \text{(Rule 2)}$$

i.e. if something is derivable from *each* of the conditions in the definiens of an atom it is also derivable from the atom itself (a dual of resolution). Note the closedness of the definition implicit in this rule.

These two rules are called $\mathcal{D}$-*right* and $\mathcal{D}$-*left* and are used to introduce atoms in the antecedent and succedent respectively. The definition $\mathcal{D}$ is regarded not as set of formulae in the antecedent of the sequent but rather as a parameter to the rules of the calculus. This approach has number of advantages when we consider extensions of the horn clause setting. See e.g. [2, 8, 3].

In addition to these we use the standard *axiom* (or initial sequent) rule for simultaneously introducing an atom in both antecedent and succedent:

$$\frac{}{\Gamma, a \vdash a} \qquad \text{(Rule 3)}$$

and rules for introducing the ordinary logical connectives such as conjunction, implication etc. in both antecedent and succedent.

Consider now the following definition

```
p <= q,a.
q <= a -> f.
a <= a.                                    (Def. 1)
```

The atomic formulas p and q both behave essentially as complex formulae through the rules of $\mathcal{D}$-*right* (resolution) and $\mathcal{D}$-*left* (definitional reflection) of definitional calculi. Consider the following proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, a \vdash a}\,Axiom(a) \qquad \dfrac{}{\Gamma, f, a \vdash C}\,\mathcal{D}\text{-}left(f)}{\Gamma, (a \text{ -> } f), a \vdash C}\,Arrow\text{-}left}{\Gamma, q, a \vdash C}\,\mathcal{D}\text{-}left(q)}{\Gamma, p \vdash C}\,\mathcal{D}\text{-}left(p) \qquad \text{(Deriv. 1)}$$

The atomic formula f is not defined by the above definition and is thus "false" according to the definition (using the principle of definitional reflection or the (implicit) "closedness" of inductive definitions). An atom defined by a clause with an empty body is dually considered (unconditionally) true.

The atom a however has a different status. a has a circular definition. We use definitions like these to "open up" the implicitly closed world of the definition. The atom f has a particular "value" according to the definition — namely (implicitly) "undefined" or "false", while the "value" of a according to this definition is (explicitly) "unknown" in a more direct sense. Another way to understand a definition like this is that a is unknown as anything but itself.

When we use a calculus of partial inductive definitions to compute functional expressions, definitions like that of a above are used to define "*data*", i.e. terms on canonical form. The datum is self-valued or self-replicating.

In traditional sequent calculi it is customary to restrict the applicability of the axiom schema to atomic formulae but as we can see from the above example this distinction is not adequate in the case of calculi of PIDs. We can distinguish

between four cases for an atom occurring in a sequent of which we are trying to find a proof:

1. The definiens of the atom is true.
   If the atom occurs in the succedent of the sequent this means that we have found a proof. If it occurs in the antecedent search must continue using a rule operating on some other formula in the sequent.
2. The definiens of the atom is false.
   This case is dual to the one above. If the atom occurs in the antecedent we have found a proof. If it occurs in the succedent search must continue using a rule operating on some other formula in the sequent.
3. The definiens of the atom is identical to the atom itself, i.e. has a "circular" definition.
   This means that we cannot (meaningfully) compute further on the atom using the $\mathcal{D}$-right or $\mathcal{D}$-left rules. But the axiom (initial sequent) rule should still be applicable. We will argue that this is the only case where the axiom rule should be applicable.
4. The definiens of an atom is neither true, false nor identical to the atom itself.
   This means we can compute on the atom. We can continue proof search in this branch of the proof tree using either the *$\mathcal{D}$-left* or the *$\mathcal{D}$-right* rule depending on whether the atom occurs in the antecedent or the succedent.
   In this case the atom is treated as if it was just a place holder for a complex formula, i.e. its definition. In this sense this kind of atom is not really an atomic formula at all.

2 Rules Relating Atoms to a Definition

For the purpose of the present paper we choose not to discuss the rules introducing complex formulae (built up from the traditional connectives) in sequent calculi as the choices involved for the exact formulation of these are in a sense orthogonal to the issues we are interested in here.

We can see the rules for introducing atomic formulae: the *$\mathcal{D}$-right*, the *$\mathcal{D}$-left*, and (we will argue) the *$\mathcal{D}$-axiom* rules as a basic system of PIDs upon which we can impose additional structure by choosing various different (e.g. substructural [5, 6]) formulations of the rules for complex formulae.

We would like to restrict the axiom (initial sequent) rule of the calculus to the case where the atom is truly atomic, i.e. is defined only by itself. A natural step is then also to restrict the applicability of the *$\mathcal{D}$-right* and *$\mathcal{D}$-left* rules to exclude this case.

First let us recapitulate the notions of "definiens" and "*a*-sufficient substitutions" in the context of variables. Equation 1 and rules 1–3 handle only the ground case. Originally these notions were introduced in [1] and [2]. [3] contains an algorithmic presentation with examples. Some variants and extensions

of these operations with inequality constraints are discussed from an implementational point of view in [7].

2.1 The Definiens Operation

The definiens operation computes the defining condition $\mathcal{D}(a)$ of an atom a given a definition $\mathcal{D}$. Let $\mathcal{D}$ be an ordered sequence (with $\cdot$ as sequence constructor) of clauses $(a_i \Leftarrow C_i)$ and a be an arbitrary atom. Then let

$$\mathcal{D}(a) = def(a, \mathcal{D}) \text{ where}$$

$$def(a, \mathcal{D}) = \begin{cases} false \text{ if } \mathcal{D} \text{ is the empty definition} \\ def(a, \mathcal{D}') \\ \quad \text{if } \mathcal{D} = (a_i \Leftarrow C_i) \cdot \mathcal{D}' \text{ and } \neg\exists\rho\,(a_i\rho = a) \\ (C_i\xi; def(a, \mathcal{D}')) \\ \quad \text{if } \mathcal{D} = (a_i \Leftarrow C_i) \cdot \mathcal{D}' \text{ and } a_i\xi = a \end{cases} \qquad \text{(Alg. 1)}$$

where ρ and ξ are substitutions. Note the disjunction (;) introduced in the last clause in the definition of $def(a, \mathcal{D})$ and that $\mathcal{D}(a)$ is *false* if a is not an instance of a defined atom in $\mathcal{D}$.

2.2 *A*-sufficient Substitutions

The definiens operation captures the case where we wish to determine if a particular atom a is an instance of some defined atom in the definition. An even more useful operation would be to determine if an atom a can be instantiated to some instance of a defined atom.

However not all such instantiations are of use to us. With the notion of an a-sufficient substitution, we capture a class of substitutions under which validity of proofs are preserved (with respect to the underlying (infinitary) theory of PID).

A substitution σ is a-sufficient w.r.t. a definition $\mathcal{D}$ provided that

$$\forall\zeta\,(\mathcal{D}(a\sigma\zeta) = (\mathcal{D}(a\sigma))\,\zeta) \qquad \text{(Eq. 2)}$$

i.e. if the definiens operation applied to the substitution instance $a\sigma$ is closed under further substitution.

There may be several a-sufficient substitutions w.r.t. a certain $\mathcal{D}$ for a given a.

2.3 The *D-right* rule

The traditional formulation (see e.g. [1,2,8]) of the *D-right* rule is almost identical to some formulations of resolution.

$$\frac{\Gamma\sigma \vdash C\sigma \quad \theta}{\Gamma \vdash c \quad \sigma\theta} \qquad \text{(Rule 4)}$$

if $(b <= C) \in \mathcal{D}\zeta$ and $\sigma = mgu(c, b)$ and ζ is a substitution that assign unique new variables (that do not occur in Γ or c) for all free variables in b and C.

It would be very natural to restrict the applicability of this rule by imposing the following proviso:

$$C\sigma\theta \neq c\sigma\theta \qquad \text{(Eq. 3)}$$

This proviso does not compromise the completeness of the calculus as there will always exist a proof for the conclusion of the rule if there exists one for the premise and the proviso is not fulfilled. To see this negate the proviso:

$$C\sigma\theta = c\sigma\theta \qquad \text{(Eq. 4)}$$

From this it follows that the premise and conclusion of the rule are identical under the computed substitution, i.e $(\Gamma\sigma \vdash C\sigma)\,\theta = (\Gamma \vdash c)\,\sigma\theta$.

In practical proof search it may of course be expensive to compute the substitution σ by some other means, but in principle the proviso does not weaken the calculus.

Note that in the context of proof *search* the proviso also acts as a primitive loop checker, i.e. terminates proof search in (some trivially) infinite branches of the search tree.

2.4 The *D-left* rule

The *D-left* rule can be similarly constrained.

$$\frac{\Gamma\sigma, \mathcal{D}(a\sigma) \vdash C\sigma \quad \theta}{\Gamma, a \vdash C \quad \sigma\theta} \qquad \text{(Rule 5)}$$

for some a-sufficient substitution σ and

$$a\sigma\theta \neq \mathcal{D}(a\sigma)\theta \qquad \text{(Eq. 5)}$$

Again the negation of this proviso implies that the premise is identical to the conclusion under the computed substitution and therefore the same comments on completeness apply to this rule as to the one above.

Note however that there may be several a-sufficient substitutions for a given definition and some may fulfil the proviso while others will not. The example of section 3.1 below will clarify this fact.

2.5 The *D-axiom* rule

We introduce a version of the axiom (initial sequent) rule that is parametrized by the definition in a way that is very similar to the *D-right* and *D-left* rules:

$$\frac{}{\Gamma, a \vdash c \quad \sigma\tau} \qquad \text{(Rule 6)}$$

where a and c are atoms and there exists an a-sufficient substitution σ such that:

$$a\sigma = \mathcal{D}(a\sigma) \qquad \text{(Eq. 6)}$$

and a most general unifier τ of $a\sigma$ and $c\sigma$:

$$\tau = mgu(a\sigma, c\sigma). \qquad \text{(Eq. 7)}$$

If not, the rule is not applicable.

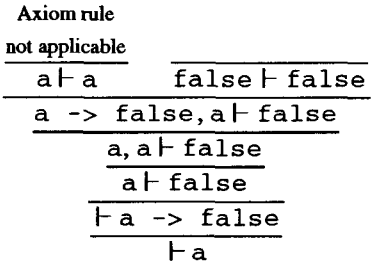
It is not, then, enough that the succedent is unifiable with some atom a in the antecedent. The definiens of $a\sigma$ for some a-sufficient substitution has to be identical to $a\sigma$ itself. This condition and the proviso of the *D-left* rule are mutually exclusive for a given a-sufficient substitution. It also means that the proviso for the *D-right* rule is false for all substitutions τ such that $\exists\xi\,(c\tau\xi = a\sigma)$ where c is the atom in the succedent.

This condition captures a notion of atomicity that is stronger than in traditional sequent calculi. Defined "atoms" are treated by the *D-left* rule while only "*proper*" atoms (i.e. with trivial definitions) are considered by the axiom rule.

Again, as noted above there may be several a-sufficient substitutions σ, for some for which the proviso (Eq. 5) is fulfilled, and others for which (Eq. 4) is instead fulfilled. For each σ one and only one of the conditions is always true.

Replacing the standard axiom rule with this restricted form of the rule clearly weakens the calculus. Note e.g. how the standard proof of the following example fails because of the proviso:

```
a <= a -> false.
```
(Def. 2)



(Deriv. 2)

3 Two examples

3.1 Example 1 – Cases

Assume that we in addition to the rules 4–6 above have the following rules for truth and falsity:

$$\frac{}{\Gamma \vdash \texttt{true}}\,(\top) \qquad \text{(Rule 7)}$$

$$\frac{}{\Gamma, \texttt{false} \vdash c}\,(\bot) \qquad \text{(Rule 8)}$$

and the following definition:

```
p(a) <= true.
p(b) <= false.
p(i) <= p(i).
```
(Def. 3)

we can then construct the following proofs:

$$\{\texttt{Z} = \texttt{a}\}\ \frac{\dfrac{}{\texttt{p(Y)} \vdash \texttt{true}}\,(\top)}{\texttt{p(Y)} \vdash \texttt{p(Z)}}\,\mathcal{D}\textit{-right}(\texttt{p(Z)}) \qquad \text{(Deriv. 3)}$$

$$\{\texttt{Y} = b\}\ \frac{\dfrac{}{\texttt{false} \vdash \texttt{p(Z)}}\,(\bot)}{\texttt{p(Y)} \vdash \texttt{p(Z)}}\,\mathcal{D}\textit{-left}(\texttt{p(Y)}) \qquad \text{(Deriv. 4)}$$

$$\{\texttt{Y} = \texttt{Z} = \texttt{i}\}\ \frac{}{\texttt{p(Y)} \vdash \texttt{p(Z)}}\,\mathcal{D}\textit{-axiom}(\texttt{p(Y)}) \qquad \text{(Deriv. 5)}$$

Note here the three distinct cases, one for each of the rules 4–6. Rules 7–8 terminate search in the syntactic (logic) domain. Other choices would be possible for these rules, e.g. rules for the linear constants of [6] or non-commutative logic principles for choosing an element in the antecedent etc.

3.2 Example 2 – Functional evaluation

Assume we have in addition to the rules 4–8 above the following rules for implication ($\rightarrow$):

$$\frac{\Gamma, A \vdash C}{\Gamma \vdash (A \rightarrow C)} \textit{Arrow-right} \qquad \text{(Rule 9)}$$

$$\frac{\Gamma, A \vdash C \quad \Gamma \vdash B}{\Gamma,\ (B \rightarrow A) \vdash C} \textit{Arrow-left} \qquad \text{(Rule 10)}$$

and the following definition:

```
0+M <= M.
s(N)+M <= succ(N+M)                                    (Def. 4)

succ(Expr) <=
    (Expr -> Value) -> s(Value)

0 <= 0.                                  % self replicating
s(N) <= s(N).                            % expressions
```

then we can construct proofs of the following kind according to a completely deterministic procedure, i.e. without search:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\{Y = 1\}\ \dfrac{}{1 \vdash Y}\ \mathcal{D}\textit{-left}}{0+1 \vdash Y}\ \mathcal{D}\textit{-axiom}}{\vdash ((0+1)\text{->}Y)} \quad \{Z = s(Y)\}\ \dfrac{}{s(Y) \vdash Z}\ \mathcal{D}\textit{-axiom}}{((0+1)\text{->}Y)\text{->}s(Y) \vdash Z}\ \textit{Arrow-left}}{\text{succ}(0+1) \vdash Z}\ \mathcal{D}\textit{-left}}{1+1 \vdash Z}\ \mathcal{D}\textit{-left} \qquad \text{(Deriv. 6)}$$

Moreover by using the restricted form of the axiom rule we eliminate a number of alternate proofs that would result in the following "lazy" answer substitutions for Z.

```
Z = s(0+1);
Z = succ(0+1).
```

For a general reference to the methodology involved in using functional programming in systems based on PID see [9] or [3].

4 Applications and future work

The main motivation behind this work is to define a calculus for PIDs that can be used for giving meaning to specifications of the procedural behaviour of a

class of logic programs. In [3] we defined the calculus $\mathcal{DOLD}$ that was used to give an operational semantics of the layered language GCLA II (based on PIDs).

On the object level of GCLA II we could achieve behaviour similar to the one described in section 3.2 by restricting the axiom rule to particular atomic expressions. These atoms were in all cases exactly those that also had "circular" definitions. The idea of restricting the axiom rule as suggested follows naturally from this observation although the exact formulation of the rules in the context of substitutions is perhaps not so obvious.

Even in non-functional programs (involving e.g. various kinds of hypothetical and default reasoning) it is very common that we want to impose restrictions on the applicability of the axiom rule. The traditional rule is far too general. Restricting the use of this rule was up to now done by writing strategies and rules that depend on domain specific (object level) information.

The version of the $\mathcal{D}$-rules presented here allows us to state this information directly on the object level in a very natural way, i.e. by giving "circular" definitions of atoms regarded as "data" (c.f. the example in section 3.2).

The $\mathcal{DOLD}$ calculus was used to interpret the rule (or meta-level) definition in GCLA II. We used functional (meta-) programs to specify the operational behaviour of (object-) programs and deterministic behaviour of functional programs was an essential property of the calculus. In [3] this was ensured by an ad hoc condition (based on the syntactic representation of object level sequents) on the axiom rule.

The calculus sketched in this paper generalizes this kind of side condition and allow us to reformulate the calculus in a more coherent and less ad hoc manner. With the introduction of the axiom rule suggested here the desired procedural properties of the $\mathcal{DOLD}$ calculus falls out naturally with out additions to the basic theory.

Future work includes investigation of the theoretical properties of calculi with this kind of restricted axiom rule. It seems ideally suited to our specific application, but it may have other interesting properties as well.

Our motivation is mainly computational, i.e. we want a calculus with reasonably deterministic behaviour. From a theoretical point of view, however, the partitioning of the term universe may have further implications. There is e.g. a question of the relation of the circular definitions to non-terminating computations, and in fact an other deficiency of the original formulation of GCLA II is related to this.

Another interesting question is whether the class of conditions provable by the restricted calculus can be characterized in some less syntactic way. It may be significant that the only examples we found so far of non-provable conditions are ones whose proofs involve the anomalous features of the theory of partial inductive definitions.

References

1. Hallnäs, L. *"Partial Inductive Definitions"*, Theoretical Computer Science vol. **87(1)** 1991 pp. 115–142.
2. Hallnäs, L.; Schroeder-Heister, P. *"A Proof Theoretic Approach to Logic Programming"*, Journal of Logic and Computation vol. **1(2)** 1990 pp. 261–283 and vol. **1(5)** 1991 pp. 635–660.
3. Kreuger, P. *"GCLA II — A definitional approach to control"*, SICS Research Report **R92:09**, Swedish Institute of Computer Science, Stockholm 1992. Also published (minus some minor additions and corrections) as Lic. Dissertation — Dept. of Computer Science, Chalmers University of Technology, Göteborg 1990 and in [11] pp. 239–297.
4. Kleene S. C. *"Introduction to Metamathematics"*, North-Holland 1952 pp. 442–448.
5. Schroeder-Heister, P. *"Structural Frameworks, substructural logics and the role of elimination inferences"*, in Plotkin, G.; Huet, G. (Eds.), *"Logical Frameworks"*. Cambridge University Press 1991 pp. 385–403.
6. Girard J.-Y., *"Linear Logic"*, in Theoretical Computer Science Vol. **50(1)**, 1987 pp. 1–102.
7. Aronsson, M., *"Implementational Issues in GCLA – A-sufficiency and the Definiens Operation"*, in [12] pp. 394–417.
8. Aronsson, M.; Eriksson, L-H.; Gäredal, A.; Hallnäs, L.; Olin, P. *"The Programming Language GCLA: A Definitional Approach to Logic Programming"* New Generation Computing vol. **7(4)** 1990 pp. 381–404.
9. Aronsson, M. *"A Definitional Approach to the Combination of Functional and Relational Programming"*, SICS Research Report **R91:10**, Swedish Institute of Computer Science, Stockholm 1991.
10. Schroeder-Heister, P. (Ed.) *"Extensions of Logic Programming, International Workshop, Tübingen, FRG, December 1989, Proceedings"*, Springer Lecture Notes in Artificial Intelligence **475**, Springer-Verlag 1991.
11. Eriksson, L-H; Hallnäs, L; Schroeder-Heister, P. (Eds.) *"Extensions of Logic Programming — ELP 91"*, in *"Extensions of Logic Programming, Second International Workshop, ELP'91, Stockholm, Sweden, January 1991, Proceedings"*, Springer Lecture Notes in Artificial Intelligence **596**, Springer-Verlag 1992.
12. Lamma, E.; Mello, P. (Eds.) *"Extensions of Logic Programming, Third International Workshop, ELP'92, Stockholm, Sweden, February 1992, Proceedings"*, Springer Lecture Notes in Artificial Intelligence **660**, Springer-Verlag 1993.