HAL
open science

E-Cyclist: Implementation of an Efficient Validation of FOL ID Cyclic Induction Reasoning (System Description)

Sorin Stratulat

To cite this version:

Sorin Stratulat. E-Cyclist: Implementation of an Efficient Validation of FOL ID Cyclic Induction Reasoning (System Description). 2020. hal-02464242v1

HAL Id: hal-02464242

https://hal.science/hal-02464242v1

Preprint submitted on 3 Feb 2020 (v1), last revised 31 Aug 2021 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

E-Cyclist: Implementation of an Efficient Validation of $\mathrm{FOL_{ID}}$ Cyclic Induction Reasoning (System Description)

Sorin Stratulat

LORIA, Department of Computer Science
Université de Lorraine
Metz, F-57000, FRANCE
sorin.stratulat@univ-lorraine.fr

Abstract. Checking the soundness of cyclic induction reasoning for first-order logic with inductive definitions ($\mathrm{FOL_{ID}}$) is decidable but the standard checking method is based on an exponential complement operation for Büchi automata. Recently, we introduced a polynomial checking method whose most expensive steps recall the comparisons done with multiset path orderings.
We describe the implementation of our method in the Cyclist prover. Referred to as E-Cyclist, it successfully checked all the proofs included in the original distribution of Cyclist. Heuristics have been devised to automatically define from the analysis of the proof derivations the ordering measures that satisfy the ordering constraints.
$\mathrm{FOL_{ID}}$ cyclic proof derivations may also be hard to certify. E-Cyclist witnesses a strong relation between the two cyclic and well-founded induction reasonings. This opens the perspective of using the known certification methods that work for well-founded induction proofs.

Introduction. Cyclic pre-proofs for the classical first-order logic with inductive predicates ($\mathrm{FOL_{ID}}$) have been extensively studied in [4, 5, 7]. They are finite sequent-based derivations where some terminal nodes, called *buds*, are labelled with sequents already occurring in the derivation, called *companions*. Bud-companion (BC) relations, graphically represented as *back-links*, are described by an induction function attached to the derivation, such that only one companion is assigned to each bud, but a node can be the companion of one or several buds. The pre-proofs can be viewed as digraphs whose cycles, if any, are introduced by the BC-relations.

It is easy to build unsound pre-proofs, for example by creating a BC-relation between the nodes labelled by the sequents from a stuttering step. The classical soundness criterion is the *global trace condition*. Firstly, the paths are annotated by traces built from inductive antecedent atoms (IAAs) found on the lhs of the sequents in the path. Then, it is shown that for every infinite path p in the cyclic derivation of a false sequent, there is some trace following p such that all successive steps starting from some point are decreasing and certain steps occurring infinitely often are strictly decreasing w.r.t. some semantic ordering. We say that

a *progress point* happens in the trace when a step is strictly decreasing. A *proof* is a pre-proof if every infinite path has an infinitely progressing trace starting from some point.

The standard checking method [5] of the global trace condition is decidable but based on an exponential complement operation for Büchi automata [9]. It has been implemented in the Cyclist prover [6] and experiments showed that the soundness checking can take up to 44% of the proof time. On the other hand, we presented in [10, 12] a less costly, polynomial-time, checking method. The pre-proof to be checked is firstly normalized into a digraph consisting of a set of derivation trees to which is attached an extended induction function. The resulting digraph counts among its roots the companions, as well as the root of the pre-proof to be checked. A sufficient condition for any pre-proof to be a proof is to show that every strongly connected component (SCC) of the digraph of the normalized pre-proof satisfies some ordering and derivability constraints, where the ordering constraints are similar to those used for certifying cyclic Noetherian induction proofs [11].

Implementation. Our method has been integrated in Cyclist, by replacing the standard checking method which leads to an extension called E-Cyclist. Cyclist builds the pre-proofs using a depth-first search strategy that aims at closing open nodes as quickly as possible. Whenever a new cycle is built, model-checking techniques provided by an external model checker are called to validate it. If the validation result is negative, the prover backtracks by trying to find another way to build new cycles. Hence, it may happen that the model checker be called several times during the construction of a pre-proof.

To each root r from the digraph $\mathcal{P}$ of a normalized pre-proof tree-set, the method attaches a measure $\mathcal{M}(r)$ consisting of a multiset of IAAs of the sequent labelling r, denoted by $S(r)$. One of the challenges is to find the good measure values that satisfy the ordering constraints. A procedure for computing these values is given by Algorithm 1.

Algorithm 1 GenOrd($\mathcal{P}$): to each root r of $\mathcal{P}$ is attached a measure $\mathcal{M}(r)$

```
for all root r do
    M(r) := ∅
end for
for all rb-path r → b from a non-singleton SCC do
    if there is a trace between an IAA A of S(b) and an IAA A' of S(r) then
        add A to M(rc) and A' to M(r), where rc is the companion of b
    end if
end for
```

Firstly, the value attached to each root is the empty set. Then, for each root-bud (rb) path from a cycle, denoted by $r \to b$, and for every trace along $r \to b$, leading some IAA of $S(r)$ to another IAA of $S(b)$, we add the corresponding

IAAs to the values of r and the companion of b, respectively. Since the number of rb-paths is finite, Algorithm 1 terminates.

Algorithm 1 may compute values that do not pass the comparison test for some non-singleton SCCs that are validated by the model checker. For this case, we considered an improvement consisting of the incremental addition of IAAs from a root sequent that are not yet in the value of the corresponding root r. Since the validating orderings are multiset extensions of multiset path orderings, such an addition does not affect the comparison value along the rp-paths starting from r. On the other hand, it may affect the comparison tests for the rp-paths ending in the companions of r. This may duplicate some IAAs from the values of the roots from the rp-paths leading to these companions. The duplicated IAAs have to be processed as any incrementally added IAA, and so on, until no changes are performed.

Table 1 illustrates some statistics about the proofs of the conjectures considered in Table 1 from [6], checked with the standard as well as our method. The IAAs are indexed in Cyclist to facilitate the construction of traces; the way they are indexed influence how the pre-proofs are built. Different indexations for a same conjecture may lead to different proofs (see the statistics for the second and third conjectures). The column labelled 'Time-E' presents the proof time measured in milliseconds by using our method. Similarly, the 'Time' column displays the proof time when using the standard method, while 'SC%' shows the percentage of time taken to check the soundness by the model checker. 'Depth' shows the depth of the proof, 'Nodes' the number of nodes in the proof, and 'Bckl.' the number of back-links in the proof. The last column gives the number of calls for pre-proof validations. The proof runs have been performed on a MacBook Pro featuring a 2,7 GHz Intel Core i7 processor and 16 GB of RAM. It can be noticed that, by using our method, the execution time is reduced by a factor going from 1.43 to 5.

Theorem	**Time-E**	Time	SC%	Depth	Nodes	Bckl.	Queries
$O_1x \vdash N_2x$	**2**	7	61	2	9	1	3
$E_1x \vee O_2x \vdash N_3x$	**4**	11	63	3	19	2	6
$E_1x \vee O_1x \vdash N_3x$	**2**	9	77	2	13	2	6
$N_1x \vdash O_2x \vee E_3x$	**3**	7	52	2	8	1	4
$N_1x \wedge N_2y \vdash Q_1(x,y)$	**297**	425	40	4	19	3	665
$N_1x \vdash Add_1(x,0,x)$	**1**	5	76	1	7	1	4
$N_1x \wedge N_2y \wedge Add_3(x,y,z) \vdash N_1z$	**8**	14	38	2	8	1	16
$N_1x \wedge N_2y \wedge Add_3(x,y,z) \vdash Add_1(x,sy,sz)$	**15**	22	32	2	14	1	14
$N_1x \wedge N_2y \vdash R_1(x,y)$	**266**	484	48	4	35	5	759
$N_1x \wedge N_2y \vdash p_1(x,y)$	**597**	?	?	4	28	3	2315

Table 1. Statistics for proofs checked with the standard and our method.

The last conjecture was not tested in [6] and refers to the 2-Hydra example [3]. A pre-proof of it, reproduced in Fig. 1, can also be generated by Cyclist, as shown in Fig. 2. In the pre-proof, the (root) companion and the buds are denoted by (a). Unfortunately, Cyclist was not able to validate it using the standard method, hence the missing figures are denoted by ? in the table.

$$
\dfrac{\dfrac{\dfrac{}{\vdash p00} \quad \dfrac{\dfrac{\dfrac{}{N0 \vdash p10} \quad \dfrac{(a)Nx, Ny \vdash pxy}{Nsx'', Nx'' \vdash psx''x''}}{Nsx'', Nx'' \vdash pssx''0}}{Nx' \vdash psx'0}\,Nx}{Nx \vdash px0} \quad \dfrac{\dfrac{\dfrac{}{N0, Nx \vdash px1} \quad \dfrac{\dfrac{(a)Nx, Ny \vdash pxy}{Nsy'', Ny'' \vdash psy''y''}}{Nsy'', Ny'' \vdash p0ssy''} \quad \dfrac{\dfrac{(a)Nx, Ny \vdash pxy}{Nx', Ny'' \vdash px'y''}}{Nsy'', Nx', Ny'' \vdash psx'ssy''}}{Nsy'', Nx, Ny'' \vdash pxssy''}\,Nx}{Nx, Ny' \vdash pxsy'}}{(a)Nx, Ny \vdash pxy}\,Ny
$$

Fig. 1. The Berardi and Tatsuta's cyclic pre-proof of the 2-Hydra example.

On the other hand, the proposed measure values may not pass some comparison tests that succeed with the standard method, even when using the improved version of Algorithm 1. Indeed, this was happened while proving $N_1x \wedge N_2y \vdash R(x, y)$. Hopefully, the prover backtracked and finally found the same proof as that built using the model checker.[1]

```
0: N_1(x) /\ N_2(y) |- p_1(x,y) (N L.Unf.) [1,2]
  1: N_1(x) /\ N_3(0) |- p_1(x,0) (N L.Unf.) [3,4]
    3: N_3(0) /\ N_4(0) |- p_1(0,0) (p R.Unf.) [5]
      5: N_3(0) /\ N_4(0) |- T (Id)
    4: N_1(y) /\ N_3(0) /\ N_4(s(y)) |- p_1(s(y),0) (N L.Unf.) [6,7]
      6: s(y)=0 /\ N_1(y) /\ N_3(0) /\ N_5(s(y)) |- p_1(s(y),0) (Ex Falso)
      7: N_1(y) /\ N_3(0) /\ N_4(y) /\ N_5(s(y)) |- p_1(s(y),0) (N L.Unf.) [8,9]
        8: N_1(0) /\ N_3(0) /\ N_5(s(0)) /\ N_6(0) |- p_1(s(0),0) (p R.Unf.) [10]
          10: N_1(0) /\ N_3(0) /\ N_5(s(0)) /\ N_6(0) |- T (Id)
        9: N_1(s(z)) /\ N_3(0) /\ N_4(z) /\ N_5(s(s(z))) /\ N_6(s(z)) |- p_1(s(s(z)),0) (p R.Unf.) [11]
          11: N_1(s(z)) /\ N_3(0) /\ N_4(z) /\ N_5(s(s(z))) /\ N_6(s(z)) |- p_1(s(z),z) (Weaken) [12]
            12: N_1(s(z)) /\ N_2(z) |- p_1(s(z),z) (Subst) [13]
              13: N_1(x) /\ N_2(y) |- p_1(x,y) (Backl) [0]
  2: N_1(x) /\ N_2(z) /\ N_3(s(z)) |- p_1(x,s(z)) (N L.Unf.) [14,15]
    14: N_2(z) /\ N_3(s(z)) /\ N_4(0) |- p_1(0,s(z)) (N L.Unf.) [16,17]
      16: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- p_1(0,s(0)) (p R.Unf.) [18]
        18: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- T (Id)
      17: N_2(y) /\ N_3(s(s(y))) /\ N_4(0) /\ N_5(s(y)) |- p_1(0,s(s(y))) (p R.Unf.) [19]
        19: N_2(y) /\ N_3(s(s(y))) /\ N_4(0) /\ N_5(s(y)) |- p_1(s(y),y) (Weaken) [20]
          20: N_1(s(y)) /\ N_2(y) |- p_1(s(y),y) (Subst) [21]
            21: N_1(x) /\ N_2(y) |- p_1(x,y) (Backl) [0]
    15: N_1(y) /\ N_2(z) /\ N_3(s(z)) /\ N_4(s(y)) |- p_1(s(y),s(z)) (N L.Unf.) [22,23]
      22: N_1(y) /\ N_3(s(0)) /\ N_4(s(y)) /\ N_5(0) |- p_1(s(y),s(0)) (p R.Unf.) [24]
        24: N_1(y) /\ N_3(s(0)) /\ N_4(s(y)) /\ N_5(0) |- T (Id)
      23: N_1(y) /\ N_2(w) /\ N_3(s(s(w))) /\ N_4(s(y)) /\ N_5(s(w)) |- p_1(s(y),s(s(w))) (p R.Unf.) [25]
        25: N_1(y) /\ N_2(w) /\ N_3(s(s(w))) /\ N_4(s(y)) /\ N_5(s(w)) |- p_1(y,w) (Weaken) [26]
          26: N_1(y) /\ N_2(w) |- p_1(y,w) (Subst) [27]
            27: N_1(x) /\ N_2(y) |- p_1(x,y) (Backl) [0]
```

Fig. 2. The screenshot of the 2-Hydra pre-proof generated by Cyclist.

[1] The source code of the implementation and the examples can be downloaded at https://members.loria.fr/SStratulat/files/e-cyclist.zip

In the following, we detail how our method has been applied for validating the 2-Hydra pre-proof from Fig. 2.

The 2-Hydra case. We explain the notations, the specification of the inductive predicates, the inference rules and the pre-proof from Fig. 2. Contrary to the pre-proof from Fig. 1, the CYCLIST pre-proof is horizontally indented by the level of nodes. The nodes are numbered and labelled by sequents where the comma (,) is replaced on the lhs of the sequents by the conjunction connector ($/\backslash$). As stated previously, the inductive predicate symbols are indexed.

The axioms defining the inductive predicates N and p are:

$$\begin{array}{rrr} & \Rightarrow p(0,0) & p(x,y) \Rightarrow p(s(x), s(s(y))) \\ \Rightarrow N(0) & \Rightarrow p(s(0),0) & p(s(y),y) \Rightarrow p(0, s(s(y))) \\ N(x) \Rightarrow N(s(x)) & \Rightarrow p(x, s(0)) & p(s(x),x) \Rightarrow p(s(s(x)),0) \end{array}$$

The applied inference rule for each sequent is pointed out at the end of the sequent.

(N L.Unf) [n_1, n_2] generates the nodes n_1 and n_2 by choosing an IAA of the form $N(t)$. If t is a variable, t will be replaced by 0 and $s(z)$, where z is a fresh variable. For the second instantiation, the IAA is replaced by $N(z)$. This represents a *progres point*. If t is of the form $s(t')$, the original sequent is reduced to another sequent by replacing the chosen IAA $N(s(t'))$ with $N(t')$.

(p R.Unf) [n] produces the node n resulting from the replacement of the consequent atom from the sequent labelling n with the condition of some axiom defining p and whose conclusion matches the atom.

(Id) and **(Ex Falso)** delete trivial conjectures. **(Weaken)** (resp., **(Subst)**) [n] is the LK's weaking (resp., substitution) rule [8] whose premise labels n. Finally, **(Backl)** [n] shows that the current node is a bud for the companion n.

The pre-proof from Fig. 2 is already normalized and there is only one non-singleton SCC. It can be easily noticed that every rb-path in a pre-proof tree has the form:

$$\dfrac{\dfrac{\cdots \qquad \cdots \qquad \dfrac{\dfrac{}{\textbf{bud}}\,\textbf{(Backl)}}{S'}\,\textbf{(Subst)}}{\vdots}}{\textbf{root}}$$

where the only time when **(Subst)** is applied in the rb-path is just before **(Backl)**.

Our validity method is based on properties to be satisfied *locally*, at the level of rb-paths. An rb-path $r \to b$ is *valid* if b is "smaller" than r w.r.t. a trace-based multiset extension relation. The standard and the trace-based definitions are:

– (*multiset extension*) $B <_{mul} A$ if there are two finite multisets X and Y such that $B = (A - X) \uplus Y$, $X \neq \emptyset$ and $\forall y \in Y$, $\exists x \in X, y < x$ holds.
– (*trace-based multiset extension*) b is "smaller" than r if, after pairwisely deleting the IAAs linked by a non-progressing trace along $r \to b$ (the result is X and Y as above), $X \neq \emptyset$ and $\forall y \in Y$, $\exists x \in X$ such that there is a progressing trace along $r \to b$ between x and y.

The three rb-paths from the unique non-singleton SCC link the root to the following nodes:

1. 27; the possible traces along this path are: $[N_1(x), N_1(x), \underline{N_1(y)}, N_1(y), N_1(y), N_1(y), N_1(x)]$ and $[N_2(y), \underline{N_2(z)}, N_2(z), \underline{N_2(w)}, N_2(w), \overline{N_2(w)}, N_2(y)]$,
2. 21; the possible traces along it are: $[N_2(y), \underline{N_2(z)}, N_2(z), \underline{N_2(y)}, N_2(y), N_2(y), N_2(y)]$ and $[N_2(y), \underline{N_2(z)}, N_2(z), N_5(s(y)), \overline{N_5(s(y))}, N_1(\overline{s(y)}), N_1(x)]$, and
3. 13; its possible traces are: $[N_1(x), N_1(x), \underline{N_1(y)}, N_1(y), N_1(s(z)), N_1(s(z)), N_1(s(z)), N_1(x)]$ and $[N_1(x), N_1(x), N_4(s(\overline{y})), \underline{N_4(y)}, \underline{N_4(z)}, N_4(z), N_4(z), N_2(y)]$.

All the above traces are progressing; we took care to underline the IAAs corresponding to progress points. By definition, these rb-paths are valid.

The improved version of Algorithm 1 can be applied to find the measure of the root. It consists of the multiset $\{N_1(x), N_2(y)\}$. In Fig. 3, we show a summary of these results for a non-optimized version of the pre-proof from Fig. 2.

```
Measures proposed for the roots in cycles:
    0: [2, 1]
Checking the link of IAAs from buds to roots:
    28 to 0: | 1 -> 1 [true ]| 2 -> 2 [true ] ==> true
    21 to 0: | 1 -> 2 [true ]| 2 -> 2 [true ] ==> true
    13 to 0: | 1 -> 1 [true ]| 2 -> 1 [true ] ==> true
The proof has succeeded
```

Fig. 3. The E-Cyclist validation of the 2-Hydra pre-proof from Fig. 2.

Even if the proof of validity for every rb-path found on the non-singleton SCCs is enough to conclude the satisfaction of the global trace condition, we will perform a last step, essential to represent the cyclic proofs as well-founded induction proofs: the introduction of the ordering constraints. For doing this, we consider only the pre-proofs for which every application of **(L.Unf.)** instantiates variables along the rb-paths. This restriction allows us to annotate with some substitution each step encountered by following an rb-path, i.e.:

– the *instantiation* substitution used by **(L.Unf.)**, for the **(L.Unf.)**-steps, and
– the *identity* substitution replacing the free variables from the current sequent by themselves, for the other steps.

Let p be an rb-path, θ the *cumulative* substitution built by composing the substitutions encountered while following p, and δ the substitution used by the **(Subst)**-step. We say that the bud b of p is *discharged* by the root r of p if there is a well-founded and 'stable under substitution' ordering $<$ such that $\mathcal{M}(c(b))\delta < \mathcal{M}(r)\theta$, where $c(b)$ is the root companion of b and $\mathcal{M}(r)\theta$ (resp., $\mathcal{M}(c(b))\delta$) is the multiset obtained by instantiating each IAA from $\mathcal{M}(r)$ with θ (resp., each IAA from $\mathcal{M}(c(b))$ with δ).

We recall the main theorem stating when a pre-proof is a proof:

Theorem 1 (adapted from [10, 12]) *Let TS be the normalized pre-proof tree-set of a pre-proof P. If there is a well-founded and 'stable under substitution' ordering such that the buds from all rb-paths encountered in the non-singleton SCCs of TS are discharged, then P is a proof.*

For the 2-Hydra example, the well-founded and 'stable under substitution' ordering can be defined as the multiset extension of some multiset path ordering (mpo) [2] built from any precedence over the symbols $\{N, s, 0\}$. The reason is that the ordering comparisons are boiled down to show that t is smaller than $s(t)$, for some terms t, which holds for every such mpo, due to its subterm property.

The comparison required for the rb-path linking the root to the node:

- 27 is $\{N(y), N(w)\} < \{N(s(y)), N(s(s(w)))\}$,
- 21 is $\{N(s(y)), N(y)\} < \{N(0), N(s(s(y)))\}$, and
- 13 is $\{N(s(z)), N(z)\} < \{N(s(s(z))), N(0)\}$.

This result allows to 'interpret' $\mathrm{FOL_{ID}}$ cyclic pre-proofs in Coq [13] as well-founded induction proofs. By using methods for certification similar to those for formula-based Noetherian induction reasoning [11], we pave the way to a solution for certifying $\mathrm{FOL_{ID}}$ cyclic reasoning, in general, and (E-)Cyclist pre-proofs, in particular.[2]

Conclusions and future work. We have implemented in Cyclist a more effective technique for validating $\mathrm{FOL_{ID}}$ cyclic pre-proofs which allows to speed up the proof runs by 5. Besides its polynomial time complexity, an important factor for its efficiency is the lack of the overhead time required to communicate with external tools.

The considered proof examples are rather small. We intend to test our method on bigger examples and on cyclic proofs from domains other than $\mathrm{FOL_{ID}}$, e.g., separation logic. We also plan to implement a certification tool for Cyclist proofs, similar to what has been developed for certifying (cyclic) well-founded induction proofs built with the SPIKE prover [1].

[2] For the reviewers, the full Coq specifications and proofs for certifying the pre-proof from Fig. 2 are given at https://members.loria.fr/SStratulat/files/hydra-coq.zip

References

1. *The SPIKE theorem prover*, 1997-2020. https://github.com/sorinica/spike-prover.
2. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
3. S. Berardi and M. Tatsuta. Classical system of Martin-Lof's inductive definitions is not equivalent to cyclic proofs. *Logical Methods in Computer Science*, 15(3), 2019.
4. J. Brotherston. Cyclic proofs for first-order logic with inductive definitions. In *Proceedings of TABLEAUX-14*, volume 3702 of *LNAI*, pages 78–92. Springer-Verlag, 2005.
5. J. Brotherston. *Sequent Calculus Proof Systems for Inductive Definitions.* PhD thesis, University of Edinburgh, November 2006.
6. J. Brotherston, N. Gorogiannis, and R. L. Petersen. A generic cyclic theorem prover. In *APLAS-10 (10th Asian Symposium on Programming Languages and Systems)*, volume 7705 of *LNCS*, pages 350–367. Springer, 2012.
7. J. Brotherston and A. Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.
8. G. Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39:176–210, 1935.
9. M. Michel. Complementation is more difficult with automata on infinite words. Technical report, CNET, 1988.
10. S. Stratulat. Cyclic proofs with ordering constraints. In R. A. Schmidt and C. Nalon, editors, *TABLEAUX 2017 (26th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods)*, volume 10501 of *LNAI*, pages 311–327. Springer, 2017.
11. S. Stratulat. Mechanically certifying formula-based Noetherian induction reasoning. *Journal of Symbolic Computation*, 80, Part 1:209–249, 2017.
12. S. Stratulat. Validating back-links of $\mathrm{FOL}_{\mathrm{ID}}$ cyclic pre-proofs. In S. Berardi and S. van Bakel, editors, *CL&C'18 (Seventh International Workshop on Classical Logic and Computation)*, number 281 in EPTCS, pages 39–53, 2018.
13. The Coq development team. *The Coq Reference Manual.* INRIA, 2020.