E-Cyclist: Implementation of an Efficient Validation of FOL ID Cyclic Induction Reasoning

Sorin Stratulat

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

Sorin Stratulat

Université de Lorraine, CNRS, LORIA
Metz, F-57000, FRANCE

sorin.stratulat@univ-lorraine.fr

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 trace-based ordering measures that guarantee the soundness property.

Introduction. Cyclic pre-proofs for the classical first-order logic with inductive predicates ($\mathrm{FOL_{ID}}$) have been extensively studied in [2, 3, 5]. 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 atoms occurring on the lhs of the sequents in the path, referred to as *inductive antecedent atoms* (IAAs). 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* occurs 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 [3] of the global trace condition is decidable but based on an exponential complement operation for Büchi automata [8]. It has been implemented in the CYCLIST prover [4] and experiments showed that the soundness checking can take up to 44% of the proof time. On the other hand, we presented in [9, 10] a less costly, polynomial-time, checking method. The pre-proof to be checked is firstly normalized into a digraph $\mathscr{P}$ consisting of a set of derivation trees to which is attached an extended induction function. The resulting digraph counts the companions among its roots, as well as the root of the pre-proof to be checked. Also, all infinite paths in the pre-proof, starting from some point, can be reconstructed by concatenating root-bud paths (*rb*-paths) in $\mathscr{P}$. Finally, a sufficient condition for ensuring the global trace condition is to show that every *rb*-path from the strongly connected components

T. Kutsia (Ed.): Symbolic Computation
in Software Science (SCSS'21)
EPTCS ??, 2021, pp. 1–7, doi:10.4204/EPTCS.??.??

(SCCs) of $\mathscr{P}$ has a trace that satisfies some trace-based ordering constraints. Therefore, in theory, if the soundness of some pre-proof can be validated with the new method, it can also be validated with the standard one.

Implementation. Our method has been integrated in the CYCLIST release labelled as CSL-LICS14, by *replacing* the standard checking method. The result was 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, the model checker may be called several times during the construction of a pre-proof.

Here is how our method works. Firstly, the pre-proof is normalized to a digraph $\mathscr{P}$. To each root r from $\mathscr{P}$, the method attaches a measure $\mathscr{M}(r)$ 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 trace-based ordering constraints. A procedure for computing these values is given by Algorithm 1.

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

for all root r **do**
  $\mathscr{M}(r) := \emptyset$
end for
for all rb-path $r \to b$ 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 $\mathscr{M}(rc)$ and A' to $\mathscr{M}(r)$, where rc is the companion of b
  end if
end for

At the beginning, the value attached to each root is the empty set. Then, for each *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 trace-based variants of multiset extension orderings, such an addition does not affect the comparison value along the rb-paths starting from r. On the other hand, it may affect the comparison tests for the rb-paths ending in the companions of r. This may duplicate some IAAs from the values of the roots from the rb-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 [4], using inductive predicates as N, E, O, and Add, referring to the naturals, even and odd numbers, as well as the addition on naturals. All inductive predicates but p are defined in [4]. The proofs have been 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 influences 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 [4] and refers to the 2-Hydra example [1]. A pre-proof of it, reproduced in Figure 1, can also be generated by CYCLIST, as shown in Figure 4. Unfortunately, CYCLIST was not able to validate it using the standard method, the missing figures being denoted by ?.

$$\dfrac{\dfrac{\dfrac{}{\vdash p00}\quad \dfrac{\dfrac{}{N0 \vdash p10}\quad \dfrac{\dfrac{(a)Nx,Ny \vdash pxy}{Nsz,Nz \vdash pszz}}{Nsz,Nz \vdash pssz0}}{Nx' \vdash psx'0}\,Nx}{Nx \vdash px0}\quad \dfrac{\dfrac{}{N0,Nx \vdash px1}\quad \dfrac{\dfrac{(a)Nx,Ny \vdash pxy}{Nsu,Nu \vdash psuu}\quad \dfrac{\dfrac{(a)Nx,Ny \vdash pxy}{Nx',Nu \vdash px'u}}{Nsu,Nx',Nu \vdash psx'ssu}\,Nx}{Nsu,Nx,Nu \vdash pxssu}}{Nx,Ny' \vdash pxsy'}\,Ny}{(a)Nx,Ny \vdash pxy}$$

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

It also may occur that the proposed measure values, as shown in Figure 5 for a non-optimised proof of 2-Hydra, may not pass some comparison tests that succeed with the standard method, even when using the improved version of Algorithm 1. Indeed, this happened while proving $N_1x \wedge N_2y \vdash R(x,y)$. Luckily, the prover backtracked and finally found the same pre-proof as that originally built with CYCLIST.[1]

We detail now how our method has been applied for validating the 2-Hydra pre-proof from Figure 4.

The 2-Hydra case. The 2-Hydra problem is a particular case showing the termination of the battle between Hercules and Hydra [6] when Hydra has at most two heads that hang on the top of necks of

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

different lengths. Hercules prevails if either Hydra has i) no heads at all, or ii) the length of the first neck is 1 unit and it has lost the second head (i.e., the length of the neck is 0), or iii) the length of the second neck is 1 unit, as in Figure 2.

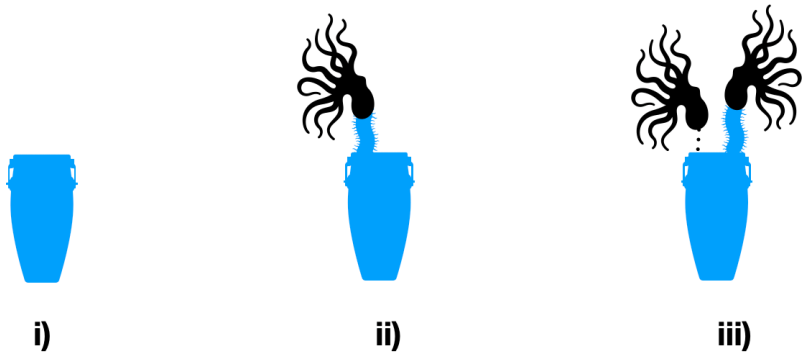

Figure 2: The cases when Hercules wins.

Hercules can cut the Hydra's necks according to the following rules. If both necks have strictly positive lengths, then Hercules can cut them such that the first neck is shorter by 1 unit and the second by 2 units (see the case iv in Figure 3). If Hydra has already lost one of the heads and the neck of the other head has a length l of at least 2 units, the first head will have a neck of length $l-1$ units and the second head a neck of length $l-2$ units (see the cases v and vi in Figure 3).

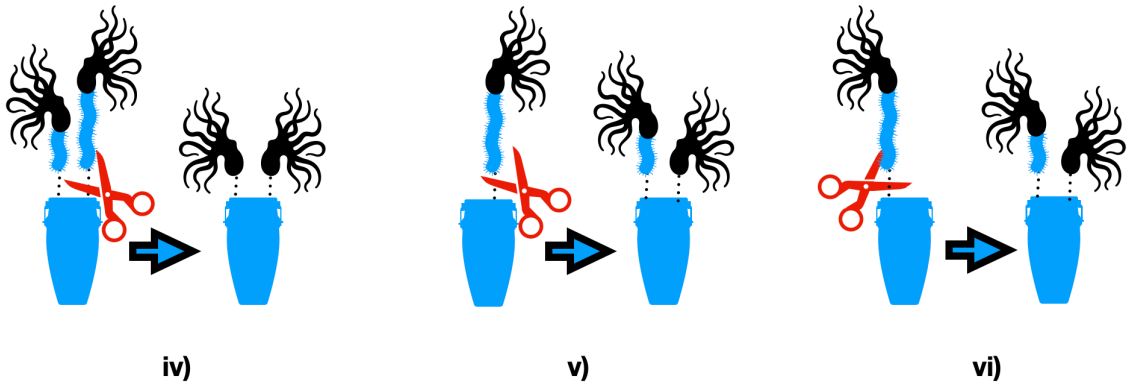

Figure 3: The cases when Hercules cuts the necks of Hydra.

Next, we introduce the notations, the specification of the inductive predicates, the inference rules, then explain the pre-proof from Figure 4. Contrary to the pre-proof from Figure 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 ($/\backslash$).

The axioms defining the inductive predicates N and p are:

$$\Rightarrow p(0,0) \qquad p(x,y) \Rightarrow p(s(x),s(s(y)))$$
$$\Rightarrow N(0) \qquad \Rightarrow p(s(0),0) \qquad p(s(y),y) \Rightarrow p(0,s(s(y)))$$
$$N(x) \Rightarrow N(s(x)) \qquad \Rightarrow p(x,s(0)) \qquad p(s(x),x) \Rightarrow p(s(s(x)),0)$$

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 weakening (resp., substitution) rule [7] whose premise labels n. Finally, **(Backl)** [n] shows that the current node is a bud for the companion n.

```
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]
```

Figure 4: The screenshot of the 2-Hydra pre-proof generated by CYCLIST.

The pre-proof from Figure 4 is already normalized and has one non-singleton SCC with three rb-paths.

Our validity method is based on properties to be satisfied *locally*, at the level of rb-paths. An rb-path $r \rightarrow b$ is *valid* if b is "smaller" than r w.r.t. a trace-based multiset extension relation. This relation guarantees the existence of traces following each infinite path p, built from the concatenation of the traces defined for the rb-paths along p. The definitions for the standard and trace-based multiset extension are:

- (*standard 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.

In Figure 5, we summarize the result of the application of the improved version of Algorithm 1 to a non-optimized version of the pre-proof from Figure 4, for which the node 27 was denoted as 28. The found measure of the root is the multiset of its IAAs indexed by 2 and 1, i.e., $\{N_2(x), N_1(y)\}$.

```
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
```

Figure 5: The E-CYCLIST validation of the 2-Hydra pre-proof from Figure 4.

In Figure 5, for each rb-path, *i* -> *j* denotes that there is a trace linking the root IAA indexed by *j* to the bud IAA indexed by *i*, `[true ]` means that the trace is progressing, and '`===> true`' informs that the rb-path is valid, as follows:

1. 0 to 28 (27 in Figure 4); the possible traces following 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 $[N_2(y), \underline{N_2(z)}, N_2(z), \underline{N_2(w)}, N_2(w), N_2(w), N_2(y)]$,
2. 0 to 21; the possible traces 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 $[N_2(y), \underline{N_2(z)}, N_2(z), N_5(s(y)), N_5(s(y)), \overline{N_1(s(y))}, N_1(x)]$, and
3. 0 to 13; the 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 $[N_1(x), N_1(x), N_4(s(y)), \underline{N_4(y)}, \underline{N_4(z)}, N_4(z), \overline{N_4(z)}, N_2(y)]$.

All the above traces are progressing, where the underlined IAAs correspond to progress points. By definition, these rb-paths are valid and conclude that the 2-Hydra pre-proof is a proof, by using arguments as in [9, 10].

Conclusions and future work. We have implemented in CYCLIST a more effective technique for validating FOL$_{\text{ID}}$ 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. In practice, our method can validate pre-proofs that cannot be validated by the CSL-LICS14 release of CYCLIST. Even if we do not have yet a clear evidence, we strongly believe that this also holds for the other way around, as this might have happened for the $N_1 x \wedge N_2 y \vdash R(x, y)$ example.

The considered pre-proof examples are rather small. We intend to test our method more extensively and on cyclic pre-proofs from domains other than FOL$_{\text{ID}}$, e.g., separation logic.

References

[1] S. Berardi & M. Tatsuta (2019): *Classical System of Martin-Lof's Inductive Definitions is not Equivalent to Cyclic Proofs*. *Logical Methods in Computer Science* 15(3), doi:10.23638/LMCS-15(3:10)2019.

[2] J. Brotherston (2005): *Cyclic Proofs for First-Order Logic with Inductive Definitions*. In: *Proceedings of TABLEAUX-14*, *LNAI* 3702, Springer-Verlag, pp. 78–92, doi:10.1007/11554554_8.

[3] J. Brotherston (2006): *Sequent Calculus Proof Systems for Inductive Definitions*. Ph.D. thesis, University of Edinburgh.

[4] J. Brotherston, N. Gorogiannis & R. L. Petersen (2012): *A Generic Cyclic Theorem Prover*. In: *APLAS-10 (10th Asian Symposium on Programming Languages and Systems)*, *LNCS* 7705, Springer, pp. 350–367, doi:10.1007/978-3-642-35182-2_25.

[5] J. Brotherston & A. Simpson (2011): *Sequent calculi for induction and infinite descent*. *Journal of Logic and Computation* 21(6), pp. 1177–1216, doi:10.1093/logcom/exq052.

[6] N. Dershowitz & G. Moser (2007): *The Hydra Battle Revisited*. *Rewriting, Computation and Proof*, pp. 1–27, doi:10.1007/978-3-540-73147-4_1.

[7] G. Gentzen (1935): *Untersuchungen über das logische Schließen. I.* *Mathematische Zeitschrift* 39, pp. 176–210, doi:10.1007/BF01201353.

[8] M. Michel (1988): *Complementation is more difficult with automata on infinite words*. Technical Report, CNET.

[9] S. Stratulat (2017): *Cyclic Proofs with Ordering Constraints*. In R. A. Schmidt & C. Nalon, editors: *TABLEAUX 2017 (26th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods)*, *LNAI* 10501, Springer, pp. 311–327, doi:10.1007/978-3-319-66902-1_19.

[10] S. Stratulat (2018): *Validating Back-links of FOL_{ID} Cyclic Pre-proofs*. In S. Berardi & S. van Bakel, editors: *CL&C'18 (Seventh International Workshop on Classical Logic and Computation)*, *EPTCS* 281, pp. 39–53, doi:10.4204/EPTCS.281.4.