Efficient Validation of $\mathsf{FOL}_{\mathsf{ID}}$ Cyclic Induction Reasoning

VeriDis + MATRYOSHKA Workshop, Amsterdam
June 12, 2019

Sorin Stratulat

INRIA, Université de Lorraine

Motivation

☞ soundness checking of cyclic pre-proofs in FOL with inductive definitions ($\mathsf{FOL}_{\mathsf{ID}}$)

pre-proof: finite derivation tree with backlinks (bud-companion relations) using CLKID^{ω} (LK + '=' rules + unfold + case) (Brotherston and Simpson [2011])

$$\Rightarrow R(0,y) \quad (1)$$

$$R(x,0) \Rightarrow R(sx,0) \quad (2)$$

$$R(ssx,y) \Rightarrow R(sx,sy) \quad (3)$$

$$\Rightarrow N(0) \quad (4)$$

$$N(x) \Rightarrow N(s(x)) \quad (5)$$

$$\dfrac{\dfrac{}{\vdash R(0,0)}\,(R.(1)) \qquad \dfrac{\dfrac{\dfrac{}{Nx' \vdash R(x',0)\ (\dagger 1)}\,(Subst)}{Nx'' \vdash R(x'',0)}\,(R.(2))}{Nx'' \vdash R(sx'',0)}\,(Case\ N)}{\dfrac{\dfrac{\dfrac{}{Nx, Ny \vdash R(x,y)\ (*1)}\,(Subst)}{Nssx', Ny' \vdash R(ssx',y')}\,(Cut) \qquad \dfrac{\dfrac{\dfrac{}{\underline{Nx'} \vdash R(x',0)\ (\dagger)}\,(R.(2))}{Nx' \vdash R(sx',0)} \qquad \dfrac{\dfrac{}{Ny \vdash R(0,y)}\,(R.(1))}{Nx', \underline{Ny} \vdash R(sx',y)}\,(Case\ N)}{Nx', Ny' \vdash R(ssx',y')}\,(R.(3))}{Nx', Ny' \vdash R(sx',sy')}\,(Case\ N)}{\underline{Nx}, Ny \vdash R(x,y)\ (*)}\,(Case\ N)$$

Motivation

☞ soundness checking of cyclic pre-proofs in FOL with inductive definitions ($\mathsf{FOL}_{\mathsf{ID}}$)

pre-proof: finite derivation tree with backlinks (bud-companion relations) using CLKID^{ω} (LK + '=' rules + unfold + case) (Brotherston and Simpson [2011])

$$\Rightarrow R(0,y) \quad (1)$$

$$R(x,0) \Rightarrow R(sx,0) \quad (2)$$

$$R(ssx,y) \Rightarrow R(sx,sy) \quad (3)$$

$$\Rightarrow N(0) \quad (4)$$

$$N(x) \Rightarrow N(s(x)) \quad (5)$$

$$\dfrac{\dfrac{}{\vdash R(0,0)}\,(R.(1)) \qquad \dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger 1)}{Nx'' \vdash R(x'',0)}\,(Subst)}{Nx'' \vdash R(sx'',0)}\,(R.(2))}{\dfrac{}{Ny \vdash R(0,y)}\,(R.(1)) \qquad \dfrac{\dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger)}{Nx' \vdash R(sx',0)}\,(R.(2)) \qquad \dfrac{\dfrac{Nx, Ny \vdash R(x,y)\ (*1)}{Nssx', Ny' \vdash R(ssx',y')}\,(Subst)}{Nx', Ny' \vdash R(ssx',y')}\,(Cut)}{Nx', Ny' \vdash R(sx',sy')}\,(R.(3))}{Nx', Ny \vdash R(sx',y)}\,(Case\ N)}{Nx, Ny \vdash R(x,y)\ (*)}\,(Case\ N)$$

Soundness checking

☞ annotate paths with traces (Brotherston and Simpson [2011])

Global trace condition: implements the 'Descente Infinie' principle

(1) by contradiction, assume that the root sequent $\Gamma \vdash \Delta$ is false
☞ *finite* unfoldings for true ind. atoms: $N(0)$, $N(s(0))$, ...

(2) show that for every infinite path p in the cyclic derivation, 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 defined over the number of unfoldings.

(3) true ind. atoms require *infinite* unfoldings. Contradiction.

Check: testing the inclusion relation between two Büchi automata

- decidable but **doubly exponential**
- implemented in the Cyclist prover; the proofs are **not certified**

Soundness checking

☞ annotate paths with traces (Brotherston and Simpson [2011])

Global trace condition: implements the 'Descente Infinie' principle

(1) by contradiction, assume that the root sequent $\Gamma \vdash \Delta$ is false
☞ *finite* unfoldings for true ind. atoms: $N(0)$, $N(s(0))$, ...

(2) show that for every infinite path p in the cyclic derivation, 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 defined over the number of unfoldings.

(3) true ind. atoms require *infinite* unfoldings. Contradiction.

Check: testing the inclusion relation between two Büchi automata

- decidable but **doubly exponential**
- implemented in the Cyclist prover; the proofs are **not certified**

Overview

Cyclic Reasoning for $\mathsf{FOL}_{\mathsf{ID}}$

A Polynomial Procedure for Checking the Global Trace Condition

Certifying Cyclic Proofs with Coq

Cyclic Reasoning for $\mathsf{FOL}_{\mathsf{ID}}$

CLKID$^{\omega}_{N}$: a particular case of CLKID$^{\omega}$

☞ Stratulat [2017a, 2018]

$$\frac{\Gamma[\{x \mapsto u\}] \vdash \Delta[\{x \mapsto u\}]}{\Gamma, x = u \vdash \Delta}$$ x is a variable not occurring in u $(= L)$

☞ particular case of $(= L)$ of CLKID$^{\omega}$ where x can also be a non-variable term

The case when the trace value is strictly decreasing

The inductive predicates are defined by axioms of the form

$$Q_1(\overline{u}_1) \wedge \ldots \wedge Q_h(\overline{u}_h) \wedge P_{j_1}(\overline{t}_1) \wedge \ldots \wedge P_{j_m}(\overline{t}_m) \Rightarrow P_i(\overline{t}) \qquad (6)$$

The $(Case)$ rule:

$$\frac{\ldots \qquad \Gamma, \overline{\mathbf{t}}' = \overline{\mathbf{t}}, Q_1(\overline{u}_1), \ldots, Q_h(\overline{u}_h), P_{j_1}(\overline{t}_1), \ldots, P_{j_m}(\overline{t}_m) \vdash \Delta \qquad \ldots}{\Gamma, P_i(\overline{t}') \vdash \Delta} \; (Case\ P_i)$$

☞ unfolding step: $P_{j_1}(\overline{t}_1), \ldots, P_{j_m}(\overline{t}_m)$ are *case descendants* of $P_i(\overline{t}')$.

Traces and progress points

inductive antecedent atoms (IAA) $\tau_1\ \tau_2\ \ldots \tau_n\ \ldots$

Definition (Trace, Progress point)
A *trace* following some (potentially infinite) path $p\ [N^1, N^2, \ldots]$ in a pre-proof tree is a sequence $(\tau_i)_{(i \geq 0)}$ of IAAs such that:

- τ_{i+1} is $\tau_i[\{x \mapsto u\}]$ if $S(N^i) \equiv (\Gamma, x = u \vdash \Delta)$ is the conclusion of $(= L)$;
- $\tau_i = \tau_{i+1}[\delta]$ if $S(N^i)$ is the conclusion of $(Subst)$ using δ;
- if $S(N^i)$ is the conclusion of a $(Case)$-rule, then either i) τ_{i+1} is τ_i, or ii) τ_i is its principal formula and τ_{i+1} is a case descendant of τ_i. In this case, i is called a *progress point*;
- $\tau_{i+1} = \tau_i$ if $S(N^i)$ is the conclusion of any other rule.

An *infinitely progressing* trace has infinitely many progress points.

Proofs

Definition (CLKID^ω_N proof)
A CLKID^ω_N pre-proof is a CLKID^ω_N *proof* if every infinite path has an infinitely progressing trace starting from some point.

☞ the global trace condition is satisfied

$$\dfrac{\dfrac{}{\vdash R(0,0)}\,(R.(1)) \quad \dfrac{\dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger 1)}{Nx'' \vdash R(x'',0)}\,(Subst)}{Nx'' \vdash R(sx'',0)}\,(R.(2))}{\dfrac{Nx \vdash R(x,0)\ (\dagger)}{Nx' \vdash R(sx',0)}\,(R.(2))}\,(Case\ N) \quad \dfrac{\dfrac{Nx, Ny \vdash R(x,y)\ (*1)}{Nssx', Ny' \vdash R(ssx', y')}\,(Subst)}{Nx', Ny' \vdash R(ssx', y')}\,(Cut) \quad \dfrac{Nx', Ny' \vdash R(ssx', y')}{Nx', Ny' \vdash R(sx', sy')}\,(R.(3))}{\dfrac{\dfrac{}{Ny \vdash R(0,y)}\,(R.(1)) \quad Nx', Ny \vdash R(sx', y)}{Nx, Ny \vdash R(x,y)\ (*)}\,(Case\ N)}\,(Case\ N)$$

Proofs

Definition (CLKID$^\omega_N$ proof)
A CLKID$^\omega_N$ pre-proof is a CLKID$^\omega_N$ *proof* if every infinite path has an infinitely progressing trace starting from some point.

☞ the global trace condition is satisfied

$$\dfrac{\dfrac{\dfrac{}{\vdash R(0,0)}\,(R.(1)) \qquad \dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger\mathbf{1})}{Nx'' \vdash R(x'',0)}\,(Subst)}{Nx'' \vdash R(sx'',0)}\,(R.(2))}{\dfrac{\dfrac{\underline{Nx'} \vdash R(x',0)\ (\dagger)}{Nx' \vdash R(sx',0)}\,(R.(2)) \qquad \dfrac{\dfrac{\dfrac{Nx, Ny \vdash R(x,y)\ (*\mathbf{1})}{Nssx', Ny' \vdash R(ssx',y')}\,(Subst)}{Nx', Ny' \vdash R(ssx',y')}\,(Cut)}{Nx', Ny' \vdash R(sx',sy')}\,(R.(3))}{Nx', \underline{Ny} \vdash R(sx',y)}\,(Case\ N)}\,(Case\ N) \qquad \dfrac{}{Ny \vdash R(0,y)}\,(R.(1))}{\underline{Nx}, Ny \vdash R(x,y)\ (*)}\,(Case\ N)$$

A Polynomial Procedure for Checking the Global Trace Condition

The checking procedure

Input: a CLKID$^{\omega}_{N}$ pre-proof P

(1) normalize P to a pre-proof tree-set TS that is *path-equivalent* to P and every path following its cycles is a concatenation of root-bud paths (*rb-paths*) starting from some point

(2) return *YES* if every rb-path found in a cycle of TS satisfies some derivability constraints

The normalization procedure

☞ exhaustive application of transformation operations to get a *pre-proof tree set*

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash \Delta}}{\Gamma[\sigma] \vdash \Delta[\sigma]}\,(Subst) \quad \text{becomes} \quad \dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma[\sigma] \vdash \Delta[\sigma]}\,(Subst) \qquad \dfrac{\vdots}{\Gamma \vdash \Delta\ (*)}\ \text{(new tree)}$$

$$\dfrac{\vdots}{\Gamma \vdash \Delta\ (*)} \quad \text{becomes} \quad \dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma \vdash \Delta}\,(Subst) \qquad \dfrac{\vdots}{\Gamma \vdash \Delta\ (*)}\ \text{(new tree)}$$

$$\dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma' \vdash \Delta'}\ \text{not}\ (Subst) \quad \text{becomes} \quad \dfrac{\dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma \vdash \Delta}\,(Subst)}{\Gamma' \vdash \Delta'}\ \text{not}\ (Subst)$$

The normalization procedure

☞ exhaustive application of transformation operations to get a *pre-proof tree set*

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash \Delta}}{\Gamma[\sigma] \vdash \Delta[\sigma]}\,(Subst)\ \vdots \quad \text{becomes} \quad \dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma[\sigma] \vdash \Delta[\sigma]}\,(Subst)\ \vdots \qquad \dfrac{\vdots}{\Gamma \vdash \Delta\ (*)}$$

(new tree)

$$\vdots\ \Gamma \vdash \Delta\ (*)\ \vdots \quad \text{becomes} \quad \dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma \vdash \Delta}\,(Subst)\ \vdots \qquad \vdots\ \Gamma \vdash \Delta\ (*)$$

(new tree)

$$\dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma' \vdash \Delta'}\ \text{not}\ (Subst)\ \vdots \quad \text{becomes} \quad \dfrac{\dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma \vdash \Delta}\,(Subst)}{\Gamma' \vdash \Delta'}\ \text{not}\ (Subst)\ \vdots$$

The normalization procedure

☞ exhaustive application of transformation operations to get a *pre-proof tree set*

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash \Delta}}{\Gamma[\sigma] \vdash \Delta[\sigma]}\,(Subst) \quad \text{becomes} \quad \dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma[\sigma] \vdash \Delta[\sigma]}\,(Subst) \qquad \dfrac{\vdots}{\Gamma \vdash \Delta\ (*)}\ \text{(new tree)}$$

$$\vdots \quad \Gamma \vdash \Delta\ (*) \quad \vdots \quad \text{becomes} \quad \dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma \vdash \Delta}\,(Subst) \qquad \vdots \quad \Gamma \vdash \Delta\ (*) \quad \text{(new tree)}$$

$$\dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma' \vdash \Delta'}\ \text{not}\ (Subst) \quad \vdots \qquad \text{becomes} \qquad \dfrac{\dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma \vdash \Delta}\,(Subst)}{\Gamma' \vdash \Delta'}\ \text{not}\ (Subst) \quad \vdots$$

The normalization procedure

☞ exhaustive application of transformation operations to get a *pre-proof tree set*

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash \Delta}}{\Gamma[\sigma] \vdash \Delta[\sigma]}\,(Subst) \quad \text{becomes} \quad \dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma[\sigma] \vdash \Delta[\sigma]}\,(Subst) \qquad \dfrac{\vdots}{\Gamma \vdash \Delta\ (*)}\ \text{(new tree)}$$

$$\begin{array}{c}\vdots\\ \Gamma \vdash \Delta\ (*)\\ \vdots\end{array} \quad \text{becomes} \quad \dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma \vdash \Delta}\,(Subst) \qquad \begin{array}{c}\vdots\\ \Gamma \vdash \Delta\ (*)\\ \text{(new tree)}\end{array}$$

$$\dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma' \vdash \Delta'}\ \text{not}\ (Subst) \quad \text{becomes} \quad \dfrac{\dfrac{\Gamma \vdash \Delta\ (*1)}{\Gamma \vdash \Delta}\,(Subst)}{\Gamma' \vdash \Delta'}\ \text{not}\ (Subst)$$

Properties of normalised pre-proofs

- all companions are root nodes
- the root of the input pre-proof tree is among the root nodes
- every rb-path root-bud in a pre-proof tree has this form

$$\dfrac{\dfrac{\ldots \qquad \ldots \qquad \dfrac{\text{bud}}{S'}\,(Subst)}{\vdots}}{\text{root}}$$

☞ the $(Subst)$-step is unique if it exists

Normalising the pre-proof of $N(x), N(y) \vdash R(x, y)$

☞ application of the second rule on $(\dagger)$

$$\dfrac{\dfrac{}{Ny \vdash R(0,y)}\,(R.(1)) \qquad \dfrac{\dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger\mathbf{1})}{Nx' \vdash R(x',0)}\,(Subst)}{Nx' \vdash R(sx',0)}\,(R.(2)) \qquad \dfrac{\dfrac{Nx, Ny \vdash R(x,y)\ (*)}{Nssx', Ny' \vdash R(ssx', y')}\,(Subst)}{Nx', Ny' \vdash R(ssx', y')}\,(Cut) \qquad \dfrac{}{Nx', Ny' \vdash R(sx', sy')}\,(R.(3))}{Nx', \underline{Ny} \vdash R(sx', y)}\,(Case\ N)}{\underline{Nx}, Ny \vdash R(x,y)\ (*)}\,(Case\ N)$$

$$\dfrac{\dfrac{}{\vdash R(0,0)}\,(R.(1)) \qquad \dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger)}{Nx'' \vdash R(x'',0)}\,(Subst)}{Nx'' \vdash R(sx'',0)}\,(R.(2))}{\underline{Nx'} \vdash R(x',0)\ (\dagger)}\,(Case\ N)$$

Derivability constraints

Idea: the global trace condition is implied by some derivability constraints

To each root r is attached a measure $\mathcal{M}(r)$ consisting of a multiset of IAAs of the sequent labelling r, denoted by $S(r)$.

- initially, the measures are empty multisets;
- for any rb-path $r \to b$ from a cycle, if there is a trace between an IAA A of $S(b)$ and an IAA A' of $S(r)$, then we add A to $\mathcal{M}(rc)$ and A' to $\mathcal{M}(r)$, where rc is the companion of b.
  ☞ A' is added only once if r is a the companion of b

Example of measures

$$\dfrac{\dfrac{}{Ny \vdash R(0,y)}\,(R.(1)) \qquad \dfrac{\dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger\mathbf{1})}{Nx' \vdash R(x',0)}\,(Subst)}{Nx' \vdash R(sx',0)}\,(R.(2)) \qquad \dfrac{\dfrac{Nx, Ny \vdash R(x,y)\ (*)}{Nssx', Ny' \vdash R(ssx',y')}\,(Subst)}{Nx', Ny' \vdash R(ssx',y')}\,(Cut)}{Nx', Ny' \vdash R(sx',sy')}\,(R.(3))}{Nx', \underline{Ny} \vdash R(sx',y)}\,(Case\ N)}{\underline{Nx}, Ny \vdash R(x,y)\ (*)}\,(Case\ N)$$

$$\dfrac{\dfrac{}{\vdash R(0,0)}\,(R.(1)) \qquad \dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger)}{Nx'' \vdash R(x'',0)}\,(Subst)}{Nx'' \vdash R(sx'',0)}\,(R.(2))}{\underline{Nx'} \vdash R(x',0)\ (\dagger)}\,(Case\ N)$$

$(*)$: $\{\}$ $\qquad\qquad$ $(\dagger)$: $\{\}$

☞ $(*) \to (\dagger 1)$ does not belong to any cycle

Example of measures

$$\dfrac{\dfrac{}{Ny \vdash R(0,y)}\,(R.(1)) \qquad \dfrac{\dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger\mathbf{1})}{Nx' \vdash R(x',0)}\,(Subst)}{Nx' \vdash R(sx',0)}\,(R.(2)) \qquad \dfrac{\dfrac{\dfrac{Nx, Ny \vdash R(x,y)\ (*)}{Nssx', Ny' \vdash R(ssx',y')}\,(Subst)}{Nx', Ny' \vdash R(ssx',y')}\,(Cut)}{Nx', Ny' \vdash R(sx',sy')}\,(R.(3))}{Nx', \underline{Ny} \vdash R(sx',y)}\,(Case\ N)}{\underline{Nx}, Ny \vdash R(x,y)\ (*)}\,(Case\ N)$$

$$\dfrac{\dfrac{}{\vdash R(0,0)}\,(R.(1)) \qquad \dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger)}{Nx'' \vdash R(x'',0)}\,(Subst)}{Nx'' \vdash R(sx'',0)}\,(R.(2))}{\underline{Nx'} \vdash R(x',0)\ (\dagger)}\,(Case\ N)$$

$(*)$: $\{\}$ $\qquad\qquad$ $(\dagger)$: $\{Nx'\}$

☞ $(*) \to (\dagger 1)$ does not belong to any cycle

Example of measures

$$\dfrac{\dfrac{}{Ny \vdash R(0,y)}\,(R.(1)) \qquad \dfrac{\dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger\mathbf{1})}{Nx' \vdash R(x',0)}\,(Subst)}{Nx' \vdash R(sx',0)}\,(R.(2)) \qquad \dfrac{\dfrac{Nx, Ny \vdash R(x,y)\ (*)}{Nssx', Ny' \vdash R(ssx',y')}\,(Subst)}{Nx', Ny' \vdash R(ssx',y')}\,(Cut) \qquad \dfrac{}{Nx', Ny' \vdash R(sx',sy')}\,(R.(3))}{Nx', \underline{Ny} \vdash R(sx',y)}\,(Case\ N)}{\underline{Nx}, Ny \vdash R(x,y)\ (*)}\,(Case\ N)$$

$$\dfrac{\dfrac{}{\vdash R(0,0)}\,(R.(1)) \qquad \dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger)}{Nx'' \vdash R(x'',0)}\,(Subst)}{Nx'' \vdash R(sx'',0)}\,(R.(2))}{\underline{Nx'} \vdash R(x',0)\ (\dagger)}\,(Case\ N)$$

$(*)$: $\{Ny\}$ $\qquad\qquad$ $(\dagger)$: $\{Nx'\}$

☞ $(*) \to (\dagger 1)$ does not belong to any cycle

The soundness checking

An rb-path $r \to b$ is *valid* if for any $A \in \mathcal{M}(b)$, there is $A' \in \mathcal{M}(r)$ such that there is a trace between A and A' and one can set a relation of multiset extension of the 'trace with progress points'.

Theorem
Let TS be the normalized pre-proof tree-set of a pre-proof P. If all rb-paths from the cycles of $\mathcal{TS}$ are valid, then P is a proof.

Proof. (*sketch*) The cycle is a concatenation of valid rb-paths. There is some root r with non-empty measure. For the nth occurrence of it r_n in any infinite path of the cycle, one can build a trace for each IAA from $\mathcal{M}(r_n)$ back to r_1, infinite when $n \mapsto \infty$.

By absurd, we assume that none of the traces is infinitely progressing when $n \mapsto \infty$. There should be an infinite sub-path for which none of the traces has progress points. For some $k > 0$, there is a trace along the path between r_{n-k} and r_n has a progress point, hence $\bot$. □

The soundness checking

An rb-path $r \to b$ is *valid* if for any $A \in \mathcal{M}(b)$, there is $A' \in \mathcal{M}(r)$ such that there is a trace between A and A' and one can set a relation of multiset extension of the 'trace with progress points'.

Theorem
Let TS be the normalized pre-proof tree-set of a pre-proof P. If all rb-paths from the cycles of $\mathcal{TS}$ are valid, then P is a proof.

Proof. (*sketch*) The cycle is a concatenation of valid rb-paths. There is some root r with non-empty measure. For the nth occurrence of it r_n in any infinite path of the cycle, one can build a trace for each IAA from $\mathcal{M}(r_n)$ back to r_1, infinite when $n \mapsto \infty$.

By absurd, we assume that none of the traces is infinitely progressing when $n \mapsto \infty$. There should be an infinite sub-path for which none of the traces has progress points. For some $k > 0$, there is a trace along the path between r_{n-k} and r_n has a progress point, hence $\bot$. □

Example of soundness checking

$$\dfrac{\dfrac{}{Ny \vdash R(0,y)}\,(R.(1)) \qquad \dfrac{\dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger\mathbf{1})}{Nx' \vdash R(x',0)}\,(Subst)}{Nx' \vdash R(sx',0)}\,(R.(2)) \qquad \dfrac{\dfrac{Nx, Ny \vdash R(x,y)\ (*)}{Nssx', Ny' \vdash R(ssx',y')}\,(Subst)}{Nx', Ny' \vdash R(ssx',y')}\,(Cut)}{Nx', \underline{Ny} \vdash R(sx',y)}\,(R.(3))}{Nx', \underline{Ny} \vdash R(sx',y)}\,(Case\ N)}{\underline{Nx}, Ny \vdash R(x,y)\ (*)}\,(Case\ N)$$

$$\dfrac{\dfrac{}{\vdash R(0,0)}\,(R.(1)) \qquad \dfrac{\dfrac{Nx' \vdash R(x',0)\ (\dagger)}{Nx'' \vdash R(x'',0)}\,(Subst)}{Nx'' \vdash R(sx'',0)}\,(R.(2))}{\underline{Nx'} \vdash R(x',0)\ (\dagger)}\,(Case\ N)$$

$(*)$: $\{Ny\}$ $\qquad$ $(\dagger)$: $\{Nx'\}$

☞ all rb-paths from cycles are valid

Implementation in Cyclist

Cyclist (Brotherston *et al.* [2012]): cyclic proofs for $\mathsf{FOL}_{\mathsf{ID}}$ and separation logic

- integrates the Spot model-checker (Duret-Lutz *et al.* [2016])
- proofs are developped using a breadth-first approach.
  ☞ Spot is called every time a cycle is built.

E-Cyclist : E(xtended)-Cyclist = Cyclist + our method

- the user can choose between the two checking methods.

The proof in E-Cyclist

```
0: N_1(x) /\ N_2(y) |- R_1(x,y) (N L.Unf.) [1,2]
  1: N_2(y) /\ N_3(0) |- R_1(0,y) (R R.Unf.) [3]
    3: N_2(y) /\ N_3(0) |- T (Id)
  2: N_1(z) /\ N_2(y) /\ N_3(s(z)) |- R_1(s(z),y) (N L.Unf.) [4,5]
    4: N_2(y) /\ N_3(s(0)) /\ N_4(0) |- R_1(s(0),y) (N L.Unf.) [6,7]
      6: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- R_1(s(0),0) (R R.Unf.) [8]
        8: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- R_1(0,0) (R R.Unf.) [10]
          10: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- T (Id)
      7: N_2(z) /\ N_3(s(0)) /\ N_4(0) /\ N_5(s(z)) |- R_1(s(0),s(z)) (R R.Unf.) [9]
        9: N_2(z) /\ N_3(s(0)) /\ N_4(0) /\ N_5(s(z)) |- R_1(s(s(0)),z) (N L.Unf.) [11,12]
          11: s(0)=0 /\ N_2(z) /\ N_4(0) /\ N_5(s(z)) /\ N_6(s(0)) |- R_1(s(s(0)),z) (Ex Falso)
          12: N_2(z) /\ N_3(0) /\ N_4(0) /\ N_5(s(z)) /\ N_6(s(0)) |- R_1(s(s(0)),z) (N Fold) [13]
            13: N_2(z) /\ N_3(0) /\ N_4(0) /\ N_5(s(z)) /\ N_7(s(s(0))) |- R_1(s(s(0)),z) (Weaken) [14]
              14: N_1(s(s(0))) /\ N_2(z) |- R_1(s(s(0)),z) (Subst) [15]
                15: N_1(x) /\ N_2(y) |- R_1(x,y) (Backl) [0]
    5: N_1(w) /\ N_2(y) /\ N_3(s(s(w))) /\ N_4(s(w)) |- R_1(s(s(w)),y) (N L.Unf.) [16,17]
      16: N_1(w) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(0) |- R_1(s(s(w)),0) (R R.Unf.) [18]
        18: N_1(w) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(0) |- R_1(s(w),0) (N L.Unf.) [20,21]
          20: N_3(s(s(0))) /\ N_4(s(0)) /\ N_5(0) /\ N_6(0) |- R_1(s(0),0) (R R.Unf.) [22]
            22: N_3(s(s(0))) /\ N_4(s(0)) /\ N_5(0) /\ N_6(0) |- R_1(0,0) (Weaken) [24]
              24: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- R_1(0,0) (Backl) [8]
          21: N_1(y) /\ N_3(s(s(s(y)))) /\ N_4(s(s(y))) /\ N_5(0) /\ N_6(s(y)) |- R_1(s(s(y)),0) (R R.Unf.) [23]
            23: N_1(y) /\ N_3(s(s(s(y)))) /\ N_4(s(s(y))) /\ N_5(0) /\ N_6(s(y)) |- R_1(s(y),0) (Weaken) [25]
              25: N_1(y) /\ N_3(s(s(y))) /\ N_4(s(y)) /\ N_5(0) |- R_1(s(y),0) (Subst) [26]
                26: N_1(w) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(0) |- R_1(s(w),0) (Backl) [18]
      17: N_1(w) /\ N_2(z) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(s(z)) |- R_1(s(s(w)),s(z)) (R R.Unf.) [19]
        19: N_1(w) /\ N_2(z) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(s(z)) |- R_1(s(s(s(w))),z) (N L.Unf.) [27,28]
          27: N_2(z) /\ N_3(s(s(0))) /\ N_4(s(0)) /\ N_5(s(z)) /\ N_6(0) |- R_1(s(s(s(0))),z) (N Fold) [29]
            29: N_2(z) /\ N_4(s(0)) /\ N_5(s(z)) /\ N_6(0) /\ N_7(s(s(s(0)))) |- R_1(s(s(s(0))),z) (Weaken) [30]
              30: N_1(s(s(s(0)))) /\ N_2(z) |- R_1(s(s(s(0))),z) (Subst) [31]
                31: N_1(x) /\ N_2(y) |- R_1(x,y) (Backl) [0]
          28: N_1(y) /\ N_2(z) /\ N_3(s(s(s(y)))) /\ N_4(s(s(y))) /\ N_5(s(z)) /\ N_6(s(y)) |- R_1(s(s(s(s(y)))),z) (N Fold) [32]
            32: N_1(y) /\ N_2(z) /\ N_4(s(s(y))) /\ N_5(s(z)) /\ N_6(s(y)) /\ N_7(s(s(s(s(y))))) |- R_1(s(s(s(s(y)))),z) (Weaken) [33]
              33: N_1(s(s(s(s(y))))) /\ N_2(z) |- R_1(s(s(s(s(y)))),z) (Subst) [34]
                34: N_1(x) /\ N_2(y) |- R_1(x,y) (Backl) [0]

Miss !!!
Root list: 8, 18, 0

Measures proposed for the roots in cycles:
    18: [1]
    0: [2]
Checking the link of IAAs from buds to roots:
    34 to 0: | 2 -> 2 [true ] ==> true
    31 to 0: | 2 -> 2 [true ] ==> true
    26 to 18: | 1 -> 1 [true ]| 3 -> 4 [false ]| 4 -> 1 [false ]| 5 -> 5 [false ] ==> true
    15 to 0: | 2 -> 2 [true ] ==> true
The proof has succeeded
```

Complexity

☞ polynomial

- The normalisation operations for a pre-proof of n nodes:
  #(non-root companions) + #(non-terminal $(Subst)$-nodes) +
  #(other nodes) $< 3n$
  If c is the maximal cost of an operation, the normalisation cost is $4nc$ (second operation duplicates it twice)
- The evaluation cost of a derivability constraint is $l * p$, where l is the average cardinality of a measure and p is the size of the rb-path: $\leq l * n$. The number of constraints is that of the buds from cycles: $< n$. The complexity of the evaluation step is $l * n^2$.

Statistics of the implementation

☞ benchmark (Brotherston *et al.* [2012])

Theorem	Time-E	Time	SC%	Depth	Nodes	Bckl.	Uns./All
$O_1 x \vdash N x$	2	7	61	2	9	1	0/1
$E_1 x \vee O_2 x \vdash N x$	4	11	63	3	19	2	0/4
$E_1 x \vee O_1 x \vdash N x$	2	9	77	2	13	2	2/5
$N_1 x \vdash O x \vee E x$	3	7	52	2	8	1	0/1
$N_1 x \wedge N_2 y \vdash Q(x, y)$	297	425	40	4	19	3	168/181
$N_1 x \vdash Add(x, 0, x)$	1	5	76	1	7	1	0/1
$N_1 x \wedge N_2 y \wedge Add_3(x, y, z) \vdash N z$	8	14	38	2	8	1	4/5
$N_1 x \wedge N_2 y \wedge Add_3(x, y, z) \vdash Add(x, sy, sz)$	15	22	32	2	14	1	9/10
$N_1 x \wedge N_2 y \vdash R(x, y)$	266	484	48	4	35	5	149/170

Configuration: MacBook Pro (13-inch, 2018)

- Processor 2,7 GHz Intel Core i7
- Memory 16 GB

Our procedure is semi-decidable

☞ the procedure may say 'NO' for sound cycles; it is not able to propose the right measures

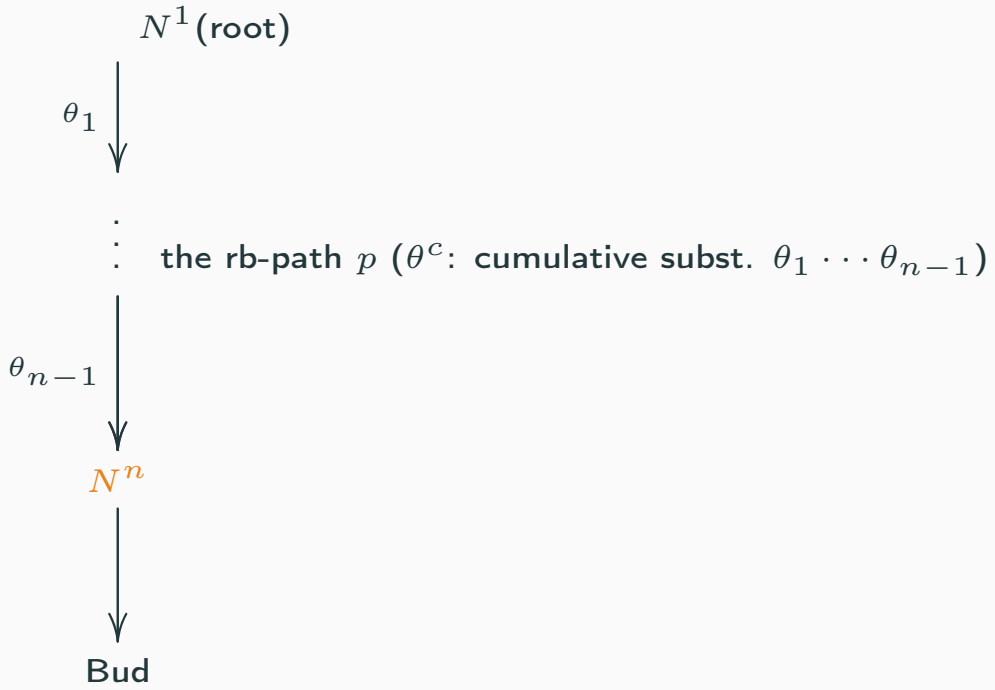
```
The proof before the abstraction step
0: N_1(x) /\ N_2(y) |- R_1(x,y) (N L.Unf.) [1,2]
  1: N_2(y) /\ N_3(0) |- R_1(0,y) (R R.Unf.) [3]
    3: N_2(y) /\ N_3(0) |- T (Id)
  2: N_1(z) /\ N_2(y) /\ N_3(s(z)) |- R_1(s(z),y) (N L.Unf.) [4,5]
    4: N_1(z) /\ N_3(s(z)) /\ N_4(0) |- R_1(s(z),0) (R R.Unf.) [6]
      6: N_1(z) /\ N_3(s(z)) /\ N_4(0) |- R_1(z,0) (Weaken) [8]
        8: N_1(z) /\ N_2(0) |- R_1(z,0) (Subst) [9]
          9: N_1(x) /\ N_2(y) |- R_1(x,y) (Backl) [0]
    5: N_1(z) /\ N_2(w) /\ N_3(s(z)) /\ N_4(s(w)) |- R_1(s(z),s(w)) (R R.Unf.) [7]
      7: N_1(z) /\ N_2(w) /\ N_3(s(z)) /\ N_4(s(w)) |- R_1(s(s(z)),w) (N L.Unf.) [10,11]
        10: N_1(z) /\ N_3(s(z)) /\ N_4(s(0)) /\ N_5(0) |- R_1(s(s(z)),0) (R R.Unf.) [12]
          12: N_1(z) /\ N_3(s(z)) /\ N_4(s(0)) /\ N_5(0) |- R_1(s(z),0) (Weaken) [14]
            14: N_1(z) /\ N_3(s(z)) /\ N_4(0) |- R_1(s(z),0) (Backl) [4]
        11: N_1(z) /\ N_2(y) /\ N_3(s(z)) /\ N_4(s(s(y))) /\ N_5(s(y)) |- R_1(s(s(z)),s(y)) (R R.Unf.) [13]
          13: N_1(z) /\ N_2(y) /\ N_3(s(z)) /\ N_4(s(s(y))) /\ N_5(s(y)) |- R_1(s(s(s(z))),y) (N L.Unf.) [15,16]
            15: N_1(z) /\ N_3(s(z)) /\ N_4(s(s(0))) /\ N_5(s(0)) /\ N_6(0) |- R_1(s(s(s(z))),0) (R R.Unf.) [17]
              17: N_1(z) /\ N_3(s(z)) /\ N_4(s(s(0))) /\ N_5(s(0)) /\ N_6(0) |- R_1(s(s(z)),0) (Weaken) [19]
                19: N_1(z) /\ N_3(s(z)) /\ N_4(s(0)) /\ N_5(0) |- R_1(s(s(z)),0) (Backl) [10]
            16: N_1(z) /\ N_2(w) /\ N_3(s(z)) /\ N_4(s(s(s(w)))) /\ N_5(s(s(w))) /\ N_6(s(w)) |- R_1(s(s(s(z))),s(w)) (R R.Unf.) [18]
              18: N_1(z) /\ N_2(w) /\ N_3(s(z)) /\ N_4(s(s(s(w)))) /\ N_5(s(s(w))) /\ N_6(s(w)) |- R_1(s(s(s(s(z)))),w) (Open)

Miss !!!
Root list: 4, 10, 0

Measures proposed for the roots in cycles:
    4: [1, 1, 1, 3, 1]
    10: [3, 1, 3, 1, 1, 3, 4, 1]
    0: [1, 1, 1, 2, 1]
Checking the link of IAAs from buds to roots:
    19 to 0: | 1 -> 1 [true ]| 3 -> 1 [false ]| 4 -> 2 [true ] ==> true
    14 to 10: | 1 -> 1 [false ]| 3 -> 3 [false ]| 4 -> 5 [false ] ==> true
    9 to 4: | 1 -> 1 [false ]| 2 -> 4 [false ] ==> false
The proof has NOT succeeded
Checking soundness starts...
Checking soundness ends, result=OK
```

Certifying Cyclic Proofs with Coq

Adding ordering constraints

Idea: build a multiset extension $<$ of a well-founded ordering over the IAAs from the measures.

☞ two IAAs $P(t_1, \ldots, t_n)$ and $P'(t'_1, \ldots, t'_{n'})$ from some trace can be compared using a rpo with a precedence where P and P' have equivalent values.



$S(N^n) < S(N^1)[\theta^c]$ (in E-Cyclist, N^n is a $(Subst)$-node)

Example

☞ any rpo ordering (we just need that x be smaller than sx, $\forall x$)

```
0: N_1(x) /\ N_2(y) |- R_1(x,y) (N L.Unf.) [1,2]
  1: N_2(y) /\ N_3(0) |- R_1(0,y) (R R.Unf.) [3]
    3: N_2(y) /\ N_3(0) |- T (Id)
  2: N_1(z) /\ N_2(y) /\ N_3(s(z)) |- R_1(s(z),y) (N L.Unf.) [4,5]
    4: N_2(y) /\ N_3(s(0)) /\ N_4(0) |- R_1(s(0),y) (N L.Unf.) [6,7]
      6: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- R_1(s(0),0) (R R.Unf.) [8]
        8: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- R_1(0,0) (R R.Unf.) [10]
          10: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- T (Id)
      7: N_2(z) /\ N_3(s(0)) /\ N_4(0) /\ N_5(s(z)) |- R_1(s(0),s(z)) (R R.Unf.) [9]
        9: N_2(z) /\ N_3(s(0)) /\ N_4(0) /\ N_5(s(z)) |- R_1(s(s(0)),z) (N L.Unf.) [11,12]
          11: s(0)=0 /\ N_2(z) /\ N_4(0) /\ N_5(s(z)) /\ N_6(s(0)) |- R_1(s(s(0)),z) (Ex Falso)
          12: N_2(z) /\ N_3(0) /\ N_4(0) /\ N_5(s(z)) /\ N_6(s(0)) |- R_1(s(s(0)),z) (N Fold) [13]
            13: N_2(z) /\ N_3(0) /\ N_4(0) /\ N_5(s(z)) /\ N_7(s(s(0))) |- R_1(s(s(0)),z) (Weaken) [14]
              14: N_1(s(s(0))) /\ N_2(z) |- R_1(s(s(0)),z) (Subst) [15]
                15: N_1(x) /\ N_2(y) |- R_1(x,y) (Backl) [0]
    5: N_1(w) /\ N_2(y) /\ N_3(s(s(w))) /\ N_4(s(w)) |- R_1(s(s(w)),y) (N L.Unf.) [16,17]
      16: N_1(w) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(0) |- R_1(s(s(w)),0) (R R.Unf.) [18]
        18: N_1(w) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(0) |- R_1(s(w),0) (N L.Unf.) [20,21]
          20: N_3(s(s(0))) /\ N_4(s(0)) /\ N_5(0) /\ N_6(0) |- R_1(s(0),0) (R R.Unf.) [22]
            22: N_3(s(s(0))) /\ N_4(s(0)) /\ N_5(0) /\ N_6(0) |- R_1(0,0) (Weaken) [24]
              24: N_3(s(0)) /\ N_4(0) /\ N_5(0) |- R_1(0,0) (Backl) [8]
          21: N_1(y) /\ N_3(s(s(s(y)))) /\ N_4(s(s(y))) /\ N_5(0) /\ N_6(s(y)) |- R_1(s(s(y)),0) (R R.Unf.) [23]
            23: N_1(y) /\ N_3(s(s(s(y)))) /\ N_4(s(s(y))) /\ N_5(0) /\ N_6(s(y)) |- R_1(s(y),0) (Weaken) [25]
              25: N_1(y) /\ N_3(s(s(y))) /\ N_4(s(y)) /\ N_5(0) |- R_1(s(y),0) (Subst) [26]
                26: N_1(w) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(0) |- R_1(s(w),0) (Backl) [18]
      17: N_1(w) /\ N_2(z) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(s(z)) |- R_1(s(s(w)),s(z)) (R R.Unf.) [19]
        19: N_1(w) /\ N_2(z) /\ N_3(s(s(w))) /\ N_4(s(w)) /\ N_5(s(z)) |- R_1(s(s(s(w))),z) (N L.Unf.) [27,28]
          27: N_2(z) /\ N_3(s(s(0))) /\ N_4(s(0)) /\ N_5(s(z)) /\ N_6(0) |- R_1(s(s(s(0))),z) (N Fold) [29]
            29: N_2(z) /\ N_4(s(0)) /\ N_5(s(z)) /\ N_6(0) /\ N_7(s(s(s(0)))) |- R_1(s(s(s(0))),z) (Weaken) [30]
              30: N_1(s(s(s(0)))) /\ N_2(z) |- R_1(s(s(s(0))),z) (Subst) [31]
                31: N_1(x) /\ N_2(y) |- R_1(x,y) (Backl) [0]
          28: N_1(y) /\ N_2(z) /\ N_3(s(s(s(y)))) /\ N_4(s(s(y))) /\ N_5(s(z)) /\ N_6(s(y)) |- R_1(s(s(s(s(y)))),z) (N Fold) [32]
            32: N_1(y) /\ N_2(z) /\ N_4(s(s(y))) /\ N_5(s(z)) /\ N_6(s(y)) /\ N_7(s(s(s(s(y))))) |- R_1(s(s(s(s(y)))),z) (Weaken) [33]
              33: N_1(s(s(s(s(y))))) /\ N_2(z) |- R_1(s(s(s(s(y)))),z) (Subst) [34]
                34: N_1(x) /\ N_2(y) |- R_1(x,y) (Backl) [0]

Miss !!!
Root list: 8, 18, 0

Measures proposed for the roots in cycles:
    18: [1]
    0: [2]
Checking the link of IAAs from buds to roots:
    34 to 0: | 2 -> 2 [true ] ==> true
    31 to 0: | 2 -> 2 [true ] ==> true
    26 to 18: | 1 -> 1 [true ]| 3 -> 4 [false ]| 4 -> 1 [false ]| 5 -> 5 [false ] ==> true
    15 to 0: | 2 -> 2 [true ] ==> true
The proof has succeeded
```

The certifying method

☞ adapt the certification method for Spike (Stratulat [2017b])

(1) to define syntactic orderings to Coccinnelle (Contejean *et al.* [2007])
   ☞ 0: `Term id_0` (S y): `Term id_S [model_nat y]`
   ☞ `less` is the wfo ordering over multisets of Coccinnelle terms

(2) attach a measure to each root formula
   ☞ `F_0:= (fun u1 u2 => (r u1 u2, ((model_nat u2)::nil))).`

(3) build the list of Fs for each strongly connected component
   ☞ `LF_0:=[F_0]`

(4) build the main lemma
   ☞ `forall F, In F LF_0 -> forall u1 u2, (forall F', In F' LF_0 -> forall e1 e2, less (snd (F' e1 e2)) (snd (F u1 u2)) -> fst (F' e1 e2)) -> fst (F u1 u2).`

(5) check all formulas in LF_0, using the well-founded induction principle
   ☞ `forall F, In F LF_0 -> forall u1 u2 u3, fst (F u1 u2).`

(6) prove the root conjectures
   ☞ `forall x y, r x y.`

Conclusions and future work

Method to effectively validate a class of CLKID$^\omega$ pre-proof trees

Related works:

- cyclic proofs with ordering constraints (Stratulat [2017a, 2018])
  ☞ the orderings are not automatically computed
- trace manifolds (Brotherston [2005]): the normalization step has exponential worst-case time complexity

Open problem:

- (strong) is there always a wfo to check a sound cycle ?
- (weak) is there always a wfo to check a sound pre-proof ?

Future work:

- automatize the certification of E-Cyclist proofs
- implementation of cyclic reasoning in Coq

Thank you !

Efficient Validation of $\mathsf{FOL}_{\mathsf{ID}}$ Cyclic Induction Reasoning

VeriDis + MATRYOSHKA Workshop, Amsterdam
June 12, 2019

Sorin Stratulat

INRIA, Université de Lorraine

References

J. Brotherston and A. Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.

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.

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.

E. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. *Frontiers of Combining Systems*, pages 148–162, 2007.

A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu. Spot 2.0 — a framework for LTL and ω-automata

manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, October 2016.

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.

S. Stratulat. Mechanically certifying formula-based Noetherian induction reasoning. *Journal of Symbolic Computation*, 80, Part 1:209–249, 2017.

S. Stratulat. Validating back-links of $\mathsf{FOL}_{\mathsf{ID}}$ 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.