Raisonnement par récurrence dans le calcul des séquents modulo

Fabrice Nahon

Phd thesis supervised by Claude Kirchner

Université Henri Poincaré

INRIA & LORIA

October 25, 2007

Necessity of induction

"Pour parvenir au plus petit théorème, il ne pourra s'affranchir de l'aide du raisonnement par récurrence parce-que c'est un instrument qui permet de passer du fini à l'infini"

H.Poincaré, "La science et l'hypothèse"

Necessity of induction

- Assume that a definition of 0, 1, and of the operation $x + 1$ is given
- Assume $0 + 1 \approx 1$, and let us define 2 by $1 + 1 \approx 2$, 3 by $2 + 1 \approx 3$ etc ...
- Let us define $x + (y + 1)$ by $x + (y + 1) \approx (x + y) + 1$ (Operations $x + 2$, $x + 3$, ... are defined *by induction*).

Necessity of induction

Assumptions:

1. $0 + 1 \approx 1$;
2. $x + (y + 1) \approx (x + y) + 1$

Conjecture: $\boxed{0 + X \approx X}$ for all integer X

- For $X = 1$, it is the first assumption.
- Let us assume that the conjecture is true for *any* X, and let us show that it is true for $X + 1$.
  - By assumption 2, $0 + (X + 1) \approx (0 + X) + 1$,
  - and since equality $0 + X \approx X$ is assumed,
  - this leads to $0 + (X + 1) \approx X + 1$ and we are done.

Necessity of induction

Since conjecture is true for $X = 1$, this leads that it is true for $X = 2$, $X = 3$, etc ...
and we have shown that it is true for *all* integer X *by induction.*

"Pourquoi donc ce jugement s'impose-t-il à nous avec une irrésistible évidence? C'est qu'il n'est que l'affirmation de la puissance de l'esprit qui se sait capable de concevoir la répétition infinie d'un même acte dès que cet acte est une fois possible"

H.Poincaré.

The Peano induction principle

The Peano induction principle is a plan to build a "Jacob's ladder":

- it consists of a base case: "proposition P is true for zero".
- and a step case: "if P is true for X, it is also true for $s(X)$, the immediate *successor* of X".

the ladder can be lengthened at any extent (proposition P is shown for *any* X) by repeating the step case.

Let us show our conjecture $\boxed{0 + X \approx X}$ with the Peano induction principle

Computation with operator $+$ can be defined by the following axioms:

$$(1)\ x + 0 \approx x \quad (2)\ x + s(y) \approx s(x + y)$$

- Base case: $0 + 0 \approx 0$, by axiom (1).
- Step case: $0 + s(X) \approx s(0 + X)$, by axiom (2)

  Now, if $0 + X \approx X$ is assumed, this leads to $0 + s(X) \approx s(X)$ and we are done.

Let us show our conjecture $\boxed{0 + X \approx X}$ with the Peano induction principle

Computation with operator $+$ can be defined by the following axioms:

$$(1)\ x + 0 \approx x \quad (2)\ x + s(y) \approx s(x + y)$$

- Base case: $0 + 0 \approx 0$, by axiom (1).
- Step case: $0 + s(X) \approx s(0 + X)$, by axiom (2)

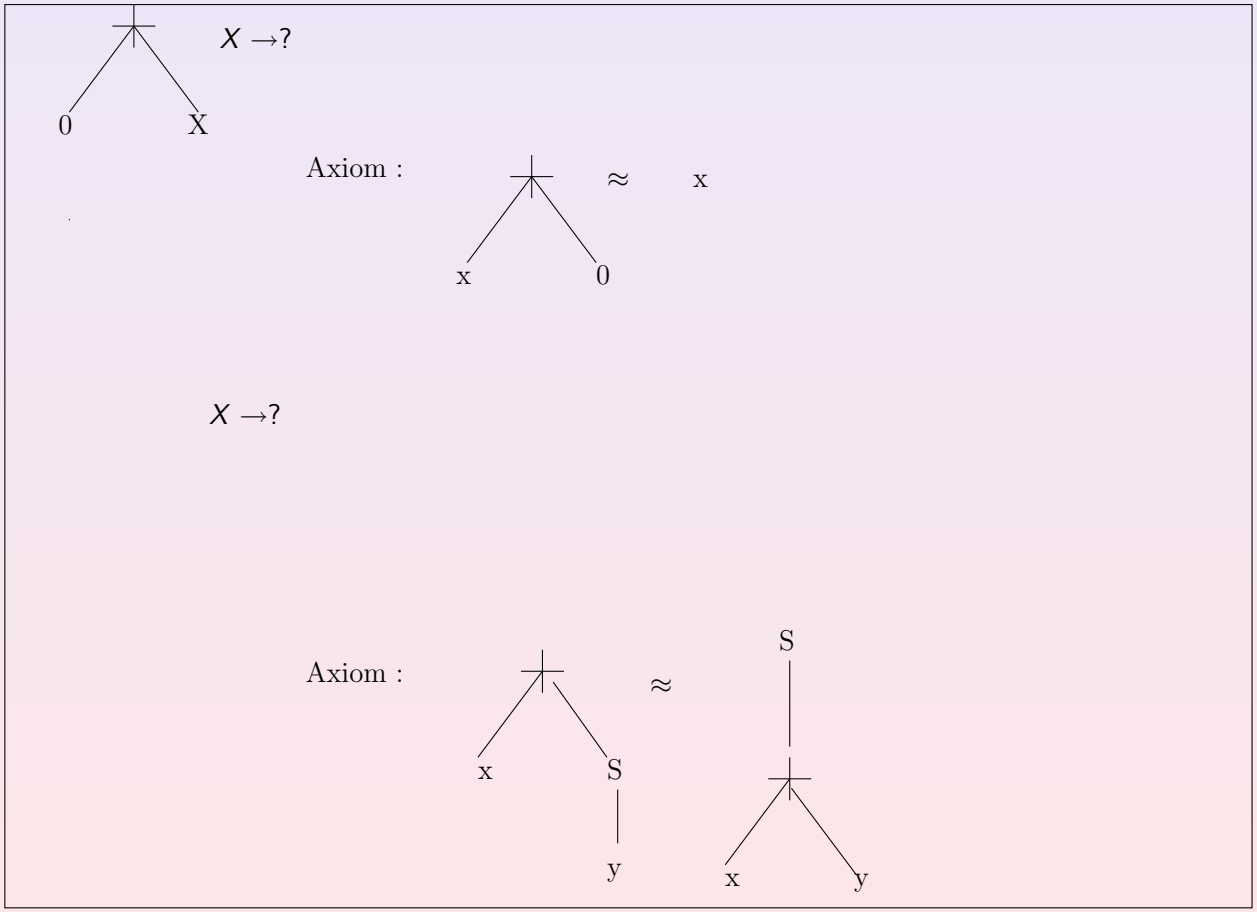
  Now, if $0 + X \approx X$ is assumed, this leads to $0 + s(X) \approx s(X)$ and we are done.

In the above proof, the replacement of X led to the application of an axiom.

This is exactly a *narrowing step*

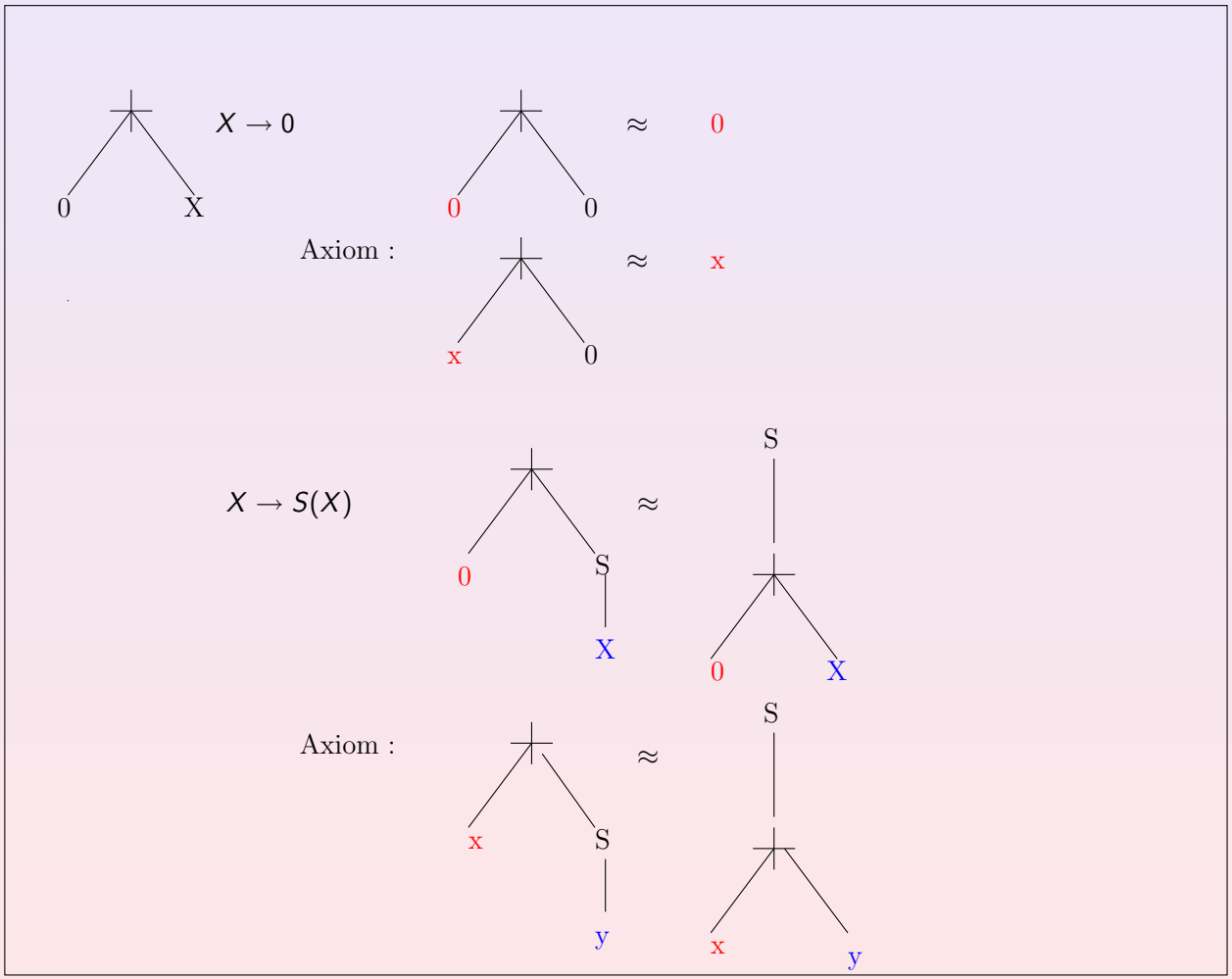
What is narrowing?

$$\boxed{0 + X \approx X}$$



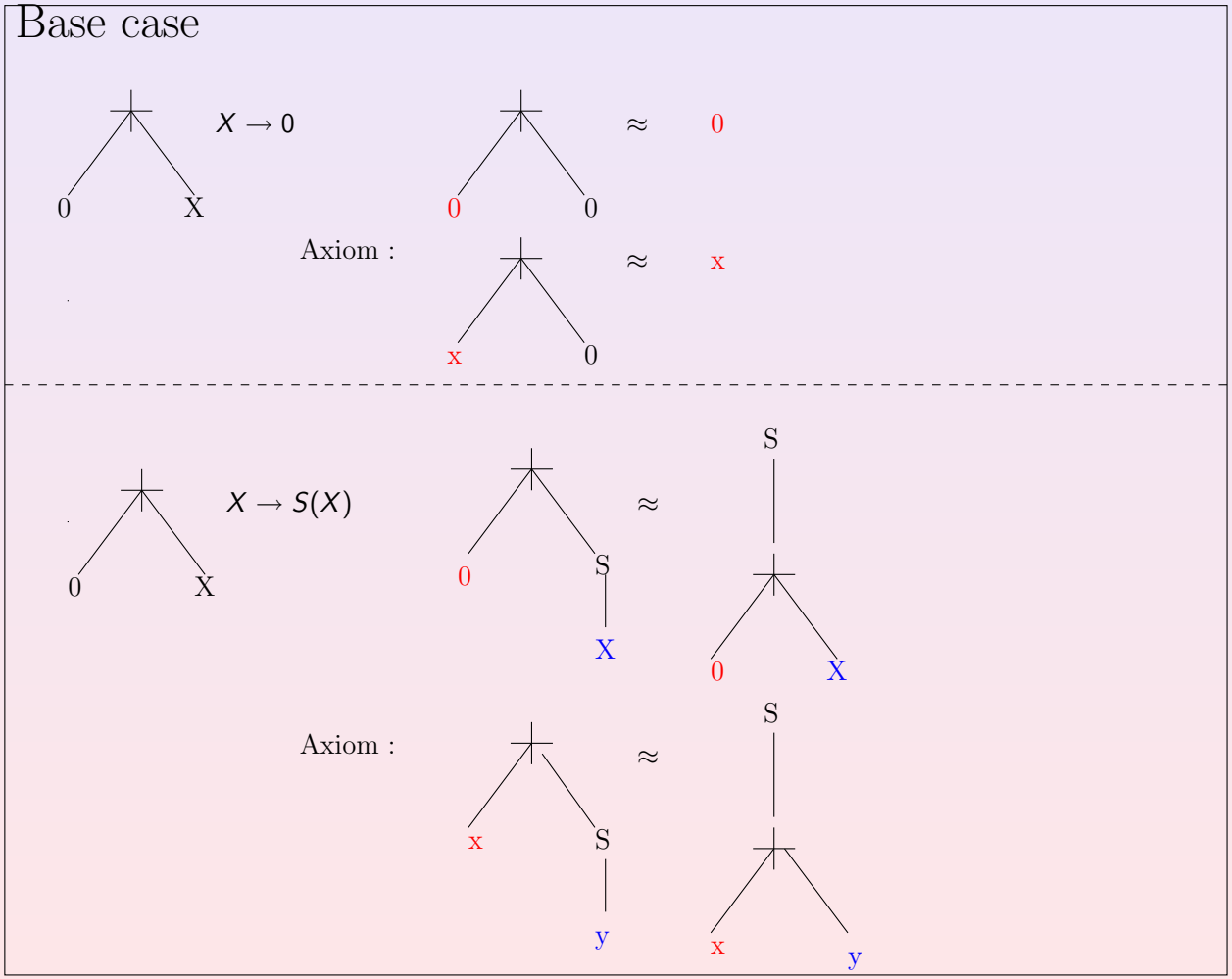
What is narrowing?

$$0 + X \approx X$$



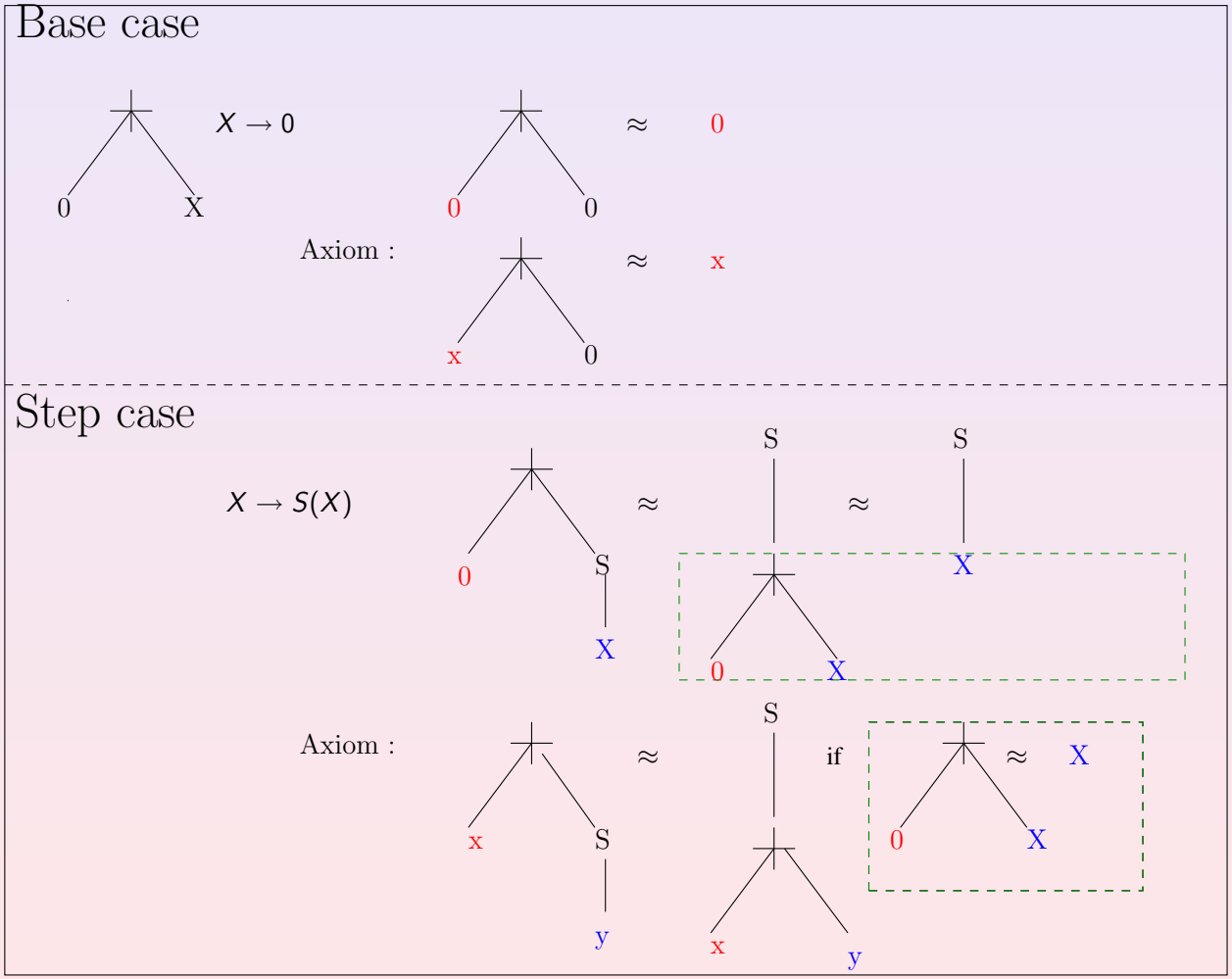
Induction vs narrowing

$$0 + X \approx X$$



Induction vs narrowing

$$0 + X \approx X$$



Induction in computer science

The Peano induction principle for integers can be generalized to any infinite set of data defined by a constructive process (lists, arrays . . .) and is therefore crucial in computer science.

Induction in computer science

The Peano induction principle for integers can be generalized to any infinite set of data defined by a constructive process (lists, arrays . . .) and is therefore crucial in computer science.

One can distinguish two approaches:

- explicit approach (COQ, ELF,HOL, Isabelle, Larch, NQTHM, PVS): the human user or an intelligent tactic must determine the most relevant induction hypothesis to show a given conjecture;

Induction in computer science

The Peano induction principle for integers can be generalized to any infinite set of data defined by a constructive process (lists, arrays . . .) and is therefore crucial in computer science.

One can distinguish two approaches:

- explicit approach (COQ, ELF,HOL, Isabelle, Larch, NQTHM, PVS): the human user or an intelligent tactic must determine the most relevant induction hypothesis to show a given conjecture;
- implicit approach (Spike, RRL ou INKA) : these automated theorem provers use the saturation and rewriting technicals to show automatically (inductive) conjectures

Induction in computer science

As a bridge between these two trends, a framework has been provided by Eric Deplagne:

- the computing steps are blindly performed without any help of the user $\longrightarrow$ implicit induction;
- the user can control the deductive steps: $\longrightarrow$ explicit induction.

His method is built over *deduction modulo* [Dowek Hardin Kirchner 1998]

What is Deduction modulo?

- A specific presentation of first-order logic
- That works *modulo* a congruence $\equiv$ on terms *and* propositions
- That makes a clear distinction between *computation* and *deduction*

Reasoning modulo

$$\frac{\Gamma, A \vdash B \quad \Gamma \vdash A, B}{\Gamma \vdash B}\text{cut}$$

Reasoning modulo

$$\frac{\Gamma, A \vdash B \quad \Gamma \vdash A, B}{\Gamma \vdash B}\text{cut}$$

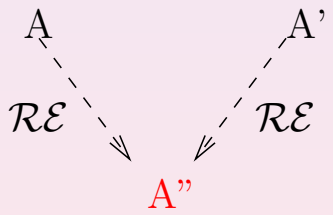
$$\frac{\Gamma, A \vdash_{\mathcal{RE}} B \quad \Gamma \vdash_{\mathcal{RE}} A', B}{\Gamma \vdash_{\mathcal{RE}} B}\text{cut}$$

Reasoning modulo

$$\frac{\Gamma, A \vdash B \quad \Gamma \vdash A, B}{\Gamma \vdash B}\text{cut}$$

$$\frac{\Gamma, A \vdash_{\mathcal{RE}} B \quad \Gamma \vdash_{\mathcal{RE}} A', B}{\Gamma \vdash_{\mathcal{RE}} B}\text{cut}$$

if:



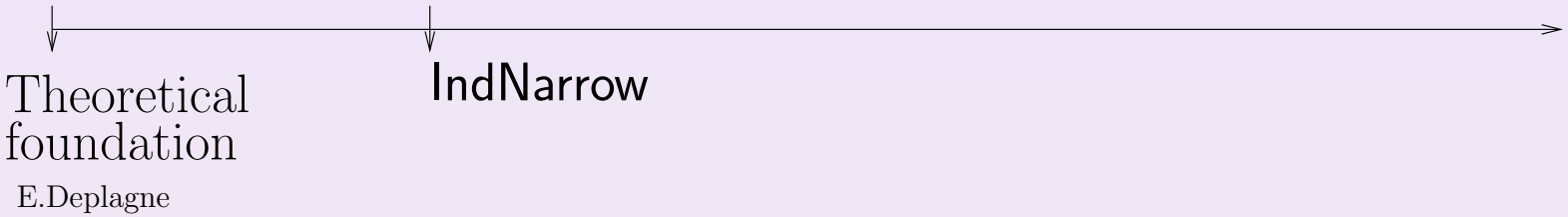
if $\equiv$-classes can be computed with $\mathcal{RE}$

Motivation:

To build an inductive proof search system, which:

- is grounded on deduction modulo;
- is narrowing based;
- performs narrowing with constructor unifiers.

The proof search system IndNarrow



- It is designed to **rewrite sequents**
- These sequents are **sequents modulo** of the form $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$

$$\boxed{\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q}$$

with $\mathcal{RE}_1$, Γ_1, $\mathcal{RE}_2$, Γ_2 such that:

- $\mathcal{RE}_1$ contains non conditional rewrite rules or equalities of the user specification;
- Γ_1 contains other axioms of the user specification;
- $\mathcal{RE}_2$ contains the induction hypotheses and other required lemmas;
- Γ_2 contains crucial propositions for induction.

Stop rules

Trivial $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} t \approx t \quad \rightarrowtail \quad \Diamond$

Refutation $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} t \approx t' \quad \rightarrowtail$ Refutation
if the proof search gets stuck

Rewriting rules

- Rewriting with the user specification:

$$\textbf{Rewrite}_1 \quad \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q \rightarrowtail \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q' \quad \text{if} \quad Q \rightarrow_{\mathcal{RE}_1} Q'$$

Rewriting rules

- Rewriting with the user specification:

$$\textbf{Rewrite}_1 \quad \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q \rightarrowtail \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q' \quad \text{if} \quad Q \rightarrow_{\mathcal{RE}_1} Q'$$

- Rewriting with the induction hypotheses or other lemmas:

$$\textbf{Rewrite}_2 \quad \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q \rightarrowtail \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q' \quad \text{if} \quad Q \rightarrow_{\mathcal{RE}_2} Q'$$

Narrowing rule: a first example

Consider the specification **Nat**:

- **Sorts:** nat.
- **Constructors:** $0 : \longrightarrow nat$; $s : nat \longrightarrow nat$;
- **Defined functions:** $+ : nat \times nat \longrightarrow nat$;
- **Rules**:

$$x + 0 \longrightarrow x$$
$$x + s(y) \longrightarrow s(x + y)$$

Narrowing rule: a first example

Consider the specification **Nat**:

- **Sorts:** nat.
- **Constructors:** $0 : \longrightarrow nat$; $s : nat \longrightarrow nat$;
- **Defined functions:** $+ : nat \times nat \longrightarrow nat$;
- **Rules**:

$$x + 0 \longrightarrow x$$
$$x + s(y) \longrightarrow s(x + y)$$

Let $\mathcal{RE}_1$ = set of rules of the specification **Nat**

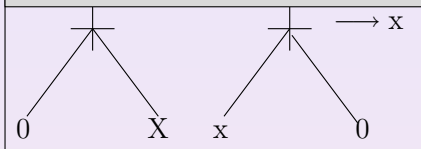
and consider the following goal: $\boxed{\emptyset | \Gamma_2 \vdash_{\mathcal{RE}_1 | \emptyset} 0 + X \approx X}$

Computing a most general unifier

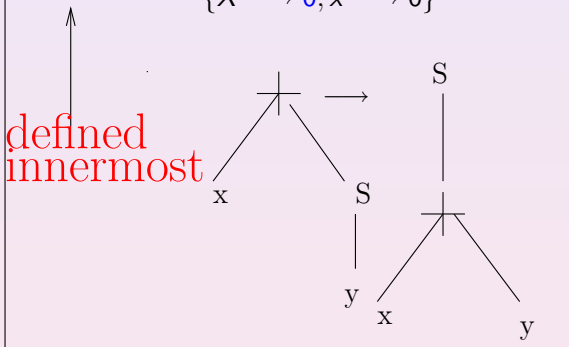
$$\emptyset | \Gamma_2 \vdash_{\mathcal{RE}_1 | \emptyset} 0 + X \approx X$$

$Q_{|\omega}$ $\quad l \longrightarrow r$

$\sigma = mgu(Q_{|\omega}, l)$



$\{X \longrightarrow 0, x \longrightarrow 0\}$



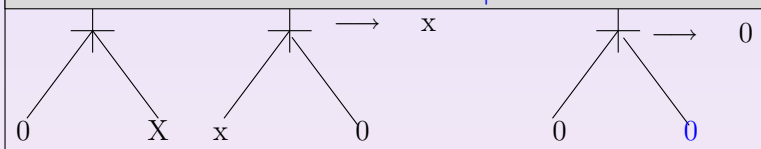
$\{X \longrightarrow S(X); x \longrightarrow 0; y \longrightarrow X\}$

Narrowing

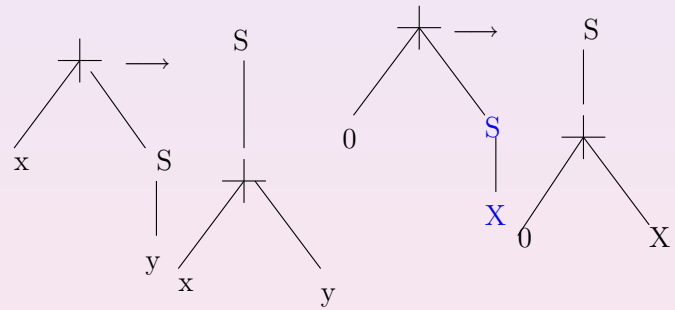
$$\emptyset | \Gamma_2 \vdash_{\mathcal{RE}_1 | \emptyset} 0 + X \approx X$$

$Q_{|\omega}$ $\quad l \longrightarrow r \quad$ $Q_{|\omega}\sigma \longrightarrow r\sigma$

$\sigma = mgu(Q_{|\omega}, l)$



$\{X \longrightarrow 0, x \longrightarrow 0\}$



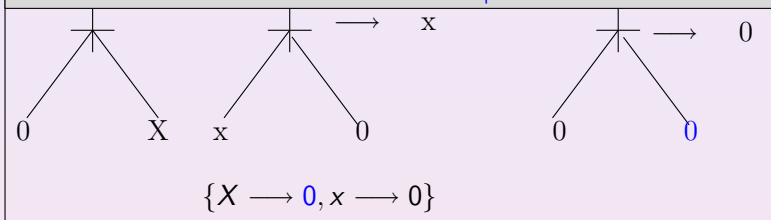
$\{X \longrightarrow S(X); x \longrightarrow 0; y \longrightarrow X\}$

Providing the subgoals

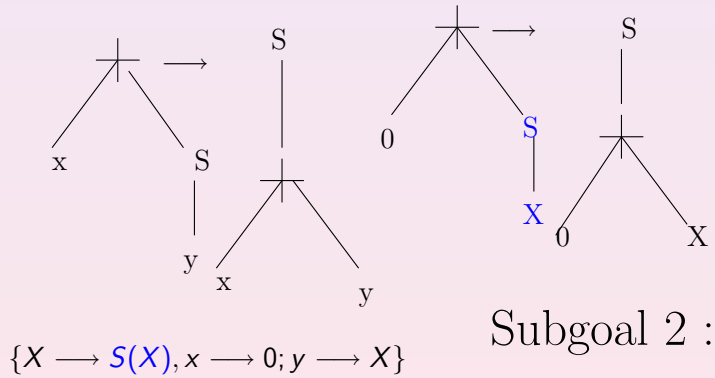
$$\boxed{\emptyset|\Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} 0 + X \approx X}$$

$Q_{|\omega}$ $\quad l \longrightarrow r$ $\quad Q_{|\omega}\sigma \longrightarrow r\sigma$ $\quad$ Subgoal: $\boxed{\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma\cup\{\mathcal{RE}_{ind}(Q,<)\sigma\}} (Q[r]_\omega)\sigma}$

$\sigma = mgu(Q_{|\omega}, l)$



Subgoal 1 : $\boxed{\emptyset|\Gamma_2 \vdash_{\mathcal{RE}_1|\{0+x\approx x \text{ if } x<0\}} 0 \approx 0}$



Subgoal 2 : $\boxed{\emptyset|\Gamma_2 \vdash_{\mathcal{RE}_1|\{0+x\approx x \text{ if } x<S(X)\}} S(0+X) \approx S(X)}$

Narrowing rule: general case

Induce $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q \rightarrowtail$

- $l \rightarrow r \in \mathcal{RE}_1$
  $\sigma = mgu(Q_{|\omega}, l)$

  $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2 \sigma \cup \{\mathcal{RE}_{ind}(Q,<)\sigma\}} (Q[r]_\omega)\sigma$

if $\omega \in DI(Q)$

Narrowing rule: general case

Induce $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q \rightarrowtail$

- $l \rightarrow r \in \mathcal{RE}_1$
  $\sigma = mgu(Q_{|\omega}, l)$

  $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2 \sigma \cup \{\mathcal{RE}_{ind}(Q,<)\sigma\}} (Q[r]_\omega)\sigma$

if $\omega \in DI(Q)$

if ω *Defined-Innermost* in Q, i.e, if $Q_{|\omega} = f t_1 \dots t_n$, with:

- f defined symbol;
- $t_1, \dots, t_n$ constructor terms.

Soundness: case of the example

$$\boxed{\emptyset | \Gamma_2 \vdash_{\mathcal{RE}_1 | \emptyset} 0 + X \approx X}$$

$\rightarrowtail$ **Induce**

$$\frac{\emptyset | \Gamma_2 \quad \vdash_{\mathcal{RE}_1 | \{0+x \approx x \text{ if } x < 0\}} 0 \approx 0 \qquad \emptyset | \Gamma_2 \quad \vdash_{\mathcal{RE}_1 | \{0+x \approx x \text{ if } x < s(X)\}} s(0+X) \approx s(X)}{\emptyset | \Gamma_2 \vdash_{\mathcal{RE}_1 | \emptyset} 0 + X \approx X}$$

Soundness: case of the example

$$\boxed{\emptyset | \Gamma_2 \vdash_{\mathcal{RE}_1 | \emptyset} 0 + X \approx X} \qquad \rightarrowtail \textbf{Induce}$$

$$\begin{array}{|l|}\hline \emptyset | \Gamma_2 \quad \vdash_{\mathcal{RE}_1 | \{0+x \approx x \text{ if } x<0\}} 0 \approx 0 \\ \hline \emptyset | \Gamma_2 \quad \vdash_{\mathcal{RE}_1 | \{0+x \approx x \text{ if } x<s(X)\}} s(0+X) \approx s(X) \\ \hline \end{array}$$

Soundness of **Induce** amounts claiming that:

there is a proof in sequent calculus modulo of the goal:

$$\boxed{\Gamma_2, X \in \mathcal{T}(\Sigma) \vdash_{\mathcal{RE}_1} 0 + X \approx X}$$

whenever there are proofs in sequent calculus modulo of the subgoals:

$$\begin{array}{|l|}\hline \Gamma_2 \quad \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \text{ if } x<0\}} 0 \approx 0 \\ \hline \Gamma_2, X \in \mathcal{T}(\Sigma) \quad \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \text{ if } x<s(X)\}} s(0+X) \approx s(X) \\ \hline \end{array}$$

Case of the example

Consider the noetherian induction principle:

$$\mathit{NoethInd}(<) \triangleq \forall P\ (\forall x\,(\forall y\,(y < x \Rightarrow P(y)) \Rightarrow P(x)) \Rightarrow \forall x\,P(x))$$

Case of the example

Consider the noetherian induction principle:

$$\mathit{NoethInd}(<) \triangleq \forall P\ (\forall x\,(\forall y\,(y < x \Rightarrow P(y)) \Rightarrow P(x)) \Rightarrow \forall x\,P(x))$$

Since $\mathit{NoethInd}(<) \subset \Gamma_2$, we can build the following derivation:

$$\boxed{\Gamma_2, X \in \mathcal{T}(\Sigma) \vdash_{\mathcal{RE}_1} 0 + X \approx X}$$

$$\vdots$$

$$\Gamma_2, X \in \mathcal{T}(\Sigma), \forall x\ (x < X \Rightarrow 0 + x \approx x) \vdash_{\mathcal{RE}_1} 0 + X \approx X.$$

The new induction hypothesis is gathered in the modulo part of the sequent:

The new induction hypothesis is gathered in the modulo part of the sequent:

$$\vdots$$

$$\Gamma_2, X \in \mathcal{T}(\Sigma) \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \text{ if } x<X\}} 0 + X \approx X$$

The new induction hypothesis is gathered in the modulo part of the sequent:

$$\vdots$$

$$\Gamma_2, X \in \mathcal{T}(\Sigma) \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \text{ if } x<X\}} 0 + X \approx X$$

and we are ready to apply the narrowing step:

The new induction hypothesis is gathered in the modulo part of the sequent:

$$\vdots$$

$$\Gamma_2, X \in \mathcal{T}(\Sigma) \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \text{ if } x<X\}} 0 + X \approx X$$

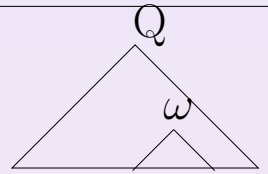
and we are ready to apply the narrowing step:

$$\vdots$$

$$\begin{array}{|ll|}\hline \Gamma_2 & \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \text{ if } x<0\}} 0 \approx 0 \\ \Gamma_2, X \in \mathcal{T}(\Sigma) & \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \text{ if } x<s(X)\}} s(0+X) \approx s(X) \\ \hline \end{array}$$

The narrowing step: going into details

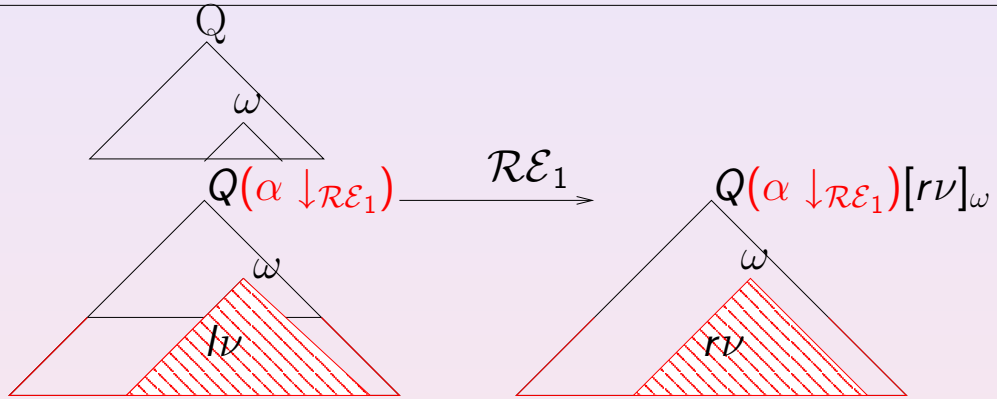
$$\Gamma_2, X \in \mathcal{T}(\Sigma) \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \text{ if } x<X\}} 0 + X \approx X$$



ω defined innermost in Q

The narrowing step: going into details

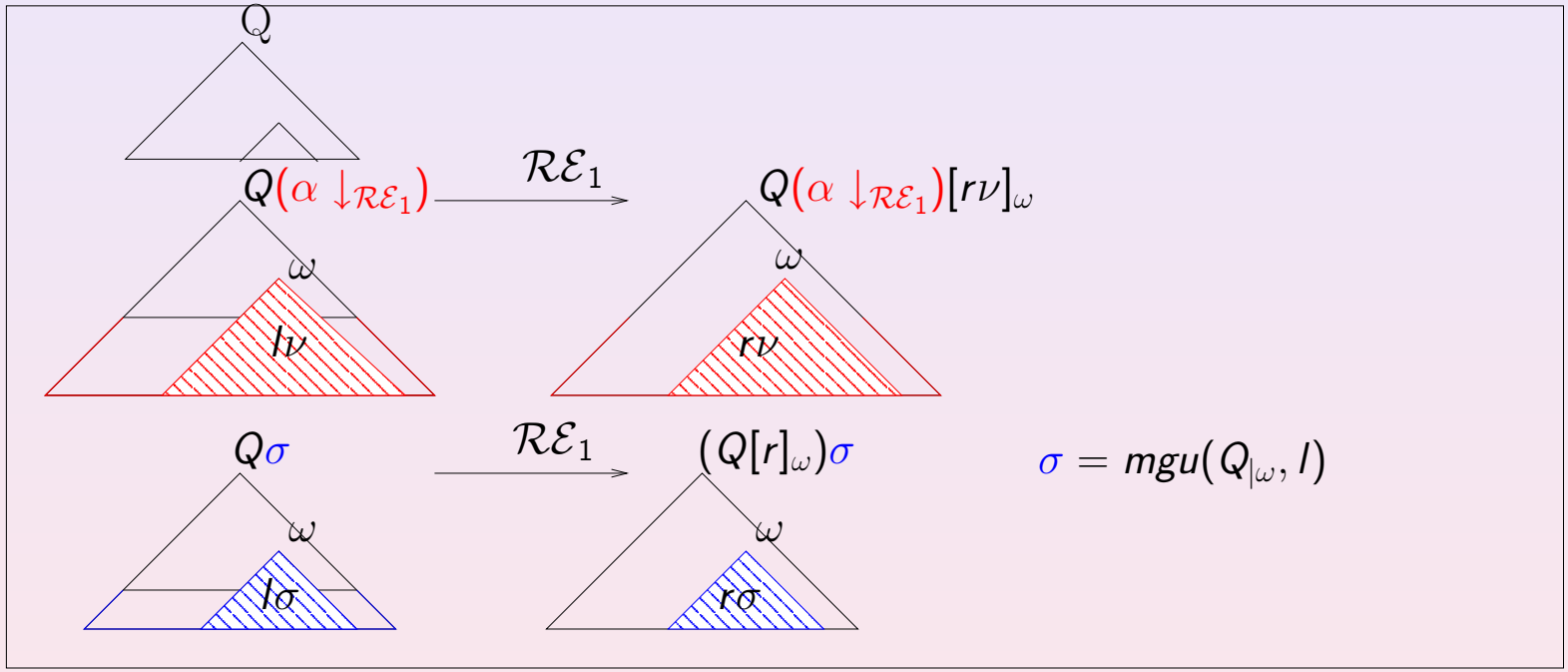
$$\Gamma_2, X \in \mathcal{T}(\Sigma) \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \text{ if } x<X\}} 0 + X \approx X$$



if $\mathcal{RE}_1$ is terminating and sufficiently complete

The narrowing step: going into details

$$\Gamma_2, X \in \mathcal{T}(\Sigma) \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \ \text{if} \ x<X\}} 0 + X \approx X$$



- $l \to r \in \mathcal{RE}_1$
  $\sigma = mgu(0 + X, l)$

$$\Gamma_2, X \in \mathcal{T}(\Sigma) \vdash_{\mathcal{RE}_1 \cup \{0+x \approx x \ \text{if} \ x<\sigma(X)\}} Q[r\sigma]_\omega$$

Assume $\mathcal{RE}_1$ terminating and sufficiently complete wrt a set of free constructors:

- **Soundness**

Theorem

$$\textit{If: } \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} Q \overset{*}{\rightarrowtail}_{\textit{IndNarrow}} \diamond$$

then sequent $\Gamma_1 \cup \Gamma_2, \overrightarrow{\mathcal{V}ar(Q)} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\emptyset} Q$ *has a proof in sequent calculus modulo;*

- **Refutational completeness**

Theorem

$$\textit{If: } \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} Q \overset{*}{\rightarrowtail}_{\textit{IndNarrow}} \textit{Refutation}$$

then sequent $\Gamma_1 \cup \Gamma_2, \overrightarrow{\mathcal{V}ar(Q)} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\emptyset} Q$ *has no proof in sequent calculus modulo.*

We have refined our system in order:

1. to apply the induction hypothesis in more cases;
2. to restrict the search space;
3. prevent infinite sequences of rewriting steps;

We have refined our system in order:

1. to apply the induction hypothesis in more cases;
2. to restrict the search space;
3. prevent infinite sequences of rewriting steps;

However, non orientable axioms can be used to increase the simplification process.

We have refined our system in order:

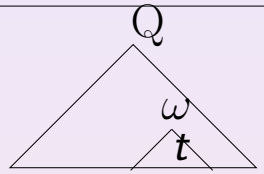
1. to apply the induction hypothesis in more cases;
2. to restrict the search space;
3. prevent infinite sequences of rewriting steps;

However, non orientable axioms can be used to increase the simplification process.

The solution consists in using equational rewriting as pioneered by Peterson and Stickel (81) and Jouannaud and Kirchner (86).

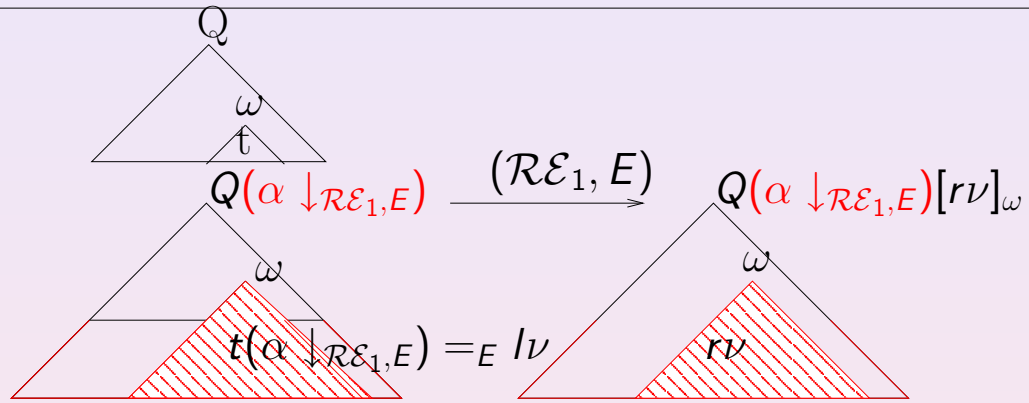
Narrowing lemma modulo: rewriting modulo step

Let us assume that E contains a finite set of equations, and consider the rewriting relation $(\mathcal{RE}_1, E)$ of Peterson and Stickel.



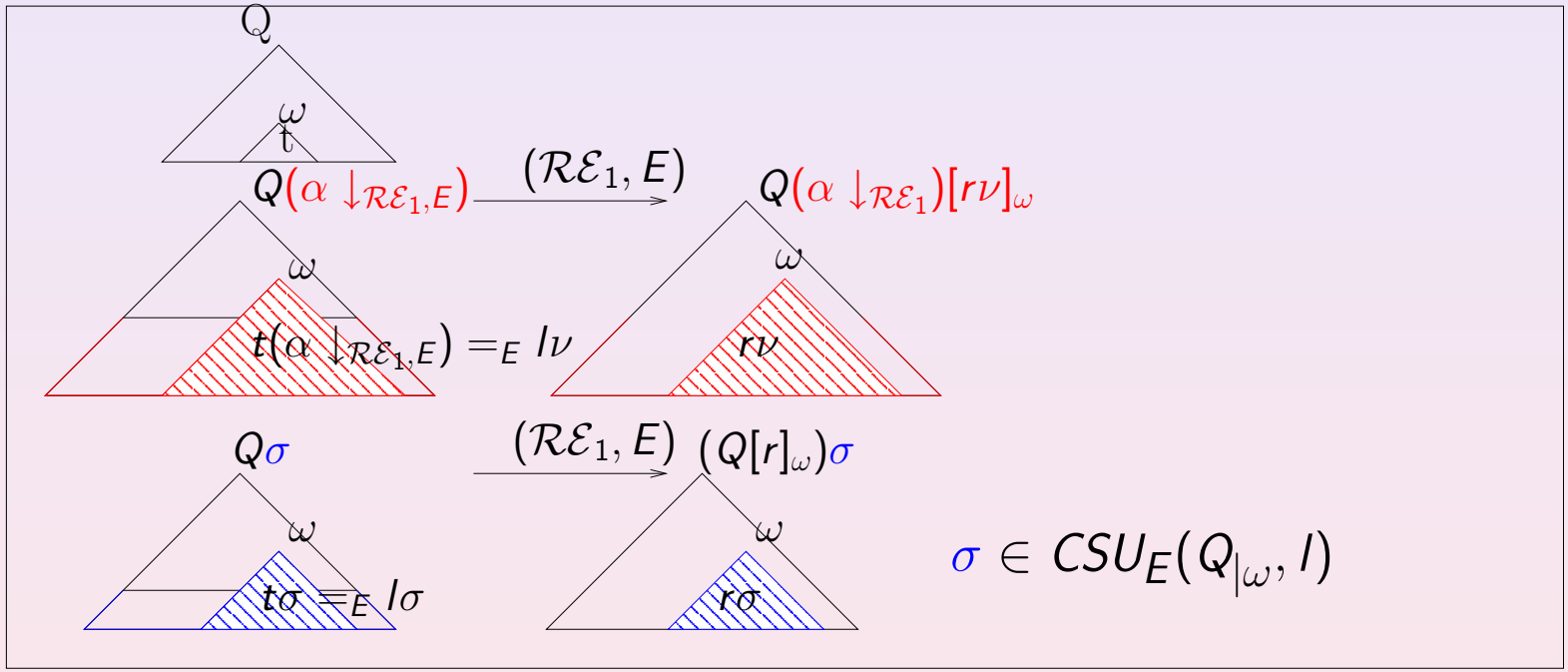
t subterm of Q at the defined innermost position ω

Narrowing lemma modulo: rewriting modulo step



if $(\mathcal{RE}_1, E)$ is terminating and sufficiently complete and since ω is *defined innermost*

Narrowing lemma: narrowing modulo step



Constructor unifiers

Moreover,
if E is constructor preserving:

- two $E-$equivalent terms are either both constructor terms or contain both defined symbols

then σ is a constructor substitution:

- it maps any variable of its domain to a constructor term

Constructor unifiers

Moreover,
if E is constructor preserving:

- two $E-$equivalent terms are either both constructor terms or contain both defined symbols

then σ is a constructor substitution:

- it maps any variable of its domain to a constructor term

A new concept: "Complete Set of Constructor Unifiers":

Correctness: every σ of $CSUC_E(s,t)$ is a constructor E-unifier of s and t;

Completeness: for any constructor E-unifier of s and t, there is $\sigma \in CSUC_E(s,t)$ and a substitution μ, such that $\theta =_E \sigma\mu\ [\mathcal{V}ar(s) \cup \mathcal{V}ar(t)]$;

Why are constructor unifiers interesting?

- since we compute $CSUC_E(Q_{|\omega}, t)$ and ω,

  they are trivial in many cases

Why are constructor unifiers interesting?

- since we compute $CSUC_E(Q_{|\omega}, t)$ and ω,

  they are trivial in many cases

Two examples ($+$ is a defined symbol):

- Let $A(+) = \{(x + y) + z \approx x + (y + z)\}$.
  Plotkin showed that $CSU_{A(+)}(x + a, a + x)$ is not finite.

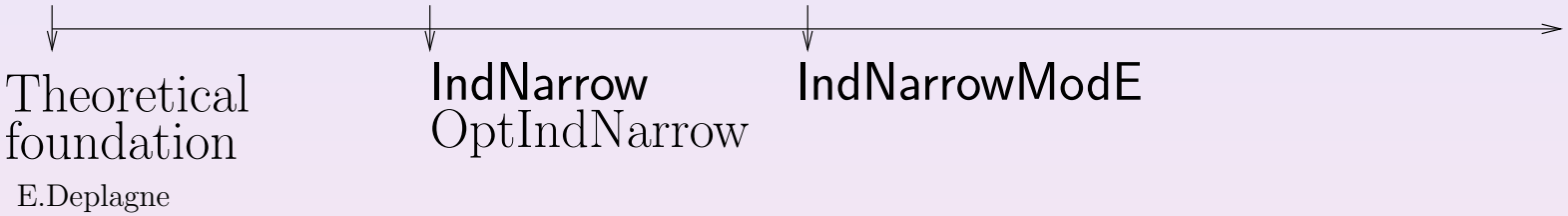
  However ...

  $CSUC_{A(+)}(x + a, a + x) = \{x \mapsto a\}$.
- Let $E(+) = \{(x + y) + z \approx (y + x) + z\}$.
  M.Echenim showed that $CSU_{E(+)}(x + y, z + y)$ is note finite.

  However ...

  $CSUC_{E(+)}(x + y, z + y) = \{x \mapsto z\}$.

The proof search system IndNarrowModE



- Stop rules: similar to IndNarrow , but E-equivalence is tested instead of equality;
- Rewriting rules: similar to IndNarrow , but we use rewriting modulo E

Narrowing rule

- in IndNarrow ,it was:

Induce $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q \rightarrowtail$

- $l \to r \in \mathcal{RE}_1$
  $\sigma = mgu(Q_{|\omega}, l)$

$\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2 \sigma \cup \{\mathcal{RE}_{ind}(Q,<)\sigma\}} (Q[r]_\omega)\sigma$
if $\omega \in DI(Q)$

- in IndNarrowModE , it is:

Induce $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q \rightarrowtail$

- $l \to r \in \mathcal{RE}_1$
  $\sigma' \in CSUC_E(Q'_{|\omega'}, l)$

$\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2 \sigma' \cup \{\mathcal{RE}_{ind}(Q,<)\sigma'\}} (Q'[r]_{\omega'})\sigma'$
if $Q' =_E Q$ and $\omega' \in DI(Q')$

Assume $\mathcal{RE}_1$ E-terminating, $(\mathcal{RE}_1, E)$-sufficiently complete and E *constructor preserving*:

Assume $\mathcal{RE}_1$ E-terminating, $(\mathcal{RE}_1, E)$-sufficiently complete and E *constructor preserving*:

- **Soundness**

Theorem

$$\textit{If: } \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} Q \overset{*}{\rightarrowtail}_{\textit{IndNarrowModE}} \diamond$$

then sequent $\Gamma_1 \cup \Gamma_2, \overrightarrow{\mathcal{V}ar(Q)} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\emptyset} Q$ *has a proof in sequent calculus modulo;*

- **Refutational completeness**

Theorem

$$\textit{If: } \Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} Q \overset{*}{\rightarrowtail}_{\textit{IndNarrowModE}} \textit{Refutation}$$

then sequent $\Gamma_1 \cup \Gamma_2, \overrightarrow{\mathcal{V}ar(Q)} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\emptyset} Q$ *has no proof in sequent calculus modulo.*

Main advantage/Main disadvantage

- Main advantage: performs unification with constructor unifiers;

Main advantage/Main disadvantage

- Main advantage: performs unification with constructor unifiers;
- Main disadvantage: requires to compute the E-equivalence class of the goal before any application of the narrowing step.

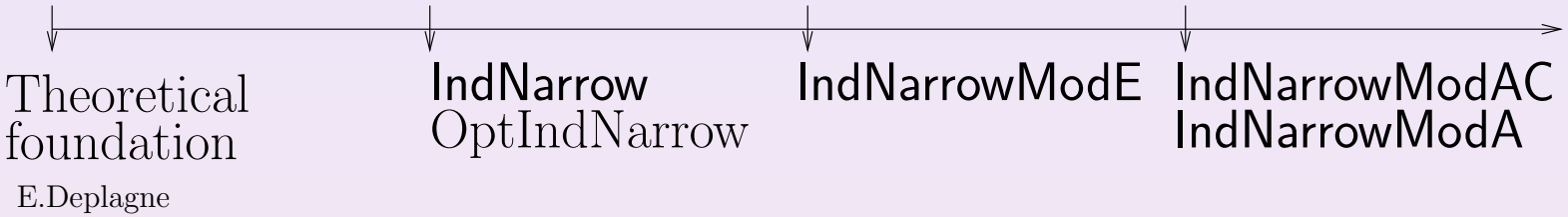
Main advantage/Main disadvantage

- Main advantage: performs unification with constructor unifiers;
- Main disadvantage: requires to compute the E-equivalence class of the goal before any application of the narrowing step.

In the case of associative commutative or associative theories, we avoid this problem by working with flattened goals [Marché93]:

- flattening a term amounts to write it without brackets:
  - Example: $\overline{((x + y) + z) + t} = x + y + z + t$

The proof search systems IndNarrowModAC and IndNarrowModA



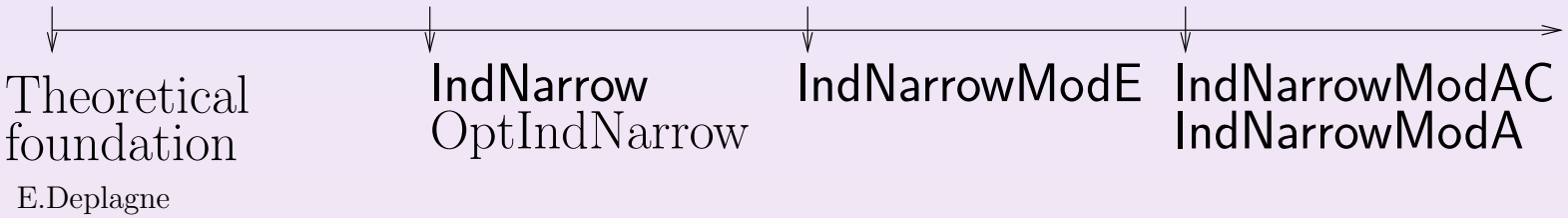
In the associative commutative case, the narrowing rule becomes:

InduceAC $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} \bar{Q} \rightarrowtail$

- $l \rightarrow r \in \mathcal{RE}_1$
  $\sigma \in CSUC_{AC}(\bar{Q}_{|\omega}, l)$

  $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2 \sigma \cup \{\mathcal{RE}_{ind}(Q,<)\sigma\}} (\bar{Q}[\bar{r}]_\omega)\sigma$
  if $\omega \in DI(\bar{Q})$

The proof search systems IndNarrowModAC and IndNarrowModA



In the associative commutative case, the narrowing rule becomes:

InduceAC $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} \bar{Q} \rightarrowtail$

- $l \to r \in \mathcal{RE}_1$
  $\sigma \in CSUC_{AC}(\bar{Q}_{|\omega}, l)$

$\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2 \sigma \cup \{\mathcal{RE}_{ind}(Q,<)\sigma\}} (\bar{Q}[\bar{r}]_\omega)\sigma$
if $\omega \in DI(\bar{Q})$

In the associative case, it is similar.

An *AC*-example

Let us consider the specification **Simple arithmetic:**

- **Sorts:** nat
- **constructors:** $0 : \ \longrightarrow nat$ $\qquad$ $s : \ nat \longrightarrow nat$
- **defined functions:**
  $+ : \ nat \times nat \longrightarrow nat$ $\qquad$ $* : \ nat \times nat \longrightarrow nat$
  $exp : \ nat \longrightarrow nat$
- **rewrite rules**:

$$x + 0 \longrightarrow x \qquad x * 0 \longrightarrow 0$$
$$x + s(y) \longrightarrow s(x + y) \qquad x * s(y) \longrightarrow x * y + x$$

$$exp(x, 0) \longrightarrow s(0)$$
$$exp(x, s(y)) \longrightarrow x * exp(x, y)$$

- **Equational axioms**: $AC(+, *)$

An *AC*-example

Consider the following goal:

$$\emptyset, A(<>) | \Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} exp(X * Y, N) \approx exp(X, N) * exp(Y, N)$$

An *AC*-example

Consider the following goal:

$$\emptyset, A(<>) | \Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} exp(X * Y, N) \approx exp(X, N) * exp(Y, N)$$

- Let us apply **InduceAC** at position 2.1.

An *AC*-example

Consider the following goal:

$$\emptyset, A(<>) | \Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} exp(X * Y, N) \approx exp(X, N) * exp(Y, N)$$

- Let us apply **InduceAC** at position 2.1.

I	$CSUC_{AC(+,*)}(exp(X, N), I)$
$exp(x, 0)$	$\sigma_1 = \{X \to X_1; N \to 0; x \to X_1\}$
$exp(x, s(y))$	$\sigma_2 = \{X \to X_1; N \to s(N_1); x \to X_1; y \to N_1\}$

- After normalization, we obtain the subgoals:

- After normalization, we obtain the subgoals:

$\emptyset, AC(+,*) \vert \Gamma_2$	$\vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_1}$ $s(0) \approx s(0)$
$\emptyset, AC(+,*) \vert \Gamma_2$	$\vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_2}$ $X_1 * Y_1 * exp(X_1 * Y_1, N_1) \approx$ $X_1 * exp(X_1, N_1) * Y_1 * exp(Y_1, N_1)$

- After normalization, we obtain the subgoals:

$\emptyset, AC(+,*) \vert \Gamma_2$	$\vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_1}$ $s(0) \approx s(0)$
$\emptyset, AC(+,*) \vert \Gamma_2$	$\vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_2}$ $X_1 * Y_1 * exp(X_1 * Y_1, N_1) \approx$ $X_1 * exp(X_1, N_1) * Y_1 * exp(Y_1, N_1)$

- **TrivialAC** gets rid of this first subgoal, and since $N_1 < s(N_1)$, the induction hypothesis can be applied to the second one.

The example with a method based on inductive schemes [Spike-ac: Bouhoula-Berregeb-96]

$$\boxed{exp(X * Y, N) \approx exp(X, N) * exp(Y, N)}$$

1. computation of induction variables: X, Y, N;
2. computation of a test set: $\{0, s(x)\}$;
3. instanciation of induction variables by elements of the test set and reduction;

$$\boxed{exp(X * Y, N) \approx exp(X, N) * exp(Y, N)}$$

A test instance is therefore:

$$\boxed{exp(s(X_1) * s(Y_1), s(N_1)) \approx \quad exp(s(X_1), s(N_1)) * exp(s(Y_1), s(N_1))}$$

$$\boxed{exp(X * Y, N) \approx exp(X, N) * exp(Y, N)}$$

A test instance is therefore:

$$\boxed{exp(s(X_1) * s(Y_1), s(N_1)) \approx \quad exp(s(X_1), s(N_1)) * exp(s(Y_1), s(N_1))}$$

after simplification, this leads to the subgoal:

$$\boxed{\begin{array}{l} s(X_1) * s(Y_1) * exp(s(X_1 + Y_1 + X_1 * Y_1), N_1) \approx \\ s(X_1) * exp(s(X_1), N_1) * s(Y_1) * exp(s(Y_1), N_1) \end{array}}$$

which cannot be simplified by the induction hypothesis

The example with another method based on narrowing at *defined-innermost* positions ([Aoto-RTA-06])

$$\boxed{exp(X * Y, N) \approx exp(X, N) * exp(Y, N)}$$

- the narrowing step must be performed in the member of the goal which is greater:

The example with another method based on narrowing at *defined-innermost* positions ([Aoto-RTA-06])

$$\boxed{exp(X * Y, N) \approx exp(X, N) * exp(Y, N)}$$

- the narrowing step must be performed in the member of the goal which is greater:
  only at position 1.1 of the goal.

The example with another method based on narrowing at *defined-innermost* positions ([Aoto-RTA-06])

$$\boxed{exp(X * Y, N) \approx exp(X, N) * exp(Y, N)}$$

- the narrowing step must be performed in the member of the goal which is greater:
  only at position 1.1 of the goal.
  we obtain:

$$\boxed{exp(X_1 * Y_1 + X_1, N_1) \approx \quad exp(X_1, N_1) * exp(s(Y_1), N_1)}$$

which cannot be simplified by the induction hypothesis.

An A-example

Consider the specification **List**:

- **Sorts:** nat, list.
- **Constructors:**

$$0 : \rightarrow nat; \quad s : nat \rightarrow nat;$$
$$Nil : \rightarrow list; \quad :: : nat \times list \rightarrow list.$$

- **Defined functions:**

$$+ : nat \times nat \rightarrow nat; \quad <> : list \times list \rightarrow list;$$
$$rev : list \rightarrow list.$$

- **Rules**:

$$0 + x \rightarrow x \qquad rev(Nil) \rightarrow Nil$$
$$s(x) + y \rightarrow s(x + y) \qquad rev(x :: l) \rightarrow rev(l) <> (x :: Nil)$$

$$Nil <> l \rightarrow l$$
$$(x :: l) <> m \rightarrow x :: (l <> m)$$

Consider the following goal:

$$\boxed{\emptyset, A(<>) | \Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} rev(L <> M) \approx rev(M) <> rev(L)}$$

Consider the following goal:

$$\emptyset, A(<>)|\Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} rev(L <> M) \approx rev(M) <> rev(L)$$

- Let us apply **InduceA** at position 1.1.

l	$CSUC_{A(<>)}(L <> M, l)$
$Nil <> l$	$\sigma_1 = \{L \to Nil; M \to M_1; l \to M_1\}$
$(x :: l) <> m$	$\sigma_2 = \{L \to X_1 :: L_1; M \to M_1;$ $x \to X_1; l \to L_1; m \to M_1\}$

$$\emptyset, A(<>) | \Gamma_2 \vdash_{\mathcal{RE}_1 | \emptyset} rev(L <> M) \approx rev(M) <> rev(L)$$

- After normalization, we obtain the subgoals:

$\emptyset, A(<>) \| \Gamma_2$	$\vdash_{\mathcal{RE}_1 \| \mathcal{RE}_{ind}(Q)\sigma_1}$ $rev(M_1) \approx rev(M_1)$
$\emptyset, A(<>) \| \Gamma_2$	$\vdash_{\mathcal{RE}_1 \| \mathcal{RE}_{ind}(Q)\sigma_2}$ $rev(L_1 <> M_1) <> (X_1 :: Nil) \approx$ $rev(M_1) <> rev(L_1) <> (X_1 :: Nil)$

$$\emptyset, A(<>)|\Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} rev(L <> M) \approx rev(M) <> rev(L)$$

- After normalization, we obtain the subgoals:

$\emptyset, A(<>)\|\Gamma_2$	$\vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_1}$ $rev(M_1) \approx rev(M_1)$
$\emptyset, A(<>)\|\Gamma_2$	$\vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_2}$ $rev(L_1 <> M_1) <> (X_1 :: Nil) \approx$ $rev(M_1) <> rev(L_1) <> (X_1 :: Nil)$

- **TrivialA** gets rid of this first subgoal, and since $(L_1, M_1) <_2 (X_1 :: L_1, M_1)$, the induction hypothesis can be applied to the second one (rule **Rewrite$_2$A**).

Main contributions

We have provided a progressive family of proof search systems for inductive theorem proving:

1. IndNarrow : shown sound and refutationally complete
2. OptIndNarrow : optimization of IndNarrow
3. IndNarrowModE :
   - designed for rewriting modulo a set E of equalities
   - performs narrowing with constructor unifiers
4. IndNarrowModAC , IndNarrowModA :refinement to rewriting modulo AC and A
   - no need to compute the equivalence class of the goal when performing the narrowing step.

This work takes place in the following constructive process:

- building a proof of an inductive conjecture with our system;

This work takes place in the following constructive process:

- building a proof of an inductive conjecture with our system;
- associating to this proof another proof in sequent calculus modulo;

This work takes place in the following constructive process:

- building a proof of an inductive conjecture with our system;
- associating to this proof another proof in sequent calculus modulo;
- translating the proof in sequent calculus modulo into a proof term, whose type is the initial conjecture.

Our system is therefore designed to collaborate with other proof assistants in a safe way, in order:

Our system is therefore designed to collaborate with other proof assistants in a safe way, in order:

1. to find the most convenient noetherian ordering and required lemmas;

Our system is therefore designed to collaborate with other proof assistants in a safe way, in order:

1. to find the most convenient noetherian ordering and required lemmas;
2. to check the proof.

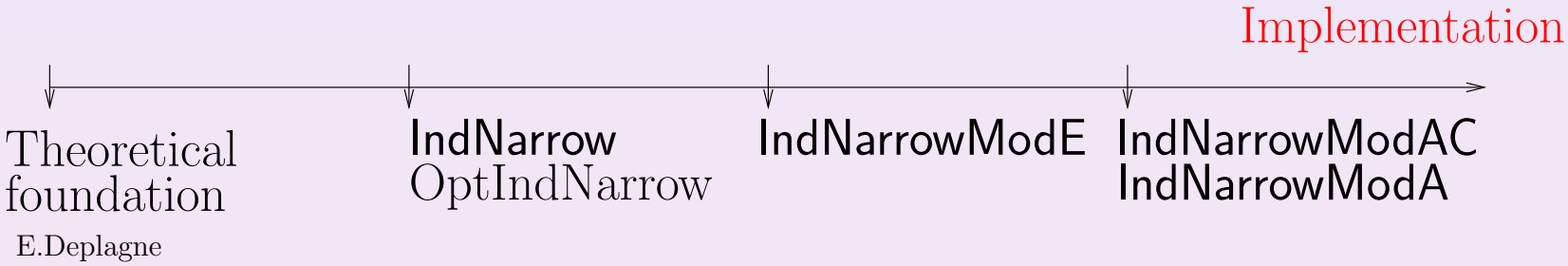
Our system is therefore designed to collaborate with other proof assistants in a safe way, in order:

1. to find the most convenient noetherian ordering and required lemmas;
2. to check the proof.

Allthough It is not yet implemented, this work can provide proof assistants a theoretical foundation based on deduction modulo.

Future work

- Constructor unification;
- Implementation



  - this novel sequent based calculus permits associative-commutative and associative theories to be deal with in an efficient manner
  - the method is quite easy for a user
  - it could be translated into a new tactic for Coq?
- Generalization to clauses

Justification

Let $E = AC$ (resp $E = A$)

Theorem

If $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail_{IndNarrowModE} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} R$, *then* $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \bar{Q} \rightarrowtail_{IndNarrowModAC} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} \bar{R}$ *(resp* $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \bar{Q} \rightarrowtail_{IndNarrowModA} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} \bar{R}$*).*

1.

Theorem

If $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \bar{Q} \rightarrowtail_{IndNarrowModAC} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} \bar{R}$, *there exists* R' *such that* $R' =_{AC} R$ *(resp* $R' =_A R$*), and* $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail_{IndNarrowModE} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} R'$

2.