



# ***Hierarchical Reflection***

***TPHOLs, 15 September 2004***

Luís Cruz-Filipe

Radboud University of Nijmegen, Netherlands

Center for Logic and Computation, Portugal

(joint work with Freek Wiedijk)

# ***Motivation***



# ***Motivation***

FTA-project at (then) University of Nijmegen

# ***Motivation***

FTA-project at (then) University of Nijmegen

Tactic to prove equalities in fields

# *Motivation*

FTA-project at (then) University of Nijmegen

Tactic to prove equalities in fields

- ⑥ intuitively “admissible” in simpler structures

# *Motivation*

FTA-project at (then) University of Nijmegen

Tactic to prove equalities in fields

- ⑥ intuitively “admissible” in simpler structures
- ⑥ uses partial reflection. . .

# *Motivation*

FTA-project at (then) University of Nijmegen

Tactic to prove equalities in fields

- ⑥ intuitively “admissible” in simpler structures
- ⑥ uses partial reflection...

Goal: hierarchy of tactics parallel to hierarchy of structures

# ***Contents***



# ***Contents***

## 1. Motivation

# ***Contents***

1. Motivation
2. Reflection

# ***Contents***

1. Motivation
2. Reflection
3. Partial reflection

# ***Contents***

1. Motivation
2. Reflection
3. Partial reflection
4. “Hierarchical” reflection

# ***Contents***

1. Motivation
2. Reflection
3. Partial reflection
4. “Hierarchical” reflection
5. Conclusions

# ***Reflection***



# ***Reflection***

Given: a predicate  $P$  on a domain  $\mathcal{D}$

# Reflection

Given: a predicate  $P$  on a domain  $\mathcal{D}$

Decision procedure  $f$  for (subset  $\mathcal{E}$  of)  $\mathcal{D}$

# Reflection

Given: a predicate  $P$  on a domain  $\mathcal{D}$

Decision procedure  $f$  for (subset  $\mathcal{E}$  of)  $\mathcal{D}$

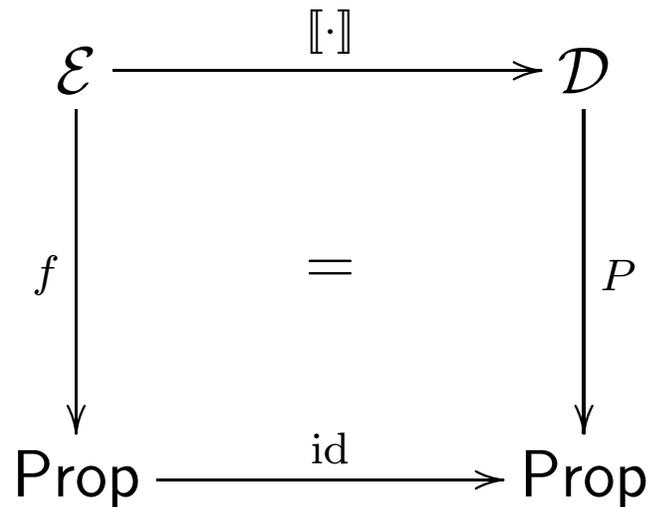
such that  $f(e) = 1 \iff P(\llbracket e \rrbracket)$

# Reflection

Given: a predicate  $P$  on a domain  $\mathcal{D}$

Decision procedure  $f$  for (subset  $\mathcal{E}$  of)  $\mathcal{D}$

such that  $f(e) = 1 \iff P(\llbracket e \rrbracket)$



# *Partial Reflection*



# *Partial Reflection*

Generalization where  $\llbracket \cdot \rrbracket : \mathcal{E} \rightarrow \mathcal{D}$  is replaced by  $\llbracket \subseteq \mathcal{E} \times \mathcal{D}$

# *Partial Reflection*

Generalization where  $\llbracket \cdot \rrbracket : \mathcal{E} \rightarrow \mathcal{D}$  is replaced by  $\llbracket \subseteq \mathcal{E} \times \mathcal{D}$

- ⑥ not (necessarily) total

# Partial Reflection

Generalization where  $\llbracket \cdot \rrbracket : \mathcal{E} \rightarrow \mathcal{D}$  is replaced by  $\llbracket \subseteq \mathcal{E} \times \mathcal{D}$

- ⑥ not (necessarily) total
- ⑥ not (necessarily) functional

# Partial Reflection

Generalization where  $\llbracket \cdot \rrbracket : \mathcal{E} \rightarrow \mathcal{D}$  is replaced by  $\llbracket \cdot \rrbracket \subseteq \mathcal{E} \times \mathcal{D}$

- ⑥ not (necessarily) total
- ⑥ not (necessarily) functional

used to get past induction-recursion required by

$$\llbracket e/f \rrbracket = \llbracket e \rrbracket / \llbracket f \rrbracket$$

# Partial Reflection

Generalization where  $\llbracket \cdot \rrbracket : \mathcal{E} \rightarrow \mathcal{D}$  is replaced by  $\llbracket \cdot \rrbracket \subseteq \mathcal{E} \times \mathcal{D}$

- ⑥ not (necessarily) total
- ⑥ not (necessarily) functional

used to get past induction-recursion required by

$$\llbracket e/f \rrbracket = \llbracket e \rrbracket / \llbracket f \rrbracket$$

(we can now write  $e \llbracket x \rightarrow f \rrbracket y \rightarrow e/f \llbracket x/y \rrbracket$ )

# ***Our tactic: Rational***



# ***Our tactic: Rational***

In our situation:

# ***Our tactic: Rational***

In our situation:

- ⑥  $\mathcal{D}$  is a field

# ***Our tactic: Rational***

In our situation:

- ⑥  $\mathcal{D}$  is a field
- ⑥  $P(x, y) := (x = y)$

# ***Our tactic: Rational***

In our situation:

- ⑥  $\mathcal{D}$  is a field
- ⑥  $P(x, y) := (x = y)$
- ⑥  $f$  computes  $\mathcal{N}(x - y)$  and checks whether it outputs 0

# ***How the normalization works***



# *How the normalization works*

For any expression  $e$ ,  $\mathcal{N}(e) = p_1/p_2$

# *How the normalization works*

For any expression  $e$ ,  $\mathcal{N}(e) = p_1/p_2$

$p_1$  and  $p_2$  are polynomials on some variables fully expanded and sorted

# *How the normalization works*

For any expression  $e$ ,  $\mathcal{N}(e) = p_1/p_2$

$p_1$  and  $p_2$  are polynomials on some variables fully expanded and sorted

take  $p_2 = 1$ : we get a tactic for rings!

# *How the normalization works*

For any expression  $e$ ,  $\mathcal{N}(e) = p_1/p_2$

$p_1$  and  $p_2$  are polynomials on some variables fully expanded and sorted

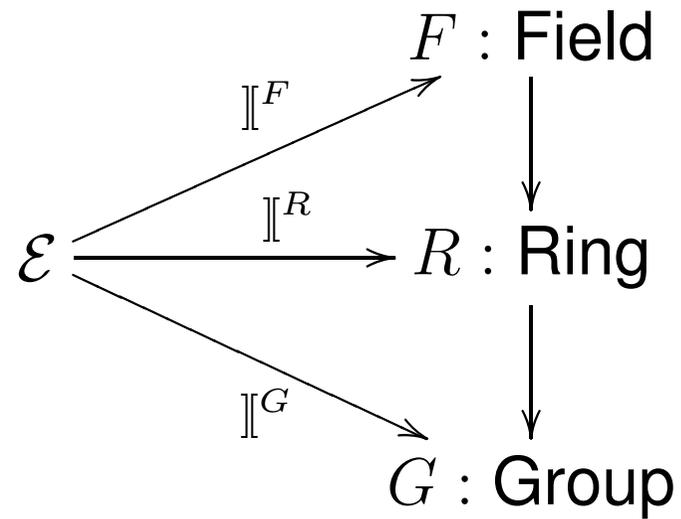
take  $p_2 = 1$ : we get a tactic for rings!

(actually even in abelian groups with some extra work...)

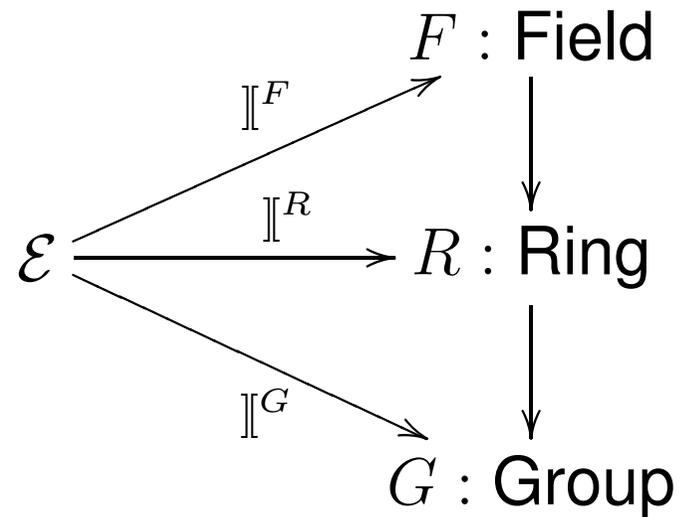
# *The big picture*



# The big picture

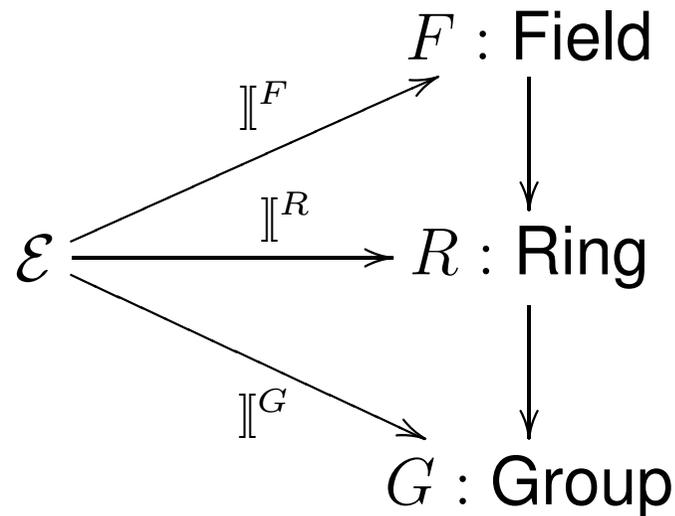


# The big picture



$e \times f \mathbb{I}^R x \times y$  but not  $e \times f \mathbb{I}^G x \times y$

# The big picture



$e \times f \mathbb{I}^R x \times y$  but not  $e \times f \mathbb{I}^G x \times y$

$e/f \mathbb{I}^F x/y$  but not  $e/f \mathbb{I}^G x/y$  or  $e/f \mathbb{I}^R x/y$

# *Good news, bad news*



# *Good news, bad news*

The good news...

# *Good news, bad news*

The good news...

- ⑥ it works!

# *Good news, bad news*

The good news...

- ⑥ it works!
- ⑥ reuse of around 60% of the code (the type  $\mathcal{E}$  and the normalization function  $\mathcal{N}$ )

# *Good news, bad news*

The good news...

- ⑥ it works!
- ⑥ reuse of around 60% of the code (the type  $\mathcal{E}$  and the normalization function  $\mathcal{N}$ )

... and the bad news

# *Good news, bad news*

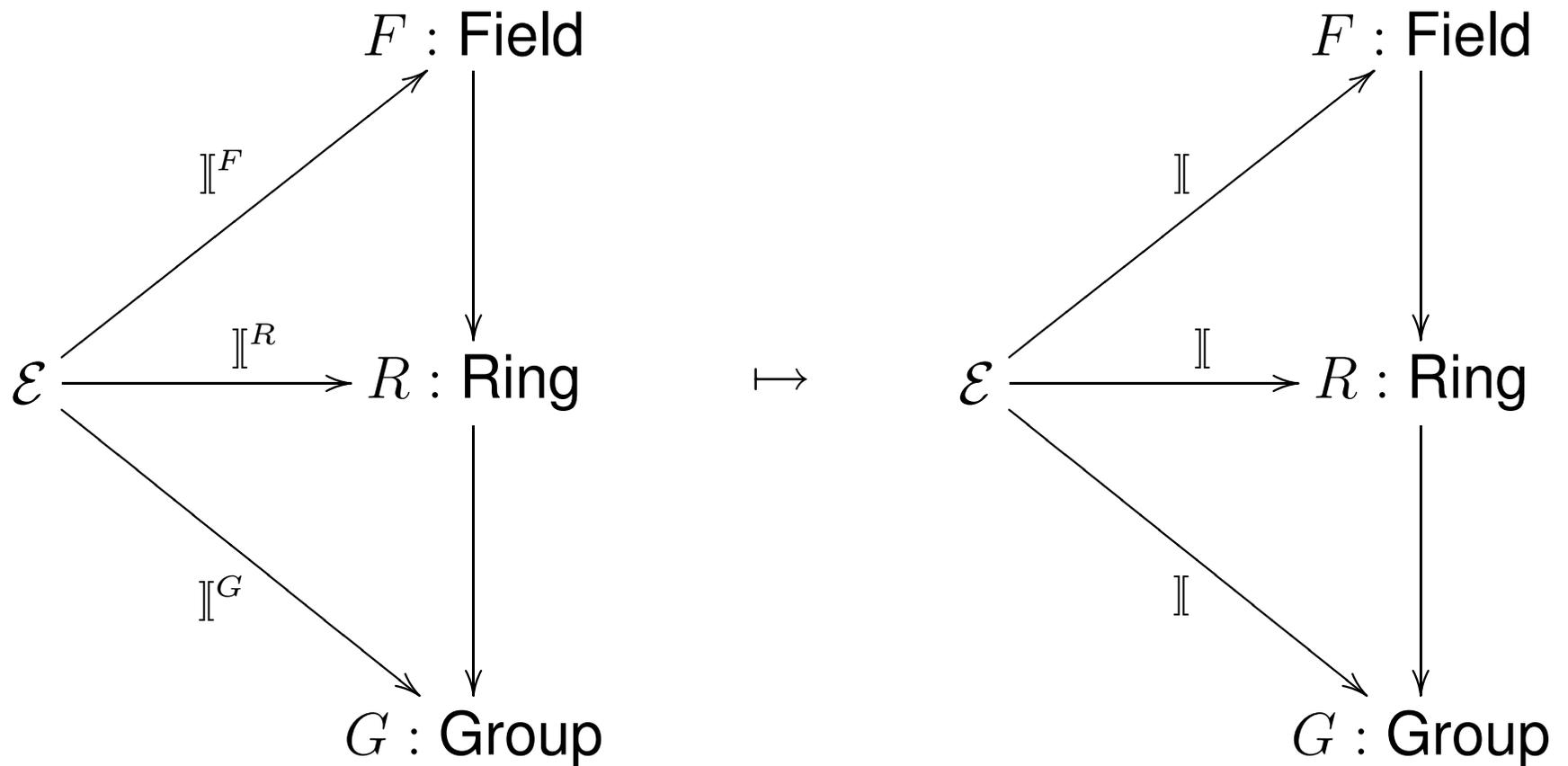
The good news...

- ⑥ it works!
- ⑥ reuse of around 60% of the code (the type  $\mathcal{E}$  and the normalization function  $\mathcal{N}$ )

... and the bad news

- ⑥ further unification requires extra axiom

# *In a perfect world (I)*



# *In a perfect world (II)*



## *In a perfect world (II)*

Instead of defining  $\mathbb{I}^G$ ,  $\mathbb{I}^R$  and  $\mathbb{I}^F$  by e.g.

## *In a perfect world (II)*

Instead of defining  $\llbracket^G$ ,  $\llbracket^R$  and  $\llbracket^F$  by e.g.

$$e \llbracket^G x \wedge f \llbracket^G y \Rightarrow e + f \llbracket^G x + y$$
$$e \llbracket^F x \wedge f \llbracket^F y \wedge y \neq 0 \Rightarrow e/f \llbracket^F x/y$$

## *In a perfect world (II)*

Instead of defining  $\llbracket^G$ ,  $\llbracket^R$  and  $\llbracket^F$  by e.g.

$$\begin{aligned} e \llbracket^G x \wedge f \llbracket^G y &\Rightarrow e + f \llbracket^G x + y \\ e \llbracket^F x \wedge f \llbracket^F y \wedge y \neq 0 &\Rightarrow e/f \llbracket^F x/y \end{aligned}$$

define  $\llbracket^- : \prod_{A:\text{Setoid}} \mathcal{E} \rightarrow A$  s.t.

## *In a perfect world (II)*

Instead of defining  $\llbracket^G$ ,  $\llbracket^R$  and  $\llbracket^F$  by e.g.

$$\begin{aligned} e \llbracket^G x \wedge f \llbracket^G y &\Rightarrow e + f \llbracket^G x + y \\ e \llbracket^F x \wedge f \llbracket^F y \wedge y \neq 0 &\Rightarrow e/f \llbracket^F x/y \end{aligned}$$

define  $\llbracket^- : \prod_{A:\text{Setoid}} \mathcal{E} \rightarrow A$  s.t.

$$\begin{aligned} A \text{ is group} \wedge e \llbracket^A x \wedge f \llbracket^A y &\Rightarrow e + f \llbracket^A x + y \\ A \text{ is field} \wedge e \llbracket^A x \wedge f \llbracket^A y \wedge y \neq 0 &\Rightarrow e/f \llbracket^A x/y \end{aligned}$$

using subtyping of algebraic structures.

# ***The $K$ -axiom***



# *The $K$ -axiom*

We cannot prove

# *The $K$ -axiom*

We cannot prove

$$e \ll^A a \wedge e \ll^A b \Rightarrow a =_A b$$

# *The $K$ -axiom*

We cannot prove

$$e \ll^A a \wedge e \ll^A b \Rightarrow a =_A b$$

without the  $K$ -axiom

# The $K$ -axiom

We cannot prove

$$e \Vdash^A a \wedge e \Vdash^A b \Rightarrow a =_A b$$

without the  $K$ -axiom

$$\langle x, y[x] \rangle = \langle x', y'[x'] \rangle \Rightarrow x = x' \wedge y = y'$$

# The $K$ -axiom

We cannot prove

$$e \Vdash^A a \wedge e \Vdash^A b \Rightarrow a =_A b$$

without the  $K$ -axiom

$$\langle x, y[x] \rangle = \langle x', y'[x'] \rangle \Rightarrow x = x' \wedge y = y'$$

The  $K$ -axiom, although consistent with, is not provable within Coq.

# ***Conclusions***



# *Conclusions*

- ⑥ Powerful tactics for equational reasoning

# Conclusions

- ⑥ Powerful tactics for equational reasoning
- ⑥ Reuse of code for fields, rings and groups

# Conclusions

- ⑥ Powerful tactics for equational reasoning
- ⑥ Reuse of code for fields, rings and groups
- ⑥ Improvement possible using  $K$ -axiom

# Conclusions

- ⑥ Powerful tactics for equational reasoning
- ⑥ Reuse of code for fields, rings and groups
- ⑥ Improvement possible using  $K$ -axiom
- ⑥ (and more on the paper)