Formalising and Reusing of Proofs

Mauricio Ayala-Rincón

Grupo de Teoria da Computação, Universidade de Brasília (UnB)

Brasília D.F., Brazil

Research funded by

Brazilian Research Agencies: CNPq, CAPES and FAPDF

LACREST MEDELLÍN 2012
REQUIREMENTS ENGINEERING & SOFTWARE TESTING
LATIN AMERICAN CONGRESS

July 14^{th}, 2012

Universidade de Brasília

Talk's Plan

Mathematical proofs - logic & deduction

Table: Rules of natural deduction for propositional logic

introduction rules	elimination rules
$\dfrac{\varphi \quad \psi}{\varphi \wedge \psi}\ (\wedge_i)$	$\dfrac{\varphi \wedge \psi}{\varphi}\ (\wedge_e)$
$\dfrac{\varphi}{\varphi \vee \psi}\ (\vee_i)$	$\dfrac{\varphi \vee \psi \quad \begin{matrix}[\varphi]^u \\ \vdots \\ \chi\end{matrix} \quad \begin{matrix}[\psi]^v \\ \vdots \\ \chi\end{matrix}}{\chi}\ (\vee_e), u, v$
$\dfrac{\begin{matrix}[\varphi]^u \\ \vdots \\ \psi\end{matrix}}{\varphi \to \psi}\ (\to_i), u$	$\dfrac{\varphi \quad \varphi \to \psi}{\psi}\ (\to_e)$
	$\dfrac{\begin{matrix}[\neg\varphi]^u \\ \vdots \\ \bot\end{matrix}}{\varphi}\ (\bot_e), u$

Universidade de Brasília

Mathematical proofs - logic & deduction

Table: Rules of Natural Deduction for Predicate logic with equality

introduction	elimination
$\dfrac{}{t = t}\ (=_i)$	$\dfrac{t_1 = t_2 \quad \varphi[x/t_1]}{\varphi[x/t_2]}\ (=_e)$
$\dfrac{\boxed{\begin{array}{c} y \text{ indep.} \\ \vdots \\ \varphi[x/y] \end{array}}}{\forall x\ \varphi}\ (\forall_i)$	$\dfrac{\forall x\ \varphi}{\varphi[x/t]}\ (\forall_e)$
$\dfrac{\varphi[x/t]}{\exists x\ \varphi}\ (\exists_i)$	$\dfrac{\exists x\ \varphi \quad \boxed{\begin{array}{c} [\varphi[x/y]]^u \\ y \text{ indep.} \vdots \\ \chi \end{array}}}{\chi}\ (\exists_e), u$

Universidade de Brasília

Mathematical proofs - logic & deduction

Table: Encoding ¬ - Rules of natural deduction for classical logic

introduction rules	elimination rules
$\dfrac{\begin{array}{c}[\varphi]^u\\ \vdots \\ \bot\end{array}}{\neg\varphi}\ (\neg_i),\ u$	$\dfrac{\varphi \quad \neg\varphi}{\bot}\ (\neg_e)$

$$\dfrac{\begin{array}{c}[\varphi]^u\\ \vdots \\ \bot\end{array}}{\varphi \to \bot}\ (\to_i),\ u \qquad\qquad \dfrac{\varphi \quad \varphi \to \bot}{\bot}\ (\to_e)$$

Universidade de Brasília

Mathematical proofs - logic & deduction

Interchangeable rules:

$$\frac{\neg\neg\phi}{\phi}\ (\neg\neg_e) \qquad \frac{}{\phi \vee \neg\phi}\ (\text{LEM}) \qquad \frac{\begin{array}{c}[\neg\phi]^a\\ \vdots\\ \bot\end{array}}{\phi}\ (\neg_e),\ a$$

Mathematical proofs - logic & deduction

Examples of deductions. Assuming $(\neg\neg_e)$, LEM holds:

$$
\begin{prooftree}
\AxiomC{$[\neg(\phi \vee \neg\phi)]^x$}
\AxiomC{$[\neg(\phi \vee \neg\phi)]^x$}
\AxiomC{$[\phi]^u$}
\RightLabel{$(\vee_i)$}
\UnaryInfC{$\phi \vee \neg\phi$}
\RightLabel{$(\neg_e)$}
\BinaryInfC{$\bot$}
\RightLabel{$(\neg_i), u$}
\UnaryInfC{$\neg\phi$}
\RightLabel{$(\vee_i)$}
\UnaryInfC{$\phi \vee \neg\phi$}
\RightLabel{$(\neg_e)$}
\BinaryInfC{$\bot$}
\RightLabel{$(\neg_i), x$}
\UnaryInfC{$\neg\neg(\phi \vee \neg\phi)$}
\RightLabel{$(\neg\neg_e)$}
\UnaryInfC{$\phi \vee \neg\phi$}
\end{prooftree}
$$

Universidade de Brasília

Mathematical proofs - logic & deduction

A derivation of Peirce's law, $((\phi \to \psi) \to \phi) \to \phi$:

$$
\dfrac{\dfrac{\dfrac{[\neg\phi]^u \quad \dfrac{[((\phi \to \psi) \to \phi)]^x \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\neg\phi]^u}{\neg\psi \to \neg\phi}\ \to_i, \emptyset \quad [\neg\psi]^v}{\neg\phi}\ (\to_e) \quad [\phi]^w}{\bot}\ (\neg_e)}{\psi}\ (\mathrm{PBC}), v}{\phi \to \psi}\ (\to_i), w}{\phi}\ (\to_e)}{\bot}\ (\neg_e)}{\phi}\ (\mathrm{PBC}), u}{((\phi \to \psi) \to \phi) \to \phi}\ (\to_i), x
$$

A very little list of related work

- Reusing proofs (T.Kolbe & C.Walter, 1994): fixing successful proof strategies through learning methods;
- Reuse of proofs in software verification (Wolfgang Reif & Kurt Stenzel, 1993): reusing proofs and proof attempts after software modifications;
- Similarities and Reuse of Proofs in Formal Software Verification (Erica Melis & Axel Schairer, 1998): reusing subproofs;
- How mathematicians prove theorems?

Learning from how mathematicians prove theorems

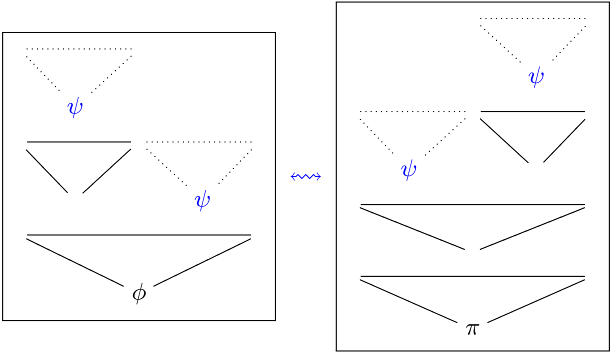


Figure: Inference of Lemmas

Learning from how mathematicians prove theorems

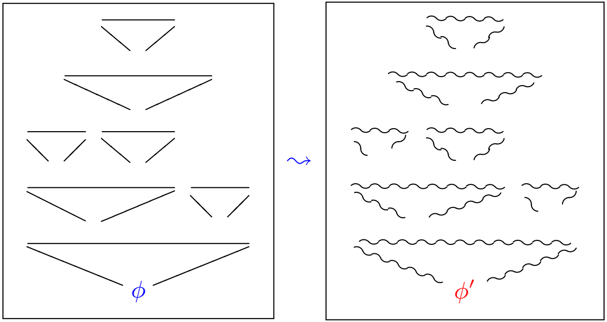


Figure: Analogy

Learning from how mathematicians prove theorems

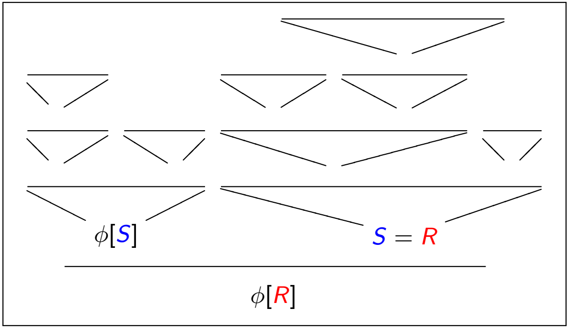


Figure: Equational reasoning

The Prototype Verification System - PVS

PVS is a verification system, developed by the SRI International Computer Science Laboratory, which consists of

1. a specification language:
   - based on higher-order logic;
   - a type system based on Church's simple theory of types augmented with subtypes and dependent types.
2. an interactive theorem prover:
   - based on sequent calculus; that is, goals in PVS are sequents of the form $\Gamma \vdash \Delta$, where Γ and Δ are finite sequences of formulae, with the usual Gentzen semantics.

GTC/Universidade de Brasília & PVS

- Term Rewriting Systems PVS library trs AR & Galdino UnB
- First-Order Unification PVS library unification AR & Avelar UnB
- Group theory PVS library groups Galdino UFG

All them available in the NASA LaRC PVS libraries:
http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html

- Air traffic CD&R (KB2D $\rightsquigarrow$ ACCoRD) AR & Galdino, Muñoz (NIA/NASA LaRC)
- Automating termination AR & Goodloe & Muñoz (NASA LaRC)
- Cryptography AR & Regô, Nantes & Fernández (King's College London)

Universidade de Brasília

Formal methods in cryptography

- Why proving mathematically security requirements?
- Authentication protocol of Needham-Schroeder
  - was considered during 17 years to be secure.
  - but Lowe detected a "man-in-the-middle" vulnerability in this protocol [Lowe 95,6].
- Example: formalisation of the security of the Dolev-Yao two-party cascade protocol [Dolev-Yao 83].
  - Joint work with Rodrigo Nogueira [2010] and Yuri Santos Rêgo [2012].

Universidade de Brasília

Cryptographic operations over monoids

- Any user $u \in U$ owns E_u and D_u.
  - $E = \{E_u \mid u \in U\}$
  - $D = \{D_u \mid u \in U\}$
- $\Sigma = E \cup D$
- Σ^* set of words over Σ.
- Monoid freely generated by Σ and congruences:

$$E_u D_u = \lambda \qquad D_u E_u = \lambda, \quad \forall u \in U \tag{1}$$

- $E_u(D_u(M)) = D_u(E_u(M)) = M, \forall M$ plain text.

Formalisation of security for cascade protocols

Theorem (Characterisation of security)

A cascade protocol P is secure iff,

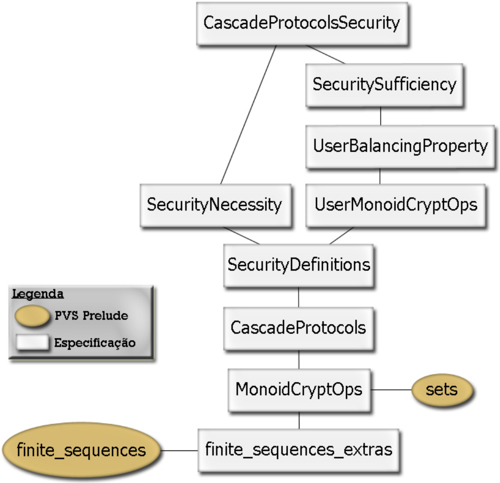
- (*i*) *it satisfies the initial security property and*
- (*ii*) *it is balanced.*

Formalisation in PVS

```
theorem1 :  THEOREM FORALL (prot :  welldefined_protocol,
                           x :  U, y :  U | x /= y, z :  U | z /= x AND z /= y) :
   secure_protocol?(prot, x, y, z) IFF
   ( alpha0ContainsE?(prot, x, y) AND balanced_cascade_protocol?(prot) )
```

Universidade de Brasília

Structure of the PVS formalisation



Reusing proofs

Why?

- Formalising is an exhaustive process that takes years.
  - Our case study on the DY security takes more than two years!
  - Size of the specification: 1.651 lines (80 KB), but
  - Size of the Formalisation: 55.300 lines (3.8 MB)!
- Small changes in the specification, implies rebuilding proofs from scratch.
- As well, use of alternative data structures, implies rebuilding proofs from scratch.

Reusing proofs - changing data structures

- Instead sequences, use lists

Reusing proofs

Definition (Isomrphism between poly-sorted signatures)

Let $\langle \mathcal{A}, \mathcal{F}, \mathcal{R} \rangle$ and $\langle \mathcal{B}, \mathcal{G}, \mathcal{P} \rangle$ be signatures consisting of families of sets $\mathcal{A} = \{A_1, \ldots, A_n\}$ and $\mathcal{B} = \{B_1, \ldots, B_n\}$, functions $\mathcal{F} = \{f_1, \ldots, f_k\}$ and $\mathcal{G} = \{g_1, \ldots, g_k\}$ and relations $\mathcal{R} = \{r_1, \ldots, r_l\}$ and $\mathcal{P} = \{p_1, \ldots, p_l\}$. An isomorphism between these structures, $\imath$ is a bijective mapping from the families of sets, and from functions into functions and relations into relations, such that the following preservation properties hold:

- *For all $f \in \mathcal{F}$, and m-tuple of well-typed arguments for f, $x_1, \ldots, x_m$, supposing f is an m-ary function, $\imath(f(x_1, \ldots, x_m)) = f^{\imath}(\imath(x_1), \ldots, \imath(x_m))$;*
- *For all $p \in \mathcal{P}$, and m-tuple of well-typed arguments for p, $x_1, \ldots, x_m$, supposing p is an m-ary predicate, $\imath(p(x_1, \ldots, x_m))$ if and only if $\imath^h(\imath(x_1), \ldots, \imath(x_m))$.*

Universidade de Brasília

Reusing proofs

$$\langle \mathcal{A}, \mathcal{F}, \mathcal{R} \rangle \xrightarrow{\quad \imath \quad} \langle \mathcal{B}, \mathcal{G}, \mathcal{P} \rangle$$

$$A \in \mathcal{A} \xrightarrow{\quad x \;\mapsto\; \imath_A(x) \quad} B \in \mathcal{B}$$

$$f \in \mathcal{F} \xrightarrow{\quad f \;\mapsto\; f^{\imath} \quad} f^{\imath} \in \mathcal{G}$$

$$\imath(f(x_1, \ldots, x_m)) = f^{\imath}(\imath(x_1), \ldots, \imath(x_m))$$

$$p \in \mathcal{R} \xrightarrow{\quad p \;\mapsto\; p^{\imath} \quad} p^{\imath} \in \mathcal{P}$$

$$\imath(p(x_1, \ldots, x_m)) \Leftrightarrow p^{\imath}(\imath(x_1), \ldots, \imath(x_m))$$

Figure: Isomorphism between poly-sorted signatures

Reusing proofs — Examples

- $$\langle \mathbb{R}, +, 0, > \rangle \xrightarrow{\imath} \langle \mathbb{R}^+, \times, 1, > \rangle$$

  $$\mathbb{R} \xrightarrow{x \;\mapsto\; \imath(x) := \exp(x)} \mathbb{R}^+$$

  $$+ \xrightarrow{+ \;\mapsto\; +^{\imath} := \times} \times$$

  $$0 \xrightarrow{0 \;\mapsto\; 0^{\imath} := 1} 1$$

  $$> \xrightarrow{> \;\mapsto\; >^{\imath} := >} >$$

Universidade de Brasília

Reusing proofs — Examples

$\imath$ is the function *ln*. Thus, one has two useful lemmas:

Lemma (isomorphism 1) $\imath \circ \imath$ is the identity in $\mathbb{R}$
Lemma (isomorphism 2) $\imath \circ \imath$ is the identity in $\mathbb{R}^+$

Homeomorphic properties for the isomorphism and its inverse:

Lemma (preservation of $+$) $\forall x, y : \mathbb{R}.\ \imath(x + y) = \imath(x) +^{\imath} \imath(y)$
Lemma (preservation of $>$1) $\forall x, y : \mathbb{R}.\ x > y \Leftrightarrow \imath(x) >^{\imath} \imath(y)$

Lemma (preservation of $\times$) $\forall x, y : \mathbb{R}^+.\ \imath(x \times y) = \imath(x) \times^{\imath} \imath(y)$
Lemma (preservation of $>$2) $\forall x, y : \mathbb{R}+ .\ x > y \Leftrightarrow \imath(x) >^{\imath} \imath(y)$

Reusing proofs — Examples

Theorem (additive inverse) $\forall x : \mathbb{R}.\ x + (-x) = 0$

Theorem (ln of mult. inverses) $\forall x : \mathbb{R}^{+}.\ \ln(x^{-1}) = -\ln(x)$

Reusing proofs — Examples

Theorem (multiplicative inverse) $\forall x : \mathbb{R}^+.\ x \times x^{-1} = 1$

can be proved as follows:

1. $x \times x^{-1} = \exp \circ \ln(x \times x^{-1})$, by Lemma isomorphism 2;
2. $\exp \circ \ln(x \times x^{-1}) = \exp(\ln(x) + \ln(x^{-1}))$, by preservation of $\times$;
3. $\exp(\ln(x) + \ln(x^{-1})) = \exp(\ln(x) + -\ln(x))$, by Theorem of ln of mult. inverses;
4. $\exp(\ln(x) + -\ln(x)) = \exp(0)$, by Theorem of additive inverse;
5. $\exp(0) = 1$, by application of the isomorphism exp.

Reusing proofs — Case of study

Changing sequences for lists in the formalisation of security of cryptographic protocols, implies construction of several operators:

$$(\mathtt{length} : \mathtt{nat}, \mathtt{seq} : [0, \mathtt{length} - 1] -> \mathtt{CryOp})$$

$\downarrow \imath$ $\qquad\qquad$ $\uparrow \imath$

$\mathtt{list}[\mathtt{CryOp}]$ $\qquad\qquad$ but also,

Reusing proofs — Case of study

For illustration, consider reusing the proof of

> Theorem(length of empty sequences)
> `s‘length = 0 IFF s = empty_seq`

to prove that the following analogous result over lists.

> Theorem(length of null list)
> `length(l) = 0 IFF l = null`

Reusing proofs — Case of study

$$\langle\{\texttt{CryOp}, \texttt{seq[CryOp]}, \mathbb{N}, \mathbb{N}, \ldots\}, \{\texttt{`length}, \texttt{`seq}, \ldots\}, \{=_{\texttt{seq[CryOp]}}, \ldots\}\rangle$$

$$\downarrow \imath$$

$$\langle\{\texttt{CryOp}, \texttt{list[CryOp]}, \mathbb{N}, \mathbb{N}^+, \ldots\}, \{\texttt{length}, \imath(\_\texttt{`seq})\ldots\}, \{=_{\texttt{list[CryOp]}}, \ldots\}\rangle$$

$\texttt{CryOpt} \longrightarrow \texttt{CryOpt}$: $op \mapsto op$

$\texttt{seq[CryOpt]} \longrightarrow \texttt{list[CryOpt]}$: $s \mapsto \imath(s)$

$\mathbb{N} \longrightarrow \mathbb{N}$: $n \mapsto n$

$\mathbb{N}(\text{index}) \longrightarrow \mathbb{N}^+(\text{position})$: $n \mapsto n+1$

$\_\texttt{`length} \longrightarrow \texttt{length}(\_)$ : $s\text{`}length \mapsto length(\imath(s))$

$\_\texttt{`seq} \longrightarrow \imath(\_\texttt{`seq})$: $s\text{`}seq \mapsto \lambda_{(\texttt{i:[1,length]})}.\texttt{nth(i,}\imath\texttt{(s))}$

$\vdots \qquad\qquad \vdots$

Figure: Isomorphism between sequences and lists of CryOps

Universidade de Brasília

Reusing proofs — Case of study

$$(\mathtt{length} : \mathtt{nat}, \mathtt{seq} : [0, \mathtt{length} - 1] -> \mathtt{CryOp})$$

$$\downarrow \imath$$

$$\mathtt{list}[\mathtt{CryOp}]$$

Specification transformation from Sequences to lists:

```
ı(s :  seq[CryOp]) RECURSIVE : list[CryOp] =
IF s‘length = 0 THEN null
ELSE cons(s‘seq(0), ı(s(1, s‘length - 1))
ENDIF
MEASURE seq‘length
```

Universidade de Brasília

Reusing proofs — Case of study

Homeomorphic properties should be formalized as, for instance:

Lemma A1	$\imath(\mathtt{s`length}) = \mathtt{length}(\imath(\mathtt{s}))$
Lemma A2	$\imath(\mathtt{s`seq}) = \lambda_{(\mathtt{i}:[1,\mathtt{s`length}])}.\mathtt{nth}(\mathtt{i}, \imath(\mathtt{s}))$
Lemma A3	$\imath(\mathtt{s`seq}(\mathtt{k})) = (\lambda_{(\mathtt{i}:[1,\mathtt{s`length}])}.\mathtt{nth}(\mathtt{i}, \imath(\mathtt{s})))\imath(\mathtt{k})$

Observe, that one has:
$(\lambda_{(\mathtt{i}:[1,\mathtt{s`length}])}.\mathtt{nth}(\mathtt{i}, \imath(\mathtt{s})))\imath(\mathtt{k}) \rightarrow_\beta$
$(\lambda_{(\mathtt{i}:[1,\mathtt{s`length}])}.\mathtt{nth}(\mathtt{i}, \imath(\mathtt{s})))(\mathtt{k}+1) \rightarrow_\beta \mathtt{nth}(\mathtt{k}+1, \imath(\mathtt{s}))$,
thus, by lemma A3, $\imath(\mathtt{s`seq}(\mathtt{k})) = \mathtt{nth}(\mathtt{k}+1, \imath(\mathtt{s}))$.

Universidade de Brasília

Reusing proofs — Case of study

$$(\texttt{length} : \texttt{nat}, \texttt{seq} : [0, \texttt{length} - 1] -> \texttt{CryOp})$$

$$\uparrow \imath$$

$$\texttt{list[CryOp]}$$

Specification transformation from lists to Sequences:

$\imath$`(l : list[CryOp]) : seq[CryOp] =`
`(# length = length(l),`
`seq =` $\lambda_{(\texttt{i}:[0,\texttt{length(l)}-1])}$`.nth(i+1, l) #)`

Reusing proofs — Case of study

Also, homeomorphic properties should be formalized, as for instance:

Lemma B1	$\imath(\mathtt{length}(\mathtt{l})) = (\imath(\mathtt{l}))`\mathtt{length}$
Lemma B2	$\imath(\mathtt{nth}(\mathtt{k},\mathtt{l})) = (\imath(\mathtt{l}))`\mathtt{seq}(\imath(\mathtt{k}))$

Notice that

$\lambda_{(\mathtt{i}:[0,\mathtt{length}(\mathtt{l})-1])}.\mathtt{nth}(\mathtt{i}+1,\mathtt{l}))(\imath(\mathtt{k})) =$
$\lambda_{(\mathtt{i}:[0,\mathtt{length}(\mathtt{l})-1])}.\mathtt{nth}(\mathtt{i}+1,\mathtt{l}))(\mathtt{k}-1) \to_\beta \mathtt{nth}(\mathtt{k},\mathtt{l})$.

Universidade de Brasília

Reusing proofs — Case of study

Formalisation of isomorphic properties is necessary:

Lemma isomorphism 1	$\forall \mathtt{s} : \mathtt{seq[CryOp]}.\ \imath \circ \imath(\mathtt{s}) = \mathtt{s}$
Lemma isomorphism 2	$\forall \mathtt{l} : \mathtt{list[CryOp]}.\ \imath \circ \imath(\mathtt{l}) = \mathtt{l}$

The presented properties are not exhaustive!

Reusing proofs — Case of study

Reusing `Theorem s‘length = 0 IFF s = empty_seq` to prove `Theorem length(l) = 0 IFF l = null`:

$\texttt{length(l)} = 0 \Leftrightarrow$	appl. of isomorphism operator
$\imath(\texttt{length(l)} = 0) \Leftrightarrow$	isomorphism properties
$\imath(\texttt{length(l)}) = \imath(0) \Leftrightarrow$	isomorphism properties
$\imath(\texttt{length(l)}) = 0 \Leftrightarrow$	isomorphism properties
$\imath(\texttt{l})\text{‘}\texttt{length} = 0$ IFF	reuse of Theorem
$\imath(\texttt{l}) = \texttt{empty\_seq} \Leftrightarrow$	application of isomorphism
$\imath(\imath(\texttt{l}) = \texttt{empty\_seq}) \Leftrightarrow$	isomorphism properties
$\imath(\imath(\texttt{l})) = \imath(\texttt{empty\_seq}) \Leftrightarrow$	isomorphism properties
$\texttt{l} = \imath(\texttt{empty\_seq}) \Leftrightarrow$	isomorphism properties
$\texttt{l} = \texttt{null}$	□

Universidade de Brasília

Reusing proofs — Case of study

Summarizing, the approach to reuse formalizations through isomorphic transformations involves two main steps:

1. Construction and formalization of isomorphisms:
    1. Construction of isomorphic transformations between data structures, functions and relations;
    2. Formalization of isomorphic and homeomorphic properties;
2. Reuse of proofs.

Once the first step is completed, proofs by reusing formalizations of equational and relational theorems follow the sketches in Fig. 6 and 7, respectively.

Reusing proofs — Case of study

$$\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$$

$$\imath \Updownarrow \imath \qquad \texttt{isomorphisms}$$

$$\mathtt{f}^{\imath}(\imath(\mathtt{x}_1), \ldots, \imath(\mathtt{x}_n))$$

$$= \qquad \texttt{Theorem}$$

$$\mathtt{g}(\mathtt{y}_1, \ldots, \mathtt{y}_m)$$

$$\imath \Updownarrow \imath \qquad \texttt{isomorphisms}$$

$$\mathtt{g}^{\imath}(\imath(\mathtt{y}_1), \ldots, \imath(\mathtt{y}_m))$$

$$\texttt{Theorem}\ \mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) = \mathtt{g}^{\imath}(\imath(\mathtt{y}_1), \ldots, \imath(\mathtt{y}_m))$$

Figure: General sketch of reusing equational proofs by isomorphisms

Universidade de Brasília

Reusing proofs — Case of study

$$p(x_1, \ldots, x_n)$$

$$\imath \Updownarrow \imath \qquad \textit{isomorphisms}$$

$$\frac{p^{\imath}(\imath(x_1), \ldots, \imath(x_n)) \qquad \texttt{Theorem}}{\texttt{Theorem}\ p(x_1, \ldots, x_n)}$$

Figure: General sketch for reusing relational proofs by isomorphisms

Conclusions

- Reusing proofs is not straightforward.
- Building poly-sorted isomorphisms works well, but is an exhaustive task.
- Although this, after specifying isomorphism operators and having proved all mundane isomorphic properties complex proofs can be reused.

Future Work

- As a case study the formalisation of security of the Dolev-Yao model is being translated to other data structures.
  - More abstract approaches are possible: starting from mathematical properties proved over algebraic structures trying to work independently of any data structure.
  - The size of the formalisation should be big enough in order to have a relatively small part related with isomorphisms. For example, the formalisation on D-Y security has size ca 80 KB and 3.8 MB specification and formalisation, respectively.
- Several related academic projects involving generation of PVS livraries are to be supervised in the GTC at the UnB.

Universidade de Brasília

References

D. Dolev and A. C. Yao.
On the Security of Public Key Protocols.
IEEE Transactions on Information Theory, 29(2):198–208, 1983.

G. Lowe.
An Attack on the Needham-Schroeder Public-Key Authentication Protocol.
Information Processing Letters, 56(3):131–133, 1995.

G. Lowe.
Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR.
Software - Concepts and Tools, 17(3):93–102, 1996.

R.B. Nogueira, M. Ayala-Rincón, A. Nascimento, and F. L.C. de Moura.
Formalization of security proofs using PVS in the Dolev-Yao model.
In Computability in Europe, 2010.

Y.S. Rêgo and M. Ayala-Rincón.
Formalization in PVS of Balancing Properties Necessary for the Security of the Dolev-Yao Cascade Protocol Model
SBL 2011 full version available, 2012.

Universidade de Brasília