A modularization mechanism for interactive proof development

Eugenio G. Omodeo
Università di L'Aquila
Dipartimento di Informatica

Jacob T. Schwartz
New York University
Courant Inst. of Mathematical Sciences

Atlanta,
March 9, 2002,
AMS meeting

Program derivation with verified transformations – I

[· · ·] *a transformational programming methodology that includes a fully operational set-theoretic proof checker* [· · ·]

[· · ·] *examples of two moderately difficult program derivations. One of these derives a high level form of an algorithm to compute the bisimulation equivalence relation* [· · ·] *the other derives an algorithm to minimize the number of states in a deterministic finite state automaton.*

[· · ·]*Based on our experience with these derivations, we believe that mechanical verification of program transformations is the most important missing ingredient to the successful use of transformational programming as part of a viable program development technology.*

J.-P. Keller, R. Paige, 1995

Program derivation with verified transformations – II

But approximately 7,000 lines of NAP text were used $[\cdots]$ *to work out the whole proof. We believe that this amount of "proofware" represents about ten times the size of a legible mathematical proof. We would hope that a high level language approach to proof construction and reuse would help.*

J.-P. Keller, R. Paige, 1995

WHAT ARCHITECTURE FOR PROOFWARE ?

- How can we 'tame' the problems which arise from large scale ?
- Is the distinction between calculus and theory adequate ?
- Any answer must ensue from an extensive experimentation plan (and activity)
- It is convenient to move from established research on the foundations of mathematics and analysis
- (Hyper-)textual scenarios are more persistent than fully interactive sessions

EXPERIMENTATION

Two current activities: NYU , MIUR 40%

- one follows the royal road of mathematics paved by the work of Cauchy, Dedekind, Frege, Cantor, Peano, Whitehead-Russell, Zermelo-Fraenkel-von Neumann, etc.
- the other aims at assessing the current state of 'proof technology', insisting particularly on the equational calculus of dyadic relations

Common denominator:

Emphasis on **set theory**

Design of
definitional extension mechanisms

ROLE OF SET THEORY

Research on foundations

- Brought into evidence the usability of set theory as a *lingua franca* for the whole of maths
- Led to an extremely valuable development of *abstract schemes* of reasoning
- *Did not* impress a technological push proportionate to the research achievements

2 examples:

$$\mathsf{Ord}(X) =: X \subseteq \mathcal{P}(X) \;\&\; (\forall\, y, z \in X)\; (\, y \in z \;\vee\; y = z \;\vee\; z \in y\,)$$

$$\mathbb{R} \quad =: \{c \subseteq \mathbb{Q} \mid (\forall\, y \in c)(\exists\, z \in c)(y < z) \;\&\; (\forall\, y \in c)(\forall\, z \in \mathbb{Q})(z < y \;\rightarrow\; z \in c)\} \setminus \{\emptyset, \mathbb{Q}\}$$

- Part of the appeal of set theory stems from the existence of a variant of it which deals with *finite* sets only (Alfred Tarski, 1924)
- It well bridges problem specification languages with *program specification* languages (where set-handling capabilities are the 'bread-and-butter')

Ingredients of Set Theory

- *Boolean* operations (save absolute complementation) $\emptyset, \cap, \cup, \setminus, \triangle$
- *extensional* $\in$ (two sets cannot have the same el'ts)

$$X = Y \leftrightarrow \forall\, v\,(\, v \in X \leftrightarrow v \in Y\,)$$

- *nesting* (which makes even individuals superfluous) X with Y, $\{\, X_1, \ldots, X_n \,\}$
- Possibility to form sets complying with *intensional* specifications

$$\begin{array}{lcl} \bigcup Y & =: & \{\, x_2 : x_1 \in Y, x_2 \in x_1 \} \\ \mathcal{P}(Y) & =: & \{\, x \,:\, x \subseteq Y \,\} \\ \mathsf{own}\mathcal{P}(Y) & =: & \{\, x \,:\, x \subseteq Y \mid x \in Y \,\} \end{array}$$

- *Well-foundedness* and *choice*:
  arb $X \in X$ with $X \;\&\; X \cap$ **arb** $X = \emptyset$
- *Recursion* based on $\in$

$$\mathsf{Ult\_mm}(S) \;=:\; S \cup \bigcup \{\, \mathsf{Ult\_mm}(x) : x \in S \,\}$$

- Existence of *infinite* sets

$$\mathbf{s}_\infty \neq \emptyset \;\&\; (\forall\, x \in \mathbf{s}_\infty)(\, \{x\} \in \mathbf{s}_\infty)$$

A citation on 'proof-hiding' – I

Wir haben oft ein Zeichen nötig, mit dem wir einen sehr zusammengesetzten Sinn verbinden. Dieses Zeichen dient uns sozusagen als Gefäß, in dem wir diesen Sinn mit uns führen können, immer in dem Bewußtsein, daß wir dieses Gefäß öffnen können, wenn wir seines Inhalts bedürfen. Gottlob Frege

We often need to associate some highly compound meaning with a symbol. Such a symbol serves us as a kind of container in which we can carry this meaning, always with the understanding that it can be opened if we need its content.

An example of 'proof-hiding' – II

The reader who remembers these key points will do well in what follows. In particular, it is now quite all right to entirely forget how the nonstandard universe was defined and to banish ultrafilters from our consciousness.

Martin Davis, Applied Nonstandard Analysis, 1977

'THEORY' EXAMPLE: ordered pair

THEORY orderedPair( )

==> (cons, car, cdr, nl, len)

$$\mathsf{car}\big(\mathsf{cons}(X,Y)\big) = X$$

$$\mathsf{cdr}\big(\mathsf{cons}(X,Y)\big) = Y$$

$$\mathsf{cons}(X,Y) = \mathsf{cons}(U,V) \rightarrow X = U \;\&\; Y = V$$

$$\mathsf{nl} \neq \mathsf{cons}(X,Y)$$

$$\mathsf{len}(\mathsf{nl}) = 0$$

$$\mathsf{len}(\mathsf{cons}(X,Y)) = \mathsf{next}(\mathsf{len}(Y))$$

END orderedPair

'Within' orderedPair we would perhaps see

$$\begin{array}{lcl}
\mathsf{cons}(X,Y) & =: & \Big\{\{X\},\big\{\{X\},\{Y,\{Y\}\}\big\}\Big\} \\
\mathsf{car}(P) & =: & \mathbf{arb}\ \mathbf{arb}\ P \\
\mathsf{cdr}(P) & =: & \mathsf{car}\big(\mathbf{arb}\ (P \setminus \{\mathbf{arb}\ P\}) \\
 & & \qquad\qquad \setminus \{\mathbf{arb}\ P\}\big) \\
\mathsf{nl} & =: & \emptyset \\
\vdots & \vdots & \vdots \\
\mathsf{len}(T) & =: & \mathbf{arb}\ \big\{\mathsf{next}(\mathsf{len}(r)) \\
 & & \qquad : r \in y \in x \in T \\
 & & \qquad \mid (\exists l)(T = [l,r])\big\}
\end{array}$$

intermixed with various proof details

ordered pair example continued

Theories will form a *hierarchy*:
'from above' orderedPair, we would see

$$\begin{array}{rcl} 0 & =: & \emptyset \\ \mathsf{next}(X) & =: & X \cup \{ X \} \\ \vdots & \vdots & \vdots \end{array}$$

(or something equivalent) before the 'invocation'

$$\mathbf{APPLY} \ ([-, -], \ hd, \ tl, \ [\,], \ ln \ ) \ \mathsf{orderedPair}$$

Defining $\mathsf{len}(-)$ inside orderedPair is certainly easier than outside it

CONSERVATIVE EXTENSIONS OF A LOGICAL FORMALISM

So far, 'THEORY' supplies us an extension mechanism more flexible, but of the same nature, as

- definitions such as $P \cap Q \;=:\; \overline{\overline{P} \cup \overline{Q}}$ for Robbins' algebra
- mechanisms such as *Skolemization*, which enables one to replace e.g. an axiom

  $\exists\, u\, (\, \forall\, x \;\; x \cdot u \;=\; x \;\;\&\;\; \forall\, y\, \exists\, v \;\; y \cdot v \;=\; u\,)$

  by the more legible

$$X \cdot 1 = X$$
$$Y \cdot Y^{-1} = 1$$

. . . . . However, *brevity*, *legibility*, *conservativity*, are not enough . . . . . .

IMPORTANCE OF DEF. MECHANISMS

Definitions serve various purposes. At their simplest they are merely abbreviations which concentrate attention on interesting constructs by assigning them names which shorten their syntactic form. (But of course the compounding of such abbreviations can change the appearance of a discourse completely, transforming what would otherwise be a exponentially lengthening welter of bewildering formulae into a sequence of sentences which carry helpful intuitions). Beyond this, definitions serve to 'instantiate', that is, to introduce the objects whose special properties are crucial to an intended argument. Like the selection of crucial lines, points, and circles from the infinity of geometric elements that might be considered in a Euclidean argument, definitions of this kind often carry a proof's most vital ideas.

Jacob T. Schwartz

REUSE OF 'GOOD TRICKS' – I
(constructions, proofs)

THEORY equiv_classes $(\mathsf{s}, \mathsf{Eq})$

$$X \in \mathsf{s} \rightarrow \mathsf{Eq}(X, X)$$
$$X, Y, Z \in \mathsf{s} \,\&\, \mathsf{Eq}(X, Y) \rightarrow \Big(\mathsf{Eq}(Y, Z) \leftrightarrow \mathsf{Eq}(Z, X)\Big)$$

==> $(\mathsf{quot}, \mathsf{cl\_of})$

$$X, Y \in \mathsf{s} \rightarrow \Big(\mathsf{Eq}(X, Y) \leftrightarrow \mathsf{Eq}(Y, X)\Big)$$
$$X \in \mathsf{s} \rightarrow \mathsf{cl\_of}(X) \in \mathsf{quot}$$
$$B \in \mathsf{quot} \rightarrow \mathbf{arb}\, B \in \mathsf{s} \,\&\, \mathsf{cl\_of}(\mathbf{arb}\, B) = B$$
$$Y \in \mathsf{s} \rightarrow \Big(\mathsf{Eq}(X, Y) \leftrightarrow \mathsf{cl\_of}(X) = \mathsf{cl\_of}(Y)\Big)$$

END equiv_classes

Note: the facts

$$\emptyset \notin \mathsf{quot}$$
$$\mathsf{s} = \bigcup \mathsf{quot}$$
$$X \in B \in \mathsf{quot} \rightarrow \mathsf{cl\_of}(X) = B$$

are not 'exported'

REUSE OF 'GOOD TRICKS' – II
(constructions, proofs)

THEORY recursive_fcn $(\mathsf{dom}, \mathsf{Lt}, \mathsf{a}, \mathsf{b}, \mathsf{P})$

$$(\forall t \subseteq \mathsf{dom})\Big(t \neq \emptyset \rightarrow (\exists\, m \in t)(\forall u \in t)\neg \mathsf{Lt}(u, m)\Big)$$

-- Lt *is thereby assumed to be*

-- *irreflexive and well-founded on* dom

$$\Longrightarrow (\mathsf{rec})$$

$$(\forall\, x, y \in \mathsf{s})\Big(\big(\mathsf{Lt}(x, y) \rightarrow \neg \mathsf{Lt}(y, x)\big) \;\&\; \neg \mathsf{Lt}(x, x)\Big)$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$(\forall\, v \in \mathsf{dom})\Big(\mathsf{rec}(v) = \mathsf{a}\Big(v, \big\{\, \mathsf{b}\big(v, w, \mathsf{rec}(w)\big) : w \in \mathsf{dom} \mid \mathsf{Lt}(w, v) \;\&\; \mathsf{P}\big(v, w, \mathsf{rec}(w)\big) \,\big\}\Big)\Big)$$

END recursive_fcn

THEORY OF Σ

A joint application of this recursive_fcn and of

THEORY finite_induction(n, P)

Finite(n) & P(n)

==> (m)

m $\subseteq$ n & P(m) & $(\forall\, k \subseteq \mathsf{m})\big(k \neq \mathsf{m} \rightarrow \neg \mathsf{P}(k)\big)$

END finite_induction

is the (easily generalizable, and giving a Σ insensitive to operand rearrangement and grouping)

THEORY sigma_add(abel, +, u)

$(\forall\, x, y \in$ abel$)(x$+$y \in$ abel -- *closure w.r.t. ...*

$\&\ x$+$y = y$+$x)$ -- *... comm. op.*

u $\in$ abel & $(\forall\, x \in$ abel$)(x$+u$ = x)$ -- *unit el't*

$(\forall\, x, y, z \in$ abel$)\big((x$+$y)$+$z = x$+$(y$+$z)\big)$ -- *assoc.*

==> (Σ) -- *summation operation*

$\Sigma(\emptyset) =$ u

$\&\ (\forall\, x \in \mathbb{N})(\forall\, y \in$ abel$)\big(\Sigma(\{[x, y]\}) = y\big)$

is_map(F) & Finite(F) & range$(F) \subseteq$ abel

$\&$ domain$(F) \subseteq \mathbb{N} \rightarrow$

$\Sigma(F) = \Sigma(F \cap G)$ + $\Sigma(F \setminus G)$ -- *additivity*

END sigma_add

INDUCTIVE SETS – I

Preliminary to a very general construction of inductive sets, we can define *into-ness*, *injectivity*, *inductive closedness*, and *surjectivity*:

$$\mathsf{Maps}(R, S, T) \quad \leftrightarrow: \quad (\forall x \in S)(\forall y)\big(R(x, y) \rightarrow y \in T\big)$$

$$\mathsf{Disj}(R, S) \quad \leftrightarrow: \quad (\forall u, v \in S)(\forall y)\big(R(u, y) \;\&\; R(v, y) \rightarrow u = v\big)$$

$$\mathsf{IndEncl}(S, R, A) \quad \leftrightarrow: \quad A \subseteq S \;\&\; \mathsf{Maps}(R, S, S \setminus A) \;\&\; \mathsf{Disj}(R, S)$$

$$\mathsf{IndClosed}(N, R, A) \quad \leftrightarrow: \quad \mathsf{IndEncl}(N, R, A) \;\&\; (\forall t)\big(A \subseteq t \;\&\; \mathsf{Maps}(R, t, t) \rightarrow N \subseteq t\big)$$

$$\mathsf{Exhs}(R, T, S) \quad \leftrightarrow: \quad (\forall y \in T)(\exists x \in S)(R(x, y))$$

INDUCTIVE SETS – II

Then we can construct:

THEORY indClosure(s, r, a)

IndEncl(s, r, a)

==> (n, indCl)

n = indCl(a)

⋮ ⋮

$B \subseteq$ a $\rightarrow$ IndClosed(indCl(B), r, B)

IndClosed(n, r, a)

⋮ ⋮

$X \in$ n $\rightarrow$ IndClosed(indCl($\{X\}$), r, $\{X\}$)

END indClosure

THEORY weakInduction(n, r, a, p)

IndClosed(n, r, a)

$X \in$ a $\rightarrow$ p(X)

$X \in$ n & p(X) & r(X, Y) $\rightarrow$ p(Y)

==>

a = $\emptyset$ $\rightarrow$ n = $\emptyset$

Exhs(r, n $\setminus$ a, n)

$X \in$ n $\rightarrow$ p(X)

END weakInduction

INDUCTIVE SETS – III

THEORY subTree(n, r, a, tree)

IndClosed(n, r, a)

$X \in \mathsf{n} \rightarrow \mathsf{IndClosed}(\mathsf{tree}(X), \mathsf{r}, \{X\})$

==>

$\vdots \quad \vdots$

$T \neq \emptyset \;\&\; T \subseteq \mathsf{n}$

$\quad \rightarrow (\exists m \in T)(\forall u \in T)\big(m \notin \mathsf{tree}(u) \setminus \{u\}\big)$

-- N.B.: this paves the way to

-- recursive constructions over n

$\vdots \quad \vdots$

END subTree

THEORY strongInduction(n, r, a, tree, p)

IndClosed(n, r, a)

$X \in n \rightarrow \mathsf{IndClosed}(\mathsf{tree}(X), \mathsf{r}, \{X\})$

$Y \in n \;\&\; (\forall x \in n)\big(Y \in \mathsf{tree}(x) \setminus \{x\} \rightarrow \mathsf{p}(x)\big)$

$\rightarrow \mathsf{p}(Y)$

==>

$X \in n \rightarrow p(X)$

END strongInduction

Classical examples of inductive sets, which can be constructed and exploited by means of the above THEORIES are: the set $\mathbb{N}$ of *natural numbers*; the set of all *finite lists* with components drawn from a fixed base set A; the set of all *terms over a signature*

CONCLUSION

The obvious goal of modularization is to avoid repeating similar steps when the proofs of two theorems are closely analogous

Modularization must also conceal the details of a proof once they have been fed into the system and successfully certified

When coupled to a powerful underlying set theory, indefinitely expansible with new function symbols generated by Skolemization, the technical notion of "theory" illustrated above appears to meet such proof-modularization requirements