Assisting Users of Proof Assistants

David Delahaye

David.Delahaye@cnam.fr

CPR Team
(CEDRIC / CNAM)

Habilitation à Diriger les Recherches

Université Pierre et Marie Curie

Conservatoire National des Arts et Métiers

CNAM, Paris

December 9, 2010

Motivations for Dependability

Several Motivations

- Fast-Growing computerization;
- Growing delegations of responsibilities given to computer systems;
- Need for reducing costs and production lead times;
- Innovation asked by consumers and users;
- Requirements regarding the quality of the provided services.

Need for Standards

- Use of safety standards in some domains which must be fail-safe (avionics, nuclear power);
- Reinforcement of security standards (Common Criteria for Information Technology Security Evaluation);
- Recent creation of safety standards for domains which were not considered as fail-safe (automobile industry).

Motivations for Dependability

Dependability = RAMS

- Reliability: continuity of correct service;
- Availability: readiness for correct service;
- Maintainability: ability for a process to undergo modifications and repairs;
- Safety: absence of catastrophic consequences on the environment.

Use of Formal Methods

- According to the required level of safety (e.g. SIL levels of IEC 61508);
- Safety-critical and high-integrity systems;
- “Critical” generally means “when human life is at stake”;
- But we must reduce the risk “As Low As Reasonably Practicable”.

Formal Verification

Basically, two approaches:

- Model checking: exhaustive exploration of the mathematical model;
- Theorem proving: ensuring properties using logical deduction.

Theorem Proving

Many Systems

- First order / Higher order logic: B, ACL2 / Coq, HOL;
- Classical / Intuitionistic logic: PVS, HOL / ALF, NuPRL;
- Set / Type theory: B, Mizar / Coq, PVS;
- Interactive / Automated: LEGO, HOL / Vampire, Gandalf;
- Logical frameworks: Isabelle, LF.

Strong Points and Difficulties

- ▲ Generation of a statement of validity and also an evidence of this validity;
- ▼ Lack of automation (especially compared to model checking);
- ▼ In the way of building specifications;
- ▼ In the way of interacting with theorem provers.

Improving Theorem Proving

Leitmotiv

How to make theorem proving easier to use?

Organization of the Memoir

Three parts:

1. Structuring:
   - Certification of airport security regulations;
   - Code generation from specifications.
2. Automating:
   - Deduction and computer algebra;
   - Certification of automated proofs.
3. Communicating:
   - From Focalize specifications to UML models;
   - A module-based model for Focalize.

Improving Theorem Proving

Leitmotiv

How to make theorem proving easier to use?

Organization of the Memoir

Three parts:

1. Structuring;
2. Automating;
3. Communicating.

Outline of the Talk

Two groups of contributions:

1. Certification of airport security regulations;
2. Deduction and computer algebra.

Part I

Certification of Airport Security Regulations

Certification of Airport Security Regulations

The EDEMOI Project

- Integrate and apply several RE and FM techniques to analyze airport security regulations in the domain of civil aviation;
- Two-step approach:
  - Analysis of the considered standards in order to build conceptual models;
  - Development of formal models using different tools (B and Focalize).

Our Motivations

- Improve the quality of the normative documents and hence increase the efficiency of the conformity assessment procedure;
- Validate the design features as well as the reasoning support offered by Focalize, and extend this environment if needed.

Standards Considered

- The international standard Annex 17 (ICAO);
- The European Directive Doc 2320 (ECAC).

Remark: the latter is supposed to refine the former.

Certification of Airport Security Regulations

The EDEMOI Project

- Integrate and apply several RE and FM techniques to analyze airport security regulations in the domain of civil aviation;
- Two-step approach:
  - Analysis of the considered standards in order to build conceptual models;
  - Development of formal models using different tools (B and Focalize).

Our Motivations

- Improve the quality of the normative documents and hence increase the efficiency of the conformity assessment procedure;
- Validate the design features as well as the reasoning support offered by Focalize, and extend this environment if needed.

People Involved (CPR Team)

- D. Delahaye, V. Donzeau-Gouge, C. Dubois, R. Laleau;
- J.-F. Étienne, PhD student (defended on July 2008), supervised by D. Delahaye and V. Donzeau-Gouge.

Preliminary Analysis

Method Used

- A variant of the KAOS goal-oriented RE methodology (use of the WHY and HOW elaboration tactics);
- But, the requirements already exist in the form of standards and recommendations;
- Identify the fundamental security properties and determine how they are decomposed into sub-properties;
- Bottom-up approach to clearly identify the intention of each specific security property.

Annex 17 Security Properties

2.1.1 *Passengers, crew, ground personnel and the general public must be protected against acts of unlawful interference.*

Preliminary Analysis

Method Used

- A variant of the KAOS goal-oriented RE methodology (use of the WHY and HOW elaboration tactics);
- But, the requirements already exist in the form of standards and recommendations;
- Identify the fundamental security properties and determine how they are decomposed into sub-properties;
- Bottom-up approach to clearly identify the intention of each specific security property.

Annex 17 Security Properties

4.1 There are no unauthorized dangerous objects on board aircraft engaged in civil aviation.

Hidden Assumptions

Annex 17 Security Properties (1)

2.1.1 *Passengers, crew, ground personnel and the general public must be protected against acts of unlawful interference.*

Annex 17 Security Properties (2)

4.1 *There are no unauthorized dangerous objects on board aircraft engaged in civil aviation.*

where “dangerous object” denotes either a weapon, an explosive, or any other dangerous device that may be introduced on board an aircraft.

Relation of Causality

A WHY question reveals that the following assumption is made:

A1 *Acts of unlawful interference can only be committed with weapons, explosives or any other dangerous devices.*

Hidden Assumptions

Annex 17 Security Properties (1)

2.1.1 *Passengers, crew, ground personnel and the general public must be protected against acts of unlawful interference.*

Annex 17 Security Properties (2)

4.1 *There are no unauthorized dangerous objects on board aircraft engaged in civil aviation.*

where “dangerous object” denotes either a weapon, an explosive, or any other dangerous device that may be introduced on board an aircraft.

Decomposition of Property 2.1.1

$$(4.1), (A1) \vdash (2.1.1)$$

Doc 2320 Security Properties

Doc 2320

- Is supposed to clarify and refine the security measures outlined in Annex 17 at the European level;
- Each security property from Doc 2320 must not be less restrictive than or must not invalidate those from Annex 17.

Differences between Annex 17 and Doc 2320

The domain knowledge is enriched.
The formulation of the security measures is different:

- New measures are introduced;
- Each existing Annex 17 security measure is considered as follows:
  - Is reformulated, but still conveys the same information;
  - Is made more precise and sometimes more restrictive;
  - Is decomposed into further security measures;
  - Is partially refined or simply not considered.

Example of Refinement (by Precision)

Property 4.2.6 of Annex 17

4.2.6 *A minimum portion of persons (other than passengers) being granted access to security restricted areas, together with items carried, must be subjected to screening.*

Property 2.3(a) of Doc 2320

2.3(a) *All staff, including flight crew, together with items carried must be screened before being allowed access into security restricted areas. The screening procedures must ensure that no prohibited article is carried and the methods used must be the same as for passengers and cabin baggage.*

Refinement Relation

$$(2.3(a)) \vdash (4.2.6)$$

Formalization using Focalize

The Focalize Environment

- Previously Foc and Focal;
- Development of certified applications;
- Specification and proof assistant tool;
- Functional and object-oriented (inheritance, parameterization);
- Algebraic specification flavor (carrier type, implementation);
- Automated (Zenon) and verified (Coq) reasoning.

A Little History:

The BiP Working Group:

- Interactions between the Coq and B communities.

The Foc Project:

- Certified library of computer algebra;
- Structures with inheritance, representation and parameterization.

Formalization using Focalize

Two Notions of Specification

- Species:
  - Contains representation, functions, and properties;
  - Structure more or less abstract;
  - Can be combined using inheritance and parameterization.
- Collection:
  - Implements a complete species;
  - Terminal object;
  - Freezes an instance of a complete species;

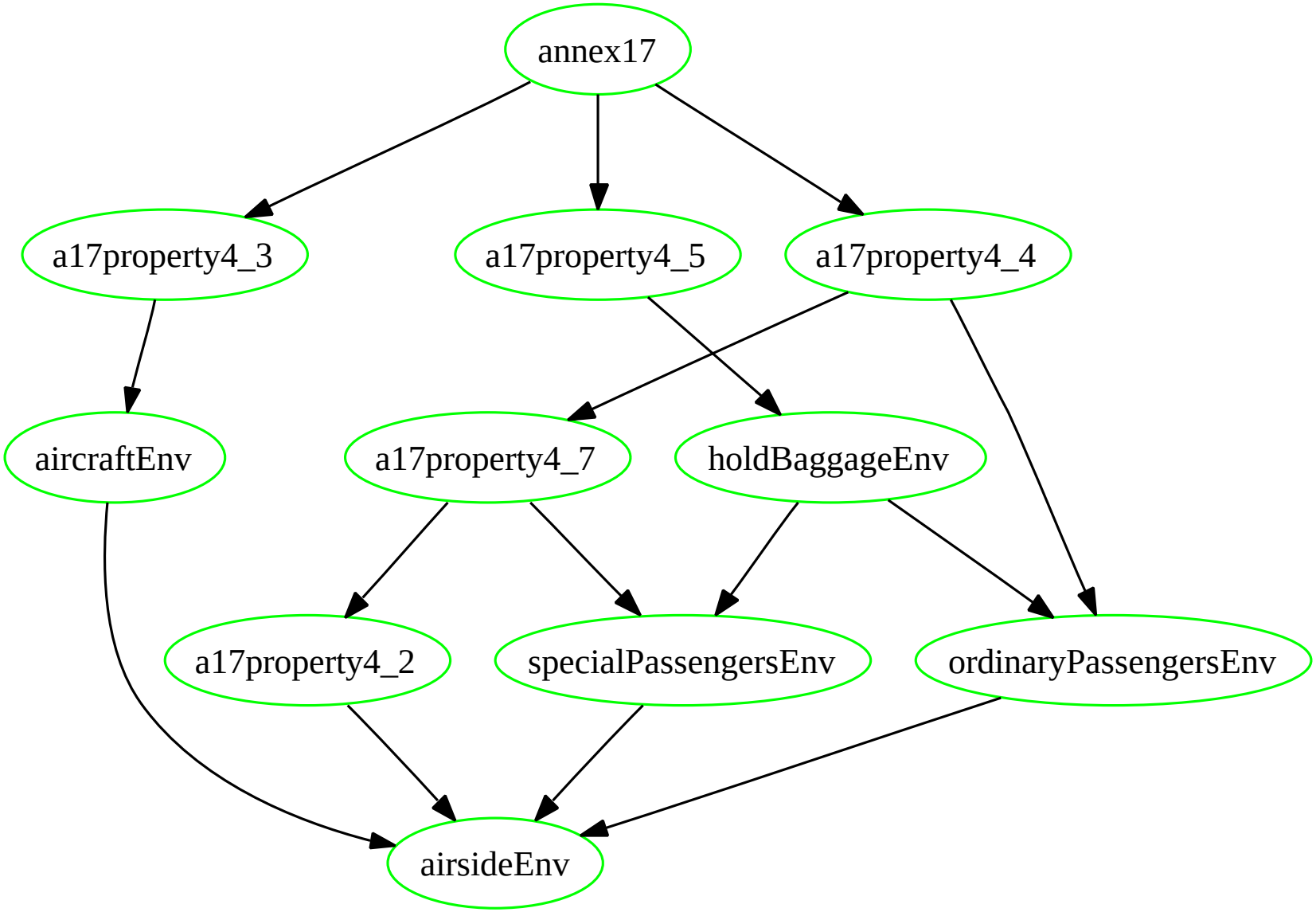
Design of a Compiler

- OCaml (execution), Coq (certification), FocDoc (documentation);
- Recently rewritten (version 0.6.0, may 2010).

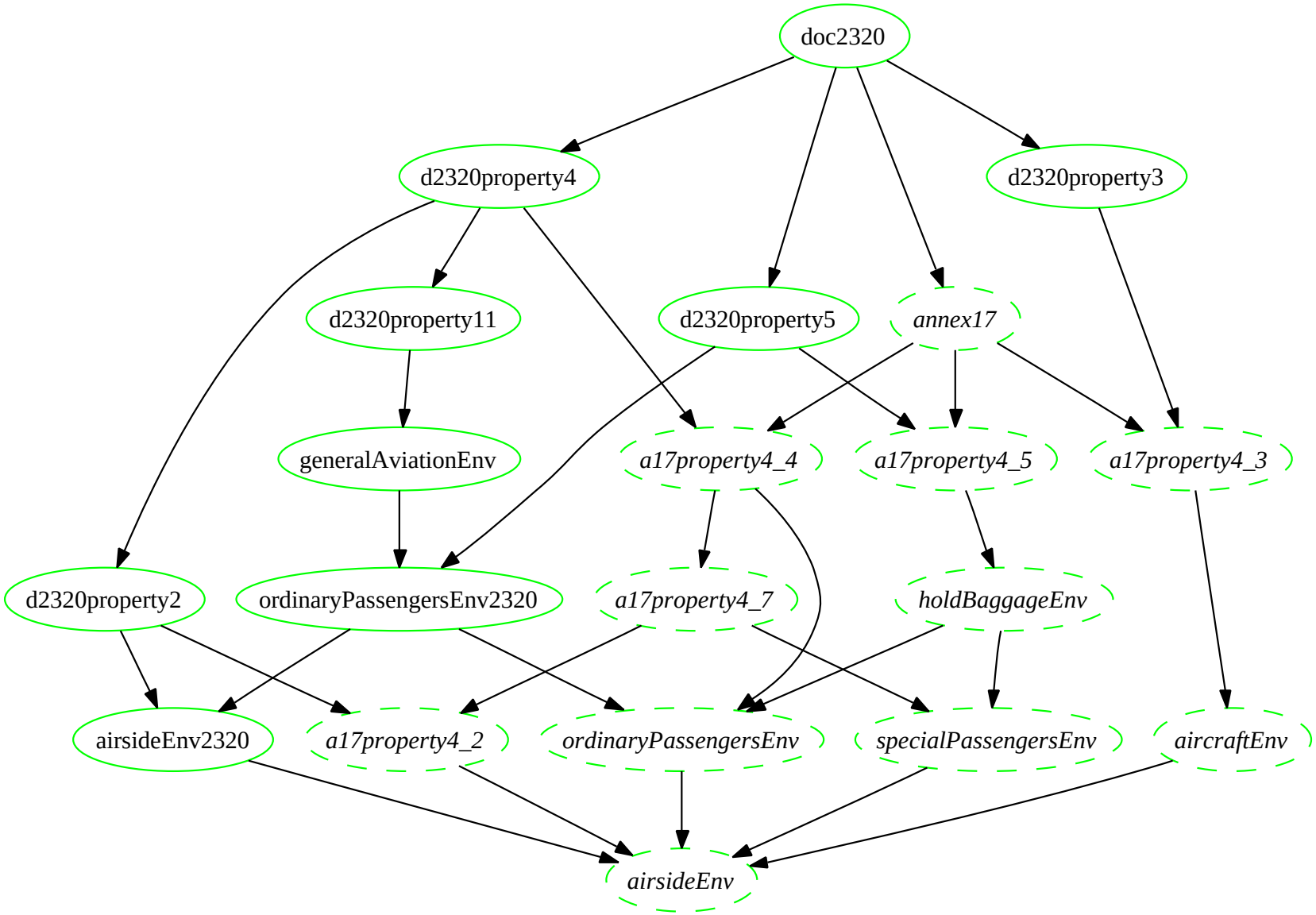
The Zenon ATP:

- First order, classical, with equality (tableaux);
- Verification by Coq (used as a proof checker).

Annex 17 Security Properties



Doc 2320 Security Properties



Remarks regarding the Formalization

The Formal Models in Figures

- 2 regulations formalized;
- About 10,000 lines of code;
- With 150 species and 200 proofs;
- 2 years to be finalized.

Some Publications

- D. Delahaye, J.-F. Étienne, and V. Viguié Donzeau-Gouge. Certifying Airport Security Regulations using the Focal Environment. In *FM*, 2006.
- D. Delahaye, J.-F. Étienne, and V. Viguié Donzeau-Gouge. Reasoning about Airport Security Regulations using the Focal Environment. In *ISoLA*, 2006.
- D. Delahaye, J.-F. Étienne, and V. Viguié Donzeau-Gouge. *Modeling and Certifying Airport Security Regulations*. Defense, Security and Strategies. 2010.

Remarks regarding the Formalization

The Formal Models in Figures

- 2 regulations formalized;
- About 10,000 lines of code;
- With 150 species and 200 proofs;
- 2 years to be finalized.

Appropriateness of Focalize

Inheritance (refinement) and parameterization (modularity):

- Separation between the domain knowledge and the security properties;
- Natural classification determined for the subjects being regulated;
- Correlation between Annex 17 and Doc 2320;
- Factorization of our development (vocabulary differences).

Remarks regarding the Formalization

The Formal Models in Figures

- 2 regulations formalized;
- About 10,000 lines of code;
- With 150 species and 200 proofs;
- 2 years to be finalized.

Appropriateness of the Reasoning Support

- The declarative-like proof language appears quite natural;
- Zenon discharged most of the proof obligations automatically;
- Quite useful in the prototyping phase to be convinced that a given lemma is correctly formulated;
- Also quite useful in the finalizing phase to obtain more readable specifications together with a reasonable compilation time.

Suitable Evolutions

- Integration of temporal mechanisms (behavioral properties).

Documentation of the Formal Models

UML Diagrams

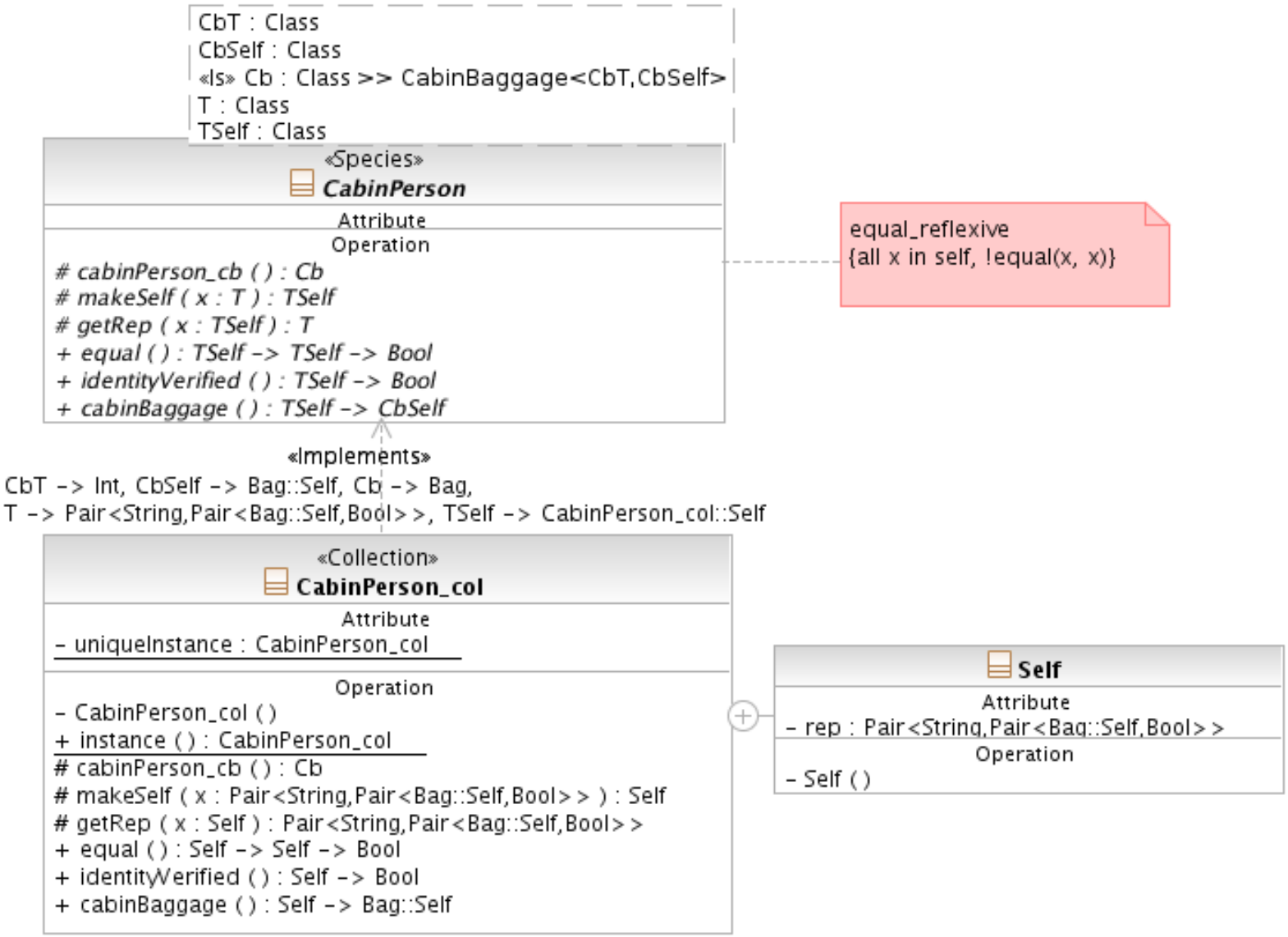
- Graphical documentation of the formal models for developers;
- Higher-level views pertinent to certification authorities.

Our Major Concern

A formal framework for an automatic transformation from Focalize to UML:

1. Formalize a subset of the UML 2.1 static structure constructs;
2. Extend the UML metamodel (via profile mechanism);
3. Describe the transformation rules from Focalize to UML;
4. Establish the soundness of the transformation.

An Example of Generated UML Model



Implementation

Two Parts

From FocDoc: XML format used by the Focalize compiler for documentation.

1. UML profile for Focalize specified with the UML2 Eclipse plug-in:
2. XSLT stylesheet that encodes the transformation rules.

Some Publications

- D. Delahaye, J.-F. Étienne, and V. Viguié Donzeau-Gouge. A Formal and Sound Transformation from Focal to UML: An Application to Airport Security Regulations. *ISSE NASA Journal*, 2008.
- D. Delahaye, J.-F. Étienne, and V. Viguié Donzeau-Gouge. Producing UML Models from Focal Specifications: An Application to Airport Security Regulations. In *TASE*, 2008.

Part II

Deduction and Computer Algebra

Skeptical Computations and Deductions

Motivations

- Provide more automation to PAs;
- Provide not only computations, but also deductions;
- Using external tools (dedicated to computation or automated deduction);
- Without breaking the consistency of the PA.

Several Approaches

- Believing approach:
  - The correction of the computation/deduction is assumed;
  - The consistency is not ensured and it is not satisfactory.
- Skeptical approach:
  - The correction of the computation/deduction is verified;
  - The consistency remains ensured.
- Autarkic approach:
  - The computation/deduction is realized within the PA;
  - The consistency remains ensured, but there is no externalization.

Skeptical Computations

Difficulties with Computations within PAs

- Constraints imposed by the environment of the PA;
- Termination required for consistency purposes;
- Choice of data structures:
  - Either suitable for proofs: e.g. Peano arithmetic;
  - Or suitable for computations: binary integers.

Reconcile Validation with Efficiency

- Relax the previous constraints;
- Notion of local correctness;
- Verify the correctness of each application of a function;
- The function is not constrained (black box).

Interactions between PAs and CASs

1. Import into Coq computations from Maple over fields;
2. Implement a quantifier elimination procedure over ACFs in Coq.

Skeptical Computations

Difficulties with Computations within PAs

- Constraints imposed by the environment of the PA;
- Termination required for consistency purposes;
- Choice of data structures:
  - Either suitable for proofs: e.g. Peano arithmetic;
  - Or suitable for computations: binary integers.

Reconcile Validation with Efficiency

- Relax the previous constraints;
- Notion of local correctness;
- Verify the correctness of each application of a function;
- The function is not constrained (black box).

People Involved

- D. Delahaye, M. Mayero;
- With the assistance of T. Coquand.

Computations from Maple to Coq

Principle

- The computations over fields are realized in Maple;
- They are then imported into Coq, which is asked to validate them;
- Neither local nor global correctness is ensured;
- It only guarantees that the computation uses operations over fields.

Choice of the Tools

- Maple: popular and easy to use;
- Coq: automatic validation thanks to the tactic “field” (written in $\mathcal{L}_{tac}$).

Exported Functions

- “simplify”: applies simplification rules to an expression;
- “factor”: factorizes a multivariate polynomial;
- “expand”: expands an expression;
- “normal”: normalizes a rational expression.

An Example of Imported Computation

Proposition to be Proved in Coq

Given x and y two non-zero elements of an ordered field:

$$(\frac{x}{y} + \frac{y}{x})\ x.y - (x.x + y.y) + 1 > 0$$

Call of Maple

- We invoke “simplify” with the left-hand side member of the inequation;
- This application returns 1 as result.

Skeptical Approach

The following equation must be generated:

$$(\frac{x}{y} + \frac{y}{x})\ x.y - (x.x + y.y) + 1 = 1$$

- To prove this equation, the tactic “field” is called;
- It succeeds and generates the condition $x.y \neq 0$ (true by hypotheses).

Implementation

Interface between Coq and Maple

- Code available as a Coq contribution;
- Quite short with about 300 lines of ML;
- Basic system of pipes between Coq and Maple;
- Extensible with other functions (with higher arities).

Some Publications

- D. Delahaye and M. Mayero. Dealing with Algebraic Expressions over a Field in Coq using Maple. *JSC*, 2005.
- D. Delahaye and M. Mayero. `Field`: une procédure de décision pour les nombres réels en Coq. In *JFLA*, 2001.

An Extension

- A quantifier elimination procedure over ACFs in Coq;
- Mainly relies on gcd computations, which can be externalized.

Quantifier Elimination Procedure over ACFs

Definition of an ACF

An algebraically closed field K is a field s.t.:

$$\forall P \in K[X].\deg(P) > 0 \Rightarrow \exists x \in K.P(x) = 0$$

We can solve systems of the form:

$$\begin{cases} P_1(X) = 0, \ldots, P_n(X) = 0 \\ Q_1(X) \neq 0, \ldots, Q_m(X) \neq 0 \end{cases}$$

Quantifier Elimination

- Gets rid of the polynomial parts which do not contain the solution;
- Heavily relies on computations of polynomial gcds;
- Many different algorithms of polynomial gcd with several complexities;
- Integrate this procedure into a PA;
- Externalize the computations of gcds using a CAS;
- Developed for Coq using the interface with Maple.

Quantifier Elimination Algorithm

General Case

Given P the gcd of P_i and Q the product of Q_i (or the lcm of Q_i):

- P and Q are relatively prime: the system is reduced to $P = 0$;
- Otherwise: the system is $P_1 = 0$ and $G \neq 0$,
  where G is the gcd of P and Q, and where P_1 is s.t. $P = GP_1$.

Example

$$\begin{cases} 3X^3 + 10X^2 + 5X + 6 = 0 \\ 2X^2 + 5X - 3 \neq 0 \end{cases}$$

Given $P = 3X^3 + 10X^2 + 5X + 6$ and $Q = 2X^2 + 5X - 3$:
The gcd of P and Q is $G = X + 3$.
P and Q are not relatively prime:
We have $P_1 = 3X^2 + X + 2$ s.t. $P = GP_1$.

Quantifier Elimination Algorithm

General Case

Given P the gcd of P_i and Q the product of Q_i (or the lcm of Q_i):

- P and Q are relatively prime: the system is reduced to $P = 0$;
- Otherwise: the system is $P_1 = 0$ and $G \neq 0$,
  where G is the gcd of P and Q, and where P_1 is s.t. $P = GP_1$.

Example

$$\begin{cases} 3X^2 + X + 2 = 0 \\ X + 3 \neq 0 \end{cases}$$

The gcd of P_1 and G is 1.
P and Q are relatively prime:
The system is $P_1 = 0$ (by definition of ACF).

Externalization of the gcd Computation

Skeptical Approach

- The verification must be stronger;
- The result must be the gcd and not only a divisor.

Bézout Relation

Given P, Q and G three non-zero polynomials:

If G divides P and Q, and if there exist two polynomials A and B s.t. $AP + BQ = G$ then G is the gcd of P and Q.

Certificates

In addition to the gcd G, we need:

- The two quotients P_1 and Q_1 s.t. $P = GP_1$ and $Q = GQ_1$;
- The two cofactors A and B.

Implementation

Proof Procedure

- Extension of the interface between Coq and Maple;
- Deal with the additional certificates;
- Verify three polynomial equations (using the tactic “field”);
- Quite transparent and automatic for the user.

Some Publications

- D. Delahaye and M. Mayero. Quantifier Elimination over Algebraically Closed Fields in a Proof Assistant using a Computer Algebra System.
  In *Calculemus*, 2005.

Conclusion: Perspectives

Development of Focalize

- Temporal mechanisms to deal with behavioral properties;
- Recursive species (e.g. real closed fields);
- Invariants over representations;
- Extensions formally and semantically founded;
- More operational model;
- Encoding in a calculus with polymorphism and dependent types.

Deduction and Computer Algebra

- Progressive line of work: fields, algebraically closed fields;
- Real closed fields: CAD as a test procedure (Focalize/Axiom);
- CAD as a proof procedure (still in the same framework);
- Local Certification of an algorithm of CAD: which certificates?

Code Generation from Specifications

Main Goal

Execute specifications.

Problem and Constraint

How to execute inductive relations?

- They may have several computational behaviors;
- Their evaluation may require backtracking.

Remain purely functional.

- Target languages for extraction are functional;
- Theorem prover languages are functional.

Our Method

In the framework of Coq (initially), and Focalize (thereafter):

- A mode analysis to know if a computation is possible;
- A code generation with heuristics to remain functional.

Code Generation from Specifications

Main Goal

Execute specifications.

Problem and Constraint

How to execute inductive relations?

- They may have several computational behaviors;
- Their evaluation may require backtracking.

Remain purely functional.

- Target languages for extraction are functional;
- Theorem prover languages are functional.

People Involved

- D. Delahaye, C. Dubois, J.-F. Étienne (Master student);
- P.-N. Tollitte, PhD student, supervised by D. Delahaye and C. Dubois.

Related Work

Semantics of Programming Languages

- Centaur (Centaur project, 1988);
- RML translator (M. Pettersson, 1996);
- Natural semantics to ML (C. Dubois, R. Gayraud, 1999);
- Maude (A. Verdejo, N. Martí-Oliet, 2006).

More General Framework

- Isabelle/HOL (S. Berghofer, T. Nipkow, 2000);
- Recently extended (S. Berghofer, L. Bulwahn, F. Haftmann, 2009).

Our approach

- No use of the logic programming paradigm;
- Extraction method formalized and soundness/completeness proved.

Examples of Code Generation

Addition Relation

```
Inductive add : nat → nat → nat → Prop :=
  | add_O : forall n : nat, add n O n
  | add_S : forall n m p : nat, add n m p → add n (S m) (S p).
```

Extraction with Mode $\{1,2\}$

```
let rec add p0 p1 = match p0, p1 with
  | n, O → n
  | n, S m → let p = add n m in S p
```

Examples of Code Generation

Addition Relation

```
Inductive add : nat → nat → nat → Prop :=
  | add_O : forall n : nat, add n O n
  | add_S : forall n m p : nat, add n m p → add n (S m) (S p).
```

Extraction with Mode $\{3, 2\}$

```
let rec add p0 p1 = match p0, p1 with
  | n, O → n
  | S p, S m → add p m
  | _ → assert false
```

Examples of Code Generation

Addition Relation

```
Inductive add : nat → nat → nat → Prop :=
  | add_O : forall n : nat, add n O n
  | add_S : forall n m p : nat, add n m p → add n (S m) (S p).
```

Extraction with Mode $\{1, 2, 3\}$

```
let rec add p0 p1 p2 = match p0, p1, p2 with
  | n, O, m when n = m→ true
  | n, S m, S p→ add n m p
  | _→ false
```

Mode Consistency Analysis

Algorithm

Data-flow analysis (inputs/outputs):

- A mode $\equiv$ a set of input positions;
- At most one output position.

Addition Relation

```
Inductive add : nat → nat → nat → Prop :=
  | add_O : forall n : nat, add n O n
  | add_S : forall n m p : nat, add n m p → add n (S m) (S p).
```

Consistency of Mode $\{1, 2\}$

- *add_0*: $S_0 = \{n\}$, $\{n\} \subseteq S_0$;
- *add_S*: $S_0 = \{n, m\}$, $\{n, m\} \subseteq S_0$, $S_1 = \{n, m, p\}$, $\{p\} \subseteq S_1$.

Mode Consistency Analysis

Algorithm

Data-flow analysis (inputs/outputs):

- A mode $\equiv$ a set of input positions;
- At most one output position.

Typing Relation

```
Inductive typing : env → term → type → Prop := ...
  | abs : forall (e : env) (t1 t2 : type) (x : var) (t : term),
    (typing (add_env (x, t1) e) t t2) →
    (typing e (Abs (x, t)) (Arr t1 t2)).
```

Inconsistency of Mode $\{1, 2\}$

t1 $\notin S_0 = \{e, x, t\}$

Code Generation

Addition Relation

```
Inductive add : nat → nat → nat → Prop :=
  | add_O : forall n : nat, add n O n
  | add_S : forall n m p : nat, add n m p → add n (S m) (S p).
```

Extraction with Mode $\{1,2\}$

```
let rec add12 (p1, p2) =
  match (p1, p2) with
  (* code generation for inductive clauses *)
```

Code Generation

Addition Relation

```
Inductive add : nat → nat → nat → Prop :=
  | add_O : forall n : nat, add n O n
  | add_S : forall n m p : nat, add n m p → add n (S m) (S p).
```

Extraction with Mode $\{1, 2\}$

```
let rec add12 (p1, p2) =
  match (p1, p2) with
  | (n, O) → (* code generation for set of premises 1 *)
  | (n, S m) → (* code generation for set of premises 2 *)
```

Code Generation

Addition Relation

```
Inductive add : nat → nat → nat → Prop :=
  | add_O : forall n : nat, add n O n
  | add_S : forall n m p : nat, add n m p → add n (S m) (S p).
```

Extraction with Mode $\{1, 2\}$

```
match f1 (t11, ..., t1n1) with
| out1 →
  (match f2 (t21, ..., t2n2) with
    | out2 →
      (... → (* result of the inductive clause *)))
```

Code Generation

Addition Relation

```
Inductive add : nat → nat → nat → Prop :=
  | add_O : forall n : nat, add n O n
  | add_S : forall n m p : nat, add n m p → add n (S m) (S p).
```

Extraction with Mode $\{1, 2\}$

```
let rec add12 (p1, p2) =
  match (p1, p2) with
  | (n, O) → n
  | (n, S m) →
    (match add12 (n, m) with
     | p → S p);
```

Formalization and Implementation

Formalization

- Extraction method formalized;
- Soundness and completeness proved.

Implementation

- Implemented for the latest version of Coq;
- Integrated to the usual extraction mechanism;
- Several extraction outputs: OCaml, Scheme, Haskell;
- Some optimizations also integrated.

Optimizations

Semantics of “while”

```
Inductive exec : store → command → store → Prop := ...
  | while1 : forall (s s1 s2 : Sigma) (b : expr) (c : command),
      (eval s b true) → (exec s c s1) →
      (exec s1 (while b do c) s2) → (exec s (while b do c) s2)
  | while2 : forall (s : Sigma) (b : expr) (c : command),
      (eval s b false) → (exec s (while b do c) s).
```

Extraction with Mode $\{1, 2\}$

```
let rec exec s c = match s, c with ...
  | s, While (b, c) →
    (match (eval s b) with
       | true →
         let s1 = exec s c in
         let s2 = exec s1 (While (b, c)) in s2
       | false → s)
```

Optimizations

Semantics of “while”

```
Inductive exec : store → command → store → Prop := ...
  | while1 : forall (s s1 s2 : Sigma) (b : expr) (c : command),
      (eval s b true) → (exec s c s1) →
      (exec s1 (while b do c) s2) → (exec s (while b do c) s2)
  | while2 : forall (s : Sigma) (b : expr) (c : command),
      (eval s b false) → (exec s (while b do c) s).
```

Larger Scale

- Able to deal with almost all examples of semantics;
- Extraction of the semantics of an intermediate language from CompCert.

Extension to Focalize

Major Evolutions

- Functional code generation within the framework of Focalize;
- Correctness theorems are also generated;
- Extraction realized by closing a set of properties.

Constraints

- Termination: structural recursion;
- Non-linearity (inputs of the conclusion): determinism.

Features to be Investigated

- A relation is never closed (inheritance), except for collections;
- New scheme of dependency computation (problem of design).