Inferring the Proof Process

Andrius Velykis

School of Computing Science
Newcastle University, UK

FM2012 Doctoral Symposium
Paris, 27 August 2012

Understand interactive proof

Focus on industrial models with *families* of similar proof obligations

Capture expert's proof process

Collect information to extract reusable proof strategies

Formal proof to justify development steps

Relevant *proof obligations* (POs) are often generated automatically and need to be discharged

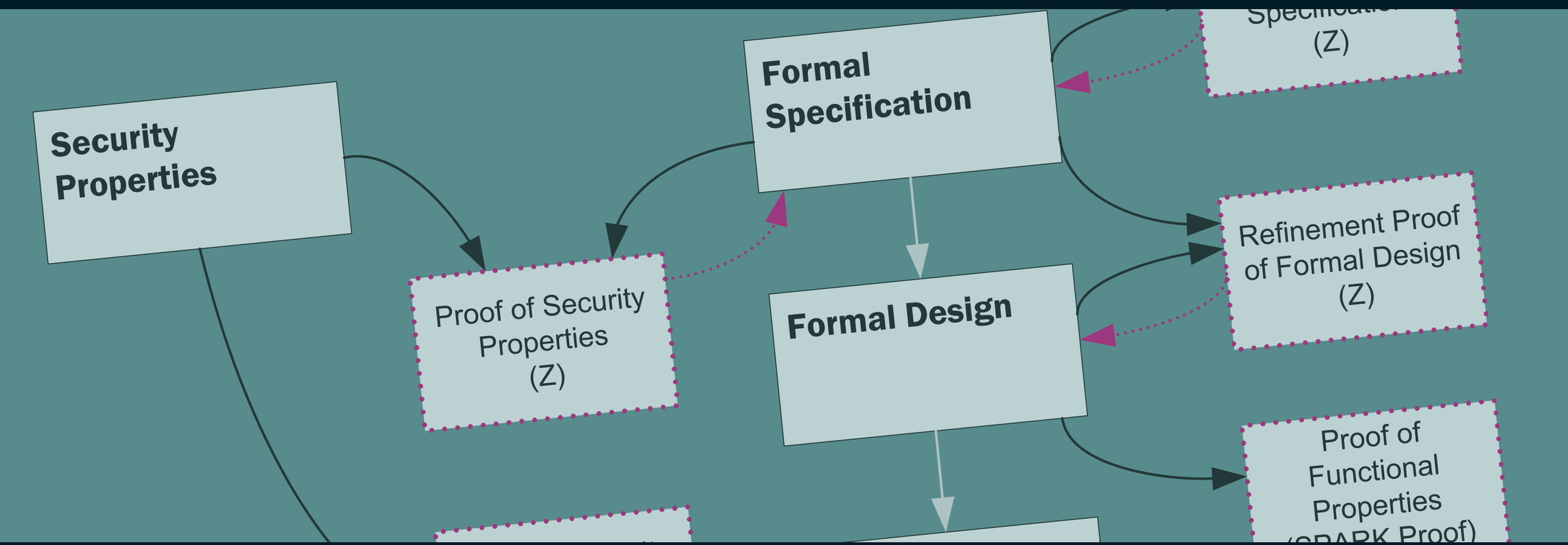


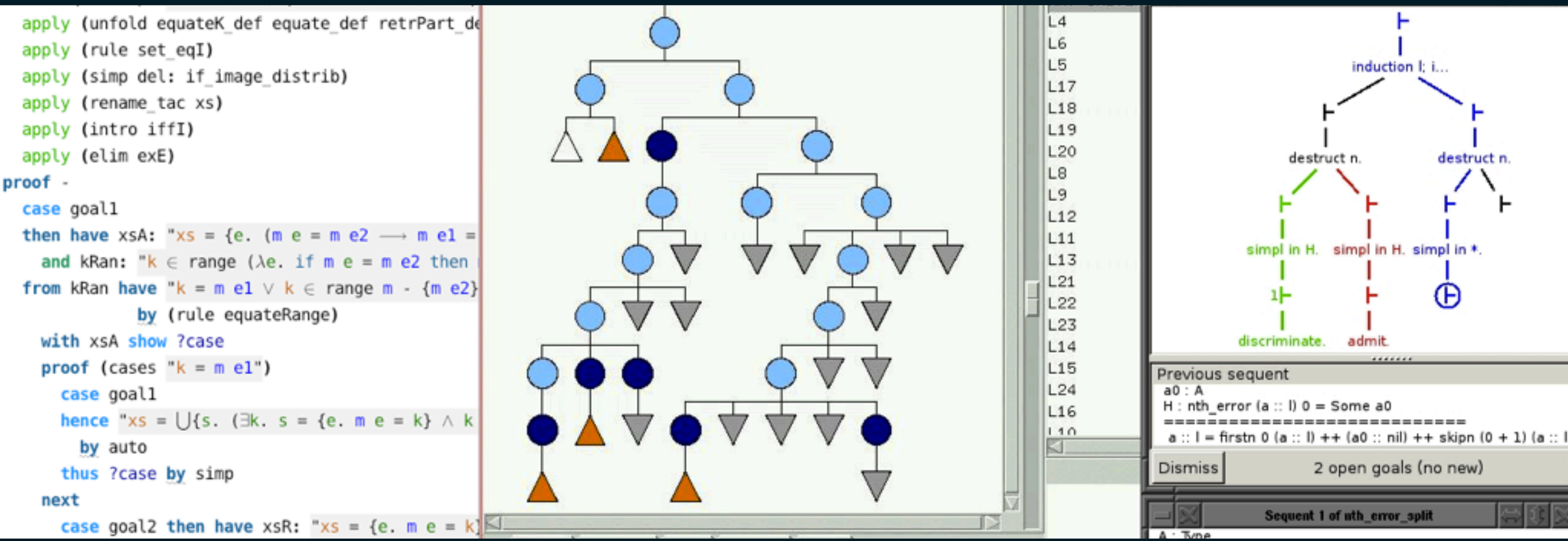
Diagram excerpt from *Tokeneer ID Station* report. Praxis, 2008.

Picking up after proof automation

- Interactive proof of remaining POs is difficult & time consuming
- Complexity / domain specific ideas
  - Generic heuristics not as effective
- Single idea for similar proofs - can we reuse it?

Motivation

- Extract & reuse high-level proof strategy (AI4FM goal)
- Capture enough information to facilitate understanding of high-level reasoning
- Can we infer the *proof process* automatically?



What describes a proof process?

Things we need to collect (and infer) to capture expert's ideas

Proof granularity

Layers of abstraction: from high-level plan to tactic steps

Proof structure

Parallel proof branches instead of a sequence of steps

Multiple attempts

Full development, including failed attempts

```
1987 1999
1988 2000  text{*computing div by shifting *}
1989 2001
1990 2002  lemma pos_zdiv_mult_2: "(0::int) \<le> a ==> (1 + 2*b) div (2*a) = b div a"
1991      - proof cases
1992      -   assume "a=0"
1993      -     thus ?thesis by simp
1994      - next
1995      -   assume "a\<noteq>0" and le_a: "0\<le>a"
1996      -   hence a_pos: "1 \<le> a" by arith
1997      -   hence one_less_a2: "1 < 2 * a" by arith
1998      -   hence le_2a: "2 * (1 + b mod a) \<le> 2 * a"
1999      -     unfolding mult_le_cancel_left
2000      -     by (simp add: add1_zle_eq add_commute [of 1])
2001      -   with a_pos have "0 \<le> b mod a" by simp
2002      -   hence le_addm: "0 \<le> 1 mod (2*a) + 2*(b mod a)"
2003      -     by (simp add: mod_pos_pos_trivial one_less_a2)
2004      -   with  le_2a
2005      -   have "(1 mod (2*a) + 2*(b mod a)) div (2*a) = 0"
2006      -     by (simp add: div_pos_pos_trivial le_addm mod_pos_pos_trivial one_less_a2
2007      -                   right_distrib)
2008      -   thus ?thesis
2009      -     by (subst zdiv_zadd1_eq,
2010      -         simp add: mod_mult_mult1 one_less_a2
2011      -                   div_pos_pos_trivial)
2012      - qed
      2003 +   using pos_divmod_int_rel_mult_2 [OF _ divmod_int_rel_div_mod]
      2004 +   by (rule div_int_unique)
2013 2005
2014 2006  lemma neg_zdiv_mult_2:
2015 2007    assumes A: "a \<le> (0::int)" shows "(1 + 2*b) div (2*a) = (b+1) div a"
```

Proof cleanup

One-liners lose ideas how the proof was reached

Intent

Tags with high-level explanation (coupled with *features*)

Proof features

What triggered the proof step?

- Shape of assumption / goal
- Available lemma
- Certain datatypes or records

Infer—or—ask the expert (*à la* Clippy in MS Office)?

ProofProcess framework

- “Wire-tap” theorem prover
  - Prototypes for Isabelle/HOL and Z/EVES
- Generic core + prover-specific extensions
- Matchers for features/structure
- Proof history
- Accommodate any proof process?

Architecture

ProofProcess model

- Intent
- Proof features

- Attempts
- Granularity
- Structure

Abstract term data

- Proof scripts
- Proof history

Create ProofProcess Model

Ask expert what's happening

Infer proof process

Apps

"Clippy"

Proof replay

App X.. ?

ProofProcess Integration

PP: Isabelle

PP: Z/Eves

PP: Rodin

PP: ...?

Theorem Proving System

Expert

Capture the role of lemmas

- Lemmas about data structures facilitate automatic proof
- Capture the need for lemma and its features
- Trace lemma usage in automatic proof tactics (e.g. `apply auto`)

Find affected goal terms

- Identify expert's path of reasoning by narrowing the scope
- Discover "unimportant" terms by inverting the scope
- Extract a toy problem

Fuzzy match with previous proofs

- Consult captured proof process to recognise a similar one
  - Partial match proof features and intents
  - Match any part of previous proof process
  - Graph matching algorithms?
- Similar to strategy extraction/reuse

Proof Process

extract and reuse proof strategies

improve proof maintenance

train interactive theorem proving

Understand interactive proof

Focus on industrial models with *families* of similar proof obligations

Capture expert's proof process

Collect information to extract reusable proof strategies

http://arnaudrius.velykis.lt