

The use of AI to support top-down formal system development

Gudmund Grov
University of Edinburgh, UK

Joint work with: Alan Bundy, Andrew Ireland, Cliff Jones,
Teresa Llano, Leo Freitas, Michael Butler and others

ETH Zürich, January 2011



THE UNIVERSITY of EDINBURGH
informatics



Outline

“In this talk I will discuss proof automation by exploring proof similarity and how to use proof failures to inform modelling”

- ▶ Structure
 - ▶ Background & motivation
 - ▶ Ideas on how to recycle proofs by abstraction
 - ▶ Suggesting re-modelling from proof-failure analysis
- ▶ Funded by UK EPSRC AI4FM project (<http://www.ai4fm.org>)
 - ▶ Edinburgh(Bundy,Grov,Lin)
 - ▶ Newcastle (Jones, Freitas, Velykis)
 - ▶ Heriot-Watt (Ireland) & Southampton (Butler)

Top-down formal methods

- ▶ **Top-down formal methods** is based upon step-wise development of systems from requirements to final product
 - ▶ (design) errors captured early – cheaper to fix!
- ▶ The user **posit** specifications which are justified by **proof**
 - ▶ Proof obligations (POs) are often automatically generated
- ▶ Handles complexity by layers of abstraction
- ▶ Many remarkably similar (with similar POs)
 - ▶ e.g, VDM, Z, B, Event-B

Motivation

- ▶ 5 – 20% of POs require interactive proofs
- ▶ TP of POs has become a bottleneck for deployment!
- ▶ POs are rarely deep (layering reduces complexity)
- ▶ Lots of POs...
- ▶ but many follows from a similar “idea”
 - ▶ thus, forming a “proof/PO family”
 - ▶ Deploy case-study: 300 POs/100 IPOs/5 families
- ▶ Lots of details (“noise”)
- ▶ Models often change (carries across layers)
- ▶ Proof failure often guides modelling changes

Reuse of proofs

- ▶ “Cut-and-paste” style reuse of tactic or proof term not robust
 - ▶ slight variations causes failure
- ▶ Need to capture **key idea/steps** of proof
- ▶ Requires us to abstract the proof
 - ▶ develop a **strategy language** to represent abstraction
- ▶ Work at the **proof plan** level
 - ▶ enables meta-reasoning about proof/tactic
- ▶ **Classify** POs into similarity-families

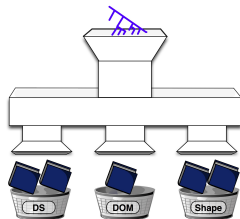
Abstracting proofs: Explanation Based Generalisation

- ▶ First coined by DeJong
- ▶ Extraction of general rules from individual observations
- ▶ Example approach to generalise a proof:
 - ▶ replace constants by variable in conjecture
 - ▶ rerun (regress) varified proof
 - ▶ rule application may cause variables to be instantiated
 - ▶ leaf nodes become assumptions
- ▶ Reminiscent of AWE project in Bremen¹
 - ▶ applied to proof term
 - ▶ type abstractions into type variables
- ▶ However, proof terms are low level
- ▶ ... and we want more structure

¹see <http://www.informatik.uni-bremen.de/~cxl/awe/>

Enhancing with structure

- ▶ Analyse over several dimensions of information:



- ▶ Need to identify key steps of proofs
- ▶ ... and chunk other steps together
 - ▶ which EBG can help with

Chunking/key steps: possible impact of structured proofs

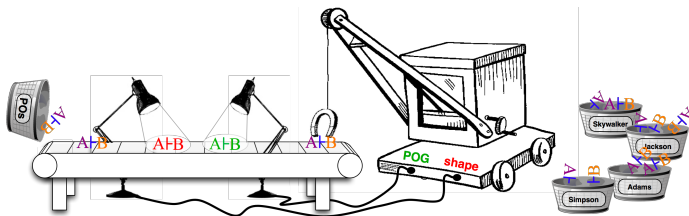
- ▶ Many possible ways of chunking steps
- ▶ Alternative: require the user to provide key steps
 - ▶ e.g. through structured proofs

```
lemma X
  apply t1
  ⋮
  apply tn
done
```

```
lemma X
proof -
  have key_step1 proof ... qed
  ⋮
  have key_stepm proof ... qed
qed
```

Proof families, classification & similarity

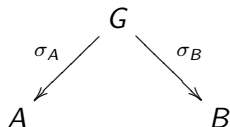
- ▶ We hope to classify POs into families from “key idea”:



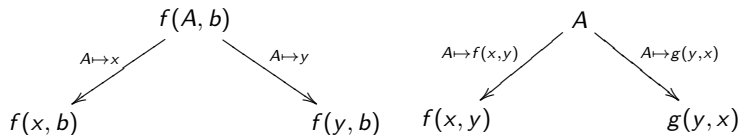
- ▶ Based upon various dimensions
- ▶ Requires a metric of similarity

Shape similarity through anti-unification

- ▶ Find the generalisation G of two term A and B

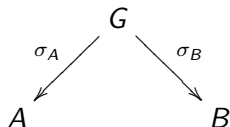


- ▶ Key is finding least general generalisation
- ▶ First-order case coined by Plotkin — examples:

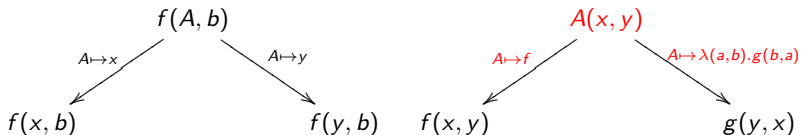


Shape similarity through anti-unification

- ▶ Find the generalisation G of two term A and B



- ▶ Key is finding least general generalisation
- ▶ **Higher-order** case studied by Krumnack et al:



Example: anti-unification of two terms

source	target	generalisation
$s(x) + y = s(x + y)$	$(h :: t)@l = ?$	$? = ?$

Example: anti-unification of two terms

source	target	generalisation
$s(x) + y = s(x + y)$	$(h :: t)@l = ?$	$? = ?$
$s(x) + y = s(x + y)$	$(h :: t)@l = ?$	$F(C(X), Y) = ?$

Example: anti-unification of two terms

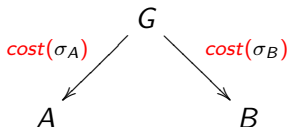
source	target	generalisation
$s(x) + y = s(x + y)$	$(h :: t)@l = ?$	$? = ?$
$s(x) + y = s(x + y)$	$(h :: t)@l = ?$	$F(C(X), Y) = ?$
$s(x) + y = s(x + y)$	$(h :: t)@l = ?$	$F(C(X), Y) = C(F(X, Y))$

Example: anti-unification of two terms

source	target	generalisation
$s(x) + y = s(x + y)$	$(h :: t)@l = ?$	$? = ?$
$s(x) + y = s(x + y)$	$(h :: t)@l = ?$	$F(C(X), Y) = ?$
$s(x) + y = s(x + y)$	$(h :: t)@l = ?$	$F(C(X), Y) = C(F(X, Y))$
$s(x) + y = s(x + y)$	$(h :: t)@l = h :: (t@l)$	$F(C(X), Y) = C(F(X, Y))$

Shape similarity through anti-unification – metric

- ▶ By augmenting costs to substitutions:



- ▶ we can find a metric of shape similarity/difference:

$$\text{diff} = \text{cost}(\sigma_A) + \text{cost}(\sigma_B)$$

- ▶ and use this in PO classification.
- ▶ This shape metric can be enhanced other dimensions
 - ▶ e.g. domain knowledge (like properties of operators) or POG

Automating remodelling from failed proofs

- ▶ Proof of POs ensures correctness of model
- ▶ Mistakes in model often manifested as failed proofs
 - ▶ models must thus be changed
 - ▶ the user must analyse proof and understand model
- ▶ **Proof critics**
 - ▶ capture common patterns of failures with patches
 - ▶ e.g. lemma speculation, generalisation, proof-by-cases
- ▶ Can we extend critics to capture patterns of failures with patterns of re-modelling?
 - ▶ automate the model/proof analysis
- ▶ Represent as suggestions to user
 - ▶ Crucial that tool does not automatically change model

A simple illustrative Event-B example

- ▶ Event-B is a FM for top-down development of discrete systems.
 - ▶ A **machine** consists of **variables**, **invariants** and **events**
- ▶ Example: (part of) a simple cruise control system:

MACHINE cruise_ctrl

VARIABLES brake, cc

INVARIANTS inv1: *cc = on \Rightarrow brake = off*

EVENTS

...

EVENT pressbrake $\hat{=}$

BEGIN

act1 : *brake := on*

END

...

END

Failed proofs

- ▶ The generated PO for pressbrake fails:

$$(cc = on \Rightarrow brake = off) \vdash \\ \{brake \mapsto on\}(cc = on \Rightarrow brake = off)$$

- ▶ One solution is to introduce an invariant, e.g.

$$inv2 : cc = off$$

- ▶ Or change the event to either

```
EVENT pressbrake  $\hat{=}$   
WHEN  
    grd1: cc = off  
THEN  
    act1 : brake := on  
END
```

```
EVENT pressbrake  $\hat{=}$   
BEGIN  
    act1 : brake := on  
    act2 : cc := off  
END
```

Extending the critics with modelling suggestions

critic (action speculation)

INPUTS:

PO_SET POs

MODEL_SET $\{M\}$

PRECONDITIONS:

1. $\exists failed_po \in \{\langle -, -, -, PO \rangle \in POs \mid failed_proof(PO)\}$.
 $failed_po = \langle M, E, _ / INV, (\Delta, X \Rightarrow Y \vdash \sigma(X \Rightarrow Y)) \rangle$
2. $\exists \tau \in sub. disjoint_sub(\tau, \sigma) \wedge provable(\Delta \vdash (\tau \cup \sigma)X \Rightarrow false)$

OUTPUTS:

GUIDE $add_action(sub2act(\tau), E, M)$

1. there exists an unproven invariant PO (implication)
2. there exists a variable update that does not interfere with the existing actions and which makes the PO provable

A similar critic can be defined to speculate a guard

Meta-information in models

- ▶ This simple example has (at least) 3 possible solutions
- ▶ In general, many solutions which fixes a model
- ▶ Add meta-information to guide critics
- ▶ Example: a priority heuristic on variables:

MACHINE *cruise_ctrl*

VARIABLES *brake, cc*

INVARIANTS *inv1: cc = on \Rightarrow brake = off*

META *priority(cc) < priority(brake)*

EVENTS

```
...
EVENT pressbrake  $\hat{=}$ 
    BEGIN
        act1 : brake := on
    END
```

```
...
END
```

Critic augmented with priority information

critic (priority action speculation)

INPUTS:

PO_SET POs

MODEL_SET $\{M\}$

PRECONDITIONS:

...

3. $priority(\tau, M) < priority(\sigma, M)$

OUTPUTS:

GUIDE $add_action(sub2act(\tau), E, M)$

1. there exists an unproven invariant PO (implication)
2. there exists a variable update that does not interfere with the existing actions and which makes the PO provable
3. update does not violate priority heuristics

Precondition 3 would fail for a priority guard critic

Toy problems

- ▶ POs often convoluted by a lot of non-relevant details
- ▶ Extend critics to generate a **toy conjecture**
 - ▶ proof of toy can be reused to prove real PO
 - ▶ i.e. has to within same family
 - ▶ must fail/succeed in the same way
 - ▶ must be simpler (abstracts unnecessary details)
- ▶ A **toy model** simplifies a model
 - ▶ help identifying mistakes/issues in models
- ▶ Idea follows from discussions with J Moore about the **ACL2 method**.

Summary

- ▶ We have discussed recycling of proof through abstraction
 - ▶ (hopefully) more robust to changes
- ▶ and how failures can be used to guide modelling
 - ▶ we call this technique **reasoned modelling**
- ▶ Many open questions like
 - ▶ What are the key ideas/steps
 - ▶ Which information is required (the why or the how)
 - ▶ Impact of recent advances within SMT
- ▶ Our approach is iterative and empirical
 - ▶ We are interested in new examples with similar observations