Experiments with Reproof Plans for Infraction

Jane Hesketh
jthesketh@pobox.com

Abstract

We describe an automated reproof generator for upbraiding effectiveness (ARGUE)[1] [2] . When reproving infractions, effectiveness is improved by comprehensive, well-organised coverage of the arguments. The search for a forceful well-expressed reproof is controlled by reproof plans, which are descriptions of the high level structure of reproofs. We have extended the concept of proof plans to the structure of reproof plans. Search control is provided by a fault lattice.

1 Introduction

Consider the following two reproofs:

A *You're late.*
Your dinner is in the dog.

B *You're late.*
You said you'd be home by seven.
You knew I was making us a special meal.
You didn't even ring to say you were delayed.
You're always late.
You take me for granted.
And another thing - you promised you'd deliver my mother's present, but you didn't.
You're unreliable.
You're self-centred.
Your dinner is in the dog.
I've gone out with the boys

Reproof A is a plain assertion of the complaint followed by a consequence. While succinct, it does not spell out the nature of the infraction or achieve the maximal potential force.

Reproof B supports the basic complaint with stated facts. As well as strengthening the reproof, the reproof plan generates the subsequent steps making the reproof incisive, unambiguous and comprehensive. The plan includes abstraction of the general nature of the complaint to allow associated tale recounting, further reinforcing the reproof.

2 Reproof Example

The example below explains the structure of a full reproof such as B. Infraction reproofs have a similar structure to an induction proof, comprising a base case and step cases. The base case provides a limited reproof. The step cases add explication and reinforcement. Presentation order emerges from the reproof plan, as indicated by the numbering.

2.1 Base Case

For dealing with infraction, it is necessary to have a well-founded ordering, for only clear knowledge of the infraction justifies reproof. For this we need the complaint's specific details both explicit and implicit. These specifics are axioms, and they provide the basis of the well-founded ordering. Each basic complaint must be supported by at least one explicit contract infraction, in our example:

You promised you wouldn't drink so much at the party on Pop's boat. (2)

Additional strength is gained by stating any further implicit contracts in force, here:

You knew how much you embarrassed me last time. (3)

[1] The research reported in this paper was supported by empirical studies conducted with the author's father.
[2] We are grateful for feedback on this paper from Phil Odor and members of the Silent Treatment, Outburst Or Stramash: Harnessing Invective Efficiently (STOOSHIE) project.

As is often the case, this complaint has a subsidiary strand:

Your costume was pathetic - carrier bags tied on with string - what was that supposed to be? (4)

2.2 Step Cases

We can step up the power of a basic complaint by up to four steps in the two dimensions: time and reproof domain in infraction space. Any number of these may be used and each more than once, increasing the strength of the reproof, as it approaches the limit in the calculus of complaint as described below.

Time: Past
Alluding to similar past misdemeanours, especially recent ones, increases the force of the current reproof. They are lemmas, so it is not necessary to detail their reproofs provided they could be produced if required. As usual, there is a library of lemmas, and the relevant ones can be referred to by name, such as *lamppost* or *freezer*, if required. References to past misdemeanours are a *non-isolation* tactic, they preemptively refute possible counter-claims that this event is an singleton.

You always drink too much. (5)

Conventionally, such expressions over all time require at least three instances to be strong enough to warrant the reproof. The complement of the *always* operator, with similar semantics, is *never*.

Infraction Space: Longitudinal
Infraction space (see Figure 1) is the lattice of types of fault, at least one of which will be that of the current fault. We take the relevant fragment here from [Odor, 2004].

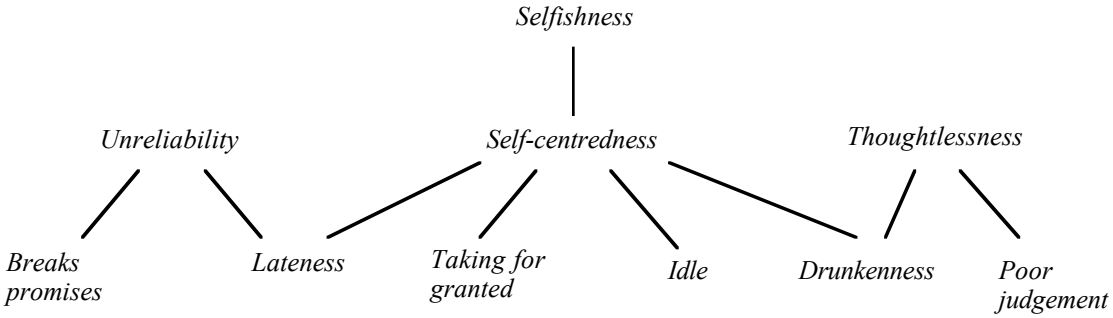

Figure 1: Fault Lattice Fragment

The lattice identifies the fault type, *Drunkenness*, of the base complaint, and that of the subsidiary base complaint *Idleness*:

You were drunk. (1)
You're idle. (6)

By looking upwards in the lattice, we can see that the *Drunkenness* is a subtype of two more abstract faults, *Thoughtlessness and Self-Centredness*:

You're thoughtless. (7)
You're self-centred. (8)

Reference to the abstract type is encapsulates the complaint neatly. We call this *wrapping*, and it is achieved using the longitudinal strategy. It also allows the retrieval of multiple types of fault inherent in the instance.

Infraction Space: Transverse
Access to the infraction space also allows the system to look down the lattice from the infraction type reached longitudinally, to find other subtypes of infraction for which there may be pertinent instances, for which *tale recounting* is then applicable. These are often announced by phrases such as "And another thing" and "Just like". In this case, *Thoughtlessness* has other instances:

Just like when you took the budgie skydiving. (9)

Transverse search through the lattice allows the inclusion of separate misdemeanours. It has a similar strengthening

effect to that achieved by the multiple instances when searching in past time.

Time: Future
The future consists of the consequences of the event.

You were blown off the boat when the wind caught the carrier bags. (10)
Your drunken thrashing filled the bags with water, so you weighed tons. (11)
We had to call the coastguard to rescue you. (12)

3 Reproof Logic

We have implemented our system in the Complaint Reasoning And Barracking (CRAB) system, written in Prolong [Clockhim and Meldrew, 1981]. CRAB is an interactive reproof planning system for intuitive gripe theory, a typed lattice logic. It is an extension of the proof planner described in [Bundy *et al*., 1990]. Since we need constructive logic for reproof synthesis, but still need proof by contradiction for excuse handling, we chose not to use a constructive logic, but to restrict use of the logic to the constructive rule subset for the reproof generation phase, and permit use of the full logic only for the excuse handling module.

In intuitive gripe theory, a proposition has multiple interpretations, and we make use of all three:

- *a* is an example of fault *A*
- *a* is a reproof for fault *A*
- *a* is a reprogram for fault *A*, (with excoriant term *a*)

Understanding the full nature of a fault should help individuals to reprogram their behaviour to avoid similar infractions. The excoriant *a* is the reprogram that for all misdemeanours there exists a reproof of that misdemeanour. A typical intuitive gripe proposition is: *You always pretend to be something you're not*.

Like proofs, gripe theory reproofs are composed of sub-reproofs, consisting either of facts or reproofs built from inferences based on facts. However, unlike a proof which is true or false, a reproof has a force ranging between 0 and 1. By using multiple reproofs on the same topic, their forces are combined according to a complaint calculus, for details see [Cross, 1987]. A direct consequence is that the selection of sub-reproofs for a reproof affects its force. Search control is therefore essential to achieve optimal force.

4 Reproof Plans

The main aspect of reproof planning is search for a reproof consisting of necessary and sufficient reproof points. While order of presentation is important, it is also straightforward as we have seen in the examples above.

4.1 Search for a Reproof

Too little reproof may fail to be comprehensive and clear, but too much can result in the argument getting lost in the details. The force of the reproof being assembled is used to assist CRAB to gauge where to expand search and where to curtail it.

The overall plan always has a base case and the *longitudinal* and *time:future* step cases. Reproof force is calculated acording to the complaint calculus, and reproof points chosen from those available to maximise the force of each branch.

The *time:past* and *transverse* step cases are incorporated if possible. This depends on the force, α, of the partial reproof, compared to the maximum effective force, β. If $\alpha < \beta$, *time:past* and *transverse* step cases are added to the reproof provided the resulting increased reproof force does not exceed β. This has the effect of limiting digression over too many other examples of the subject's infractions, and making the optimal choice from those available. In this example it leads CRAB to omit *that time you knitted our Christmas cards* as having less force than (9) but using both would take the reproof force above β.

5 Discussion and Further Work

We plan to continue testing and developing the fault lattice. Careful study of logic indicates avoiding the *You're just like your {mother | father}* tactic as having no additional value except to raise the temperature of debate. The *Just like a man*, and *My mother warned me* tactics fall into the same category. However, this has encountered some resistance from experimental subjects, so to enable this we intend to introduce user-selectable lattice labelling. The calculus of complaint will also be developed to permit user-defined weightings on the fault lattice to allow users to influence the relative subjective significance of factors.

Incorporating participant roles: partner, teenager, parent, child, boss and subordinate into the model will appear in a later phase. Management of discursive styles such as self-pity or ranting will be a future feature.

Modelling of the subsequent stages of the argument is underway. The next phase of reproof is excuse handling.

Anticipating the *Oh no I didn't, oh yes you did* tactic, we chose not to use an underlying constructive logic as contradiction would be required. A module on *Changing the Subject* detection is being investigated.

6 Results and Conclusions

We have described a novel approach to the generation of reproof plans which are comprehensive and detailed. Search is controlled using a fault lattice and weighted reproof tree.

The approach has been successfully tested on the generation of the following reproofs:

- You forgot my birthday
- This place is a tip
- You've got us lost

In the heat of an argument logic can be an early casualty. Our system is a significant contribution to this problem.

References

[Bundy *et al*., 1990] A. Bundy, F. van Harmelen, J. Hesketh and A. Smaill. Experiments with proof plans for induction. *Journal of Automated Reasoning* 1990.

[Clockhim and Meldrew, 1981] W. Clockhim and V. Meldrew. *Reprogramming with Prolong: A Reproof Language with Objection-Oriented Extensions.* Winter Verlag 1981.

[Cross, 1987] I.M. Cross. *Complaint Calculus.* Apathetic Press 1987.

[Odor, 2004] J.P. Odor. *Fault Lattices: Unstable Fault Lines Across Relationships*. MegaDodo Publications 2004.