CONCURRENT DEDUCTION: CLASSICAL AND MODAL

Jim Cunningham

Department of Computing, Imperial College,
London SW7 2BZ, UK

Abstract

We provide an informal report of work seeking to introduce concurrency into deduction by exploiting modularity. The work arose in the context of a theorem prover for an applied modal action logic, has led to the re-discovery and generalisation of earlier work by Nelson and Oppen and promises a fibered approach to deduction in multi-modal representations of rationality.

1. Background

An automated tableau theorem prover was developed for the UK Forest project first order *modal action* logic in order to demonstrate industrial applications in validating specifictions rather than mathematically interesting theorems. It has been used to prove properties safety-critical specifications for systems of non-trivial size, without being adequate for systems of full industrial scale (see, for example Atkinson and Cunningham [1991]).

While the full battery of mechanised deduction methods with sorting, theory unification etc., would undoubtedly have provided further enhancement for this system, there seemed to be an underlying need for a "divide and rule" approach which will reduce deep problems into small shallow parts. Motivated by this we re-explored the salient work of Nelson and Oppen [1979] on co-operating decision processes. As a consequence we were able to discover, in turn, a simple new procedure for co-operating classical tableaux, and on further analysis, a comparable procedure for concurrent action tableau which had previously been elusive.

Our interests in mechanising a richer class of multi-modal logics let to the creation of a European Esprit project (Medlar). While we report elsewhere on the work of the Medlar project (see, for example Cunningham, Gabbay and Ohlbach 1991), we indicate in the final section of this paper the possible development and application of a fibered form of concurrency.

2. Communication between Derivation Processes

We address the following question: *what has to be communicated when (semi-) decision procedures are used co-operatively to infer joint consequences*. For a full answer the detailed processes of derivation must be analysed, but some pre-requisites for success can be derived at a more abstract level. Let A and B be distinct theories in some logic L and call their common extension $A \cup B$ the joint theory. For example, A might be the theory of finite lists with membership and B a simple arithmetic. Suitably formulated, the joint theory could be lists with arithmetic elements.

Suppose we wish to prove some goal G in the joint theory, but only have the deductive machinery for deriving goals in theories A and B separately. In this case we may wish to separate the goal, so that $G = G_A \cup G_B$, and derive one subgoal G_A in a subproof using the machinery for theory A and the other subgoal G_B in a subproof using the machinery for theory B. By what means can we (i) *separate* the goals, (ii) under what conditions is it *sufficient* to prove the sub-goals, and (iii) if it is not necessary to prove the subgoals in their separate theories is there a *complete* procedure to infer G from the successful derivation of the subgoals?

In other words, we seek an exploitable relation between derivation in the combined theory (a) below, and derivations in the separate theories (b):

(a) $A \cup B \vdash G$ and (b) $A \vdash G_A$ & $B \vdash G_B$

(i) First let us address the means of separation. Nelson and Oppen have observed and exploited the separation of joint derivations in a quantifier-free theory in classicallogic with identity. They separate the constituent theories by the introduction of free term variables, leaving the equality symbol and the free term variables as the remaining common parameters. A similar technique for separation of theories can be achieved by the introduction of free propositional variables in classical propositional logic. This is also used in the final

example, where certain application modules and a richer syntax are presumed. But if there are no preconceived sub-theories we can subdivide the extra-logical symbols in a way which is convenient for the proof. This is the method in the simple examples of sections 3 and 4.

(ii) It can be observed that (a) above follows from (b) in classical logic by monotonicity[1]. So if G_A and G_B can be derived we can infer G if the logic L is monotonic. It follows too that if the goal G cannot be derived from the joint theory, then one of the subgoals cannnot be derived from its theory. These are indeed conditions for soundly infering, or refuting, the joint derivation. But to reduce the joint derivation to derivations of the subgoals we need complete criteria too.

(iii) So now let us suppose $A \cup B \vdash G$, i.e. $A \cup B \vdash G_A \cup G_B$. Provided the extension is conservative it follows that $A \vdash (B \vdash G_A \cup G_B)$ and $B \vdash (A \vdash G_A \cup G_B)$. Then, by weakening, $A \vdash (B \vdash G_B)$ and $B \vdash (A \vdash G_A)$

This just says that in the context of A, from B we can deduce G_B and vice versa. But what does the machinery for $B \vdash G_B$ need from A, everything?

A crucial observation now is that if the Craig interpolation lemma holds for L (in strict form) there is an interpolant Z containing only parameters common to A and B such that $A \vdash Z$ and $Z \vdash (B \vdash G_B)$. Now we have a breakthrough. In the lattice under classical inference there is also a strongest interpolant Z_0, such that $A \vdash Z_0$, $Z_0 \vdash (B \vdash G_B)$, and Z_0 contains only common parameters. (Nelson and Oppen speak of the strongest *residue*). Now if there if there is a complete (semi-) derivation procedure in A, and only a finite set of common parameters, successive monotonic approximations to Z_0, can be communicated to the co-operating derivation process for B

Let us call a logic *modular* if it is monotonic and has a Craig interpolation lemma. We have now outlined a way of obtaining a complete (semi-) decision procedure for derivation of a combined goal for a union of the theories in a modular logic by communication between complete (semi-) decision procedures for the individual theories.

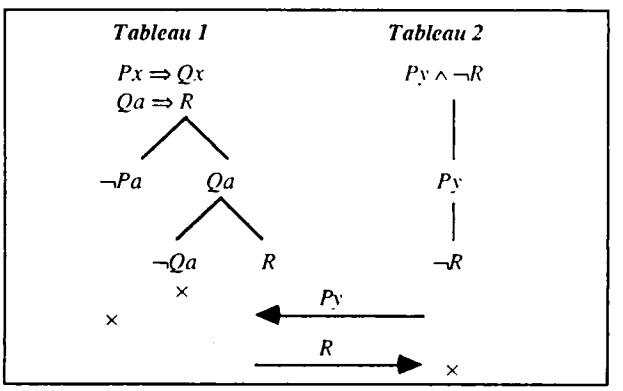
3. Concurrent Classical Tableaux

So far we have presented ideas which are largely contained in Nelson and Oppen's paper, although in different context. As mentioned above, an intermediate adaption is to produce concurrent tableaux for classical predicate logic. Classical logic is of course, modular by our definition (provided it is equipped with sufficient logical symbols, in particular a denotation for falsehood). In example 1 we separate the theorem heuristically, developing one tableau for the antecedent, another for the negation of the conclusion, leaving common parameters.

Example 1. Show that

$$(\forall x.Px \Rightarrow Qx) \wedge (\exists x.Qx \Rightarrow R) \Rightarrow (\exists x.Px \Rightarrow R)$$

Proceed by putting into the premise in Skolem prenex normal form, and use the resulting matrix as the basis for one tableau, and base another tableau on the matrix from the denial of the conclusion, propagating positive residuals in the common parameters P, R, a as illustrated in the figure below. The restriction to positive interpolants is one of a number of possible optimisations. The significance of the residue is clear in context. In this instance the weaker derivations from alternative instantiations of the free variable x are irrelevant for closure.



We make three observations about this example. The *first* is that it is indeed a demonstration of co-operating tableaux and as such, modular deduction in a non-trivial sense. *Secondly*, we have discovered that Smullyan (1968) himself describes a related idea, called *clashing tableaux* , the difference being that Smullyan has cross-checking clash rules for closing the branches of the different tableaux, whereas we have a notion of communication between processes. *Thirdly*, although this is in practice a way of trading search space for communication effort, in the absence of a natural separation into theories from the application it is difficult to exploit arbitary separation because there are "worst cases" where forcing separation on closely coupled formulae forces communication without benefit (e.g, with the unsatisfiable list $p \vee q$, $p \vee \neg q$, $\neg p \vee q$, $\neg p \vee \neg q$).

4. The modal context

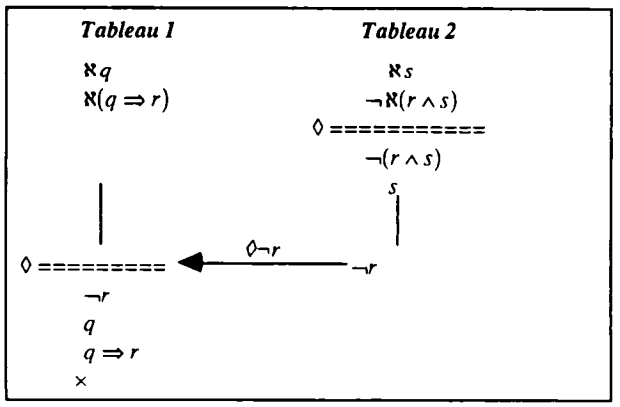
Normal modal logics also have interpolation theorems in the propositional case (see, Fitting 1983). In this case the

[1] Monotonicity is not strictly necessary, as relevant logic and some other

interpolant for inferring a modal formula is also modal. The following example illustrates this in the case of a simple K-tableau where we use the symbol $\aleph$ for necessity.

Example 2.

Show that $\aleph q \wedge \aleph(q \Rightarrow r) \Rightarrow (\aleph s \Rightarrow \aleph(r \wedge s))$ in modal K



While the simple nature of concurrent classical tableaux was unknown to us before, as is pointed out in the earlier report, concurrent deduction with the Forest logic was proving elusive. Progress only came when the logical foundations of section 2 were exposed. It then turned out that only the propositional case of the Forest action logic was modular in its existing form. On working out the interpolants all became clear. We include as an appendix a diagram of division between three modules; *producer*, *buffer*, and *consumer* in a simple application environment. In this system we seek to prove:

$$empty \wedge cont(L) = nil \Rightarrow [put(i)][get]cont(L) \neq nil$$

To accommodate the rules of sharing of the application modules we introduce propositional variables to separate the languages of the joint goal between the three tableaux of the appendix.

5. Fibered Deduction

Given that the form of concurrency we have explored is essentially dependent on modularity, it is attractive to consider applications where multi-modal logics are used to express modular theories of rationality. Modality can be regarded as an object level representation of some meta-level categorisation of logical information. This view finesses arguments about individual operators of propositional attitude (see, for example Cohen and Levesque 1990), since acceptable characteristics for actions, belief, and temporal modalities have been explored in studies of modal logics. (These become details of the modules). We are left to implement the interaction beween modalities by communication between modules, e.g. that a naive robot *believes* what it *senses* in example 3 below.

Example 3. Specifying a Naive robot.

Assume a standard multi-modal logic with normal KD modalities B, W, S, A_a, repectively for representing logically consistent but inconsistent *Beliefs*, *Wants*, *Senses*, and *Actions*, for each parameter a. Let D_a be a predicate parameterised by a, for each action term a, what the agent *Does*.

The joint theory of the robot will presume each of the following as a an axiom schema, where α is a metavariable ranging over propositions:

Immediate belief:	$S\alpha \Rightarrow B\alpha$
Rapaciousness:	$W\alpha \wedge BA_a\alpha \Rightarrow D_a$

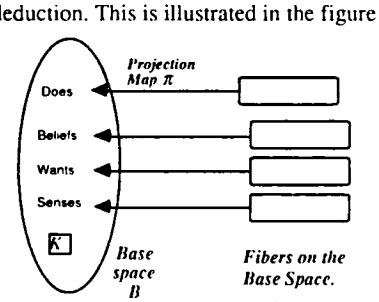
This system is vacuous without some *wants*, and some *beliefs* about *actions*. We can deduce behaviour when there are contingent facts to *sense*, or this behaviour may be realised by symbolic execution. We proceed a little further and consider a car painting robot:

The beliefs of the robot may contain the formula:

$$\forall x.aligned(x) \Leftrightarrow A_{paint}painted(x)$$

Assume the robot can, in effect, instantiate and use modus ponens. If it believes that a particular object c is aligned it can also deduce that it believes $A_{paint}painted(c)$. Of course, whether it believes c is aligned depends on communication from the sense module, and what it does will depend on what it *wants*, and what it can *plan*, and perhaps what it *remembers*.

We can hope to do better by refining the modalities to allow for concious introspection, distinguishing speech acts from physical action and senses, and by introducing new modalities to represent notions of time, memory, and intent in thinking about actions and plans. We then see this as a sort of fibered deduction system, which we will not elaborate here, but which is a way of tackling problems which comes from topology. The fibered view has the disjoint theories of *beliefs*, *wants* and *senses*, perhaps with agentive affect through a *does* fibre. The interaction axioms are the constraints which bind the system together by forcing communication between the fibres of deduction. This is illustrated in the figure below



To see the way the interaction constraints, K, determine communication between the fibers, consider the robot sensing that a car, c, is unpainted, say $S\,U(c)$. By the immediate belief axiom this forces the belief that c is unpainted. But instead of treating this axiom as a logical constraint in the joint language we could for each car c introduce a propositional variable U_c common to the otherwise disjoint languages of sense and belief, and require in the sense fiber that $S\,U(c) \Rightarrow U_c$ and in the belief fiber that $U_c \Rightarrow B\,U(c)$. Then the proposition U_c is a message which must be communicated from the sense fiber to the belief fiber.

The effect of the rapaciousness axiom on communication is clearer if it is replaced by the more directed form below, where the arrow now indicates information flow between the fibers:

$$W\alpha \Rightarrow (B\neg\alpha \wedge BA_a\alpha) \Rightarrow D_a$$

6. Conclusions

We have sketched a route to concurrent deduction through the exploitation of modularity. Although our early examples are based on tableaux stytems, the actual style of deduction engine used by individual modules is not critical. Much remains unstated in this short abstract. We are working to underpin the foundations, quantify and explore the feasibility of the technique in prototype development and make the resulting method of fibered deduction useful for multi-modal applications. The engineering and knowledge representation applications we have in mind depend in on the use of first order multi-modal logics. This factor alone has driven us back to basics in order to recover the necessary properties.

7. References

Atkinson W., Cunningham J., "Proving Properties of a Safety-Critical System" *IEE Software Engineering Journal*, 6-2 1991, pp41-50

Cohen, P.R, Levesque, H.J., "Persistence, Intention, and Committment", in P.Cohen, J.Morgan, Martha Pollack, (Eds) *Intention and Communication*, MIT Press, 1990

Cunningham J., Gabbay D, Ohlbach H-J, "Towards the Medlar Framework", Proc. Esprit'91 (CEC), 1991

Fitting, M.C. *Proof Methods for Modal and Intuitionistic Logic* D.Reidel 1983

Nelson G., Oppen D.C., "Simplification by Co-operating Decision Procedure" ACM Transactions on Programming Lannguages and Systems, 1- 2, 1979, pp 245-257

Smullyan R.M., *First Order Logic* Springer Verlag 1968

8. Acknowledgement

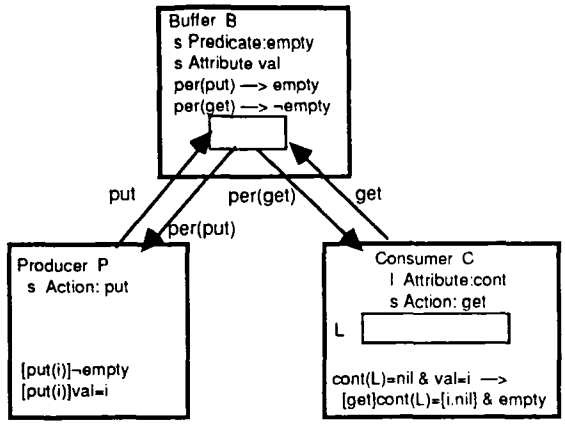
My thanks to Marcello D'Agostino and Jeremy Pitt for their assistance in preparing this paper.

Appendix

Forest-style Producer-Consumer Example

Joint Goal G:

$empty \wedge cont(L) = nil \Rightarrow [put(i)][get]cont(L) \neq nil$



Separated Goals:

G_C		
	$empty$	
		G_P
G_B	$cont(L) = nil$	
$\neg[put(i)]X$	$X = [get]Y$	$Y = (cont(L) \neq nil)$

Concurrent Tableaux

Producer Tableau	***Buffer Tableau***	***Consumer Tableau***
P++	**B++**	**C++**
$\xrightarrow{[put(i)]\neg empty}$		
$\xrightarrow{[put(i)]val=i}$		
	$empty$	
		$cont(L) = nil$
$\neg[put(i)]X$	$X = [get]Y$	$Y = (cont(L) \neq nil)$
$\xrightarrow{\neg[put(i)]X}$		
	$per(put)$	
$\xleftarrow{per(put)}$		
	$put(i)$ ======	
	$\neg empty$	
	$val = i$	
	$\xrightarrow{val=i}$	
	$\neg[get]Y$	
	$\xrightarrow{\neg[get]Y}$	
	$per(get)$	
	$\xrightarrow{per(get)}$	
		get ======
		$cont(L) = [i.nil]$
		$empty$
$(\neg\neg)cont(L) = nil$		
		$\times$

Concurrent tableaux for Forest-style producer-consumer