Appendix (continued)

Separated Goals:

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

Concurrent Tableaux

Producer Tableau	***Buffer Tableau***	***Consumer Tableau***
P++	**B++**	**C++**
[put(i)]¬empty →		
[put(i)]val=i →		
	empty	
		$cont(L) = nil$
$\neg[put(i)]X$	$X = [get]Y$	$Y = (cont(L) \neq nil)$
¬[put(i)]X →		
	$per(put)$	
← *per(put)*		
	$put(i)$ ======	
	¬empty	
	$val = i$	
	val=i →	
	$\neg[get]Y$	
	¬[get]Y →	
	$per(get)$	
	per(get) →	
		get =======
		$cont(L) = [i.nil]$
		empty
		$(\neg\neg)cont(L) = nil$
		×

Concurrent tableaux for the Forest-style producer-consumer

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

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. Further development

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 method of concurrency useful for real applications. The engineering and knowledge representation applications we have in mind depend on the use of first order multi-modal logics. The conditions for modularity have driven us back to basics in order to recover the necessary properties.

6. References

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

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

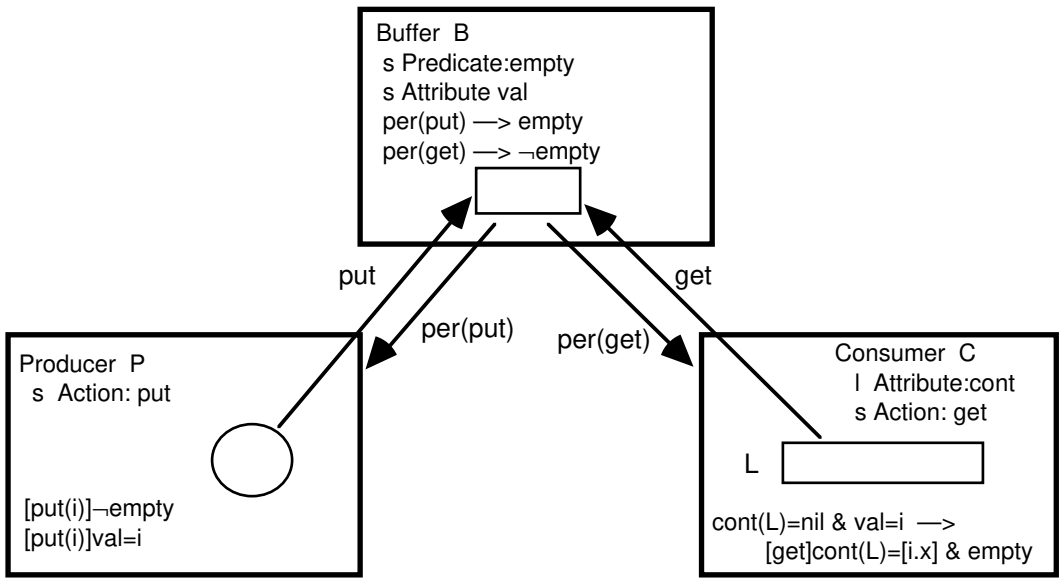
Nelson G., Oppen D.C., "Simplification by Co-operating Decision Procedure" ACM Transactions on Programming Lannguages and Systems, **1**- 2, 1979, pp 245-257

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

7. Acknowledgement

My thanks to Marcello D'Agostino, Jeremy Pitt and Stuart Kent for their assistance in preparing this and the earlier report.

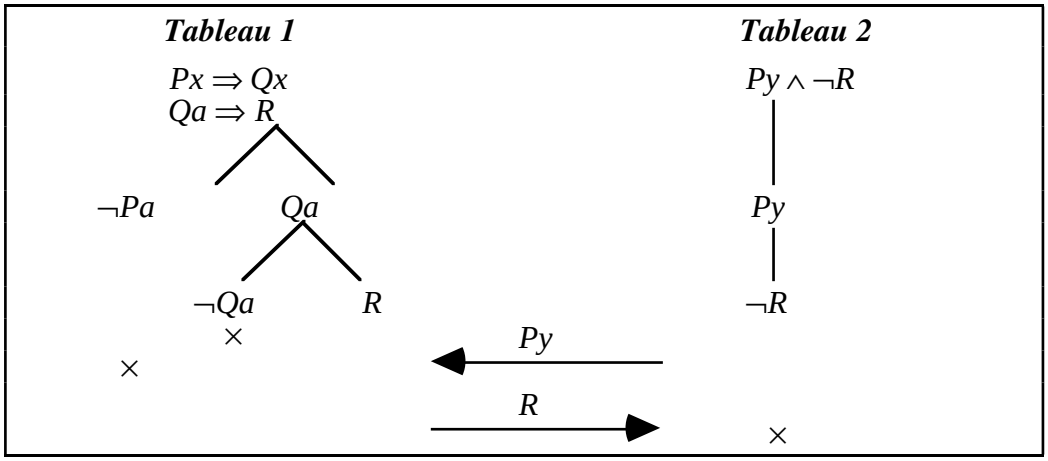
Appendix



Forest-style Producer-Consumer Example

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

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.

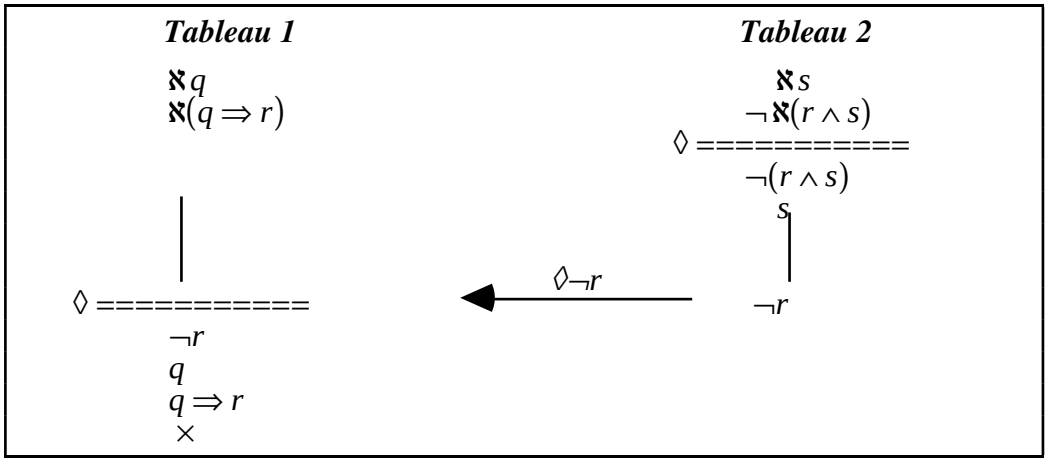


We make three observations about this example. The *first* is that it is indeed a demonstration of co-operating tableaux and as such, modular deduction in a non-trivial sense. *Secondly*, we have discovered that Smullyan (1968) himself describes a related idea, called *clashing tableaux* , the difference 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*, as can readily be verified, there is no restriction to just two co-operating tableaux, provided communications are propagated appropriately.

4. The modal context

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

logic 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[2]. 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 the logic *modular* if it is monotonic and has a Craig interpolation lemma. We have now outlined the argument for a conjecture:

Conjecture: If there are complete (semi-) decision procedures for individual theories in a modular logic then a complete procedure for derivation of a combined goal in the union of the theories can be obtained by communication between (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 presented with different context. As mentioned above, an intermediate adaption for our purpose is to produce concurrent tableaux for classical predicate logic. Classical logic is of course, modular by our definition (provided it is equiped with sufficient logical symbols, in particular a denotation for falsehood). In this case we separate a theorem on heuristic grounds, developing one tableau for the antecedent of a implication, 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

[2]Monotonicity is not strictly necessary, as relevant logic and some other sub-classical logics have properties sufficient with the corresponding interpretation of &.

CONCURRENT TABLEAUX

Jim Cunningham
Department of Computing
Imperial College

Extended Abstract

In this paper we provide an informal report of previously unpublished work by the author which was done as part of an investigation into modular deduction with action logics[1]. Initial results are now being pursued within the ambit of the Esprit Medlar project, with a more complete paper intended.

We report the work under five headings, *background*, *communication, between derivation processes*, *concurrent classical tableau*, *the modal case*, *& further development*:, providing a more applied example as an appendix.

1. Background

The automated tableau theorem prover for the UK Forest project action logic was developed to demonstrate industrial applications rather than mathematically interesting theorems. It has been used to prove properties as part of the process of validating 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 provide further enhancement, there is 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.

2. Communication between Derivation Processes

We need to know 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):

$$\text{(a) } A \cup B \vdash G \quad \text{and} \quad \text{(b) } A \vdash G_A \;\&\; B \vdash G_B$$

(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 classical

---

[1] *Modular Deduction* , UK Forest Research Project deliverable 3.3 (14pp)