Electronic News Journal on Reasoning about Actions and Change 51
Vol. 3: *51-65*, 30.4.1999
http://www.etaij.org/rac/notes/1999/02/

Simulation, Ramification, and Linear Logic

Graham White
Queen Mary and Westfield College, London, UK

Abstract

This article first argues that formalisations of the "frame problem" should have certain desirable logical features; it then proposes a treatment of the frame problem, using linear logic together with modal operators, which fulfils these *desiderata* and seems to be successful in other respects.

"*In logic, there are no morals. Everybody is at liberty to build up his own logic, i.e. his own form of language, as he wishes. All that is required of him is that, if he wishes to discuss it, he must state his methods clearly and give syntactical rules instead of philosophical arguments.*

The tolerant attitude here suggested is, as far as special mathematical calculi are concerned, the attitude which is tacitly shared by the majority of mathematicians." Carnap [2, §17 p. 52].

1 Desiderata

This article will be partly methodological and partly substantive. The methodological part will argue that we should pay more attention to the logical form of the theories that we use in reasoning about the frame problem. I shall be systematically asking the question of what sort of mathematical structure we are given as an alleged solution of the frame problem; and I shall be attempting to find a good answer. There has been surprisingly little investigation of this question, and, as I hope to show, it deserves investigation, simply because of the utility of the answers one gets.

This area is, in general, surprisingly unclear. There is a great deal of talk about how one might solve the frame problem, but most of it seems to be described in terms of the *process* of theorising – [8] is fairly typical of these descriptions. One first produces several items, some of which describe the actions and some of which describe the domain. Those which describe the actions are generally called "action postconditions" or "effect axioms"; the domain is described by "state constraints"; these two should, as far as possible, be separate (see [14, p. 11]). Having assembled this collection of axioms, one then produces frame axioms from these by some appropriate procedure. This collection of axioms, in some appropriate logic, is then supposed to solve the problem at hand.

What we have, then, is merely the description of a process of axiomatisation: we are not given an independent account of the result of the process. What one can gain from this description rather modest: it is simply the idea that our logical theories ought to have two components. The distinction between these components is epistemological, rather than logical: one component – call it the *immediate component* – should be derived from the immediate insights that we have about the situation and the actions, and it is this which becomes the action postconditions, the effect axioms, and the domain constraints. The other component – the frame axioms – should be derived from the first component by some purely formal procedure; one is not supposed to add extra insights when deriving the frame axioms.

The notion of *locality* also plays an important role in this epistemological description. Sandewall formulates the basic insight thus:

> Consider applications where every scenario describes a number of separate but interconnected objects, different scenarios involve different configurations, and each action has its immediate effects on one or a few of the objects, but indirect effects on objects which are connected to the first ones, in some sense of the word 'connected'. Then, it would be completely unreasonable to let the action laws contain different cases which enumerate the possible configurations. Instead, action laws should only specify the "immediate" or "primary" effects of the action, and logical inference should be used for tracing how some changes "cause" other changes across the structure of the configuration at hand. [14, p. 11]

We have, then, some sort of basic insight into what objects are immediately connected to other objects, into what events are immediately caused by other events; and it is *only* this knowledge of local connections which should go into our immediate description of the situation.

Now let us consider the logical form of these two components. The immediate component should give *sufficient* conditions for change: if some fluent is an immediate effect of some action, then it is an immediate effect regardless of what else goes on elsewhere. If locality amounts to anything, it should amount to that sort of non-interference. Consequently, if we add new action types (with their own effect axioms), the existing effect axioms, and domain rules, should still be valid.

So the immediate axioms will have Σ_1 antecedents; that is, their antecedents should be existential quantifications of unquantified sentences. For example, we will have effect axioms like

$$\mathsf{move}(a,b,l) \quad \to \quad \mathsf{at}(b,l)$$

which say that, if an actor a moves a block bl to a location l, then the block ends up at l. This has free variables, which disguise an existential quantification (over a) in the antecedent.

The frame axioms, on the other hand, give necessary conditions for change, and so, when expressed as clauses, they will have antecedents which are Π_1 or worse. If, in the above example, the only actions which can change the

	Effect Axioms	Frame Axioms
Role	Sufficient for Change	Necessary For Change
Character of Data	Local	Global
Type of Cause	Individual	All Causes
New causes?	Remain Valid	Broken
Logical Form of Antecedent	$\Sigma_1 : \exists x. P(x)$	$\Pi_1 : \forall x.P(x)$ (or worse)
Logic Programming	Horn Clauses	Clarke's Completion

Table 1: Effect Axioms and Frame Axioms

position of a block are moving it, or moving a block that it is on, then the frame axioms will have to have at least the consequences that, if the block is not at l to begin with, and if no action of certain sorts have occurred, then the block is not at l in the new situation. So if we formulate this as a clause, it will have to have an antecedent which is a negated existential, i.e. which is Π_1. We should also notice that frame axioms are broken by the addition of new action types. They give *necessary* conditions for change, and so, if we work out the frame axioms for a given set of immediate axioms, and then add new action types, then the frame axioms will generally not remain valid.

We may compare this with the situation in logic programming. Here, we use clauses to define the extension of a predicate; their antecedents give sufficient conditions for terms to be in the extension of a predicate. If we use negation as failure, then we also get necessary conditions; Clark's completion gives a reformulation of these necessary conditions in logical terms.

So, if we gather all of this together, we get the comparison in Table 1. It is, of course, rather idealised, and at this stage we do not know if we can achieve working theories which have this logical form. Nevertheless it is worth aiming at; it would give us theories which were very elaboration tolerant (because adding new action types would not break the effect axioms, and the frame axioms are produced automatically).

2 Minimisation

2.1 The Procedure

We use Lifschitz' description of the circumscription procedure [8].

1. Select appropriate fluents for the situation.
2. Secondly, "we need to describe first when a combination of values of the frame fluents is 'consistent', that is, attained by some situation".
3. Next we introduce actions and their postconditions.

4. There is also the general machinery of the situation calculus in particular the generic law of inertia

$$\neg\mathsf{noninertial}(f, a, s) \quad \rightarrow \quad [\mathsf{holds}(f, \mathsf{result}(a, s)) \leftrightarrow \mathsf{holds}(f, s)]$$

(this is a somewhat simplified version of [8, 5.14, p. 333]).

5. Finally we compute the resulting state by circumscription: we circumscribe the extent of the noninertial predicate while holding fixed the postconditions of the actions and obeying the constraints.

2.1.1 What are Fluents?

We should, in passing, ask a question: what are fluents? On the one hand, they *look* very propositional – they are such things as $\mathsf{at}(b_1, l_2)$ – and, given a fluent ϕ and a situation s, we can obtain a truth-value by means of the $\mathsf{holds}(\cdot, \cdot)$ predicate. So they might, conceivably, be families of propositions parametrised by situations (and, indeed, they are called "propositional fluents": [8, p. 328]).

However, the propositional appearance is deceptive: they are, as Lifschitz remarks, "terms and not formulae" [8, p. 328]. We cannot, for example, apply truth-functional connectives to them: we can say $\neg\mathsf{holds}(\mathsf{at}(b_1, l_2), s)$, but not $\mathsf{holds}(\neg\mathsf{at}(b_1, l_2), s)$, and so on.[1] And there is a subclass of the fluents – the "frame fluents" – to which the circumscription procedure applies.

So fluents have two sides: extensional and intensional. Extensionally, they can be regarded as assignments of truth values to situations, and, regarded in this way, there would be no reason why we should not form arbitrary truth-functional combinations of fluents. Intensionally, however, fluents are terms: we cannot necessarily form arbitrary truth-functional combinations of them.

2.2 The Example

Let us try this on an example.
Example 1. There are two balls, b_1 and b_2, connected by a string of length 1. There are three locations, l_0 which is on a table, l_1 at height 1 above l_0, and l_2 at height 2 above l_0. Initially both balls are at l_0 with the string slack: at time 1, b_1 is raised to l_2, thus:

Before	**After**
b_2 at l_0	b_1 at l_2
b_1 at l_0	b_2 at l_1
string slack	sling taut

The minimisation procedure would be applied as follows.

- We start with a suitable language, $\mathcal{L}$, which contains constants $b_i(i = 1, 2)$ (the balls), $l_j(i = 0, 1, 2)$ (the locations), and and s (the string). It will also have predicates $\mathsf{at}(\cdot, \cdot)$, $\mathsf{slack}(\cdot)$, and $\mathsf{taut}(\cdot)$.

[1] We may, of course, define what Shanahan calls "compound fluents" [15, pp. 115f.], and we can define recursive conditions for $\mathsf{holds}(f, s)$ to be true when f is such a fluent. However, these fluents play no role in the minimisation procedure; and it is this role in the minimisation procedure that we are concerned with.

$$
\begin{aligned}
\mathsf{holds}(\mathsf{at}(b_2, l_0), \sigma) &\leftrightarrow (\mathsf{holds}(\mathsf{slack}(s), \sigma) \wedge \mathsf{holds}(\mathsf{at}(b_1, l_0), \sigma)) \\
&\quad \vee(\mathsf{holds}(\mathsf{taut}(s), \sigma) \wedge \mathsf{holds}(\mathsf{at}(b_1, l_1), \sigma)) \\
\mathsf{holds}(\mathsf{at}(b_2, l_1), \sigma) &\leftrightarrow (\mathsf{holds}(\mathsf{taut}(s), \sigma) \wedge \mathsf{holds}(\mathsf{at}(b_1, l_0), \sigma)) \\
&\quad \vee(\mathsf{holds}(\mathsf{slack}(s), \sigma) \wedge \mathsf{holds}(\mathsf{at}(b_1, l_1), \sigma)) \\
&\quad \vee(\mathsf{holds}(\mathsf{taut}(s), \sigma) \wedge \mathsf{holds}(\mathsf{at}(b_1, l_2), \sigma)) \\
\mathsf{holds}(\mathsf{at}(b_2, l_2), \sigma) &\leftrightarrow (\mathsf{holds}(\mathsf{taut}(s), \sigma) \wedge \mathsf{holds}(\mathsf{at}(b_1, l_1))) \\
&\quad \vee(\mathsf{holds}(\mathsf{slack}(s), \sigma) \wedge \mathsf{holds}(\mathsf{at}(b_1, l_2), \sigma)) \\
&\quad \vee(\mathsf{holds}(\mathsf{taut}(s), \sigma) \wedge \mathsf{holds}(\mathsf{at}(b_1, l_3), \sigma)) \\
\mathsf{holds}(\mathsf{at}(b, l), \sigma) \wedge & \\
\mathsf{holds}(\mathsf{at}(b, l'), \sigma) &\rightarrow l = l' \\
\mathsf{holds}(\mathsf{slack}(s), \sigma) &\leftrightarrow \neg\mathsf{holds}(\mathsf{taut}(s), \sigma)
\end{aligned}
$$

Table 2: Theory $\mathcal{T}$

- The fluents will will be those of the form $\mathsf{at}(b, l)$, where b is a block and l is a location, and two fluents describing the condition of the string: $\mathsf{taut}(s)$ and $\mathsf{slack}(s)$.
- The state constraints will be given by the *local* constraints of the situation, namely those given in Table 2. Call this *Theory* $\mathcal{T}$.
- Circumscription gives the wrong solution, namely:

Before	**After**
b_2 at l_0	b_1 at l_2
b_1 at l_0	b_1 at l_2
string slack	sling slack

  This is because, in any solution satisfying the state constraints, the position of the balls must change; so the position of the balls makes no difference, one way or another, to the circumscription. However, we can still circumscribe the $\mathsf{noninertial}(\mathsf{slack}(s))$; if we do this, we force the string to remain slack.

2.3 Fixing the State Constraints

We might fix up the state constraint, so that it works, with something like this:

> We should add to this extra constraints of the form: A ball B at height H is either on the table ($H = 0$), or held at height H, or hanging on a taut string S from a ball $B2$ at height $H2 > H$.[2]

This works in this case. You could justify it in two ways:

[2] This was a referee's comment on [28].

1. ‘support’ is a natural concept, and we all have it, so we should use it, or
2. we should try to find an analogous concept in all problems of a suitable sort.

We can deal with each of these justifications in turn.

Justification 1: closely analogous problems can't be rescued by the concept of support. For example, we could consider
Example 2. As with Example 1, but suppose that there are magnets at each of the locations, weak enough so that pulling on a string will nevertheless move a ball, but strong enough so that if a ball gets to be at a location it is happy to remain there.
Example 3. As with Example 1, but now suppose that the locations are spread out horizontally, on a suitably rough surface. Here everything is supported all of the time.

Notice that in both of these variants the constraints that I've given *do* yield exactly what Lifschitz says they should – they “describe … when a combination of values of the frame fluents is ‘consistent’, that is, attained by some situation”; any situation satisfying the constraints is attained by some physically possible, stable situation.

So we cannot, universally, use the concept of support.

Justification 2: The alternative justification would say that we should find an analogous concept in all such problems. So how *do* we find such a concept?

Notice that the concept is not local: it is not given by talk of individual components of the situation and their individual interactions with one another. We have to know *all* of the reasons for a ball remaining in the same place.

To see this, notice, firstly, that this axiom is broken by the addition of extra effect axioms giving different reasons for balls remaining fixed or moving – as with Example 2 where we had magnets at each of the locations.

Secondly, notice that we can generalise the problem to that of a set of balls, connected by strings of varying lengths. Here a string will become taut if it is on the shortest path from the lifted ball to one of its ends; and being on the shortest path is not a local concept, that is, we cannot decide whether it holds or not simply by knowing about the string and what it is connected to.

Now finding such a concept – that is, a concept which takes into account all of the causes affecting a particular fluent – is precisely what circumscription is supposed to do. So what the suggested addition does is to add, to the state constraints, something looking very like a frame axiom. And the fact that, in some circumstances, people can naturally come up with concepts like support surely means that frame axioms are quite natural things. The problem, however, is this: the recommended minimisation procedure, which was supposed to come up with the state constraints, did not work.

$\mathcal{L}$	$\mathcal{L}'$
$\mathsf{holds}(\mathsf{at}(b_1, l_0), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_1, l_0), \sigma)$
$\mathsf{holds}(\mathsf{at}(b_1, l_0), \sigma) \vee \mathsf{holds}(\mathsf{at}(b_1, l_1), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_1, l_1), \sigma)$
$\mathsf{holds}(\mathsf{at}(b_1, l_0), \sigma) \vee \mathsf{holds}(\mathsf{at}(b_1, l_1), \sigma)$ $\vee \mathsf{holds}(\mathsf{at}(b_1, l_2), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_1, l_2), \sigma)$
$\vdots$	$\vdots$
$\mathsf{holds}(\mathsf{at}(b_2, l_0), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_2, l_0), \sigma)$
$\mathsf{holds}(\mathsf{at}(b_2, l_0), \sigma) \vee \mathsf{holds}(\mathsf{at}(b_2, l_1), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_2, l_1), \sigma)$
$\mathsf{holds}(\mathsf{at}(b_2, l_0), \sigma) \vee \mathsf{holds}(\mathsf{at}(b_2, l_1), \sigma)$ $\vee \mathsf{holds}(\mathsf{at}(b_2, l_2), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_2, l_2), \sigma)$
$\vdots$	$\vdots$
$\mathsf{holds}(\mathsf{at}(b_1, l_0), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_1, l_0), \sigma)$
$\mathsf{holds}(\mathsf{at}(b_1, l_1), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_1, l_1), \sigma)$ $\wedge \neg \mathsf{holds}(\boldsymbol{\beta}(b_1, l_0), \sigma)$
$\mathsf{holds}(\mathsf{at}(b_1, l_2), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_1, l_2), \sigma)$ $\wedge \neg \mathsf{holds}(\boldsymbol{\beta}(b_1, l_1), \sigma)$
$\vdots$	$\vdots$
$\mathsf{holds}(\mathsf{at}(b_2, l_0), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_2, l_0), \sigma)$
$\mathsf{holds}(\mathsf{at}(b_2, l_1), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_2, l_1), \sigma)$ $\wedge \neg \mathsf{holds}(\boldsymbol{\beta}(b_2, l_0), \sigma)$
$\mathsf{holds}(\mathsf{at}(b_2, l_2), \sigma)$	$\mathsf{holds}(\boldsymbol{\beta}(b_2, l_2), \sigma)$ $\wedge \neg \mathsf{holds}(\boldsymbol{\beta}(b_2, l_1), \sigma)$
$\vdots$	$\vdots$

Table 3: Translating between $\mathcal{L}$ and $\mathcal{L}'$

2.4 Intensionality of the Data

Consider also the following. Let us define a new language, $\mathcal{L}'$, which will be much the same as $\mathcal{L}$ except that it will have primitives $\boldsymbol{\beta}(\cdot, \cdot)$ in place of $\mathsf{at}(\cdot, \cdot)$. We can easily define translations in both directions between the two languages: they are given in Table 3, and are clearly mutually inverse. We can use these translations to come up with a theory $\mathcal{T}'$, equivalent in first order logic to Theory $\mathcal{T}$, but expressed in $\mathcal{L}'$. (Intuitively, $\boldsymbol{\beta}(b, l)$ will mean that b is at l or below; but I do not wish to concentrate on the precise meaning of $\boldsymbol{\beta}$, merely on the fact that we can translate from $\mathcal{T}$ into what ought to be an equivalent theory in a different vocabulary.)

Circumscription is somewhat more successful for this theory than for Theory $\mathcal{T}$, because the fluents are different: the position of b_2 is now described in terms of fluents $\boldsymbol{\beta}(b_2, l_0)$, $\boldsymbol{\beta}(b_2, l_1)$, and $\boldsymbol{\beta}(b_2, l_2)$. In the intended model, the last two of these remain true; so circumscription gives us this model as well as the usual unintended model where the string remains slack. We can then use a circumscription policy to get rid of the unintended model.

This shows that the data to which the circumscription procedure are applied are not the obvious extensional ones – namely a theory up to logical equivalence – but contain, in addition, something intensional. The data consist, namely, of a theory together with a set of atoms in its language (the atoms which Lifschitz calls the "frame fluents"). These atoms are involved in the circumscription. We have, then, a basically intensional procedure, used together with a formalism – first order logic and model theory – that is thoroughly extensional; we may expect confusion to follow, and it does.

What is awkward is not the intensionality *per se*. If we read the philosophical literature – particularly [3] – the intensionality comes as no surprise; a good deal of our talk about causality just *is* intensional. The awkwardness comes from not taking the intensionality seriously enough, and expecting the orthodox logical approach, which is very much tied to extensionality, to be able to cope.

We can see the awkwardness if we ask when we have two different presentations of the same mathematical structure. This is a fairly basic requirement: as Quine puts it,

> [w]e have an acceptable notion of class, or physical object, or attribute, or any other sort of object, only insofar as we have an acceptable principle of individuation for that sort of object. There is no entity without identity. ([26, p. 102]; see [6].)

So: what is our principal of individuation for these mathematical structures, that is, for the nonmonotonic theories that are "solutions to the frame problem"?

Now for classical logic we have a good answer to this: two axiom systems, in the same language, count as the same if they define the same consequence relation. (See, for example, Tarski's papers of the thirties: [23, 21, 20].) For non-monotonic logics, too, we have a reasonably well-behaved notion of consequence relation [9]. Lifschitz' remark, that "a classical axiomatic theory can be viewed simply as a set of sentences – its axioms" [8, p. 307] is clearly wrong: it takes merely typographic differences far too seriously. We can have many different sets of axioms that define the same theory.

In classical logic, we also have a good account of how to compare axiom systems in two different languages, using the notion of an interpretation of one language in another; these concepts are very prominent in Carnap's work [2, §61 p. 222f.], and are in line with his emphasis on toleration.

It is the idea of interpretation which goes awry in this case. We might like to think that the language with at and the language with $\boldsymbol{\beta}$ are intertranslatable; they certainly are in the case of classical logic. And this conforms to our intuitions; the choice of different vocabularies means that our theories look different, but they remain intertranslatable. However, in the case of our nonmonotonic logic, two things depend on vocabulary choice; one is the concepts expressed by our language, and these are intertranslatable. The other, however, is the choice of fluents, and this choice is not thus translatable.

So in the case of this nonmonotonic logic, we have not been told when two different sets of data define the same theory: this example shows that the classical answer to this question does not work, but it would seem to be rather difficult to find an alternative. Fluents, as described, seem to be merely typographic entities: they are terms, and we cannot perform the usual logical operations on them without breaking the minimisation procedure. As Galton argues, we need this sort of reification because of operators which are not truth-functional [5, p. 1198].[3]

3 Rewrites

I propose a much more direct solution of this problem: namely simulating it with a rewrite system. This is very similar to a procedure, originated by Hölldobler and Schneeberger [7] and described by Thielscher [24]. We define the following rewrite system. Firstly the rewrites of the system will be applied to situation terms, which are collections of atoms put together with an associative commutative connective $\bullet$.

There are three sorts of rules, firstly the rules which say how the string can change the positions of balls:

$$\mathsf{at}(b, l_i) \bullet \mathsf{at}(b', l_j) \quad \rightsquigarrow \quad \mathsf{at}(b, l_i) \bullet \mathsf{at}(b', l_{i-1}) \qquad i = 1, 2, 3, j < i - 1$$

and secondly the rules which say how the position of balls affects the condition of the string:

$$\begin{array}{rcl} \mathsf{at}(b, l_i) \bullet \mathsf{at}(b', l_j) \bullet \mathsf{slack}(s) & \rightsquigarrow & \mathsf{at}(b, l_i) \bullet \mathsf{at}(b', l_j) \bullet \mathsf{taut}(s) \\ & & i, j = 0, 1, 2, 3, i \neq j \\ \mathsf{at}(b, l_i) \bullet \mathsf{at}(b', l_i) \bullet \mathsf{taut}(s) & \rightsquigarrow & \mathsf{at}(b, l_i) \bullet \mathsf{at}(b', l_i) \bullet \mathsf{slack}(s) \\ & & i = 0, 1, 2, 3 \end{array}$$

and, finally, the rewrite which says what the action does:

$$\begin{array}{rcl} \mathsf{time}(T) \bullet \mathsf{occurs}(\mathsf{lift})(T) \bullet \mathsf{at}(b, l_i) & \rightsquigarrow & \mathsf{time}(T+1) \bullet \mathsf{at}(b2, l_2) \\ & & i = 0, 1 \end{array}$$

Note that these rewrites are all local, and they are pleasingly direct: they say what balls and string do.

Now start with a situation in which we simply have the postconditions of the action, and apply the rewrites until we can't any more:

$$\begin{array}{rl} \mathsf{time}(0) \bullet \mathsf{occurs}(\mathsf{lift})(0) \bullet \mathsf{at}(b, l_0) \bullet \mathsf{at}(b', l_0) \bullet \mathsf{slack}(s) & \rightsquigarrow \\ \mathsf{time}(1) \bullet \mathsf{at}(b, l_2) \bullet \mathsf{at}(b', l_0) \bullet \mathsf{slack}(s) & \rightsquigarrow \\ \mathsf{time}(1) \bullet \mathsf{at}(b, l_2) \bullet \mathsf{at}(b', l_1) \bullet \mathsf{slack}(s) & \rightsquigarrow \\ \mathsf{time}(1) \bullet \mathsf{at}(b, l_2) \bullet \mathsf{at}(b', l_1) \bullet \mathsf{taut}(s) & \end{array}$$

and this does the job; that is, we can prove that any sequence of such rewrites will terminate at the desired situation.

[3] We should note that fluents only seem to have become typographical items *when circumscription was applied to the frame problem.* McCarthy and Hayes originally [12, p. 478] define fluents as functions from situations to propositions; this is a very natural (albeit higher order) definition of fluents. McCarthy's original paper on circumscription [10] discusses object and predicate circumscription, and does not apply circumscription to the frame problem; he does apply circumscription to the blocks world, but applies it to a predicate $\mathsf{prevents}(\cdot, \cdot)$ where the first argument is a fluent and the second is an action. It is only in [11] that McCarthy describes what seems to have become the canonical treatment of the frame problem by means of circumscription: and, instead of "predicate circumscription", we now have "formula circumscription" [11, p. 90].

3.1 Correctness and Tractability

Notice two things about this system. Firstly, it is correct: Thielscher [25] and his colleagues have a large collection of examples (mostly to do with electric circuits) which minimisation gets wrong, but which are handled correctly by such rewrite systems.

Secondly, such systems are very efficient, basically because we can do all the work locally. They are much more tractable than the usual sort of non-monotonic logic.

3.2 Local and Global

Notice that we still have a form of our original distinction between local and global; that is, we can distinguish between parts of the reasoning given by local data and that given by global data. The individual rewrites are given by local data: if we introduce new types of cause, they remain valid. Detecting whether the rewrites terminate, however, is a global matter: if we introduce new types of cause, then a situation which was previously stable might be open to further rewrites.

4 Logic After All

4.1 The Local Theory: Linear Logic

You could argue, though, that this solution has fatal defects precisely because it isn't logic: this, we cannot reason with incomplete information, we can only reason in one direction, and so on.

But yes, it is logic. Here is a (rudimentary) sequent calculus for linear logic.

$$\frac{}{A \vdash A}\,\text{Axiom} \qquad \frac{}{\vdash \mathbf{1}}\,\mathbf{1}\text{R}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\,\otimes\text{L} \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta'}\,\otimes\text{R}$$

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \multimap B \vdash \Delta, \Delta'}\,\multimap\text{L} \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\,\multimap\text{R}$$

Now using this we can deal with rewrites.[4] Look at a typical one. Suppose we have a simple rewrite rule, say $A \rightsquigarrow A'$, and apply it thus:

$$A \bullet B \bullet C \ldots \rightsquigarrow A' \bullet B \bullet C \ldots$$

[4] The corresponding proof search procedure was, in fact, also described by Wolfgang Bibel some time before Girard's paper on linear logic; see [1].

Girard, however, remains as the one who provided a full proof theory for linear logic, and who noticed that this logic had remarkably good metatheoretical properties.

$$\frac{\Gamma,\, A \,\vdash\, \Delta}{\Gamma,\, \Box A \,\vdash\, \Delta}\,\Box\mathrm{L} \qquad \frac{\Gamma,\, \Box A \,\vdash\, B, \Delta}{\Gamma,\, \Box A \,\vdash\, \Box B,\, \Delta}\,\Box\mathrm{R}_1 \qquad \frac{\Gamma \,\vdash\, \Diamond A,\, B,\, \Delta}{\Gamma \,\vdash\, \Diamond A, \Box B,\, \Delta}\,\Box\mathrm{R}_2$$

$$\frac{\Gamma,\, A \,\vdash\, \Diamond B,\, \Delta}{\Gamma,\, \Diamond A \,\vdash\, \Diamond B,\, \Delta}\,\Diamond\mathrm{L}_1 \qquad \frac{\Gamma,\, A,\, \Box B \,\vdash\, \Delta}{\Gamma,\, \Diamond A,\, \Box B \,\vdash\, \Delta}\,\Diamond\mathrm{L}_2 \qquad \frac{\Gamma \,\vdash\, A,\, \Delta}{\Gamma \,\vdash\, \Diamond A,\, \Delta}\,\Diamond\mathrm{R}$$

Table 4: Rules for Variant Linear Modalities

$$\frac{\Gamma, A \,\vdash\, \Delta}{\Gamma, \Box A \,\vdash\, \Delta}\,\Box\mathrm{L} \qquad \frac{\Box\Gamma \,\vdash\, B, \Diamond\Delta}{\Box\Gamma \,\vdash\, \Box B, \Diamond\Delta}\,\Box\mathrm{R}$$

$$\frac{\Box\Gamma, A \,\vdash\, \Diamond\Delta}{\Box\Gamma, \Diamond A \,\vdash\, \Diamond\Delta}\,\Diamond\mathrm{L} \qquad \frac{\Gamma \,\vdash\, A, \Delta}{\Gamma \,\vdash\, \Diamond A, \Delta}\,\Diamond\mathrm{R}$$

Table 5: Sequent Calculus Rules for S4

We can represent this as a linear proof, thus:

$$\dfrac{\dfrac{}{A \,\vdash\, A} \qquad \dfrac{\dfrac{}{A' \,\vdash\, A'} \qquad \dfrac{}{B \otimes C \ldots \,\vdash\, B \otimes C \ldots}}{B \otimes C \ldots, A' \,\vdash\, A' \otimes B \otimes C \otimes D \ldots}}{\dfrac{A, B \otimes C \ldots, A \multimap A' \,\vdash\, A' \otimes B \otimes C \ldots}{A \otimes B \otimes C \ldots, A \multimap A' \,\vdash\, A' \otimes B \otimes C \ldots}}$$

We can extend this to the following.
Proposition 1. *Given a set of ACI rewrite rules, we can thus find a set of linear logic formulae, $\{\alpha_i \mid i \in I\}$, such that*

1. *If we have an application of a rewrite rule, $A \bullet B \bullet \ldots \leadsto A' \bullet B' \bullet \ldots$, then there is an α_i such that the sequent $A \otimes B \otimes \ldots, \alpha_i \vdash A' \otimes B' \ldots$ is provable, and*

2. *Any proof of a sequent of the form $A \otimes B \otimes \ldots, \alpha_i \vdash A' \otimes B' \ldots$ corresponds to an application of a rewrite rule.*

Proof. The first half is elementary; we simply define a suitable α_i for each rewrite rule, as above. For the second half, we need to use cut elimination for linear logic, in order to show that the only proofs of these sequents correspond to rewrites. □

This gives us, then, a translation of the local component of our problem into linear logic. However, we still need to deal with two things: the composition of rewrites into arbitrary sequences, and being able to tell when those sequences have terminated.

4.2 The Global Theory: Linear Modalities

Consider the rules for modal operators given in Table 4. Notice that they are not the same as the normal rules for an S4 modality – that is, the rules in Table 5. The side conditions on the left $\Diamond$-rule and the right $\Box$-rule are different; whereas the normal S4 rules require *all* of the other formulae in these rules to be modalised, our rules only require that *at least one* other formula should be appropriately modalised.

McCarthy and Hayes [12, p. 472] do propose such a reading of modal operators in terms of rewrites, but their modalities are normal Kripkean ones in a classical theory, and they are unable to make a very precise application of the modal logic.

Consider now the relation which holds between A and B if we have $A \vdash \Diamond B$. This looks very like the rewrite relation. Firstly, it is transitive, since if we have $A \vdash \Diamond B$ and $B \vdash \Diamond C$, then we also have $A \vdash \Diamond C$ by

$$\dfrac{\begin{array}{c}\vdots\\ A \vdash \Diamond B\end{array} \qquad \dfrac{\begin{array}{c}\vdots\\ B \vdash \Diamond C\end{array}}{\Diamond B \vdash \Diamond C}}{A \vdash \Diamond C}$$

Secondly, it is stable under $\otimes$: that is, if we have $A \vdash \Diamond B$, we also have $A \otimes C \vdash \Diamond(B \otimes C)$, by cutting with

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B} \qquad \dfrac{}{C \vdash C}}{B,\, C \vdash B \otimes C}}{B,\, C \vdash \Diamond(B \otimes C)}}{\Diamond B,\, C \vdash \Diamond(B \otimes C)}}{(\Diamond B) \otimes C \vdash \Diamond(B \otimes C)}$$

So we need a few technical lemmas for these modalities; detailed proofs are given in [29]. First we have cut elimination:

Proposition 2. *Linear logic, together with the modalities given in Table 4, satisfies cut elimination: any proof can be transformed into a proof without the cut rule.*

Now this is not quite good enough, because in practice we will have some given ramification behaviour, that we are trying to model, and we will want to produce a modality that behaves appropriately. So we have the following.

Definition 1. *Given a set of basic rewrites,*

$$\{A_{i,1} \otimes A_{i,2} \ldots \otimes A_{i,r_i} \rightsquigarrow B_{i,1} \otimes B_{i,2} \ldots \otimes B_{i,s_i} \mid i \in I\},$$

where the $A_{i,j}$ and the $B_{i,j}$ are atoms, then the associated set of modal axioms is

$$\{A_{i,1} \otimes A_{i,2} \ldots \otimes A_{i,r_i} \vdash \Diamond(B_{i,1} \otimes B_{i,2} \ldots \otimes B_{i,s_i}) \mid i \in I\},$$

In [29] we show how to introduce these axioms in a cleaner way – that is, as rules – and we prove cut elimination for the system with these rules.

Finally we need to be able to detect termination. We will do this with an operator $\boldsymbol{\tau}(\cdot)$, which is introduced by the rule

$$\frac{\{\Gamma \vdash A_i,\, \Delta\}_{A \rightsquigarrow A_i}}{\Gamma \vdash \boldsymbol{\tau}(A_i),\, \Delta}$$

where the A_i are all of the states such that $A \rightsquigarrow A_i$. We can (given some additional left rules for $\boldsymbol{\tau}(\cdot)$) also prove cut elimination for the system with this new operator. Cut elimination then implies that, if A represents a state, we can prove $A \vdash \boldsymbol{\tau}(A)$ iff A is terminal under $\rightsquigarrow$.

Notice certain things about this representation of the problem. Firstly, it is elaboration tolerant: the rules governing $\Diamond$ and $\boldsymbol{\tau}$ are generic, and all we change, when we add new interactions to the situation, is the set of modal axioms. We change this simply by adding new members to it.

Secondly, we can change vocabulary easily; in [29] we show that we have a good notion of interpretation for theories of this sort, and that, in our example, we can change from the at vocabulary to the $\boldsymbol{\beta}$ vocabulary without breaking anything.

Thirdly, we have represented our problem as a problem of proof search in a system with good theoretical properties. We can thus reason with incomplete information, we can use the system for explanation as well as for prediction, and so on.

Fourthly, systems like this are easy to implement. There is a well-developed technology for implementing proof search in these systems [13, 27], and thus these results are more than simply a theoretical curiosity.

References

[1] Wolfgang Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.

[2] Rudolf Carnap. *The Logical Syntax of Language.* Kegan Paul, Trench, Trubner and Co., London, 1937. Translated by Amethe Smeaton, Countess von Zeppelin.

[3] Donald Davidson. The logical form of action sentences. In *Essays on Actions and Events*, pages 105–148. Oxford University Press, 1980. Originally published in N. Rescher (ed.), *The Logic of Decision and Action*, University of Pittsburgh Press, 1967.

[4] Dov M. Gabbay, C.J. Hogger, and J.A. Robinson. *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3. Clarendon, 1994. Volume Coordinator D. Nute.

[5] Anthony Galton. Reified temporal theories and how to unreify them. In John Myopoulos and Ray Reiter, editors, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1177–1182. 1991.

[6] Roger Gibson. Quine on matters ontological. *Electronic Journal of Analytic Philosophy*, 1997. Available from `http://www.phil.indiana.edu/ejap/1997.spring/gibson976.html`.

[7] Gerd Große, Steffen Hölldobler, Josef Schneeberger, Ute Sigmund, and Michael Thielscher. Equational Logic Programming, Actions, and Change. In K. Apt, editor, *Proceedings of the International Joint Conference and Symposium on Logic Programming (IJCSLP)*, pages 177–191. MIT Press, 1992.

[8] Vladimir Lifschitz. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming* [4], pages 297–352. Volume Coordinator D. Nute.

[9] David Makinson. General patterns in nonmonotonic reasoning. In *Handbook of Logic in Artificial Intelligence and Logic Programming* [4], pages 35–110. Volume Coordinator D. Nute.

[10] John McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[11] John McCarthy. Applications of circumscription to formalizing commonsense reasoning. *Artificial Intelligence*, 28:89–116, 1986.

[12] John McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[13] David J. Pym and James A. Harland. A uniform proof-theoretic investigation of linear logic programming. *Journal of Logic and Computation*, 4:175–207, 1994.

[14] Erik Sandewall. Underlying semantics for action and change with ramification. *Linköping Electronic Articles in Computer and Information Science*, 1(2), 1996. Available from `http://www.ep.liu.se/ea/cis/1996/002`.

[15] Murray Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* MIT Press, 1997.

[16] Alfred Tarski. Fundamentale Begriffe der Methodologie der deduktiven Wissenchaften. I. *Monatshefte für Mathematik und Physik*, 37:361–404, 1930.

[17] Alfred Tarski. Über einige fundamentale Begriffe der Metamathematik. *Comtes Rendues des séances de la Société des Sciences et des lettres de Varsovie*, 23 c. iii:22–29, 1930. Based on a lecture to the Warsaw branch of the Polish Mathematical Society, 1928.

[18] Alfred Tarski. Grundzüge des Systemenkalkül, Erster Teil. *Fundamenta Mathematicae*, 25:503–26, 1935.

[19] Alfred Tarski. Grundzüge des Systemenkalkül, Zweiter Teil. *Fundamenta Mathematicae*, 26:283–301, 1936.

[20] Alfred Tarski. Foundations of the calculus of systems. In *Logic, Semantics, Metamathematics: Papers from 1923 to 1938* [22], pages 342–383. An English translation of [18, 19].

[21] Alfred Tarski. Fundamental concepts of the methodology of the deductive sciences. In *Logic, Semantics, Metamathematics: Papers from 1923 to 1938* [22]. An English translation of [16].

[22] Alfred Tarski. *Logic, Semantics, Metamathematics: Papers from 1923 to 1938.* Clarendon, Oxford, 1956. Translated by J.H. Woodger.

[23] Alfred Tarski. On some fundamental concepts of metamathematics. In *Logic, Semantics, Metamathematics: Papers from 1923 to 1938* [22], pages 30–37. An English translation of [17].

[24] Michael Thielscher. Computing Ramifications by Postprocessing. In C. S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1994–2000, Montreal, Canada, August 1995. Morgan Kaufmann.

[25] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89:317–364, 1997.

[26] Willard van Ormond Quine. *Theories and Things.* Harvard, Cambridge MA, 1981.

[27] G. Graham White. The design of a situation-based Lygon metainterpreter: I. simple changes and persistence. Technical Report 729, Department of Computer Science, Queen Mary and Westfield College, University of London, 1996. Available from `ftp://ftp.dcs.qmw.ac.uk/pub/applied_logic/graham/metadesign.ps`.

[28] G. Graham White. Balls and string: Simulations and theories. Paper presented at Common Sense 98; available from `ftp://ftp.dcs.qmw.ac.uk/pub/applied_logic/graham/simulation.ps`, 1997.

[29] G. Graham White. Actions, ramification and linear modalities. Technical Report 749, Department of Computer Science, Queen Mary and Westfield College, 1998. Available from `ftp://ftp.dcs.qmw.ac.uk/pub/applied_logic/graham/ramMod.ps`.