Stratified Causal Theories for Reasoning about Deterministic Devices and Protocols

Antony Galton
University of Exeter
Exeter, UK
apgalton@ex.ac.uk

Juan Carlos Augusto
University of Southampton
Southampton, UK.
jca@ecs.soton.ac.uk

Abstract

We present a method for formalising the behaviour of simple deterministic devices and protocols in a way that makes explicit the causal dependencies amongst the component elements, thereby allowing true causal (as opposed to purely temporal) reasoning. Our intention is to handle such systems effectively in the simplest possible way, without invoking additional problematic considerations (concerning, for example, non-monotonicity) that may be necessary for modelling a more general range of scenarios.

1 Introduction

By way of motivation, consider the theory of the gas heater in [1]. The formalisation includes the rules:

```
init(coolDown,cold,[],[]).
init(coolDown,thermoVOpen,[],[]).
init(coolDown,burning,[],[pilotOn]).
```

These state that a cooling down event will bring about the following states: it is cold, the thermostatic valve is open, and, so long as the pilot light is on, the gas is burning. This is indeed what happens when the heater functions correctly. Although these may be thought of as causal rules, they do not capture the true causal sequence: the event `coolDown` causes the state `cold`; this in turn causes the state `thermoVOpen`, which in turn causes `burning`. Thus while `coolDown` does cause `burning`, it does so only indirectly, by way of the sequence just described. In Cervesato and Montanari's model, the first two rules above play no part in inferring that `burning` is initiated as a result of `coolDown`, this being secured by the third rule alone. Hence their model does not correctly capture the causal structure of the gas heater's operation. Similar examples can be given from their paper.

Our goal in this paper is to find a simple and clear method of formalising devices such as the gas heater and protocols (such as ATM transactions), in a way that makes explicit the causal dependencies, thereby allowing true causal (as opposed to merely temporal) reasoning. Our aim is to incorporate explicit reference to states and events and their causal relations in a system which allows both forward reasoning (for simulation) and backward reasoning (for explanation and prediction). The non-monotonic element is limited to, first, the persistence of states in the absence of any explicit causal explanation for their changing, and second, a closed-world assumption to the effect that the only impingement of the external world on the system is through those initiating events explicitly stated to occur. Our intention is to handle these notions effectively in the simplest possible way, for example without presupposing all the apparatus of the Situation Calculus, Event Calculus and other such formalisms.

2 Outline of Formal Theory

We assume a set of *atomic states*, with each positive atomic state S paired with its negation $-S$. We use $\overline{S}$ for the state paired with S: if S is positive, then $\overline{S} = -S$, while if $S = -S_1$ is negative then $\overline{S} = S_1$. We write $S_1 \sqcap S_2$ to denote the (non-atomic) state that holds exactly when S_1 and S_2 both hold.

The set $\mathcal{S}$ of all atomic states (positive and negative) is partitioned into a set $\mathcal{S}_I$ of *independent* states and a set $\mathcal{S}_D$ of *dependent* states. An independent state does not depend causally on other states holding at the same time, whereas a dependent state can do so. An independent state can only be initiated by the occurrence of initiating or terminating *events*. We call S *co-independent* if $\overline{S}$ is independent. Note that S and $-S$ may be both dependent, both independent, or one of each.

We use events to model impingements on the system

Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning (TIME'02)
1530-1311/02 $17.00 © 2002 IEEE

from outside. These are modelled as *initiation events* of the form $Init(S)$, where S is a state. $Init(S)$ is an event which ensures that S holds immediately after; it is not necessary to characterise further the specific external trigger which causes this to happen, since our main concern is to model the internal workings of the system.

Time is represented as a discrete series of *atomic intervals*, labelled by the natural numbers. We write $HoldsAt(S,t)$ to mean that state S holds at atomic interval t. The rule for negative states is

$$HoldsAt(-S,t) \leftrightarrow \neg HoldsAt(S,t).$$

Events occur at the instants which separate neighbouring atomic intervals. The instant between t and $t+1$ is denoted t^*. Initiation events bring about states according to the rule:

$$Occurs(Init(S),t^*) \rightarrow HoldsAt(S,t+1).$$

Note that it is not required that $\overline{S}$ holds at t, although this will often be the case in practice.

There are two kinds of *causal rule*:

Same-time rules: $S_1 \sqcap S_2 \sqcap \cdots \sqcap S_n \rightsquigarrow S$
Next-time rules: $S_1 \sqcap S_2 \sqcap \cdots \sqcap S_n \rightsquigarrow \bigcirc S$

where each S_i is an atomic state and $S \in \mathcal{S}_D$.

Same-time rules are required to be *stratified*, as follows:

1. A Stage 1 rule is a rule $S_1 \sqcap S_2 \sqcap \cdots \sqcap S_n \rightsquigarrow S$, where $S_1,\ldots,S_n$ are all independent. In this case S is *1-dependent*. (The independent states are *0-dependent*.)

2. A Stage k rule is a rule $S_1 \sqcap \cdots \sqcap S_n \rightsquigarrow S$, where each of $S_1,\ldots,S_n$ is at most $(k-1)$-dependent, and at least one of them is $(k-1)$-dependent. Then S is *k-dependent*, and $\overline{S}$ is *co-k-dependent*.

3. A set of same-time rules is stratified so long as for every rule in the set there is a number k such that the rule is a Stage k rule.

A same-time rule $S_1 \sqcap S_2 \sqcap \cdots \sqcap S_n \rightsquigarrow S$ can be *applied* at time t so long as

$$HoldsAt(S_1,t) \wedge \cdots \wedge HoldsAt(S_n,t).$$

The result of applying it is that we may assert $HoldsAt(S,t)$. Under the same condition we can also apply a next-time rule $S_1 \sqcap S_2 \sqcap \cdots \sqcap S_n \rightsquigarrow \bigcirc S$, and the result of applying it is that we may assert $Holds(S,t+1)$. If a rule can be applied, it is said to be *live*.

3 Forward Reasoning

Given a stratified set of same-time rules, a set of next-time rules, an *initial condition* specified by the truth value of $HoldsAt(S,0)$ for each $S \in \mathcal{S}_I$, and an *event list*, which is a set of formulae of the form $Occurs(Init(S),t^*)$, we compute the resulting history as follows:

1. At $t=0$, apply any live same-time rules, in order of increasing Stage (note that some rules may become live as a result of applying others). When all possible same-time rules have been applied, apply any live next-time rules. Any positive state S whose value is not already determined at $t=0$ is now assumed to be false, i.e., $HoldsAt(-S,0)$.

2. For $t = 1,2,3,\ldots$,

   (a) For each event $Occurs(Init(S),(t-1)^*)$, if $Holds(\overline{S},t-1)$, assert $HoldsAt(S,t)$.

   (b) For each co-independent state S, if $HoldsAt(S,t-1)$ and it has not already been asserted that $HoldsAt(\overline{S},t)$, then assert $HoldsAt(S,t)$. This is called 'applying persistence' to the state S.

   (c) For $k = 1,2,3,\ldots$, first apply any live same-time rules of Stage k, then apply persistence: for any co-k-dependent state S, if $HoldsAt(\overline{S},t)$ has not already been asserted, and $HoldsAt(S,t-1)$, then assert $HoldsAt(S,t)$.

   (d) Apply any live next-time rules.

We illustrate the algorithm, with same-time rules only, using the gas-heater example from [1]. The independent states are $\pm Power$, $\pm Gas$, $\pm Cold$, $\pm Lighter$, $\pm Disable$, and the dependent states are $\pm TVOpen$, $\pm SVOpen$, $\pm Sparking$, $\pm Pilot$, $\pm Burning$. The same-time rules are:

Stage 1:	1.	$Power \sqcap Cold \rightsquigarrow TVOpen$
	2.	$Power \sqcap -Cold \rightsquigarrow -TVOpen$
	3.	$-Gas \rightsquigarrow -Pilot$
	4.	$Disable \rightsquigarrow SVOpen$
	5.	$Lighter \sqcap Power \rightsquigarrow Sparking$
	6.	$-Power \rightsquigarrow -Sparking$
	7.	$-Lighter \rightsquigarrow -Sparking$
Stage 2:	8.	$-Pilot \sqcap -Disable \rightsquigarrow -SVOpen$
	9.	$Sparking \sqcap Gas \sqcap SVOpen \rightsquigarrow Pilot$
	10.	$-TVOpen \rightsquigarrow -Burning$
	11.	$-Pilot \rightsquigarrow -Burning$
Stage 3:	12.	$Pilot \sqcap TVOpen \rightsquigarrow Burning$

Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning (TIME'02)
1530-1311/02 $17.00 © 2002 IEEE

$Time$	0	1	2	3	4	5	6	7	8
$Power$	$-$	$-$	$+$	$+$	$+$	$+$	$+$	$-$	$-$
Gas	$-$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
$Cold$	$+$	$+$	$+$	$+$	$+$	$+$	$-$	$-$	$+$
$Lighter$	$-$	$-$	$-$	$+$	$-$	$-$	$-$	$-$	$-$
$Disable$	$-$	$-$	$-$	$+$	$+$	$-$	$-$	$-$	$-$
$TVOpen$	$-$	$-$	$+$	$+$	$+$	$+$	$-$	$-$	$-$
$SVOpen$	$-$	$-$	$-$	$+$	$+$	$+$	$+$	$+$	$+$
$Sparking$	$-$	$-$	$-$	$+$	$-$	$-$	$-$	$-$	$-$
$Pilot$	$-$	$-$	$-$	$+$	$+$	$+$	$+$	$+$	$+$
$Burning$	$-$	$-$	$-$	$+$	$+$	$+$	$-$	$-$	$-$

Figure 1. The gas-heater example

Assume the initial condition is $HoldsAt(Cold, 0)$, and let the event-list be

$$Occurs(Init(Gas), 0^*),$$
$$Occurs(Init(Power), 1^*),$$
$$Occurs(Init(Disable), 2^*),$$
$$Occurs(Init(Lighter), 2^*),$$
$$Occurs(Init(-Lighter), 3^*),$$
$$Occurs(Init(-Disable), 4^*),$$
$$Occurs(Init(-Cold), 5^*),$$
$$Occurs(Init(-Power), 6^*),$$
$$Occurs(Init(Cold), 7^*)$$

The history computed by means of the algorithm is illustrated in Figure 1.

4 Backward Reasoning

The forward-reasoning procedure is appropriate for simulating behaviour, but less so for prediction and explanation. For this it is preferable to use the backward reasoning algorithm we have developed. We do not have space to do more than describe this in outline.

Before causal inference begins, we collect together information about the causal relations amongst the states. These relations are represented by a set of trees, in which each node is labelled by a state. Each state appears as the label of the root node of one of the trees; within a tree, a node labelled by S has as its descendents one node for each state on which S immediately depends by one of the same-time rules. Thus the independent states always appear as leaves of the trees; each independent state also gives rise to a tree with a single node. A tree is *activated* at time t if at least one of its independent states is known to hold at t; it is *fully activated* at t if *all* its independent states hold at t.

The algorithm requires us to find the set of trees supporting a given state S. This is achieved recursively: for each rule $B_1 \sqcap \cdots \sqcap B_n \rightsquigarrow S$, we apply the process to find the trees supporting each of $B_1, \ldots, B_n$, and each combination of such trees is joined together to form a new tree. The base cases of the recursion are provided by the independent states, whose trees consist of single nodes.

This process can lead to combinatorial explosion, and is thus not suited to practical implementation of large rule sets. Our purpose here is to establish the correct logical principles first before addressing complexity issues. Note that the stratification of the rules means that the tree-construction will always terminate.

5 Next-time rules

Next-time rules express delayed causation, where states holding at one time influence the states holding at the next time. The forward-reasoning algorithm already tells us how to handle them; we illustrate with the example of an automatic hand-drier with a time-out mechanism. The key notion embodied by next-time rules is that causality operates from one time to the next, it does not 'leap' over an interval. Thus a device with a time-out mechanism has to have an explicit 'memory' of how much time it has remaining.

We use states On, $ButtonPressed$, and $TimeLeft(n)$ (for $n = 0, 1, 2, 3$). The rules are

1. $ButtonPressed \rightsquigarrow On$
2. $ButtonPressed \rightsquigarrow TimeLeft(3)$
3. $On \sqcap -ButtonPressed \sqcap TimeLeft(n) \rightsquigarrow \bigcirc TimeLeft(n-1)$
4. $On \sqcap TimeLeft(1) \rightsquigarrow \bigcirc(-On)$

Assume that at $t = 0$ we have $-On$, $-ButtonPressed$, and $TimeLeft(0)$, and let the event list be

$$Occurs(Init(ButtonPressed), 2^*),$$
$$Occurs(Init(-ButtonPressed), 3^*),$$
$$Occurs(Init(ButtonPressed, 5^*),$$
$$Occurs(Init(-ButtonPressed), 8^*).$$

The button is pressed after two time steps and released after a further time step; it is then pressed again after a further two steps, and held down for three steps before being released. Using the rules, we can compute that the drier is on from times 3 to 11 inclusive.

References

[1] I. Cervesato and A. Montanari. A calculus of macro-events: Progress report. In A. Trudel and S. Goodwin, editors, *Proceedings of the Seventh International Workshop on Temporal Representation and Reasoning*, pages 47–58. IEEE Computer Society Press, 2000.

Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning (TIME'02)
1530-1311/02 $17.00 © 2002 **IEEE**