Operators vs Arguments

The Ins and Outs of Reification

Antony Galton
University of Exeter, UK

What is Reification?

$White(x)$

$Colour(x, white)$

All balls in the bag are the same colour:

$$\exists x \forall y(Ball(y) \wedge In(y, b) \rightarrow Colour(y, x))$$

Reification of times, states, properties, actions, events, . . .

Reification of Times

Tense Logic: Pp

Standard Logic: $\exists t'(t' \prec t \wedge p(t'))$

Prior: 'Tense Logic and the Logic of Earlier and Later' (in *Papers on Time and Tense*, 1968)

Massey: 'Tense Logic! Why Bother?' (*Noûs*, 1969)

van Benthem: 'Tense Logic and Standard Logic' (*Logique et Analyse*, 1977)

Part of wider debate on Classical vs Non-classical Logic (e.g., Gabbay 1994)

The Reification Debate in AI

Heyday in the 1980s and 1990s

Problem for Planning and Knowledge Representation: How to express in precise logical terms the general temporal knowledge which an autonomous agent must have if it is to be able to formulate plans, understand natural language, and generally hold its own in a constantly changing world.

How should we express knowledge about states, actions, facts, events, . . . ?

To reify or not to reify?

Prior (1968)

Relationship between Tense Logic and U-calculi.

Tense Logic: tense operators act on formulae to produce formulae: Pp, Fp, Hp, Gp.

U-calculus: explicit reference to times, and a symbol T relating propositions to times: Ttp.

What is the logical form of Ttp?

If p is a self-standing proposition, then Tt acts as a modal operator. We could write $\mathbf{True}_t(P)$ or $\mathbf{True}(P, t)$ (cf. Rescher's $R_t(p)$).

Alternatively, treat p, etc, as '*predicates* of the instants "at" which they are . . . said to be true'. So T_p is a predicate, $P(\_)$, and we can rewrite Ttp as $P(t)$: the **Method of Temporal Arguments (MTA)**.

But there is a third possibility, not considered by Prior . . .

Model-theory encoding

We treat the p in Ttp not as a formula or a predicate, but as a *term*.

This means that T becomes a *predicate* ('is true at').

We write $T(p, t)$.

Since what was a proposition has been metamorphosed into a term, this is a reified logic.

What do propositional terms denote?

One answer: they denote formulae.
The reified logic is a *first-order encoding of the model theory for Tense Logic*.

Some comparisons

Semantic rules for '$\rightarrow$' and '$\mathbf{G}$'.

- U-calculus
  $Tt(p \rightarrow q) \rightarrow (Ttp \rightarrow Ttq)$
  $TtGp \leftrightarrow \forall t'(t \prec t \rightarrow Ttp)$

- Modal version
  $\mathbf{True}(P \rightarrow Q, t) \rightarrow (\mathbf{True}(P, t) \rightarrow \mathbf{True}(Q, t))$
  $\mathbf{True}(\mathbf{G}(P), t) \leftrightarrow \forall t'(t \prec t' \rightarrow \mathbf{True}(P, t'))$

- Method of temporal arguments
  $(P \Rightarrow Q)(t) \rightarrow (P(t) \rightarrow Q(t))$
  $[\mathbf{G}(P)](t) \rightarrow \forall t'(t \prec t' \rightarrow P(t'))$

- Reified version
  $True(if(p, q), t) \rightarrow (True(p, t) \rightarrow True(q, t))$
  $True(g(p), t) \leftrightarrow \forall t'(t \prec t' \rightarrow True(p, t))$

Han Reichgelt

'Semantics for Reified Temporal Logic' (In Hallam and Mellish, *Advances in Artificial Intelligence*, 1987).

Modal temporal logic TM ($\approx$ Tense Logic)
Reified temporal logic TR

TR is a formalisation of TM in a many-sorted first-order predicate logic.

Sorts: *expressions* (propositional, individual)
denotations

Reichgelt writes $Holds(p, t)$ instead of $True(p, t)$.

Unlike Prior, he quantifies over proposition expressions.

Quantification over propositional terms as a *sine qua non* for calling a logic reified?

Yoav Shoham

'Temporal Logics in AI: Semantical and Ontological Considerations' (*Artificial Intelligence*, 1987)

TRUE(t_1, t_2, p)
'proposition p is true over interval $[t_1, t_2]$'

Shoham: TRUE is neither a predicate nor a modal operator.
Hence p is neither a term nor a formula.
Shoham: p is a 'primitive proposition'.

Shoham presents his logic as a *alternative* to reifying propositions, but calls it a 'new reified temporal logic'.

Reichgelt calls it 'a hybrid of a modal approach and the method of temporal arguments'.
Bacchus *et al* (1991) say it's 'closer to the spirit of an *intensional* logic'.

Why Reify?

Shoham (1987)

If time is represented as an argument . . . to predicates, then there is nothing general you can say about the temporal aspect of assertions. For example, you cannot say that "effects cannot precede their causes"; at most you can say that about specific causes and effects.

Reichgelt (1987)

In TM . . . it is impossible to express general temporal knowledge such as "effects cannot precede their causes". Because one cannot talk about effects and causes in general, one can at most make the above statements about specific causes and effects.

Bacchus, Tenenberg and Koomen (1991)

One advantage that is possessed by reified logics is that they allow quantification over propositions. For example, one can express the assertion that "effects cannot precede their causes" in a reified logic.

"Effects cannot precede their causes"

Reichgelt

$$\forall p \forall q (Causes(p, q) \rightarrow \forall t \forall t' (Holds(p, t) \wedge Holds(q, t') \rightarrow \neg(t' \prec t)))$$

But this says that *no* occurrence of q can precede *any* occurrence of p!

Bacchus *et al*

$$\forall y \forall t_2 (\exists x Causes(x, y) \wedge Holds(y, t_2) \rightarrow \exists z \exists t_1 (Causes(z, y) \wedge Holds(z, t_1) \wedge t_1 \prec t_2))$$

This says that if something has a cause, then *at least one* of its causes must precede it.

Shoham

No attempt to express this proposition!

What's really going on here?

'The alarm's ringing at time t_1 caused John to wake at time t_2.'

Reified language
We use a predicate $Causes$:

1. $Causes(ring(alarm), t_1, wake(john), t_2))$

2. $\forall x, x', t, t'(Causes(x, t, x', t') \rightarrow t \preceq t')$

Unreified language
We use a 'modal connective' $\mathbf{Causes}$:

3. $\mathbf{Causes}(Ring(alarm, t_1), Wake(john, t_2))$

4. $\forall t, t'(\mathbf{Causes}(\Phi(t), \Psi(t')) \rightarrow t \preceq t')$

To express the equivalent of (2), we need a second-order formula:

5. $\forall \Phi, \Psi, t, t'(\mathbf{Causes}(\Phi(t), \Psi(t')) \rightarrow t \preceq t')$

Denotata of propositional terms

What can propositional terms denote, if not formulae?

A frequent answer: States and Events!

The $Causes$ predicate already seems to presuppose this.

Events John flies from London to Paris
John leaves London
John arrives in Paris

States John is in London
John is in the plane
John is in Paris

Events *happen* or *occur* whereas states *hold* or *obtain*.

This idea underlies some influential temporal logics that appeared in the AI community in the early 1980s.

Drew McDermott

'A Temporal Logic for Reasoning about Processes and Plans' (*Cognitive Science*, 1982)

State
'An instantaneous snapshot of the universe'
Date
A real number $d(s)$ associated with a state s
Chronicle
'A complete possible history of the universe'

Fact A set of states
Event A set of intervals

A fact may be *true* in a state: $True(p, s)$
This is 'syntactic sugar' for $s \in p$.

An event may *occur* on an interval:
$Occurs(e, [s_1, s_2])$
This is syntactic sugar for $[s_1, s_2] \in e$.

McDermott uses his logic to support an account of causality, reasoning about continuous change, and planning.

James Allen

'Towards a General Theory of Action and Time' (*Artificial Intelligence*, 1984)

$Occurs(e, i)$: an occurrence of type e occurs over the interval i.

$Holds(p, i)$: the property p holds throughout the interval i.

$\forall p \forall i(Holds(p, i) \rightarrow \forall j(In(j, i) \rightarrow Holds(p, j)))$
$\forall e \forall i(Occurs(e, i) \rightarrow \forall j(In(j, i) \rightarrow \neg Occurs(e, j))$

Event-causality
$Ecause(e_1, i_1, e_2, i_2)$ says that event e_1, if it occurs on i_1, causes event e_2 to occur on i_2.

'An event cannot cause events prior to its occurrence':

$\forall e \forall e' \forall i \forall i'(Ecause(e, i, e', i') \rightarrow NotBefore(i', i))$,

Criticisms of McDermott and Allen

Shoham

Allen was wrong to enshrine the distinction between properties, processes, and events in the logic—this is unnecessary for some purposes, insufficient for others.

Likewise with McDermott's fact vs event.

Hence Shoham's proposal to have a single syntactic category of 'primitive propositions' (though what these are, Shoham does not make clear).

Reichgelt only grudgingly concedes that these systems 'can be interpreted as' reified temporal logics—proposing instead his own system as the correct form of such a logic.

Unreified Logics I

Bacchus, Tenenberg and Koomen

'A non-reified temporal logic' (*Artificial Intelligence*, 1991)

They use the Method of Temporal Arguments, replacing $True(on(a,b),t)$ by $On(a,b,t)$.
(Or $On(a(t),b(t),t)$.)

Systematic translation of Shoham's system into theirs; the two logics are equivalent. (But Shoham's isn't really reified.)

Unreified Logics II

Brian Haugh
'Non-standard Semantics for the Method of Temporal Arguments' (*IJCAI* 1987)

'Representing the times of occurrences of events requires a different sort of semantics than the times of the holding of ordinary facts'

He uses $Occurs(e, i)$, where e is a token event. (Allen's and McDermott's events are types.)

But he also allows event types:

$$Occurs(throw(john, ball1), i).$$

This is not a reified logic, since events are 'clearly distinguished from propositions'. Even propositional terms could be allowed, so long as there is no 'truth predicate'.

So what exactly is a reified temporal logic?

How to Unreify a Reified Logic

Antony Galton, 'Reified Temporal Logics and How to Unreify Them' (*IJCAI*, 1991)

Replace types by tokens.

Event tokens: cf. Davidson
State tokens: cf. situations of McCarthy and Hayes (also McDermott's states)

Reified
$Occurs(fly(john, london, paris), i)$
$Holds(in(john, paris), t)$

Unreified
$\exists e(Fly(john, london, paris, e) \wedge Occurs(e, i))$
$\exists s(In(john, paris, s) \wedge Holds(s, t))$

This is token-reification, as opposed to 'true' reification (type-reification).

'Effects cannot precede their causes':

$$\forall e, e'(Causes(e, e') \rightarrow \neg(time(e') \prec time(e))).$$

Event-Token Reification

This goes back to Davidson 'The Logical Form of Action Sentences' (1967).

John ate an egg with a spoon in the kitchen at lunchtime.

$$\exists e(Eat(e) \wedge \\ Agent(e, john) \wedge Patient(e, egg) \wedge \\ Instrument(e, spoon) \wedge Place(e, kitchen) \wedge \\ Time(e, lunchtime))$$

$$\exists e(Eat(john, egg, e) \wedge Instrument(e, spoon) \wedge \\ Place(e, kitchen) \wedge Time(e, lunchtime))$$

This kind of analysis (but with e a constant rather than a variable) was used in the *Event Calculus* of Kowalski and Sergot (1986). Their inspiration was not Davidson, however, but the Situation Calculus.

The Situation Calculus

John McCarthy and Pat Hayes, 'Some Philosophical Problems from the Standpoint of Artificial Intelligence' (1969)

Key concept: a situation, 'the complete state of the universe at a given instant of time'.

We can think of this as a 'global state token' (analogous to McDermott's states).

Propositional fluents map situations to truth values: $Raining(x, s)$ says that in situation s, it is raining in place x.

Situation Calculus allows a kind of reification by λ-abstraction: $Holds(\lambda s'.Raining(x, s'), s)$.

These reified fluents are analagous to McDermott's facts.

Actions in the Situation Calculus

Actions are represented as types: generic actions that can be performed in different situations.

$result(a, s)$ denotes the situation resulting from action a being performed in situation s.

Reification of fluents allows general temporal assertions such as

$$\forall p \forall a \forall s (Happens(a, s) \wedge Initiates(a, p) \rightarrow Holds(p, result(a, s)))$$

'An effect cannot precede its cause':

$$\forall a \forall s \neg (time(result(a, s)) \prec time(s)).$$

Another Approach to Token-Reification

Lluís Vila and Han Reichgelt, 'The Token Reification Approach to Temporal Reasoning' (*Artificial Intelligence*, 1996)

State and event tokens $p(x_1, \ldots, x_n, i)$, where i is a temporal argument:

$$Occurs(fly(john, london, paris, i_1))$$
$$Holds(in(john, paris, i_2))$$

Type information is incorporated in a complex event-token symbol. (In the other approach, event-tokens were unstructured.)

'Causes precede their effects':

$$\forall e \forall s(Cause(e, s) \rightarrow (Occurs(e) \rightarrow (Holds(s) \wedge begin(e) \prec begin(s))))$$

Token identity

A token is constructed from a type and a time. This rules out simultaneous tokens of the same type.

Hence 'John opens a window' is not a good type (since John *can* open two windows simultaneously).

Instead of

$$Occurs(openwindow(john, t))$$

we must write

$$\exists x(Window(x) \wedge Occurs(open(john, x, t))).$$

John opens two windows simultaneously:

$$\begin{aligned}\exists x \exists y(&Window(x) \wedge Window(y) \wedge \\ &Occurs(open(john, x, t)) \wedge \\ &Occurs(open(john, y, t)))\end{aligned}$$

(This is reification of windows!)

A few final thoughts

Do states and events exist? How do we handle causal relations?

We can ask these questions about types or about tokens.

To reify is to presuppose some sort of existence, especially if we quantify over the reified entities.

We then have a responsibility to determine criteria of identity for them—a key concern for Davidson, but rather neglected in the AI literature.

The drive to do everything in first-order logic leads us to handle time in ways very different from Prior. It remains to this day a prolific strand in AI.

Minsky: "It's tying your hands behind your back".