The Productions of Time

Mark Steedman, Informatics, University of Edinburgh

Sinn & Bedeutung, Osnabrück 2001

Eternity is in love with the productions of time

— William Blake, *The Marriage of Heaven and Hell*.

`www.cogsci.ed.ac.uk/~steedman/papers.html`

Temporal Ontology

The Perfect

- Tenses/Aspects aren't primarily to do with time at all.

(1) I have lost my watch (# but I have found it again)

(2) Yesterday, I lost my watch (but I (have) found it again)

(3) a. #I have breathed
    b. #Einstein has visited New York

The Progressive

- Tenses/Aspects are non-uniform with respect to factive entailment:

(4) a. Keats was writing $\models$ Keats wrote

b. Keats was writing a sonnet $\not\models$ Keats wrote a sonnet

(5) A proposition ϕ holding of an interval t is downward entailing if it entails that ϕ also holds of all subintervals of t down to some reasonable minimum size.

- Hence we must establish an ontology of *types* of events.

- A standard move in natural language semantics since Aristotle, Jespersen, and Vendler 1967.

Aspectual Coercion

- Aspect is not inherent in verbs.
- Suppose we know that Keats writes a sonnet to time every Sunday afternoon, and that on 23rd of December 1816 he wrote one in just fifteen minutes. (In fact, he did—it was called “In Disgust of Vulgar Superstition”.) We can say:

  (6) Keats wrote in fifteen minutes.

The Tense/Aspect System in NL Semantics

- Moens and Steedman (1988) proposed an ontology for durative and telic events to support natural language semantics of tense and temporality, linking world knowledge to aspectual coercion.

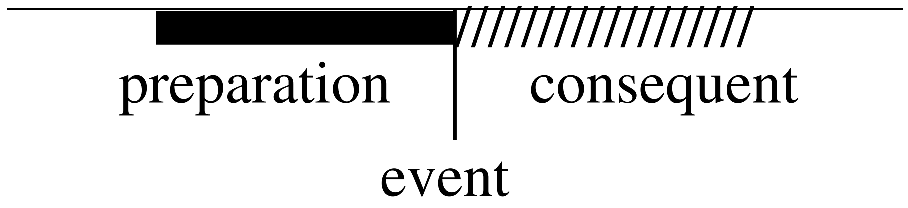

Figure 1: The event nucleus (adapted from Moens and Steedman 1988)

- In particular, durative events were represented in terms of inceptive and culminative events with progressive states as their respectively consequent and preparatory states.
- But what they offered is something less than a semantics.

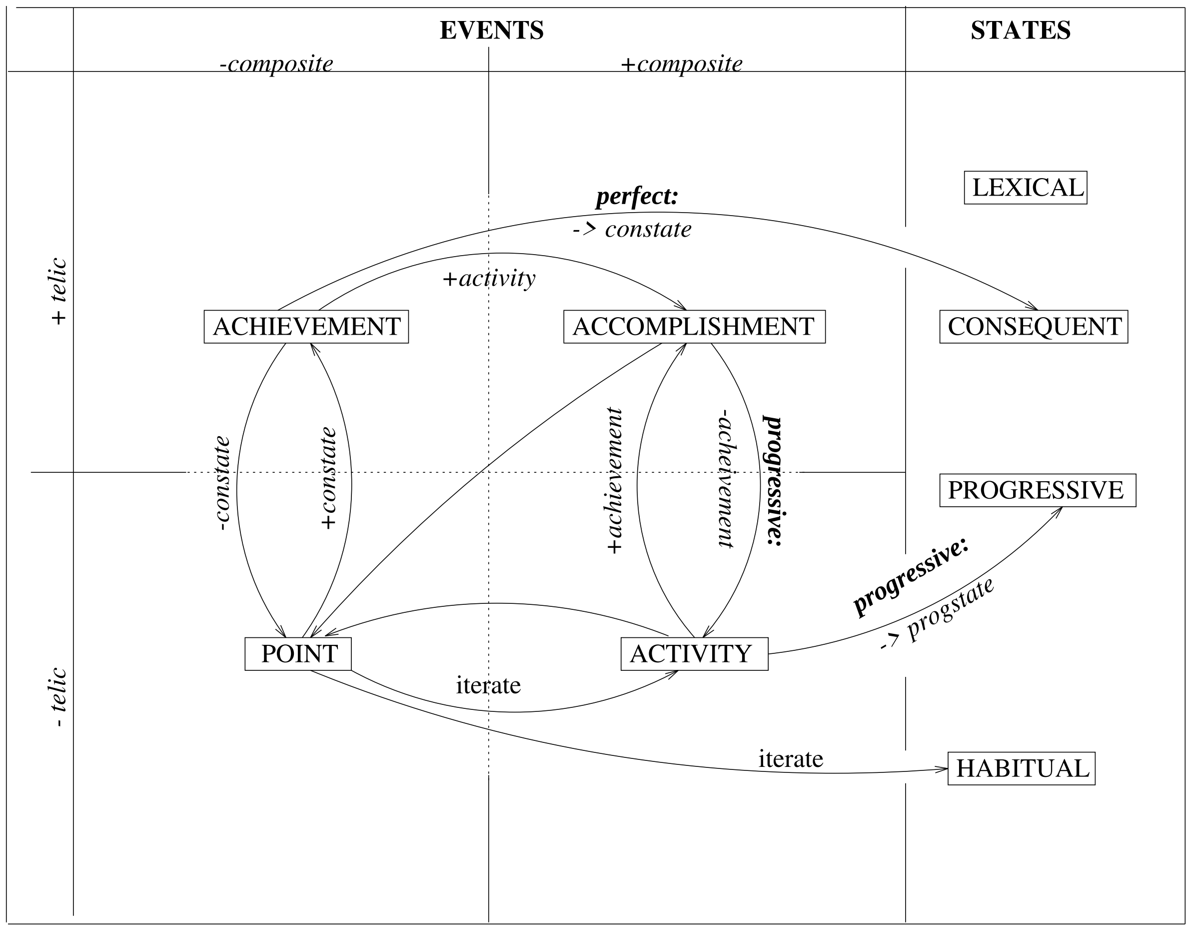

Figure 2: A scheme of aspectual coercion (Moens and Steedman 1988)

The Dynamic Event Calculus

- Basic Dynamic Logic:

(7) $n \geq 0 \Rightarrow [\alpha](y = F(n))$

(8) $n \geq 0 \Rightarrow \langle\alpha\rangle(y = F(n))$

- The particular dynamic logic that we are dealing with here is one that includes the following dynamic axiom (the operator ; is *sequence*, an operation related to functional composition of functions of type *situation* $\rightarrow$ *situation*):

  (9) $[\alpha][\beta]P \Rightarrow [\alpha;\beta]P$

- Composition is one of the most primitive *combinators*, or operations combining functions, which Curry and Feys (1958) call **B**, writing the above sequence $\alpha;\beta$ as $\mathbf{B}\alpha\beta$, where

  (10) $\mathbf{B}\alpha\beta \equiv \lambda s.\alpha(\beta s)$

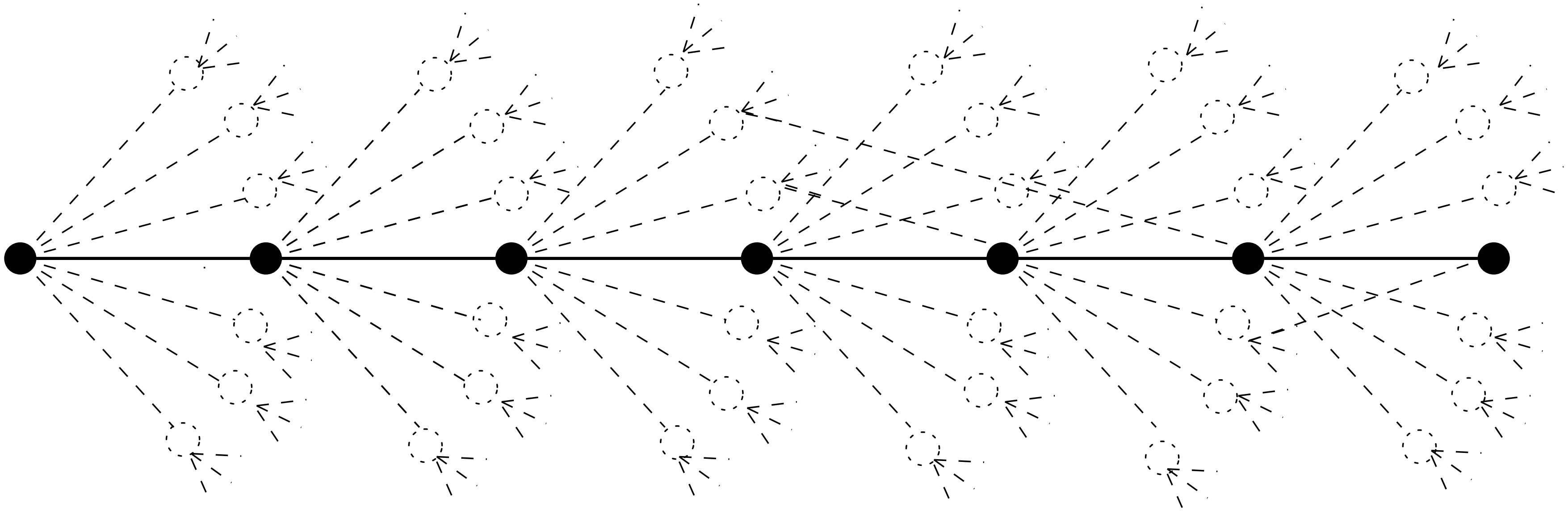

Figure 3: Kripke Model of Causal Accessibility Relation

Situation/Event Calculi and the Frame Problem

- The Situation Calculus (McCarthy and Hayes 1970) and its descendants can be seen as reified versions of Dynamic Logic.
- These calculi are heir to the "Frame Problem," which arises from the fact that humans conceptualize events in terms of very localized changes to situations.
- For example, the effects of an event of *My eating a hamburger* are confined to specific aspects of myself and the hamburger. The color of the walls, the day of the week, the leadership of the Conservative and Unionist party, and countless other aspects of the situation remain unchanged.
- This character of the knowledge representation raises the Frame Problem in two forms: the "Representational" and "Inferential" versions.

The Representational Frame Problem

- Since change is local, it is cumbersome to explicitly represent the input effect of each event on each fluent by innumerable rules such as

(11) $color(wall,x) \Rightarrow [eat(hamburger)]color(wall,x)$

- Kowalski (1979) solved the representational problem using the equivalent of reified Frame Axioms such as:

(12) $p \wedge (p \neq here(hamburger)) \Rightarrow [eat(hamburger)]p$

- This keeps rules defining the positive effects of eating hamburgers as simple as (7).

- But if we ever need to know what the color of the walls is after a sequence of, say, five hamburger eating events, then we have to do costly theorem proving search. This is the *Inferential* form of the Frame Problem.

The Inferential Frame Problem

- The STRIPS program (Fikes and Nilsson 1971) solved both representational and inferential problems by representing the situation as a changing *database* and representing change as sets of *preconditions* and localized *database updates*:

OPERATOR:	$\lambda x.eat(x)$
PRECONDITIONS:	$hamburger(x)$
	$here(x)$
	$hungry$
DELETIONS:	$here(x)$
ADDITIONS:	$thirsty$

- Such representations were initially derided by logicians (because of their nonmonotonicity), but are in fact reminiscent of Linear Logic.

STRIPS Update as Linear Implication

- We can represent events involving doors in this notation for a world (simplified for purposes of exposition) in which there are two places *out* and *in* separated by a door which may be *open* or *shut*, as follows:

- First we need rules to capture the fact that pushing a door changes it from open to shut and vice versa:

(13) a. $shut(x) \multimap [push(x)]open(x)$

b. $open(x) \multimap [push(x)]shut(x)$

(14) a. $in \multimap [go\text{-}through(x)]out$

b. $out \multimap [go\text{-}through(x)]in$

- These rules use *linear* logical implication $\multimap$, rather than intuitionistic implication $\Rightarrow$ in those rules that change the value of fluents.

STRIPS Update as Linear Implication (Contd.)

- The effect of such systems can be exemplified as follows: if the initial situation is that you are in and the door is shut:

  (15) $in \wedge door(d) \wedge shut(d)$

  —then the linear rules (13) mean that an attempt to prove the proposition in (16) concerning the state of the door in the situation that results from pushing the door will fail because rule (13a) has removed the fact in question from the database that results from the action $push(d)$:

  (16) $[push(d)]shut(d)$

STRIPS Update as Linear Implication (Contd.)

- On the other hand, attempts to prove the following will all succeed, since they are all facts in the database that results from the action $push(d)$ in the initial situation (15):

(17) a. $[push(d)]open(d)$
b. $[push(d)]door(d)$
c. $[push(d)]in$

- Using linear implication (or the equivalent rewriting logic devices or state update axioms of Thielscher (1999) and Martí-Oliet and Meseguer (1999)) for STRIPS-like rules makes frame axioms along the lines of (12) unnecessary. Instead, they are theorems of the linear logic representation.

STRIPS Preconditions as Intuitionistic Implication

- Preconditions are defined in terms of a predicate *possible*:

(18) a. $door(x) \wedge open(x)$
$\Rightarrow possible(go\text{-}through(x))$

b. $door(x) \Rightarrow possible(push(x))$

- We also need to define the transitive property of the possibility relation, as follows, using the definition 9 of event sequence composition:

(19) $\models possible(\alpha) \wedge [\alpha]possible(\beta) \Rightarrow possible(\alpha;\beta)$

STRIPS Planning in Dynamic Event Calculus

- This fragment supports a simple-minded form of planning, in which starting from the world (20) in which I am *in*, and the door is *shut* and stating the goal (21) meaning "find a possible series of actions that will get me *out*," can be made to automatically deliver a constructive proof that one such plan is (22):

  (20) $in \wedge door(d) \wedge shut(d)$

  (21) $possible(\alpha) \wedge [\alpha]out$

  (22) $\alpha = push(d); go\text{-}through(d)$.

- The situation that results from executing this plan in the start situation (15) is one in which the following conjunction of facts is directly represented by the database:

  (23) $out \wedge door(d) \wedge open(d)$

The Monkey and the Bananas

- A more interesting example, still simplified: grabbing something gets you to the state of having it, and if you were 6 ft higher than where you are you could grab the bananas (the latter a grotesque hack to avoid axiomatizing arithmetic):

(24) a. $[grab(x)]have(x)$

b. $at((here + 3) + 3) \Rightarrow possible(grab(bananas))$

- If something is a box you can climb on it:

(25) $box(b) \Rightarrow possible(climb\text{-}on(b))$

- —and if you are at a place and you climb on a box you are at a place that is higher by 3ft:

(26) $at(p) \multimap [climb\text{-}on(b)]at(p+3)$

The Monkey and the Bananas, (Contd.)

- if two things have nothing on top of them and are not the same thing you can put one on the other:

(27) $clear(x) \wedge clear(y) \wedge (x \neq y) \Rightarrow possible(puton(x,y))$

- —and if x is on something and you put it on something else then that something becomes clear and x is on that something else:

(28) a. $on(x,z) \multimap [puton(x,y)]clear(z)$
b. $clear(y) \multimap [puton(x,y)]on(x,y)$

An Aside on "Bunched" Implication

- It is worth noting that the use of a hybrid logic in which fluents like *clear* can be antecedents to both nonlinear precondition or "qualification" rules like (27) and linear consequence or "ramification" rules like (28) means that we avoid reintroducing the frame problem by having to explicitly state that $clear(x)$ in the consequent of (28b).

- Such a logic is a logic of "bunched implication". O'Hearn and Pym (1999); Pym (1999) prove a cut elimination theorem etc. for a similar bunched implication logic.

The Monkey and the Bananas, (Contd.)

- If the initial state of the world is as follows:

(29) $at(here) \wedge box(b1) \wedge box(b2) \wedge clear(b1) \wedge clear(b2)$

- —then the goal (30a) gives rise to (30b) as one possible plan

(30) a. $possible(\alpha) \wedge [\alpha]have(bananas)$

b. $\alpha = [puton(b1, here); climb\text{-}on(b1);$
$puton(b2, b1); climb\text{-}on(b2); grab(bananas)]$

- However, we have said nothing yet about the problem of *Search* implicit in identifying such plans.

Planning in Animals

- Some animals can make plans involving tools of the kind in (30) (Köhler 1925).

Figure 4: From Köhler 1925

Figure 5: From Köhler 1925

- Such planning seems to be *reactive* to the presence of the tool and *forward-chaining*, rather than backward-chaining (working from goal to tool).
- This seems a good way for an animal to plan, and implies that actions are accessed via perception of the objects that mediate them—in other words that actions are represented as the *Affordances* of objects, in Gibson's (1966) terms.
- The relation between an object and its affordances can be thought of in terms of a second simple combinator, called **T**, or **Type-raising**, where:

  (31) $\mathbf{T}a = \lambda f.fa$

- Relatives of **B** and **T** play a central role in Combinatory Categorial Grammar (CCG, Steedman 2000, *passim*), suggesting that grammar may be evolutionarily and developmentally an adaptation of the planning apparatus.

Durative and Telic Events in DEC

- If you aren't already moving and you are somewhere, then its possible to start moving to somewhere else:

(32) $\neg in\_progress(move(x, l_0, l)) \wedge at(x, l_0)$
$\Rightarrow possible(start(move(x, l_0, l)))$

- If you do so, then you stop being where you were, and start being in motion:

(33) $at(x, l_0) \multimap [start(move(x, l_0, l))] in\_progress(move(x, l_0, l))$

Durative and Telic Events in DEC (Contd.)

- Once you are in motion, its possible to stop, and its also possible to reach where you're going(there are other conditions on the latter that are not directly under your control):

(34) a. $\mathit{in\_progress}(p) \Rightarrow \mathit{possible}(\mathit{stop}(p))$

b. $\mathit{in\_progress}(\mathit{move}(x, l_0, l)) \Rightarrow \mathit{possible}(\mathit{reach}(x, l))$

- If you stop moving from l_0 to l, then you are no longer in motion and are between l_0 to l. If you reach l, then you are no longer in motion and are at l.

(35) a. $\mathit{in\_progress}(\mathit{move}(x, l_0, l)) \multimap [\mathit{stop}(\mathit{move}(x, l_0, l))]\mathit{between}(x, l_0, l)$

b. $\mathit{in\_progress}(\mathit{move}(x, l_0, l)) \multimap [\mathit{reach}(x, l)]\mathit{at}(x, l)$

The Progressive Again

- The progressive of an activity *act* is true if *act* is in progress. The progressive of an accomplishment made up of an activity *act* and an achievement *ach* is true if the progressive of the activity is true. The progressive of an achievement *ach* is true if the progressive of an accomplishment made up of *ach* and its preparatory activity *act* is true:

(36) a. $in\_progress(act) \Rightarrow progressive(activity(act))$

b. $progressive(activity(act))$
$\Rightarrow progressive(accomplishment(activity(act), achievement(ach)))$

c. $preparation(act, ach)$
$\wedge progressive(accomplishment(activity(act), achievement(ach)))$
$\Rightarrow progressive(achievement(ach))$

The Imperfective Paradox Regained

- Moving to someplace is the preparatory activity for reaching that place:

(37) $preparation(move(x, l_0, l), reach(x, l))$

- so (simplifying somewhat) if a car is moving from Edinburgh to Auchterader, the car is reaching Auchterader:

(38) $in\_progress(move(car, Edinburgh, Auchterarder))$
$\models progressive(achievement(reach(car, Auchterarder)))$

- *However the truth of this progressive doesn't depend on the realization of the achievement itself.*

That Man Keats

- *In Disgust of Vulgar Superstition* is a sonnet:

  (39) $sonnet(in\_disgust)$

- Writing is the preparatory activity for completing a sonnet:

  (40) $sonnet(y) \Rightarrow preparation(write(x,y), complete(x,y))$

- If you aren't already writing, then its possible to start:

  (41) $\neg in\_progress(write(x,y)) \Rightarrow possible(start(write(x,y)))$

- If you do so, then writing is in progress:

  (42) $[start(write(x,l_0,l))]in\_progress(write(x,y))$

That Man Keats (Contd.)

- Once you are writing something, its possible either to stop, or to complete it (there are other conditions on the latter that are not directly under your control):

(43) a. $in\_progress(p) \Rightarrow possible(stop(p))$
b. $in\_progress(write(x,y)) \Rightarrow possible(complete(x,y)))$

- Make a plan to get Keats to be completing *In Disgust of Vulgar Superstition*:

(44) $possible(\alpha) \wedge [\alpha] progressive(achievement(complete(keats, in\_disgust)))$

- He should start writing it:

(45) $\alpha = start(write(keats, in\_disgust))$

Cause and Necessity in Dynamic Event Calculus

- The Calculus does not yet really capture the idea of effective planning.
- Consider the effect of adding the following axioms concerning arrival:

(46) a. $[arrive(x)]present(x)$

b. $\neg present(x) \Rightarrow possible(arrive(x))$

the earlier goal (44)

(47) $possible(\alpha) \wedge [\alpha]progressive(achievement(complete(keats, in\_disgust)))$

—-now gives rise to foolish "plans":

(48) $\alpha = start(write(keats, in\_disgust))$

$\alpha = start(write(keats, in\_disgust)); arrive(chapman)$

$\alpha = start(write(keats, in\_disgust)); arrive(chapman); arrive(homer)$

$\ldots$

—although the following is correctly excluded:

(49) $\alpha = start(write(keats, sonnet)); arrive(x); stop(write(keats, sonnet))$

Cause and Contingency (Contd.)

- We need to distinguish the *possible* accessibility relation from the relation *necessary*:

(50) $(possible(\alpha) \wedge [\alpha]necessary(\beta)) \Rightarrow possible(\alpha;\beta)$

- Suppose we now add an axiom saying that anyone else being present causes Keats to stop writing things:

(51) $present(x) \wedge (x \neq keats) \wedge in\_progress(write(keats, y))$
$\Rightarrow necessary(stop(write(keats, y)))$

Cause and Contingency (Contd.)

- The following goal then finds the plan (53a) which makes Keats necessarily stop writing, rather than those like (53b) that merely make it possible:

(52) $possible(\alpha) \wedge [\alpha]necessary(stop(write(keats, y)))$

(53) a. $\alpha = start(write(keats, y)); arrive(x)$
b. $\alpha = start(write(keats, y))$

- Goldman's "Generation" relation can be captured in this way.

(54) $[flip(switch)]necessary(turnon(light))$

The (Solvable) Ramification Problem (cf. Kowalski, Shanahan)

The Problem: *The normal consequences of an action may be overridden.*

Start State (simplified by omitting the individual)

(55) $at(here) \wedge color(green)$

Update consequences of spraying something a color d:

(56) $color(c) \multimap [start(spray(d))]color(d)$

Then we correctly fail to prove the following:

(57) $[start(move(car, here, there)); start(spray(pink));$
$stop(spray(pink)); reach(car, there)]color(green)$

(58) $[start(spray(pink)); start(move(car, here, there));$
$stop(move(here, there)); reach(car, there)]at(here)$

Ramification: Yale Shootings, Stockholm Delivery Scenarios, etc.

- Suppose we need to get a pizza from Edinburgh to Auchterarder using a car. We need to know about the behaviour of containers, namely that if a thing is in a container and the container is at a place then the thing is there too:

(59) $in(x,y) \wedge at(y, place) \Rightarrow at(x, place)$

- Also the preconditions of putting something in a container:

(60) $at(thing, place) \wedge at(container, place)$
$\Rightarrow possible(put\_in(thing, container))$

- Also the updates of putting a thing in a container:

(61) $at(thing, place) \multimap [put\_in(thing, container)]in(thing, container)$

The Auchterarder Ramifying Pizza Delivery Scenario

- A pizza and a car are at Edinburgh:

(62) $at(pizza, Edinburgh), at(car, Edinburgh)$

- How to get a pizza to Auchterarder using the earlier axioms for *move*:

(63) a. $possible(\alpha) \wedge [\alpha]at(pizza, Auchterarder)$

b. $\alpha = put\_in(pizza, car); start(move(car, Edinburgh, Auchterarder))$
$; reach(car, Auchterarder)$

The Auchterarder Ramifying Pizza Delivery Scenario (Contd.)

- Axiom concerning the consequences of throwing things from cars:

  (64) $in(x,y) \multimap [eject(x,y)]$

- As a consequence, any attempt to prove that the pizza is in Auchterarder after the following sequence will fail:

  (65) $\alpha = put\_in(pizza, car); start(move(car, Edinburgh, Auchterarder))$
  $; eject(pizza, car); reach(car, Auchterarder)$

Clock-time

- Its also easy enough to fix up DEC with temporal indices identifying clock times—one possible notation is the following:

(66) $[_{t_0}\alpha]_{t_1}[_{t_1}\beta_{t_2}]_{t_3}P \Rightarrow [_{t_0}\alpha_{t_1};\beta_{t_2}]_{t_3}P$

- We thereby get to prove the following:

(67) $[_0 start(move(here,there))_1; start(spray(pink))_1;$
$stop(spray(pink))_2; reach(car,there)_2]_2 color(pink)$

(68) $[_0 start(spray(pink))_1; start(move(here,there))_1;$
$stop(move(here,there))_2; reach(car,there)_2]_2 at(there)$

Delayed Action

- With this apparatus, just as it is possible to capture the imperfective paradox by the use of $in_progress$ fluents it is also possible to capture delayed action using consequent state fluents.

- Preconditions and updates for dying:

(69) a. $alive \Rightarrow possible(die)$
b. $alive \multimap [die]$

- The consequent state of *drinking arsenic in one's tea at time* t_1 holds if that is what you do at that time:

(70) $[_{t_0} drink(As)_{t_1}]_{t_2} constate(drink(As)_{t_1}) \wedge clocktime(t_2)$

- If you are in that state and four hours has elapsed, you die:

(71) $constate(drink(As)_{t_1}) \wedge clocktime(t_2) \wedge t_2 = (t1 + 4)$
$\Rightarrow necessary(die_{t_2})$

- If we want to achieve a state $\neg alive$ then we have the goal

(72) $possible(\alpha) \wedge [\alpha]necessary(die)_4$

(73) $\alpha = drink(As)_0$

The (Unsolvable) Qualification Problem

You can never state all the conditions of an action.

- I am sitting in my car and I want to drive. I know that to drive the engine must be running, that for it to be running I must switch it on, that the precondition for switching on is to have the key in the lock, and the way to get it there is to put it there, for which the precondition is being there.
- There are several, possibly infinitely many, further preconditions. For example, if there is a potato on the tailpipe of a car, switching on will not result in the engine running. By assumption, this practical precondition is not explicit in the knowledge representation that I am using. What do I do?
- **either:** Some hard scientific reasoning from first principles according to the Hypothetico Deductive Method, a.k.a. **Active Learning**.
- **or:** Kick the tires a.k.a. **Reactive Planning**.

Moral for Temporal Semantics

- Aristotelian/Vendlerian States are fluents, or facts in a databases.
- There are many sorts of states, including progressive, futurate, and consequent states related to aspects.
- Aristotelian/Vendlerian events are instantaneous changes to fluents or facts, accompanied by updates as well as preconditions.
- “Temporal” categories are primarily **Causal** in their ontology: quantification over instants on a timeline plays very little part in defining their semantics.
- Question: is this a Logic at all? (Answer: Yes but that is the least interesting thing about it.)

References

Curry, H. B. and Feys, R. (1958). *Combinatory Logic: Vol. I*. North Holland, Amsterdam.

Köhler, W. (1925). *The Mentality of Apes*. Harcourt Brace and World, New York.

Kowalski, R. (1979). *Logic for Problem Solving*. North Holland, Amsterdam.

Martí-Oliet, N. and Meseguer, J. (1999). Action and change in rewriting logic. In Pareschi, R. and Fronhöfer, B., editors, *Dynamic Worlds*, pages 1–53. Kluwer, Dordrecht.

Moens, M. and Steedman, M. (1988). Temporal ontology and temporal reference. *Computational Linguistics*, 14:15–28.

O'Hearn, P. and Pym, D. (1999). The logic of bunched implications. *Bulletin of Symbolic Logic*, 5:215 – 244.

Pym, D. (1999). On bunched predicate logic. In *Proceedings of the 14th Symposium on Logic In Computer Science (LICS)*. IEEE Computer Society.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA.

Thielscher, M. (1999). From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem. *Artificial Intelligence*, 111:277–299.