Some notes on the Tamworth 1870 accident scenario

To appreciate the attached fragments of an LSD account, you need to have the picture of the accident scenario with signalmen E and H and points labelled A and B in mind. The 'driver' agent here is the driver of the Irish Mail train. (This is the artefact that, in conjunction with the referent - in your imagination! - informs the LSD account.) Note that there are many physical factors, relating to lengths of trains, location and distances between points, feasibility of observation of various kinds etc, about which the artefact gives useful insight but that are *not in* the LSD account. Some of this information is so obvious that we wouldn't dream of identifying it explicitly when we develop the account. For instance, the Irish mail train encounters the points at A before it reaches the points at B (isn't that obvious?) - and trains don't come up the river either (but even that could be different in some surreal computer game).

Standard protocols ("from training manual")

```
agent driver {
        oracle distantsignal
        handle speed_of_train
        protocol
                distant_signal==CAUTION --> speed_of_train = SLOW

}
```

.. this is OK if you don't consider brake failure or icy etc - otherwise:

```
agent driver {
        oracle distantsignal
        handle brakePos
        derivate speed_of_train = f(brakePos, ... )
                                ## '...' on RHS for environmental factors
        protocol
                distant_signal==CAUTION --> brakePos = ON

}
```

A first approximation to the North signalman agent account is:

```
agent signalmanE {
        oracle next_train, homeUPsignal_status, homeMsignal_status, pointA_status
        derivate
                next_kind_of_train is typeof(next_train)
                status_or_next_train is
                        f(next_kind_of_train, schedule)       // status is STOPPING or THROUGH
        handle homeMsignal_status, pointsA_status
        derivate homeUPsignal_status = if (homeMsignal_status = CLEAR) then DANGER else CLEAR;
        protocol
                (next_kind_of_train == STOPPING)
                        --> pointsA_status = TO_PLATFORM; homeMsignal_status = DANGER;
                (next_kind_of_train == THROUGH)
                        --> pointsA_status = TO_MAIN; homeMsignal_status = CLEAR;
                ....
}
```

Note that this doesn't permit both home signals to be set at DANGER. This is a serious flaw, not only because this is an essential state from the point of view of safety (imagine e.g. if the whole station were to be on fire), but also since both home signals at DANGER is a precondition for it to be possible to switch the points at A. This means that the derivate is really an unsatisfactory simplification: indeed, the positions of the home signals can *at any time* be set at DANGER.

A better account captures the idea that the signalman follows two different protocols to set up the points and signals for the arrival of stopping and through trains. Note that there is an invariant relationship (guaranteed by the interlocking mechanism) whereby `homeMsignal_status == CLEAR => pointsA_status == TO_MAIN`. (It is not mechanically possible to set the home mainline signal to clear unless the points at A are correctly set, and once the signal has been lowered, it is mechanically impossible to change the points at A.) On that basis, the privilege

```
(pointsA_status == TO_MAIN) and (homeMsignal_status = CLEAR) --> homeMsignal_status = DANGER
```

can be 'equivalently' expressed as:

```
(homeMsignal_status = CLEAR) --> homeMsignal_status = DANGER.
```

A similar observation applies to setting the home platform signal to danger ("the signalman can change a signal at CLEAR to DANGER at any time"). Notice here that in accepting that such relationships cannot be otherwise we are taking steps towards a view of the mechanical technology as 'completely reliable' that is part of the much larger move towards "taking mechanisms for granted" that is characteristic of classical computing. There is probably some justification for this, when we consider that, once the technology was mature, mechanical failure of interlocking mechanisms was much less likely than human error to be at the root of railway accidents.

```
agent signalmanE {
        oracle next_train, homeUPsignal_status, homeMsignal_status, pointA_status
        derivate
                next_kind_of_train is typeof(next_train)
                status_or_next_train is
                        f(next_kind_of_train, schedule)       // status is STOPPING or THROUGH
        handle homeMsignal_status, pointsA_status
        protocol
                (next_kind_of_train == STOPPING) --> setUPsignalclear(),
                (next_kind_of_train == THROUGH) --> setMsignalclear();
                ....
}

agent setUPsignalclear {
        handle homeMsignal_status, pointA_status, homeUPsignal_status
        protocol
                (pointsA_status == TO_MAIN) and (homeMsignal_status = CLEAR)
                        --> homeMsignal_status = DANGER;
                (pointsA_status == TO_MAIN) and (homeMsignal_status = DANGER)
                        --> pointsA_status = TO_PLATFORM;
                (pointsA_status == TO_PLATFORM) and (homeMsignal_status = DANGER)
                        --> homeUPsignal_status = CLEAR
}

agent setMsignalclear {
        handle homeMsignal_status, pointA_status, homeUPsignal_status
        protocol
                (pointsA_status == TO_PLATFORM) and (homeUPsignal_status = CLEAR)
                        --> homeUPsignal_status = DANGER;
                (pointA_status == TO_PLATFORM) and (homeUPsignal_status = DANGER)
                        --> pointsA_status = TO_MAIN;
                (pointsA_status == TO_MAIN) and (homeUPsignal_status = DANGER)
                        --> homeMsignal_status = CLEAR
}
```

It's possible to regard the `setUPsignalclear` and `setMsignalclear` agents as roles for the signalman. This would be most plausible if achieving the "neutral" / "safe" state of readiness, when both home signals are set to danger, was acknowledged as a separate role (introducing a "`setupsignalssafe`" agent). Note further that the notion that the next train to arrive is a stopping train is merely a motivation for the intelligent and conscientious signalman to prepare the station for its arrival. As the circumstances of the accident show, there is nothing to oblige the signalman to adopt what is necessarily the most appropriate role (in principle, the signalman could decide to set both signals to DANGER and take the day off). Neither can the timing of the signalman's actions be considered in isolation from the real situation: as the accident revealed, in some circumstances, setting the home signals to DANGER was not an adequate precaution against disaster, as there was insufficient time to bring the train safely to a stop. And quite obviously, the changing of the status of signals and points is not something that could be synchronised arbitrarily with the passing of a train etc.

The above discussion is relevant to one aspect of the accident scenario, though it doesn't take account of the role of the gong and disc protocol within the operational framework (a rather controversial issue in the BoT report, since E and H gave conflicting evidence about the communication procedures followed). To account for another very significant aspect of the situation: the failure of E's watch, we also need to refine the idea of `next_train` as a function of the time (`timenow`) and the schedule:

```
agent signalmanE {
        oracle
                timenow
                next_train, home1signal_status, home2signal_status, pointB_status
        derivate
                next_train is f(timenow, schedule)
                next_kind_of_train is typeof(next_train)
                status_or_next_train is
                        f(next_kind_of_train, schedule)       // status is STOPPING or THROUGH
        handle homeMsignal_status, pointsA_status
        protocol
                (next_kind_of_train == STOPPING) --> setUPsignalclear(),
                (next_kind_of_train == THROUGH) --> setMsignalclear();
```

```
                ....
}
```

... where we ideally expect the signalman to know the current time:

```
agent watchH{
        state timenow
        derivate timenow is actualtime

}
```

Some fairly complex derivate or maybe procedural component is needed to determine whether next_train is STOPPING or not (probably routinely defined by a dependency if the schedule is being followed in normal operation). (The BoT report refers to the fact that "Notice by telegraph was, indeed, received at the Tamworth station of the approach of the Irish mail train in advance of the goods train; but this notice was not communicated to the signalman ...".)

Anomalous situations

The above discussion raises the issue of non-standard contexts. Already have such an issue to consider in connection with the unreliable watch. Deal with this by attributing different roles to the watch - the classification of observables becomes dependent on context.

For a watch, have two contexts - watch can be going or stopped.

```
agent watchH {
        state timenow, status (= GOING, STOPPED)
        agent watchHgoing {
                state           error (= +/- n seconds )
                derivate        timenow is time + error
                                        ## watch could be slow or fast
                                        LIVE is (status == GOING)
        }

        agent watchHstopped {
                state           timenow = time0
                derivate        LIVE is (status == STOPPED)
        }
}

agent watchHmech {
        handle status
        oracle powersourceOK
        derivate status is if (powersourceOK) then GOING else STOPPED
}
```

With a clockwork clock, might want to take into account the responsibility for winding it - and so invoke agency of signalmanH again here.

Actually (of course!) the roles of the signalmen are affected by whether there is or isn't a train present at the station. As in the case of the Railway Arrival-Departure Animation, it is necessary to instantiate some observables that are only meaningful if and when there *is* a train in the station.

```
agent train_at_station {
        state   position
                        speed
                        platform
        derivate
                        LIVE is f(position, ...)
                        ## "at station" might mean between points A and B, or perhaps
                        ## should mean between points A and the distant signal
                        ## cf observable 'engaging' that features in the Railway Station Animation
}
```

Can now frame roles for the signalmen according to whether trains are present or not etc., bearing in mind that there may be more than one train present at the station at any one time. Something more complex but along the lines of the account of watch roles is what's needed here.

There are aspects of the analysis of the accident scenario that provoke reflections about possible actions that might not be part of the training manual. Perhaps signalman H could have diverted the express back on to the main track for instance, by shifting the points at B. (As the BoT report suggests, such a diversion might have worked in principle since the train successfully negotiated the points at A at a speed estimated at 45 mph, and was eventually travelling at about 15 mph. Even at that speed the momentum of a train was quite sufficient to cause catastrophe given the strength of the buffers and the proximity to the river bank etc.)

How could we model such an interventionary action on the part of H?

```
agent signalmanH {
        oracle  is_train_at_station
                        train_at_station
                        pos_train_at_station
                        speed_train_at_station
                        kind_train_at_station
                        ## could make into state observables of "train_at_station" agent
                        ## or better "train" agent "at station" here
        handle  pointsB_status
        privilege
                (kind_train_at_station==THROUGH) && (pos_train_at_station==PLATFORM_LINE)
                        --> pointsB_status = ...
}
```

Note the issues here about what signalmanH can observe. Does he know that the train on the PLATFORM_LINE is a THROUGH train: from the schedule? because it's going unusually fast? because he recognises that it is the Irish Mail? etc.

Might also ask - is he *supposed* to be able to observe such things? - i.e. is it part of his job description? If it were, then this would be a flaw in the railway system conception, since a train going the other way could interrupt his observation. A related question is: in what ways was the telegraph expected to enhance the communication between the two signalmen and improve upon the pre-existing gong and disc mechanisms?

The above discussion indicates how developing an LSD account can serve a purpose that is primarily provocative rather than functional, stimulating us to ask questions that might otherwise not occur to us. In general, the development of an account and an EM artefact may usefully proceed together. Such an exercise could be particularly useful if carried out in conjunction with a close reading of the BoT accident report at http://www.railwaysarchive.co.uk/documents/BoT_Tamworth1870.pdf, which includes additional detail about the precise distances between key points, the weather conditions, the gradients and slipperiness of the track, the composition of the train, including weights of the component vehicles, condition of the brakes, and the eye-witness reports from railway personnel both on and off the train.