Event-driven Rules in AutoHAN

John Bates

This is some background material on rules and some examples for the AutoHAN proposal. It goes into very little details about what is going on but I hope it is illustrative. It all relies on an underlying distributed event system, involving a paradigm of registration and notification. We have services for event brokering and a scheme for event class hierarchies (e.g. imagine a generic event class `PointAt` and sub-classes `PointedAtTV` and `PointedAtPhone`). AutoHAN components are functionally equivalent to distributed objects; each AutoHAN component generates events and is controllable remotely.

1 Rule Syntax

Event driven rules can be specified, of the form

```
rule <name> = <event expression>; {<list of invocations>}
```

An event expression specifies one or a combination of events which will cause this rule to *fire*. A list of invocations on any of the objects in the application context, are performed if the rule fires. The rule name represents an event which occurs when the rule fires. It can be incorporated into the event expressions of other rules.

1.1 Basic Operators

`-> evexpr`: This statement specifies that the expression `evexpr` must occur after any previous expressions.

`-> evexpr1 - evexpr2`: This composite event expression is detected if the expression `evexpr1` occurs after any previous expressions `without` any intervening occurrences of `evexpr2`.

`=> evexpr1`:

`=> evexpr1 - evexpr2`: In addition to the single arrow (`->`), the double arrow (`=>`) can be used. The single arrow indicates interest in only the first relevant occurrence of an event. The double arrow indicates interest in every occurrence of an event .

`evexpr1 | evexpr2`: This expression is detected if either `evexpr1` **or** `evexpr2` is detected. These must be monitored for concurrently since each expression may be temporally complex.

`evexpr1 ^ evexpr2`: This expression is detected if `evexpr1` **and** `evexpr2` have been detected, in any temporal order.

1.2 Time Comparison Operators

`evexpr<t`: This expression is detected if e occurs within time period t.

`evexpr>t`: This expression is detected if e does not occur within time period t.

`evexpr,t`: This expression is detected if evexpr has not occurred for time period t.

`evexpr1,evexpr2,t`: This expression is detected if evexpr2 occurs. It is rejected if e does not repeatedly occur within time period t.

2 Implementing Rule Monitoring

A service to support the monitoring of composite event rules have been constructed. Events are notified to such a service, which updates its state accordingly. If the entire expression is detected, it will notify a specified client.

Composite event rules offer the best of both worlds: High-level declarative specification and efficient underlying implementation. Monitoring for event expressions is achieved by constructing non-deterministic finite state machines. If a relevant event occurs, the machine can change state. States are either accepting, which indicate the expression has been detected, or non-accepting. The current state is marked with a token. If an expression is concerned with every instance of an event (marked as `=>` transitions in an expression) then multiple tokens may be active inside the same machine, each corresponding to a different event matching sequence.

When attempting to order events from different distributed sources, one must take into account delays such that the order in which one receives events may not be the order in which they occurred. The Cambridge event work takes this into account by allowing an application to specify how long it is prepared to keep events before consuming them, pending the arrival of earlier occurrences. Services use regular heartbeat events to assure the composite event service that there are no outstanding events.

3 Examples

In these examples, we are assuming the home environment is equipped with:

- *Ubiquitous endpoints*
  - *AutoHAN home entertainment* – TV, hi-fi, video.
  - *Live endpoints* – Cameras, microphones everywhere.
  - *AutoHAN comms* – phone, videophone, Internet.
- *Location monitoring equipment* – active badges, active doors, video motion detection.
- *AutoHAN central heating*
- *AutoHAN kitchen* – cooker, microwave, washing machine.

3.1 Automatic TV Mute

The television volume is required to be muted if the telephone or the door bell rings.

```
rule tv-mute => TVOn(TVID) ->
                (telephone-rings(PHONEID) | door-bell(DOORID))
                - TVOff(TVID);
{
    TVID.AudioOut.mute();
}
```

This expression starts monitoring when a television is switched on. It monitors independently for each television (due to the => transition). The CE accepts when a telephone or door-bell rings without the television being switched off first.

3.2 Security Monitoring

The ability to monitor the environment enables security applications to be implemented. Each home dweller wears an Active Badge. The absence of badge events gives a strong indication that no one is present in the building. Consider the following CE expression which implements the request "...if no family member has been seen in a home for 10 minutes then send in a security guard....":

```
rule security -> ABadge(LastPerson, Home, LastLocation),600.0 =>
                 (Motion(Home, Location, Camera) | Door(Home, Location, Status)) -
                 ABadge(Anyperson, Home, AnyLocation);
{
    Control.Security.Alert(Home, Location);
}
```

This expression initially monitors for badge events. After a period of 600 seconds (10 minutes) in which no badge events have occurred anywhere in the home, video motion and door activity are monitored for. The CE accepts if either one of these events occurs without a badge event. The application then informs a security control centre that unauthorised movement has been seen within the home.

This rule can either be monitored for locally, in the home, or remotely. A security control centre can monitor ALL the homes in its vicinity using this one rule.

3.3 Mobile Home Objects

In other research, we have developed mobile multimedia objects – media endpoints that can move location without loss of state and can manage hand-over of communications channels. These can be used in the HAN. For example, in conjunction with infra-red wands. Currently active TV sessions can be moved between screens by dragging them through the air using a wand:

```
rule tv-move => PointedAtTV(TVID,SessionID) -> DragAndDrop(TVID2) ;
{
    Session(SessionID).Move(TVID,TVID2);
}
```

Because this rule has an initial spawning state (=>) it applies to any TV in the house.

Additionally, we can specify rules so that the music we are listening to follows us around the house:

```
rule music-follow =>  MovedTo(Person,Location) ->
                      MusicOn(Location,Track,Session) ->
                      MovedTo(Person,Location2) ;
{
    HiFi.Audio(Location2).switch(Track,Session)
    HiFi.Audio(Location).clear();

}
```

3.4 Intelligent Comfort

It is desirable for equipment to turn itself on to aid human comfort. An example is "if the kids come home, turn on the central heating and get their dinner ready – it's in the microwave; 20 minutes on high please. If they email their arrival time then turn it on 20 minutes before they expect to get in":

```
rule message-from-kids ->
             EmailCommand('KidsAccount', 'Arriving', ExpectedTime);
{
    RegisterTimerEvent('TimeForTea', (ExpectedTime-20.0));
}

rule kids-home -> ((EnterHome('John') ^ EnterHome('Rebekah')) |
                  TimeforTea)
{
    Kitchen.Microwave.On('High', 20.0);
    CentralHeating.On();
}
```

Another example is "please record Coronation Street tonight". A simple way in which this can be done is, for example, by pointing at the video with a wand and saying "Coronation Street". The following rule generalises this for any television programme, using speech recognition and a lookup in the *Radio Times*:

```
rule set-video => PointedAtVideo(VideoID,SessionID) ->
                  SpeechDetected(Programme) ->
                  state RadioTimes(Programme, Date, Time, Channel) ;
{
    SetVideo(Programme, Date, Time, Channel);
}
```

3.5 A Context-Aware Telephone Directory

A convenient application is one which displays a telephone directory on the nearest display when a telephone handset is picked up. To facilitate this, consider the following CE expression:

```
rule phone-dir => (PhonePickedUp(PhoneID) ->
                   (state Proximity('PHONE', PhoneID, '2.0', 'TV', TVID) |
                    PointedAtTV(TVID))
                   - PhoneDropped(PhoneID)) ->
                  state ScreenActivity(TVID, Name, Userid) > 60.0;
{
    Applications.PhoneDir.show(TVID);
}
```

This firstly looks for any telephone that has been picked-up. It then waits to see if either a TV is within $2m$ of the telephone or a screen is selected using a wand. Finally, a state lookup is made to determine whether the equipment has been idle for at least 60 seconds. The CE will reject if, after the initiating phone 'pickup' event, the same phone's handset is put back down.