Noise, Non-Determinism and Spatial Uncertainty

Murray Shanahan

Department of Computer Science,
Queen Mary and Westfield College,
Mile End Road, London E1 4NS, England.
mps@dcs.qmw.ac.uk

Abstract

This paper presents a logical account of sensor data assimilation in a mobile robot, based on abduction. Unlike previous work, the present formulation handles sensor noise as well as motor noise. In addition, it incorporates two significant technical advances. The use of determining fluents to deal with non-determinism obviates the need for a special form of abduction, and the use of uncertain object boundaries alleviates a problem with multiple explanations.

Introduction

In [Shanahan, 1996], a logical characterisation of robot sensor data assimilation via abduction is presented. The methodology used in that paper, as well as the present work, comprises the following three steps. First, design a generic logical formalism for representing and reasoning about action, change, space and shape. Second, use this formalism to build a theory of the robot's relationship to the world, that is to say, a theory describing the effect of the robot's actions on the world and the impact of the world on the robot's sensors. Third, view sensor data assimilation as abduction with this theory. The role of abduction is to hypothesise a collection of objects whose shapes and locations are sufficient to explain the robot's sensor data. Any algorithm for deployment on the actual robot which is provably correct with respect to this abductive characterisation is, in effect, a map-building algorithm.

The key features of the informatic situation confronting a mobile robot are *incompleteness*, due to the robot's limited window on the world, and *uncertainty*, due to noise in its sensors and motors. The topic of noisy motors is discussed at length in [Shanahan, 1996]. In that paper, noise is considered as a kind of non-determinism, and a consistency-based form of abduction is used. But the treatment of incompleteness and uncertainty is unsatisfactory for a number of reasons. Chief among these is the profligacy of the collection of explanations licensed by abduction for a given stream of sensor data. Other reasons include the difficulty of performing, or even proving properties of, consistency-based abduction.

This paper sets out to remedy the deficiencies of the earlier work through the provision of a cleaner and more versatile logical account of sensor data assimilation. To deal with the multiple explanations problem, objects are represented with *uncertain boundaries*. To cope with noise, the method of *determining fluents* replaces the use of non-deterministic trajectories in [Shanahan, 1996]. This permits the use of standard, as opposed to consistency-based, abduction.

Finally, the robot that formed the basis of the earlier work was equipped with a very simple collection of sensors, namely three bump switches. The robot under discussion here has a suite of eight infra-red proximity sensors, supplying much richer sensory input. This forces us to confront the issue of sensor noise. (There is no sensor noise as such with a bump switch.) Accordingly, the present paper, unlike [Shanahan, 1996], supplies a treatment of sensor noise as well as motor noise.

1 Representing Action

The first step in the methodology outlined in the introduction is the development of a generic formalism for representing and reasoning about action, change, space and shape. This section concerns the part of the formalism for reasoning about action and continuous change, which is based on the circumscriptive event calculus [Shanahan, 1997]. The same formalism is employed in [Shanahan, 1996]. Because this material is presented in considerable detail elsewhere, the description here will be kept very brief.

A many sorted language is assumed, with variables for *fluents*, *actions* (or *events*), and *time points*. We have the following axioms, whose conjunction will be denoted CEC. Their main purpose is to constrain the predicate HoldsAt. HoldsAt(f,t) represents that fluent f holds at time t. All variables are universally quantified with maximum scope, unless otherwise indicated.

$$\text{HoldsAt}(f,t) \leftarrow \text{Initially}_P(f) \wedge \neg \text{Clipped}(0,f,t) \quad \text{(EC1)}$$

$$\neg \text{HoldsAt}(f,t) \leftarrow \text{Initially}_N(f) \wedge \neg \text{Declipped}(0,f,t) \quad \text{(EC2)}$$

$$\text{HoldsAt}(f,t2) \leftarrow \text{Happens}(a,t1) \wedge \text{Initiates}(a,f,t1) \wedge t1 < t2 \wedge \neg \text{Clipped}(t1,f,t2) \quad \text{(EC3)}$$

$$\neg \text{HoldsAt}(f,t2) \leftarrow \text{Happens}(a,t1) \wedge \text{Terminates}(a,f,t1) \wedge t1 < t2 \wedge \neg \text{Declipped}(t1,f,t2) \quad \text{(EC4)}$$

$$\text{Clipped}(t1,f,t2) \leftrightarrow \exists\, a,t\ [\text{Happens}(a,t) \wedge t1 < t < t2 \wedge [\text{Terminates}(a,f,t) \vee \text{Releases}(a,f,t)]] \quad \text{(EC5)}$$

Declipped(t1,f,t2) ↔ (EC6)
  ∃ a,t [Happens(a,t) ∧ t1 < t < t2 ∧
    [Initiates(a,f,t) ∨ Releases(a,f,t)]]

HoldsAt(f2,t2) ← (EC7)
  Happens(a,t1) ∧ Initiates(a,f1,t1) ∧ t1 < t2 ∧
    t2 = t1 + d ∧ Trajectory(f1,t1,f2,d) ∧
      ¬ Clipped(t1,f1,t2)

A particular *domain* is described in terms of Initiates, Terminates, Releases, and Trajectory formulae. Initiates(a,f,t) represents that fluent f starts to hold after action a at time t. Conversely, Terminates(a,f,t) represents that f starts not to hold after action a at t. Releases(a,f,t) represents that fluent f is no longer subject to the common sense law of inertia after action a at t. The Trajectory predicate is used to capture continuous change. Trajectory(f1,t,f2,d) represents that f2 holds at time t+d if f1 starts to hold at time t.

A particular *narrative* of events is described in terms of Happens and Initially formulae. The formulae $\text{Initially}_P(f)$ and $\text{Initially}_N(f)$ respectively represent that fluent f holds at time 0 and does not hold at time 0. Happens(a,t) represents that action or event a occurs at time t.

In rough terms, if E is a domain description and N is a narrative description, then the frame problem is overcome with circumscription, by considering,

CIRC[N ; Happens] ∧
  CIRC[E ; Initiates, Terminates, Releases] ∧ CEC.

However, care must be taken when domain constraints and triggered events are included. The former must be conjoined to CEC, while the latter are conjoined to N. A detailed account of this formalism and the accompanying solution to the frame problem can be found in [Shanahan, 1997].

2 Space

Continuing with the first step of the methodology outlined in the introduction, the subject of this section is the representation of space and shape in the required generic formalism. In the present paper as in previous work, the plane $\mathbb{R}^2$ formed the basis of this aspect of the formalism. In previous work, objects in the robot's workspace occupied regions, which were considered to be open, path-connected subsets of the plane [Shanahan, 1996]. The region occupied by an object was defined using set membership.

The formulation in [Shanahan, 1996] has a serious problem with *multiple explanations*. The incompleteness and uncertainty of the common sense informatic situation entail that many slightly different configurations of objects can be responsible for the same set of sensor data. In [Shanahan, 1996], the level of abstraction at which spatial occupancy is represented is too low to admit a convenient description of the set of all possible configurations of objects which can explain a given stream of sensor data.

In the formalism of the present paper there is no ontological commitment to regions at all. Instead, the spatial ontology comprises *curves* and *points*, which are each assigned their own sort in the language. Curves are open and directed. The sort of curves includes the sub-sort of (straight) *lines* of finite, non-zero length.

In the sequel, we will confine our attention to interpretations in which these sorts have their intuitive meanings in $\mathbb{R}^2$. The formalism also includes the following spatial predicates and functions, for which the same assumption is made. The formula On(p,c) represents that the point p is on the curve c, the formula FromTo(c,y1,y2) represents that the line c has start point y1 and end point y2, the term Lng(c) denotes the length of line c, and the formula Bng(c) denotes the bearing of line c relative to North (a South–North line has bearing 0).

The robot's map of the world will be represented as a graph of lines joining significant locations. In general, the start and end points of these lines will only be known to within certain bounds (similar techniques are used by Davis [1986]). The approach taken is very close to that of [Kuipers, *et al.*, 1993], and indeed the present paper can be thought of as supplying a logical reconstruction of certain aspects of Kuipers' work.

3 The Robot's Effect on the World

Having designed a generic formalism for representing and reasoning about action, change, space and shape, the next step is to use that formalism to build a theory of the robot's relationship to the world. This theory must describe the effect of the robot's actions on the world, and the effect the world has on the robot's sensors. This section concerns the effect of the robot's actions on the world.

The subject of this paper's formalisation is the Khepera robot, a low-cost miniature robot platform with two drive wheels and a suite of eight infra-red proximity sensors around its circumference, as depicted on the left of Figure 1.

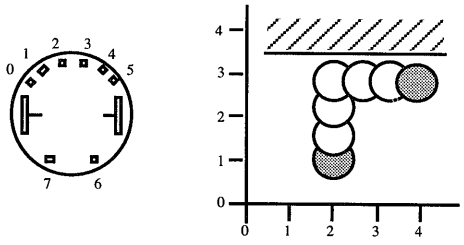

Figure 1: The Khepera Robot (left), Meeting a Wall (right)

Although the Khepera can move in a curve, we will consider a repertoire of just three actions: Go, Rotate, and Stop. When it performs a Go action, the robot starts to move forwards. When it performs a Rotate action, the robot starts to turn on its axis. The Go action arrests any rotational motion, the Rotate action arrests any translational motion, and the Stop action arrests both translational and rotational motion.

For a variety of reasons, such as wheel slippage or an uneven work surface, the robot's motors must be considered "noisy". Because of this noise, the Go and Rotate actions have non-deterministic effects. In [Shanahan, 1996], this

non-determinism was incorporated into the robot's trajectory as it moved forwards. The present paper takes a somewhat different approach, which I will now outline.

The method for dealing with non-determinism adopted here involves the use of *determining fluents* (see [Shanahan, 1997, Chapter 15]), which are related to the *noise terms* introduced for the same purpose by Poole [1995]. With the method of determining fluents, actions with non-deterministic effects are transformed into actions with deterministic effects. The trick is simply that the outcome of an action with non-deterministic effects is made to depend on a fluent, called a determining fluent, which doesn't appear elsewhere in the formalisation, and whose value at the time of the performance of the action is unknown. This treatment of non-determinism obviates the need for a special form of abduction, such as the consistency-based form of abduction used in [Shanahan, 1996]. Instead, as we'll see later on, the determining fluents are simply made abducible.

In the formalisation of the Khepera robot, the Go and Rotate actions have non-deterministic effects: the velocity of the robot after a Go action is only known within certain bounds, and the angular velocity of the robot after a Rotate action is only known within certain bounds. In what follows, the constant V denotes the robot's median velocity, measured in robot radii per unit of time, and the constant W denotes the robot's median angular velocity in degrees per unit of time. The formalisation also uses the constants ε_V and ε_W. After a Go action, the robot's velocity is $V \pm \varepsilon_V$. Similarly, after a Rotate action, the robot's angular velocity is $W \pm \varepsilon_W$.

To capture the effect of the Go and Rotate actions using determining fluents, the functions VelNoise and RotNoise are introduced. These fluents perturb the robot's (angular) velocity at the outset of a period of motion. If a Go action is performed at time t, then the term VelNoise(t) denotes the difference e, where $-\varepsilon_V \leq e \leq \varepsilon_V$, between the robot's actual velocity after t and the median velocity V. Similarly, if a Rotate action is performed at time t, the term RotNoise(t) denotes the difference e, where $-\varepsilon_W \leq e \leq \varepsilon_W$, between the robot's actual angular velocity after t and the median angular velocity W.

We have the following Initiates and Terminates formulae, which introduce the two new fluents Moving and Turning. The fluent Moving(v) holds when the robot is moving at velocity v, and the fluent Turning(w) holds when the robot is rotating with angular velocity w.

Initiates(Go,Moving(V+VelNoise(t)),t) (E1)

Initiates(Rotate,Turning(W+RotNoise(t)),t) (E2)

The following two formulae constrain the values of VelNoise and RotNoise to fall within the required range.

$-\varepsilon_V \leq \text{VelNoise}(t) \leq \varepsilon_V$ (B1)

$-\varepsilon_W \leq \text{RotNoise}(t) \leq \varepsilon_W$ (B2)

Next, we need to define the continuous variation that takes place in the robot's bearing and location while, respectively, the Turning and Moving fluents hold. The fluent Facing(r) holds when the robot's compass bearing is r degrees relative to North. The fluent Location(y) holds when the robot's centre is at point y. The continuous variation in the Location and Facing fluents is captured by the following Trajectory formulae.

$$\text{Trajectory}(\text{Moving}(v),t,\text{Location}(y1),\frac{\text{Lng}(c)}{v}) \leftarrow \quad \text{(T1)}$$
$$\text{HoldsAt}(\text{Facing}(r),t) \wedge \text{HoldsAt}(\text{Location}(y2),t) \wedge \text{FromTo}(c,y2,y1) \wedge \text{Bng}(c,r)$$

$$\text{Trajectory}(\text{Turning}(w),t,\text{Facing}(r+d.w),d) \leftarrow \quad \text{(T2)}$$
$$\text{HoldsAt}(\text{Facing}(r),t)$$

After a Go action, the Location fluent is no longer subject to default persistence. Similarly, after a Rotate action, the Facing fluent is no longer subject to default persistence.

Releases(Go,Location(y),t) (E3)

Releases(Rotate,Facing(r),t) (E4)

The following Initiates and Terminates formulae capture the effects of the Stop action.

Terminates(Stop,Moving(v),t) (E5)

Terminates(Stop,Turning(w),t) (E6)

$$\text{Initiates}(\text{Stop},\text{Location}(y),t) \leftarrow \quad \text{(E7)}$$
$$\text{HoldsAt}(\text{Moving}(v),t) \wedge v > 0 \wedge \text{HoldsAt}(\text{Location}(y),t)$$

$$\text{Initiates}(\text{Stop},\text{Facing}(r),t) \leftarrow \quad \text{(E8)}$$
$$\text{HoldsAt}(\text{Turning}(v),t) \wedge w \neq 0 \wedge \text{HoldsAt}(\text{Facing}(r),t)$$

Taken in conjunction with the event calculus axioms of Section 1, the formulae above can be used deductively for prediction (temporal projection). Given a description of a sequence of robot actions, the robot's location at any given time can be predicted within the tolerances allowed by the non-determinism.

Let's take a look at an example. Let M be the conjunction of the following formulae.

InitiallyP(Location(L0)) InitiallyP(Facing(R0))

Let N be the conjunction of the following formulae.

Happens(Go,1000) Happens(Stop,3000)

Happens(Rotate,4000) Happens(Stop,5000)

Happens(Go,6000)

These actions are illustrated on the right of Figure 1.

In addition, we require some uniqueness-of-names axioms. The following will suffice for now, although we will adopt a different set in the next section.

UNA[Go, Rotate, Stop] (3.1)

UNA[Moving, Turning, Location, Facing] (3.2)

Let E be the conjunction of (E1) to (E8). Let B be the conjunction of,

- the event calculus axioms CEC,
- the background axioms (B1) to (B4),
- the Trajectory formulae (T1) and (T2), and
- the uniqueness-of-names axioms (3.1) and (3.2).

Following the prescription for overcoming the frame problem set out in Section 1, we're interested in the logical consequences of the following formula, which will be denoted Σ.

CIRC[N ; Happens] ∧
CIRC[E ; Initiates, Terminates, Releases] ∧ M ∧ B

The logical consequences of Σ are expected to describe the path of the robot. However, the non-deterministic nature of the robot's actions, reflecting the fact that its motors are noisy, means that Σ doesn't fix the exact location of the robot at any time after its first Go action. Rather, we should expect Σ to yield consequences of the form Δ ← Γ, where Δ describes the robot's location to the bounds within which it can be known, and Γ constrains the determining fluents VelNoise and RotNoise accordingly. Δ will characterise the robot's location in terms of a graph of curves and locations corresponding to the path the robot followed to get there. The following proposition gives an example.

Proposition 3.3. Let Δ denote the conjunction of the following formulae, in which c1, y1, c2, and y2 are free.

$$\begin{aligned}&\text{FromTo}(c1,L0,y1) \wedge \text{Bng}(c1) = R0 \wedge \\ &\quad 2000.(V - \varepsilon_V) \leq \text{Lng}(c1) \leq 2000.(V + \varepsilon_V) \wedge \\ &\quad\quad \text{HoldsAt}(\text{Location}(y1),3000)] \\ &\text{FromTo}(c2,y1,y2) \wedge 1000.(W - \varepsilon_W) \leq \\ &\quad \text{Bng}(c2) - \text{Bng}(c1) \leq 1000.(W + \varepsilon_W) \wedge \\ &\quad\quad 2000.(V - \varepsilon_V) \leq \text{Lng}(c2) \leq 2000.(V + \varepsilon_V) \wedge \\ &\quad\quad\quad \text{HoldsAt}(\text{Location}(y2),8000)\end{aligned}$$

Let Γ denote the conjunction of the following formulae, in which c1, y1, c2, and y2 are also free.

$$\text{VelNoise}(1000) = \frac{\text{Lng}(c1)}{2000} - V$$

$$\text{RotNoise}(4000) = \frac{\text{Bng}(c2) - \text{Bng}(c1)}{1000} - W$$

$$\text{VelNoise}(6000) = \frac{\text{Lng}(c2)}{2000} - V$$

We have $\Sigma \models \exists\ c1,y1,c2,y2\ [\Delta \leftarrow \Gamma]$.

Proof. See full paper □

4 The Effect of the World on the Robot

The formulae of the previous section describe the effect of the robot's actions on the world. Now we need to characterise the way the robot's interactions with the world effect its sensors. Once again, noise is a big issue. This time the noise is in the robot's sensors, which deliver uncertain information about the world. As before, noise is considered as non-determinism.

Each of the Khepera's infra-red proximity sensors supplies an unbroken (but discretely sampled) stream of values between 0 and 1023. A high value suggests the proximity of an obstacle, but unsurprisingly no straightforward functional correspondence exists between the value of a sensor and the distance to the nearest object. As well as being subject to random fluctuations and spikes, the value delivered by a sensor at any time depends on the sizes, shapes, orientations and reflective properties of the objects within its range. So the question arises of how to extract useful distance information from the sensor signal.

A standard way would be to use a Gaussian and/or Kalman filter. Although benefit can still be derived from pre-filtering sensor signals, the approach taken in the present paper places the burden of interpreting sensor data at a higher level. To begin with, raw sensor data is transformed into a sequence of sensor events. Two types of sensor event are incorporated into the formalisation: LatchOn and LatchOff, which will be parameterised by the identifier of the sensor in question. Sensor data assimilation is then performed through abduction, whose task is to construct an explanation of these sensor events in terms of a map of the obstacles in the robot's workspace.

Intuitively, a LatchOn event indicates that an object has been encountered, while a LatchOff event indicates that the robot has found free space. An obvious way to implement this is to trigger a LatchOn event whenever the sensor signal exceeds a threshold, and a LatchOff event whenever it dips below that threshold. However, with this method, the presence of random noise in the sensor signal gives rise to the possibility of a flurry of LatchOn and LatchOff events when the signal is close to the threshold value. Accordingly, in the present approach, sensor events are associated with two threshold values δ1 and δ2, where δ1 > δ2. When a sensor value exceeds δ1 then a LatchOn event occurs. Conversely, when the sensor value dips below δ2, a LatchOff event occurs.

The fluent High holds when the sensor signal is greater than δ1, and the fluent Low holds when the sensor signal is less than δ2. Like the LatchOn and LatchOff events, these fluents are parameterised by the identifier of the sensor in question. Sensors will be grouped into pairs whose values will be aggregated, and our main concern will be with the pair 0 and 1, which will be called the left sensor, and the pair 4 and 5, which will be called the right sensor (see Figure 1). The following two clauses constrain the High and Low fluents for the left sensor.

$$\begin{aligned}&\text{Holds}(\text{Low}(\text{Left}),t) \leftarrow \qquad \text{(B3)} \\ &\quad \neg\, \exists\ y,r,c,x\ [\text{HoldsAt}(\text{Location}(y),t) \wedge \\ &\quad\quad \text{HoldsAt}(\text{Facing}(r),t) \wedge \\ &\quad\quad\quad \text{HoldsAt}(\text{Boundary}(x,c),t) \wedge \\ &\quad\quad\quad\quad F_{sen}(y,r-45,c) + \text{SenNoise}(\text{Left},t) > \delta 2] \\ &\text{HoldsAt}(\text{High}(\text{Left}),t) \leftarrow \qquad \text{(B4)} \\ &\quad \text{HoldsAt}(\text{Location}(y),t) \wedge \text{HoldsAt}(\text{Facing}(r),t) \wedge \\ &\quad\quad \text{HoldsAt}(\text{Boundary}(x,c),t) \wedge \\ &\quad\quad\quad F_{sen}(y,r-45,c) + \text{SenNoise}(\text{Left},t) > \delta 1]\end{aligned}$$

The first of these formulae describes the conditions under which the value of the left sensor falls below δ2 at time t. First, there has to be an object whose boundary bears a suitable relation to the position and direction of the left sensor at t. (The term Boundary(x,c) denotes a fluent which holds if the curve c is part of the boundary of object x.) The influence of such an object on the value of the sensor is given by the function F_{sen}, which allows us to abstract away from the sensor's characteristics. $F_{sen}(y,r,c)$ denotes the median value the sensor would have if located at y on a bearing r in a world containing only the object boundary c. This value is combined with the effect of sensor noise at

time t, given by the term SenNoise(Left,t), to yield the actual value of the sensor.

Symmetrical formulae are required for the right sensor.

$$\text{Holds}(\text{Low}(\text{Right}),t) \leftarrow \quad \text{(B5)}$$
$$\neg \exists\, y,r,c,x\ [\text{HoldsAt}(\text{Location}(y),t) \wedge \text{HoldsAt}(\text{Facing}(r),t) \wedge \text{HoldsAt}(\text{Boundary}(x,c),t) \wedge F_{sen}(y,r+45,c) + \text{SenNoise}(\text{Right},t) > \delta 2]$$
$$\text{HoldsAt}(\text{High}(\text{Right}),t) \leftarrow \quad \text{(B6)}$$
$$\text{HoldsAt}(\text{Location}(y),t) \wedge \text{HoldsAt}(\text{Facing}(r),t) \wedge \text{HoldsAt}(\text{Boundary}(x,c),t) \wedge F_{sen}(y,r+45,c) + \text{SenNoise}(\text{Right},t) > \delta 1$$

The SenNoise function plays a similar role to the VelNoise and RotNoise functions. The effect of sensor noise is assumed fall within in a certain range although, of course, the exact amount of sensor noise at any time is unknown.

$$-\varepsilon_s \leq \text{SenNoise}(t) \leq \varepsilon_s \quad \text{(B7)}$$

The effects of sensor events and the conditions under which they are triggered are given by the following formulae, which are generic to both the left and right sensors.

$$\text{Happens}(\text{LatchOn}(d),t) \leftarrow \quad \text{(H1)}$$
$$\neg\, \text{HoldsAt}(\text{On}(d),t) \wedge \text{HoldsAt}(\text{High}(d),t)$$
$$\text{Happens}(\text{LatchOff}(d),t) \leftarrow \quad \text{(H2)}$$
$$\text{HoldsAt}(\text{On}(d),t) \wedge \text{HoldsAt}(\text{Low}(d),t)$$
$$\text{Initiates}(\text{LatchOn}(d),\text{On}(d),t) \quad \text{(E9)}$$
$$\text{Terminates}(\text{LatchOff}(d),\text{On}(d),t) \quad \text{(E10)}$$

The On and Off fluents ensure that LatchOn and LatchOff events only occur when a threshold is passed.

Finally, we have two uniqueness-of-names axioms.

$$\text{UNA}[\text{Go, Rotate, Stop, LatchOn, LatchOff}] \quad \text{(B7)}$$
$$\text{UNA}[\text{Moving, Turning, Location, Facing, Boundary, On, High, Low}] \quad \text{(B8)}$$

We're now in a position to view sensor data assimilation as abduction. Given a sequence of sensor events, the task is to construct an explanation of those events by postulating the existence of objects of suitable shapes and locations. This task is essentially abductive. Roughly speaking, if the sequence of sensor events is represented by the formula Ψ, and we are given formulae Σ, representing the relationship between the robot and the world, and Δ_N, representing the robot's actions, then we are seeking a formula Δ_M such that,

$$\Sigma \wedge \Delta_N \wedge \Delta_M \models \Psi.$$

Care must be taken when using abduction to explain observations in the context of actions with non-deterministic effects. The problem is that an observation — in this case Ψ — can have a greater degree of precision than can be explained by the occurrence of an action with a non-deterministic effect. However, if the method of determining fluents is used, this difficulty can be circumvented by including values for those fluents in the explanation.

Δ_M is, first and foremost, a map of the world in which, because of motor and sensor noise, the boundaries of objects are not precisely given. But because noise is treated as non-determinism, Δ_M will also have to contain a certain amount of unwanted junk. This takes the form of formulae that assign values to the determining fluents representing the noise present in the motors and sensors. These formulae will be stripped from Δ_M.

Before moving on to an example, a further definition is required. In order to rule out explanations which posit phantom objects, we want our explanations to entail not only the occurrence of the events in Ψ, but also the non-occurrence of events not in Ψ. Accordingly, we will be looking for explanations of $\Psi \wedge \text{COMP}[\Psi]$, where $\text{COMP}[\Psi]$ is defined as follows.

Definition 4.1.

$$\text{COMP}[\Psi] \equiv_{\text{def}} [\text{Happens}(a,t) \wedge [a = \text{LatchOn}(d) \vee a = \text{LatchOff}(d)]] \rightarrow \bigvee_{\langle \alpha,\tau \rangle \in \Gamma} [a = \alpha \wedge t = \tau]$$

where $\Gamma = \{\langle \alpha,\tau \rangle \mid \text{Happens}(\alpha,\tau) \in \Psi\}$. □

5 A Worked Example

Figure 1 shows the robot encountering a wall. After detecting the wall, the robot rotates so that it is sideways on to the wall, then follows it for a while until coming to a halt. At some time during the robot's turn, its left sensor will detect the wall. This is the sensor event we want to explain.

Let M1 be the conjunction of the following formulae.

$\text{Initially}_P(\text{Location}(L0))$

$\text{Initially}_P(\text{Facing}(R0))$

Let N be the conjunction of (H1) and (H2) with the following formulae.

$\text{Initially}_N(\text{On}(\text{Left}))$	$\text{Initially}_N(\text{On}(\text{Right}))$
$\text{Happens}(\text{Go},1000)$	$\text{Happens}(\text{Stop},3000)$
$\text{Happens}(\text{Rotate},4000)$	$\text{Happens}(\text{Stop},5000)$
$\text{Happens}(\text{Go},6000)$	$\text{Happens}(\text{Stop},8000)$

Let Ψ be the following formula.

$\text{Happens}(\text{LatchOn}(\text{Left}),4500)$

Let E be the conjunction of (E1) to (E10). Let B be the conjunction of,

- the event calculus axioms CEC,
- the background axioms (B1) to (B8), and
- the Trajectory formulae (T1) and (T2).

Let M2 be the conjunction of formulae (5.1) to (5.11) below. As we'll see shortly, M2 is an explanation of Ψ. M2 has several components. For the interval up to the time the robot stops having detected the wall, we have,

$$\text{FromTo}(C1,L0,Y1) \wedge \text{Bng}(C1) = R0 \wedge 2000.(V - \varepsilon_V) \leq \text{Lng}(C1) \leq 2000.(V + \varepsilon_V) \quad (5.1)$$
$$\neg \exists\, c,x,t\ [\text{Initially}_P(\text{Boundary}(x,c)) \wedge 1000 \leq t < 3000 \wedge F_{sen}(\text{LOC}(t),\text{Bng}(C1)+45,c) + \text{SenNoise}(\text{Right},t) > \delta 1] \quad (5.2)$$

$$\neg \exists\, c,x,t\ [\text{Initially}_P(\text{Boundary}(x,c)) \wedge \quad (5.3)$$
$$1000 \le t < 3000 \wedge$$
$$F_{sen}(LOC(t),Bng(C1)–45,c) +$$
$$SenNoise(Left,t) > \delta 1].$$

This collection of three formulae is typical. In essence, it says that there are no objects within a certain range (constrained by the characteristics of the sensors) either side of a line C1 which starts at L0 and ends at Y1. The locations of C1's start and end points are not precisely known. Note the inclusion of the noise terms in (5.2) and (5.3).

LOC and DIR are defined, using standard trigonometric functions, to yield the location and bearing of the robot for any given time, relative to landmark locations and bearings, such as L0, Y1, Bng(C1) and Bng(C2).

For time 4500, when the left sensor detects the wall, we have,

$$\exists\, c,x\ [\text{Initially}_P(\text{Boundary}(c,x)) \wedge \quad (5.4)$$
$$F_{sen}(Y1,Bng(C1)–45,c) +$$
$$SenNoise(Left,4500) > \delta 1].$$

M2 doesn't require a component for the interval between times 3000 and 4500, since neither its location nor its sensor data change during that time. For the interval between the left sensor's detection of the wall and the end of the rotation, we have,

$$\exists\, c,x\ [\text{Initially}_P(\text{Boundary}(c,x)) \wedge \quad (5.5)$$
$$\forall\, t\ [4500 < t \le 5000 \rightarrow$$
$$F_{sen}(Y1,DIR(t)–45,c) + SenNoise(Left,t) > \delta 2]].$$

Note that (5.4) refers to the higher threshold $\delta 1$, while (5.5) refers to the lower threshold $\delta 2$. This is because the sensor value has to exceed $\delta 1$ at least momentarily for a LatchOn event to occur. For the On fluent to continue holding, however, it's only necessary for it to remain above $\delta 2$. For the interval during which the robot is wall-following, we have,

$$FromTo(C2,Y1,Y2) \wedge \quad (5.6)$$
$$1000.(W - \varepsilon_W) \le Bng(C2) - Bng(C1) \le$$
$$1000.(W + \varepsilon_W) \wedge$$
$$2000.(V - \varepsilon_V) \le Lng(C2) \le 2000.(V + \varepsilon_V)$$

$$\neg \exists\, c,x,t\ [\text{Initially}_P(\text{Boundary}(x,c)) \wedge \quad (5.7)$$
$$6000 < t < 8000 \wedge$$
$$F_{sen}(LOC(t),Bng(C2)+45,c) +$$
$$SenNoise(Right,t) > \delta 1]$$

$$\exists\, c,x\ [\text{Initially}_P(\text{Boundary}(c,x)) \wedge \quad (5.8)$$
$$\forall\, t\ [6000 < t < 8000 \rightarrow$$
$$F_{sen}(LOC(t),Bng(C2)–45,c) +$$
$$SenNoise(Left,t) > \delta 2]].$$

Finally, we require some values for the VelNoise and RotNoise functions (see Proposition 3.3).

$$VelNoise(1000) = \frac{Lng(C1)}{2000} - V \quad (5.9)$$

$$RotNoise(4000) = \frac{Bng(C2) - Bng(C1)}{1000} - W \quad (5.10)$$

$$VelNoise(6000) = \frac{Lng(C2)}{2000} - V \quad (5.11)$$

Proposition 5.12.

$$CIRC[N ; Happens] \wedge$$
$$CIRC[E ; Initiates, Terminates, Releases] \wedge$$
$$M1 \wedge M2 \wedge B \models \Psi \wedge COMP[\Psi]$$

Proof. See full paper □

Proposition 5.12 asserts that M2 is indeed an explanation of Ψ. But as it stands, it's not very useful as a map of the world. To extract a useful representation, we need to exploit the known bounds on the noise terms. We can then pick out useful consequences of M2, and discard unwanted junk. For example, from (B7) we have the following consequence of M2.

$$\exists\, c,x\ [\text{Initially}_P(\text{Boundary}(c,x)) \wedge$$
$$\forall\, p\ [On(p,C2) \rightarrow$$
$$\delta 2 - \varepsilon_S \le F_{sen}(p,Bng(C2)–45,c) \le \delta 2 + \varepsilon_S]]$$

Concluding Remarks

While the usefulness of logic in the study of high-level cognitive skills is widely accepted, it's natural to question the need for a logical account of low-level perceptual tasks. But, as many contemporary cognitive scientists agree, there is no clean separation between cognitive and motor-perceptual systems in an embodied agent. If they're right, then a logical approach to common sense reasoning can only succeed if logic permeates all levels of the system.

Two key features of the explanations supplied by the foregoing abductive account are determining fluents (noise terms) and uncertain boundaries. Noise terms eliminate the need for consistency-based abduction, while uncertain boundaries enable a whole set of possible configurations of objects to be captured in a single explanation.

The question of robot control is outside the scope of this paper. But it's assumed that the robot has some form of exploration strategy for gathering sufficient sensor data to construct a map. Work in progress takes a logic programming approach to implementation. (The formalisation of the present paper isn't intended to be used directly.) The use of abductively acquired maps for navigation is also under investigation.

References

[Davis, 1986] E.Davis, *Representing and Acquiring Geographic Knowledge*, Pitman (1986).

[Kuipers, *et al.*, 1993] B.Kuipers, R.Froom, W.-Y.Lee and D.Pierce, The Semantic Hierarchy in Robot Learning, in *Robot Learning*, ed. J.Connell and S.Mahadevan, Kulwer Academic Publishers (1993), pp. 141-170.

[Poole, 1995] D.Poole, Logic Programming for Robot Control, *Proceedings IJCAI 95*, pp. 150-157.

[Shanahan, 1996] M.P.Shanahan, Noise and the Common Sense Informatic Situation for a Mobile Robot, *Proceedings AAAI 96*, pp. 1098-1103.

[Shanahan, 1997] M.P.Shanahan, *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*, MIT Press (1997).