Deriving Fluents from Sensor Data for Mobile Robots

Mark Witkowski, David Randell and Murray Shanahan

Intelligent and Interactive Systems Group
Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine
Exhibition Road
London SW7 2BT
United Kingdom
{m.witkowski, d.randell, m.shanahan}@ic.ac.uk

Abstract

In this paper we consider some practical issues of grounding logical symbols derived from low-level sensor streams in the context of a miniature mobile robot. We identify three distinct layers, primitive, derived and synthetic, at which symbols may be anchored in order to connect a model-based view of the robot's world with its sensory input. The Event Calculus, a logical formalism for reasoning about action, forms the basis of an abductive approach to several related robot tasks, sensor data assimilation, planning and map building.

1 Introduction

In formal logic, symbols are given an interpretation (i.e. are given semantic values) by a function that maps them to entities in a domain. These interpretations may be abstract or concrete. Without this ascription of semantic values, the logical system remains uninterpreted, and its constituent symbols semantically inert. That is to say, the symbols cannot be said to be about anything - the predicate and constant names in the language do not denote or refer to anything. Any sentences constructible in that formal language (apart from those expressing tautologies) consequently remain neither true nor false.

Within logic, the ascription of semantic values to the set of symbols is simply stipulated. But in the case of an autonomous robot using a logical language to represent and reason about objects in its domain, the interpretation is typically concrete. For example, the robot will need to identify, represent and reason about physical objects, including other robots, and their relationships. In order to plan, it will also need to reason about sequences of states, or events, either actual or hypothetical. Either way, the robot has to be able to maintain the correspondence of the symbols in its language, to the set of actual or inferred entities in its domain, so that it can act effectively upon the world. Following Coradeschi and Saffiotti (2000), the process of establishing and maintaining the correspondence between abstract representations and features of the real world (derived from sensor data) will be referred to as *anchoring.* The anchoring process defines the point of contact between the symbolic system and the real world, and ensures that the otherwise semantically inert symbolic representations used by the robot can genuinely be said to be *about* the world (Konolige et al. 1997; Coradeschi and Saffiotti 2000). The anchoring issue first appeared in the AI literature as the "symbol grounding problem", after Harnad (1990), although the general problem of the relationship between sensation and meaning has long been the subject of debate in philosophy (e.g. Frege 1892; Russell 1905; Quine 1960).

In this paper, we describe techniques used to address the problem of symbol grounding in a mobile robot. Details of the underlying logic formalisation and implementation approach described in this paper can be found elsewhere (Shanahan and Witkowski 2000). Specifically, we show how a stream of continuous data from the robot's sensors is meaningfully transformed into a sequence of discrete symbols, which then participate in the logical reasoning processes that determine the robot's actions. Various robot tasks are discussed, including navigation, map building and localisation. Central to the discussion is the notion of a *fluent*, a function whose value changes over time. Fluents act as a pivot point between the robot and the abstract, logical representation it uses. The problems of map building, localisation and planning for mobile robots have been effectively addressed using numerical and probabilistic techniques (e.g. Thrun 1988). In marked contrast our primary motivation is to investigate (i) explicit logical representations of actions, events and the juxtaposition of objects in the robot's environment, and (ii) show how well understood automated reasoning tasks applied to these structures can be used to implement robot control.

In section 2, we consider three levels at which symbols may be said to be grounded. Section 3 summarises the Event Calculus, a formalism for reasoning about action that can be used to capture the relationship between sensor events, the robot's actions, and the actions of other agents.

Section 4 describes how fluents are derived from actual sensor readings in a miniature Khepera robot.

Section 5 introduces the role of *abductive reasoning* (reasoning from events to possible causes). Abduction forms the basis of the strategy we use to control robots by logic. Section 6 summarises how abduction may be used to interpret the incoming sensor data to perform a number of robot tasks. In the first task, *sensor data assimilation*, abduction is used to discover plausible explanations for incoming sensor fluents in the context of an existing map. In the second tasks, *map building*, incoming fluents are interpreted as new features that can be added to the robot's incomplete map of the environment. In the third case, *planning*, the robot abduces a sequence of possible actions to achieve some goal, such as being at a specified location. In the final task, we briefly consider *localisation*, which the robot carries out if no interpretation can be found for its incoming sensor data that is consistent with the location it currently thinks it is in.

In this paper it is not our intention to provide a detailed description of, or justification for, the Event Calculus, as several accounts are available elsewhere (for instance, Shanahan 1997). Neither do we consider the logical inference processes that underpin the abductive reasoning methods, as these are also considered in detail elsewhere (Shanahan 1996; Shanahan and Witkowski 2000). Rather, we focus on the question of the grounding of the symbols employed in the reasoning system that mediates between the low-level sensory information supplied by the robot's sensors and the low-level control commands issued to its actuators.

2 Levels of Anchoring

In order to identify the precise interface point between logical symbolic descriptions of the world and the incoming sensor data in a physically embodied system, we identify several levels of increasing abstraction and structure: (i) primitive, (ii) derived, and (iii) synthetic. At the lowest (primitive) level raw sensor data streaming from the sensors is measured. The values obtained are directly tied to the sensors themselves and are continuous in nature.

At the second processing level, various sensor values (over time) are combined. It is at this second level, and this only, that events and states are recognised. This derived information is characterised by the use of generic descriptions of features of the robot's perceived world. For example, sufficient information from the combined sensory values will be available to be able to match the sensor data to a generic description of features, as distinct from a specific instance of that feature type. That is to say, the generation of sensory symbols, or rather the mapping of a sensory-symbol type to incoming sensor data provides a necessary but not sufficient condition in the recognition of particular instances of corners, doorways, walls and any other distinguishing features of the robot's world.

Having now been able to recognise the signature of specific features in the robot's domain, and with a pre-defined symbolic model of the world, the robot interacting with its environment becomes capable of naming ground instances of those features in the world, e.g. corner1, corner2, and so on. Here, the interpretation stemming from the model, not only allows recognition of a corner, as a specific corner, but it also allows re-identification of that corner. At this synthetic level, the same (derived) input, related to the model provides a necessary and sufficient condition for object naming and the maintenance of the anchoring of the symbols to objects and features of the robot's world.

These three levels express a concrete interpretation of the world, i.e. the objects so anchored refer to actual physical features in the robot's domain. An additional *abstract level* is also recognised where alternative models exist that do not (necessarily) denote actual physical features, realised in the actual world. The three main levels (the primitive, derived and synthetic) with their increasing levels of abstraction should not be seen in terms of an increase of complexity of information maintained; rather they should be seen in terms of the distance of the point where observations made reach sufficient bounds in order to be verified against a model, and where, for example, re-planning may be required in order to satisfy consistency of logical model used. In this sense then, the anchoring process points to the very heart of hypothesis generation and testing, where a "sanity check" against the world is rendered not only desirable, but more importantly possible.

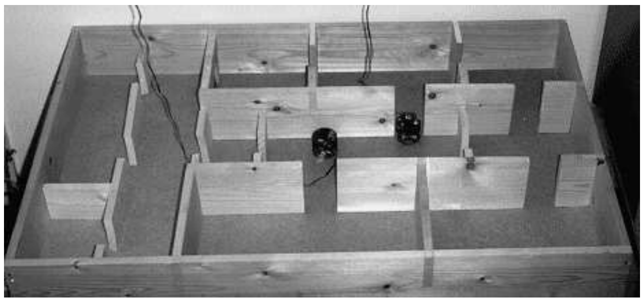
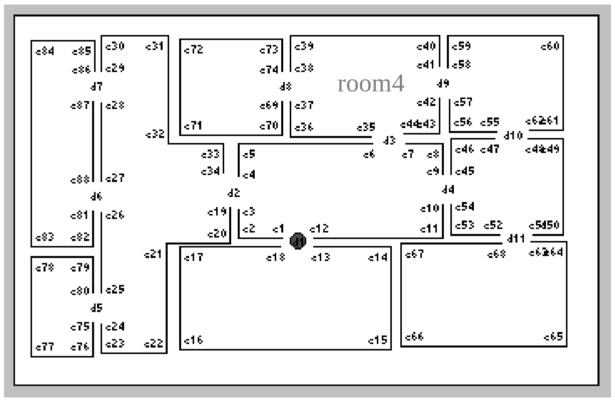


Figure 1: The Miniature Office Environment

In this paper we discuss how a small mobile robot may construct, using a logic representation, a complex synthetic view of its world, building on a stream of simple sensor events that arise following actions by the robot. The robot's environment is to be interpreted in terms of "walls", "corners", "doors" and "rooms", in keeping with our intuitive view of an "office" environment. An example is shown in the photograph of figure 1. A formal description of the environment (for room4 only), and its visualisation is shown in figure 2. This description can be constructed solely from an *interpretation* placed on the stream of sensor-derived fluents within a logical model. The symbolic description shown here was created automatically using a map building procedure, though such maps may also be handcrafted.

Specific features in the environment, such as the individual corners, doors and rooms, must be interpreted as such, and assigned unique, synthetic names, consistent with the logical model. Each of sensor-derived fluents relates only to a simple, local feature, for instance, that the robot has encountered - a corner, or a door. Each is anonymous, incoming symbols are not specific to individual features, but report a feature type.

Sensing on the Khepera robots (K-Team 1995) used is very restricted. The robot must be within 2cm of a wall or other feature to detect it. We also use odometry to determine the length of travel along wall features. Despite their limitations, these senses, together with certain restrictions on the design of the environment are sufficient to drive the logical model and create an interpretation of a larger map.



```
/* Room 4 */
next_corner(r4,c35,c36).      door(d3,c35,c44).
next_corner(r4,c36,c37).      door(d8,c37,c38).
next_corner(r4,c37,c38).      door(d9,c41,c42).
next_corner(r4,c38,c39).      inner(c36).
next_corner(r4,c39,c40).      inner(c39).
next_corner(r4,c40,c41).      inner(c40).
next_corner(r4,c41,c42).      inner(c43).
next_corner(r4,c42,c43).      connects(d3,r4,r1).
next_corner(r4,c43,c44).      connects(d8,r4,r7).
next_corner(r4,c44,c35).      connects(d9,r4,r8).
```

Figure 2: Visualisation of rooms, and room 4 description

3 Fluents and the Event Calculus

When applying a logic formalism to robotics, it becomes clear that the scheme used must be able to represent the effects of actions and the consequential changes that occur to the robot and the environment. Equally, it must be able to represent the effects of exogenous events on the robot, as detected via its sensors. To achieve this, the underlying ontology (the primitive or given features of a language) of the Event Calculus is based on *fluents*, the description of entities that can change state with time; *events* (or *actions*), that can cause the state of a fluent to change; and *time points*, the instants of time at which changes occur. Fluents can represent the state of a sensor, the position of a robot, or the state a feature in the environment (for instance, whether a door is open or not). Action events may be initiated by the robot (possibly as the result of planning), or represent other exogenous events within the environment, causing fluents to change independently of the robot. Time points are ordered. In a more formal treatment of the Event Calculus this ordering would be made explicit, here it will be assumed.

The Event Calculus also defines seven basic predicates, which fully represent the ways in which fluents and actions interact, and the time ordering between them:

- Initially(*f*), indicating that the fluent *f* holds a value of true (Initially$_1$) or false (Initially$_0$) at time 0.
- HoldsAt(*f*,*t*), indicates that the fluent *f* holds true at an instant, *t*.
- Happens(*a*,*t1*,*t2*), indicates that the action or event *a* occurs during the time range bounded by *t1* and *t2*. In practice this will be qualified by preconditions, placing restriction on when a robot might perform an action, or when an exogenous event is possible.
- Initiates(*a*,*f*,*t*), indicates that fluent *f* will hold after an occurrence of action *a* at time *t*.
- Terminates(*a*,*f*,*t*), indicates that *f* will no longer hold true after an occurrence of *a* at time *t*.
- Clipped(*t1*,*f*,*t2*), indicates that the state of fluent *f* has altered during the range of times *t1* to *t2*.
- Before(*t1*,*t2*), makes explicit the ordering relationship between a pair of time points.

The Event Calculus has previously been proposed as a solution to the *frame-problem*, (Shanahan 1997), as it overcomes the need to explicitly maintain knowledge about what *does not change* as a consequence of performing actions or due to the occurrence of exogenous events. This is clearly a major concern when applied to robotics, but is by no means restricted to robotics tasks, and the Event Calculus axioms may be used as a "wrapper" to augment other logic representations where time, change and the effects of actions must be considered. In principle, it is possible to record a complete history of events and changes to fluents (encoded as "Initially" and "Happens" formulae), though in a robot environment this may be neither possible, nor desirable, if the reasoning process is not to become overwhelmed with extraneous "memories".

4 Deriving Fluents from Sensors

This section describes the effects of the action events and generation of sensor event fluents that may be used to characterise model office environments of the form shown in figure 1 when used with miniature (6cm high) Khepera mobile robots. As only two of the Khepera's sensor modalities are used, six of the eight infra-red proximity sensors (with an effective range of about 2cm, and located as indicated in the robot outline in figure 3), and wheel

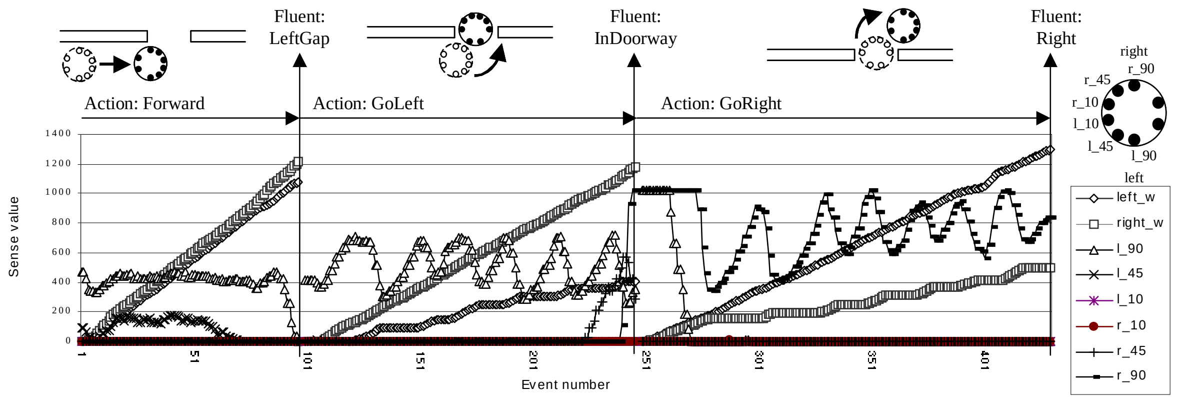


Figure 3: Deriving Fluents from Raw Sensor Events at the Robot Level

odometry, a number of restrictions are placed on the design of this environment, which must be rectilinear. All rooms must be connected only by doors and doorways must follow strict dimensional criteria if the robot is to be able to detect both doorposts with its sensors.

Sensor events within the Event Calculus program detect discontinuities ("caused" by the corners, door-posts and doorways) between features. At the robot level, these sensor conditions only identify the type of feature, and must be combined and named to provide a coherent description and map of the environment.

In the model office environment, we use seven distinct and mutually exclusive conditions to sense all the significant transition events (fluents) used by the Event Calculus programs (Left, Right, LeftAndFront, RightAndFront, LeftGap, RightGap and InDoorway). These are complemented by five action commands (Forward, GoLeft, GoRight, Turn and Back). Fluents arise from actions, and also terminate them.

The sensor fluents Left and Right indicate that the robot is beside a wall (on its left or right hand side, respectively) and may follow it Forward until the next feature. The fluents LeftAndFront and RightAndFront indicate that the robot has moved forward and encountered another wall at a concave corner. The robot may then perform a GoRight or GoLeft action to align itself with that connecting wall (a Back action allows it to return along the wall it is on). The fluents LeftGap and RightGap indicate that the robot has overshot the wall it is following. This occurs in two distinct cases, that of a convex corner (e.g. c21 or c32 of figure 2) or when a doorpost is encountered (e.g. c1, c26 or c38). The robot cannot directly disambiguate these two cases using its sensors, and so must perform a GoLeft (or GoRight) to follow round the corner. Sensing a Left or Right indicates a concave corner, an InDoorway that it was a doorpost. When in a door, the robot may proceed with actions GoLeft or GoRight to enter the next room, or perform a Turn (spin round by 180°) followed by a GoLeft or GoRight to continue in the same room.

Fluent events are detected by a combination of changing sensor values and the context in which those changes are detected. Figure 3 charts the values for individual readings of the sensors, and the cumulative rotations of the wheels ('left_w', diamond markers and 'right_w', square markers) for a sequence of three consecutive actions, Forward, GoLeft then GoRight. This gives rise to three fluents, LeftGap, InDoorway and Right, as it approaches, enters and finally leaves a doorway in the environment. At the beginning of the sequence the robot is aligned with a wall to its left (consequently the last fluent delivered would have been Left). The Forward action initiates a period of wall-following, during which the robot controller attempts to keep a constant distance from the wall to its left while it progresses forward. This distance is measured by the 'l_90' ("left at 90°") infra-red proximity sensor (triangle markers), and a simple servo-loop established to maintain the value within a narrow target range by differentially adjusting the left and right motor speeds. The infra-red sensors return a 10bit value; the value of 400 corresponds to approximately 1cm from the wall. Note the small correction at the beginning of the sequence, indicating that the robot was not exactly aligned parallel with the wall by the previous command. Towards the end of the Forward sequence the value of l_90 rises slightly as the robot turns into the corner to compensate for the otherwise diminishing sensor value. As the robot overshoots the corner, the value from l_90 drops rapidly. This precipitates the LeftGap fluent and terminates the action. The robot is programmed to automatically reverse to recapture the wall just prior to the gap.

In executing the next command, GoLeft, to enter the doorway, it may be seen that the robot edges around the corner (that is the doorpost) by keeping the speed of the right motor constant and starting and stopping the left motor to keep the l_90 sensor within a (broader) band of values. The opposing doorpost is detected by the rapid rise of sensor r_90 (horizontal bar markers) and the InDoorway fluent generated. Had the previous LeftGap fluent been part of a convex corner, the value of r_90 would have remained low, and a Left fluent would have been generated after the robot had completed a 90° turn. It may be seen that the GoRight action operates in a similar fashion to exit the doorway and align the robot with the wall in the next room.

While fluents are reported to the logical layer as instantaneous events, the robot internally treats this symbol as a "state", which persists until the next valid action. The actions available in any of these states is restricted to only those which can themselves lead to another valid fluent. For instance, if the robot has detected a concave corner, say RightAndFront, it may perform a GoLeft action to align itself with the next wall, or a Back to return along the right hand wall. It may not, in these circumstances, attempt to drive or turn into the wall with a Forward or GoRight action. These action commands and sensor fluents have been embedded into an "extended BIOS" for the Khepera, which may be downloaded into the robot's RAM (or blown into a replacement ROM), and become available to any high-level control program via the Khepera's RS232 communications link. In the next part of the paper we consider how the logic based robot controller processes fluents using abduction.

5 Abduction

Abduction is a form of reasoning which attempts to provide explanations, by established proof procedures, for given events. Abduction is therefore particularly relevant to the application of reasoning in logic to robotics, where we expect a stream of events to be generated by the normal process of the robot sensing conditions arising (either through its own actions, or through the occurrence of extrinsic events) within the robot's environment. For example, a robot may encounter an obstacle in its path. Several possible explanations might be considered. If that obstacle is already recorded within the robot's description of the world, the obstacle's presence is trivially explained, and the robot may perform some action to avoid the obstruction. However, if the robot were currently in an unexplored part of its environment, the explanation would clearly involve adding knowledge about the obstacle to the robot's model. If not, several alternate explanations could be formulated from the robot's description of the world and it's properties. It might be that an external agent had deposited the obstacle while it was not being observed. If the obstacle is modelled as "immovable", this explanation may have to be discarded. Finally, in this example, the robot may be forced to the conclusion that it has become disoriented in its environment, and that object naming is now without foundation.

Reasoning by abduction is related to, but differs from deductive reasoning. The process of abduction is directed toward an *explanadum* (a fact or observation to be explained), given a background theory (in this case the Event Calculus axioms, the robot program and other components introduced in the previous section). As with other forms of reasoning, only explanations supported by the model may be generated. As already noted, abductive reasoning may give rise to several alternative explanations, which are (by definition) equally supported, although not necessarily equally desirable. Boutilier and Becher (1995) introduce a *preference ordering* in the context of belief revision to resolve this problem. We note that the generation of more than one explanation will have different effects according to the task being addressed, sometimes indicative, sometimes benign and sometimes detrimental. Abductive reasoning has been used to good effect in model-based diagnosis, where possible explanations of mal-function must be formed (Davis 1984; de Kleer and Williams 1987).

6 Abduction for Sensor Data Assimilation, Map Building and Planning

A cognitive robot controller using the Event Calculus properly consists of the set of Event Calculus axioms (styled as "EC"), which define the underlying rules for reasoning about time and change and a *domain theory*, robot programs in the event calculus (styled "Σ") that describe various interactions between the actual robot and its environment. An event calculus robot program will, in turn, consist of:

1) The effects of the robot's low-level actions on the environment.
2) A description of impact of the environment on the robot's sensors.
3) The effects of high-level actions (for hierarchical planning).
4) High-level actions in terms of component lower-level actions.
5) The historical "narrative" of past events (styled as "Δ").
6) A map of the environment encoded as Event Calculus axioms.
7) Which predicates are abducible.

Robot control is embedded in a "sense-plan-act" cycle, which continues *ad-infinitum*. Short bursts of planning activity are inter-leaved with actions and sensor gathering. Planning is a computationally demanding task, often more so when conducted in a formal reasoning environment than when performed by *ad-hoc* planning algorithms. To alleviate this problem, plans are created hierarchically, initially from high-level action descriptions. Once a high-level plan is available (for instance, at the room level), only the first step is expanded (and the first step of that, etc.) until a starting sequence composed of only low-level actions is formed. This is *progression order planning* (Shanahan and Witkowski 2000).

This section considers how the abductive reasoning scheme might be applied to a range of different tasks. In each case the robot controller is presented with some event (styled "Γ"), either actual, as in the case of an incoming sensor event (Γ_S), or desired, such as the goal in a plan (Γ_G), which must be explained or otherwise interpreted by creating a residue of "Happens" formulae (styled "Ψ") by automated reasoning. In map building a novel sensor event (representing an environmental feature) must be assimilated into the robot's map and model of its world. In planning the robot must also generate sequences of actions, recorded as "Happens" formulae, to achieve its goals. Because these abductive processes are all similar we note that the bulk of the event calculus description remains identical across all the activities. According to the task, the detailed processes invoked will differ, and in particular, the set of items declared abducible changes (for example, to generate plan items during planning but map descriptions during map building, etc.)

6.1 Sensor Data Assimilation

In sensor assimilation, the explanation of a sensor event (Ψ_S) is encapsulated by the abductive entailment:

$$EC \,\&\, \Sigma \,\&\, \Delta \,\&\, \Psi_S \models \Gamma_S$$

That is, generate some new explanation, Ψ_S, that, when taken with the Event Calculus axioms (EC), the existing robot control program and map, (Σ), and the narrative of past events, which entails the current sensor input Γ_S. It is this step that maintains the anchoring over time of symbolic names (e.g. c23 or d3 of figure 2) to the physical objects they denote. Note that this process is not one of anchor tracking as the percept of the object is not maintained over time, but rather one of anchor re-acquisition. Normally, of course, a sensor event will be consistent with the current map, and so be trivially explained. Changes to the environment, such as a door being closed, may equally be explained in this manner according to the definition Σ. Due to the restrictions inherent in the Khepera's sensing, when a door is in place ("closed"), it appears to form part of a long "wall", and the next fluent encountered will be that of the next feature around the room. Where the next fluent encountered can be explained by the closure of a door, the new state of the door can be noted and the fluent accepted.

The abduced residue must be consistent with the current plan being executed, if it is not, the plan must be abandoned and a new one initiated, taking account of the changes. Where the residue is neither consistent with the existing map nor be explained in terms of a new map feature, the robot can conclude it has become disoriented in the environment and initiate a localisation process.

6.2 Map Building

$$EC \,\&\, \Sigma \,\&\, \Delta \,\&\, \Psi_M \models \Gamma_S$$

Map building is a variant of this abductive scheme. If some sensor event Γ_S occurs that cannot be explained by the map, but could be if the map were extended by the residue Ψ_M, then new knowledge has been acquired and the map can be augmented. New features in the map are automatically named (using a successor function) and added to the map description formulae. Clearly, in these circumstances, the abductive process must give rise to a single interpretation before the map can be extended. In contrast to sensor data assimilation, this step creates new symbols and anchors them to the new feature.

The accuracy of the Khepera's odometry sense is inadequate to localise a room feature to a unique place (this effect may be noted from figure 2, where rooms appear not to align as they should and walls appear to have varying thickness). This error is sufficient to cause positional ambiguity between a group of corners such as c8, c43, c46 and c56. We are therefore obliged to add *integrity constraints* (Shanahan and Witkowski 2000), making explicit, for instance, that a corner cannot be located in two rooms to restrict the abductively generated alternatives.

We normally consider map building in this way to be a specific process, rather than an opportunistic activity, with a complete room explored by progressing around it in a clockwise (or anti-clockwise) direction. Mapping a complete environment consists of exploring a room, and then planning a path through known space to a doorway that leads to an unexplored room.

6.3 Planning

$$EC \,\&\, \Sigma \,\&\, \Delta_0 \,\&\, \Psi_P \models \Gamma_G$$

In planning, some desired event Γ_G is postulated (such as "HoldsAt(At(C19),*t*)"), given a current place in the narrative of events (Δ_0) and a residue constructed (Ψ_P) then describes a sequence of action events that lead to the desired goal condition (for instance, "Happens(GoLeft, T100), Happens(Forward, T101), Happens(GoRight, T102), Happens(Forward, T103), Happens(GoLeft, T104), Happens(GoRight, T105)"). In this case there may be several, equally valid, residues, equating to multiple possible paths through the environment. Several strategies can be devised to select between them, apart, of course, from taking the first plan formulated. Minimising the number of steps is a reasonable measure, and minimising distance travelled. In this instance though, timing information is more significant, as the elapsed time to traverse corners is generally greater than that to follow walls. Such information is available at the lower, robot level, but may not be made accessible to the logic level.

6.4 Localisation

$$EC \mathbin{\&} \Sigma \mathbin{\&} \Delta_R \mathbin{\&} \Psi_L \models \Gamma_S$$

In localisation we can attempt to build a residue (Ψ_L) comprising exactly one abductive explanation of the current sensor fluent (Γ_S) and the recent history (Δ_R, that since the loss of localisation was detected), which is uniquely consistent with the existing map, and so defines a current, specific location within the map. While there are multiple abductive explanations, the robot could still be at one of a number of locations and further actions are required to disambiguate these. Should the residue Ψ_L ever become empty, the map is no longer valid, and must be reconstructed.

7 Multiple Robots

We have conducted a number of experiments with multiple robots operating in the same environment in order to determine the issues that arise. These issues fall into four categories: 1) To determine whether the existing set of sensory fluents was adequate to detect other robots, and indeed, to determine whether the sensors were adequate to reliably detect another robot. 2) Changes to the Event Calculus robot program to describe the effects of encountering other robots, and to allow for additional abductive explanations about contact with those robots. 3) Changes to the planning mechanism to account for additional robots, with the possibility of cooperation between the devices. 4) The role and type of communication between individual robots.

In general, other robots were found to be effectively detected by the proximity sensors, but it was not possible to (reliably) distinguish between an encounter with a second robot with that with any other object. It is also the case that pairs of robots might encounter each other at any point within the environment. If this occurs on a long “straight”, both detect a (spurious) concave corner. If one robot is in a doorway, the other robot will, when following a wall, treat it as a closed door, although the detour around the curved aspect of the first robot will extend the apparent length of the combined wall before the next feature. Encounters between robots when both were turning into a convex corner or doorway (a GoLeft or GoRight after a RightGap or LeftGap), or when exiting from a doorway were found to be particularly problematic. Four sensory fluents were added to detect the conditions where two robots meet while one (or both) are in the process of turning a convex corner (e.g. c32 of figure 2), or entering or leaving a doorway. The four fluents OCCL and OCCR (Obstacle Convex Corner Left/Right) and OEDL and OEDR (Obstacle Exit Door Left/Right) are specific to this case, they do not occur with a single robot. Unlike the other fluents, they may therefore be interpreted abductively without ambiguity.

We have not modified the planning component to operate in the multiple robot case, but note that, as the robots may not pass each other, one must retreat to allow the other to pass. This, and the entirely practical need to ensure that the robots’ power and communications cables do not become entangled, the planning and navigation task will surely focus on avoiding encounters with other robots.

8 Discussion

In this paper we have described our approach to symbol grounding, using miniature robots to illustrate the processes involved. We have argued that it is possible to construct sensory fluents from individual primitive sensor readings. These fluents act as a pivot point between the physical robot and a logical, synthetic, model of the world the robot inhabits. We further indicated how abductive reasoning may be applied to this stream of derived symbols to interpret them as specific identified and named features within that world. According to circumstances the incoming fluents can confirm expectations held by the robot controller, precipitate changes to planned activities, or be integrated into a partial map of the environment, extending the description the robot holds about its world. By integrating a symbolic representation and the physical world in this way we provide a means by which the robot controller can be seen to be validating an otherwise purely syntactic internal model.

We have referred to sensory fluents as derived, when they appear as atomic to the robot, and therefore might be better considered as primitive by our own classification. While it would undoubtedly be impractical to do so (the inferencing must operate within the time constraints due to the dynamics of the robot), it would nevertheless be possible to extend the robot logical model to provide an interpretation that allowed for the derivation of the sensory fluents from the individual sensor readings. For instance, “primitive fluents” l_90_high (sensor l_90 >600), l_90_mid (<=600 & >400) and l_90_low (<=400 & >20), taken in the context of a forward action (first part of figure 3) would describe the wall following activity. Similarly, l_10_high (>400) would generate the LeftAndFront sensory fluent, while l_90_off (<=20) a LeftGap fluent. Note that this is equivalent to an anchor maintenance process, where a continuous stream of percepts is anchored to a single object (i.e. the current wall or corner).

As previously noted, the odometry estimates generated by the Khepera movements are far from perfect, and the derived positions of features in the environment appear to drift with successive actions. This is a problem common to almost all mobile real robot implementations (for instance, Thrun 1998). Our approach is to treat these as “crisp” identifications, either the feature falls within an acceptable bound (i.e. there is no other equivalent feature that allows a conflicting interpretation), or it does not. This is in direct contrast, say, to the use of fuzzy boundaries or partial matching for anchoring tasks (for example, Coradeschi and Saffiotti 1999, 2001). Equally, primitive and sensory fluents are crisply generated when the conditions that characterise them are encountered, and not otherwise. The effect of positional drift is to cause anchoring to fail, in which case plan regeneration or localisation must occur.

The Khepera based model presented here is a simplification of what might be expected from a mobile robot operating in a real environment, one shared with people. It is characterised by a restricted fluent stream in a heavily constrained working environment. Our work continues with larger mobile robots, designed to work in a full-scale environment, using stereoscopic vision as the primary sense modality and wireless communications.

To provide a rigorous foundation to the analysis of camera data, we have developed a visual *Region Occlusion Calculus* (ROC), (Randell, Witkowski and Shanahan 2001). The Region Occlusion Calculus extends the earlier *Lines of Sight Calculus* of Galton (1994) and builds on the Region Connection Calculus of Randell, Cui and Cohn (1992). ROC reduces the possible alignments of pairs of objects in the visual field to 20 relations. Implicit in the use of this calculus in a robotics application is the ability to anchor specific regions in the visual image field to named objects. This is, in effect, the object permanence phenomenon - objects do not cease to exist or change their identity when obscured. Such input inevitably generates streams of many fluents as objects pass in front and behind each other in the visual field. Abductive reasoning can as a filter to remove sensory events that are expected/trivially explained by the model (such as those caused when the robot moves in a field of stationary objects). The (hopefully) small residue of conditions, such as those caused by objects that move in the field, or visual artefacts such as shadows and reflections, will require detailed explanation or inference.

Acknowledgements

Work described here has been supported by EPSRC project GR/N13104, "Cognitive Robotics II".

References

Boutilier, C. and Becher, V. 1995. Abduction as Belief Revision", *Artificial Intelligence*, 77:43-94

Coradeschi, S. and Saffiotti, A. 1999. Anchoring Symbols to Vision Data by Fuzzy Logic. In: Hunter, A. and Parsons, S. (eds.) *Quantitative and Qualitative Approaches to Reasoning with Uncertainty*, LNAI, Berlin: Springer (pre-print)

Coradeschi, S. and Saffiotti, A. 2001. Perceptual Anchoring of Symbols for Action. In proc. 17th IJCAI, 407-412

Coradeschi, S. and Saffiotti, A. 2000. Anchoring Symbols to Sensor Data; Preliminary Report. In Proc. 17th AAAI, 129-135

Davis, R. 1984. Diagnostic Reasoning Based on Structure and Behavior. *Artificial Intelligence* 24:347-410

de Kleer, J. and Williams, B. C. 1987. Diagnosing Multiple Faults. *Artificial Intelligence* 32:97-130

Frege, G. 1892. Über Sinn und Bedeutung (On Sense and Meaning). *Zeitschrift für Philosophie und Philosophische Kritik*. Translation: Geach, P. and Black, M. eds. 1980. *Translations from the Philosophical Writings of Gottlob Frege*, 25-50. Oxford: Blackwell.

Galton, A.P. 1994. Lines of Sight. *AISB Workshop on Spatio-temporal Reasoning*

Harnad, S. 1990. The Symbol Grounding Problem. *Physica D*, 42:335-346

K-Team. 1995. Khepera User Manual. K-Team SA, Ch. du Vasset, CP111, 1028 Préverenges, Switzerland, Version 4.06, November 1995

Konolige, K., Myers, K., Ruspini, E. and Saffiotti, A. 1997. The Saphira Architecture: A Design for Autonomy. *Journal of Experimental and Theoretical Artificial Intelligence* 9(1):215-235

Quine, W. V. 1960. *Word and Object*. MIT Press

Randell, D., Cui, Z. and Cohn, A.G. 1992. A Spatial Logic Based on Regions and Connections. In proc. 3rd Int. Conf. On Knowledge Representation and Reasoning, 165-176.

Randell, D., Witkowski, M. and Shanahan, M. 2001. From Images to Bodies: Modelling and Exploiting Spatial Occlusion and Motion Parallax. In proc. 17th IJCAI, 57-63

Russell, B. 1905. On Denoting. *Mind* XIV:479-493

Shanahan, M. P. 1996. Robotics and the Common Sense Informatic Situation. In proc. 12th Euro. Conf. on Artificial Intelligence (ECAL-96), 684-688

Shanahan, M. P. 1997. Solving the Frame problem: A Mathematical Investigation of the Common Sense Law of Inertia, MIT Press.

Shanahan, M. P. and Witkowski, M. 2000. High-level Robot Control Through Logic. In proc. *Agent Theories, Architectures and Languages* (ATAL-2000), Boston, 100-113

Thrun, S. 1998. Learning Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99:85-111