Cognitive Robotics: On the Semantic Knife-edge

Mark Witkowski, Murray Shanahan, Paulo Santos and David Randell
Intelligent and interactive Systems Research Group
Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine
Exhibition Road,
London SW7 2BT
{m.witkowski, m.shanahan, p.santos, d.randell}@ic.ac.uk

5/4/2001

Abstract

This paper describes some aspects of recent and ongoing work in the area of Cognitive Robotics in the Department of Electrical and Electronic Engineering at Imperial College. Our approach to Cognitive Robotics has been to apply abductive reasoning procedures using the Event Calculus, an extension to First Order Predicate Calculus (FOPC), to provide a unified view of several related mobile robotics tasks: sensor data assimilation, map-building and planning. Cognitive robotics depends on an explicit declarative representation. While this greatly facilitates reasoning about domain knowledge, it comes with an extra computational overhead. This is the basis of the *semantic knife-edge*, maintaining a delicate balance between expressivity and efficient implementation.

1 Introduction

"Cognitive Robotics" refers to an area of study that applies formal logical representations to physical robots, with the long-term goal to endow them with higher-level cognitive skills. Cognitive Robotics [4, 10, 11, 17, 21] proceeds from the conviction that advances in developing a clear semantics for robots and their environments, within the principled framework of logical description, will lead to further advances in our overall understanding of intelligence in robots and artificial agents [8]. Despite a promising early start with projects such as SRI's SHAKEY [13, 20], the rigours of applying formal logic to real robots remain problematic. In part, these problems arise from the "semantic knife-edge" [14], the observation that logic representations are apparently inevitably balanced between a lack of expressivity, which renders them useless for the tasks they are to be employed for, and too much detail, which renders them intractable to automated proof generation methods.

The use of explicit logical formalisms in the context of the Event Calculus, allows dynamic, non-trivial real world domains to be rigorously described and reasoned about. The declarative nature of the representational formalism and approach used facilitates the description of structure within the formalism, which can be exploited by specialist reasoning techniques. It also lays bare the primitive features of the language used and highlights how this maps directly to observable events and properties of objects in the modelled domain. Consequently, we show how it is possible to start with a high-level description of the world and eventually map this directly to sense data; or rather abduce from sensory information, hypotheses about the world that can be tested and validated. In this way complex and meaningful behaviour observed in autonomous robots not only becomes possible, the use of logic as the main representational language allows both a uniform framework to identify and exploit meta-level structure, but also provides the means to rigorously specify and establish program correctness, and lays the foundations for robots and autonomous artificial agents that can reason about their own behaviour in a physical world.

The first part of this paper introduces aspects of the Event Calculus [19, 21], then describes how sensor events and the effects of actions are represented in the Event Calculus for a mobile robot. It then describes how abductive reasoning is applied to important mobile robot tasks, sensor data assimilation, map-building, planning and localisation. We next provide a description of how an Event Calculus based robot controller is interfaced to a

Proc. TIMR 01 – Towards Intelligent Mobile Robots, Manchester 2001.
Technical Report Series, Department of Computer Science, Manchester University, ISSN 1361 – 6161
Report number UMCS-010401. http://www.cs.man.ac.uk/cstechrep/titles01.html

Khepera [9], a real, if miniature, mobile robot. In the second part we widen the discussion by introducing a new visual *Region Occlusion Calculus* (ROC), and indicate how it may be used to formally describe a greater range of more complex sensory events and in turn generate a spatial segmentation of a robot's environment, in which the robot may navigate and reason about its surroundings.

2 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*, a 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 I 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*, [19]; 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".

3 Robot Control - Building on the Event Calculus

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 conducted 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* [21].

4 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.

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 [1] 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 malfunction must be formed [3, 5].

5 Using Abduction for Sensor Data Assimilation, Map-building, Planning and Localisation

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.)

5.1 Sensor Data Assimilation

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

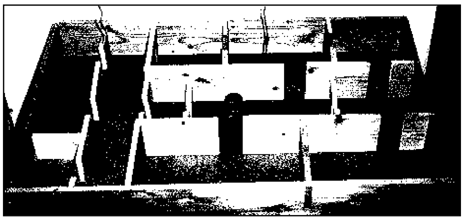
$$EC \,\&\, \Sigma \,\&\, \Delta \,\&\, \Psi_S \models \Gamma_S$$

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 ($\models$) the new sensor input Γ_S. 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. Sensing is very restricted in the robot used, relying only on short-range (2cm) proximity detectors and wheel odometry. The robot operates in its (miniature) model of an office environment, figure 1, by following the walls and edging around corners, so as to maintain a continuous sensory stream. 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 corner around the room. The abduced residue must be consistent with the current plan, if it is not, the current plan must be abandoned and a new initiated. 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.

5.2 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 the abductive process must give rise to a single interpretation before the map can be extended. The accuracy of the Khepera's odometry sense is inadequate to localise a room feature to a unique place. We are therefore obliged to add *integrity constraints* [21], making explicit, for instance, that a corner cannot be located in two rooms to guide the abductive reasoning. 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 to a doorway that leads to an unexplored room.



/* 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).

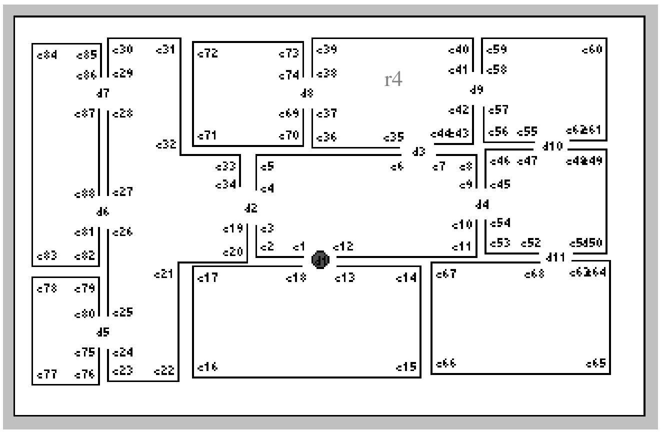


Figure 1: Actual office environment model, room 4 axioms (left) and visualisation of map axioms (right)

5.3 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 control level, but may not be made available to the logic level.

5.4 Localisation

$$EC \,\&\, \Sigma \,\&\, \Delta_R \,\&\, \Psi_L \models \Gamma_S$$

In localisation we 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 ever become empty, the map is no longer valid, and must be reconstructed.

6 Environment to Events: Robot Actions and Fluents

This section describes the effects of the action events and generation of sensor event fluents that characterise model office environments of the form shown in figure 1 when used with miniature (6cm high) Khepera mobile robots from K-Team [9]. The environment is defined in terms of “walls”, “corners”, “doors” and “rooms”. 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 2), and wheel odometry, a number of restrictions are placed on the design of this environment. It 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 seen by the Event Calculus program detect discontinuities (the corners, door-posts and doorways) between these features. At the robot level, these sensor conditions are completely anonymous (that is, they provide no identifying information about the features they detect) and must be combined and built-up 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 terminate them. The sensor fluents Left and Right indicate that the robot is beside a wall 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 1) or when a doorpost is encountered (e.g. C1, C26 or C38). The robot cannot directly distinguish these two cases, and so must perform a GoLeft (or GoRight) to follow round the corner. Sensing a Left or Right indicates a concave corner, 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 followed by a GoLeft or GoRight to continue in the same room. Figure 2 charts the values of the sensors, and the cumulative rotations of the wheels (diamond and square markers) for a sequence of three actions, Forward, GoLeft then GoRight giving rise to three fluents, LeftGap, InDoorway and Right, as it negotiates a doorway in the environment. Fluent events are detected by a combination of changing sensor value and distance moved.

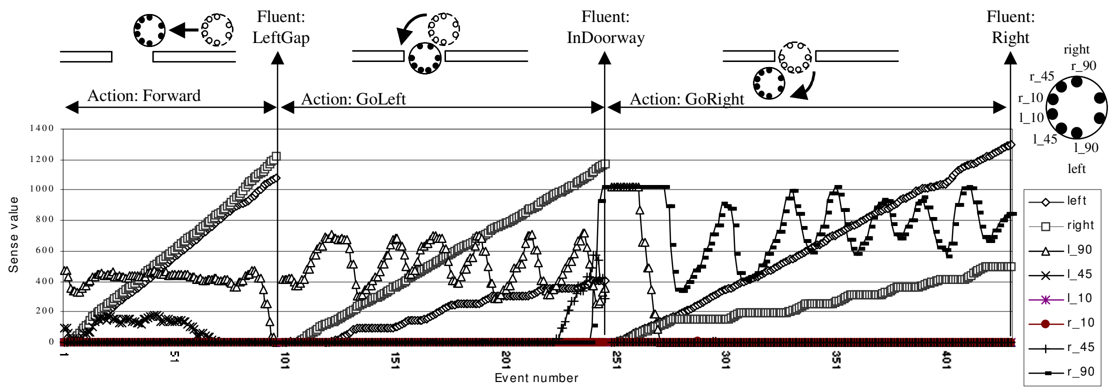


Figure 2: Actions and Fluents at the Robot Level

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.

7 Generating Fluents from Visual Occlusion

As with the Khepera robots using its primitive sensors to provide the raw data from which features in the model are constructed, so also with a robot using machine vision. In the Khepera case, discontinuities and invariants in the Khepera's physical environment are used to infer the existence of features such as walls, corners and doorposts. These give rise to fluents, which may then used within the Event Calculus to map build and navigate through its environment. In the second this part of the paper we now consider the extraction of fluents from machine vision data. In particular information garnered from occlusion events and motion parallax. Occlusion events map directly to fluents, and actions of the robot may bring about a change in the relative positions and alignments between objects with respect to the robot's current viewpoint. In this way, sensory data tied to movement of the robot again leads to explanations of what it senses, and in turn provides a way to disambiguate what it sees, or if not, the basis to re-plan a set of actions in order to achieve this.

To fully characterise the various relationships between arbitrary-shaped objects in a visual field we have developed the *Region Occlusion Calculus* (ROC). The Region Occlusion Calculus is a first-order logical theory that describes the spatial relationships between bodies as seen from a robot's viewpoint. The theory encapsulates two things: (a) spatial occlusion (or *interposition*) between objects, and (b) the means to reason about how these occlusion events will change with respect to changes in the robot's viewpoint, for instance in the motion parallax effect, whereby a change in viewpoint causes relative displacements of objects at different distances in the visual field. Both cues are exploited to build up an awareness of three-dimensional form and distance from a two-dimensional image.

Occlusion events help to determine where an object's boundary lies, or to infer why an object cannot be seen, and what needs to be done in order to render it visible. For example, consider two objects A and B in a robot's visual field (Figure 3). Suppose a robot moves to its left, while keeping these objects in sight. If object A passes across B, or when moving toward A, B becomes completely obscured the robot can infer that A is in front of B. Similarly, if, when moving to the right, no relative change arises, the robot may infer that A and B are far away, or close by and possibly moving in the same direction as itself. Conversely, if A, when visible, always appears to be subtended by B, the robot may infer that A and B are physically connected. In each case, occlusion events and motion parallax are being used to derive an objective model of the world from a naturally restricted viewpoint.

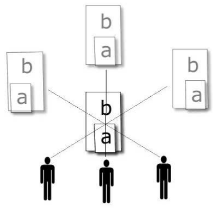
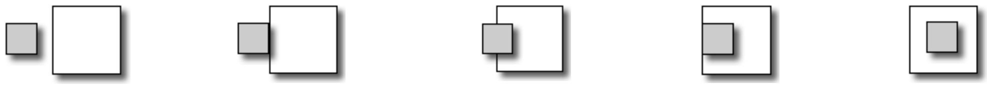


Figure 3: Spatial occlusion at work. Occlusion events arise with a change in viewpoint.

The Region Occlusion Calculus extends the earlier *Lines of Sight calculus* of Galton [7], and builds on the *Region Connection Calculus* of Randell, Cohn and Cui [16]. Distinguishing features of the new calculus are that it can represent concave objects, and so is therefore able to represent objects that can mutually occlude each other, and it encapsulates a notion of depth and comparative distance between objects. Both are essential in a logical theory if it is to find practical application in robotics. Region Occlusion Calculus reduces the description of possible alignments between objects to 20 relations. These 20 relations are *jointly exhaustive and pairwise disjoint* (JEPD), which is to say they completely account for all possible cases, and no two relation subsumes or is a more general case of another. All are reducible to a primitive relation of connection (x connects with y) and total occlusion (x totally occludes y from viewpont v), and a function that maps three-dimensional bodies and a viewpoint to their corresponding images in the visual field (the image of x from viewpoint v). The theory is further augmented with relations that express comparative distances between bodies (x is nearer to y than z), and their left/right orientations with respect to a restricted viewport (x is to the left/right of y from viewpoint v).

As each spatial relation describes a possible alignment in the world between two bodies in the visual space, plus the fact we have a JEPD set of these, it is possible to model all continuous transitions via sequences of discrete spatial relations. In the implemented logic these changes map to changes in the apparent connectivity of images in the visual space as either positions of the bodies or the viewpoint changes. These can be formulated into a transition graph (c.f. Freksa's [6] *conceptual neighbourhoods*), which captures these direct transitions, with paths through the graph interpreted as possible sequences of instantaneous changes between occlusion events over time. Transitions can also be expressed as a set of envisioning axioms [15] to form the basis of a qualitative simulation program [2], or can be re-worked directly within the Event Calculus that explicitly models change over time.



NonOccludesDC *(DC: DisConnected)* | *NonOccludesEC* *(EC: External Connect)* | *PartiallyOccludesPO* *(PO: Partial Overlap)* | *PartiallyOccludesTPP* *(TPP: Tangential Proper Part)* | *PartiallyOccludesNTPP* *(NTPP: Non Tangential Proper Part)*

Figure 4: Some of the Occlusion Relationships

Figure 4 shows five of the 20 occlusion relationships: NonOccludesDC(*x*,*y*,*v*), NonOccludesEC(*x*,*y*,*v*), PartiallyOccludesPO(*x*,*y*,*v*), PartiallyOccludesTPP(*x*,*y*,*v*), PartiallyOccludesNTPP(*x*,*y*,*v*). In each case, from the robot's viewpoint v, NonOccludesDC(*x*,*y*,*v*) describes the case where from viewpoint *v*, two bodies *x* and *y* appear separated. NonOccludesEC(*x*,*y*,*v*) the case where external boundary of *x* exactly aligns the external boundary of *y*. PartiallyOccludesPO(*x*,*y*,*v*) the case where *x* partially overlaps *y*; PartiallyOccludesTPP(*x*,*y*,*v*) the case where *x* is appears inside *y*, and their boundaries align; and PartiallyOccludesNTPP(*x*,*y*,*v*) where *x* appears completely inside *y*. Note that NonOccludesEC(*x*,*y*,*v*) and PartiallyOccludesTPP(*x*,*y*,*v*) capture alignments between objects *x* and *y*, so PartiallyOccludesTPP(*x*,*y*,*v*) is the case whenever *x* is both inside and edge aligned, there will be many such instances, and so for the others. Similarly PartiallyOccludesPO(*x*,*y*,*v*) describes all the instances where the occluding object *x* partially overlaps the occluded object *y*'s boundary.
Within the Event Calculus, these predicates now re-appear as functions indexed to time points, or to intervals where the fluent holds, e.g. HoldsAt(NonOccludesDC(*x*,*y*,*v*),*t*). To these fluents we add actions taken by the robot and map actions to changes in the apparent connectivity between pairs of objects as the robot's viewpoint, or the positions of the bodies with respect to its viewpoint changes. We now describe how we might map these high level descriptions to data extracted from the visual field.

8 From Visual Occlusion to Mapping Space

The goal of this element of our on-going research is the development of a map building and robot navigation system based on a rigorous first order formalism for representing and reasoning about objects and about the free space between them. In general terms, the framework is constituted at three levels: first, at the sensor data level, the data from a stereo vision system is transformed in a depth profile of the image which are then described by logic formulae. A sequence of such profiles constitutes a transition of scenes involving changes in the states of objects; our logic reasoning system encodes these profile transitions and entails the relations between the objects. These relations constitute the second level of reasoning: that of image interpretation. On using a subset of the relations defined in the previous level (more specifically, the relations concerned with object occlusion) a tessellation of the environment is constructed, this is the third level: the map level. A rigorous description of the reasoning system is outside scope of the present paper. The aim of this section is to discuss some of the main principles involved in developing this framework.

8.1 Level 1: Sensor data

Consider a robot that receives information from the environment through a stereo vision system; thus, from the viewpoint of the robot, the objects of the external world are 2D regions in a camera image representing the depths of the objects. In order to transform the sensor data from the vision system into a high-level description of the relations between the objects, we consider a simplification of the initial data. Instead of analysing the whole stereogram of a scene we analyse initially a horizontal cut of it, creating then a one dimensional *depth profile* of the scene (figure 5b). The whole image can be analysed considering a grid of depth profiles with a granularity depending on the complexity of the required scene description.
A depth profile (figure 5b) is a 2D chart representing, in the vertical axis (l), the relative depths of the objects in a robot's viewpoint and, in the horizontal axis (Θ), the relative size of the objects measured in terms of angular coordinates w.r.t. the field of view of the robot (θ_n). The l axis is constrained by the furthest point that can be noted by the robot's sensors, this limiting distance is represented by the letter L in the profile in figure 5b. Therefore the

vertical peaks in a profile represent objects and relation between objects in the environment, thus we also use the term *object profile* referring to a peak in a depth profile. It is worth noting that the values of l and Θ are qualitative rather than quantitative, and they can be encoded in terms of fuzzy notions such as *less near*, *near* and *very near* for the depth variable l, and similar notions for the variable Θ.

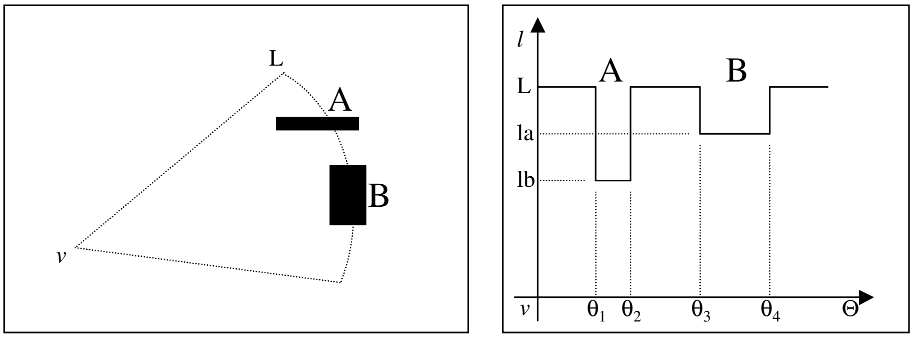


Figure 5: a) Two objects *A, B* inside of a robot's (ν) field of view; b) depth profile relative to the field of view

Another important feature of the depth profiles is the shape of the peaks. For the moment we are only interested in two categories of shapes of object profiles: *single* and *composed*. Single peaks are those depicted in figure 5b, while a composed peak has the shape of a step (figure 6.P3 below). The importance of this classification is briefly explained as follows.

8.2 Level 2: Image interpretation - occlusion

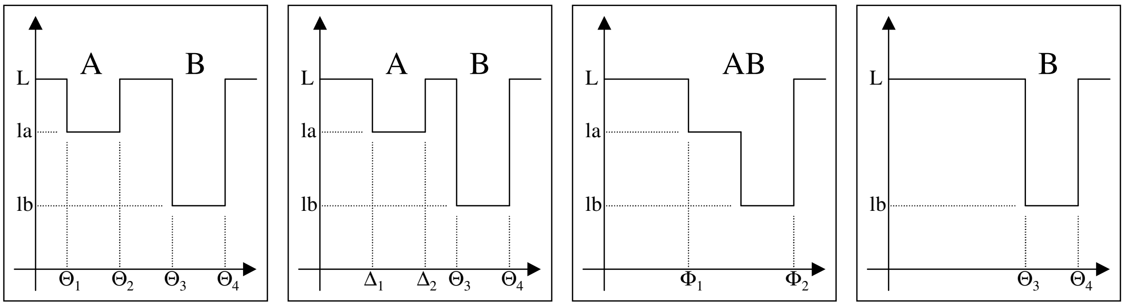


Figure 6: A depth profile sequence P1, P2, P3 and P4.

Many conclusions about an object's relationships can be asserted from the analysis of sequences of depth profiles. One of them has an immediate interest to the present work, the interpretation of occlusion transitions. Figure 6 is an example of a depth profile sequence involving this kind of transition. The first step towards the logic reasoning system is the description of the main features of the profile transitions in such a way that higher level interpretations are logically entailed by this description. Intuitively, the transition from **P1** to **P2** (**P1->P2**), in figure 6, suggests that either the object **A** approached the object **B** or the observer made a semi-circle centred in **B** in the direction of decreasing the distance between the peaks **A** and **B**. The logic reasoning system should be capable of entailing these possibilities.

From the logic description of transition **P2 -> P3** (which should contain the information that the single peaks **A** and **B**, in profile **P2**, became a composed peak **AB** in profile **P3**), the reasoning system should be able to entail that object **B** *partially occludes* **A**. Similarly, from the transition sequence **P2->P3->P4** mutually occlusion between **A** and **B** should be entailed from the information that the peaks **A** and **B** changed from two single peaks (in **P2**) to one composed (in **P3**) and eventually to one single peak (in **P4**). Note that, as a simplification, we use the same notation for the peaks **A** and **B** and for the possible objects they might denote. Rigorously, this association should be an abductive inference since a single peak (from a particular viewpoint) can split in two (possibly composed) peaks from another viewpoint.

8.3 Level 3: The Map

A simple solution to the problem of map building begins by making the robot turn 360° around each of the objects in the environment. While turning around an object, the initial task of the map building system is to observe the occlusion relations between this object and any other object in the robot's field of view. The aim of observing this relation is to note the point where it terminates holding; in other words, the initial task of the map building system is to note (analysing the profile sequences) the transition when (given two objects A and B and adjacent viewpoints v and v'), for instance, PartiallyOccludesPO(A,B,v) becomes NonOccludesEC(A,B,v'). The next task of the map building system is, then, to store the pair of points [v, p], where p is any point in object A that is the nearest to B (figure7a). Such pairs of points define a straight line, which, for the purpose of this work, we name a *line of sight*.

The purpose of noting these lines is to use them to construct a tessellation (figure 7c) of space defined by the intersections of the lines of sight between the objects in the environment. This tessellation is a map of the environment within which the robot can locate itself and plan paths through the world. This process is similar to the construction of orientation regions (Levitt and Lawton, [12]). A suitable data structure must be created in order to store the lines of sight and the tessellation of the environment, maintaining the order among the lines and the regions, so that path planning can be reduced to efficient search. This data structure might be similar to a quad-tree, modified in order to consider the ordering of regions and lines; Schlieder [18] presents a possible way to define such an ordering.

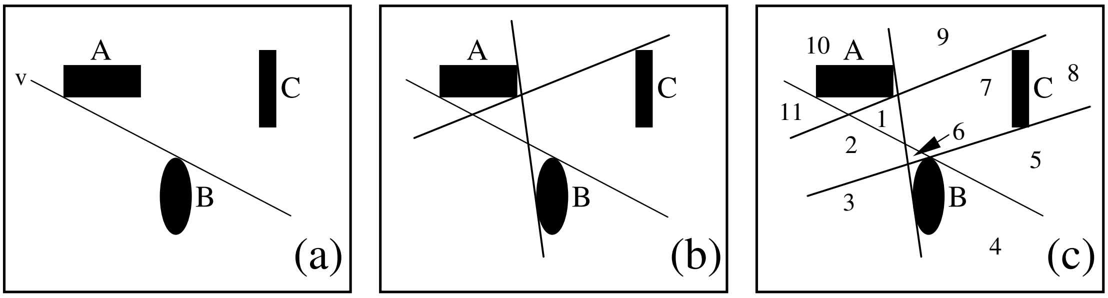


Figure 7: Construction of orientation regions from lines of sight (aerial view).

Another point of concern in this framework is the time efficiency of the map building process briefly described above. In order analyse the main factors influencing time efficiency of this map building process, consider that the environment is composed of N objects. The worst case occurs, thus, when each one of the N objects defines lines of sight to each other from some viewpoint. In this case, for every two objects there are two different lines of sight to be defined. Thus, the time efficiency of this task is $O(n^2)$.

However, the most influencing issue in efficiency is the task of constructing orientation regions. Clearly, this process depends on the number of intersection points of lines of sight, which depends on the geometrical distribution of the objects in the environment. If all of the intersections between lines of sight are calculated, the algorithm would produce a large number of polygons, most of them would be far too small for the purpose of robot navigation (as is the case of region 6 in figure 7c). A solution to this problem would be to discard some of the lines of sight using a preference criterion (or a conjunction of criteria) while storing these lines; an example of such criteria is choosing lines that connect only objects that measure twice the size of the robot (limiting in this way storing lines of sight between objects that can have their positions altered by the robot's motion).

9 Conclusions

We have noted that the use of logic to control robots appears to be balanced on a "semantic knife-edge", between a representation that is too sparse to allow adequate reasoning about the domain in hand, and one too rich to be tractable to current automated proof techniques. We have presented an approach to four closely interrelated mobile robot tasks, namely sensor assimilation, planning (and navigation), map-building and localisation, within a single logical framework based on the Event Calculus. By careful choice of the primitive events synthesised at the robot level and made available to the reasoning part, we then argued that it is possible to construct and reason with complex descriptions and maps built from many, but very low-level, sensor readings. Event Calculus programs are typically concise for each of these tasks when compared to equivalent code written procedurally, but automated reasoning is computationally intensive and performance suffers greatly as a consequence compared to conventional approaches to map building and navigation, [22] for example. Our on-going research not only investigates better ways of representing robot knowledge, but also better ways of exploiting it through reasoning. We suspect, however, that the semantic knife-edge problem will never be fully resolved, for each time we make an advance in the speed or effectiveness of automated reasoning it will inevitably be countered by an increase in the ambition to generate more detailed descriptions of the world and extend the bridge between grounded, sensor and motor based, reasoning with ever greater degrees of cognitive skill.

Acknowledgements

This work has been supported by EPSRC project GR/N13104, "Cognitive Robotics II".

References

[1] Boutilier, C. and Becher, V. (1995) "Abduction as Belief Revision", Artificial Intelligence, Vol. 77, pages 43-94

[2] Cui, Z, Cohn, A.G and Randell D.A. (1992) "Qualitative simulation based on a logical formalism of space and time", *Proceedings AAAI-92*, AAAI Press, Menlo Park, California, pages 679-684

[3] Davis, R. (1984) "Diagnostic Reasoning Based on Structure and Behavior", *Artificial Intelligence*, Vol. 24, pages 347-410

[4] De Giacomo, G. (ed.), (1998) *Cognitive Robotics: Papers from the 1998 AAAI Fall Symposium*, Orlando, Florida, AAAI Technical Report FS-98-02

[5] de Kleer, J. and Williams, B. C. (1987) "Diagnosing Multiple Faults", *Artificial Intelligence*, Vol. 32, pp. 97-130

[6] Freksa, C. (1992) "Temporal reasoning based on semi-intervals", *Artificial Intelligence*, Vol. 54, pages 199-227

[7] Galton, A.P. (1994) "Lines of Sight", *AISB Workshop on Spatial and Spatio-Temporal Reasoning*.

[8] Huhns, M.N. and Singh, M.P. (1997) "Agents and Multiagent Systems: Themes, Approaches and Challenges", *Readings in Agents*, Morgan Kaufmann Publishers, pages 1-23.

[9] K-Team (1995) "Khepera User Manual", K-Team SA, Ch. du Vasset, CP111, 1028 Préverenges, Switzerland, Version 4.06, November 1995

[10] Lespérance, Y., Levesque, H.J. Lin, F., Marcu, D. Reiter, R. and Scherl, R.B. (1994) "A Logical Approach to High-Level Robot Programming: A Progress Report", in B. Kuipers (ed.), *Control of the Physical World by Intelligent Systems: Papers from the 1994 AAAI Fall Symposium,* New Orleans, pages 79-85

[11] Levesque, H., Reiter, R., Lespérance, Y., Lin, F. and Scherl, R.B. (1997) "GOLOG: A Logic Programming Language for Dynamic Domains", *The Journal of Logic Programming*, Vol. 31, pages 59–83

[12] Levitt, T. S. and Lawton, D. T. (1990) "Qualitative Navigation for Mobile Robots", *Artificial Intelligence*, Vol. 44, pages 305-360

[13] Nilsson, N. J. (1984) "Shakey the Robot", SRI Technical Note no. 323, SRI, Menlo Park, CA.

[14] Pratt, I. (1999) "Qualitative Spatial Representation Languages with Convexity", *J. of Spatial Cognition and Computation*, Vol. 1, pages 181-204

[15] Randell, D. A. (1991) "Analysing the Familiar: Reasoning About Space and Time in the Everyday World", PhD thesis, University of Warwick, Coventry, UK, 1991

[16] Randell, D. A., Cui, Z. and Cohn, A. G. (1992) "A Spatial Logic Based on Regions and Connection", in proc. *3rd Int. Conf. on Knowledge Representation and Reasoning*, pages 165-176

[17] Reiter, R. (1998) "Sequential, temporal GOLOG", *Proceedings 1998 Knowledge Representation Conference (KR 98)*, pages 547–556

[18] Schlieder, C. (1995) "Reasoning About Ordering", in: COSIT'95, Springer LNCS, Vol. 988, pages 341-349

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

[20] Shanahan, M.P. (2001) "Reinventing Shakey", in J. Minker (ed.) *Logic-Based Artificial Intelligence*, Kluwer Academic

[21] Shanahan, M. P. and Witkowski, C. M. (2000) "High-level Robot Control Through Logic", in proc. *Agent Theories, Architectures and Languages* (ATAL-2000), Boston, pages 100-113

[22] Thrun, S. (1998) "Learning Maps for Indoor Mobile Robot Navigation", *Artificial Intelligence*, Vol. 99, pages. 85-111